Time-Optimal Control of Two-Link Manipulators

A Thesis Submitted to the College of

Graduate Studies and Research

in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy

in the Department of Mechanical Engineering

University of Saskatchewan

Saskatoon, Saskatchewan

by

Reza Fotouhi-Chahouki

Fall 1996

UNIVERSITY OF SASKATCHEWAN

College of Graduate Studies and Research

SUMMARY OF DISSERTATION

Submitted in partial fulfilment

of the requirements for the

DEGREE OF DOCTOR OF PHILOSOPHY

by

REZA FOTOUHI-CHAHOUKI

Department of Mechanical Engineering

College of Engineering

University of Saskatchewan

Fall 1996

Examining Committee:

Dr. T.S. Sidhu	Dean's Designate, Chair
	College of Graduate Studies and Research
Dr. B. Sparling	Department of Civil Engineering
Dr. A.T. Dolovich	Department of Mechanical Engineering
Dr. L.G. Watson	Department of Mechanical Engineering
Dr. W. Szyszkowski	Advisor, Department of Mechanical Engineering

External Examiner:

Dr. O.G. Vinogradov

Department of Mechanical Engineering

University of Calgary

Calgary, Alberta

T2N 1N4

Time-Optimal Control of Two-Link Manipulators

Time-optimal maneuvers of rigid Two-Link Manipulators (TLM) are analysed using the Pontryagin's Minimum Principle. The problem is formulated using Optimal Control Theory and ideas developed in the Calculus of Variations. The dynamics of rigid TLM is obtained using the Lagrange's equations of motion.

The optimal solutions with bang-bang controls are found by solving the corresponding nonlinear Two-Point Boundary Value Problems (TPBVP). Since the problem is very sensitive to the unknown initial costates, a strategy which combines the Forward-Backward Method (FBM) and the Shooting Method (SM) is proposed and used successfully.

The FBM is a numerical procedure, which finds a non-optimal solution that satisfies the state equations and the initial and the final boundary conditions. This solution is used to obtain approximate locations of the switch times and the initial values of the costates. The initial costates are required to initiate the SM which then is used to find the optimal solution that satisfies the state equations, the costate equations, and all the boundary conditions. Usefulness of the FBM for generating the initial costates such that the SM converges is demonstrated by numerical examples. Linear and nonlinear systems with single, or double controls are considered.

Several rigid TLM with different mechanical properties undergoing various maneuvers are discussed in detail.

It is shown that for the time-optimal maneuvers of the TLM the number of switches is related to the magnitude of the maneuver and is not constant as suggested in earlier works. In general, for the cases investigated, shorter maneuvers required three switches while longer maneuvers required four switch times.

The effects on TLM time-optimal maneuvers of physical parameters such as geometry, link masses and masses at both link tips as well as changes in the limits of control forces are also examined. It is shown that, the maneuver time can be further shorten by altering the lengths of the links and the ratio of the controls applied at the two joints.

For analysis of the effects of flexibility of the manipulator, the forces obtained from optimal solution of the rigid TLM are applied to the flexible manipulator and the response is simulated by the nonlinear finite element method. The effects of flexibility are measured in terms of the amount of vibrations generated and the distance between the tip of the manipulator at the end of the maneuver and the target point. The performance of such forces to control the flexible TLM is found to be satisfactory if the slenderness of the links is sufficiently small.

Permission To Use

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, the author has agreed that the Libraries of this University may make it freely available for inspection. The author further has agreed that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised this thesis work or, in their absence, by the Head of the Department or the Dean of the College in which this thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without the author's written permission. It is also understood that due recognition shall be given to the author and to the University of Saskatchewan in any scholarly use which may be made of any material in this thesis.

Request for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Mechanical Engineering
University of Saskatchewan
57 Campus Drive
Saskatoon, Saskatchewan, Canada
S7N 5A9

Abstract

Time-optimal maneuvers of rigid Two-Link Manipulators (TLM) are analysed using Pontryagin's Minimum Principle. The problem is formulated using Optimal Control Theory and ideas developed in the Calculus of Variations. The dynamics of rigid TLM are obtained using the Lagrange equations of motion.

The optimal solutions with bang-bang controls are found by solving the corresponding nonlinear Two-Point Boundary Value Problems (TPBVP). Since the problem is very sensitive to the unknown initial costates, a strategy which combines the Forward-Backward Method (FBM) and the Shooting Method (SM) is proposed and used successfully.

The FBM is a numerical procedure which finds a non-optimal solution that satisfies the state equations and the initial and the final boundary conditions. This solution is used to obtain approximate locations of the switch times and the initial values of the costates. The initial costates are required to initiate the SM which then is used to find the optimal solution that satisfies the state equations, the costate equations, and all the boundary conditions. Usefulness of the FBM for generating the initial costates such that the SM converges is demonstrated by numerical examples. Linear and nonlinear systems with single, or double controls are considered.

Several rigid TLM with different mechanical properties undergoing various maneuvers are discussed in detail.

It is shown that for the time-optimal maneuvers of the TLM the number of switches is related to the magnitude of the maneuver and is not constant as suggested in earlier works. In general, for the cases investigated, shorter maneuvers required three switches while longer maneuvers required four switch times.

The effects on TLM time-optimal maneuvers of physical parameters such as geometry, link masses and masses at both link tips as well as changes in the limits of control forces are also examined. It is shown that the maneuver time can be further shortened by altering the lengths of the links and the ratio of the controls applied at the two joints.

For analysis of the effects of flexibility of the manipulator, the forces obtained from optimal solution of the rigid TLM are applied to the flexible manipulator and the response is simulated by the nonlinear finite element method. The effects of flexibility are measured in terms of the amount of vibration generated and the distance between the tip of the manipulator at the end of the maneuver and the target point. The performance of such forces to control the flexible TLM is found to be satisfactory if the slenderness of the links is sufficiently small.

Acknowledgements

The author would like to thank his supervisor Dr. Walerian Szyszkowski for his help and valuable guidance throughout the Doctoral program. Also the author would like to thank Jonathan Moore and particularly Ian MacPhedran from Engineering Computing Services who helped with the computer software and system operation. The author's thanks also goes to the members of the advisory committee, Dr. B. Sparling of the Civil Engineering Department, Dr. A.T. Dolovich, and Dr. L.G. Watson of the Mechanical Engineering Department. It was an honour for the author to have Dr. O.G. Vinogradov, Department of Mechanical Engineering, University of Calgary, Alberta as his external examiner. The author also would like to honour the contribution of the late Dr. H.T. Danyluk of the Mechanical Engineering Department.

Financial support for this thesis work obtained from the Ministry of Culture and Higher Education of the Islamic Republic of Iran, the Natural Science and Engineering Research Council of Canada operating grant of Dr. W. Szyszkowski, and a University of Saskatchewan Graduate Scholarship is gratefully acknowledged.

The author would like to thank his parents for their care, especially providing good education during the high school years. Finally and most importantly, the author would like to thank his dear wife Ashraf for her unconditional moral support.

Dedication

This thesis is dedicated to the author's family Ashraf and Sohael.

Table Of Contents

List of Tables

List of Figures

Nomenclature

Listed below are the symbols and abbreviations used most frequently in the text. Occasionally, the same symbol may have a different meaning defined in the text.

$a(.)$	nonlinear function of states	a_{11}	nonlinear function of rotation
a_{12}	nonlinear function of rotation	a_{13}	nonlinear function of rotation
a_{22}	constant	a_{14}	nonlinear function of rotation
a_{24}	nonlinear function of rotation	ar_i	aspect ratio of links
B	initial costate	$c(.)$	nonlinear function of states
C	damping matrix	d	closeness norm
d_i	diameter of links	e	discontinuity error
E	moduli of elasticity	E_k	kinetic energy
E_p	potential energy	$f(.)$	a function
$f_{ik}(.)$	function of control and state	F	force vector
$F(.)$	function of state and costate	FBM	Forward-Backward Method
$g(.)$	functional to be minimised	$g_a(.)$	functional to be minimised
g_r	gravitational acceleration	G	switch functions
$h(.)$	functional of final states	$\mathcal{H}$	Hamiltonian
I_0	mass moment of inertia of the shoulder joint	I_1	mass moment of inertia of shoulder link with respect to its centre of mass
I_2	mass moment of inertia of elbow link with respect to its centre of mass	I_a	mass moment of inertia of the shoulder tip
I_b	mass moment of inertia of the elbow tip	I	performance measure
J	performance measure	k	iteration number
kf	final iteration	K	stiffness matrix
$K(.)$	initial value of state and costate	l	number of final error
l_1	length of the shoulder link	l_2	length of the elbow link
l_{c1}	centre of mass of shoulder link with respect to its joint	l_{c2}	centre of mass of elbow link with respect to its joint

L	final error	$\mathcal{L}$	Lagrangian
m_1	mass of the shoulder link	m_2	mass of the elbow link
m_a	mass at the shoulder tip	m_b	mass at the elbow tip
m	number of controls	M	mass matrix
n	number of states	p	costate variables
PMP	Pontryagin's Minimum Principle	Q	generalised force
R	weighting matrix	R_I	inertia ratios of the links
R_L	length ratios of the links	R_U	ratios of control limits
SM	Shooting Method	SOC	Semi-Optimal Control
t	time	t_0	initial time
t_f	final time	t_{si}	switch times
TLM	Two-Link Manipulators	TPBVP	Two-Point Boundary Value Problem
u_1	control force for shoulder joint	u_2	control force for elbow joint
u	optimal control forces	$\tilde{u}$	control forces
u^{fd}	forward control	u^{bd}	backward control
U	limit of control forces	v	weight functions
w	weight functions		
x_1	rotation of shoulder link	x_2	angular velocity of shoulder link
x_3	rotation of elbow link	x_4	angular velocity of elbow link
x	state variables	x^{fd}	forward state
x^{bd}	backward state	x_0	initial states
x_f	final states	X	state and costate variables
α_k	positive scalar	ϵ_s	convergence norm
δ	linear variation of a function	Δ	increment of a function
ρ	positive scalar	ρ_i	density of the links
φ	degree of freedom	φ_1	rotation of shoulder link
φ_{1_0}	initial rotation of shoulder link	φ_{1_f}	final rotation of shoulder link
φ_2	rotation of elbow link	$\omega_1 = \dot{\varphi}_1$	angular velocity of shoulder link
$\omega_2 = \dot{\varphi}_2$	angular velocity of elbow link		

Chapter 1

Introduction

1.1 Background

Most automated manufacturing tasks are carried out by robots designed to perform specific functions in a manufacturing process. The Czech word *robota* is considered to be the origin of the word *robot*. The meaning of the word in Webster's dictionary is (quoted from [2]):

> an automatic device that performs functions ordinarily ascribed to human beings.

The Robot Institute of America has a more precise description of industrial robots [2]:

> A robot is a re-programmable multi-functional manipulator designed to move materials, parts, tools, or specialised devices, through variable programmed motions for the performance of a variety of tasks.

An industrial robot is a general purpose manipulator with several rigid links which is controlled by a computer. An example of the popular industrial robot PUMA [1] is shown in Figure 1.1.

Since manipulators are typically used to repeat a prescribed task a large number of times, even small improvements in their performances may result in large monetary savings.

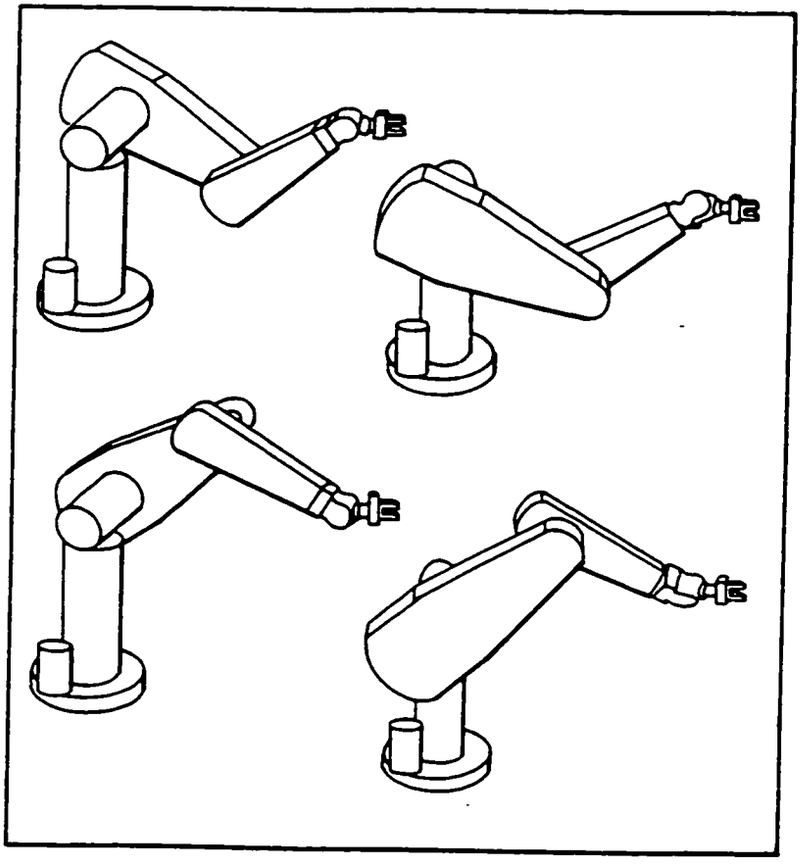

Figure 1.1: The industrial robot PUMA [1].

Here an attempt is made to minimise the maneuver time (time-optimal control) between two points if the controls (the driving forces) are constrained by their maximum and minimum values available from the installed motors. Any trajectory that can be realized by applying the available driving forces and connecting the starting point with the target point can be used to implement the maneuver. The major objective of this thesis is to develop a method that would determine the time-optimal trajectory and the corresponding values of driving forces for two-link manipulators (i.e. a robotic arm consisting of two segments).

1.2 Problem Formulation

The dynamics of manipulators relates the control $u(t)$ with their dynamic response $x(t)$. Typically, there are two kinds of problems when dynamics are used to design manipulators. The problem is called *inverse dynamics* when the trajectory $x(t)$ is known and the control forces have to be determined. This can be symbolised as $u(t) \Longleftarrow x(t)$. The other type is the *direct dynamics* problem where the behaviour of the manipulator for the assumed forces is to be determined, and can be symbolised as $u(t) \Longrightarrow x(t)$. It should be emphasised that, formally, when using the governing equations of dynamics in the design process, either x or u must be known.

In the classical design of manipulators, the inverse dynamics problem ($u(t) \Longleftarrow x(t)$) and the direct dynamics problem ($u(t) \Longrightarrow x(t)$) are used alternatively to gradually improve the performance and establish the control as $u(t)$ or $u(x)$.

The objective of optimal control theory, which nowadays plays an important role in the design of advanced systems, is to determine simultaneously such $u(t)$ and $x(t)$ that would minimise a specific performance measure. In optimal control the performance is written as

$$J(u) = \int_{t_0}^{t_f} g(x, u, t)dt \quad \longrightarrow \min \tag{1.1}$$

This imposes extra conditions on the variables $u(t)$ and $x(t)$ related through the governing equations of dynamics and combines the search for the best control with the search for the best trajectory into one optimisation process. It is expected that at least one $x(t)$ and one $u(t)$ exists that satisfies condition (1.1). Such a solution is recognised as an optimal solution.

An optimal control problem requires a mathematical model of the process to be controlled, a statement of physical constraints, and a performance measure. Here different aspects of the problem formulation for the time-optimal control of manipulators are reviewed.

1.2.1 Dynamics of Manipulators

In control theory the state of the system, which here is meant to represent the mathematical model of the dynamics of the manipulators, is usually written in the form

$$\dot{x} = a(x, u, t) \tag{1.2}$$

where x is the vector of state variables, u is the vector of controls, and a is the vector of nonlinear functions of states and controls.

In structural dynamics the equations of motion are usually written in the form

$$M(\varphi)\ddot{\varphi}(t) + C(\varphi)\dot{\varphi} + K(\varphi)\varphi(t) = F(t) \tag{1.3}$$

where $M(\varphi)$, $C(\varphi)$, and $K(\varphi)$ are nonlinear mass, damping, and stiffness matrices, φ is the vector of Degrees Of Freedom (DOF) or generalised displacement, and $F(t)$ is the vector of control forces driving the system. Equations (1.3) can be easily rewritten in the form of (1.2). Each component of φ_i in the DOF vector forms two state variables: $x_k^d = \varphi_i$ and $x_l^v = \dot{\varphi}_i$, where $k = 2i - 1$, $l = 2i = k + 1$, $i = 1, \ldots, n$. The superscripts ^{d} and ^{v} denote displacement and the rate of change of displacement, respectively. It allows the state vector x to be split into two parts: $x = [x_1^d, x_2^v, x_3^d, x_4^v, \ldots, x_{2n-1}^d, x_{2n}^v]^T$. Substituting into (1.3), the equations of motion in terms of the state variables are

$$\begin{aligned} \dot{x}_k^d &= x_l^v \\ \dot{x}_l^v &= M_{ij}^{-1}[F_j - (C_{ji}\, x_l^v + K_{ji}\, x_k^d)] \end{aligned} \tag{1.4}$$

Where $i, j = 1, \ldots, n$. Thus any system with n DOF can be defined by $2n$ state variables.

1.2.2 Boundary Conditions and Constraints

The mathematical model of the problem discussed earlier should include the boundary conditions and the physical constraints imposed on the states or controls. The boundary conditions for a manipulator are:

$$x(t_0) = x_0 \qquad\qquad x(t_f) = x_f \tag{1.5}$$

where $x(t_0)$ and $x(t_f)$ represent the positions and velocities of the links at the beginning and at the end of the maneuver, respectively. There may also be some constraints imposed on the states such as:

$$x_{min} \leq x(t) \leq x_{max} \tag{1.6}$$

Any state trajectory which satisfies these state constraints for the entire time of motion is called an *admissible trajectory.*

The constraints on the controls are:

$$U_i^- \leq u_i(t) \leq U_i^+ \tag{1.7}$$

where U_i^- and U_i^+ are the minimum and maximum forces or moments which can be generated by particular driving motors. The term *admissible control* is used when a control history satisfies the control constraints during the entire motion.

1.2.3 Performance Index

In optimal control a performance index (measure) is minimised or maximised. The designer might be required to consider several performance measures before selecting one.

For example, large structures/manipulators used in space applications are made flexible to reduce the high cost of lifting mass into orbit. However, more flexibility may

introduce some vibrations, affecting the accuracy of the maneuvers. Manipulators to be used in space might be optimised with respect to their precision as well as to their weight. This can be achieved by vector optimisation which can include those two goals.

To drive a manipulator for which a general task is given within particular limits for controls ($U^{\mp}$) and space (x_0, x_f), one can use optimal control theory, in which a performance index

$$J(u) = \int_{t_0}^{t_f} g(x, u, t)dt \tag{1.8}$$

has to be minimised. Note that, formally, the left hand side of (1.8) should be written as $J(x, u)$; however, since u and x are related via the state equation, the performance is dependent only on the control.

The accomplishment of such an optimal control depends on the particular form of $g(x, u, t)$. If $g = g_1(x)$, the corresponding performance index can be used to suppress the vibration or follow a particular path. For example, for $g = g_1(x) = x^T K x$, where K is the stiffness matrix, the performance represents the strain energy of the system. If $g = g_2(u)$, the corresponding performance index can be used to minimise the fuel consumption or minimise the use of power. Using $g = g_2(u) = u^T Q u$, where Q is a given matrix, suppresses the magnitude of the control forces.

If $g = c$, where c is a constant, the performance index represents the minimum time; that is

$$J(u) = \int_{t_0}^{t_f} c\, dt = c\,(t_f - t_0) \tag{1.9}$$

Since t_0 is known, this performance index will minimise t_f. This is a time-optimal control problem.

1.2.4 Formulation of Optimal Control Problem

The purpose of the optimal control problem is to determine the control $u(t)$ that minimises the performance index $J(u)$. From the physical point of view, the state x

must be continuous, but the control u can be discontinuous. For a better performance, the control may be required to change suddenly from its maximum value U_i^+ to its minimum value U_i^-. Such an instant is referred to as the switch time. If the control is performed using only the extreme values, switching between minimum and maximum, it is referred to as *bang-bang control.*

Optimal control means finding an admissible control $u(t)$ which makes the system (1.2) follow an admissible trajectory $x(t)$ and minimises the performance measure (1.8). Such u and x are optimal controls and optimal state trajectories. Minimum $J(u) = J^*(u)$ means that

$$J^*(u) = \int_{t_0}^{t_f} g(x^*, u^*, t)dt \leq \int_{t_0}^{t_f} g(x, u, t)dt \tag{1.10}$$

for all admissible states and all admissible controls. The quantities J^*, x^*, u^* are the optimal values of the parameters. They define the global minimum of J. The inequality of (1.10) may also be met only for some range of states ($\|x\| < b$), where $\| \ \|$ means a norm of admissible trajectories and b is a positive value. In this case (1.10) would define a local minimum. For simplicity the superscript * will be dropped throughout the thesis, unless its existence is considered to be necessary.

Example 1: Linear Mass and Spring System

For illustration of the time-optimal control problem a linear mass and spring system shown in Figure 1.2 is used. At $t = t_0 = 0$ the mass M is at location B specified by the initial stretch $x_1(0) = x_B$ and the initial velocity $x_2(0) = v_B$. This mass must be brought to the unstretched configuration A where $(x_1(t_f) = 0 = x_2(t_f))$ in a shortest time given that the control force is limited by $|u(t)| \leq U$. Without u the system would vibrate indefinitely. If u was proportional to the current displacement and velocity, as it would be assumed in classical control (the proportionality coefficient would constitute the gain matrix of the feedback), theoretically, the mass would also vibrate indefinitely with decreasing amplitude. It is desired here that the mass be

stopped completely at t_f.

Using the time-optimal control strategy with $g(x, u, t) = 1$, the performance index would be

$$J(u) = \int_{t_0}^{t_f} dt = t_f \quad \longrightarrow \min$$

The equation of motion of such a linear system with one degree of freedom is given by

$$M\ddot{\varphi} + K\varphi = F(t)$$

and can be transformed into the state equation in the form

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= -x_1 + u \end{aligned} \tag{1.11}$$

where x_1 represents non-dimensional displacement and x_2 non-dimensional velocity, and u non-dimensional control force (see Section 6.1.1). Note that the state equation (1.11) cannot be solved independently since the control $u(t)$ is still unknown.

Details of the solution to this problem are given in Section 6.1. Here only some typical results are shown. For time-optimal trajectory the control u takes extremal values (bang-bang control), switching two times during the whole maneuver as shown in Figure 1.3. The optimal force should be $u = U^-$ (pressing down) during the intervals $0 \leq t < t_{s1}$ and $t_{s2} < t \leq t_f$, and $u = U^+$ (pulling up) in the interval $t_{s1} < t < t_{s2}$. The locations of the mass and its velocity at the beginning of the maneuver, at each switch time, and at the final time are indicated in Figure 1.2. The solution of a time-optimal control problem with bounded control, such as this example, is always a bang-bang control where the control take its extreme values.

The numerical values of non-dimensional boundary conditions and limits on con-

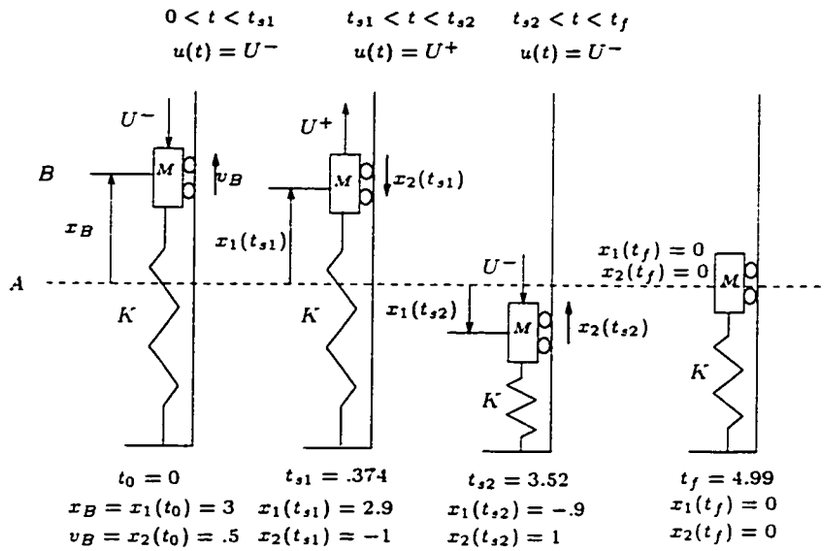

Figure 1.2: Optimal motion of linear mass and spring system.

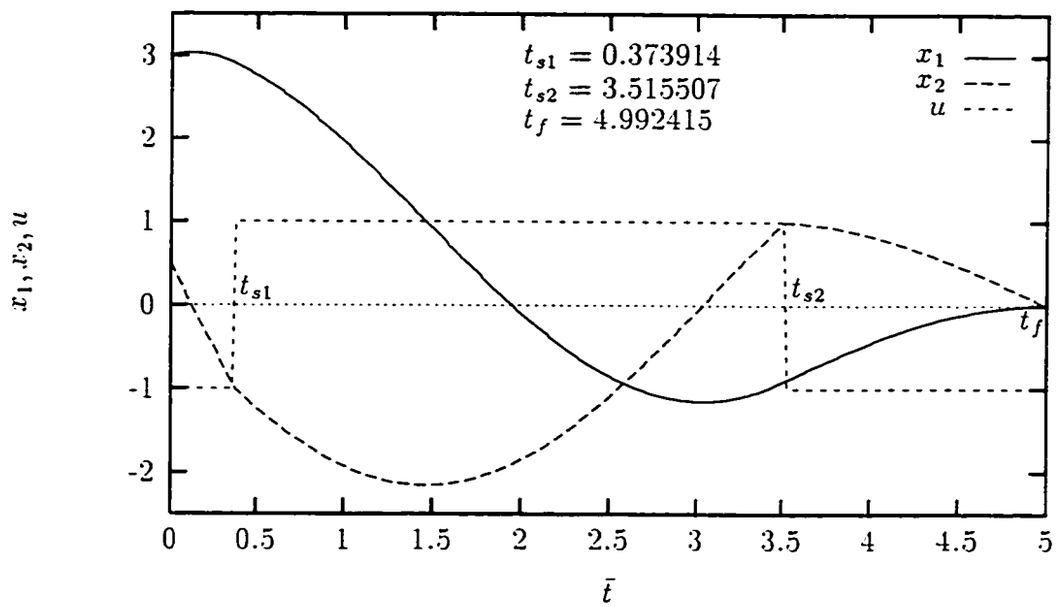

Figure 1.3: The optimal control solution and trajectory for example 1. x_1 and x_2 are states, u is optimal control.

trol for this problem are (after [3]),

$$x_1(0) = 3.0 \qquad x_2(0) = 0.5$$
$$x_1(t_f) = 0.0 \qquad x_2(t_f) = 0.0$$
$$-1 = U^- \leq u \leq U^+ = +1$$

All the values presented here are also non-dimensional (see Section 6.1). Note that at the target point, which is reached in time t_f (the time of optimal maneuver), the mass remains at rest when the control process is terminated.

1.2.5 Types of Optimal Control

Optimal control law or optimal control policy provides an admissible control history in the form $u(t) = f(x(t), t)$. From the view point of control it is considered a *closed loop control system* if it depends on the state. The optimal control law can be a linear, time invariant feedback of the states if $u(t) = Cx(t)$, where C is real constant matrix. The optimal control has an *open loop* form if $u(t) = f(t)$, because it does not depend on the state. Open loop control has several applications. Radar tracking of a satellite when the orbit is set, or industrial robot manipulators with a specific task are examples of the open loop control systems in that sense. Here, optimal control of a two-link manipulator is considered.

1.3 Solution of Optimal Control Problem

Optimal control, like all optimisation problems, can be approached using two methods: the direct and the indirect. The direct method is one approach in which the sets $(x^{(k)}, u^{(k)})$ and $(x^{(k+1)}, u^{(k+1)})$ should be selected in two consecutive iterations in such a way that $J^{(k+1)} < J^{(k)}$. Thus the performance is directly minimised, while trying to meet all the constraints, using various search techniques. Usually direct methods use parameter optimisation methods such as penalty, gradient, conjugate gradient, etc.

The direct methods (also referred to as parametric optimisation), which can solve any optimisation problem, are quite inefficient due to the large number of parameters involved [4] and because they are exhaustively time consuming (see Appendix A).

The alternative approach is the indirect method. This method is more analytical than the direct method. The conditions to be met on the optimal path must be derived first. These conditions form Pontryagin's Minimum Principle (PMP) and are the necessary conditions for an optimum solution. The next step is to determine the controls and the trajectory that meet these conditions. When successful, the indirect methods, in general, converge more quickly, but can experience problems with convergence. They may also become very complicated mathematically. Due to this complexity, the indirect methods are now mostly used only for verification of the solutions found with the help of other optimisation methods. Here, the indirect method for determination of time-optimal control of Two-Link Manipulators (TLM) is used. The optimal solutions are found by solving the optimality conditions formulated with the help of the PMP.

For solving optimal control problems of complicated systems such as TLM, a numerical approach is needed. Since the PMP for time-optimal control includes the initial and final boundary conditions, the problem is referred to as the Two-Point Boundary Value Problem (TPBVP). The shooting method is one of the basic ways of solving TPBVP. However, the method is extremely sensitive and difficult to converge for optimal control of TLM. Here a Forward-Backward Method is introduced and combined with the shooting method. With the help of the Forward-Backward Method and the shooting method, the very sensitive TPBVP of time-optimal control of TLM are successfully solved and analysed.

1.4 Concluding Remarks

If the control is bounded, and only the maneuver time is minimised, the optimal trajectory requires a bang-bang control, in which the controls take their maximum or minimum values throughout the time domain. The corresponding non-singular, nonlinear TPBVP can be formulated using the PMP. This problem may be solved by the shooting method or other analytical or numerical methods to determine the switching times for the optimal sequence of the control. However, unless the system is linear, time invariant, and of low order, there is little hope of determining the optimal control law analytically [5]. In general, for nonlinear systems, optimal control solutions are very difficult to obtain [6].

1.5 Scope and Objectives

In Chapter 2, the literature is reviewed and categorised into four major groups, from very general to a more specific. In Chapter 3, mathematical aspects of the optimal control theory are discussed. In Chapter 4, time-optimal maneuvers of rigid two-link manipulators (Figure 1.4) are analysed using Pontryagin's Minimum Principle. The dynamics of rigid TLM is obtained using the Lagrange equations of motion. In Chapter 5, the optimal solutions with bang-bang controls are found by solving the corresponding nonlinear two-point boundary value problems. Since the problem is very sensitive to the unknown initial costates, a strategy which combines the Forward-Backward Method and the Shooting Method is proposed and used successfully. In Chapter 6, results and analysis obtained for TLM are presented. Several numerical examples of optimal motion of TLM for different geometries of the target point are shown. It is shown that for the time-optimal maneuvers of the TLM the number of switches is related to the magnitude of the maneuver and is not constant as suggested in earlier works. Also, effects of flexibility of two-link manipulators are discussed there. For analysis of the effects of flexibility of the manipulator, the forces obtained

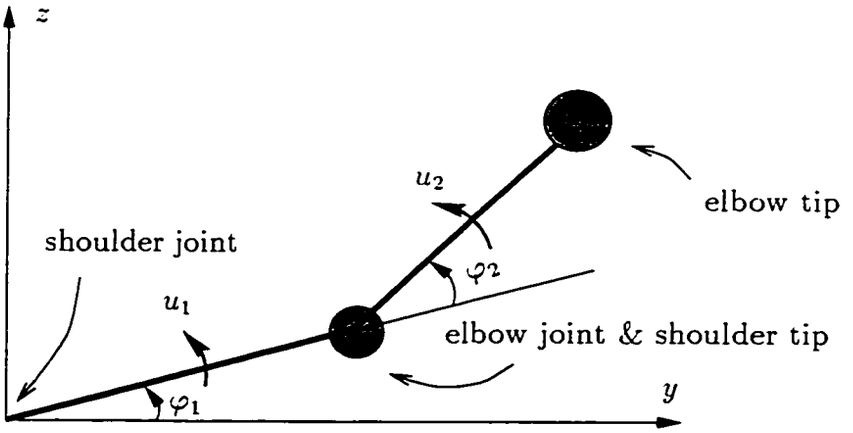

Figure 1.4: Rigid two-link manipulators.

from optimal solution of the rigid TLM are applied to the flexible manipulator and the response is simulated by a nonlinear finite element analysis. The effects on TLM time-optimal maneuvers of physical parameters such as geometry, link masses and masses at both link tips as well as changes in the limits of control forces are also examined. It is shown that the maneuver time can be further shortened by altering the lengths of the links and the ratio of the controls applied at the two joints. Chapter 7 contains the conclusion and the possible future work in this research area. In Appendix A the direct method of solving optimal control problems is outlined. In Appendix B the details of the analytical derivations are given. In Appendix C the computer code (OC2ADC13 of Section 5.4) that implements the Forward-Backward Method for TLM is explained. In Appendix D the computer code (XRKOC7 of Section 5.2) that implements the shooting method is given.

Chapter 2

Literature Review

A large number of papers and books covering various aspects of optimal control have been published in recent years. The topics considered here are categorised into four major groups, starting from general optimisation problems to more specific topics related directly to the time-optimal control of the Two-Link Manipulators (TLM). The first group of papers deals with formulation of optimal control problems from the view point of vector optimisation. The second group presents the application of optimal control to flexible manipulators/structures. The third category consists of the time-optimal control problem and its applications, which is the main topic of this thesis. The fourth category outlines the numerical methods for solving different optimal control problems.

2.1 Multiple Criteria Optimisation

Typically, several aspects of a design process should be looked at and optimised. This creates an optimisation problem with more than one objective referred to as *vector optimisation*. A vector optimisation involving optimal control problems can be written as

$$J_i(u) = \int_{t_0}^{t_f} g_i(x, u, t)dt \longrightarrow \min \tag{2.1}$$

where $i = 1, \ldots, m$ and m is the number of criteria to be optimised. With *multiple criteria* or vector optimisation, one deals with a design variable vector which satisfies

all the constraints and minimises the components of a vector of objective functions.

One of the characteristic features of the multicriteria optimisation problems is the appearance of an objective conflict. That means that none of the solutions allows the simultaneous minimisation of all objectives. This sometimes is called a *compromise performance index.* This problem is normally reduced to a scalar optimisation problem by formulating the substitute problems or a substitute performance function. Using the *Method of Objective Weighting* a substitute problem is solved:

$$J(u) = \int_{t_0}^{t_f} \sum_{i=1}^{m} w_i \times g_i(x, u, t)dt \tag{2.2}$$

The single performance index is determined by a linear combination of the criteria $J_1, \ldots, J_m$ and the corresponding weighting factors $w_1, \ldots, w_m$. It is possible to generate Pareto-optima for the original problem by changing the weight factors w_i in the performance index. A Pareto-optima or Pareto solution set is referred to the solution of a multicriteria optimisation problem. It is a reference to V. Pareto (1848-1923), the French-Italian economist who established the multicriteria optimisation concept [7].

There are other methods proposed to handle the vector optimisation problem such as *Method of Distance Functions*, *Method of Constraint Oriented Transformation (Trade-off Method)*, and *Method of Min-Max Formulation.* Details of these methods can be found in [7]. The following papers on optimisation problems deal with this type of optimal control problem.

A survey of multicriteria optimisation in mechanics is carried out in [8]. A Pareto set of a particular multicriteria optimisation problem dominates almost all of the optimal solutions in the literature.

In [9], simultaneous control and structure design is considered. The objective combines the weight of the structure and the robustness of the closed-loop control system. The robustness is represented by the sensitivity of the closed-loop eigenvalues with respect to uncertain parameters. The optimum design was obtained by using the method of feasible directions for constrained minimisation and the finite difference

gradients.

An optimisation procedure that combines the structural and control design criteria into a single problem formulation is also discussed in [10]. The necessary conditions for Pareto optimality are derived there.

Combined structures-controls-integrated optimisation using distributed parameters models is presented in [11]. An explicit closed-form expression for a clamped lattice-truss, using an equivalent anisotropic Timoshenko beam model is presented there.

Path following and the time-optimal control of robot manipulators are the objectives of the optimisation discussed in [12]. Extended Pontryagin's Minimum Principle (PMP) and parameter optimisation are used to solve the problem. It is found that one and only one control is always in saturation in every time interval along optimal path, while the other controls take on values within their bounds to guarantee the motion along prescribed path.

A minimum time and minimum control effort (see Section 3.1) of manipulators is considered in [13]. The approach presented there uses approximated dynamics in which the parameters are kept constant in each sampling time. The authors claimed that this process can be implemented on-line. The control sign is determined using simplified decoupled dynamics before hand and updated later using an averaging process.

A numerical solution to the optimal control problem with the compromise performance index of time and fuel using the minimum principle is the topic of discussion in [14]. The numerical method used for solving the corresponding two point boundary value problem is called the *"target practice method."* The initial costates are guessed first and corrected later.

A combined energy and travelling time control of the two-link manipulator is the topic in [15]. The problem is formulated using the PMP. The corresponding Two-Point Boundary Value Problem (TPBVP) is solved using a numerical algorithm

which employs the gradient method and the shooting method. A method named *"transition-matrix algorithm"* is used for solving the time-optimal control problem (when the weighting function for energy is set to zero.)

Simultaneous optimal control of the maneuver time and vibration of a flexible spacecraft (similar to a flexible single link manipulator) is the concern of [16]. A perturbation method is employed to solve the problem. The slewing of the rigid spacecraft is regarded as a zero order problem while the vibration is regarded as a first order problem.

Minimum-time trajectory of a two-degree of freedom system under the average heat generation is investigated in [17]. The performance index is a combination of minimum time and heat generation. The solution is obtained in two phases. First a minimum time solution is obtained when the coefficient of heat generation is assumed using the parameter optimisation method (finite element and steepest descent). Then a search for the coefficient is determined considering the average heat generation for rigid dynamics for the motion of the manipulator.

2.2 Optimal Control of Flexible Structures

Application of optimal control to flexible bodies with highly nonlinear governing equations are the topic of the following papers.

In [18] adaptive decentralised control of flexible multi body structures is considered. The decentralised technique allows the model to be divided into a set of subsystems, each of which is controlled independently.

Feedback control for end-effector tracking for flexible manipulators is considered in [19], where the feedback law is derived by using a series of steady state regulators at intermediate configurations along the desired path.

In [20], approximate optimal control of a flexible manipulator with the objective function which combines minimum time and tip tracking is solved using finite element

and recursive quadratic programming. As an initial control, the control configuration for a rigid arm has been used. The optimal control solution is not bang-bang even when only time is the objective function for the rigid arm.

Combined structural and control optimisation for flexible systems is considered in [21]. The objective of the optimisation is to minimise the mass and vibrational energy simultaneously. The control part is a linear regulator problem which can be minimised using the Riccatti equation. For the structural optimisation the gradient search is used. The algorithm combines finite element, optimal control, and gradient search.

A survey on the control of flexible structures is done in [22]. The work includes model reduction, active and passive control, hierarchical/decentralised control, combined structural-control, optimal design, and actuator/sensor location selection for flexible structures.

In [23], optimal control of a flexible single link manipulator is analysed by using the finite element method and Pontryagin's Minimum Principle.

Optimal control of a flexible single link manipulator is discussed in [24]. Using the technique of modal expansion, a finite dimensional dynamic model is developed. Pontryagin's Minimum Principle is employed to derive the differential equation of motion.

Dynamics and control of a space based mobile flexible manipulator is studied in [25]. The performance index is designed to suppress deviation of the tip of the link from its prescribed path. The nonlinear equation of motion is linearised about the planned path. With the linearised equation the problem is converted to a linear quadratic regulator problem which is solved using the Riccatti matrix equation. A routine DGEAR from IMSL/LIB is used for numerical solution.

Large angle maneuvers of a flexible spacecraft with a fixed final time is discussed in [26]. The control is accomplished in two phases. In the first phase a frequency shaped open loop solution for the nonlinear rigid body is used to control the band

width of the control system. The corresponding TPBVP is handled by the continuation method which progresses from a simplified solution to the full solution. In the second phase, the feedback control of the flexible body is obtained by linearisation along the rigid body solution. The performance measure includes the response of the system as well as the control. It is a combined measure of power consumption and vibration suppression.

In [27] multicriteria optimisation of nonlinear structural (mass and stress) and control is investigated and the numerical results for a flexible robot arms are discussed. The method uses a step by step integration procedure. The analogy between mechanical and electrical systems is shown. The gradient based optimisation method is then used to solve the problem.

2.3 Time-Optimal Control

Time-Optimal Control (TOC) problems can be represented by the following performance index

$$J(u) = \int_{t_0}^{t_f} dt = t_f - t_0 \tag{2.3}$$

where the final time t_f is unknown. A characteristic feature of these problems is that the controls are not usually continuous. Two types of problems can be considered here: for a known trajectory, the control u has to be found (path following problem in the shortest time), or both trajectory and u have to be found (minimum time problem). TOC problems are the main topic of the following papers.

In [28], where a path planning problem for the two-link manipulator is considered, the extended PMP was used. It was argued there that one and only one control torque is in saturation along the optimal trajectory. So the solution is not a bang-bang form.

Time-optimal control for arbitrary point-to-point transfers for three different robots is discussed in [29]. One of the robots is a robot with a horizontal articulated arm with two links. The parameter optimisation method is used for the numerical

analysis. The author divided the solution into six categories based on final position of the inner link. Singular solutions were found in two categories.

Mathematical discussion of the time-optimal control of rigid robot arms with friction is the topic of [30]. The author analytically proved that, for a rigid manipulator such as a TLM with friction, a time-optimal control always exists if the final state can be reached from the initial state by the manipulator (Fillipov's theorem). He also concludes that for almost all times, at least one of the controls should take one of its extreme values.

The existence of regular or singular control solutions of the time-optimal point-to-point control of rigid manipulators is discussed in [31].

A note about the time-optimal control of two kinds of manipulators with relatively simple governing equations and symmetric boundary conditions is presented in [32]. A short discussion about the number of switches, when a singular arc exists, is given.

A short discussion on the time-optimal control of a single link manipulator is presented in [33]. The authors claimed that the number of switches of bang-bang control for a one-bending mode model would be, at most, three.

A numerical method for a special family of the time-optimal control problems (i.e. minimum-time pointing control of a two-link manipulator when the final conditions of the links are not specified completely) is discussed in [34]. An interesting discussion about the pattern of switches of bang-bang control is presented there.

2.4 Numerical Methods for Optimal Control

Numerical solutions are the most common way of solving optimal control problems. The shooting method for optimal control problems was used in [3] to solve a few simple examples for which it was possible to obtain a good guess for initial costates intuitively. It was found that the shooting method is highly sensitive to the guess of the initial values of costates.

A shooting method to find the switching times for the bang-bang control for the near-minimum-time problem is used in [35]. The initial costates are generated with the help of a quasilinearisation technique. The authors also applied their method for solving the minimum time attitude maneuver of a rigid spacecraft [36]. The minimum time is determined by sequentially shortening the slewing time. A set of methods based on the Euler's principal axis rotation is developed to estimate the unknown initial costates and the final time as well as to generate the nominal solutions for starting the quasilinearisation algorithm.

In [37] a perturbation method for the numerical solution of optimisation problems subject to inequality constraints explicitly containing the control is introduced. The solution is not a bang-bang type because of the nature of the constraints on controls. Perturbation is along the nominal trajectory, and uses the linear terms only. Two methods of convergence based on the norm-reduction and on magnitude-limitation were investigated. The latter converges more rapidly than the former. However the latter case was prone to divergence.

A method used in [38] permits determination of the number and the location of the zeros of the switching functions. The cost function includes the switching times and the final time of a bang-bang control. A direct search method (gradient method) is used to minimise the cost function. The expressions for the gradient of the cost function with respect to the switch times are obtained explicitly. The author also applied the approach to a linear plant with bounded control and a quadratic performance index (an optimal regulator problem [39].)

A numerical procedure to compute non-singular, time-optimal solutions for non-linear systems that are linear in control and have fixed initial and final states and bounded control is presented in [40]. The authors introduced a numerical test that determines whether bang-bang solutions satisfy Pontryagin's Minimum Principle. They stated there that *"the test reveals the fact that, for non-linear systems, with linear control and dimension n, the probability that a bang-bang solution with more than*

$n-1$ *switches satisfies Pontryagin's Minimum Principle is almost zero."* This claim is challenged in this thesis (see Section 6.1).

A general purpose subroutine which solves optimal control problems with a fixed terminal time using control parameterisation is introduced in [41]. The approach uses a direct optimisation technique (a zeroth order spline) based on the concept of sequential quadratic programming.

The superiority of the conjugate gradient method over the steepest descent for a class of optimal control problems is claimed in [42]. For a fixed final time with nonlinear or linear state equations and a linear or nonlinear performance index, the conjugate gradient method is employed. The constrained problem is converted to an unconstrained one using the penalty function. The authors conclude that the conjugate gradient method is faster than steepest descent for this group of optimal control problems but slower than second order methods.

In [43], the choice for initial multipliers for the quasilinearisation solution of some optimal control problems is introduced. The authors use an auxiliary minimisation problem with diagonal matrices of weighting coefficients in the performance index. This index includes the cumulative error in the constraints and the optimality conditions in the original problem. The authors believe that if the initial choice of the nominal function for the optimal solution is close enough, the chance of convergence of the quasilinearisation algorithm to the optimal solution of the control problem would be high.

The *method of particular solutions*, which is a general technique for solving nonlinear TPBVP, is introduced in [44]. In that method the equations and the boundary conditions are first linearised about a nominal function which satisfies the boundary conditions. The method of particular solutions is then employed to get the perturbation of nominal function to its varied function.

In [45] a gradient approach with bounded controls is introduced. The method uses an adjustable control-variation weight matrix rather than a penalty function to

enforce the bounded controls. Like other approaches which use the gradient method. it has the weakness that the objective function is independent of the sign of switch functions; consequently, it converges to solutions which are not necessarily bang-bang.

It is notable that some papers do not calculate the costates; the solution (control) then is not verified with the switch functions and may not be optimal. For example. in [46], a numerical method for solving the minimum-time problem of robot manipulator is introduced. The application of the method to the two-link rigid manipulator is presented. In that algorithm a vector of bang-bang control forces as well as the switch times are first assumed. Also, an initial value for the control vector is guessed. With a gradient based method the switch times are corrected in each iteration to minimise the error of the states at the target. However, the solution obtained that way did not consider the costates and was found not to be optimal [47], as will be discussed also later in Section 6.1.

2.5 The Contribution of the Candidate

The author has contributed to the third and fourth groups of the above literature review. For the first time, the complete solutions (containing the controls, the states, and the costates) to the time-optimal maneuver problem of Two-Link Manipulators (TLM) are obtained directly from the PMP. The solutions allow for analysis of some characteristic features of the maneuver, such as the number of switches, for example. Details of the numerical method presented in this thesis have been published in several papers.

A Forward-Backward Method (FBM) that solves the time-optimal control problem (see Section 5.4) is discussed in [47], [48], and [49]. In the method, the switching times were directly approximated in each iteration from a solution of the state equations that meets the initial and final conditions. The FBM generates the initial costates which then are used in the shooting method. The results show that the FBM is

capable of generating a good initial guess for systems with one or two control forces. Cases discussed in [48] had the order of the state equation of less than two. Those cases included a single link robot with a single control force and a linear or nonlinear system with two state variables or less. The results of applying the FBM to two-link manipulators was presented in [47] and [49]. The time-optimal control problem for two-link manipulators was also formulated there. Next, a description of the numerical approach that solves two-point boundary value problem was given. The approach combined the shooting and forward-backward methods in order to obtain the time-optimal control solution. The method was also used to control the motion of flexible manipulators [50]. The finite element method was used to solve the dynamics of flexible manipulators. The expected number of switches for the time-optimal control was investigated in [51].

Chapter 3

Mathematical Statement of Problem

The task of optimal control is to determine the controls (the driving forces and moments) that minimise a performance index and simultaneously satisfy the physical constraints.

For evaluating the performance of a system, selection of a performance measure is necessary. For the time-optimal control problems, the performance measure is clearly the elapsed time. However, in some cases the selection of a performance index is a subjective matter. In the following, a few performance measures indicating different optimal control problems are mentioned.

3.1 Review of Some Optimal Control Problems

The performance functional for an optimal control problem can be written in a different form (see Equation 3.25) as

$$J(u) = h(x(t_f), t_f) + \int_{t_0}^{t_f} g(x(t), u(t), t)dt \tag{3.1}$$

where g and h are functionals of states, controls, and time. Here h represents the state at the target and g is a function depending on the whole time domain. This form can be easily transformed to the form of Equation (1.8). (see Section 3.3.1).

The objective of optimal control is to find control u which causes the system

$$\dot{x}(t) = a(x(t), u(t), t) \tag{3.2}$$

to follow a trajectory x with the given boundary conditions and minimises $J(u)$. Some typical control problems are as follows.

3.1.1 Time Minimum Problem

The task is to transfer a system from initial state $x(t_0) = x_0$ to a target state $x(t_f) = x_f$ in minimum time. The performance index to be minimised is

$$J(u) = \int_{t_0}^{t_f} dt = t_f - t_0 \tag{3.3}$$

3.1.2 Problem of Terminal Control

Deviation of the final state of a system from its desired value $r(t_f)$ must be minimised. The performance measure in this case can be

$$J(u) = h(x(t_f), t_f) = \| x(t_f) - r(t_f) \| \tag{3.4}$$

where $\|\|$ denotes a norm of the vector $[x(t_f) - r(t_f)]$.

3.1.3 Problem of Minimum Control Effort

The problem is to minimise the expenditure of control effort for transferring a system from the initial state x_0 to a target state x_f. The performance index may be in one of the following forms:

$$J(u) = \int_{t_0}^{t_f} | u(t) | dt \tag{3.5}$$

or

$$J(u) = \int_{t_0}^{t_f} (u^T R u) dt \tag{3.6}$$

where R is a real, symmetric, positive definite weighting matrix. The latter performance index is sometimes called a *control energy problem.*

3.1.4 Path Following Problem

A tracking problem or a path following problem is to maintain the state $x(t)$ as close as possible to the desired state $r(t)$ during the motion for $t \in [t_0, t_f]$.

$$J(u) = \int_{t_0}^{t_f} [x(t) - r(t)]^2 dt \tag{3.7}$$

Moreover, if the states are to be close to their desired values at the final time as well as being close to the desired state $r(t)$ during the motion, the performance index can be set to

$$J(u) = \| x(t_f) - r(t_f) \| + \int_{t_0}^{t_f} [x(t) - r(t)]^2 dt \tag{3.8}$$

Clearly, if $x(t) = r(t)$ for the optimal solution the indices defined by (3.7) and (3.8) are equivalent. A special case of the tracking problem is *the regulator problem* when the states are zero all the time.

3.2 Calculus of Variations

The objective in optimal control problems is to determine a function that minimises a functional called the performance measure or performance index. Calculus of Variations is a branch of mathematics which deals with analytical problems of minimising functionals. Typically only well-behaved functions (that is the functions which are continuous up to a sufficient order) are used to define the governing equations (the Euler-Lagrange equations). Since the functions describing the admissible control and the states are less well-behaved (the controls are, in general, discontinuous), the rules

of Calculus of Variations should be slightly modified to be applicable in optimal control. Nevertheless, the strategies used in Calculus of Variations and in Optimal Control are analogous and are briefly discussed in this chapter.

3.2.1 The Fundamental Theorem of Calculus of Variations

Consider the functional (function of function) $J(x)$ defined as

$$J(x) = \int_{t_0}^{t_f} g(x(t), \dot{x}(t), \ddot{x}(t), \ldots, t)dt \tag{3.9}$$

The increment of J, where J is said to be differentiable on x, can be written as

$$\Delta J(x, \delta x) = J(x + \delta x) - J(x) = \delta J(x, \delta x) + \text{higher order terms} \tag{3.10}$$

The variation δJ is linear in δx.

The fundamental theorem of calculus of variations says that if x^* is an extremal which minimises (or maximises) $J(x)$, then the variation of J must vanish:

$$\delta J(x^*, \delta x) = 0 \tag{3.11}$$

for all admissible δx. (The admissibility of δx means that $x^* + \delta x$ is a member of the same class as x^* and δx, e.g. both are continuous functions.) The proof of this theorem can be found in [5].

3.2.2 Variation of Functionals of Several Functions

Consider a functional of the vector x made of n independent functions x_i (where $i = 1 \ldots n$) and their first derivatives. The functional $J(x)$ is

$$J(x) = \int_{t_0}^{t_f} g(x(t), \dot{x}(t), t)dt \tag{3.12}$$

This case is commonly used in mechanics where $x \in C^1$ (meaning that the functions x and their first derivatives are continuous). Also g has continuous partial first and second derivatives with respect to x, $\dot{x}$, and t. Assume that t_0 and $x(t_0)$ are specified and t_f and $x(t_f)$ are not specified (Figure 3.1).

To make use of (3.11) the extremal x^* and a neighbouring curve $x = x^* + \delta x$ can be compared. Note that $\delta x(t_f)$ is the difference in x at t_f and δx_f is the difference in x of the endpoints of the two curves. In general $\delta x(t_f) \neq \delta x_f$.

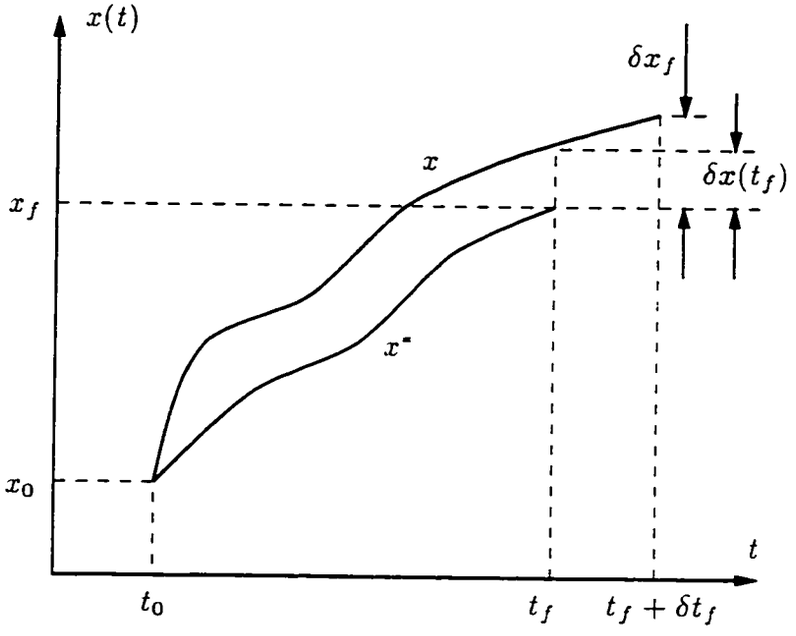

Figure 3.1: Extremal x^* and another neighbouring curve x for free final point.

The fundamental theorem of calculus of variations must be used to find the necessary conditions for functions x that must be satisfied to obtain minimum of J. The increment $\Delta J(x, \delta x)$ using the standard manipulations can be determined following [5]. The star denoting the extremal curve is omitted for convenience.

$$\begin{aligned} \Delta J(x, \delta x) &= J(x + \delta x) - J(x) \\ &= \int_{t_0}^{t_f + \delta t_f} g(x + \delta x, \dot{x} + \delta \dot{x}, t)dt - \int_{t_0}^{t_f} g(x, \dot{x}, t)dt \end{aligned} \tag{3.13}$$

$$= \int_{t_0}^{t_f} \left\{ \frac{\partial g}{\partial x}(x, \dot{x}, t) - \frac{d}{dt}\left[\frac{\partial g}{\partial \dot{x}}(x, \dot{x}, t)\right] \right\} (\delta x) dt + O(\cdot)$$
$$+ \left[\frac{\partial g}{\partial \dot{x}}(x, \dot{x}, t)\right]_{t_f} \delta x(t_f) + [g(x, \dot{x}, t)]_{t_f} \delta t_f$$

Where $O(.)$ denotes terms of higher than first order. From Figure 3.1, it can be seen

$$\delta x(t_f) = \delta x_f - \dot{x}(t_f)\delta t_f \tag{3.14}$$

substituting the equation (3.14) into (3.13) one can get the variation $\delta J(x, \delta x)$ (first order terms of the increment ΔJ) as

$$\delta J(x, \delta x) = \int_{t_0}^{t_f} \left\{ \frac{\partial g}{\partial x}(x, \dot{x}, t) - \frac{d}{dt}\left[\frac{\partial g}{\partial \dot{x}}(x, \dot{x}, t)\right] \right\} (\delta x) dt \tag{3.15}$$
$$+ \left[\frac{\partial g}{\partial \dot{x}}(x, \dot{x}, t)\right]_{t_f} \delta x_f + \left[g(x, \dot{x}, t) - \left(\frac{\partial g}{\partial \dot{x}}(x, \dot{x}, t)\right)\dot{x}\right]_{t_f} \delta t_f = 0$$

For this variation to vanish for arbitrary δx (the fundamental theorem of calculus of variation) the following conditions must be satisfied.

$$\frac{\partial g}{\partial x}(x, \dot{x}, t) - \frac{d}{dt}\left[\frac{\partial g}{\partial \dot{x}}(x, \dot{x}, t)\right] = 0 \tag{3.16}$$

for all $t \in [t_0, t_f]$. This is the *Euler-Lagrange equation* for the problem (3.12), the integration of which enables one to define the optimal x. The second part of (3.15) involves the boundary condition at the final time which, if (3.16) is met, makes the variation δJ equal to:

$$\delta J(x, \delta x) = \left[\frac{\partial g}{\partial \dot{x}}(x, \dot{x}, t)\right]_{t_f} \delta x_f + \left[g(x, \dot{x}, t) - \left(\frac{\partial g}{\partial \dot{x}}(x, \dot{x}, t)\right)\dot{x}\right]_{t_f} \delta t_f = 0 \tag{3.17}$$

Note that the conditions $x(t_0) = x_0$ is assumed to be always satisfied. Equations (3.16) and (3.17) are *the necessary conditions* that must be satisfied by an extremal

curve.

Boundary conditions

From (3.17) the boundary conditions at the final time can be easily specified for some particular cases. Three such cases are as follows.

Case 1: If $x(t_f)$ is specified and t_f is free, then $\delta x_f = 0$, and the boundary conditions are:

$$\begin{aligned} x(t_f) &= x_f \\ g(x, \dot{x}, t)_{t_f} &- \left(\frac{\partial g}{\partial \dot{x}}(x, \dot{x}, t)\right)_{t_f} \dot{x}(t_f) = 0 \end{aligned} \tag{3.18}$$

Case 2: If $x(t_f)$ is free and t_f is specified, then $\delta t_f = 0$, $\delta x_f \neq 0$, and from (3.17) then:

$$\frac{\partial g}{\partial \dot{x}}(x, \dot{x}, t)_{t_f} = 0 \tag{3.19}$$

Case 3: If both $x(t_f)$ and t_f are free and independent of each other, then the boundary conditions which satisfy (3.17) are:

$$\begin{aligned} &\frac{\partial g}{\partial \dot{x}}(x, \dot{x}, t)_{t_f} = 0 \\ &g(x, \dot{x}, t)_{t_f} = 0 \end{aligned} \tag{3.20}$$

Note that Equation (3.16) with appropriate boundary conditions can be written for each component of the vector x.

3.3 Calculus of Variations and Optimal Control

In the discussion above it was assumed that the components of x are independent of each other. However, in control problems the functional representing the performance

index contains n states x and m controls u. Thus, such functional includes $n + m$ functions of x and u, but only m of them are independent (controls u). The state equations imposes n constraints on x and u in the form of (3.2). Now the necessary conditions (3.16) and (3.17) have to be extended to handle such a constraint problem.

When the controls are not bounded the necessary conditions of optimality can still be derived using classical calculus of variations. However, if the controls are bounded, some continuity requirements used in calculus of variations are not met. Therefore, a generalisation of the fundamental theorem of the calculus of variations has to be introduced. This generalisation will lead to the *Pontryagin's Minimum Principle (PMP)*. Time-optimal control problems using the PMP are discussed at the end of this section.

3.3.1 Unbounded Optimal Control

Consider the admissible control vector u containing m components that drives the system defined by n components of the state vector x and satisfying the equation

$$\dot{x}(t) = a(x(t), u(t), t) \tag{3.21}$$

Optimal control must minimise the performance index

$$J(u) = h(x(t), t)_{t_f} + \int_{t_0}^{t_f} g(x(t), u(t), t)dt \tag{3.22}$$

Here it is assumed that $x(t_0)$ and t_0 are fixed, and x is determined by control u which does not have limits (unbounded control). This problem is considered as an initial fixed point and final free point problem in the calculus of variations.

To be consistent with the formulation in the previous section, $h(x, t)_{t_f}$ can be

included in the integral sign assuming that h is differentiable such that

$$J(u) = \int_{t_0}^{t_f} \left\{ g(x(t), u(t), t) + \frac{d}{dt} h(x(t), t) \right\} dt \tag{3.23}$$

In order to include the constraints (3.21) the augmented functional I is formed by introducing n Lagrange multipliers as the *costates* vector $p(t)$. One can write (3.23) as

$$\begin{aligned} I(u) \;=\; & \int_{t_0}^{t_f} \left\{ g(x(t), u(t), t) + \left[\frac{\partial h}{\partial x}(x(t), t) \right] \dot{x}(t) + \frac{\partial h}{\partial t}(x(t), t) \right. \\ & \left. + p^T(t) \left[a(x(t), u(t), t) - \dot{x}(t) \right] \right\} dt \end{aligned} \tag{3.24}$$

$I(u)$ by analogy to (3.12) can be rewritten as

$$I(u) = \int_{t_0}^{t_f} g_a(x(t), \dot{x}(t), u(t), p(t), t) dt \tag{3.25}$$

where

$$g_a(x, \dot{x}, u, p, t) = g(x, u, t) + p^T \left[a(x, u, t) - \dot{x} \right] + \left[\frac{\partial h}{\partial x}(x, t) \right] \dot{x} + \frac{\partial h}{\partial t}(x, t) \tag{3.26}$$

Replacing $g(x, u, t)$ by $g_a(x, \dot{x}, u, p, t)$ in (3.15) and introducing the variations δx, $\delta \dot{x}$, δu, δp, and δt_f the variation of I can be obtained in the form

$$\begin{aligned} \delta I(u) = \int_{t_0}^{t_f} & \left\{ \left[\frac{\partial g_a}{\partial x}(x(t), \dot{x}(t), u(t), p(t), t) - \frac{d}{dt} \frac{\partial g_a}{\partial \dot{x}}(x(t), \dot{x}(t), u(t), p(t), t) \right] \delta x(t) \right. \\ & + \left[\frac{\partial g_a}{\partial u}(x(t), \dot{x}(t), u(t), p(t), t) \right] \delta u(t) + \left[\frac{\partial g_a}{\partial p}(x(t), \dot{x}(t), u(t), p(t), t) \right] \delta p(t) \Big\} dt + \\ & + \left[\frac{\partial g_a}{\partial \dot{x}}(x, \dot{x}, u, p, t) \right]_{t_f} \delta x_f + \left[g_a(x, \dot{x}, u, p, t) - \left(\frac{\partial g_a}{\partial \dot{x}}(x, \dot{x}, u, p, t) \right) \dot{x} \right]_{t_f} \delta t_f = 0 \end{aligned} \tag{3.27}$$

If the chain rule is applied to the terms involving h inside the integral, they will cancel

each other out. Then the remaining terms of the integral are

$$\int_{t_0}^{t_f} \Bigg\{ \left[\left[\frac{\partial g}{\partial x}(x(t), u(t), t) \right] + p^T(t) \left[\frac{\partial a}{\partial x}(x(t), u(t), t) \right] + \frac{d}{dt} p^T(t) \right] \delta x(t)$$
$$+ \left[\left[\frac{\partial g}{\partial u}(x(t), u(t), t) \right] + p^T(t) \left[\frac{\partial a}{\partial u}(x(t), u(t), t) \right] \right] \delta u(t)$$
$$+ \Big[a(x(t), u(t), t) - \dot{x}(t) \Big] \delta p(t) \Bigg\} dt \tag{3.28}$$

Regardless of the boundary conditions this integral must be equal to zero for $\delta I(u)$ to vanish for admissible variations δx, δu, and δp. If the constraints in (3.21) are satisfied, then the coefficient of the $\delta p(t)$ is automatically zero. The state equations represented by (3.21) is repeated here for convenience:

$$\dot{x}(t) = a(x, u, t) \tag{3.29}$$

Since the costates p are arbitrary, one can select them in such a way that the coefficient of $\delta x(t)$ also vanishes on the extremal, that is

$$\dot{p}(t) = -p^T(t) \left[\frac{\partial a}{\partial x}(x, u, t) \right] - \frac{\partial g}{\partial x}(x, u, t) \tag{3.30}$$

This set of n ODEs is called the *costate equations*. Since the variation of $\delta u(t)$ is independent, its coefficient must be set to zero as well, resulting in

$$\frac{\partial g}{\partial u}(x, u, t) + p^T(t) \left[\frac{\partial a}{\partial u}(x, u, t) \right] = 0 \tag{3.31}$$

This set of m algebraic equations is to define the controls u as a function of x and p. In mechanics $a(x, u, t)$ is a linear function of u; if $g(x, u, t)$ is also linear in terms of u (or independent of u), then Equation (3.31) can not be used. Also, this equation is not valid for bounded or discontinuous controls. For the cases mentioned above, Equation (3.31) must be modified as explained in Section 3.3.2.

The boundary conditions are obtained by setting the remaining part of the vari-

ation $\delta I(u)$ given by (3.27) to zero. Substituting back the function g_a as defined by (3.26) into the term representing boundary condition gives

$$\left[\frac{\partial h}{\partial x}(x,t) - p(t)\right]_{t_f} \delta x_f + \left[g(x,u,t) + \frac{\partial h}{\partial t}(x,t) + p^T(t)[a(x,u,t)]\right]_{t_f} \delta t_f = 0 \quad (3.32)$$

The above equation is similar to (3.17). This boundary condition must be satisfied at the final time.

Note that n state equations (3.29), n costate equations (3.30), and a set of m algebraic relations (3.31) constitute the *necessary conditions* which must be satisfied throughout the interval $[t_0, t_f]$.

By defining the following function $\mathcal{H}$, called the *Hamiltonian*,

$$\mathcal{H}(x,u,p,t) = g(x,u,t) + p^T(t)[a(x,u,t)] \quad (3.33)$$

the necessary conditions (3.29), (3.30), (3.31) can be written as:

$$\dot{x}(t) = \frac{\partial \mathcal{H}}{\partial p}(x,u,p,t) \quad (3.34)$$

$$\dot{p}(t) = -\frac{\partial \mathcal{H}}{\partial x}(x,u,p,t) \quad (3.35)$$

$$0 = \frac{\partial \mathcal{H}}{\partial u}(x,u,p,t) \quad (3.36)$$

These equations must be satisfied at all time $t \in [t_0, t_f]$. The boundary conditions at initial time t_0 are $x(t_0) = x_0$ and at final time t_f as given by (3.32) which can be rewritten as:

$$\left[\frac{\partial h}{\partial x}(x,t) - p(t)\right]_{t_f} \delta x_f + \left[\mathcal{H}(x,u,p,t) + \frac{\partial h}{\partial t}(x,t)\right]_{t_f} \delta t_f = 0 \quad (3.37)$$

The conditions (3.34) and (3.35) form a set of $2n$ first order ODE's and the condition (3.36) a set of m algebraic relations. The set of functions to be determined includes n state variables x, n costate variables p, and m controls u.

Boundary conditions:

The initial condition is always given as $x(t_0) = x_0$. The boundary condition at the final time can be obtained from (3.37) as follows:

Final time fixed: If the final time t_f is specified, then $\delta t_f = 0$. If the final state is also specified then $\delta x_f = 0$ and (3.37) is satisfied automatically. The boundary conditions at t_f are

$$x(t_f) = x_f \tag{3.38}$$

If the final state is free, then $\delta x_f \neq 0$. Using (3.37) the following boundary conditions at t_f can be obtained

$$\frac{\partial h}{\partial x}(x,t)_{t_f} - p(t_f) = 0 \tag{3.39}$$

Final time free: If the final time t_f is free, then $\delta t_f \neq 0$. When the final state $x(t_f)$ is fixed, then $\delta x_f = 0$. Using (3.37) the boundary conditions are

$$\begin{aligned} &x(t_f) = x_f \\ &\mathcal{H}(x,u,p,t)_{t_f} + \frac{\partial h}{\partial t}(x,t)_{t_f} = 0 \end{aligned} \tag{3.40}$$

This case is characteristic for the time-optimal control problem for which the initial and final configuration of the links are known and the final time is minimised.

When the final state is free, then $\delta x_f \neq 0$ and $\delta t_f \neq 0$. Using (3.37) provides the following boundary conditions at the final time.

$$\begin{aligned} &\frac{\partial h}{\partial x}(x,t)_{t_f} - p(t_f) = 0 \\ &\mathcal{H}(x,u,p,t)_{t_f} + \frac{\partial h}{\partial t}(x,t)_{t_f} = 0 \end{aligned} \tag{3.41}$$

There are other possibilities for final states that are not considered here. The char-

acteristic feature of these conditions is that they are imposed on *both ends* of the trajectory at t_0 and t_f creating the two-point boundary value problem as mentioned before.

3.3.2 Bounded Controls

Up to now, the admissible controls were not constrained. However, in real situations, physical actuators have limitations. For example, electrical and mechanical motors can produce torques only up to a certain amount. The necessary conditions derived before have to be modified to deal with such limits on controls. The generalisation of the fundamental theorem of Calculus of Variations when the controls are limited or discontinued is called Pontryagin's Minimum Principle.

If u is the optimal control, for all admissible controls close to this control, the functional J has relative minimum when

$$\Delta J(u, \delta u) = J(u + \delta u) - J(u) = \delta J(u, \delta u) + \text{higher order terms} \geq 0 \qquad (3.42)$$

where δJ is linear with respect to δu. When the norm of δu approaches zero, the higher order terms approach zero too. As in the previous section (unbounded control), a necessary condition for u to be extremal is that the variation $\delta J(u, \delta u)$ be zero for all admissible δu with a sufficiently small norm. However since the controls are bounded, some of the variation of δu might lie in an inadmissible region. So, when all admissible variations $\| \delta u \|$ are small enough such that the sign of ΔJ is determined by δJ, a necessary condition for u to minimise J is

$$\delta J(u, \delta u) \geq 0 \qquad (3.43)$$

when u lies *on the boundary of admissible controls* in any time interval $t \in [t_0, t_f]$, and

$$\delta J(u, \delta u) = 0 \qquad (3.44)$$

when u is *within the boundary of admissible controls* for the whole time interval $[t_0, t_f]$.

Using the definition of the Hamiltonian, and (3.27), 3.28), the variation $\delta J(u, \delta u)$ has the form

$$\begin{aligned}\delta J(u,\delta u) &= \int_{t_0}^{t_f}\left\{\left[\frac{\partial \mathcal{H}}{\partial x}(x,u,p,t)+\dot{p}(t)\right]\delta x+\left[\frac{\partial \mathcal{H}}{\partial p}(x,u,p,t)-\dot{x}(t)\right]\delta p\right. \\ &\left.+\left[\frac{\partial \mathcal{H}}{\partial u}(x,u,p,t)\right]\delta u\right\}dt+\left[\frac{\partial h}{\partial x}(x,t)-p\right]_{t_f}\delta x_f \\ &+\left[\mathcal{H}(x,u,p,t)+\frac{\partial h}{\partial t}(x,t)\right]_{t_f}\delta t_f \end{aligned} \tag{3.45}$$

When the costates are selected to satisfy (3.35), the coefficient of δx in the integral vanishes. If the state equations (3.34) are satisfied, the coefficient of δp in the integral vanishes. If also the boundary conditions (3.37) are satisfied, from (3.45) one can have

$$\delta J(u,\delta u)=\int_{t_0}^{t_f}\left[\frac{\partial \mathcal{H}}{\partial u}(x,u,p,t)\right]\delta u dt \tag{3.46}$$

which can be written as

$$\delta J(u,\delta u)=\int_{t_0}^{t_f}[\mathcal{H}(x,u+\delta u,p,t)-\mathcal{H}(x,u,p,t)]dt \tag{3.47}$$

Since $\delta J(u, \delta u) \geq 0$ from (3.43), the necessary condition for u to minimise the functional J is

$$\mathcal{H}(x,\overbrace{u+\delta u}^{\hat{u}},p,t)\geq \mathcal{H}(x,u,p,t) \tag{3.48}$$

$$\text{or} \qquad \mathcal{H}(x,\hat{u},p,t)\xrightarrow{\hat{u}}\min$$

for all admissible variation δu and for all $t \in [t_0, t_f]$. Here $\hat{u}$ defines any admissible control including non-optimal controls. This equation shows that *optimal control must minimise the Hamiltonian.* It represents an extension of (3.36) for the cases in which u is bounded or discontinued.

In summary Pontryagin's Minimum Principle for bounded controls is

$$\dot{x}(t) = \frac{\partial \mathcal{H}}{\partial p}(x, u, p, t) \tag{3.49}$$

$$\dot{p}(t) = -\frac{\partial \mathcal{H}}{\partial x}(x, u, p, t) \tag{3.50}$$

$$\mathcal{H}(x, u, p, t) \leq \mathcal{H}(x, \hat{u}, p, t) \tag{3.51}$$

for all time $t \in [t_0, t_f]$, where u is optimal control and $\hat{u}$ is admissible control. The boundary condition at final time t_f is identical to (3.37), that is

$$\left[\frac{\partial h}{\partial x}(x,t) - p(t)\right]_{t_f} \delta x_f + \left[\mathcal{H}(x, u, p, t) + \frac{\partial h}{\partial t}(x,t)\right]_{t_f} \delta t_f = 0 \tag{3.52}$$

It should be mentioned that u is a control that causes $\mathcal{H}(x, u, p, t)$ to assume its absolute or global minimum. Also, the set of equations (3.49), (3.50), (3.51), (3.52) are *necessary conditions* of optimality in general; that is, Pontryagin's Minimum Principle can be applied to problems with bounded or unbounded controls.

Other necessary conditions:

The variation of $\mathcal{H}(x, u, p, t)$ in Equation (3.47) is

$$\delta\mathcal{H} = \frac{\partial \mathcal{H}}{\partial x}\delta x + \frac{\partial \mathcal{H}}{\partial p}\delta p + \frac{\partial \mathcal{H}}{\partial u}\delta u + \frac{\partial \mathcal{H}}{\partial t}\delta t$$

Using the necessary conditions (3.34), (3.35), (3.36) one can have

$$\frac{\partial \mathcal{H}}{\partial x}\delta x + \frac{\partial \mathcal{H}}{\partial p}\delta p = -\dot{p}\delta x + \dot{x}\delta p = -\dot{p}\dot{x}\delta t + \dot{x}\dot{p}\delta t = 0$$

For optimal control either $\frac{\partial \mathcal{H}}{\partial u} = 0$ for u within the bound, or $\delta u = 0$ for u on the bound; therefore, $\delta\mathcal{H} = \frac{\partial \mathcal{H}}{\partial t}\delta t$. When the Hamiltonian does not depend on time

explicitly, then along any optimal trajectory:

$$\delta\mathcal{H} = 0 \tag{3.53}$$

There are two more necessary conditions when the Hamiltonian does not depend on time explicitly, derived from (3.53) by Pontryagin [52] as follows.

1. When the final time is fixed ($\delta t_f = 0$) then, using (3.53), the Hamiltonian must be constant for all time $t \in [t_0, t_f]$.

$$\mathcal{H}(x, u, p) = \text{constant} \tag{3.54}$$

2. When the final time is free ($\delta t_f \neq 0$), and when $\frac{\partial h}{\partial t} = 0$ then, from (3.52) and (3.53), the Hamiltonian must be zero identically on a trajectory for all time $t \in [t_0, t_f]$.

$$\mathcal{H}(x, u, p) = 0 \tag{3.55}$$

3.4 Time-Optimal Control Problem

The objective of the time-optimal control is to transfer a system from an arbitrary initial state to a target state in minimum time. Then the performance measure is

$$J(u) = \int_{t_0}^{t_f} dt = t_f - t_0 \tag{3.56}$$

where t_0 is always known. The PMP is used to determine the time-optimal control for a particular class of systems governed by the equation of motion in the form.

$$\dot{x}(t) = a(x, t) + c(x, t)u(t) \tag{3.57}$$

where x is the vector of n state variables, a is an n-dimensional vector and c is an $n \times m$ matrix of nonlinear functions of states. Note that the states are linear in controls,

which is the case for mechanical systems. For a time-optimal control problem, the state departing from the initial conditions:

$$x(t_0) = x_0 \tag{3.58}$$

must reach the final conditions:

$$x(t_f) = x_f \tag{3.59}$$

in a minimum time. Each component of the admissible control vector u is bounded as

$$U_i^- \leq \hat{u}_i(t) \leq U_i^+ \quad i = 1, \ldots, m \tag{3.60}$$

for $t \in [t_0, t_f]$ where U_i^- and U_i^+ are lower and upper bounds of controls.

Since $h(x,t)_{t_f} = 0$ and $g(x,u,t) = 1$ in the performance measure (3.56), the Hamiltonian $\mathcal{H}(x,u,p,t)$ defined by (3.33) is now given as:

$$\mathcal{H}(x,u,p,t) = 1 + p^T(t)[a(x,t) + c(x,t)u(t)] \tag{3.61}$$

Note that for this case, because $\mathcal{H}$ is a linear function of u and $g = 1$, the condition (3.36) does not provide any useful relation between the controls and the states.

Using Pontryagin's Minimum Principle, the optimal solution must satisfy the necessary conditions (3.49), (3.50), (3.51):

$$\dot{x}(t) = \frac{\partial \mathcal{H}}{\partial p}(x,u,p,t) \tag{3.62}$$

$$\dot{p}(t) = -\frac{\partial \mathcal{H}}{\partial x}(x,u,p,t) \tag{3.63}$$

$$\mathcal{H}(x,u,p,t) \leq \mathcal{H}(x,\hat{u},p,t) \tag{3.64}$$

Substituting (3.61) into (3.64) one can have

$$p^T(t)c(x,t)u(t) \leq p^T(t)c(x,t)\hat{u}(t) \tag{3.65}$$

for all admissible controls $\hat{u}(t)$ and for all $t \in [t_0, t_f]$. The optimal control $u(t)$ causes $p^T(t)c(x,t)\hat{u}(t)$ to take its minimum value (the term $p^T(t)c(x,t)\hat{u}(t)$ should be minimised with respect to any control $\hat{u}(t)$). It forces the control $\hat{u}(t)$ to take its extremal values. Since the components of the controls are independent of each other, the optimal controls $u_i(t)$ to satisfy (3.64) or (3.65) must be assumed as:

$$u_i(t) = \left\{ \begin{array}{lll} U_i^+ & \text{for} & p^T(t)c_i(x,t) < 0 \\ U_i^- & \text{for} & p^T(t)c_i(x,t) > 0 \\ \text{unknown} & \text{for} & p^T(t)c_i(x,t) = 0 \end{array} \right\} \tag{3.66}$$

Here $c_i(x,t)$ is the ith column of matrix $c(x,t)$. The function $G_i(x,p,t) = p^T(t)c_i(x,t)$ is called the switch function corresponding to the control u_i. When $G_i(x,p,t) = p^T(t)c_i(x,t) = 0$ for a period of time, the PMP is not able to define any optimal value of u_i. It is called a *singular* condition, and the control is referred to as singular control. Thus, for singular control, the necessary condition that u must minimise $\mathcal{H}$ provides no information about the value of the controls u. If the switch functions G_i are zero at individual points only, the control is nonsingular. This type of control (3.66) is referred to as *bang-bang control.* The controls take their extremal values throughout the whole motion to minimise the maneuver time.

Additionally, since the Hamiltonian is not an explicit function of time, it must satisfy the additional necessary condition (3.55):

$$\mathcal{H}(x,u,p) = 0 \text{ for all } t \tag{3.67}$$

The set of differential equations (3.62) and (3.63), the requirements (3.66) and (3.67), and the initial and the final conditions (3.58) and (3.59) completely define a time-optimal control problem. This problem belongs to a class of two-point boundary value problems. The solution of n state (x) and n costate (p) equations must satisfy $2n$ initial and final conditions imposed on the state only. It is assumed that any

optimal trajectory remains bounded on a closed, bounded interval containing t_0 and t_f.

If the state equations are nonlinear, from the mathematical view point there is little known about characteristics of the above problem. However, the existence of the solution was considered and proven in [6], [30]. For time-invariant linear systems of the form:

$$\dot{x}(t) = Ax + Cu \tag{3.68}$$

where A and C are constant matrices, the following observation can be made [5].

1. If all of the eigenvalues of A have non-positive real parts, then there is an optimal control that transfers any initial state x_0 to the origin.
2. If an optimal control exists, it is unique. In other words, the PMP is both a necessary and sufficient condition for the time-optimal control of such systems.
3. If all of the eigenvalues of A are real, and a time-optimal control exists, then each control component can switch at most $(n-1)$ times when n is the number of states.

The case of Two-Link Manipulators (TLM) considered here is inherently nonlinear and the above observations do not apply. In [40] an attempt was made to extend rule 3 to a TLM. However, as it will be shown in Chapter 6, such an extension, in general, is not correct.

Chapter 4

Time-Optimal Control of Two-Link Manipulators

4.1 Necessary Conditions

The purpose of this chapter is to derive the time-optimal control problem for the Two-Link Manipulators (TLM) shown in Figure 4.1. Here, the states $x_1 = \varphi_1, x_2 = \dot{\varphi}_1$ are rotation and angular velocity of the first link (shoulder link), and the states $x_3 = \varphi_2, x_4 = \dot{\varphi}_2$ are rotation and angular velocity of the second link (elbow link) as it is shown in Figure 4.1.

The TLM is considered rigid and driven by two control torques u_1 and u_2 applied in the first joint (shoulder joint) and the second joint (elbow joint). Any given mass or geometric characteristics of the links as well as the joints of a real manipulator can be specified in terms of the parameters indicated in the Figure 4.1.

Here the objective is to transfer a system moving in a plane (y, z), from an arbitrary initial state x_0 to a target state x_f in minimum time. Then the performance measure is to minimise the maneuver time t_f

$$J = \int_{t_0}^{t_f} dt = t_f - t_0 \tag{4.1}$$

Since t_0 can be set arbitrarily (here $t_0 = 0$), the time-optimal control problem for the TLM can be summarised as follows

$$\text{minimise} \quad J = t_f$$

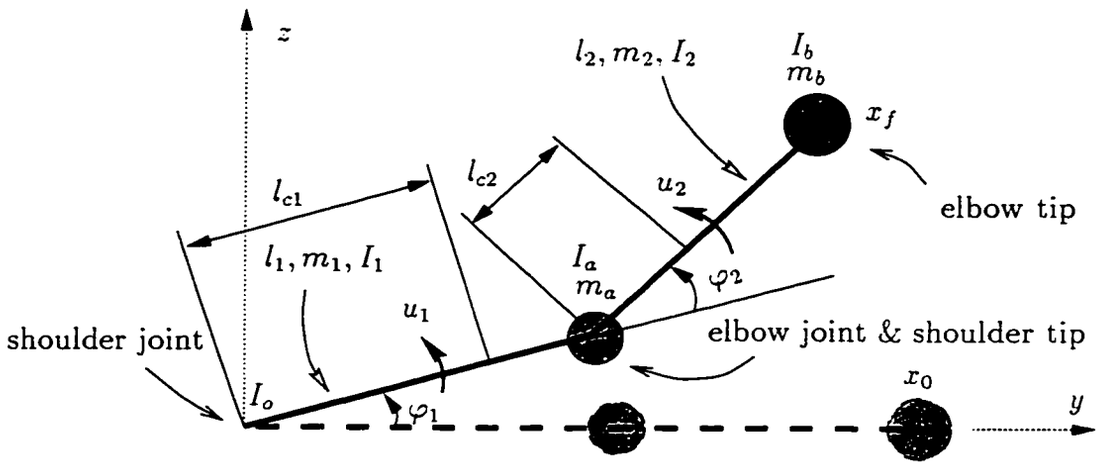

Figure 4.1: Physical parameters used in formulation of two-link manipulators.

subject to the equation of motion in the form

$$\dot{x}_i(t) = a_i(x) + c_{ij}(x)u_j(t) \qquad i = 1, \ldots, 4, \; j = 1, 2 \tag{4.2}$$

where x is vector of 4 state variables, a is vector of 4 nonlinear functions of states, and c is 4×2 matrix of nonlinear functions of states. The detailed form of (4.2) is derived in Section 4.2.

The boundary conditions are:

$$\begin{aligned} x_0 &= \left\{ \begin{array}{cccc} x_{10} & x_{20} & x_{30} & x_{40} \end{array} \right\}^T \\ x_f &= \left\{ \begin{array}{cccc} x_{1f} & x_{2f} & x_{3f} & x_{4f} \end{array} \right\}^T \end{aligned} \tag{4.3}$$

The Hamiltonian $\mathcal{H}$ is defined as

$$\mathcal{H}(x, u, p, t) = 1 + p^T(t)[a(x) + c(x)u(t)] \tag{4.4}$$

As it was stated in the previous chapter, using Pontryagin's Minimum Principle (PMP), the optimal solution must satisfy the necessary conditions (3.62), (3.63),

(3.64):

$$\dot{x}(t) = \frac{\partial \mathcal{H}}{\partial p}(x, u, p, t) \tag{4.5}$$

$$\dot{p}(t) = -\frac{\partial \mathcal{H}}{\partial x}(x, u, p, t) \tag{4.6}$$

$$\mathcal{H}(x, u, p, t) \leq \mathcal{H}(x, \hat{u}, p, t) \tag{4.7}$$

Due to the special form of (4.4), the costate equations can be obtained as:

$$\dot{p_i}(t) = p_k \; f_{ik}(x, u) \qquad i, k = 1, \ldots, 4 \tag{4.8}$$

where the functions $f_{ik}(x, u)$ are nonlinear function of states and controls. These functions are defined in Section 4.3.

The admissible control bounds $\hat{u}$ are

$$\left\{ \begin{array}{l} U_1^- \leq \hat{u}_1(t) \leq U_1^+ \\ U_2^- \leq \hat{u}_2(t) \leq U_2^+ \end{array} \right\} \tag{4.9}$$

for $t \in [t_0, t_f]$ where U_i^- and U_i^+ are lower and upper bounds of controls. The forces for time-optimal control obtained from the PMP are:

$$u_i(t) = \left[\begin{array}{lll} U_i^+ & for & G_i < 0 \\ U_i^- & for & G_i > 0 \end{array} \right] \tag{4.10}$$

where $G_i(x, p) = \sum_{j=1}^{4} p_j(t) c_{ji}(x, t)$ is the switch function corresponding to the control u_i, and $c_i(x, t)$ is the ith column of matrix $c(x, t)$. The function G_i are formulated in detail in Section 4.4.

Since the final time is free and the Hamiltonian does not depend on time explicitly, it must be identically zero on an optimal trajectory for all time $t \in [t_0, t_f]$, or

$$\mathcal{H}(x, u, p) = 0 \tag{4.11}$$

4.2 The State Equations of Dynamics

Since the rigid TLM is a two degree of freedom system, its equation of motion can be derived by minimising the kinetic and potential energies of the system. The resulting set of equations is called the *Lagrange equations* [53]. For the TLM one has to include the kinetic and potential energies due to rotations and translations of the links and joints.

The energy functional I_E for the rigid TLM can be written as:

$$I_E(\varphi) = \int_{t_0}^{t_f} \mathcal{L}(\varphi(t), \dot{\varphi}(t), t)dt \tag{4.12}$$

where $\mathcal{L}$ is the *Lagrangian* defined as

$$\mathcal{L}(\varphi, \dot{\varphi}, t) = E_k(\varphi, \dot{\varphi}, t) - E_p(\varphi, t) \tag{4.13}$$

where E_k, E_p are the kinetic and the potential energies of the TLM, respectively. Minimising the functional energy (4.12), the following Lagrange equations are obtained:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\varphi}}(\varphi, \dot{\varphi}, t) - \frac{\partial \mathcal{L}}{\partial \varphi}(\varphi, \dot{\varphi}, t) = Q^T \tag{4.14}$$

where Q is the vector of generalised forces. For the TLM the potential energy is independent of the velocity $\left(\frac{\partial E_p}{\partial \dot{\varphi}} = 0\right)$; therefore, substituting (4.13) into (4.14) gives

$$\frac{d}{dt}\frac{\partial E_k}{\partial \dot{\varphi}} - \frac{\partial E_k}{\partial \varphi} + \frac{\partial E_p}{\partial \varphi} = Q^T \tag{4.15}$$

The kinetic energy of the TLM with rigid links shown in Figure 4.2 can be written as follows (terms in the first bracket represent the kinetic energy of the first link and the terms in the second bracket are the kinetic energy of the second link):

$$2 \times E_k = \left\{(I_0 + I_1 + I_a)\omega_1^2 + m_1 v_{c1}^2 + m_a v_a^2\right\} + \left\{(I_2 + I_b)\omega_2^2 + m_2 v_{c2}^2 + m_b v_b^2\right\} \tag{4.16}$$

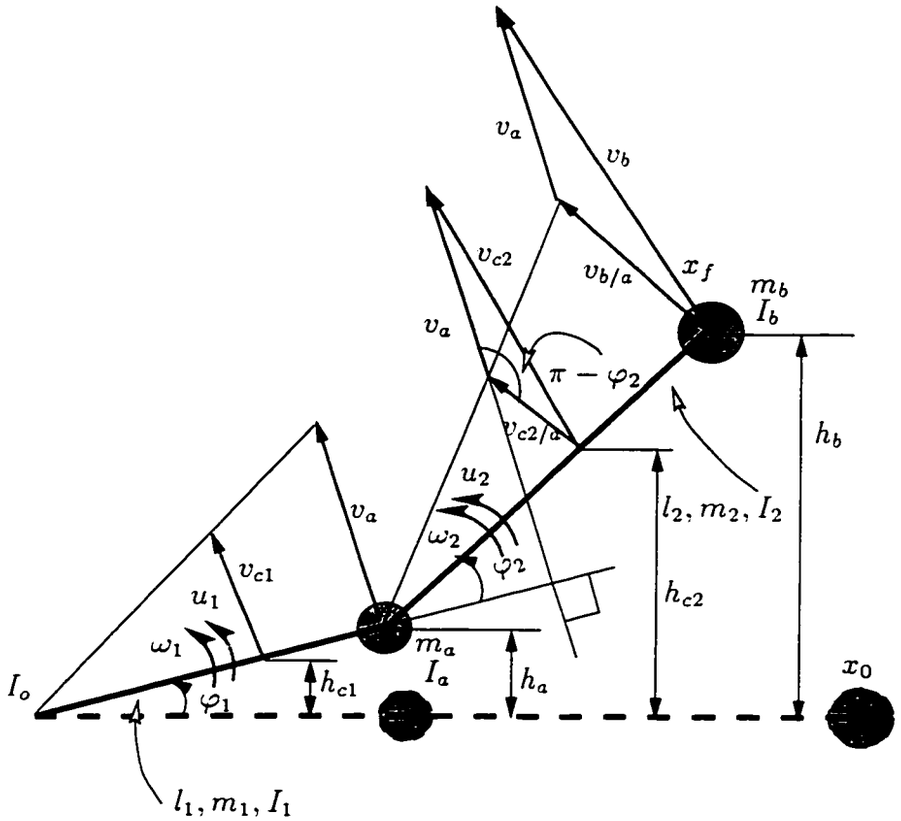

Figure 4.2: Motion of rigid two-link manipulators.

where I_1 and I_2 are mass moment of inertia of the links w.r.t. their centres of gravity, I_a and I_b are mass moment of inertia of the masses m_a and m_b w.r.t. their centres. I_0 is mass moment of inertia of the mass at the shoulder joint including the inertia of the shoulder motor. The terms m_1 and m_2 are the masses of the shoulder and elbow links, respectively, m_a is mass at the elbow joint, and m_b mass at the end of the manipulator. The terms v_{c1} and v_{c2} are linear velocities of the centres of the links, and v_a and v_b are linear velocities of tips of the links. The terms ω_1 and ω_2 are angular velocities of the links as well as the angular velocities for m_a and m_b, respectively. The lengths l_1 and l_2 are the lengths of the links, l_{c1} and l_{c2} locate the centres of gravity of the links.

The linear and angular velocities in Equation (4.16) can be obtained as:

$$\begin{aligned} \omega_1 &= \dot{\varphi}_1 & v_{c1} &= \omega_1 l_{c1} & v_a &= \omega_1 l_1 \\ \omega_2 &= \dot{\varphi}_1 + \dot{\varphi}_2 & v_{c2/a} &= \omega_2 l_{c2} & v_{b/a} &= \omega_2 l_2 \end{aligned}$$

$$\begin{aligned} v_{c2}^2 &= v_a^2 + v_{c2/a}^2 + 2 v_a v_{c2/a} \cos(\varphi_2) \\ v_b^2 &= v_a^2 + v_{b/a}^2 + 2 v_a v_{b/a} \cos(\varphi_2) \end{aligned}$$

Substituting into (4.16) gives

$$2 \times E_k = c_1 \dot{\varphi_1}^2 + c_2(\dot{\varphi}_1 + \dot{\varphi}_2)^2 + 2 c_3 \dot{\varphi}_1(\dot{\varphi}_1 + \dot{\varphi}_2)\cos(\varphi_2) \tag{4.17}$$

where the constants c_1, c_2, c_3 are

$$\begin{aligned} c_1 &= (I_0 + I_1 + I_a) + m_1 l_{c1}^2 + (m_2 + m_a + m_b) l_1^2 \\ c_2 &= (I_2 + I_b) + m_2 l_{c2}^2 + m_b l_2^2 \\ c_3 &= l_1 (m_2 l_{c2} + m_b l_2) \end{aligned}$$

Assuming motion in a gravitational field, the potential energy for the rigid links for a vertical (y, z) plane can be written as follows

$$E_p = \Big\{ [m_1 h_{c1} + m_a h_a] + [m_2 h_{c2} + m_b h_b] \Big\} g_r \tag{4.18}$$

where g_r is the gravitational acceleration and

$$\begin{aligned} h_{c1} &= l_{c1} \sin(\varphi_1) & h_a &= l_1 \sin(\varphi_1) \\ h_{c2} &= l_1 \sin(\varphi_1) + l_{c2} \sin(\varphi_1 + \varphi_2) & h_b &= l_1 \sin(\varphi_1) + l_2 \sin(\varphi_1 + \varphi_2) \end{aligned}$$

Substituting into (4.18) gives

$$E_p = \Big\{ c_4 \sin(\varphi_1) + c_5 \sin(\varphi_1 + \varphi_2) \Big\} g_r \tag{4.19}$$

where the constants c_4, c_5 are

$$c_4 = m_1 l_{c1} + (m_2 + m_a + m_b) l_1$$
$$c_5 = m_2 l_{c2} + m_b l_2$$

If the plane of motion is horizontal or the gravitational field does not exist, then $E_p = 0$. For that case, g_r is set to zero in the equations of motion.

The derivatives necessary for the Lagrange equations (4.15) are given in Appendix (B). The generalised forces corresponding to φ_1 and φ_2 are $Q_1 = u_1$ and $Q_2 = u_2$, respectively. Substituting into (4.15) the equations of motion are obtained as

$$a_{11}\ddot{\varphi}_1 + a_{12}\ddot{\varphi}_2 - a_{13}(2\dot{\varphi}_1 + \dot{\varphi}_2)\dot{\varphi}_2 + a_{14} g_r = u_1 \tag{4.20}$$

$$a_{12}\ddot{\varphi}_1 + a_{22}\ddot{\varphi}_2 + a_{13}\dot{\varphi}_1{}^2 + a_{24} g_r = u_2 \tag{4.21}$$

where:

$$\begin{aligned}
a_{11} &= c_1 + c_2 + 2c_3\cos(\varphi_2) & c_1 &= (I_0 + I_1 + I_a) + m_1 l_{c1}^2 + (m_2 + m_a + m_b) l_1^2 \\
a_{12} &= c_2 + c_3\cos(\varphi_2) & c_2 &= (I_2 + I_b) + m_2 l_{c2}^2 + m_b l_2^2 \\
a_{13} &= c_3\sin(\varphi_2) & c_3 &= l_1(m_2 l_{c2} + m_b l_2) \\
a_{22} &= c_2 & c_4 &= m_1 l_{c1} + (m_2 + m_a + m_b) l_1 \\
a_{14} &= c_4\cos(\varphi_1) + c_5\cos(\varphi_1 + \varphi_2) & c_5 &= m_2 l_{c2} + m_b l_2 \\
a_{24} &= c_5\cos(\varphi_1 + \varphi_2)
\end{aligned} \tag{4.22}$$

Using the states defined as:

$$\begin{aligned}
x_1 &= \varphi_1 \quad x_2 = \dot{\varphi}_1 \\
x_3 &= \varphi_2 \quad x_4 = \dot{\varphi}_2
\end{aligned} \tag{4.23}$$

the equations of motion (4.20) and (4.21) can be transformed to the form (4.2); that is:

$$\dot{x}_i(t) = a_i(x) + c_{ij}(x) u_j(t) \tag{4.24}$$

where:

$$
\begin{aligned}
a_1(x) &= x_2 \\
a_2(x) &= \tfrac{+a_{12}}{\Delta} \times [a_{22}(2x_2 + x_4)x_4 + a_{12}x_2^2] - \tfrac{g_r}{\Delta} \times (a_{14}a_{22} - a_{24}a_{12}) \\
a_3(x) &= x_4 \\
a_4(x) &= \tfrac{-a_{12}}{\Delta} \times [a_{12}(2x_2 + x_4)x_4 + a_{11}x_2^2] + \tfrac{g_r}{\Delta} \times (a_{14}a_{12} - a_{24}a_{11})
\end{aligned}
\tag{4.25}
$$

and

$$
c(x) = \tfrac{1}{\Delta} \begin{bmatrix} 0 & 0 \\ +a_{22} & -a_{12} \\ 0 & 0 \\ -a_{12} & +a_{11} \end{bmatrix} \quad u(t) = \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \tag{4.26}
$$

The parameters a_{ij} are given by the relations in (4.22) in which φ_1 and φ_2 are replaced by x_1 and x_3, respectively. The dominator parameter Δ in Equation (4.25) and Equation (4.26) is defined as:

$$
\Delta = a_{11}a_{22} - a_{12}^2 = c_1c_2 - c_3^2\cos^2(x_3) \tag{4.27}
$$

It can be shown that $\Delta > 0$ for any configuration of the manipulator.

4.3 The Costate Equations

As it was stated in the previous chapter, the optimal solution must satisfy the necessary conditions (4.5), (4.6), (4.7) resulting from Pontryagin's Minimum Principle. The second necessary condition (4.6) represents the costate equation. For calculating the costate equation the Hamiltonian can be defined as (3.33) or as (3.61)

$$\mathcal{H}(x,u,p,t) = 1 + p_k(t)[a_k(x) + c_{kj}(x)u_j(t)] \tag{4.28}$$

where $k = 1,\ldots,4$ and $j = 1,2$.

The costate equations then are

$$\dot{p}_i(t) = -\frac{\partial \mathcal{H}}{\partial x_i}(x,u,p,t) = p_k\ f_{ik}(x,u) \tag{4.29}$$

where

$$f_{ik}(x,u) = -\left[\frac{\partial a_k}{\partial x_i} + \frac{\partial c_{kj}}{\partial x_i}u_j\right] \tag{4.30}$$

where $i,k = 1,\ldots,4$ and $j = 1,2$.

The partial derivatives of $\frac{\partial a_k}{\partial x_i}$ and $\frac{\partial c_{kj}}{\partial x_i}$ are given in Appendix (B). Substituting those derivatives into (4.30) gives the costate equations in the following form (4.29) as:

$$\begin{aligned}
\dot{p}_1 &= p_2 \times f_{12}(x) + p_4 \times f_{14}(x) \\
\dot{p}_2 &= -p_1 + p_2 \times f_{22}(x) + p_4 \times f_{24}(x) \\
\dot{p}_3 &= p_2 \times f_{32}(x,u) + p_4 \times f_{34}(x,u) \\
\dot{p}_4 &= -p_3 + p_2 \times f_{42}(x) + p_4 \times f_{44}(x)
\end{aligned} \tag{4.31}$$

where the non-zero functions $f_{ik}(x,u)$ are:

$$f_{12}(x) = \tfrac{g_r}{\Delta} \times [(a_{12} - a_{22})c_5 sin(x_1 + x_3) - a_{22}c_4 sin(x_1)]$$

$$f_{14}(x) = -\tfrac{g_r}{\Delta} \times [(a_{11} - a_{12})c_5 sin(x_1 + x_3) - a_{12}c_4 sin(x_1)]$$

$$f_{21} = -1$$

$$f_{22}(x) = \tfrac{-2}{\Delta} \times a_{13}(a_{12}x_2 + a_{22}x_4)$$

$$f_{24}(x) = \tfrac{+2}{\Delta} \times a_{13}(a_{11}x_2 + a_{12}x_4)$$

$$\begin{aligned} f_{32}(x,u) = -\tfrac{1}{\Delta} \times \Big\{ & (a_{12} - a_{22})\left[a_{22}(2x_2 + x_4)x_4 + a_{12}x_2^2\right] - a_{13}^2 x_2^2 \\ & + a_{13}u_2 - g_r\left[(a_{12} - a_{22})c_5 sin(x_1 + x_3) + a_{13}a_{24}\right]\Big\} \\ + \tfrac{1}{\Delta^2} \times 2a_{13}(a_{12} - a_{22}) \Big\{ & a_{13}\left[a_{22}(2x_2 + x_4)x_4 + a_{12}x_2^2\right] \\ & + a_{22}u_1 - a_{12}u_2 - g_r\left[a_{14}a_{22} - a_{24}a_{12}\right]\Big\} \end{aligned}$$

$$\begin{aligned} f_{34}(x,u) = \tfrac{1}{\Delta} \times \Big\{ & (a_{12} - a_{22})\left[a_{12}(2x_2 + x_4)x_4 + a_{11}x_2^2\right] - a_{13}^2[(x_2 + x_4)^2 + x_2^2] \\ & + a_{13}(2u_2 - u_1) - g_r\left[(a_{11} - a_{12})c_5 sin(x_1 + x_3) + a_{13}(2a_{24} - a_{14})\right]\Big\} \\ - \tfrac{1}{\Delta^2} \times 2a_{13}(a_{12} - a_{22}) \Big\{ & a_{13}\left[a_{12}(2x_2 + x_4)x_4 + a_{11}x_2^2\right] \\ & + a_{12}u_1 - a_{11}u_2 - g_r\left[a_{14}a_{12} - a_{24}a_{11}\right]\Big\} \end{aligned}$$

$$f_{41} = -1$$

$$f_{42}(x) = \tfrac{-2}{\Delta} \times a_{13}a_{22}(x_2 + x_4)$$

$$f_{44}(x) = \tfrac{+2}{\Delta} \times a_{13}a_{12}(x_2 + x_4)$$

4.4 The Switch Functions and Controls

The optimal control forces can be obtained from the PMP as

$$u_j(t) = \begin{bmatrix} U_j^+ & for & G_j < 0 \\ U_j^- & for & G_j > 0 \end{bmatrix} \tag{4.32}$$

The switch function (G_j, $j = 1, 2$) corresponding to control u_j are:

$$G_j(x, p) = \sum_{i=1}^{4} p_i c_{ij}(x) \tag{4.33}$$

where the matrix c_{ij} are given in Equation (4.26). Using this matrix in the explicit form the switch functions which define the controls u_1 and u_2 are:

$$\begin{aligned} G_1(x, p) &= \tfrac{1}{\Delta} \times (+p_2 a_{22} - p_4 a_{12}) \\ G_2(x, p) &= \tfrac{1}{\Delta} \times (-p_2 a_{12} + p_4 a_{11}) \end{aligned} \tag{4.34}$$

Whenever G_j changes sign the corresponding control u_j switches from one extreme to another extreme (bang-bang control). The complete sequence of controls is given by Equation (4.32).

Chapter 5

Method of Solving Optimal Control Problems

5.1 Shooting Method

The time-optimal control problem defined in Chapter 4 belongs to the Two-Point Boundary Value Problems (TPBVP) with the initial and the final boundary conditions to be met. Usually no analytical solutions to these problems can be found; therefore, substantial efforts have been made to solve them numerically. The Shooting Method (SM) is one of the basic numerical approaches of solving TPBVP.

In the SM the initial conditions are used explicitly but the final conditions are replaced by unspecified conditions at the initial point (the unknown initial conditions). With a guessed set of unknown initial conditions, the differential equations can be integrated numerically up to the final point (shooting to the target final conditions.) If the computed final states satisfy the given final states, the problem is solved. If these final values do not satisfy the given target point (which is usually the case), the errors at the target are used to correct the unknown initial conditions. For the nonlinear differential equations or nonlinear boundary conditions, the correction of unknown initial conditions should be done iteratively.

The shooting methods are quite general and are applicable to a variety of differential equations of any order (even or odd). They are relatively easy to implement in a computer code. However, for nonlinear problems the shooting methods may be very sensitive to the unknown initial conditions. It makes the method difficult to

converge. Such a situation arises when the SM is applied to TPBVP for time-optimal control of Two-Link Manipulators (TLM). Several authors have tried unsuccessfully to apply the SM to the TLM [29].

A brief description of the shooting methods for TPBVP is as follows. The set of n nonlinear Ordinary Differential Equations (ODE) can be written as

$$\dot{x}(t) = a(x, t) \tag{5.1}$$

where x is an n vector and a is an n vector which is a nonlinear function of the dependent variable x and assumed to be twice differentiable with respect to x, and t is the independent variable.

In general the initial conditions may be given as:

$$x_i(t_0) = c_i, \qquad i = 1, \ldots, r \tag{5.2}$$

also the final conditions are given as:

$$x_m(t_f) = c_m, \qquad m = r+1, \ldots, n \tag{5.3}$$

Clearly $r < n$ for having the boundary conditions on both ends. In the case of the TLM, $n = 8$ and $r = 4$.

In the SM the unknown initial conditions are corrected by an amount which depends on the difference between the calculated final values and the given final conditions.

If there are $n - r$ final conditions (5.3), the error $L_m = x_m(t_f) - c_m$ calculated at t_f is function of r initial conditions $c_1, \ldots, c_r$ and $n - r$ unknown initial conditions denoted as $x_{r+1}(t_0), \ldots, x_n(t_0)$. The purpose of the SM is to find such

$x_{r+1}(t_0), \ldots, x_n(t_0)$ that makes $L_m = 0$; that is,

$$L_m(x_{r+1}, x_{r+2}, \ldots, x_n)_{t_0} = 0, \qquad m = 1, \ldots, n-r \tag{5.4}$$

Different approaches are used for solving the system of equations (5.4). For a linear system (5.1), the set (5.4) represents m linear algebraic equations which can be easily solved. For nonlinear problems the SM reduces to a set of nonlinear equations which should be solved by iterations applying the Newton-Raphson method, for example. The method of adjoints and the method of complementary functions are two of the most frequently used shooting methods [54]. The former employs backward integration of the adjoint equations, while the later employs forward integration of the variational equations (see Appendix B.4).

For stability of numerical procedures such as the SM, it is required that small changes to the initial values $x_{r+1}(t_0), \ldots, x_n(t_0)$ should result in small changes in the solution $x_{r+1}(t_f), \ldots, x_n(t_f)$. Such problems are called well conditioned. Any problem which does not has this property is called ill conditioned. Such problems are numerically unstable, and very sensitive to the initial guess of the unknown initial conditions. Unfortunately the TPBVP for the TLM belongs to this class of problems.

5.2 Shooting Method for Two-Link Manipulators

Here the SM for a particular class of the TPBVP for time-optimal control problems is discussed. In these problems, the control is bang-bang, and the final time is unknown. Experience shows that this TPBVP is sensitive to the initial conditions because of strong interaction between the states and controls, the controls and switch functions and the switch functions and costates.

The SM for solving the TPBVP for time-optimal problems with a single control was originally proposed in [3]. Here the method is expanded to include multiple controls and to handle Two-Link Manipulators as defined in Chapter 4.

The equations of states, costates, initial and final conditions, control forces, and Hamiltonian [(4.5), (4.6), (4.3), (4.10), (4.11)] can be rewritten as

$$\dot{X} = F[X(t), u(t)] \tag{5.5}$$

$$X(t_0) = K[x_0, B] \tag{5.6}$$

$$L[x(B, t_f), t_f] = 0 \tag{5.7}$$

$$u_i(t) = \begin{bmatrix} U_i^+ & if & G_i(X) < 0 \\ U_i^- & if & G_i(X) > 0 \end{bmatrix} \tag{5.8}$$

Where, $X = [x, p]^T$ is vector of $2n$ variables which includes both states x and costates p. $F[X, u]$ is vector of nonlinear functions of states and costates as given by the right hand side of (4.5), (4.6). Equation (5.5) formally combines (4.5) and (4.6) together.

Equation (5.6) represents the initial conditions where x_0 is a known n dimensional vector of initial states, and $B = p(t_0)$ is an n dimensional vector of unknown initial conditions (costates). Thus $K[x_0, B]$ is $2n$ dimensional vector of initial states as well as unknown initial conditions B (costates).

The final conditions are handled by Equation (5.7), where $L[x(B, t_f), t_f]$ is an l dimensional vector, with $l \geq n$, representing the error at the target point. This error depends on the choice of B and t_f when integrating Equation (5.5). The vector L

includes the final conditions of states (4.3), and the extra condition for Hamiltonian (4.11) to be met at the target point. For the TLM the components of the L vector are:

$$L_i[x(B,t_f),t_f] = v_i \times [x_i(t_f) - x_{if}] \quad i = 1,\ldots,n$$
$$L_l[x(B,t_f),t_f] = v_l \times \mathcal{H}(t_f) \tag{5.9}$$

Here $x_i(t_f)$ are calculated final states and x_{if} are final conditions for states, and v_i are weight functions to accommodate the difference between the dimensions of the states as well as of the Hamiltonian. The choice of weight functions v_i is discussed in Appendix D.

U_i^+, U_i^- are upper and lower controls bounds, and $G_i(X)$ are switching functions defined by (4.34).

In order to reduce $L[x(B,t_f),t_f]$ to zero, the values of $B^{(k)}$, $t_f^{(k)}$ in iteration k, have to be corrected in the next iteration using the following formula:

$$\begin{bmatrix} B^{(k+1)} \\ t_f^{(k+1)} \end{bmatrix} = \begin{bmatrix} B^{(k)} \\ t_f^{(k)} \end{bmatrix} + \begin{bmatrix} \Delta B^{(k)} \\ \Delta t_f^{(k)} \end{bmatrix} \tag{5.10}$$

In each iteration the correction $\Delta B^{(k)}$, $\Delta t_f^{(k)}$ can be computed by minimising a norm of the target error, L as:

$$\|L\| = \left(\sum_{i=1}^{l} L_i^2 \right)^{1/2} \tag{5.11}$$

The Newton-Raphson method is used for this purpose. The vector of corrections is defined as:

$$\begin{bmatrix} \Delta B^{(k)} \\ \Delta t_f^{(k)} \end{bmatrix} = -\alpha_k \begin{bmatrix} \delta B^{(k)} \\ \delta t_f^{(k)} \end{bmatrix} \tag{5.12}$$

where $\delta B^{(k)}$ and $\delta t_f^{(k)}$ are calculated as:

$$\begin{bmatrix} \frac{\partial L[B^{(k)},t_f^{(k)}]}{\partial B} & \frac{\partial L[B^{(k)},t_f^{(k)}]}{\partial t_f} \end{bmatrix} \begin{bmatrix} \delta B^{(k)} \\ \delta t_f^{(k)} \end{bmatrix} = L[B^{(k)}, t_f^{(k)}] \tag{5.13}$$

The positive scalar α_k, calculated in each iteration, is added for improvement of the convergence and is chosen from:

$$\alpha_k = \min \left\{ 1.0, \frac{\rho \| [B^{(k)}, t_f^{(k)}] \|}{\| [\delta B^{(k)}, \delta t_f^{(k)}] \|} \right\} \tag{5.14}$$

where $0 < \rho < 1$.

The scalar ρ effects the value of α_k. Intuition is required when choosing ρ for convergence of the SM. If the unknown initial conditions B are far from their optimal values then ρ can be chosen to be small, typically $0.01 \leq \rho \leq 0.05$. If they are close to their optimal values then ρ can be chosen close to one ($0.9 \leq \rho \leq 0.99$).

Components of the partial derivative matrix $\begin{bmatrix} \frac{\partial L}{\partial B} & \frac{\partial L}{\partial t_f} \end{bmatrix}_{l \times l}$ can be calculated as follows:

$$\frac{\partial L}{\partial B} = \left(\frac{\partial L}{\partial X} \frac{\partial X}{\partial B} + \frac{\partial L}{\partial B} \right)_{t=t_f}, \quad \frac{\partial L}{\partial t_f} = \left(\frac{\partial L}{\partial X} \frac{dX}{dt} + \frac{\partial L}{\partial t_f} \right)_{t=t_f} \tag{5.15}$$

All of the above partial derivatives can be obtained analytically, with the exception of $\frac{\partial X}{\partial B}$. For computing $\frac{\partial X(t_f)}{\partial B}$ the following equation derived from (5.5) can be used:

$$\begin{aligned} \frac{\partial X(t_f)}{\partial B} = \frac{\partial K[x_0, B]}{\partial B} &+ \sum_{j=1}^{N+1} \int_{t_{j-1}}^{t_j} \left(\frac{\partial F[X(\tau), u(\tau)]}{\partial X} \frac{\partial X(\tau)}{\partial B} \right) d\tau \\ &+ \sum_{j=1}^{N} \left\{ F[X,u]_{t=t_j^-} - F[X,u]_{t=t_j^+} \right\} \frac{\partial t_j}{\partial B} \end{aligned} \tag{5.16}$$

The following sequence of ODE can also be used:

$$\frac{d}{dt} \left(\frac{\partial X(t)}{\partial B} \right) = \left(\frac{\partial F}{\partial X} \right) \left(\frac{\partial X(t)}{\partial B} \right) \tag{5.17}$$

valid for $t_{j-1} \leq t \leq t_j$, where t_j is the jth switch time $(1 \leq j \leq N+1), t_{N+1} = t_f$, and the initial value for the above equation is:

$$\frac{\partial X(t_0)}{\partial B} = \frac{\partial K[x_0, B]}{\partial B} \tag{5.18}$$

There is a jump in the value of $\frac{\partial X(t)}{\partial B}$ at every switch (t_j) which is calculated as:

$$\frac{\partial X(t_j^+)}{\partial B} = \frac{\partial X(t_j^-)}{\partial B} + \left\{ F[X,u]_{t=t_j^-} - F[X,u]_{t=t_j^+} \right\} \frac{\partial t_j}{\partial B} \tag{5.19}$$

This equation indicates that $\frac{\partial X}{\partial B}$ is discontinuous at a switch time if the corresponding $F[X,u]$ is discontinuous. For the TLM it occurs only for X_2, X_4, and X_7 because the derivatives of these components of X explicitly contain the control u.

Since at switch time t_j the corresponding switch function $G_i[X(t_j)] = 0$, then

$$\left\{ \frac{\partial G_i}{\partial X} \left[\frac{\partial X}{\partial B} + \frac{dX}{dt} \frac{\partial t}{\partial B} \right] + \frac{\partial G_i}{\partial t} \frac{\partial t}{\partial B} \right\}_{t=t_j^-} = 0 \tag{5.20}$$

This can be rearrange as

$$\left\{ \left[\frac{\partial G_i}{\partial X} \frac{dX}{dt} + \frac{\partial G_i}{\partial t} \right] \frac{\partial t}{\partial B} + \frac{\partial G_i}{\partial X} \frac{\partial X}{\partial B} \right\}_{t=t_j^-} = 0 \tag{5.21}$$

The derivative $\frac{\partial t_j}{\partial B}$ for each switch function G_i . using the total derivative, can be calculated as $\left(\text{where } \frac{dG_i}{dt} = \frac{\partial G_i}{\partial X}\frac{dX}{dt} + \frac{\partial G_i}{\partial t}\right)$:

$$\left(\frac{\partial t_j}{\partial B} \right)_{G_i} = - \left\{ \left(\frac{dG_i}{dt} \right)^{-1} \frac{\partial G_i}{\partial X} \frac{\partial X}{\partial B} \right\}_{t=t_j^-} \tag{5.22}$$

For a nonsingular system, it is assumed that $\frac{dG_i}{dt} \neq 0$ at every switch time.

The equations (5.5), (5.6), (5.7), (5.8), (5.10), (5.11), (5.12), (5.13), (5.14), (5.15), (5.17), (5.18), (5.19), (5.22) are used in the SM for the TLM.

For starting the iterations a guess of the values for $B^{(0)}$ and $t_f^{(0)}$ is needed. It is

known (see [48]) that convergence of the SM is very sensitive to these initial guesses. If the guess is sufficiently close to the optimal values of the initial costates, the SM works as well as the regular Newton-Raphson method for solving nonlinear algebraic equations. For the TLM, however, it is extremely difficult to have a good guess of $B^{(0)}$ and $t_f^{(0)}$ intuitively, because the costates do not have any simple interpretation or physical meaning. Therefore a new method is proposed which automatically generates a sufficiently good guess of the unknown initial conditions B (costates). This method is referred to as Forward-Backward Method and is discussed in Section 5.3 and Section 5.4.

5.3 Forward-Backward Method (Single Control)

The Forward-Backward Method (FBM) is a numerical method used to find an approximate solution of TPBVP. In particular, it satisfies the initial and final conditions on the state automatically [47], [48]. In the first step the FBM uses the states only to find the switch times that would allow a bang-bang control to move the state of the system from the initial to final location. However, neither the number of switches nor their location is optimal in the sense that switch functions are not necessarily zero at the switch points. The second step of the FBM is to find the costates. In the subsequent iterations, when both the states and costates are available, the optimality conditions are checked and the number and locations of the switch times are corrected correspondingly.

The procedure is first explained for the case of single control. Suppose the bang-bang solution for time-optimal control problem has only one switch. Using the initial and final conditions for the states, and assuming one switch for the single control u together with the assumption of the sign of the control, the switch time is located. For a control u and a single switch function G, the following steps should be taken.

5.3.1 Iteration One

Step 1: Finding one switch time

The single control u can take the value U^+ or U^- at x_0 and x_f. At x_0 the control u most likely should be U^+ (to accelerate the mass from rest) and when approaching x_f it should be U^- (to decelerate the mass until it stops). Therefore starting from the initial conditions x_0, it is assumed that the control $u^{fd} = U^+$ and the state equation integrated forward in time up to an assumed final time t_f^a to obtain $x(x_0, u^{fd}, t)$ as shown in Figure 5.1. Simultaneously, starting from the final conditions x_f, the control is taken as $u^{bd} = U^-$ and the state equation is integrated backward in time from the same assumed final time t_f^a to obtain $x(x_f, u^{bd}, t_f - t)$. Here the superscripts

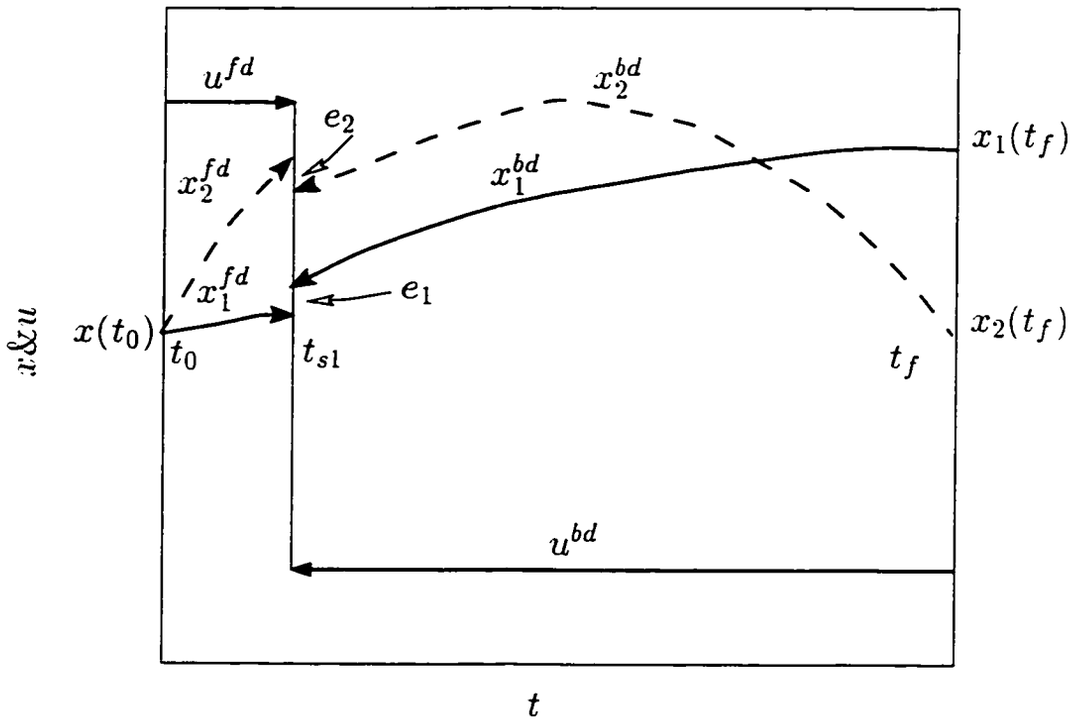

Figure 5.1: **Step 1** of the FBM, time domain.

(^{fd}) and (^{bd}) denote the control values in forward, and backward integration paths, respectively. On the phase plane, the separation between the forward and backward paths in each time instant is defined by the vector $e(t, t_f)$ (Figure 5.2), where:

$$e(t, t_f) = \left\{ \begin{array}{c} e_1(t, t_f) \\ \\ e_2(t, t_f) \end{array} \right\} = \left\{ \begin{array}{c} \left| x_1(x_0, u^{fd}, t) - x_1(x_f, u^{bd}, t_f - t) \right| \\ \\ \left| x_2(x_0, u^{fd}, t) + x_2(x_f, u^{bd}, t_f - t) \right| \end{array} \right\} \tag{5.23}$$

Here x_1 and x_2 are the states of the system. For example, for a mass and spring system they are the displacement and the velocity respectively. When $e(t, t_f) \rightarrow 0$, for $t \rightarrow t_s$, the continuity of states is satisfied, and consequently

$$x(x_0, u^{fd}, t_s) = x(x_f, u^{bd}, t_f - t_s) \tag{5.24}$$

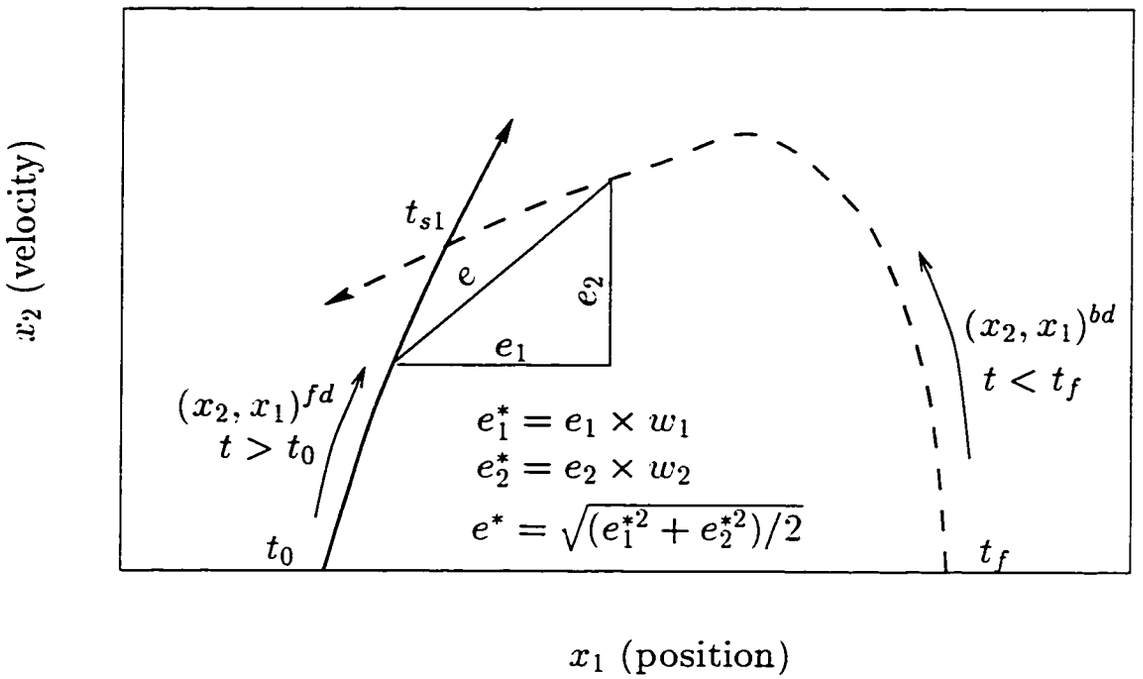

Figure 5.2: **Step 1** of the FBM, phase domain.

This point is the first switch t_s, and the corresponding t_f is the final time of the trajectory that meets the initial condition x_0, and the final condition x_f. Numerically, the values of t_s and t_f are found (when the number of states is $n = 2$) by minimising the following norm of $e(t, t_f)$:

$$e^{\blacksquare} = \sqrt{\frac{e_1^{\blacksquare 2} + e_2^{\blacksquare 2}}{2}} \tag{5.25}$$

$$e_1^{\blacksquare} = e_1 \times w_1 \qquad e_2^{\blacksquare} = e_2 \times w_2$$

where w_1, w_2 are weight functions to accommodate the difference in dimensions of the states. If (5.24) can not be met, forward integration with $u^{fd} = U^-$ and backward integration with $u^{bd} = U^+$ should be carried out.

Step 2: Finding the costates

At the end of **Step 1** t_s and t_f are given, where t_s is the first switch time and t_f is the final time. It is still not known whether t_f is the minimum maneuver time. To verify this the costates and the switch function have to be calculated. Assuming that t_s is the first switch time for optimal maneuver and since (4.11) is valid for all time, it can be used at t_s together with (4.10) to determine the costates (p_1 and p_2) at t_s

$$\left[\begin{array}{l} H(t_s) = 0 \\ G(t_s) = 0 \end{array}\right] \Longrightarrow p(t_s) \tag{5.26}$$

where $G(t_s)$ is switch function used in (4.10).

Now one can obtain the costates for the entire time domain. First the state equations should be integrated from t_0 to t_s to obtain $x(t_s)$. Then the states and costates should be integrated simultaneously backward from t_s to t_0 using $x(t_s)$, and $p(t_s)$ as the initial conditions. Also the states and costates must be integrated simultaneously forward from t_s to t_f using $x(t_s)$, and $p(t_s)$ as the initial conditions. Using (4.10) more switch times can be computed, if they exist (control u can now be defined by the switch function $G(x,p)$). This step is illustrated in Figure 5.3. The subsequent iterations are described below.

5.3.2 Subsequent Iterations

Step 1 of **Iteration one** is repeated using $p(t_0)$ and $p(t_f)$ from the previous iteration as initial and final values for the costates, and x_0, x_f as the initial and the final values for the states, and (4.10) is used for defining the controls. There is no need to guess the sign of control forces at second and subsequent iterations, because they are defined by the sign of switch function $G(x,p)$. Also, this switch function determines the new set of the switch times t_{si} found from the condition $G(x,p) = 0$. The continuity

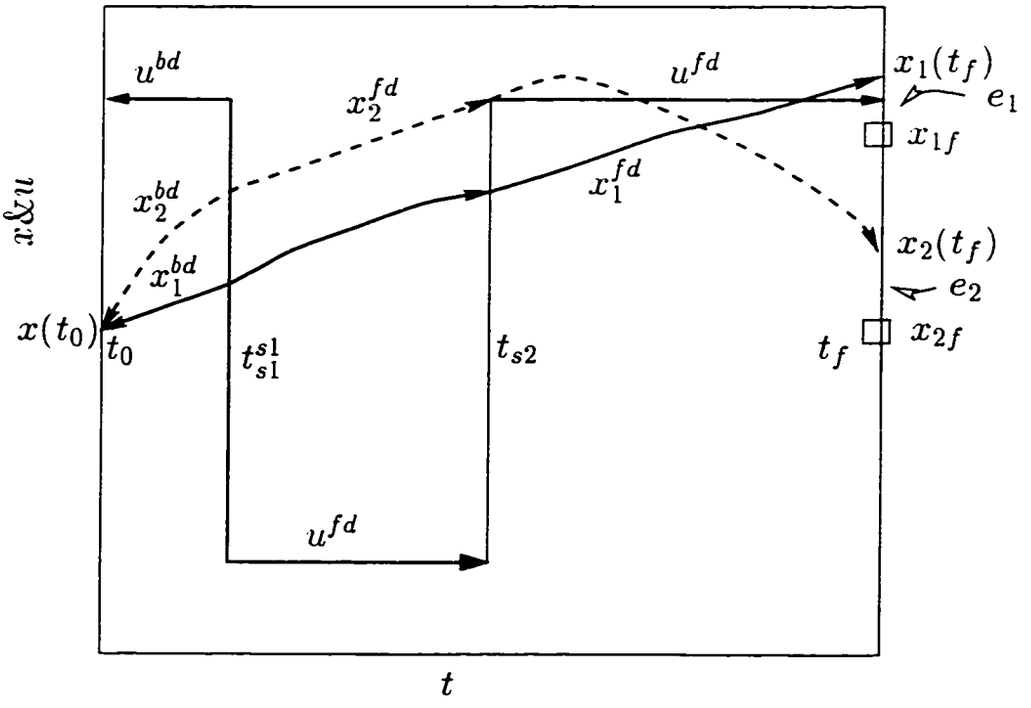

Figure 5.3: **Step 2** of the FBM, calculation of costate and another switch.

criteria of the states for multiple switches, t_{si}, now is:

$$x(x_0, u^{fd}, t_{si}) = x(x_f, u^{bd}, t_f - t_{si}) \tag{5.27}$$

where u^{fd}, u^{bd} are determined by the switch function $G(x,p)$ from (4.10) with the initial and final costates known from the previous iteration. Satisfying (5.27) means that the state equations are met for controls obtained from (4.10) starting with $p(t_0)$ of the previous iteration. However, when performing **Step 2**, in which the costates are calculated for the current iteration, the new switch times may be different than those in **Step 1**.

The following convergence norm, defining optimality of the solution, can be spec-

ified at the end of each iteration (indicated here as iteration $k+1$):

$$\epsilon_s = \sqrt{(\epsilon_{s1}^2 + \cdots + \epsilon_{sm}^2)/m} \tag{5.28}$$

where:

$$\epsilon_{sj} = \left| t_{sj}^{(k+1)} - t_{sj}^{(k)} \right| / t_f^{(k)}$$

and where $t_{sj}^{(k)}$ is the jth switch time, and $t_f^{(k)}$ is the final time calculated in the iteration k. If $\epsilon_s \rightarrow 0$, the state and the costate equations will be satisfied and the optimal solution will be generated.

5.3.3 Combining Forward-Backward & Shooting Methods

It has been found that the convergence of the FBM procedure is slow or difficult to obtain. In particular, it is difficult to apply the Newton-Raphson method to reduce the error (5.28). Better results were obtained by combining the Forward-Backward Method with the Shooting Method (SM). Implementation of the FBM on several examples has proved that the costates generated at each iteration are relatively close to the optimal costates.

It should be emphasised that, for the FBM iterations, the initial and the final conditions for the states are always met; however, some discontinuity in the states, defined by (5.25), may be present at one point of the trajectory. The SM (Section 5.2) can be started with:

$$B = p(t_0)_{FBM} \tag{5.29}$$

where $p(t_0)_{FBM}$ are taken from the FBM.

This way, a random guessing of the initial costates for starting the shooting method can be avoided. For most cases solved in Chapter 6 only one FBM iteration was needed to obtain satisfactory initial costates to converge the SM. It should be noted that the states and costates integrated by the SM are always continuous; however, there may

be some error at the target as defined by (5.9).

5.4 Forward-Backward Method for TLM

In the first step the forward-backward method uses the states only to find the switch times. Physically the motion is from x_0 to x_f with the controls being $U_i^{\mp}$. However, as explained in previous section, neither the number of switches nor their location is optimal in that sense that switch functions are not necessarily zero at the switch points. After obtaining x^{FBM} and u^{FBM}, the approximation of the initial costates $p(t_0)$ is obtained by assuming that the corresponding switch function G_i is zero at switch time t_{sj}. In Section 5.3 the FBM for cases with one control, when the order of the state equation was two, was discussed. The initial costates were obtained there using the Hamiltonian and a switch function at any single switch time. However, when the number of states is larger than two, more switch times are needed to start the integration of the costates. In case of the Two-Link Manipulators (TLM), there are four state variables, four costate variables, and two control forces (and consequently two switch functions) to be considered. Since at each switch time one switch function as well as the Hamiltonian vanish, at least two switch times are needed for numerical integration of the four costate equations. The number of switches will increase during the iteration process since for optimal solutions of various manoeuvres of TLM, it should be between 3 to 5 switches as reported in [29]. Discussion about the number of switches can be also found in [38], [40], and [51].

The forward-backward method for TLM is stated as follows. Suppose the bang-bang solution for time-optimal control problem has j switches. If the locations of $j-1$ switches are known one can use the initial and the final states to find the location of the jth switch. In the first step of the FBM it is assumed that there is only one switch time. If **Step 1** fails, then one can suppose that there are $j \geq 2$ switches, in which $j-1$ switches are assumed, and the last switch is determined. For the FBM

for TLM with two controls the following steps have to be taken.

5.4.1 Iteration One

Step 1: Finding one switch

Starting from the initial conditions x_0, it is assumed that the control $u_i^{fd} = U_i^+$ and the state equation integrated forward in time up to an assumed final time t_f^a to obtain $x(x_0, u_i^{fd}, t)$. Simultaneously using the final conditions x_f, the control is taken $u_i^{bd} = U_i^-$ and the state equation is integrated backward in time up to the same assumed final time t_f^a to obtain $x(x_f, u_i^{bd}, t_f - t)$. As before the superscripts $(^{fd})$ and $(^{bd})$ denote the values of parameters in forward, and backward integration paths respectively (Figure 5.4, Figure 5.5).

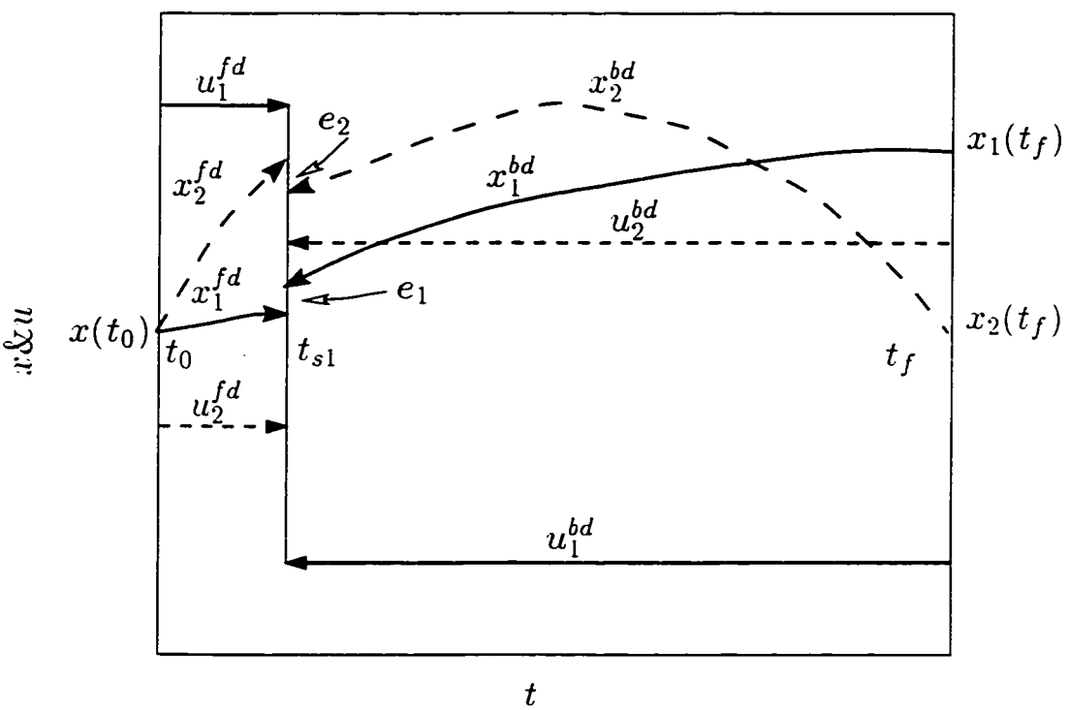

Figure 5.4: **Step 1** and sub **Step 2.1** of the FBM, controls and states of shoulder link (x_1, x_2).

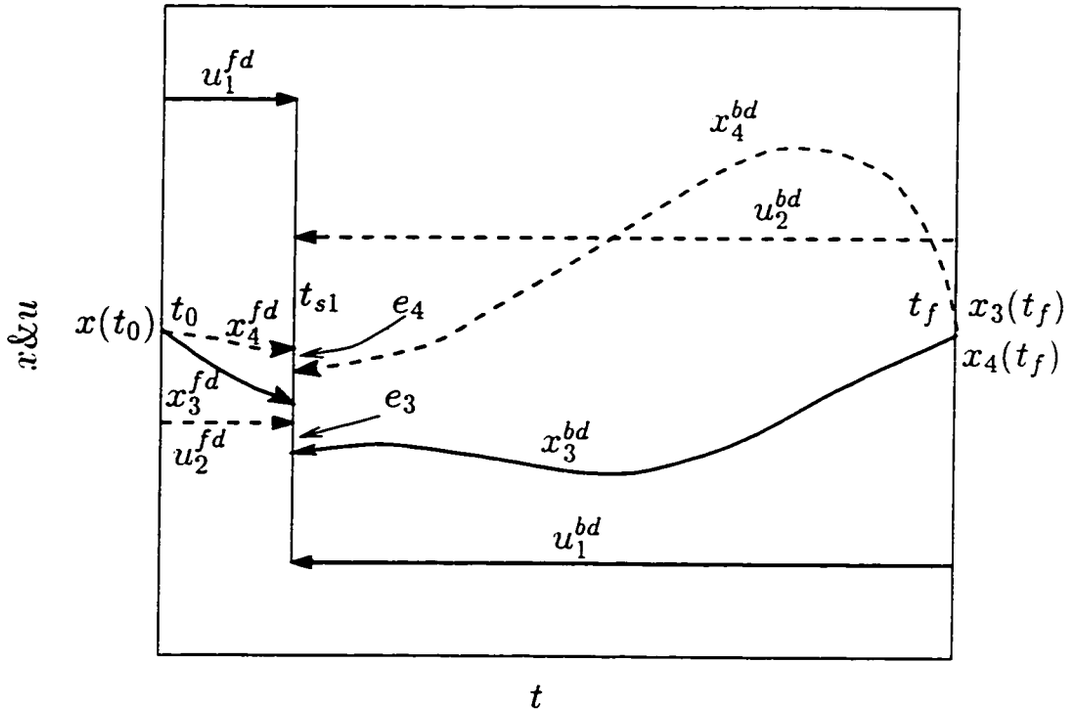

Figure 5.5: **Step 1** and sub **Step 2.1** of the FBM, controls and states for elbow link (x_3, x_4).

The separation between the forward and backward paths in each time instant is defined by the the vectors e_{12}, e_{34} (Figure 5.9), where:

$$e(t,t_f) = \left\{ \begin{array}{c} e_{12}(t,t_f) \\ \\ e_{34}(t,t_f) \end{array} \right\} \quad e_{12}(t,t_f) = \left[\begin{array}{c} e_1(t,t_f) \\ \\ e_2(t,t_f) \end{array} \right] \quad e_{34}(t,t_f) = \left[\begin{array}{c} e_3(t,t_f) \\ \\ e_4(t,t_f) \end{array} \right] \tag{5.30}$$

$$e_{12}(t,t_f) = \left\{ \begin{array}{c} \left| x_1(x_0,u_i^{fd},t) - x_1(x_f,u_i^{bd},t_f-t) \right| \\ \\ \left| x_2(x_0,u_i^{fd},t) + x_2(x_f,u_i^{bd},t_f-t) \right| \end{array} \right\}$$

$$e_{34}(t,t_f) = \left\{ \begin{array}{l} \left| x_3(x_0, u_i^{fd}, t) - x_3(x_f, u_i^{bd}, t_f - t) \right| \\ \\ \left| x_4(x_0, u_i^{fd}, t) + x_4(x_f, u_i^{bd}, t_f - t) \right| \end{array} \right\}$$

For TLM the states $x_1 = \varphi_1$ and $x_2 = \dot{\varphi}_1$ are the displacement and the velocity of the shoulder link respectively, and the states $x_3 = \varphi_2$ and $x_4 = \dot{\varphi}_2$ are the displacement and the velocity of the elbow link respectively. When $e(t,t_f) \rightarrow 0$, for $t \rightarrow t_{s1}$ the continuity of states is satisfied, and consequently

$$x(x_0, u_i^{fd}, t_{s1}) = x(x_f, u_i^{bd}, t_f - t_{s1}) \tag{5.31}$$

The trajectory meets the initial condition x_0, and the final condition x_f where t_{s1} is the first switch time, and the corresponding t_f is the final time. Numerically, the values of t_{s1} and t_f are found by minimising the following norm of $e^*(t,t_f)$:

$$e^* = \left(\frac{1}{4} \sum_{i=1}^{4} e_i^{*2} \right)^{1/2} \qquad e_i^* = e_i \times w_i \tag{5.32}$$

where w_i, $i = 1,\ldots,4$ are weight functions to accommodate the difference in magnitudes of the states. If $e^*_{min} > 0$ and can not be reduced, the continuity condition (5.31) can not be met. In that case, other possible combinations of the control values for forward and backward integrations should be tried. The following eight cases are all possibilities that may be considered in this step:

case	u_1^{fd}	u_2^{fd}	u_1^{bd}	u_2^{bd}
1	U_1^-	U_2^-	U_1^+	U_2^+
2	U_1^-	U_2^-	U_1^+	U_2^-
3	U_1^-	U_2^+	U_1^+	U_2^+
4	U_1^-	U_2^+	U_1^+	U_2^-

case	u_1^{fd}	u_2^{fd}	u_1^{bd}	u_2^{bd}
5	U_1^+	U_2^-	U_1^-	U_2^+
6	U_1^+	U_2^-	U_1^-	U_2^-
7	U_1^+	U_2^+	U_1^-	U_2^+
8	U_1^+	U_2^+	U_1^-	U_2^-

There are eight more possibilities for which $u_1^{fd} = u_1^{bd}$. However, if the motion of

the shoulder link is less than a specified value (e.g. $\varphi_1^c = 0.98[rad]$ for the IBM 7535 B 04 robot considered in [29]), these possibilities can be ruled out instantly because of the limit on the number of switches of the shoulder link (one switch).

For example, if one wishes to go from x_0 to x_f with only one switch as shown in Figure 5.6, cases 5 to 8 are good candidates for selecting the control torques and performing the forward and backward integrations. If none of the four candidates satisfies the continuity of all the states, it can be concluded that there must be either separate switches for the shoulder and elbow links or more switches. To handle this, the strategy discussed in **Step 2** should be used.

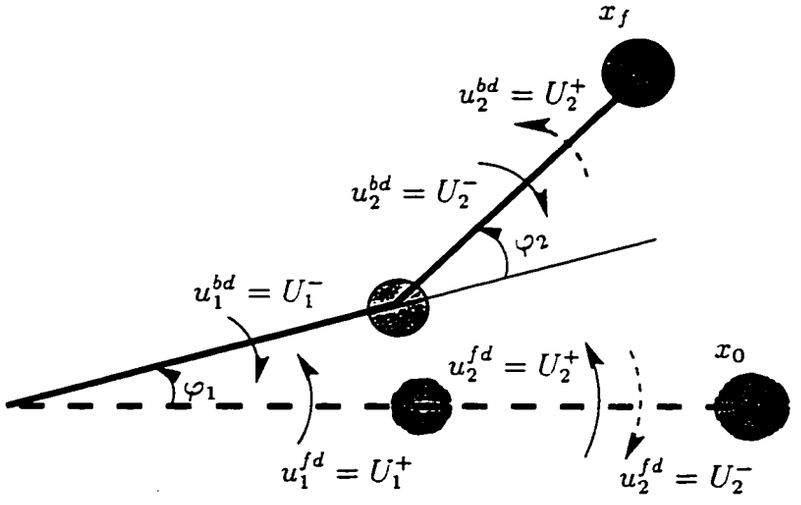

Figure 5.6: Different possibilities for u_1 and u_2 for the FBM for TLM.

Step 2: Finding two switches in one step

This step consists of two sub steps.

Sub Step 2.1: If **Step 1** fails to satisfy the continuity of all the states, which is usually the case, the continuity of the states of the shoulder link might be satisfied first, and the states of the elbow link be considered later. For this purpose **Step 1** considering only ϵ_{12} as the measure of the state continuity should be repeated. When

$e_{12}(t, t_f) \rightarrow 0$ at $t \rightarrow t_{s1}$, and consequently

$$x_i(x_0, u_i^{fd}, t_{s1}) = x_i(x_f, u_i^{bd}, t_f - t_{s1}) \tag{5.33}$$

where $i = 1, 2$ this instant is called the first switch t_{s1}, and the corresponding t_f is called the final time of the solution that meet the initial condition x_{0i} and the final condition x_{fi}, $i = 1, 2$ (see Figure 5.4 and Figure 5.5). One can use the same switch time for the elbow link; however, the continuity of this link may not be met in this sub step.

Sub Step 2.2: Now **Step 1** including both e_{12} and e_{34} in the measure $e^*(t, t_f)$ should be repeated. This can be done using the following controls (see Figure 5.7 and Figure 5.8):

$$u_i^{fd}(t) = \begin{bmatrix} u_1^{fd(s2.1)} & \text{if} \quad t \leq t_{s1} \\ u_1^{bd(s2.1)} & \text{if} \quad t > t_{s1} \\ u_2^{fd(s2.1)} & \end{bmatrix} \quad u_i^{bd}(t) = \begin{bmatrix} u_1^{bd(s2.1)} \\ u_2^{bd(s2.1)} \end{bmatrix} \tag{5.34}$$

Here $u_i^{fd(s2.1)}$, $u_i^{bd(s2.1)}$ are the control forces which are chosen in sub **Step 2.1**. When $e^*(t, t_f) \rightarrow 0$ at $t \rightarrow t_{s2}$, continuity of all the states and consequently (5.31) is satisfied. The new switch time is t_{s2}, and the corresponding t_f is the new final time of the solution that meet the initial condition x_{0i} and the final condition x_{fi}, $i = 1, \ldots, 4$. If the continuity of all four states still cannot be satisfied, it means that more switches are required, and **Step 3** should be carried out. Experience shows that for TLM at least three switches are needed.

Step 3: Using pre-specified switches

If **Step 1** and **Step 2** fail to satisfy the continuity of the states (5.31), one or two pre-specified switch times for u_2 should be used (t_{s1} and t_{s2} in Figure 5.10 and Figure 5.11). The location of these pre-specified switches may be corrected later.

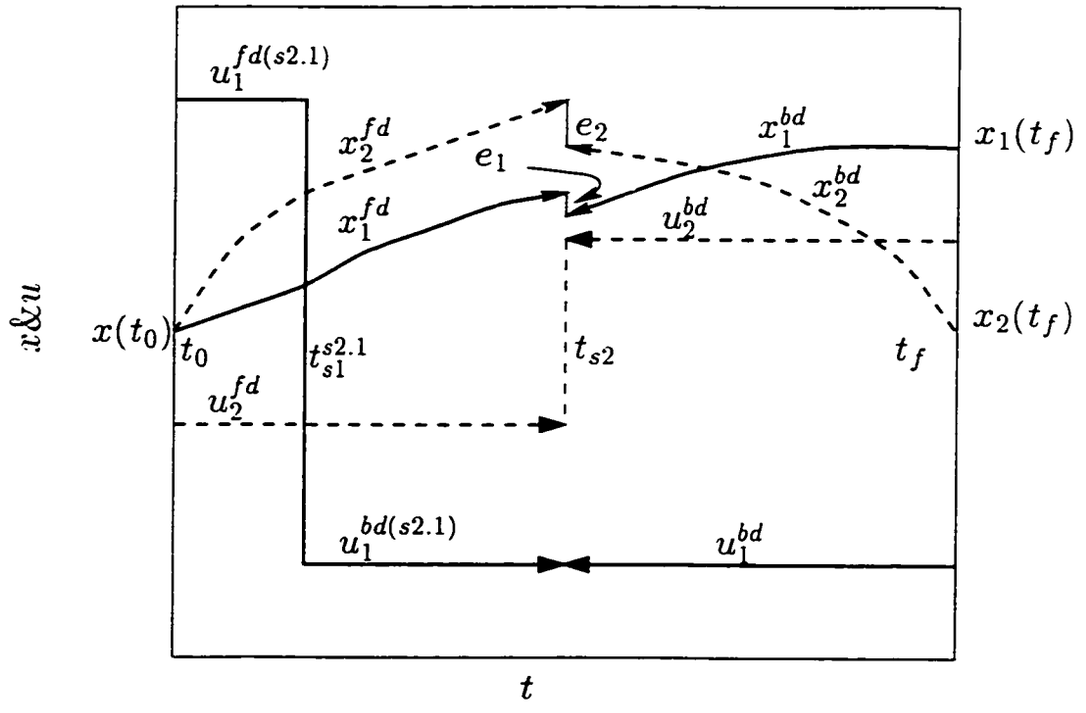

Figure 5.7: Sub **Step 2.2** of the FBM, controls and states for shoulder link (x_1, x_2).

Step 2 can be repeated with t_{s1} as the first switch and Δt_s, as the difference between the second switch and the assumed final time ($\Delta t_s = t_f^a - t_{s2}$). In forward integration and in backward integration the following controls (5.35) may be used:

$$u_i^{fd}(t) = \begin{bmatrix} u_1^{fd(s1)} & \\ u_2^{fd(s1)} & if \quad t \leq t_{s1} \\ -u_2^{fd(s1)} & if \quad t > t_{s1} \end{bmatrix} \quad u_i^{bd}(t) = \begin{bmatrix} u_1^{bd(s1)} & \\ u_2^{bd(s1)} & if \quad t \leq \Delta t_s \\ -u_2^{bd(s1)} & if \quad t > \Delta t_s \end{bmatrix} \tag{5.35}$$

The control forces $u_i^{fd(s1)}$, $u_i^{bd(s1)}$ are chosen from the candidate controls in **Step 1**. When $e^*(t, t_f) \rightarrow 0$ at $t \rightarrow t_{s3}$, continuity of all the states and consequently (5.31) is satisfied. This instant is the new switch time t_{s3}, and the corresponding t_f the final time of the solution that meet the initial condition x_0 and the final condition x_f. Like **Step 2**, this step is also performed in two sub steps, sub **Step 3.1** (Figure 5.10 and

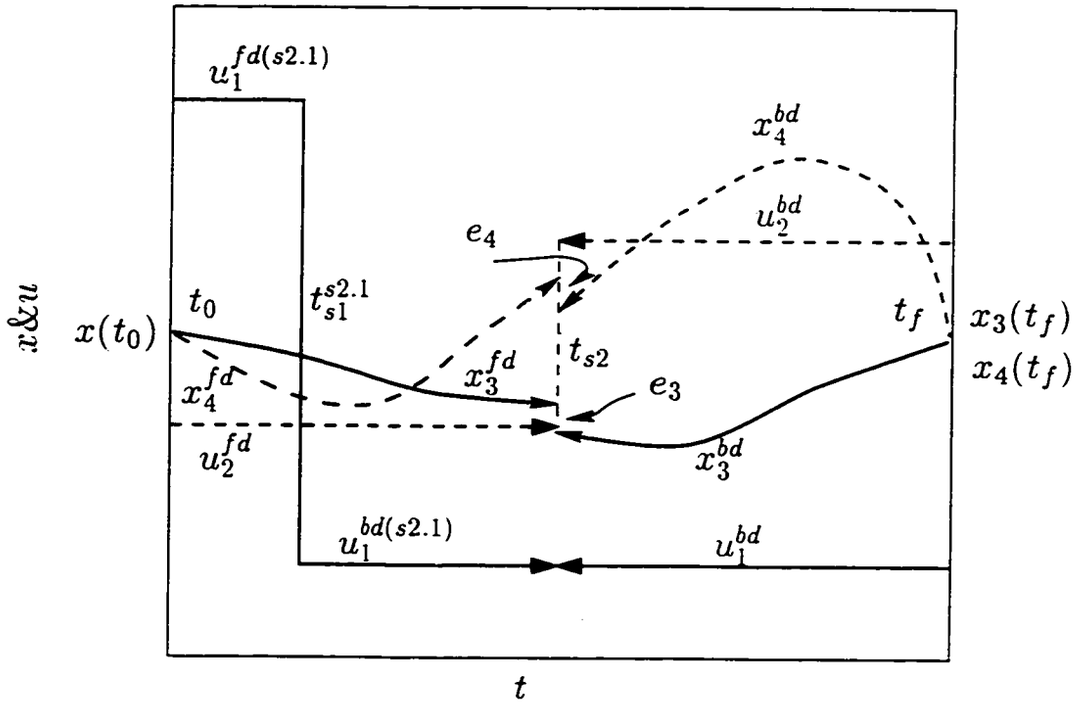

Figure 5.8: Sub **Step 2.2** of the FBM, controls and states for elbow link (x_3, x_4).

Figure 5.11), and sub **Step 3.2** (Figure 5.12 and Figure 5.13). The controls in sub **Step 3.2** are,

$$u_i^{fd}(t) = \begin{bmatrix} u_1^{fd(s3.1)} & \text{if} \quad t \leq t_{s3} \\ u_1^{bd(s3.1)} & \text{if} \quad t > t_{s3} \\ u_2^{fd(s3.1)} & \end{bmatrix} \quad u_i^{bd}(t) = \begin{bmatrix} u_1^{bd(s3.1)} \\ u_2^{bd(s3.1)} \end{bmatrix} \tag{5.36}$$

where $u_i^{fd(s3.1)}$, $u_i^{bd(s3.1)}$ are the control forces in sub **Step 3.1**.

The numerical results indicate that the continuity of all the states can be obtained upon correcting t_{s1} and Δt_s. The initial values of t_{s1} and Δt_s can be approximated using the results of **Step 2** as:

$$t_{s1}^{(0)} = t_{s1}^{(s2.2)}/2 \qquad \Delta t_s{}^{(0)} = (t_f - t_{s1})^{(s2.2)}/2 \tag{5.37}$$

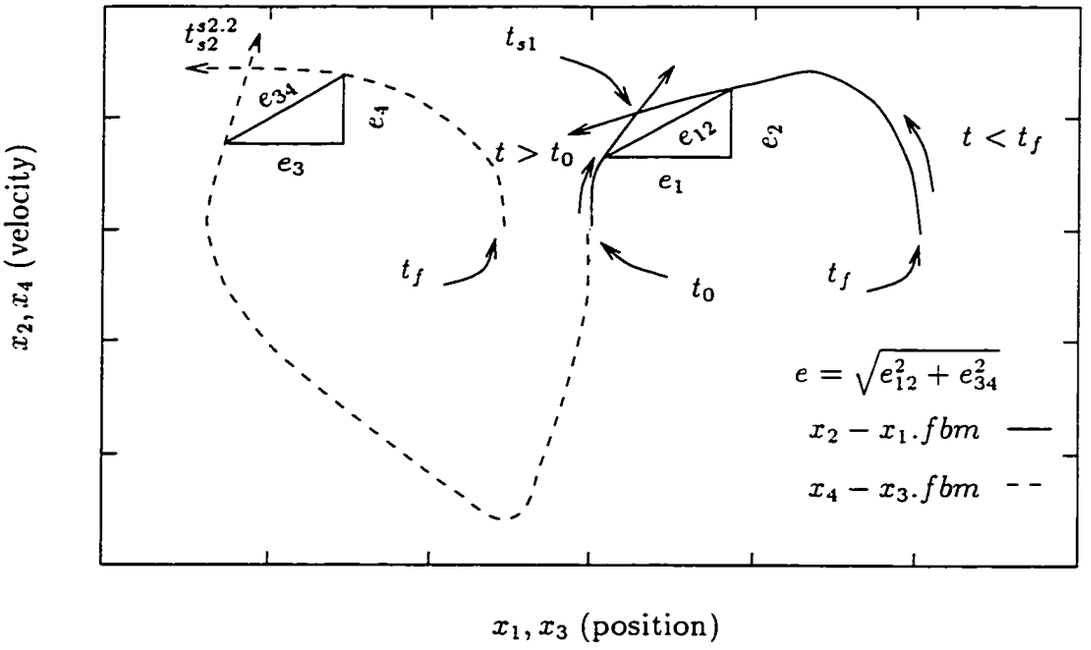

Figure 5.9: **Step 2** or **Step 3** of the FBM, state trajectory for both links.

The values of t_{s1} and Δt_s can be corrected by a gradient search method for the minimum differences of the forward and backward states, (e_{12}, e_{34}) in the following time intervals:

$$0 \leq t_{s1} \leq t_{s3}^{(s3.2)} \qquad 0 \leq \Delta t_s \leq (t_f - t_{s3})^{(s3.2)} \tag{5.38}$$

Note that when this step is completed the initial and the final boundary conditions are satisfied; however, there still may be some discontinuity when the states integrated from the initial and the final points are matched. This discontinuity is defined by (5.32).

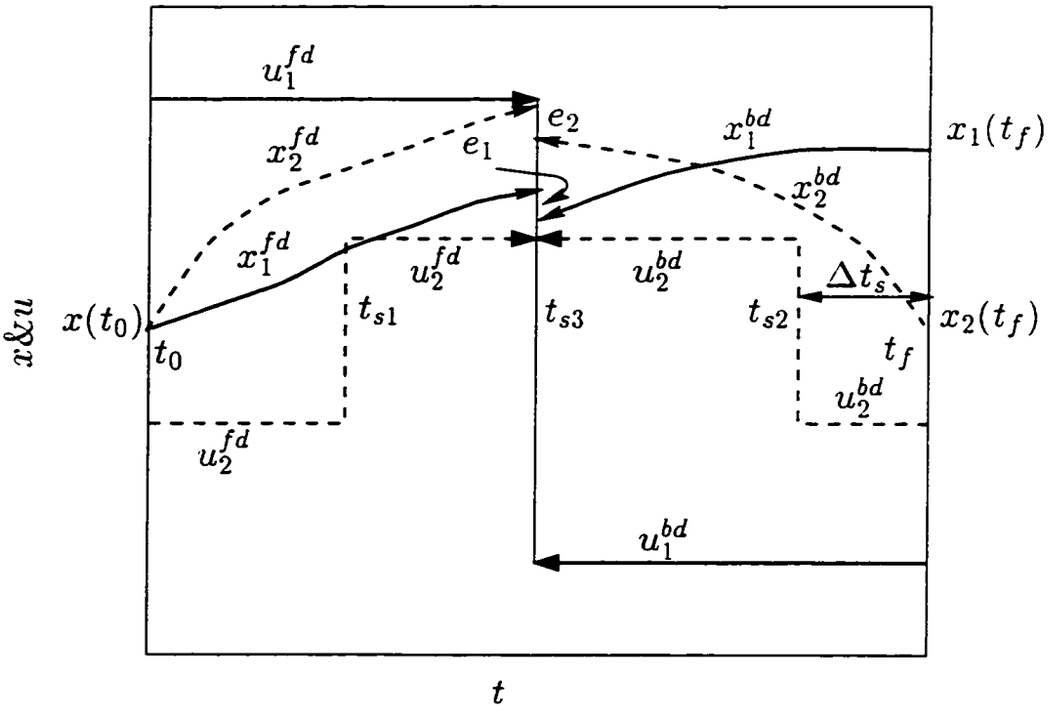

Figure 5.10: **Step 3.1** of the FBM, controls and states for shoulder link (x_1, x_2).

Step 4: Finding the costates

At the end of **Step 3** two or three switches, t_{s1}^*, t_{s2}^*, t_{s3}^*, in increasing order, and the final time, t_f are given such that:

$$t_{s1}^* < t_{s2}^* < t_{s3}^* < t_f \tag{5.39}$$

Step 4 can be carried out, even if **Step 3** is not completed, i.e. even if some discontinuity of the states is still present. As it was stated at the beginning of this section, two time instances are needed for numerical integration of the four costate equations. The first two switch times, t_{s1}^*, t_{s2}^*, are used to calculate the costates. Since (4.11) is valid for all time, one can use (4.10) and (4.11) to get costates in $t_{s1}^* \leq t \leq t_{s2}^*$. Suppose the control u_1 switches at t_{s2}^*, and u_2 switches at t_{s1}^*. Then there are four equations and four unknown costates, that can be solved using a TPBVP solver with

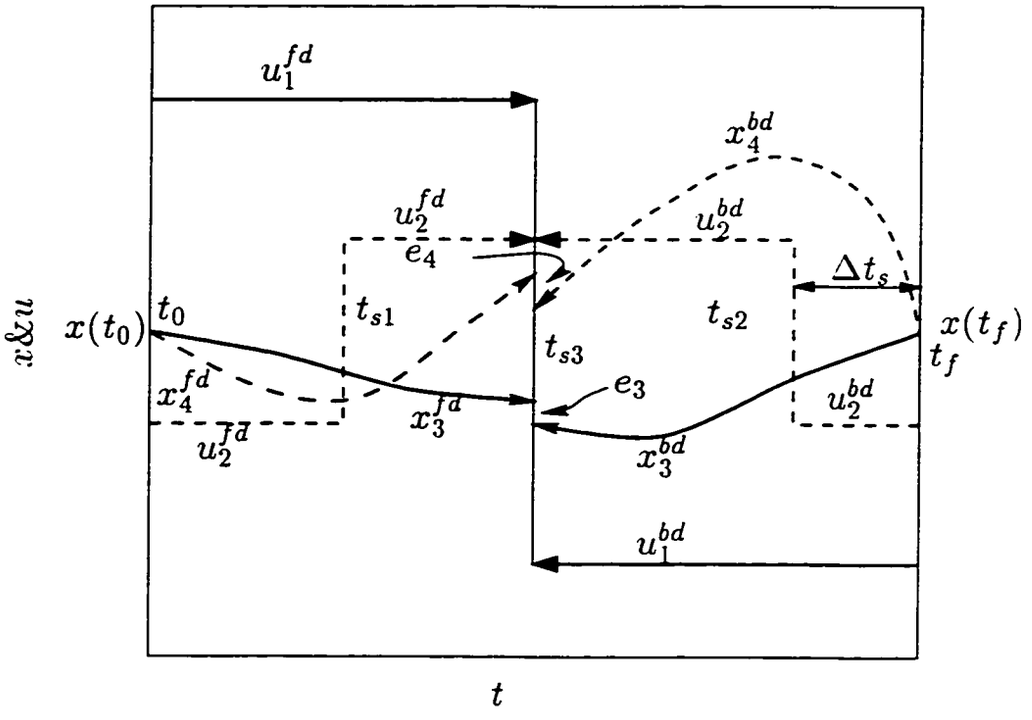

Figure 5.11: **Step 3.1** of the FBM, controls and states for elbow link (x_3, x_4).

the following boundary conditions,

$$\left[\begin{array}{c} H(t_{s1}^*) = 0 \\ G_2(t_{s1}^*) = 0 \\ H(t_{s2}^*) = 0 \\ G_1(t_{s2}^*) = 0 \end{array}\right] \Longrightarrow p(t) \quad \text{for} \quad t_{s1}^* \leq t \leq t_{s2}^* \tag{5.40}$$

where $G_2(t_{s1}^*)$ and $G_1(t_{s2}^*)$ are switch functions (4.10) at t_{s1}^* and at t_{s2}^*. Since the costates are dependent on the states, the state and costate equations have to be integrated simultaneously. Consequently, the states are needed in t_{s1}^* or t_{s2}^*. First the state equations should be integrated from t_0 to t_{s1}^* to obtain $x(t_{s1}^*)$. Then, using $x(t_{s1}^*)$ and the implicit boundary conditions (5.40) for p, the states and the costates should be integrated simultaneously forward from t_{s1}^* to t_{s2}^*. In order to obtain $p(t_0)$, the

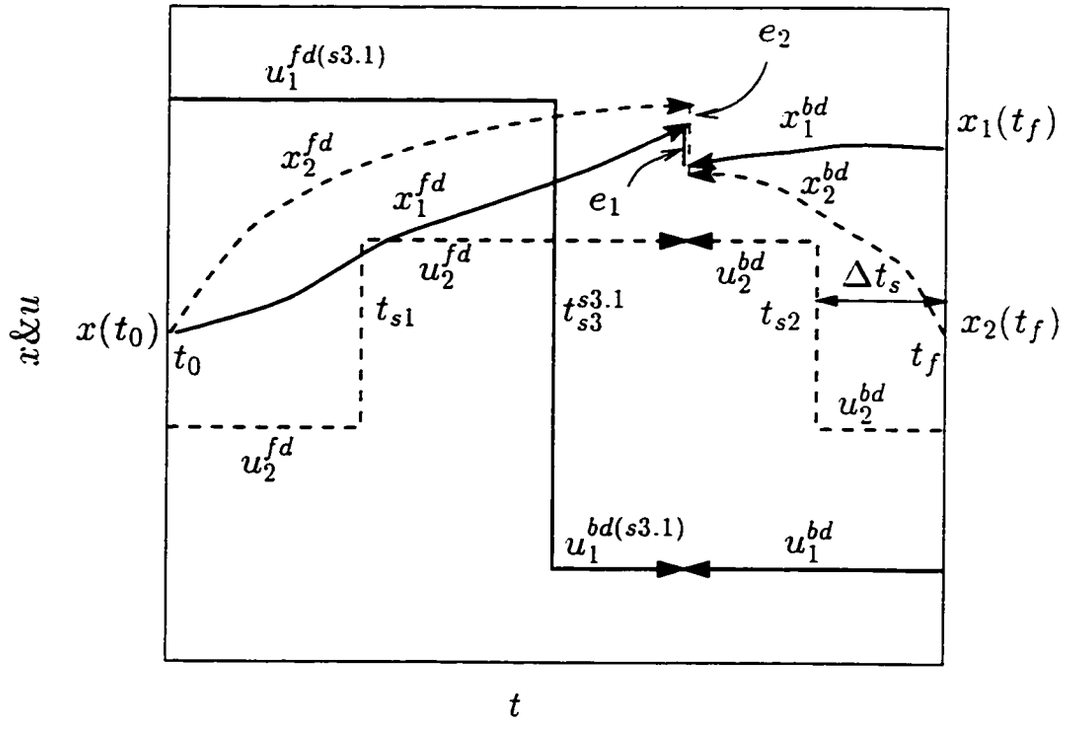

Figure 5.12: **Step 3.2** of the FBM, controls and states for shoulder link (x_1, x_2).

controls of **Step 3** can be used and the states and the costates can be integrated simultaneously backward from t_{s1}^* to t_0 using $x(t_0)$ and $p(t_{s1}^*)$ as the boundary conditions. For calculating $p(t_f)$ the states and the costates have to be integrated simultaneously forward in time with the sequence of controls as in **Step 3**. The states and the costates can be integrated simultaneously forward from t_{s2}^* to t_{s3}^* using $p(t_{s2}^*)$ and $x(t_{s2}^*)$ as initial conditions. Finally, the states and the costates have to be integrated simultaneously forward from t_{s3}^* to t_f using $p(t_{s3}^*)$ and $x(t_{s3}^*)$ as initial conditions. If there are more than three switch times, the integration can be continued forward in time from t_{si}^* to $t_{s(i+1)}^*$.

This strategy ensures that the boundary conditions for the states are satisfied (though at one of the switch times the discontinuity may still be present) and the continuity of the costates over the entire time domain is ensured. Thus, as explained

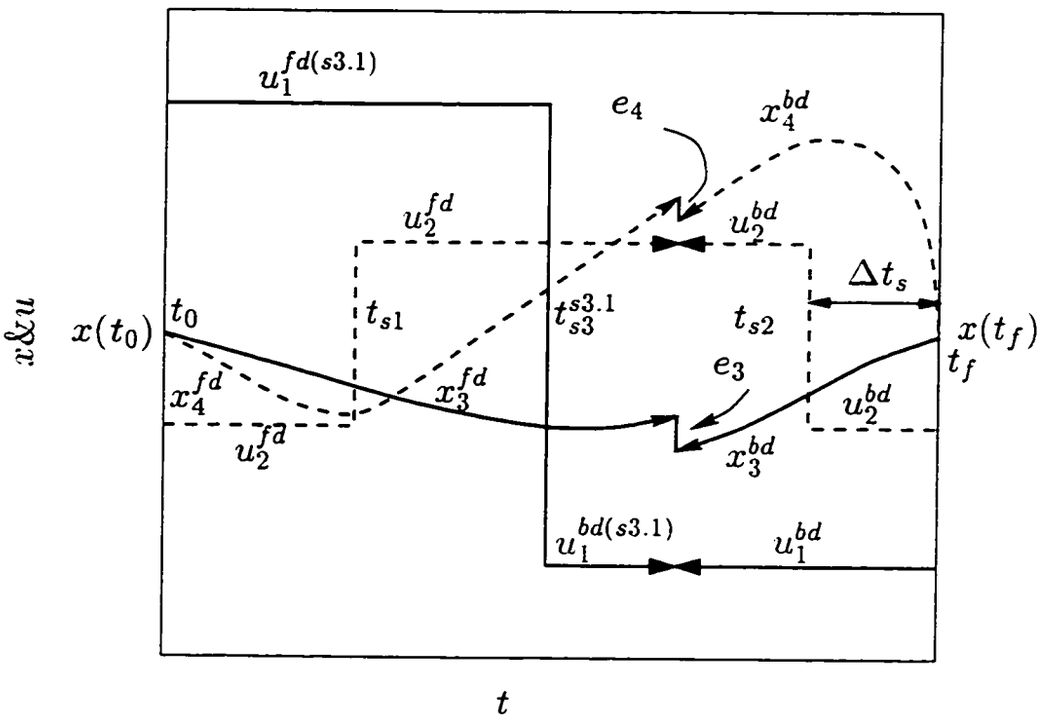

Figure 5.13: **Step 3.2** of the FBM, controls and states for elbow link (x_3, x_4).

above, for calculating the costates only the first two switches have been used. In fact, for this purpose any two switches could be used.

At this moment the state and the costate are determined. However, even if the states are continuous ($e^* \longrightarrow 0$) the switch functions probably do not match the calculated switch times. That is at t_{si} when one of the controls switches, the corresponding switch functions is not zero. This problem is considered in the next and subsequent iterations as discussed below.

5.4.2 Subsequent Iterations

Iteration One is repeated from Step 3 using $p(t_0)$ and $p(t_f)$ from the previous iteration as initial values for the costates, and x_0, x_f as initial and final values for the states, and (4.10) is used for defining the controls. Since the switch functions $G_i(x, p)$ are

now explicitly calculated, there is no need to guess the sign of control forces in this iteration. Also, these switch functions determine the new set of the switch times $t_{si}^{(k+1)}$ found from the conditions $G_i(x,p) = 0$. The criterion for continuity of the states at the switch time t_{si} is now:

$$x(x_0, u_i^{fd}, t_{si}) = x(x_f, u_i^{bd}, t_f - t_{si}) \tag{5.41}$$

where u_i^{fd}, u_i^{bd} are determined by the switch functions (4.10).

The following convergence norm, defining optimality of the solution, can be specified at the end of each iteration:

$$\epsilon_s = \sqrt{(\epsilon_{s1}^2 + \cdots + \epsilon_{sm}^2)/m} \tag{5.42}$$

where:

$$\epsilon_{sj} = \left| t_{sj}^{(k+1)} - t_{sj}^{(k)} \right| / t_f^{(k)}$$

Here $t_{sj}^{(k)}$ is the jth switch time, and $(t_f^{(k)})$ is the final time calculated in the iteration k. Satisfying (5.41) means that the state equations are met for controls obtained from (4.10) starting with $p(t_0)$ and $p(t_f)$ of the previous iteration.

Note that if $\epsilon_s \longrightarrow 0$, the states and costates equations will be satisfied and the optimal solution will be generated. It means that the global error in each iteration is going to zero and the switch times are matching the switch functions. This can only happen when there is no state discontinuity; i.e. only if $e^* \longrightarrow 0$.

5.4.3 Combining Forward-Backward & Shooting Methods

As mentioned before, in practice the process of reducing ϵ_s to zero was found to be slow. Better results were obtained by combining the Forward-Backward Method with the Shooting Method (SM). It has been found numerically that if $\epsilon_s < \epsilon_0$, the values of $p(t_0)$ generated by the FBM are good enough to cause the convergence of the SM.

The value of ϵ_0 depends on the physical parameters of the manipulator and the task. For the examples tried in Chapter 6 the FBM was terminated and the SM was started when $\epsilon_0 \approx 0.5$.

It should be emphasised that for the FBM iterations the initial and the final conditions for the states are always met; however, some discontinuity in the states may be present at one point of the trajectory. For the SM iterations, which starts with the initial conditions of the states and the initial conditions of the costates ($B = p(t_0)_{FBM}$) generated by the FBM, all the states are continuous but the final conditions have to be met in final iteration.

Here in order to explain the FBM, the trajectories with up to three switches have been considered. As soon as three switch solutions are obtained, then, if needed, the optimal initial costates of this class of motion can be used as starting point for the SM to obtain the optimal solution with four switches. Then using the continuation method, explained in Section (B.4), one can obtain the solution of original problem (using the optimal initial costate of the previous problem as initial guess for the next problem.)

Chapter 6

Results and Analysis

6.1 Numerical Examples

Three examples to show the usefulness of the method presented in Chapter 5 are discussed. The first example is a linear system with two states. This example was solved in [3] using the shooting method. The second and third examples illustrate the application of the method for the time-optimal control of Two-Link Manipulators (TLM).

6.1.1 Example One: Linear Problem

Here the linear mass and spring system introduced briefly in Chapter 1 is analysed. The equation of motion of that system (see Figure 1.2) is,

$$M\ddot{\varphi}(t) + K\varphi(t) = F(t)$$

with the limit on control force $|F(t)| \leq F_0$. Using non-dimensional variables $\bar{t} = \sqrt{\frac{K}{M}}t$, $\bar{\varphi} = \varphi\frac{K}{F_0}$, and $u(t) = \frac{F(t)}{F_0}$, this equation can be transformed to an ODE in the following form

$$\frac{d^2\bar{\varphi}}{d\bar{t}^2} + \bar{\varphi} = \frac{F}{F_0} = u$$

The non-dimensional equation of motion can be written in the form of the state equations if $x_1 = \bar{\varphi}$, $x_2 = \dot{x_1} = \frac{d\bar{\varphi}}{dt}$ and where $|u| = |F/F_0| \leq 1$. Now the time-optimal motion of the mass can be analysed as the following optimal control problem.

$$\text{minimise} \qquad t_f$$

subject to the state equations,

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= -x_1 + u \end{aligned} \tag{6.1}$$

The control bounds are,

$$-1 = U^- \leq u \leq U^+ = +1 \tag{6.2}$$

This problem is identical to the problem in [3] if the initial conditions are assumed as,

$$x_1(0) = 3.0 \quad x_2(0) = 0.5 \tag{6.3}$$

The final conditions are,

$$x_1(t_f) = 0.0 \quad x_2(t_f) = 0.0 \tag{6.4}$$

Using Pontryagin's Minimum Principle (PMP) the costate equations are,

$$\begin{aligned} \dot{p_1} &= p_2 \\ \dot{p_2} &= -p_1 \end{aligned} \tag{6.5}$$

and the Hamiltonian is

$$\mathcal{H}(x, p, u) = 1 + p_1\ x_2 + p_2\ (-x_1 + u) \tag{6.6}$$

The control forces can be obtained from the PMP as

$$u = \left\{ \begin{array}{ll} U^+ & \text{if } p_2 < 0 \\ U^- & \text{if } p_2 > 0 \end{array} \right\} \tag{6.7}$$

The equations (6.1), (6.3), (6.4), (6.5), (6.7) form the Two-Point Boundary Value Problem (TPBVP) for $0 \leq t \leq t_f$.

For starting the Shooting Method (SM) the state with $x_1(0)$, $x_2(0)$ and the costate equations with $p_1(0) = B_1$, $p_2(0) = B_2$ have to be integrated from t_0 to t_f, where B_1, B_2 are the assumed unknown initial costates. The target error vector given by (5.7) should vanish at the final time t_f. The components of this vector (see Equation (5.9)) are:

$$L_1 = x_1(t_f) \quad L_2 = x_2(t_f) \quad L_3 = \mathcal{H}(U^-, t_f) \times \mathcal{H}(U^+, t_f) = p_2^2(t_f) - 1 \tag{6.8}$$

Note that $\mathcal{H}(t_f) = 1 + p_2(t_f)u(t_f)$, where $u(t_f)$ can take the values of ∓ 1. In order to avoid guessing the sign of $u(t_f)$, the component L_3 of target error is formulated in such a way that it should be zero in both cases ($u = +1$ or $u = -1$). The target error norm $\|L\|$ is:

$$\|L\| = \sqrt{(L_1^2 + L_2^2 + L_3^2)} \tag{6.9}$$

The shooting method is terminated when this norm is less than a small positive value, $\|L\| = 10^{-6}$, in this case. Here the unknowns are B_1, B_2, t_f. This example has been executed several times with different initial values of B_1, B_2, t_f.

In order to discuss convergence the following norm that defines the closeness of the initial and the converged values of the calculated parameters is introduced:

$$d = \sqrt{\left(\sum_{i=1}^{n} (B_i^{iv} - B_i^{ov})^2\right) + (t_f^{iv} - t_f^{ov})^2} \tag{6.10}$$

where B_i^{iv}, B_i^{ov} are initial values and optimal values for the initial costates, and n is

the number of costates. The parameters t_f^{iv}, t_f^{ov} are initial values and optimal values for the final time. This norm is only a rough indication of the closeness of the starting point to the optimal point, mainly because not all of the costates in (6.10) appear in the switch functions which define the controls.

Table 6.1 summarises the results of these executions. If the initial costates are not close enough to their optimal values, the SM fails to converge (e.g. $d < 1.93$ for this example). The set of initial costates marked by *literature* in the Table refers to the starting point for the SM used in [3]. This set was not a blind guess for B and t_f. It was chosen by using some available information. The FBM set, shown in the second column of Table 6.1, indicates the set of initial costates obtained from the first iteration of the FBM described in Section 5.3. The approximated initial costates from FBM were sufficiently close to their optimal values and made the SM converge. The set *random* was chosen to cause divergence using trial and error from numbers close to the *FBM* and *literature*. The purpose of this set of initial data was to show that the SM is very sensitive to the starting point, and it may not converge, if the starting point is not close to the solution. As explained earlier, the norm d is only an indication of the closeness, and $d < 1.93$ for this example does not give a definite conclusion that the SM never converges for the values $d \geq 1.93$.

Table 6.1: Initial and final values of parameters for the SM (Example One).

parameters	*literature*	*FBM*	*random*	*converged*
B_1	0.89	0.75	$-.93$	0.93
B_2	0.44	0.68	0.71	0.36
t_f	5.00	5.26	4.61	4.99
d	0.09	0.46	1.93	0.00

Figure 6.1 shows the convergence of the SM with different initial values. The *random* set has diverged. This Figure shows that the convergence of the SM can be fast if the unknown initial costates are sufficiently close to their optimal values (small d), or that convergence may never happen if they are not sufficiently close to

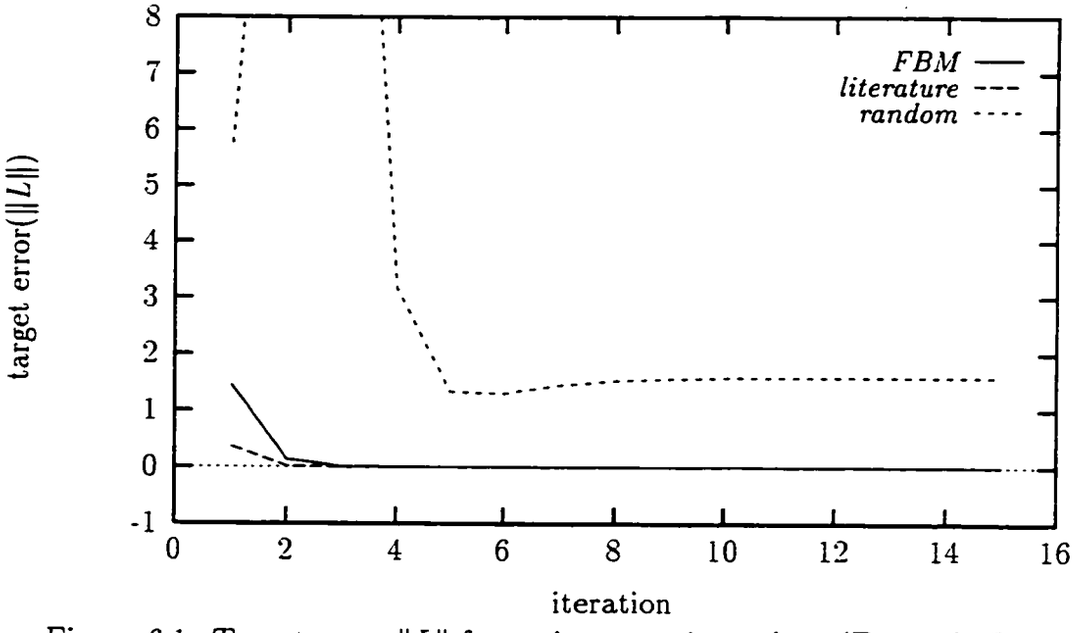

Figure 6.1: Target error $\|L\|$ for various starting values (Example One).

the optimal. Figure 6.2 shows the time-optimal control solution for this example. As can be seen, the switch function G is matched with the control u. Each time $G = 0$, there is a change in the sign of the control (u switches from one extreme to another). Figure 6.3 shows the first iteration of FBM. The superscripts indicate the number of switches, and the subscripts indicate the time. The state integration starts from A (forward) and from B (backward). The corresponding trajectory is denoted as x^1 with one switch at $x^1_{t_1}$ denoted as C_1. After calculating the costates and integrating the states x^2 and costates p^2 in reverse from C_1, another switch is located at $x^2_{t_1}$. Continuing the reverse integration up to $t_0 = 0$ the state reaches the point x^2_0 and the costate point p^2_0. For comparison the exact solutions denoted as x and p are also shown. As can be observed, the FBM in the second step of the first iteration (calculating the costates) produced a poor approximation for the state variables but a good approximation for the costate variables.

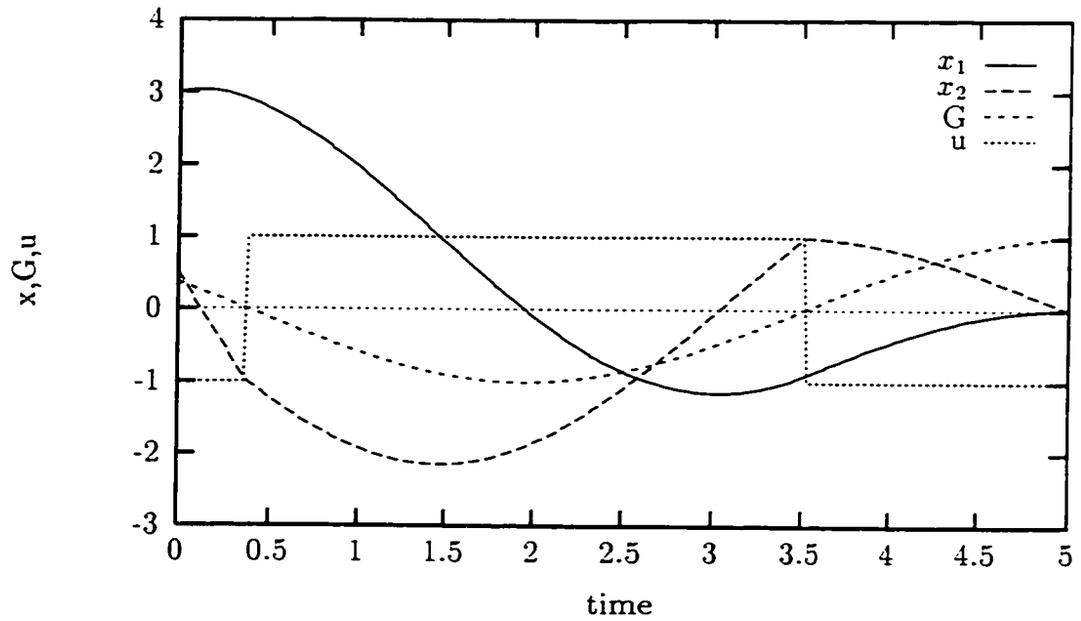

Figure 6.2: Time-optimal control solution (Example One).

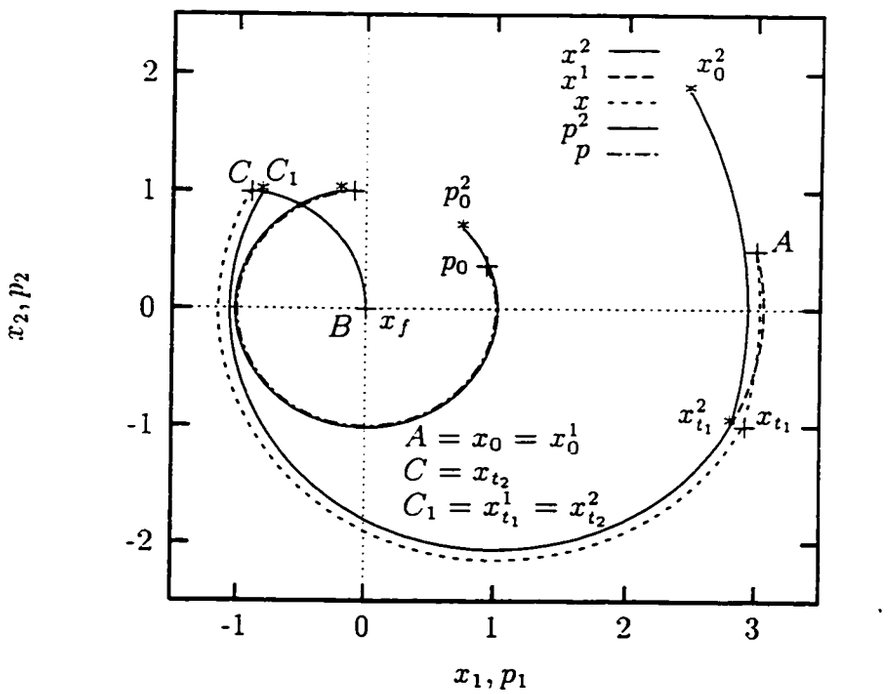

Figure 6.3: Iteration One of the FBM (Example One).

Examples for two-link manipulators

Here two examples of application of the procedure that combines the FBM with the SM for the time-optimal control of Two-Link Manipulators (TLM) is discussed. It should be noted that only two examples of the time-optimal control of TLM were found in the literature. The solutions reported were found using different methods from the proposed method, as discussed in the introduction. To verify the method presented here, these two cases, referred as Example Two and Example Three, are analysed. Example Two is a rest-to-rest motion from straight-to-straight configurations ($\varphi_2(0) = 0, \varphi_2(t_f) = 0$) with four state variables and two controls (Figure 6.4). The physical properties have been taken from [46]. It will be shown that results presented in [46] are, in fact, non optimal. Example Three is also a rest-to-rest motion from straight-to-straight configurations ($\varphi_2(0) = 0, \varphi_2(t_f) = 0$) with four state variables and two controls (Figure 6.14). However, the physical properties have been taken from [29] and [40]. With this example, it is shown that some conclusions reached in those papers related to the number of switches in the time-optimal motion are not correct.

For starting the Shooting Method (SM) the state with x_0, and the costate equations with $p_i(t_0) = B_i$ have to be integrated from t_0 to t_f, where B_i are the initial values of costates. The target error vector (5.7), should vanish at the final time t_f. The components of this vector as indicated by Equation (5.9) are:

$$L_i[x(B, t_f), t_f] = v_i \times [x_i(t_f) - x_{if}] \quad i = 1, \ldots, 4$$
$$L_5[x(B, t_f), t_f] = v_5 \times \mathcal{H}(t_f) \tag{6.11}$$

and the target error norm $\|L\|$ is:

$$\|L\| = \sqrt{L_1^2 + L_2^2 + L_3^2 + L_4^2 + L_5^2} \tag{6.12}$$

Here, v_i are weight functions to accommodate the difference of the magnitudes of the states as well as of the Hamiltonian. The choice of v_i is discussed in Appendix D. The unknowns are B_1, B_2, B_3, B_4, t_f. When this norm ($\|L\|$) is less than a small positive pre-specified value, the iterations in the shooting method are terminated. For the numerical examples the convergence criteria is set to $\|L\| = 10^{-6}$. The closeness norm of the initial and the converged values of the calculated parameters is the same as in Equation (6.10).

6.1.2 Example Two: Motion in a Vertical Plane (Gravity)

This example is a rest-to-rest motion of a two-link manipulator in the gravitational field from straight-to-straight configurations ($\varphi_2(0) = 0, \varphi_2(t_f) = 0$). The physical parameters as reported in [46] are as follows:

$$\begin{array}{lll} l_1 = 2l_{c1} = 0.2\ [m] & U_1^{\mp} = \mp 10.0\ [Nm] & \\ l_2 = 2l_{c2} = 0.2\ [m] & U_2^{\mp} = \mp 5.0\ [Nm] & I_1 = 0.004167\ [kg.m^2] \\ m_1 = 1.0\ [kg] & m_a = 0 = m_b & I_2 = 0.004167\ [kg.m^2] \\ m_2 = 1.0\ [kg] & I_o = I_a = I_b = 0 & g_r = 9.81\ [m.s^{-2}] \end{array} \tag{6.13}$$

where I_1 and I_2 are the mass moment of inertia of the links with respect to their centres of gravity, I_a and I_b are mass moment of inertia of the masses m_a and m_b with respect to their centres, I_0 is mass moment of inertia of the mass at the shoulder joint including the inertia of the shoulder motor. The terms m_1 and m_2 are the masses of the shoulder and elbow links respectively, m_a is the mass at the elbow joint, and m_b the mass at the tip of the manipulator. The lengths l_1 and l_2 are of the links, l_{c1} and l_{c2} locate the centres of gravity of the links, $U_1^{\mp}$ and $U_2^{\mp}$ are bounds of controls (Figure 4.1). Following [46] the gravitational acceleration g_r is Earth gravity. To compare with the results reported in [46] where $\varphi_{1_0} = \varphi_1(0) = 60^0 = 1.047\ [rad]$, the

initial and the final conditions of the states in $[rad]$ and $[rad/s]$ are (Figure 6.4):

$$x(0) = \begin{bmatrix} 1.047 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T$$
$$x(t_f) = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \tag{6.14}$$

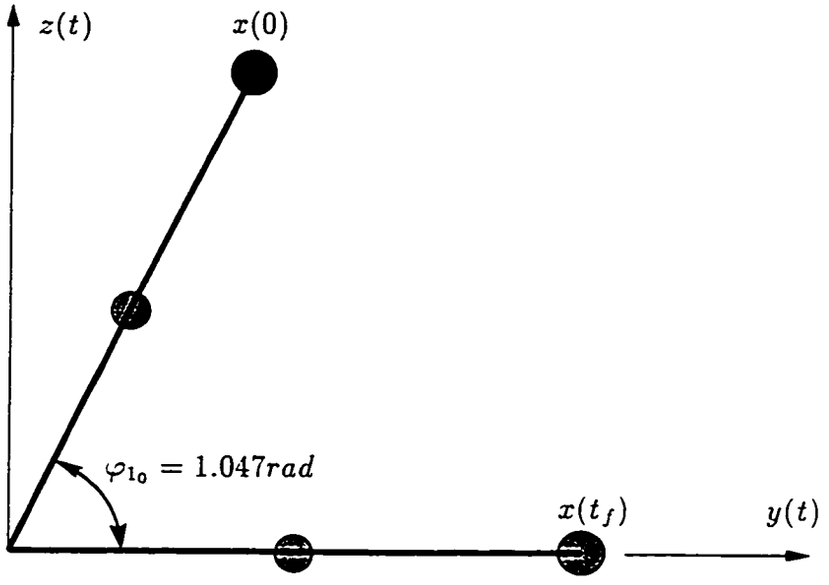

Figure 6.4: The initial and the final conditions of the TLM (Example Two).

This example has been executed several times using different methods to obtain the initial values for B_i, and final time t_f to start the SM method. Table 6.2 summarises the initial sets and the converged values. The Semi-Optimal Control (SOC) set was obtained using the location of the switch times t_{si} and t_f from the optimal control solution reported in [46]. This information and the boundary conditions (5.40) for the costates permit calculations of a set of initial costates by applying the procedure described in **Step 4** of **Iteration One** of the FBM method (see Section 5.4). The reason this set is called semi-optimal control is because the information of the semi-optimal solution reported in [46] has been used to generate the set of initial

Table 6.2: Initial and final values of parameters for the SM (Example Two).

parameters	*SOC*	*FBM*	*random*	*converged*
B_1	0.067684	1.141600	0.5	0.063566
B_2	0.006206	0.058565	0.5	0.006176
B_3	0.014805	0.470310	0.2	0.013303
B_4	0.001834	0.020709	0.2	0.001817
t_f	0.210000	0.191800	0.2	0.194385
d	0.016218	1.172229	0.713	0.0

costates and final time for the SM. The set defined as *FBM* was obtained using the FBM method, and the set *random* was selected to cause divergence using trial and error from numbers close to the *FBM* and *SOC* as explained for Example One.

Figure 6.5 shows the convergence of the SM with different initial values of B and t_f. The set *SOC* converged very quickly. However, it was based on the previous knowledge of the location of the switch times known from literature. Somewhat surprisingly it was found that the set *FBM* converged to two different solutions indicated by FBM(run1) and FBM(run2). In these two runs, different sets of weighting functions (v_i in Equation (6.11)) were used. The set *random* did not converge. Figure 6.6 shows the convergence of the final states in the SM with the initial values of costates generated by the FBM. As can be seen there, the final velocities of the links ($x_2(t_f)$, $x_4(t_f)$) converge more slowly than the rotations. Figure 6.7 shows the convergence of the switch times in the SM with initial values generated by the FBM. Comparing Figure 6.6 and Figure 6.7 indicate that when the number of switches is right (three here) deviation of the final calculated states from the given states are small. Figure 6.8 shows the trajectories of end points of the elbow and the shoulder at final iteration ($kf = 55$), as well as the path of elbow end point at 15th and 16th iterations ($k1 = 15$ and $k2 = 16$). In iteration $k2 = 16$ the number of switches was picked up correctly (see Figure 6.7); nevertheless, the trajectory missed the target point substantially.

There are two optimal control solutions generated from the *FBM* set that satisfy

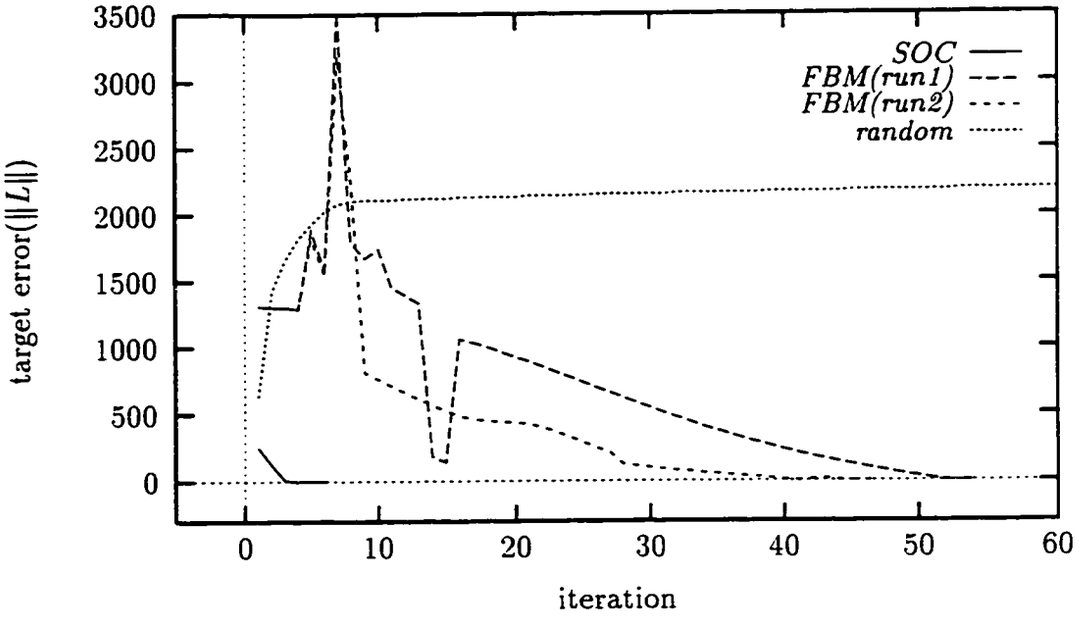

Figure 6.5: Target error $\|L\|$ for various starting values (Example Two).

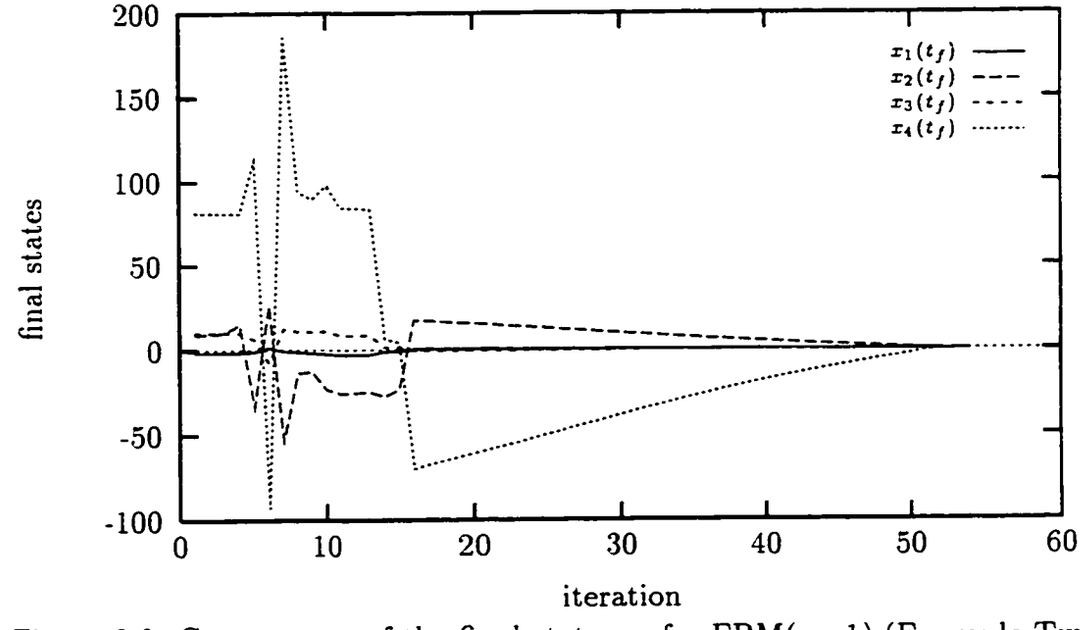

Figure 6.6: Convergence of the final states x_i for FBM(run1) (Example Two).

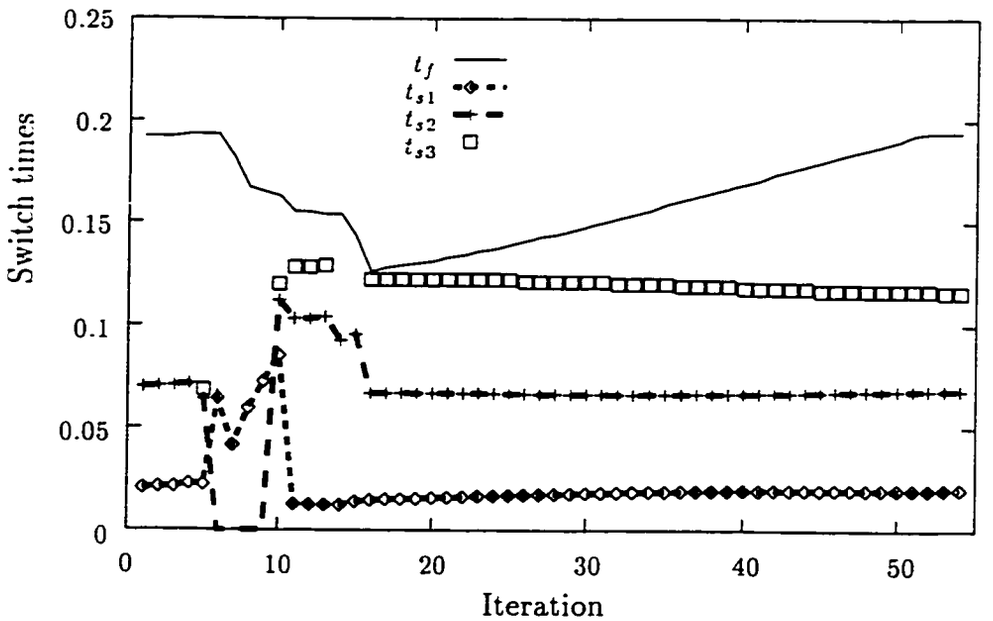

Figure 6.7: Convergence of the switch times t_{si} for FBM(run1) (Example Two).

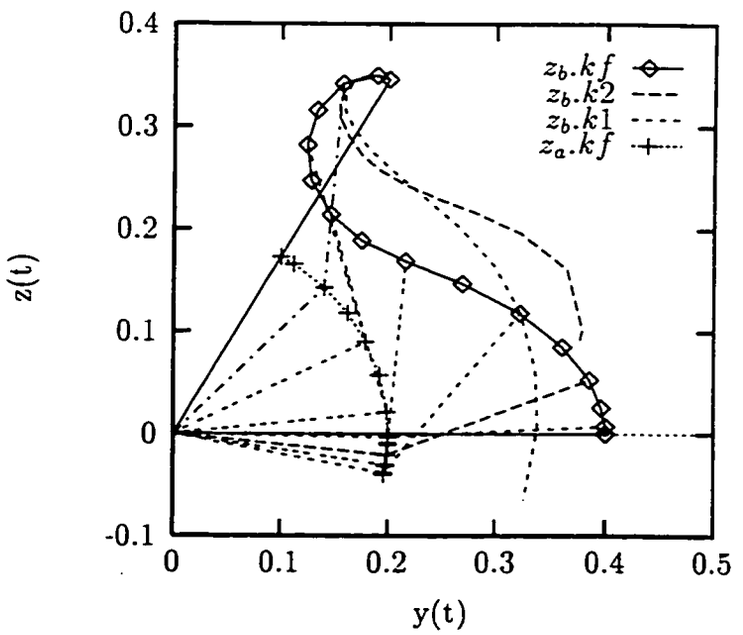

Figure 6.8: Trajectories of tip of the elbow link (y_b, z_b) and tip of the shoulder link (y_a, z_a) for FBM(run1) (Example Two).

the PMP; each represents a local minimum. The FBM(run1) solution which gives a smaller t_f and should be considered as a global optimum. It has three switches, ($t_{s1} = 0.019758, t_{s2} = 0.068059, t_{s3} = 0.116491$ sec), and the final time $t_f = 0.194385sec$ with the sequence for controls as in Figure 6.9. The FBM(run2) solution has four switches, ($t_{s1} = 0.012310, t_{s2} = 0.053549, t_{s3} = 0.086854, t_{s4} = 0.173929$ sec), and the final time $t_f = 0.197307sec$ with the sequence for controls as in Figure 6.10. In terms of t_f this solution is only 1.5% worse. Note that, for this example, a semi-optimal control solution with three switches was reported in [46] with $t_f = 0.2053sec$ (5.6% worse than FBM(run1) solution.) Because there are only two solutions for this initial and final conditions (three and four switch solutions), the shorter final time is considered global and the other local. As demonstrated in Section 6.2 this rotation angle $\varphi_{1_0} = 1.047$ is the border interval of three and four switch times solutions. The optimal solution has either three or four switch times as is explained in Section 6.2.

Figure 6.9 shows the scaled switch functions (250 G_i) of the time-optimal control of the manipulator for the FBM(run1) solution with three switches. Figure 6.10 shows the scaled switch functions (250 G_i) of the time-optimal control of the manipulator for the FBM(run2) solution with four switches. Figure 6.11 and Figure 6.12 show the optimal states and controls for the FBM(run1) solution with three switches. As can be seen, from these two figures, the magnitude of the velocities are considerably bigger than the magnitude of the rotations.

Figure 6.13 shows the trajectory of elbow link (y_b, z_b) for the FBM(run1) solution (3 switches) versus the FBM(run2) solution (4 switches) as well as semi-optimal control (SOC) solution with three switches as reported in [46]. Note that these two (FBM(run1) and FBM(run2)) optimal solutions have quite different trajectories despite almost identical final times.

As it is seen from Table 6.2, if the initial costates are not close enough to their optimal values, the SM fails to converge ($d \approx 0.713$ for this example). The initial costates obtained from the first iteration of the FBM ($d = 1.172229$) are close enough

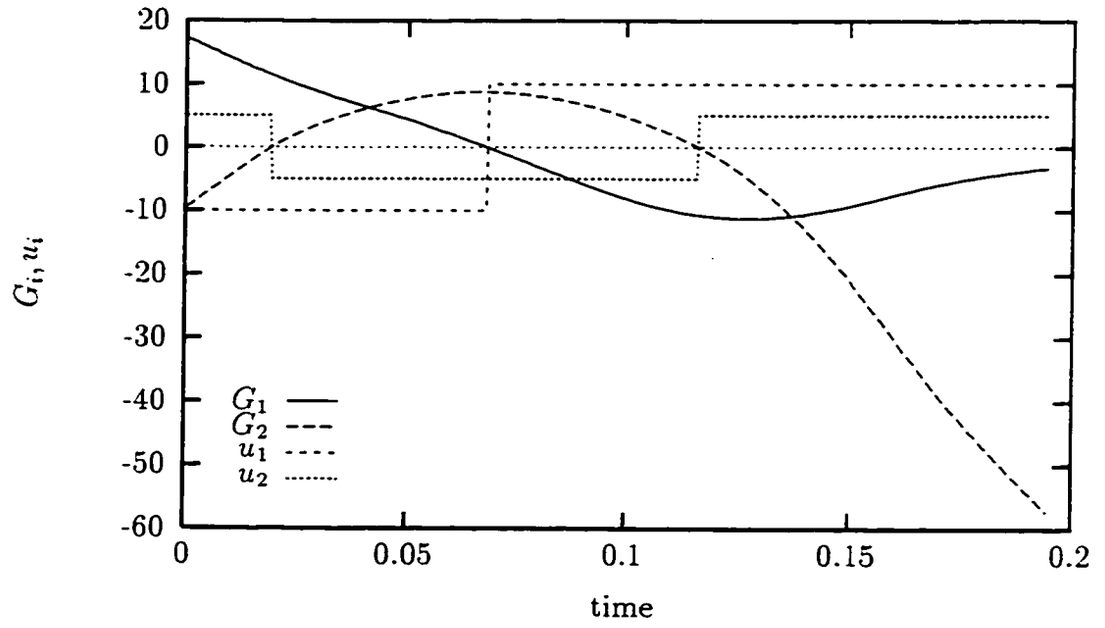

Figure 6.9: Optimal control, and scaled (250 G_i) switch functions FBM(run1) (Example Two).

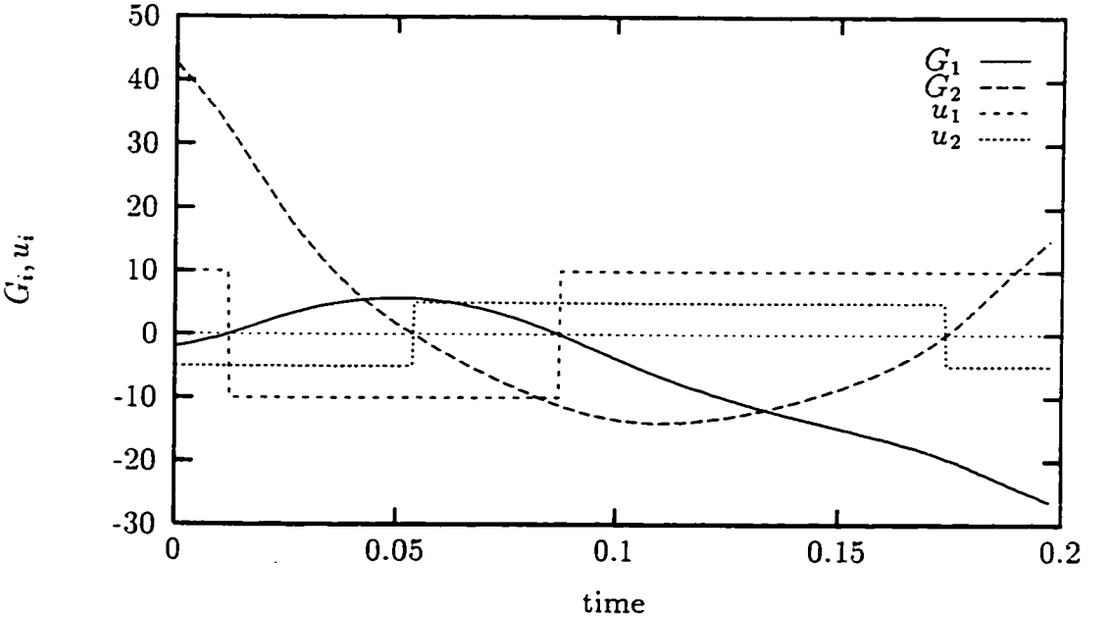

Figure 6.10: Optimal control, and scaled (250 G_i) switch functions FBM(run2) (Example Two).

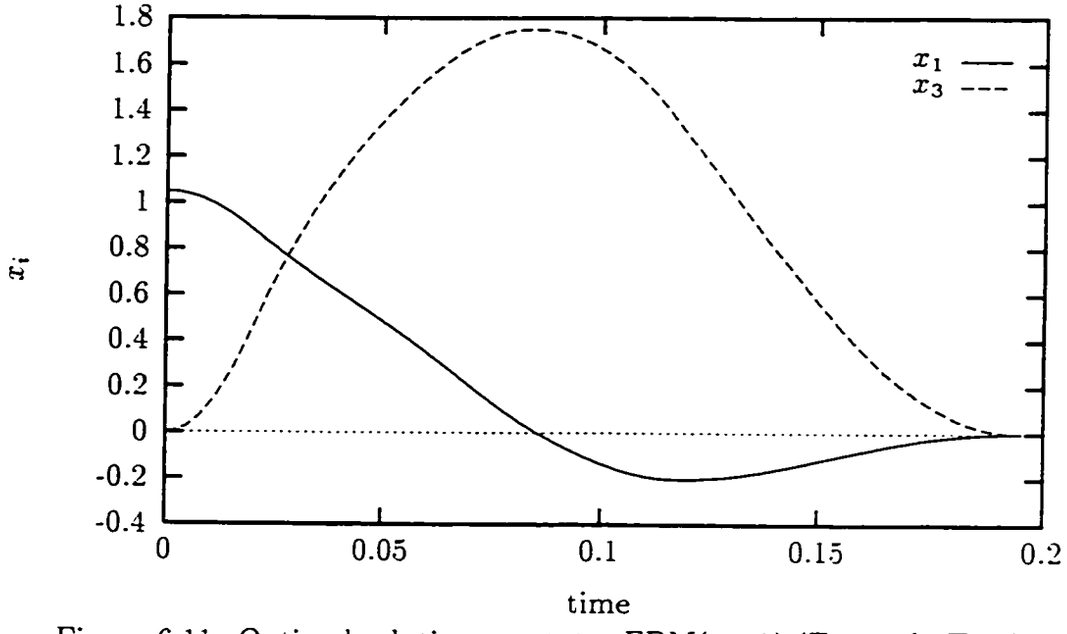

Figure 6.11: Optimal solution, x_i states FBM(run1) (Example Two).

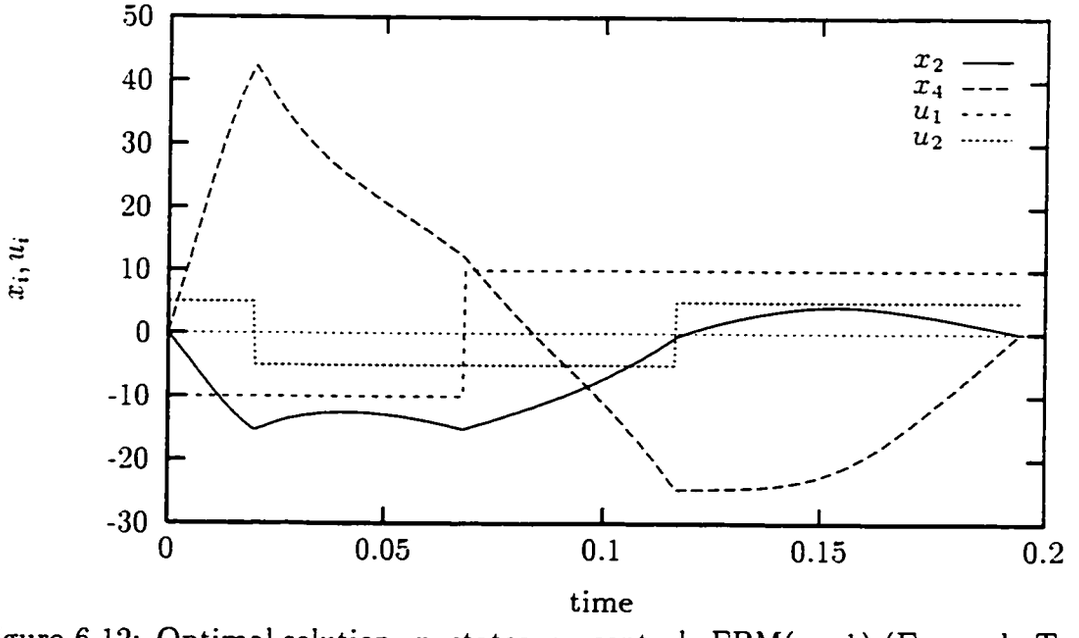

Figure 6.12: Optimal solution, x_i states, u_i controls FBM(run1) (Example Two).

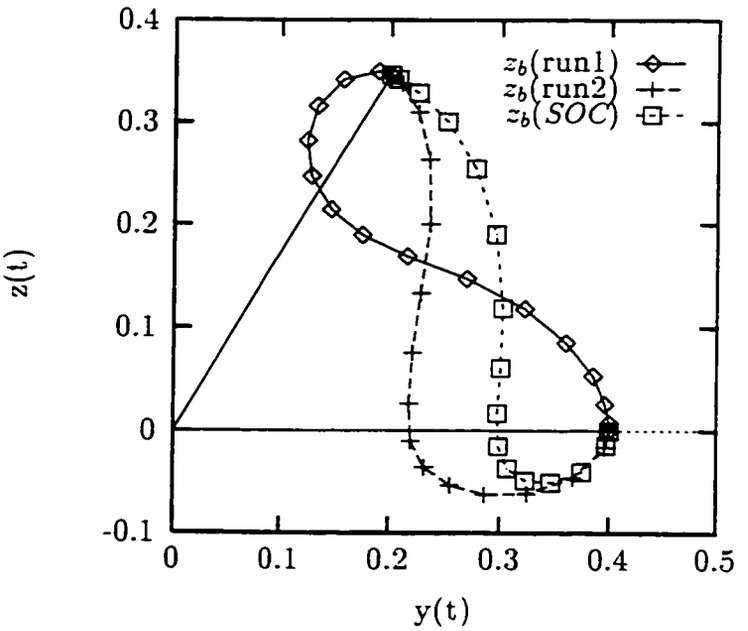

Figure 6.13: Trajectory of tip of elbow link (y_b, z_b) for FBM(run1) with $t_f = 0.1944$, FBM(run2) with $t_f = 0.1973$, SOC with $t_f = 0.2053$ (Example Two).

to their optimal values to allow the SM to converge. Although the parameter d for the *random* set is lower than the one for the FBM, the SM did not converge. As it was explained when the closeness norm d was defined in Equation 6.10, the norm d may not be a good indication of the closeness of the starting point to the optimal point. Particularly for this Example, the costates (p_1 and p_3), which define the controls and do not appear in the switch functions G_j, have the same weight as those which appear in the switch function (p_2 and p_4). Probably, this is the reason for having a smaller norm ($d = 0.713$) for the *random* set, which did not converge, than the FBM set norm ($d = 1.172229$), which did converge.

6.1.3 Example Three: Motion in a Horizontal Plane

This example is a rest-to-rest motion of a two-link manipulator without gravity from straight-to-straight configurations. The physical parameters are for the *IBM 7535 B 04 robot* as reported in [29] and [40] and are given as:

$$\begin{array}{lll} l_1 = 2l_{c1} = 0.40\ [m] & U_1^{\mp} = \mp 25.0\ [Nm] & \\ l_2 = 2l_{c2} = 0.25\ [m] & U_2^{\mp} = \mp 9.0\ [Nm] & I_1 = 0.416739\ [kg.m^2] \\ m_1 = 29.58\ [kg] & m_a = 0,\ m_b = 6.0\ [kg] & I_2 = 0.205625\ [kg.m^2] \\ m_2 = 15.00\ [kg] & I_o = I_a = I_b = 0 & g_r = 0\ [m.s^{-2}] \end{array} \tag{6.15}$$

In this example the initial and the final conditions of the states in $[rad]$ and $[rad/s]$ (Figure 6.14) are identical to those in [40]:

$$\begin{aligned} x(0) &= \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \\ x(t_f) &= \begin{bmatrix} 0.975 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \end{aligned} \tag{6.16}$$

Similarly, this example has been executed several times with different initial values for B_i, t_f. Table 6.3 summarises the initial sets and the converged values. The initial costates *SOC* (Semi-Optimal Control) were calculated in a similar manner to those in Example Two using information about the switch times for the optimal solution reported in [29] and [40].

Table 6.3: Initial and final values of parameters for the SM (Example Three).

parameters	*SOC*	*FBM*	*random*	*converged*
B_1	-1.086330	-0.990	2.0	-0.456692
B_2	-0.419652	-0.366	-2.0	-0.298622
B_3	-0.323267	-0.280	2.0	-0.093548
B_4	-0.111253	-0.949	-2.0	-0.074258
t_f	1.085000	0.795	2.0	1.083378
d	0.6821	1.0826	4.2260	0.0

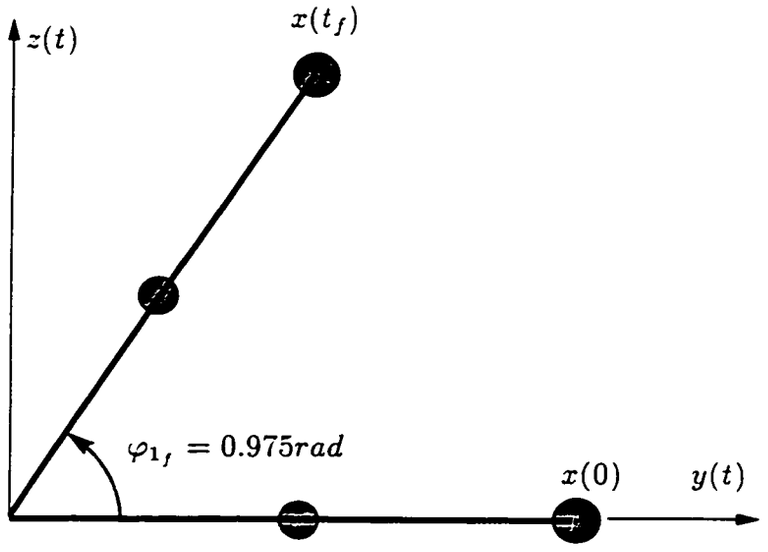

Figure 6.14: The initial and the final conditions of the TLM (Example Three).

Figure 6.15 shows the iterations of the SM with different initial values: the *SOC* and the *FBM* that converged, and the *random* set which did not converge. Figure 6.16 shows the convergence of the final states in the SM with initial values generated by the FBM. In this Figure all of the final states should go to zero except the final rotation of the shoulder link, which should be $x_1(t_f) = 0.975$. The optimal control solution has three switches, ($t_{s1} = 0.08751933, t_{s2} = 0.5416889, t_{s3} = 0.5872438$ *sec*), and the final time $t_f = 1.083378sec$. Figure 6.17 shows the convergence of the switch times in the SM with initial values generated by the FBM. Again comparing Figure 6.16 and Figure 6.17 indicate that when the number of switches is appropriate (three here) the SM continue to converge smoothly. Figure 6.18 shows the trajectories of end points of the elbow and the shoulder at final iteration ($kf = 22$), as well as the path of elbow end point at 5th and 7th iterations ($k1 = 5$ and $k2 = 7$). In those iterations the correct number of switch times are determined by the SM. Figure 6.19 shows the convergence of the initial costates B_i with initial values generated by the FBM for Example Three. As can be seen from that Figure, even if the variation of the initial costates are not

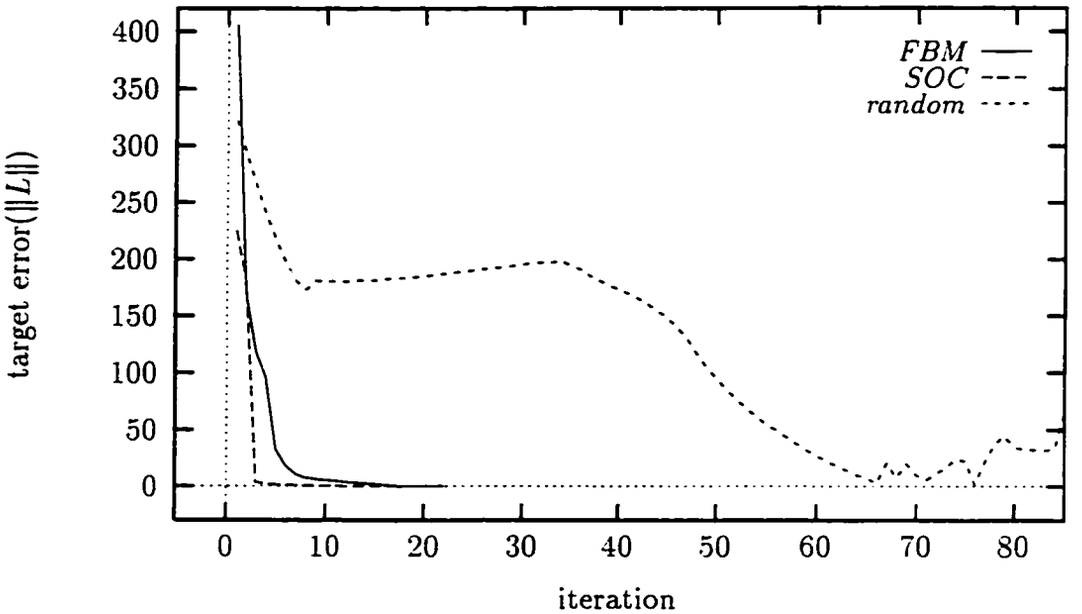

Figure 6.15: Target error $\|L\|$ for various starting values (Example Three).

much in magnitude ($-1.2 \leq B_i \leq 0$), but it has big effects on the convergence of the SM (see Figure 6.16), on the number of switches (see Figure 6.17), and on the trajectory of the tip of elbow (see Figure 6.18). Figure 6.20 shows the scaled switch functions ($250\ G_i$) of the time-optimal control solution of the manipulator. Figure 6.21 shows the optimal states solution for Example Three. The figure clearly indicates that the two-point boundary conditions are satisfied. As can be seen from Table 6.3, if the initial costates are not close enough to their optimal values, the SM fails to converge ($d \approx 4.22$ for this example). The initial costates obtained from the first iteration of FBM ($d = 1.0826$) are sufficiently close to their optimal values and the SM converges. The SM converged faster for the *SOC* set than for the *FBM* set, while for the *random* set did not converge during the test period.

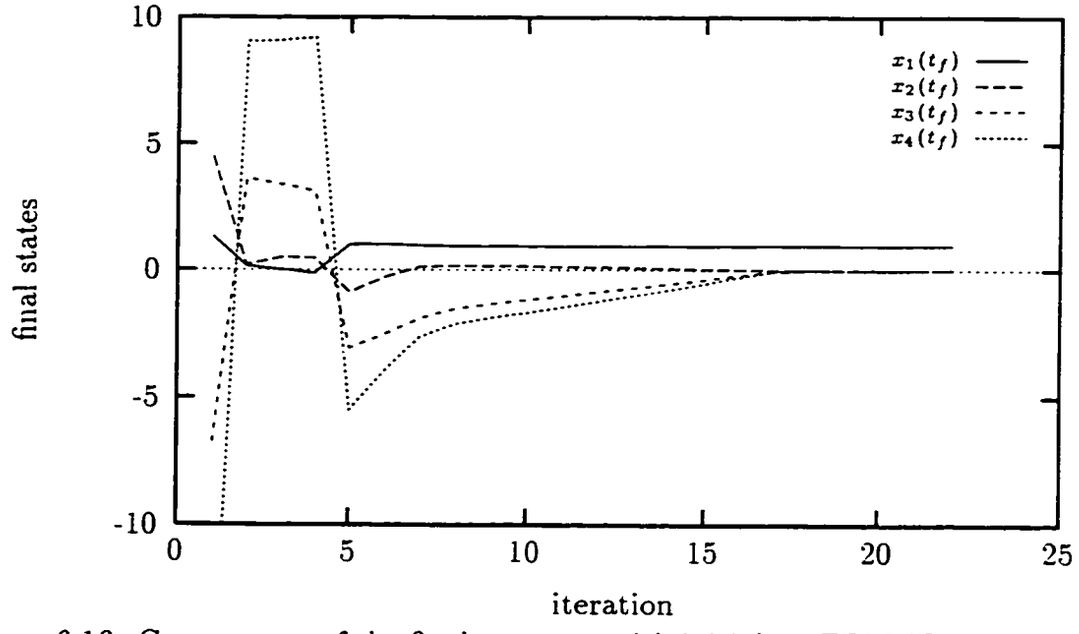

Figure 6.16: Convergence of the final states x_i with initial set FBM (Example Three).

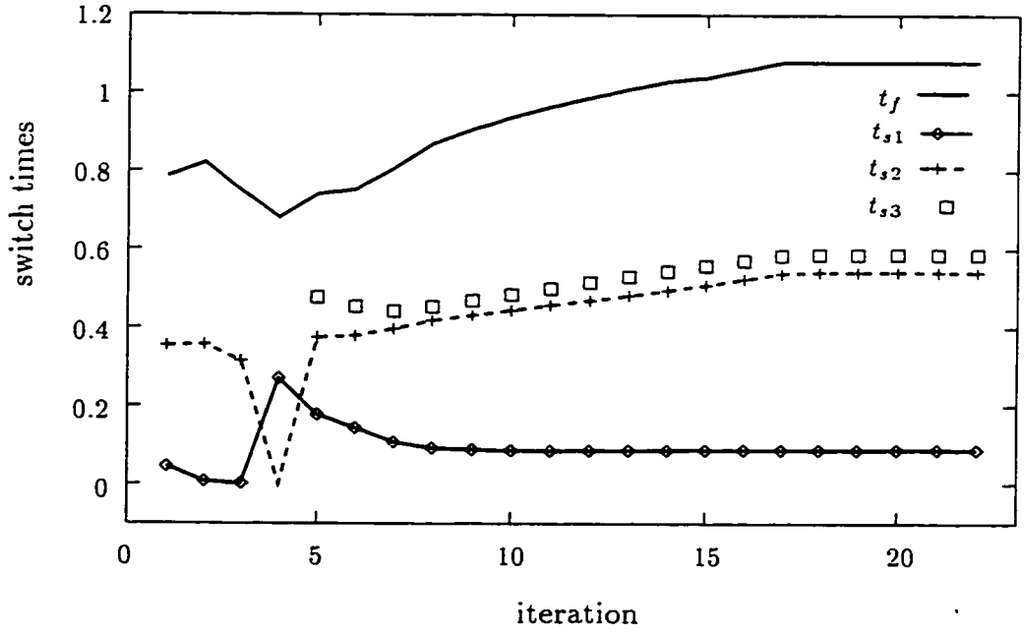

Figure 6.17: Convergence of the switch times t_{si} with initial set FBM (Example Three).

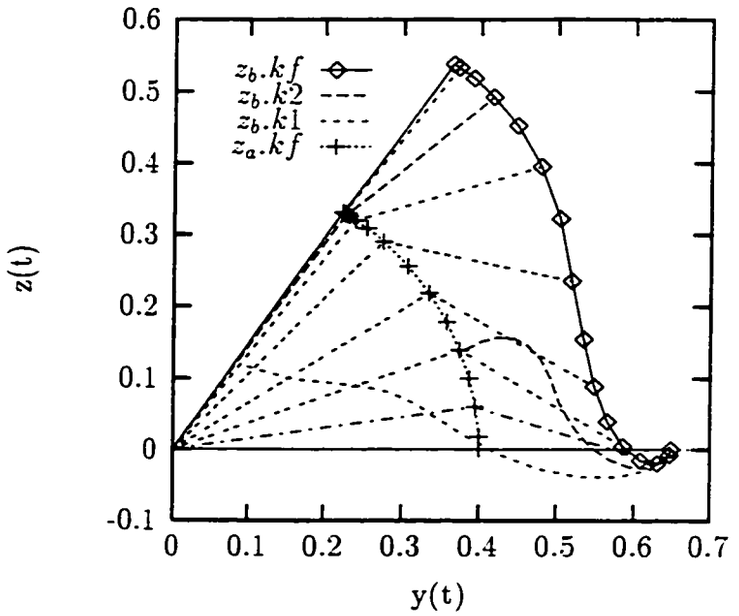

Figure 6.18: Trajectories of tip of the elbow link (y_b, z_b) and tip of the shoulder link (y_a, z_a) (Example Three).

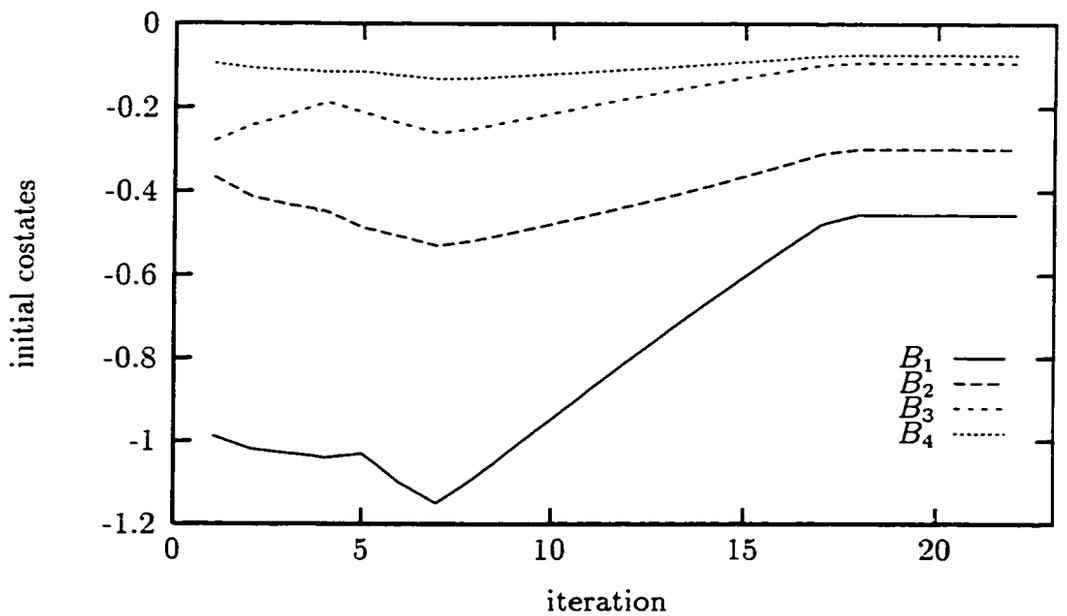

Figure 6.19: Convergence of the initial costates B_i with initial set FBM (Example Three).

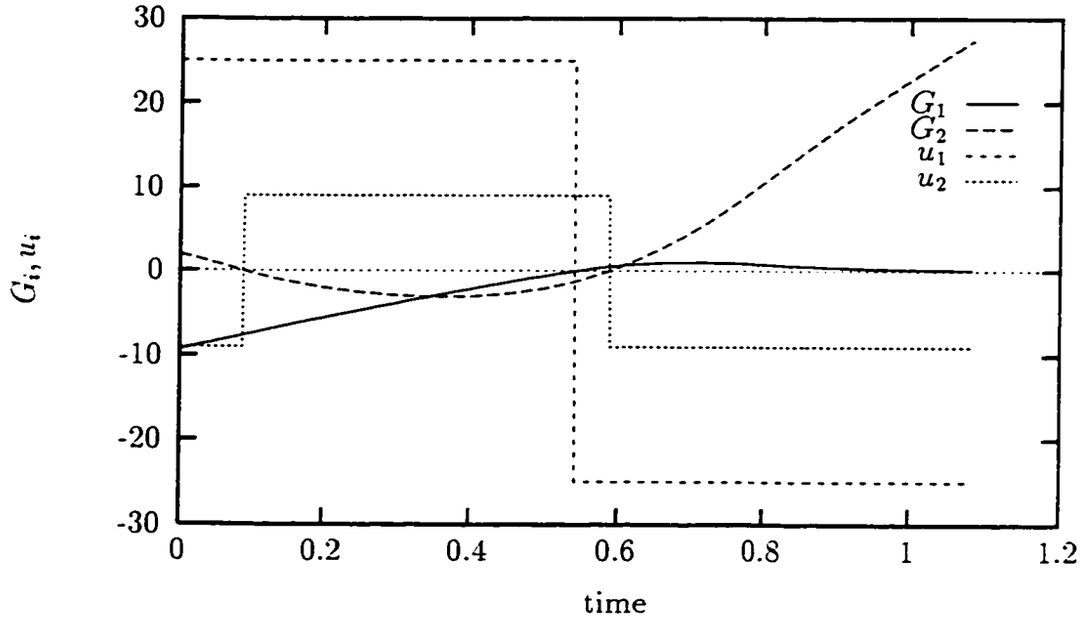

Figure 6.20: Optimal control, and scaled (250 G_i) switch functions (Example Three).

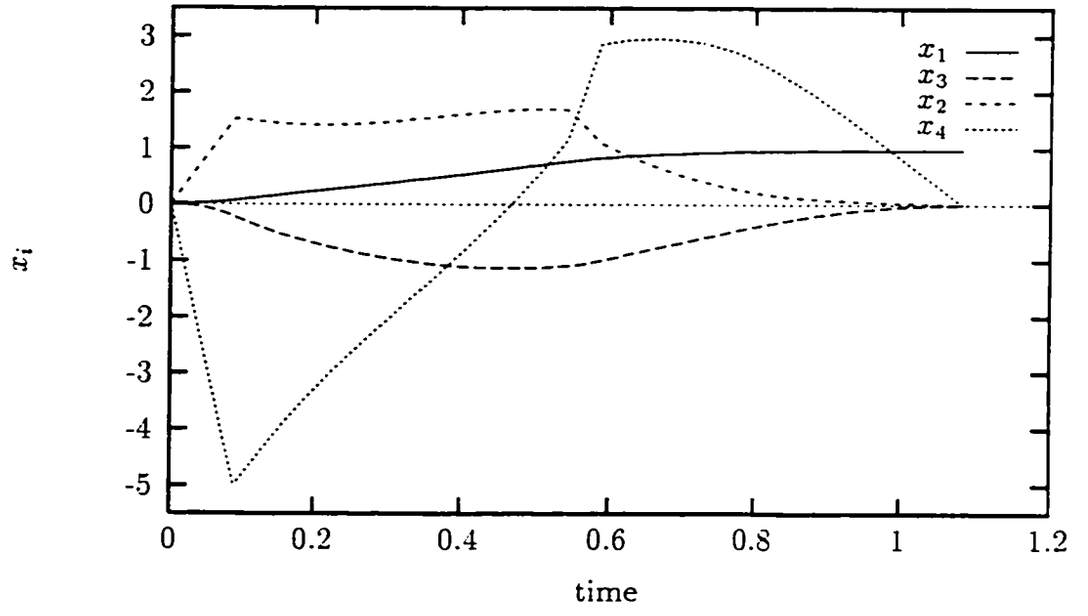

Figure 6.21: Optimal solution, x_i states (Example Three).

6.2 Effects of the Magnitude of Maneuver

In the cases of time-optimal controls of Two-Link Manipulators (TLM) presented in Section 6.1, the maneuvers were characterised by the angle $\varphi_{1_0} = \varphi_1(0)$ or $\varphi_{1_f} = \varphi_1(t_f)$ which was equal to $\varphi_{1_0} = 60^0$ for Example Two and $\varphi_{1_f} = 55.9^0$ for Example Three. Here the effects of magnitude of the maneuver on the time-optimal trajectories of the TLM in rest-to-rest motion from straight-to-straight configurations are analysed. First, the TLM presented in Example Two (see Section 6.1.2) with the physical parameters as reported in [46] and given in Equation (6.13), is considered. Next, the TLM presented in Example Three (see Section 6.1.3) with the physical parameters as reported in [29] and [40] and given in Equation (6.15), is discussed.

6.2.1 Two-Link Manipulator from Example Two (Gravity)

This example is a rest-to-rest motion of a two-link manipulator in a gravitational field from straight-to-straight configurations ($\varphi_2(t_0) = 0, \varphi_2(t_f) = 0$). The physical parameters as in [46] are given in Equation (6.13). The case $\varphi_{1_0} = 60^0 = 1.047[rad]$ was presented in Section 6.1.2, where two different time-optimal trajectories were found with almost identical maneuver times t_f, but different numbers of switches. Here, it is shown that the optimal solution for this TLM should have three and four switches. Schematically this situation is presented in Figure 6.22. The method presented here generated optimal solutions with only three switches for $\varphi_{1_0} < 1.047$, with only four switches for $\varphi_{1_0} > 1.19$, and double (three and four switches) solutions for $1.047 < \varphi_{1_0} < 1.19$. Double solutions interval is because the three and four switches solutions are overlapped. The typical configuration of optimal controls with three switches was shown in Figure 6.9, and with four switches is shown in Figure 6.23.

The following problem is an example when the number of switch times is four. Note that this solution contradicts the statement in [40] according to which the chance of having more than three switches for TLM is *"almost zero"*.

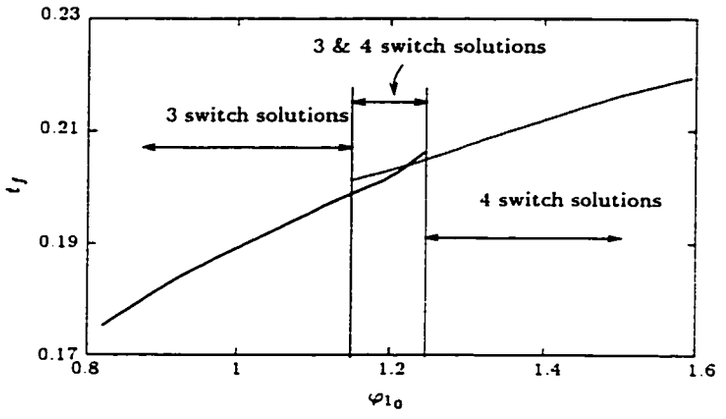

Figure 6.22: Number of switches in optimal maneuver (Example Two).

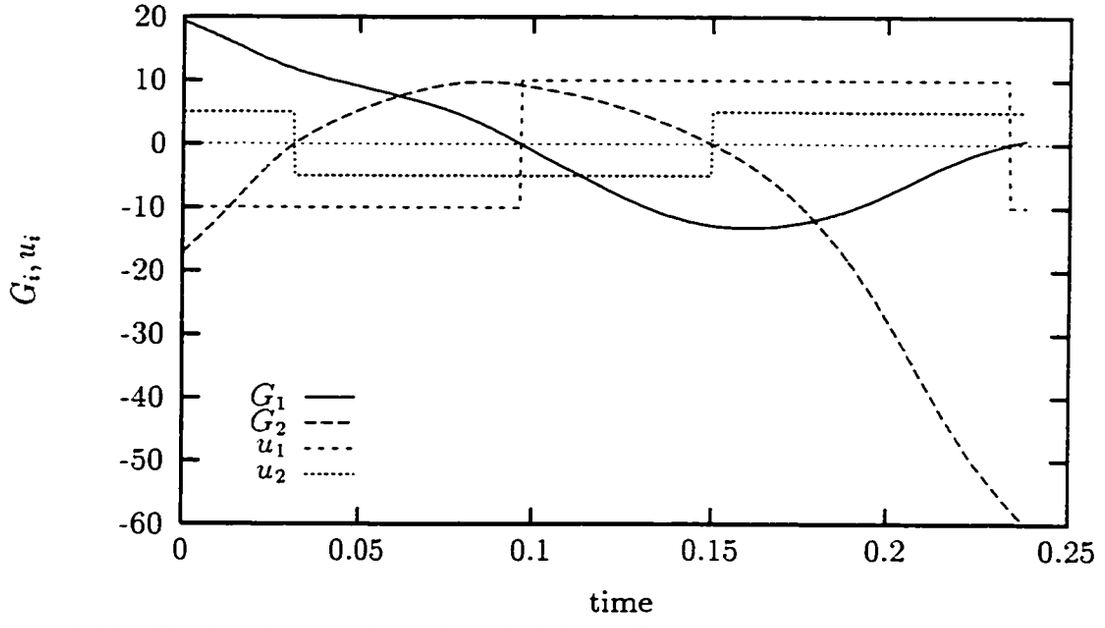

Figure 6.23: Optimal control, and scaled (250 G_i) switch functions (Example Two).

The initial and the final conditions of the states in $[rad]$ and $[rad/s]$ are:

$$x(t_0) = \begin{bmatrix} 2.09044 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T$$
$$x(t_f) = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \tag{6.17}$$

Figure 6.24 shows the trajectory of the optimal control solution which, for this case, ($\varphi_{1_0} = 119.8^0$) has four switches ($t_{s1} = 0.031250$, $t_{s2} = 0.096590$, $t_{s3} = 0.149692$, $t_{s4} = 0.216585$), and the final time $t_f = 0.233483$ *sec*. The sequence of controls and the switch functions (250 G_i) of the time-optimal control of the manipulators are shown in Figure 6.23. Figure 6.25 and Figure 6.26 show the optimal states; the controls are shown again for reference.

The calculations were carried out for φ_{1_0} varying from 0.01 to 2.09044 for this

Table 6.4: Converged switch times and final time for various φ_{1_0} (Example Two).

φ_{1_0}	t_f	t_{s1}	t_{s2}	t_{s3}	t_{s4}
0.01	$2.262984e-2$	$3.009902e-3$	$6.875248e-3$	$1.210933e-2$	
0.1	$7.099302e-2$	$9.080023e-3$	$2.165279e-2$	$3.869921e-2$	
0.3	$1.185223e-1$	$1.352152e-2$	$3.689333e-2$	$6.793785e-2$	
0.6	$1.583224e-1$	$1.655116e-2$	$5.137521e-2$	$9.393380e-2$	
0.9	$1.843600e-1$	$1.873540e-2$	$6.286723e-2$	$1.105064e-1$	
0.98	$1.899912e-1$	$1.929065e-2$	$6.571662e-2$	$1.139097e-1$	
1.047	$1.943858e-1$	$1.975870e-2$	$6.805973e-2$	$1.164911e-1$	
1.15	$2.006384e-1$	$2.049171e-2$	$7.159919e-2$	$1.200212e-1$	
1.19	$2.029181e-1$	$2.078223e-2$	$7.295600e-2$	$1.212581e-1$	
1.2	$2.034756e-1$	$2.090897e-2$	$7.322001e-2$	$1.217578e-1$	0.2033833
1.21	$2.040258e-1$	$2.108171e-2$	$7.341957e-2$	$1.224270e-1$	0.2037624
1.23	$2.051043e-1$	$2.141474e-2$	$7.383069e-2$	$1.237116e-1$	0.2045193
1.25	$2.061552e-1$	$2.173289e-2$	$7.425562e-2$	$1.249310e-1$	0.2052730
1.28	$2.076834e-1$	$2.218753e-2$	$7.491466e-2$	$1.266509e-1$	0.2063951
1.31	$2.091580e-1$	$2.262054e-2$	$7.559537e-2$	$1.282540e-1$	0.2075046
1.34	$2.105835e-1$	$2.303393e-2$	$7.629408e-2$	$1.297535e-1$	0.2086000
1.4	$2.133022e-1$	$2.381872e-2$	$7.773601e-2$	$1.324795e-1$	0.2107484
1.5	$2.175017e-1$	$2.503373e-2$	$8.024226e-2$	$1.363486e-1$	0.2142098
1.571	$2.202720e-1$	$2.584695e-2$	$8.207840e-2$	$1.386705e-1$	0.2165857
1.9	$2.318255e-1$	$2.935283e-2$	$9.110341e-2$	$1.464363e-1$	0.2272525
2.0	$2.351313e-1$	$3.035957e-2$	$9.396661e-2$	$1.481993e-1$	0.2305007
2.05	$2.367784e-1$	$3.085508e-2$	$9.541383e-2$	$1.490326e-1$	0.2321436
2.065	$2.372727e-1$	$3.100194e-2$	$9.584972e-2$	$1.492785e-1$	0.2326394
2.08	$2.377673e-1$	$3.114861e-2$	$9.628634e-2$	$1.495228e-1$	0.2331367
2.081	$2.377838e-1$	$3.115350e-2$	$9.630091e-2$	$1.495310e-1$	0.2331533
2.09	$2.381118e-1$	$3.125059e-2$	$9.659065e-2$	$1.496922e-1$	0.2334837

TLM. The number and the locations of switches as well as the final time of the bang-bang time-optimal control for these motions are given in Table 6.4 and plotted in Figure 6.27. These solutions have three or four switches. The optimal initial costates are given in Table 6.5. Figure 6.28 and Figure 6.29 show the initial costates $B_i = p_i(t_0)$ versus the rotation φ_{1_0}. As can be seen, in the time-optimal solutions, the number of the switches are three if $\varphi_{1_0} < 1.047$, and four if $\varphi_{1_0} > 1.19$.

Table 6.5: Converged initial costates for various φ_{1_0} (Example Two).

φ_{1_0}	B_1	B_2	B_3	B_4
0.01	$1.131076e-0$	$7.778124e-3$	$3.566842e-1$	$2.452987e-3$
0.1	$3.447834e-1$	$7.602686e-3$	$1.076539e-1$	$2.388014e-3$
0.3	$1.740699e-1$	$7.041198e-3$	$5.168767e-2$	$2.177734e-3$
0.6	$1.035751e-1$	$6.514514e-3$	$2.759441e-2$	$1.971925e-3$
0.9	$7.321899e-2$	$6.249575e-3$	$1.681304e-2$	$1.855829e-3$
0.98	$6.768495e-2$	$6.206348e-3$	$1.480558e-2$	$1.834003e-3$
1.047	$6.356638e-2$	$6.176809e-3$	$1.330301e-2$	$1.817900e-3$
1.15	$5.798168e-2$	$6.141888e-3$	$1.125638e-2$	$1.796556e-3$
1.19	$5.602172e-2$	$6.131519e-3$	$1.053674e-2$	$1.789308e-3$
1.2	$5.539384e-2$	$6.112890e-3$	$1.029867e-2$	$1.781736e-3$
1.21	$5.464557e-2$	$6.080387e-3$	$1.001038e-2$	$1.769187e-3$
1.23	$5.322102e-2$	$6.019279e-3$	$9.463446e-3$	$1.745497e-3$
1.25	$5.188338e-2$	$5.962862e-3$	$8.952294e-3$	$1.723507e-3$
1.28	$5.002142e-2$	$5.886020e-3$	$8.244985e-3$	$1.693334e-3$
1.31	$4.831331e-2$	$5.817481e-3$	$7.600884e-3$	$1.666153e-3$
1.34	$4.674115e-2$	$5.756377e-3$	$7.012765e-3$	$1.641664e-3$
1.4	$4.395515e-2$	$5.654160e-3$	$5.984043e-3$	$1.599869e-3$
1.5	$4.020567e-2$	$5.536446e-3$	$4.639369e-3$	$1.549111e-3$
1.571	$3.812295e-2$	$5.489864e-3$	$3.925378e-3$	$1.526292e-3$
1.9	$3.323645e-2$	$5.661760e-3$	$2.619796e-3$	$1.557207e-3$
2.0	$3.294882e-2$	$5.838763e-3$	$2.725410e-3$	$1.610092e-3$
2.05	$3.294835e-2$	$5.947118e-3$	$2.837967e-3$	$1.643378e-3$
2.065	$3.296335e-2$	$5.982022e-3$	$2.878014e-3$	$1.654192e-3$
2.08	$3.298465e-2$	$6.017973e-3$	$2.920700e-3$	$1.665364e-3$
2.081	$3.298546e-2$	$6.019189e-3$	$2.922166e-3$	$1.665743e-3$
2.09	$3.300296e-2$	$6.043597e-3$	$2.951868e-3$	$1.673345e-3$

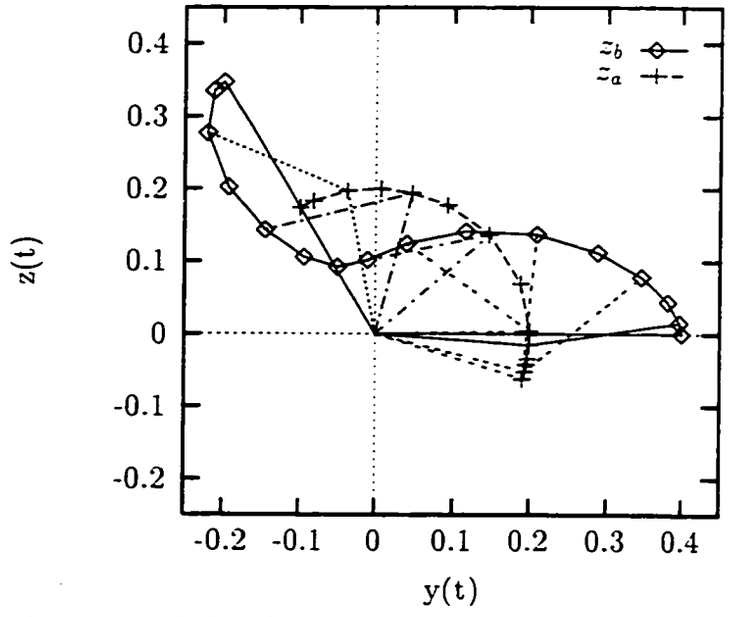

Figure 6.24: Trajectories of tip of the elbow link (y_b, z_b) and tip of the shoulder link (y_a, z_a) (Example Two).

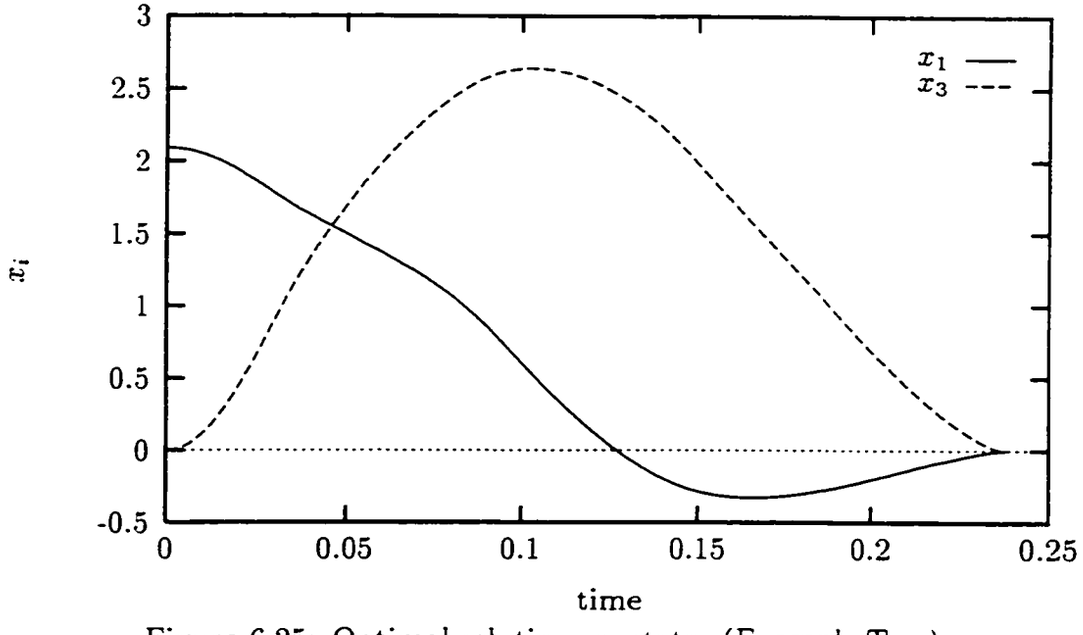

Figure 6.25: Optimal solution, x_i states (Example Two).

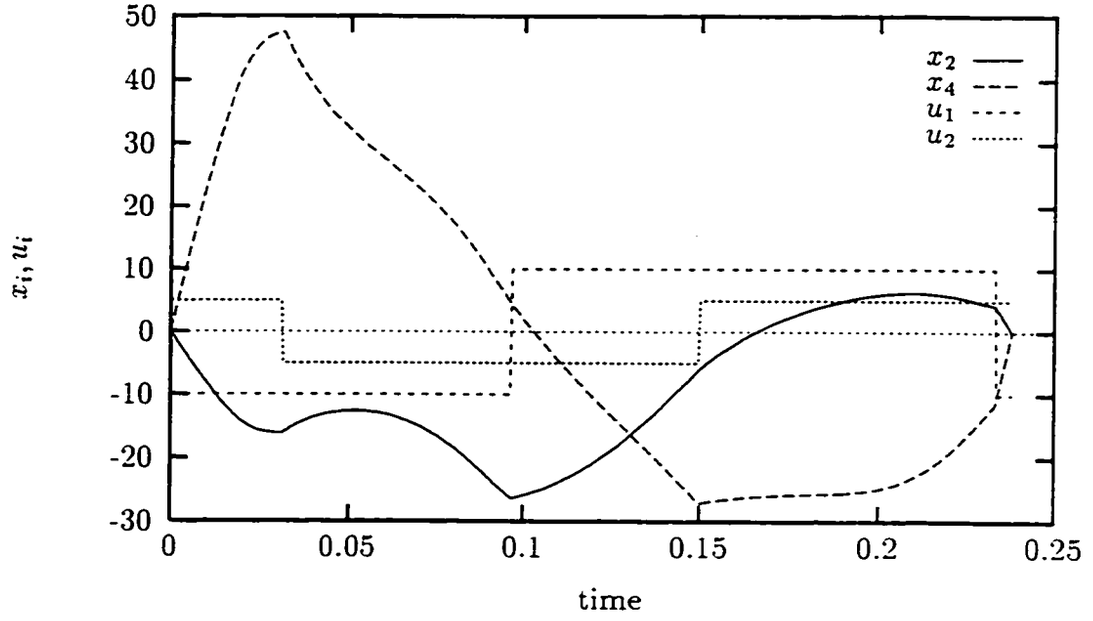

Figure 6.26: Optimal solution, x_i states, u_i controls (Example Two).

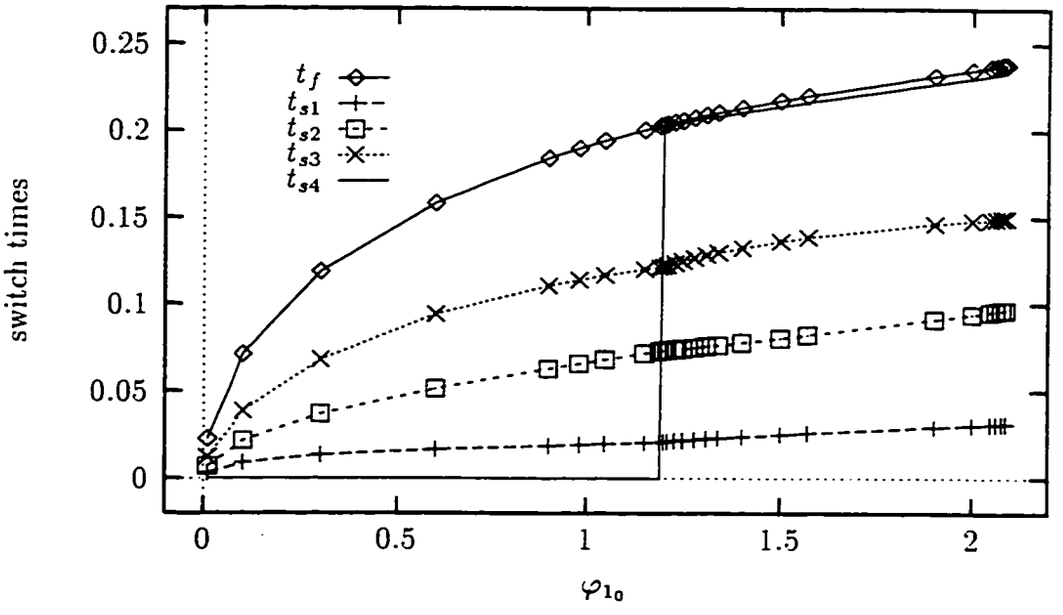

Figure 6.27: Optimal switch times t_{si} and final time t_f (Example Two).

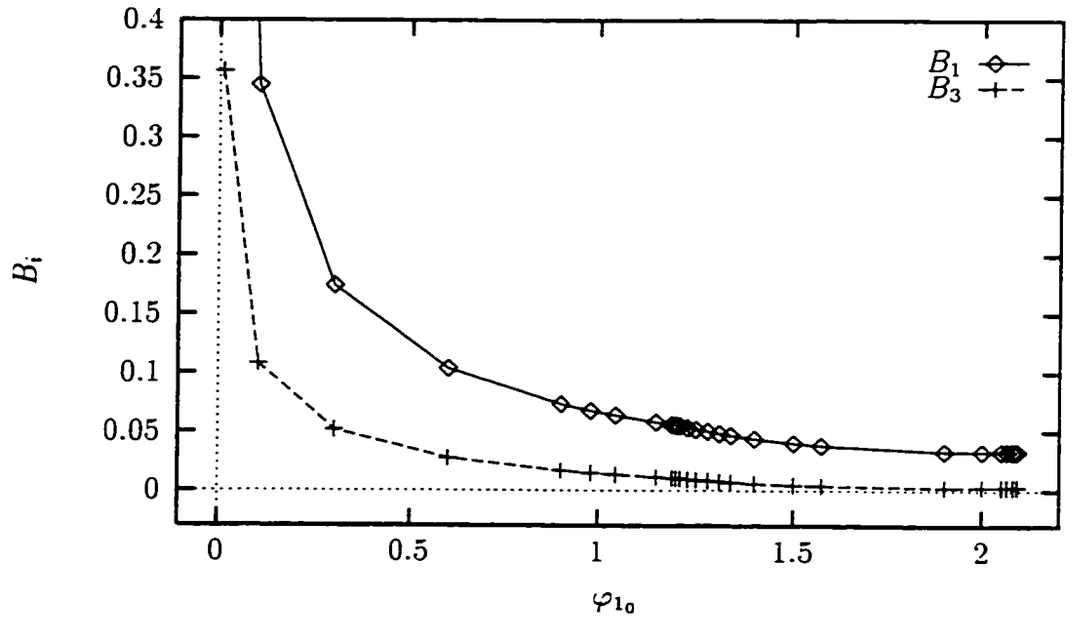

Figure 6.28: Optimal initial costates B_1, B_3 (Example Two).

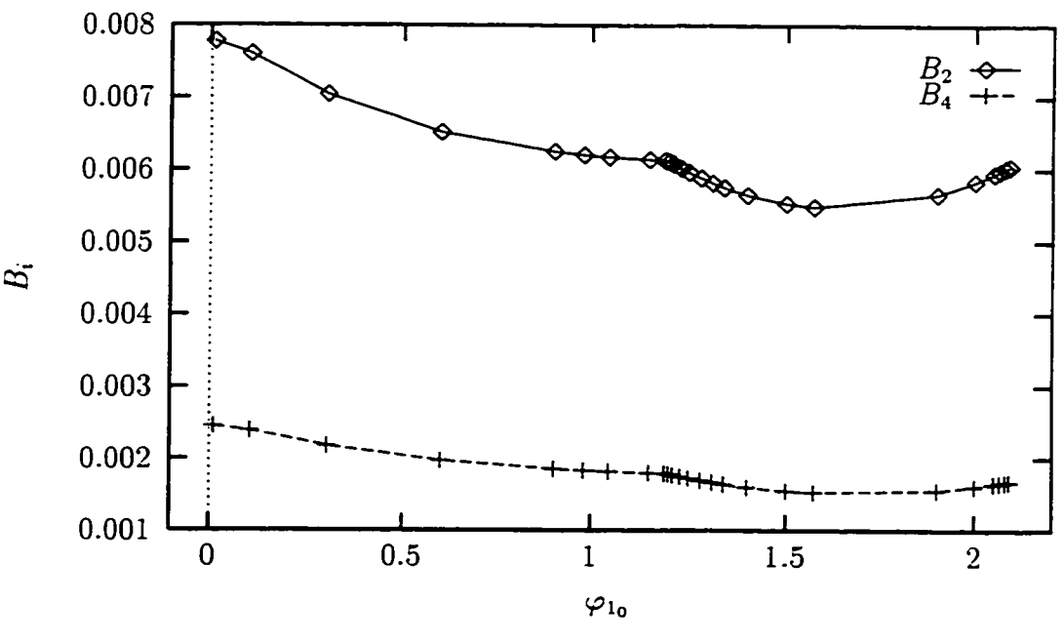

Figure 6.29: Optimal initial costates B_2, B_4 (Example Two).

6.2.2 Two-Link Manipulator from Example Three

This example is also a rest-to-rest motion of a two-link manipulator from straight-to-straight configurations ($\varphi_2(t_0) = 0, \varphi_2(t_f) = 0$). The physical parameters as in [29] and [40] are given in Equation (6.15). In [29] the time-optimal trajectories with three and four switches were reported; however, in [40] it was concluded that only the solutions with three switches were optimal. Here we show that, as in the previous case, the optimal solutions have three switches for smaller φ_{1_f} and four switches for larger φ_{1_f}. Here the results are presented for $0.76 \leq \varphi_{1_f} \leq 3.1415$ $[rad]$. It is expected that for $\varphi_{1_f} < 0.76$ the solutions will have always three switches. For this example, the only parameter which varies is the rotation of shoulder link φ_{1_f}. Again the optimal solutions have three or four switches. A typical time-optimal solution with three switches was presented in Section 6.1.3 for $\varphi_{1_f} = 55.9^0 = 0.975$ $[rad]$ (see Figure 6.20).

The following is an example of time-optimal control when the number of switch times is four. The initial and the final conditions of the states in $[rad]$ and $[rad/s]$ are:

$$\begin{aligned} x(t_0) &= \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \\ x(t_f) &= \begin{bmatrix} 3.1415 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \end{aligned} \tag{6.18}$$

Figure 6.30 shows the trajectory of the optimal control solution which has four switches, ($t_{s1} = 0.061168$, $t_{s2} = 0.634483$, $t_{s3} = 0.817584$, $t_{s4} = 1.242685$), and the final time $t_f = 1.512833$ $sec.$ The sequence for controls and the scaled switch functions ($250\ G_i$) of the time-optimal control of the manipulators are shown in Figure 6.31. Figure 6.32 shows the optimal states for Example Three.

For the TLM in this example, the calculations again were carried out for φ_{1_f} varying from 0.76 to 3.1415. The optimal control solution has three or four switches as reported in Table 6.6. Figure 6.33 indicates the number and the locations of the switches as well as the final time of the bang-bang optimal control for these motions. The optimal initial costates are given in Table 6.7. Figure 6.34 and Figure 6.35 show

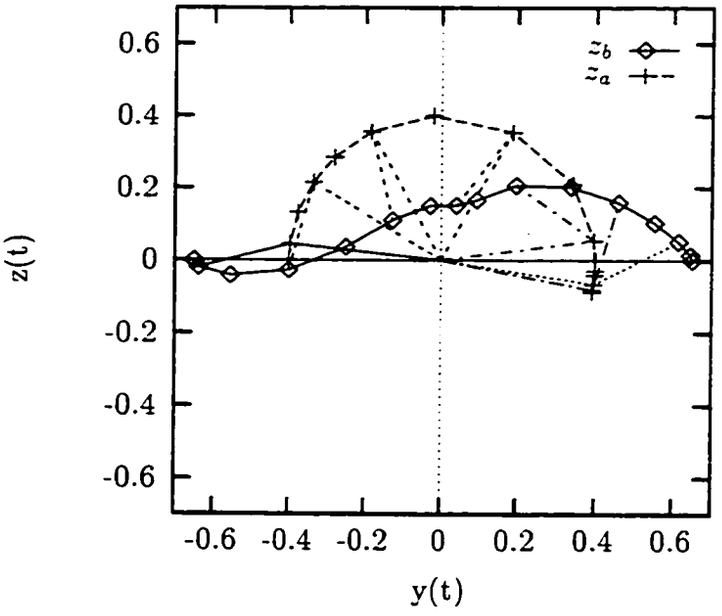

Figure 6.30: Trajectories of tip of the elbow link (y_b, z_b) and tip of the shoulder link (y_a, z_a) (Example Three).

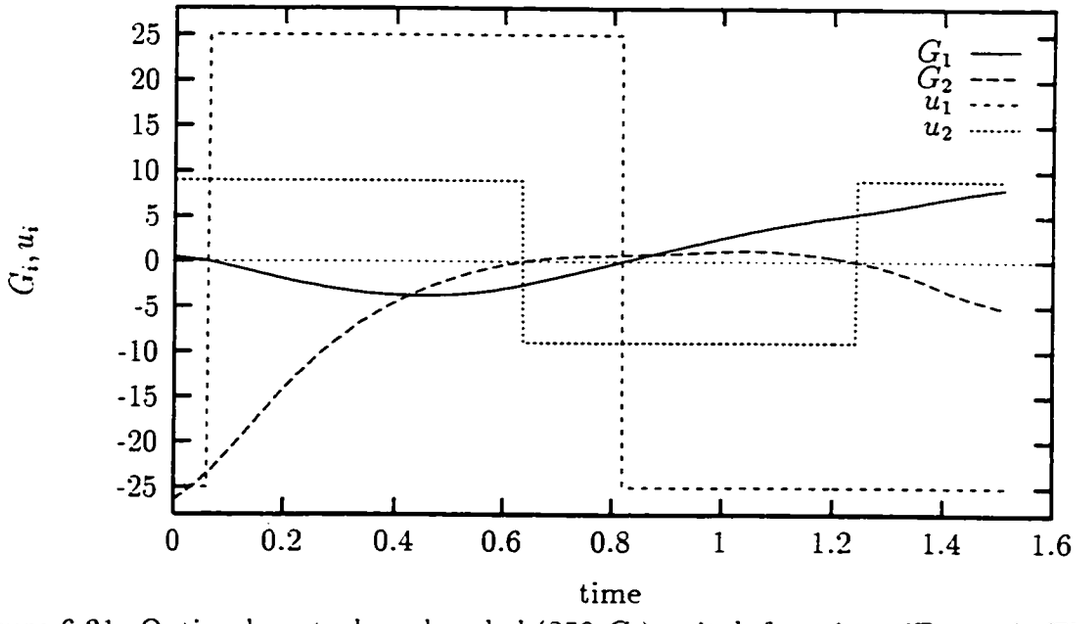

Figure 6.31: Optimal control, and scaled (250 G_i) switch functions (Example Three).

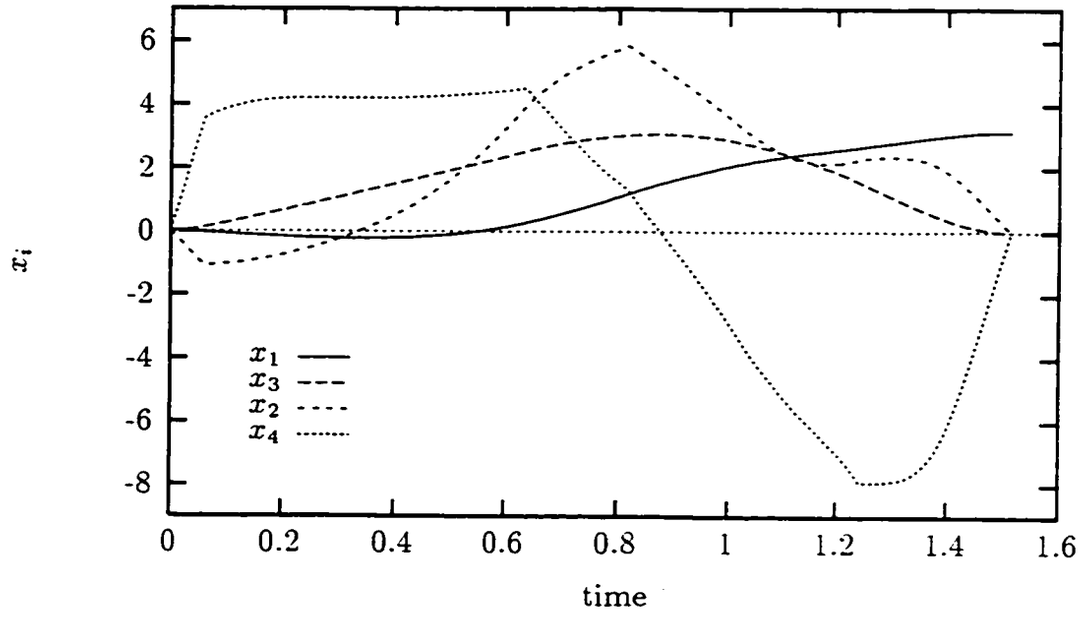

Figure 6.32: Optimal solution, x_i states, u_i controls (Example Three).

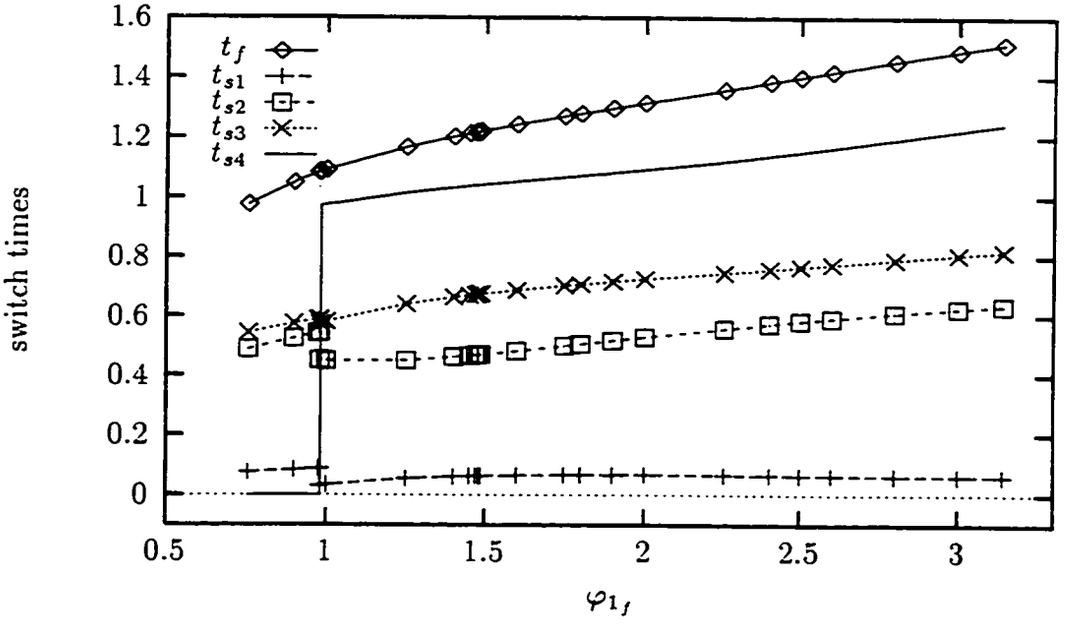

Figure 6.33: Optimal switch times t_{si} and final time t_f (Example Three).

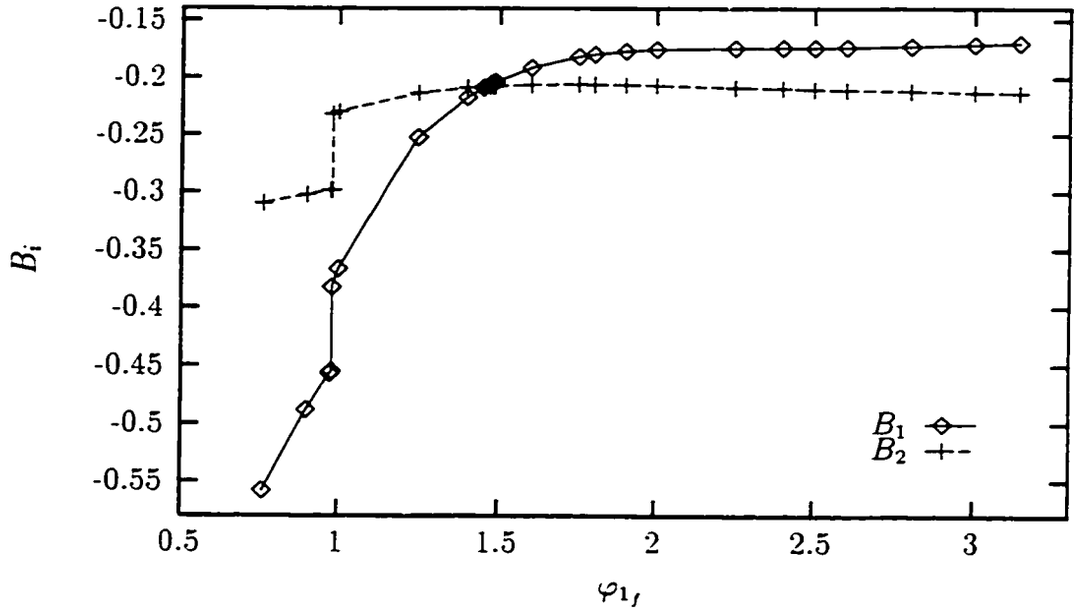

Figure 6.34: Optimal initial costates B_1, B_2 (Example Three).

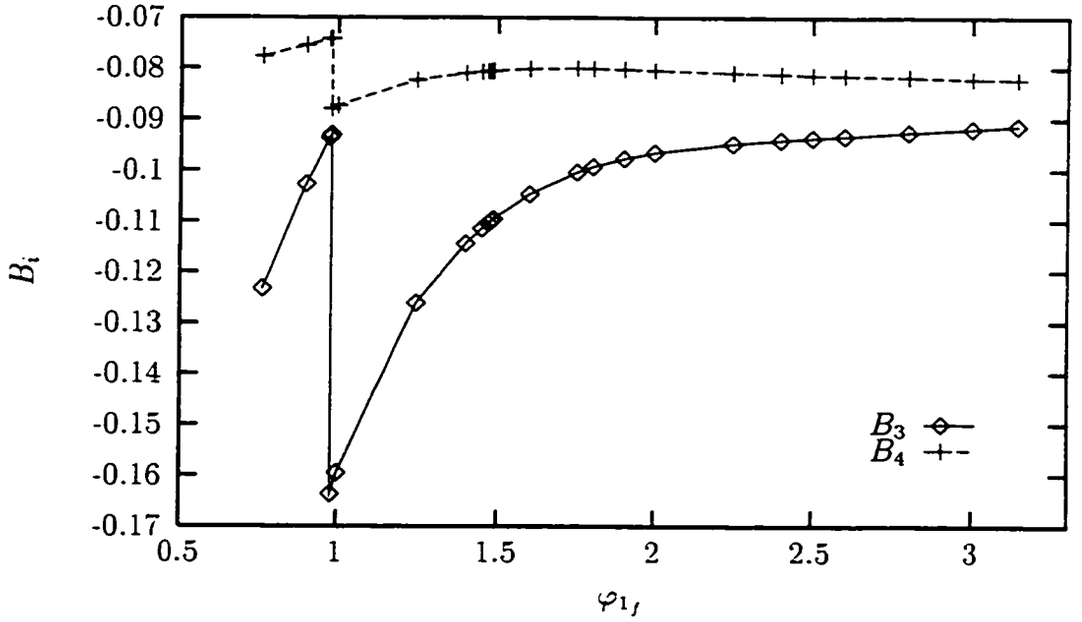

Figure 6.35: Optimal initial costates B_3, B_4 (Example Three).

Table 6.6: Converged switch times and final time for various φ_{1_f} (Example Three).

φ_{1_f}	t_f	t_{s1}	t_{s2}	t_{s3}	t_{s4}
0.76	0.974963	$7.571537e-2$	$4.874812e-1$	$5.426642e-1$	
0.9	1.047979	$8.346379e-2$	$5.239894e-1$	$5.739498e-1$	
0.975	1.083378	$8.751933e-2$	$5.416889e-1$	$5.872438e-1$	
0.98	1.085656	$8.778758e-2$	$5.428282e-1$	$5.880497e-1$	
0.98	1.084077	$2.946801e-2$	$4.507253e-1$	$5.715065e-1$	0.970975
1.0	1.091563	$3.325808e-2$	$4.479418e-1$	$5.790395e-1$	0.974051
1.25	1.166824	$5.507388e-2$	$4.493728e-1$	$6.384861e-1$	1.013591
1.4	1.201874	$6.106139e-2$	$4.612327e-1$	$6.619982e-1$	1.031755
1.45	1.212537	$6.238261e-2$	$4.658690e-1$	$6.686512e-1$	1.037215
1.47	1.216692	$6.284398e-2$	$4.678144e-1$	$6.711898e-1$	1.039333
1.48	1.218747	$6.306144e-2$	$4.688051e-1$	$6.724350e-1$	1.040380
1.485	1.219770	$6.316696e-2$	$4.693047e-1$	$6.730518e-1$	1.040900
1.486	1.219974	$6.318782e-2$	$4.694050e-1$	$6.731747e-1$	1.041004
1.4865	1.220076	$6.319821e-2$	$4.694551e-1$	$6.732361e-1$	1.041056
1.6	1.242467	$6.507204e-2$	$4.814797e-1$	$6.863057e-1$	1.052456
1.75	1.270386	$6.638350e-2$	$4.987245e-1$	$7.015764e-1$	1.066996
1.8	1.279425	$6.660462e-2$	$5.046588e-1$	$7.063171e-1$	1.071866
1.9	1.297261	$6.680875e-2$	$5.166141e-1$	$7.154392e-1$	1.081809
2.0	1.314894	$6.676847e-2$	$5.285101e-1$	$7.242152e-1$	1.092145
2.25	1.358624	$6.600738e-2$	$5.569972e-1$	$7.453195e-1$	1.120216
2.4	1.384789	$6.521420e-2$	$5.725402e-1$	$7.576089e-1$	1.138713
2.5	1.402213	$6.467714e-2$	$5.823972e-1$	$7.657838e-1$	1.151630
2.6	1.419613	$6.403436e-2$	$5.914844e-1$	$7.738407e-1$	1.165029
2.8	1.454282	$6.291964e-2$	$6.087622e-1$	$7.900607e-1$	1.192825
3.0	1.488693	$6.195136e-2$	$6.246889e-1$	$8.062979e-1$	1.221707
3.1415	1.512833	$6.116775e-2$	$6.344832e-1$	$8.175843e-1$	1.242685

the initial costates $B_i = p_i(t_0)$ versus φ_{1_f}. As can be seen, the number of the switches is three if $\varphi_{1_f} \leq 0.98$, and four if $\varphi_{1_f} \geq 0.98$.

Table 6.7: Converged initial costates for various φ_{1_f} (Example Three).

φ_{1_f}	B_1	B_2	B_3	B_4
0.76	$-5.579808e-1$	$-3.098280e-1$	$-1.232082e-1$	$-7.768285e-2$
0.9	$-4.879275e-1$	$-3.024319e-1$	$-1.026793e-1$	$-7.542207e-2$
0.975	$-4.566919e-1$	$-2.986221e-1$	$-9.354806e-2$	$-7.425754e-2$
0.98	$-4.547446e-1$	$-2.983736e-1$	$-9.298028e-2$	$-7.418156e-2$
0.98	$-3.825371e-1$	$-2.321680e-1$	$-1.636708e-1$	$-8.799163e-2$
1.0	$-3.664889e-1$	$-2.300996e-1$	$-1.595359e-1$	$-8.735937e-2$
1.25	$-2.523828e-1$	$-2.137025e-1$	$-1.260441e-1$	$-8.231080e-2$
1.4	$-2.175666e-1$	$-2.092486e-1$	$-1.144341e-1$	$-8.098576e-2$
1.45	$-2.092150e-1$	$-2.082189e-1$	$-1.114584e-1$	$-8.067103e-2$
1.47	$-2.062550e-1$	$-2.078754e-1$	$-1.103778e-1$	$-8.056600e-2$
1.48	$-2.048512e-1$	$-2.077176e-1$	$-1.098597e-1$	$-8.051774e-2$
1.485	$-2.041678e-1$	$-2.076420e-1$	$-1.096061e-1$	$-8.049464e-2$
1.486	$-2.040326e-1$	$-2.076272e-1$	$-1.095558e-1$	$-8.049010e-2$
1.4865	$-2.039651e-1$	$-2.076198e-1$	$-1.095307e-1$	$-8.048784e-2$
1.6	$-1.914926e-1$	$-2.064584e-1$	$-1.046779e-1$	$-8.013287e-2$
1.75	$-1.818154e-1$	$-2.061424e-1$	$-1.003636e-1$	$-8.003625e-2$
1.8	$-1.798278e-1$	$-2.062494e-1$	$-9.932808e-2$	$-8.006898e-2$
1.9	$-1.771410e-1$	$-2.066682e-1$	$-9.770493e-2$	$-8.019700e-2$
2.0	$-1.756708e-1$	$-2.072619e-1$	$-9.653656e-2$	$-8.037851e-2$
2.25	$-1.745321e-1$	$-2.089963e-1$	$-9.477485e-2$	$-8.090867e-2$
2.4	$-1.742458e-1$	$-2.098603e-1$	$-9.406859e-2$	$-8.115854e-2$
2.5	$-1.740387e-1$	$-2.104567e-1$	$-9.368122e-2$	$-8.133900e-2$
2.6	$-1.736486e-1$	$-2.108799e-1$	$-9.328249e-2$	$-8.145342e-2$
2.8	$-1.726499e-1$	$-2.118907e-1$	$-9.257682e-2$	$-8.176465e-2$
3.0	$-1.712244e-1$	$-2.129118e-1$	$-9.188735e-2$	$-8.209856e-2$
3.1415	$-1.698736e-1$	$-2.133731e-1$	$-9.133232e-2$	$-8.223627e-2$

6.3 Effects of Flexibility of the Manipulator

In this section maneuvers of Two-Link Flexible Manipulators (TLFM) are discussed. The following strategy is used. First, the control forces, using Pontryagin's Minimum Principle are obtained assuming rigid links. Next, these forces are applied to the flexible manipulator and the response is simulated by a nonlinear finite element method. The effects of flexibility are measured in terms of the amount of vibrations generated and the distance from the tip of the manipulator to the target point at the end of the maneuver. A quantitative relationship between these effects and the slenderness of the links is obtained.

The dynamics of flexible manipulators can be simulated using the Finite Element Method (FEM). At present, numerical routines capable of including all the flexibility effects in the equations of motion and securing high accuracy of the time integration schemes are available in commercial software. Of course this integration can be carried out only if the forces applied to the manipulator are known. For example, in [23] time-optimal control of very flexible single link manipulators is analysed by using the FEM program ADINA. In [20], approximate optimal control of a flexible manipulator is solved combining the FEM and the recursive quadratic programming.

Here, the motion of flexible links is simulated by ANSYS [55]. This FEM software package solves problems involving nonlinear equations of motion by direct integration. The forces required to drive the manipulator to the target are obtained using the methods described in Chapter 5. Structural damping was not considered for the flexible manipulator. Total number of iterations is set to 20000 with maximum time step of 0.0015 sec. The maximum time step was chosen around $T/20$, where T is the period of the dominant mode of vibration. The numerical damping is set to 0.005. Details of the elements used here and the FEM can be found in [55].

6.3.1 Dynamics of Flexible Manipulators

The equations of motion for flexible manipulators can be derived using the Lagrange equations in which the functional of energy includes the potential and kinetic energies, and the strain energy of the links. However, the problem of specifying an appropriate set of Degrees Of Freedom (DOF) describing such systems is more challenging [56], if one is to follow the derivation presented in Chapter 4. This problem is solved conveniently in the FEM formulation in which DOF are the displacements and rotations of the assumed nodal points. The finite element formulation is used here, in which the equations of motion are written in the form:

$$M(\varphi)\ddot{\varphi}(t) + C(\varphi)\dot{\varphi} + K(\varphi)\varphi(t) = u(t) \tag{6.19}$$

where $M(\varphi)$, $C(\varphi)$, and $K(\varphi)$ are nonlinear mass, damping, and stiffness matrices. The vector φ represents Degrees Of Freedom at each nodal point (two components of displacement and rotation for a 2-Dimensional beam element are shown in Figure 6.36). The vector of control forces $u(t)$ is obtained from time-optimal solutions for rigid manipulators. Here an ANSYS program in which the motion is considered in the (Y, Z) coordinate system is used. The matrices M, C, and K which define the inertia, damping and stiffness properties of the element IJ in this coordinate system, change as the element changes its orientation during the maneuver.

6.3.2 Modelling of Flexible Manipulators

The two-dimensional elastic beam element (BEAM3) of the ANSYS element library has been used to model the two links of the flexible manipulator. This element uses the standard approximation of the displacements of any point of the element in the form of:

$$\left\{ \begin{array}{l} v_s = \alpha_1 + \alpha_2 s \\ v_z = \alpha_3 + \alpha_4 s + \alpha_5 s^2 + \alpha_6 s^3 \end{array} \right\} \tag{6.20}$$

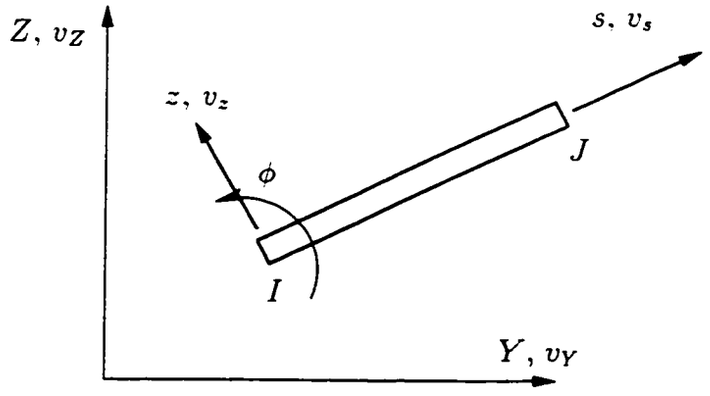

Figure 6.36: Beam element (BEAM3).

where v_s and v_z are displacements components in the local coordinate system (s, y) shown in Figure 6.36, and α_i are constants. The constants α_i are related to six DOF describing three components of displacement (v_s, v_z, ϕ) at each node. Ten such elements in each link have been used. In total, twenty elements of type BEAM3 modelled the links of the TLM. The structural mass element (MASS21) was used to include the lumped mass at the tip of each links and at the hub. This element has one node and three DOF. Two such elements (MASS21), one at the tip of the shoulder and one at the tip of the elbow link, have been used to include m_a and m_b. The joints were generated by coupling of the translational DOF at the corresponding nodal points. The driving moments were applied at these joints.

Two cases will be examined here to show the effects of flexibility of the manipulators under the optimal control obtained from the solution for rigid links. The first case (Example Four) is a rest-to-rest motion where its optimal solution has three switch times. The second case (Example Five) is also a rest-to-rest motion but its optimal solution has four switch times. For comparing response of the rigid links with flexible ones, structural damping was not considered for the flexible manipulator.

6.3.3 Example Four: Straight-to-Straight Configurations

This example is a rest-to-rest motion of the TLFM from straight-to-straight configurations (where for rigid links $\varphi_2(t_0) = 0$, $\varphi_2(t_f) = 0$). The optimal solution obtained for a rigid manipulator had three switch times. The physical parameters are the same as in [50] and have the following values:

$$\begin{array}{lll} l_1 = 2l_{c1} = 0.40\ [m] & U_1^{\mp} = \mp 0.25\ [Nm] & \\ l_2 = 2l_{c2} = 0.25\ [m] & U_2^{\mp} = \mp 0.10\ [Nm] & I_1 = 3.2688e-3\ [kg.m^2] \\ m_1 = 0.245\ [kg] & m_a = 0,\ m_b = 0.5\ [kg] & I_2 = 0.7986e-3\ [kg.m^2] \\ m_2 = 0.15315\ [kg] & I_o = I_a = I_b = 0 & g = 0\ [m.s^{-2}] \end{array} \tag{6.21}$$

These data had to be translated into the ANSYS input data format for BEAM3 element. For each link the cross sectional area of the beam A_i, the area moment of inertia J_i, and the height H_i of the cross section are required. Here $i = 1, 2$ indicate the link number. Assuming that the links are made of solid cylindrical rods with diameter d_i, and density ρ_i the relations for I_i and m_i are

$$I_i = \frac{m_i l_i^2}{12} + \frac{m_i d_i^2}{16} \qquad m_i = \rho_i l_i A_i = \rho_i l_i \pi d_i^2/4 \tag{6.22}$$

Using (6.21) and (6.22) the input data for element BEAM3 for this example is:

$$\begin{array}{lll} d_1 = 0.01\ m & A_1 = \pi d_1^2/4 = 7.854e-5\ m^2 & A_2 = A_1 \\ \rho_1 = 7800\ kg/m^3 & J_1 = \pi d_1^4/64 = 4.909e-10\ m^4 & J_2 = J_1 \\ & H_1 = d_1 = 0.01\ m & H_2 = H_1 \end{array} \tag{6.23}$$

A typical ANSYS input file for the TLFM can be found in Appendix B.

Here, slenderness of the links is indicated by the geometrical aspect ratio of each link defined as $ar_i = \frac{l_i}{d_i}$. These aspect ratios for the TLM analysed in Example Four

are:

$$ar_1 = \frac{l_1}{d_1} = \frac{0.4}{0.01} = 40 \qquad ar_2 = \frac{l_2}{d_2} = \frac{0.25}{0.01} = 25 \tag{6.24}$$

The initial and the final conditions of the states in $[rad]$ and $[rad/s]$ are:

$$\begin{aligned} x(0) &= \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \\ x(t_f) &= \begin{bmatrix} 0.376 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \end{aligned} \tag{6.25}$$

Rigid Links

Using the numerical approach described in Chapter 5, the optimal control solution has been obtained. It has three switches, ($t_{s1} = 0.039623$, $t_{s2} = 0.616536$, $t_{s3} = 0.664537$), and the final time $t_f = 1.233072$, where all time are given in seconds.

Figure 6.37 shows the scaled switch functions (0.1 G_i) of time-optimal control of

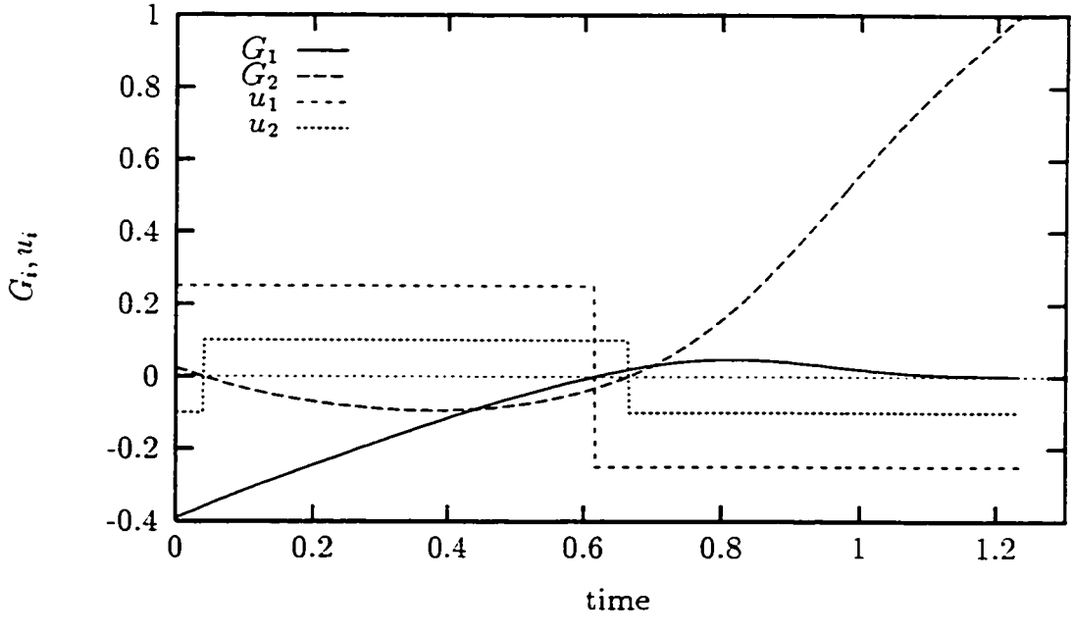

Figure 6.37: Optimal control, and scaled (0.1 G_i) switch functions of rigid links (Example Four).

the manipulator of rigid links. The moments $u_1(t)$ and $u_2(t)$ as given in Figure 6.37 are applied to the shoulder and elbow joints of the flexible manipulator.

Flexible Links

For the ANSYS analysis the links are specified by data given in Equation (6.23). Flexibility of the TLM can be numerically manipulated by assuming different values of the modulus of elasticity E of the material. Here, three moduli of elasticity to represent Steel ($E = 200\ GPa$), Aluminium 6061-T6 ($E = 69\ GPa$), Magnesium Alloy AM 100A ($E = 44.8\ GPa$) have been used.

In the graphs showing the results, the symbols $.r$ and $.fA$ are used meaning rigid and flexible links (made from Aluminium), respectively. Figure 6.38 compares the planar trajectories of motion of tip of the elbow link for the rigid and the flexible manipulator. For flexible manipulators angles of rotation of nodal points along each link vary to account for their deformability.

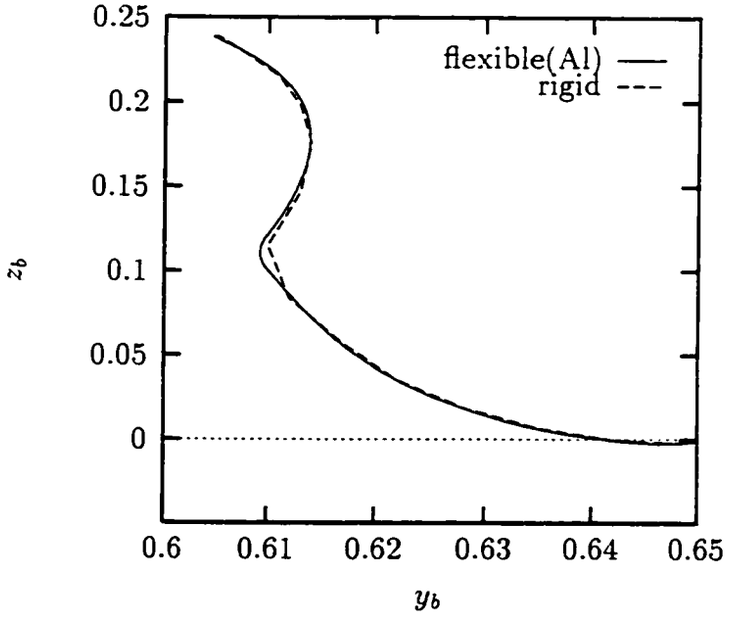

Figure 6.38: Planar motion of elbow's tip (rigid & flexible (Al), Example Four).

Here only the rotation and angular velocities of the tip of shoulder link and the tip of elbow link are shown and compared with the rotation and angular velocity of the corresponding rigid links. These rotations are indicated in Figure 6.39 as φ_1^{tip} for the shoulder link ($\dot{\varphi}_1^{tip}$ for velocity) and as φ_2^{tip} for the elbow link ($\dot{\varphi}_2^{tip}$ for velocity). Figure 6.40 compares the rotations of the links for rigid manipulator ($x_1.r = \varphi_1$ and $x_3.r = \varphi_2$) and the rotations of tip of the flexible links ($x_1.fA = \varphi_1^{tip}$, $x_3.fA = \varphi_2^{tip}$).

It should be noted that the trajectories and the states presented in this section were obtained by the FEM. For rigid links these plots agree very closely with the results calculated with the help of the method given in Chapter 5. This supports correctness of all the derivatives used there. As can be seen the angles of rotation of the rigid TLM and the TLFM made of Aluminium are very close. Differences are more visible when plotting the angular velocities. Figure 6.41 compares the angular velocity of the rigid shoulder link ($x_2.r = \dot{\varphi}_1$) with the angular velocity of tip of the flexible shoulder link ($x_2.fA = \dot{\varphi}_1^{tip}$). Figure 6.42 compares the angular velocity for the rigid elbow link ($x_4.r = \dot{\varphi}_2$) with the angular velocity of tip of the flexible elbow link ($x_4.fA = \dot{\varphi}_2^{tip}$). These plots (Figure 6.41 and Figure 6.42) indicate some vibration

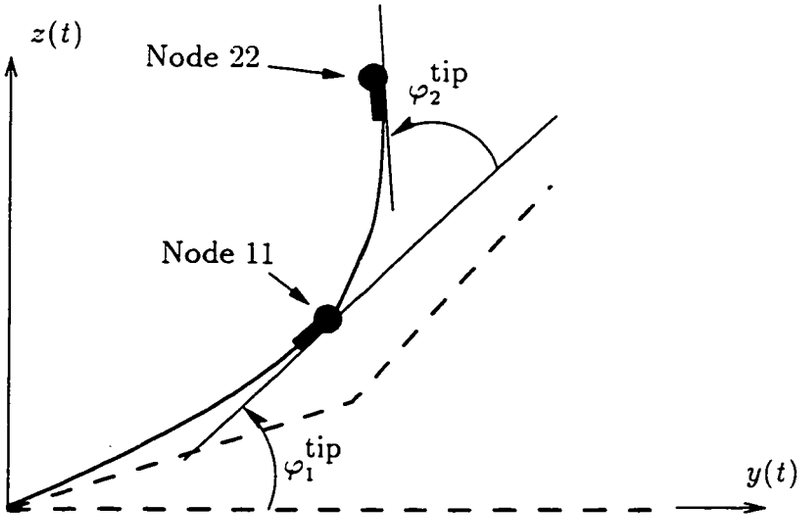

Figure 6.39: A two-link flexible manipulator.

of the links which, however, affect the trajectory of the tip of the manipulator very little.

The maneuver error due to the flexibility, e_a, is defined as the absolute difference (in meters) between the tip positions of the rigid and flexible manipulators.

$$e_a = \sqrt{(y_b^r - y_b^f)^2 + (z_b^r - z_b^f)^2} \tag{6.26}$$

The relative error is defined as $e_r = e_a/L$, where $L = l_1 + l_2$ is the total length of the two links. The dynamic responses of the flexible links for other materials to this control are very similar. The tip position and the corresponding errors are listed in Table 6.8. As can be seen from this Table that the tip position of the manipulator is almost identical for the rigid and the flexible manipulators, even with the least rigid material (Magnesium).

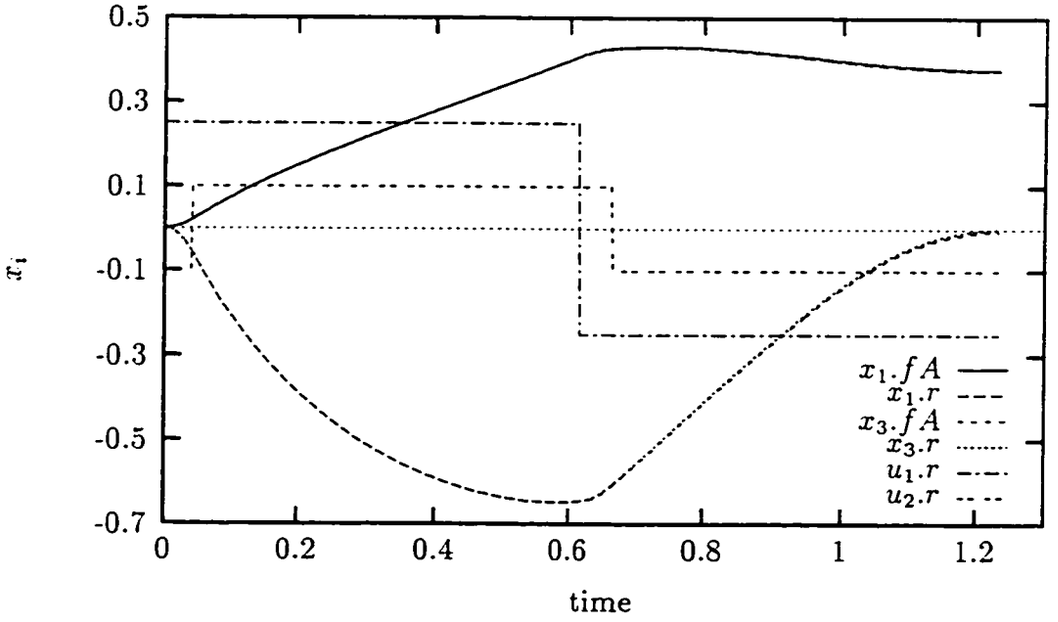

Figure 6.40: Rotations of links (rigid & flexible (Al), Example Four).

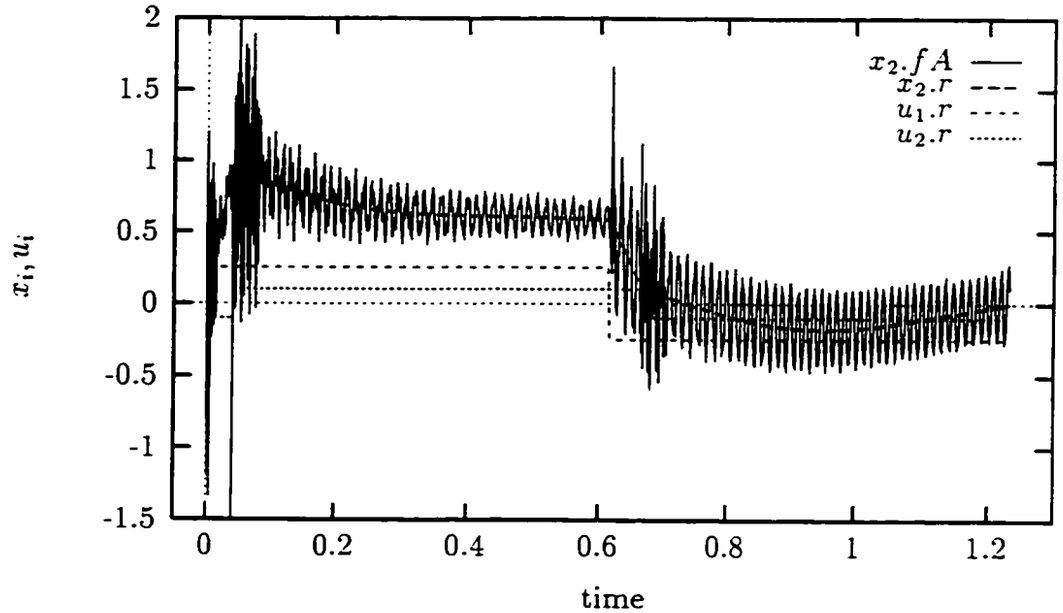

Figure 6.41: Angular velocity of shoulder's tip (rigid & flexible (Al), Example Four).

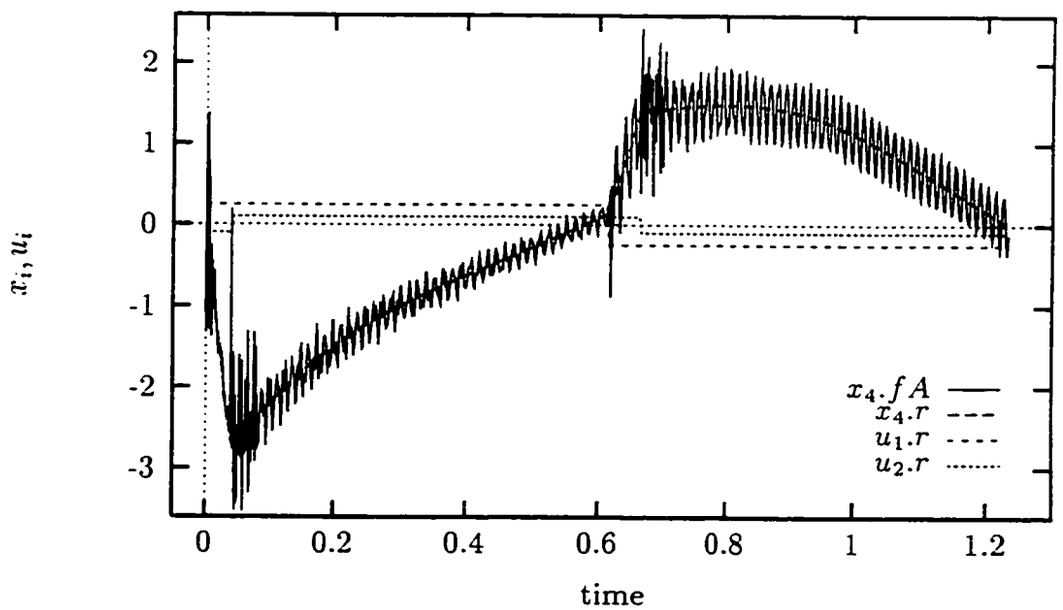

Figure 6.42: Angular velocity of elbow's tip (rigid & flexible (Al), Example Four).

Table 6.8: The tip position of elbow (y_b, z_b) (Example Four).

case	$E\ [GPa]$	*tip position*		*error*	
		y_b	z_b	e_a	e_r
rigid	-	0.604605	0.238649	0.0	0.0
Steel	200	0.604610	0.238632	1.772e-5	2.726e-5
Aluminium	69	0.604609	0.238633	1.649e-5	2.537e-5
Magnesium	44.8	0.604653	0.238512	1.452e-4	2.323e-4

6.3.4 Example Five: Straight-to-Broken Configurations

This example is a rest-to-rest motion of the TLFM from straight-to-broken configurations (where for rigid links $\varphi_2(t_0) = 0$, $\varphi_2(t_f) \neq 0$). The optimal solution obtained for rigid manipulator has four switch times. The physical parameters are the same as in [50] and have the following values:

$$\begin{array}{lll} l_1 = 2l_{c1} = 1.0\ [m] & U_1^{\mp} = \mp 3.90\ [Nm] & \\ l_2 = 2l_{c2} = 0.625\ [m] & U_2^{\mp} = \mp 1.56\ [Nm] & I_1 = 51.0547e-3\ [kg.m^2] \\ m_1 = 0.61261\ [kg] & m_a = 0,\ m_b = 1.25\ [kg] & I_2 = 12.4660e-3\ [kg.m^2] \\ m_2 = 0.38288\ [kg] & I_o = I_a = I_b = 0 & g = 0\ [m.s^{-2}] \end{array} \tag{6.27}$$

Again these data have to be translated into BEAM3 of ANSYS input data format. The input data for element BEAM3 of ANSYS for this example is the same as the data given by Equation (6.23). Here, however, the links are more slender. The geometrical aspect ratios for the TLM analysed in Example Five are:

$$ar_1 = \frac{l_1}{d_1} = \frac{1.0}{0.01} = 100 \qquad ar_2 = \frac{l_2}{d_2} = \frac{0.625}{0.01} = 62.5 \tag{6.28}$$

The initial and the final conditions of the states in $[rad]$ and $[rad/s]$ are:

$$\begin{aligned} x(0) &= \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \\ x(t_f) &= \begin{bmatrix} 0.8 & 0.0 & 0.2 & 0.0 \end{bmatrix}^T \end{aligned} \tag{6.29}$$

Rigid Links

Using the numerical approach described in Chapter 5, the time-optimal control solution has been obtained. It has four switches, (t_{s1} = 0.122441, t_{s2} = 0.708397, t_{s3} = 1.188230, t_{s4} = 1.518571), and the final time t_f = 1.620396, where all time are given in seconds. Figure 6.43 shows the scaled switch functions (0.1 G_i) of time-optimal control of the manipulator of rigid links. The moments $u_1(t)$ and $u_2(t)$ as given in Figure 6.43 are applied to the shoulder and elbow joints of the flexible manipulator.

Flexible Links

For the ANSYS analysis the links are specified by data given in Equation (6.23). Flexibility of the TLM again can be numerically manipulated by assuming different values of the modulus of elasticity E of the material. Here, two moduli of elasticity

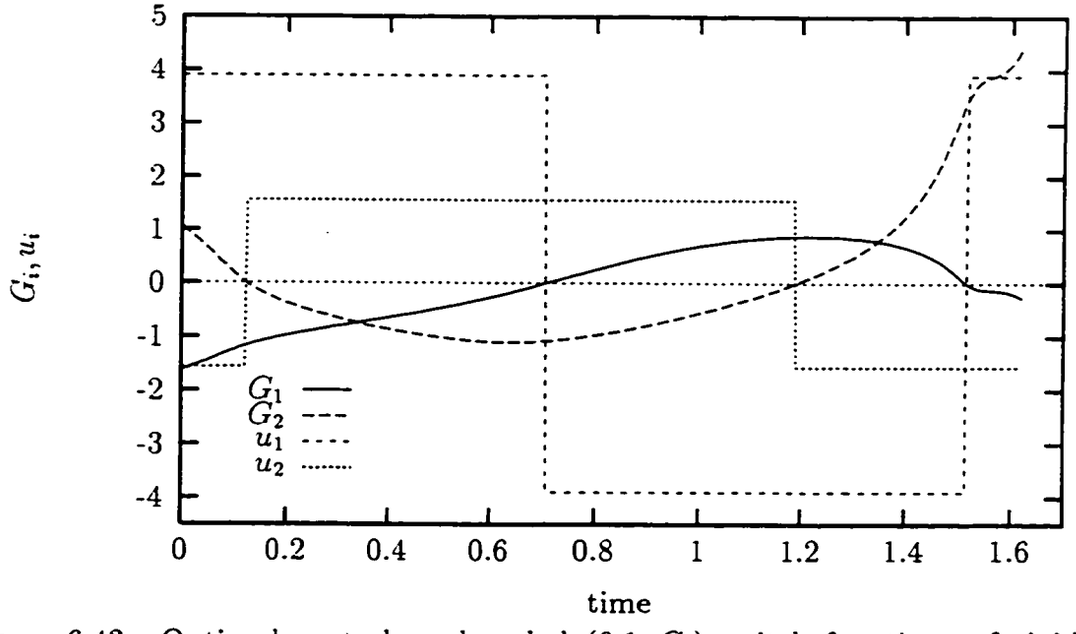

Figure 6.43: Optimal control, and scaled (0.1 G_i) switch functions of rigid links (Example Five).

to represent Steel ($E = 200\ GPa$) and Aluminium 6061-T6 ($E = 69\ GPa$) have been used.

In plots that follow the rotations and velocities of the flexible manipulator are denoted similar to that of Example Four. Figure 6.44 compares the planar trajectories of motion of tip of the elbow link for the rigid and the flexible manipulator. The symbols $.r$ and $.fS$ indicate rigid and flexible links made from Steel respectively. Figure 6.45 compares the rotations of the links for rigid manipulator ($x_1.r = \varphi_1$ and $x_3.r = \varphi_2$) and the rotations of tip of the flexible links ($x_1.fS = \varphi_1^{tip}$, $x_3.fS = \varphi_2^{tip}$). Figure 6.46 compares the angular velocity of the rigid shoulder link ($x_2.r = \dot{\varphi}_1$) with the angular velocity of tip of the flexible shoulder link ($x_2.fS = \dot{\varphi}_1^{tip}$). Figure 6.47 compares the angular velocity of the rigid elbow link ($x_4.r = \dot{\varphi}_2$) with the angular velocity of tip of the flexible elbow link ($x_4.fS = \dot{\varphi}_2^{tip}$). These plots (Figure 6.46 and Figure 6.47) indicate vibrations of the links. For this case, because of higher slenderness of the links (the geometrical aspect ratio), the vibrations are stronger

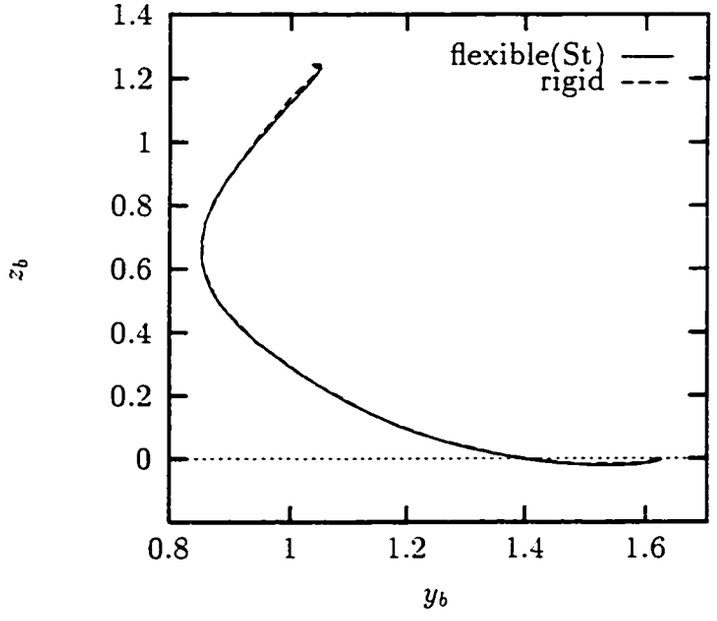

Figure 6.44: Planar motion of elbow's tip (rigid & flexible, Steel, Example Five).

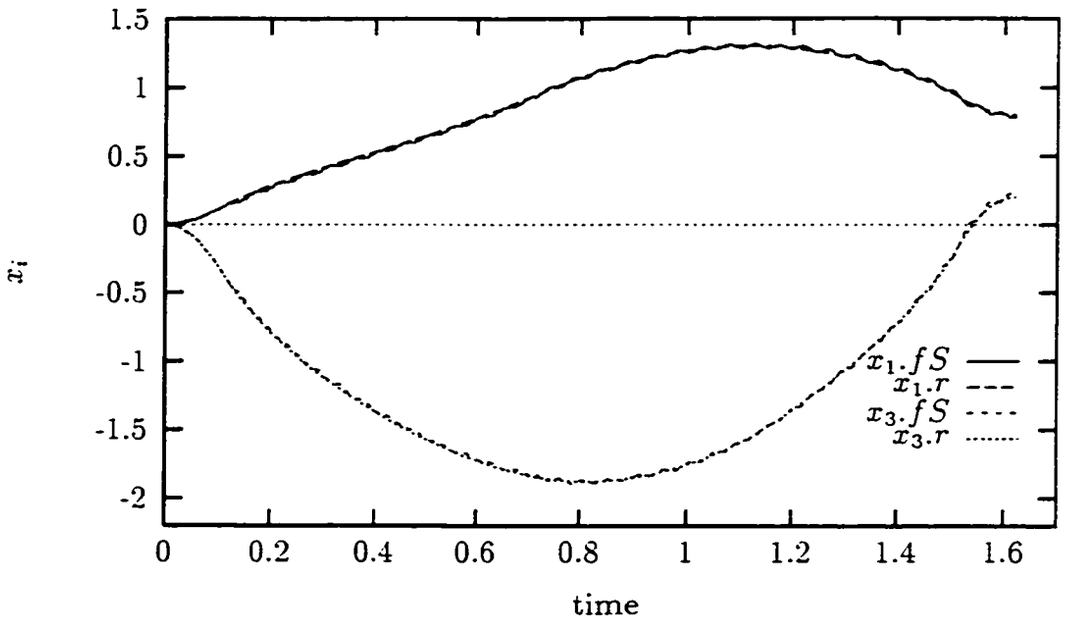

Figure 6.45: Rotations of links (rigid & flexible, Steel, Example Five).

than for Example Four. However, these vibrations still have very little affect on the trajectory of the tip of the manipulator. The dynamic responses of the flexible links for other materials to this control are very similar. The tip position and the corresponding errors are listed in Table 6.9. As can be seen from this Table the tip position of the manipulator is almost identical for the rigid and the flexible manipulators, even for Aluminium. The maneuver error, e_a, is defined in Equation (6.26).

Table 6.9: The tip position of elbow (y_b, z_b) (Example Five).

case	E $[GPa]$	*tip position*		*error*	
		y_b	z_b	e_a	e_r
rigid	–	1.03436	1.24321	0.0	0.0
Steel	200	1.03446	1.24135	1.863e-3	1.146e-3
Aluminium	69	1.03434	1.23884	4.370e-3	2.689e-3

In conclusion, Finite Element Method may be used to verify the performance of real (that is flexible) manipulators driven by the forces obtained by the time-optimal

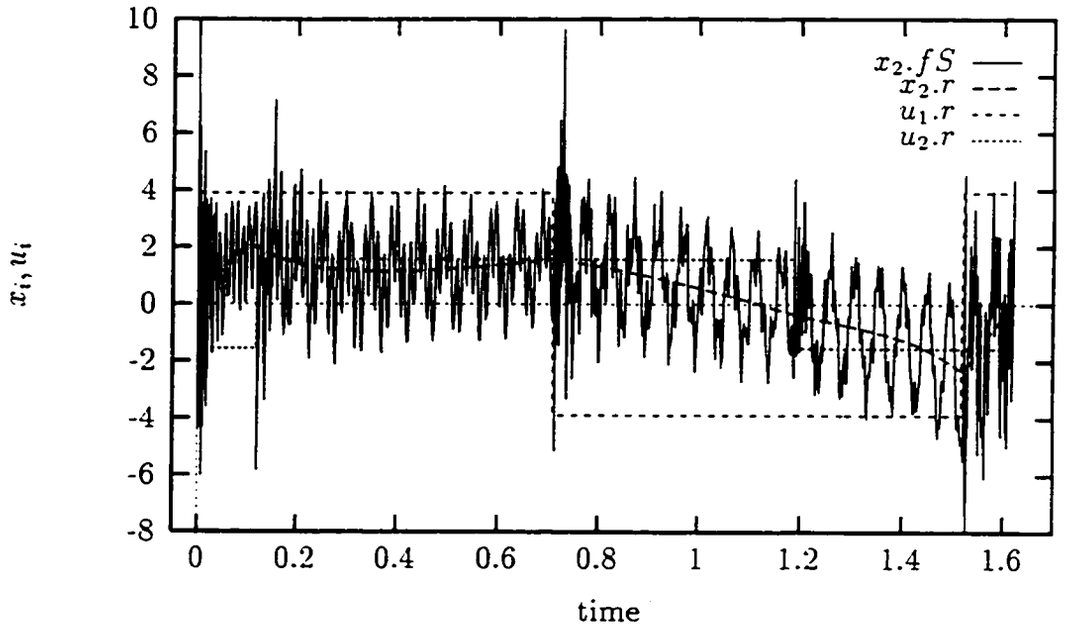

Figure 6.46: Angular velocity of shoulder's tip (rigid & flexible, Steel, Example Five).

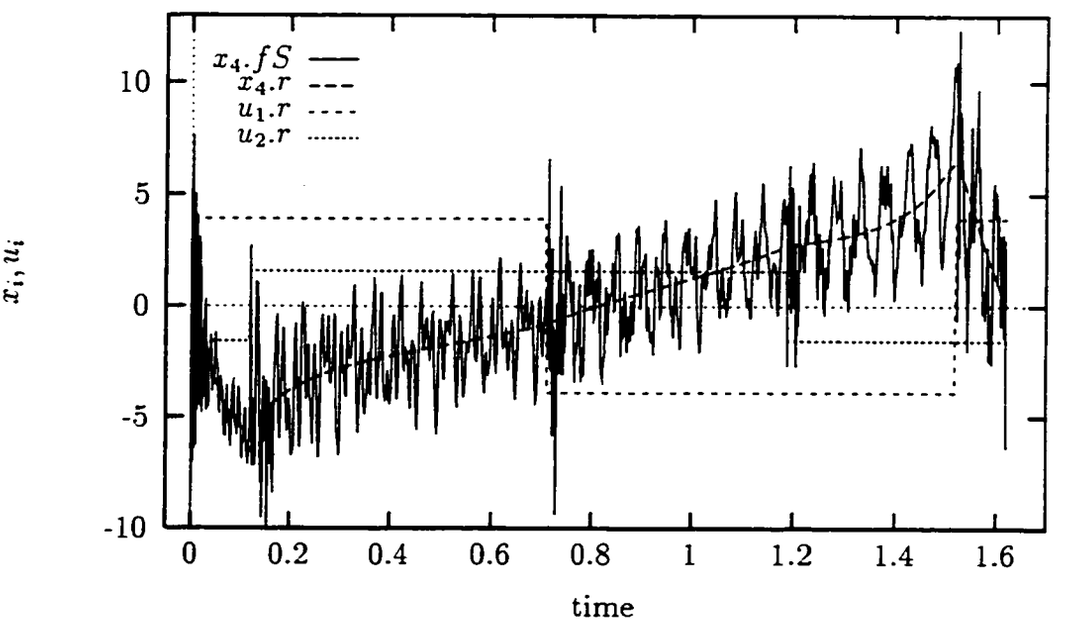

Figure 6.47: Angular velocity of elbow's tip (rigid & flexible, Steel, Example Five).

control of rigid manipulators. The results show that the vibrations due to flexibility have a greater effect on the response of the links than the motion of the tip of the manipulator. In general, the dynamic performance of the flexible manipulators under the controls obtained from the *rigid solution* is satisfactory if the slenderness of the links is sufficiently small. Particularly, with the geometrical aspect ratios, $ar_1 = 100$ and $ar_2 = 62.5$, and for the Steel links the performance is satisfactory.

6.4 Physical Parameters & Control Forces

Here the effects of physical parameters (such as geometry and masses of the links as well as masses at the tips of the links) and the effects of changing the limits of control forces on time-optimal maneuvers of the Two-Link Manipulator (TLM) are discussed. Effects of the length of links of TLM are shown in Example Six, with two different maneuvers. In that example the total length of the links is constant, as is the total mass of the links, but their ratio is varied. Example Seven shows the effects of different limits of control forces on time-optimal maneuvers of TLM. In that example the total magnitude of moments applied at the joints is constant, but their ratio is varied. Effects of the mass m_a at the tip of shoulder link and the mass m_b at the tip of the elbow are investigated in Example Eight, and effects of the gravity are investigated in Example Nine. The following ratios of the link lengths, the link inertias and the applied moments are defined as

$$R_L = \frac{l_1}{l_2} = \frac{m_1}{m_2} = R_M \qquad R_U = \frac{U_1}{U_2} \tag{6.30}$$

Other ratios related to R_L and R_U are also used.

$$R_I = \frac{I_1}{I_2} \qquad R_{UI} = \frac{R_U}{R_I} = \frac{U_1}{I_1} \Big/ \frac{U_2}{I_2} \tag{6.31}$$

6.4.1 Example Six: Effects of Length of the Links

This example is rest-to-rest motion of a two-link manipulator from straight-to-straight configurations with two different maneuver angles $\varphi_{1_f} = \varphi_1(t_f)$. The length ratio of the links R_L, defined in (6.30), is varied here. The physical parameters of this example are modified from Example Three of Section 6.1 and are given as follows:

$$\begin{array}{lll} l_1 = 2l_{c1} & U_1^{\mp} = \mp 25\ [Nm] & d_1 = d_2 = 0.1098723\ [m] \\ l_2 = 2l_{c2} & U_2^{\mp} = \mp 9\ [Nm] & I_1 = m_1(l_1^2/12 + d_1^2/16)\ [kg.m^2] \\ l_1 + l_2 = 0.65\ [m] & m_a = m_b = 0 & I_2 = m_2(l_2^2/12 + d_2^2/16)\ [kg.m^2] \\ m_1 + m_2 = 48.07\ [kg] & I_o = I_a = I_b = 0 & g_r = 0 \end{array} \tag{6.32}$$

where d_1 and d_2 are diameters of the cylindrical links. The total length of the links is constant, as is the total mass of the links; however, their ratio is varied.

The initial conditions of the states for both maneuvers are $x_i(t_0) = 0,\ i = 1, \ldots, 4$. The final conditions for the maneuver angles $\varphi_{1_f} = 0.76$ and $\varphi_{1_f} = 0.6$, respectively, are:

$$x(t_f) = [\ 0.76 \quad 0.0 \quad 0.0 \quad 0.0\ ]^T \tag{6.33}$$

$$x(t_f) = [\ 0.6 \quad 0.0 \quad 0.0 \quad 0.0\ ]^T \tag{6.34}$$

For $\varphi_{1_f} = 0.76$ the ratio R_L is varied from 1.0 to 1.60 and for $\varphi_{1_f} = 0.6$ the ratio R_L is varied from 1.60 to 2.0, while the $R_U = 2.77778$ is kept constant. The reason for choosing two different φ_{1_f} is to show the similarity and also the difference of the effects of the R_L on the minimum maneuver time w.r.t. φ_{1_f}.

Table 6.10 shows the change in the physical parameters of the manipulator when R_L changes. Table 6.11 shows the change in the optimal final time with the change in R_L. Since the cross sectional area of the links is kept constant, the variation of R_L will also change the values of R_I and R_{UI} shown in Table 6.11. Figure 6.48 shows

optimal switch times t_{si} and final time t_f with the length ratio R_L for $\varphi_{1_f} = 0.76$ and Figure 6.49 for $\varphi_{1_f} = 0.6$. Figure 6.50 shows optimal initial costates as functions of R_L for $\varphi_{1_f} = 0.76$ and Figure 6.51 for $\varphi_{1_f} = 0.6$. These two figures show the trend of changes in the initial costates as function of R_L and that the changes are smooth. Figure 6.52 shows optimal maneuver time t_f as function of R_L for $\varphi_{1_f} = 0.76$ and Figure 6.53 for optimal maneuver time t_f for $\varphi_{1_f} = 0.6$.

As can be observed, if R_L and R_I increase, the maneuver time decreases. It indicates that when designing for minimum maneuver time, the shoulder link should be as long as possible and the elbow link should be as short as possible when other parameters are constant. The decrease of t_f with R_L is sharper for $\varphi_{1_f} = 0.76$ than for $\varphi_{1_f} = 0.6$.

Table 6.10: Physical parameters of (Example Six, $R_U = 2.777778$).

R_L	l_1	l_2	m_1	m_2	I_1	I_2
1.00	.3250000	.3250000	24.03499	24.03499	.2296922	.2296922
1.05	.3329268	.3170732	24.62121	23.44877	.2459950	.2141446
1.10	.3404762	.3095238	25.17951	22.89047	.2622403	.2000225
1.20	.3545454	.2954546	26.21999	21.84999	.2944426	.1754324
1.30	.3673913	.2826087	27.16999	20.89999	.3261084	.1548718
1.40	.3791666	.2708334	28.04082	20.02916	.3571028	.1375413
1.50	.3900000	.2600000	28.84198	19.22799	.3873332	.1228252
1.60	.4000000	.2500000	29.58152	18.48846	.4167393	.1102435
1.65	.4047169	.2452831	29.93036	18.13962	.4311212	.1046319
1.70	.4092592	.2407408	30.26628	17.80370	.4452851	.0994189
1.75	.4136363	.2363637	30.58998	17.48000	.4592294	.0945693
1.80	.4178571	.2321429	30.90212	17.16785	.4729531	.0900515
1.85	.4219298	.2280702	31.20331	16.86666	.4864562	.0858372
1.90	.4258620	.2241380	31.49412	16.57586	.4997392	.0819010
2.00	.4333333	.2166667	32.04665	16.02333	.5256495	.0747734

Table 6.11: Optimal final time t_f as function of R_L (Example Six, $R_U = 2.777778$).

R_L	φ_{1_f}	R_I	R_{UI}	t_f
1.00	0.76	1.000	2.777778	0.8962093
1.05	0.76	1.149	2.418123	0.8890710
1.10	0.76	1.311	2.118736	0.8818791
1.20	0.76	1.678	1.655033	0.8682559
1.30	0.76	2.106	1.319192	0.8561131
1.40	0.76	2.596	1.069886	0.8454381
1.50	0.76	3.154	0.880846	0.8360224
1.60	0.76	3.780	0.734828	0.8276434
1.60	0.60	3.780	0.734829	0.7546849
1.65	0.60	4.120	0.674159	0.7518880
1.70	0.60	4.479	0.620195	0.7492062
1.75	0.60	4.856	0.572029	0.7466287
1.80	0.60	5.252	0.528896	0.7441462
1.85	0.60	5.667	0.490151	0.7417496
1.90	0.60	6.102	0.455243	0.7394319
2.00	0.60	7.030	0.395138	0.7350008

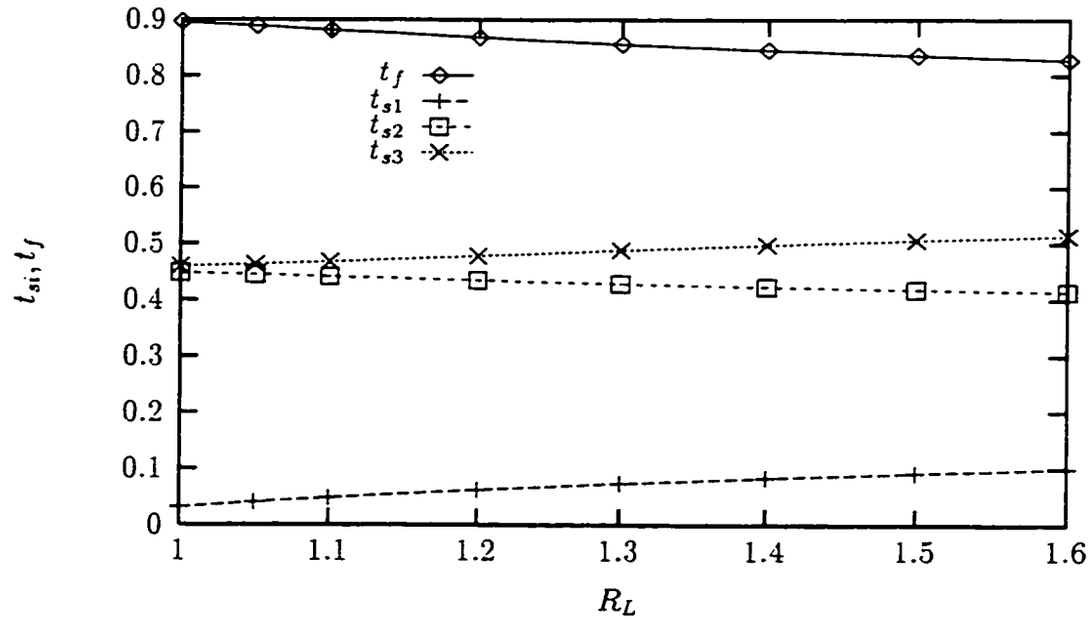

Figure 6.48: Optimal switch times t_{si} & t_f as function of R_L (Example Six, $\varphi_{1_f} = 0.76$)

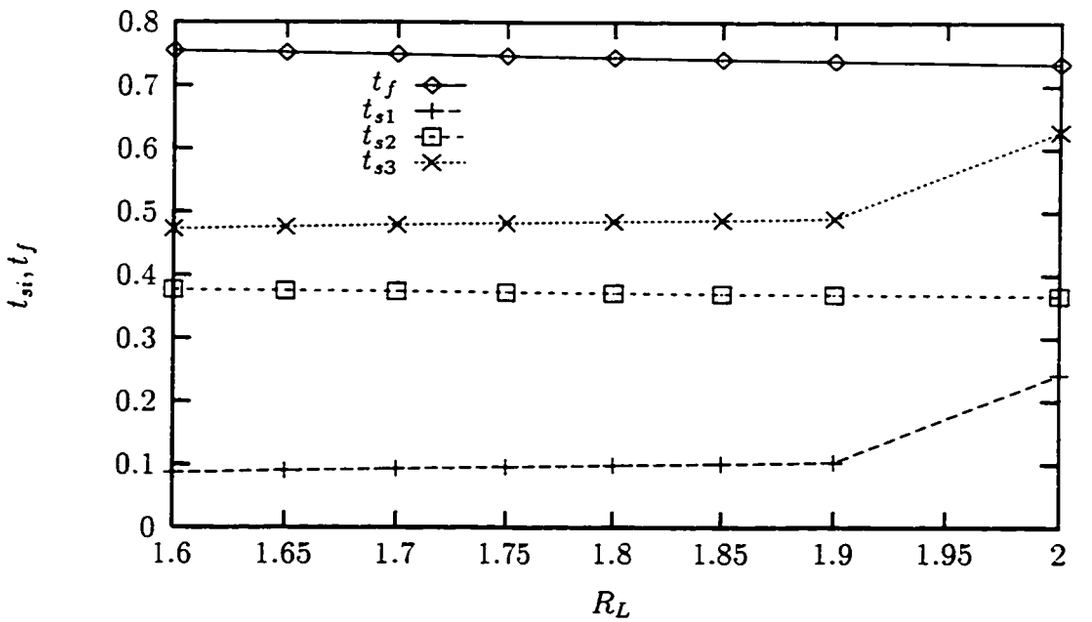

Figure 6.49: Optimal switch times t_{si} & t_f as function of R_L (Example Six, $\varphi_{1_f} = 0.6$).

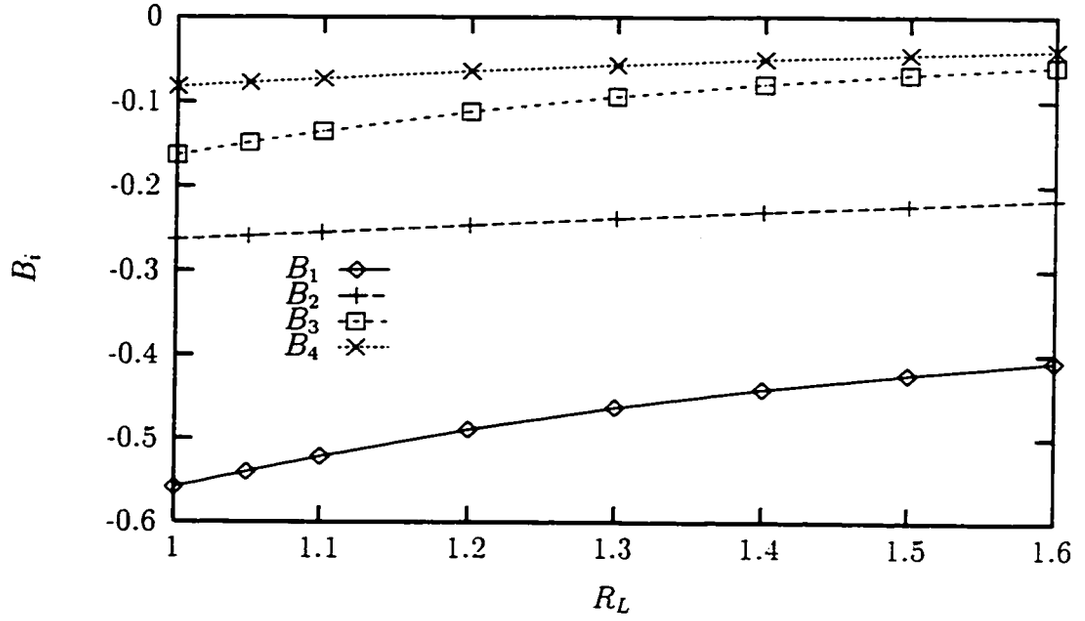

Figure 6.50: Optimal initial costates B_i as function of R_L (Example Six, $\varphi_{1_f} = 0.76$).

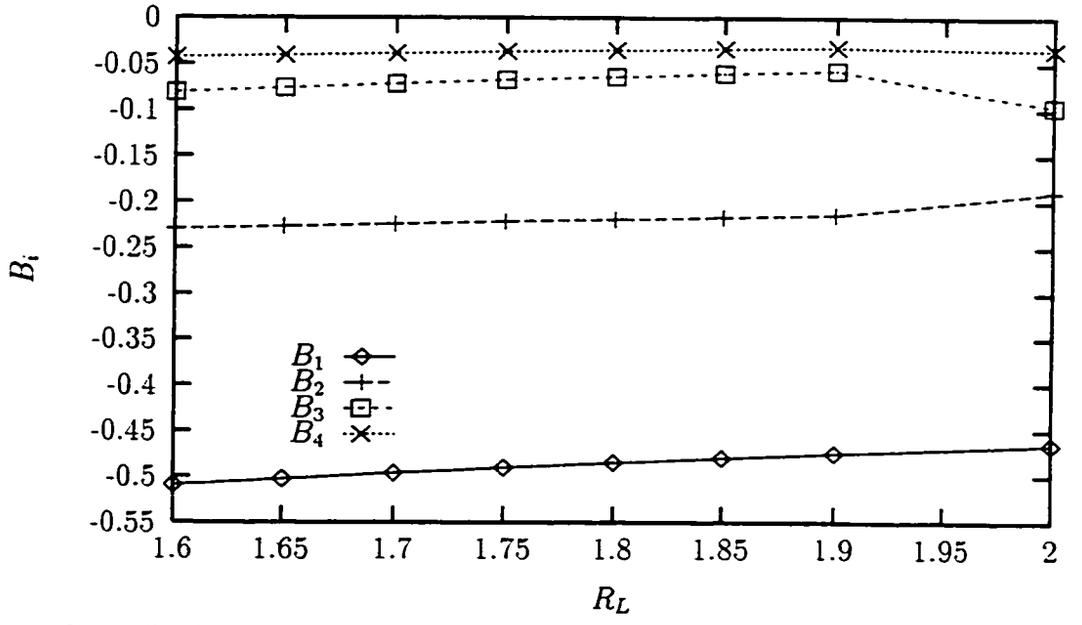

Figure 6.51: Optimal initial costates B_i as function of R_L (Example Six, $\varphi_{1_f} = 0.6$).

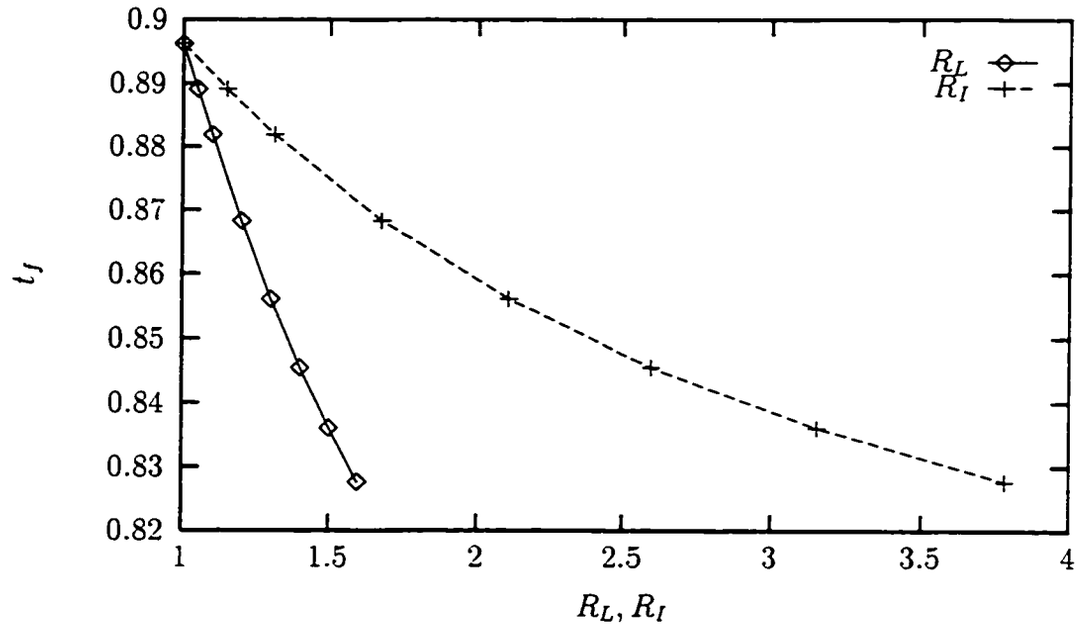

Figure 6.52: Optimal final time t_f as function of R_L (Example Six, $\varphi_{1_f} = 0.76$).

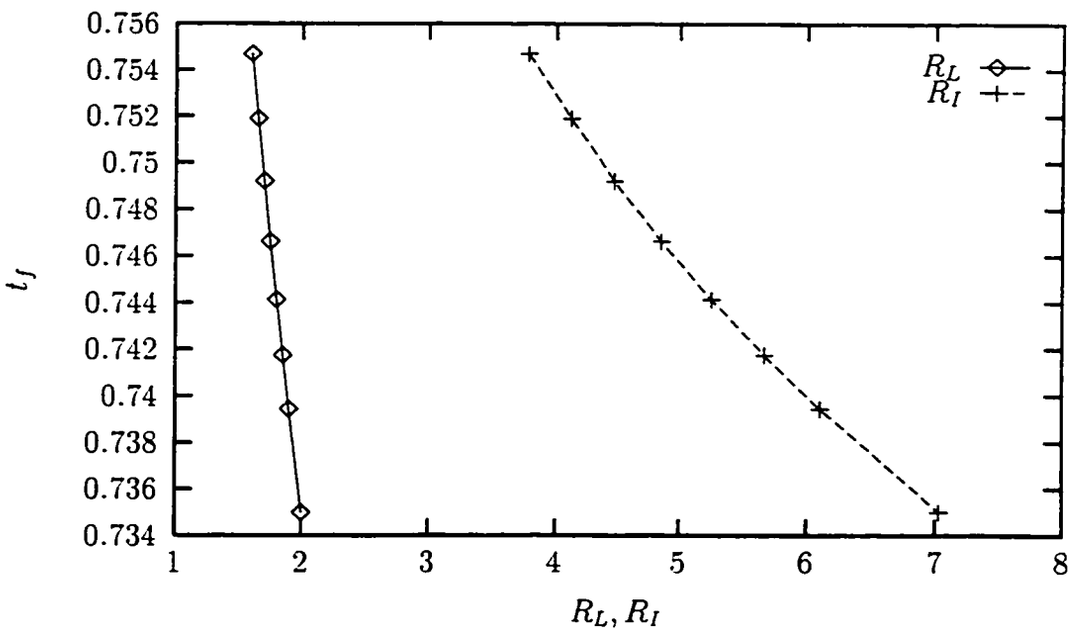

Figure 6.53: Optimal final time t_f as function of R_L (Example Six, $\varphi_{1_f} = 0.6$).

6.4.2 Example Seven: Effects of Control Forces

This example is one of the cases of Example Six with $R_L = 1.0$. In this example the total magnitudes of the control forces are constant, but their ratio R_U, defined in (6.30), is varied. The physical parameters of TLM are given in (6.35). This example is also a rest-to-rest motion of a two-link manipulator from straight-to-straight configurations. The initial and the final conditions of the states are the same as (6.33) and (6.34).

$$\begin{array}{lll} l_1 = 2l_{c1} = .325\ [m] & U_1^- + U_2^- = -34\ [Nm] & d_1 = d_2 = 0.1098723\ [m] \\ l_2 = 2l_{c2} = .325\ [m] & U_1^+ + U_2^+ = +34\ [Nm] & I_1 = 0.229692\ [kg.m^2] \\ m_1 = 24.035\ [kg] & m_a = m_b = 0 & I_2 = 0.229692\ [kg.m^2] \\ m_2 = 24.035\ [kg] & I_o = I_a = I_b = 0 & g_r = 0 \end{array} \tag{6.35}$$

Table 6.12 shows the effects of the change in the control forces applied to the manipulator (variation of R_U) on the optimal final time for $\varphi_{1_f} = 0.76$ and for $\varphi_{1_f} = 0.6$ The parameters U_1/I_1, U_2/I_2, and R_{UI} are also given there. Because $R_L = 1.0$ and consequently $I_1 = I_2$, then $R_{UI} = R_U$.

Table 6.12: Control forces and optimal t_f (Example Seven, $\varphi_{1_f} = 0.76, 0.6$, $R_L = 1.0$).

R_U	U_1	U_2	U_1/I_1	U_2/I_2	R_{UI}	$t_f(0.76)$	$t_f(0.6)$
0.78947	15.0	19.0	65.30477	82.71938	0.78947	0.8724345	0.8170388
0.88889	16.0	18.0	69.65842	78.36573	0.88889	0.8680668	0.8106437
1.00000	17.0	17.0	74.01208	74.01208	1.00000	0.8642263	0.8052761
1.26667	19.0	15.0	82.71938	65.30477	1.26667	0.8597692	0.7963664
1.42857	20.0	14.0	87.07303	60.95112	1.42857	0.8599207	0.7935635
1.61538	21.0	13.0	91.42668	56.59747	1.61538	0.8621044	0.7918445
1.83333	22.0	12.0	95.78033	52.24382	1.83333	0.8664952	0.7922263
2.09091	23.0	11.0	100.1340	47.89017	2.09091	0.8739482	0.7941729
2.40000	24.0	10.0	104.4876	43.53652	2.40000	0.8847785	0.7977509
2.77778	25.0	9.00	108.8413	39.18286	2.77778	0.8962094	0.8004655
3.00000	25.5	8.50	111.0181	37.00604	3.00000	0.8978919	0.7987488

Figure 6.54 shows optimal final time t_f as function of R_U for $\varphi_{1_f} = 0.76$ and $\varphi_{1_f} = 0.6$. For $\varphi_{1_f} = 0.76$, as R_U increases, final time decreases down to a minimum point $t_f = 0.8597692$ at $R_U = 1.27$. For $\varphi_{1_f} = 0.6$ the minimum point is $t_f = 0.7918445$ at $R_U = 1.62$. Figure 6.55 shows optimal final time t_f as function of U_1/I_1 and U_2/I_2 for $\varphi_{1_f} = 0.76$ and for $\varphi_{1_f} = 0.6$. From the practical point, these plots help to make the best choice of U_1 and U_2 for the manipulator design. In other words, it is practical to choose the best ratio of the control forces R_U such that the maneuver time is shortened further. For this example, the minimum final time corresponds to the point $R_U = 1.27$, $U_1/I_1 = 82.72$, $U_2/I_2 = 65.30$ for $\varphi_{1_f} = 0.76$, and $R_U = 1.62$, $U_1/I_1 = 91.43$, $U_2/I_2 = 56.6$ for $\varphi_{1_f} = 0.6$. This figure has another practical use that would allow to choose the appropriate control forces in relation to the mass moment of inertia of the links. Figure 6.56 shows optimal switch times t_{si} and final time t_f for various combination of control forces R_U for maneuver $\varphi_{1_f} = 0.76$. Figure 6.57 is the plot of optimal switch times t_{si} and final time t_f as function of R_U for $\varphi_{1_f} = 0.6$. Figure 6.58 and Figure 6.59 show optimal initial costates as functions of R_U for $\varphi_{1_f} = 0.76$ and $\varphi_{1_f} = 0.6$ respectively.

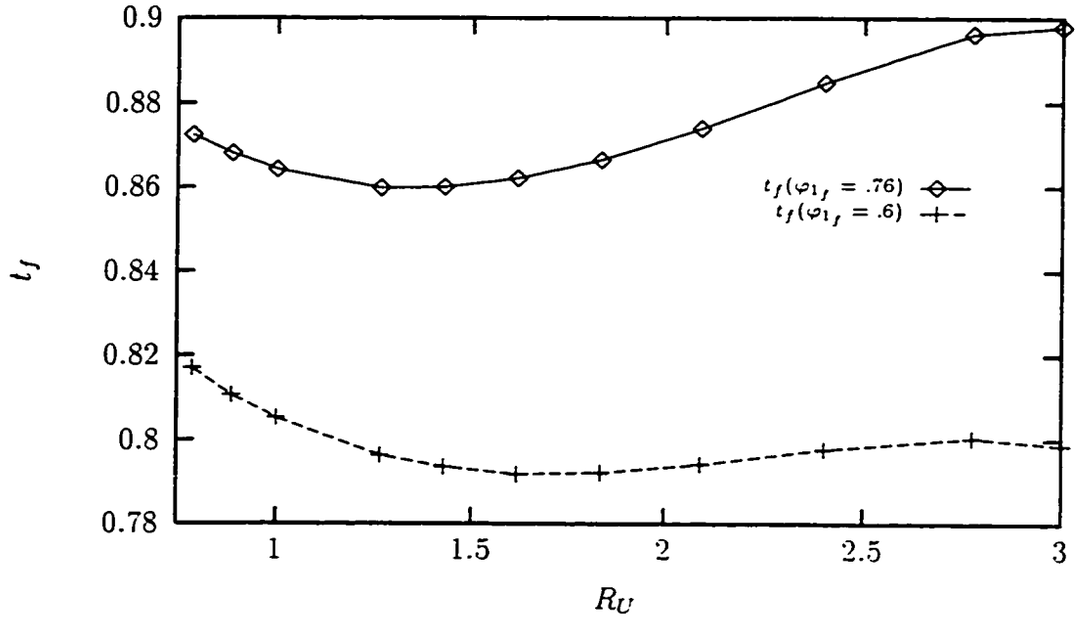

Figure 6.54: Optimal final time t_f as function of R_U (Example Seven, $\varphi_{1_f} = 0.76, 0.6$).

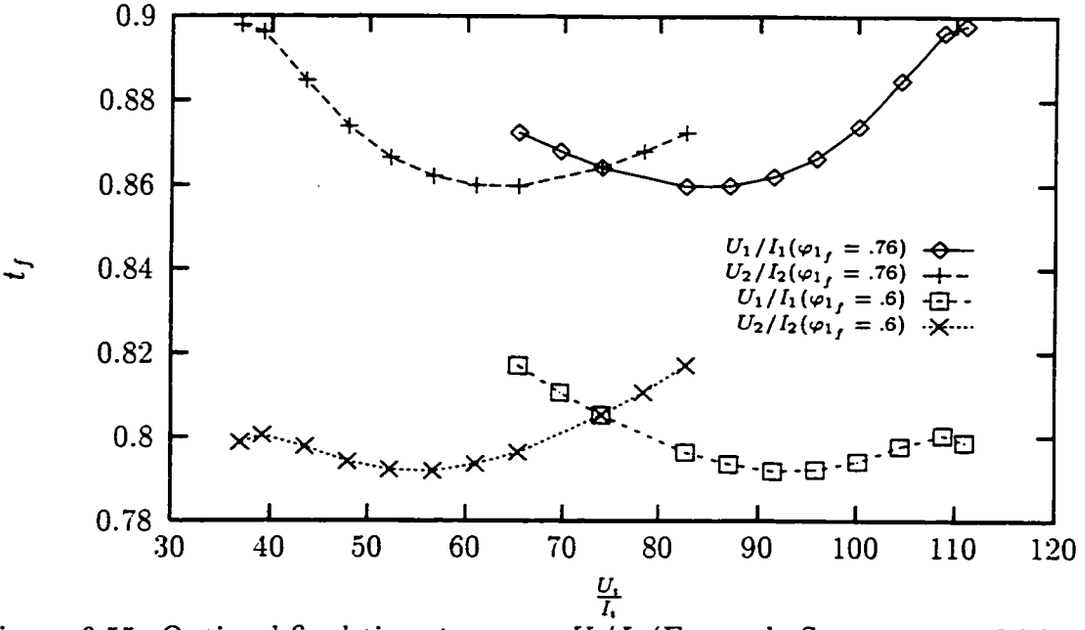

Figure 6.55: Optimal final time t_f versus U_i/I_i (Example Seven, $\varphi_{1_f} = 0.76, 0.6$).

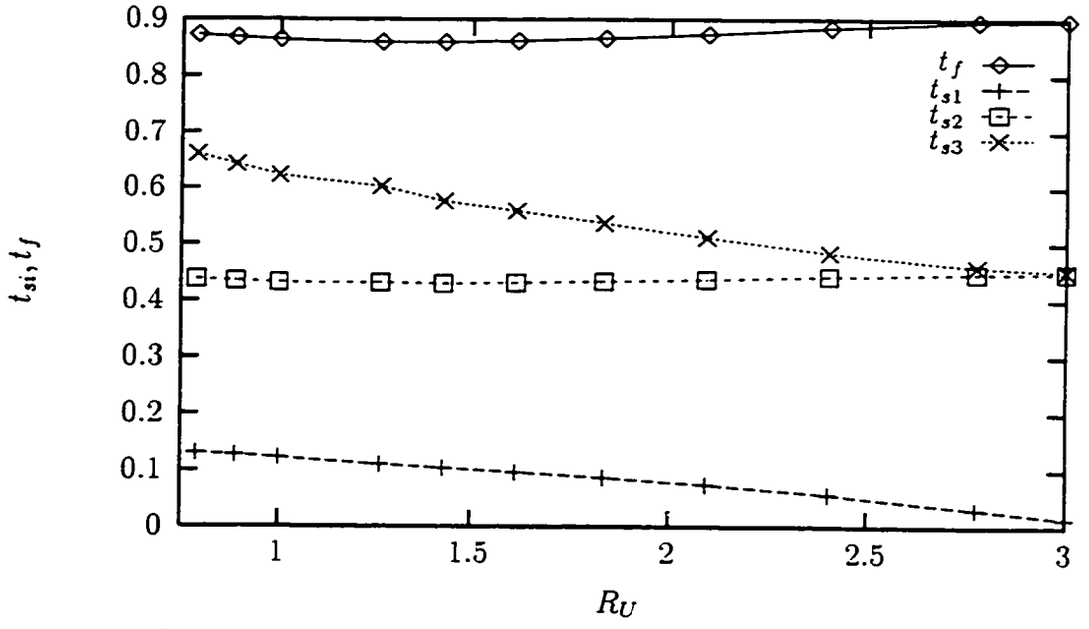

Figure 6.56: Optimal switch times t_{si} & t_f versus R_U (Example Seven, $\varphi_{1_f} = 0.76$).

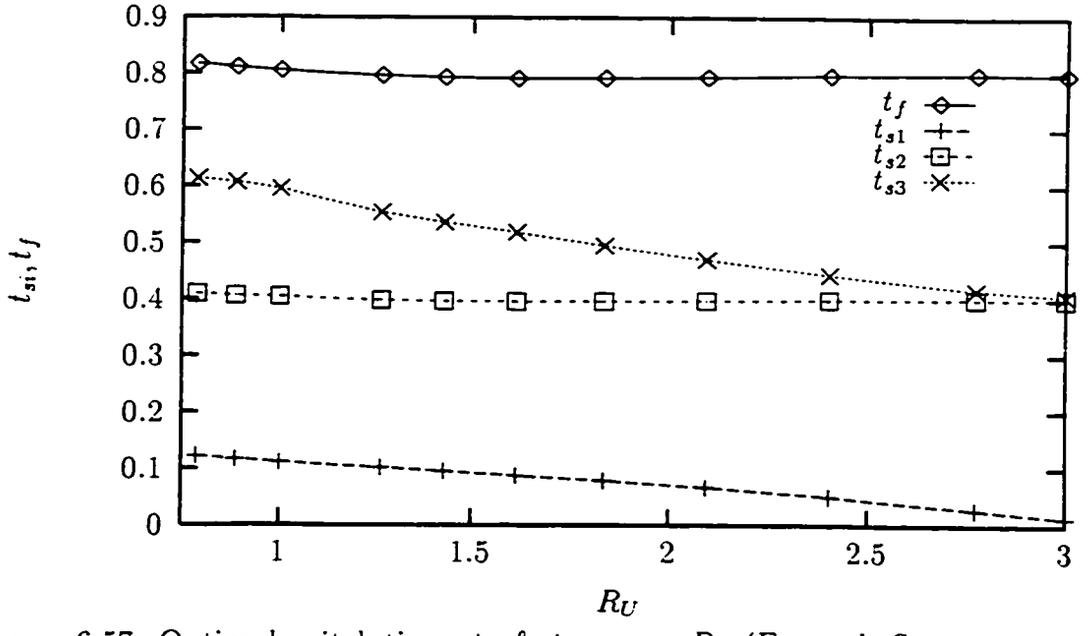

Figure 6.57: Optimal switch times t_{si} & t_f versus R_U (Example Seven, $\varphi_{1_f} = 0.6$).

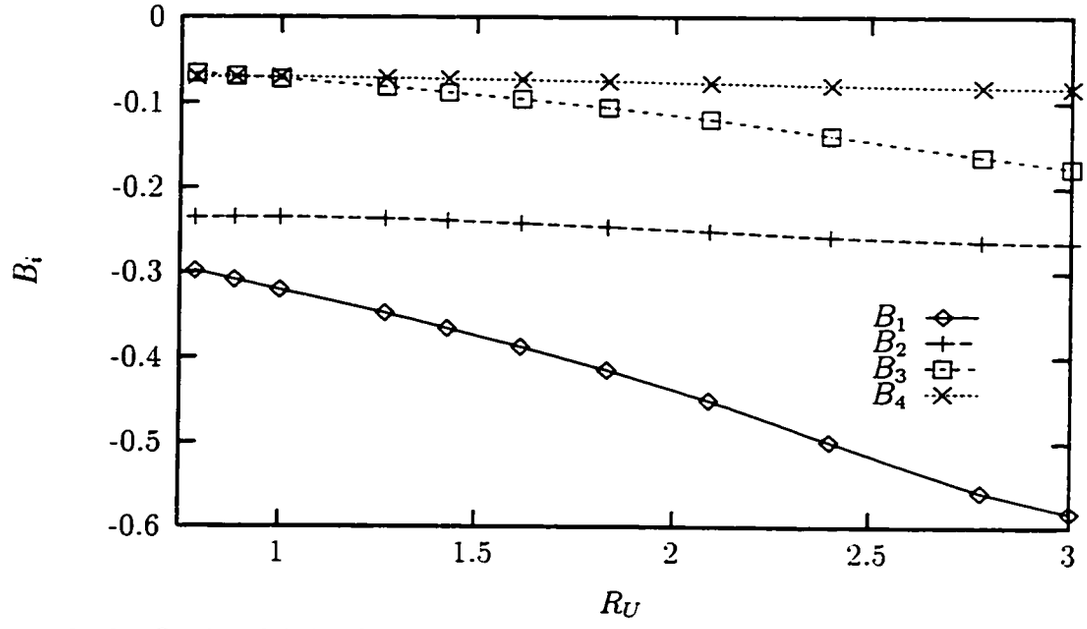

Figure 6.58: Optimal initial costates B_i as function of R_U (Example Seven, $\varphi_{1_f} = 0.76$)

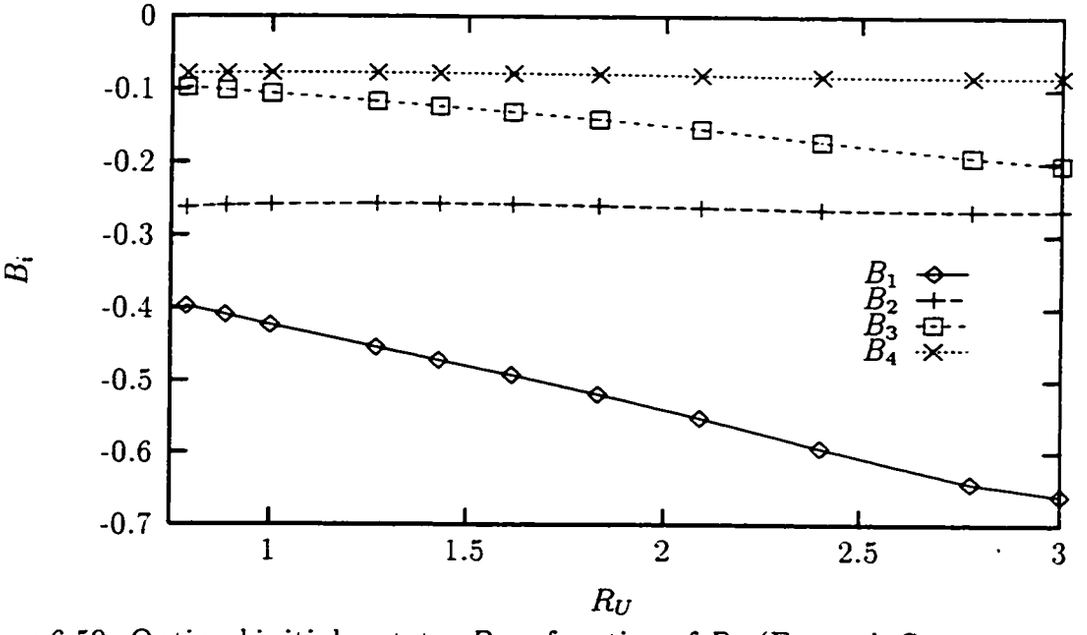

Figure 6.59: Optimal initial costates B_i as function of R_U (Example Seven, $\varphi_{1_f} = 0.6$).

6.4.3 Example Eight: Effects of Joint/Tip Masses

To show the effects of the mass m_a at the tip of shoulder link and the mass m_b at the tip of elbow link two cases are considered. In the first case, for which m_a varies and $m_b = 0$, it is assumed that $R_L = 1.6$ and $R_U = 2.78$. The physical parameters are given as:

$$\begin{array}{lll} l_1 = 2l_{c1} = .40\ [m] & U_1^{\mp} = \mp 25.0\ [Nm] & d_1 = d_2 = 0.1098723\ [m] \\ l_2 = 2l_{c2} = .25\ [m] & U_2^{\mp} = \mp 9.0\ [Nm] & I_1 = 0.416739\ [kg.m^2] \\ m_1 = 29.58152\ [kg] & m_b = 0 \leq m_a \leq 5 & I_2 = 0.110244\ [kg.m^2] \\ m_2 = 18.48846\ [kg] & I_o = I_a = I_b = 0 & g_r = 0 \end{array} \tag{6.36}$$

In the second case, for which m_b varies and $m_a = 0$, it is assumed that $R_L = 1.0$, $R_U = 1.83$. The physical parameters are given as:

$$\begin{array}{lll} l_1 = 2l_{c1} = .325\ [m] & U_1^{\mp} = \mp 22.0\ [Nm] & d_1 = d_2 = 0.1098723\ [m] \\ l_2 = 2l_{c2} = .325\ [m] & U_2^{\mp} = \mp 12.0\ [Nm] & I_1 = 0.229692\ [kg.m^2] \\ m_1 = 24.035\ [kg] & m_a = 0 \leq m_b \leq 12 & I_2 = 0.229692\ [kg.m^2] \\ m_2 = 24.035\ [kg] & I_o = I_a = I_b = 0 & g_r = 0 \end{array} \tag{6.37}$$

These example are again rest-to-rest motion from straight-to-straight configurations. The initial and the final conditions of the states are same as (6.34).

Figure 6.60 and Figure 6.61 show optimal switch times t_{si}, final time t_f, and optimal initial costates as functions of shoulder tip mass m_a for $\varphi_{1_f} = 0.6$. Figure 6.62 and Figure 6.63 show optimal switch times t_{si}, final time t_f, and optimal initial costates as functions of elbow tip mass m_b for $\varphi_{1_f} = 0.6$. From these plots one can observe that the increase in the tip masses m_a and m_b increases the final time of maneuver. Figure 6.62 indicates the jump from three switches solution to four switches solution at $m_b = 11$. As well Figure 6.60 and Figure 6.62 show sharper change of maneuver time for m_b than the change for m_a, which is instinctively predictable.

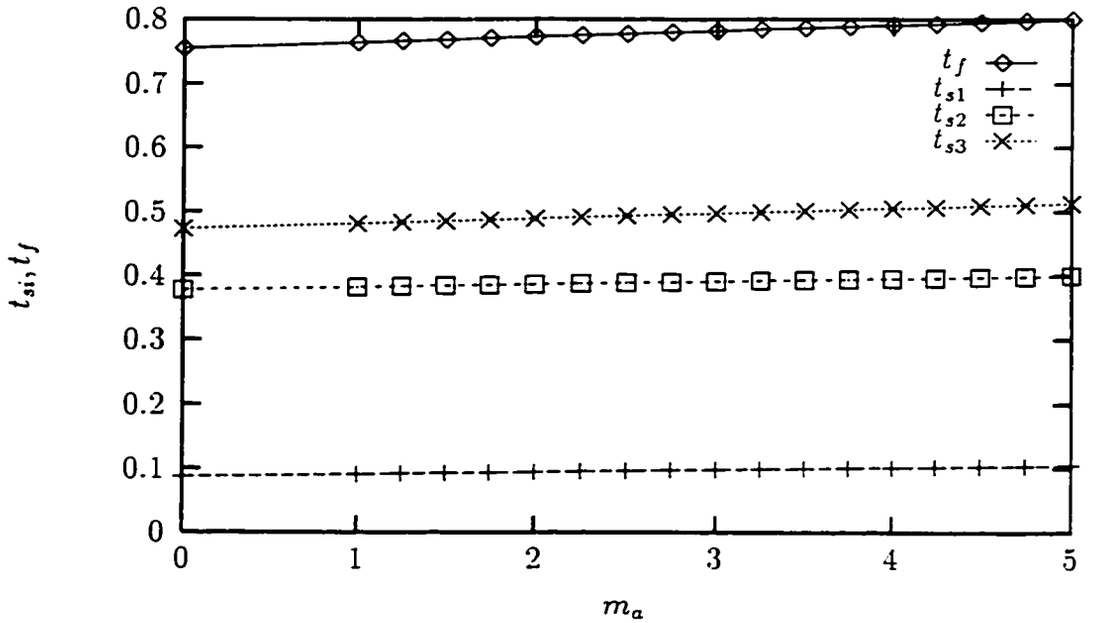

Figure 6.60: Optimal switch times t_{si} & t_f versus m_a(Example Eight, $\varphi_{1_f} = 0.6$).

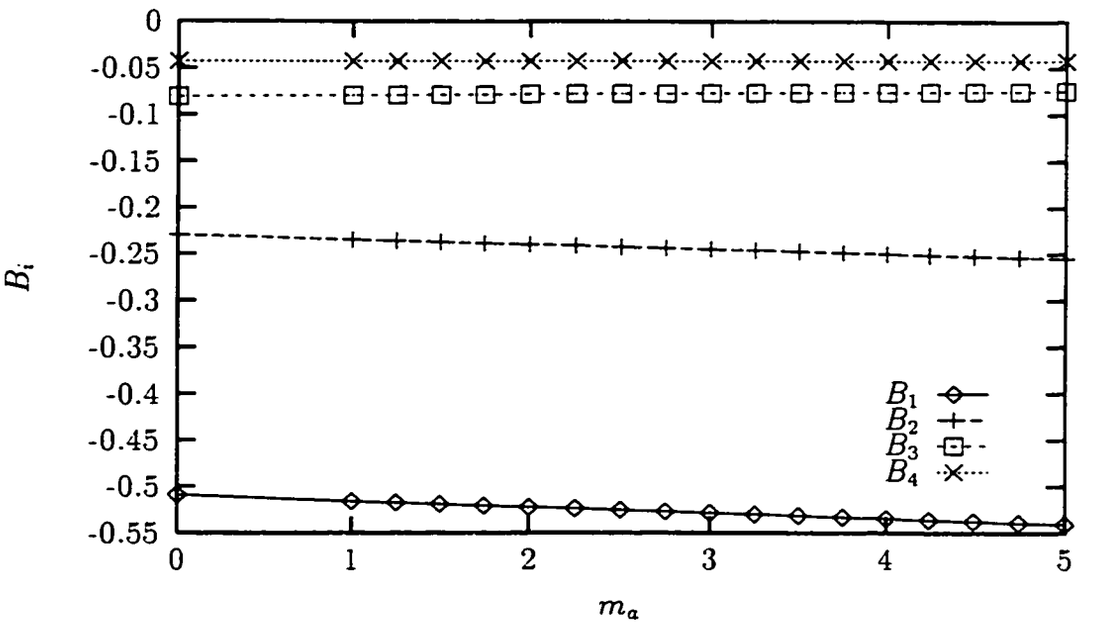

Figure 6.61: Optimal initial costates B_i as function of m_a (Example Eight, $\varphi_{1_f} = 0.6$).

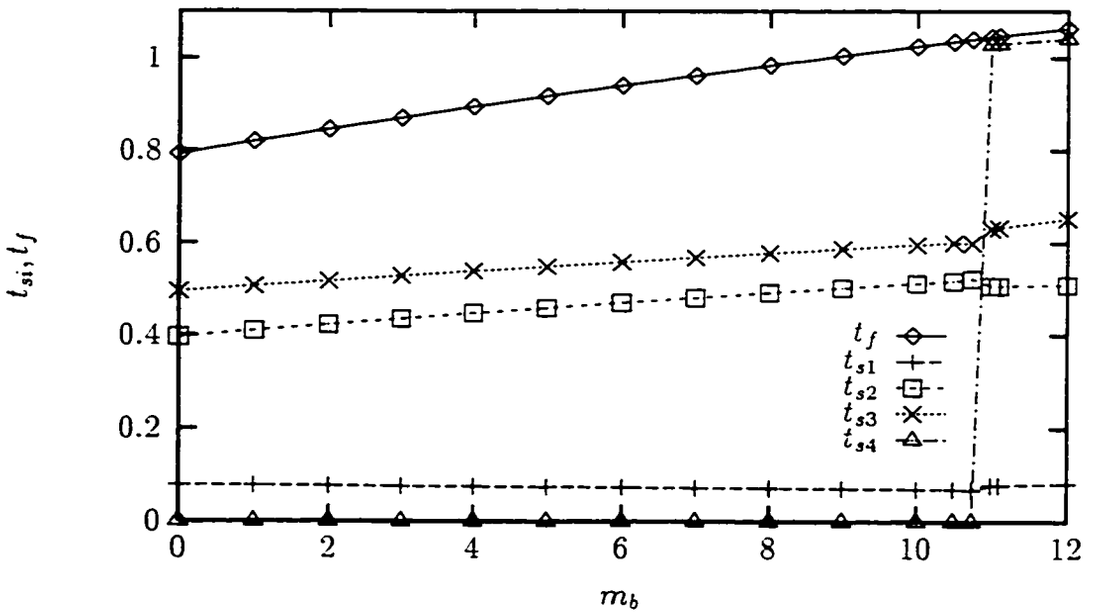

Figure 6.62: Optimal switch times t_{si} & t_f versus m_b(Example Eight, $\varphi_{1_f} = 0.6$).

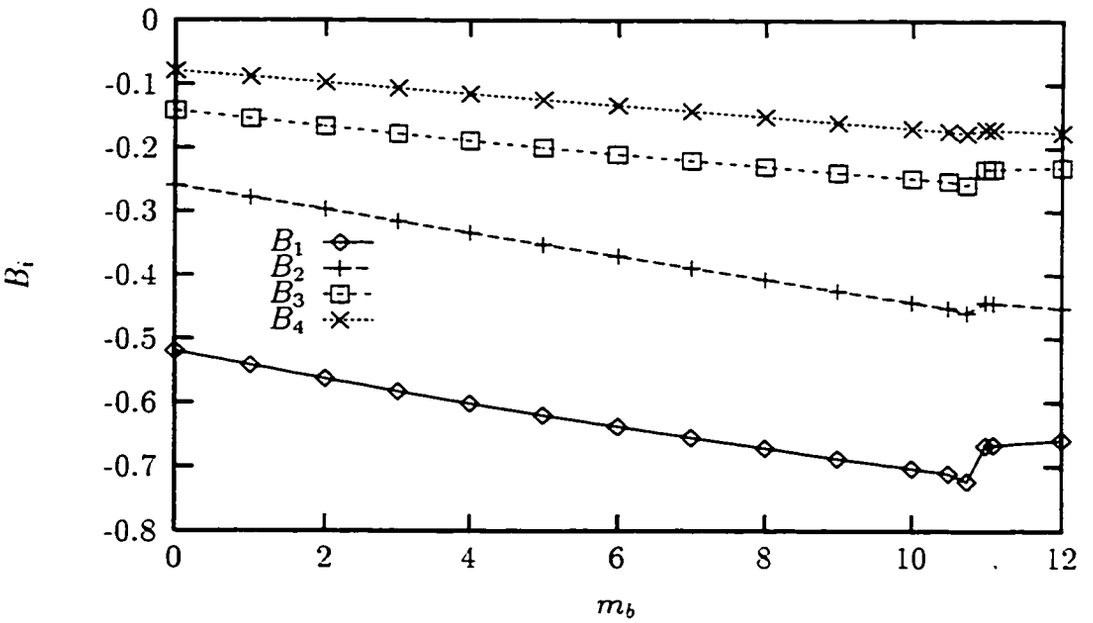

Figure 6.63: Optimal initial costates B_i as function of m_b(Example Eight, $\varphi_{1_f} = 0.6$).

6.4.4 Example Nine: Effects of Gravity

The effects of the gravity on optimal rest-to-rest motion of two-link manipulators from straight-to-straight configurations is discussed here. The physical parameters are given as:

$$\begin{array}{lll} l_1 = 2l_{c1} = 0.2\ [m] & U_1^{\mp} = \mp 10.0\ [Nm] & \\ l_2 = 2l_{c2} = 0.2\ [m] & U_2^{\mp} = \mp 5.0\ [Nm] & I_1 = 0.004167\ [kg.m^2] \\ m_1 = 1.0\ [kg] & m_a = 0 = m_b & I_2 = 0.004167\ [kg.m^2] \\ m_2 = 1.0\ [kg] & I_o = I_a = I_b = 0 & 0 \leq g_r \leq 9.81\ [m.s^{-2}] \end{array} \tag{6.38}$$

In this example the gravity g_r is varied. Physically, such a variation can be implemented by tilting the plane of motion between being vertical (when $g_r = 9.81\ [m.s^{-2}]$) to being horizontal (when $g_r = 0$).

The initial and the final conditions of the states in $[rad]$ and $[rad/s]$ are:

$$\begin{aligned} x(t_0) &= \begin{bmatrix} 1.047 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \\ x(t_f) &= \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \end{aligned} \tag{6.39}$$

Figure 6.64 shows optimal switch times t_{si} and final time t_f for various gravitation g_r. As can be seen, the increase in the gravity increases the final time of maneuver only slightly. Figure 6.65 shows optimal initial costates as functions of g_r for this example. Interestingly, time-optimal maneuvers have four switches if $g_r \leq 7\ [m.s^{-2}]$ and three switches if $g_r \geq 8\ [m.s^{-2}]$. Also note that for the parameters selected for this TLM, the maximum equivalent moments due to the Earth gravity (for the TLM in horizontal configuration) are about 1 NM in the elbow joint and about 4 NM in the shoulder joint. These moments represent about 20% of the control moment in the elbow joint and about 40% of the control moment in the shoulder joint. However they increase the maneuver time t_f by about 9%.

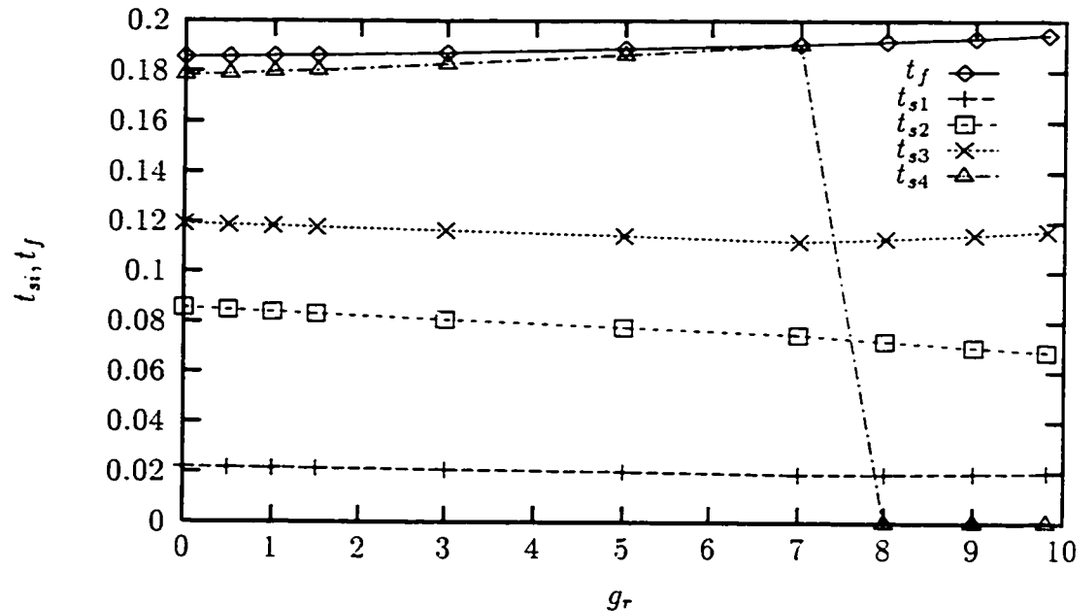

Figure 6.64: Optimal switch times t_{si} & t_f versus g_r(Example Nine, $\varphi_{10} = 1.047$).

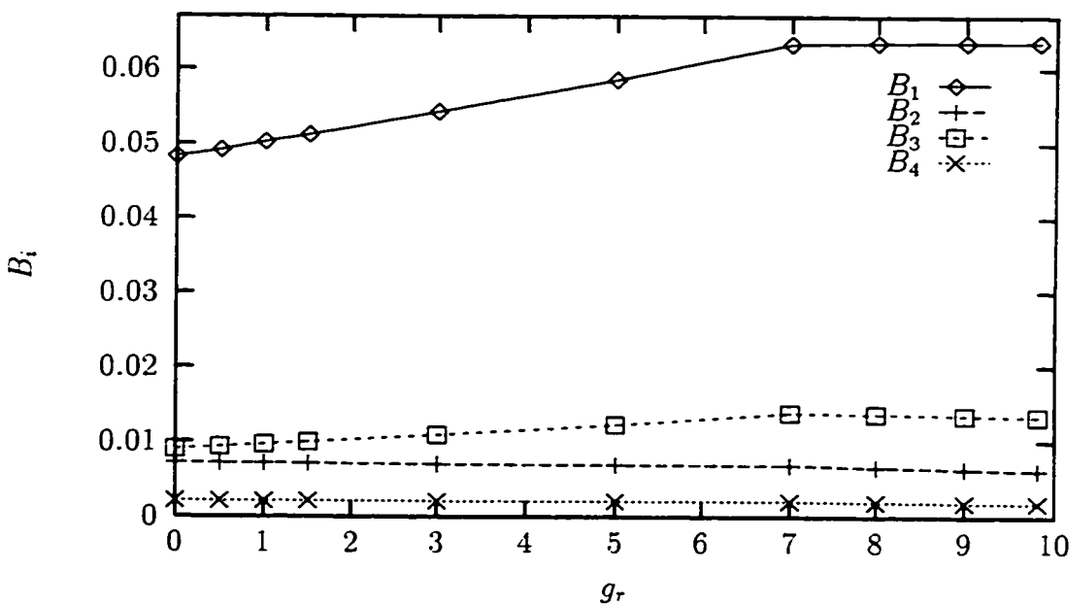

Figure 6.65: Optimal initial costates B_i as function of g_r(Example Nine, $\varphi_{10} = 1.047$).

Chapter 7

Closing

In closing, the main points of the research presented here are summarised and concluded. Also, possible future work of this research area is suggested.

7.1 Summary

A rest-to-rest maneuver time of Two-Link Manipulators (TLM) is minimised by the optimal use of the forces available to drive it. The problem is formulated using the optimal control theory. This theory mostly uses the ideas developed in Calculus of Variations as shown in Chapter 3. Also Pontryagin's Minimum Principle (PMP). which is applied to determine optimal paths, was explained in that chapter. The Lagrange equations for dynamics of rigid TLM were obtained in Chapter 4. Detailed formulation of the time-optimal control problem of TLM, including the explicit form of the state and costate equations and the corresponding switch functions was presented in that chapter. This problem was solved numerically as a Two-Point Boundary Value Problem (TPBVP) by the method described in Chapter 5. The results were presented in Chapter 6.

7.2 Conclusion

The TPBVP describing time-optimal control of two-link manipulators are not well behaved, and the solutions are not easy to obtain. Nevertheless results presented in Chapter 6 prove that this strategy, which directly uses the PMP to generate optimal solutions, can be successful. The results of are obtained with high numerical precision. Particularly, the values of the switch times and the final time of the bang-bang control are calculated more accurately than any other methods. When using general optimisation methods (the parametric optimisation outlined in Appendix A) accuracy of the solutions presented in the literature, especially in terms of location of the switch times, was usually quite poor. It can be only speculated that in order to achieve the precision of the results shown in this thesis, it would probably have required an unreasonably large number of iterations. In the following sections the discussion of more important conclusions of the solution method and the results is presented.

7.2.1 Solution Method

Shooting methods (SM) were used to solve TPBVP arising from time-optimal control problems for TLM.

Since the problem was very sensitive to the unknown initial conditions (costates), a procedure had to be worked out that would provide the initial costates sufficiently close to their optimal values for the SM to converge.

This was achieved by introducing a numerical approach, called the Forward-Backward Method (FBM). This method by itself can solve time-optimal control problems, although the convergence of the FBM was difficult to obtain. Nevertheless, the initial costates generated by the FBM were found to be sufficiently close to their optimal values. It allowed the combination of the FBM and the SM into the procedure that was able to solve TPBVP for time-optimal control of TLM successfully.

The usefulness of the FBM for generation of the initial costate for linear and nonlinear systems with single, or double controls was demonstrated on numerical examples. Since the FBM does not use any linearisation in generating the initial costates, it can be applied to any nonlinear problems.

7.2.2 Number of Switches of the Bang-Bang Control

The number of switches for bang-bang time-optimal control has been investigated by many researchers. In [40] it was concluded that only the solutions with three switches could be optimal. Here it was shown that the number of switches is not constant and is related to the magnitude of the maneuver. In general, the optimal solution has three switches for smaller tasks and four switches for bigger tasks.

7.2.3 Effects of Flexibility

The effects of flexibility of the manipulator were investigated in Section 6.3. The finite element method was used to verify the performance of the flexible manipulators under the control calculated for time-optimal control problems of *rigid* manipulators. The results indicated that the vibrations due to flexibility affected the response (angular velocity) of the links more than the motion of the tip of the manipulator. In general, the dynamic performance of the flexible manipulators under such controls was found to be satisfactory if the slenderness of the links was sufficiently small as explained earlier.

7.2.4 Physical Parameters of Manipulator

The effects of physical parameters such as geometry and masses of the links as well as masses at the tips of the links and the effects of changing the limits of control forces on time-optimal maneuvers of the TLM were investigated.

It was concluded that a shorter maneuver time could be obtained by maximising

the ratio of the length of shoulder link over the length of elbow link. It was also shown that there is an optimal ratio of control force applied in the shoulder joint over the control force applied in the elbow joint, at which the time of optimal maneuver is minimised.

The effects of the mass m_a at the tip of the shoulder link and the mass m_b at the tip of the elbow link were found to be as intuitively predicted. The increase in the tip masses m_a and/or m_b increases the final time of maneuver.

The effects of the gravitation were also determined. Interestingly, even though the equivalent forces at joints due to gravity were almost half of the control forces applied, they caused only slight increase to the final time of maneuver (in comparison with the maneuver without the gravity).

7.3 Future Work

As an extension of this research the future work should include study towards further improvement of the numerical procedure calculating the optimal paths of rigid TLM. Also, effects of flexibility of the links, which were briefly touched here, should be more throughly investigated. This can be done in one of the following two ways.

7.3.1 Flexible Manipulator with Control of Rigid Dynamics

The optimal control forces were obtained assuming the dynamics for the rigid links. Consequently the vibrations due to those control forces were observed. Those vibrations were not significant, at least for the examples that were used. However, for more slender links one may expect some significant vibrations during the maneuver. For reducing those vibrations, one can use the sensitivity of the final conditions and a shooting method to correct the control forces applied. In this approach the control forces obtained using the rigid dynamics become the initial guess for the iterative method which includes the flexible dynamics. The control configuration can be cor-

rected through the iteration by changing the locations of switch times and the final time (using parametric optimisation for example). Using the error of calculated final configuration of flexible manipulator and the given final conditions, the vibration of the links can be reduced to zero. For the dynamics of flexible links an FEM program such as ANSYS can be used. This can be considered as the iteration of the approach used in Section 6.3 and parametric optimisation when the locations of switch times and the final time are the optimisation variables. The similar approach has been successfully implemented for a single-link manipulator in [23]. This approach however would be a semi time-optimal control rather than time-optimal control.

7.3.2 Optimal Control of Flexible Manipulator

The other alternative is to use the dynamics of flexible manipulators when solving the time-optimal control problem of TLM. However, for this approach, the explicit form of the dynamics of the flexible TLM is required. Deriving the state and the costate equations for such a system would probably be too difficult. However, a simplified form of the explicit dynamics rather than the exact dynamics for flexible TLM might be used. One can always use a finite element approximation for the dynamics. As soon as a set of equations to calculate the costates is established and an explicit form of switch functions becomes available, then the numerical approach introduced in Chapter 5 can be used to calculate the optimal control forces.

References

[1] J. J. Craig, *Introduction to Robotics.* Reading, Massachusetts: Addison-Wesley, 1986.

[2] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence.* New York, N.Y.: McGraw-Hill Book Company, 1987.

[3] G. J. Lastman, "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.

[4] J. S. Arora, *Introduction to Optimum Design.* New York, NY: McGraw-Hill Book Company, 1989.

[5] D. E. Kirk, *Optimal Control Theory an Introduction.* Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1970.

[6] E. R. Pinch, *Optimal Control and the Calculus of Variations.* New York: Oxford University Press Inc., 1993.

[7] H. Eschenauer, J. Koski, and A. Osyczka, *Multicriteria Design Optimization.* Heidelberg, Germany: Springer-Verlag, 1990.

[8] W. Stadler, "Multicriteria optimization in mechanics: A survey," *Journal of Applied Mechanics Reviews*, vol. 37, no. 3, pp. 227–286, 1984.

[9] A. Adamian and J. Gibson, "Integrated control/structure design and robustness," in *Society of Automotive Engineers, Inc.*, pp. 7.1336–7.1343, 1987.

[10] M. Milman, M. Salama, R. Scheid, and R. Bruno, "Combined control-structural optimization," *Computational Mechanics*, vol. 8, pp. 1–18, 1991.

[11] A. V. Balakrishnan, "Combined structures-controls-integrated optimization using distributed parameter models," *Computational Mechanics*, vol. 8, pp. 125–133, 1991.

[12] Y. Chen, S. Y. P. Chien, and A. A. Desrochers, "General structure of time-optimal control of robotic manipulators moving along prescribed paths," *International Journal of Control*, vol. 56, no. 4, pp. 767–782, 1992.

[13] B. K. Kim and K. G. Shin, "Suboptimal control of industrial manipulators with a weighted minimum time-fuel criterion," *IEEE Transactions on Automatic Control*, vol. AC-30, no. 1, pp. 1–10, 1985.

[14] K. G. Grigor'ev, M. P. Zapletin, and D. A. Silaev, "Optimal lift-off of a spacecraft from the lunar surface into the circular orbit of a lunar satellite," *Journal of Cosmic Research*, vol. 29, no. 5, pp. 595–603, 1991.

[15] G. Bessonnet and J. P. Lallemand, "Optimal trajectories of robot arms minimising constrained actuators and travelling time," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 112–117, 1990.

[16] L. Meirovitch and Y. Sharony, "Optimal vibration control of flexible spacecraft during a minimum-time maneuver," *Journal of Optimization Theory And Applications*, vol. 69, no. 1, pp. 31–54, 1991.

[17] T. Teramoto, K. Ono, and O. Turhan, "High-speed motion control of mechanisms under average heat generation restriction (minimum-time trajectory planning of robotic manipulator and calculated examples of two degree-of-freedom system)," *JSME International Journal, Series C*, vol. 37, no. 2, pp. 315–321, 1994.

[18] G. T. Flowers and V. B. Venkayya, "Adaptive decentralised control of flexible multibody structures," in *AIAA-92-4741-CP*, pp. 404–414, AIAA, 1992.

[19] J. Carusone, K. S. Buchan, and G. M. T. D'Eleuterio, "End-effector tracking control for structurally flexible manipulators," in *Proceedings of American Control Conference*, (Pittsburgh, PA), June 21-23 1989.

[20] G. R. Eisler, R. D. Robinett, D. Segalman, and J. Feddema, "Approximate optimal trajectories for flexible-link manipulator slewing using recursive quadratic programming," *Journal of Dynamic Systems, Measurement, and Control*, vol. 115, pp. 405–410, 1993.

[21] D. F. Miller and J. Shim, "Gradient-based combined structural and control optimization," *Journal of Guidance, Control and Dynamics*, vol. 10, no. 3, pp. 291–298, 1987.

[22] S. S. Rao, T. S. Pan, and V. B. Venkayya, "Modelling, control, and design of flexible structures: A survey," *Journal of Applied Mechanics Reviews*, vol. 43, no. 5, pp. 99–117, 1990.

[23] W. Szyszkowski and D. Youck, "Optimal control of a flexible manipulator," *Journal of Computers & Structures*, vol. 47, no. 4/5, pp. 801–813, 1993.

[24] S. K. Biswas and R. D. Klafter, "Dynamic modelling and optimal control of flexible robotic manipulators," in *Proceedings of IEEE International Conference on Robotics and Automation*, (Philadelphia), pp. 15–20, May 1988.

[25] V. J. Modi and J. K. Chan, "Performance of an flexible mobile manipulator," in *unknown*, pp. 375–384, unknown.

[26] H. M. Chun, J. D. Turner, and J.-N. Juang, "Frequency-shaped large-angle maneuvers," *The Journal of the Astronautical Science*, vol. 36, no. 3, pp. 219–243, 1988.

[27] H. Yamakawa, "Simultaneous optimization of nonlinear structural and control system," in *AIAA-92-4742-CP*, pp. 415–423, AIAA, 1992.

[28] Y. Chen and A. A. Desrochers, "Structure of minimum-time control law for robotic manipulators with constrained paths," in *Proceedings of IEEE International Conference on Robotics and Automation*, (Scottsdale, AZ), pp. 971–976, May 14-19 1989.

[29] H. P. Geering, L. Guzzella, S. A. R. Hepner, and C. H. Onder, "Time-optimal motions of robots in assembly tasks," *IEEE Transactions on Automatic Control*, vol. AC-31, no. 6, pp. 512–518, 1986.

[30] L. G. Van Willigenburg, "First order controllability and the time optimal control problem for rigid articulated arm robots with friction," *International Journal of Control*, vol. 51, no. 6, pp. 1159–1171, 1990.

[31] J. Y. Fourquet, "Optimal control theory and complexity of the time-optimal problem for rigid manipulators," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Yokohama, Japan), pp. 84–90, July 26-30 1993.

[32] A. M. Formal Sky and S. N. Osipov, "On the problem of the time-optimal manipulator arm turning," *IEEE Transactions on Automatic Control*, vol. 35, no. 6, pp. 714–719, 1990.

[33] L. Y. Pao and G. F. Franklin, "Time-optimal control of flexible structures," in *Proceedings of The 29th IEEE conference on Decision and Control*, (Honolulu, HI), pp. 2580–2581, Dec. 1990.

[34] B. Wie, C.-H. Chuang, and J. Sunkel, "Minimum-time pointing control of a two-link manipulator," *Journal of Guidance, Control and Dynamics*, vol. 13, no. 5, pp. 867–873, 1990.

[35] F. Li and P. M. Bainum, "An improved shooting method for solving minimum-time maneuver problems," *Advances in Dynamics and Control of Flexible Spacecraft and Space-based Manipulators*, vol. 20, pp. 13–18, 1990.

[36] P. M. Bainum and F. Li, "Rapid in-plane maneuvering of the flexible orbiting scole," *The Journal of the Astronautical Science*, vol. 39, no. 2, pp. 233–248, 1991.

[37] G. J. Lastman and B. D. Tapley, "Optimization of non-linear systems with inequality constraints explicitly containing the control," *International Journal of Control*, vol. 12, no. 3, pp. 497–510, 1978.

[38] H. R. Sirisena, "A gradient method for computing optimal bang-bang controls," *International Journal of Control*, vol. 19, no. 2, pp. 257–264, 1974.

[39] H. R. Sirisena, "Optimal control of saturating linear plants for quadratic performance indices-ii," *International Journal of Control*, vol. 12, no. 5, pp. 739–752, 1970.

[40] L. G. Van Willigenburg and R. P. H. Loop, "Computation of time-optimal controls applied to rigid manipulators with friction," *International Journal of Control*, vol. 54, no. 5, pp. 1097–1117, 1991.

[41] C. J. Goh and K. L. Teo, "Optimal initial choice of multipliers in the quasilinearization method for optimal control problems with bounded controls," *Automatica*, vol. 24, no. 1, pp. 3–18, 1988.

[42] L. S. Lasdon, S. K. Mitter, and A. D. Waren, "The conjugate gradient method for optimal control problems," *IEEE Transactions on Automatic Control*, vol. AC-12, no. 2, pp. 132–138, 1967.

[43] B. P. Yeo, K. J. Waldron, and B. S. Goh, "Optimal initial choice of multipliers in the quasilinearization method for optimal control problems with bounded controls," *International Journal of Control*, vol. 20, no. 1, pp. 17–33, 1974.

[44] A. Miele and R. R. Iyer, "General technique for solving nonlinear, two-point boundary-value problems via the method of particular solutions," *Journal of Optimization Theory And Applications*, vol. 5, no. 5, pp. 382–399, 1970.

[45] A. Weinreb and A. E. Bryson, "Optimal control of systems with hard control bounds," *IEEE Transactions on Automatic Control*, vol. AC-30, no. 11, pp. 1135–1138, 1985.

[46] M. Yamamoto and A. Mohri, "Planning of quasi-minimum time trajectories for robot manipulators (generation of a bang-bang control)," *Robotica*, vol. 7, pp. 43–47, 1989.

[47] W. Szyszkowski and R. Fotouhi-C, "A numerical method for time-optimal control of double arms robot," *IEEE Transactions on Automatic Control*, 1995. submitted.

[48] R. Fotouhi-C. and W. Szyszkowski, "An algorithm for time-optimal control problems," *Journal of Dynamic Systems, Measurement, and Control*, 1995. submitted.

[49] R. Fotouhi-C and W. Szyszkowski, "A numerical approach for time-optimal control of double arms robot," in *The fourth IEEE Conference on Control Applications (CCA)*, (Albany, NY), September 28-29 1995.

[50] R. Fotouhi-C and W. Szyszkowski, "Effects of flexibility on time-optimal control of two links manipulators," in *Eleventh International Conference on Systems Engineering, University of Nevada*, (Las Vegas, Nevada), July 9-11 1996.

[51] W. Szyszkowski and R. Fotouhi-C, "On the number of switches in time-optimal control of manipulators," *Optimal Control, Applications & Methods*, 1994. submitted.

[52] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko, *The Mathematical Theory of Optimal Processes.* New York: Interscience Publishers, Inc., 1962.

[53] H. Goldstein, *Classical Mechanics.* Reading, Massachusetts: Addison-Wesley Publishing Company, 1980.

[54] S. M. Roberts and J. S. Shipman, *Two-Point Boundary Value Problems: Shooting Methods.* New York, N.Y.: American Elsevier Publishing Company, Inc., 1972.

[55] I. Swanson Analysis Systems, "Beam3 2-d elastic beam and mass21 structural mass," in *Ansys User's Manual for Revision 5.0, Volume III*, pp. 4–13, 4–113, Houston, PA: Swanson Analysis Systems, Inc., 1992.

[56] A. R. Fraser and R. W. Daniel, *Perturbation Techniques for Flexible Manipulators.* Boston, Massachusetts: Kluwer Academic Publishers, 1991.

[57] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran: The Art of Scientific Computing, Second Edition.* Boston, Massachusetts: Cambridge University Press, 1992.

Appendix A

Parametric Optimisation

Here the *Direct Method (DM)* of solving optimal control problems is briefly outlined. This method sometimes is called *Parametric Optimisation Method.*

The optimal control problems can be transformed to optimum design problems and treated by the parametric optimisation methods as described in [4].

First the integral for the performance measure J is replaced by the summation.

$$J(u) = \int_{t_0}^{t_f} G(x,u,t)dt = J(u_k,t_f) = \sum_{k=0}^{n-1} \int_{t_k}^{t_{k+1}} G(x_k,u_k,t_k)dt \qquad \text{(A.1)}$$

where n is the number of time steps, thus $t_n = t_f$. In the time interval $t_k < t < t_{k+1}$ the control u_k is assumed constant, and the state $x_k(t)$ in that time interval is obtained

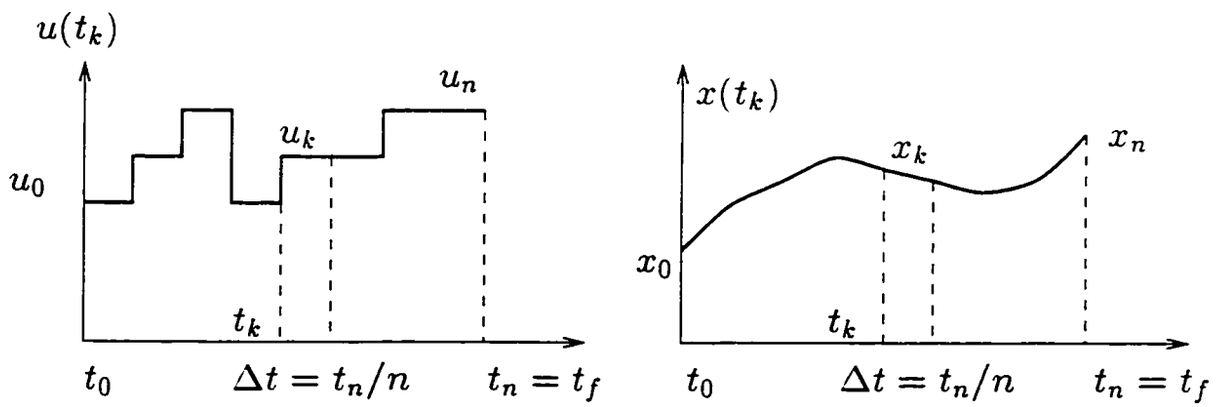

Figure A.1: Control, states

from integration of the state equation (dynamic constraint),

$$\dot{x_k} = a(x_k) + c(x_k)u_k \tag{A.2}$$

where x_k and u_k are the vectors of states and controls at time step t_k.

The controls constraint is,

$$U^- \leq u_k \leq U^+ \tag{A.3}$$

The boundary conditions constraint are

$$x_0(t_0) = x_0 \qquad x_n(t_f) = x_f \tag{A.4}$$

Here the objective function is $J(u_k, t_f)$ and the optimisation variables are $(u_1, \ldots, u_n, t_f)$. The control must meet the inequality constraints given in equation (A.3), and the state variables $x_k(t)$ must satisfy the equality constraints, equation (A.4). Note that the equality constraints (A.2) are automatically satisfied when integrating the state equations.

For time-optimal control if one attempts only bang-bang controls then the number of optimisation variables u_k can be reduced. Instead of using equal intervals dt one can use the switch times as optimisation variables and assume $u_k = U^-$ or $u_k = U^+$. Thus

$$J(t_{si}, t_f) = \sum_{si=1}^{m} \int_{t_{si}}^{t_{si+1}} G(x_{si}, u_{si}, t_{si}) dt \tag{A.5}$$

where $m + 1$ is the number of optimisation variables or switch times plus final time.

Some of numerical methods described in [4] can solve formulated optimal control problems. It must be noted that the costate equations does not come into the calculation here. Only the state equations are used.

Appendix B

Analytical Derivations

B.1 Derivations for Calculus of Variations

The complete derivations of the Section 3.2 is as followed. Let x be a vector of admissible functions which satisfy the end conditions, and determine the variation $\delta J(x, \delta x)$ from the increment $\Delta J(x, \delta x)$,

$$\begin{aligned}
\Delta J(x, \delta x) &= J(x + \delta x) - J(x) && \text{(B.1)} \\
&= \int_{t_0}^{t_f + \delta t_f} g(x + \delta x, \dot{x} + \delta \dot{x}, t) dt - \int_{t_0}^{t_f} g(x, \dot{x}, t) dt && \text{(B.2)} \\
&= \int_{t_0}^{t_f} \left[g(x + \delta x, \dot{x} + \delta \dot{x}, t) - g(x, \dot{x}, t) \right] dt \\
&\quad + \int_{t_f}^{t_f + \delta t_f} g(x + \delta x, \dot{x} + \delta \dot{x}, t) dt && \text{(B.3)}
\end{aligned}$$

where $\delta \dot{x} = \frac{d}{dt} \delta x$. Eventually, ΔJ will be expressed entirely in terms of x, $\dot{x}$, and δx. The first integrand can be expanded about x, $\dot{x}$ in Taylor series to give ΔJ as

$$\begin{aligned}
\Delta J(x, \delta x) &= \int_{t_0}^{t_f} \left\{ \left[\frac{\partial g}{\partial x}(x, \dot{x}, t) \right] \delta x + \left[\frac{\partial g}{\partial \dot{x}}(x, \dot{x}, t) \right] \delta \dot{x} \right\} dt + O(\delta x, \delta \dot{x}) \\
&\quad + \int_{t_f}^{t_f + \delta t_f} g(x + \delta x, \dot{x} + \delta \dot{x}, t) dt && \text{(B.4)}
\end{aligned}$$

Where $O(\delta x, \delta \dot{x})$ or simply $O(.)$ denotes terms of higher than first order. Also we can write the second integrand as

$$\int_{t_f}^{t_f + \delta t_f} g(x + \delta x, \dot{x} + \delta \dot{x}, t) dt = \left[g(x + \delta x, \dot{x} + \delta \dot{x}, t) \right]_{t_f} \delta t_f + O(\delta t_f) \qquad \text{(B.5)}$$

or at $t = t_f$ using the Taylor series expansion

$$g(x+\delta x, \dot{x}+\delta\dot{x}, t)_{t_f} = g(x,\dot{x},t)_{t_f} + \left[\frac{\partial g}{\partial x}(x,\dot{x},t)\right]_{t_f} \delta x(t_f) + \left[\frac{\partial g}{\partial \dot{x}}(x,\dot{x},t)\right]_{t_f} \delta\dot{x}(t_f) + O(\cdot) \tag{B.6}$$

Integrating (B.4) and substituting (B.5) and (B.6), we get

$$\begin{aligned} \Delta J(x,\delta x) &= \int_{t_0}^{t_f} \left\{ \frac{\partial g}{\partial x}(x,\dot{x},t) - \frac{d}{dt}\left[\frac{\partial g}{\partial \dot{x}}(x,\dot{x},t)\right] \right\} (\delta x) dt + O(\cdot) \\ &\quad + \left[\frac{\partial g}{\partial \dot{x}}(x,\dot{x},t)\right]_{t_f} \delta x(t_f) + [g(x,\dot{x},t)]_{t_f} \delta t_f \end{aligned} \tag{B.7}$$

The variation of J, which is δJ, includes only first order terms of the increment ΔJ, and the dependence of $\delta x(t_f)$ to δt_f and δx_f is linearly approximated. From Figure 3.1 we have

$$\delta x(t_f) = \delta x_f - \dot{x}(t_f)\delta t_f \tag{B.8}$$

substituting this equation (B.8) into (B.7) we can get the variation $\delta J(x,\delta x)$ as

$$\begin{aligned} \delta J(x,\delta x) = 0 = &\int_{t_0}^{t_f} \left\{ \frac{\partial g}{\partial x}(x,\dot{x},t) - \frac{d}{dt}\left[\frac{\partial g}{\partial \dot{x}}(x,\dot{x},t)\right] \right\} (\delta x) dt \\ &+ \left[\frac{\partial g}{\partial \dot{x}}(x,\dot{x},t)\right]_{t_f} \delta x_f + \left[g(x,\dot{x},t) - \left(\frac{\partial g}{\partial \dot{x}}(x,\dot{x},t)\right)\dot{x} \right]_{t_f} \delta t_f \end{aligned} \tag{B.9}$$

Applying the fundamental theorem of calculus of variation (vanishing the first variation) we find that a necessary condition for x to be an extremal is

$$\frac{\partial g}{\partial x}(x,\dot{x},t) - \frac{d}{dt}\left[\frac{\partial g}{\partial \dot{x}}(x,\dot{x},t)\right] = 0 \tag{B.10}$$

for all $t \in [t_0, t_f]$. This is the *Euler equation* for fixed end points problem. It must be noted that an extremal for this free end point problem must also be an extremal for certain fixed end points problem. So x is the solution of fixed as well as free end point problems. The boundary condition then at the final time are determined by the relationship

$$\delta J(x,\delta x) = 0 = \left[\frac{\partial g}{\partial \dot{x}}(x,\dot{x},t)\right]_{t_f} \delta x_f + \left[g(x,\dot{x},t) - \left(\frac{\partial g}{\partial \dot{x}}(x,\dot{x},t)\right)\dot{x} \right]_{t_f} \delta t_f \tag{B.11}$$

Equations (B.10) and (B.11) are *the necessary conditions* that must be satisfied by an extremal curve.

B.2 Derivations for State Equations

The linear and angular velocities in Equation (4.16) can be obtained as:

$$\begin{aligned} \omega_1 &= \dot{\varphi_1} & v_{c1} &= \omega_1 \times l_{c1} & v_a &= \omega_1 \times l_1 \\ \omega_2 &= \dot{\varphi_1} + \dot{\varphi_2} & v_{c2/a} &= \omega_2 \times l_{c2} & v_{b/a} &= \omega_2 \times l_2 \end{aligned}$$

$$\begin{aligned} \vec{v}_{c2} &= \vec{v}_a + \vec{v}_{c2/a} & v_{c2}^2 &= v_a^2 + v_{c2/a}^2 + 2v_a v_{c2/a}\cos(\varphi_2) \\ \vec{v}_b &= \vec{v}_a + \vec{v}_{b/a} & v_b^2 &= v_a^2 + v_{b/a}^2 + 2v_a v_{b/a}\cos(\varphi_2) \end{aligned}$$

$$\begin{aligned} v_{c1}^2 &= l_{c1}^2 {\dot{\varphi_1}}^2 \\ v_a^2 &= l_1^2 {\dot{\varphi_1}}^2 \\ v_{c2}^2 &= l_1^2 {\dot{\varphi_1}}^2 + l_{c2}^2(\dot{\varphi_1} + \dot{\varphi_2})^2 + 2[l_1\dot{\varphi_1}][l_{c2}(\dot{\varphi_1} + \dot{\varphi_2})]\cos(\varphi_2) \\ v_b^2 &= l_1^2 {\dot{\varphi_1}}^2 + l_2^2(\dot{\varphi_1} + \dot{\varphi_2})^2 + 2[l_1\dot{\varphi_1}][l_2(\dot{\varphi_1} + \dot{\varphi_2})]\cos(\varphi_2) \end{aligned}$$

The kinetic energy for the rigid links is (4.17)

$$2 \times E_k = c_1 {\dot{\varphi_1}}^2 + c_2(\dot{\varphi_1} + \dot{\varphi_2})^2 + 2c_3\dot{\varphi_1}(\dot{\varphi_1} + \dot{\varphi_2})\cos(\varphi_2) \tag{B.12}$$

Also, the potential energy for the rigid links is (4.19)

$$E_p = \Big\{c_4 \sin(\varphi_1) + c_5 \sin(\varphi_1 + \varphi_2)\Big\} g \tag{B.13}$$

The derivatives necessary for the Euler-Lagrange equation (4.15) using (B.12) and

(B.13) are

$$\frac{\partial E_k}{\partial \dot{\varphi}_1} = c_1\dot{\varphi}_1 + c_2(\dot{\varphi}_1 + \dot{\varphi}_2) + c_3(2\dot{\varphi}_1 + \dot{\varphi}_2)\cos(\varphi_2)$$
$$\frac{\partial E_k}{\partial \dot{\varphi}_2} = c_2(\dot{\varphi}_1 + \dot{\varphi}_2) + c_3\dot{\varphi}_1\cos(\varphi_2)$$
$$\frac{d}{dt}\cos(\varphi_2) = -\dot{\varphi}_2\sin(\varphi_2)$$
$$\frac{d}{dt}\frac{\partial E_k}{\partial \dot{\varphi}_1} = (c_1 + c_2 + 2c_3\cos(\varphi_2))\ddot{\varphi}_1 + (c_2 + c_3\cos(\varphi_2))\ddot{\varphi}_2 - c_3(2\dot{\varphi}_1 + \dot{\varphi}_2)\dot{\varphi}_2\sin(\varphi_2)$$
$$\frac{d}{dt}\frac{\partial E_k}{\partial \dot{\varphi}_2} = (c_2 + c_3\cos(\varphi_2))\ddot{\varphi}_1 + c_2\ddot{\varphi}_2 - c_3\dot{\varphi}_1\dot{\varphi}_2\sin(\varphi_2)$$
$$\frac{\partial E_k}{\partial \varphi_1} = 0$$
$$\frac{\partial E_k}{\partial \varphi_2} = -c_3\dot{\varphi}_1(\dot{\varphi}_1 + \dot{\varphi}_2)\sin(\varphi_2)$$
$$\frac{\partial E_p}{\partial \varphi_1} = \Big\{c_4\cos(\varphi_1) + c_5\cos(\varphi_1 + \varphi_2)\Big\}g$$
$$\frac{\partial E_p}{\partial \varphi_2} = \Big\{c_5\cos(\varphi_1 + \varphi_2)\Big\}g$$

The parameters a_{ij} are given by relations (4.22) in which φ_1 and φ_2 are replaced by x_1 and x_3 respectively.

$$\begin{aligned}
a_{11} &= c_1 + c_2 + 2c_3\cos(x_3) & c_1 &= (I_0 + I_1 + I_a) + m_1 l_{c1}^2 + (m_2 + m_a + m_b)l_1^2 \\
a_{12} &= c_2 + c_3\cos(x_3) & c_2 &= (I_2 + I_b) + m_2 l_{c2}^2 + m_b l_2^2 \\
a_{13} &= c_3\sin(x_3) & c_3 &= l_1(m_2 l_{c2} + m_b l_2) \\
a_{22} &= c_2 & c_4 &= m_1 l_{c1} + (m_2 + m_a + m_b)l_1 \\
a_{14} &= c_4\cos(x_1) + c_5\cos(x_1 + x_3) & c_5 &= m_2 l_{c2} + m_b l_2 \\
a_{24} &= c_5\cos(x_1 + x_3) & \Delta &= a_{11}a_{22} - a_{12}^2 = c_1 c_2 - c_3^2\cos^2(x_3)
\end{aligned} \tag{B.14}$$

B.3 Derivations for Costate Equations

The equation of motion (4.2) can also be written in the form:

$$\dot{x}_i(t) = A_i(x, u, t) \tag{B.15}$$

where

$$A_1(x, u) = x_2$$

$$A_2(x,u) = \frac{+1}{\Delta} \times \left\{ a_{13} \left[a_{22}(2x_2 + x_4)x_4 + a_{12}x_2^2 \right] - g(a_{14}a_{22} - a_{24}a_{12}) + (a_{22}u_1 - a_{12}u_2) \right\}$$

$$A_3(x,u) = x_4$$

$$A_4(x,u) = \frac{-1}{\Delta} \times \left\{ a_{13} \left[a_{12}(2x_2 + x_4)x_4 + a_{11}x_2^2 \right] - g(a_{14}a_{12} - a_{24}a_{11}) + (a_{12}u_1 - a_{11}u_2) \right\}$$

Using the state equations (B.15) for TLM, and the following definitions for $a_{ij}(x_1, x_3)$:

$$\begin{aligned}
a_{11}(x_3) &= c_1 + c_2 + 2c_3\cos(x_3) \\
a_{12}(x_3) &= c_2 + c_3\cos(x_3) \\
a_{13}(x_3) &= c_3\sin(x_3) \\
a_{22} &= c_2 \\
a_{14}(x_1, x_3) &= c_4\cos(x_1) + c_5\cos(x_1 + x_3) \\
a_{24}(x_1, x_3) &= c_5\cos(x_1 + x_3) \\
\Delta(x_3) &= a_{11}a_{22} - a_{12}^2 = c_1c_2 - c_3^2\cos^2(x_3)
\end{aligned}$$

the partial derivatives $\frac{\partial a_{ij}}{\partial x_i}$ are:

$$\begin{aligned}
\frac{\partial a_{11}}{\partial x_1} &= 0 & \frac{\partial a_{11}}{\partial x_3} &= -2c_3\sin(x_3) = -2a_{13} \\
\frac{\partial a_{12}}{\partial x_1} &= 0 & \frac{\partial a_{12}}{\partial x_3} &= -c_3\sin(x_3) = -a_{13} \\
\frac{\partial a_{13}}{\partial x_1} &= 0 & \frac{\partial a_{13}}{\partial x_3} &= c_3\cos(x_3) = a_{12} - a_{22} \\
\frac{\partial a_{22}}{\partial x_1} &= 0 & \frac{\partial a_{22}}{\partial x_3} &= 0 \\
\frac{\partial a_{14}}{\partial x_1} &= -c_4\sin(x_1) - c_5\sin(x_1 + x_3) & \frac{\partial a_{14}}{\partial x_3} &= -c_5\sin(x_1 + x_3) \\
\frac{\partial a_{24}}{\partial x_1} &= -c_5\sin(x_1 + x_3) & \frac{\partial a_{24}}{\partial x_3} &= -c_5\sin(x_1 + x_3) \\
\frac{\partial \Delta}{\partial x_1} &= 0 & \frac{\partial \Delta}{\partial x_3} &= 2c_3^2\sin(x_3)\cos(x_3) = 2a_{13}(a_{12} - a_{22})
\end{aligned}$$

and $\frac{\partial a_{ij}}{\partial x_2} = 0$, $\frac{\partial a_{ij}}{\partial x_4} = 0$, $\frac{\partial \Delta}{\partial x_2} = 0$, $\frac{\partial \Delta}{\partial x_4} = 0$.

Then, the partial derivatives $\frac{\partial A_j}{\partial x_i}$ in equation (4.29) can be obtained as follows:

$$\begin{aligned}
&\frac{\partial A_1}{\partial x_1} = 0 \quad \frac{\partial A_1}{\partial x_2} = 1 \quad \frac{\partial A_1}{\partial x_3} = 0 \quad \frac{\partial A_1}{\partial x_4} = 0 \\
&\frac{\partial A_2}{\partial x_1} = 0 \quad \frac{\partial A_2}{\partial x_2} = 0 \quad \frac{\partial A_2}{\partial x_3} = 0 \quad \frac{\partial A_2}{\partial x_4} = 1
\end{aligned}$$

$$\frac{\partial A_2}{\partial x_1} = -\frac{g}{\Delta} \times \left[\frac{\partial a_{14}}{\partial x_1} a_{22} - \frac{\partial a_{24}}{\partial x_1} a_{12}\right] = \frac{-g}{\Delta} \times [(a_{12} - a_{22}) c_5 \sin(x_1 + x_3) - a_{22} c_4 \sin(x_1)]$$

$$\frac{\partial A_4}{\partial x_1} = +\frac{g}{\Delta} \times \left[\frac{\partial a_{14}}{\partial x_1} a_{12} - \frac{\partial a_{24}}{\partial x_1} a_{11}\right] = \frac{g}{\Delta} \times [(a_{11} - a_{12}) c_5 \sin(x_1 + x_3) - a_{12} c_4 \sin(x_1)]$$

$$\frac{\partial A_2}{\partial x_2} = +\frac{2}{\Delta} \times a_{13}[a_{12} x_2 + a_{22} x_4]$$

$$\frac{\partial A_4}{\partial x_2} = -\frac{2}{\Delta} \times a_{13}[a_{11} x_2 + a_{12} x_4]$$

$$\frac{\partial A_2}{\partial x_4} = +\frac{2}{\Delta} \times a_{13} a_{22}[x_2 + x_4]$$

$$\frac{\partial A_4}{\partial x_4} = -\frac{2}{\Delta} \times a_{13} a_{12}[x_2 + x_4]$$

For calculating $\frac{\partial A_2}{\partial x_3}$ and $\frac{\partial A_4}{\partial x_3}$ the auxiliary variables $A_2 = \frac{N_2}{\Delta}$ and $A_4 = \frac{N_4}{\Delta}$, defined as follows, are needed.

$$N_2 = \Big\{a_{13}\left[a_{22}(2x_2 + x_4)x_4 + a_{12}x_2^2\right] - g(a_{14}a_{22} - a_{24}a_{12}) + (a_{22}u_1 - a_{12}u_2)\Big\}$$

$$N_4 = -\Big\{a_{13}\left[a_{12}(2x_2 + x_4)x_4 + a_{11}x_2^2\right] - g(a_{14}a_{12} - a_{24}a_{11}) + (a_{12}u_1 - a_{11}u_2)\Big\}$$

$$\frac{\partial A_2}{\partial x_3} = \frac{1}{\Delta^2} \times \left[\frac{\partial N_2}{\partial x_3}\Delta - \frac{\partial \Delta}{\partial x_3} N_2\right] = \frac{1}{\Delta}\frac{\partial N_2}{\partial x_3} - \frac{1}{\Delta^2}\frac{\partial \Delta}{\partial x_3} N_2$$

$$\frac{\partial N_2}{\partial x_3} = \frac{\partial a_{13}}{\partial x_3}\left[a_{22}(2x_2 + x_4)x_4 + a_{12}x_2^2\right] + a_{13}\left[\frac{\partial a_{22}}{\partial x_3}(2x_2 + x_4)x_4 + \frac{\partial a_{12}}{\partial x_3}x_2^2\right]$$
$$+ \left[\frac{\partial a_{22}}{\partial x_3}u_1 - \frac{\partial a_{12}}{\partial x_3}u_2\right] - g\left[\frac{\partial a_{14}}{\partial x_3}a_{22} + a_{14}\frac{\partial a_{22}}{\partial x_3} - \frac{\partial a_{24}}{\partial x_3}a_{12} - a_{24}\frac{\partial a_{12}}{\partial x_3}\right]$$

$$\frac{\partial A_4}{\partial x_3} = \frac{1}{\Delta^2} \times \left[\frac{\partial N_4}{\partial x_3}\Delta - \frac{\partial \Delta}{\partial x_3} N_4\right] = \frac{1}{\Delta}\frac{\partial N_4}{\partial x_3} - \frac{1}{\Delta^2}\frac{\partial \Delta}{\partial x_3} N_4$$

$$\frac{\partial N_4}{\partial x_3} = \frac{\partial a_{13}}{\partial x_3}\left[a_{12}(2x_2 + x_4)x_4 + a_{11}x_2^2\right] + a_{13}\left[\frac{\partial a_{12}}{\partial x_3}(2x_2 + x_4)x_4 + \frac{\partial a_{11}}{\partial x_3}x_2^2\right]$$
$$+ \left[\frac{\partial a_{12}}{\partial x_3}u_1 - \frac{\partial a_{11}}{\partial x_3}u_2\right] - g\left[\frac{\partial a_{14}}{\partial x_3}a_{12} + a_{14}\frac{\partial a_{12}}{\partial x_3} - \frac{\partial a_{24}}{\partial x_3}a_{11} - a_{24}\frac{\partial a_{11}}{\partial x_3}\right]$$

substituting the partial derivatives $\frac{\partial a_{ij}}{\partial x_3}$ and $\frac{\partial \Delta}{\partial x_3}$ into the above relations results:

$$\frac{\partial A_2}{\partial x_3} = +\frac{1}{\Delta} \times \Big\{(a_{12} - a_{22})\left[a_{22}(2x_2 + x_4)x_4 + a_{12}x_2^2\right] - a_{13}^2 x_2^2$$
$$+ a_{13}u_2 - g\left[(a_{12} - a_{22})c_5 sin(x_1 + x_3) + a_{13}a_{24}\right]\Big\}$$
$$-\frac{1}{\Delta^2} \times 2a_{13}(a_{12} - a_{22})\Big\{a_{13}\left[a_{22}(2x_2 + x_4)x_4 + a_{12}x_2^2\right]$$
$$+ a_{22}u_1 - a_{12}u_2 - g\left[a_{14}a_{22} - a_{24}a_{12}\right]\Big\}$$

$$\frac{\partial A_4}{\partial x_3} = -\frac{1}{\Delta} \times \Big\{(a_{12} - a_{22})\left[a_{12}(2x_2 + x_4)x_4 + a_{11}x_2^2\right] - a_{13}^2[(x_2 + x_4)^2 + x_2^2]$$
$$+ a_{13}(2u_2 - u_1) - g\left[(a_{11} - a_{12})c_5 sin(x_1 + x_3) + a_{13}(2a_{24} - a_{14})\right]\Big\}$$
$$+\frac{1}{\Delta^2} \times 2a_{13}(a_{12} - a_{22})\Big\{a_{13}\left[a_{12}(2x_2 + x_4)x_4 + a_{11}x_2^2\right]$$
$$+ a_{12}u_1 - a_{11}u_2 - g\left[a_{14}a_{12} - a_{24}a_{11}\right]\Big\}$$

B.4 Details of Shooting Method

In general the initial conditions may be given as:

$$x_i(t_0) = c_i, \qquad i = 1, \ldots, r \tag{B.16}$$

also the final conditions are given as:

$$x_{i_m}(t_f) = c_{i_m}, \qquad m = 1, \ldots, n - r \tag{B.17}$$

Clearly $r < n$ for having the boundary conditions on both ends. In case of TLM, $n = 8$ and $r = 4$. The subscripts on the final conditions are c_{i_m} to include the boundary conditions which are not disjoint. For example if $n = 6$, and $x_1(t_0)$, $x_2(t_0)$, $x_3(t_0)$, $x_2(t_f)$, $x_4(t_f)$, $x_6(t_f)$ are specified as boundary conditions, then $i_1 = 2$, $i_2 = 4$, $i_3 = 6$, so $x_{i_1}(t_f) = x_2(t_f)$, $x_{i_2}(t_f) = x_4(t_f)$, $x_{i_3}(t_f) = x_6(t_f)$.

For stability of the SM a method is said to be convergent if $x_i \rightarrow x(t_i)$ when $h \rightarrow o$ for all starting values which approach the true initial conditions as $h \rightarrow o$. Here h is the integration step size, and x_i is true solution. Partial stability may occur in some numerical methods like Runge-Kutta, when the numerical solution approximates the true solution for step size h below a specific value h_0. The method however fails when h is greater than h_0.

B.4.1 Method of Adjoints

One of the most important shooting methods is the method of adjoints. If we have a **linear nth order TPBVP** of the form

$$\dot{x}(t) = A(t)x + f(t) \tag{B.18}$$

with the initial conditions (B.16) and the final conditions (B.17). We can form the following adjoint equations

$$\dot{y}(t) = -A^T(t)y \tag{B.19}$$

where y is the n vector of adjoint variables. The boundary conditions of adjoint system are related to the boundary conditions of the original system (B.18) by an identity. With simple calculus one can find the following relationship between the boundary conditions of systems (B.18) and (B.19), i.e.; multiplying (B.18) by y, and (B.19) by x and adding the results, then integrating the sum over $[t_0, t_f]$.

$$\sum_{i=1}^{n} y_i(t_f)x_i(t_f) - \sum_{i=1}^{n} y_i(t_0)x_i(t_0) = \int_{t_0}^{t_f} \sum_{i=1}^{n} y_i(t)f_i(t)dt \tag{B.20}$$

This is called the fundamental identity for the method of adjoints. For any choice of boundary conditions of y variable, this is a linear relationship between the unspecified $x_i(t_0)$. We can integrate the adjoint equations $(n-r)$ times backward with the following boundary conditions.

$$y_i^{(m)}(t_f) = \left\{ \begin{array}{ll} 1 & i = i_m \\ 0 & i \neq i_m \end{array} \right\} \qquad m = 1, \ldots, n-r \tag{B.21}$$

where (m) is for the mth backward integration of the adjoint equations and i_m is for the specified final conditions $x_i^{(m)}(t_f)$ of (B.17). Using (B.21) one can write (B.20) as

$$\sum_{i=r+1}^{n} y_i^{(m)}(t_0)x_i(t_0) = x_{i_m}(t_f) - \sum_{i=1}^{r} y_i^{(m)}(t_0)x_i(t_0) - \int_{t_0}^{t_f} \sum_{i=1}^{n} y_i^m(t)f_i(t)dt \tag{B.22}$$

This is a set of $(n-r)$ linear algebraic relations for $(n-r)$ unknown initial conditions which can be solved for $x_{r+1}(t_0), x_{r+2}(t_0), \ldots, x_n(t_0)$. The $y_i^{(m)}(t_0)$ in LHS of (B.22) is known from backward integration of adjoint equations with (B.21) as boundary conditions. As it can be seen, the unknown initial conditions of linear TPBVP can be obtained in one step by the method of adjoints.

For **non linear TPBVP** of the form

$$\dot{x}(t) = a(x, t) \tag{B.23}$$

with the initial conditions (B.16) and the final conditions (B.17) we have to use iterative approach to find the unknown initial conditions. If $x(t)$ is the solution of

(B.23), and $\delta x(t)$ is the first variation of this solution, then $x+\delta x$ is the nearby solution of (B.23) as well. Substituting the nearby solution in (B.23), using Taylors series expansion, and cancelling equal terms from the sides, give the following variational equations for δx

$$\delta \dot{x}_i = \sum_{j=1}^{n} \frac{\partial a_i}{\partial x_j} \delta x_j \qquad i = 1, \ldots, n \tag{B.24}$$

The adjoint equations for the variational equations can be written similar to the linear TPBVP (B.19) as

$$\dot{y}_i = -\sum_{j=1}^{n} \frac{\partial a_j}{\partial x_i} y_j \qquad i = 1, \ldots, n \tag{B.25}$$

Here equations (B.24) and (B.25) are linear ODE's. Then similar to (B.20) the fundamental identity of the method of adjoints for these systems would be

$$\sum_{i=1}^{n} y_i(t_f)\delta x_i(t_f) - \sum_{i=1}^{n} y_i(t_0)\delta x_i(t_0) = 0 \tag{B.26}$$

For iteration (k), the variation of x_i can be calculated as

$$\delta x_i^{(k)} = x_{i_{true}} - x_i^{(k)} \qquad i = 1, \ldots, n \qquad k = 0, 1, \ldots \tag{B.27}$$

Since the r initial conditions are specified, we have

$$\delta x_i^{(k)}(t_0) = 0 \qquad i = 1, \ldots, r \tag{B.28}$$

also, for r final conditions which are not specified we have

$$\delta x_{i_m}^{(k)}(t_f) = 0 \qquad m = n - r + 1, \ldots, n \tag{B.29}$$

Also for $(n - r)$ specified final conditions we use the following

$$\delta x_{i_m}^{(k)}(t_f) = x_{i_m}(t_f) - x_{i_m}^{(k)}(t_f) \qquad m = 1, \ldots, n - r \tag{B.30}$$

Like before, the fundamental identity (B.26) can be used to obtain the corrections to the unknown initial conditions $x_i(t_0)$. If the Kronecker delta (B.21) be used as

terminal conditions of adjoint variables $y_i(t_f)$, then (B.26) can be rewritten as

$$\begin{bmatrix} y_{r+1}^{(1)}(t_0) & \cdots & y_{r+1}^{(1)}(t_0) \\ \cdot & \cdots & \cdot \\ y_{r+1}^{(n-r)}(t_0) & \cdots & y_{r+1}^{(n-r)}(t_0) \end{bmatrix} \begin{Bmatrix} \delta x_{r+1}^{(k)}(t_0) \\ \cdot \\ \delta x_n^{(k)}(t_0) \end{Bmatrix} = \begin{Bmatrix} \delta x_{i_1}^{(k)}(t_f) \\ \cdot \\ \delta x_{i_{n-r}}^{(k)}(t_f) \end{Bmatrix} \tag{B.31}$$

The correction terms $\delta x_i^{(k)}(t_0)$ can be calculated now, since everything else in (B.31) are known. The unknown initial conditions then are

$$x_i^{(k+1)}(t_0) = x_i^{(k)}(t_0) + \delta x_i^{(k)}(t_0) \qquad i = r+1, \ldots, n \tag{B.32}$$

B.4.2 Newton-Raphson Method

It should be emphasised here that the method of adjoints is identical with the *Newton-Raphson Method* for solving non linear TPBVP. If we look back at the function of missing final distance L_m, $m = 1, \ldots, n-r$ which should be set to zero to enforce the final conditions.

$$L_m = x_{i_m}(t_f) - x_{i_m}^{(k)}(t_f) \qquad i = r+1, \ldots, n \tag{B.33}$$

It can be written as function of unknown initial conditions as

$$L_m(x_i)_{t_0} = 0 \qquad m = 1, \ldots, n-r, \;\; i = r+1, \ldots, n \tag{B.34}$$

Then the correction terms based on the Newton-Raphson method would be

$$x_i^{(k+1)}(t_0) = x_i^{(k)}(t_0) - \left(\frac{\partial L_m}{\partial x_i^{(k)}}\right)_{t_0}^{-1} L_m(x_i^{(k)})_{t_0} \tag{B.35}$$

Comparing this equation with the similar equation in the method of adjoints (B.32), it can be concluded that

$$y_i^{(m)}(t_0) = -\left(\frac{\partial L_m}{\partial x_i^{(k)}}\right)_{t_0}^{-1} \tag{B.36}$$

$$\delta x_{i_m}^{(k)}(t_f) = L_m(x_i^{(k)})_{t_0} \tag{B.37}$$

$$\delta x_i^{(k)}(t_0) = -\left(\frac{\partial L_m}{\partial x_i^{(k)}}\right)_{t_0}^{-1} L_m(x_i^{(k)})_{t_0} \tag{B.38}$$

Details of the proof can be found in [54].

B.4.3 The Continuation Method

Computational experience shows that if the method of adjoints is going to converge, it would do so within the first 10 iterations. Also it is better to choose smaller time step and employ fewer iterations than using larger time steps and employ more iterations.

When the SM is diverging for the TPBVP which are too sensitive to the unknown initial conditions, the *continuation method* may help [54]. When a problem cannot be solved by the regular SM, it may be solved in conjunction with continuation. The idea of which is to solve a sequence of TPBVP for a sequence of time $[t_0, t_1]$, $[t_0, t_2]$,...,$[t_0, t_f]$ where $t_0 < t_1 < \ldots < t_f$. The initial conditions and final conditions of the sequence TPBVP are the same. The initial conditions determined by the solution of the TPBVP when $t \in [t_0, t_1]$ are taken as the initial guess of unknown initial conditions for the TPBVP when $t \in [t_0, t_2]$. This method allows for avoiding numerical overflow or underflow in computer for the problem with sensitive initial conditions.

B.5 ANSYS Input File

A typical ANSYS input file for maneuver of a Two-Link Flexible Manipulator of Section 6.3 is as follows.

```
/core,,,1000000
/prep7
/title,Example-4(2D-Robot, rest to rest, flexible arms(Mg), no gravity)
et,1,3
et,3,21,,,3

r1=.005
a1=3.1415926535*(r1**2)
in1=(3.1415926535/4.0)*(r1**4)
h1=2*r1                !(circular cross section)
r,1,a1,in1,h1
r,3,0.5,0              !(end mass without inertia)
ex,1,69.0e+9           !(flexible arms, Aluminium 6061-T6)
dens,1,7800            !(Steel density)
nuxy,1,.3
ts1=3.962303e-2
ts2=6.165361e-1
ts3=6.645379e-1
tf=1.233072
tds=15.0
itot=20000
ddt=0.0015
dts2=(ts2-ts1)/tds+ts1
dts3=(ts3-ts2)/tds+ts2
dtf=(tf-ts3)/tds+ts3
is1=(ts1-0.001)/tf*itot
is2=(ts2-dts2)/tf*itot
is3=(ts3-dts3)/tf*itot
if=(tf-dtf)/tf*itot
ids2=is2/tds
ids3=is3/tds
idf=if/tds
itp=30+is1+is2+is3+if+ids2+ids3+idf
kan,4
kay,5,2
kay,6,1
kay,9,1
gamma,0.005
c***acel,,9.81           !(no gravity)

n,1
n,11,.4
fill
n,12,.4
n,22,.65
fill
type,1
real,1
e,1,2
egen,10,1,1
e,12,13
egen,10,1,11
type,3
real,3
e,22
cp,1,uy,11,12
```

```
cp,2,ux,11,12
d,1,ux
d,1,uy
kbc,1
time,0.001
f,1,mz,+.25
f,11,mz,+.1
f,12,mz,-.1
iter,20,0,1
lwrite

kbc,1
time,ts1,ddt
f,1,mz,+.25
f,11,mz,+.1
f,12,mz,-.1
iter,-is1,0,1
lwrite

kbc,1
time,dts2,(ddt/3.0)
f,1,mz,+.25
f,11,mz,-.1
f,12,mz,+.1
iter,-ids2,0,1
lwrite

kbc,1
time,ts2,ddt
f,1,mz,+.25
f,11,mz,-.1
f,12,mz,+.1
iter,-is2,0,1
lwrite

kbc,1
time,dts3,(ddt/3.0)
f,1,mz,-.25
f,11,mz,-.1
f,12,mz,+.1
iter,-ids3,0,1
lwrite

kbc,1
time,ts3,ddt
f,1,mz,-.25
f,11,mz,-.1
f,12,mz,+.1
iter,-is3,0,1
lwrite

kbc,1
time,dtf,(ddt/3.0)
f,1,mz,-.25
f,11,mz,+.1
f,12,mz,-.1
iter,-idf,0,1
lwrite

kbc,1
time,tf,ddt
f,1,mz,-.25
f,11,mz,+.1
f,12,mz,-.1
```

```
iter,-if,0,1
lwrite

afwrite
finish

/input,27
finish

/post26
numvar,25
disp,2,1,rotz,ph1b
disp,3,11,rotz,ph1e
disp,4,12,rotz,te2b
disp,5,22,rotz,te2e
disp,6,1,ux,zero
add,7,2,4,,ph2b,,,-1
add,8,3,5,,ph2e,,,-1
deriv,9,2,1,,dp1b
deriv,10,7,1,,dp2b
deriv,11,3,1,,dp1e
deriv,12,8,1,,dp2e
nforce,13,1,1,mz,mz1b
nforce,14,11,12,mz,mz2b
add,15,13,6,,U1,,,-1
add,16,14,6,,U2,,,-1
deriv,17,4,1,,dt2b
deriv,18,5,1,,dt2e
disp,19,22,ux,ux2e
disp,20,22,uy,uy2e
int1,21,6,1,,zero,auxl,,,,0.65
add,22,19,21,,uxee
lines,itp
/output,ocl8fb,out   !(base displacements)
prvar,2,9,7,10,15,16
/output
/output,ocl8fe,out   !(end displacements)
prvar,3,11,8,12,22,20
/output
finish
```

Appendix C

Computer Code for the Forward-Backward Method

The computer code OC2ADC13 that implements the Forward-Backward Method (FBM) of Section 5.4 for time-optimal control of Two-Link Manipulators (TLM) is included and explained here. This program, which uses the *Fortran* programming language, contains a main body and ten subroutines, and in total is **2506** lines long. One of the main subroutines of this code, DBVPMS, which solves a Two Point Boundary Value Problem (TPBVP) with known RHS for ODE's using multiple shooting method is taken from IMSL/LIB. The code OC2ADC13, basically is a driver program for DBVPMS.

C.1 Input Files

The input data files SYSD.IN, SYSD2.IN, SYSD3.IN for initialising the FBM are:

SYSD.IN

t_0	t_f^a	$BTOL$	$DTOL$	$MAXIT$

SYSD2.IN

u_1^{fd}	u_2^{fd}	u_1^{bd}	u_2^{bd}	
$SIGMA$	$ALFA$	$BETA$	$NITR$	
$t_{s1}^{(0)}$	$\Delta t_s{}^{(0)}$			
$KCONT$	$SIGM1$			
g_r				
U_1^-	U_1^+	U_2^-	U_2^+	
l_1	l_2	l_{c1}	l_{c2}	
m_1	m_2	m_a	m_b	
I_1	I_2	I_0	I_a	I_b

SYSD3.IN

x_{10}	x_{20}	x_{30}	x_{40}
x_{1f}	x_{2f}	x_{3f}	x_{4f}
$p_1(t_0)$	$p_2(t_0)$	$p_3(t_0)$	$p_4(t_0)$
$p_1(t_f)$	$p_2(t_f)$	$p_3(t_f)$	$p_4(t_f)$

Note that specifying $p_i(t_0)$, $p_i(t_f)$, $i = 1, \ldots, 4$ is necessary, if one wishes to skip Iteration One of the FBM and go to the Subsequent Iterations from the beginning.

Some of the parameters which are not defined before are:

BTOL: Boundary conditions error tolerance

DTOL: Differential equation error tolerance

MAXIT: Maximum number of Newton iterations allowed

SIGMA: The initial value for $e^*(t, t_f)$ of Equation (5.32)

ALFA: Program constant, use +1.0

BETA: Program constant, use -1.0

NITR: Maximum number of iterations of the main loop

KCONT: Continuity key $12 \rightarrow e^*_{12}$, $1234 \rightarrow e^*$ of Equation (5.32)

SIGM1: A positive small number such as 0.00001

The input files SYSD.IN, SYSD2.IN, SYSD3.IN for Example Two of Section 6.1.2 are:

```
SYSD.IN
-------
0.0  0.205296  5.0E-4  5.0E-4  41

SYSD2.IN
--------
-10.0  -5.0  +10.0  +5.0
0.050001  +1.0  -1.0  19
0.072  0.018
12  0.00001
9.81
-10.0  +10.0  -5.0  +5.0
0.2 0.2 0.1 0.1
1.0 1.0 0.0 0.0
4.167E-03 4.167E-03  0.0  0.0  0.0
```

```
SYSD3.IN
--------
1.047   0.0   0.0   0.0
0.0     0.0   0.0   0.0
0.0     0.0   0.0   0.0
0.0     0.0   0.0   0.0
```

C.2 Output Files

The code creates several output files, some of the important ones are:

File name	Contents
T2D15.DAT	States in Step 1 in each iteration
T2D15C.DAT	Costates in Step 1 in each iteration
T2DS.DAT	States and controls at the end of each iteration
T2DC.DAT	Costates and switch functions at the end of each iteration
TSF1.DAT	States and controls at the end of first iteration
TCF1.DAT	Costates and switch functions at the end of first iteration
TSF2.DAT	States and controls at the end of second iteration
TCF2.DAT	Costates and switch functions at the end of second iteration
TSF3.DAT	States and controls at the end of third iteration
TCF3.DAT	Costates and switch functions at the end of third iteration
T2D7.DAT	First switch time t_{s1}, final time t_f, and error e^* in each iteration
T2D8.DAT	Switch times t_{si}, final time t_f, and error e^* in each iteration
T2D16.DAT	Input data, $p(t_0)$, $p(t_f)$, t_{si}, t_f, and e^* in each iteration

However, there are other output files such as

T2D3.DAT, T2D5.DAT, T2D10.DAT, T2D3C.DAT, T2D11.DAT, T2D12.DAT , T2D21.DAT, T2D22.DAT, T2D23.DAT, T2D27.DAT, T2D28.DAT

which prints different parameters or states, costates, and switch functions in different stages of the FBM.

C.3 Weight Functions W_i

The weight functions w_i in Equations (5.32) are to accommodate the difference of the dimensions of the states. They are calculated and selected as follows. For convenience Equation (5.30), (5.32) is repeated here.

$$e(t,t_f) = \left\{ \begin{array}{c} e_{12}(t,t_f) \\ \\ e_{34}(t,t_f) \end{array} \right\} \quad e_{12}(t,t_f) = \left[\begin{array}{c} e_1(t,t_f) \\ \\ e_2(t,t_f) \end{array} \right] \quad e_{34}(t,t_f) = \left[\begin{array}{c} e_3(t,t_f) \\ \\ e_4(t,t_f) \end{array} \right] \tag{C.1}$$

$$e_{12}(t,t_f) = \left\{ \begin{array}{l} \left| x_1(x_0,u_i^{fd},t) - x_1(x_f,u_i^{bd},t_f-t) \right| \\ \\ \left| x_2(x_0,u_i^{fd},t) + x_2(x_f,u_i^{bd},t_f-t) \right| \end{array} \right\}$$

$$e_{34}(t,t_f) = \left\{ \begin{array}{l} \left| x_3(x_0,u_i^{fd},t) - x_3(x_f,u_i^{bd},t_f-t) \right| \\ \\ \left| x_4(x_0,u_i^{fd},t) + x_4(x_f,u_i^{bd},t_f-t) \right| \end{array} \right\}$$

$$e^* = \left(\frac{1}{4} \sum_{i=1}^{4} e_i^{*2} \right)^{1/2} \qquad e_i^* = e_i \times w_i, \quad i = 1,\ldots,4 \tag{C.2}$$

The weight functions w_i^* are calculated as follows:

$$w_i^{fd} = max|x_i^{fd}(t)|, \quad i = 1,\ldots,4$$

$$w_i^{bd} = max|x_i^{bd}(t)|, \quad i = 1,\ldots,4$$

$$\bar{w}_i = max(w_i^{fd}, w_i^{bd}), \quad i = 1,\ldots,4$$

$$w_i = \frac{\bar{w}_1}{\bar{w}_i}, \quad i = 1,\ldots,4$$

Note that w_i is non-dimensional and usually is $0 < w_i \leq 1$. Presence of weight functions w_i in (C.2) insures that error norm e^* is not dominated by magnitude of one or more states, because of the difference in the magnitudes of rotations and angular velocities. In other words, error vector e^* is considered a dimensionless vector.

C.4 The Listing of the Program OC2ADC13

The computer code OC2ADC13 that implements the FBM for time-optimal control of two-link manipulators is as follows.

```
CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
CCCCC67---STARTING A NEW STEP----------49----------72
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
INTEGER LDY,NEQNS,NMAX,LDY2,NEQNS2
C1CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------.72
      PARAMETER (NEQNS=4,NMAX=301,LDY=NEQNS,NEQNS2=2*NEQNS,LDY2=2*LDY)
      INTEGER I,MAXIT,NINIT,IS,IT,NITR,KCONT,II,KK
DIMENSION KS(10),KSB(10)
DOUBLE PRECISION
     1 FCNEQNL,FCNEQNU,FCNEQNC
     &  ,FCNBCSL,FCNBCSU,FCNBCLI,FCNBCUI
     &  ,FCNBC,FCNJACB
     2 ,FLOAT,SQRT,ABS,SIN,COS,MAX,SSET,DTOL,BTOL,XLEFT,XLEND,XRIGHT
     &  ,XREND,E1,E2,E3,E4,EB1,EB2,EB3,EB4,EB1234,EB,EBF,E1F,E2F
     &  ,E3F,E4F,WFL1,WFL2,WFL3,WFL4,WFU1,WFU2,WFU3,WFU4,SIGMA,EBS0,EBS
     &  ,EBI1,EBI2,EBI3,EBI4,EBI1234,EBI,ES1F,ES2F,EBIS,EBFI,DX
     3 ,X(2*NMAX),XL(2*NMAX),XU(2*NMAX),XLC(2*NMAX),XUC(2*NMAX)
     &  ,XLI(2*NMAX),XUI(2*NMAX),XB(2*NMAX),XF(2*NMAX),XFI(2*NMAX)
     &  ,XFR(2*NMAX),XSW(10,20),EBJ(NMAX*NMAX)
     4 ,YSS(LDY2,20),YSST(LDY2,20),YL(LDY2,2*NMAX),YU(LDY2,2*NMAX)
     &  ,YLC(LDY2,2*NMAX),YUC(LDY2,2*NMAX),YF(LDY2,2*NMAX)
     &  ,YFI(LDY2,2*NMAX),YFR(LDY2,2*NMAX)
     &  ,YLI(LDY2,2*NMAX),YUI(LDY2,2*NMAX),YB(LDY2,2*NMAX)
     5 ,GC1(2*NMAX),GC2(2*NMAX),GS1(2*NMAX),GS2(2*NMAX)
     6 ,UL1(2*NMAX),UL2(2*NMAX),UU1(2*NMAX),UU2(2*NMAX)
     &  ,UF1(2*NMAX),UF2(2*NMAX),UCF1(2*NMAX),UCF2(2*NMAX)
     7 ,XSF(10,NMAX),XS1T(20),XS12(20),XS3TT(20),XS3C(20),XTT1
     &  ,XTT(20),XT(20),XBS(10,20)
     &  ,SIGNT(10,20),YSW(LDY2,10),YBS(LDY2,10)
     8 ,H(10,2*NMAX)
     9 ,L1,L2,M1,M2,MA,MB,LC1,LC2,RI1,RI2,X1,X2,X3,X4,P1,P2,P3,P4
     &  ,A11,A12,A13,A22,A14,A24,DE,T1,T2,ALFA,BETA
C---67---COMMON PARAMETERS----------72
      INTEGER J11,KFJ,NSW,IEQ,IBC,IJB
DIMENSION KSW(10)
DOUBLE PRECISION
     +YBCL(8),YBCR(8),XS1(20),XS2(20),XS3(20),SIGN(8,20)
     +,YSCL(8,20),YSCU(8,20),YSLB(8,20),YSUB(8,20),UFW(10,2)
     +,C1,C2,C3,C4,C5,GR,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1
COMMON /PARAM/ YBCL,YBCR,XS1,XS2,XS3,SIGN
     +,YSCL,YSCU,YSLB,YSUB,UFW,C1,C2,C3,C4,C5,GR
     +,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1,KSW,J11,KFJ,NSW,IEQ,IBC,IJB
C---67---COMMON PARAMETERS----------72

INTRINSIC FLOAT,SQRT,ABS,SIN,COS,MAX
EXTERNAL DBVPMS,SSET
     1 ,FCNEQNL,FCNEQNU,FCNEQNC
     &  ,FCNBCSL,FCNBCSU,FCNBCLI,FCNBCUI
     &  ,FCNBC,FCNJACB
      COMMON /WORKSP/ RWKSP
      REAL RWKSP(64328)
      CALL IWKIN(64328)

OPEN(UNIT=1, FILE='t2ds.dat',STATUS= 'UNKNOWN')
OPEN(UNIT=3, FILE='t2d3.dat',STATUS= 'UNKNOWN')
OPEN(UNIT=5, FILE='t2d5.dat',STATUS= 'UNKNOWN')
OPEN(UNIT=7, FILE='t2d7.dat',STATUS= 'UNKNOWN')
OPEN(UNIT=8, FILE='t2d8.dat',STATUS= 'UNKNOWN')
OPEN(UNIT=10, FILE='t2d10.dat',STATUS= 'UNKNOWN')
OPEN(UNIT=15, FILE='t2d15.dat',STATUS= 'UNKNOWN')
OPEN(UNIT=16, FILE='t2d16.dat',STATUS= 'UNKNOWN')
OPEN(UNIT=18, FILE='t2d15c.dat',STATUS= 'UNKNOWN')
OPEN(UNIT=19, FILE='t2d3c.dat',STATUS= 'UNKNOWN')
OPEN(UNIT=20, FILE='t2dc.dat',STATUS= 'UNKNOWN')
OPEN(UNIT=27, FILE='t2d27.dat',STATUS= 'UNKNOWN')
OPEN(UNIT=28, FILE='t2d28.dat',STATUS= 'UNKNOWN')

WF1=0.0
WF2=0.0
WF3=0.0
WF4=0.0

OPEN(UNIT=13, FILE='sysd2.in',STATUS='OLD')
READ(13,*) SIGN(1,1),SIGN(2,1),SIGN(3,1),SIGN(4,1),SIGMA
   &,ALFA,BETA,NITR,XS2(1),XS3(1),KCONT,SIGM1
```

```
      READ(13,*) GR,T1MIN,T1MAX,T2MIN,T2MAX
      READ(13,*) L1,L2,LC1,LC2,M1,M2,MA,MB,RI1,RI2
CLOSE(UNIT=13)

      C1=M1*LC1*LC1+RI1 + (M2+MA+MB)*L1*L1
      C2=M2*LC2*LC2+RI2 + MB*L2*L2
      C3=(M2*LC2+MB*L2)*L1
      C4=M1*LC1 + (M2+MA+MB)*L1
      C5=(M2*LC2+MB*L2)

WRITE(16,*) SIGN(1,1)
WRITE(16,*) SIGN(2,1)
WRITE(16,*) SIGN(3,1)
WRITE(16,*) SIGN(4,1)
WRITE(16,*) SIGMA
WRITE(16,*) ALFA
WRITE(16,*) BETA
WRITE(16,*) NITR
WRITE(16,*) XS2(1)
WRITE(16,*) XS3(1)
WRITE(16,*) KCONT
WRITE(16,*) SIGM1
      WRITE(16,*) GR,T1MIN,T1MAX,T2MIN,T2MAX
      WRITE(16,*) L1,L2,LC1,LC2
      WRITE(16,*) M1,M2,MA,MB
      WRITE(16,*) RI1,RI2

C2CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
OPEN(UNIT=17, FILE='sysd3.in',STATUS='OLD')
DO J2=1,NEQNS
READ(17,*) YSCL(J2,1)
END DO
DO J2=1,NEQNS
READ(17,*) YSCU(J2,1)
END DO
DO J2=NEQNS+1,NEQNS2
READ(17,*) YSCL(J2,1)
END DO
DO J2=NEQNS+1,NEQNS2
READ(17,*) YSCU(J2,1)
END DO
CLOSE(UNIT=17)

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
CCCCC67---STARTING THE ITERATION LOOP-------49----------72
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
DO 1450 J11=1,NITR

OPEN(UNIT=2, FILE='sysd.in',STATUS='OLD')
READ(2,*) XLEND,XREND,BTOL,DTOL,MAXIT
CLOSE(UNIT=2)
XLEFT=XLEND
XRIGHT=XREND
    WRITE(5,'(20X,A,I5/20X,A)')'ITR=',J11,'-------'
    WRITE(5,'(5X,A,E15.6,5X,A,E15.6/)')'XLFT0=',XLEFT,'XRGTF=',XRIGHT
IS=0
JS=0

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
CCCCC67---INTEGRATION   OF  STATES-------49----------72
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72

    IF(J11.GE.2) THEN
      WRITE(16,'(18X,A,18X,A)') 'YSCL','YSCU'
DO J2=1,NEQNS
YSCL(J2,J11)=YSCL(J2,1)
YSCU(J2,J11)=YSCU(J2,1)
      WRITE(16,*) YSCL(J2,J11),YSCU(J2,J11)
END DO
DO J2=NEQNS+1,NEQNS2
YSCL(J2,J11)=YF(J2,1)
YSCU(J2,J11)=YFR(J2,NFR)
      WRITE(16,*) YSCL(J2,J11),YSCU(J2,J11)
END DO
    END IF
IEQ=0
IBC=0
IJB=0

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
CCCCC67---INTEGRATION OF STATES FROM XT0-XTF---49----------72
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
NINIT=NMAX
C DEFINE THE SHOOTING POINTS
DO 10 I=1, NINIT
  X(I)=XLEFT+FLOAT(I-1)/FLOAT(NINIT-1)*(XRIGHT-XLEFT)
    IF(J11.EQ.1) THEN
```

```
   CALL SSET (NEQNS2, 0.0, YL(1,I), 1)
   CALL SSET (NEQNS2, 0.0, YU(1,I), 1)
      END IF
10 CONTINUE

    IF(J11.GE.2) THEN
      DO I=1,NFL
  DO J=1,NEQNS2
YL(J,I)=YF(J,I)
  END DO
END DO
      DO I=1,NFR
  I2=NFR-I+1
  DO J=1,NEQNS2
YU(J,I)=YFR(J,I2)
  END DO
END DO
DO J=1,NEQNS2
YL(J,1)=YSCL(J,J11)
YU(J,1)=YSCU(J,J11)
END DO
    END IF

C SOLVE PROBLEM
KFJ=10
CALL DBVPMS (FCNEQNL,FCNJACB,FCNBCSL,NEQNS2,XLEFT,XRIGHT,DTOL,
   &     BTOL,MAXIT,NINIT,X,YL,LDY2,NMAX,NFL,XL,YL,LDY2)
OPEN(UNIT=11, FILE='t2d11.dat',STATUS= 'UNKNOWN')
WRITE(11,'(5X,A,6X,A,5X,A,6X,A,5X,A,8X,A,8X,A/)')
   &  'T','X1=PH1','X2=DPH1','X3=PH2','X4=DPH2','U1','U2'

WFL1=0.0
WFL2=0.0
WFL3=0.0
WFL4=0.0
DO 111 I=1,NFL
C6CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
      X3=YL(3,I)
      P2=YL(6,I)
      P4=YL(8,I)
      A11=C1+C2+2*C3*COS(X3)
      A12=C2+C3*COS(X3)
      A13=C3*SIN(X3)
      A22=C2
      DE=A11*A22-A12*A12
      GS1(I)=(P2*A22-P4*A12)/DE
      GS2(I)=(-P2*A12+P4*A11)/DE
      IF(GS1(I).LE.0.0) THEN
           UL1(I)=T1MAX
      ELSE
           UL1(I)=T1MIN
      END IF
      IF(GS2(I).LE.0.0) THEN
           UL2(I)=T2MAX
      ELSE
           UL2(I)=T2MIN
      END IF

IF(J11.EQ.1) THEN
IF(ABS(XS2(J11)).LE.SIGM1) THEN
  UL2(I)=SIGN(2,J11)
ELSE IF(ABS(XS2(J11)).GT.SIGM1) THEN
  IF(XL(I).LE.XS2(J11)) THEN
UL2(I)=SIGN(2,J11)
  ELSE
UL2(I)=-SIGN(2,J11)
  END IF
END IF
UL1(I)=SIGN(1,J11)
END IF
C1CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
WRITE(11,9992) XL(I),(YL(J2,I),J2=1,NEQNS),UL1(I),UL2(I)
WFL1=MAX(ABS(YL(1,I)),WFL1)
WFL2=MAX(ABS(YL(2,I)),WFL2)
WFL3=MAX(ABS(YL(3,I)),WFL3)
WFL4=MAX(ABS(YL(4,I)),WFL4)
111 CONTINUE
CLOSE(UNIT=11)

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
CCCCC67---INTEGRATION OF STATES FROM XTF-XT0---49----------72
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
     WRITE(5,'(5X,A,E15.6,5X,A,E15.6/)')'XLFT0=',XLEFT,'XRGTF=',XRIGHT
KFJ=20
  CALL DBVPMS (FCNEQNU,FCNJACB,FCNBCSU,NEQNS2,XLEFT,XRIGHT,DTOL,
   &     BTOL,MAXIT,NINIT,X,YU,LDY2,NMAX,NFU,XU,YU,LDY2)
OPEN(UNIT=12, FILE='t2d12.dat',STATUS= 'UNKNOWN')
```

```
WRITE(12,'(5X,A,6X,A,5X,A,6X,A,5X,A,8X,A,8X,A/)')
    &  'T','X1=PH1','X2=DPH1','X3=PH2','X4=DPH2','U1','U2'
WFU1=0.0
WFU2=0.0
WFU3=0.0
WFU4=0.0
DO 113 I=1,NFU
C6CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
      X3=YU(3,I)
      P2=YU(6,I)
      P4=YU(8,I)
      A11=C1+C2+2*C3*COS(X3)
      A12=C2+C3*COS(X3)
      A13=C3*SIN(X3)
      A22=C2
      DE=A11*A22-A12*A12
      GS1(I)=(P2*A22-P4*A12)/DE
      GS2(I)=(-P2*A12+P4*A11)/DE
      IF(GS1(I).LE.0.0) THEN
           UU1(I)=T1MAX
      ELSE
           UU1(I)=T1MIN
      END IF
      IF(GS2(I).LE.0.0) THEN
           UU2(I)=T2MAX
      ELSE
           UU2(I)=T2MIN
      END IF

IF(J11.EQ.1) THEN
IF(ABS(XS3(J11)).LE.SIGM1) THEN
  UU2(I)=SIGN(4,J11)
ELSE IF(ABS(XS3(J11)).GT.SIGM1) THEN
  IF(XU(I).LE.XS3(J11)) THEN
UU2(I)=-SIGN(4,J11)
  ELSE
UU2(I)=SIGN(4,J11)
  END IF
END IF
UU1(I)=SIGN(3,J11)
END IF
C1CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
WRITE(12,9992) XU(I),YU(1,I),-YU(2,I),YU(3,I),-YU(4,I)
    &,UU1(I),UU2(I)
WFU1=MAX(ABS(YU(1,I)),WFU1)
WFU2=MAX(ABS(YU(2,I)),WFU2)
WFU3=MAX(ABS(YU(3,I)),WFU3)
WFU4=MAX(ABS(YU(4,I)),WFU4)
113 CONTINUE
CLOSE(UNIT=12)

CCCCC67-9-------9-------9-------9-------9-------9-------9-------72
CCCCC67---FIRSET MINIMIZATION FOR PLACE OF XS1(CONTINUITY OF S&C)---72
CCCCC67-9-------9-------9-------9-------9-------9-------72
EBF=10000.0
EBFI=10000.0
WRITE(7,'(20X,A,I5/20X,A)')'ITR=',J11,'-------'
WRITE(10,'(20X,A,I5/20X,A)')'ITR=',J11,'-------'
WRITE(10,'(4E15.6)') WF1,WF2,WF3,WF4
WF1=MAX(WFL1,WFU1)
WF2=WF1/MAX(WFL2,WFU2)
WF3=WF1/MAX(WFL3,WFU3)
WF4=WF1/MAX(WFL4,WFU4)
WF1=WF1/WF1
WRITE(10,9735)
WRITE(10,'(4E15.6)') WF1,WF2,WF3,WF4
WRITE(7,9731)
JC=1
DO I=1,NFL
  DO J=1,NFU
      E1=ABS( YL(1,I) - YU(1,J) )*WF1
       E2=ABS( YL(2,I) + YU(2,J) )*WF2
      E3=ABS( YL(3,I) - YU(3,J) )*WF3
       E4=ABS( YL(4,I) + YU(4,J) )*WF4
EB1=SQRT((E1**2+E2**2)/2.0)
EB2=SQRT((E3**2+E4**2)/2.0)
EB3=SQRT((E1**2+E3**2)/2.0)
EB4=SQRT((E2**2+E4**2)/2.0)
EB1234=SQRT((EB3**2+EB4**2)/2.0)
IF(KCONT.EQ.12) EB=EB1
IF(KCONT.EQ.13) EB=EB3
IF(KCONT.EQ.34) EB=EB2
IF(KCONT.EQ.1234) EB=EB1234
      IF(EB.LT.EBF) THEN
 EBF=EB
 EBJ(JC)=EBF
 JC=JC+1
```

```
END IF
  END DO
END DO
SIGMA=EBJ(JC-11)
EBF=10000.0
    WRITE(10,'(20X,A,I2,A,E15.4/20X,A)')'SIGMA',J11,'=',SIGMA,'----'
WRITE(10,9732)

DO 117 I=1,NFL
DO 119 J=1,NFU

      E1=ABS( YL(1,I) - YU(1,J) )**WF1
      E2=ABS( YL(2,I) + YU(2,J) )**WF2
      E3=ABS( YL(3,I) - YU(3,J) )**WF3
      E4=ABS( YL(4,I) + YU(4,J) )**WF4
EB1=SQRT((E1**2+E2**2)/2.0)
EB2=SQRT((E3**2+E4**2)/2.0)
EB3=SQRT((E1**2+E3**2)/2.0)
EB4=SQRT((E2**2+E4**2)/2.0)
EB1234=SQRT((EB3**2+EB4**2)/2.0)
IF(KCONT.EQ.12) EB=EB1
IF(KCONT.EQ.13) EB=EB3
IF(KCONT.EQ.34) EB=EB2
IF(KCONT.EQ.1234) EB=EB1234
    IF(EB .LT. SIGMA) THEN
     IF(EB.LT.EBF) THEN
 EBF=EB
 IST=I
 JST=J
 ITT=IST+JST
 XS1T(J11)=XL(IST)
 XTT(J11)=XL(IST)+XU(JST)
 XS3TT(J11)=XTT(J11)-XS3(J11)
    WRITE(10,9730)IST,ITT,XS2(J11),XS1T(J11),XS3TT(J11),XTT(J11),EBF

CCCCC67-9-----9-----9-----9-----9-----9-----9----72
CCCCC67---INTEGRATION OF STATES FROM (XS1-D)-(XS1+D)----------72
CCCCC67-9-----9-----9-----9-----9-----9-----9----72
XLEFT=XL(IST-1)
IF(IST.EQ.1) XLEFT=XL(IST)
XRIGHT=XL(IST+1)
IF(IST.EQ.NFL) XRIGHT=XL(IST)
WRITE(5,'(1X,A,E12.4,2X,A,E12.4,4X,A,I4,2X,A,I4)')'XLFT=',XLEFT
   &,'XRGT=',XRIGHT,'IST=',IST,'ITT=',ITT
DO 114 J2=1,NEQNS2
  YSLB(J2,J11)=YL(J2,IST-1)
  IF(IST.EQ.1) YSLB(J2,J11)=YL(J2,IST)
114 CONTINUE
DO 107 I2=1, NINIT
  XLI(I2)=XLEFT+FLOAT(I2-1)/FLOAT(NINIT-1)*(XRIGHT-XLEFT)
  CALL SSET (NEQNS2, 0.0, YLI(1,I2), 1)
107 CONTINUE
KFJ=10
CALL DBVPMS (FCNEQNL,FCNJACB,FCNBCLI,NEQNS2,XLEFT,XRIGHT,DTOL,
   &  BTOL,MAXIT,NINIT,XLI,YLI,LDY2,NMAX,NFLI,XLI,YLI,LDY2)

CCCCC67-9-----9-----9-----9-----9-----9-----9----72
CCCCC67---INTEGRATION OF STATES FROM ((XT-XS1)-D)-((XT-XS1)+D)--- --72
CCCCC67-9-----9-----9-----9-----9-----9-----9----72
XLEFT=XU(JST-1)
IF(JST.EQ.1) XLEFT=XU(JST)
XRIGHT=XU(JST+1)
IF(JST.EQ.NFU) XRIGHT=XU(JST)
WRITE(5,'(1X,A,E12.4,2X,A,E12.4,4X,A,I4,2X,A,I4)')'XLFT=',XLEFT
   &,'XRGT=',XRIGHT,'JST=',JST,'ITT=',ITT
DO 116 J2=1,NEQNS2
  YSUB(J2,J11)=YU(J2,JST-1)
  IF(JST.EQ.1) YSUB(J2,J11)=YU(J2,JST)
116 CONTINUE
DO 108 J2=1, NINIT
  XUI(J2)=XLEFT+FLOAT(J2-1)/FLOAT(NINIT-1)*(XRIGHT-XLEFT)
  CALL SSET (NEQNS2, 0.0, YUI(1,J2), 1)
108 CONTINUE
KFJ=20
CALL DBVPMS (FCNEQNU,FCNJACB,FCNBCUI,NEQNS2,XLEFT,XRIGHT,DTOL,
   &  BTOL,MAXIT,NINIT,XUI,YUI,LDY2,NMAX,NFUI,XUI,YUI,LDY2)

CCCCC67-9-----9-----9-----9-----9-----9-----9----72
CCCCC67---SECOND MINIMIZATION FOR PLACE OF XS1(CONTINUITY OF S&C)---72
CCCCC67-9-----9-----9-----9-----9-----9-----9----72
DO 122 I2=1,NFLI
DO 124 J2=1,NFUI
C3CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
      E1F=ABS( YLI(1,I2) - YUI(1,J2) )**WF1
      E2F=ABS( YLI(2,I2) + YUI(2,J2) )**WF2
      E3F=ABS( YLI(3,I2) - YUI(3,J2) )**WF3
      E4F=ABS( YLI(4,I2) + YUI(4,J2) )**WF4
```

```
 EBI1=SQRT((E1F**2+E2F**2)/2.0)
 EBI2=SQRT((E3F**2+E4F**2)/2.0)
 EBI3=SQRT((E1F**2+E3F**2)/2.0)
 EBI4=SQRT((E2F**2+E4F**2)/2.0)
 EBI1234=SQRT((EBI3**2+EBI4**2)/2.0)
 IF(KCONT.EQ 12) EBI=EBI1
 IF(KCONT.EQ.13) EBI=EBI3
 IF(KCONT.EQ.34) EBI=EBI2
 IF(KCONT.EQ.1234) EBI=EBI1234
      IF(EBI.LT.EBFI) THEN
 EBFI=EBI
 ISF=I2
 JSF=J2
 ITF=ISF+JSF
 IS=I
 JS=J
 IT=IS+JS
 XSF(1,J11)=XLI(ISF)
      XS1(J11)=XSF(1,J11)
 XSF(4,J11)=XLI(ISF)+XUI(JSF)
 XT(J11)=XSF(4,J11)
 XS3C(J11)=XT(J11)-XS3(J11)
 DO 120 J3=1,NEQNS2
  YSS(J3,J11)=YLI(J3,ISF)
120  CONTINUE
 ES1F=0.0
 DO 1081 JJ2=1,NEQNS
ES1F=ES1F+YLI(JJ2,I2)*YLI(JJ2,I2)
1081  CONTINUE
 ES1F=SQRT(ES1F)
 ES2F=0.0
 DO 1082 JJ2=1,NEQNS
ES2F=ES2F+YUI(JJ2,J2)*YUI(JJ2,J2)
1082  CONTINUE
 ES2F=SQRT(ES2F)
       EBIS=ABS( ES1F - ES2F )
     END IF
124 CONTINUE
122 CONTINUE
WRITE(7,973)IS,IT,XS1(J11),XT(J11),EBFI,EBIS
      END IF
    END IF
119 CONTINUE
117 CONTINUE
IF(XT(J11).LE.0.0) THEN
  WRITE(15,*) 'CONTINUITY OF STATES OF FIRST ARM DIDNOT HAPPEN'
  WRITE(18,*) 'CONTINUITY OF STATES OF FIRST ARM DIDNOT HAPPEN'
  GO TO 10000
END IF

CCCCC67-9-----9-----9-----9-----9-----9-----9----.72
CCCCC67---PRINT RESULTS-------------------------------.72
C2CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------.72
     WRITE(15,'(20X,A,I5/20X,A)')'ITR=',J11,'------'
WRITE(15,'(5X,A,7X,A,6X,A,6X,A,5X,A,8X,A,8X,A)')
    &  'T','X1=PH1','X2=DPH1','X3=PH2','X4=DPH2','U1','U2'
     WRITE(18,'(20X,A,I5/20X,A)')'ITR=',J11,'------'
WRITE(18,'(5X,A,7X,A,6X,A,6X,A,5X,A,8X,A,8X,A)')
    &  'T','P1-PH1','P2-DPH1','P3-PH2','P4-DPH2','U1','U2'
DO 127 I=1,IS
WRITE(15,9992) XL(I),(YL(J3,I),J3=1,NEQNS),UL1(I),UL2(I)
WRITE(18,9992) XL(I),(YL(J3,I),J3=NEQNS+1,NEQNS2),UL1(I),UL2(I)
127 CONTINUE
DO 129 I=JS,1,-1
WRITE(15,9992) XT(J11)-XU(I),YU(1,I),-YU(2,I),YU(3,I),-YU(4,I)
    &,UU1(I),UU2(I)
WRITE(18,9992) XT(J11)-XU(I),(YU(J3,I),J3=NEQNS+1,NEQNS2)
    &,UU1(I),UU2(I)
129 CONTINUE



CCCCC67-9-----9-----9-----9-----9-----9-----9----.72
CCCCC67---INTEGRATION OF STATES FROM XT0-XTF FOR PLACE OF TS2-----72
CCCCC67-9-----9-----9-----9-----9-----9-----9----.72
NINIT=NMAX
XLEFT=XLEND
XRIGHT=XT(J11)
     WRITE(5,'(5X,A,E15.6,5X,A,E15.6/)')'XLFT0=',XLEFT,'XRGTT=',XRIGHT
DO I=1, NINIT
  X(I)=XLEFT+FLOAT(I-1)/FLOAT(NINIT-1)*(XRIGHT-XLEFT)
  CALL SSET (NEQNS2, 0.0, YL(1,I), 1)
  CALL SSET (NEQNS2, 0.0, YU(1,I), 1)
END DO

KFJ=11
CALL DBVPMS (FCNEQNL,FCNJACB,FCNBCSL,NEQNS2,XLEFT,XRIGHT,DTOL,
```

```
     &    BTOL,MAXIT,NINIT,X,YL,LDY2,NMAX,NFL,XL,YL,LDY2)

 OPEN(UNIT=21, FILE='t2d21.dat',STATUS= 'UNKNOWN')
 WRITE(21,'(5X,A,6X,A,5X,A,6X,A,5X,A,8X,A,8X,A/)')
     &  'T','X1=PH1','X2=DPH1','X3=PH2','X4=DPH2','U1','U2'
 WFL1=0.0
 WFL2=0.0
 WFL3=0.0
 WFL4=0.0
 DO 1111 I=1,NFL
 C6CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
       X3=YL(3,I)
       P2=YL(6,I)
       P4=YL(8,I)
       A11=C1+C2+2*C3*COS(X3)
       A12=C2+C3*COS(X3)
       A13=C3*SIN(X3)
       A22=C2
       DE=A11*A22-A12*A12
       GS1(I)=(P2*A22-P4*A12)/DE
       GS2(I)=(-P2*A12+P4*A11)/DE
       IF(GS1(I).LE.0.0) THEN
            UL1(I)=T1MAX
       ELSE
            UL1(I)=T1MIN
       END IF
       IF(GS2(I).LE.0.0) THEN
            UL2(I)=T2MAX
       ELSE
            UL2(I)=T2MIN
       END IF

 IF(J11.EQ.1) THEN
    IF(XS1(J11).NE.0.0) THEN
 IF(XL(I).LE.XS1(J11)) THEN
 UL1(I)=SIGN(1,J11)
 ELSE
 UL1(I)=SIGN(3,J11)
 END IF
    ELSE
 UL1(I)=SIGN(1,J11)
    END IF

    IF(ABS(XS2(J11)).LE.SIGM1) THEN
 UL2(I)=SIGN(2,J11)
    ELSE IF(ABS(XS2(J11)).GT.SIGM1) THEN
 IF(XL(I).LE.XS2(J11)) THEN
 UL2(I)=SIGN(2,J11)
 ELSE
 UL2(I)=-SIGN(2,J11)
 END IF
    END IF

 END IF
 WRITE(21,9992) XL(I),(YL(J2,I),J2=1,NEQNS),UL1(I),UL2(I)
 WFL1=MAX(ABS(YL(1,I)),WFL1)
 WFL2=MAX(ABS(YL(2,I)),WFL2)
 WFL3=MAX(ABS(YL(3,I)),WFL3)
 WFL4=MAX(ABS(YL(4,I)),WFL4)
 1111 CONTINUE
 CLOSE(UNIT=21)

 CCCCC67-9-------9-------9-------9-------9-------9-------9-------72
 CCCCC67---INTEGRATION OF STATES FROM XTF-XT0 FOR PLACE OF TS2-------72
 CCCCC67-9-------9-------9-------9-------9-------9-------9-------72
      WRITE(5,'(5X,A,E15.6,5X,A,E15.6/)')'XLFT0=',XLEFT,'XRGTT=',XRIGHT
 KFJ=21
 CALL DBVPMS (FCNEQNU,FCNJACB,FCNBCSU,NEQNS2,XLEFT,XRIGHT,DTOL,
     &     BTOL,MAXIT,NINIT,X,YU,LDY2,NMAX,NFU,XU,YU,LDY2)
 OPEN(UNIT=22, FILE='t2d22.dat',STATUS= 'UNKNOWN')
 WRITE(22,'(5X,A,6X,A,5X,A,6X,A,5X,A,8X,A,8X,A/)')
     &  'T','X1=PH1','X2=DPH1','X3=PH2','X4=DPH2','U1','U2'
 WFU1=0.0
 WFU2=0.0
 WFU3=0.0
 WFU4=0.0
 DO 1131 I=1,NFU
 C6CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
       X3=YU(3,I)
       P2=YU(6,I)
       P4=YU(8,I)
       A11=C1+C2+2*C3*COS(X3)
       A12=C2+C3*COS(X3)
       A13=C3*SIN(X3)
       A22=C2
       DE=A11*A22-A12*A12
       GS1(I)=(P2*A22-P4*A12)/DE
```

```
      GS2(I)=(-P2**A12+P4**A11)/DE
      IF(GS1(I).LE.0.0) THEN
           UU1(I)=T1MAX
      ELSE
           UU1(I)=T1MIN
      END IF
      IF(GS2(I).LE.0.0) THEN
           UU2(I)=T2MAX
      ELSE
           UU2(I)=T2MIN
      END IF

    IF(J11.EQ.1) THEN
    IF(ABS(XS3(J11)).LE.SIGM1) THEN
     UU2(I)=SIGN(4,J11)
    ELSE IF(ABS(XS3(J11)).GT.SIGM1) THEN
     IF(XU(I).LE.XS3(J11)) THEN
    UU2(I)=-SIGN(4,J11)
     ELSE
    UU2(I)=SIGN(4,J11)
     END IF
    END IF
    UU1(I)=SIGN(3,J11)
    END IF
    WRITE(22,9992) XU(I),YU(1,I),-YU(2,I),YU(3,I),-YU(4,I)
       &,UU1(I),UU2(I)
    WFU1=MAX(ABS(YU(1,I)),WFU1)
    WFU2=MAX(ABS(YU(2,I)),WFU2)
    WFU3=MAX(ABS(YU(3,I)),WFU3)
    WFU4=MAX(ABS(YU(4,I)),WFU4)
    1131 CONTINUE
    CLOSE(UNIT=22)

    CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
    CCCCC67---FIRSET MINIMIZATION FOR PLACE OF TS2(CONTINUITY OF S&C)---72
    CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
    OPEN(UNIT=23, FILE='t2d23.dat',STATUS= 'UNKNOWN')
       WRITE(10,*)
    WRITE(23,9733)
    EBF=10000.0
    EBFI=10000.0
    E1F=10000.0
    E2F=10000.0
    E3F=10000.0
    E4F=10000.0
    WF1=MAX(WFL1,WFU1)
    WF2=WF1/MAX(WFL2,WFU2)
    WF3=WF1/MAX(WFL3,WFU3)
    WF4=WF1/MAX(WFL4,WFU4)
    WF1=WF1/WF1
    WRITE(10,9736)
    WRITE(10,'(4E15.6)') WF1,WF2,WF3,WF4
        WRITE(10,9737)
    DO 1171 I=1,NFL
    DO 1191 J=1,NFU

    C3CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-----72
         E1=ABS( YL(1,I) - YU(1,J) )**WF1
         E2=ABS( YL(2,I) + YU(2,J) )**WF2
         E3=ABS( YL(3,I) - YU(3,J) )**WF3
         E4=ABS( YL(4,I) + YU(4,J) )**WF4
    EB1=SQRT((E1**2+E2**2)/2.0)
    EB2=SQRT((E3**2+E4**2)/2.0)
    EB3=SQRT((E1**2+E3**2)/2.0)
    EB4=SQRT((E2**2+E4**2)/2.0)
    EB1234=SQRT((EB3**2+EB4**2)/2.0)
    EB=SQRT((EB1**2+EB2**2)/2.0)
       IF(EB.LT.EBF) THEN
     EBF=EB
     IST=I
     JST=J
     ITT=IST+JST
     XS12(J11)=XL(IST)
     XTT(J11)=XL(IST)+XU(JST)
     XS3C(J11)=XTT(J11)-XS3(J11)
        WRITE(10,9730)IST,ITT,XS1(J11),XS12(J11),XTT(J11),EBF
     DO J3=1,NEQNS2
      YSST(J3,J11)=YL(J3,IST)
     END DO
        END IF

        IF(E1.LT.E1F.AND.I.GT.1) THEN
    E1F=E1
    ITT1=I+J
    XTT1=XL(I)+XU(J)
       WRITE(23,9734)I,ITT1,XS1(J11),XL(I),XTT1,E1,E2,E3,E4,EB
        END IF
```

```
      IF(E2.LT.E2F.AND.I.GT.1) THEN
 E2F=E2
 ITT1=I+J
 XTT1=XL(I)+XU(J)
      END IF

      IF(E3.LT.E3F.AND.I.GT.1) THEN
 E3F=E3
 ITT1=I+J
 XTT1=XL(I)+XU(J)
      END IF

      IF(E4.LT.E4F.AND.I.GT.1) THEN
 E4F=E4
 ITT1=I+J
 XTT1=XL(I)+XU(J)
      END IF

1191 CONTINUE
1171 CONTINUE
CLOSE(UNIT=23)

II=0
DO I=1,IST
  IF(XL(I).LE.XS12(J11)) THEN
II=II+1
XF(II)=XL(I)
DO J=1,NEQNS2
YF(J,II)=YL(J,I)
END DO
UF1(II)=UL1(I)
UF2(II)=UL2(I)
  END IF
END DO
DO I=JST,1,-1
  IF((XTT(J11)-XU(I)).GT.XS12(J11)) THEN
II=II+1
XF(II)=XTT(J11)-XU(I)
DO J=1,NEQNS2
YF(J,II)=YU(J,I)
IF((J.EQ.2).OR.(J.EQ.4)) YF(J,II)=-YF(J,II)
END DO
UF1(II)=UU1(I)
UF2(II)=UU2(I)
  END IF
END DO

J40=J11+40
J50=J11+50
IF(J11.EQ.1) THEN
OPEN(UNIT=J40, FILE='tcf1.dat',STATUS= 'UNKNOWN')
OPEN(UNIT=J50, FILE='tsf1.dat',STATUS= 'UNKNOWN')
ELSE IF(J11.EQ.2) THEN
OPEN(UNIT=J40, FILE='tcf2.dat',STATUS= 'UNKNOWN')
OPEN(UNIT=J50, FILE='tsf2.dat',STATUS= 'UNKNOWN')
ELSE IF(J11.EQ.3) THEN
OPEN(UNIT=J40, FILE='tcf3.dat',STATUS= 'UNKNOWN')
OPEN(UNIT=J50, FILE='tsf3.dat',STATUS= 'UNKNOWN')
END IF
WRITE(J40,'(5X,A,7X,A,6X,A,6X,A,5X,A,8X,A,8X,A)')
    &  'T','P1-PH1','P2-DPH1','P3-PH2','P4-DPH2' ,'U1','U2'
WRITE(J50,'(5X,A,7X,A,6X,A,6X,A,5X,A,8X,A,8X,A)')
    &  'T','X1=PH1','X2=DPH1','X3=PH2','X4=DPH2' ,'U1','U2'
DO I=1,II
WRITE(J50,9992) XF(I),(YF(J3,I),J3=1,NEQNS),UF1(I),UF2(I)
WRITE(J40,9992) XF(I),(YF(J3,I),J3=NEQNS+1,NEQNS2),UF1(I),UF2(I)
END DO

KK=0
DO J=1,II-1
 IF(UF1(J+1)*UF1(J).LT.0.0) THEN
  KK=KK+1
  KS(KK)=J
  KSW(KK)=1
  UFW(KK,1)=UF1(J)
  UFW(KK,2)=UF2(J)
  XSW(KK,J11)=XF(J)+(XF(J+1)-XF(J))/2.0
  DO J2=1,NEQNS2
    YSW(J2,KK)=YF(J2,J)+(YF(J2,J+1)-YF(J2,J))/2.0
  END DO
 ELSE IF(UF2(J+1)*UF2(J).LT.0.0) THEN
  KK=KK+1
  KS(KK)=J
  KSW(KK)=2
  UFW(KK,1)=UF1(J)
  UFW(KK,2)=UF2(J)
```

```
      XSW(KK,J11)=XF(J)+(XF(J+1)-XF(J))/2.0
      DO J2=1,NEQNS2
        YSW(J2,KK)=YF(J2,J)+(YF(J2,J+1)-YF(J2,J))/2.0
      END DO
     END IF
    END DO

     J=II
     KK=KK+1
     KS(KK)=J
     KSW(KK)=0
     UFW(KK,1)=UF1(J)
     UFW(KK,2)=UF2(J)
     XSW(KK,J11)=XF(J)
     DO J2=1,NEQNS2
      YSW(J2,KK)=YF(J2,J)
     END DO

    WRITE(J40,'(3X,A,3X,A,7X,A,10X,A,10X,A,10X,A,10X,A)')'KK','KS'
       + ,'XSW1','YSW5','YSW6','YSW7','YSW8'
    WRITE(J50,'(3X,A,3X,A,7X,A,10X,A,10X,A,10X,A,10X,A)')'KK','KS'
       + ,'XSW1','YSW1','YSW2','YSW3','YSW4'
    DO JK=1,KK
     WRITE(J50,'(2I5,1P5E14.6)') JK,KS(JK),XSW(JK,J11)
       +    ,(YSW(J2,JK),J2=1,NEQNS)
     WRITE(J40,'(2I5,1P5E14.6)') JK,KS(JK),XSW(JK,J11)
       +    ,(YSW(J2,JK),J2=NEQNS+1,NEQNS2)
    END DO
    WRITE(J40,'(3X,A,3X,A,2X,A,11X,A,10X,A,10X,A)')'KK','KS'
       + ,'KSW','XSW','UFW1','UFW2'
    WRITE(J50,'(3X,A,3X,A,2X,A,11X,A,10X,A,10X,A)')'KK','KS'
       + ,'KSW','XSW','UFW1','UFW2'
    DO JK=1,KK
     WRITE(J50,'(3I5,3E14.6)') JK,KS(JK),KSW(JK),XSW(JK,J11)
       +    ,UFW(JK,1),UFW(JK,2)
     WRITE(J40,'(3I5,3E14.6)') JK,KS(JK),KSW(JK),XSW(JK,J11)
       +    ,UFW(JK,1),UFW(JK,2)
    END DO

    IF(XTT(J11).LE.0.0) THEN
     WRITE(J40,*) 'CONTINUITY OF STATES OF BOTH ARMS DIDNOT HAPPEN'
     WRITE(J50,*) 'CONTINUITY OF STATES OF BOTH ARMS DIDNOT HAPPEN'
    END IF

    CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
    C--67--INTEGRATION OF STATES FROM XT0-XSW1 ----------72
    CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
    DO J=1,NEQNS2
    YBCL(J)=YSCL(J,J11)
    YBCR(J)=YF(J,KS(1))
    END DO
    XLEFT=XLEND
    XRIGHT=XSW(1,J11)
          DX=ABS(XRIGHT-XLEFT)
          NINIT=NMAX*(DX/ABS(XREND-XLEND))
          NINIT=MAX(NINIT,10)
         WRITE(5,'(5X,A,E15.6,5X,A,E15.6/)')'XLFT0=',XLEFT,'XRGTS1=',XRIGHT
    C                  Define the Shooting Points
    DO I=1, NINIT
      XF(I)=XLEFT+FLOAT(I-1)/FLOAT(NINIT-1)*(XRIGHT-XLEFT)
      CALL SSET (NEQNS2, 0.0, YF(1,I), 1)
    END DO

    KFJ=30
    CALL DBVPMS (FCNEQNL,FCNJACB,FCNBC,NEQNS2,XLEFT,XRIGHT,DTOL,
       &     BTOL,MAXIT,NINIT,XF,YF,LDY2,NINIT,NFL,XF,YF,LDY2)

    CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
    C--67--INTEGRATION OF STATES & COSTATES FROM XSW1-XSW2--------72
    CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
    DO J=1,NEQNS2
    YBCL(J)=YF(J,1)
    YBCR(J)=YF(J,NFL)
    END DO
    XLEFT=XSW(1,J11)
    XRIGHT=XSW(2,J11)
    DX=ABS(XRIGHT-XLEFT)
    NINIT=NMAX*(DX/ABS(XREND-XLEND))
    NINIT=MAX(NINIT,10)
        WRITE(5,'(5X,A,E15.6,5X,A,E15.6/)')'XLFS1=',XLEFT,'XRGTS2=',XRIGHT
    C                  Define the Shooting Points
          DO I=1, NINIT
            XFI(I)=XLEFT+FLOAT(I-1)/FLOAT(NINIT-1)*(XRIGHT-XLEFT)
            CALL SSET (NEQNS2, 0.0, YFI(1,I), 1)
    END DO

    C                  Solve Problem
```

```
KFJ=32
CALL DBVPMS (FCNEQNL,FCNJACB,FCNBC,NEQNS2,XLEFT,XRIGHT,DTOL,
    & BTOL,MAXIT,NINIT,XFI,YFI,LDY2,NINIT,NFLI,XFI,YFI,LDY2)

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C---67---INTEGRATION OF STATES & COSTATES FROM XLEND-XSW1-------72
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
DO J=1,(NEQNS2/2)
YBCL(J)=YSCL(J,J11)
YBCR(J)=YFI(J,NFLI)
END DO
DO J=(NEQNS2/2+1),NEQNS2
YBCL(J)=YFI(J,1)
YBCR(J)=YFI(J,NFLI)
END DO
XLEFT=XLEND
XRIGHT=XSW(1,J11)
DX=ABS(XRIGHT-XLEFT)
NINIT=NMAX*(DX/ABS(XREND-XLEND))
NINIT=MAX(NINIT,10)
     WRITE(5,'(5X,A,E15.6,5X,A,E15.6/)')'XLFT0=',XLEFT,'XRGTS1=',XRIGHT
C                 Define the Shooting Points
DO I=1, NINIT
  XF(I)=XLEFT+FLOAT(I-1)/FLOAT(NINIT-1)*(XRIGHT-XLEFT)
  CALL SSET (NEQNS2, 0.0, YF(1,I), 1)
END DO

KFJ=31
CALL DBVPMS (FCNEQNL,FCNJACB,FCNBC,NEQNS2,XLEFT,XRIGHT,DTOL,
    &     BTOL,MAXIT,NINIT,XF,YF,LDY2,NINIT,NFL,XF,YF,LDY2)

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
CCCCC67-----------Print Results-----------------------72
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
WRITE(1,'(6X,A,5X,A,4X,A,5X,A,4X,A,7X,A,7X,A,10X,A)')
    &  'T','X1=PH1','X2=DPH1','X3=PH2','X4=DPH2' ,'U1','U2','H'
WRITE(20,'(6X,A,5X,A,4X,A,5X,A,4X,A,7X,A,7X,A,10X,A)')
    &  'T','P1-PH1','P2-DPH1','P3-PH2','P4-DPH2' ,'G1','G2','H'

KFJ=31
      DO I=1,NFL
X1=YF(1,I)
X2=YF(2,I)
X3=YF(3,I)
X4=YF(4,I)
P1=YF(5,I)
P2=YF(6,I)
P3=YF(7,I)
P4=YF(8,I)
  IF(KFJ.EQ.31.OR.KFJ.EQ.30) THEN
UF1(I)=UFW(1,1)
UF2(I)=UFW(1,2)
  END IF
  IF(KFJ.EQ.32) THEN
UF1(I)=UFW(2,1)
UF2(I)=UFW(2,2)
  END IF
T1=UF1(I)
T2=UF2(I)
DE = C1*C2-C3**2*COS(X3)**2
A11 = C1+C2+2*C3*COS(X3)
A12 = C2+C3*COS(X3)
A13 = C3*SIN(X3)
A22 = C2
A14 = C4*COS(X1)+C5*COS(X1+X3)
A24 = C5*COS(X1+X3)
      H(I,J11)=
    &1+P1*X2+P3*X4+P2/DE*(-GR*(A22*A14-A12*A24)+A22*T1-A12*T2+A13*(A12*
    +X2**2+A22*(2*X2*X4+X4**2)))+P4/DE*(GR*(A12*A14-A11*A24)-A12*T1+A11
    +*T2-A13*(A11*X2**2+A12*(2*X2*X4+X4**2)))
      GC1(I)=(P2*A22-P4*A12)/DE
      GC2(I)=(-P2*A12+P4*A11)/DE
     WRITE(1,1007)XF(I),(YF(J1,I),J1=1,NEQNS),UF1(I),UF2(I),H(I,J11)
     WRITE(20,1007)XF(I),(YF(J1,I),J1=NEQNS+1,NEQNS2),GC1(I),GC2(I)
    &,H(I,J11)
END DO

KFJ=32
      DO I=1,NFLI
X1=YFI(1,I)
X2=YFI(2,I)
X3=YFI(3,I)
X4=YFI(4,I)
P1=YFI(5,I)
P2=YFI(6,I)
P3=YFI(7,I)
P4=YFI(8,I)
```

```
   IF(KFJ.EQ.31.OR.KFJ.EQ.30) THEN
 UF1(I)=UFW(1,1)
 UF2(I)=UFW(1,2)
   END IF
   IF(KFJ.EQ.32) THEN
 UF1(I)=UFW(2,1)
 UF2(I)=UFW(2,2)
   END IF
 T1=UF1(I)
 T2=UF2(I)
 DE = C1*C2-C3**2*COS(X3)**2
 A11 = C1+C2+2*C3*COS(X3)
 A12 = C2+C3*COS(X3)
 A13 = C3*SIN(X3)
 A22 = C2
 A14 = C4*COS(X1)+C5*COS(X1+X3)
 A24 = C5*COS(X1+X3)
       H(I,J11)=
     &1+P1*X2+P3*X4+P2/DE*(-GR*(A22*A14-A12*A24)+A22*T1-A12*T2+A13*(A12*
     +X2**2+A22*(2*X2*X4+X4**2)))+P4/DE*(GR*(A12*A14-A11*A24)-A12*T1+A11
     +*T2-A13*(A11*X2**2+A12*(2*X2*X4+X4**2)))
       GC1(I)=(P2*A22-P4*A12)/DE
       GC2(I)=(-P2*A12+P4*A11)/DE
      WRITE(1,1007)XFI(I),(YFI(J1,I),J1=1,NEQNS),UF1(I),UF2(I),H(I,J11)
      WRITE(20,1007)XFI(I),(YFI(J1,I),J1=NEQNS+1,NEQNS2),GC1(I),GC2(I)
     &,H(I,J11)
END DO

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
CCCCC67---INTEGRATION OF STATES & COSTATES FROM XSW2-XREND-------72
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
DO J=1,NEQNS2
YBCL(J)=YFI(J,1)
YBCR(J)=YFI(J,NFLI)
END DO
    DO 130 JK=3,KK
NSW=JK
XLEFT=XSW(JK-1,J11)
XRIGHT=XSW(JK,J11)
DX=ABS(XRIGHT-XLEFT)
NINIT=NMAX*(DX/ABS(XREND-XLEND))
NINIT=MAX(NINIT,10)
     WRITE(5,'(5X,A,I1,A,E15.6,5X,A,I1,A,E15.6/)')'XLFS',JK-1,'='
    +,XLEFT,'XRGTS',JK,'=',XRIGHT
C                Define the Shooting Points
      DO I=1, NINIT
        XFR(I)=XLEFT+FLOAT(I-1)/FLOAT(NINIT-1)*(XRIGHT-XLEFT)
        CALL SSET (NEQNS2, 0.0, YFR(1,I), 1)
END DO

C                   Solve Problem
KFJ=33
CALL DBVPMS (FCNEQNL,FCNJACB,FCNBC,NEQNS2,XLEFT,XRIGHT,DTOL,
   & BTOL,MAXIT,NINIT,XFR,YFR,LDY2,NINIT,NFR,XFR,YFR,LDY2)
DO J=1,NEQNS2
YBCL(J)=YFR(J,1)
YBCR(J)=YFR(J,NFR)
END DO
      DO I=1,NFR
X1=YFR(1,I)
X2=YFR(2,I)
X3=YFR(3,I)
X4=YFR(4,I)
P1=YFR(5,I)
P2=YFR(6,I)
P3=YFR(7,I)
P4=YFR(8,I)
UF1(I)=UFW(JK,1)
UF2(I)=UFW(JK,2)
T1=UF1(I)
T2=UF2(I)
DE = C1*C2-C3**2*COS(X3)**2
A11 = C1+C2+2*C3*COS(X3)
A12 = C2+C3*COS(X3)
A13 = C3*SIN(X3)
A22 = C2
A14 = C4*COS(X1)+C5*COS(X1+X3)
A24 = C5*COS(X1+X3)
      H(I,J11)=
   &1+P1*X2+P3*X4+P2/DE*(-GR*(A22*A14-A12*A24)+A22*T1-A12*T2+A13*(A12*
   +X2**2+A22*(2*X2*X4+X4**2)))+P4/DE*(GR*(A12*A14-A11*A24)-A12*T1+A11
   +*T2-A13*(A11*X2**2+A12*(2*X2*X4+X4**2)))
     GC1(I)=(P2*A22-P4*A12)/DE
     GC2(I)=(-P2*A12+P4*A11)/DE
    WRITE(1,1007)XFR(I),(YFR(J1,I),J1=1,NEQNS),UF1(I),UF2(I),H(I,J11)
    WRITE(20,1007)XFR(I),(YFR(J1,I),J1=NEQNS+1,NEQNS2),GC1(I),GC2(I)
   &,H(I,J11)
```

```
 END DO
 130  CONTINUE
 CLOSE(UNIT=J40)
 CLOSE(UNIT=J50)

 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
 CCCCC67---CHEKING THE CONVERGENCE CRITERIA----------72
 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
 EBS=0.0
 DO JK=1,KK
  XSW(JK,0)=0.0
  EBS= (XSW(JK,J11)-XSW(JK,J11-1))**2+EBS
 END DO
 EBS=SQRT(EBS/KK)
 WRITE(8,'(10X,A/5X,A,5X,A,9X,A,9X,A,9X,A)') '********
    &'INTERMEDIATE RESULTS********','ITR','XSW1','XSW2','XSW3','XSW4'
 WRITE(8,'(50X,A,10X,A,9X,A/)') 'XS1','XT','EBS'
       WRITE(8,1002) J11,(XSW(JK,J11),JK=1,KK)
       WRITE(8,1012) XS12(J11),XTT(J11),EBS
       WRITE(8,*) '--------------------'

 IF((J11.GE.2).AND.(EBS.LT.DTOL)) THEN
  WRITE(8,'(10X,A/5X,A,4X,A,4X,A,4X,A,4X,A)') '********
    &'FINAL RESULTS********','ITR','XSW1','XSW2','XSW3','XSW4'
  WRITE(8,'(50X,A,10X,A,9X,A/)') 'XS12','XTT','EBS'
     WRITE(8,1002) J11,(XSW(JK,J11),JK=1,KK)
     WRITE(8,1012) XS12(J11),XTT(J11),EBS
      WRITE(8,*) '--------------------'
   GOTO 10000
 END IF

 1450 CONTINUE
 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
 CCCCC67-END OF THE ITERATION LOOP----------72
 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72

 WRITE(1,'(10X,A)') 'THE PROGRAM DID NOT CONVERGE.'
 WRITE(8,'(10X,A)') 'THE PROGRAM DID NOT CONVERGE.'
 WRITE(15,'(10X,A)') 'THE PROGRAM DID NOT CONVERGE.'
 WRITE(18,'(10X,A)') 'THE PROGRAM DID NOT CONVERGE.'
 WRITE(20,'(10X,A)') 'THE PROGRAM DID NOT CONVERGE.'
 GOTO 10000

 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
 CCCCC67---FINDING COSTATES FROM X0 TO XS1----------72
 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
 NINIT=IS
 IF(NINIT.EQ.1) NINIT=2

 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
 CCCCC67---DEFINING INITIAL COSTATES USING HAM. & SWITCH FUNCS.---72
 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
 C2CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
       A11=C1+C2+2*C3*COS(X3)
       A12=C2+C3*COS(X3)
       A13=C3*SIN(X3)
       A22=C2
 A14 = C4*COS(X1)+C5*COS(X1+X3)
 A24 = C5*COS(X1+X3)
       DE=A11*A22-A12*A12

 P1=ALFA*X1
 P3=BETA*X3
 T1=UL1(IS)
 T2=UL2(IS)
 CCCCC67-9---- P2G1=0 -9-----9-----9-----9-----9-----72
       P2=
    &DE*(1.0+P1*X2+P3*X4)/(A22*GR*A11*A24-A22*A11*T2+A22*A13*A11*X2**2-
    +A12**2*GR*A24+A12**2*T2-A12**2*A13*X2**2)*A12
 CCCCC67-9---- P4G1=0 -9-----9-----9-----9-----9-----72
       P4=
    &A22*DE*(1.0+P1*X2+P3*X4)/(A22*GR*A11*A24-A22*A11*T2+A22*A13*A11*X2
    +**2-A12**2*GR*A24+A12**2*T2-A12**2*A13*X2**2)
 XLEFT=XLEND
 XRIGHT=XS1(J11)
     WRITE(5,'(1X,A,E12.4,2X,A,E12.4,4X,A,I4,2X,A,I4)')'XLFTC=',XLEFT
    &,'XRGTC=',XRIGHT,'I=',1,'IS=',IS
 IF(XRIGHT.LE.XLEFT) THEN
 NFLC=1
 XLC(NFLC)=XLEFT
 UCF1(NFLC)=UL1(NFLC)
 UCF2(NFLC)=UL2(NFLC)
 DO 1472 J2=1,NEQNS2
  YLC(J2,NFLC)=YSCL(J2,J11)
 1472 CONTINUE
 KK=0
 GO TO 1340
```

```
 END IF

 C DEFINE THE SHOOTING POINTS
 DO 137 I=1, NINIT
   XLC(I)=XLEFT+FLOAT(I-1)/FLOAT(NINIT-1)*(XRIGHT-XLEFT)
   CALL SSET (NEQNS2, 0.0, YLC(1,I), 1)
 137 CONTINUE
 KFJ=12
 C CALL DBVPMS (FCNEQNC,FCNJACB,FCNBCLC,NEQNS2,XLEFT,XRIGHT,DTOL,
 C    & BTOL,MAXIT,NINIT,XLC,YLC,LDY2,NINIT,NFLC,XLC,YLC,LDY2)
 DO 1371 J=1,NFLC
 C6CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
       X3=YLC(3,J)
       P2=YLC(6,J)
       P4=YLC(8,J)
       A11=C1+C2+2*C3*COS(X3)
       A12=C2+C3*COS(X3)
       A13=C3*SIN(X3)
       A22=C2
       DE=A11*A22-A12*A12
       GC1(J)=(P2*A22-P4*A12)/DE
       GC2(J)=(-P2*A12+P4*A11)/DE
       IF(GC1(J).LE.0.0) THEN
            UCF1(J)=T1MAX
       ELSE
            UCF1(J)=T1MIN
       END IF
       IF(GC2(J).LE.0.0) THEN
            UCF2(J)=T2MAX
       ELSE
            UCF2(J)=T2MIN
       END IF
 1371   CONTINUE
 KK=0
 DO 1331 J=1,NFLC-1
  IF(GC1(J+1)*GC1(J).LT.0.0) THEN
   KK=KK+1
   KS(KK)=J
   XSW(KK,J11)=XLC(J)-(XLC(J+1)-XLC(J))*GC1(J)/(GC1(J+1)-GC1(J))
   DO 1332 J2=1,NEQNS2
   YSW(J2,KK)=YLC(J2,J)-(YLC(J2,J+1)-YLC(J2,J))*GC1(J)/
     &            (GC1(J+1)-GC1(J))
 1332   CONTINUE
  END IF
  IF(GC2(J+1)*GC2(J).LT.0.0) THEN
   KK=KK+1
   KS(KK)=J
   XSW(KK,J11)=XLC(J)-(XLC(J+1)-XLC(J))*GC2(J)/(GC2(J+1)-GC2(J))
   DO J2=1,NEQNS2
   YSW(J2,KK)=YLC(J2,J)-(YLC(J2,J+1)-YLC(J2,J))*GC2(J)/
     &            (GC2(J+1)-GC2(J))
  END DO
  END IF
 1331 CONTINUE

 1340 CONTINUE
 WRITE(3,1003) J11
 WRITE(19,1006) J11
 DO 134 I=1,NFLC
 WRITE(3,9991) XLC(I), (YLC(J2,I),J2=1,NEQNS) ,UCF1(I),UCF2(I)
      WRITE(19,9991)XLC(I),(YLC(J2,I),J2=NEQNS+1,NEQNS2),UCF1(I),UCF2(I)
 134 CONTINUE
   KK=KK+1
   KS(KK)=IS
   XSW(KK,J11)=XS1(J11)

 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
 CCCCC67---FINDING COSTATES FROM XS1 TO XT----------72
 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
 NINIT=IT-IS
 IF(NINIT.EQ.1) NINIT=2
 XLEFT=XS1(J11)
 XRIGHT=XT(J11)
      WRITE(5,'(1X,A,E12.4,2X,A,E12.4,4X,A,I4,2X,A,I4)')'XLFTC=',XLEFT
     &,'XRGTC=',XRIGHT,'IS=',IS,'IT=',IT
 IF(XRIGHT.LE.XLEFT) THEN
 NFUC=1
 XUC(NFUC)=XLC(NFLC)
 UCF1(NFUC)=UCF1(NFLC)
 UCF2(NFUC)=UCF2(NFLC)
 DO 1342 J2=1,NEQNS2
   YUC(J2,NFUC)=YLC(J2,NFLC)
 1342 CONTINUE
 GO TO 140
 END IF

 C DEFINE THE SHOOTING POINTS
```

```
      DO 138 I=1, NINIT
        XUC(I)=XLEFT+FLOAT(I-1)/FLOAT(NINIT-1)*(XRIGHT-XLEFT)
             CALL SSET (NEQNS2, 0.0, YUC(1,I), 1)
      138 CONTINUE
      KFJ=22
      C CALL DBVPMS (FCNEQNC,FCNJACB,FCNBCUC,NEQNS2,XLEFT,XRIGHT,DTOL,
      C     &        BTOL,MAXIT,NINIT,XUC,YUC,LDY2,NINIT,NFUC,XUC,YUC,LDY2)

      DO 1381 J=1,NFUC
      C6CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
            X3=YUC(3,J)
            P2=YUC(6,J)
            P4=YUC(8,J)
            A11=C1+C2+2*C3*COS(X3)
            A12=C2+C3*COS(X3)
            A13=C3*SIN(X3)
            A22=C2
            DE=A11*A22-A12*A12
            GC1(J)=(P2*A22-P4*A12)/DE
            GC2(J)=(-P2*A12+P4*A11)/DE
            IF(GC1(J).LE.0.0) THEN
                 UCF1(J)=T1MAX
            ELSE
                 UCF1(J)=T1MIN
            END IF
            IF(GC2(J).LE.0.0) THEN
                 UCF2(J)=T2MAX
            ELSE
                 UCF2(J)=T2MIN
            END IF
      1381  CONTINUE
      DO 1385 J=2,NFUC-1
       IF(GC1(J+1)*GC1(J).LT.0.0) THEN
        KK=KK+1
        KS(KK)=J
        XSW(KK,J11)=XUC(J)-(XUC(J+1)-XUC(J))*GC1(J)/(GC1(J+1)-GC1(J))
        DO 1386 J2=1,NEQNS2
             YSW(J2,KK)=YUC(J2,J)-(YUC(J2,J+1)-YUC(J2,J))*GC1(J)/
          & (GC1(J+1)-GC1(J))
      1386  CONTINUE
       END IF
       IF(GC2(J+1)*GC2(J).LT.0.0) THEN
        KK=KK+1
        KS(KK)=J
        XSW(KK,J11)=XUC(J)-(XUC(J+1)-XUC(J))*GC2(J)/(GC2(J+1)-GC2(J))
        DO J2=1,NEQNS2
        YSW(J2,KK)=YUC(J2,J)-(YUC(J2,J+1)-YUC(J2,J))*GC2(J)/
          &            (GC2(J+1)-GC2(J))
       END DO
       END IF
      1385 CONTINUE

      DO 141 I=2,NFUC
      I4=I+IS-1
      WRITE(3,9991) XUC(I), (YUC(J2,I),J2=1,NEQNS) ,UCF1(I),UCF2(I)
          WRITE(19,9991)XUC(I),(YUC(J2,I),J2=NEQNS+1,NEQNS2),UCF1(I),UCF2(I)
      141 CONTINUE
      140 CONTINUE

      C6CCC67-THIS LINE/LINES SHOULD BE DROPED----------------72
      CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
      C DEFINING THE SIGN OF CONTROL FOR THE NEXT ITERATION
      CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
       SIGN(1,J11)=UF2(1)
       SIGNT(3,J11)=UF2(IS+1)
       SIGN(2,J11)=UF2(IS-1)
       SIGN(4,J11)=UF2(IT-1)
      WRITE(3,'(2X,A,E8.2,5X,A,E8.2,5X,A,E8.2,5X,A,E8.2)')'S1=',SIGN(
          &1,J11),'S1T=',SIGNT(1,J11),'S3=',SIGN(3,J11),'S3T=',SIGNT(3,J11)
      WRITE(3,'(2X,A,E8.2,5X,A,E8.2,5X,A,E8.2,5X,A,E8.2)')'S2=',SIGN(
          &2,J11),'S2T=',SIGNT(2,J11),'S4=',SIGN(4,J11),'S4T=',SIGNT(4,J11)

      C6CCC67-THIS LINE/LINES SHOULD BE DROPED----------------72

      C9CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
      WRITE(3,'(/3X,A,3X,A,7X,A,1GX,A,10X,A,10X,A,10X,A)')'KK','KS'
          +  ,'XSW1','YSW1','YSW2','YSW3','YSW4'
      WRITE(19,'(/3X,A,3X,A,7X,A,10X,A,10X,A,10X,A,10X,A)')'KK','KS'
          +  ,'XSW1','YSW1','YSW2','YSW3','YSW4'
      DO 1413 JK=1,KK
        WRITE(3,'(2I5,1P5E14.6)') JK,KS(JK),XSW(JK,J11)
          +    ,(YSW(J2,JK),J2=1,NEQNS)
        WRITE(19,'(2I5,1P5E14.6)') JK,KS(JK),XSW(JK,J11)
          +    ,(YSW(J2,JK),J2=NEQNS+1,NEQNS2)
      1413 CONTINUE

      CCCCC67---INTEGRATING BOTH STATES & CO AT THE END OF EACH ITERATION-72
```

```
CCCCC67-USING THE SOLUTION OF COSTS AS INITIAL CONDITION FOR YC(X(0))-72
DO 1491 J3=1,NEQNS
  J4=NEQNS+J3
C   YSCB(J3,J11)=YSCL(J3,J11)
C   YSCB(J4,J11)=YSCL(J4,J11+1)
1491 CONTINUE
XLEFT=XLEND
XRIGHT=XT(J11)
      WRITE(5,'(5X,A,E15.6,5X,A,E15.6/)')'XLFTB=',XLEFT,'XRGTB=',XRIGHT
NINIT=IT-1

C DEFINE THE SHOOTING POINTS
DO 151 I=1, NINIT
   XB(I)=XLEFT+FLOAT(I-1)/FLOAT(NINIT-1)*(XRIGHT-XLEFT)
   CALL SSET (NEQNS2, 0.0, YB(1,I), 1)
151 CONTINUE
KFJ=12
C CALL DBVPMS (FCNEQNC,FCNJACB,FCNBCB,NEQNS2,XLEFT,XRIGHT,DTOL,
C     &   BTOL,MAXIT,NINIT,XB,YB,LDY2,NINIT,NFB,XB,YB,LDY2)

CCCCC67---CALCULATING THE HAMILTONIAN-------------------72
CCCCC67---PRINTING THE FINAL RESULTS OF EACH ITERATION--------72
WRITE(1,'(/1X,A)')'-------------------------'
WRITE(1,'(11X,A,14X,A,14X,A,14X,A)'),'YSCB1','YSCB2','YSCB3'
    &,'YSCB4'
C WRITE(1,'(1P4E19.11/)') (YSCB(J2,J11),J2=1,NEQNS)
WRITE(20,'(11X,A,14X,A,14X,A,14X,A)'),'YSCB5','YSCB6','YSCB7'
    &,'YSCB8'
C WRITE(20,'(1P4E19.11/)') (YSCB(J2,J11),J2=NEQNS+1,NEQNS2)
WRITE(1,1005) J11
WRITE(20,1008) J11

DO 153 J=1,NFB
C1CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
      X1=YB(1,J)
      X2=YB(2,J)
      X3=YB(3,J)
      X4=YB(4,J)
      P1=YB(5,J)
      P2=YB(6,J)
      P3=YB(7,J)
      P4=YB(8,J)
      A11=C1+C2+2*C3*COS(X3)
      A12=C2+C3*COS(X3)
      A13=C3*SIN(X3)
      A22=C2
A14 = C4*COS(X1)+C5*COS(X1+X3)
A24 = C5*COS(X1+X3)
      DE=A11*A22-A12*A12
      GC1(J)=(P2*A22-P4*A12)/DE
      GC2(J)=(-P2*A12+P4*A11)/DE
      IF(GC1(J).LE.0.0) THEN
            UF1(J)=T1MAX
      ELSE
            UF1(J)=T1MIN
      END IF
      IF(GC2(J).LE.0.0) THEN
            UF2(J)=T2MAX
      ELSE
            UF2(J)=T2MIN
      END IF
T1=UF1(J)
T2=UF2(J)
      H(J,J11)=
   &1+P1*X2+P3*X4+P2/DE*(-GR*(A22*A14-A12*A24)+A22*T1-A12*T2+A13*(A12*
   +X2**2+A22*(2*X2*X4+X4**2)))+P4/DE*(GR*(A12*A14-A11*A24)-A12*T1+A11
   +*T2-A13*(A11*X2**2+A12*(2*X2*X4+X4**2)))
WRITE(1,1007) XB(J),(YB(J1,J),J1=1,NEQNS),UF1(J),UF2(J),H(J,J11)
WRITE(20,1007) XB(J),(YB(J1,J),J1=NEQNS+1,NEQNS2),UF1(J),UF2(J)
    &,H(J,J11)
153   CONTINUE
KKB=0
DO 1531 J=1,NFB-1
 IF(GC1(J+1)*GC1(J).LT.0.0) THEN
  KKB=KKB+1
  KSB(KKB)=J
  XBS(KKB,J11)=XB(J)-(XB(J+1)-XB(J))*GC1(J)/(GC1(J+1)-GC1(J))
  DO 1532 J2=1,NEQNS2
        YBS(J2,KKB)=YB(J2,J)-(YB(J2,J+1)-YB(J2,J))*GC1(J)/(GC1(J+1)
    & -GC1(J))
1532   CONTINUE
 END IF
 IF(GC2(J+1)*GC2(J).LT.0.0) THEN
  KKB=KKB+1
  KSB(KKB)=J
  XBS(KKB,J11)=XB(J)-(XB(J+1)-XB(J))*GC2(J)/(GC2(J+1)-GC2(J))
  DO J2=1,NEQNS2
```

```
          YBS(J2,KKB)=YB(J2,J)-(YB(J2,J+1)-YB(J2,J))*GC2(J)/(GC2(J+1)
     & -GC2(J))
   END DO
  END IF
 1531 CONTINUE

 C5CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
 WRITE(1,'(/3X,A,3X,A,7X,A,10X,A,10X,A,10X,A,10X,A)')
     +  'KKB','KSB','XBS','YBS1','YBS2','YBS3','YBS4'
 WRITE(20,'(/3X,A,3X,A,7X,A,10X,A,10X,A,10X,A,10X,A)')
     +  'KKB','KSB','XBS','YBS5','YBS6','YBS7','YBS8'
 DO 1533 JK=1,KKB
   WRITE(1,'(2I5,1P5E14.6)') JK,KSB(JK),XBS(JK,J11)
     +    ,(YBS(J2,JK),J2=1,NEQNS)
   WRITE(20,'(2I5,1P5E14.6)') JK,KSB(JK),XBS(JK,J11)
     +    ,(YBS(J2,JK),J2=NEQNS+1,NEQNS2)
 1533 CONTINUE
      WRITE(1,'(/A,3X,A,9X,A,14X,A,14X,A,14X,A)')'  IS','IT','XS2','XS1'
     *,'XS3','XT'
      WRITE(1,'(2I5,1P4E17.9/)')IS,IT,XS2(J11),XS1(J11),XS3(J11),XT(J11)

 CCCCC67---CHEKING THE CONVERGENCE CRITERIA----------------72
 EBS=0.0
 DO 157 JK=1,KK
   XSW(JK,0)=0.0
   EBS= (XSW(JK,J11)-XSW(JK,J11-1))**2+EBS
 157 CONTINUE
 XT(0)=0.0
 EBS=(XT(J11)-XT(J11-1))**2+EBS
 EBS=SQRT(EBS/(KK+1))
       WRITE(8,1002) J11,(XSW(JK,J11),JK=1,KK)
       WRITE(8,1012) XS1(J11),XT(J11),EBS
       WRITE(8,*) '-------------------------------'

 C3CCC67-THIS LINE/LINES SHOULD BE DROPED----------------72
 EBS0=SQRT(((XS1(J11)-XS1(J11-1))**2+(XSF(2,J11)-XSF(2,J11))*
     &*2+(XSF(3,J11)-XSF(3,J11))**2+(XT(J11)-XT(J11-1))**2)/4)
      WRITE(16,1001) J11,XSF(2,J11),XS1(J11),XSF(3,J11),XT(J11),EBS0
 C3CCC67-THIS LINE/LINES SHOULD BE DROPED----------------72

 IF(EBS.LT.DTOL) THEN
  WRITE(8,'(10X,A/5X,A,4X,A,4X,A,4X,A,4X,A)') '********
     &*'FINAL RESULTS********','ITR','XSW1','XSW2','XSW3','XSW4'
  WRITE(8,'(50X,A,10X,A,9X,A/)') 'XS1','XT','EBS'
      WRITE(8,1002) J11,(XSW(JK,J11),JK=1,KK)
      WRITE(8,1012) XS1(J11),XT(J11),EBS
       WRITE(8,*) '-------------------------------'
   GOTO 10000
 END IF

 C3CCC67-THIS LINE/LINES SHOULD BE DROPED----------------72
 IF(EBS0.LT.DTOL) THEN
  WRITE(16,'(10X,A/6X,A,10X,A,10X,A,10X,A,9X,A,10X,A)') '********
     &*'FINAL RESULTS************','ITR','TS2','TS1','TS3','TF','EBS0'
      WRITE(16,1001) J11,XSF(2,J11),XS1(J11),XSF(3,J11),XT(J11),EBS0
   GOTO 10000
 END IF
 C3CCC67-THIS LINE/LINES SHOULD BE DROPED----------------72

 CCCCC67-END OF THE ITERATION LOOP----------------72
 145 CONTINUE
 WRITE(1,'(10X,A)') 'THE PROGRAM DID NOT CONVERGE.'
 WRITE(3,'(10X,A)') 'THE PROGRAM DID NOT CONVERGE.'
 WRITE(8,'(10X,A)') 'THE PROGRAM DID NOT CONVERGE.'
 WRITE(15,'(10X,A)') 'THE PROGRAM DID NOT CONVERGE.'
 WRITE(18,'(10X,A)') 'THE PROGRAM DID NOT CONVERGE.'
 WRITE(19,'(10X,A)') 'THE PROGRAM DID NOT CONVERGE.'
 WRITE(20,'(10X,A)') 'THE PROGRAM DID NOT CONVERGE.'

 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
 CCCCC67-FORMATS----------72
 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
 C27CC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
 973     FORMAT(2I6,4E16.6)
 9730     FORMAT(2I6,5E13.4)
 9731    FORMAT(4X,'IS',4X,'IT'
     & ,9X,'XS1',14X,'XT',12X,'EBFI',12X,'EBIS')
 9732    FORMAT(1X,'IST',3X,'ITT'
     & ,9X,'XS2',9X,'XS1T',8X,'XS3TT',10X,'XTT',10X,'EBF')
 9733    FORMAT(1X,'IST',1X,'ITT',1X,'XS1',6X,'XS2',6X,'XT',7X
     & ,'E1',7X,'E2',7X,'E3',7X,'E4',7X,'EB')
 9734    FORMAT(2I4,8E9.1)
 9735    FORMAT(5X,'WF1.1',10X,'WF2.1',10X,'WF3.1',10X,'WF4.1')
 9736    FORMAT(5X,'WF1.2',10X,'WF2.2',10X,'WF3.2',10X,'WF4.2')
 9737  FORMAT(1X,'IST',3X,'ITT',10X,'XS1',10X,'XS12',9X,'XTT',10X,'EBF')
 999  FORMAT(F10.4,1P5E14.5)
 9992  FORMAT(F8.4,1P6E12.4)
```

```
1001    FORMAT(I10,5E13.5/'————')
1002    FORMAT(I8,9F8.4)
9991  FORMAT(F8.4,1P6E12.3)
1003 FORMAT(/,30X,'STATES',10X,'ITR=',I5,//,6X,'T',
     &  6X,'X1=PH1',5X,'X2=DPH1',6X,'X3=PH2',5X,'X4=DPH2',9X,'UC1'
     &  ,9X,'UC2'/)
1006 FORMAT(/,30X,'COSTATES',10X,'ITR=',I5,//,6X,'T',
     &  7X,'P1~X1',7X,'P2~X2',7X,'P3~X3',7X,'P4~X4',9X,'UC1'
     &  ,9X,'UC2'/)
1004 FORMAT(/,30X,'COSTATES',10X,'ITR=',I5,//,4X,'T',
     &  8X,'X1=Y',7X,'X2=Y'',7X,'P1~X1',7X,'P2~X2',9X,'UC2',10X,'UC'/)
1005 FORMAT(16X,'STATES',22X,'ITR=',I2//5X,'T',
     &  3X,'X1=PH1',4X,'X2=DPH1',5X,'X3=PH2',4X,'X4=DPH2',6X,'UF1'
     &  ,6X,'UF2',10X,'H'/)
1008 FORMAT(16X,'COSTATES',22X,'ITR=',I2//5X,'T',
     &  4X,'P1~X1',6X,'P2~X2',6X,'P3~X3',6X,'P4~X4',6X,'UF1'
     &  ,6X,'UF2',10X,'H'/)
1007 FORMAT(F7.4,4E11.3,2F9.4,E11.3)
1012    FORMAT(42X,1P3E12.4)

10000 CONTINUE
END
CCCCC67-9———9———9———9———9———9———9———72
CCCCC67-END OF MAIN PROGRAM————————————————————72
CCCCC67-9———9———9———9———9———9———9———72

CCCCC67-9———9———9———9———9———9———9———72
CCCCC67-SUBPROGRAM FOR DEFINING STATES & COSTATES EQUATIONS AT LEFT-72
CCCCC67-9———9———9———9———9———9———9———72
SUBROUTINE FCNEQNL(NEQNS2,X,Y,P,DYDX)
INTEGER NEQNS2
DOUBLE PRECISION  X,Y(NEQNS2),P,DYDX(NEQNS2)
C—67—COMMON PARAMETERS————————————————————72
      INTEGER J11,KFJ,NSW,IEQ,IBC,IJB
DIMENSION KSW(10)
DOUBLE PRECISION
    +YBCL(8),YBCR(8),XS1(20),XS2(20),XS3(20),SIGN(8,20)
    +,YSCL(8,20),YSCU(8,20),YSLB(8,20),YSUB(8,20),UFW(10,2)
    +,C1,C2,C3,C4,C5,GR,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1
COMMON /PARAM/ YBCL,YBCR,XS1,XS2,XS3,SIGN
    +,YSCL,YSCU,YSLB,YSUB,UFW,C1,C2,C3,C4,C5,GR
    +,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1,KSW,J11,KFJ,NSW,IEQ,IBC,IJB
C—67—COMMON PARAMETERS————————————————————72

DOUBLE PRECISION X1,X2,X3,X4,P1,P2,P3,P4,T1,T2,A11,A12,A13,A22
   &,DE,G1,G2,G
DOUBLE PRECISION SIN,COS,SQRT
INTRINSIC SIN,COS,SQRT

C DEFINE PDE
G=GR
X1 = Y(1)
X2 = Y(2)
X3 = Y(3)
X4 = Y(4)
P1 = Y(5)
P2 = Y(6)
P3 = Y(7)
P4 = Y(8)
      A11=C1+C2+P**2*C3*COS(X3)
      A12=C2+P*C3*COS(X3)
      A13=P*C3*SIN(X3)
      A22=C2
      DE=A11*A22-A12*A12
    IF(J11.EQ.2) THEN
IEQ=IEQ+1
MIF=MOD(IEQ,100000)
IF(ABS(DE).LE.10.0E-10) THEN
       WRITE(27,'(/3I10,A)') J11,IEQ,MIF,'  WARNING DEY'
       WRITE(27,'(5E15.6)') X,(Y(J2),J2=1,NEQNS2/2)
       WRITE(27,'(5E15.6)') X,(Y(J2),J2=NEQNS2/2+1,NEQNS2)
END IF
DO J=1,NEQNS2
 IF(ABS(Y(J)).LT.10.0E-20.AND.ABS(Y(J)).NE.0.0) THEN
       WRITE(27,'(/3I10,A)') J11,IEQ,MIF,'  WARNING Y'
       WRITE(27,'(5E15.6)') X,(Y(J2),J2=1,NEQNS2/2)
       WRITE(27,'(5E15.6)') X,(Y(J2),J2=NEQNS2/2+1,NEQNS2)
GO TO 11
 ELSE IF(ABS(Y(J)).GT.10.0E+20) THEN
       WRITE(27,'(/3I10,A)') J11,IEQ,MIF,'  WARNING Y'
       WRITE(27,'(5E15.6)') X,(Y(J2),J2=1,NEQNS2/2)
       WRITE(27,'(5E15.6)') X,(Y(J2),J2=NEQNS2/2+1,NEQNS2)
GO TO 11
 END IF
END DO
11 CONTINUE
    END IF
```

```
      G1=(P2*A22-P4*A12)/DE
      G2=(-P2*A12+P4*A11)/DE
      IF(G1.LT.0.0) THEN
           T1=T1MAX
      ELSE
           T1=T1MIN
      END IF
      IF(G2.LT.0.0) THEN
           T2=T2MAX
      ELSE
           T2=T2MIN
      END IF
IF(J11.EQ.1) THEN
  IF(KFJ.EQ.10) THEN
IF(ABS(XS2(J11)).LE.SIGM1) THEN
 T2=SIGN(2,J11)
ELSE IF(ABS(XS2(J11)).GT.SIGM1) THEN
 IF(X.LE.XS2(J11)) THEN
T2=SIGN(2,J11)
 ELSE
T2=-SIGN(2,J11)
 END IF
END IF
T1=SIGN(1,J11)
  END IF
  IF(KFJ.EQ.11) THEN
   IF(XS1(J11).NE.0.0) THEN
IF(X.LE.XS1(J11)) THEN
T1=SIGN(1,J11)
ELSE
T1=SIGN(3,J11)
END IF
   ELSE
T1=SIGN(1,J11)
   END IF
IF(ABS(XS2(J11)).LE.SIGM1) THEN
 T2=SIGN(2,J11)
ELSE IF(ABS(XS2(J11)).GT.SIGM1) THEN
 IF(X.LE.XS2(J11)) THEN
T2=SIGN(2,J11)
 ELSE
T2=-SIGN(2,J11)
 END IF
END IF
  END IF

  IF(KFJ.EQ.31.OR.KFJ.EQ.30) THEN
T1=UFW(1,1)
T2=UFW(1,2)
  END IF
  IF(KFJ.EQ.32) THEN
T1=UFW(2,1)
T2=UFW(2,2)
  END IF
  IF(KFJ.EQ.33) THEN
T1=UFW(NSW,1)
T2=UFW(NSW,2)
  END IF
END IF
DYDX(1)=X2
DYDX(2)=
   +P/(C1*C2-C3**2*COS(X3)**2)*(-G*(C2*(C4*COS(X1)+C5*COS(X1+X3))-(C2+
   +C3*COS(X3))*C5*COS(X1+X3))+C2*T1-(C2+C3*COS(X3))*T2+C3*SIN(X3)*((C
   +2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2)))
DYDX(3)=X4
DYDX(4)=
   +P/(C1*C2-C3**2*COS(X3)**2)*(G*((C2+C3*COS(X3))*(C4*COS(X1)+C5*COS(
   +X1+X3))-(C1+C2+2*C3*COS(X3))*C5*COS(X1+X3))-(C2+C3*COS(X3))*T1+(C1
   ++C2+2*C3*COS(X3))*T2-C3*SIN(X3)*((C1+C2+2*C3*COS(X3))*X2**2+(C2+C3
   +*COS(X3))*(2*X2*X4+X4**2)))
     DYDX(5)=
   +-P*(-P2/(C1*C2-C3**2*COS(X3)**2)*G*(C2*(-C4*SIN(X1)-C5*SIN(X1+X3))
   ++(C2+C3*COS(X3))*C5*SIN(X1+X3))+P4/(C1*C2-C3**2*COS(X3)**2)*G*((C2
   ++C3*COS(X3))*(-C4*SIN(X1)-C5*SIN(X1+X3))+(C1+C2+2*C3*COS(X3))*C5*S
   +IN(X1+X3)))
     DYDX(6)=
   +-P*(P1+P2/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*(C2+C3*COS(X3))*X
   +2+2*C2*X4)-P4/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*(C1+C2+2*C3*C
   +OS(X3))*X2+2*(C2+C3*COS(X3))*X4))
     DYDX(7)=
   +-P*(-2*P2/(C1*C2-C3**2*COS(X3)**2)**2*(-G*(C2*(C4*COS(X1)+C5*COS(X
   +1+X3))-(C2+C3*COS(X3))*C5*COS(X1+X3))+C2*T1-(C2+C3*COS(X3))*T2+C3*
   +SIN(X3)*((C2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2)))*C3**2*COS(X3)*
   +SIN(X3)+P2/(C1*C2-C3**2*COS(X3)**2)*(-G*(-C2*C5*SIN(X1+X3)+C3*SIN(
   +X3)*C5*COS(X1+X3)+(C2+C3*COS(X3))*C5*SIN(X1+X3))+C3*SIN(X3)*T2+C3*
   +COS(X3)*((C2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2))-C3**2*SIN(X3)**
   +2*X2**2)-2*P4/(C1*C2-C3**2*COS(X3)**2)**2*(G*((C2+C3*COS(X3))*(C4*
```

```
     +COS(X1)+C5*COS(X1+X3))-(C1+C2+2*C3*COS(X3))*C5*COS(X1+X3))-(C2+C3*
     +COS(X3))*T1+(C1+C2+2*C3*COS(X3))*T2-C3*SIN(X3)*((C1+C2+2*C3*COS(X3
     +))*X2**2+(C2+C3*COS(X3))*(2*X2*X4+X4**2)))*C3**2*COS(X3)*SIN(X3)+P
     +4/(C1*C2-C3**2*COS(X3)**2)*(G*(-C3*SIN(X3)*(C4*COS(X1)+C5*COS(X1+X
     +3))-(C2+C3*COS(X3))*C5*SIN(X1+X3)+2*C3*SIN(X3)*C5*COS(X1+X3)+(C1+C
     +2+2*C3*COS(X3))*C5*SIN(X1+X3))+C3*SIN(X3)*T1-2*C3*SIN(X3)*T2-C3*CO
     +S(X3)*((C1+C2+2*C3*COS(X3))*X2**2+(C2+C3*COS(X3))*(2*X2*X4+X4**2))
     +-C3*SIN(X3)*(-2*C3*SIN(X3)*X2**2-C3*SIN(X3)*(2*X2*X4+X4**2))))
       DYDX(8)=
     +-P*(P3+P2/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*C2*(2*X2+2*X4)-P4/(C
     +1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(C2+C3*COS(X3))*(2*X2+2*X4))
      Y(1) = X1
      Y(2) = X2
      Y(3) = X3
      Y(4) = X4
      Y(5) = P1
      Y(6) = P2
      Y(7) = P3
      Y(8) = P4
      IF(J11.EQ.2) THEN
IF(MIF.EQ.1) THEN
        WRITE(27,'(/3I10,1E20.6)') J11,IEQ,MIF,DE
        WRITE(27,'(9E9.1)') X,(Y(J),J=1,NEQNS2)
        WRITE(27,'(9E9.1)') X,(DYDX(J),J=1,NEQNS2)
END IF
IF(ABS(X).LE.10.0E-15) THEN
        WRITE(27,'(/3I10,1E20.6)') J11,IEQ,MIF,DE
        WRITE(27,'(9E9.1)') X,(Y(J),J=1,NEQNS2)
        WRITE(27,'(9E9.1)') X,(DYDX(J),J=1,NEQNS2)
END IF
IF(ABS(DE).LE.10.0E-10) THEN
        WRITE(27,'(/3I10,A)') J11,IEQ,MIF,' WARNING DEDYDX'
        WRITE(27,'(5E15.6)') X,(DYDX(J2),J2=1,NEQNS2/2)
        WRITE(27,'(5E15.6)') X,(DYDX(J2),J2=NEQNS2/2+1,NEQNS2)
END IF
DO J=1,NEQNS2
  IF(ABS(DYDX(J)).LT.10.0E-20.AND.ABS(DYDX(J)).NE.0.0) THEN
        WRITE(27,'(/3I10,A)') J11,IEQ,MIF,' WARNING DYDX'
        WRITE(27,'(5E15.6)') X,(DYDX(J2),J2=1,NEQNS2/2)
        WRITE(27,'(5E15.6)') X,(DYDX(J2),J2=NEQNS2/2+1,NEQNS2)
GO TO 12
  ELSE IF(ABS(DYDX(J)).GT.10.0E+20) THEN
        WRITE(27,'(/3I10,A)') J11,IEQ,MIF,' WARNING DYDX'
        WRITE(27,'(5E15.6)') X,(DYDX(J2),J2=1,NEQNS2/2)
        WRITE(27,'(5E15.6)') X,(DYDX(J2),J2=NEQNS2/2+1,NEQNS2)
GO TO 12
  END IF
END DO
12 CONTINUE
    END IF
RETURN
END

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
CCCCC67-SUBPROGRAM FOR DEFINING STATES & COSTATES EQUATIONS AT RIGHT-72
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
SUBROUTINE FCNEQNU(NEQNS2,X,Y,P,DYDX)
INTEGER NEQNS2
DOUBLE PRECISION  X,Y(NEQNS2),P,DYDX(NEQNS2)

C---67---COMMON PARAMETERS-------------------------72
      INTEGER J11,KFJ,NSW,IEQ,IBC,IJB
DIMENSION KSW(10)
DOUBLE PRECISION
    +YBCL(8),YBCR(8),XS1(20),XS2(20),XS3(20),SIGN(8,20)
    +,YSCL(8,20),YSCU(8,20),YSLB(8,20),YSUB(8,20),UFW(10,2)
    +,C1,C2,C3,C4,C5,GR,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1
COMMON /PARAM/ YBCL,YBCR,XS1,XS2,XS3,SIGN
    +,YSCL,YSCU,YSLB,YSUB,UFW,C1,C2,C3,C4,C5,GR
    +,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1,KSW,J11,KFJ,NSW,IEQ,IBC,IJB
C---67---COMMON PARAMETERS-------------------------72

DOUBLE PRECISION X1,X2,X3,X4,P1,P2,P3,P4,T1,T2,A11,A12,A13,A22
    &,DE,G1,G2,G
DOUBLE PRECISION SIN,COS,SQRT
INTRINSIC SIN,COS,SQRT

C DEFINE PDE
G=GR
X1 = Y(1)
X2 = Y(2)
X3 = Y(3)
X4 = Y(4)
P1 = Y(5)
P2 = Y(6)
P3 = Y(7)
P4 = Y(8)
```

```
      A11=C1+C2+P**2*C3*COS(X3)
      A12=C2+P**C3*COS(X3)
      A13=P**C3*SIN(X3)
      A22=C2
      DE=A11*A22-A12**A12
    IF(J11.EQ.2) THEN
IBC=IBC+1
MIF=MOD(IBC,100000)
IF(ABS(DE).LE.10.0E-10) THEN
       WRITE(28,'(/3I10,A)') J11,IBC,MIF,'  WARNING DEY'
       WRITE(28,'(5E15.6)') X,(Y(J2),J2=1,NEQNS2/2)
       WRITE(28,'(5E15.6)') X,(Y(J2),J2=NEQNS2/2+1,NEQNS2)
END IF
DO J=1,NEQNS2
 IF(ABS(Y(J)).LT.10.0E-20.AND.ABS(Y(J)).NE.0.0) THEN
       WRITE(28,'(/3I10,A)') J11,IBC,MIF,'  WARNING Y'
       WRITE(28,'(5E15.6)') X,(Y(J2),J2=1,NEQNS2/2)
       WRITE(28,'(5E15.6)') X,(Y(J2),J2=NEQNS2/2+1,NEQNS2)
GO TO 11
 ELSE IF(ABS(Y(J)).GT.10.0E+20) THEN
       WRITE(28,'(/3I10,A)') J11,IBC,MIF,'  WARNING Y'
       WRITE(28,'(5E15.6)') X,(Y(J2),J2=1,NEQNS2/2)
       WRITE(28,'(5E15.6)') X,(Y(J2),J2=NEQNS2/2+1,NEQNS2)
GO TO 11
 END IF
END DO
11 CONTINUE
    END IF
      G1=(P2*A22-P4*A12)/DE
      G2=(-P2*A12+P4*A11)/DE
      IF(G1.LT.0.0) THEN
            T1=T1MAX
      ELSE
            T1=T1MIN
      END IF
      IF(G2.LT.0.0) THEN
            T2=T2MAX
      ELSE
            T2=T2MIN
      END IF
IF(J11.EQ.1) THEN
  IF(KFJ.EQ.20.OR.KFJ.EQ.21) THEN
IF(ABS(XS3(J11)).LE.SIGM1) THEN
 T2=SIGN(4,J11)
ELSE IF(ABS(XS3(J11)).GT.SIGM1) THEN
 IF(X.LE.XS3(J11)) THEN
T2=-SIGN(4,J11)
 ELSE
T2=SIGN(4,J11)
 END IF
END IF
T1=SIGN(3,J11)
  END IF
END IF
DYDX(1)=X2
DYDX(2)=
   +P/(C1*C2-C3**2*COS(X3)**2)*(-G*(C2*(C4*COS(X1)+C5*COS(X1+X3))-(C2+
   +C3*COS(X3))*C5*COS(X1+X3))+C2*T1-(C2+C3*COS(X3))*T2+C3*SIN(X3)*((C
   +2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2)))
DYDX(3)=X4
DYDX(4)=
   +P/(C1*C2-C3**2*COS(X3)**2)*(G*((C2+C3*COS(X3))*(C4*COS(X1)+C5*COS(
   +X1+X3))-(C1+C2+2*C3*COS(X3))*C5*COS(X1+X3))-(C2+C3*COS(X3))*T1+(C1
   ++C2+2*C3*COS(X3))*T2-C3*SIN(X3)*((C1+C2+2*C3*COS(X3))*X2**2+(C2+C3
   +*COS(X3))*(2*X2*X4+X4**2)))
     DYDX(5)=
   +-P*(-P2/(C1*C2-C3**2*COS(X3)**2)*G*(C2*(-C4*SIN(X1)-C5*SIN(X1+X3))
   ++(C2+C3*COS(X3))*C5*SIN(X1+X3))+P4/(C1*C2-C3**2*COS(X3)**2)*G*((C2
   ++C3*COS(X3))*(-C4*SIN(X1)-C5*SIN(X1+X3))+(C1+C2+2*C3*COS(X3))*C5*S
   +IN(X1+X3)))
     DYDX(6)=
   +-P*(P1+P2/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*(C2+C3*COS(X3))*X
   +2+2*C2*X4)-P4/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*(C1+C2+2*C3*C
   +OS(X3))*X2+2*(C2+C3*COS(X3))*X4))
     DYDX(7)=
   +-P*(-2*P2/(C1*C2-C3**2*COS(X3)**2)**2*(-G*(C2*(C4*COS(X1)+C5*COS(X
   +1+X3))-(C2+C3*COS(X3))*C5*COS(X1+X3))+C2*T1-(C2+C3*COS(X3))*T2+C3*
   +SIN(X3)*((C2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2)))*C3**2*COS(X3)*
   +SIN(X3)+P2/(C1*C2-C3**2*COS(X3)**2)*(-G*(-C2*C5*SIN(X1+X3)+C3*SIN(
   +X3)*C5*COS(X1+X3)+(C2+C3*COS(X3))*C5*SIN(X1+X3))+C3*SIN(X3)*T2+C3*
   +COS(X3)*((C2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2))-C3**2*SIN(X3)**
   +2*X2**2)-2*P4/(C1*C2-C3**2*COS(X3)**2)**2*(G*((C2+C3*COS(X3))*(C4*
   +COS(X1)+C5*COS(X1+X3))-(C1+C2+2*C3*COS(X3))*C5*COS(X1+X3))-(C2+C3*
   +COS(X3))*T1+(C1+C2+2*C3*COS(X3))*T2-C3*SIN(X3)*((C1+C2+2*C3*COS(X3
   +))*X2**2+(C2+C3*COS(X3))*(2*X2*X4+X4**2)))*C3**2*COS(X3)*SIN(X3)+P
   +4/(C1*C2-C3**2*COS(X3)**2)*(G*(-C3*SIN(X3)*(C4*COS(X1)+C5*COS(X1+X
   +3))-(C2+C3*COS(X3))*C5*SIN(X1+X3)+2*C3*SIN(X3)*C5*COS(X1+X3)+(C1+C
```

```
     +2+2*C3*COS(X3))*C5*SIN(X1+X3))+C3*SIN(X3)*T1-2*C3*SIN(X3)*T2-C3*CO
     +S(X3)*((C1+C2+2*C3*COS(X3))*X2**2+(C2+C3*COS(X3))*(2*X2*X4+X4**2))
     +-C3*SIN(X3)*(-2*C3*SIN(X3)*X2**2-C3*SIN(X3)*(2*X2*X4+X4**2))))
       DYDX(8)=
     +-P*(P3+P2/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*C2*(2*X2+2*X4)-P4/(C
     +1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(C2+C3*COS(X3))*(2*X2+2*X4))
      Y(1) = X1
      Y(2) = X2
      Y(3) = X3
      Y(4) = X4
      Y(5) = P1
      Y(6) = P2
      Y(7) = P3
      Y(8) = P4
     IF(J11.EQ.2) THEN
IF(MIF.EQ.1) THEN
        WRITE(28,'(/3I10,1E20.6)') J11,IBC,MIF,DE
        WRITE(28,'(9E9.1)') X,(Y(J),J=1,NEQNS2)
        WRITE(28,'(9E9.1)') X,(DYDX(J),J=1,NEQNS2)
END IF
IF(ABS(X).LE.10.0E-15) THEN
        WRITE(28,'(/3I10,1E20.6)') J11,IBC,MIF,DE
        WRITE(28,'(9E9.1)') X,(Y(J),J=1,NEQNS2)
        WRITE(28,'(9E9.1)') X,(DYDX(J),J=1,NEQNS2)
END IF
IF(ABS(DE).LE.10.0E-10) THEN
        WRITE(28,'(/3I10,A)') J11,IBC,MIF,' WARNING DEDYDX'
        WRITE(28,'(5E15.6)') X,(DYDX(J2),J2=1,NEQNS2/2)
        WRITE(28,'(5E15.6)') X,(DYDX(J2),J2=NEQNS2/2+1,NEQNS2)
END IF
DO J=1,NEQNS2
  IF(ABS(DYDX(J)).LT.10.0E-20.AND.ABS(DYDX(J)).NE.0.0) THEN
        WRITE(28,'(/3I10,A)') J11,IBC,MIF,' WARNING DYDX'
        WRITE(28,'(5E15.6)') X,(DYDX(J2),J2=1,NEQNS2/2)
        WRITE(28,'(5E15.6)') X,(DYDX(J2),J2=NEQNS2/2+1,NEQNS2)
GO TO 12
  ELSE IF(ABS(DYDX(J)).GT.10.0E+20) THEN
        WRITE(28,'(/3I10,A)') J11,IBC,MIF,' WARNING DYDX'
        WRITE(28,'(5E15.6)') X,(DYDX(J2),J2=1,NEQNS2/2)
        WRITE(28,'(5E15.6)') X,(DYDX(J2),J2=NEQNS2/2+1,NEQNS2)
GO TO 12
  END IF
END DO
12 CONTINUE
    END IF
RETURN
END

CCCCC67-9-------9-------9-------9-------9-------9-------9---72
CCCCC67-SUBPROGRAM FOR DEFINING STS & COSTS EQUS AFTER FINDING XS1---72
CCCCC67-9-------9-------9-------9-------9-------9-------9---72
SUBROUTINE FCNEQNC(NEQNS2,X,Y,P,DYDX)
INTEGER NEQNS2
DOUBLE PRECISION  X,Y(NEQNS2),P,DYDX(NEQNS2)

C---67---COMMON PARAMETERS-----------------------72
      INTEGER J11,KFJ,NSW,IEQ,IBC,IJB
DIMENSION KSW(10)
DOUBLE PRECISION
    +YBCL(8),YBCR(8),XS1(20),XS2(20),XS3(20),SIGN(8,20)
    +,YSCL(8,20),YSCU(8,20),YSLB(8,20),YSUB(8,20),UFW(10,2)
    +,C1,C2,C3,C4,C5,GR,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1
COMMON /PARAM/ YBCL,YBCR,XS1,XS2,XS3,SIGN
    +,YSCL,YSCU,YSLB,YSUB,UFW,C1,C2,C3,C4,C5,GR
    +,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1,KSW,J11,KFJ,NSW,IEQ,IBC,IJB
C---67---COMMON PARAMETERS-----------------------72

DOUBLE PRECISION X1,X2,X3,X4,P1,P2,P3,P4,T1,T2,A11,A12,A13,A22
    &,DE,G1,G2,G
DOUBLE PRECISION SIN,COS,SQRT
INTRINSIC SIN,COS,SQRT
C DEFINE PDE
G=GR
X1 = Y(1)
X2 = Y(2)
X3 = Y(3)
X4 = Y(4)
P1 = Y(5)
P2 = Y(6)
P3 = Y(7)
P4 = Y(8)
      A11=C1+C2+P*2*C3*COS(X3)
      A12=C2+P*C3*COS(X3)
      A13=P*C3*SIN(X3)
      A22=C2
      DE=A11*A22-A12*A12
C9CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
```

```
      G1=(P2*A22-P4*A12)/DE
      G2=(-P2*A12+P4*A11)/DE
      IF(G1.LT.0.0) THEN
           T1=T1MAX
      ELSE
           T1=T1MIN
      END IF
      IF(G2.LT.0.0) THEN
           T2=T2MAX
      ELSE
           T2=T2MIN
      END IF
 DYDX(1)=X2
 DYDX(2)=
     +P/(C1*C2-C3**2*COS(X3)**2)*(-G*(C2*(C4*COS(X1)+C5*COS(X1+X3))-(C2+
     +C3*COS(X3))*C5*COS(X1+X3))+C2*T1-(C2+C3*COS(X3))*T2+C3*SIN(X3)*((C
     +2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2)))
 DYDX(3)=X4
 DYDX(4)=
     +P/(C1*C2-C3**2*COS(X3)**2)*(G*((C2+C3*COS(X3))*(C4*COS(X1)+C5*COS(
     +X1+X3))-(C1+C2+2*C3*COS(X3))*C5*COS(X1+X3))-(C2+C3*COS(X3))*T1+(C1
     ++C2+2*C3*COS(X3))*T2-C3*SIN(X3)*((C1+C2+2*C3*COS(X3))*X2**2+(C2+C3
     +*COS(X3))*(2*X2*X4+X4**2)))
       DYDX(5)=
     +-P*(-P2/(C1*C2-C3**2*COS(X3)**2)*G*(C2*(-C4*SIN(X1)-C5*SIN(X1+X3))
     ++(C2+C3*COS(X3))*C5*SIN(X1+X3))+P4/(C1*C2-C3**2*COS(X3)**2)*G*((C2
     ++C3*COS(X3))*(-C4*SIN(X1)-C5*SIN(X1+X3))+(C1+C2+2*C3*COS(X3))*C5*S
     +IN(X1+X3)))
       DYDX(6)=
     +-P*(P1+P2/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*(C2+C3*COS(X3))*X
     +2+2*C2*X4)-P4/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*(C1+C2+2*C3*C
     +OS(X3))*X2+2*(C2+C3*COS(X3))*X4))
       DYDX(7)=
     +-P*(-2*P2/(C1*C2-C3**2*COS(X3)**2)**2*(-G*(C2*(C4*COS(X1)+C5*COS(X
     +1+X3))-(C2+C3*COS(X3))*C5*COS(X1+X3))+C2*T1-(C2+C3*COS(X3))*T2+C3*
     +SIN(X3)*((C2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2)))*C3**2*COS(X3)*
     +SIN(X3)+P2/(C1*C2-C3**2*COS(X3)**2)*(-G*(-C2*C5*SIN(X1+X3)+C3*SIN(
     +X3)*C5*COS(X1+X3)+(C2+C3*COS(X3))*C5*SIN(X1+X3))+C3*SIN(X3)*T2+C3*
     +COS(X3)*((C2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2))-C3**2*SIN(X3)**
     +2*X2**2)-2*P4/(C1*C2-C3**2*COS(X3)**2)**2*(G*((C2+C3*COS(X3))*(C4*
     +COS(X1)+C5*COS(X1+X3))-(C1+C2+2*C3*COS(X3))*C5*COS(X1+X3))-(C2+C3*
     +COS(X3))*T1+(C1+C2+2*C3*COS(X3))*T2-C3*SIN(X3)*((C1+C2+2*C3*COS(X3
     +))*X2**2+(C2+C3*COS(X3))*(2*X2*X4+X4**2)))*C3**2*COS(X3)*SIN(X3)+P
     +4/(C1*C2-C3**2*COS(X3)**2)*(G*(-C3*SIN(X3)*(C4*COS(X1)+C5*COS(X1+X
     +3))-(C2+C3*COS(X3))*C5*SIN(X1+X3)+2*C3*SIN(X3)*C5*COS(X1+X3)+(C1+C
     +2+2*C3*COS(X3))*C5*SIN(X1+X3))+C3*SIN(X3)*T1-2*C3*SIN(X3)*T2-C3*CO
     +S(X3)*((C1+C2+2*C3*COS(X3))*X2**2+(C2+C3*COS(X3))*(2*X2*X4+X4**2))
     +-C3*SIN(X3)*(-2*C3*SIN(X3)*X2**2-C3*SIN(X3)*(2*X2*X4+X4**2))))
       DYDX(8)=
     +-P*(P3+P2/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*C2*(2*X2+2*X4)-P4/(C
     +1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(C2+C3*COS(X3))*(2*X2+2*X4))
      Y(1) = X1
      Y(2) = X2
      Y(3) = X3
      Y(4) = X4
      Y(5) = P1
      Y(6) = P2
      Y(7) = P3
      Y(8) = P4
 RETURN
 END

 CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
 CCCCC67-SUBPROGRAM FOR BOUNDARY CONDITIONS OF LEFT STS & COSTS-----.72
 CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
 SUBROUTINE FCNBCSL (NEQNS2, YLEFT, YRIGHT, P, F)
 INTEGER NEQNS2
 DOUBLE PRECISION YLEFT(NEQNS2),YRIGHT(NEQNS2),P,F(NEQNS2)
 C---67---COMMON PARAMETERS-----------------------72
       INTEGER J11,KFJ,NSW,IEQ,IBC,IJB
 DIMENSION KSW(10)
 DOUBLE PRECISION
     +YBCL(8),YBCR(8),XS1(20),XS2(20),XS3(20),SIGN(8,20)
     +,YSCL(8,20),YSCU(8,20),YSLB(8,20),YSUB(8,20),UFW(10,2)
     +,C1,C2,C3,C4,C5,GR,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1
 COMMON /PARAM/ YBCL,YBCR,XS1,XS2,XS3,SIGN
     +,YSCL,YSCU,YSLB,YSUB,UFW,C1,C2,C3,C4,C5,GR
     +,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1,KSW,J11,KFJ,NSW,IEQ,IBC,IJB
 C---67---COMMON PARAMETERS-----------------------72

 C  DEFINE BOUNDARY CONDITIONS
 DO 11 J2=1,NEQNS2
   F(J2) = YLEFT(J2)-YSCL(J2,J11)
 11 CONTINUE
 RETURN
 END
```

```
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
CCCCC67-SUBPROGRAM FOR BOUNDARY CONDITIONS OF RIGHT STS & COSTS----72
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
SUBROUTINE FCNBCSU (NEQNS2, YLEFT, YRIGHT, P, F)
INTEGER NEQNS2
DOUBLE PRECISION YLEFT(NEQNS2),YRIGHT(NEQNS2),P,F(NEQNS2)
C---67---COMMON PARAMETERS-----------------------72
      INTEGER J11,KFJ,NSW,IEQ,IBC,IJB
DIMENSION KSW(10)
DOUBLE PRECISION
     +YBCL(8),YBCR(8),XS1(20),XS2(20),XS3(20),SIGN(8,20)
     +,YSCL(8,20),YSCU(8,20),YSLB(8,20),YSUB(8,20),UFW(10,2)
     +,C1,C2,C3,C4,C5,GR,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1
COMMON /PARAM/ YBCL,YBCR,XS1,XS2,XS3,SIGN
     +,YSCL,YSCU,YSLB,YSUB,UFW,C1,C2,C3,C4,C5,GR
     +,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1,KSW,J11,KFJ,NSW,IEQ,IBC,IJB
C---67---COMMON PARAMETERS-----------------------72

C  DEFINE BOUNDARY CONDITIONS
DO 11 J2=1,NEQNS2
  F(J2) = YLEFT(J2)-YSCU(J2,J11)
11 CONTINUE
RETURN
END

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
CCCCC67-SUBPROGRAM FOR BOUNDARY CONDITIONS OF LEFT INTERMEDIATE STS-72
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
SUBROUTINE FCNBCLI (NEQNS2, YLEFT, YRIGHT, P, F)
INTEGER NEQNS2
DOUBLE PRECISION YLEFT(NEQNS2), YRIGHT(NEQNS2), P, F(NEQNS2)
C---67---COMMON PARAMETERS-----------------------72
      INTEGER J11,KFJ,NSW,IEQ,IBC,IJB
DIMENSION KSW(10)
DOUBLE PRECISION
     +YBCL(8),YBCR(8),XS1(20),XS2(20),XS3(20),SIGN(8,20)
     +,YSCL(8,20),YSCU(8,20),YSLB(8,20),YSUB(8,20),UFW(10,2)
     +,C1,C2,C3,C4,C5,GR,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1
COMMON /PARAM/ YBCL,YBCR,XS1,XS2,XS3,SIGN
     +,YSCL,YSCU,YSLB,YSUB,UFW,C1,C2,C3,C4,C5,GR
     +,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1,KSW,J11,KFJ,NSW,IEQ,IBC,IJB
C---67---COMMON PARAMETERS-----------------------72

C  DEFINE BOUNDARY CONDITIONS
DO 11 J2=1,NEQNS2
  F(J2) = YLEFT(J2)-YSLB(J2,J11)
11 CONTINUE
RETURN
END

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
CCCCC67-SUBPROGRAM FOR BOUNDARY CONDITIONS OF RIGHT INTERMEDIATE STS-72
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
SUBROUTINE FCNBCUI (NEQNS2, YLEFT, YRIGHT, P, F)
INTEGER NEQNS2
DOUBLE PRECISION YLEFT(NEQNS2), YRIGHT(NEQNS2), P, F(NEQNS2)
C---67---COMMON PARAMETERS-----------------------72
      INTEGER J11,KFJ,NSW,IEQ,IBC,IJB
DIMENSION KSW(10)
DOUBLE PRECISION
     +YBCL(8),YBCR(8),XS1(20),XS2(20),XS3(20),SIGN(8,20)
     +,YSCL(8,20),YSCU(8,20),YSLB(8,20),YSUB(8,20),UFW(10,2)
     +,C1,C2,C3,C4,C5,GR,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1
COMMON /PARAM/ YBCL,YBCR,XS1,XS2,XS3,SIGN
     +,YSCL,YSCU,YSLB,YSUB,UFW,C1,C2,C3,C4,C5,GR
     +,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1,KSW,J11,KFJ,NSW,IEQ,IBC,IJB
C---67---COMMON PARAMETERS-----------------------72

C  DEFINE BOUNDARY CONDITIONS
DO 11 J2=1,NEQNS2
  F(J2) = YLEFT(J2)-YSUB(J2,J11)
11 CONTINUE
RETURN
END

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
CCCCC67-SUBPROGRAM FOR BOUNDARY CONDITIONS OF LEFT STS & COSTS----72
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
SUBROUTINE FCNBC (NEQNS2, YLEFT, YRIGHT, P, F)
INTEGER NEQNS2
DOUBLE PRECISION YLEFT(NEQNS2),YRIGHT(NEQNS2),P,F(NEQNS2)
DOUBLE PRECISION X1,X2,X3,X4,P1,P2,P3,P4,HL,HR,G1L,G1R,G2L,G2R
   &,DE,A11,A12,A13,A22,A14,A24,T1,T2
C---67---COMMON PARAMETERS-----------------------72
      INTEGER J11,KFJ,NSW,IEQ,IBC,IJB
DIMENSION KSW(10)
DOUBLE PRECISION
```

```
     +YBCL(8),YBCR(8),XS1(20),XS2(20),XS3(20),SIGN(8,20)
     +,YSCL(8,20),YSCU(8,20),YSLB(8,20),YSUB(8,20),UFW(10,2)
     +,C1,C2,C3,C4,C5,GR,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1
  COMMON /PARAM/ YBCL,YBCR,XS1,XS2,XS3,SIGN
     +,YSCL,YSCU,YSLB,YSUB,UFW,C1,C2,C3,C4,C5,GR
     +,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1,KSW,J11,KFJ,NSW,IEQ,IBC,IJB
  C—67—COMMON PARAMETERS————————————————72

  C  DEFINE BOUNDARY CONDITIONS
  G=GR
    IF(KFJ.EQ.31.OR.KFJ.EQ.30) THEN
  T1=UFW(1,1)
  T2=UFW(1,2)
    END IF
    IF(KFJ.EQ.32) THEN
  T1=UFW(2,1)
  T2=UFW(2,2)
    END IF
    IF(KFJ.EQ.33) THEN
  T1=UFW(NSW,1)
  T2=UFW(NSW,2)
    END IF
  X1=YLEFT(1)
  X2=YLEFT(2)
  X3=YLEFT(3)
  X4=YLEFT(4)
  P1=YLEFT(5)
  P2=YLEFT(6)
  P3=YLEFT(7)
  P4=YLEFT(8)

  DE = C1*C2-C3**2*COS(X3)**2
  A11 = C1+C2+2*C3*COS(X3)
  A12 = C2+C3*COS(X3)
  A13 = C3*SIN(X3)
  A22 = C2
  A14 = C4*COS(X1)+C5*COS(X1+X3)
  A24 = C5*COS(X1+X3)

       HL=
     &1+P1*X2+P3*X4+P2/DE*(-G*(A22*A14-A12*A24)+A22*T1-A12*T2+A13*(A12*X
     +2**2+A22*(2*X2*X4+X4**2)))+P4/DE*(G*(A12*A14-A11*A24)-A12*T1+A11*T
     +2-A13*(A11*X2**2+A12*(2*X2*X4+X4**2)))
       G1L=
     &0.1E1/DE*(P2*A22-P4*A12)
       G2L=
     &0.1E1/DE*(-P2*A12+P4*A11)

  X1=YRIGHT(1)
  X2=YRIGHT(2)
  X3=YRIGHT(3)
  X4=YRIGHT(4)
  P1=YRIGHT(5)
  P2=YRIGHT(6)
  P3=YRIGHT(7)
  P4=YRIGHT(8)

  DE = C1*C2-C3**2*COS(X3)**2
  A11 = C1+C2+2*C3*COS(X3)
  A12 = C2+C3*COS(X3)
  A13 = C3*SIN(X3)
  A22 = C2
  A14 = C4*COS(X1)+C5*COS(X1+X3)
  A24 = C5*COS(X1+X3)
       HR=
     &1+P1*X2+P3*X4+P2/DE*(-G*(A22*A14-A12*A24)+A22*T1-A12*T2+A13*(A12*X
     +2**2+A22*(2*X2*X4+X4**2)))+P4/DE*(G*(A12*A14-A11*A24)-A12*T1+A11*T
     +2-A13*(A11*X2**2+A12*(2*X2*X4+X4**2)))
       G1R=
     &0.1E1/DE*(P2*A22-P4*A12)
       G2R=
     &0.1E1/DE*(-P2*A12+P4*A11)

     IF(KFJ.EQ.30) THEN
  DO I=1,NEQNS2
  F(I) = YLEFT(I)-YBCL(I)
  END DO
     END IF
     IF(KFJ.EQ.31) THEN
  DO I=1,(NEQNS2/2)
  F(I) = YLEFT(I)-YBCL(I)
  END DO
  DO I=(NEQNS2/2+1),NEQNS2
  F(I) = YRIGHT(I)-YBCL(I)
  END DO
     END IF
     IF(KFJ.EQ.32) THEN
```

```
 DO I=1,(NEQNS2/2)
 F(I) = YLEFT(I)-YBCR(I)
 END DO
 IF((KSW(1).EQ.2).AND.(KSW(2).EQ.1)) THEN
   F(5) = G1R
   F(6) = HR
   F(7) = G2L
   F(8) = HL
 ELSE IF((KSW(1).EQ.1).AND.(KSW(2).EQ.2)) THEN
   F(5) = G1L
   F(6) = HR
   F(7) = G2R
   F(8) = HL
 END IF
     END IF
     IF(KFJ.EQ.33) THEN
 DO I=1,NEQNS2
 F(I) = YLEFT(I)-YBCR(I)
 END DO
     END IF
 RETURN
 END

 CCCCC67-9-----9-----9-----9-----9-----9-----9----72
 CCCCC67-SUBPROGRAM FOR DEFINING JACOBIAN-----------72
 CCCCC67-9-----9-----9-----9-----9-----9-----9----72
 SUBROUTINE FCNJACB (NEQNS2, X, Y, P, DYPDY)
 INTEGER NEQNS2
 DOUBLE PRECISION X, Y(NEQNS2), P, DYPDY(NEQNS2,NEQNS2)
 C---67---COMMON PARAMETERS-------------------------72
       INTEGER J11,KFJ,NSW,IEQ,IBC,IJB
 DIMENSION KSW(10)
 DOUBLE PRECISION
     +YBCL(8),YBCR(8),XS1(20),XS2(20),XS3(20),SIGN(8,20)
     +,YSCL(8,20),YSCU(8,20),YSLB(8,20),YSUB(8,20),UFW(10,2)
     +,C1,C2,C3,C4,C5,GR,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1
 COMMON /PARAM/ YBCL,YBCR,XS1,XS2,XS3,SIGN
     +,YSCL,YSCU,YSLB,YSUB,UFW,C1,C2,C3,C4,C5,GR
     +,T1MIN,T1MAX,T2MIN,T2MAX,SIGM1,KSW,J11,KFJ,NSW,IEQ,IBC,IJB
 C---67---COMMON PARAMETERS-------------------------72

 DOUBLE PRECISION X1,X2,X3,X4,P1,P2,P3,P4,T1,T2,A11,A12,A13,A22
     &,DE,G1,G2,G
 DOUBLE PRECISION SIN,COS,SQRT,SN3,CS3,CS1,SN13,CS13
 INTRINSIC SIN,COS,SQRT

 C4CCC67-THIS LINE/LINES SHOULD BE CHANGED FOR ECACH EXAMPLE-------72
 G=GR
 X1 = Y(1)
 X2 = Y(2)
 X3 = Y(3)
 X4 = Y(4)
 P1 = Y(5)
 P2 = Y(6)
 P3 = Y(7)
 P4 = Y(8)
       A11=C1+C2+P**2*C3*COS(X3)
       A12=C2+P*C3*COS(X3)
       A13=P*C3*SIN(X3)
       A22=C2
       DE=A11*A22-A12*A12
       G1=(P2*A22-P4*A12)/DE
       G2=(-P2*A12+P4*A11)/DE
       IF(G1.LT.0.0) THEN
             T1=T1MAX
       ELSE
             T1=T1MIN
       END IF
       IF(G2.LT.0.0) THEN
             T2=T2MAX
       ELSE
             T2=T2MIN
       END IF

IF(J11.EQ.1) THEN

   IF(KFJ.EQ.10) THEN
IF(ABS(XS2(J11)).LE.SIGM1) THEN
  T2=SIGN(2,J11)
ELSE IF(ABS(XS2(J11)).GT.SIGM1) THEN
  IF(X.LE.XS2(J11)) THEN
T2=SIGN(2,J11)
  ELSE
T2=-SIGN(2,J11)
  END IF
END IF
T1=SIGN(1,J11)
```

```
   END IF

   IF(KFJ.EQ.11) THEN
    IF(XS1(J11).NE.0.0) THEN
IF(X.LE.XS1(J11)) THEN
T1=SIGN(1,J11)
ELSE
T1=SIGN(3,J11)
END IF
    ELSE
T1=SIGN(1,J11)
    END IF
    IF(ABS(XS2(J11)).LE.SIGM1) THEN
T2=SIGN(2,J11)
    ELSE IF(ABS(XS2(J11)).GT.SIGM1) THEN
IF(X.LE.XS2(J11)) THEN
  T2=SIGN(2,J11)
ELSE
  T2=-SIGN(2,J11)
END IF
    END IF
   END IF

   IF(KFJ.EQ.20.OR.KFJ.EQ.21) THEN
IF(ABS(XS3(J11)).LE.SIGM1) THEN
  T2=SIGN(4,J11)
ELSE IF(ABS(XS3(J11)).GT.SIGM1) THEN
  IF(X.LE.XS3(J11)) THEN
T2=-SIGN(4,J11)
  ELSE
T2=SIGN(4,J11)
  END IF
END IF
T1=SIGN(3,J11)
   END IF

   IF(KFJ.EQ.31.OR.KFJ.EQ.30) THEN
T1=UFW(1,1)
T2=UFW(1,2)
   END IF
   IF(KFJ.EQ.32) THEN
T1=UFW(2,1)
T2=UFW(2,2)
   END IF
   IF(KFJ.EQ.33) THEN
T1=UFW(NSW,1)
T2=UFW(NSW,2)
   END IF
END IF

DO 11 I=1,NEQNS2
  DO 12 J=1,NEQNS2
  DYPDY(J,I)=0.0
12  CONTINUE
11  CONTINUE
      DYPDY(1,2)=1.0
      DYPDY(2,1)=
    +-P/(C1*C2-C3**2*COS(X3)**2)*G*(C2*(-C4*SIN(X1)-C5*SIN(X1+X3))+(C2+
    +C3*COS(X3))*C5*SIN(X1+X3))
      DYPDY(2,2)=
    +P/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*(C2+C3*COS(X3))*X2+2*C2*X
    +4)
      DYPDY(2,3)=
    +-2*P/(C1*C2-C3**2*COS(X3)**2)**2*(-G*(C2*(C4*COS(X1)+C5*COS(X1+X3)
    +)-(C2+C3*COS(X3))*C5*COS(X1+X3))+C2*T1-(C2+C3*COS(X3))*T2+C3*SIN(X
    +3)*((C2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2)))*C3**2*COS(X3)*SIN(X
    +3)+P/(C1*C2-C3**2*COS(X3)**2)*(-G*(-C2*C5*SIN(X1+X3)+C3*SIN(X3)*C5
    +*COS(X1+X3)+(C2+C3*COS(X3))*C5*SIN(X1+X3))+C3*SIN(X3)*T2+C3*COS(X3
    +)*((C2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2))-C3**2*SIN(X3)**2*X2**
    +2)
      DYPDY(2,4)=
    +P/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*C2*(2*X2+2*X4)
      DYPDY(3,4)=1.0
      DYPDY(4,1)=
    +P/(C1*C2-C3**2*COS(X3)**2)*G*((C2+C3*COS(X3))*(-C4*SIN(X1)-C5*SIN(
    +X1+X3))+(C1+C2+2*C3*COS(X3))*C5*SIN(X1+X3))
      DYPDY(4,2)=
    +-P/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*(C1+C2+2*C3*COS(X3))*X2+
    +2*(C2+C3*COS(X3))*X4)
      DYPDY(4,3)=
    +-2*P/(C1*C2-C3**2*COS(X3)**2)**2*(G*((C2+C3*COS(X3))*(C4*COS(X1)+C
    +5*COS(X1+X3))-(C1+C2+2*C3*COS(X3))*C5*COS(X1+X3))-(C2+C3*COS(X3))*
    +T1+(C1+C2+2*C3*COS(X3))*T2-C3*SIN(X3)*((C1+C2+2*C3*COS(X3))*X2**2+
    +(C2+C3*COS(X3))*(2*X2*X4+X4**2)))*C3**2*COS(X3)*SIN(X3)+P/(C1*C2-C
    +3**2*COS(X3)**2)*(G*(-C3*SIN(X3)*(C4*COS(X1)+C5*COS(X1+X3))-(C2+C3
    +*COS(X3))*C5*SIN(X1+X3)+2*C3*SIN(X3)*C5*COS(X1+X3)+(C1+C2+2*C3*COS
    +(X3))*C5*SIN(X1+X3))+C3*SIN(X3)*T1-2*C3*SIN(X3)*T2-C3*COS(X3)*((C1
```

```
     ++C2+2*C3*COS(X3))*X2**2+(C2+C3*COS(X3))*(2*X2*X4+X4**2))-C3*SIN(X3
     +)*(-2*C3*SIN(X3)*X2**2-C3*SIN(X3)*(2*X2*X4+X4**2)))
       DYPDY(4,4)=
     +-P/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(C2+C3*COS(X3))*(2*X2+2*X4)
       DYPDY(5,1)=
     +-P*(-P2/(C1*C2-C3**2*COS(X3)**2)*G*(C2*(-C4*COS(X1)-C5*COS(X1+X3))
     ++(C2+C3*COS(X3))*C5*COS(X1+X3))+P4/(C1*C2-C3**2*COS(X3)**2)*G*((C2
     ++C3*COS(X3))*(-C4*COS(X1)-C5*COS(X1+X3))+(C1+C2+2*C3*COS(X3))*C5*C
     +OS(X1+X3)))
       DYPDY(5,2)=0.0
       DYPDY(5,3)=
     +-P*(2*P2/(C1*C2-C3**2*COS(X3)**2)**2*G*(C2*(-C4*SIN(X1)-C5*SIN(X1+
     +X3))+(C2+C3*COS(X3))*C5*SIN(X1+X3))*C3**2*COS(X3)*SIN(X3)-P2/(C1*C
     +2-C3**2*COS(X3)**2)*G*(-C2*C5*COS(X1+X3)-C3*SIN(X3)*C5*SIN(X1+X3)+
     +(C2+C3*COS(X3))*C5*COS(X1+X3))-2*P4/(C1*C2-C3**2*COS(X3)**2)**2*G*
     +((C2+C3*COS(X3))*(-C4*SIN(X1)-C5*SIN(X1+X3))+(C1+C2+2*C3*COS(X3))*
     +C5*SIN(X1+X3))*C3**2*COS(X3)*SIN(X3)+P4/(C1*C2-C3**2*COS(X3)**2)*G
     +*(-C3*SIN(X3)*(-C4*SIN(X1)-C5*SIN(X1+X3))-(C2+C3*COS(X3))*C5*COS(X
     +1+X3)-2*C3*SIN(X3)*C5*SIN(X1+X3)+(C1+C2+2*C3*COS(X3))*C5*COS(X1+X3
     +)))
       DYPDY(5,4)=0.0
       DYPDY(5,5)=0.0
       DYPDY(5,6)=
     +P/(C1*C2-C3**2*COS(X3)**2)*G*(C2*(-C4*SIN(X1)-C5*SIN(X1+X3))+(C2+C
     +3*COS(X3))*C5*SIN(X1+X3))
       DYPDY(5,7)=0.0
       DYPDY(5,8)=
     +-P/(C1*C2-C3**2*COS(X3)**2)*G*((C2+C3*COS(X3))*(-C4*SIN(X1)-C5*SIN
     +(X1+X3))+(C1+C2+2*C3*COS(X3))*C5*SIN(X1+X3))
       DYPDY(6,1)=0.0
       DYPDY(6,2)=
     +-P*(P2/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*C2+2*C3*COS(X3))-P4/
     +(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*C1+2*C2+4*C3*COS(X3)))
       DYPDY(6,3)=
     +-P*(-2*P2/(C1*C2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*(2*(C2+C3*C
     +OS(X3))*X2+2*C2*X4)*COS(X3)+P2/(C1*C2-C3**2*COS(X3)**2)*C3*COS(X3)
     +*(2*(C2+C3*COS(X3))*X2+2*C2*X4)-2*P2/(C1*C2-C3**2*COS(X3)**2)*C3**
     +2*SIN(X3)**2*X2+2*P4/(C1*C2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*
     +(2*(C1+C2+2*C3*COS(X3))*X2+2*(C2+C3*COS(X3))*X4)*COS(X3)-P4/(C1*C2
     +-C3**2*COS(X3)**2)*C3*COS(X3)*(2*(C1+C2+2*C3*COS(X3))*X2+2*(C2+C3*
     +COS(X3))*X4)-P4/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(-4*C3*SIN(X3)
     +*X2-2*C3*SIN(X3)*X4))
       DYPDY(6,4)=
     +-P*(2*P2/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*C2-P4/(C1*C2-C3**2*CO
     +S(X3)**2)*C3*SIN(X3)*(2*C2+2*C3*COS(X3)))
       DYPDY(6,5)=-1.0
       DYPDY(6,6)=
     +-P/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*(C2+C3*COS(X3))*X2+2*C2*
     +X4)
       DYPDY(6,7)=0.0
       DYPDY(6,8)=
     +P/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*(C1+C2+2*C3*COS(X3))*X2+2
     +*(C2+C3*COS(X3))*X4)
       DYPDY(7,1)=
     +-P*(2*P2/(C1*C2-C3**2*COS(X3)**2)**2*G*(C2*(-C4*SIN(X1)-C5*SIN(X1+
     +X3))+(C2+C3*COS(X3))*C5*SIN(X1+X3))*C3**2*COS(X3)*SIN(X3)-P2/(C1*C
     +2-C3**2*COS(X3)**2)*G*(-C2*C5*COS(X1+X3)-C3*SIN(X3)*C5*SIN(X1+X3)+
     +(C2+C3*COS(X3))*C5*COS(X1+X3))-2*P4/(C1*C2-C3**2*COS(X3)**2)**2*G*
     +((C2+C3*COS(X3))*(-C4*SIN(X1)-C5*SIN(X1+X3))+(C1+C2+2*C3*COS(X3))*
     +C5*SIN(X1+X3))*C3**2*COS(X3)*SIN(X3)+P4/(C1*C2-C3**2*COS(X3)**2)*G
     +*(-C3*SIN(X3)*(-C4*SIN(X1)-C5*SIN(X1+X3))-(C2+C3*COS(X3))*C5*COS(X
     +1+X3)-2*C3*SIN(X3)*C5*SIN(X1+X3)+(C1+C2+2*C3*COS(X3))*C5*COS(X1+X3
     +)))
       DYPDY(7,2)=
     +-P*(-2*P2/(C1*C2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*(2*(C2+C3*C
     +OS(X3))*X2+2*C2*X4)*COS(X3)+P2/(C1*C2-C3**2*COS(X3)**2)*(C3*COS(X3
     +)*(2*(C2+C3*COS(X3))*X2+2*C2*X4)-2*C3**2*SIN(X3)**2*X2)+2*P4/(C1*C
     +2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*(2*(C1+C2+2*C3*COS(X3))*X2
     ++2*(C2+C3*COS(X3))*X4)*COS(X3)+P4/(C1*C2-C3**2*COS(X3)**2)*(-C3*CO
     +S(X3)*(2*(C1+C2+2*C3*COS(X3))*X2+2*(C2+C3*COS(X3))*X4)-C3*SIN(X3)*
     +(-4*C3*SIN(X3)*X2-2*C3*SIN(X3)*X4)))
CCCCC67-9-----9-----9-----9-----9-----9-----72
SN3=SIN(X3)
CS3=COS(X3)
CS1=COS(X1)
SN13=SIN(X1+X3)
CS13=COS(X1+X3)
       DYPDY(7,3)=
     +-P*(8*P2/(C1*C2-C3**2*CS3**2)**3*(-G*(C2*(C4*CS1+C5*CS13)
     +-(C2+C3*CS3)*C5*CS13)+C2*T1-(C2+C3*CS3)*T2+C3*SN3*
     +((C2+C3*CS3)*X2**2+C2*(2*X2*X4+X4**2)))*C3**4*CS3**2*SN3**2
     +-4*P2/(C1*C2-C3**2*CS3**2)**2*(-G*(-C2*C5*SN13+
     +C3*SN3*C5*CS13+(C2+C3*CS3)*C5*SN13)+C3*SN3*T2+
     +C3*CS3*((C2+C3*CS3)*X2**2+C2*(2*X2*X4+X4**2))-C3**2*SN3**2*X2**2)
     +*C3**2*CS3*SN3+2*P2/(C1*C2-C3**2*CS3**2)**2*(-G*(C2*(C4*CS1+
     +C5*CS13)-(C2+C3*CS3)*C5*CS13)+C2*T1-(C2+C3*CS3)*T2+
     +C3*SN3*((C2+C3*CS3)*X2**2+C2*(2*X2*X4+X4**2)))*C3**2*SN3**2-
```

```
     +2*P2/(C1*C2-C3**2*CS3**2)**2*(-G*(C2*(C4*CS1+C5*CS13)-
     +(C2+C3*CS3)*C5*CS13)+C2*T1-(C2+C3*CS3)*T2+C3*SN3*
     +((C2+C3*CS3)*X2**2+C2*(2*X2*X4+X4**2)))*C3**2*CS3**2+
     +P2/(C1*C2-C3**2*CS3**2)*(-G*(-C2*C5*CS13+C3*CS3*C5*CS13-
     +2*C3*SN3*C5*SN13+(C2+C3*CS3)*C5*CS13)+C3*CS3*T2-C3*SN3*
     +((C2+C3*CS3)*X2**2+C2*(2*X2*X4+X4**2))-3*C3**2*CS3*SN3*X2**2)+
     +8*P4/(C1*C2-C3**2*CS3**2)**3*(G*((C2+C3*CS3)*(C4*CS1+C5*CS13)-
     +(C1+C2+2*C3*CS3)*C5*CS13)-(C2+C3*CS3)*T1+(C1+C2+2*C3*CS3)*T2-
     +C3*SN3*((C1+C2+2*C3*CS3)*X2**2+(C2+C3*CS3)*(2*X2*X4+X4**2)))*
     +C3**4*CS3**2*SN3**2-4*P4/(C1*C2-C3**2*CS3**2)**2*(G*(-C3*SN3*
     +(C4*CS1+C5*CS13)-(C2+C3*CS3)*C5*SN13+2*C3*SN3*C5*CS13+
     +(C1+C2+2*C3*CS3)*C5*SN13)+C3*SN3*T1-2*C3*SN3*T2-C3*CS3*
     +((C1+C2+2*C3*CS3)*X2**2+(C2+C3*CS3)*(2*X2*X4+X4**2))-C3*SN3*(
     +-2*C3*SN3*X2**2-C3*SN3*(2*X2*X4+X4**2)))*C3**2*CS3*SN3+
     +2*P4/(C1*C2-C3**2*CS3**2)**2*(G*((C2+C3*CS3)*(C4*CS1+C5*CS13)-
     +(C1+C2+2*C3*CS3)*C5*CS13)-(C2+C3*CS3)*T1+(C1+C2+2*C3*CS3)*T2-
     +C3*SN3*((C1+C2+2*C3*CS3)*X2**2+(C2+C3*CS3)*(2*X2*X4+X4**2)))*
     +C3**2*SN3**2-2*P4/(C1*C2-C3**2*CS3**2)**2*(G*((C2+C3*CS3)*(C4*CS1+
     +C5*CS13)-(C1+C2+2*C3*CS3)*C5*CS13)-(C2+C3*CS3)*T1+(C1+C2+2*C3*CS3)
     +*T2-C3*SN3*((C1+C2+2*C3*CS3)*X2**2+(C2+C3*CS3)*(2*X2*X4+
     +X4**2)))*C3**2*CS3**2+P4/(C1*C2-C3**2*CS3**2)*(G*(-C3*CS3*
     +(C4*CS1+C5*CS13)-2*C3*SN3*C5*SN13-(C2+C3*CS3)*C5*CS13+2*C3*CS3*
     +C5*CS13+(C1+C2+2*C3*CS3)*C5*CS13)+C3*CS3*T1-2*C3*CS3*T2+C3*SN3*
     +((C1+C2+2*C3*CS3)*X2**2+(C2+C3*CS3)*(2*X2*X4+X4**2))-
     +2*C3*CS3*(-2*C3*SN3*X2**2-C3*SN3*(2*X2*X4+X4**2))-
     +C3*SN3*(-2*C3*CS3*X2**2-C3*CS3*(2*X2*X4+X4**2))))
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
      DYPDY(7,4)=
     +-P*(-2*P2/(C1*C2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*C2*(2*X2+2*
     +X4)*COS(X3)+P2/(C1*C2-C3**2*COS(X3)**2)*C3*COS(X3)*C2*(2*X2+2*X4)+
     +2*P4/(C1*C2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*(C2+C3*COS(X3))*
     +(2*X2+2*X4)*COS(X3)+P4/(C1*C2-C3**2*COS(X3)**2)*(-C3*COS(X3)*(C2+C
     +3*COS(X3))*(2*X2+2*X4)+C3**2*SIN(X3)**2*(2*X2+2*X4)))
       DYPDY(7,5)=0.0
       DYPDY(7,6)=
     +-P*(-2/(C1*C2-C3**2*COS(X3)**2)**2*(-G*(C2*(C4*COS(X1)+C5*COS(X1+X
     +3))-(C2+C3*COS(X3))*C5*COS(X1+X3))+C2*T1-(C2+C3*COS(X3))*T2+C3*SIN
     +(X3)*((C2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2)))*C3**2*COS(X3)*SIN
     +(X3)+1/(C1*C2-C3**2*COS(X3)**2)*(-G*(-C2*C5*SIN(X1+X3)+C3*SIN(X3)*
     +C5*COS(X1+X3)+(C2+C3*COS(X3))*C5*SIN(X1+X3))+C3*SIN(X3)*T2+C3*COS(
     +X3)*((C2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2))-C3**2*SIN(X3)**2*X2
     +**2))
       DYPDY(7,7)=0.0
       DYPDY(7,8)=
     +-P*(-2/(C1*C2-C3**2*COS(X3)**2)**2*(G*((C2+C3*COS(X3))*(C4*COS(X1)
     ++C5*COS(X1+X3))-(C1+C2+2*C3*COS(X3))*C5*COS(X1+X3))-(C2+C3*COS(X3)
     +)*T1+(C1+C2+2*C3*COS(X3))*T2-C3*SIN(X3)*((C1+C2+2*C3*COS(X3))*X2**
     +2+(C2+C3*COS(X3))*(2*X2*X4+X4**2)))*C3**2*COS(X3)*SIN(X3)+1/(C1*C2
     +-C3**2*COS(X3)**2)*(G*(-C3*SIN(X3)*(C4*COS(X1)+C5*COS(X1+X3))-(C2+
     +C3*COS(X3))*C5*SIN(X1+X3)+2*C3*SIN(X3)*C5*COS(X1+X3)+(C1+C2+2*C3*C
     +OS(X3))*C5*SIN(X1+X3))+C3*SIN(X3)*T1-2*C3*SIN(X3)*T2-C3*COS(X3)*((
     +C1+C2+2*C3*COS(X3))*X2**2+(C2+C3*COS(X3))*(2*X2*X4+X4**2))-C3*SIN(
     +X3)*(-2*C3*SIN(X3)*X2**2-C3*SIN(X3)*(2*X2*X4+X4**2))))
       DYPDY(8,1)=0.0
       DYPDY(8,2)=
     +-P*(2*P2/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*C2-2*P4/(C1*C2-C3**2*
     +COS(X3)**2)*C3*SIN(X3)*(C2+C3*COS(X3)))
       DYPDY(8,3)=
     +-P*(-2*P2/(C1*C2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*C2*(2*X2+2*
     +X4)*COS(X3)+P2/(C1*C2-C3**2*COS(X3)**2)*C3*COS(X3)*C2*(2*X2+2*X4)+
     +2*P4/(C1*C2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*(C2+C3*COS(X3))*
     +(2*X2+2*X4)*COS(X3)-P4/(C1*C2-C3**2*COS(X3)**2)*C3*COS(X3)*(C2+C3*
     +COS(X3))*(2*X2+2*X4)+P4/(C1*C2-C3**2*COS(X3)**2)*C3**2*SIN(X3)**2*
     +(2*X2+2*X4))
       DYPDY(8,4)=
     +-P*(2*P2/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*C2-2*P4/(C1*C2-C3**2*
     +COS(X3)**2)*C3*SIN(X3)*(C2+C3*COS(X3)))
       DYPDY(8,5)=0.0
       DYPDY(8,6)=
     +-P/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*C2*(2*X2+2*X4)
       DYPDY(8,7)=-1.0
       DYPDY(8,8)=
     +P/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(C2+C3*COS(X3))*(2*X2+2*X4)
      Y(1) = X1
      Y(2) = X2
      Y(3) = X3
      Y(4) = X4
      Y(5) = P1
      Y(6) = P2
      Y(7) = P3
      Y(8) = P4
RETURN
END
```

Appendix D

Computer Code for the Shooting Method

The computer code XRKOC7.FOR that implements the Shooting Method (SM) of Section 5.2 for time-optimal control of Two-Link Manipulators (TLM) is included and explained here. This program, which also uses the *Fortran* programming language, contains a main body and seventeen subroutines, and in total is **1911** lines long. Some of the subroutines of this code, which perform standard jobs, are taken from Numerical Recipes in Fortran [57], and with minor or major modification are used by the main program.

There is another computer code, XRKOC8, that does the same thing as XRKOC7. The only difference with XRKOC7 is that XRKOC8 uses adaptive step size 4th order Runge-Kutta method when calculating $\frac{\partial X(t_f)}{\partial B}$ for integration of Equation (5.17), while XRKOC7 uses constant step size 4th order Runge-Kutta method for integration of the ODE's.

The XRKOC7 algorithm is quite complicated. However, a brief description of main body of the program is as following. The program XRKOC7 uses the unknown initial costates generated by the Forward Backward Method (FBM) implemented by program OC2ADC13.FOR, and reads two input files XRKOC1.DAT and XRKOC2.DAT. The program then uses the SM and solve the associated Two Point Boundary Value Problem (TPBVP) with time-optimal control of TLM.

D.1 Input Files

The input data files XRKOC1.DAT, XRKOC2.DAT for initialising the SM are:

XRKOC1

$NITR$	$K1$	$K2$	KKM	SCG			
U_1^-	U_1^+	U_2^-	U_2^+				
ρ	$EBSIL$	$SIGMA$					
x_{10}	x_{20}	x_{30}	x_{40}				
x_{1f}	x_{2f}	x_{3f}	x_{4f}				
t_0							
B_1	B_2	B_3	B_4	t_f			
$EBSXO$	$H1XO$	SM	ρ_a	ρ_m	ρ_{min}	v_i^{min}	α_k^{max}
END							

XRKOC2

g_r				
l_1	l_2	l_{c1}	l_{c2}	
m_1	m_2	m_a	m_b	
I_1	I_2	I_0	I_a	I_b

Some of the parameters which are not defined before are:

NITR:	Number of maximum iterations
K1, K2:	Iteration numbers for plotting the (y_b, z_b) trajectory
KKM:	Maximum number of switches to be printed in output file TEMP9.OUT
SCG:	Scale factor for switch functions G_j
EBSIL:	Convergence criteria for $\|L[x(B, t_f), t_f]\|$
SIGMA:	Convergence criteria for calculating switch times from $G_j = 0$
EBSXO:	Criteria for boundary conditions for adaptive step size Runge-Kutta in XRKOC8
H1XO:	Initial value for adaptive step size Runge-Kutta in XRKOC8
SM:	The criteria of small in routine SVDCMP
ρ_a, ρ_m, ρ_{min}:	values for adjusting ρ
v_i^{min}:	Minimum allowable value for v_i
α_k^{max}:	Maximum allowable value for α_k

The input files XRKOC1.DAT, XRKOC2.DAT for Example Two of Section 6.1.2 are:

```
XRKOC1.DAT
----------

  70   1   2   7  250.0

 -1.000000E+01  1.000000E+01 -5.000000E+00  5.000000E+00

  4.000000E-02  1.000000E-05  1.000000E-11

  1.047000E+00  0.000000E+00  0.000000E+00  0.000000E+00

  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00

  0.000000E+00

 6.7684E-02 6.2063E-03 1.4805E-02 1.834003E-03 2.100000E-01

 1.00E-06 2.00E-01 1.00E-08 5.0E-03 0.0E+00 1.00E-02 1.00E-02 1.0E+00
END
```

```
XRKOC2.DAT
----------
  9.810000E+00
  2.000000E-01  2.000000E-01  1.000000E-01  1.000000E-01
  1.000000E+00  1.000000E+00  0.000000E+00  0.000000E+00
  4.167000E-03  4.167000E-03  0.000000E+00  0.000000E+00  0.00E+00
```

D.2 Output Files

The code creates several output files, some of the important ones are:

File name	Contents
TEMP3.OUT	Contents of input files XRKOC1, XRKOC2, plus switch times t_{si}, convergence parameters ρ, α_k, RLC= $\|L[x(B,t_f),t_f]\|$, RDBT= $\|[\delta B,\delta t_f]\|$, initial costates B_i and final time t_f for each iteration
TEMPS.OUT	Optimal states x_i, Hamiltonian, and optimal Control forces u_j in time if the program converges
TEMPC.OUT	Optimal costates p_i, Hamiltonian, and scaled switch functions G_j in time if the program converges
TEMPS1.OUT	Planar trajectory (y,z) of points corresponding to m_a, l_{c2}, m_b (see Figure 4.1) for three different iterations, if the program converges
TEMPS2.OUT	Links configuration in (y,z) plane for final iteration, if the program converges

However, there are other output files (TEMP1.OUT, TEMP2.OUT, TEMP4.OUT, TEMP9.OUT, TEMP91.OUT, TEMP92.OUT, TEMPC1.OUT) which prints different parameters or states, costates, and switch functions in different stages of the SM.

D.3 Weight Functions v_i

The weight functions v_i in Equation (6.11) are to accommodate the difference of the dimensions of the states as well as of the Hamiltonian. They are calculated and selected as follows. For convenience Equations (6.11) and (6.12) are repeated here.

$$L_j[x(B,t_f),t_f] = v_j \times [x_j(t_f) - x_{if}] \quad j = 1,\ldots,4$$
$$L_5[x(B,t_f),t_f] = v_5 \times \mathcal{H}(t_f) \tag{D.1}$$

and the target error norm $\|L\|$ is:

$$\|L\| = \sqrt{L_1^2 + L_2^2 + L_3^2 + L_4^2 + L_5^2} \tag{D.2}$$

The weight functions v_i $(i = 1,\ldots,5)$ are calculated as follows:

$$\bar{v}_j = max|x_j(t)|, \quad j = 1,\ldots,4 \qquad \bar{v}_5 = max|\mathcal{H}(t)|$$
$$v_i^{(k)} = \frac{(k-1) \times v_i^{(k-1)} + \bar{v}_i}{k}$$
$$v_{max} = max(\frac{1}{v_i^{(k)}})$$
$$v_i^* = \frac{\frac{1}{v_i^{(k)}}}{v_{max}} \qquad v_i = \frac{1}{v_i^*} \tag{D.3}$$

Note that here k is iteration number and the non-dimensional v_i^* is always $0 < v_i^* \leq 1$.

Presence of weight functions v_i in (D.1) insures that the target error norm $\|L\|$ is not dominated by the magnitude of one or more states or Hamiltonian, because of the difference in the magnitudes of rotations, the angular velocities, and the Hamiltonian. In this way the target error vector $L[x(B,t_f),t_f]$ is considered as a dimensionless vector. In Equation (D.1), weight functions v_i have been used rather than v_i^*. The reason for this choice is that the SM usually converges much faster with v_i than with v_i^*.

D.4 The Listing of the Program XRKOC7

The computer code XRKOC7 that implements the SM for time-optimal control of Two-Link Manipulators is as follows.

```
      PROGRAM XRKOC7
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C     DRIVER FOR ROUTINES RK4D,RK4H,RKDUMBH,DXSVB,SVDCMP,PYTHAG,SVBKSB
C     (CONSTANT STEP SIZE RUNGE-KUTTA 4TH ORDER METHOD)
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
      REAL SCG
      INTEGER K,K1,K2,JK1,JK2,IK1,IK2,KKM,IPR
      INTEGER N,M1,M2,NM,NMAX,NMXI,M1P,NSTEP
      DOUBLE PRECISION EBSXO,H1XO,SM
      PARAMETER(N=8,M1=N/2,M2=2,NM=N*M1,NMAX=35,NMXI=201,M1P=M1+1
     +,NSTEP=500)
        DIMENSION IS(NMAX),JSN(NMAX),ISW(NMAX)
        DOUBLE PRECISION V(N),DV(N),VS(N),VJ(N),DVJ(N),YS(N,NMAX),VF(N)
     1,DFDV(N,N),DFDY(N,N),XS(NMAX),CU(NSTEP+1,M2),G(NSTEP+1,M2)
     2,BTF(M1P,NMXI),XX(NSTEP+1),Y(N,NSTEP+1),KB(NM),XH(NSTEP+1)
     3,YH(NM,NSTEP+1),EBSIL,SIGMA
     4,FL(N,NMAX),FR(N,NMAX),F(N,NMAX),CS(2*M2)
     5,GM(M2),DGM(M2),DGX(M2,N),YHSL(NM),YHSR(NM),FS(N),LC(M1P),DYF(N)
     6,DLDX(M1P,N),DLDB(M1P,M1),DLDTF(M1P),YHF(NM),D(M1P,M1P),YF(N)
     7,BTFF(M1P),RLC(NMXI),RDBT(NMXI),RBT(NMXI),VSS(N)
     8,DBTC(M1P),AK(NMXI),ALK(NMXI),SIGST(NMAX),ROIA,ROIM,ROMIN,WSM,ALKM
     9,DT(M1P,M1P),DBT(M1P),DSIN,DCOS,DSQRT,ABS
     1,HAM,HA(NSTEP+1),WF(M1P),WFMAX,WFMIN,WFO(M1P),CPM,CPB,P2L,P4L
     2,XV(N),YV(N),XYSP(3,6,NSTEP+1)
     3,A11,A12,A13,A14,A24,P4G1,P2G2,DE,DEG1,DEG2,DISI(M1P),DIS(NMXI)
        DOUBLE PRECISION X,X1,X2,X2S,X3,XJ,XHJ,H,HJ,H2,HT
        DOUBLE PRECISION MHJ,HJ1,DMHJ,H0
     1,L1,L2,AM1,AM2,MA,MB,LC1,LC2,RI1,RI2,RI0,RIA,RIB
        DOUBLE PRECISION UCS(2),C1,C2,C3,C4,C5,GR
        COMMON /PATH/UCS,C1,C2,C3,C4,C5,GR
        CHARACTER TXT*3
        EXTERNAL DERIVS,DERIVSH,GDGTX,DTDBS,LDXBT,DLSBT,DYDBT0,UCONT
     +,DJACOB,HAM
        INTRINSIC DSIN,DCOS,DSQRT
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C    OUTPUT FILES INCLUDE INTERMEDIATE RESULTS OF EACH ITERATION
        OPEN(UNIT=1, FILE='temp1.out')
        OPEN(UNIT=2, FILE='temp2.out')
        OPEN(UNIT=3, FILE='temp3.out')
        OPEN(UNIT=4, FILE='temp4.out')
        OPEN(UNIT=9, FILE='temp9.out')
        OPEN(UNIT=91, FILE='temp91.out')
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C       INPUT FILE INCLUDE GEOMETERY OF THE MOTION AND CONTROL FORCES
        OPEN(UNIT=7,FILE='xrkoc1.dat',STATUS='OLD')
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
        READ(7,*)
        READ(7,*) NITR,K1,K2,KKM,SCG
        READ(7,*)
        READ(7,*) (CS(J),J=1,2*M2)
        READ(7,*)
        READ(7,*) RO,EBSIL,SIGMA
        READ(7,*)
        READ(7,*) (VS(J),J=1,M1)
        READ(7,*)
        READ(7,*) (VF(J),J=1,M1)
        READ(7,*)
        READ(7,*) X0
        READ(7,*)
        READ(7,*) (BTF(J,1),J=1,M1P)
        READ(7,*)
        READ(7,*) EBSXO,H1XO,SM,ROIA,ROIM,ROMIN,WSM,ALKM
        READ(7,'(A3)') TXT
        IF (TXT.NE.'END') THEN
          WRITE(*,*) ' DATA FILE IS INCOMPLETE, TRY A NEW SET OF DATA'
          GO TO 9900
        END IF
        CLOSE(UNIT=7)
        WRITE(*,*) '  DATA FILE (xrkoc1.dat) IS COMPLETE'

CCCCC67-9-----9-----9-----9-----9-----9-----9-----9-----72
C       INPUT FILE INCLUDE PHYSICAL PARAMETERS OF THE ARMS
        OPEN(UNIT=8,FILE='xrkoc2.dat',STATUS='OLD')
CCCCC67-9-----9-----9-----9-----9-----9-----9-----9-----72
        READ(8,*) GR,L1,L2,LC1,LC2,AM1,AM2,MA,MB,RI1,RI2,RI0,RIA,RIB
        CLOSE(UNIT=8)
```

```
      WRITE(*,*) '  DATA FILE (xrkoc2.dat) IS COMPLETE'
      WRITE(*,*) ' THE     PROGRAM  IS     RUNNING '

      WRITE(3,'(10X,A)') 'INPUT FILES "xrkoc1.dat" AND "xrkoc2.dat"'
      WRITE(3,'(8X,45A)') ('=',J=1,45)
     WRITE(3,'(T10,A,T21,A,T33,A,T43,A,T53,A,T68,A)')
    & 'NITR','K1(ITR)','K2(ITR)','KKM(NPTs)','SCG(scale,G)','(NSTEP)'
      WRITE(3,'(4I12,F12.4,I12)') NITR,K1,K2,KKM,SCG,NSTEP
     WRITE(3,'(T11,A,T25,A,T39,A,T53,A)')'U1^-','U1^+','U2^-','U2^+'
      WRITE(3,'(1P4E14.6)') (CS(J),J=1,2*M2)
     WRITE(3,'(T10,A,T24,A,T38,A)')'   RO','EBSIL','SIGMA'
      WRITE(3,'(1P4E14.6)') RO,EBSIL,SIGMA
     WRITE(3,'(T12,A,T26,A,T40,A,T54,A)')'X10','X20','X30','X40'
      WRITE(3,'(1P4E14.6)') (VS(J),J=1,M1)
     WRITE(3,'(T12,A,T26,A,T40,A,T54,A)')'X1F','X2F','X3F','X4F'
      WRITE(3,'(1P4E14.6)') (VF(J),J=1,M1)
     WRITE(3,'(T13,A)')'T0'
      WRITE(3,'(1P4E14.6)') X0
     WRITE(3,'(T13,A,T28,A,T43,A,T58,A,T73,A)')
    &'P10','P20','P30','P40',' TF'
      WRITE(3,'(1P5E15.6)') (BTF(J,1),J=1,M1P)
      WRITE(3,'(A3)') TXT
      WRITE(3,'(70A)') ('-',J=1,70)
     WRITE(3,'(T8,A)')'GRAVITY'
      WRITE(3,'(1P4E14.6)') GR
     WRITE(3,'(T12,A,T26,A,T40,A,T54,A)')' L1',' L2','LC1','LC2'
      WRITE(3,'(1P4E14.6)') L1,L2,LC1,LC2
     WRITE(3,'(T13,A,T27,A,T41,A,T55,A)')'M1','M2','MA','MB'
      WRITE(3,'(1P4E14.6)') AM1,AM2,MA,MB
     WRITE(3,'(T13,A,T27,A,T41,A,T55,A,T69,A)')'I1','I2','I0','IA','IB'
      WRITE(3,'(1P5E14.6)') RI1,RI2,RI0,RIA,RIB
     WRITE(3,'(T5,A,T16,A,T26,A,T36,A,T46,A,T55,A,T66,A,T74,A)')
    & 'EBSXO','H1XO','SMDX','ROIA','ROIM','ROMIN','WSM','ALKM'
     WRITE(3,'(1P7E10.2,1PE8.1)')EBSXO,H1XO,SM,ROIA,ROIM,ROMIN,WSM,ALKM
     WRITE(3,'(A)')'END'
      WRITE(3,'(78A)') ('=',J=1,78)

      C1=(RI0+RI1+RIA) + AM1*LC1*LC1 + (AM2+MA+MB)*L1*L1
      C2=(RI2+RIB) + AM2*LC2*LC2 + MB*L2*L2
      C3=(AM2*LC2+MB*L2)*L1
      C4=AM1*LC1 + (AM2+MA+MB)*L1
      C5=(AM2*LC2+MB*L2)

      WRITE(3,'(10X,A)') 'OUTPUT FILE "temp3.out" from "xrkoc7.for"'
      WRITE(3,'(8X,45A)') ('=',J=1,25)
      WRITE(3,'(2X,A,9X,A,11X,A,11X,A,11X,A,11X,A)') 'K','XS1','XS2'
    &,'XS3','XS4','...'
      WRITE(3,'(26X,A,12X,A,12X,A,11X,A)') 'RO','ALK','RLC','RDBT'
      WRITE(3,'(19X,55A)') ('-',J=1,55)
     WRITE(9,'(A,1X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X
    &,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A)')
    &'#','K','ALK','RLC',' RD',' B1',' B2',' B3',' B4',' B5'
    &,' Y1',' Y2',' Y3',' Y4','G10','G20','XS1','XS2','XS3','XS4','...'
     WRITE(91,'(A,1X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X,A,7X
    &,A,7X,A,7X,A,7X,A,7X,A)')'#','K',' W1',' W2',' W3',' W4',' W5'
    &,'LC1','LC2','LC3','LC4','LC5','HT0','HTF','CPM','CPB'
      JK1=5
      JK2=10
      DO 125 I=1,M1P
       WFO(I)=0.0D0
125     CONTINUE
CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
C    STARTING THE MAIN ITERATION LOOP
CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
      WRITE(*,*) ' STARTING THE MAIN ITERATION LOOP'
     DO 1 K=1,NITR
      WRITE(*,'(/A,I4)') ' THE PROGRAM IS RUNNING, STARTING INTERATION
    & NO ',K

CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
C    INTEGRATION OF STATES & COSTATES USING GUESS VALUES FOR COSTATES
     WRITE(*,*) ' INTEGRATION OF STATES & COSTATES USING GUESS VALUES'
      H=BTF(M1P,K)/NSTEP
      KK=0
      IS(0)=0
      X=X0
      DO 110 J=1,KKM
       XS(J)=0.0D0
110     CONTINUE
      DO 2 J=1,M1
      J2=M1+J
2     VS(J2)= BTF(J,K)

CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
C      MODIFYING THE INITIAL COSTATES(P2,P4) FORCING H(t0)=0
      P2L=VS(6)
      P4L=VS(8)
```

```
      CALL UCONT(VS,CS,UCS)
      CALL DERIVS(N,X,VS,DV)
          HA(1)=HAM(M1,X,VS,DV)
      CPM=-DV(2)/DV(4)
      CPB=-1.0D0/DV(4)
      WRITE(4,*) 'MODIFYING THE INITIAL COSTATES(P2,P4) FORCING H=0'
      WRITE(4,'(1P2E15.6)') CPM,CPB
      WRITE(4,991) X,(DV(J),J=1,M1)
      WRITE(4,991) X,(VS(J),J=M1+1,N),HA(1),(UCS(J),J=1,M2)

      RBT(K)=0.0D0
      DO 3 I=1,M1P
3     RBT(K)=RBT(K)+BTF(I,K)*BTF(I,K)
      RBT(K)=DSQRT(RBT(K))
      AK(K)=MAX(RBT(K),1.0D0)
      RDBT(1)=AK(1)*EBSIL+10.0D0*EBSIL

      DO 10 J=1,N
         Y(J,1)=VS(J)
10       V(J)=VS(J)
      XX(1)=X
      CALL GDGTX(N,M2,X,V,GM,DGM,DGX,DERIVS)
      DO 101 J=1,M2
101    G(1,J)=GM(J)
      DO 11 I=1,NSTEP
      CALL UCONT(V,CS,UCS)
      DO 112 J=1,M2
112      CU(I,J)=UCS(J)
      CALL DERIVS(N,X,V,DV)
          HA(I)=HAM(M1,X,V,DV)
      CALL STPA(L1,L2,LC1,LC2,V,XV,YV)
      DO 114 J=1,3
       J1=2*J-1
       J2=2*J
     IF(K1.GE.1.AND.K2.GE.1) THEN
      IF(K.LT.K1.AND.K.LT.K2) THEN
       IF(K.EQ.JK1) THEN
          XYSP(1,J1,I)=XV(J)
          XYSP(1,J2,I)=YV(J)
       ELSE IF(K.EQ.JK2) THEN
          XYSP(2,J1,I)=XV(J)
          XYSP(2,J2,I)=YV(J)
       END IF
      ELSE IF(K.GE.K1.AND.K.LT.K2) THEN
       IF(K.EQ.K1) THEN
          XYSP(1,J1,I)=XV(J)
          XYSP(1,J2,I)=YV(J)
       ELSE IF(K.EQ.JK1.OR.K.EQ.JK2) THEN
          XYSP(2,J1,I)=XV(J)
          XYSP(2,J2,I)=YV(J)
       END IF
      ELSE IF(K.GE.K1.AND.K.GE.K2) THEN
       IF(K.EQ.K1) THEN
         XYSP(1,J1,I)=XV(J)
         XYSP(1,J2,I)=YV(J)
       ELSE IF(K.EQ.K2) THEN
         XYSP(2,J1,I)=XV(J)
         XYSP(2,J2,I)=YV(J)
       END IF
      END IF
     ELSE IF(K1.LE.0.AND.K2.LE.0) THEN
       IF(K.EQ.JK1) THEN
         XYSP(1,J1,I)=XV(J)
         XYSP(1,J2,I)=YV(J)
       ELSE IF(K.EQ.JK2) THEN
         XYSP(2,J1,I)=XV(J)
         XYSP(2,J2,I)=YV(J)
       END IF
     END IF
      XYSP(3,J1,I)=XV(J)
      XYSP(3,J2,I)=YV(J)
114      CONTINUE
      CALL RK4D(V,DV,N,X,H,V,DERIVS)
      X=X+H
      XX(I+1)=X
      DO 13 J=1,N
13     Y(J,I+1)=V(J)
      CALL GDGTX(N,M2,X,V,GM,DGM,DGX,DERIVS)
      DO 131 JG=1,M2
       G(I+1,JG)=GM(JG)

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C        CHECKING THE SIGN OF SWITHCHING FUNCTION
      IF( G(I,JG)*G(I+1,JG).LT.0.0D0 ) THEN
         JT=0
         KK=KK+1
      WRITE(*,'(A,I4)') ' FINDING THE SWITCH TIME NO ',KK
```

```
      XJ=XX(I)
      H0=-H*G(I,JG)/(G(I+1,JG)-G(I,JG))
      HJ=H0

      DO 14 J1=1,N
14     VJ(J1)=Y(J1,I)
15     CALL UCONT(VJ,CS,UCS)
      JT=JT+1
      CALL DERIVS(N,XJ,VJ,DVJ)
      CALL RK4D(VJ,DVJ,N,XJ,HJ,VJ,DERIVS)

      XHJ=XJ+HJ
      CALL UCONT(VJ,CS,UCS)
      CALL GDGTX(N,M2,XHJ,VJ,GM,DGM,DGX,DERIVS)
      MHJ=GM(JG)
      DMHJ=DGM(JG)
      HJ1=-MHJ/DMHJ

      IF(MIN(ABS(MHJ),ABS(HJ1)).LT.SIGMA) THEN
        WRITE(*,'(A,I4)')' FOUND SWITCH TIME NO ',KK
        SIGST(KK)=MAX(ABS(MHJ),ABS(HJ1))
        SIGST(KK)=10.D0*MAX(SIGST(KK),SIGMA)
        WRITE(4,'(/5X,A,10X,A,10X,A,9X,A,I1,A)')
     &   '—MHJ—','—HJ1—','SIGMA','SIGST(',KK,')'
        WRITE(4,'(1P5E15.7)') ABS(MHJ),ABS(HJ1),SIGMA,SIGST(KK)
        JSN(KK)=JT
        IS(KK)=I
        ISW(KK)=JG
        XS(KK)=XHJ
        DO 16 J1=1,N
16       YS(J1,KK)=VJ(J1)
        H2=SIGMA
        CALL DERIVS(N,XHJ,VJ,DVJ)
        CALL RK4D(VJ,DVJ,N,XHJ,H2,VJ,DERIVS)
        XHJ=XHJ+H2
        H2=XX(I+1)-XHJ
        CALL UCONT(VJ,CS,UCS)
        CALL DERIVS(N,XHJ,VJ,DVJ)
        CALL RK4D(VJ,DVJ,N,XHJ,H2,V,DERIVS)
        DO 161 J1=1,N
161      Y(J1,I+1)=V(J1)
        CALL GDGTX(N,M2,X,V,GM,DGM,DGX,DERIVS)
        G(I+1,JG)=GM(JG)
        GO TO 17
      END IF

      XJ=XHJ
      HJ=HJ1
      GO TO 15
17   END IF
131  CONTINUE
11   CONTINUE
     CALL UCONT(V,CS,UCS)
     DO 113 J=1,M2
113     CU(NSTEP+1,J)=UCS(J)
     CALL DERIVS(N,X,V,DYF)
     DO 111 J=1,N
111   YF(J)=V(J)
        HA(NSTEP+1)=HAM(M1,X,YF,DYF)
     CALL STPA(L1,L2,LC1,LC2,YF,XV,YV)
     DO 116 J=1,3
      J1=2*J-1
      J2=2*J
   IF(K1.GE.1.AND.K2.GE.1) THEN
    IF(K.LT.K1.AND.K.LT.K2) THEN
     IF(K.EQ.JK1) THEN
        IK1=JK1
        XYSP(1,J1,NSTEP+1)=XV(J)
        XYSP(1,J2,NSTEP+1)=YV(J)
     ELSE IF(K.EQ.JK2) THEN
        IK2=JK2
        XYSP(2,J1,NSTEP+1)=XV(J)
        XYSP(2,J2,NSTEP+1)=YV(J)
     END IF
    ELSE IF(K.GE.K1.AND.K.LT.K2) THEN
     IF(K.EQ.K1) THEN
        IK1=K1
        XYSP(1,J1,NSTEP+1)=XV(J)
        XYSP(1,J2,NSTEP+1)=YV(J)
     ELSE IF(K.EQ.JK1.OR.K.EQ.JK2) THEN
        IF(K.EQ.JK1) IK2=JK1
        IF(K.EQ.JK2) IK2=JK2
        XYSP(2,J1,NSTEP+1)=XV(J)
        XYSP(2,J2,NSTEP+1)=YV(J)
     END IF
    ELSE IF(K.GE.K1.AND.K.GE.K2) THEN
     IF(K.EQ.K1) THEN
```

```
          IK1=K1
          XYSP(1,J1,NSTEP+1)=XV(J)
          XYSP(1,J2,NSTEP+1)=YV(J)
        ELSE IF(K.EQ.K2) THEN
          IK2=K2
          XYSP(2,J1,NSTEP+1)=XV(J)
          XYSP(2,J2,NSTEP+1)=YV(J)
        END IF
      END IF
     ELSE IF(K1.LE.0.AND.K2.LE.0) THEN
        IF(K.EQ.JK1) THEN
          IK1=JK1
          XYSP(1,J1,NSTEP+1)=XV(J)
          XYSP(1,J2,NSTEP+1)=YV(J)
        ELSE IF(K.EQ.JK2) THEN
          IK2=JK2
          XYSP(2,J1,NSTEP+1)=XV(J)
          XYSP(2,J2,NSTEP+1)=YV(J)
        END IF
     END IF
      XYSP(3,J1,NSTEP+1)=XV(J)
      XYSP(3,J2,NSTEP+1)=YV(J)
116     CONTINUE
       IF(K.EQ.JK1) JK1=JK1+10
       IF(K.EQ.JK2) JK2=JK2+10

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
      WRITE(4,*) 'MAXIMUM VALUES OF STATES & HA. FOR SCALING THE LCs'
      DO 115 J=1,M1P
        WF(J)=0.0D0
115     CONTINUE
      DO 119 I=1,(NSTEP+1)
        DO 117 J=1,M1
          WF(J)=MAX(WF(J),ABS(Y(J,I)))
117       CONTINUE
        WF(M1P)=MAX(WF(M1P),ABS(HA(I)))
119     CONTINUE
      DO 120 I=1,M1P
        WF(I)=(WFO(I)*(K-1)+WF(I))/K
        WFO(I)=WF(I)
120     CONTINUE
      WFMAX=0.0D0
      DO 121 I=1,M1P
        WF(I)=1.0D0/WF(I)
        WFMAX=MAX(WFMAX,WF(I))
121     CONTINUE
      DO 123 I=1,M1P
        WF(I)=WF(I)/WFMAX
123     CONTINUE
      WRITE(4,'(11X,A,11X,A,11X,A,11X,A,11X,A)')
     & 'WF(1)','WF(2)','WF(3)','WF(4)','WF(5)'
      WRITE(4,'(6E16.8)') (WF(J),J=1,M1P)
      WRITE(4,*) 'SATURATION OF WS FOR SCALING THE LCs'
      WFMAX=0.0D0
      WFMIN=1.0D0
      DO 124 I=1,M1P
        WFMAX=MAX(WFMAX,WF(I))
        WFMIN=MIN(WFMIN,WF(I))
124     CONTINUE
      DO 1241 I=1,M1P
       IF(WF(I).LT.WSM) WF(I)=(1.0D0+(WF(I)-WFMIN)/(WFMAX-WFMIN))*WSM
1241   CONTINUE
      IF(WSM.GT.0.5D0) THEN
        DO 1242 I=1,M1P
          WF(I)=1.0D0
1242      CONTINUE
      END IF
      WRITE(4,'(11X,A,11X,A,11X,A,11X,A,11X,A)')
     &'WF(1)','WF(2)','WF(3)','WF(4)','WF(5)'
      WRITE(4,'(6E16.8)') (WF(J),J=1,M1P)

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C     PRINTING THE RESULTS OF INTEGRATION OF STATES & COSTATES
      WRITE(*,*)' PRINTING THE RESULTS OF INTEGRATION OF STATES & C.'
     WRITE(1,'(20X,A,I5/20X,A)')'ITR=',K,'-----'
      WRITE(1,'(6X,A,5X,A,4X,A,5X,A,4X,A,10X,A,8X,A,8X,A)')
     & 'T','X1=PH1','X2=DPH1','X3=PH2','X4=DPH2','H','U1','U2'
     WRITE(2,'(20X,A,I5/20X,A)')'ITR=',K,'-----'
      WRITE(2,'(6X,A,5X,A,4X,A,5X,A,4X,A,10X,A,8X,A,8X,A)')
     & 'T','P1-PH1','P2-DPH1','P3-PH2','P4-DPH2','H','G1','G2'
      DO 21 I=1,(NSTEP+1)
      WRITE(1,991) XX(I),(Y(J,I),J=1,M1),HA(I),(CU(I,J),J=1,M2)
      WRITE(2,991) XX(I),(Y(J,I),J=M1+1,N),HA(I),(G(I,J),J=1,M2)
21     CONTINUE
       WRITE(1,992)
       WRITE(2,992)
       DO 19 J=1,KK
```

```
        WRITE(1,993) J,IS(J),ISW(J),XS(J),(YS(J1,J),J1=1,M1)
        WRITE(2,993) J,IS(J),ISW(J),XS(J),(YS(J1,J),J1=M1+1,N)
19      CONTINUE

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C    CHEKING THE CONVERGENCE CRITERIA
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
      WRITE(*,*)' CHEKING THE CONVERGENCE CRITERIA'
      X2=BTF(M1P,K)
      CALL UCONT(YF,CS,UCS)
     CALL LDXBT(N,M1,X2,YF,VF,DFDY,LC,DLDX,DLDB,DLDTF,WF,DERIVS,DJACOB)
      WRITE(4,'(/A,I4)') 'STATES AT FINAL TIME AT ITR NO',K
      WRITE(4,'(12X,A,10X,A,10X,A,10X,A,10X,A,5X,A,5X,A)') 'X'
    &,'YF1','YF2','YF3','YF4','U1','U2'
     WRITE(4,'(5E13.5,2F7.1)')X2,(YF(J),J=1,M1),(CU(NSTEP+1,J),J=1,M2)
     WRITE(4,'(5E13.4,2F7.1)')X2,(YF(J),J=M1+1,N)
      RLC(K)=0.0D0
      DO 191 I=1,M1P
       RLC(K)=RLC(K)+LC(I)*LC(I)
191     CONTINUE
      RLC(K)=DSQRT(RLC(K))
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
      IF( MAX(RDBT(K)/AK(K),RLC(K)).LT.EBSIL) THEN
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
        DO 192 I=1,M1P
192      BTFF(I)=BTF(I,K)
        WRITE(*,*) ' THE PROGRAM CONVERGED AFTER'
        WRITE(*,'(5X,I3,5X,A)') K, 'ITERATION'
        WRITE(1,*) ' THE PROGRAM CONVERGED AFTER'
        WRITE(1,'(5X,I3,5X,A)') K, 'ITERATION'
        WRITE(1,1000)
        WRITE(1,1001) K,(BTFF(I),I=1,M1P)
        WRITE(3,'(I3,1P10E14.6)') K,(XS(L),L=1,KK)
        WRITE(3,'(18X,1PE10.2,1P3E15.6)') RO,ALK(K),RLC(K),RDBT(K)
        WRITE(3,'(78A)') ('-',J=1,78)
        WRITE(3,1000)
        DO 193 I=1,K
         WRITE(3,1001) I,(BTF(J,I),J=1,M1P)
193     CONTINUE
        WRITE(3,1000)
        WRITE(3,1001) K,(BTFF(I),I=1,M1P)
        WRITE(3,'(78A)') ('=',J=1,78)
        WRITE(3,'(T5,A,I3,T15,A,I3,T35,A,I3,T45,A,I3,T70,A,I3)')
   &     'JK1=',JK1-10,'JK2=',JK2-10,'IK1=',IK1,'IK2=',IK2,' KF=',K

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C PRINTING THE DISTANCE OF CONVERGED INITIAL COSTATES & INITIAL COSTATES

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C    OUTPUT FILE INCLUDE INTERMEDIATE RESULTS OF EACH ITERATION
       OPEN(UNIT=92, FILE='temp92.out')
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
       WRITE(92,'(A,1X,A,9X,A,9X,A,9X,A,9X,A,9X,A,9X,A)')
   &    '#','K',' D',' D1',' D2',' D3',' D4',' D5'
       DO 1931 I=1,K
        DIS(I)=0.0D0
        DO 1932 J=1,M1P
         DISI(J)=0.0D0
         DISI(J)=ABS(BTFF(J)-BTF(J,I))
         DIS(I)=DISI(J)*DISI(J)+DIS(I)
1932       CONTINUE
        DIS(I)=DSQRT(DIS(I))
        WRITE(92,'(I3,1P6E12.4)')I,DIS(I),(DISI(J),J=1,M1P)
        IF( I.EQ.1) THEN
         WRITE(3,'(78A)') ('-',J=1,78)
         WRITE(3,'(A,1X,A,9X,A,9X,A,9X,A,9X,A,9X,A,9X,A)')
   &      '#','K',' D',' D1',' D2',' D3',' D4',' D5'
         WRITE(3,'(I3,1P6E12.4)')I,DIS(I),(DISI(J),J=1,M1P)
        END IF
1931    CONTINUE

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C       PRINTING THE CONVERGED RESULTS OF STATES & COSTATES

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C    OUTPUT FILES INCLUDE OPTIMAL STATES, COSTATES, AND COSTATE PRODUCTS
       OPEN(UNIT=5,FILE='temps.out')
       OPEN(UNIT=6,FILE='tempc.out')
       OPEN(UNIT=61,FILE='tempc1.out')
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
       WRITE(5,'(A,5X,A,5X,A,4X,A,5X,A,4X,A,10X,A,8X,A,8X,A)')
   &    '#','T','X1=PH1','X2=DPH1','X3=PH2','X4=DPH2','H','U1','U2'
    WRITE(6,'(A,5X,A,5X,A,4X,A,5X,A,4X,A,10X,A,1X,F7.3,A,1X,F7.3,A)')'
   &#','T','P1=PH1','P2=DPH1','P3=PH2','P4=DPH2','H',SCG,'G1',SCG,'G2'
    WRITE(61,'(A,T7,A,T16,A,T27,A,T38,A,T49,A,T61,A,T71,A,T80,A,T91,A,
   &T102,A,T115,A,T126,A,T135,A,T146,A)')'#','T','A11','A12','A13','A1
   &4','C2','A24','DELTA','P4G1','P2G2','P2','P4','DEG1','DEG2'
```

```
          DO 194 I=1,(NSTEP+1)
            A11=C1+C2+2*C3*DCOS(Y(3,I))
            A12=C2+C3*DCOS(Y(3,I))
            A13=C3*DSIN(Y(3,I))
            A14=C4*DCOS(Y(1,I))+C5*DCOS(Y(1,I)+Y(3,I))
            A24=C5*DCOS(Y(1,I)+Y(3,I))
            DE=C1*C2-C3**2*DCOS(Y(3,I))**2
            DEG1=G(I,1)*DE/C2
            P4G1=Y(8,I)*A12/C2
            DEG2=G(I,2)*DE/A11
            P2G2=Y(6,I)*A12/A11
            WRITE(61,9911) XX(I),A11,A12,A13,A14,C2,A24,DE
     +                    ,P4G1,P2G2,Y(6,I),Y(8,I),DEG1,DEG2
            WRITE(5,991) XX(I),(Y(J,I),J=1,M1),HA(I),(CU(I,J),J=1,M2)
          WRITE(6,991) XX(I),(Y(J,I),J=M1+1,N),HA(I),(SCG*G(I,J),J=1,M2)
194         CONTINUE
            WRITE(5,994)
            WRITE(6,994)
            DO 195 J=1,KK
             WRITE(5,995) '#',J,IS(J),ISW(J),XS(J),(YS(J1,J),J1=1,M1)
             WRITE(6,995) '#',J,IS(J),ISW(J),XS(J),(YS(J1,J),J1=M1+1,N)
195         CONTINUE
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C    OUTPUT FILE INCLUDE TRAJECTORIES OF 3 ITERATIONS OF THE HAND
          OPEN(UNIT=51,FILE='temps1.out')
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
       WRITE(51,'(A,T5,A,I3,T20,A,I3,T35,A,I3)')'#'
     &,'IK1=',IK1,'IK2=',IK2,' KF=',K
      WRITE(51,'(A,4X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A
     &,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A)')'#'
     &,' X1.1',' Y1.1','X2C.1','Y2C.1',' X2.1',' Y2.1'
     &,' X1.2',' Y1.2','X2C.2','Y2C.2',' X2.2',' Y2.2'
     &,' X1.F',' Y1.F','X2C.F','Y2C.F',' X2.F',' Y2.F'
          IPR=NSTEP/15
          DO 196 I=1,(NSTEP+1),IPR
           WRITE(51,'(1P18E10.2)') ((XYSP(L,J,I),J=1,6),L=1,3)
196         CONTINUE
            WRITE(51,'(1P18E10.2)') ((XYSP(L,J,NSTEP+1),J=1,6),L=1,3)
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C    OUTPUT FILE INCLUDE OPTIMAL TRAJECTORY OF THE HAND
          OPEN(UNIT=52,FILE='temps2.out')
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
       WRITE(52,'(A,4X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A
     &,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A,5X,A)')'#'
     &,'  X.1','  Y.1','  X.3','  Y.3','  X.5','  Y.5'
     &,'  X.7','  Y.7','  X.9','  Y.9',' X.11',' Y.11'
     &,' X.13',' Y.13',' X.15',' Y.15',' X.17',' Y.17'
          IPR=IPR*2
       WRITE(52,'(18A)') ((' 0.00E-01',J=1,2),I=1,(NSTEP+1),IPR)
     &,(' 0.00E-01',J=1,2)
       WRITE(52,'(1P18E10.2)') ((XYSP(3,J,I),J=1,2),I=1,(NSTEP+1),IPR)
     &,(XYSP(3,J,NSTEP+1),J=1,2)
       WRITE(52,'(1P18E10.2)') ((XYSP(3,J,I),J=5,6),I=1,(NSTEP+1),IPR)
     &,(XYSP(3,J,NSTEP+1),J=5,6)
          CLOSE(UNIT=5)
          CLOSE(UNIT=6)
          CLOSE(UNIT=51)
          CLOSE(UNIT=52)
          CLOSE(UNIT=61)
          CLOSE(UNIT=92)
          GOTO 9900
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
        END IF
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C  CALCULATING THE PARTIAL DERIVATIVES OF Y'S WRT B'S (DY/DB)
        WRITE(*,*)' CALCULATING THE PARTIAL DERIVATIVES OF (DY/DB)'
C  INTEGRATION OF DY/DB FROM X0-XS(1)
        WRITE(*,*)' INTEGRATION OF DY/DB FROM X0-XS(1)'
        CALL DYDBT0(N,M1,KB)
        X1=X0
        DO 371 I=1,N
        VSS(I)=VS(I)
371      CONTINUE

        DO 40 L=1,KK
        X2S=XS(L)
        X2=XS(L)-SIGST(L)
        X3=XS(L)+SIGST(L)
        JS=IS(L)-IS(L-1)
        WRITE(*,'(A,I2,A,I4)') ' JS OF SWITCH NO (',L,')= ', JS
        IF(JS.LE.0) THEN
          JS=NSTEP
          WRITE(*,'(A,I2,A,I4)')' MODIFIED JS OF SWITCH NO(',L,')= ',JS
        END IF
        CALL UCONT(VSS,CS,UCS)
```

```
      WRITE(4,'(/A,2I4)')'STATES AT SWITCHING TIMES AT SWITCH NOS',L-1,L
      WRITE(4,'(12X,A,10X,A,10X,A,10X,A,10X,A,5X,A,5X,A)') 'X'
     &,'VS1','VS2','VS3','VS4','U1','U2'
      WRITE(4,'(5E13.5,2F7.1)')X1,(VSS(J),J=1,M1),(UCS(J),J=1,M2)
      WRITE(4,'(5E13.4,2F7.1)')X1,(VSS(J),J=M1+1,N)
       CALL RKDUMBH(N,NM,VSS,DFDV,KB,X1,X2,JS,XH,YH,DERIVS,DERIVSH)
      WRITE(4,'(5E13.5,2F7.1)')X2,(VSS(J),J=1,M1),(UCS(J),J=1,M2)
      WRITE(4,'(5E13.4,2F7.1)')X2,(VSS(J),J=M1+1,N)
      WRITE(4,'(12X,A,10X,A,10X,A,10X,A,10X,A,5X,A,5X,A)') 'X'
     &,'YS1','YS2','YS3','YS4','U1','U2'
      WRITE(4,'(5E13.5,2F7.1)')X2,(YS(J,L),J=1,M1),(UCS(J),J=1,M2)
      WRITE(4,'(5E13.4,2F7.1)')X2,(YS(J,L),J=M1+1,N)

      WRITE(4,'(/A)')' INTEGRATION OF DY/DB FROM XS(KK-1)-XS(KK)'
      WRITE(4,'(9X,A,E14.6,5X,A,I4,A,E14.6//9X,A,13X,A,10X,A)')
     &'XH(1)=',XH(1),'XH(',JS+1,')=',XH(JS+1),'I','YH`1(I)','YH`JS+1(I)'
       DO 27 J=1,NM
        WRITE(4,998) J,YH(J,1),YH(J,JS+1)
27      CONTINUE

CCCCC67-9-----9-----9-----9-----9-----9-----72
C     CALCULATING THE JUMP VECTOR F AT SWHICHING TIMES
      WRITE(*,*)' CALCULATING THE JUMP VECTOR F AT SWHICHING TIMES'
      WRITE(4,'(/6X,A,I3,10X,A,I3)') 'ITR=',K,'SWITCH NO=',L
      HT=-SIGST(L)
      DO 29 I=1,N
29     V(I)=YS(I,L)
      CALL UCONT(V,CS,UCS)
      CALL DERIVS(N,X2S,V,DV)
      CALL RK4D(V,DV,N,X2S,HT,V,DERIVS)
      CALL UCONT(V,CS,UCS)
      CALL DERIVS(N,X2,V,DV)
     WRITE(4,'(/10X,A,10X,A,10X,A,10X,A,4X,A,4X,A,6X,A)')
     & 'X','DYS2','DYS4','DYS7','U1','U2','(-HT)HT'
     WRITE(4,'(F11.8,3E14.6,2F6.1,E13.5)')
     &X2,DV(2),DV(4),DV(7),(UCS(J),J=1,M2),HT
      DO 30 I=1,N
      FL(I,L)=DV(I)
30     V(I)=YS(I,L)
      HT=SIGST(L)
      CALL UCONT(V,CS,UCS)
      CALL DERIVS(N,X2S,V,DV)
      CALL RK4D(V,DV,N,X2,HT,V,DERIVS)
      CALL UCONT(V,CS,UCS)
      CALL DERIVS(N,X3,V,DV)
     WRITE(4,'(F11.8,3E14.6,2F6.1,E13.5)')
     &X3,DV(2),DV(4),DV(7),(UCS(J),J=1,M2),HT
      WRITE(4,'(/20X,A/)') 'THE F VECTOR'
      DO 31 I=1,N
      FR(I,L)=DV(I)
      F(I,L)=FL(I,L)-FR(I,L)
      FS(I)=F(I,L)
      VSS(I)=V(I)
      WRITE(4,'(20X,A,I2,A,I1,A,1PE20.10)') 'F(',I,',',L,') =',F(I,L)
31     CONTINUE
CCCCC67-9-----9-----9-----9-----9-----9-----72
C     CALCULATING THE INTIAL CONDITION FOR (DY/DB) AFTER SWITCHING TIME
     WRITE(*,*)' CALCULATING THE INT COND FOR (DY/DB) AFTER SW TIME'
      DO 32 I=1,NM
32     YHSL(I)=YH(I,JS+1)
      CALL GDGTX(N,M2,X3,VSS,GM,DGM,DGX,DERIVS)
      ISF=ISW(L)
      CALL DTDBS(N,M1,M2,NM,ISF,YHSL,DGM,DGX,FS,YHSR)
      WRITE(4,'(/5X,A,23X,A/)') 'YH BEFORE JUMP','YH AFTER JUMP'
      DO 33 I=1,NM
      WRITE(4,'(5X,A,I2,A,1PE20.10,5X,A,I2,A,1PE20.10)') 'YHSL(',I
    &,') =',YHSL(I),'YHSR(',I,') =',YHSR(I)
      KB(I)=YHSR(I)
33     CONTINUE
      X1=X3
40     CONTINUE
      X2=BTF(M1P,K)
      JS=NSTEP-IS(KK)
     WRITE(*,'(A,I2,A,I4)') ' JS TO END AFTER SWITCH NO (',KK,')= ',JS
      IF(JS.LE.0) THEN
       JS=NSTEP
       WRITE(*,'(A,I2,A,I4)')' MODIFIED JS OF SWITCH NO(',KK,')= ',JS
      END IF
      CALL UCONT(VSS,CS,UCS)
     WRITE(4,'(/A,2I4)')'STATES AT SWITCHING TIMES AT SWITCH NOS',L-1,L
      WRITE(4,'(12X,A,10X,A,10X,A,10X,A,10X,A,5X,A,5X,A)') 'X'
    &,'YS1','YS2','YS3','YS4','U1','U2'
     WRITE(4,'(5E13.5,2F7.1)')X1,(VSS(J),J=1,M1),(UCS(J),J=1,M2)
     WRITE(4,'(5E13.4,2F7.1)')X1,(VSS(J),J=M1+1,N)
      CALL RKDUMBH(N,NM,VSS,DFDV,KB,X1,X2,JS,XH,YH,DERIVS,DERIVSH)
     WRITE(4,'(5E13.5,2F7.1)')X2,(VSS(J),J=1,M1),(UCS(J),J=1,M2)
     WRITE(4,'(5E13.4,2F7.1)')X2,(VSS(J),J=M1+1,N)
```

```
      WRITE(4,'(12X,A,10X,A,10X,A,10X,A,10X,A,5X,A,5X,A)') 'X'
     &,'YF1','YF2','YF3','YF4','U1','U2'
      WRITE(4,'(5E13.5,2F7.1)')X2,(YF(J),J=1,M1),(CU(NSTEP+1,J),J=1,M2)
      WRITE(4,'(5E13.4,2F7.1)')X2,(YF(J),J=M1+1,N)
      WRITE(4,'(/A)')' INTEGRATION OF DY/DB FROM XS(KK)-XF'
      WRITE(4,'(9X,A,E14.6,5X,A,I4,A,E14.6//9X,A,13X,A,10X,A)')
     &'XH(1)=',XH(1),'XH(',JS+1,')=',XH(JS+1),'I','YH`1(I)','YH`JS+1(I)'
        DO 42 J=1,NM
          WRITE(4,998) J,YH(J,1),YH(J,JS+1)
42      CONTINUE

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C     CALCULATING THE DEVIATION MATRIX D AT THE END
      WRITE(*,*) ' CALCULATING THE DEVIATION MATRIX D AT THE END  '
        DO 44 I=1,NM
44       YHF(I)=YH(I,JS+1)
        CALL DLSBT(N,M1,YHF,DLDX,DLDB,DLDTF,DYF,D)
        DO 46 I=1,M1P
        DO 48 J=1,M1P
            DT(J,I)=D(J,I)
48        CONTINUE
         DBT(I)=LC(I)
46       CONTINUE
        WRITE(4,'(/A)') 'THE DEVIATION MATRIX AT FINAL TIME'
        WRITE(4,'(9X,A,9X,A,9X,A,9X,A,9X,A)')
     &'DT(1,J)','DT(2,J)','DT(3,J)','DT(4,J)','DT(5,J)'
        DO 45 I=1,M1P
45        WRITE(4,'(6E16.8)') (DT(I,J),J=1,M1P)

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C     CALLING SUBROUTINE DXSVB FOR SOLVING LINEAR ALG. EQUATION DT*DBT=LC
        CALL DXSVB(M1P,SM,DT,LC,DBT)
         WRITE(4,'(/A)')'THE SOLUTION & COMPARISON'
         WRITE(4,'(10X,A,10X,A,10X,A,10X,A,10X,A)')
     &'DBT(1)','DBT(2)','DBT(3)','DBT(4)','DBT(5)'
         WRITE(4,'(6E16.8)') (DBT(J),J=1,M1P)
         WRITE(4,'(A)')'ORIGINAL RIGHT-HAND SIDE VECTOR LC.'
         WRITE(4,'(11X,A,11X,A,11X,A,11X,A,11X,A)')
     &'LC(1)','LC(2)','LC(3)','LC(4)','LC(5)'
         WRITE(4,'(6E16.8)') (LC(J),J=1,M1P)
        DO 47 I=1,M1P
        LC(I)=0.0D0
        DO 49 J=1,M1P
         LC(I)=LC(I)+D(I,J)*DBT(J)
49      CONTINUE
47      CONTINUE
        WRITE(4,'(A)')'RESULT OF (MATRIX)*(SOL N VECTOR):'
        WRITE(4,'(11X,A,11X,A,11X,A,11X,A,11X,A)')
     &'LC(1)','LC(2)','LC(3)','LC(4)','LC(5)'
        WRITE(4,'(6E16.8)') (LC(J),J=1,M1P)
      WRITE(4,*) '*********************************'

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C     MODIFYING THE INTIAL GUESS FOR COSATATES & FINIAL TIME
      WRITE(*,*)' MODIFYING THE INTIAL GUESS FOR COSATATE & FINIAL TIME'
       RDBT(K+1)=0.0D0
       DO 51 I=1,M1P
51       RDBT(K+1)=RDBT(K+1)+DBT(I)*DBT(I)
       RDBT(K+1)=DSQRT(RDBT(K+1))
       ALK(K)=MIN(ALKM,RO*RBT(K)/RDBT(K+1))
       DO 52 I=1,M1P
         DBTC(I)=-ALK(K)*DBT(I)
         BTF(I,K+1)=BTF(I,K)+DBTC(I)
52      CONTINUE
       WRITE(4,'(/A)')'RESULTS OF MODIFIED PARAMETERS'
       WRITE(4,'(10X,A,14X,A,13X,A)')'ALK(K)','RLC(K)','RDBT(K+1)'
       WRITE(4,'(1P3E20.10)') ALK(K),RLC(K),RDBT(K+1)
        WRITE(4,'(10X,A,10X,A,10X,A,10X,A,10X,A)')
     &'BTF(1)','BTF(2)','BTF(3)','BTF(4)','BTF(5)'
        WRITE(4,'(1P5E16.8)') (BTF(J,K+1),J=1,M1P)
        WRITE(4,'(9X,A,9X,A,9X,A,9X,A,9X,A)')
     &'BTFO(1)','BTFO(2)','BTFO(3)','BTFO(4)','BTFO(5)'
        WRITE(4,'(1P5E16.8)') (BTF(J,K),J=1,M1P)
            WRITE(4,'(25X,A,5X,I3)') 'END OF ITERATION=',K
       WRITE(4,'(40A)') ('-',J=1,40)
       WRITE(3,'(I3,1P10E14.6)') K,(XS(L),L=1,KK)
       WRITE(3,'(18X,1PE10.2,1P3E15.6)') RO,ALK(K),RLC(K),RDBT(K+1)
       WRITE(3,'(19X,55A)') ('-',J=1,55)
      WRITE(9,'(I3,1P22E10.2)')K,ALK(K),(RLC(K)/M1P),(RDBT(K)/M1P)
     &,(BTF(J,K),J=1,M1P),(YF(J),J=1,M1),(G(1,J),J=1,M2),(XS(L),L=1,KKM)
      WRITE(91,'(I3,1P14E10.2)')K,(WF(J),J=1,M1P),(LC(J),J=1,M1P)
     &,HA(1),HA(NSTEP+1),CPM,CPB

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C       ADJUSTING THE STEP SIZE FACTOR RO
      IF(ALK(K).NE.ALKM) THEN
        IF(K.EQ.1) THEN
```

```
       IF(RLC(K).LT.5.0D3) THEN
           RO=RO+ROIA
       ELSE IF(RLC(K).GT.5.0D3) THEN
           RO=RO-ROIA
       END IF
      ELSE IF(K.EQ.2) THEN
       IF(RLC(K).LT.RLC(K-1)) THEN
           RO=RO+ROIA
       ELSE IF(RLC(K).GT.RLC(K-1)) THEN
           RO=RO-ROIA
       END IF
      ELSE IF(K.GT.2) THEN
       IF(RLC(K).LT.RLC(K-1).AND.RLC(K-1).LT.RLC(K-2) ) THEN
           RO=RO+ROIA+RO*ROIM
           GO TO 53
       ELSE IF(RLC(K).GT.RLC(K-1).AND.RLC(K-1).GT.RLC(K-2) ) THEN
           RO=RO-ROIA-RO*ROIM
           GO TO 53
       ELSE IF(RLC(K).LT.RLC(K-1)) THEN
           RO=RO+ROIA
       ELSE IF(RLC(K).GT.RLC(K-1)) THEN
           RO=RO-ROIA
       END IF
      END IF
53       IF(RO.GT.0.999D0) RO=0.999D0
     IF(RO.LT.ROMIN) RO=ROMIN

1    CONTINUE
CCCCC67-9-----9-----9-----9-----9-----9-----9---72
C    END OF THE ITERATION LOOP
CCCCC67-9-----9-----9-----9-----9-----9-----9---72

     WRITE(*,*) ' END OF THE ITERATION LOOP '
       WRITE(3,'(2X,A,9X,A,9X,A,9X,A,9X,A,9X,A)')
    &'K','BTF(1)','BTF(2)','BTF(3)','BTF(4)','BTF(5)'
     DO 60 K=1,NITR
     WRITE(3,'(I3,1P5E15.6)') K,(BTF(J,K),J=1,M1P)
60    CONTINUE
     WRITE(3,'(40A)') ('-',J=1,39)
     WRITE(3,'(T5,A,I3,T15,A,I3,T35,A,I3,T45,A,I3,T70,A,I3)')
    &'JK1=',JK1-10,'JK2=',JK2-10,'IK1=',IK1,'IK2=',IK2,' KF=',NITR
     WRITE(3,'(40A)') ('-',J=1,39)
     WRITE(*,*) ' THE PROGRAM DID NOT CONVERGED AFTER'
     WRITE(*,'(10X,I3,5X,A)') NITR, 'ITERATION'
     WRITE(1,*) ' THE PROGRAM DID NOT CONVERGED AFTER'
     WRITE(1,'(10X,I3,5X,A)') NITR, 'ITERATION'
     WRITE(2,*) ' THE PROGRAM DID NOT CONVERGED AFTER'
     WRITE(2,'(10X,I3,5X,A)') NITR, 'ITERATION'
     WRITE(3,*) ' THE PROGRAM DID NOT CONVERGED AFTER'
     WRITE(3,'(10X,I3,5X,A)') NITR, 'ITERATION'
     WRITE(4,*) ' THE PROGRAM DID NOT CONVERGED AFTER'
     WRITE(4,'(10X,I3,5X,A)') NITR, 'ITERATION'
     CLOSE(UNIT=1)
     CLOSE(UNIT=2)
     CLOSE(UNIT=3)
     CLOSE(UNIT=4)
     CLOSE(UNIT=9)
     CLOSE(UNIT=91)

990   FORMAT(/'ITR =',I5/15X,'STATES',24X,'COSTATES'//3X,'T',8X,'X1=Y'
    ,7X,'X2=Y'',7X,'X3=P1',7X,'X4=P2',8X,'U',10X,'EB'/)
991        FORMAT(F7.4,1P5E11.3,1P2E10.2)
9911        FORMAT(F7.4,1P13E11.3)
992   FORMAT(/1X,'KK',1X,'IS',2X,'JG',6X,'XS',8X,'YS1',8X,'YS2',8X
    &,'YS3',8X,'YS4')
993        FORMAT(I2,I4,I4,1P6E11.3)
994   FORMAT('#','KK',1X,'IS',2X,'JG',6X,'XS',8X,'YS1',8X,'YS2',8X
    &,'YS3',8X,'YS4')
995        FORMAT(A,I2,I4,I4,1P6E11.3)
998     FORMAT(I10,1P2E20.10)
1000       FORMAT(12X,'BF(1)',10X,'BF(2)',10X,'BF(3)',10X,'BF(4)',13X,'TF')
1001       FORMAT(I3,1P5E15.6)

CCCCC67-9-----9-----9-----9-----9-----9-----9---72
9900   END
CCCCC67-9-----9-----9-----9-----9-----9-----9---72


CCCCC67-9-----9-----9-----9-----9-----9-----9---72
C    USER SUPLLIED SUBROUTINE FOR DERIVATIVES OF STATES & COSTATES
CCCCC67-9-----9-----9-----9-----9-----9-----9---72
      SUBROUTINE DERIVS(N,X,Y,DYDX)
      DOUBLE PRECISION UCS(2),C1,C2,C3,C4,C5,GR
      COMMON /PATH/UCS,C1,C2,C3,C4,C5,GR
      DOUBLE PRECISION X,P,Y(N),DYDX(N)
      DOUBLE PRECISION X1,X2,X3,X4,P1,P2,P3,P4,T1,T2,G
```

```
      DOUBLE PRECISION SIN,COS
      INTRINSIC SIN,COS

      P=1.0D0
      G=GR
      X1 = Y(1)
      X2 = Y(2)
      X3 = Y(3)
      X4 = Y(4)
      P1 = Y(5)
      P2 = Y(6)
      P3 = Y(7)
      P4 = Y(8)
      T1=UCS(1)
      T2=UCS(2)
      DYDX(1)=X2
      DYDX(2)=
     +P/(C1"C2-C3""2"COS(X3)""2)"(-G"(C2"(C4"COS(X1)+C5"COS(X1+X3))-(C2+
     +C3"COS(X3))"C5"COS(X1+X3))+C2"T1-(C2+C3"COS(X3))"T2+C3"SIN(X3)"((C
     +2+C3"COS(X3))"X2""2+C2"(2"X2"X4+X4""2)))
      DYDX(3)=X4
      DYDX(4)=
     +P/(C1"C2-C3""2"COS(X3)""2)"(G"((C2+C3"COS(X3))"(C4"COS(X1)+C5"COS(
     +X1+X3))-(C1+C2+2"C3"COS(X3))"C5"COS(X1+X3))-(C2+C3"COS(X3))"T1+(C1
     ++C2+2"C3"COS(X3))"T2-C3"SIN(X3)"((C1+C2+2"C3"COS(X3))"X2""2+(C2+C3
     +"COS(X3))"(2"X2"X4+X4""2)))
      DYDX(5)=
     +-P"(-P2/(C1"C2-C3""2"COS(X3)""2)"G"(C2"(-C4"SIN(X1)-C5"SIN(X1+X3))
     ++(C2+C3"COS(X3))"C5"SIN(X1+X3))+P4/(C1"C2-C3""2"COS(X3)""2)"G"((C2
     ++C3"COS(X3))"(-C4"SIN(X1)-C5"SIN(X1+X3))+(C1+C2+2"C3"COS(X3))"C5"S
     +IN(X1+X3)))
      DYDX(6)=
     +-P"(P1+P2/(C1"C2-C3""2"COS(X3)""2)"C3"SIN(X3)"(2"(C2+C3"COS(X3))"X
     +2+2"C2"X4)-P4/(C1"C2-C3""2"COS(X3)""2)"C3"SIN(X3)"(2"(C1+C2+2"C3"C
     +OS(X3))"X2+2"(C2+C3"COS(X3))"X4))
      DYDX(7)=
     +-P"(-2"P2/(C1"C2-C3""2"COS(X3)""2)""2"(-G"(C2"(C4"COS(X1)+C5"COS(X
     +1+X3))-(C2+C3"COS(X3))"C5"COS(X1+X3))+C2"T1-(C2+C3"COS(X3))"T2+C3"
     +SIN(X3)"((C2+C3"COS(X3))"X2""2+C2"(2"X2"X4+X4""2)))"C3""2"COS(X3)"
     +SIN(X3)+P2/(C1"C2-C3""2"COS(X3)""2)"(-G"(-C2"C5"SIN(X1+X3)+C3"SIN(
     +X3)"C5"COS(X1+X3)+(C2+C3"COS(X3))"C5"SIN(X1+X3))+C3"SIN(X3)"T2+C3"
     +COS(X3)"((C2+C3"COS(X3))"X2""2+C2"(2"X2"X4+X4""2))-C3""2"SIN(X3)""
     +2"X2""2)-2"P4/(C1"C2-C3""2"COS(X3)""2)""2"(G"((C2+C3"COS(X3))"(C4"
     +COS(X1)+C5"COS(X1+X3))-(C1+C2+2"C3"COS(X3))"C5"COS(X1+X3))-(C2+C3"
     +COS(X3))"T1+(C1+C2+2"C3"COS(X3))"T2-C3"SIN(X3)"((C1+C2+2"C3"COS(X3
     +))"X2""2+(C2+C3"COS(X3))"(2"X2"X4+X4""2)))"C3""2"COS(X3)"SIN(X3)+P
     +4/(C1"C2-C3""2"COS(X3)""2)"(G"(-C3"SIN(X3)"(C4"COS(X1)+C5"COS(X1+X
     +3))-(C2+C3"COS(X3))"C5"SIN(X1+X3)+2"C3"SIN(X3)"C5"COS(X1+X3)+(C1+C
     +2+2"C3"COS(X3))"C5"SIN(X1+X3))+C3"SIN(X3)"T1-2"C3"SIN(X3)"T2-C3"CO
     +S(X3)"((C1+C2+2"C3"COS(X3))"X2""2+(C2+C3"COS(X3))"(2"X2"X4+X4""2))
     +-C3"SIN(X3)"(-2"C3"SIN(X3)"X2""2-C3"SIN(X3)"(2"X2"X4+X4""2))))
      DYDX(8)=
     +-P"(P3+P2/(C1"C2-C3""2"COS(X3)""2)"C3"SIN(X3)"C2"(2"X2+2"X4)-P4/(C
     +1"C2-C3""2"COS(X3)""2)"C3"SIN(X3)"(C2+C3"COS(X3))"(2"X2+2"X4))
      Y(1) = X1
      Y(2) = X2
      Y(3) = X3
      Y(4) = X4
      Y(5) = P1
      Y(6) = P2
      Y(7) = P3
      Y(8) = P4
       RETURN
       END

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C     USER SUPLLIED SUBROUTINE FOR SECOND DERIVATIVES (JACOBIAN) OF S+C
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
      SUBROUTINE DJACOB(N,X,Y,DFDY)
DOUBLE PRECISION UCS(2),C1,C2,C3,C4,C5,GR
COMMON /PATH/UCS,C1,C2,C3,C4,C5,GR
DOUBLE PRECISION X,P,Y(N),DFDY(N,N)
DOUBLE PRECISION X1,X2,X3,X4,P1,P2,P3,P4,T1,T2,G
DOUBLE PRECISION SIN,COS,S3,O3,A11,A12,A13,A14,A24,DE
     &,A241,A12X2,DL4,DL5,DL6,DLTA1
INTRINSIC SIN,COS

P=1.0D0
G=GR
X1 = Y(1)
X2 = Y(2)
X3 = Y(3)
X4 = Y(4)
P1 = Y(5)
P2 = Y(6)
P3 = Y(7)
P4 = Y(8)
```

```
T1=UCS(1)
T2=UCS(2)

DO 12 I=1,N
       DO 11 J=1,N
11      DFDY(J,I)=0.0D0
12    CONTINUE

DFDY(1,2)=1.0D0
DFDY(2,1)=
   +-P/(C1**C2-C3**2*COS(X3)**2)**G*(C2*(-C4*SIN(X1)-C5*SIN(X1+X3))+(C2+
   +C3*COS(X3))*C5*SIN(X1+X3))
DFDY(2,2)=
   +P/(C1**C2-C3**2*COS(X3)**2)**C3*SIN(X3)*(2*(C2+C3*COS(X3))*X2+2*C2**X
   +4)
DFDY(2,3)=
   +-2*P/(C1**C2-C3**2*COS(X3)**2)**2*(-G*(C2*(C4*COS(X1)+C5*COS(X1+X3)
   +)-(C2+C3*COS(X3))*C5*COS(X1+X3))+C2*T1-(C2+C3*COS(X3))*T2+C3*SIN(X
   +3)*((C2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2)))*C3**2*COS(X3)*SIN(X
   +3)+P/(C1**C2-C3**2*COS(X3)**2)*(-G*(-C2*C5*SIN(X1+X3)+C3*SIN(X3)*C5
   +*COS(X1+X3)+(C2+C3*COS(X3))*C5*SIN(X1+X3))+C3*SIN(X3)*T2+C3*COS(X3
   +)*((C2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2))-C3**2*SIN(X3)**2*X2**
   +2)
DFDY(2,4)=
   +P/(C1**C2-C3**2*COS(X3)**2)**C3*SIN(X3)*C2*(2*X2+2*X4)
DFDY(3,4)=1.0D0
DFDY(4,1)=
   +P/(C1**C2-C3**2*COS(X3)**2)**G*((C2+C3*COS(X3))*(-C4*SIN(X1)-C5*SIN(
   +X1+X3))+(C1+C2+2*C3*COS(X3))*C5*SIN(X1+X3))
DFDY(4,2)=
   +-P/(C1**C2-C3**2*COS(X3)**2)**C3*SIN(X3)*(2*(C1+C2+2*C3*COS(X3))*X2+
   +2*(C2+C3*COS(X3))*X4)
DFDY(4,3)=
   +-2*P/(C1**C2-C3**2*COS(X3)**2)**2*(G*((C2+C3*COS(X3))*(C4*COS(X1)+C
   +5*COS(X1+X3))-(C1+C2+2*C3*COS(X3))*C5*COS(X1+X3))-(C2+C3*COS(X3))*
   +T1+(C1+C2+2*C3*COS(X3))*T2-C3*SIN(X3)*((C1+C2+2*C3*COS(X3))*X2**2+
   +(C2+C3*COS(X3))*(2*X2*X4+X4**2)))*C3**2*COS(X3)*SIN(X3)+P/(C1**C2-C
   +3**2*COS(X3)**2)*(G*(-C3*SIN(X3)*(C4*COS(X1)+C5*COS(X1+X3))-(C2+C3
   +*COS(X3))*C5*SIN(X1+X3)+2*C3*SIN(X3)*C5*COS(X1+X3)+(C1+C2+2*C3*COS
   +(X3))*C5*SIN(X1+X3))+C3*SIN(X3)*T1-2*C3*SIN(X3)*T2-C3*COS(X3)*((C1
   ++C2+2*C3*COS(X3))*X2**2+(C2+C3*COS(X3))*(2*X2*X4+X4**2))-C3*SIN(X3
   +)*(-2*C3*SIN(X3)*X2**2-C3*SIN(X3)*(2*X2*X4+X4**2)))
DFDY(4,4)=
   +-P/(C1**C2-C3**2*COS(X3)**2)**C3*SIN(X3)*(C2+C3*COS(X3))*(2*X2+2*X4)
DFDY(5,1)=
   +-P*(-P2/(C1**C2-C3**2*COS(X3)**2)*G*(C2*(-C4*COS(X1)-C5*COS(X1+X3))
   ++(C2+C3*COS(X3))*C5*COS(X1+X3))+P4/(C1**C2-C3**2*COS(X3)**2)*G*((C2
   ++C3*COS(X3))*(-C4*COS(X1)-C5*COS(X1+X3))+(C1+C2+2*C3*COS(X3))*C5*C
   +OS(X1+X3)))
DFDY(5,2)=0.0D0
DFDY(5,3)=
   +-P*(2*P2/(C1**C2-C3**2*COS(X3)**2)**2*G*(C2*(-C4*SIN(X1)-C5*SIN(X1+
   +X3))+(C2+C3*COS(X3))*C5*SIN(X1+X3))*C3**2*COS(X3)*SIN(X3)-P2/(C1**C
   +2-C3**2*COS(X3)**2)*G*(-C2*C5*COS(X1+X3)-C3*SIN(X3)*C5*SIN(X1+X3)+
   +(C2+C3*COS(X3))*C5*COS(X1+X3))-2*P4/(C1**C2-C3**2*COS(X3)**2)**2*G*
   +((C2+C3*COS(X3))*(-C4*SIN(X1)-C5*SIN(X1+X3))+(C1+C2+2*C3*COS(X3))*
   +C5*SIN(X1+X3))*C3**2*COS(X3)*SIN(X3)+P4/(C1**C2-C3**2*COS(X3)**2)*G
   +*(-C3*SIN(X3)*(-C4*SIN(X1)-C5*SIN(X1+X3))-(C2+C3*COS(X3))*C5*COS(X
   +1+X3)-2*C3*SIN(X3)*C5*SIN(X1+X3)+(C1+C2+2*C3*COS(X3))*C5*COS(X1+X3
   +)))
DFDY(5,4)=0.0D0
DFDY(5,5)=0.0D0
DFDY(5,6)=
   +P/(C1**C2-C3**2*COS(X3)**2)*G*(C2*(-C4*SIN(X1)-C5*SIN(X1+X3))+(C2+C
   +3*COS(X3))*C5*SIN(X1+X3))
DFDY(5,7)=0.0D0
DFDY(5,8)=
   +-P/(C1**C2-C3**2*COS(X3)**2)*G*((C2+C3*COS(X3))*(-C4*SIN(X1)-C5*SIN
   +(X1+X3))+(C1+C2+2*C3*COS(X3))*C5*SIN(X1+X3))
DFDY(6,1)=0.0D0
DFDY(6,2)=
   +-P*(P2/(C1**C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*C2+2*C3*COS(X3))-P4/
   +(C1**C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*C1+2*C2+4*C3*COS(X3)))
DFDY(6,3)=
   +-P*(-2*P2/(C1**C2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*(2*(C2+C3*C
   +OS(X3))*X2+2*C2*X4)*COS(X3)+P2/(C1**C2-C3**2*COS(X3)**2)*C3*COS(X3)
   +*(2*(C2+C3*COS(X3))*X2+2*C2*X4)-2*P2/(C1**C2-C3**2*COS(X3)**2)*C3**
   +2*SIN(X3)**2*X2+2*P4/(C1**C2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*
   +(2*(C1+C2+2*C3*COS(X3))*X2+2*(C2+C3*COS(X3))*X4)*COS(X3)-P4/(C1**C2
   +-C3**2*COS(X3)**2)*C3*COS(X3)*(2*(C1+C2+2*C3*COS(X3))*X2+2*(C2+C3*
   +COS(X3))*X4)-P4/(C1**C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(-4*C3*SIN(X3)
   +*X2-2*C3*SIN(X3)*X4))
DFDY(6,4)=
   +-P*(2*P2/(C1**C2-C3**2*COS(X3)**2)*C3*SIN(X3)*C2-P4/(C1**C2-C3**2*CO
   +S(X3)**2)*C3*SIN(X3)*(2*C2+2*C3*COS(X3)))
DFDY(6,5)=-1.0D0
DFDY(6,6)=
```

```
     +-P/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*(C2+C3*COS(X3))*X2+2*C2*
     +X4)
 DFDY(6,7)=0.0D0
 DFDY(6,8)=
     +P/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(2*(C1+C2+2*C3*COS(X3))*X2+2
     +*(C2+C3*COS(X3))*X4)
 DFDY(7,1)=
     +-P*(2*P2/(C1*C2-C3**2*COS(X3)**2)**2*G*(C2*(-C4*SIN(X1)-C5*SIN(X1+
     +X3))+(C2+C3*COS(X3))*C5*SIN(X1+X3))*C3**2*COS(X3)*SIN(X3)-P2/(C1*C
     +2-C3**2*COS(X3)**2)*G*(-C2*C5*COS(X1+X3)-C3*SIN(X3)*C5*SIN(X1+X3)+
     +(C2+C3*COS(X3))*C5*COS(X1+X3))-2*P4/(C1*C2-C3**2*COS(X3)**2)**2*G*
     +((C2+C3*COS(X3))*(-C4*SIN(X1)-C5*SIN(X1+X3))+(C1+C2+2*C3*COS(X3))*
     +C5*SIN(X1+X3))*C3**2*COS(X3)*SIN(X3)+P4/(C1*C2-C3**2*COS(X3)**2)*G
     +*(-C3*SIN(X3)*(-C4*SIN(X1)-C5*SIN(X1+X3))-(C2+C3*COS(X3))*C5*COS(X
     +1+X3)-2*C3*SIN(X3)*C5*SIN(X1+X3)+(C1+C2+2*C3*COS(X3))*C5*COS(X1+X3
     +)))
 DFDY(7,2)=
     +-P*(-2*P2/(C1*C2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*(2*(C2+C3*C
     +OS(X3))*X2+2*C2*X4)*COS(X3)+P2/(C1*C2-C3**2*COS(X3)**2)*(C3*COS(X3
     +)*(2*(C2+C3*COS(X3))*X2+2*C2*X4)-2*C3**2*SIN(X3)**2*X2)+2*P4/(C1*C
     +2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*(2*(C1+C2+2*C3*COS(X3))*X2
     ++2*(C2+C3*COS(X3))*X4)*COS(X3)+P4/(C1*C2-C3**2*COS(X3)**2)*(-C3*CO
     +S(X3)*(2*(C1+C2+2*C3*COS(X3))*X2+2*(C2+C3*COS(X3))*X4)-C3*SIN(X3)*
     +(-4*C3*SIN(X3)*X2-2*C3*SIN(X3)*X4)))
 S3=SIN(X3)
 O3=COS(X3)
 A11=(C1+C2+2*C3*COS(X3))
 A12=(C2+C3*COS(X3))
 A13=C3*SIN(X3)
 A14=(C4*COS(X1)+C5*COS(X1+X3))
 A24=C5*COS(X1+X3)
 DE=(C1*C2-C3**2*COS(X3)**2)
 A241=C5*SIN(X1+X3)
 A12X2=(A12*X2**2+C2*(2*X2*X4+X4**2))
 DL4=(A11*X2**2+A12*(2*X2*X4+X4**2))
 DL5=(-G*(C2*A14-A12*A24)+C2*T1-A12*T2+A13*A12X2)
 DL6=(-2*A13*X2**2-A13*(2*X2*X4+X4**2))
 DLTA1=(G*(A12*A14-A11*A24)-A12*T1+A11*T2-A13*DL4)
 DFDY(7,3)=-P*(8*P2/DE**3*DL5*C3**4*O3**2*S3**2
     +-4*P2/DE**2*(-G*(-C2*A241+A13*A24+A12*A241)+A13*T2+C3*O3*A12X2
     +-C3**2*S3**2*X2**2)*C3**2*O3*S3
     ++2*P2/DE**2*DL5*C3**2*S3**2-2*P2/DE**2*DL5*C3**2*O3**2
     ++P2/DE*(-G*(-C2*A24+C3*O3*A24-2*A13*A241+A12*A24)+C3*O3*T2
     +-A13*A12X2-3*C3**2*O3*S3*X2**2)
     ++8*P4/DE**3*DLTA1*C3**4*O3**2*S3**2
     +-4*P4/DE**2*(G*(-A13*A14-A12*A241+2*A13*A24+A11*A241)+A13*T1
     +-2*A13*T2-C3*O3*DL4-A13*DL6)*C3**2*O3*S3
     ++2*P4/DE**2*DLTA1*C3**2*S3**2
     +-2*P4/DE**2*DLTA1*C3**2*O3**2
     ++P4/DE*(G*(-C3*O3*A14-2*A13*A241-A12*A24+2*C3*O3*A24+A11*A24)
     ++C3*O3*T1-2*C3*O3*T2+A13*DL4-2*C3*O3*DL6
     +-A13*(-2*C3*O3*X2**2-C3*O3*(2*X2*X4+X4**2))))
 DFDY(7,4)=
     +-P*(-2*P2/(C1*C2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*C2*(2*X2+2*
     +X4)*COS(X3)+P2/(C1*C2-C3**2*COS(X3)**2)*C3*COS(X3)*C2*(2*X2+2*X4)+
     +2*P4/(C1*C2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*(C2+C3*COS(X3))*
     +(2*X2+2*X4)*COS(X3)+P4/(C1*C2-C3**2*COS(X3)**2)*(-C3*COS(X3)*(C2+C
     +3*COS(X3))*(2*X2+2*X4)+C3**2*SIN(X3)**2*(2*X2+2*X4)))
 DFDY(7,5)=0.0D0
 DFDY(7,6)=
     +-P*(-2/(C1*C2-C3**2*COS(X3)**2)**2*(-G*(C2*(C4*COS(X1)+C5*COS(X1+X
     +3))-(C2+C3*COS(X3))*C5*COS(X1+X3))+C2*T1-(C2+C3*COS(X3))*T2+C3*SIN
     +(X3)*((C2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2)))*C3**2*COS(X3)*SIN
     +(X3)+1/(C1*C2-C3**2*COS(X3)**2)*(-G*(-C2*C5*SIN(X1+X3)+C3*SIN(X3)*
     +C5*COS(X1+X3)+(C2+C3*COS(X3))*C5*SIN(X1+X3))+C3*SIN(X3)*T2+C3*COS(
     +X3)*((C2+C3*COS(X3))*X2**2+C2*(2*X2*X4+X4**2))-C3**2*SIN(X3)**2*X2
     +**2))
 DFDY(7,7)=0.0D0
 DFDY(7,8)=
     +-P*(-2/(C1*C2-C3**2*COS(X3)**2)**2*(G*((C2+C3*COS(X3))*(C4*COS(X1)
     ++C5*COS(X1+X3))-(C1+C2+2*C3*COS(X3))*C5*COS(X1+X3))-(C2+C3*COS(X3)
     +)*T1+(C1+C2+2*C3*COS(X3))*T2-C3*SIN(X3)*((C1+C2+2*C3*COS(X3))*X2**
     +2+(C2+C3*COS(X3))*(2*X2*X4+X4**2)))*C3**2*COS(X3)*SIN(X3)+1/(C1*C2
     +-C3**2*COS(X3)**2)*(G*(-C3*SIN(X3)*(C4*COS(X1)+C5*COS(X1+X3))-(C2+
     +C3*COS(X3))*C5*SIN(X1+X3)+2*C3*SIN(X3)*C5*COS(X1+X3)+(C1+C2+2*C3*C
     +OS(X3))*C5*SIN(X1+X3))+C3*SIN(X3)*T1-2*C3*SIN(X3)*T2-C3*COS(X3)*((
     +C1+C2+2*C3*COS(X3))*X2**2+(C2+C3*COS(X3))*(2*X2*X4+X4**2))-C3*SIN(
     +X3)*(-2*C3*SIN(X3)*X2**2-C3*SIN(X3)*(2*X2*X4+X4**2))))
 DFDY(8,1)=0.0D0
 DFDY(8,2)=
     +-P*(2*P2/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*C2-2*P4/(C1*C2-C3**2*
     +COS(X3)**2)*C3*SIN(X3)*(C2+C3*COS(X3)))
 DFDY(8,3)=
     +-P*(-2*P2/(C1*C2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*C2*(2*X2+2*
     +X4)*COS(X3)+P2/(C1*C2-C3**2*COS(X3)**2)*C3*COS(X3)*C2*(2*X2+2*X4)+
     +2*P4/(C1*C2-C3**2*COS(X3)**2)**2*C3**3*SIN(X3)**2*(C2+C3*COS(X3))*
     +(2*X2+2*X4)*COS(X3)-P4/(C1*C2-C3**2*COS(X3)**2)*C3*COS(X3)*(C2+C3*
```

```
     +COS(X3))*(2*X2+2*X4)+P4/(C1*C2-C3**2*COS(X3)**2)*C3**2*SIN(X3)**2*
     +(2*X2+2*X4))
 DFDY(8,4)=
     +-P*(2*P2/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*C2-2*P4/(C1*C2-C3**2*
     +COS(X3)**2)*C3*SIN(X3)*(C2+C3*COS(X3)))
 DFDY(8,5)=0.0D0
 DFDY(8,6)=
     +-P/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*C2*(2*X2+2*X4)
 DFDY(8,7)=-1.0D0
 DFDY(8,8)=
     +P/(C1*C2-C3**2*COS(X3)**2)*C3*SIN(X3)*(C2+C3*COS(X3))*(2*X2+2*X4)
      Y(1) = X1
      Y(2) = X2
      Y(3) = X3
      Y(4) = X4
      Y(5) = P1
      Y(6) = P2
      Y(7) = P3
      Y(8) = P4
 RETURN
 END

 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
 C    SUBROUTINE FOR CALCULATING DERIVATIVES OF DX/DB
 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
       SUBROUTINE DERIVSH(N,NVAR,X,YS,DFDY,Y,DYDX,DJACOB)
       PARAMETER(NT=16,M1T=NT/2)
 DOUBLE PRECISION UCS(2),C1,C2,C3,C4,C5,GR
 COMMON /PATH/UCS,C1,C2,C3,C4,C5,GR
      DOUBLE PRECISION Y(NVAR),DYDX(NVAR),DFDY(N,N),DYDB(NT,M1T),YS(N)
     &,DYHDT(NT,M1T),SUM
        DOUBLE PRECISION X
        INTEGER N,M1,NVAR
       EXTERNAL DJACOB
        M1=N/2
        CALL DJACOB(N,X,YS,DFDY)
        DO 14 I=1,M1
          DO 13 J=1,N
            IJ=(I-1)*N+J
 13        DYDB(J,I)=Y(IJ)
 14      CONTINUE
        DO 17 L=1,M1
           DO 16 J=1,N
           LJ=(L-1)*N+J
           SUM=0.0D0
              DO 15 I=1,N
 15            SUM = DFDY(J,I)*DYDB(I,L)+SUM
           DYHDT(J,L)=SUM
           DYDX(LJ)=DYHDT(J,L)
 16      CONTINUE
 17      CONTINUE
        RETURN
        END

 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
 C    USER SUPLIED SUBROUTINE FOR CONTROL FORCES/TORQUES
 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
      SUBROUTINE UCONT(Y,CS,UC)
 DOUBLE PRECISION UCS(2),C1,C2,C3,C4,C5,GR
 COMMON /PATH/UCS,C1,C2,C3,C4,C5,GR
      DOUBLE PRECISION Y(*),CS(*),UC(*),G(2)
      DOUBLE PRECISION A11,A12,A13,A22,DE,DSIN,DCOS
      INTRINSIC DSIN,DCOS
 A11=C1+C2+2*C3*DCOS(Y(3))
 A12=C2+C3*DCOS(Y(3))
 A13=C3*DSIN(Y(3))
 A22=C2
 DE=A11*A22-A12*A12
 G(1)=(+Y(6)*A22-Y(8)*A12)/(A11*A22-A12*A12)
 G(2)=(-Y(6)*A12+Y(8)*A11)/(A11*A22-A12*A12)
      IF(G(1).GT.0.0D0) THEN
         UC(1)=CS(1)
      ELSE
         UC(1)=CS(2)
      END IF
      IF(G(2).GT.0.0D0) THEN
  UC(2)=CS(3)
      ELSE
  UC(2)=CS(4)
      END IF
      RETURN
      END

 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
 C    USER SUPPLIED SUBROUTINE FOR CALCULATING SWITCH FUNCTIONS AND
 C      ITS DERIVATIVES (G,DG/DT,DG/DX)
 CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
```

```
      SUBROUTINE GDGTX(N,M2,X,Y,G,DGDT,DGX,DERIVS)
      PARAMETER(NT=16)
 DOUBLE PRECISION UCS(2),C1,C2,C3,C4,C5,GR
 COMMON /PATH/UCS,C1,C2,C3,C4,C5,GR
 DOUBLE PRECISION X,Y(N),G(M2),DGDT(M2),DGX(M2,N),DYDX(NT)
     DOUBLE PRECISION A11,A12,A13,A22,DE,DSIN,DCOS
     INTRINSIC DSIN,DCOS
 DO 11 I=1,M2
        DO 12 J=1,N
 12      DGX(I,J)=0.0D0
 11     CONTINUE
 A11=C1+C2+2*C3*DCOS(Y(3))
 A12=C2+C3*DCOS(Y(3))
 A13=C3*DSIN(Y(3))
 A22=C2
 DE=A11*A22-A12*A12
 G(1)=(+Y(6)*A22-Y(8)*A12)/(A11*A22-A12*A12)
 G(2)=(-Y(6)*A12+Y(8)*A11)/(A11*A22-A12*A12)
 DGX(1,3)=A13*(Y(8)-2.0D0*(A12-A22)*G(1))/(A11*A22-A12*A12)
 DGX(1,6)=+A22/(A11*A22-A12*A12)
 DGX(1,8)=-A12/(A11*A22-A12*A12)
     DGX(2,3)=A13*(Y(6)-2.0D0*(Y(8)+(A12-A22)*G(2)))/(A11*A22-A12*A12)
 DGX(2,6)=-A12/(A11*A22-A12*A12)
 DGX(2,8)=+A11/(A11*A22-A12*A12)
 CALL DERIVS(N,X,Y,DYDX)
       DO 14 I=1,M2
        DGDT(I)=0.0D0
        DO 13 J=1,N
 13      DGDT(I)=DGX(I,J)*DYDX(J)+DGDT(I)
 14     CONTINUE
       RETURN
       END

CCCCC67-9-----9-----9-----9-----9-----9-----9----72
C     USER SUPPLIED SUBROUTINE FOR CALCULATING VECTOR L(X,B,TF) AND ITS
C     DERIVATIVES AT THE FINAL TIME (L,DL/DX,DL/DB,DL/DTF)
CCCCC67-9-----9-----9-----9-----9-----9-----9----72
      SUBROUTINE LDXBT(N,M1,X,Y,VF,DFDY,LC,DLDX,DLDB,DLDTF,WF,DERIVS
    &,DJACOB)
      PARAMETER(NT=16)
 DOUBLE PRECISION UCS(2),C1,C2,C3,C4,C5,GR
 COMMON /PATH/UCS,C1,C2,C3,C4,C5,GR
      DOUBLE PRECISION Y(N),LC(M1+1),DLDX(M1+1,N),DLDB(M1+1,M1)
    & ,DLDTF(M1+1),DYDX(NT),DFDY(N,N),VF(N),SUM,WF(M1+1)
      DOUBLE PRECISION X
      EXTERNAL DERIVS,DJACOB
      CALL DERIVS(N,X,Y,DYDX)
      CALL DJACOB(N,X,Y,DFDY)
      SUM=0.0D0
      DO 15 I=1,M1
       LC(I)=Y(I)-VF(I)
       SUM=Y(M1+I)*DYDX(I)+SUM
 15    CONTINUE
      LC(M1+1)=1.0D0+SUM
      DO 13 I=1,M1+1
       DO 11 J=1,N
 11     DLDX(I,J)=0.0D0
       DO 12 J=1,M1
 12     DLDB(I,J)=0.0D0
 13    CONTINUE
      DO 16 I=1,M1
       DLDX(I,I)=1.0D0
       SUM=0.0D0
       DO 17 J=1,M1
 17      SUM=Y(M1+J)*DFDY(J,I)+SUM
       DLDX(M1+1,I)=SUM
       DLDX(M1+1,M1+I)=DYDX(I)
 16    CONTINUE
      DO 14 I=1,M1+1
 14    DLDTF(I)=0.0D0

      DO 19 I=1,M1+1
  LC(I)=LC(I)/WF(I)
CWS   LC(I)=LC(I)*WF(I)
       DO 18 J=1,N
        DLDX(I,J)=DLDX(I,J)/WF(I)
CWS       DLDX(I,J)=DLDX(I,J)*WF(I)
 18     CONTINUE
 19    CONTINUE
      RETURN
      END

CCCCC67-9-----9-----9-----9-----9-----9-----9----72
C     USER SUPLLIED SUBROUTINE FOR INITIAL CONDITIONS OF DX/DB
CCCCC67-9-----9-----9-----9-----9-----9-----9----72
      SUBROUTINE DYDBT0(N,M1,DKDB)
      PARAMETER(NT=16,M1T=NT/2)
```

```
      DOUBLE PRECISION DYDB(NT,M1T),DKDB(N*M1)
      DO 12 I=1,M1
       DO 11 J=1,N
11      DYDB(J,I)=0.0D0
12     CONTINUE
      DO 15 I=1,M1
       DYDB(M1+I,I)=1.0D0
15     CONTINUE
      DO 14 I=1,M1
       DO 13 J=1,N
        IJ=(I-1)*N+J
13      DKDB(IJ)=DYDB(J,I)
14     CONTINUE
      RETURN
      END

CCCCC67-9-----9-----9-----9-----9-----9-----9-----9----72
C     SUBROUTINE FOR CALCULATING JUMP OF (DY/DB) AT SWITCHING POINTS
CCCCC67-9-----9-----9-----9-----9-----9-----9-----9----72
SUBROUTINE DTDBS(N,M1,M2,NM,ISF,YHSL,DGDT,DGX,F,YHSR)
      PARAMETER(NT=16,M1T=NT/2,M2T=M1T/2)
DOUBLE PRECISION UCS(2),C1,C2,C3,C4,C5,GR
COMMON /PATH/UCS,C1,C2,C3,C4,C5,GR
      DOUBLE PRECISION YHSL(NM),YHSR(NM),YHT(NT,M1T),GYH(M2T,M1T)
   &,DGDT(M2),DTDB(M1T),FTB(NT,M1T),F(N),DGX(M2,N),SUM
      DO 15 I=1,M1
       DO 14 J=1,N
        IJ=(I-1)*N+J
14      YHT(J,I)=YHSL(IJ)
15     CONTINUE
DO 12 J=1,M1
      SUM=0.0D0
      DO 11 I=1,N
11  SUM = DGX(ISF,I)*YHT(I,J)+SUM
  GYH(ISF,J)=SUM
12     CONTINUE
DO 16 J=1,M1
16  DTDB(J)=-GYH(ISF,J)/DGDT(ISF)
DO 19 J=1,M1
      DO 18 I=1,N
18      FTB(I,J)= F(I)*DTDB(J)
19     CONTINUE
      DO 21 I=1,M1
       DO 20 J=1,N
       IJ=(I-1)*N+J
       YHT(J,I)=YHT(J,I)+FTB(J,I)
20      YHSR(IJ)=YHT(J,I)
21     CONTINUE
       RETURN
      END

CCCCC67-9-----9-----9-----9-----9-----9-----9-----9----72
C     SUBROUTINE FOR CALCULATING (D-MATRIX)DEVIATION OF Y'S AT THE END
CCCCC67-9-----9-----9-----9-----9-----9-----9-----9----72
      SUBROUTINE DLSBT(N,M1,YHF,DLDX,DLDB,DLDTF,DVF,D)
      PARAMETER(NT=16,M1T=NT/2)
DOUBLE PRECISION UCS(2),C1,C2,C3,C4,C5,GR
COMMON /PATH/UCS,C1,C2,C3,C4,C5,GR
    DOUBLE PRECISION YHF(N*M1),DLDX(M1+1,N),DLDB(M1+1,M1),DLDTF(M1+1)
   &,DVF(N),DLSDB(M1T+1,M1T),DLSDT(M1T+1),D(M1+1,M1+1),YHT(NT,M1T),SUM
      DO 12 I=1,M1
       DO 11 J=1,N
        IJ=(I-1)*N+J
11      YHT(J,I)=YHF(IJ)
12     CONTINUE
      DO 15 L=1,M1+1
         DO 14 J=1,M1
         SUM=0.0D0
            DO 13 I=1,N
13           SUM = DLDX(L,I)*YHT(I,J)+SUM
         DLSDB(L,J)=SUM+DLDB(L,J)
14        D(L,J)=DLSDB(L,J)
15     CONTINUE
      DO 17 L=1,M1+1
         SUM=0.0D0
         DO 16 I=1,N
16        SUM = DLDX(L,I)*DVF(I)+SUM
         DLSDT(L)=SUM+DLDTF(L)
17        D(L,M1+1)=DLSDT(L)
       RETURN
      END

CCCCC67-9-----9-----9-----9-----9-----9-----9-----9----72
C     FUNCTION FOR CALCULATING HAMILTONIANS
CCCCC67-9-----9-----9-----9-----9-----9-----9-----9----72
    FUNCTION HAM(M1,X,Y,DYDX)
    DOUBLE PRECISION X,Y(*),DYDX(*),HAM
```

```
      HAM=0.0D0
      DO 11 I=1,M1
       HAM=Y(M1+I)*DYDX(I)+HAM
11    CONTINUE
      HAM=1.0D0+HAM
     RETURN
     END

CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
C    SUBROUTINE FOR CALCULATING SPACIAL TRAJECTORY PATH
CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
     SUBROUTINE STPA(L1,L2,LC1,LC2,Y,XP,YP)
     DOUBLE PRECISION Y(*),XP(*),YP(*),L1,L2,LC1,LC2,DSIN,DCOS
     INTRINSIC DSIN,DCOS
XP(1)=L1*DCOS(Y(1))
YP(1)=L1*DSIN(Y(1))
XP(2)=L1*DCOS(Y(1))+LC2*DCOS(Y(1)+Y(3))
YP(2)=L1*DSIN(Y(1))+LC2*DSIN(Y(1)+Y(3))
XP(3)=L1*DCOS(Y(1))+L2*DCOS(Y(1)+Y(3))
YP(3)=L1*DSIN(Y(1))+L2*DSIN(Y(1)+Y(3))
     RETURN
     END

CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
C    SUBROUTINE ONE STEP INTEGRATION USING 4TH ORDER RUNGE-KUTTA METHOD
CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
     SUBROUTINE RK4D(Y,DYDX,N,X,H,YOUT,DERIVS)
     PARAMETER (NMAX=80)
     DOUBLE PRECISION Y(N),DYDX(N),YOUT(N),YT(NMAX),DYT(NMAX),DYM(NMAX)
     DOUBLE PRECISION H,HH,H6,XH,X
     EXTERNAL DERIVS
     HH=H*0.5D0
     H6=H/6.D0
     XH=X+HH
     DO 11 I=1,N
      YT(I)=Y(I)+HH*DYDX(I)
11   CONTINUE
     CALL DERIVS(N,XH,YT,DYT)
     DO 12 I=1,N
      YT(I)=Y(I)+HH*DYT(I)
12   CONTINUE
     CALL DERIVS(N,XH,YT,DYM)
     DO 13 I=1,N
      YT(I)=Y(I)+H*DYM(I)
      DYM(I)=DYT(I)+DYM(I)
13   CONTINUE
     CALL DERIVS(N,X+H,YT,DYT)
     DO 14 I=1,N
      YOUT(I)=Y(I)+H6*(DYDX(I)+DYT(I)+2.D0*DYM(I))
14   CONTINUE
     RETURN
     END
C (C) COPR. 1986-92 NUMERICAL RECIPES SOFTWARE (MODIFIED)

CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
C    SUBROUTINE ONE STEP INTEGRATION USING 4TH ORDER RUNGE-KUTTA METHOD
CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
     SUBROUTINE RK4H(Y,DYDX,N,NVAR,X,H,YS,DFDY,YOUT,DERIVS,DERIVSH)
     PARAMETER (NMAX=80)
     DOUBLE PRECISION Y(NVAR),DYDX(NVAR),YOUT(NVAR),YT(NMAX),DYT(NMAX)
    &,DYM(NMAX)
     DOUBLE PRECISION YS(N),DYS(NMAX),DFDY(N,N)
     DOUBLE PRECISION H,HH,H6,XH,X
     DOUBLE PRECISION UCS(2),C1,C2,C3,C4,C5,GR
     COMMON /PATH/UCS,C1,C2,C3,C4,C5,GR
     EXTERNAL DERIVSH,DJACOB,DERIVS
     HH=H*0.5D0
     H6=H/6.D0
     XH=X+HH
     DO 11 I=1,NVAR
      YT(I)=Y(I)+HH*DYDX(I)
11   CONTINUE
     CALL DERIVSH(N,NVAR,XH,YS,DFDY,YT,DYT,DJACOB)
     DO 12 I=1,NVAR
      YT(I)=Y(I)+HH*DYT(I)
12   CONTINUE
     CALL DERIVSH(N,NVAR,XH,YS,DFDY,YT,DYM,DJACOB)
     DO 13 I=1,NVAR
      YT(I)=Y(I)+H*DYM(I)
      DYM(I)=DYT(I)+DYM(I)
13   CONTINUE
     CALL DERIVSH(N,NVAR,X+H,YS,DFDY,YT,DYT,DJACOB)
     DO 14 I=1,NVAR
      YOUT(I)=Y(I)+H6*(DYDX(I)+DYT(I)+2.D0*DYM(I))
14   CONTINUE
     CALL DERIVS(N,X,YS,DYS)
     CALL RK4D(YS,DYS,N,X,H,YS,DERIVS)
```

```
      RETURN
      END
C  (C) COPR. 1986-92 NUMERICAL RECIPES SOFTWARE (MODIFIED)

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C   SUBROUTINE INTEGRATION OF ODE's USING CONSTANT STEP SIZE RUNGE-KUTTA
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
      SUBROUTINE RKDUMBH(N,NVAR,VS,DFDV,VSTART,X1,X2,NSTEP,XH,YH,DERIVS
     &,DERIVSH)
      PARAMETER (NMAX=80)
      DOUBLE PRECISION VSTART(NVAR),V(NMAX),DV(NMAX),VS(N),DFDV(N,N)
      DOUBLE PRECISION XH(NSTEP+1),YH(NVAR,NSTEP+1)
      DOUBLE PRECISION X1,X2,X,H
DOUBLE PRECISION UCS(2),C1,C2,C3,C4,C5,GR
COMMON /PATH/UCS,C1,C2,C3,C4,C5,GR
       EXTERNAL DERIVS,DERIVSH,DJACOB
      DO 11 I=1,NVAR
        V(I)=VSTART(I)
YH(I,1)=V(I)
11    CONTINUE
      XH(1)=X1
      X=X1
      H=(X2-X1)/NSTEP
      DO 13 K=1,NSTEP
        CALL DERIVSH(N,NVAR,X,VS,DFDV,V,DV,DJACOB)
        CALL RK4H(V,DV,N,NVAR,X,H,VS,DFDV,V,DERIVS,DERIVSH)
        IF(X+H.EQ.X)PAUSE 'Stepsize not significant in RKDUMB.'
        X=X+H
XH(K+1)=X
        DO 12 I=1,NVAR
  YH(I,K+1)=V(I)
12      CONTINUE
13    CONTINUE
      RETURN
      END
C  (C) COPR. 1986-92 NUMERICAL RECIPES SOFTWARE (MODIFIED)

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C     SUBROUTINE FOR SOLVING LINEAR ALGEBRIC EQUATION A*X=C
C     DRIVER FOR ROUTINE SVBKSB, WHICH CALLS ROUTINE SVDCMP
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
       SUBROUTINE DXSVB(N,SM,A,C,X)
      INTEGER MP,NP
      PARAMETER(MP=20,NP=20)
      INTEGER J,K,L,N
      DOUBLE PRECISION A(N,N),C(N),X(N)
      DOUBLE PRECISION U(NP,NP),V(NP,NP),W(NP),AT(MP,NP)
      DOUBLE PRECISION WMAX,WMIN,SM
C     COPY A INTO U
      DO 12 K=1,N
        DO 11 L=1,N
          U(K,L)=A(K,L)
11      CONTINUE
12    CONTINUE
C     DECOMPOSE MATRIX A
      CALL SVDCMP(U,N,N,NP,NP,W,V)
      WRITE(4,*) 'CHECK PRODUCT AGAINST ORIGINAL MATRIX:'
      WRITE(4,*) 'ORIGINAL MATRIX:'
        WRITE(4,'(10X,A,10X,A,10X,A,10X,A,10X,A)')
     &'A(1,J)','A(2,J)','A(3,J)','A(4,J)','A(5,J)'
        DO 45 L=1,N
45      WRITE(4,'(6E16.8)') (A(L,J),J=1,N)
      WRITE(4,*) 'PRODUCT U*W*(V-TRANSPOSE):'
      DO 18 K=1,N
        DO 19 L=1,N
          AT(K,L)=0.0D0
          DO 21 J=1,N
            AT(K,L)=AT(K,L)+U(K,J)*W(J)*V(L,J)
21        CONTINUE
19      CONTINUE
        WRITE(4,'(6E16.8)') (AT(K,L),L=1,N)
18    CONTINUE
      WRITE(4,*) '**********************************'
C     FIND MAXIMUM SINGULAR VALUE
      WRITE(4,*) 'FIND MAXIMUM SINGULAR VALUE'
      WRITE(4,'(11X,A,11X,A,11X,A,11X,A,11X,A)')
     &' W(1)',' W(2)',' W(3)',' W(4)',' W(5)'
      WRITE(4,'(6E16.8)') ( W(K),K=1,N)
      WMAX=0.0D0
      DO 13 K=1,N
        IF (W(K).GT.WMAX) WMAX=W(K)
13    CONTINUE
C     DEFINE "SMALL"
      WMIN=WMAX*SM
C     ZERO THE "SMALL" SINGULAR VALUES
      DO 14 K=1,N
        IF (W(K).LT.WMIN) W(K)=0.0D0
```

```
14    CONTINUE
      WRITE(4,*) 'ZERO THE "SMALL" SINGULAR VALUES'
      WRITE(4,'(6E16.8)') ( W(K),K=1,N)
C     BACKSUBSTITUTE FOR RIGHT-HAND SIDE VECTOR
       CALL SVBKSB(U,W,V,N,N,NP,NP,C,X)
      END
C  (C) COPR. 1986-92 NUMERICAL RECIPES SOFTWARE (MODIFIED)

CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
C     SUBROUTINE FOR SOLVING LINEAR ALGEBRIC EQUATION A*X=C
C     ROUTINE SVDCMP
CCCCC67-9-----9-----9-----9-----9-----9-----9-----.72
      SUBROUTINE svdcmp(a,m,n,mp,np,w,v)
      INTEGER m,mp,n,np,NMAX
      DOUBLE PRECISION a(mp,np),v(np,np),w(np)
      PARAMETER (NMAX=500)
CU    USES pythag
      INTEGER i,its,j,jj,k,l,nm
      DOUBLE PRECISION anorm,c,f,g,h,s,scale,x,y,z,rv1(NMAX),pythag
      g=0.0d0
      scale=0.0d0
      anorm=0.0d0
      do 25 i=1,n
        l=i+1
        rv1(i)=scale*g
        g=0.0d0
        s=0.0d0
        scale=0.0d0
        if(i.le.m)then
          do 11 k=i,m
            scale=scale+abs(a(k,i))
11        continue
          if(scale.ne.0.0d0)then
            do 12 k=i,m
              a(k,i)=a(k,i)/scale
              s=s+a(k,i)*a(k,i)
12          continue
            f=a(i,i)
            g=-sign(sqrt(s),f)
            h=f*g-s
            a(i,i)=f-g
            do 15 j=l,n
              s=0.0d0
              do 13 k=i,m
                s=s+a(k,i)*a(k,j)
13            continue
              f=s/h
              do 14 k=i,m
                a(k,j)=a(k,j)+f*a(k,i)
14            continue
15          continue
            do 16 k=i,m
              a(k,i)=scale*a(k,i)
16          continue
          endif
        endif
        w(i)=scale *g
        g=0.0d0
        s=0.0d0
        scale=0.0d0
        if((i.le.m).and.(i.ne.n))then
          do 17 k=l,n
            scale=scale+abs(a(i,k))
17        continue
          if(scale.ne.0.0d0)then
            do 18 k=l,n
              a(i,k)=a(i,k)/scale
              s=s+a(i,k)*a(i,k)
18          continue
            f=a(i,l)
            g=-sign(sqrt(s),f)
            h=f*g-s
            a(i,l)=f-g
            do 19 k=l,n
              rv1(k)=a(i,k)/h
19          continue
            do 23 j=l,m
              s=0.0d0
              do 21 k=l,n
                s=s+a(j,k)*a(i,k)
21            continue
              do 22 k=l,n
                a(j,k)=a(j,k)+s*rv1(k)
22            continue
23          continue
            do 24 k=l,n
              a(i,k)=scale*a(i,k)
```

```
24          continue
          endif
        endif
        anorm=max(anorm,(abs(w(i))+abs(rv1(i))))
25    continue
      do 32 i=n,1,-1
        if(i.lt.n)then
          if(g.ne.0.0d0)then
            do 26 j=l,n
              v(j,i)=(a(i,j)/a(i,l))/g
26          continue
            do 29 j=l,n
              s=0.0d0
              do 27 k=l,n
                s=s+a(i,k)*v(k,j)
27            continue
              do 28 k=l,n
                v(k,j)=v(k,j)+s*v(k,i)
28            continue
29          continue
          endif
          do 31 j=l,n
            v(i,j)=0.0d0
            v(j,i)=0.0d0
31        continue
        endif
        v(i,i)=1.0d0
        g=rv1(i)
        l=i
32    continue
      do 39 i=min(m,n),1,-1
        l=i+1
        g=w(i)
        do 33 j=l,n
          a(i,j)=0.0d0
33      continue
        if(g.ne.0.0d0)then
          g=1.0d0/g
          do 36 j=l,n
            s=0.0d0
            do 34 k=l,m
              s=s+a(k,i)*a(k,j)
34          continue
            f=(s/a(i,i))*g
            do 35 k=i,m
              a(k,j)=a(k,j)+f*a(k,i)
35          continue
36        continue
          do 37 j=i,m
            a(j,i)=a(j,i)*g
37        continue
        else
          do 38 j= i,m
            a(j,i)=0.0d0
38        continue
        endif
        a(i,i)=a(i,i)+1.0d0
39    continue
      do 49 k=n,1,-1
        do 48 its=1,30
          do 41 l=k,1,-1
            nm=l-1
            if((abs(rv1(l))+anorm).eq.anorm)  goto 2
            if((abs(w(nm))+anorm).eq.anorm)  goto 1
41        continue
1         c=0.0d0
          s=1.0d0
          do 43 i=l,k
            f=s*rv1(i)
            rv1(i)=c*rv1(i)
            if((abs(f)+anorm).eq.anorm) goto 2
            g=w(i)
            h=pythag(f,g)
            w(i)=h
            h=1.0d0/h
            c= (g*h)
            s=-(f*h)
            do 42 j=1,m
              y=a(j,nm)
              z=a(j,i)
              a(j,nm)=(y*c)+(z*s)
              a(j,i)=-(y*s)+(z*c)
42          continue
43        continue
2         z=w(k)
          if(l.eq.k)then
            if(z.lt.0.0d0)then
```

```
            w(k)=-z
            do 44 j=1,n
              v(j,k)=-v(j,k)
44          continue
          endif
          goto 3
        endif
        if(its.eq.30) pause 'no convergence in svdcmp'
        x=w(l)
        nm=k-1
        y=w(nm)
        g=rv1(nm)
        h=rv1(k)
        f=((y-z)*(y+z)+(g-h)*(g+h))/(2.0d0*h*y)
        g=pythag(f,1.0d0)
        f=((x-z)*(x+z)+h*((y/(f+sign(g,f)))-h))/x
        c=1.0d0
        s=1.0d0
        do 47 j=l,nm
          i=j+1
          g=rv1(i)
          y=w(i)
          h=s*g
          g=c*g
          z=pythag(f,h)
          rv1(j)=z
          c=f/z
          s=h/z
          f= (x*c)+(g*s)
          g=-(x*s)+(g*c)
          h=y*s
          y=y*c
          do 45 jj=1,n
            x=v(jj,j)
            z=v(jj,i)
            v(jj,j)= (x*c)+(z*s)
            v(jj,i)=-(x*s)+(z*c)
45        continue
          z=pythag(f,h)
          w(j)=z
          if(z.ne.0.0d0)then
            z=1.0d0/z
            c=f*z
            s=h*z
          endif
          f= (c*g)+(s*y)
          x=-(s*g)+(c*y)
          do 46 jj=1,m
            y=a(jj,j)
            z=a(jj,i)
            a(jj,j)= (y*c)+(z*s)
            a(jj,i)=-(y*s)+(z*c)
46        continue
47      continue
        rv1(l)=0.0d0
        rv1(k)=f
        w(k)=x
48    continue
3     continue
49  continue
    return
    END
C  (C) Copr. 1986-92 Numerical Recipes Software j:!&1415S.

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
C     SUBROUTINE FOR SOLVING LINEAR ALGEBRIC EQUATION A*X=C
C     FUNCTION pythag
CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
    FUNCTION pythag(a,b)
    DOUBLE PRECISION a,b,pythag
    DOUBLE PRECISION absa,absb
    absa=abs(a)
    absb=abs(b)
    if(absa.gt.absb)then
      pythag=absa*sqrt(1.d0+(absb/absa)**2)
    else
      if(absb.eq.0.d0)then
        pythag=0.d0
      else
        pythag=absb*sqrt(1.d0+(absa/absb)**2)
      endif
    endif
    return
    END
C  (C) Copr. 1986-92 Numerical Recipes Software j:!&1415S.

CCCCC67-9-----9-----9-----9-----9-----9-----9-----72
```

```
C     SUBROUTINE FOR SOLVING LINEAR ALGEBRIC EQUATION A*X=C
C     SUBROUTINE svbksb
CCCCC67-9-------9-------9-------9-------9-------9-------9-------72
      SUBROUTINE svbksb(u,w,v,m,n,mp,np,b,x)
      INTEGER m,mp,n,np,NMAX
      DOUBLE PRECISION b(mp),u(mp,np),v(np,np),w(np),x(np)
      PARAMETER (NMAX=500)
      INTEGER i,j,jj
      DOUBLE PRECISION s,tmp(NMAX)
      do 12 j=1,n
        s=0.d0
        if(w(j).ne.0.d0)then
          do 11 i=1,m
            s=s+u(i,j)*b(i)
11        continue
          s=s/w(j)
        endif
        tmp(j)=s
12    continue
      do 14 j=1,n
        s=0.d0
        do 13 jj=1,n
          s=s+v(j,jj)*tmp(jj)
13      continue
        x(j)=s
14    continue
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software j:!&1415S.
```