國 立 交 通 大 學

機 械 工 程 學 系

博 士 論 文

動態系統最佳化設計之計算方案與其應用

Computational Schemes for Dynamic System Optimal Design and its Applications

研 究 生： 黃 智 宏

指導教授： 洪 景 華 教授

曾 錦 煥 教授

中 華 民 國 九十五 年 五 月

動態系統最佳化設計之計算方案與其應用

Computational Schemes for Dynamic System Optimal Design and its Applications

研 究 生： 黃 智 宏　　Student: Chih-Hung Huang

指導教授： 洪 景 華　　Advisor: Ching-Hua Hung
曾 錦 煥　　Ching-Huan Tseng

國 立 交 通 大 學

機械工程學系

博士論文

A Dissertation

Submitted to Department of Mechanical Engineering

College of Engineering

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Mechanical Engineering

May 2006

Hsinchu, Taiwan, Republic of China

中 華 民 國 九十五 年 五 月

國立交通大學

論文口試委員會審定書

本校 機械工程 學系博士班 黃智宏 君

所提論文(中文) 動態系統最佳化設計之計算方案與其應用

(英文) Computational Schemes for Dynamic System Optimal Design and its Applications

合於博士資格水準、業經本委員會評審認可。

口試委員：蔡忠杓　洪景華

蕭德瑛　王安邦

宋震國

林清安

指導教授：洪景華

系主任：傅武雄 教授

中華民國 九十五 年 五 月 五 日

動態系統最佳化設計之計算方案與其應用

研究生：黃智宏　　指導教授：洪景華 曾錦煥

國立交通大學機械工程學系

摘 要

ABSTRACT (IN CHINESE)

動態系統所引發的特性一直困擾著工程設計人員，而只在靜態系統模式下，採用最佳化設計方法所求得的設計，則往往在實際的應用上有所不足。本文主要依據最佳設計與最佳控制理論基礎，結合動態分析與數值分析求解技巧，發展一套通用之動態系統最佳設計方法與軟體。

一般動態系統之最佳化問題可以轉換成標準的最佳控制問題，再透過離散技術轉換成非線性規劃問題，如此便可利用現有之最佳化軟體進行求解。在本文中，首先將動態系統的解題方法與流程發展為最佳控制分析模組，再將該模組與最佳化分析軟體 (MOST) 整合得到整合最佳控制軟體，可以用來解決各種類型的最佳控制問題。為驗證軟體的效能與準確性，利用本文所發展之整合最佳控制軟體求解文獻資料中所提出之各類型最佳控制問題。藉由分析結果之數值與控制軌跡曲線的比對，整合最佳控制軟體所求出之數值解，在效能與準確性上都能與文獻資料所獲得的最佳解吻合，確認該整合最佳控制軟體的確可以用來解決我們工程應用上的最佳控制問題。

另外，針對工程設計中存在的離散（整數）最佳控制問題，本文依據混合整數非線性規劃法(mixed integer nonlinear programming) 做進一步的研究。猛撞型控制 (bang-bang type control) 是常見的離散最佳控制問題，其複雜與難解的特性更是吸引諸多文獻探討的主因。許多文獻針對此一問題所提出的方法多在控制函數的切換點數量為已知的假設

條件下所推導，但這並不符合實際工程上的應用需求，因為控制函數的切換點數量大多在求解完成後才會得知。因此，本文針對此類型問題發展出兩階段求解的方法，第一階段先粗略求解該問題在連續空間下的解，並藉此求得控制函數可能的切換點資訊，第二階段再利用混合整數非線性規劃法求解該問題的真實解。發展過程中，加強型的分支界定演算法 (enhanced branch-and-bound method)被實際應用並且納入前一階段所開發的整合最佳控制軟體中，這也使得這個軟體可以同時處理實際動態系統中最常見的連續及離散最佳控制問題。

最後，本文將所發展的整合最佳控制軟體用來求解兩個實際的工程應用問題：飛航高度控制問題與車輛避震系統設計問題。兩個問題都屬於高階非線性控制問題，首先利用本文中所建議的解題步驟建構完成這兩個問題的數學模型，接著直接利用本研究所發展的軟體求解符合問題要求的最佳解。經由這些實際應用案例的驗證，顯示本文所發展的方法與軟體的確可以提供工程師、學者與學生一個便利可靠的動態系統設計工具。

Computational Schemes for Dynamic System Optimal Design and its Applications

Student: Chih-Hung Huang Advisor: Ching-Hua Hung
Ching-Huan Tseng

Department of Mechanical Engineering
National Chiao Tung University

ABSTRACT

The nonlinear behaviors of dynamic system have been of continual concern to both engineers and system designers. In most applications, the designs – based on a static model and obtained by traditional optimization methods – can never work perfectly in dynamic cases. Therefore, researchers have devoted themselves to find an optimal design that is able to meet dynamic requirements. This dissertation focuses on developing a general-purpose optimization method, based on optimization and optimal control theory, one that integrates dynamic system analysis with numerical technology to deal with dynamic system design problems.

A dynamic system optimal design problem can be transformed into an optimal control problem (OCP). Many scholars have proposed methods to solve optimal control problems and have outlined discretization techniques to convert the optimal control problem into a nonlinear programming problem that can then be solved using extant optimization solvers. This dissertation applies this method to develop a direct optimal control analysis module that is then integrated into the optimization solver, MOST. The numerical results of the study indicate that the solver produces quite accurate results and performs even better than those reported in the earlier literatures. Therefore, the capability and accuracy of the optimal control problem solver is indisputable, as is its suitability for engineering applications.

A second theme of this dissertation is the development of a novel method for solving discrete-valued optimal control problems arisen in many practical designs; for example, the bang-bang type control that is a common problem in time-optimal control problems. Mixed-integer nonlinear programming methods are applied to deal with those problems in this dissertation. When the controls are assumed to be of the bang-bang type, the time-optimal control problem becomes one of determining the switching times. Whereas several methods for determining the time-optimal control problem (TOCP) switching times have been studied extensively in the literature, these methods require that the number of switching times be known before their algorithms can be applied. Thus, they cannot meet practical demands because the number of switching times is usually unknown before the control problems are solved. To address this weakness, this dissertation focuses on developing a computational method to solve discrete-valued optimal control problems that consists of two computational phases: first, switching times are calculated using existing continuous optimal control methods; and second, the information obtained in the first phase is used to compute the discrete-valued control strategy. The proposed algorithm combines the proposed OCP solver with an enhanced branch-and-bound method and hence can deal with both continuous and discrete optimal control problems.

Finally, two highly nonlinear engineering problems – the flight level control problem and the vehicle suspension design problem – are used to demonstrate the capability and accuracy of the proposed solver. The mathematical models for these two problems can be successfully established and solved by using the procedure suggested in this dissertation. The results show that the proposed solver allows engineers to solve their control problems in a systematic and efficient manner.

ACKNOWLEDGEMENT (誌謝)

博士論文的完成，首先要感謝及感念的是我的首位指導教授，『最佳化實驗室』的大家長—曾錦煥教授，沒有他在學術課業上的指導，生活上的照應，待人處事經驗上的分享，不會有這份論文的完成，僅將此論文獻給來不及在這上面簽名，我亦師亦父亦友的 曾錦煥博士。

感謝我另一位指導教授 洪景華博士以及齒輪實驗室 蔡忠杓教授，在我們失去曾老師、頓失依靠之際，毫無保留的給予我們最大的協助，讓這份論文得以順利完成，兩位老師真心付出，讓學生滿心感激。

對於我的口試委員：清華大學動機系蕭主任德瑛、清華大學動機系宋震國老師、交通大學機械系學蔡忠杓老師、台灣大學醫學工程研究所呂東武老師與崴昊科技陳申岳老師，感謝您們不辭辛勞，能撥冗前來指導學生的論文口試，並給予許多適當的指正與寶貴的建議，使學生的論文內容更加充實與完整。

從民國八十年進入『最佳化實驗室』至今已有 15 年，其間與數屆的學長姐、學弟妹相互學習、討論與砥勉，讓我的學習及研究得以長進，也讓我感受到另一個家庭的溫暖。在此，我要一併感謝他們，尤其是東武、宏榮與接興等學長們在我學習遭遇挫折與困頓時給予我極大的鼓勵與支持，在此特別感謝。另外。對於我的同窗好友也是我事業上的伙伴—寬賢，感謝你在這些年來給予我多方面的協助，讓我能有機會在學術研究之外，增長實務與管理上的見識。

漫長的學習與研究過程中，需要的除了耐心與努力外，家庭溫暖的親情是支持我堅持下去的主要動力。從小父母親在南部農村的困頓環境下，縮衣節食，即使四處借貸，也堅持給我們就學的機會，讓我得以順利取得家族中第一個碩士學位。之後，內人雅惠在我攻讀博士其間更是無怨無悔的照料家中大小事務，哺育兩個年幼的兒子(國賜與家富)，她長年的包容與等待，讓我可以無後顧之憂，在此論文完成之際，對他們除了感謝還是感謝！

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

$\boldsymbol{b}$	design variables
$\boldsymbol{d}$	descent direction defined in SQP algorithm
k	iteration counter
s	the wing surface area
t_0	start time
t_f	terminal time
$\boldsymbol{u}$	control variable vector
$\boldsymbol{u_c}$	piecewise continuous control variable vector
$\boldsymbol{u_d}$	discontinuous control variable vector
$\boldsymbol{x}$	state variable vector
$\boldsymbol{B}_k$	BFGS approximation of Hessian matrix
C_L	dimensionless lift coefficient
C_D	dimensionless drag coefficient
D	the aerodynamic drag force of a aircraft
$\boldsymbol{H}$	Hamiltonian function
$\boldsymbol{H_L}$	Hessian matrix of the Lagrangian function
I	interpolation function of control variables
J_{max}	Cost bound of mixed-integer NLP
J_0	function of performance index (cost function)
L	the aerodynamic lift force of a aircraft
$\boldsymbol{L}_c$	(augmented) Lagrangian function
N	number of time intervals
N_s	maximum iteration number of AOCP algorithm
N_e	number of equality constraints

N_T	number of inequality constraints
$\boldsymbol{P}$	extended design variable vector
$\mathbf{S_D}$	discretized control variable vector
$\mathbf{S}$	Parameter vector of interpolation function for controls
T	the thrust exerted by the aircraft engine
$\mathbf{T}$	time grid vector
T_i	i^{th} time grid point
$\mathcal{U}$	class of all such admissible controls
$\mathcal{U}_d$	class of all discontinuous admissible controls
α	step size of SQP algorithm
ε	convergence parameter of SQP algorithm
$\boldsymbol{\lambda}$	vector of Lagrange multipliers
$\zeta_1^{(l)}$	i^{th} parameter of interpolation function for l time grid
ρ	the air density
Φ_i	function of terminal state
$\mathcal{L}_i$	integral part of performance index and functional constraints
ψ_j	dynamic constraint function

CHAPTER 1
INTRODUCTION

1.1 Dynamic Optimization and Optimization Control Problems

Over the past decade, applications for dynamic systems in engineering have increased significantly. In most applications, the designs, which are based on a static model and obtained by traditional optimization methods, cannot work perfectly in dynamic cases because of their nonlinear behaviors. Therefore, researchers have devoted themselves to find an optimal design that is able to meet dynamic requirements.

Most engineering applications are modeled dynamically using differential algebraic equations (DAE) whose formulation consists of (a) differential equations that describe the dynamic behavior – such as mass and energy balances – of the state of a given system and (b) algebraic equations that ensure physical and dynamic relations. Usually, the dynamic behaviors of a given system can be influenced by the choice of certain control variables. For instance, a vehicle can be controlled by the steering wheel, the accelerator pedal, and the brakes. At the same time, the state and/or control variables cannot assume any value but are subject to certain restrictions, often resulting from safety regulations or physical limitations, such as the altitude of an aircraft being above ground level or the steering angle of a vehicle having a maximum limitation. In addition, engineers are particularly interested in those state and control variables that fulfill all restrictions while also minimizing or maximizing a given objective function. These problems are typically ones of dynamic optimization. By applying modeling and optimization technologies, a dynamic optimization problem can be reformulated as an optimal control problem (OCP).

Even though optimal control problems arise in various disciplines, not all engineers are familiar with optimal control theory. On the other hand, most optimal control problems are interpreted as an extension of nonlinear programming (NLP) problems to an infinite number of variables and solved by numerical methods. For engineers who are inexperienced in

numerical techniques, implementing these numerical techniques is another obstacle in solving dynamic optimization problems. Consequently, a general-purpose solver for optimal control problems coupled with a systematic procedure could assist engineers in solving various optimal control problems.

Time-optimal control problems (TOCP) have attracted the interest of researchers in optimal control because, even they often arise in practical applications, their solutions are difficult. In practical applications, one of the most common types of control function is the piecewise-constant function by which a sequence of constant inputs is used to control a given system with suitable switching times. Nevertheless, many methods proposed in the literature – for example, the switching time computations algorithm (Lucas and Kaya, 2001) – assume that the number of switching times is known before their algorithms are applied. In reality, however, the number of switching times is generally unknown before most control problems are solved. Therefore, an efficient algorithm for determining the switching times of TOCP becomes important and attracts the interest of researchers.

1.2 Literature Review

1.2.1 Methods for Optimal Control Problems

Optimal control problems can be solved by a variational method (Pontryagin *et al.*, 1962) or by nonlinear programming approaches (Huang and Tseng, 2003, 2004; Hu *et al.*, 2002; Jaddu and Shimemura, 1999). The variational or indirect method is based on the solution of first-order necessary conditions for optimality obtained from Pontryagin's maximum principle (Pontryagin *et al.*, 1962). For problems without inequality constraints, the optimality conditions can be formulated as a set of differential-algebraic equations, often in the form of a two-point boundary value problem (TPBVP). The TPBVP can be addressed using many approaches, including single shooting, multiple shooting, invariant embedding, or a discretization method such as collocation on finite elements. On the other hand, if the problem

requires that active inequality constraints be handled, finding the correct switching structure, as well as suitable initial guesses for the state and costate variables, is often very difficult.

Much attention has been paid in the literature to the development of numerical methods for solving optimal control problems (Hu *et al.*, 2002; Pytlak, 1999; Jaddu and Shimemura, 1999; Teo, and Wu, 1984; Polak, 1971), the most popular approach in this field is the reduction of the original problem to a NLP problem. Nevertheless, in spite of extensive use of nonlinear programming methods to solve optimal control problems, engineers still spend much effort reformulating nonlinear programming problems for different control problems. Moreover, implementing the corresponding programs for the nonlinear programming problem is tedious and time consuming. Therefore, a general OCP solver coupled with a systematic computational procedure for various optimal control problems has become an imperative for engineers, particularly for those who are inexperienced in optimal control theory or numerical techniques.

Additionally, in many practical engineering applications, the control action is restricted to a set of discrete values. These systems can be classified as switched systems consisting of several subsystems and switching laws that orchestrate the active subsystem at each time instant. Optimal control problems for switched systems, which require solution of both the optimal switching sequences and the optimal continuous inputs, have recently drawn the attention of many researchers. The primary difficulty with these switched systems is that the range set of the control is discrete and hence not convex. Moreover, choosing the appropriate elements from the control set in an appropriate order is a nonlinear combinatorial optimization problem. In the context of time optimal control problems, as pointed out by Lee *et al*. (1997), serious numerical difficulties may arise in the process of identifying the exact switching points. Therefore, an efficient numerical method is still needed to determine the exact control switching times in many practical engineering problems.

1.2.2 Time-Optimal Control Problems

The TOCP is one of most common types of OCP, one in which only time is minimized and the control is bounded. In a TOCP, a TPBVP is usually derived by applying Pontryagin's maximum principle (PMP). In general, time-optimal control solutions are difficult to obtain (Pinch, 1993) because, unless the system is of low order and is time invariant and linear, there is little hope of solving the TPBVP analytically (Kirk, 1970). Therefore, in recent research, many numerical techniques have been developed and adopted to solve time-optimal control problems.

One of the most common types of control function in time-optimal control problems is the piecewise-constant function by which a sequence of constant inputs is used to control a given system with suitable switching times. Additionally, when the control is bounded, a very commonly encountered type of piecewise-constant control is the bang-bang type, which switches between the upper and lower bounds of the control input. When the controls are assumed to be of the bang-bang type, the time-optimal control problem becomes one of determining the switching times, several methods for which have been studied extensively in the literature (see, *e.g.*, Kaya and Noakes, 1996; Bertrand and Epenoy, 2002; Simakov *et al*., 2002). However, as already mentioned, in contrast to practical reality, these methods require that the number of switching times be known before their algorithms can be applied. To overcome the numerical difficulties arising during the process of finding the exact switching points, Lee *et al*. (1997) proposed the control parameterization enhancing transform (CPET), which they also extended to handle the optimal discrete-valued control problems (Lee *et al*., 1999) and applied to solve the sensor-scheduling problem (Lee *et al*., 2001).

In similar manner, this dissertation focuses on developing a numerical method to solve time-optimal control problems. This method consists of the two-phase scheme: first, switching times are calculated using existing optimal control methods; and second, the

resulting information is used to compute the discrete-valued control strategy. The proposed algorithm, which integrates the admissible optimal control problem formulation with an enhanced branch-and-bound method (Tseng *et al*., 1995), is then implemented and applied to some examples.

1.3 Objectives

The major purpose of this dissertation is to develop a computational method to solve the time-optimal control problems and find the corresponding discrete-valued optimal control laws. The other purpose of this dissertation is to implement a general OCP solver and provide a systematic procedure for solving OCPs that provides engineers with a systematic and efficient procedure to solve their optimal control problems.

1.4 Outlines

The dissertation is organized as follows. Chapter 2 introduces the formulations for various optimal control problems and the general methods for solving such problems. Also briefly discussed are problem-solving procedures and the difficulties with direct and indirect methods. Chapter 3 specifically addresses the computational methods for solving optimal control problems and presents the theoretical basis and numerical preliminaries for developing a general optimal control problem solver. The architecture of the OCP solver and the systematic procedure for solving the OCP are described in Chapter 4, which also present the details of the implementation and user interface of the proposed solver. Here, the *van der Pol* oscillator problem with various types of terminal conditions and the time-optimal control problem of overhead crane control are used to demonstrate and verify the capability and accuracy of the proposed OCP solver. Chapter 5 introduces a two-phase scheme that integrates the admissible optimal control problem method and the enhanced branch-and-bound algorithm to efficiently solve the bang-bang control problems in the field of engineering. In Chapter 6, the proposed

solver is applied to two practical engineering applications: the flight level control problem and the vehicle suspension design problem. Finally, Chapter 7 draws some conclusions and makes suggestions for further research.

CHAPTER 2
METHODS FOR SOLVING OPTIMAL CONTROL PROBLEMS

2.1 Introduction

Optimal control theory has been of considerable importance in a wide variety of disciplines. Over the years, the theory has been developed for various applications in many different fields, *e.g.*, mechanical systems (Kim and Ha, 2001), automotive vehicle design (Panagiotis, 2000; Jalili and Esmailzadeh, 2001), and manufacturing processes (Samaras and Simaan, 2001). However, because most real-world problems are becoming too complex to be solved analytically (Kirk, 1970), using computational algorithms to solve them is becoming inevitable. As a result, several successful families of algorithms are now available in the literature.

Techniques for the numerical solution of optimal control problems can be broadly divided into direct and indirect methods (Bock, 1978; Stryk and Bulirsch, 1992). In the direct method, the state and/or control variables are parameterized using a piecewise polynomial approximation. Inserting these approximations into the cost functional, dynamic equations, and constraints and boundary conditions leads to a static parameter optimization problem. On the other hand, the indirect method is based on the solution of the first-order necessary conditions for optimality obtained from Pontryagin's maximum principle (Pontryagin *et al.*, 1962) or derived from the Hamilton-Jacobi-Bellman equation (Bellman, 1957).

Two early methods commonly used to solve optimal control problems are Bellman's dynamic programming (Bellman, 1957) and Pontryagin's maximum principle (Pontryagin *et al*., 1962). Dynamic programming, developed by Bellman in the late 1950s (Bellman, 1957; Bellman and Dreyfus, 1962; Bellman and Kalaba, 1965), is a computational technique that extends the decision-making concept to sequences of decisions, which together define an optimal policy and trajectory. Subsequently, Soviet mathematician Pontryagin and his

colleagues (Pontryagin *et al*., 1962) developed the calculus of variations approach using a maximum principle. Although both the dynamic programming method and PMP have been used to solve optimal control problems, many practical problems described by strongly nonlinear differential equations cannot be easily solved by either technique. As a result, many approximation methods based on NLP methods are used to solve these practical problems (see, *e.g.*, Lin, 1992; Pytlak, 1999; Jaddu and Shimemura, 1999; Hu *et al*., 2002).

A nonlinear programming problem consists of a multivariable function subject to multiple inequality and equality constraints. The solution to the nonlinear programming problem is found by solving the Kuhn-Tucker points of equalities given by the first-order boundary conditions. Conceptually, this procedure is analogous to solving optimal control problems using Pontryagin's maximum principle. Depending on the discretization technique applied, methods that apply NLP solvers can be classified into two groups: simultaneous or sequential strategies. In the simultaneous methods, the state and control variables are fully discretized and thus usually lead to large-scale NLP problems that require special solution strategies (Cervantes and Biegler, 1998; Betts and Huffman, 1992). However, in sequential methods – also known as control variable parameterization methods – only the control variables are discretized. Based on initial conditions and a set of control parameters, the system equations are integrated with an ordinary differential equation (ODE) solver at each iteration to produce cost functional (performance index) and constraint values used by a nonlinear programming solver to find the control parameterization's optimal coefficient values. The sequential approach is a feasible path method, *i.e.*, in each of iteration, the system equation is solved. However, this procedure is robust only when the system contains stable modes. Otherwise, finding a feasible solution for a given set of control parameters may be difficult. In this dissertation, a different discretization technique – the shooting method – is implemented and used in conjunction with sequential quadratic programming (SQP) to solve various types of

optimal control problems.

The shooting method serves as a bridge between sequential and simultaneous approaches by partitioning the time domain into smaller time intervals and integrating the system equations separately in each interval. Control variables are treated in the same manner as in the sequential approach. Moreover, to obtain gradient information, sensitivities are obtained for both the control variables and the initial state conditions in each time interval. Finally, equality constraints are added to the nonlinear program in order to link the time intervals and ensure that the states are continuous across each time interval. This method allows inequality constraints for both the state and the controls to be imposed directly at the grid points. Thus, the admissible optimal control problem (AOCP) formulation based on the shooting method is adopted as the core of the proposed method.

2.2 Canonical Formulation of Optimal Control Problems

Considering a dynamical system described by the following nonlinear differential equations on [0, t_f]:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}\left(t, \boldsymbol{b}, \boldsymbol{x}(t), \boldsymbol{u}(t)\right), \qquad t \in \left[0, t_f\right] \tag{2.1}$$

with the initial condition

$$\boldsymbol{x}(0) = \boldsymbol{x}_0, \tag{2.2}$$

where t_f is the terminal time, $\boldsymbol{b} \in \mathbb{R}^{\pi}$ is the vector of design variables, $\boldsymbol{u}(t) \equiv \left[u_1(t), u_2(t), \cdots, u_m(t)\right]^{\mathrm{T}} \in \mathbb{R}^m$ is a vector of the control functions and $\boldsymbol{x}(t) \equiv \left[x_1(t), x_2(t), \cdots, x_n(t)\right]^{\mathrm{T}} \in \mathbb{R}^n$ is a vector of the state variables. The function $\boldsymbol{f} : \mathbb{R} \times \mathbb{R}^{\pi} \times \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n$ is assumed to be continuously differentiable with respect to all its arguments, and $\boldsymbol{x}_0$ is a given vector in $\mathbb{R}^n$. It is assumed that the process starts from $t_0 = 0$ and ends at the fixed terminal time $t_f > 0$. A process that starts from $t_0 \neq 0$ may be transformed to satisfy this assumption by suitable shifting on the time axis. Let $\mathcal{U}$ be the class of all such

admissible controls. Then an optimal control problem may be stated formally as follows: Given the dynamical system expressed in Eqs. (2.1) and (2.2), find $\boldsymbol{u} \in \mathcal{U}$ such that the cost functional (performance index)

$$J_0 = \Phi_0\left(\boldsymbol{b}, \boldsymbol{x}(t_f), t_f\right) + \int_0^{t_f} \mathcal{L}_0\left(\boldsymbol{b}, \boldsymbol{u}(t), \boldsymbol{x}(t), t\right) dt \tag{2.3}$$

is minimized subject to the constraint

$$J_i = \Phi_i\left(\boldsymbol{b}, \boldsymbol{x}(t_f), t_f\right) + \int_0^{t_f} \mathcal{L}_i\left(\boldsymbol{b}, \boldsymbol{u}(t), \boldsymbol{x}(t), t\right) dt \begin{cases} = 0; & i = 1, ..., N_e \\ \le 0; & i = N_e + 1, ..., N_T \end{cases} \tag{2.4}$$

and the following continuous inequality constraint on the function of the state and control:

$$\psi_j\left(\boldsymbol{b}, \boldsymbol{u}(t), \boldsymbol{x}(t), t\right) \le 0; j = 1, ..., q \qquad , \forall t \in \left[0, t_f\right]. \tag{2.5}$$

where Φ_0, $\mathcal{L}_0$, Φ_i, $\mathcal{L}_i$ and ψ_j are continuously differentiable with respect to their respective arguments. This problem is referred to as problem (P$\mathcal{v}$). A control $\boldsymbol{u} \in \mathcal{U}$ is said to be a feasible control if it satisfies constraints (2.4) and (2.5).

The preceding definition extends the original Bolza problem to account for inequality constraints because the original Bolza formulation, containing only equality constraints, is not general for the OCP. It also fails to treat the design variables $\boldsymbol{b}$, which may serve a variety of useful purposes apart from the obvious design parameters, *e.g.*, weight and velocity of a vehicle. Moreover, when the terminal time t_f is unconstrained (for optimization), a free-time problem occurs. Otherwise, a fixed-time problem is given. In addition, the initial conditions are separated from the functional constraints in Eq. (2.4) for practical considerations, and the terminal conditions are treated as equality constraints in the first term of Eq. (2.4). The differential equations for the system in Eq. (2.1) are written in general first-order form. Equation (2.5) represents the mixed state and control inequality dynamic constraints.

According to the constraints encountered in practical applications, most constraints can be classified under one of the following categories (Teo *et al.*, 1991):

Type 1. Control bounds:

$$\boldsymbol{u}_{\min} \le \boldsymbol{u}(t) \le \boldsymbol{u}_{\max} \qquad , \forall t \in \left[0, t_f\right] \tag{2.6}$$

Type 2. Terminal state constraint with fixed terminal time:

$$\Phi_i\left(\boldsymbol{b}, \boldsymbol{x}(t_f), t_f\right) \begin{cases} = 0; & i = 1, ..., N_e \\ \le 0; & i = N_e + 1, ..., N_T \end{cases} \qquad , t_f \text{ is fixed.} \tag{2.7}$$

Type 3. Terminal state constraint with free terminal time:

$$\Phi_i\left(\boldsymbol{b}, \boldsymbol{x}(t_f), t_f\right) = 0 \qquad , t_f \text{ is unspecified.} \tag{2.8}$$

Type 4. Interior point state constraint:

$$\Phi_i\left(\boldsymbol{b}, \boldsymbol{x}(t_l), t_l\right) = 0 \qquad , 0 < t_l < t_f \tag{2.9}$$

Type 5. Integral constraint:

$$\int_0^{t_f} \mathcal{L}_i\left(\boldsymbol{b}, \boldsymbol{u}(t), \boldsymbol{x}(t), t\right) dt \begin{cases} = 0; & i = 1, ..., N_e \\ \le 0; & i = N_e + 1, ..., N_T \end{cases} \tag{2.10}$$

Type 6. Continuous equality constraint on the function of the state and control:

$$\Phi_i\left(\boldsymbol{b}, \boldsymbol{x}(t), \boldsymbol{u}(t), t\right) = 0 \qquad , \forall t \in \left[0, t_f\right] \tag{2.11}$$

Type 7. Continuous inequality constraint on the function of the state and control:

$$\Phi_i\left(\boldsymbol{b}, \boldsymbol{x}(t), \boldsymbol{u}(t), t\right) \le 0 \qquad , \forall t \in \left[0, t_f\right] \tag{2.12}$$

To develop a general optimal control solver, any constraint of type 2 to type 7 can be regarded as a special case of Eqs. (2.4) and (2.5).

2.3 First-Order Necessary Condition – Euler Lagrangian Equation

The first-order necessary condition for optimality, known as the Euler-Lagrangian equation, can be found in many research studies (*e.g.*, Teo *et al*., 1991; Kirk, 1970). Given an optimal control problem where control $\boldsymbol{u} \in \mathcal{U}$ is chosen such that the cost functional defined in Eq. (2.3) is minimized, then

$$J_0 = \Phi_0\left(\boldsymbol{x}(t_f), t_f\right) + \int_0^{t_f} \mathcal{L}_0\left(\boldsymbol{u}(t), \boldsymbol{x}(t), t\right) dt \tag{2.13}$$

where Φ_0 and $\mathcal{L}_0$ are continuously differentiable with respect to their respective arguments.

It should be noted that the cost functional may be regarded as depending explicitly only on $\boldsymbol{u}$, as $\boldsymbol{x}$ is implicitly determined by $\boldsymbol{u}$ from Eqs. (2.1) and (2.2). In addition, the design variables vector, $\boldsymbol{b}$, is treated as a constant and is not involved. The system equations (2.1) and (2.2) can be appended to the cost functional by introducing the appropriate Lagrange multiplier $\boldsymbol{\lambda} \in \mathbb{R}^n$:

$$\begin{aligned} \bar{J}_0\left(\boldsymbol{u}\right) = \Phi_0\left(\boldsymbol{x}\left(t_f\right), t_f\right) + \int_0^{t_f} \Big\{ & \mathcal{L}_0\left(t, \boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right)\right) \\ & + \left(\boldsymbol{\lambda}\left(t\right)\right)^T \left[\boldsymbol{f}\left(t, \boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right)\right) - \dot{\boldsymbol{x}}\left(t\right)\right] \Big\} dt \end{aligned} \tag{2.14}$$

The Hamiltonian function is defined as follows:

$$\boldsymbol{H}\left(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}\right) = \mathcal{L}_0\left(t, \boldsymbol{x}, \boldsymbol{u}\right) + \boldsymbol{\lambda}^T \boldsymbol{f}\left(t, \boldsymbol{x}, \boldsymbol{u}\right) \tag{2.15}$$

It should again be noted that, if the system equation is satisfied, the appended cost functional $\bar{J}_0$ is indifferent to the original J_0. The time dependent Lagrange multiplier is referred to as the *costate vector*, also known as the *adjoint vector*.

Substituting Eq. (2.15) into Eq. (2.14) and integrating the last term by parts, the cost functional becomes

$$\begin{aligned} \bar{J}_0\left(\boldsymbol{u}\right) = \Phi_0\left(\boldsymbol{x}\left(t_f\right)\right) - \left(\boldsymbol{\lambda}\left(t_f\right)\right)^T \boldsymbol{x}\left(t_f\right) + \left(\boldsymbol{\lambda}\left(0\right)\right)^T \boldsymbol{x}\left(0\right) \\ + \int_0^{t_f} \left\{ \boldsymbol{H}\left(t, \boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right), \boldsymbol{\lambda}\left(t\right)\right) + \left(\dot{\boldsymbol{\lambda}}\left(t\right)\right)^T \boldsymbol{x}\left(t\right) \right\} dt \end{aligned} \tag{2.16}$$

For a small variation c in $\boldsymbol{u}$, the corresponding first-order variations in $\boldsymbol{x}$ and $\bar{J}_0$ are $\delta\boldsymbol{x}$ and $\delta\bar{J}_0$, respectively, where $\delta\bar{J}_0$ is obtained by the chain rule:

$$\delta \bar{J}_0(\boldsymbol{u}) = \left[\frac{\partial \Phi_0(\boldsymbol{x}(t))}{\partial \boldsymbol{x}} - \left(\boldsymbol{\lambda}(t_f)\right)^T \right] \delta \boldsymbol{x}(t_f) + \left(\boldsymbol{\lambda}(0)\right)^T \delta \boldsymbol{x}(0)$$
$$+ \int_0^{t_f} \left\{ \left[\frac{\partial \boldsymbol{H}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t))}{\partial \boldsymbol{x}} + \left(\dot{\boldsymbol{\lambda}}(t)\right)^T \right] \delta \boldsymbol{x}(t) \right. \tag{2.17}$$
$$\left. + \frac{\partial \boldsymbol{H}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t))}{\partial \boldsymbol{u}} \delta \boldsymbol{u}(t) \right\} dt$$

Since $\boldsymbol{\lambda}(t)$ is arbitrary so far, it can be set as

$$\left(\dot{\boldsymbol{\lambda}}(t)\right)^T = -\frac{\partial \boldsymbol{H}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t))}{\partial \boldsymbol{x}} \tag{2.18}$$

with boundary condition:

$$\left(\boldsymbol{\lambda}(t_f)\right)^T = \frac{\partial \Phi_0(\boldsymbol{x}(t_f))}{\partial \boldsymbol{x}} \tag{2.19}$$

As the initial condition $\boldsymbol{x}(0)$ is fixed, $\delta \boldsymbol{x}(0)$ vanishes and Eq. (2.17) reduces to

$$\delta \bar{J}_0(\boldsymbol{u}^*) = \int_0^{t_f} \left\{ \frac{\partial \boldsymbol{H}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t))}{\partial \boldsymbol{u}} \delta \boldsymbol{u}(t) \right\} dt \tag{2.20}$$

For a local minimum, it is necessary that $\delta \bar{J}_0$ vanishes for arbitrary $\delta \boldsymbol{x}$. Therefore, it is necessary that

$$\frac{\partial \boldsymbol{H}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t))}{\partial \boldsymbol{u}} = 0 \tag{2.21}$$

for all $t \in [0, t_f]$, except on a finite set. It should be noted that this holds only if no bounds on $\boldsymbol{u}$ exist; otherwise, the Pontryagin's maximum principle to be discussed later will be applied. Equations (2.1), (2.2), (2.18), (2.19), and (2.21) are the well-known Euler-Lagrangian equations whose results can be summarized in the following theorem.

Theorem 2.1 If $\boldsymbol{u}^*(t)$ is a control that yields a local minimum for the cost functional (2.13), and $\boldsymbol{x}^*(t)$ and $\boldsymbol{\lambda}^*(t)$ are the corresponding state and costate, then it is necessary that

$$\left(\dot{\boldsymbol{x}}^*(t)\right)^T = \left[\frac{\partial \boldsymbol{H}\left(t, \boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t)\right)}{\partial \boldsymbol{\lambda}}\right]^T = f\left(t, \boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t)\right) \quad (2.22a)$$

$$\boldsymbol{x}^*(0) = \boldsymbol{x}_0 \quad (2.22b)$$

$$\left(\dot{\boldsymbol{\lambda}}^*(t)\right) = -\left[\frac{\partial \boldsymbol{H}\left(t, \boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t)\right)}{\partial \boldsymbol{x}}\right]^T \quad (2.22c)$$

$$\boldsymbol{\lambda}(t_f) = \left[\frac{\partial \Phi_0\left(\boldsymbol{x}^*(t_f)\right)}{\partial \boldsymbol{x}}\right]^T \quad (2.22d)$$

and, for all $t \in [0,\ t_f]$, except possibly on a finite subset of $[0,\ t_f]$,

$$\frac{\partial \boldsymbol{H}\left(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t)\right)}{\partial \boldsymbol{u}} = 0 \quad (2.22e)$$

It should be noted that Eqs. (2.22a)-(2.22d) constitute $2n$ differential equations with n boundary conditions for $\boldsymbol{x}^*$ specified at $t = 0$ and n boundary conditions for $\boldsymbol{\lambda}^*$ specified at $t = t_f$. This is referred to as a two-point boundary value problem. In principle, the dependence on $\boldsymbol{u}^*$ can be removed by solving $\boldsymbol{u}^*$ as a function of $\boldsymbol{x}^*$ and $\boldsymbol{\lambda}^*$ from the m algebraic equations in Eq. (2.22e) via the implicit function theorem, provided that the Hessian $\boldsymbol{H}_{uu} \equiv \frac{\partial}{\partial \boldsymbol{u}}\left[\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{u}}\right]^T$ is nonsingular at the optimal point.

2.4 Methods for Solving Optimal Control Problems

2.4.1 Indirect Methods

As mentioned in Section 1.2.1, the indirect method is based on the solution of the first-order necessary conditions for optimality obtained from Pontryagin's maximum principle (Pontryagin *et al.*, 1962), which has been modified and applied in various applications (see,

e.g., Xu and Antsaklis, 2004; Chyba *et al.*,2003; Steindl and Troger, 2003). For problems without inequality constraints, the optimality conditions can be formulated as a set of differential-algebraic equations (DAEs). Obtaining a solution to DAEs requires careful attention to the boundary conditions because the state variables frequently have specified initial conditions and costate (adjoint) variables whose final conditions result in a TPBVP that is notoriously difficult to solve analytically and requires the use of iterative numerical techniques (Kirk, 1970). On the other hand, if the problem requires that active inequality constraints be handled, finding the correct switching structure together with suitable initial guesses for state and costate variables is often very difficult because of a lack of physical significance and the need for prior knowledge of the control's switching structure. Many numerical techniques, including single shooting, invariant embedding, and multiple shooting, can be used to solve TPBVP, but PMP does not deal well with nonlinear optimal control problems. Figure 2.1 shows a solution process based on indirect methods.

Pontryagin's Maximum Principle

According to the Euler-Lagrangian equation for the unconstrained optimal control problem of Section 2.3 depicts that the Hamiltonian function must necessarily be stationary with respect to the control, *i.e.* $\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{u}} = 0$ at optimality. However, the optimality condition obtained in Section 2.3 does not have to be satisfied if the control is constrained to lie on the boundary of a subset $\mathbf{U}_s$. Here, $\mathbf{U}_s$ is a compact subset of $\mathbb{R}^r$. Then, the Pontryagin's maximum principle can be described by the following theorem:

Theorem 2.2 Given the problem, where the cost functional (2.13) is to be minimized over $\boldsymbol{\mathcal{U}}$ subjected to the system equations (2.1) and (2.2), if $\boldsymbol{u}^*(t) \in \boldsymbol{\mathcal{U}}$ is an optimal control, and $\boldsymbol{x}^*(t)$ and $\lambda^*(t)$ are the corresponding state and costate, then it is necessary that

$$\left(\dot{\boldsymbol{x}}^*(t)\right)^T = \left[\frac{\partial \boldsymbol{H}\left(t, \boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t)\right)}{\partial \boldsymbol{\lambda}}\right]^T = f\left(t, \boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t)\right) \quad (2.23a)$$

$$\boldsymbol{x}^*(0) = \boldsymbol{x}_0 \quad (2.23b)$$

$$\left(\dot{\boldsymbol{\lambda}}^*(t)\right) = -\left[\frac{\partial \boldsymbol{H}\left(t, \boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t)\right)}{\partial \boldsymbol{x}}\right]^T \quad (2.23c)$$

$$\boldsymbol{\lambda}(t_f) = \left[\frac{\partial \Phi_0\left(\boldsymbol{x}^*(t_f)\right)}{\partial \boldsymbol{x}}\right]^T \quad (2.23d)$$

and, for all $t \in [0,\ t_f]$, except possibly on a finite subset of $[0,\ t_f]$,

$$\boldsymbol{H}\left(t, \boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t)\right) \le \boldsymbol{H}\left(t, \boldsymbol{x}^*(t), \boldsymbol{u}(t), \boldsymbol{\lambda}^*(t)\right) \quad (2.23e)$$

for all $t \in [0, t_f]$.

Dynamic Programming

Dynamic programming (DP), based on Bellman's principle of optimality (Bryson and Ho, 1975; Bellman and Dreyfus, 1962; Bellman, 1957), requires solution of the Hamilton-Jacobi-Bellman partial differential equation in a domain of the state space that contains the optimal solution. In dynamic programming, the optimal control problem is expressed as a state-variable feedback in graphical or tabular form. Optimal control strategies must be determined by working backward from the final stages. In other words, this method operates in sweeps through the state set, performing a full backup operation on each state. Each backup updates the value of one state based on the value of all possible successor states.

The computational procedure for dynamic programming can be described briefly by the following steps.

Step 1: Approximating the continuous-time system using a discrete-time system.

In this step, the time interval, $[t_0, t_f]$, is divided into N equal spaced intervals, Δt, and the performance index and state equations are converted into discrete form. Then, by applying the principle of optimality, the performance index can be converted into recurrent form:

$$\begin{aligned} \boldsymbol{J}^*_{N-k,N}\left(\boldsymbol{x}\left(N-k\right)\right) = \min_{u(N-k)} \{ & \mathcal{L}_{\mathrm{D}}\left(\boldsymbol{b}, \boldsymbol{u}(N-k), \boldsymbol{x}(N-k)\right) \\ & + \boldsymbol{J}^*_{N-(k-1),N}(\Delta t \cdot \boldsymbol{f}(\boldsymbol{b}, \boldsymbol{x}(N-k), \boldsymbol{u}(N-k)))\} \end{aligned} \tag{2.24}$$

Step 2: Quantizing the admissible state and control values into a finite number of levels.

Step 3: Calculating and storing the minimum values of the performance index of each stage from final state to initial state. In each stage, every quantized control value is tried at each quantized state value to discover the corresponding state values of the next stage. Additionally, the value of the performance index from current stage to final stage is calculated and compared. The minimum performance index is then chosen and stored. If the corresponding state values of the next stage are not in the quantized grid points, interpolation is required.

Step 4: Showing the results.

2.4.2 Direct Methods

Direct methods try to solve the dynamic optimization problem directly without explicitly solving the necessary conditions. Usually, these methods are based on an iterative procedure that generates approximations to the optimal solution of the dynamic optimization problem within each iteration step. For instance, the SQP method uses quadratic subproblems to approximate a general nonlinear programming problem locally.

As mentioned in Section 2.1, most direct methods that apply NLP solvers can be classified into simultaneous and sequential strategies. The important question for these numerical direct methods is whether these iterative approximate algorithms converge to a solution of the original problem or not. A solution process based on such methods is shown in Figure 2.2 and

their details will be introduced in Chapter 3.

2.5 Summary

The primary objective of this chapter has been to survey methods of the optimal control problems and provide formulations of various types of optimal control problems. The first-order necessary condition (Euler-Lagrangian equation) has also been briefly introduced to provide the theoretical foundation for Pontryagin's maximum principle. In addition, the chapter has described two typical methods for solving optimal control problems – indirect and direct approaches – whose advantages and drawbacks are listed in Table 2.1. Understanding the advantages of and difficulties with these methods will help engineers apply them to problem solving.

As regards applicability, dynamic programming (DP) is sometimes thought to be limited because of "*the curse of dimensionality*" (Bellman, 1957), *i.e.*, the fact that the number of states often grows exponentially with the number of state variables. In reality, even though large state sets do create difficulties, these are the inherent difficulties of the problem not of DP as a solution method. In fact, the DP method can be used with today's computers to solve optimal control problems with millions of states. In particular, dynamic programming can deal with multistage optimal control problems that are difficult to solve using other methods. Nevertheless, even though dynamic programming can be used to solve optimal control problems in nonlinear time-variant systems, using it to deal with time-optimal trajectory planning is difficult in practice because it relies on the exact dynamic models of the system. Yet, unfortunately, the time-optimal control problem is a very common application of the optimal control problem.

In contrast, Pontryagin's maximum principle, which provides the analytical foundation for this study, can deal with various types of optimal control problem. However, in any such control problem, PMP unfortunately leads to a nonlinear two-point boundary value problem

that, as earlier mentioned, is notoriously difficult to solve analytically and requires the use of iterative numerical techniques (Kirk, 1970).

Furthermore, neither DP nor PMP can serve as a convenient and complete method for reformulating different control problems. Rather, engineers either have to derive the state equations, costate equations, and boundary conditions from PMP or have to reformulate the discrete form of the system equations and performance index by applying the DP algorithm. Engineers must then also implement numerical programs to solve the TPBVP using PMP or execute recurrence equations using DP. For engineers inexperienced in optimal control theory or numerical techniques, carrying out these theoretical derivations and program implementations is difficult. Thus, a general-purpose solver is needed for various types of optimal control problems.

From a practical viewpoint, of the two types of NLP methods compared in Section 2.1 (simultaneous and sequential strategies), the sequential NLP methods are the best for developing a general-purpose problem solver.

Table 2.1 Comparison of the methods for solving optimal control problems.

Method	Advantages	Disadvantages / Difficulties
Dynamic programming method	1. Can obtain global optimal solutions. 2. Can deal with nonlinear constrained time-variant systems. 4. Suits multistage optimal control problems. 3. Is straightforward to program.	1. Hard to apply the algorithms for time-optimal trajectory planning in practice. 2. Inconvenient to reuse.
Pontryagin's minimum principle	1. Provides the analytical foundation. 2. Can deal with various types of optimal control problem.	1. Leads to a nonlinear TPBVP that is difficult to solve. 2. Inconvenient to reuse.
Simultaneous NLP methods	1. Can deal with path constraint problems. 2. Can be implemented as a general OCP solver.	1. The computational efficiency is slowed for large-scale problems. 2. Needs extra efforts to deal with inconsistency problem between state equations and controls. 3. Needs a proper initial guess to obtain the optimal solution.
Sequential NLP methods	1. Can deal with various types of nonlinear optimal control problem. 2. Easy to implement as a general OCP solver. 3. Many well-developed numerical schemes can be applied to solve initial value problems. 4. Higher computational efficiency for solving large-scale problems.	1. Needs a proper initial guess to obtain the optimal solution. 2. Path constraints for the states may not be satisfied between grid points.

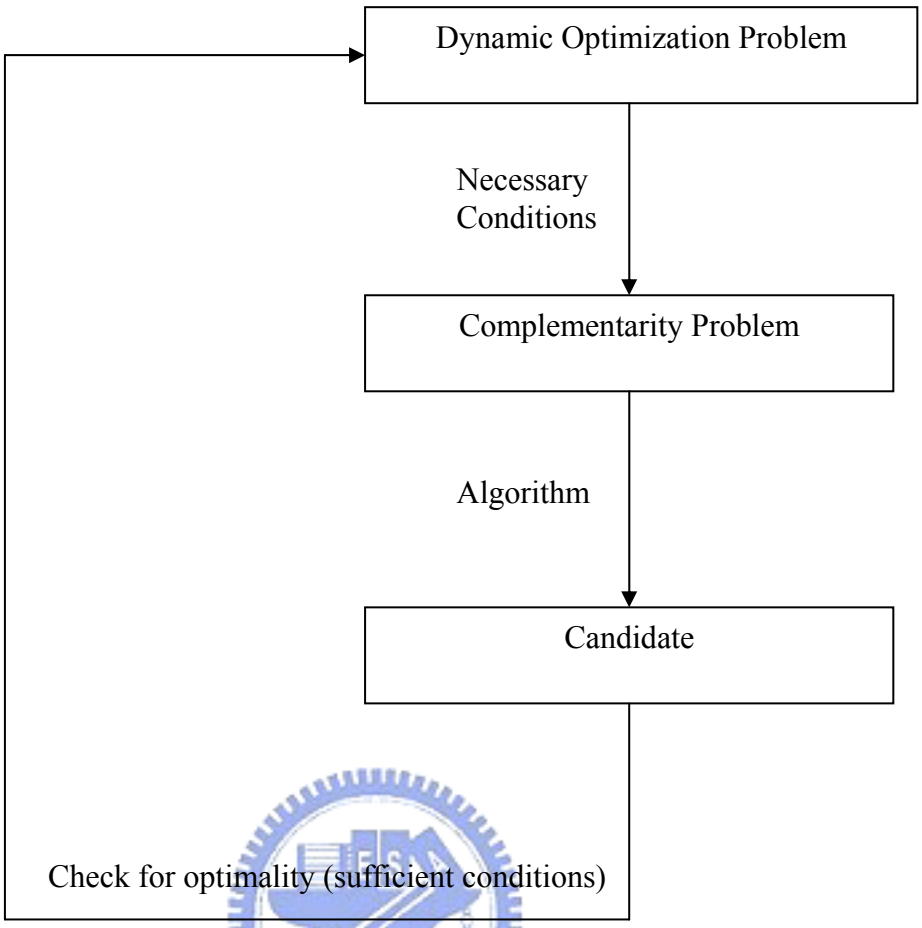


Figure 2.1 Solution process based on indirect methods.

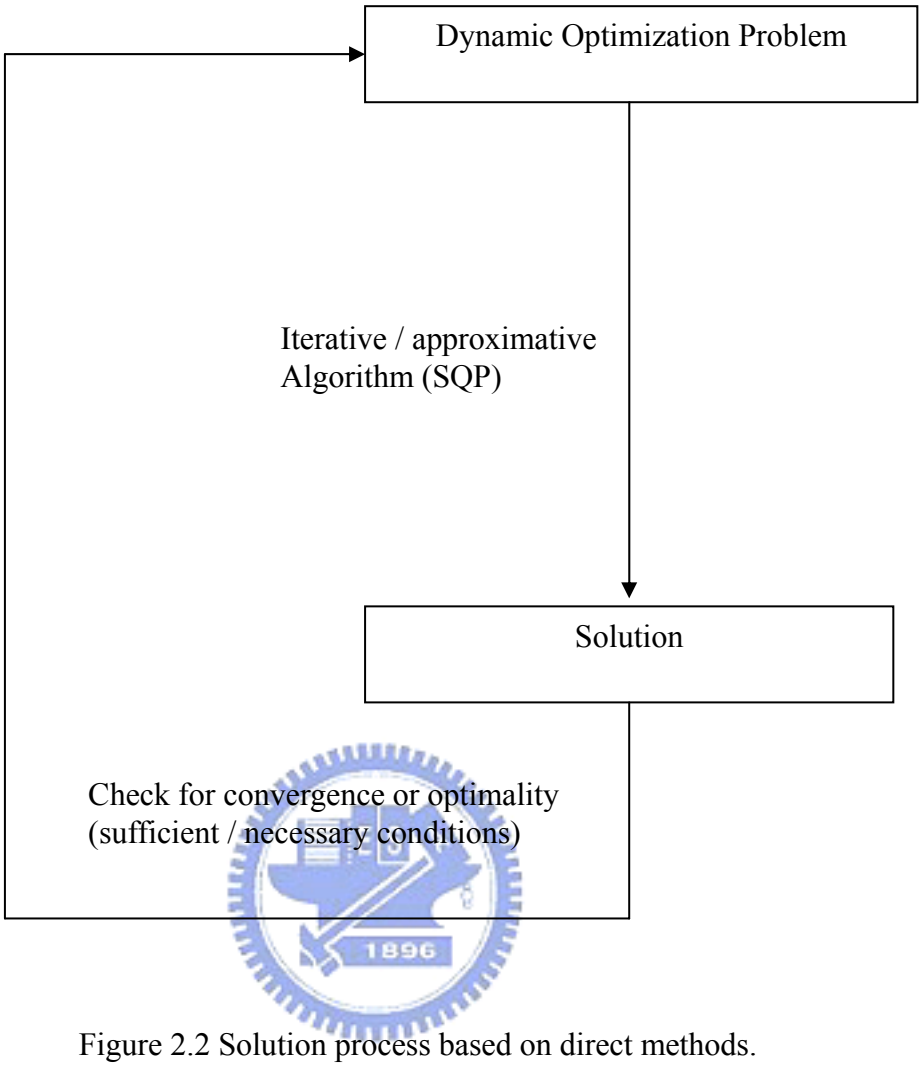


Figure 2.2 Solution process based on direct methods.

CHAPTER 3
COMPUTATIONAL METHODS AND NUMERICAL PRELIMINARIES FOR SOLVING OCP

3.1 Introduction

The rapid advancements in modern computers have brought about a revolution in the solutions to many physical and engineering problems, including optimal control problems. However, most real-world problems are becoming too complex to allow analytical solution; thus, computational methods must inevitably be used in solving them. As a result, computational methodology has attracted the interest of many engineers and mathematicians, and over the last two decades, many state-of-the-art computational methods for optimal control theory – including collocation transcription and the AOCP method – have been developed (see, *e.g.,* Betts, 1998 and 2001; Hu *et al*., 2002; Jaddu and Shimemura, 1999; Lin, 1992; Pytlak, 1999).

Some earlier computational methods for solving optimal control problems were based on the indirect approach that assumes the direct solution of a set of necessary optimality conditions resulting from Pontryagin's maximum principle. The adjoint (co-state) equations are combined with the original state equations to form a TPBVP. This problem may be efficiently solved using the shooting method discussed earlier, which guesses the unknown initial values of the adjoint variables, integrates both system and adjoint equations forward, and then reestimates the initial guesses from residuals at the end point (Bulirsch, 1971; Lastman, 1978). Nevertheless, because of difficulties arising from the sensitivity and instability of the solutions to the initial guesses, Bulirsch and his coworkers (1971, 1980) introduced multiple shooting algorithms to improve convergence and stability. *Multiple shooting* refers to the breaking up of a trajectory into subintervals, on each of which an initial-value problem is defined. The solutions are then adjusted in successive iterations until the boundary conditions and continuity properties at the ends of the subintervals are satisfied.

Multiple shooting is much more successful than its ancestor, the simple shooting method, in which a single initial-value problem is defined. However, even though especially good convergence properties are attributed to multiple-shooting algorithms, the necessity to define the proper control structure and initialize the adjoint variables within a sufficient vicinity of the optimal values still remains a serious limitation.

To avoid the drawbacks of shooting techniques, the direct methods have been studied extensively during the last two decades (Betts, 1993; Barclay, 1997; Gill *et al*., 2002). One of the most widely used methods for solving optimal control problems is the direct method whose basis is the transformation of the optimal control problem into a NLP problem using either the discretization or parameterization technique (see, *e.g.*, Goh and Teo, 1988; Xu and Antsaklis, 2004; Jaddu, 2002; Lee *et al*., 1999).

When the discretization technique is applied, the optimal control problem is converted into a nonlinear programming problem with a large number of unknown parameters and constraints (Betts, 1998). On the other hand, parameterizing the control variables (Goh and Teo, 1988; Teo *et al.*, 1991) requires integration of the system equations. Moreover, the simultaneous parameterization of both the state variables and the control variables also results in a nonlinear programming problem with a large number of parameters and equality constraints.

As a prelude to discussing computational methods for solving optimal control problems, the following sections introduce some fundamental NLP concepts. Also introduced is one of the best and most frequently applied NLP methods for solving optimal control problems, sequential quadratic programming (see Barclay, *et al.*, 1998; Betts, 2000; Gill *et al.*, 2002; Kraft, 1994; Stryk, 1993). Subsequently, the AOCP method, which uses the discretization technique to convert an OCP into a NLP problem, is proposed, and then a standard SQP algorithm is applied to solve it. Also discussed are the dynamic constraint treatments and

design sensitivity analysis used in AOCP.

3.2 Nonlinear Programming Problem

Mathematically, the general form of a constrained NLP problem can be expressed as follows:

minimize

$$f(\boldsymbol{x})$$

subject to (3.1)

$$\boldsymbol{g}(\boldsymbol{x}) \le 0\,, \quad \boldsymbol{x}^{\mathrm{T}} = (x_1, \ldots, x_n)$$

$$\boldsymbol{h}(\boldsymbol{x}) = 0$$

where $f(\boldsymbol{x})$ is the objective function, and $\boldsymbol{h}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$ are the equality and inequality constraint functions, respectively. It should be noted that in the inequality constraint functions $\boldsymbol{g}(\boldsymbol{x})$, the simple bounds of the design variables ($\boldsymbol{x}_L \le \boldsymbol{x} \le \boldsymbol{x}_U$) are considered and classified. Because maximization problems can be converted to minimization ones by negating their objectives, only minimization problems are considered here, without loss of generality.

A general continuous constrained NLP problem is defined in Eq. (3.1) in which $\boldsymbol{x}$ is a vector of continuous variables. Over the past three decades, a variety of methods has been produced in a wide body of research to solve the general constrained continuous optimization problem (Betts, 2001; Michalewicz *et al*., 1996; Horst and Tuy, 1993; Floudas and Pardalos, 1992; Hansen, 1992). Based on different problem formulations, existing methods can be classified into three categories: penalty formulations, direct solutions, and Lagrangian methods. Figure 3.1 classifies these methods according to their formulations, and the details of these methods and their comparisons can be found in Wu (2000). Here, the SQP method based on the Lagrangian method and adopted as an NLP solver in the AOCP algorithm is introduced briefly.

In general, because Lagrangian methods work on equality constraints, inequality

constraints are first transformed into their equal equivalents before Lagrangian methods are applied. For example, an inequality constraint can be transformed into an equality constraint by adding a slack variable (Luenberger, 1984). Thus, a general continuous equality constrained optimization problem can be formulated as follows:

minimize

$$f(\boldsymbol{x}) \tag{3.2}$$

subject to

$$\boldsymbol{h}(\boldsymbol{x}) = [h_1(\mathrm{x}), \ldots, h_\mathrm{m}]^\mathrm{T} = \mathbf{0}$$

where $\boldsymbol{x}^\mathrm{T} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mathrm{n})$ is a vector of the continuous variables. Both $f(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ are assumed to be continuous functions that are at least first-order differentiable. The augmented Lagrangian function in continuous space in Eq. (3.2) is then defined as

$$\mathbf{L}_c(\boldsymbol{x},\boldsymbol{\lambda}) \equiv \nabla_x f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \nabla_x \boldsymbol{h}(\boldsymbol{x}) + \frac{1}{2}\left\|\boldsymbol{h}(\boldsymbol{x})\right\|^2 \tag{3.3}$$

where $\boldsymbol{\lambda}$ is a vector of the Lagrange multipliers. Compared to the conventional Lagrangian function in continuous space defined as $\mathbf{L}_c(\boldsymbol{x},\boldsymbol{\lambda}) \equiv f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{h}(\boldsymbol{x})$, the augmented Lagrangian function reduces the possibility of ill conditioning and is, therefore, more stable.

Various continuous Lagrangian methods have been developed to find the (local) optimum, all based on first-order necessary conditions. To state these conditions, the concept of regular points must first be introduced.

Definition 3.1. A point $\boldsymbol{x}$, which satisfies constraints $\boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}$, is said to be a *regular point* (Luenberger, 1984) if the gradient vectors $\nabla h_1(\boldsymbol{x}), \nabla h_2(\boldsymbol{x}), \ldots, \nabla h_m(\boldsymbol{x})$ at point $\boldsymbol{x}$ are linearly independent.

First-order necessary conditions for continuous constrained NLP problems.

Letting $\boldsymbol{x}$ be a (local) optimal solution of $f(\boldsymbol{x})$ subject to constraints $\boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}$, and assuming that $\boldsymbol{x}$ is a regular point, then there exists $\boldsymbol{\lambda} \in \mathbb{R}^m$ such that

$$\nabla_x f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \nabla_x \boldsymbol{h}(\boldsymbol{x}) = 0 \tag{3.4}$$

Based on the definition of a Lagrangian function, the necessary conditions for $\boldsymbol{x}$ to be a constrained (local) optimal solution can be written as follows:

$$\begin{aligned} \nabla_{\boldsymbol{x}} \mathbf{L}_c(\boldsymbol{x}, \boldsymbol{\lambda}) &= 0 \\ \nabla_{\boldsymbol{\lambda}} \mathbf{L}_c(\boldsymbol{x}, \boldsymbol{\lambda}) &= 0 \end{aligned} \tag{3.5}$$

To ensure that the equilibrium point is an optimal solution, second-order sufficient conditions are used to check that the solution is a strictly relative minimum subject to constraints (Luenberger, 1984). These second-order sufficient conditions require second-order derivatives, and the Hessian matrix of the Lagrangian function is needed to satisfy certain conditions (Luenberger, 1984) if the solution to Eq. (3.5) is to be a strictly (local) optimal solution. Here, The Hessian matrix of the Lagrangian can be defined as

$$\boldsymbol{H}_{\boldsymbol{L}} \equiv \nabla_x \left[\nabla_x \boldsymbol{L}_c(\boldsymbol{x}, \boldsymbol{\lambda}) \right]^T = \nabla_{xx} \boldsymbol{L}_c(\boldsymbol{x}, \boldsymbol{\lambda})$$

. Many searching methods based on first-order necessary conditions in continuous space have been developed for solving constrained optimization problems (Bertsekas, 1982; Luenberger, 1984), including the first-order method, Newton's method, modified Newton's methods, quasi-Newtonian methods, and sequential quadratic programming (Hribar, 1996; Boggs and Tolle, 1995). A major advantage of these methods is that solving the first-order conditions exactly matches the goal of locating a (local) optimal solution. Therefore, these algorithms are usually efficient for solving continuous constrained NLPs. One of most popular methods for solving constrained optimization is sequential quadratic programming, discussed briefly in the next section.

3.3 Sequential Quadratic Programming Method

SQP, one of most popular Lagrangian methods for solving constrained NLPs, is also widely applied to develop computational methods for solving optimal control problems (Betts, 2000; Buskens and Maurer, 2000; Volkwein, 2000; Barclay *et al*., 1997). SQP methods have

proven reliable and efficient for many practical constrained optimization problems. The method described here is implemented in MOST (Tseng *et al*., 1993) and is similar to the algorithm employed by SNOPT (Gill *et al*., 2002) software.

The SQP method is actually a generalization of Newton's method (Luenberger, 1984) for unconstrained optimization in the sense that it obtains search directions from a sequence of quadratic programming (QP) subproblems. Each QP subproblem minimizes a quadratic model of a certain Lagrangian function subject to linearized constraints. In its simplest form, an SQP algorithm replaces $f(\boldsymbol{x})$ in the Lagrangian function with a quadratic approximation and the weighted constraint functions $\boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x})$ with their linear approximations:

$$\begin{aligned} Q(\boldsymbol{d}) &= \nabla f(\boldsymbol{x})^T \boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T \nabla^2_{xx} \mathbf{L}_c(\boldsymbol{x},\boldsymbol{\lambda})\boldsymbol{d} \\ &= \nabla f(\boldsymbol{x})^T \boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T \boldsymbol{H}_L \boldsymbol{d} \end{aligned} \tag{3.6}$$

where $\boldsymbol{d}$ refers to the descent direction (or search direction) and $\boldsymbol{H}_L$ to the Hessian matrix of the Lagrangian function. The descent direction $\boldsymbol{d}^{(k)}$ of the k^{th} iteration of SQP can be found by solving the following quadratic problem, assuming equality constraints only:

minimize

$$Q(\boldsymbol{d}^{(k)})$$

subject to

$$\boldsymbol{h}(\boldsymbol{x}^{(k)}) + \nabla \boldsymbol{h}(\boldsymbol{x}^{(k)})^T \boldsymbol{d}^{(k)} = \boldsymbol{0} \tag{3.7}$$

where $\boldsymbol{x}^k$ is used to represented the values of the design variables of the k^{th} iteration of SQP.

The local convergence property of SQP is well defined when $(\boldsymbol{x}, \boldsymbol{\lambda})$ satisfies the second-order sufficient conditions (Luenberger, 1984). That is, if point $(\boldsymbol{x}, \boldsymbol{\lambda})$ is sufficiently close to an optimal solution $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$, then the sequence generated using the descent direction $\boldsymbol{d}$ and the appropriate step size α will converge to $\boldsymbol{x}^*$ at a second-order rate. The search step size α can be obtained by applying a line search method.

The SQP method described here requires a more precise computation of the Hessian matrix, $\nabla^2_{xx}\mathbf{L}_c\left(\boldsymbol{x}^k,\boldsymbol{\lambda}^k\right)$, at each step. However, it is usually replaced with a BFGS approximation (Arora, 1989) $\boldsymbol{B}_k$ updated at each iteration. Using a BFGS formula allows the following simple update strategy to be defined:

$$\begin{aligned}\boldsymbol{\eta}_k &= \boldsymbol{x}_{k+1}-\boldsymbol{x}_k \\ \boldsymbol{\beta}_k &= \nabla_x\mathbf{L}_c\left(\boldsymbol{x}_{k+1},\boldsymbol{\lambda}_k\right)-\nabla_x\mathbf{L}_c\left(\boldsymbol{x}_k,\boldsymbol{\lambda}_k\right) \\ \boldsymbol{B}_{k+1} &= \boldsymbol{B}_k-\frac{\boldsymbol{B}_k\boldsymbol{\eta}_k\boldsymbol{\eta}_k^T\boldsymbol{B}_k}{\boldsymbol{\eta}_k^T\boldsymbol{B}_k\boldsymbol{\eta}_k}+\frac{\boldsymbol{\beta}_k\boldsymbol{\beta}_k^T}{\boldsymbol{\beta}_k^T\boldsymbol{\eta}_k}\end{aligned} \tag{3.8}$$

Once the descent direction has been determined, the step size must be calculated based on simultaneously decreasing the objective, as well as improving constraint satisfaction. To accomplish this goal, a suitable unconstrained function must be developed upon which to base the step size determination. Many unconstrained optimization methods, such as the golden section search method, can be found in the literature (see Arora, 1989) and applied to calculate the search step size, α.

Nevertheless, although SQP methods are generally efficient, they often require that functions be differentiable and therefore cannot be applied directly to solving NLPs containing discrete variables. Thus, a modified SQP algorithm in cooperation with an enhanced branch-and-bound method is proposed here to solve discrete-valued NLP problems. The details of this modified algorithm are introduced in Chapter 5 of this dissertation. The details of SQP method implementation can be found in the literature (*e.g.*, Arora, 1989; Boggs and Tolle, 1995; Gill *et al.*, 2002). Figure 3.2 presents a conceptual flowchart of the SQP method, whose algorithm is briefly described below.

Algorithm: Sequential Quadratic Programming

Step 1. Choose $\boldsymbol{x}_0$, N_s (maximum number of iteration)

ε_i (for convergence and stopping), $k = 1$ (iteration counter).

Step 2. Find the descent direction $\boldsymbol{d}$ by solving the QP subproblem defined in Eqs. (3.6)

and Eq. (3.7).

Step 3. Check feasible and convergence criteria.

(a) Convergence for SQP:

IF ($h_j = 0, j = 1, \ldots, m$) **THEN**

IF (KT condition is satisfied) **THEN**

Algorithm converged, Stop.

(b) Stopping criteria:

$\Delta \boldsymbol{x} = \boldsymbol{x}^{k+1} - \boldsymbol{x}^{k}$

IF ($\Delta \boldsymbol{x}^{\mathrm{T}} \Delta \boldsymbol{x} \le \varepsilon_i$) **THEN** Stop. (design variable not changing)

IF ($k = N_s$) **THEN** Stop. (maximum iteration reached)

Continue

Step 4. Calculate the step size α.

Step 5. Update Hessian matrix $\boldsymbol{H_L}$ by applying BFGS approximation $\boldsymbol{B}_k$ (Eq. 3.8).

Step 6. $k = k + 1$; Go to Step 2.

3.4 Admissible Optimal Control Problem Method

The admissible optimal control problem method is a direct method that transcribes an optimal control problem into a NLP problem, a process shown in Figure 3.3. Whereas an NLP problem consists of a finite set of variables and constraints, an optimal control problem can involve continuous functions and be treated as an infinite-dimensional extension of an NLP problem. However, most practical methods for solving optimal control problems require Newton-based iterations with a finite set of variables and constraints. Therefore, a discretization technique is needed to convert the infinite-dimensional problem into a finite-dimensional approximation. On the other hand, a general optimal control problem may include some dynamic constraints that make the problem complex and difficult to solve. Thus, an efficient dynamic constraint treatment becomes more important for developing a general

optimal control solver. Presented below are two common design sensitivity analysis (DSA) methods used to determine the effect of a change in the current design on the cost functional and the constraint functions.

3.4.1 Discretization and Parameterization Techniques

Various discretization and parameterization techniques for state and control variables allow for an optimal solution for the OCP via nonlinear programming. Jaddu and Shimemura (1999) used quasi-linearization and state parameterization using Chebyshev polynomials to solve constrained nonlinear optimal control problems. Hu *et al*. (2002) applied an enhanced scheme based on the direct collocation and nonlinear programming problem (DCNLP) to transform the system dynamics into constraints for nonlinear programming. Nevertheless, although these simultaneous discretization methods are applied to many numerical examples and solve them successfully, using a full discretization strategy sharply increases the number of design variables. Therefore, a different discretization technique, in conjunction with SQP, is implemented here, one used to solve various types of optimal control problems (Betts, 2000; Barclay *et al*., 1997). This technique is based on the sequential method in which only the control variables $\boldsymbol{u}$ are approximated by some interpolation function in each time interval. The approximate trajectories $\boldsymbol{x}$ are generated by solving the initial-value problem defined in Eqs. (2.1) and (2.2). This method, first proposed by Sage and White (1977), is termed the AOCP method.

Control function parameterization. Parameterization of the control functions can be carried out using the following process. First, the entire time interval $t \in \left[t_0, t_f \right]$ is subdivided into N general unequal time intervals and the time grids are designated as

$$t_0 = 0,\ t_1,\ t_2, \ldots,\ t_{N-1},\ t_N = t_f \tag{3.9}$$

The time intervals between the grid points are defined in a vector form as

$$\boldsymbol{T} = [\, T_1, T_2, \ldots, T_N \,]^T \tag{3.10}$$

where $T_l = t_l - t_{l-1}$ and $\sum_{l=1}^{N} T_l = t_f - t_0$.

If at each time grid, control $\boldsymbol{u}^{(l)}$ is treated as a set of m unknown parameters, then interval $\left[t_0, t_f\right]$ will have an Nm unknown parameter and can be represented as

$$\begin{aligned} \mathbf{S_D} &= [\, \boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)}, \ldots, \boldsymbol{u}^{(N)} \,]^T \\ &= [\, u_1(t_0), \ldots, u_m(t_0), u_1(t_1), \ldots, u_m(t_1), \ldots, u_1(t_{N-1}), \ldots, u_m(t_{N-1}) \,]^T \\ &= [\, S_1, \ldots, S_m, S_{m+1}, \ldots, S_{2m}, S_{2m+1}, \ldots, S_{(N-1)m+1}, \ldots, S_{mN} \,]^T \end{aligned} \tag{3.11}$$

where $\boldsymbol{u}^{(l)} \in \mathbb{R}^m$ is the vector of the control variables for the l^{th} time interval $[t_l, t_{l+1}]$. This formulation can be treated as a subset of the design variable vector, resulting in a total number of $k+N+Nm$ design variables:

$$\boldsymbol{P} = [\, b_1, \ldots, b_r, T_1, \ldots, T_N, \ldots, S_1, \ldots, S_{N+1}, S_{N+2}, \ldots, S_{mN} \,]^T \tag{3.12}$$

Finding an accurate solution for any practical application requires one set of fine time grid intervals. However, discretizing control functions with a fine time interval increases the number of design variables considerably, especially for a practical optimal control problem with a large number of control variables. Hence, certain parameterization techniques have been developed to overcome this problem. If parameterization techniques are applied, control function $\boldsymbol{u}(t)$ may be represented by an interpolation function, and the coefficients of the interpolation function may be considered design variables instead of T_i and S_i in Eq. (3.12). For example, if the time grid is not considered a design variable, the interpolation function based on a third-order polynomial $\boldsymbol{u}^{(l)} = \zeta_1^{(l)} + \zeta_2^{(l)} \times t + \zeta_3^{(l)} \times t^2 + \zeta_4^{(l)} \times t^3$ can be used to represent the first component of the control forces in $\boldsymbol{u}(t)$ and $\zeta_1^{(l)}$, $\zeta_2^{(l)}$, $\zeta_3^{(l)}$, and $\zeta_4^{(l)}$ can be treated as a subset of the design variables. Therefore, the control functions can be approximated by interpolation functions $\boldsymbol{I}(t)$, where $\boldsymbol{I}(t) : \left[t_0, t_f\right] \in \mathbb{R}^m$. The continuous time

optimal control problem with interpolation functions ***I***(t) is thus reformulated as an NLP problem without using approximate discretization. As noted earlier, the control functions, ***u***(*t*), are treated as a subset of the design variable vector ***P***. Similarly, the terminal time t_f can be treated as one of the design variables in time interval vector ***T***, *e.g.*, $t_f = \sum_{i=1}^{N} T_i + t_0$. The admissible control functions are represented in the form ***u***(*t*) = ***I***(***S, T,*** *t*), and the state variables are written in the form ***x***(***b, S, T,*** *t*) to emphasize that they are functions of the design variable vector ***P***. Here, ***S*** represents the parameter vector of interpolation function. As a result, the admissible optimal control problem in an NLP formulation can be rewritten as follows.

A design variable vector $\boldsymbol{P} = [\boldsymbol{b}^{\mathrm{T}}, \boldsymbol{T}^{\mathrm{T}}, \mathbf{S}^{\mathrm{T}}]^{\mathrm{T}}$ must be found that minimizes the cost functional

$$J_0 = \Phi_0\left(\boldsymbol{b}, \boldsymbol{x}(\boldsymbol{b},\boldsymbol{S},\boldsymbol{T},t_f), t_f\right) + \sum_{l=1}^{N-1}\left[\int_{t_l}^{t_{l+1}} \mathcal{L}_0^{(l)}\left(\boldsymbol{b}, \boldsymbol{I}(\boldsymbol{S},\boldsymbol{T},t), \boldsymbol{x}(\boldsymbol{b},\boldsymbol{S},\boldsymbol{T},t), t\right) dt\right] \tag{3.13}$$

subject to

$$J_i^{(l)} = \Phi_i\left(\boldsymbol{b}, \boldsymbol{x}(\boldsymbol{b},\boldsymbol{S},\boldsymbol{T},t_f), t_f\right) + \int_{t_l}^{t_{l+1}} \mathcal{L}_i^{(l)}\left(\boldsymbol{b}, \boldsymbol{I}(\boldsymbol{S},\boldsymbol{T},t), \boldsymbol{x}(\boldsymbol{b},\boldsymbol{S},\boldsymbol{T},t), t\right) dt \begin{cases} = 0; & i = 1, \ldots\ldots\ldots, N_e \\ \le 0; & i = N_e + 1, \ldots, N_T \end{cases} \tag{3.14}$$

$$\psi_j^{(l)}\left(\boldsymbol{b}, \boldsymbol{I}(\boldsymbol{S},\boldsymbol{T},t), \boldsymbol{x}(\boldsymbol{b},\boldsymbol{S},\boldsymbol{T},t), t\right) \le 0 \; ; j = 1, \ldots\ldots\ldots, q; \; \forall t \in \left[t_l, t_{l+1}\right]. \tag{3.15}$$

and the system equation is represented as

$$\dot{\boldsymbol{x}}^{(l)} = \boldsymbol{f}\left(t, \boldsymbol{b}, \boldsymbol{x}(\boldsymbol{b},\boldsymbol{S},\boldsymbol{T},t), \boldsymbol{I}(\boldsymbol{S},\boldsymbol{T},t)\right), \; t \in \left[t_l, t_{l+1}\right] \tag{3.16}$$

with the initial condition

$$\boldsymbol{x}^{(l)}(t_l) = \boldsymbol{x}_l; \; \boldsymbol{x}^{(0)}(t_0) = \boldsymbol{x}_0 \tag{3.17}$$

where *l* is used to indicate the index of the time grid, and the original optimal control problem is divided into *N* subproblems. These NLP subproblems are solved sequentially from time

grid t_0 to terminal time t_f, and the solutions of the state variables obtained in each time interval are then applied as initial values in the next time interval.

With AOCP, the system equation in Eq. (3.16), together with the initial conditions in Eq. (3.17), forms an initial-value problem (IVP), and the corresponding values for the state variables can be calculated by solving the problem using the design variable values in each iteration. For integrating the state equations in Eqs. (3.16) and (3.17), some good first-order differential equation methods are available that have a variable step size and error control, *e.g.*, Adam's method, the Runge-Kutta-Fehlberg method, and the backward difference formulae (BDF) (Press *et al*., 1992). Those solvers can give accurate results with user-defined error control. The state trajectories are internally approximated using interpolation functions in the differential equation solvers. Values of the state and control variables between the grid points can be also obtained with different types of interpolation schemes.

3.4.2 Dynamic Constraint Treatments

The continuum dynamic constraints in Eq. (2.5) must be satisfied over the entire time interval at the optimum solution. Some procedures to eliminate time from these constraints are employed to convert an admissible optimal control problem into an NLP problem. The many treatments proposed to deal with the dynamic constraint problem – *e.g.*, the equivalent functional formulation (Haug and Arora, 1979) and worst-case design formulation (Hsieh and Arora, 1984) – are introduced below.

(a) Conventional Formulation

With conventional formulation, discretization of the entire time interval $[t_0, t_f]$ in equation (3.9) into N subintervals can be carried out by fixing a time grid at the current design iteration. Conceptually, the simplest way of replacing the continuum dynamic constraints of equation (3.15) is to impose the constraint at all grid points. This approach is hereafter referred to as the conventional formulation.

(b) Worst Case Design Formulation

Here, the dynamic constraints are treated as a worst-case design formulation (Hsieh and Arora, 1984). Each continuum constraint of equation (3.15) is replaced by constraints at the worst-response time points:

$$\phi_\beta[\boldsymbol{b}, \boldsymbol{I}(\boldsymbol{S},\boldsymbol{T},t), \boldsymbol{x}(\boldsymbol{b},\boldsymbol{S},\boldsymbol{T},t), t]\Big|_{t=t_j} \le 0;\ \ j = 1,2,...,m(\beta) \tag{3.18}$$

where t_j is a point of local maximum for the function ϕ_β, and $m(\beta)$ is the total number of max points for the β^{th} constraint. A numerical procedure is then used to locate all points t_j for the constraint in a given design and impose the constraint thereon.

(c) Subdomain Functional Formulation

For this formulation, the dynamic constraints are transformed into several equivalent functional constraints by dividing the entire time domain into several subdomains, each containing one local maximum point (Hsieh and Arora, 1984). Thus, the dynamic constraints are replaced by the following constraints:

$$\int_{t_{1j}}^{t_{2j}} \phi_\beta\left[\boldsymbol{b}, \boldsymbol{I}(\boldsymbol{S},\boldsymbol{T},t), \boldsymbol{x}(\boldsymbol{b},\boldsymbol{S},\boldsymbol{T},t), t\right] dt \le 0,\ \ j = 1,2,...,m(\beta) \tag{3.19}$$

where $[t_{1j}, t_{2j}]$ is a small subdomain around a local maximum point t_j for the constraint function ϕ_β.

(d) Equivalent Functional Formulation

Here, the dynamic constraints can be transformed into an equivalent functional form by integrating them over the time interval (Haug and Arora, 1979) as

$$\int_{t_0}^{t_f} \phi_\beta\left[\boldsymbol{b}, \boldsymbol{I}(\boldsymbol{S},\boldsymbol{T},t), \boldsymbol{x}(\boldsymbol{b},\boldsymbol{S},\boldsymbol{T},t), t\right] dt \le 0 \tag{3.20}$$

where

$$\int_{t_0}^{t_f} \phi_\beta\left[\boldsymbol{b}, \boldsymbol{I}(\boldsymbol{S},\boldsymbol{T},t), \boldsymbol{x}(\boldsymbol{b},\boldsymbol{S},\boldsymbol{T},t), t\right] dt \begin{cases} = 0, \ if\ \phi_\beta < 0; \\ = \phi_\beta, \ if\ \phi_\beta \ge 0; \end{cases} \tag{3.21}$$

at any time.

In this dissertation, the conventional formulation, the worst-case design formulation, and the hybrid treatment for conventional and worst-case approach (shown in Figure 3.4) are applied to deal with the dynamic constraint problem.

3.4.3 Design Sensitivity Analysis

It is important that a numerical method of optimization determine the effect of a change in the current design on the performance index and constraint functions. In other words, the gradients of the performance index and constraint functions with respect to design variables must be evaluated using what is generally referred to as design sensitivity analysis (DSA). The design sensitivity coefficients may be used directly in gradient-based iterative optimization algorithms. Two methods for computing these gradients are the direct differentiation method (DDM) and the adjoint variable method (AVM).

In the DDM, the first variation of the state equation is performed and then the forward numerical integration scheme is used to solve the initial-value problem. The result is substituted directly into the variation of the functional constraint or dynamic constraints with respect to the design variables. The AVM transposes an adjoint vector and then solves the terminal value problem using a backward numerical integration scheme. The result is then used to solve the sensitivity coefficients. The details and implementation of these two sensitivity analysis methods are discussed in Tseng (1987).

3.4.4 ODE Solvers for Solving Initial Value Problem

The dynamic behaviors for most optimal control problems are determined by a system of ordinary differential equations with a given initial state. Because this system forms an IVP, the numerical solution of the IVP for ordinary differential equations (ODEs) is fundamental to most optimal control methods.

Finding accurate and efficient solution procedures for solving ODEs has long been a

problem of importance. However, in many practical situations, an analytical solution is either impossible to find or extremely difficult to evaluate. Therefore, numerical solution procedures for approximating solutions have become increasingly popular. Most numerical schemes for solving ODEs can be classified as either one-step or multistep methods; however, one-step methods like Euler's are seldom used in practical situations because of their poor accuracy. Rather, multistep methods – the most popular being the fourth-order Runge-Kutta method, the backward differentiation method, and the Adams-Bashforth method – are most commonly used to solve ODEs because of their high efficiency and accuracy.

Many well-developed packages or subroutines exist for solving differential equations, incuding IMSL, Maple, Mathematica, and MATLAB. Most of these are based on the Runge-Kutta method, the Adams formula, or the backward-differentiation formula. For this dissertation, DDERKF, DDEABM, and DDEBDF, all developed by Sandia Laboratory, were selected for the integration of state or design sensitivity equations. DDEBDF is based on the variable-order (1–5) backward-differentiation formula, DDERKF is a fifth-order Runge-Kutta code, and DDEABM is a variable-order (1–12) Adams-Bashforth code. These equation solvers use variable-step-size algorithms and have good error control. The DDERKF and DDEABM can be used to solve nonstiff and mildly stiff differential equations, while DDEBDF is suitable for stiff equations. If the differential equation is very stiff, DDEBDF is more efficient than DDERKF and DDEABM. In contrast, DDEBDF is far less efficient than DDERKF and DDEABM for nonstiff equations. Since it is not known a priori whether the differential equations are stiff, DDERKF and DDEABM may not be converged. Therefore, DDEBDF must be used. To handle this situation, a subroutine has been developed that controls the use of differential equations solvers. This subroutine first uses DDEABM and then, if the intermediate output shows the problem to be stiff, a switch is made to DDEDBF. With this implementation, the differential equation solvers can be used more reliably and

efficiently.

For most optimal control problems, their dynamical behaviors are determined by a system of ordinary differential equations with a given initial state. It forms an IVP and hence the numerical solution of the IVP for ODEs is fundamental to most optimal control methods.

3.4.5 Numerical Integration Methods

Two common integration schemes, Simpson's rule and Gaussian quadrature, are adopted in this dissertation to integrate the sensitivity coefficients and integral part of the functional constraints into the adjoint variable method (AVM).

3.4.6 Interpolation Functions

For the admissible optimal control formulation, interpolation schemes are needed at several places. First, to integrate the performance index and functional constraints, information between the grid points is needed for a variable-step-size integration rule, *e.g.*, a Gaussian quadrature formula. Secondly, since variable-step-size differential equation solvers are used in this dissertation, values for the state and control variables between the grid points are needed to calculate the right-hand side of the sensitivity equations. Hence, an interpolation scheme is required to obtain the information between grid points. Finally, in the treatment of dynamic constraints, an interpolation scheme is necessary for locating the maximum points for the worst-case formulation or for evaluating the integral for the functional formulation. In the OCP solver developed in this dissertation, zero-order, first-order, and piecewise cubic-spline interpolation functions are adopted.

3.4.7 Computational Algorithm of AOCP

For solving the optimal control problem, the essential idea of AOCP is to treat optimal control problems as initial-value problems by using iterative methods of nonlinear programming. The SQP method is selected to solve the nonlinear programming problems

transcribed from the discretization model of the original optimal control problem. Because SQP is a generalized gradient-descent optimization method and subsequently converges to a local rather than a global optimum, it solves the subproblem by providing both the direction of design improvement and the step size along the search direction. In this dissertation, the algorithms of AOCP and SQP are combined to form a general purpose solver, the OCP solver.

The architectural framework of the OCP solver, as illustrated in Figure 3.5, is composed of two computational blocks: the SQP algorithm and the OCP solver. Because the SQP algorithm is a well-known algorithm for optimization (Arora, 1989; Chong and Zak, 1996; Rao, 1996), its implementation details can be found in a wide body of research and are therefore skipped in this dissertation. Basically, in each iteration of SQP, the values of the design variables are handed over to the OCP solver, which then uses them to calculate the values of the cost functional and the constraints. As shown in Figure 3.5, the OCP solver contains three major computational modules: discretization, calculation of current values of the state variables by applying the ODE solver, and estimation of the values of the cost functional and the constraints. Hence, the AOCP algorithm based on the SQP method can be described as following:

AOCP Algorithm:

Step 1. Choose $\boldsymbol{b}_0$, $\boldsymbol{u}_0$, N_s (maximum number of iteration)

N (number of time intervals)

ε_i (for convergence and stopping), $k = 0$ (iteration counter).

Step 2. Execute the discretization and parameterization process and calculate values for the following variables:

Time intervals $\boldsymbol{T} = [\, T_1, T_2, \ldots, T_N \,]^T$ defined in Eq. (3.10).

Interpolation parameters $\mathbf{S}^{(k)}$ by applying $\boldsymbol{u}_k(\boldsymbol{T}) = \boldsymbol{I}(\boldsymbol{S}^{(k)}, \boldsymbol{T}, t)$.

Design variable vector $\boldsymbol{P}^{(k)} = \left[b_1^{(k)}, \cdots, b_\pi^{(k)}, T_1^{(k)} \cdots, T_N^{(k)}, S_1^{(k)}, \cdots, S_{mN}^{(k)} \right]^T$.

Step 3. Determinate the state variables $\boldsymbol{x}^{(k)}$ by solving the initial-value problem defined in Eqs. (3.16) and (3.17) with $\boldsymbol{P}^{(k)}$.

Step 4. Calculate the values of the cost functional Eq. (3.13) and the constraints Eqs.(3.14) and (3.15).

Step 5. Calculate the gradients of the cost functional and the constraints.

Step 6. Find the descent direction $\boldsymbol{d}$ by solving the QP subproblem defined in Eqs. (3.6) and (3.7).

Step 7. Check feasible and convergence criteria.

(a) Convergence for SQP:

IF (KT condition is satisfied) **THEN**

Algorithm converged, Stop.

(b) Stopping criteria:

$\Delta \boldsymbol{P} = \boldsymbol{P}^{(k+1)} - \boldsymbol{P}^{(k)}$

IF ($\Delta \boldsymbol{P}^{\mathrm{T}} \Delta \boldsymbol{P} \le \varepsilon_i$) **THEN** Stop. (design variable not changing)

IF ($k = N_s$) **THEN** Stop. (maximum iteration reached)

Continue

Step 8. Calculate the step size $\alpha^{(k)}$.

Step 9. Update the Hessian matrix $\boldsymbol{H}^{(k)}$ by applying a BFGS approximation defined in Eq. (3.8).

Step 10. k = k +1; Go to Step 2.

3.5 Summary

This chapter has introduced the numerical preliminaries – including NLP formulation, the SQP method, the control parameterization technique, and dynamic constraint treatments – for developing the OCP solver. Also discussed were methods for solving the NLP and the

first-order necessary condition for continuous constrained NLP problems. The text also introduced the SQP method that serves as the kernel of proposed method, and provided its algorithm. In addition, because this dissertation uses control parameterization with an interpolation function to decrease the numbers of design variables and so make the solver more efficient, discretization and parameterization techniques that transcribe the optimal control problems into NLP problems were developed. Also introduced were several numerical schemes involved in the proposed solver – including ODE solvers and integration and interpolation schemes – and finally, the computational algorithm of the ACOP method that will help implement the OCP solver.

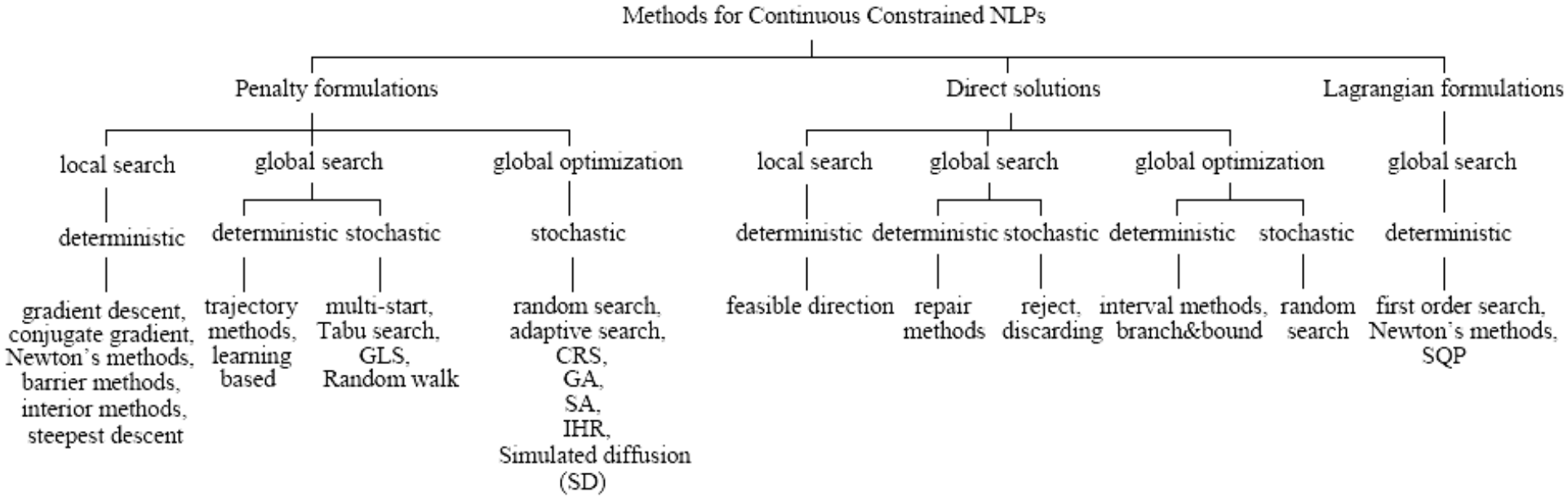


Figure 3.1 Methods for continuous constrained NLPs (Wu, 2000).

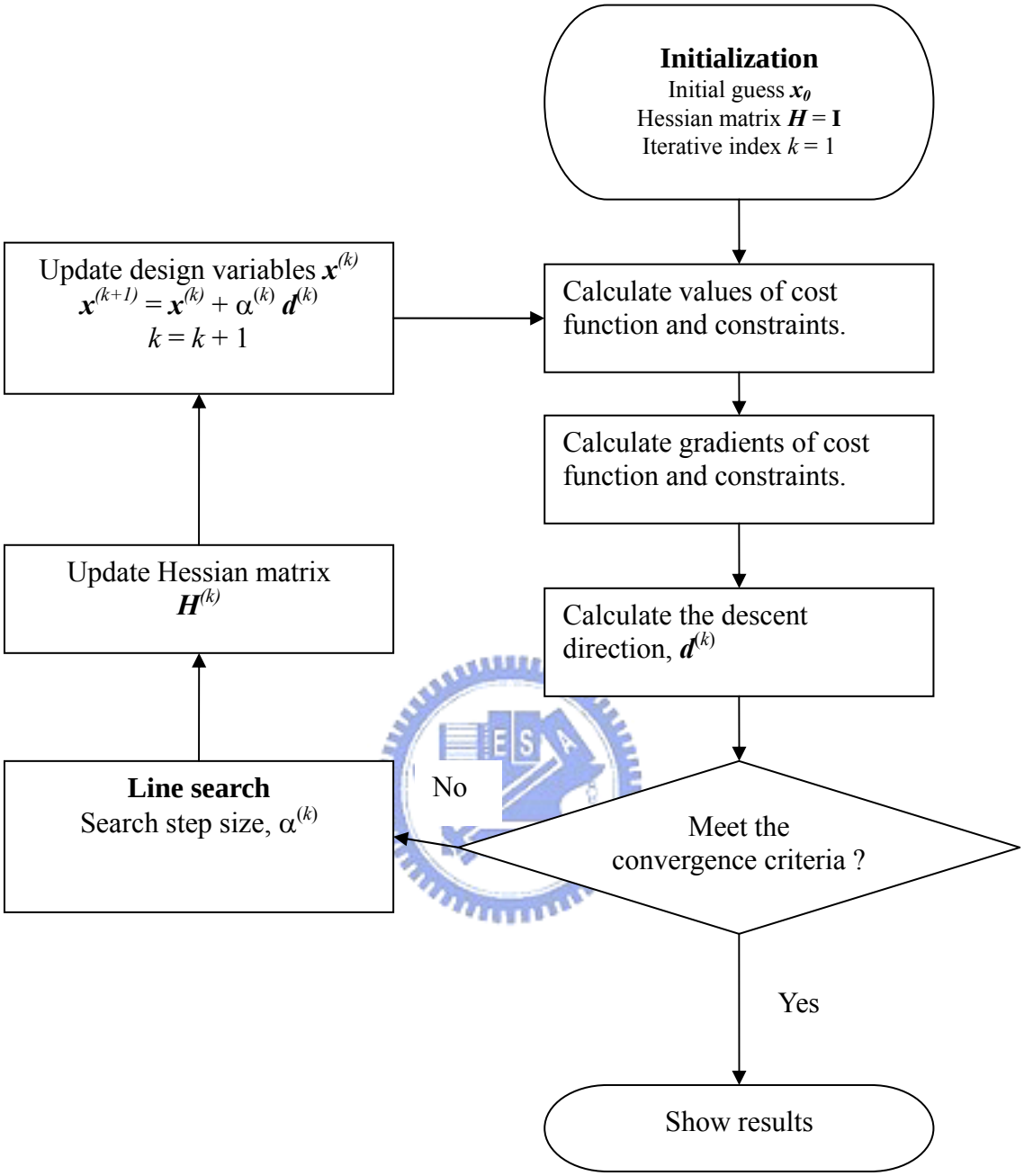


Figure 3.2 Conceptual flowchart of the SQP method.

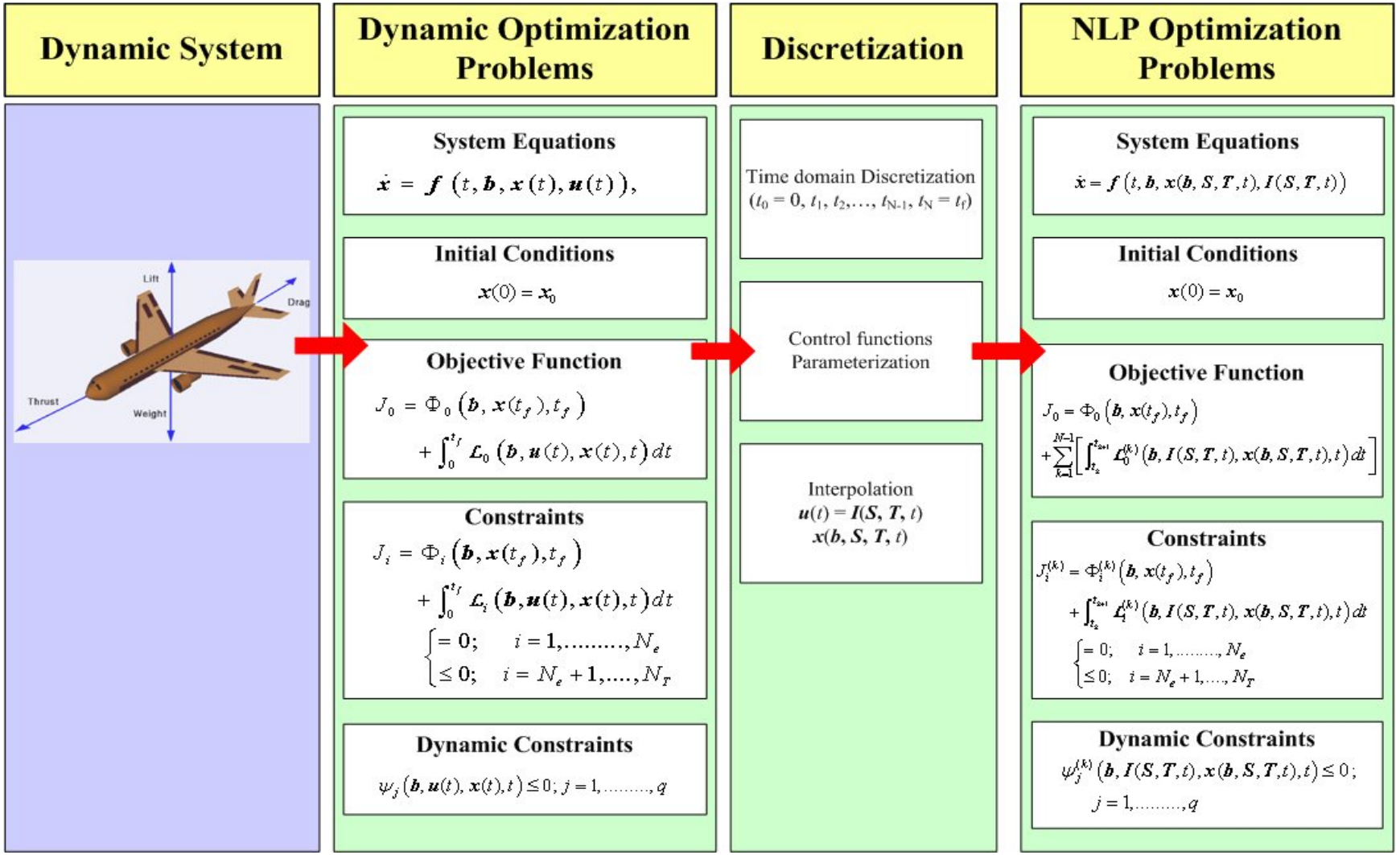
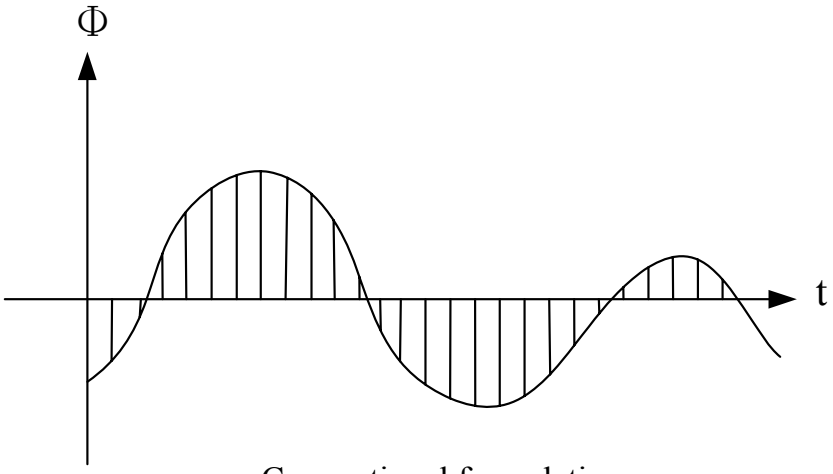
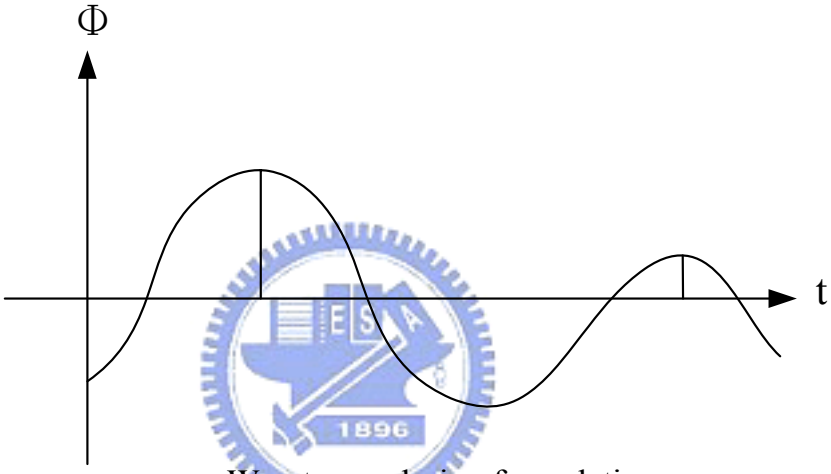


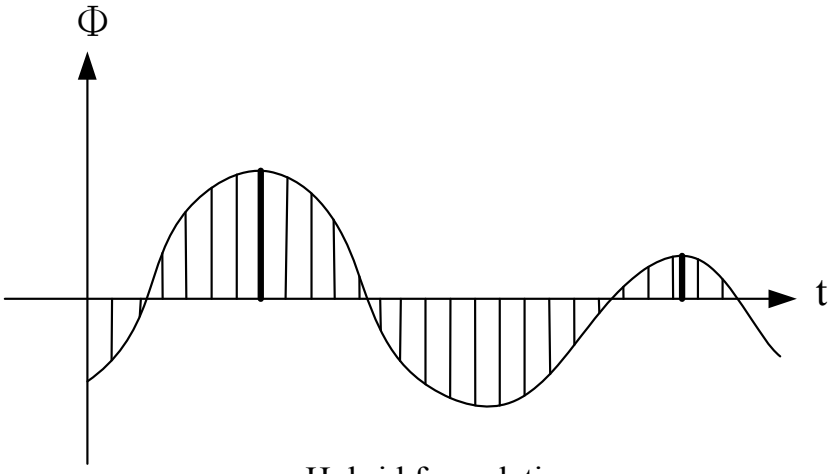
Figure 3.3 Problem-transcribing Process.



Conventional formulation



Worst-case design formulation



Hybrid formulation

Figure 3.4 Dynamic constraint treatments.

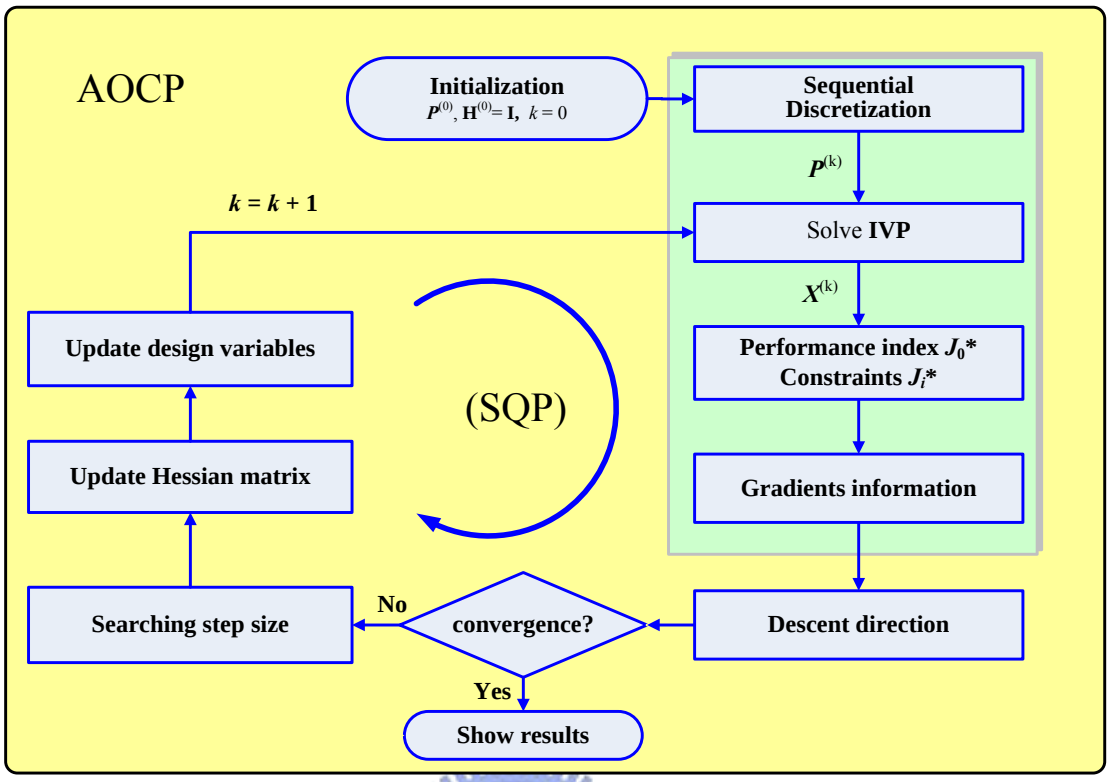


Figure 3.5 Conceptual flowchart of the AOCP method.

CHAPTER 4
A CONVENIENT SOLVER FOR SOLVING OPTIMAL CONTROL PROBLEMS

4.1 Introduction

Even though, over the last two decades, theoretical and numerical methods for solving optimal control problems have been extensively studied and many well-designed algorithms have been proposed, engineers must still expend much effort to reformulate the nonlinear programming problems for different control problems. On the other hand, reworking the corresponding programs for the nonlinear programming problem is a waste of time and even more tedious. Therefore, developing a general OCP solver that offers a systematic process for solving various optimal control problems has become imperative for engineers, particularly for those who are inexperienced in optimal control theory or numerical techniques.

As mentioned in the previous chapter, many well-developed subroutines or program units for numerical analysis are involved in developing a general OCP solver, *e.g.*, integration routines, ODE solvers, interpolation schemes, and general constrained optimization solvers. A general OCP solver, named the OCP solver, was designed that consists of the subsystems or components for such numerical subroutines. Its modular programming features enable the OCP solver to choose different numerical schemes flexibly and be easily upgraded by replacing subroutines with new versions. The implementation details of the kernel module and user interface of the OCP solver are presented in following sections.

4.2 Multifunctional Optimization System Tool - MOST

In the AOCP method, the optimal control problem is converted into an NLP program so that any reliable nonlinear constrained optimization solver can be applied to solve it numerically. A great deal of attention has been paid to using the SQP method to solve NLP problems (Tseng, 1987; Jaddu and Shimemura, 1999). For this dissertation, the

multifunctional optimization system tool MOST (Tseng, 1993), based on the SQP method, has been chosen to solve the NLP problem. This powerful optimization software was developed to solve multi-objective optimization problems with both continuous and discrete design variables (Tseng *et. al*., 1993). The MOST software contains three main modules for dealing with continuous variables, discrete variables, and multi-objective optimization, respectively.

In the primary module, a SQP method (Arora, 2004) is employed to perform the single-objective optimization for problems with continuous design variables. The SQP is selected because of its accuracy, efficiency, and robustness. MOST's accuracy and stability has been tested in the research using 115 test problems with 2 to 96 design variables given by Hock and Schittkowski (1980) The results were satisfactory and also indicated that MOST can handle large-scale engineering optimization with excellent convergence (Tseng *et. al*., 1988a; 1988b). To cope with the discrete-valued optimal control problems arising from discrete design variables, an enhanced branch-and-bound method (Tseng *et. al*., 1995) was integrated into the program. In this module, the original design space of discrete variables is converted into one with continuous variables by dropping the noncontinuous restrictions sequentially. In each of the converted continuous design spaces, the SQP module described above is then utilized to find the optimal values.

Nevertheless, in many engineering applications there frequently exist several mutually conflicting or competing objectives and requirements. Therefore, multi-objective (vector) optimization offers a very promising way to handle such problems. For multi-objective optimization, MOST provides decision makers, goal programming, compromise programming, and the surrogate worth trade-off method (Evans, 1984; Tseng and Lu, 1990) to help users determine the best compromised solutions to nonlinear problems.

It is known that a rigorous formulation of the design problem helps the designer better understand the problem and a proper mathematical formulation leads to a good solution.

MOST provides an input data file that includes the initial design, and it transcribes the design problem by coding subroutines that include evaluation of the cost, in routine *cusermf*, and constraint functions, in routine *cusercf* (see Figure 4.1). Because of its user-defined subroutines, MOST can be extended to flexibly integrate other analysis packages, *e.g.*, ANSYS (Yang *et al*. 1992; Lin *et al*., 1992), EUCLIS-IS (Wang, 1993), and MATHEMATICA (Su, 1994). Moreover, with the assistance of an interface coupler, MOST can deal with the complexity and large size of engineering systems that have no explicit relationships between inputs and system outputs (Huang, 1994). The architecture of the MOST interface coupler is shown in Figure 4.2. The IAOS, a new distributed version of the interface coupler, was developed to deal with analysis packages installed on different machines (Huang, 1994). In IAOS, MOST and the interface coupler are merged into a powerful new optimizer (see Figure 4.3).

4.3 Structure of the Proposed OCP Solver

The kernel of the OCP solver is written in FORTRAN and has been tested on a UNIX platform. Because of the increased popularity and technical developments in the calculation ability of the personal computer (PC), for this dissertation, the OCP solver has been transplanted onto a PC platform. The structured chart for the entire OCP solver is given in Figure 4.4, and the connections between the OCP solver and MOST are shown in Figure 4.9. Four primary independent modules of the OCP solver are introduced below.

CTRLMF module: The structure of this module, given in Figure 4.5, links with MOST's user-defined subroutine *cusermf* and contains six subroutines. The control routine *CTRLMF* is used to calculate the value of the performance index, to partition the array, and to check available memory. The pseudocode for the *CTRLMF* module of the AOCP algorithm is shown in Table 4.1.

CTRLCF module: This module, whose structure is shown in Figure 4.6, calculates the

constraint function values. First, the information on the state trajectory is passed from the *CTRLMF* module and then the values of the functional constraints and dynamic constraints are calculated in routine *CTRLCF*. In this module, subroutine *PTCST* is used to calculate the number and values of dynamic constraints for the alternative treatments described in Section 3.4.2.

CTRLMG module: This module, illustrated in Figure 4.7, calculates the design derivatives of the performance index by using the direct differentiation or adjoint variable method. In the *DDMMG* routine, the design derivatives are calculated using direct differentiation (DDM); in the *AVMMG* routine, by the adjoint variable method (AVM). The *ANSAVM* routine determines the terminal conditions and integrates the adjoint differential equation using a backward numerical integration scheme.

CTRLCG module: This module, outlined in Figure 4.8, is in charge of design sensitivity analysis, which determines the effect of a change in the current design on the performance index and constraint functions. Two common methods, DDM and AVM, described in Section 3.4.3, are implemented in this module. The control routine *DDMCG* is evaluated to obtain the design derivatives using DDM. The information for state variable derivatives with respect to the design variables is then passed from the *CTRLMG* module. If the AVM is needed to calculate the design derivatives, the *AVMCG* routine is executed.

In addition to these four modules, many useful routines are shared by a variety of modules, *e.g.*, *DIFSOL*, *INGSOL*, *DDERKF*, *DDEABM*, *DDEBDF*, *SIMPSN*, *GAUSS*, *TBFIT*, *TGVAL*, and *CTRLF*. The routine *DIFSOL* contains three differential equation solvers (RKF, ABM and DBF), and the routine *INGSOL* is used to calculate the value of an integral. In *INGSOL*, the *SIMPSN* (bases on Simpson's rule) and *GAUSS* (following the Gaussian quadrature formula) are called on to evaluate the integral. Both the *DDERKF* and the *DDEABM* can integrate a system of first-order differential equations, the first using the

Runge-Kutta-Fehlberg method and the second using the Adams-Bashforth-Moulton predicator-corrector formulas for orders one through. The latter implements a backward differentiation formula in the routine. The *TBFIT* is used to calculate the coefficients of the interpolation function and the *TGVAL*, to calculate the value of functions and their derivatives.

4.4 The OCP Solver in Cooperation with MOST

Because of the extension flexibility of MOST, the OCP solver is herein treated as a MOST module. The linkage of MOST to the OCP solver is composed of four user-defined MOST subroutines: *cusermf*, *cusermg*, *cusercf* and *cusercg*. The OCP solver also has four user-defined subroutines for defining an optimal control problem: *USERMF*, *USERMG*, *USERCF* and *USERCG*. The architecture for MOST and the OCP solver is illustrated in Figure 4.9.

In the OCP solver, the subroutine *USERMF* provides the cost functional value (performance index value); subroutine *USERCF* provides the constraint function values; and subroutines *USERMG* and *USERCG* provide the cost function gradient and the gradients of the active constraints, respectively. A fifth subroutine, *USEROU*, can be developed by the user to perform a subsequent optimality analysis for the optimal solution and obtain more output. If the analytic expressions for the gradients in *USERMG* and *USERCG* are not available, MOST provides the option of calculating gradients using a finite difference method that can be specified as forward, backward, or central. More details are provided in the MOST 1.1 User's Manual (Tseng *et al*., 1993). As the architecture given in Figure 4.9 shows, connections exist between the optimizer MOST, the five user subroutines, and the four independent modules, *CTRLMF*, *CTRLMG*, *CTRLCF* and *CTRLCG*, for the OCP solver. The four modules contain the performance index, the functional and dynamic constraints, the gradient of the performance index, and the constraint function gradients for the optimal control problems, respectively. They can be connected to each other to form a general

constrained optimization solver.

4.5 User Interface for the OCP Solver

The user interface for the OCP solver consists of two parameter files and four subroutines. Users can specify the optimization parameters and numerical schemes in the parameter files of both MOST and the OCP solver, such as acceptable violation of constraints for feasible designs, the differential equation solver, the integration rules, and the interpolation scheme. The user interfaces for the OCP solver and MOST are shown in Figure 4.10.

The MOST optimal parameter file, shown in Table 4.2, is used to configure the optimizer parameters (details can again be found in the MOST user manual [Tseng *et. al.*, 1993]). Details of the OCP parameter file *fort.11* – which contains information on the number of equations, grid points, equality and inequality functional constraints, and the parameters for numerical schemes – can be found in Tseng (1987). Table 3 gives the OCP parameter file for the *van der Pol* oscillator problem to be introduced in Section 4.7.1. Once all four necessary user-defined subroutines, *FFN*, *GFN*, *HFN*, and *Z0FN,* are ready, they should be linked to the OCP through the MOST kernel. Figure 4.10 shows the relationship between the user-defined routines and the MOST modules. *FFN* evaluates the integral terms of the performance index or functional constraints, while *GFN* calculates the values of the first term of the performance index or functional constraints and the dynamic constraints. *HFN* is used to evaluates the system state equation $\boldsymbol{f(b}$, $\boldsymbol{u}(t)$, $\boldsymbol{x}(t),t)$, and finally, *Z0FN* is responsible for calculating the initial values of the state variables, $\boldsymbol{x}(t_0)$. Figure 4.11 gives a flowchart for the OCP solver.

4.6 Systematic Procedure for Solving the OCP

In this dissertation, the OCP is converted into an NLP problem using an admissible optimal control problem formulation then the optimizer based on the SQP method is used to solve the NLP problem numerically. These procedures can now be directly implemented, and

the complicated details of transformation and programming automatically completed, in the proposed OCP solver. Because the optimal control and state trajectories are obtained and recorded in the output files, engineers can follow an efficient and systematic procedure to solve various optimal control problems. The procedure for solving the OCP with the OCP solver is as follows:

1) Defining the OCP problem following the formulation defined in Section 2.2.
2) Preparing the parameter files and user-defined subroutines according to the formulation.
3) Compiling the user's subroutines and linking with the OCP solver.
4) Executing the OCP solver and obtaining the optimal results.

4.7 Illustrative Examples

Two types of optimal control problems mentioned in the literature have been used as test problems to evaluate the performance of the proposed method. In the AOCP method, both the acceptable violation of constraints for feasible designs and the acceptable tolerance for the convergence parameter are 10^{-3}. The numerical results for all example problems were obtained on a Pentium 4 Celeron 1.2 GHz computer with 384 MB of RAM.

4.7.1 The *van der Pol* Oscillator Problem

The *van der Pol* oscillator problem was given and solved by Bullock and Franklin (1967) using a second variation method. The problem was also used by Jaddu and Shimemura (1999) to verify their computational method. In this dissertation, it is further used to evaluate the performance and capabilities of the proposed method and the OCP solver. The *van der Pol* oscillator problem can be formulated by the following minimization

$$J_0 = \frac{1}{2}\int_0^5 (x_1^2 + x_2^2 + u^2)\,dt \tag{4.1}$$

subject to

$$\begin{aligned} \dot{x}_1 &= x_2 \ , \\ \dot{x}_2 &= -x_1 + (1 - x_1^2)x_2 + u. \end{aligned} \tag{4.2}$$

with initial states $\boldsymbol{x}^{\mathrm{T}}(\mathbf{0}) = [1, 0]^{\mathrm{T}}$.

Based on this problem, Jaddu and Shimemura considered three cases that can also be solved by the OCP solver: the unconstrained problem, the terminal state constrained problem, and the terminal states and control constrained problem.

Case I: Free end point and no control constraints

The optimal solution for this problem found by Bullock and Franklin (1967) using a second variation method was $J_0^* = 1.433508$, while that found by Jaddu and Shimemura (1999) using a ninth-order Chebyshev series to approximate $x_1(t)$ was $J_0^* = 1.4334872$. Using the OCP solver, in which the control variable $\boldsymbol{u}$ is discretized into 21 grid points, the optimal value is $J_0^* = 1.4334723$, smaller than both earlier reported results. The numerical parameters for MOST are listed in Table 4.2, and the optimal control and state trajectories are shown in Figure 4.12.

Case II: Terminal state constraint

$$\varphi\left(\boldsymbol{x}\left(t_f\right)\right) = 1 - x_2\left(t_f\right) + x_1\left(t_f\right) = 0 \tag{4.3}$$

For this problem, Bullock and Franklin (1967), again using the second variation method, found an optimal value of $J_0^* = 1.6905756$, while Jaddu and Shimemura (1999), also using a ninth-order Chebyshev series to approximate $x_1(t)$, found an optimal value of $J_0^* = 1.6857113$. In this study, the terminal state constraint is treated as an equality constraint and the other number parameters, the same as in case I. With the OCP package, the value obtained is $J_0^* = 1.6856957$. Figure 4.13 shows the optimal control and state trajectories for the proposed OCP solver.

Case III: Terminal state constraints and saturation constraints on control

The terminal state constraints and the saturation constraints on control are described in the following equation:

$$\begin{aligned} &\psi_1 = x_1\left(t_f\right) + 1 = 0, \\ &\psi_2 = x_2\left(t_f\right) = 0, \\ &and \;\; \psi_3 = u\left(t\right) - 0.75 \le 0. \end{aligned} \tag{4.4}$$

When Bashein and Enns (1972) solved the problem, they obtained J_0^* = 2.1439039, while Jaddu and Shimemura (1999), this time using a twelfth-order Chebyshev series to approximate $x_1(\tau)$, found an optimal value of J_0^* = 2.1443893. The solution produced by the OCP solver is an optimal value of J_0^* = 2.1375360. The optimal control and state trajectories for the OCP solver are shown in Figure 4.14.

4.7.2 Time-optimal Control Problem: Overhead Crane System

Overhead cranes are widely used in factories and workplaces to transport objects. An overhead crane system, like that sketched in Figure 4.15, is a high-order nonlinear system that consists of a cart with a point load suspended by cables. The control problem is to transfer the load from an arbitrary point A to point B in minimal time subject to the requirement of zero residual vibration at point B. The control inputs are the horizontal acceleration of the cart and the hoisting acceleration of the cable. Hu *et al.* (2002) proposed this problem and solved it using an enhanced DCNLP method. In this dissertation, this problem will be used to demonstrate the ability of the proposed method to solve a high-order time-optimal control problem.

Given $x_1 = z,\ x_2 = \dot{z},\ x_3 = \theta,\ x_4 = \dot{\theta} = \omega,\ x_5 = l,\ x_6 = \dot{l}$ as the state variables, and $u_1 = \ddot{z}$, $u_2 = \ddot{l}$ as the control inputs, the OCP formulation of the overhead crane system can be minimized as follows

$$J_0 = t_f \tag{4.5}$$

subject to

$$\begin{aligned}
&\dot{x}_1 = x_2 \,, \\
&\dot{x}_2 = u_1 \,, \\
&\dot{x}_3 = x_4 \,, \\
&\dot{x}_4 = (u_1 \cos x_3 - 2x_4 x_6 - g \sin x_3)/x_5 \,, \\
&\dot{x}_5 = x_6 \,, \\
&and \quad \dot{x}_6 = u_2 .
\end{aligned} \tag{4.6}$$

with initial conditions $\boldsymbol{x}^{\mathrm{T}}(t_0) = [0, 0, 0, 0, 4, 0]^{\mathrm{T}}$ and terminal conditions $\boldsymbol{x}^{\mathrm{T}}(t_f) = [10, 0, 0, 0, 4, 0]^{\mathrm{T}}$ where g is the gravitational acceleration.

The state and control constraints are as follows:

$$0 \le x_2(t) \le 1, \qquad |x_6(t)| \le 1, \qquad for \qquad t_0 \le t \le t_f \tag{4.7}$$

$$|u_i(t)| \le 0.5, \quad i = 1,2 \qquad for \qquad t_0 \le t \le t_f \tag{4.8}$$

Using the admissible control formulation delineated in the previous chapter, the control variables are converted into design variables that can then be treated as design variable boundaries. Furthermore, the state constraints are transferred into standard constraint form as follows:

$$\begin{aligned}
&\psi_1 = -x_2(t) \le 0 \,, \\
&\psi_2 = x_2(t) - 1 \le 0 \,, \\
&\psi_3 = -x_6(t) - 1 \le 0 \,, \\
&and \quad \psi_4 = x_6(t) - 1 \le 0 .
\end{aligned} \tag{4.9}$$

In this problem, both the state and the control variables are divided into 101 grid points. The minimum time $J^* = t_f = 12.0004$ is solved in the OCP solver by applying a cubic piecewise interpolation scheme to the control function. Two local optimal solutions are obtained by the OCP solver with different initial points. The trajectories of the rope angle and angular velocity are shown in Figure 4.16, in which the solid line represents the results obtained by the DCNLP method (Hu. *et al*., 2002) and the dashed line represents a second

optimal solution obtained by the OCP solver. As the figure illustrates, the solid line totally matches the results obtained by Hu *et al.* (2002), meaning that one of the optimal solutions found by the OCP solver tallies exactly with the trajectories obtained by the DCNLP method (Hu. *et al*., 2002). In addition, the performance index (terminal time, t_f) obtained by the OCP solver ($t_f \cong 12.00$) is very close to the result using the DCNLP method ($t_f \cong 12.00$). Moreover, according to the trajectories shown in Figure 4.16, the amplitudes of rope angle and angular velocity for the second optimal solution obtained by the OCP solver, as represented by the dashed line, are smaller than the others. Figures 4.17 and Figure 4.18 depict the corresponding inputs and states with local optimal solutions, respectively. In Figure 4.17, the trajectories of the control inputs conform to the dynamic control constraints given in Eq. (4.8). According to the state trajectories in Figure 4.18, the initial conditions, $\boldsymbol{x}^T(t_0) = [0, 0, 0, 0, 4, 0]^T$,and the terminal conditions, $\boldsymbol{x}^T(t_f) = [10, 0, 0, 0, 4, 0]^T$ are satisfied. Obviously, all constraints are fulfilled, thereby proving the correctness of the solutions. In other word, both solutions solved by the OCP solver are local optimal solutions. In practice, small amplitudes of rope angle and angular velocity for an overhead crane will be adopted because they benefit operational safety.

As the numerical results show, both examples convert successfully into NLP problems using the admissible control formulation and can then be solved using the AOCP method. Moreover, the results of the numerical schemes of the proposed method are quite accurate. With the OCP solver, users need not spend a vast amount of effort on programming to obtain solutions. Rather, once the problems are formulated, the solver can be implemented and the problems solved easily. In addition, rapidly advancing computer capabilities will ensure that computing time for the OCP solver will decrease. Thus, users will be able to obtain optimal results more quickly than before.

4.8 Numerical Study

To investigate how the numerical schemes affect the validity of the solution and computational efficiency, both the *van der Pol* oscillator problem and the overhead crane control problem are solved again using different numerical schemes and time intervals that introduced in the previous chapter. Here, the finite-difference method (DSA=FDM) and the DDM for sensitivity analysis are selected to evaluate their performance. Simpson's rule (INTG=SIMPSN) and the Gaussian quadrature formula (INTG=GAUSS) are used to carry out the numerical integration over the time interval. DDERKF and DDEABM with the option to switch to DDEBDF are selected for solving first-order differential equations with a relative scalar error of 1.0E-8 and a scalar error of 1.0E-10. Three common interpolation schemes – zero-order (INTP=Zero), first-order (INTP=First), and piecewise cubic-spline (INTP=Cubic) – are chosen to estimate their effects upon the performances.

Table 4.4, which gives data types collected for the 24 cases of the case I *van der Pol* oscillator problem, shows the total number of iterations (NIT) of the AOCP method, the optimal value of the performance index (J_0^*), and the convergence parameter (Conv. Par.) and CPU time for the entire iterative process. The lower part of the table includes the mean values for different conditions. In contrast, in the case III *van der Pol* oscillator problem, a dynamic constraint is imposed that can be used to compare the dynamic treatments. The conventional design (DTC=ALL), worst-case design (DTC=MAX), and hybrid design (DTC=HYB) for dynamic constraints are selected to evaluate their performance. In this case, two different time grid points are chosen to compare the effects on performance of either a coarse or fine mesh. The comparison among these dynamic treatments with different mesh points is list in Table 4.5. These results indicate that the worst-case design needs more iterations and CPU time to converge. In addition, the optimum performance index with a fine mesh is more accurate than that with a coarse mesh. Similarly, the overhead crane control problem can be solved using

different numerical schemes (the results are shown in Table 4.6).

Overall, the results in Table 4.4 and Table 4.6 show the performance of all numerical schemes to be quite accurate. However, the design sensitivity analysis using the finite-difference method (FDM), the differential equation solver with integration using a fifth-order Runge-Kutta algorithm (DDERKF), the Gaussian quadrature formula (GAUSS), and the interpolation scheme with a first-order interpolation function give better performance with respect to efficiency. From the Table 4.5 data, it is obvious that the fine mesh increases computational cost, while dynamic constraint treatment using the conventional design treatment (ALL) gives more efficient performance than the worst-case design treatment (MAX).

4.9 Summary

This chapter has presented the major modules for the general optimal control problem solver being developed, namely, the OCP solver. Discussed first was the multifunctional optimization solver, MOST, which is used as the kernel of the proposed solver. To develop the OCP solver, use of the interface coupler that integrates the MOST optimizer with other analyzers via standard input/output files has been extended. In addition, the AOCP algorithm has been implemented as an external analyzer module and involved in the OCP solver. The discussion also presented the implementation details of the interface between MOST and the AOCP module, as well as the user interface for the OCP solver.

The primary purpose of this dissertation is to present a systematic procedure for solving optimal control problems with the OCP solver provided. To this end, the *van der Pol* oscillator problem with various terminal states and control constraints and the overhead crane control problem, a high-order nonlinear time-optimal control problem, were used to evaluate the capability and accuracy of the OCP solver. A performance comparison among different numerical schemes involved in the OCP solver was also carried out. The results indicate that

the proposed OCP solver can truly facilitate the solving of engineering control problems in a systematic and efficient way.

Table 4.1 Pseudo-code for the CTRLMF module of the AOCP algorithm.

CTRLMF(***P***, J_0, *nv*, *nobj*, *itr_k*, *ierr*)

BEGIN

Assign the values of the discretized control variables from design variable vector ***P***.

Calculate the values of control variables at any time in [t_0, t_f] by applying interpolation schemes.

Substitute the values of control variables into the system equations Eq. (3.16) so the system equations with the initial conditions in Eq. (3.17) form an initial value problem (IVP).

IF the IVP is non-stiff **THEN**

Calculate the values of the state variables by solving the IVP with the Rung-Kutta-Fehlberg (RKF) method.

ELSE (the IVP is stiff**)**

Calculate the values of the state variables by solving the IVP with the backward differentiation formulas (BDF) method.

ENDIF

Calculate the state of the control variables at any time in [t_0, t_f] by applying interpolation schemes.

Calculate the values of the performance indexes J_0 and return those values.

END

Table 4.2 MOST input file for the *van der Pol* oscillator problem.

```
tit      = van der Pol oscillator problem, case I
nv        = 21
nobj    = 1
neql    = 0
niql    = 0
ntrs    = 100
ipr      = 3
iact    = 5
igrad    = 1
del      = 1.0000e-05
acs      = 1.0000e-03
acv      = 1.0000e-03
act      = 1.0000e-12
X[ 1]= 1.0    -10.0    10.0
X[ 2]= 1.0    -10.0    10.0
X[ 3]= 1.0    -10.0    10.0
X[ 4]= 1.0    -10.0    10.0
X[ 5]= 1.0    -10.0    10.0
X[ 6]= 1.0    -10.0    10.0
X[ 7]= 1.0    -10.0    10.0
X[ 8]= 1.0    -10.0    10.0
X[ 9]= 1.0    -10.0    10.0
X[10]= 1.0    -10.0    10.0
X[11]= 1.0    -10.0    10.0
X[12]= 1.0    -10.0    10.0
X[13]= 1.0    -10.0    10.0
X[14]= 1.0    -10.0    10.0
X[15]= 1.0    -10.0    10.0
X[16]= 1.0    -10.0    10.0
X[17]= 1.0    -10.0    10.0
X[18]= 1.0    -10.0    10.0
X[19]= 1.0    -10.0    10.0
X[20]= 1.0    -10.0    10.0
X[21]= 1.0    -10.0    10.0
```

Table 4.3 Parameter file for the *van der Pol* oscillator problem.

```
2    0    0    0    0
0.    2.0
3    0    2    2    64    1    0    1    3
2    1    2    101    3
0 1 1 0 0 0
1 1 0 0 1 0
2 1 0 0 1 0
0 0 0 0 0 0
1 0 0 0 0 0
2 0 0 0 0 0
1 1 0 0 1 0
2 1 0 1 1 0
```

Table 4.4 Performance comparison of various numerical schemes for the oscillator problem, case I.

DSA	INTG	DIFF	INTP	NIT	J_0*	Conv. Par.	CPU
DDM	SIMPSN	DDERKF	Zero	11	1.4530	4.9577e-4	19.639
			First	21	1.4333	7.6480e-4	6.069
			Cubic	20	1.4334	5.4521e-4	18.146
		DDEABM	Zero	11	1.4530	4.9577e-4	18.455
			First	21	1.4333	7.6480e-4	19.147
			Cubic	20	1.4334	5.4521e-4	15.202
	GAUSS	DDERKF	Zero	12	1.4422	9.7245e-4	20.499
			First	21	1.4328	8.4954e-4	4.665
			Cubic	20	1.4334	5.4616e-4	14.392
		DDEABM	Zero	12	1.4422	9.7245e-4	19.078
			First	21	1.4328	8.4954e-4	17.686
			Cubic	20	1.4334	5.4616e-4	11.776
FDM	SIMPSN	DDERKF	Zero	13	1.4530	3.6313e-4	3.615
			First	21	1.4333	7.6261e-4	1.382
			Cubic	20	1.4334	5.4124e-4	1.542
		DDEABM	Zero	13	1.4530	3.6306e-4	3.615
			First	21	1.4333	7.5959e-4	3.244
			Cubic	20	1.4334	5.4139e-4	2.333
	GAUSS	DDERKF	Zero	14	1.4422	5.3821e-4	3.445
			First	21	1.4328	8.4715e-4	0.752
			Cubic	20	1.4334	5.4218e-4	0.871
		DDEABM	Zero	14	1.4422	5.3905e-4	3.555
			First	21	1.4328	8.3944e-4	2.754
			Cubic	20	1.4334	5.4269e-4	1.603
Averages							
DDM	—	—	—	17.5	1.4380	6.957E-04	15.396
FDM	—	—	—	18.2	1.4380	5.983E-04	2.393
—	SIMPSN	—	—	17.3	1.4407	5.922E-04	12.707
—	GAUSS	—	—	18.0	1.4361	7.154E-04	8.423
—	—	DDERKF	—	17.8	1.4380	6.474E-04	7.918
—	—	DDEABM	—	17.8	1.4380	6.466E-04	9.871
—	—	—	Zero	12.5	1.4476	5.925E-04	11.488
—	—	—	First	21.0	1.4331	8.047E-04	6.962
—	—	—	Cubic	20.0	1.4334	5.438E-04	8.233
Bullock and Franklin (1967)							J* = 1.433508
Jaddu and Shimemura (1999)							J* = 1.433487

Table 4.5 Various dynamic constraint treatments for the oscillator problem, case III.

NGP	DCT	NIT	Max. Vio.	Conv. Para.	J_0*	NMF	NCF	NTG	CPU
21	ALL	27	5.01403e-9	8.79668e-5	2.13771	594	594	340	2.273
	MAX	100	6.38948e-6	6.42630e-3	2.13758	2200	2200	243	8.341
	HYB	30	9.76433e-9	7.59396e-5	2.13772	660	660	388	2.594
101	ALL	41	8.03759e-8	6.72651e-5	2.13657	4183	4183	1764	34.710
	MAX	100	1.91428e-5	1.23887e-4	2.13658	10205	10205	1110	79.635
	HYB	50	2.60229e-6	7.37412e-5	2.13658	5107	5107	2193	43.564
Averages									
	ALL	34	4.26950E-8	7.76160E-5	2.13714	2388.5	2388.5	1052	18.4915
	MAX	100	1.27661E-5	3.27509E-3	2.13708	6202.5	6202.5	676.5	43.988
	HYB	40	1.30603E-6	7.48404E-5	2.13715	2883.5	2883.5	1290.5	23.079

Table 4.6 Comparison of various numerical schemes for the overhead crane system.

INTG	DIFF	INTP	NIT	Max. vio.	Conv. Para.	J_0*	CPU
SIMPSN	DDERKF	Zero	101	6.73320E-06	8.42785E-04	12.17290	1590.31
		First	95	5.31769E-09	9.40788E-04	12.04890	116.29
		Cubic	92	2.16952E-09	7.97492E-04	12.04730	165.252
	DDEABM	Zero	102	1.46089E-05	9.37978E-04	12.17290	1668.69
		First	93	4.02915E-06	9.99191E-04	12.04880	800.555
		Cubic	60	3.72381E-06	6.89197E-04	12.00040	280.323
GAUSS	DDERKF	Zero	101	6.73320E-06	8.42785E-04	12.17290	1584.946
		First	95	5.31769E-09	9.40788E-04	12.04890	114.797
		Cubic	92	2.16952E-09	7.97492E-04	12.04730	162.769
	DDEABM	Zero	102	1.46089E-05	9.37978E-04	12.17290	1599.916
		First	93	4.02915E-06	9.99191E-04	12.04880	799.932
		Cubic	60	3.72381E-06	6.89197E-04	12.00040	280.25
Average							
SIMPSN	—	—	90.5	4.85042E-06	8.67905E-04	12.08187	770.2
GAUSS	—	—	90.5	4.85042E-06	8.67905E-04	12.08187	757.1
—	DDERKF	—	96.0	2.24690E-06	8.60355E-04	12.08970	622.4
—	DDEABM	—	85.0	7.45395E-06	8.75455E-04	12.07403	904.9
—	—	Zero	101.5	1.06711E-05	8.90382E-04	12.17290	1611.0
—	—	First	94.0	2.01723E-06	9.69990E-04	12.04885	457.9
—	—	Cubic	76.0	1.86299E-06	7.43345E-04	12.02385	222.1
Hu *et al.* (2002)						≅ 12	

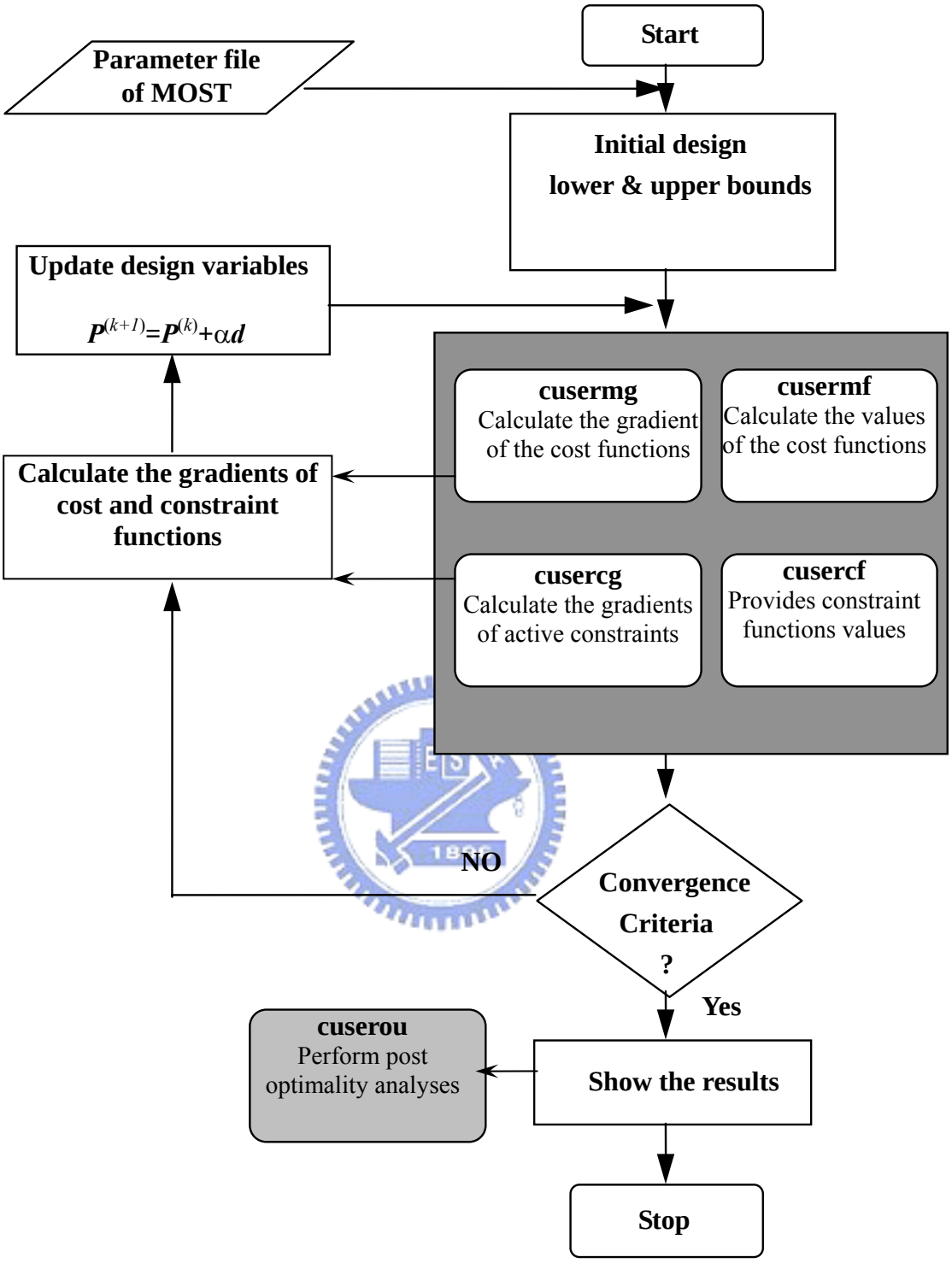


Figure 4.1 The architecture of MOST.

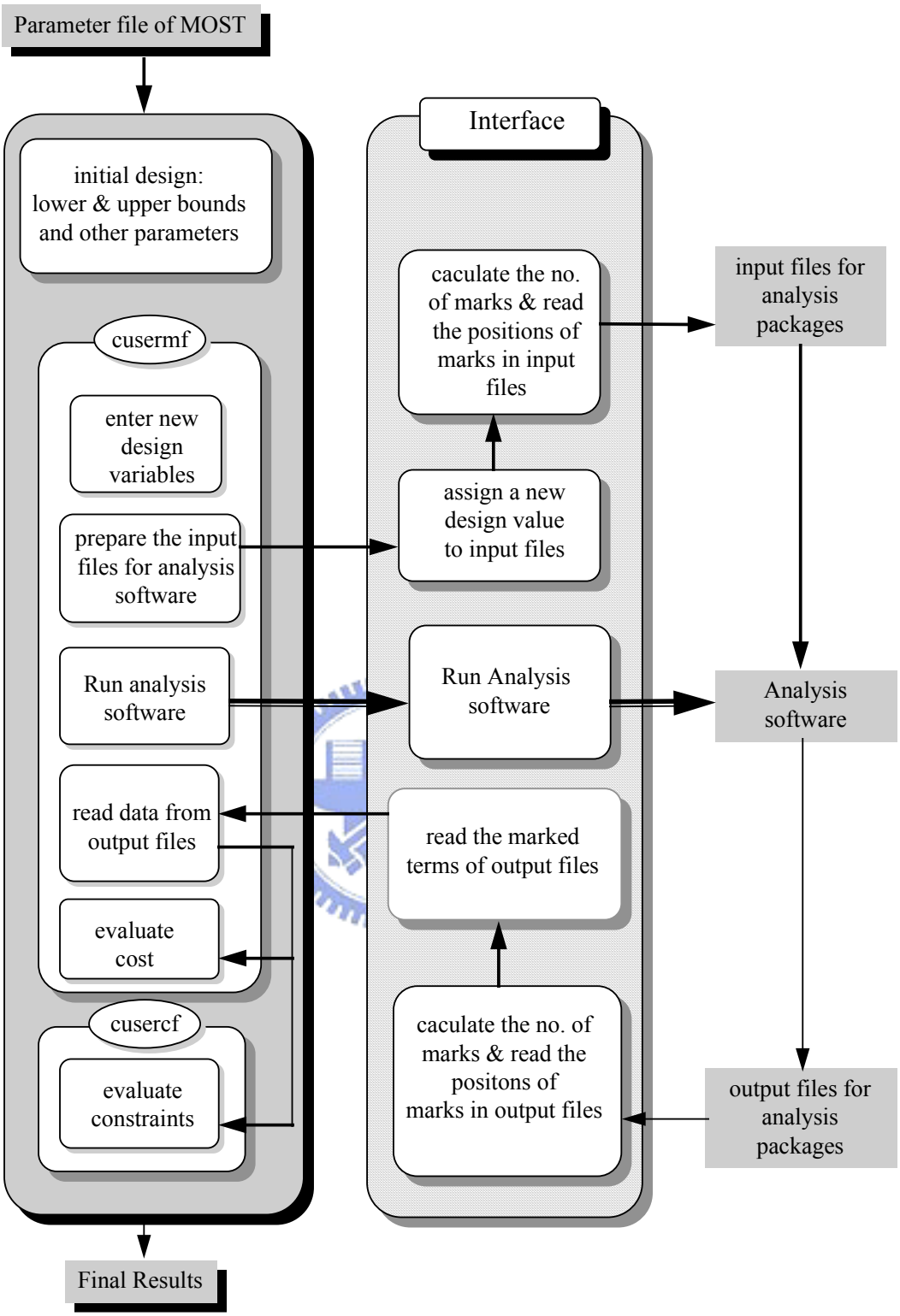


Figure 4.2 Architecture of the interface coupler.

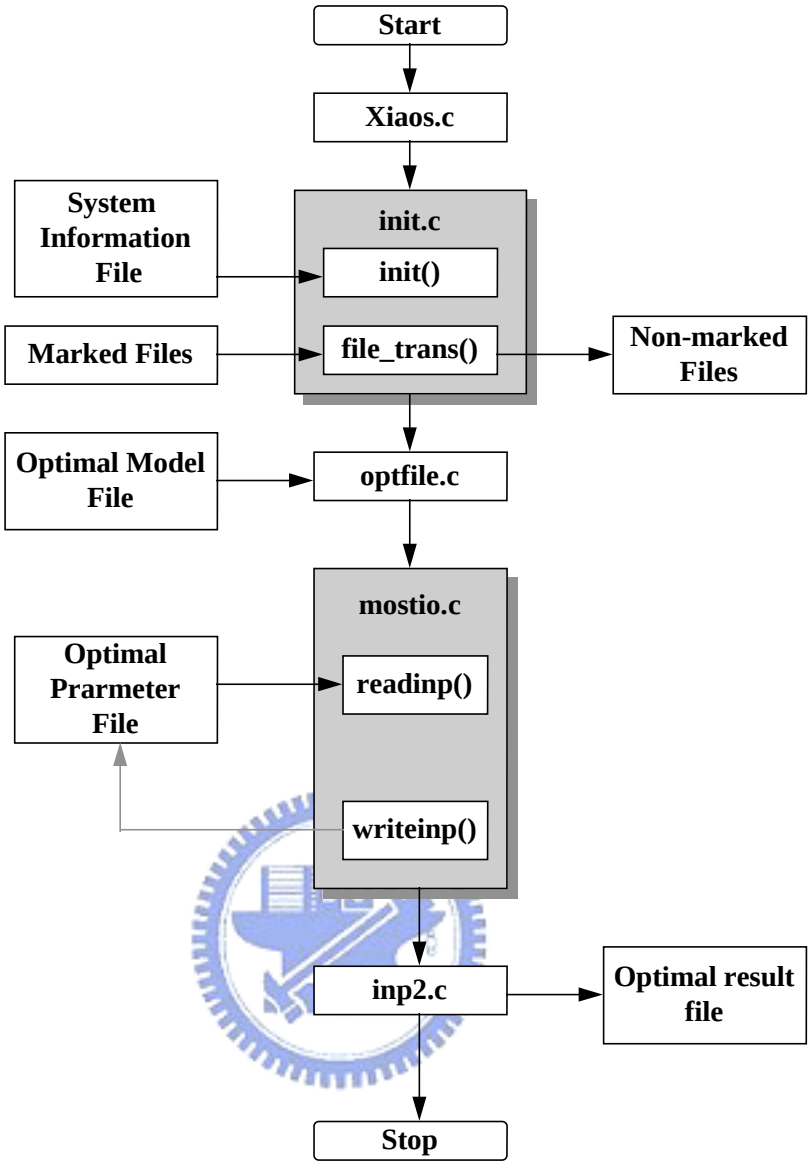


Figure 4.3 Architecture of the new interface coupler – IAOS.

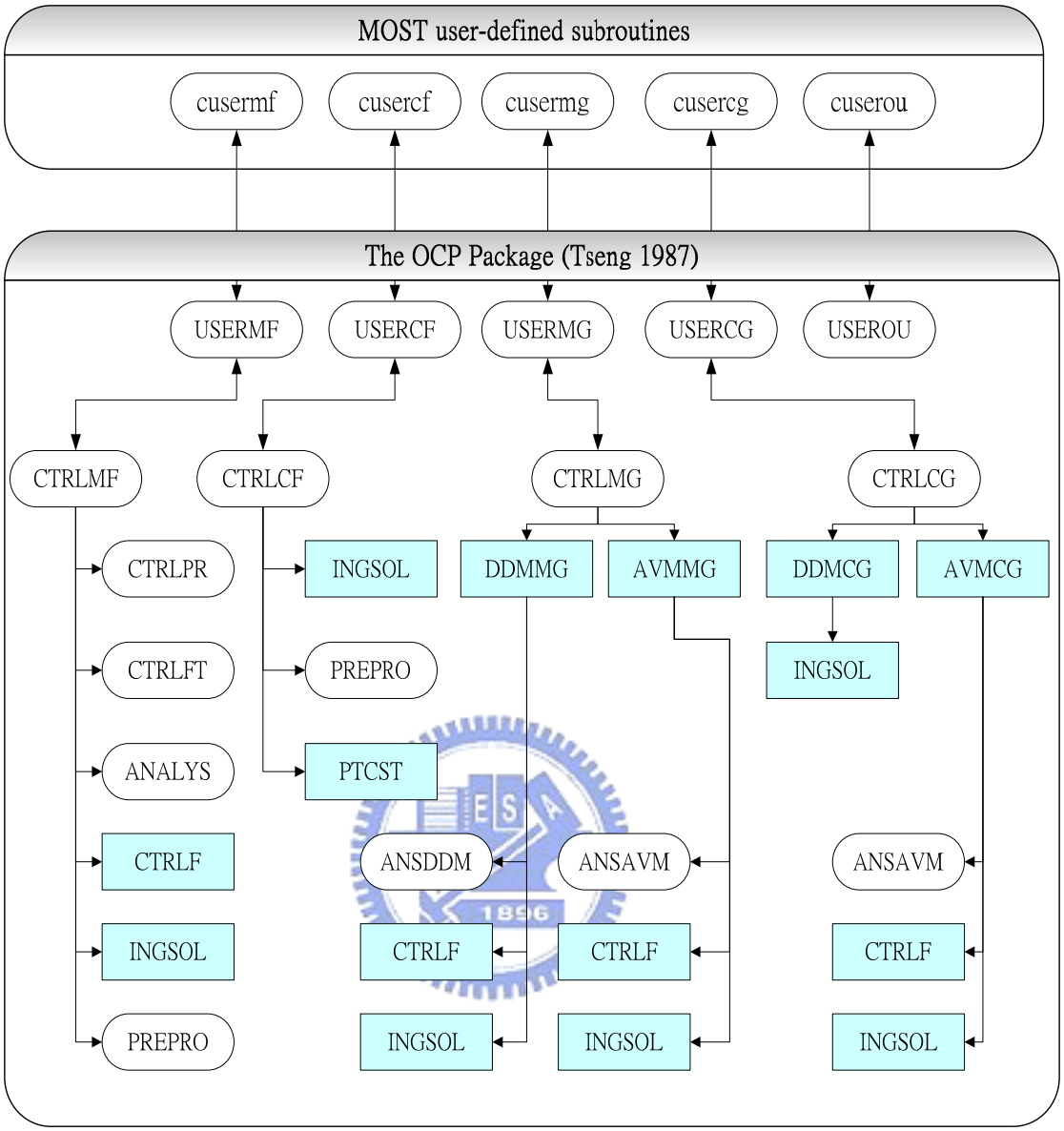


Figure 4.4 Structure chart of the OCP Solver.

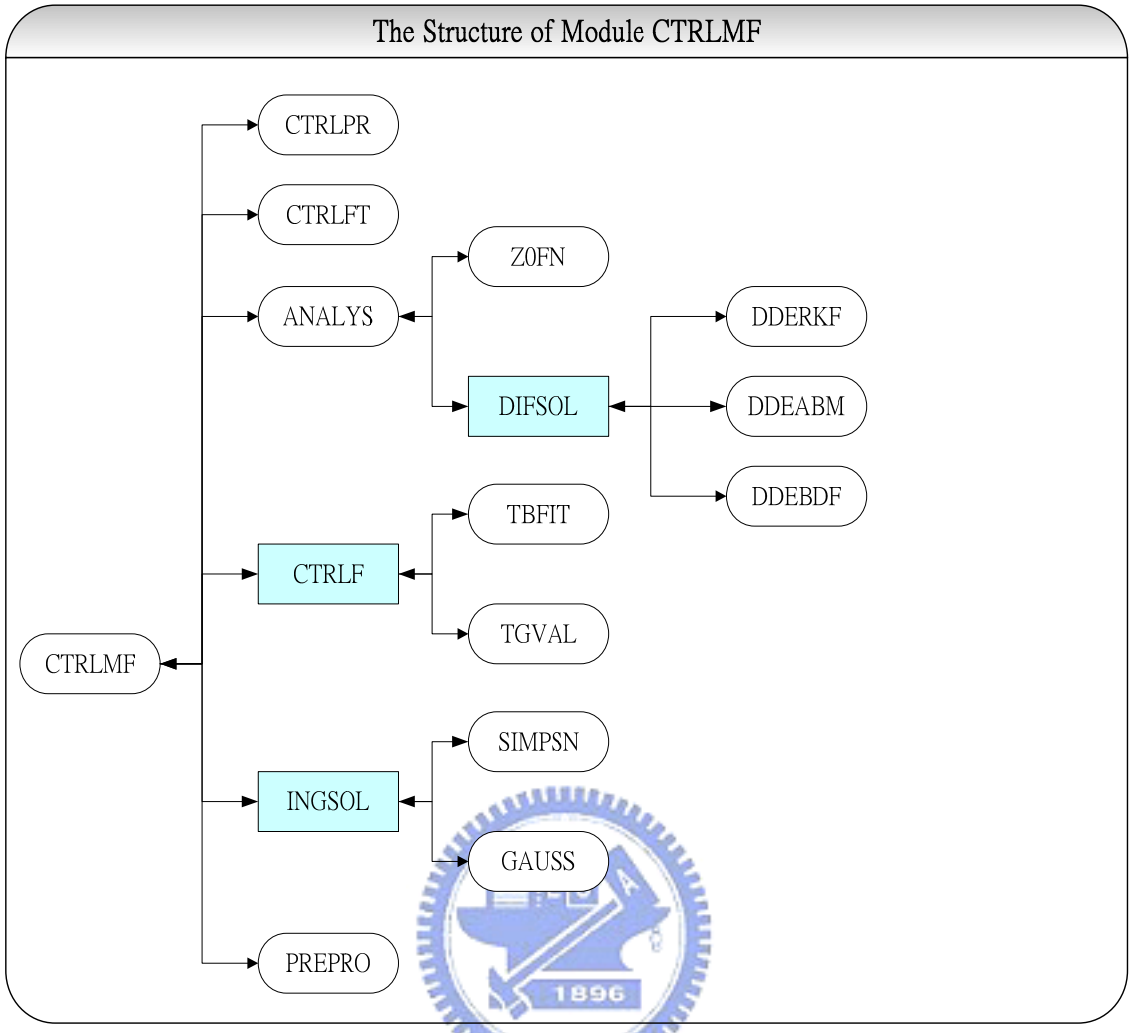


Figure 4.5 CTRLMF module.

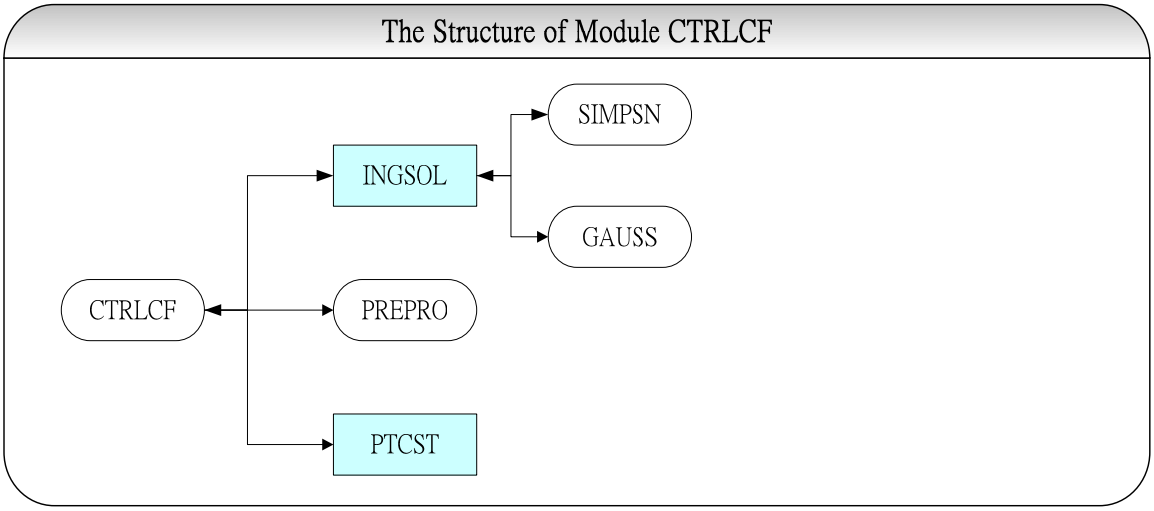


Figure 4.6 CTRLCF module.

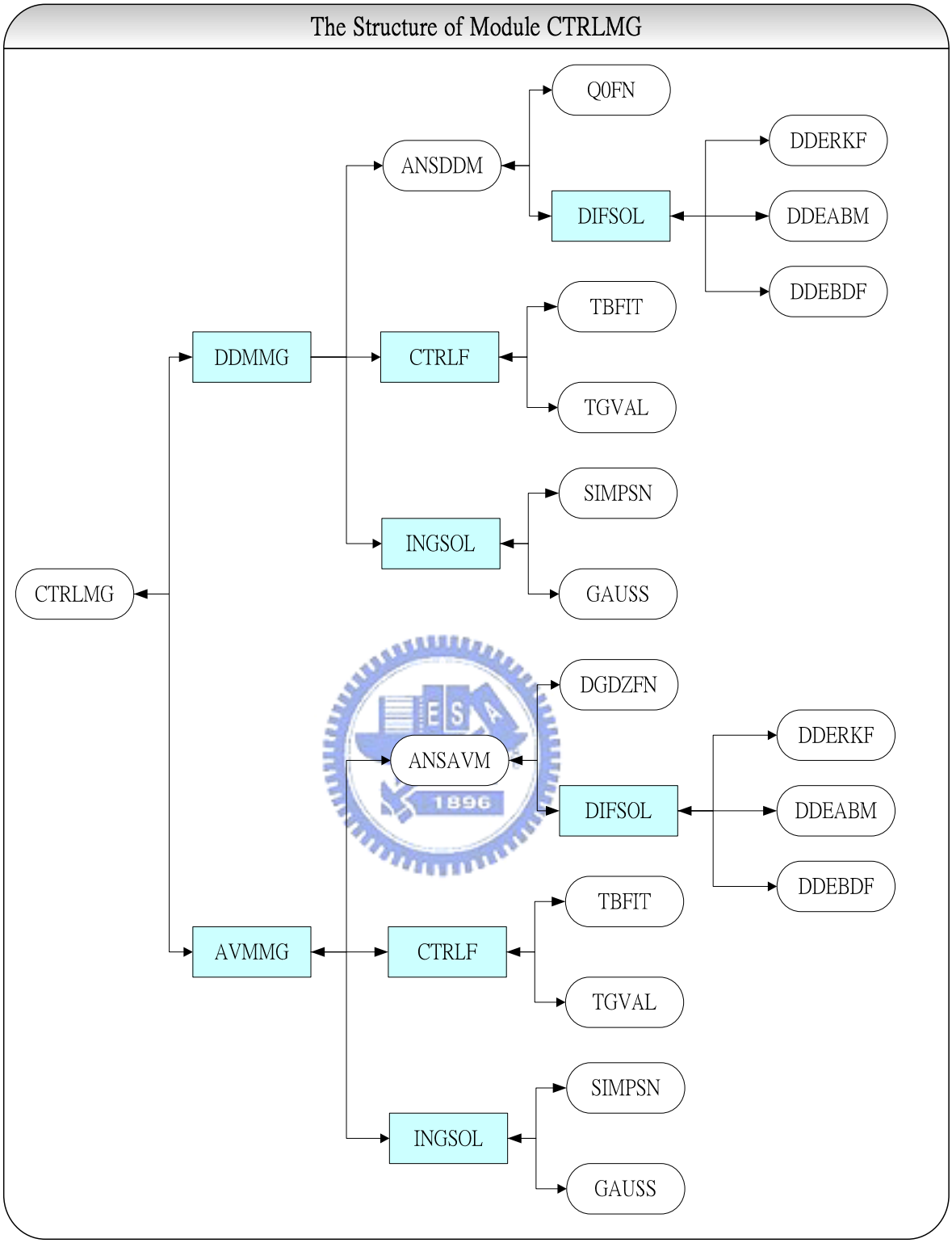


Figure 4.7 CTRMG module.

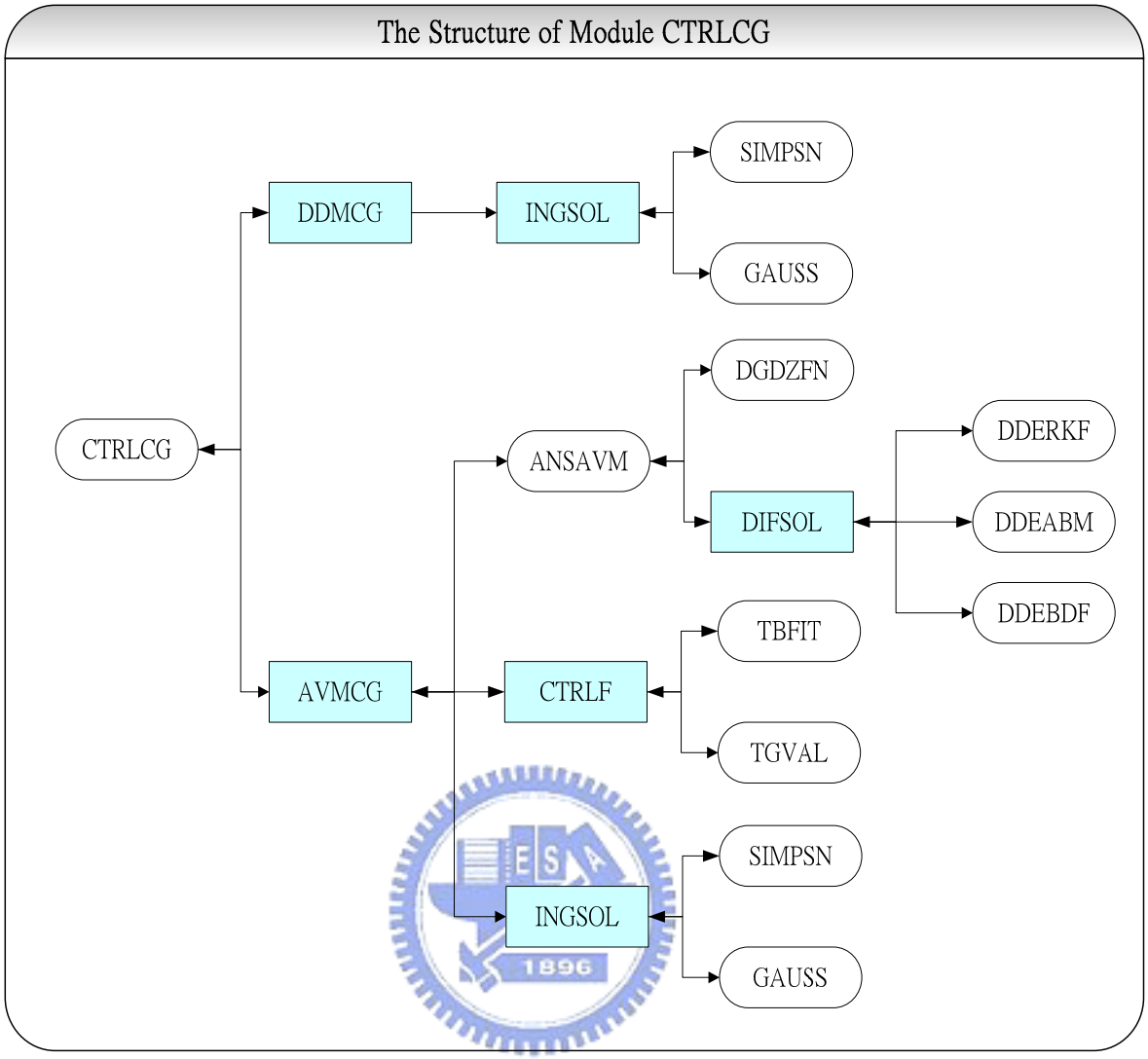


Figure 4.8 CTRCG module.

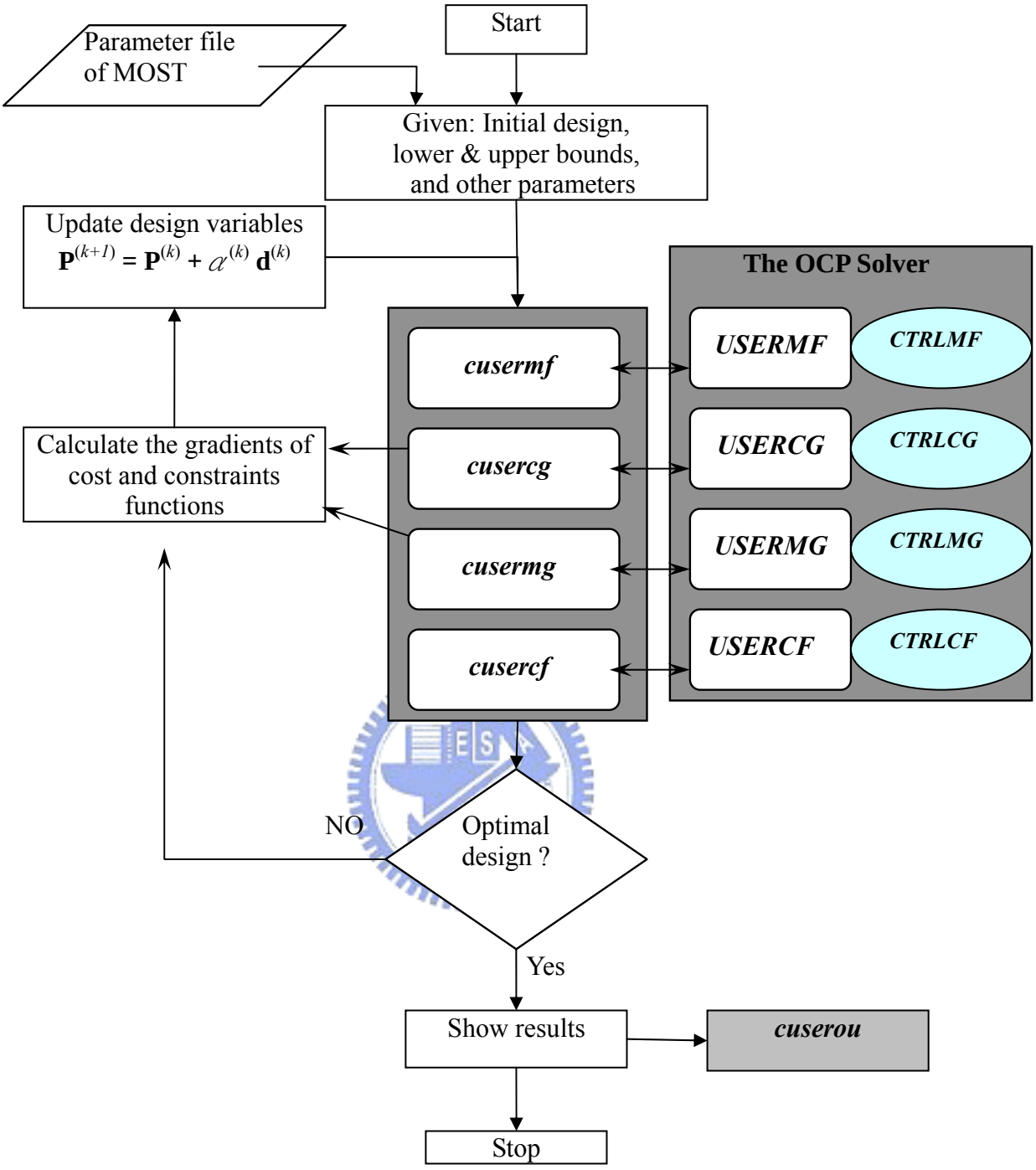


Figure 4.9 Connection architecture of MOST and OCP solver.

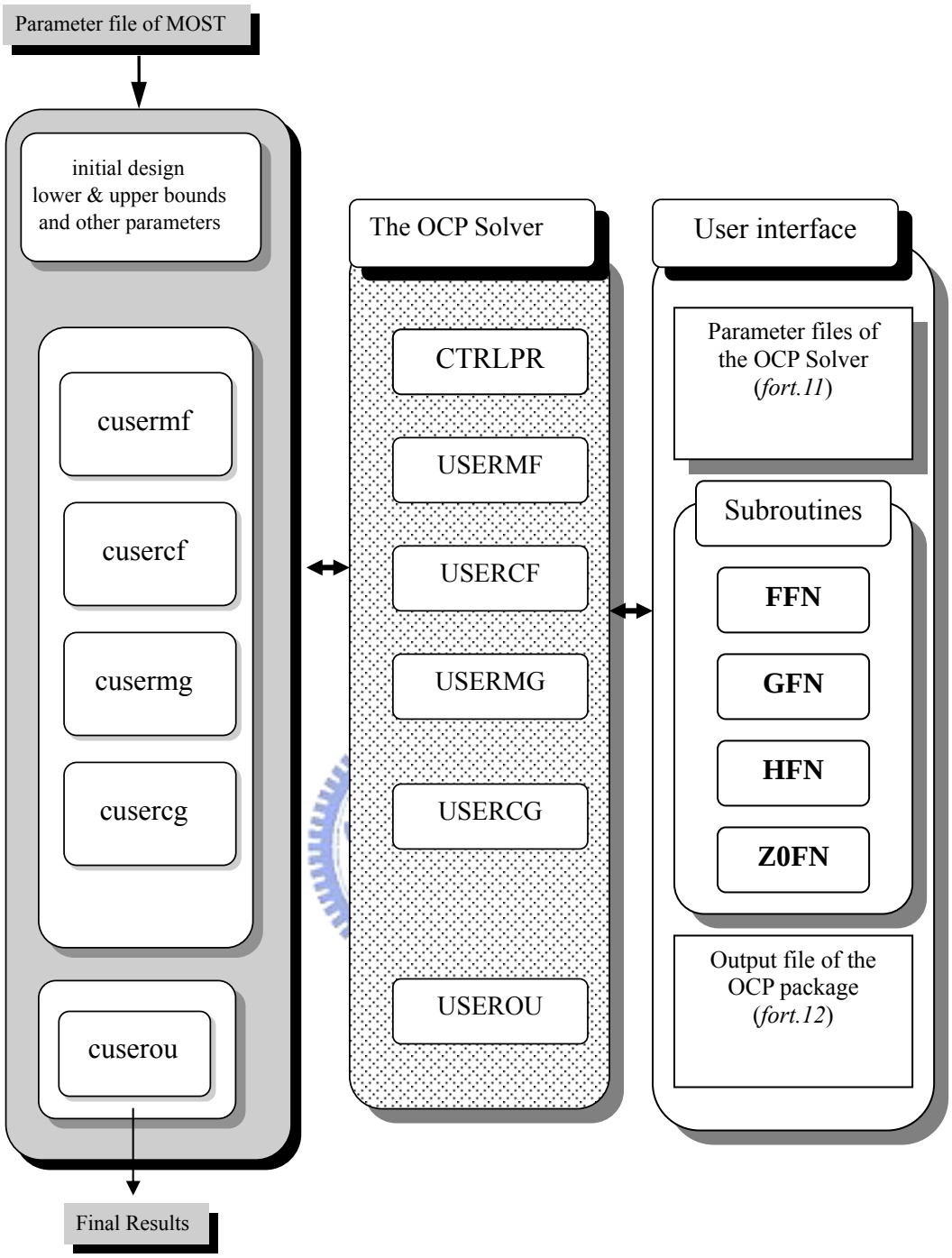


Figure 4.10 User interfaces for MOST and the OCP solver.

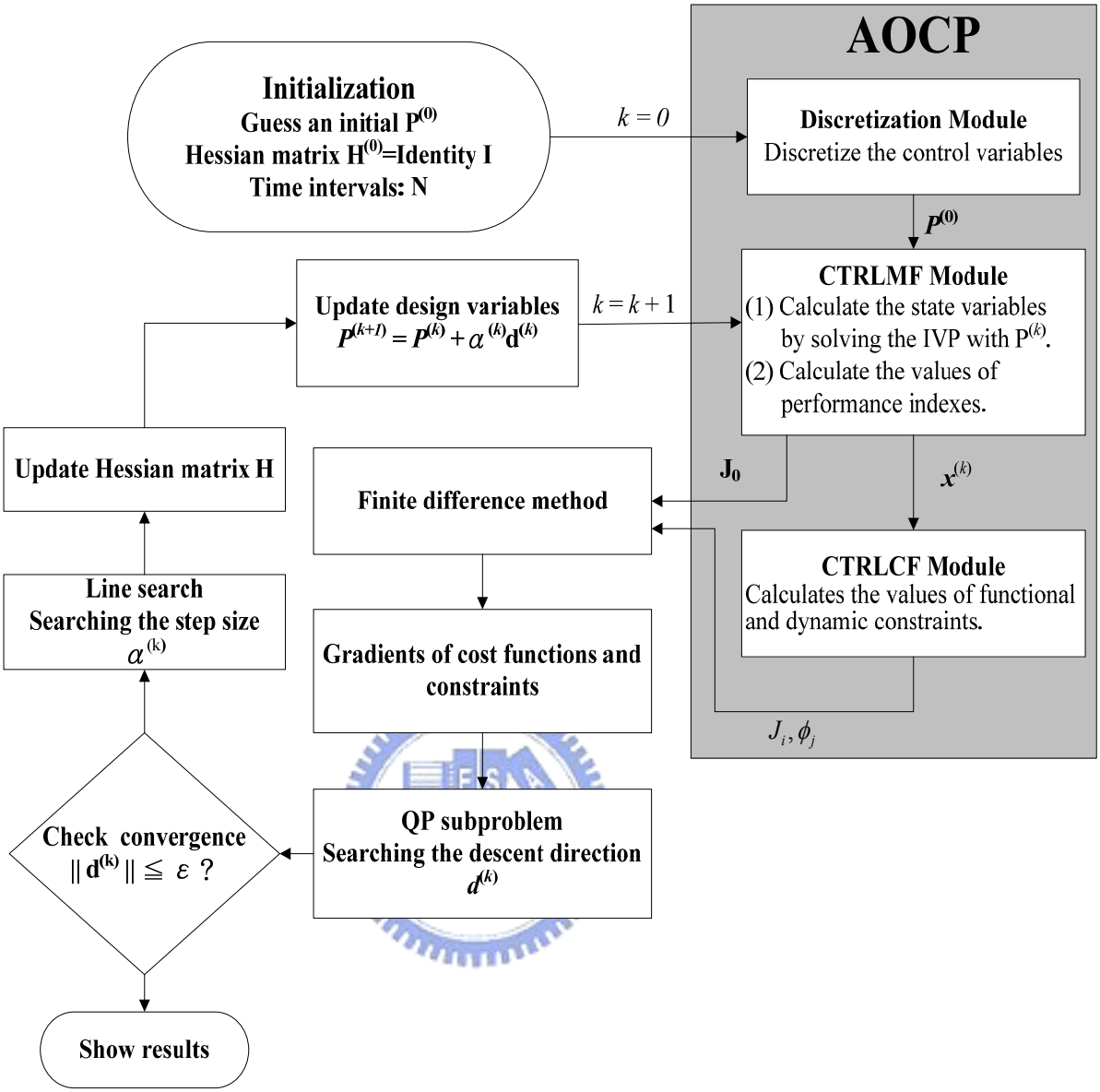


Figure 4.11 Flowchart for the OCP solver.

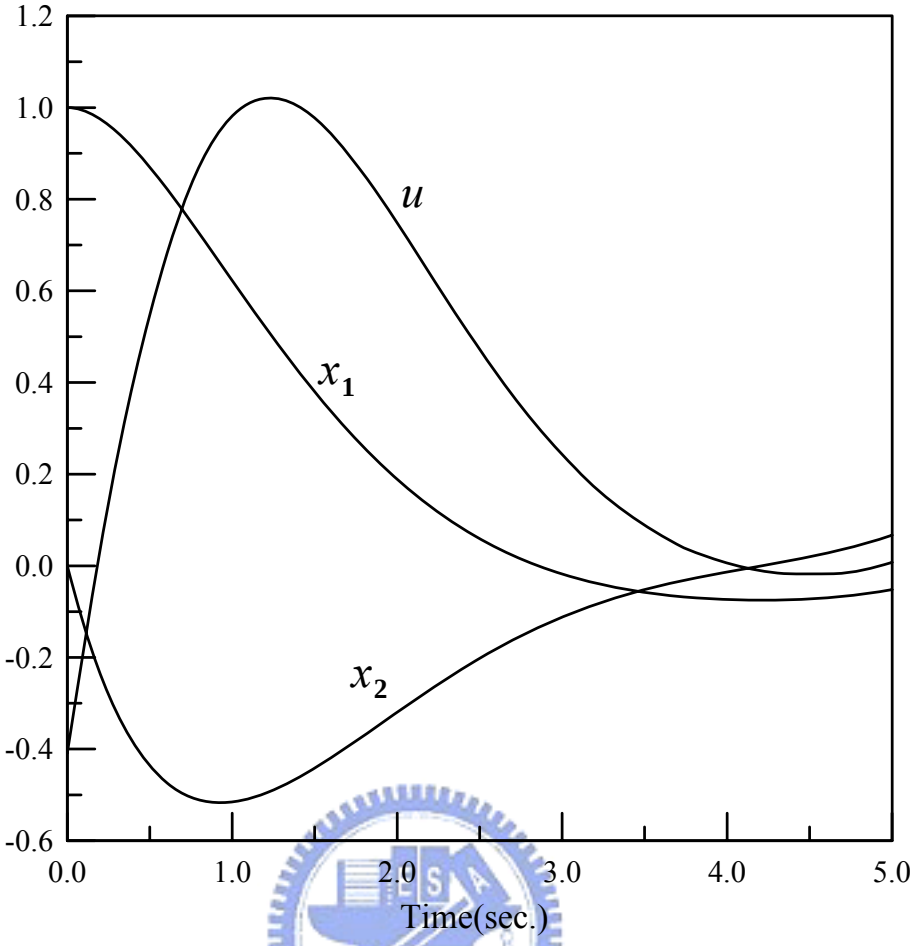


Figure 4.12 Control and state trajectories for *van de Pol* oscillator problem, case I.

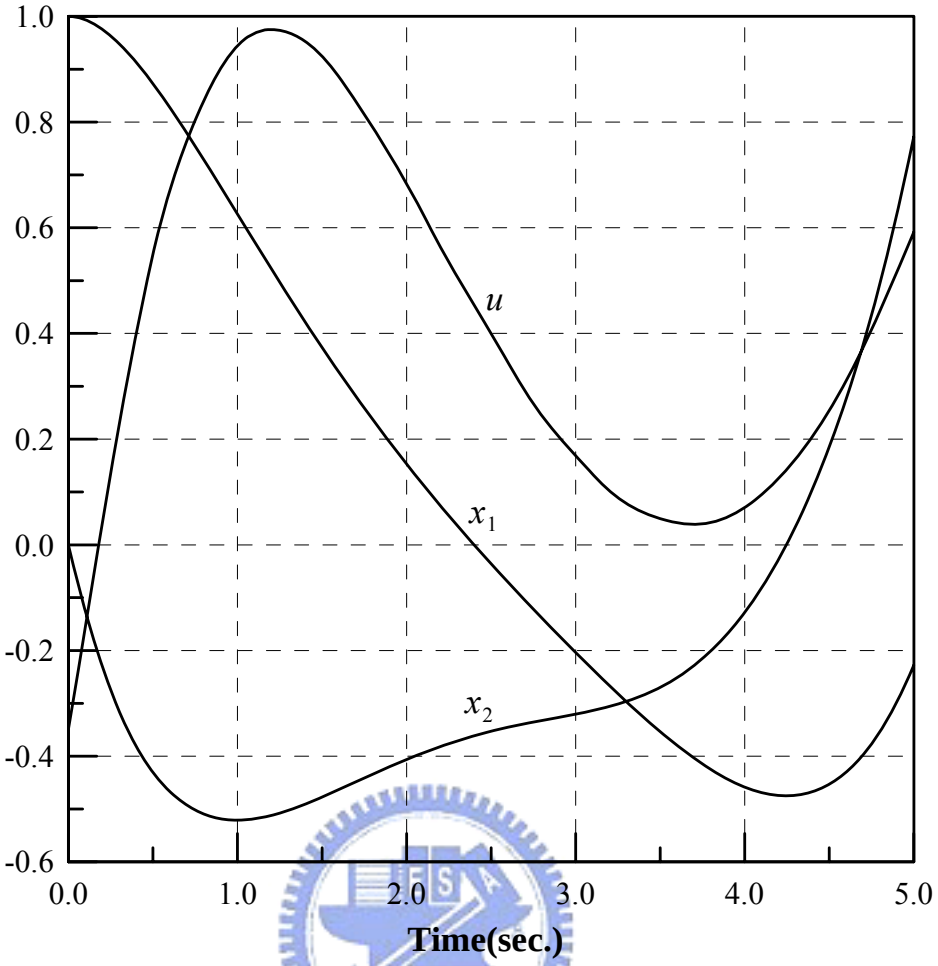


Figure 4.13 Control and state trajectories for *van de Pol* oscillator problem, case II.

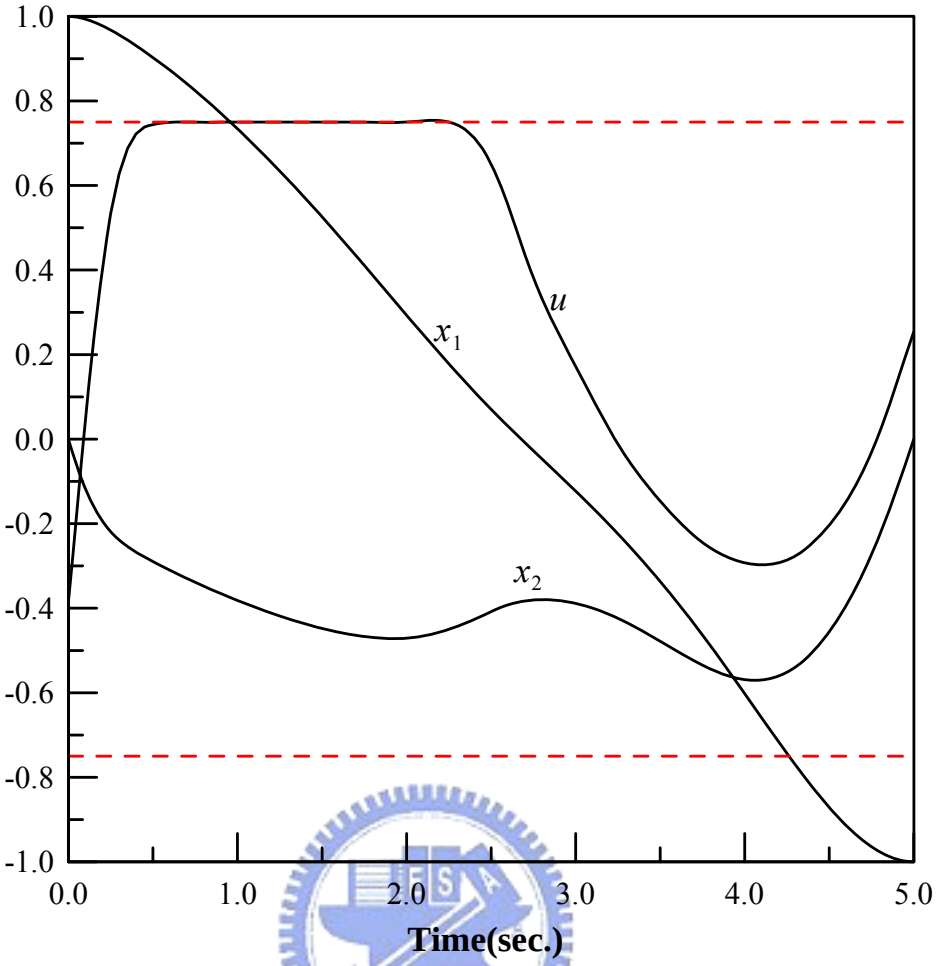


Figure 4.14 Control and state trajectories of *van de Pol* oscillator problem, case III.

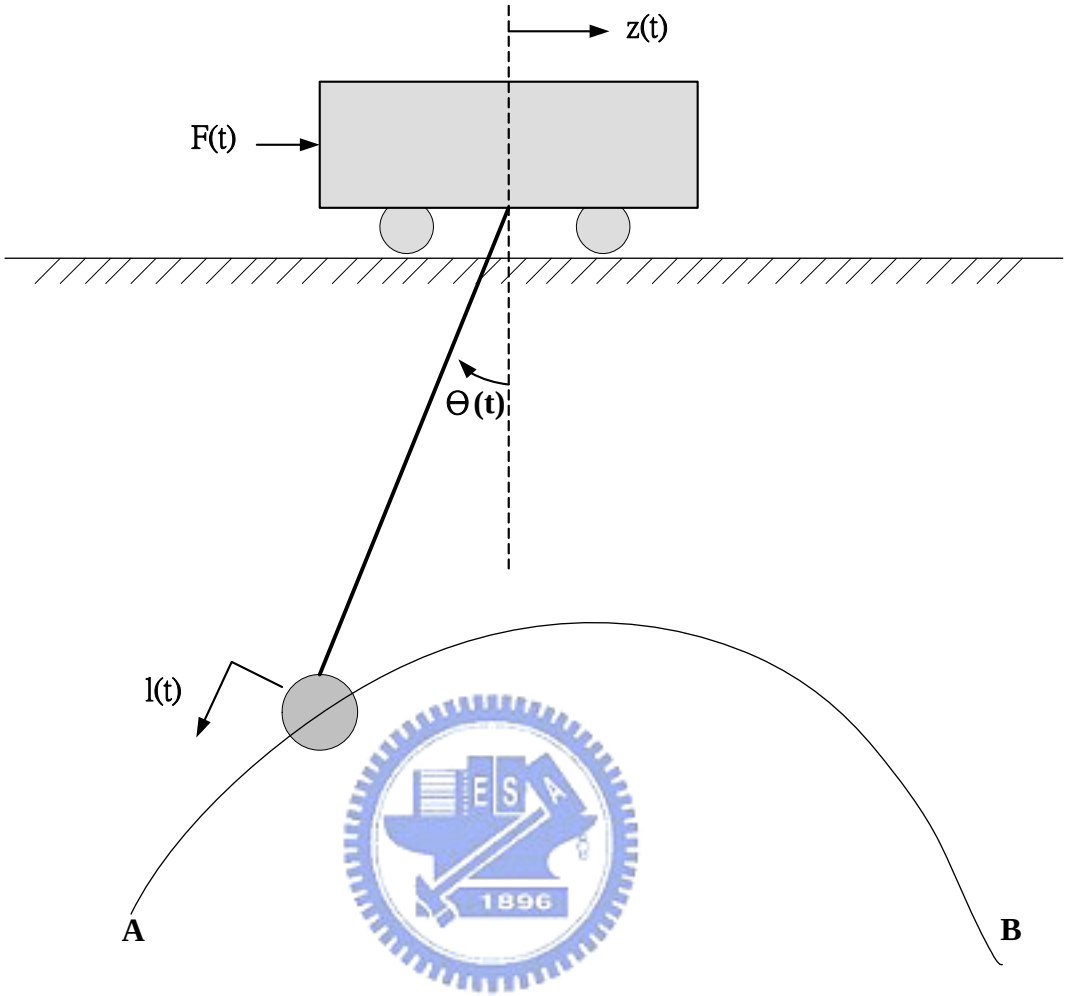
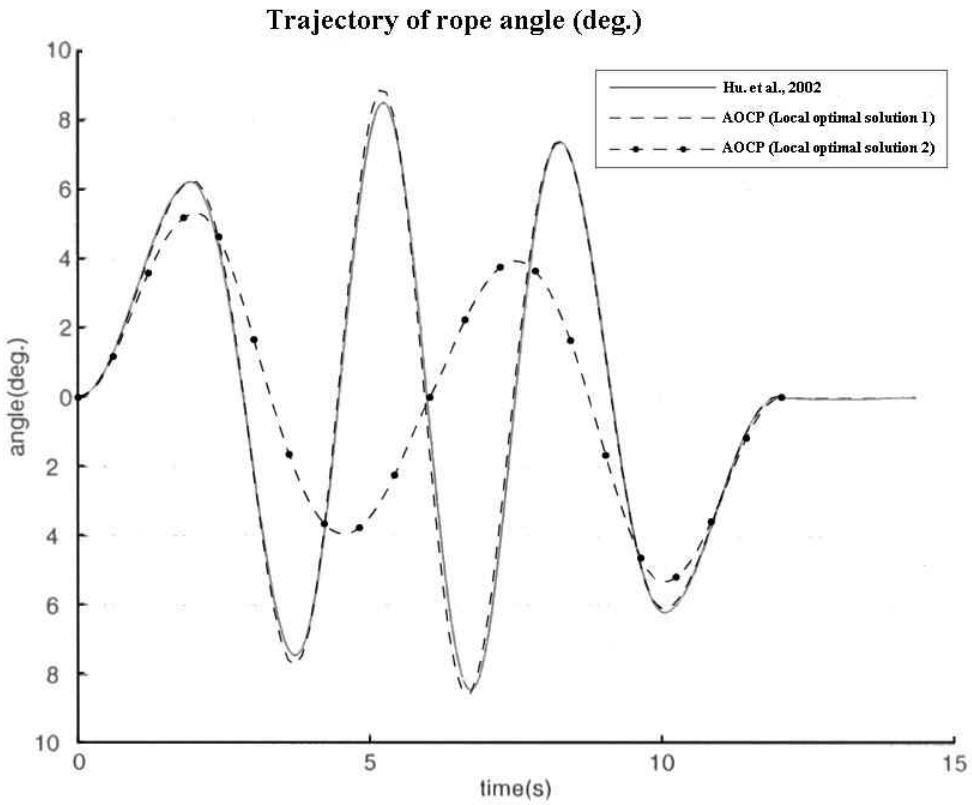
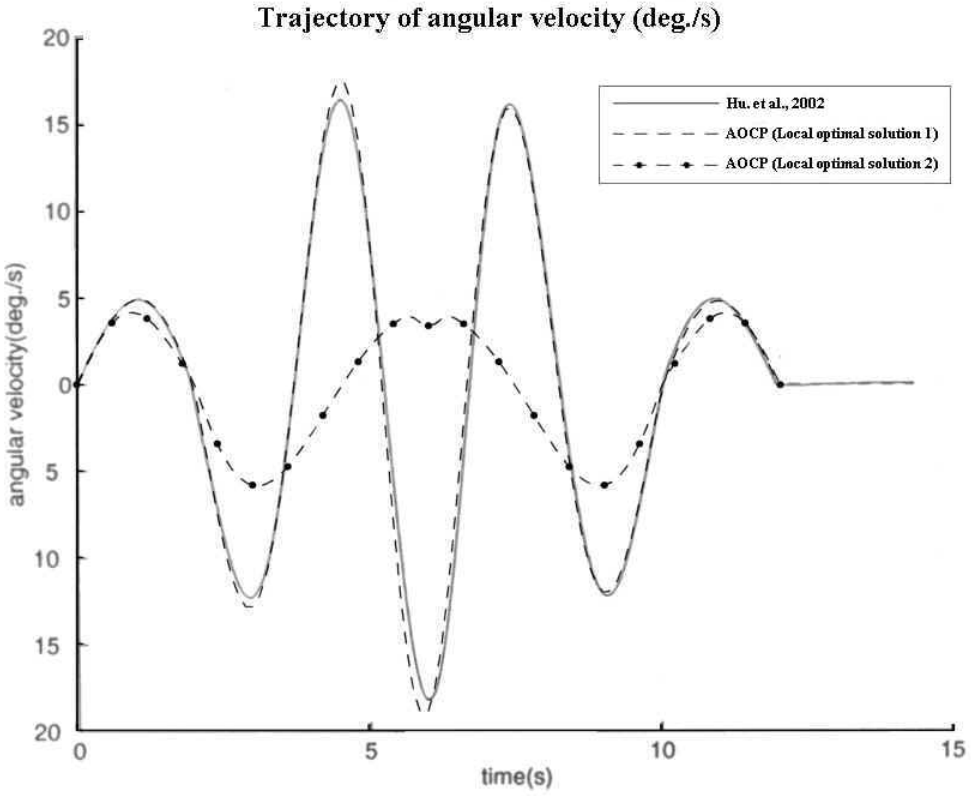


Figure 4.15 Schematic of the overhead crane system (Hu *et al*., 2002).

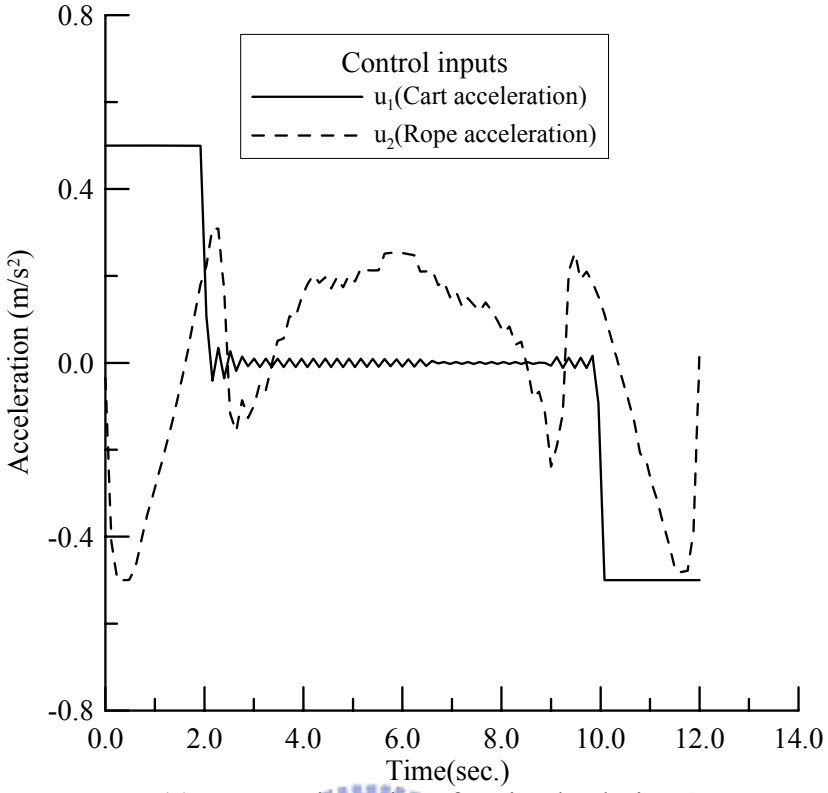


(a) Rope angle trajectory.

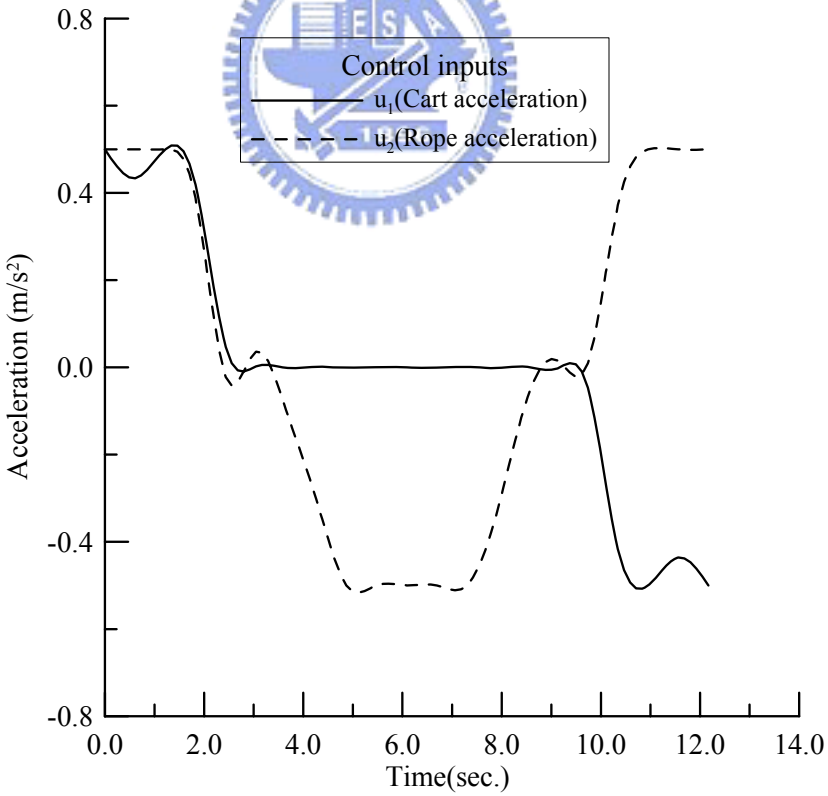


(b) Angular velocity trajectory.

Figure 4.16 State trajectories of the overhead crane system.

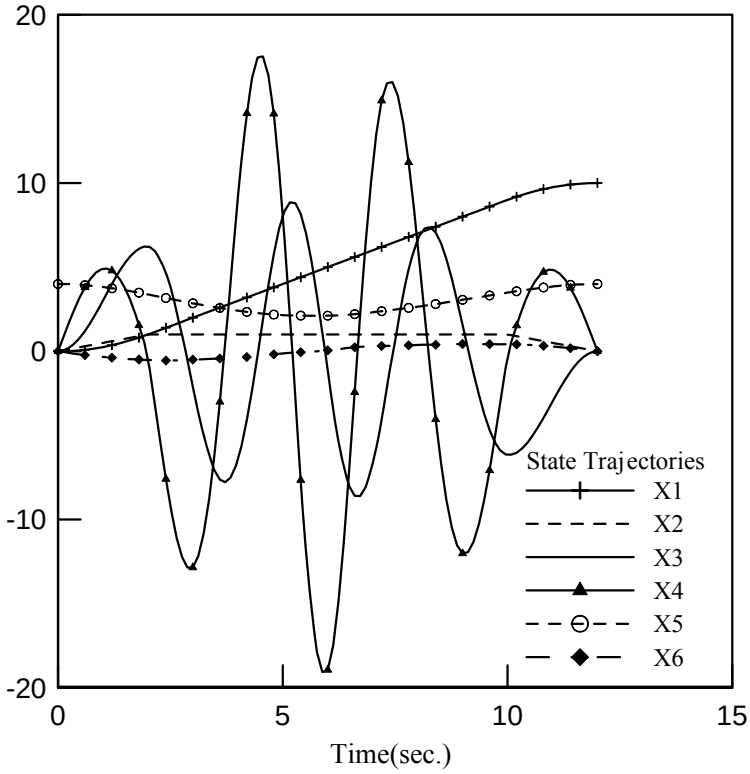


(a) Input trajectories of optimal solution 1.

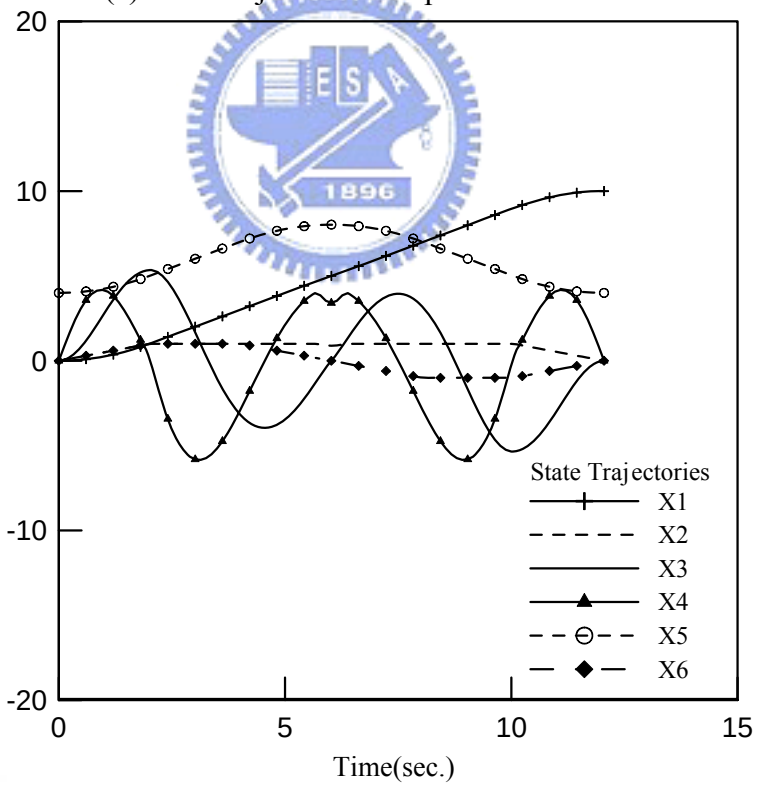


(b) Input trajectories of optimal solution 2.

Figure 4.17 Control trajectories for overhead crane system.



(a) State trajectories of optimal solution 1.



(b) State trajectories of optimal solution 2.

Figure 4.18 State trajectories with different initial guess for overhead crane system.

CHAPTER 5
A COMPUTATIONAL SCHEME FOR SOLVING THE DISCRETE-VALUED OPTIMAL CONTROL PROBLEM

5.1 Introduction

Time-optimal control problems have attracted the interest of researchers in the area of optimal control because they often occur in practical applications. Thus a series of essential results has been published concerning applications of Pontryagin's maximum principle to the time-optimal control of finite dimensional linear systems and low-order nonlinear systems. However, in the case of state- and/or control-constrained high-order nonlinear systems, solving the two-point boundary value problem that results from Pontryagin's maximum principle is difficult. Moreover, analytic solutions are impractical if the dimension of a system exceeds three (Kirk, 1970). Therefore, in recent research, many numerical techniques have been developed and adopted to solve time-optimal control problems.

For a time-optimal control problem, one of the most common types of control function is the piecewise-constant function by which a sequence of constant inputs is used to control a given system with suitable switching times. Additionally, when the control is bounded, a very commonly encountered type of piecewise-constant control is bang-bang, which switches between the upper and lower bounds of the control input. When the controls are assumed to be of the bang-bang type, the time-optimal control problem becomes one of determining the switching times. Several methods for determining TOCP switching times have been extensively studied in the literature (see, *e.g.*, Kaya and Noakes, 1996; Bertrand and Epenoy, 2002; Simakov, 2002). However, in these methods, the number of switching times must be known before their algorithms can be applied. In most practical cases, however, as pointed out earlier, the number of switching times is unknown before the control problems are solved. To overcome the numerical difficulties that arise during the process of finding the exact switching points, Lee *et al*. (1997) propose the control parameterization enhancing transform

(CPET), which is also extended to deal with optimal discrete-valued control problems (Lee *et al.*, 1999) and applied to solve the sensor-scheduling problem (Lee *et al.*, 2001).

In like manner, this dissertation focuses on developing a computational method to solve discrete-valued optimal control problems. This method consists of two computational phases: first, switching times are calculated using existing optimal control methods and second, the resulting information is used to compute the discrete-valued control strategy. The proposed algorithm, which integrates the existing optimal control solver with an enhanced branch-and-bound method (Tseng *et al*., 1995), is implemented and applied to some example systems, including that of the F-8 fighter aircraft.

5.2 Problem Formulations

5.2.1 Optimal Discrete-valued Control Problems

In many practical engineering applications, the control action is restricted to a set of discrete values that forms a discrete-valued control problem. An optimal discrete-valued control problem can be viewed as exactly determining the switching points of the optimal discrete-valued control. The major difference between continuous and discrete-valued control problems is the control function. For a piecewise-constant function $\boldsymbol{u}_{\mathbf{d}}$: $\left[0, t_f\right) \mapsto \mathbf{U}_d$ where $\mathbf{U}_d$ is a finite set in $\mathbb{R}^m$, if $\boldsymbol{u}_d$ has a finite number of discontinuous or switching points, it is referred to as an admissible control. Letting $\mathcal{U}_d$ be the class of all such admissible controls, in like manner to problem ($\mathrm{P}_{\mathcal{U}}$), the optimal discrete-valued control problem may be stated formally as follows: Given the dynamic system (2.1, 2.2), $\boldsymbol{u}_{\mathbf{d}} \in \mathcal{U}_d$ must be found such that the cost functional (performance index)

$$J_0 = \Phi_0\left(\boldsymbol{b}, \boldsymbol{x}(t_f), t_f\right) + \int_0^{t_f} \mathcal{L}_0\left(\boldsymbol{b}, \boldsymbol{u}_d(t), \boldsymbol{x}(t), t\right) dt \tag{5.1}$$

is minimized subject to the constraint

$$J_i = \Phi_i\left(\boldsymbol{b}, \boldsymbol{x}(t_f), t_f\right) + \int_0^{t_f} \mathcal{L}_i\left(\boldsymbol{b}, \boldsymbol{u}_d(t), \boldsymbol{x}(t), t\right) dt \begin{cases} = 0; & i = 1, \ldots\ldots\ldots, N_e \\ \le 0; & i = N_e + 1, \ldots., N_T \end{cases} \tag{5.2}$$

and the following continuous inequality constraint on the function of the state and control:

$$\psi_j\left(\boldsymbol{b}, \boldsymbol{u}_d(t), \boldsymbol{x}(t), t\right) \le 0 \;; j = 1, \ldots\ldots\ldots, q \quad , \forall t \in \left[0, t_f\right]. \tag{5.3}$$

It should be noted that, for a given $\boldsymbol{u}_{\mathbf{d}} \in \mathcal{U}_d$, the right hand side of Eq. (2.1) may be discontinuous at the switching points of $\boldsymbol{u}_d$. Denoting these switching points as $t_1, t_2, \ldots, t_N$ and also defining t_0 and t_{N+1} such that $0 = t_0 < t_1 < t_2 < \ldots < t_N < t_{N+1} = t_f$, the solution $\boldsymbol{x}(t)$ is then obtained in a piecewise manner by successive integration over each time interval $[t_i, t_{i+1}]$, $i = 0, 1, 2, \ldots, N$. The resulting $\boldsymbol{x}(t)$ is continuous and piecewise differentiable on $(0, t_f)$.

5.2.2 Mixed-Discrete Optimal Control Problems

Mixed-discrete control problems that control functions are mixed with continuous and discrete functions is considered to meet the generality. For a continuous control variable, any piecewise continuous function $\boldsymbol{u}_c$ from $[0, t_f]$ into $\mathbb{R}^m$ may be taken as an admissible control. For optimal discrete-valued control problems, a piecewise-constant function $\boldsymbol{u}_d$, $\boldsymbol{u}_d : \left(0, t_f\right] \mapsto \mathbf{U}_\mathrm{d}$, may be taken as an admissible control, where $\mathbf{U}_\mathrm{d}$ is a finite set in $\mathbb{R}^m$. Letting $\mathcal{U}$ be the class of all such admissible controls, then a mixed-discrete optimal control problem may be stated formally as follows: Given the dynamical system (2.1, 2.2), $\boldsymbol{u} = [\, \boldsymbol{u}_c^{\mathrm{T}}, \boldsymbol{u}_d^{\mathrm{T}}\,]^{\mathrm{T}} \in \mathcal{U}$ must be found such that the cost functional (performance index)

$$J_0 = \Phi_0\left(\boldsymbol{b}, \boldsymbol{x}(t_f), t_f\right) + \int_0^{t_f} \mathcal{L}_0\left(\boldsymbol{b}, \boldsymbol{u}(t), \boldsymbol{x}(t), t\right) dt \tag{5.4}$$

is minimized subject to the constraint

$$J_i = \Phi_i\left(\boldsymbol{b}, \boldsymbol{x}(t_f), t_f\right) + \int_0^{t_f} \mathcal{L}_i\left(\boldsymbol{b}, \boldsymbol{u}(t), \boldsymbol{x}(t), t\right) dt \begin{cases} = 0; & i = 1, ..., N_e \\ \le 0; & i = N_e + 1, ..., N_T \end{cases} \tag{5.5}$$

and the following continuous inequality constraint on the function of the state and control:

$$\psi_j\left(\boldsymbol{b}, \boldsymbol{u}(t), \boldsymbol{x}(t), t\right) \le 0; j = 1, ..., q \qquad \forall t \in \left[0, t_f\right]. \tag{5.6}$$

5.2.3 Time-Optimal Control Problems

For a time-optimal control problem, the terminal time, t_f, is not fixed and is treated as a design variable in $\boldsymbol{b}$. The system govern equations described by Eq. (2.1) are expressed in general first-order form. Equation (5.5) represents the mixed state and control constraints, and the terminal conditions are treated as equality constraints in its first term. Then the class of time optimal control problems can be stated formally in the following manner.

Subject to the system (2.1, 2.2) together with the final condition,

$$\boldsymbol{x}(t_f) = \boldsymbol{x}_f, \tag{5.7}$$

control $\boldsymbol{u} \in \mathcal{U}$ must be found such that t_f is minimized, where $\boldsymbol{x}_f$ is a given vector in $\mathbb{R}^n$. For convenience, this time-optimal control problem will be referred to as problem (TP) whose cost functional is then t_f. Clearly, the problem (TP) can be written as follows: Given the dynamical system (2.1, 2.2), $\boldsymbol{u} \in \mathcal{U}$ must be found such that the cost functional

$$J_0 = \int_0^{t_f} dt = t_f, \tag{5.8}$$

is minimized subject to the constraint Eq.(5.5) and continuous inequality constraint Eq.(5.6).

5.3 Mixed-Integer NLP Algorithm for Solving TOCP

Most discrete programming methods are based on the assumption that discontinuous optimization problems are transformed into multiple continuous optimization subproblems to take advantage of well-established continuous optimization algorithms. These continuous optimization problems are solved by imposing constraints on the discreteness of the design variables. The optimal discrete solution is taken from among the continuous solutions obtained in the optimization sub-problems. However, the large number of discontinuous design variables greatly increases the number of the continuous optimization subproblems. Tseng *et al*. (1995) presented an enhanced branch-and-bound method for reducing the number of executions of the continuous-optimization scheme by intelligently selecting the bounding route. Because such an enhanced branch-and-bound method dramatically reduces the total number of continuous optimization runs executed and speeds up its convergence (Tseng *et al*., 1995), it is adopted herein and integrated with the AOCP to develop a mixed integer NLP algorithm for solving time-optimal control problems.

5.3.1 Integrating the AOCP and Enhanced Branch-and-Bound Method

The algorithm developed in this dissertation consists of three major processes: branching, the AOCP, and bounding. Initially, all discrete-valued restrictions are relaxed and the resulting continuous NLP problem is solved using the AOCP. If the solution of continuous optimum design problem occurs when all discrete-valued variable values are in the discrete set $\mathbf{U_d}$, which is preset by the user to meet practical requirements, then an optimal solution is determined and the procedure ends. Otherwise, the algorithm selects one of the discrete-valued variables whose value is not in the discrete set $\mathbf{U_d}$ – for example, the *j-th* design variable, P_j, with value $\hat{P}_j$ – and branches on it.

Branching process: In the branching process, the original design domain is divided into

three subdomains by two allowable discrete values, $\bar{u}_i$ and $\bar{u}_{i+1}$, that are nearest to the continuous optimum, as shown in Figure 5.1. Among the three subdomains, subdomain II, included in the continuous solution but not in the feasible discontinuous solution, is dropped. In the other two subdomains, called nodes, two new NLP problems are formed by adding simple bounds, $\hat{P}_j \le \bar{u}_i$ and $\hat{P}_j \ge \bar{u}_{i+1}$, respectively, to the continuous NLP problems. One of the two new NLP problems is selected and solved next. Many search methods based on tree searching – including depth-first search, breadth-first search and best-first search – can be applied to choose the next branching node. The branching process is repeated in each of the subdomains until the feasible optimal solution is found in which all the discrete variables have allowable discrete values. Obviously, the number of subdomains may grow exponentially so that a great deal of computing time is required. Thus in the enhanced branch-and-bound method (Tseng *et al*., 1995), multiple branching and unbalanced branching strategies have been developed to improve the efficiency of the method.

Bounding process: In discrete optimization, the minimum cost is always greater than or equal to the cost of the original regular optimal design that was originally branched. This fact provides a guideline for when branching should be stopped. If the branching process yields a feasible discontinuous solution, then the corresponding cost value can be considered a bound. Any other subdomain that imposes a continuous minimum cost larger than this bound need not be branched further. This bounding strategy can be used to select the branching route intelligently and avoid the need for a complete search over all the branches.

5.3.2 Algorithm for Solving Discrete-valued Optimal Control Problems

In this dissertation, the AOCP algorithm proposed in Section 3.4.7 is used as the core iterative routine of the enhanced branch-and-bound method. All candidates will be evaluated and finally an optimal solution can be found. Here, symbol ***S*** defined in Eq. (3.11) is used to represent the discretized control variable set and the ***P*** defined in Eq. (3.12) is the design

variable vector. Assuming that the problem at least has one feasible solution, it can then be proven that an optimal solution exists and can be found by the proposed method. The details of the two-phase scheme algorithm are as follows and Figure 5.2 presents a schematic flow chart of the algorithm for solving discrete-valued optimal control problems.

Algorithm: Combines AOCP and enhanced branch-and-bound methods

Initialization: Relax all discrete-valued restrictions and then place the resulting continuous NLP problem on the branching tree.

Set the cost bound $J_{max} = \infty$.

while (there are pending nodes in the branching tree) **do**

1. Select an unexplored node from the branching tree.
2. Control discretization.
3. **Repeat** (for *k*-th AOCP iteration)
   (1). Solve the initial value problem for state variable $\boldsymbol{x}^{(k)}$ of AOCP.
   (2). Calculate the values of the cost function, J_0, and the constraints.
   (3). Solve the $QP^{(k)}$ problem by applying the BFGS method to obtain the descent direction $\boldsymbol{d}^{(k)}$.
   (4). **if** ($QP^{(k)}$ is feasible and convergent) **then exit** AOCP.
   (5). Find the step size $\alpha^{(k)}$ of the SQP method by using the line search method.
   (6). Update the design variable vector: $\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)} + \alpha^{(k)} \boldsymbol{d}^{(k)}$.
4. **if** (NLP is optimal) and ($J_0 < J_{max}$) **then**

   if ($\mathbf{S}^{(k+1)}$ is feasible) **then**

      Update the current best point by setting the cost bound $J_{max} = J_0$.

      Add this node to the feasible node matrix.

   else

      Evaluate the values of criteria for selecting the branch node.

Choose a discrete-valued variable $\mathbf{S}_l^{(k+1)} \notin \mathcal{U}$ and branch it.

Add two new NLP problems into the branching tree.

Drop this node.

endif

else

Stop branching on this node.

endif

end while.

5.4 Two-Phase Scheme for Solving TOCP

The mixed integer NLP algorithm developed in this dissertation is one type of switching time computation (STC) method. Most switching time computation methods (see, *e.g.*, Kaya and Noakes, 1996; Lucas and Kaya, 2001; Simakov *et al*., 2002) assume that the structure of the control is bang-bang and the number of switching times is known. Unfortunately, the information on the switchings of several practical time-optimal control problems is unknown and hard to compute using analytical methods. Hence, to overcome this difficulty, this dissertation proposes a two-phase Scheme that consists of the AOCP plus the mixed integer NLP method. In Phase I, the AOCP is used to calculate the information on switchings with rough time grids so that the information can be used in Phase II as the feasible initial design of the mixed integer NLP method. This scheme is described briefly below.

Phase I: Find the information about the switching times and terminal time.

1. Solve the time-optimal control problem using continuous controls by following the steps of the AOCP method proposed in Section 3.4.
2. Based on the numerical results, extract information about the switching times and terminal time, t_f.

Phase II: Calculate the exact solutions

3. Based on the information about switching times obtained in Phase I, treat the switchings as design variables and add them into the time grid vector ***T*** defined in Eq. (3.10). It should be noted that each interval between the upper and lower bounds on each of those design variables must include one switching.
4. Insert the terminal time, t_f, into the design variable vector ***P*** (see Eq. 3.12).
5. Discretize each control variable into the number of switchings plus one. Then the discrete control vector, ***S***, defined in Eq. (3.11) can be added to the design variable vector ***P*** and the corresponding upper and lower bounds be limited by the original bounds of the controls.
6. Solve the problem by applying the mixed integer NLP method, and then find the optimal discrete-type control trajectories.

A third-order system shown in Section 5.5.1 is used to demonstrate the processes of this numerical scheme.

5.5 Illustrative Examples

The numerical results for the following examples are obtained on an Intel Celeron 1.2 GHz computer with 512 MB of RAM memory. The AOCP is coded in FORTRAN, and C language is used to implement the enhanced branch-and-bound method. The Visual C++ 5.0 and Visual FORTRAN 5.0 installed in a Windows 2000 operating system are adopted to compile the corresponding programs. The total CPU times for solving the F-8 fighter craft problem in Phase I and Phase II are 3.605 and 1.782 seconds, respectively.

5.5.1 Third-Order System

The following system of differential equations is a model of the third-order system dynamics taken from Wu (1999).

$$\dot{x}_1 = x_2\,, \tag{5.9}$$

$$\dot{x}_2 = x_3\,, \tag{5.10}$$

$$\dot{x}_3 = -10x_3 + 10u\,. \tag{5.11}$$

The problem here is to find the control $|u| \le 10$ in order to bring the system from the initial state $[-10, 0, 0]^T$ to the final state $[0, 0, 0]^T$ in minimum time.

First, this problem is solved directly by the mixed integer NLP method. Assuming four switching times (T_1, T_2, T_3, T_4) and five control arcs have values in the discrete set, $\mathbf{U_d}$: {-10, 10}, the terminal time, t_f, is treated as a design variable, so the design variable vector $\boldsymbol{P}$ can be expressed as $[T_1, T_2, T_3, T_4, t_f, U_{d1}, U_{d2}, U_{d3}, U_{d4}, U_{d5}]^T$. Most notably, the final conditions of the state variables are transferred to the equality constraints. Thus, the TOCP problem becomes one of determining the switching times. Figure 5.3(a) presents the continuous solution obtained by using the AOCP and the discrete solution determined by applying the mixed integer NLP method proposed herein. The results indicate that the control trajectory determined by the mixed integer NLP method is of the bang-bang type and the solution consistent with the results obtained by Wu (1999).

As stated in Section 5.4, several assumptions must be made when the mixed integer NLP method is applied to solving TOCP directly. Unfortunately, these assumptions cannot be guaranteed to hold in practical cases. Consequently, the two-phase scheme proposed in this dissertation is needed. For illustration, the third-order system is again solved using this two-phase scheme. In Phase I, the two switching times are found to be $[0.330, 0.725]^T$ and the terminal time t_f is 0.7864. In the first phase, these switching data need not be accurate because they are only used to help users decide on the number of switching times, the control arcs and their corresponding boundaries. Thus, in Phase II, the design variable vector $\boldsymbol{P}$ is re-formed as $[T_1, T_2, t_f, U_{d1}, U_{d2}, U_{d3}]^T$; the numerical result obtained by applying the mixed integer NLP method is as presented in Figure 5.3(b). In Phase II, the switching times of the discrete control

input are $[0.323, 0.713]^T$, and the terminal time t_f is 0.7813 seconds. The control trajectory also agrees with that obtained by Wu (1999).

5.5.2 Fourth-Order Systems: A Flexible Mechanism

A flexible mechanism was proposed and solved by Wu (1999). The OCP formulation of this problem is to minimize

$$J_0 = t_f \tag{5.12}$$

subject to

$$\begin{aligned} &\dot{x}_1(t) = x_2\,, \\ &\dot{x}_2 = -\frac{k}{m_1}(x_1 - x_3) + \frac{u}{m_1}\,, \\ &\dot{x}_3 = x_4\,, \\ &\dot{x}_4 = \frac{k}{m_2}(x_1 - x_3). \end{aligned} \tag{5.13}$$

control constraints

$$\phi_1 = |u(t)| \le M \tag{5.14}$$

with boundary conditions $\boldsymbol{x}^T(0) = [0, \quad 0, \quad 0, \quad 0]^T$ and $\boldsymbol{x}^T(t_f) = [1, \quad 0, \quad 1, \quad 0]^T$.

With admissible control formulation, the control variables are converted into design variables and the control constraints are treated as the dynamic constraint. In this dissertation, the system is solved by the OCP solver with the following parameters: k = 1.0 N-m-rad^{-1}, $m_1 = m_2$ = 1.0 kg-m^2, and M = 1.0 N-m. The numbers of time-grid points for the control function (NGP) are selected as 5, 11 and 51 to study the effect of coarser or finer mesh. Two initial guesses, u (0) = 0.0 and u(0) = 1.0, for the control function with three piecewise interpolation schemes – zero order, first order, and cubic spline – are used in this problem. The hybrid method that combines the DDM and AVM for design sensitivity analysis is used to calculate the design sensitivity coefficients.

The optimal solution for this problem is given in Table 5.2 and the trajectories of state

variables are shown in Figure 5.4 and Figure 5.5 shows the comparison of control trajectories between Phase I and Phase II. All 18 test runs are successfully solved with the proposed method, but the runs with a small number of control grid points (NGP) give higher optimum values and less CPU time. The terminal time, t_f, and the trajectories obtained in this work agree with the results, $t_f \cong 4.3$, obtained by Wu (1999). The numerical results also show that the proposed method has the capability to deal with the high-order time-optimal control problem.

5.5.3 F-8 Fighter Aircraft

The F-8 fighter aircraft has been considered in several pioneering studies (*e.g.*, Kaya and Noakes, 1996; Banks and Mhana, 1992; Simakov *et al*., 2002) and has become a standard for testing various optimal control strategies. A nonlinear dynamic model of the F-8 fighter aircraft is considered below. The model is represented in state space by the following differential equations:

$$\dot{x}_1 = -0.877x_1 + x_3 - 0.088x_1x_3 + 0.47x_1^2 - 0.019x_2^2 - x_1^2x_3 + 3.846x_1^3 - 0.215u + 0.28x_1^2u - 0.47x_1u^2 + 0.63u^3, \tag{5.15}$$

$$\dot{x}_2 = x_3, \tag{5.16}$$

$$\dot{x}_3 = -4.208x_1 - 0.396x_3 - 0.47x_1^2 - 3.564x_1^3 - 20.967u + 6.265x_1^2u + 46x_1u^2 + 61.4u^3, \tag{5.17}$$

where x_1 is the angle of attack in radians, x_2 is the pitch angle, x_3 is the pitch rate and the control input u represents the tail deflection angle. For convenience of comparison, the standard settings (Kaya and Noakes, 1996; Lee *et al*., 1997) are used. A control $|u| \le 0.05236$ must be found that brings the system from its initial state $\left[26.7\pi/180,\ \ 0,\ \ 0\right]^T$ to the final state $\left[0,\ \ 0,\ \ 0\right]^T$ in minimum time.

When the two-phase scheme is applied, as described in Section 5.4, the switching times computed in Phase I are 0.115, 2.067, 2.239, 4.995, and 5.282, and the terminal time is t_f = 5.7417. These switching data are used to set the design variables and their corresponding bounds, and then the problem is solved by the mixed integer NLP method. Finally, the switching times for the discrete control input are 0.098, 2.027, 2.199, 4.944, and 5.265, and the terminal time t_f is 5.74216. Figure 5.6 shows the comparison of the controls between Phase I and Phase II, while Figure 5.7 shows the trajectories of the states and the control of Phase I and Phase II. This example is also solved by Kaya and Noakes (1996) using the switching time computation method and by Lee *et al.* (1997) using the Control Parameterization Enhancing Transform (CPET) method. Table 5.1 shows the terminal time t_f, switching times and the accuracy of terminal constraints computed by various methods for this problem. According to the numerical results, the two-phase scheme provides a better solution, and the accuracy of the terminal constraints is acceptable.

5.6 Summary

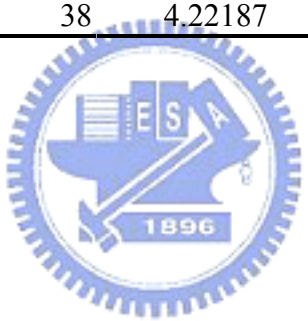
This chapter has proposed a novel numerical method for solving time-optimal control problems with discrete-type control inputs that include the bang-bang type most commonly encountered when the control is bounded. This two-phase computational scheme for finding a discrete optimal control for time-optimal control problems is novel because its discrete control can be more easily implemented than continuous control in practical applications. A simple example, a third-order system, was presented to demonstrate the usage of the proposed scheme. A flexible mechanism control problem and an F-8 fighter aircraft control problem were also considered and solved by application of the proposed scheme. Numerical results were obtained efficiently and accurately and provide evidence that the two-phase scheme constitutes a viable method for solving time-optimal control problems with discrete-valued controls.

Table 5.1 Results of various methods for the F-8 fight aircraft problem.

Method	t_f	Switching Times	Accuracy of Terminal Constraints
STC (Kaya and Noakes, 1996)	6.3867	0.0761, 5.4672, 5.8241, 6.3867	$\leq 10^{-5}$
CPET (Lee *et al*., 1997)	6.0350	2.188, 2.352, 5.233, 5.563	$\leq 10^{-10}$
Two-phase scheme	5.7422	0.098, 2.027, 2.199, 4.944, 5.265	$\leq 10^{-10}$

Table 5.2 Optimal results for the fourth-order system.

$u(t_0)$	NGP	INTP	NIT	J_0^*	Conv.Par.	CPU
0.0	5	Zero	5	4.33196	1.04E-05	0.131
		First	5	4.86764	5.69E-06	0.07
		Cubic	5	4.90565	4.67E-07	0.06
	11	Zero	15	4.30699	5.71E-09	1.382
		First	7	4.28066	5.65E-06	0.35
		Cubic	10	4.30041	1.52E-08	0.34
	51	Zero	50	4.26239	1.38E-07	43.803
		First	44	4.22087	3.50E-07	18.596
		Cubic	40	4.22187	1.10E-08	12.659
1.0	5	Zero	6	4.33197	6.16E-07	0.12
		First	7	4.86765	2.25E-07	0.091
		Cubic	9	4.90560	3.72E-05	0.12
	11	Zero	7	4.36249	3.51E-08	0.651
		First	8	4.28064	1.09E-05	0.43
		Cubic	10	4.30041	3.50E-07	0.39
	51	Zero	49	4.26229	7.21E-08	42.872
		First	42	4.22087	2.20E-06	17.315
		Cubic	38	4.22187	2.88E-06	11.847



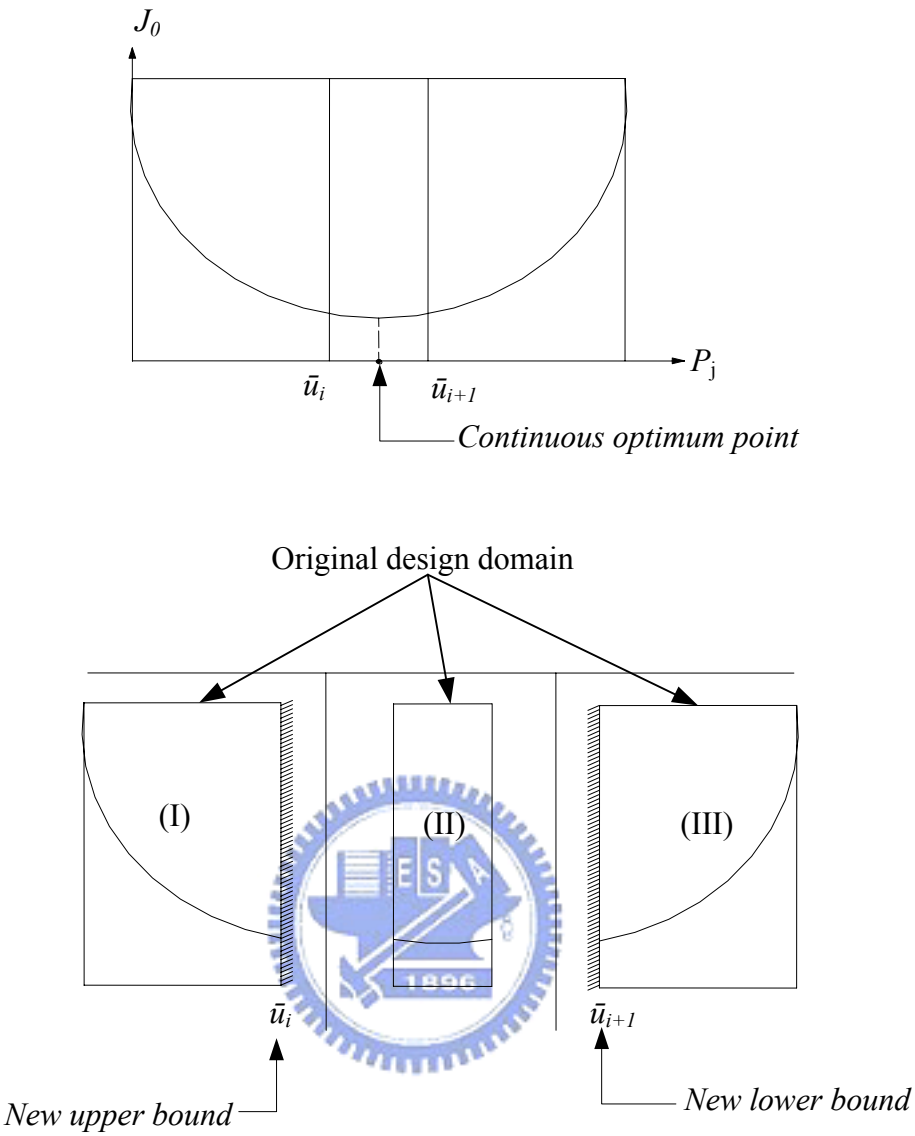


Figure 5.1 Conceptual layout of the branching process.

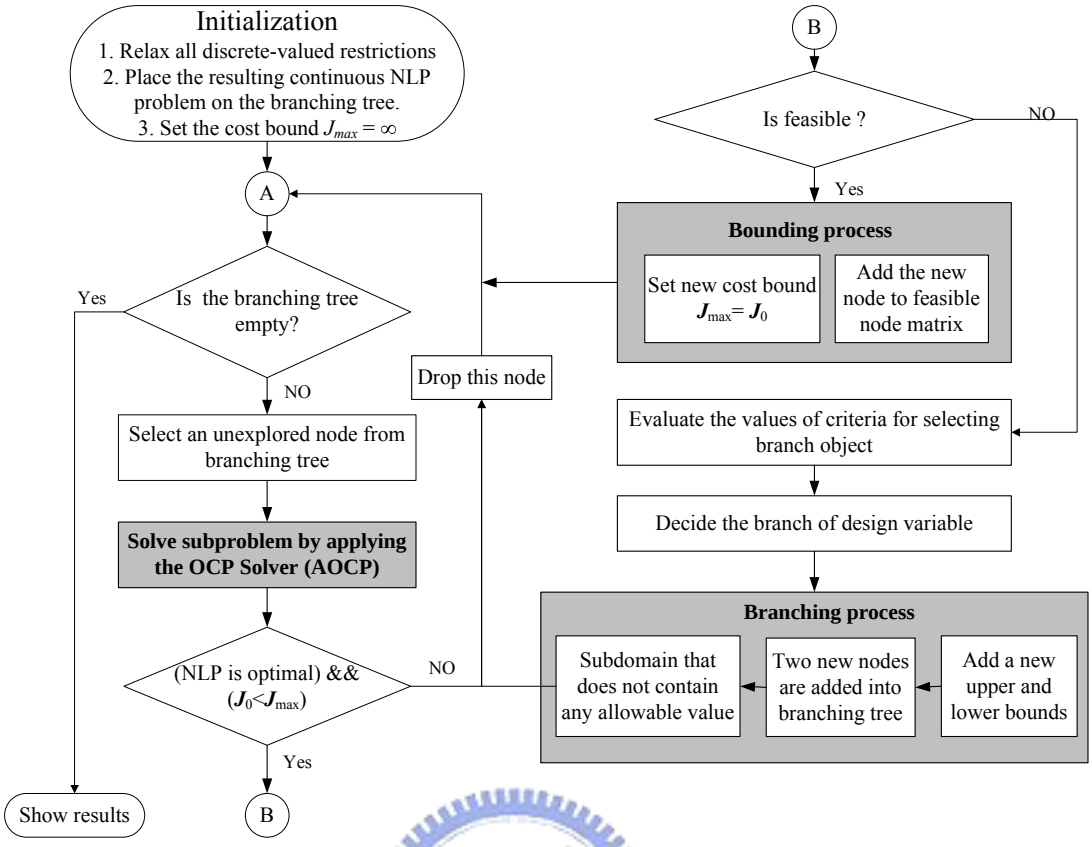
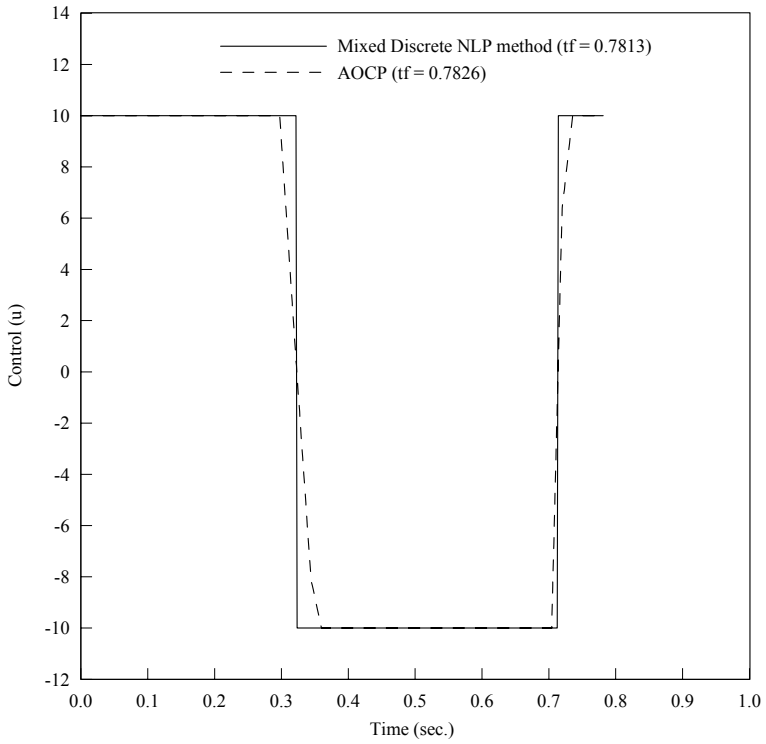
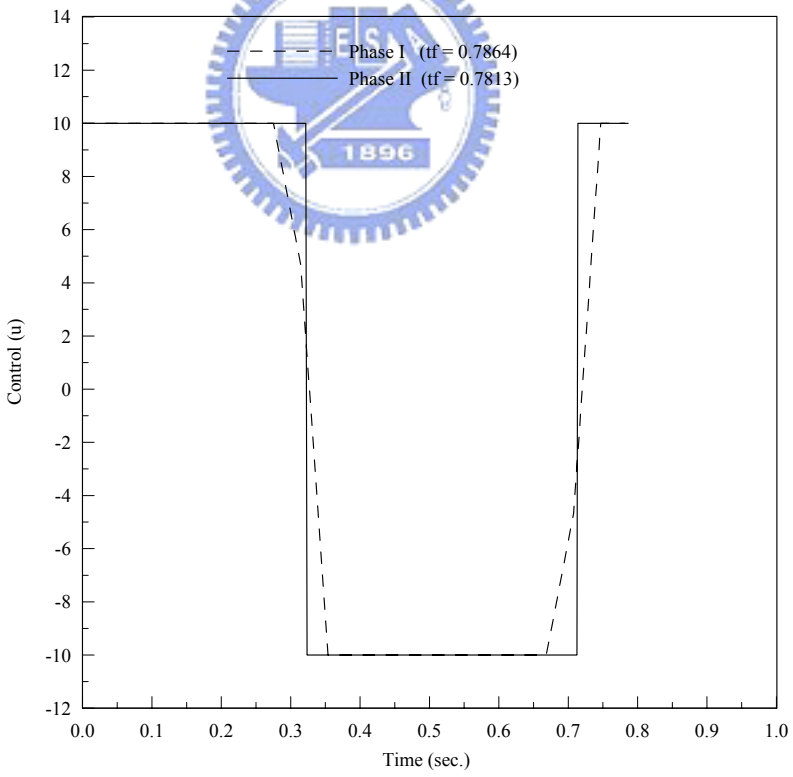


Figure 5.2 Flow chart of the algorithm for solving discrete-valued optimal control problems.

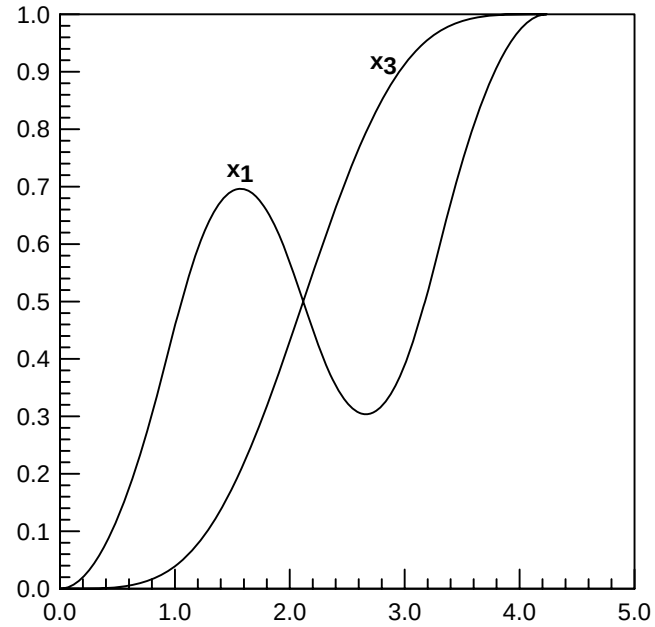


(a) AOCP vs. a mixed-integer NLP method.

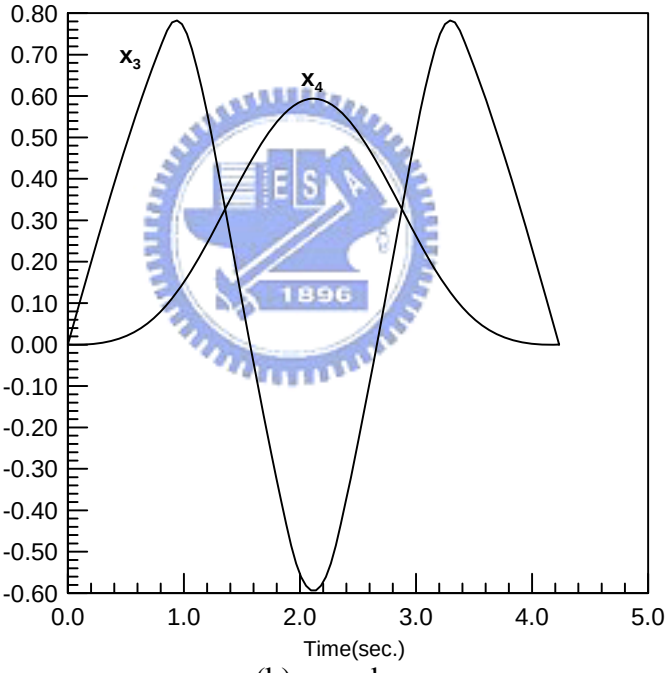


(b) Phase I vs. Phase II.

Figure 5.3 Control trajectories for the third-order system.



(a) x_1 and x_3



(b) x_2 and x_3

Figure 5.4 State trajectories for the fourth-order system (Phase II).

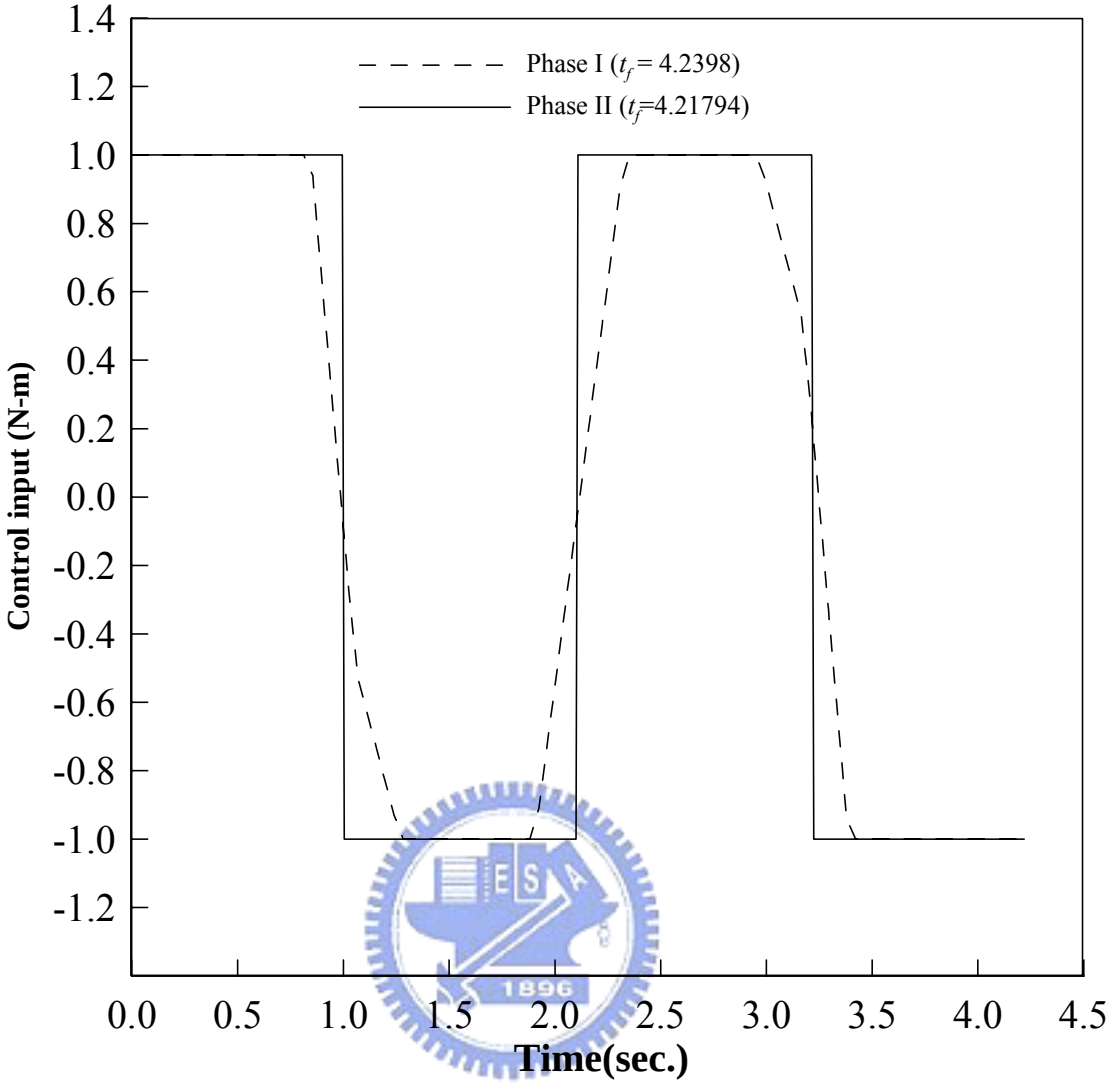


Figure 5.5 Control trajectories for the fourth-order system.

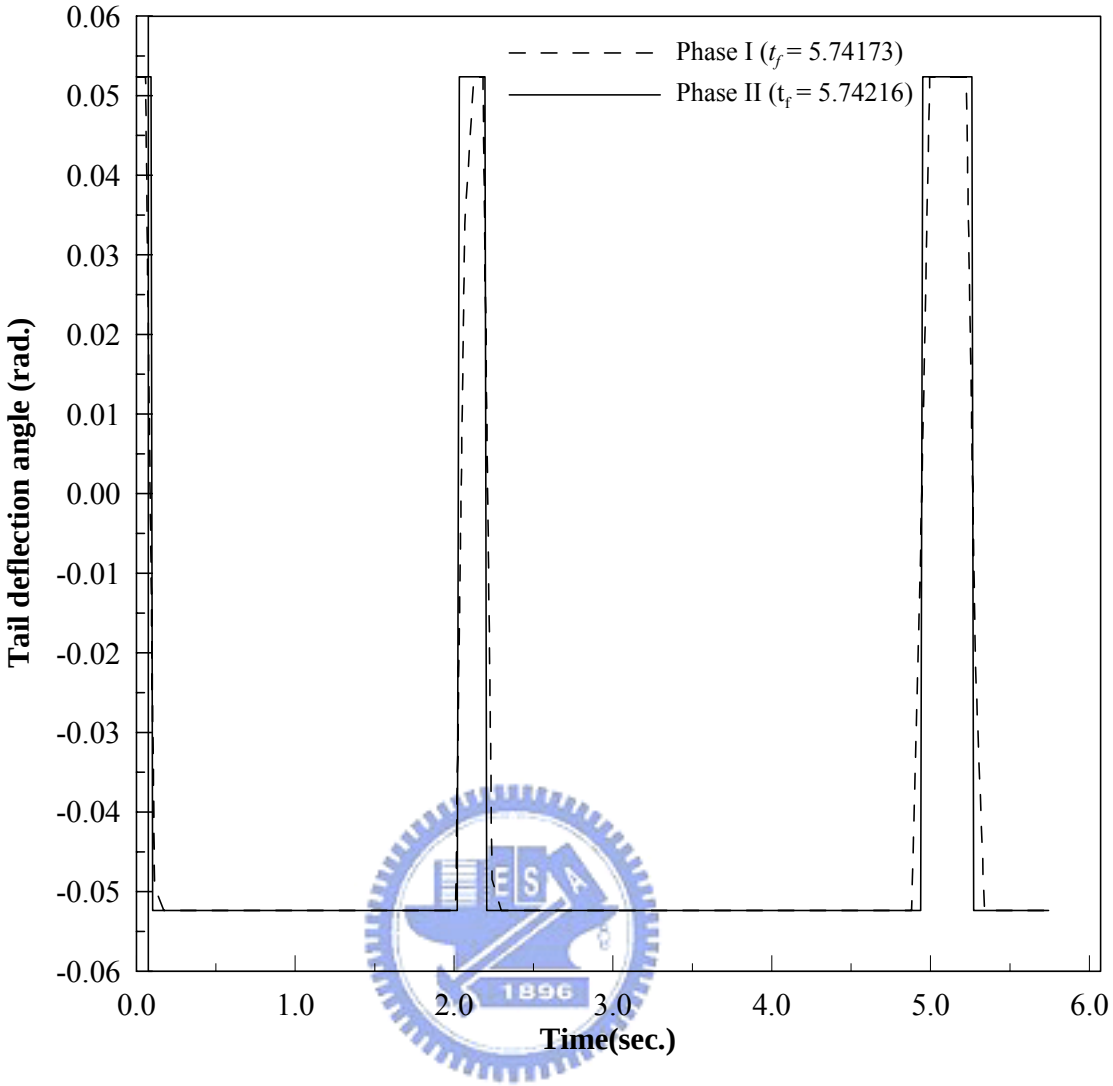
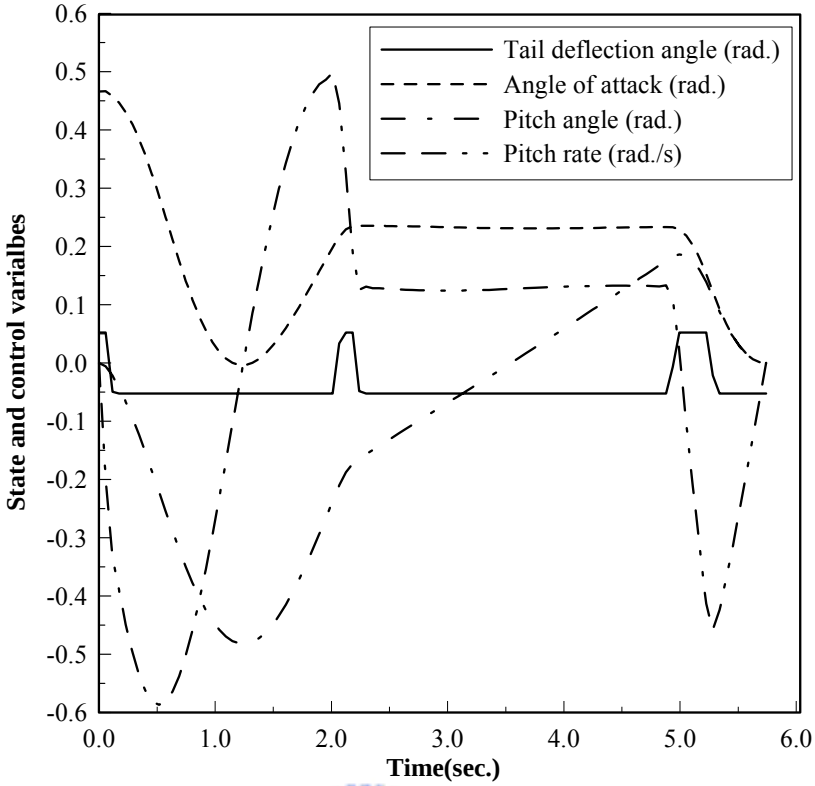
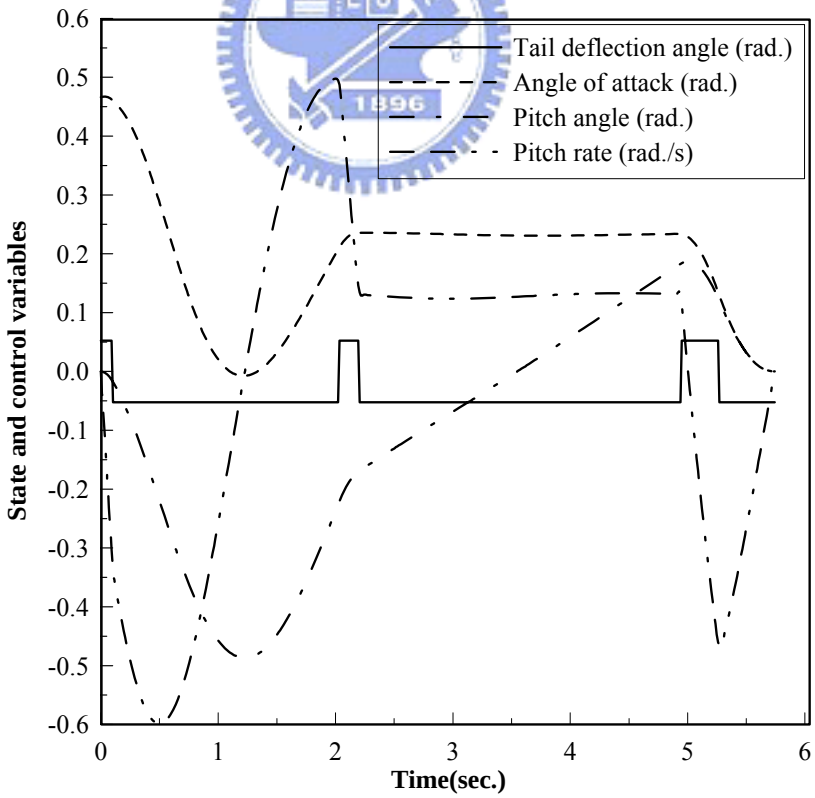


Figure 5.6 Control trajectories for the F-8 fighter aircraft.



(a) Phase I.



(b) Phase II.

Figure 5.7 Trajectories of the states and control input for the F-8 fighter aircraft.

CHAPTER 6
ENGINEERING APPLICATIONS

6.1 Flight Level Control Problem

The flight level tracking that plays an important role in autopilot systems has received considerable attentions from many researchers (Lygeros, 2003; Lygeros *et al.*, 1999; Cook, 1997; Tomlin *et al.*, 1996; Etkin and Redi, 1996). A commercial aircraft's cruising altitude is typically assigned a flight level by air traffic control (ATC). To ensure aircraft separation, each aircraft has its own flight level separated by a few hundred feet; however, changes in flight level do happen occasionally and must be cleared by ATC. At all other times, the aircraft crew must ensure that they remain within the allowed bounds of their assigned level. At the same time, they must also maintain limits on factors such as speed, flight path angle, and acceleration imposed by limitations of airframe and engine and passenger comfort requirements or to avoid dangerous situations such as aerodynamic stall. In this paper, the flight level tracking problem is formulated into an optimal control problem. For safety reasons, the speed of the aircraft and the flight path angle must be kept within a safe "aerodynamic envelope" (Tomlin *et al.*, 1996) that can be translated into the dynamic constraints of the optimal control problem. A flight level tracking problem and a minimum time problem are outlined in the following sections and then solved using the proposed solver.

6.1.1 Aircraft Model

Much ATC research (*e.g.*, Cook, 1997; Etkin and Redi, 1996) has applied a point mass model to describe aircraft motion, considering only aircraft movement in a lateral direction. In Figure 6.1, three coordinate frames are used to describe aircraft motion: X_g-Y_g denotes the ground frame; X_b-Y_b, the body frame; and X_w-Y_w, the wind frame. In addition, θ, γ, and α denote the rotation angle between the frames; $V \in \mathbb{R}$ represents the speed of the aircraft, which is aligned with the positive X_w direction; and h is the aircraft's altitude.

The equations of the motion can be derived from the force balance relationships:

$$\begin{aligned} & m\dot{V} = T\cos\alpha - D - mg\sin\gamma, \\ & and \;\; mV\dot{\gamma} = L + T\sin\alpha - mg\cos\gamma, \end{aligned} \tag{6.1}$$

where T is the thrust exerted by the engine, D is the aerodynamic drag, and L is the aerodynamic lift. By applying basic aerodynamics, the lift (L) and drag (D) can be approximated by

$$\begin{aligned} & L = \frac{C_L s\rho V^2}{2}(1+c\alpha) = a_L V^2(1+c\alpha), \\ & and \quad D = \frac{C_D s\rho V^2}{2} = a_D V^2, \end{aligned} \tag{6.2}$$

where C_L, C_D, and c are dimension-less lift and drag coefficients, s is the wing surface area and ρ is the air density.

According to the admissible optimal control formulation described in Section 3.4, the air model can be formulated by a three-state model with a state variable vector $\boldsymbol{x}(t) = [x_1, x_2, x_3]^T = [V, \gamma, h]^T$ and a control input vector $\boldsymbol{u}(t) = [u_1, u_2]^T = [T, \theta]^T$. By approximating α with a small angle, the equations of the motion (system equations) can be written as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -\frac{a_D}{m}x_1 - g\sin x_2 \\ \frac{a_L}{m}x_1(1-cx_2) - g\frac{\cos x_2}{x_1} \\ x_1\sin x_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{m} & 0 \\ 0 & \frac{a_L}{m}x_1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{6.3}$$

This model, proposed by Lygeros *et al.* (1999) and adopted here, extends the three dimensions of an aerodynamic envelope protection problem. Taking into the consideration of safety conditions, the aircraft speed and flight path angle are bounded in a rectangular limitation called a "*safe aerodynamic envelop*." Following Tomlin *et al.* (1996), Lygeros (2003) proposed a simplified aerodynamic envelope that is adopted in this paper and translated into the following dynamic constraints:

$$
\begin{aligned}
&V_{\min} \le x_1 \le V_{\max}\,, \\
&\gamma_{\min} \le x_2 \le \gamma_{\max}\,, \\
&h_{\min} \le x_3 \le h_{\max}\,,
\end{aligned}
\tag{6.4}
$$

Based on the NLP formulation described in Section 2.2, these constraints can be treated as dynamic constraints and rewritten as follows:

$$
\begin{aligned}
&\phi_1 : -x_1 + V_{\min} \le 0\,, \quad \phi_2 : x_1 - V_{\max} \le 0\,, \\
&\phi_3 : -x_2 + \gamma_{\min} \le 0\,, \quad \phi_4 : x_2 - \gamma_{\max} \le 0\,, \\
&\phi_5 : -x_3 + h_{\min} \le 0\,, \quad \phi_6 : x_3 - h_{\max} \le 0\,,
\end{aligned}
\tag{6.5}
$$

To illustrate the capabilities of the proposed method, the flight level tracking problem and the minimum time problem have been chosen.

Case I: Flight level tracking problem

This tracking problem is to find the controls that will maintain the system state $\boldsymbol{x}(t)$ as close as possible to the desired state $\boldsymbol{r}(t)$ in the interval $[t_0, t_f]$. The performance index for the tracking problem can be written as

$$
J_0 = \int_{t_0}^{t_f} \left\| \boldsymbol{x}(t) - \boldsymbol{r}(t) \right\|_{\boldsymbol{Q}(t)}^2 dt \tag{6.6}
$$

where $\boldsymbol{Q}(t)$ is a real symmetric $n \times n$ matrix that is positive semi-definite for all $t \in \left[t_0,\ \ t_f \right]$. The flight level tracking problem involves keeping the aircraft as near as possible to the desired level and aircraft speed. Therefore, the performance index can be represented as

$$
J_0 = \frac{1}{2} \int_{t_0}^{t_f} \left[\left(x_1 - x_{1d} \right)^2 + \left(x_2 - x_{2d} \right)^2 + \left(x_3 - x_{3d} \right)^2 \right] dt \tag{6.7}
$$

where $x_{1\mathrm{d}}$ is the desired aircraft speed, $x_{2\mathrm{d}}$ is desired flight path angle and $x_{3\mathrm{d}}$ is the assigned altitude.

Case II: Minimum time problem

The minimum time problem is to transfer a system from an arbitrary initial state $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ to a specified target set $\mathbf{S_t}$ in minimum time. The performance index for the minimum time

problem can be written as

$$J_0 = t_0 - t_f = \int_{t_0}^{t_f} dt \tag{6.8}$$

where t_f is the first instant of time when $\boldsymbol{x}(t)$ and $\mathbf{S_t}$ intersect. In some emergencies, the aircraft crew is asked to change their level as soon as possible.

6.1.2 Numerical examples

The following parameters, outlined here for case I, are used in both cases:

a_L = 65.3 Kg/m,	a_D = 3.18 Kg/m,	$m = 160\times10^3$ Kg,
g = 9.81 m/s^2,	θ_{min} = -20°,	γ_{min} = -20,
c = 6,	θ_{max} = 25°,	γ_{max} = 25,
$T_{min} = 60\times10^3$ N,	$T_{min} = 120\times10^3$ N,	V_{min} = 92 m/s,
V_{max} = 170 m/s,	h_{min} = -150 m,	h_{max} = 150 m

The initial values of the state variables are

$$\boldsymbol{x}_0 = [100, 20, -120]^T \tag{6.9}$$

and the purpose of this problem is to find a suitable control for maintaining the flight level and keeping the aircraft altitude at the assigned level. Thus the desired states are set with following values

$$\boldsymbol{r}(t) = [150, 0, 0]^T. \tag{6.10}$$

In addition to the dynamic constraints proposed in Eq. (6.5), the control inputs are also limited within the following bounds:

$$\begin{aligned} &T_{min} \le u_1 \le T_{max}\ , \\ &and \quad \theta_{min} \le u_2 \le \theta_{max}\ . \end{aligned} \tag{6.11}$$

Substituting these parameters into Eqs. (6.3) and (6.7), the flight tracking problem is solved by the OCP solver. The numerical results are shown in Figure 6.2. As shown in Figure 6.2(a), all states meet the constraints, and the flight level and aircraft speed return to the

desired states. Table 6.1 shows the user subroutines for this case. Obviously, the OCP solver provides an easily usable tool for solving dynamic optimization problems.

Case II: Minimum time problem

In this problem, the aircraft crew is asked to increase their altitude in minimum time. The initial and final altitude are h_0 = 0 m and h_f = 500 m, respectively. All constraints imposed on case I are also imposed on this case. The initial state $\boldsymbol{x}_0 = [100, 0, 0]^T$, the final time, t_f, obtained by using the AOCP, is 73.98 seconds, and the final altitude is 499.928 m. The control histories shown in Figure 6.3(a) and (b) give the state trajectories, which, as the figure illustrates, all fall within the safe "aerodynamic envelope" (*i.e.*, meet the dynamic constraints).

6.2 Vehicle Suspension Design Problem

Many studies have treated the vehicle as a dynamic system, starting with the basic properties of vehicle suspension, the stiffness and damping coefficients (Gillespie, 1992). Thus the design of vehicle suspension systems has received much attention in the automotive industry. Numerous researchers have examined semi-active and active vibration isolation for suspension systems. Yet, despite recent advances in active and semi-active suspension technology, vehicles with passive suspension systems still dominate current car production. Tools must therefore be made available to vehicle designers for optimizing passive suspension systems.

The model described here is a half-car model that allows independent vertical inputs to the front and rear wheels and can thus simulate pitching and bouncing motions due to road inputs. Two longitudinal forces, which can be positive to represent traction or negative to represent braking, are applied to the front and rear axles to simulate the effects of vehicle acceleration or deceleration. Cases of braking and accelerating while moving straight ahead are used to validate the longitudinal vehicle dynamics.

An optimal design problem in relation to vehicle suspension is considered to maximize vehicle ride performance, which may be evaluated according to passenger discomfort. The response to driver's seat to acceleration is commonly used as the objective of suspension design. Three road profiles that excite pitch and bounce motions at a constant vehicle speed are used to calculate the optimal suspension parameters. In this optimal design problem, the objective is to minimize the extreme acceleration of the driver's seat under a number of constraints on the dynamic response and the design parameters. The optimal design of a vehicle suspension system can be applied in diverse fields of research, including traction force control, speed control, braking system design, to name a few. In this dissertation, an emergency stop – a special case of vehicle speed control problem – is treated as a time-optimal control problem and solved by the proposed AOCP method.

6.2.1 Derivation of the Vehicle Model

Half-car model

Although the quarter-car model has been commonly used in assessing vehicle ride performance, it does not fully represent the rigid body motions that a motor vehicle may exhibit. For example, the quarter-car model disregards pitching motions, which may be important, particularly when the car travels over obstructions like road bumps and potholes. Moreover, the quarter-car model is a multi-input system that responds with both pitch motions and vertical bounce because of the longitudinal distance between the axles. These pitch and bounce motions must be understood because they provide useful information on vertical and longitudinal vibrations. As a result of these quarter-car limitations, half-car and full-car models are used in several studies on suspension. Figure 6.4 depicts a nonlinear half-car model with six degrees of freedom, modified from the model of Haug and Arora (1979). Two additional longitudinal forces, traction or braking forces, are applied to the axles, allowing the vehicle to be accelerated or decelerated. Shock absorbers are assumed to be rigidly joined to

the chassis without displacement or deflection in the longitudinal direction. Based on this assumption, the longitudinal forces only change the speed and pitch angle of vehicle. The governing equations for the vehicle can be derived from Lagrange's equations

$$\frac{d}{dt}\left[\frac{\partial \mathrm{T}}{\partial \dot{\mathbf{z}}_i}\right]-\frac{\partial \mathrm{T}}{\partial \mathbf{z}_i}+\frac{\partial \mathrm{V}}{\partial \mathbf{z}_i}-Q_i=0\,,\ \mathrm{i}=1,\ldots,6 \tag{6.12}$$

where T and V represent the system's kinetic and potential energy, and Q_i represent nonconservative generalized forces. In Figure 6.4, the kinetic energy of the system can be expressed as

$$T=\frac{1}{2}m_1\dot{z}_1^2+\frac{1}{2}m_2\dot{z}_2^2+\frac{1}{2}m_3\dot{z}_3^2+\frac{1}{2}m_4\dot{z}_4^2+\frac{1}{2}m_5\dot{z}_5^2+\frac{1}{2}m_6\dot{z}_6^2 \tag{6.13}$$

The potential energy V of the conservative forces is

$$\begin{aligned}V=&\frac{1}{2}k_1(z_2+\frac{L}{12}z_3-z_1)^2+\frac{1}{2}k_2(z_4-z_2-\frac{L}{3}z_3)^2\\&+\frac{1}{2}k_3(z_5-z_2+\frac{2L}{3}z_3)^2+\frac{1}{2}k_4(z_4-R_1(z_6))^2\\&+\frac{1}{2}k_5(z_5-R_2(z_6))^2\end{aligned} \tag{6.14}$$

and the virtual work done by the nonconservative forces is

$$\begin{aligned}\delta w=&-c_1(\dot{z}_2+\frac{L}{12}\dot{z}_3-\dot{z}_1)(\delta z_2+\frac{L}{12}\delta z_3-\delta z_1)\\&-c_2(\dot{z}_4-\dot{z}_2-\frac{L}{3}\dot{z}_3)(\delta z_4-\delta z_2-\frac{L}{3}\delta z_3)\\&-c_3(\dot{z}_5-\dot{z}_2+\frac{2L}{3}\dot{z}_3)(\delta z_5-\delta z_2+\frac{2L}{3}\delta z_3)\\&-c_4(\dot{z}_4-\dot{R}_1)\delta z_4-c_5(\dot{z}_5-\dot{R}_2)\delta z_5+F_t\delta z_6+F_tH\delta z_3\\\equiv&\sum_{i=1}^{6}Q_i\delta z_i\end{aligned} \tag{6.15}$$

where $F_t = F_f + F_r$ is the total traction/braking force imposed on the vehicle. From Eq. (6.12), the system equations describing the motion of half-vehicle model can be derived as follows

$$m_1\ddot{z}_1+c_1\dot{z}_1-c_1\dot{z}_2-\frac{L}{12}c_1\dot{z}_3+k_1z_1-k_1z_2-\frac{L}{12}k_1z_3=0\,, \tag{6.16}$$

$$\begin{aligned}&m_2\ddot{z}_2+c_1\dot{z}_1+\dot{z}_2(c_1+c_2+c_3)+\dot{z}_3(\frac{L}{12}c_1+\frac{L}{3}c_2-\frac{2L}{3}c_3)\\&-c_2\dot{z}_4-c_3\dot{z}_5-k_1z_1+z_2(k_1+k_2+k_3)+z_3(\frac{L}{12}k_1+\frac{L}{3}k_2-\frac{2L}{3}k_3)\end{aligned} \tag{6.17}$$

$$-k_2 z_4 - k_3 z_5 = 0\,,$$

$$\begin{aligned} I\,\ddot{z}_3 &- \frac{L}{12}c_1\dot{z}_1 + \dot{z}_2\left(\frac{L}{12}c_1 + \frac{L}{3}c_2 - \frac{2L}{3}c_3\right) + \dot{z}_3\left(\frac{L^2}{144}c_1 + \frac{L^2}{9}c_2 - \frac{4L^2}{9}c_3\right) \\ &- \frac{L}{3}c_2\dot{z}_4 + \frac{2L}{3}c_3\dot{z}_5 - \frac{L}{12}k_1 z_1 + z_2\left(\frac{L}{12}k_1 + \frac{L}{3}k_2 - \frac{2L}{3}k_3\right) \\ &+ z_3\left(\frac{L^2}{144}k_1 + \frac{L^2}{9}k_2 + \frac{4L^2}{9}k_3\right) - \frac{L}{3}k_2 z_4 + \frac{2L}{3}k_2 z_5 = F_t H\,, \end{aligned} \tag{6.18}$$

$$\begin{aligned} m_4\ddot{z}_4 &+ c_2\dot{z}_2 - \frac{L}{3}c_2\dot{z}_3 + \dot{z}_4(c_2 + c_4) - k_2 z_2 - \frac{L}{3}k_2 z_3 + z_4(k_2 + k_4) \\ &= k_4 R_1(z_6) + c_4\dot{R}_1(z_6)\,, \end{aligned} \tag{6.19}$$

$$\begin{aligned} m_5\ddot{z}_5 &- c_3\dot{z}_2 + \frac{2L}{3}c_3\dot{z}_3 + \dot{z}_5(c_3 + c_5) - k_3 z_2 + \frac{2L}{3}k_3 z_3 + z_5(k_3 + k_5) \\ &= k_5 R_2(z_6) + c_5\dot{R}_2(z_6)\,, \end{aligned} \tag{6.20}$$

$$\text{and}\quad (m_1 + m_2 + m_4 + m_5)\,\ddot{z}_6 = F_t\,. \tag{6.21}$$

where m_i represents the masses of the seat and driver, the main body, and the wheel and axles, respectively. The parameters k_i and c_i represent the known stiffness and damping coefficients of the suspension system. The moment of main body inertia about its center of mass is denoted as I, while H is the vertical distance from the center of gravity (C.G.) to the ground, and L is the total length of the wheel base. The functions $R_1(y)$ and $R_2(y)$ represent displacements of the front and rear wheels, caused by undulations of the road surface on which the vehicle is traveling. Once $z_{6+i} = \dot{z}_i, i = 1,...,6$ is defined, the vehicle system can be transformed into a state-space equation of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{W} \tag{6.22}$$

where $\boldsymbol{x}(t) = [z_1, z_2, z_3, \ldots, z_{12}]^{\mathrm{T}}$ represents the vector of state variables and the nonzero elements of matrices ***A, B*** and ***W*** are given as follows:

$$A_{1,7}=1 \qquad A_{2,8}=1$$
$$A_{3,9}=1 \qquad A_{4,10}=1$$
$$A_{5,11}=1 \qquad A_{6,12}=1$$
$$A_{7,2}=k_1/m_1 \qquad A_{7,1}=-k_1/m_1$$
$$A_{7,7}=-c_1/m_1 \qquad A_{7,3}=Lk_1/(12m_1)$$
$$A_{7,9}=Lc_1/(12m_1) \qquad A_{7,8}=c_1/m_1$$
$$A_{8,2}=-(k_1+k_2+k_3)/m_2 \qquad A_{8,1}=k_1/m_2$$
$$A_{8,4}=k_2/m_2 \qquad A_{8,3}=-L(k_1+4k_2-8k_3)/(12m_2)$$
$$A_{8,7}=c_1m_2 \qquad A_{8,5}=k_3/m_2$$
$$A_{8,9}=-L(c_1+4c_2-8c_3)/(12m_2) \qquad A_{8,8}=-(c_1+c_2+c_3)/m_2$$
$$A_{8,11}=c_3/m_2 \qquad A_{8,10}=c_2/m_2$$
$$A_{9,2}=-L(k_1+4k_2-8k_3)/(12I) \qquad A_{9,1}=Lk_1/(12I)$$
$$A_{9,4}=Lk_2/(3I) \qquad A_{9,3}=-L^2(k_1+16k_2+64k_3)/(144I)$$
$$A_{9,7}=Lc_1/(12I) \qquad A_{9,5}=-2Lk_3/(3I)$$
$$A_{9,9}=-(c_1+16c_2+64c_3)L^2/(144I) \qquad A_{9,8}=-(c_1+4c_2-8c_3)L/(12I)$$
$$A_{9,11}=-2Lc_3/(3I) \qquad A_{9,10}=Lc_2/(3I)$$
$$A_{10,3}=Lk_2/(3m_4) \qquad A_{10,2}=k_2/m_4$$
$$A_{10,8}=c_2/m_4 \qquad A_{10,4}=-(k_2+k_4)/m_4$$
$$A_{10,10}=-(c_2+c_4)/m_4 \qquad A_{10,9}=Lc_2/(3m_4)$$
$$A_{11,3}=-2Lk_3/(3m_5) \qquad A_{11,2}=k_3/m_5$$
$$A_{11,8}=c_3/m_5 \qquad A_{11,5}=-(k_3+k_5)/m_5$$
$$A_{11,11}=-(c_3+c_5)/m_5 \qquad A_{11,9}=-2Lc_3/(3m_5)$$
$$B_9=(m_1+m_2+m_4+m_5)H/I \qquad B_{12}=1/(m_1+m_2+m_4+m_5)$$
$$W_{10}=\left[k_4R_1(y)+c_4\dot{R}_1(y)\right]/m_4 \qquad W_{11}=\left[k_5R_2(y)+c_5\dot{R}_2(y)\right]/m_5$$

For safety and comfort, six dynamic constraints are imposed on the system, whose constraint equations may be written as

$$\left|\ddot{z}_1(t)\right|\le\theta_1,\quad 0\le t\le t_f \tag{6.23}$$

$$\left|z_2(t)+\frac{L}{12}z_3(t)-z_1(t)\right|\le\theta_2,\quad 0\le t\le t_f \tag{6.24}$$

$$\left|z_4(t)-z_2(t)-\frac{L}{3}z_3(t)\right|\le\theta_3,\quad 0\le t\le t_f \tag{6.25}$$

$$\left| z_5(t) - z_2(t) + \frac{2L}{3} z_3(t) \right| \le \theta_4, \quad 0 \le t \le t_f \tag{6.26}$$

$$\left| z_4(t) - R_1(t) \right| \le \theta_5, \quad 0 \le t \le t_f \tag{6.27}$$

$$\left| z_5(t) - R_2(t) \right| \le \theta_6, \quad 0 \le t \le t_f \tag{6.28}$$

$$z_{12}(t) \le v_{allow}, \quad 0 \le t \le t_f \tag{6.29}$$

where θ_2 to θ_6 are the maximum allowable displacements and v_{allow} is the maximum allowable speed.

Road surface displacement function

Because the dynamic response depends strongly on the vertical displacement history of the wheels on the road surface, the input road conditions are very important. Most data used in establishing the ride comfort criteria were obtained using sinusoidal inputs. Thus the road surface displacement function plotted in Figure 6.5 is defined as a sinusoidal undulation with amplitude $\boldsymbol{x}_0$ and variable half-wavelength l_i (Haug and Arora, 1979). The front tire displacement $\bar{v}(y)$ at position y is thus defined as

$$\bar{v}(y) = \begin{cases} x_0 \left[1 - cos \dfrac{\pi (y - y^{i-1})}{l_i} \right], & y^{i-1} \le y \le y^i, \quad i \ \ is \ \ odd \\ x_0 \left[1 + cos \dfrac{\pi (y - y^{i-1})}{l_i} \right], & y^{i-1} \le y \le y^i, \quad i \ \ is \ \ even \end{cases} \tag{6.30}$$

where y is a coordinate measured along the road and $y^i = \sum_{j=1}^{i} l_j$. The vertical displacement function for the front wheel can therefore be defined as

$$R_1(y) = \begin{cases} \bar{v}(y), & 0 \le y \le y_t \\ 0, & otherwise \end{cases} \tag{6.31}$$

where y_t is the final position of the road undulation. The vertical displacement of the rear wheel has the same value as that of the front wheel but with a wheelbase lag. Therefore,

$$R_2(y) = R_1(y - L) \tag{6.32}$$

where $R_1(y)$ is defined in Eq. (6.31).

6.2.2 Numerical Examples

This paper uses the numerical data from Haug and Arora (1979) to validate the model specified in Section 6.2.1. The following parameters in the vehicle system equations are fixed during the calculations; m_1g = 290 lb, m_2g = 4500 lb, $m_4g = m_5g$ = 96.6 lb, I = 41,000 lb-in-sec^2, H = 20 in, L = 120 in, $k_4 = k_5$ = 1500 lb / in, v_{allow} = 1056 in / sec (60 mph), and $c_4 = c_5$ = 5 lb-sec / in. The coefficients of the suspension system are selected as design variables, $\boldsymbol{b} = [k_1, k_2, k_3, c_1, c_2, c_3]^T$. The lower and upper bounds on $\boldsymbol{b}$ are $[50, 200, 200, 2, 5, 5]^T$ and $[500, 1000, 1000, 50, 80, 80]^T$, respectively. The maximum allowable values for the state variable constraints in Eqs. (6.23) – (6.29) are selected to be $[400, 2, 5, 5, 2, 2, 1056]^T$. The units of z_1, z_2, z_4, z_5 and z_6 are inches and those of z_3 are radians.

Model validation

The physical phenomenon of rigid body motion can be used to confirm the correctness of the vehicle model. Therefore, cases of braking and accelerating while traveling straight ahead are considered here to validate the longitudinal vehicle dynamics. For convenience of observation, the vehicle is assumed to travel along a straight path such that $R_1(y) = R_2(y) = 0$. In cases of acceleration, the control problem is to determine a feasible acceleration trajectory along which a vehicle with various initial speeds can arrive at a destination in minimal time. Hence, one additional terminal constraint is imposed:

$$z_6(t_f) = y_t \tag{6.33}$$

where y_t is the destination. Similarly, one additional terminal constraint is included in cases of braking

$$z_{12}(t_f) = 0 \tag{6.34}$$

All the acceleration and braking test cases are transformed into time-optimal control

problems and solved by applying the proposed NLP method. Figure 6.6 and Figure 6.7 show the velocity trajectories of the vehicle with various starting speeds. As Figure 6.6 illustrates, the vehicle accelerates at the maximum allowable acceleration until the speed constraint defined in Eq. (6.29) becomes pertinent, from which point the speed is maintained. In contrast to the cases of acceleration, the vehicle decelerates with maximal allowable deceleration until it stops. Figure 6.8 shows the driver's seat acceleration trajectory for the case of straight-ahead braking. According to these results, the vehicle motion is consistent with the motion of a rigid body, meaning that the longitudinal vehicle dynamics of the proposed model are validated.

Optimal design of the vehicle suspension system

The vertical displacement functions and system equations specified in Section 3 can be used to define an optimal suspension design problem. The driver is to be made as comfortable as possible over a range of road conditions and traveling speeds. Thus, the design objective is to minimize the maximum absolute acceleration of the driver's seat by adjusting the vehicular suspension properties subject to the constraints that certain relative displacements do not exceed imposed limits. The objective function is therefore

$$J_0 = \max_{t\in[0,t_f]} \left|\ddot{z}_1(t)\right| \tag{6.35}$$

where $\ddot{z}_1(t)$ is the acceleration of the driver's seat under the road conditions $R_1(y)$ and $R_2(y)$ as defined by Eqs. (6.31) and (6.32).

Two design cases considered by Haug and Arora (1979) are used here to examine the correctness of the proposed method. Figure 6.9 represents the road displacement profiles in the test cases. In case 1, the road surface profile includes a cavity. Case 2 involves two road displacement profiles, presented in Figure 6.9(b) and (c). The speed of the vehicle in case 1 is 450 in/sec and that in case 2 is 960 in/sec. Table 6.2 gives the optimal solutions. A comparison with the results present in the research sources (Hsieh and Arora, 1984; Haug and Arora, 1979)

shows that the results obtained by the proposed method are quite accurate.

Vehicle speed control problem

In most emergency situations, drivers must stop the vehicle quickly and safely. Changing the speed of the vehicle according to the conditions of the road and the distance from the current position to the site of accident is a vehicle speed control problem that the vehicle model and system equations derived in Section 3 can be used to solve. In this case, the initial speed of the vehicle is 880 in / sec (50 mph) and the road surface profile is as plotted in Figure 6.10. According to the definition in Section 2.2, the emergency braking problem is transformed into a time-optimal control problem that is then solved using the proposed NLP method. The minimum time, t_f = 3.4 seconds, and the terminal displacement, z_6 = 1585.7 inches, are obtained using the OCP solver. Figure 6.11 shows the trajectories of the vehicle speed and acceleration. Figure 6.12 plots the trajectories of the acceleration and pitch angle of the passenger seat, which are of interest to vehicle designers. The solid bold curves at the bottom of Figure 6.11 and Figure 6.12 represent the corresponding road profile. The numerical results indicate that all the constraints are satisfied and the optimal control law that solves the emergency braking problem is determined.

6.3 Summary

In this chapter, two practical applications, the flight level control problem and the vehicle suspension design problem – both highly nonlinear optimal control problems – have been formulated following the procedure suggested in this dissertation and solved by the proposed OCP solver. In the case of the flight level control problem, two common types of optimal control problem, the tracking problem and the minimum time problem, were derived to simulate practical situations. The vehicle suspension design problem provided a useful example of dynamic system design. After the problem has been formulated and the proper constraints imposed, users can solve their dynamic optimization problems by applying the

proposed method.

Because the proposed solver provides a convenient tool for solving dynamic optimization problems, proper modeling and formulating the physical problem become the major decisive factors in whether the solution is meaningful or not. Moreover, the constraints must meet actual environmental conditions or the solution will make no sense. Overconstraining the problem will considerably increase the computational efforts and make obtaining the solution harder. In contrast, loosely constraining the problem will provide no practically applicable solution.

Table 6.1 User subroutines for the flight level tracking problem.

```
//-------Program parameters ---------------------------------------------
//B: Discrete design parameters of design variable vector. (INPUT)
// U: Admissible control function vector. (INPUT)
// Z: State variable vector. (INPUT)
// T: Given time grid point. (INPUT)
// G: First term of performance index or functional constraint or
//          dynamic constraint. (OUTPUT)
// NV: Number of design variables for optimizer (INPUT)
// NU: Number of control functions. (INPUT)
// NEQ: Number of state equations (INPUT)
// N: Index of current number of function evaluation. (INPUT)
//
// --------------------------------FFN()-----------------------------------
// Routine to calculate the integral term of the performance index
// or functional constraint
void ffn(double *B, double *U, double *Z, double *T, double *F,
          int NV, int NU, int NEQ, int N, int NBJ)
{
    if (N==0)
      *F = 0.5*((Z[0]-150.0)*(Z[0]-150.0)) + (Z[1]*PI/180.0)
           * ( Z[1] * PI/180.0 ) + (Z[2]*Z[2]);
    else
          *F = 0.0;
}
//---------------------- GFN() ----------------------------------------------
// Routine to calculate the first term of the performance index or
// functional constraint or dynamic constraint
void gfn(double *B, double *U, double *Z, double *T, double *G,
           int NV, int NU, int NEQ, int N, int NBJ)
{
    switch (N)
    {
        case 0:
           *G = 0.0;      break;
      case 1:
           *G = -1 * Z[0] + 92.0; break;
      case 2:
           *G = Z[0] – 170.0; break;
      case 3:
           *G = -1 * Z[1] -20.0; break;
      case 4:
           *G = Z[1] – 25.0; break;
      case 5:
           *G = -1 * Z[2] -150.0; break;
      case 6:
           *G = Z[2] – 150.0;    break;
    };
}
```

Table 6.1 (cont.) User subroutines for the flight level tracking problem.

```
//---------------------- HFN() ----------------------------------------------------
//Routine to calculate the state trajectory.
void hfn(double *B, double *U, double *Z, double *DZ, double *T,
            int NV, int NU, int NEQ )
{
     DZ[0] = -1*((aD*Z[0]*Z[0]/m) + (g*sin(Z[1]*PI/180.0))) + U[0]
                  *10000 / m;
     DZ[1] = (aL*Z[0]*(1-c*Z[1])/m) - (g*cos(Z[1]*PI/180.0)/Z[0]) +
                aL*c*Z[0]*U[1]/m;
     DZ[2] = Z[0]*sin(Z[1]*PI/180.0);
}
//---------------------- Z0FN() ---------------------------------------------------
// Routine to calculate the initial state vector.
void z0fn(double *B, double *ZINT, int NV, int NEQ )
{
   ZINT[0] = 100.0;
   ZINT[1] = 20.0;
   ZINT[2] = -120.0;
}
```

Table 6.2 Optimal solutions for vehicle suspension.
(a) Case 1

	Haug and Arora (1979)	Hsieh and Arora (1984)	Proposed Method
k_1	50.00	50.00	50.00
k_2	200.00	204.10	200.00
k_3	241.90	293.90	200.00
c_1	12.89	30.87	39.96
c_2	77.52	76.94	77.35
c_3	80.00	80.00	80.00
Cost	257.40	255.80	254.00

(b) Case 2

	Haug and Arora (1979)	Proposed Method
k_1	50.00	191.90
k_2	200.00	200.00
k_3	200.00	200.00
c_1	8.93	8.52
c_2	45.92	25.24
c_3	37.81	29.16
Cost	125.50	125.60

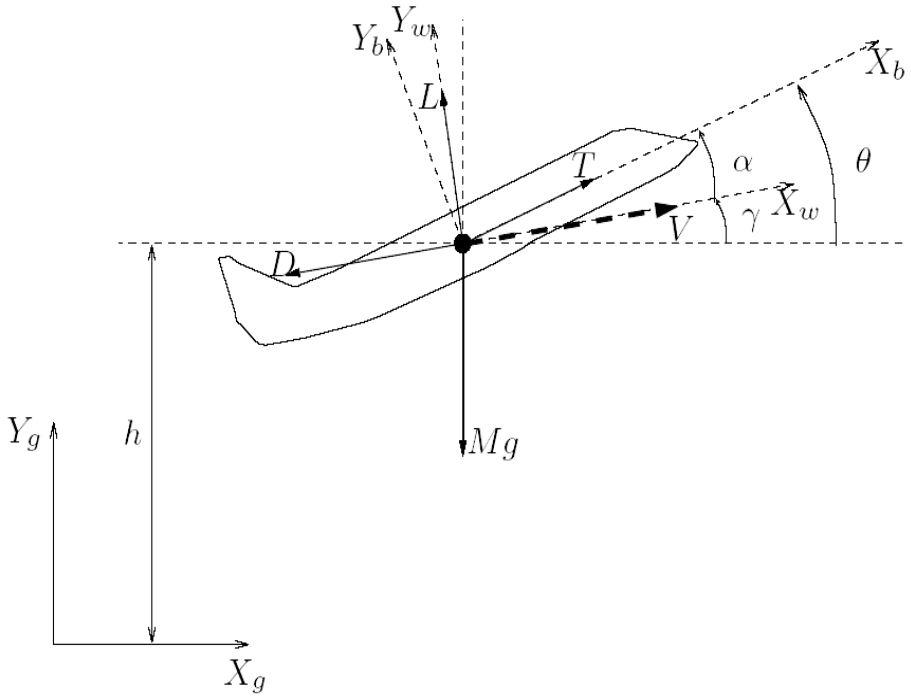
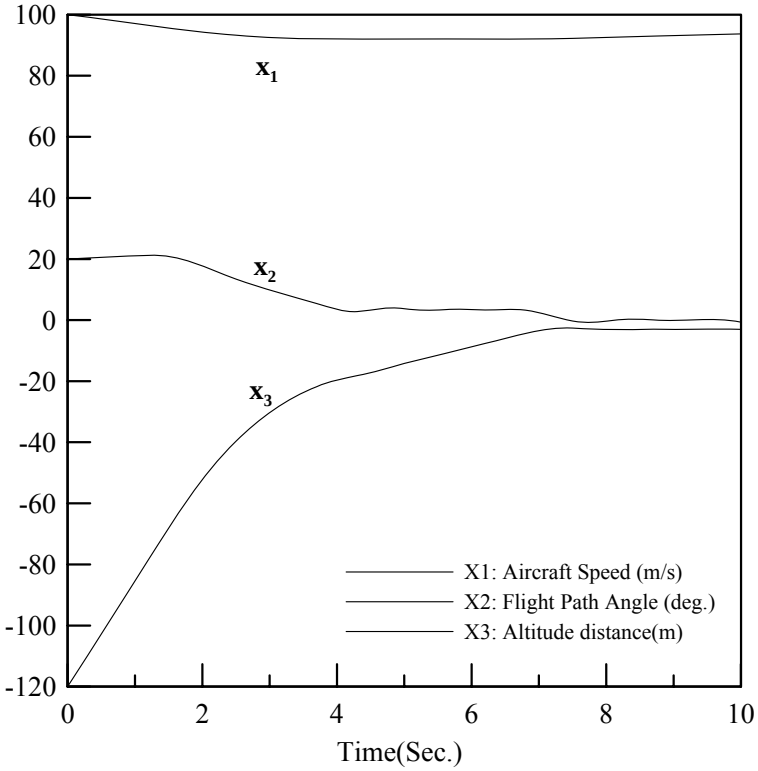
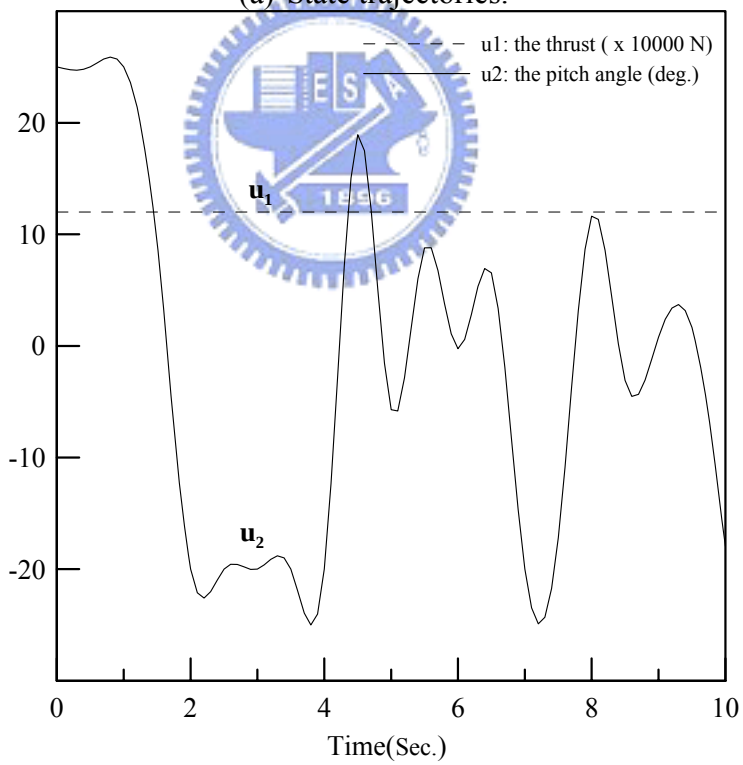


Figure 6.1 Aircraft model (Lygeros, 2003).

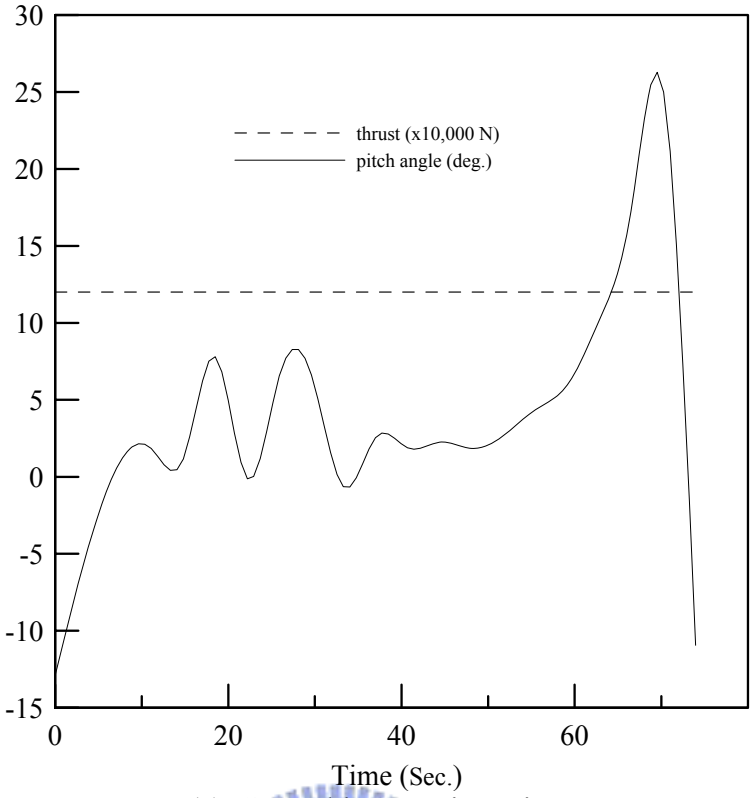


(a) State trajectories.

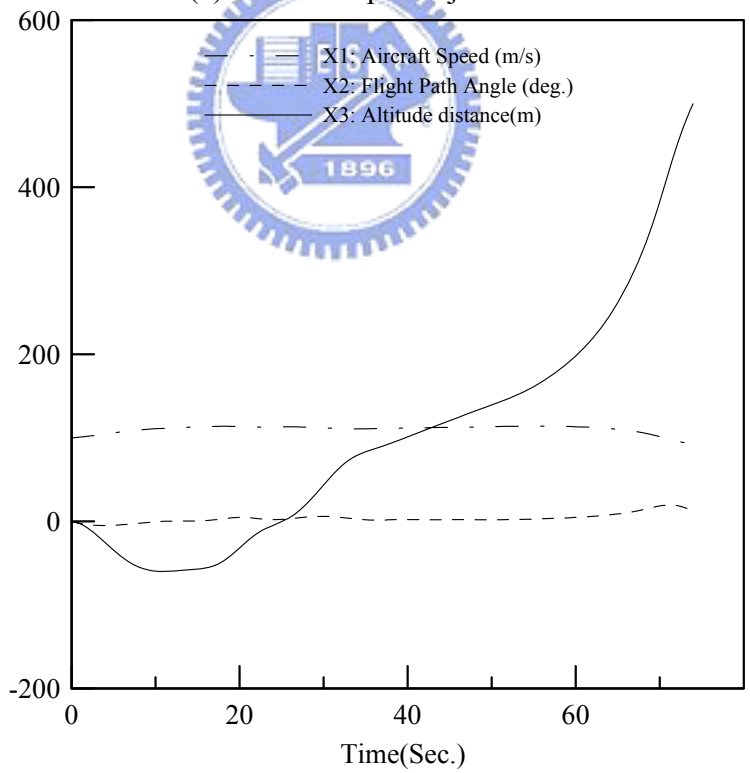


(b) The thrust trajectory.

Figure 6.2 Numerical results for the tracking problem.



(a) Control input trajectories.



(b) State trajectories.

Figure 6.3 Trajectories for the minimum time problem.

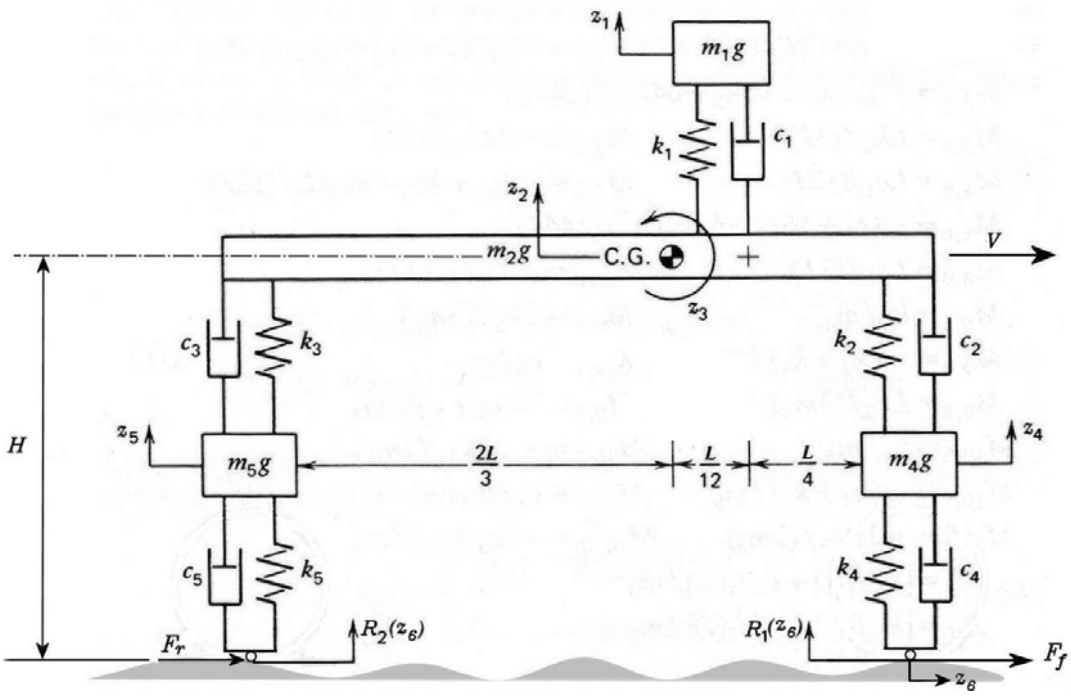


Figure 6.4 Six-degrees-of-freedom vehicle model

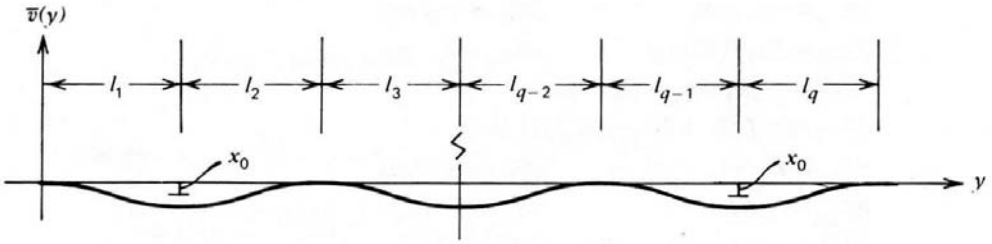


Figure 6.5 Sinusoidal displacement functions.

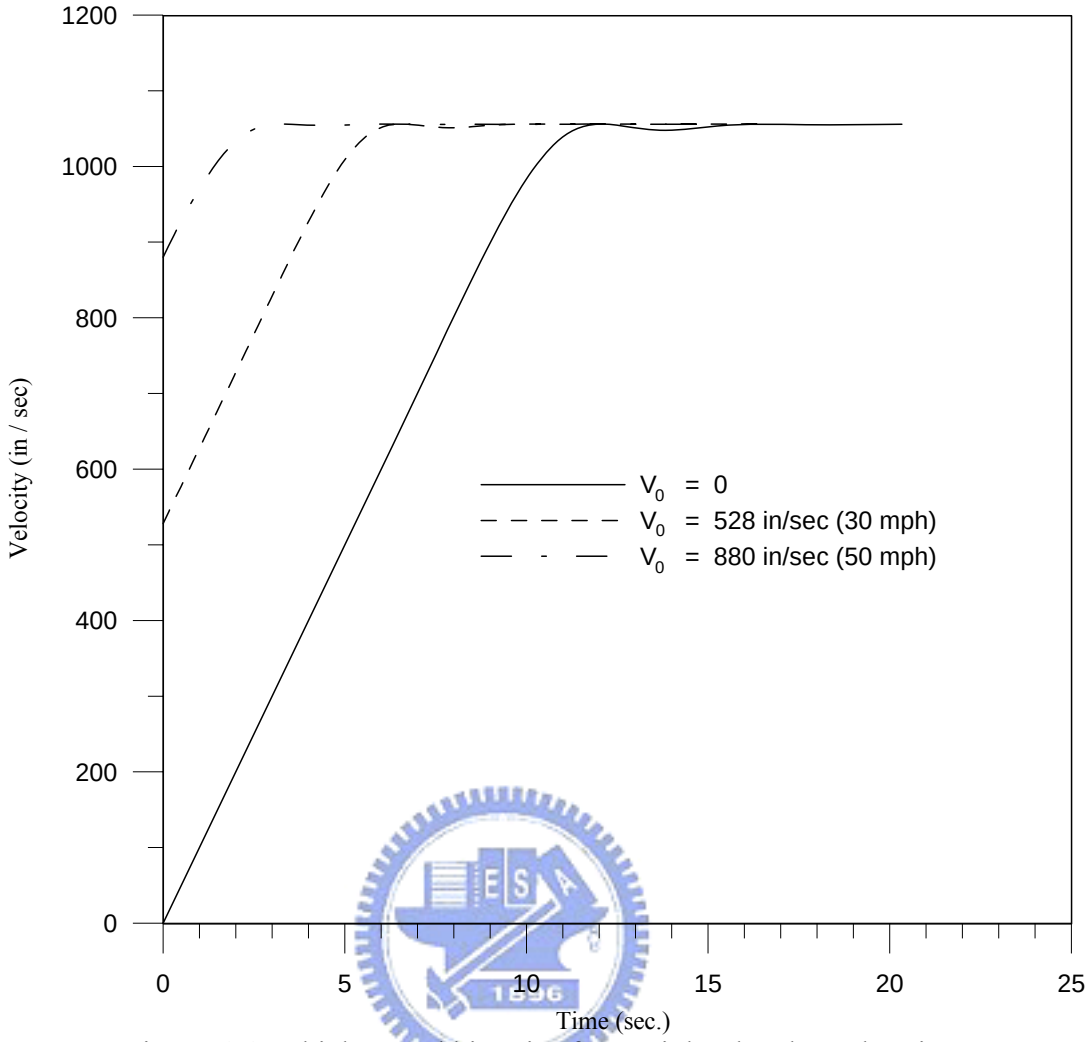


Figure 6.6 Vehicle speed histories for straight-ahead accelerating.

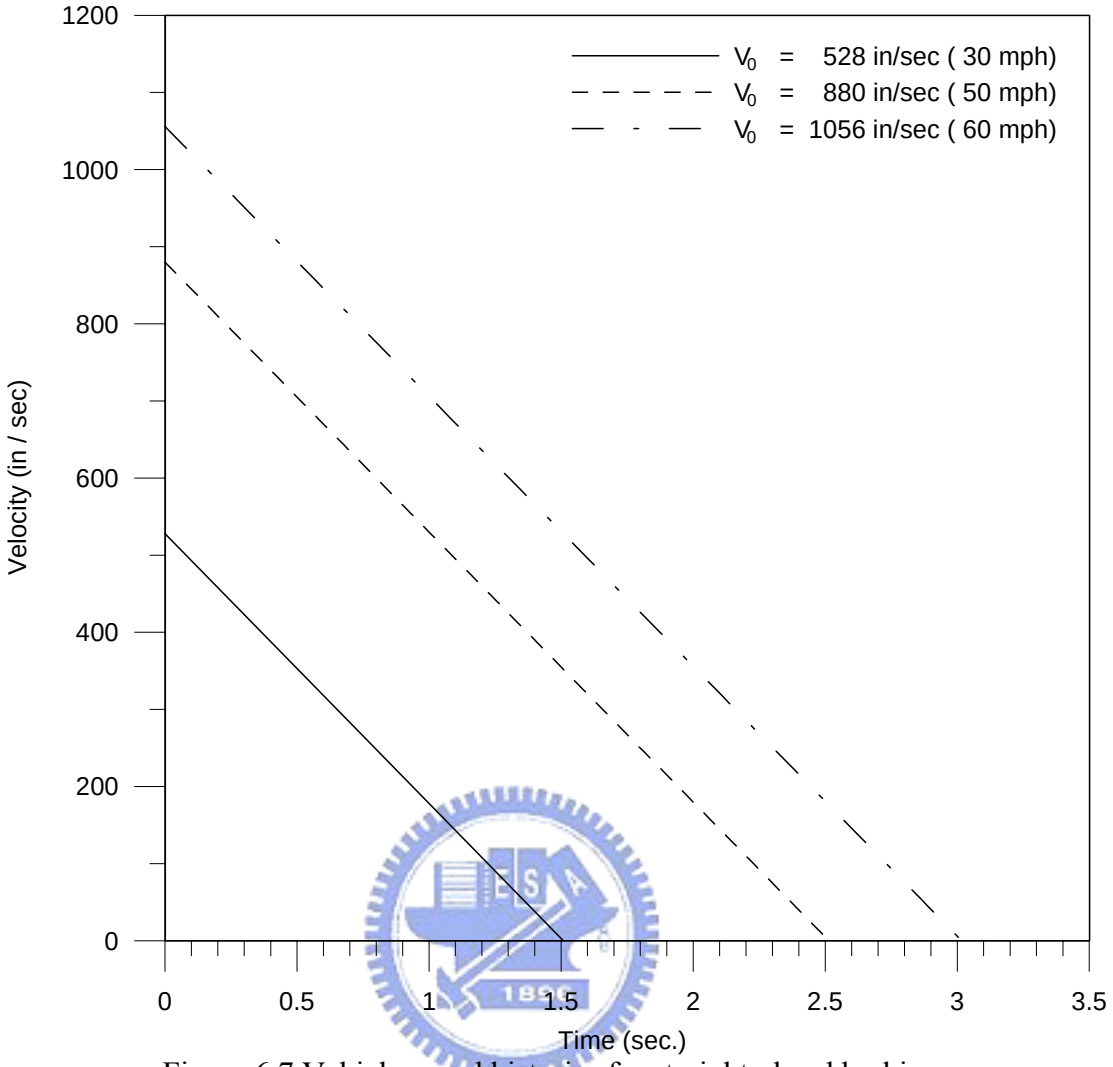


Figure 6.7 Vehicle speed histories for straight-ahead braking.

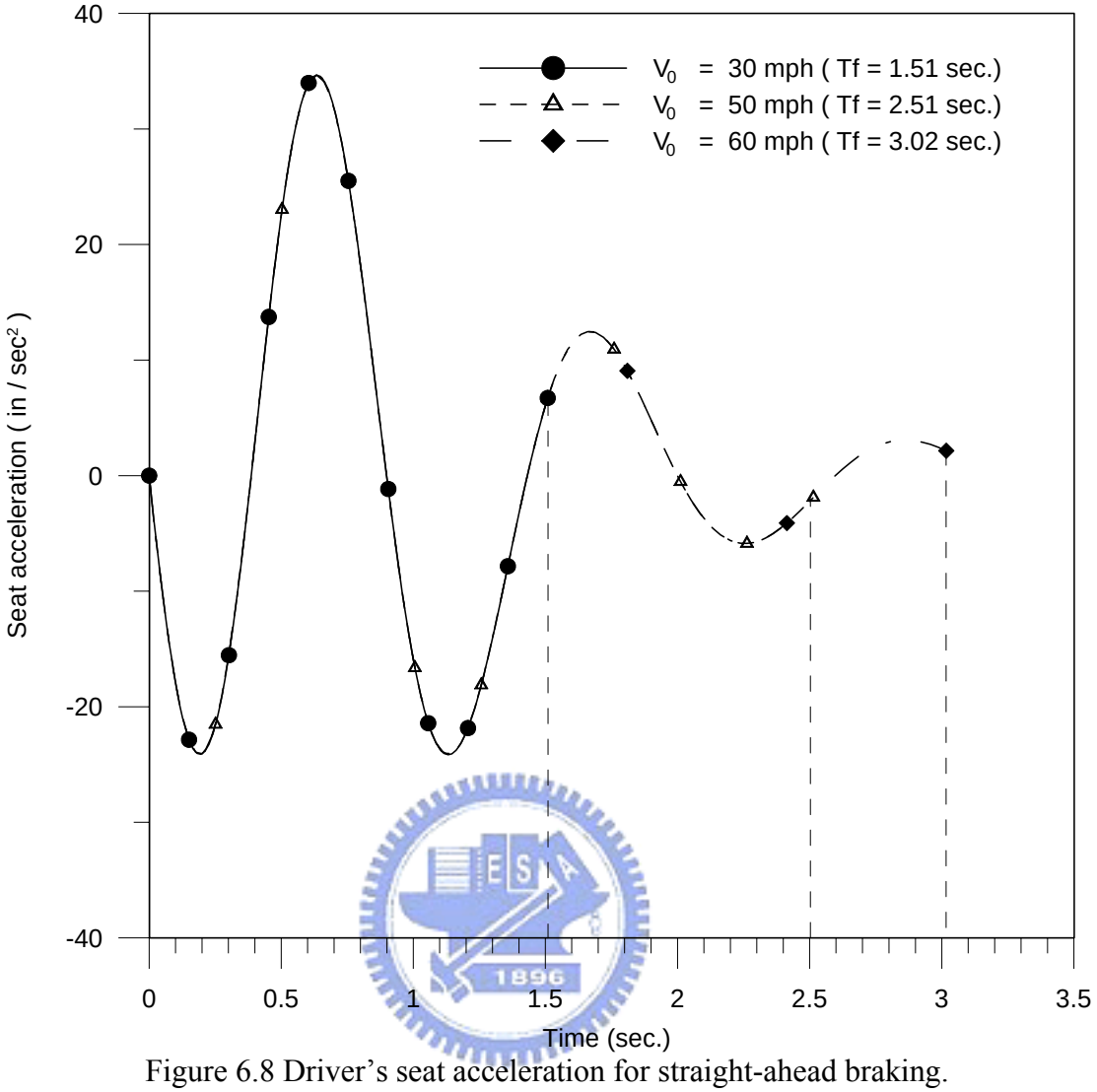
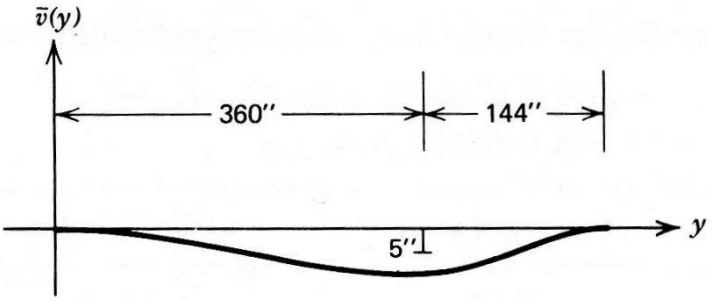
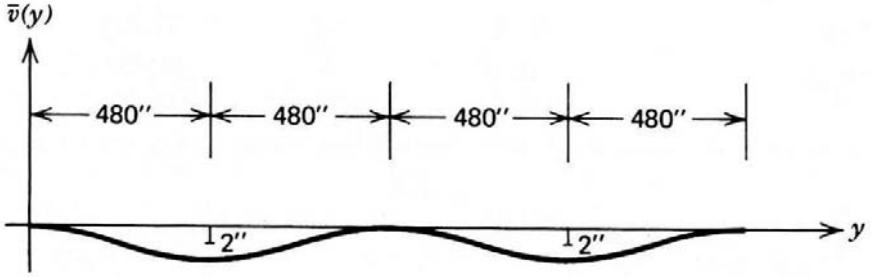


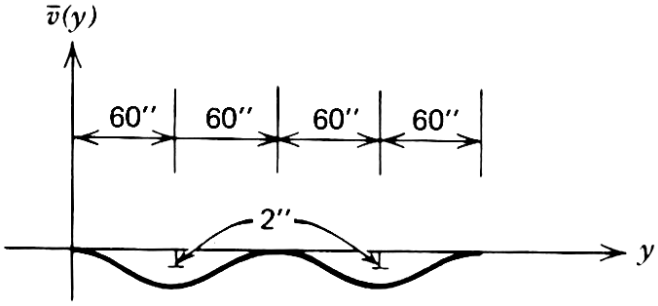
Figure 6.8 Driver's seat acceleration for straight-ahead braking.



(a) Case I



(b) Case II – road profile 1.



(c) Case II – road profile 2.

Figure 6.9 Road displacement profiles for model validation (Haug and Arora 1979).

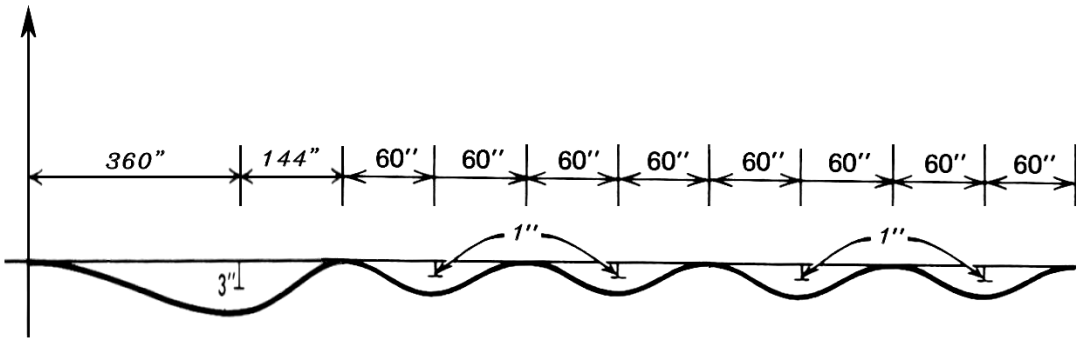


Figure 6.10 Road displacement profiles for emergency braking.

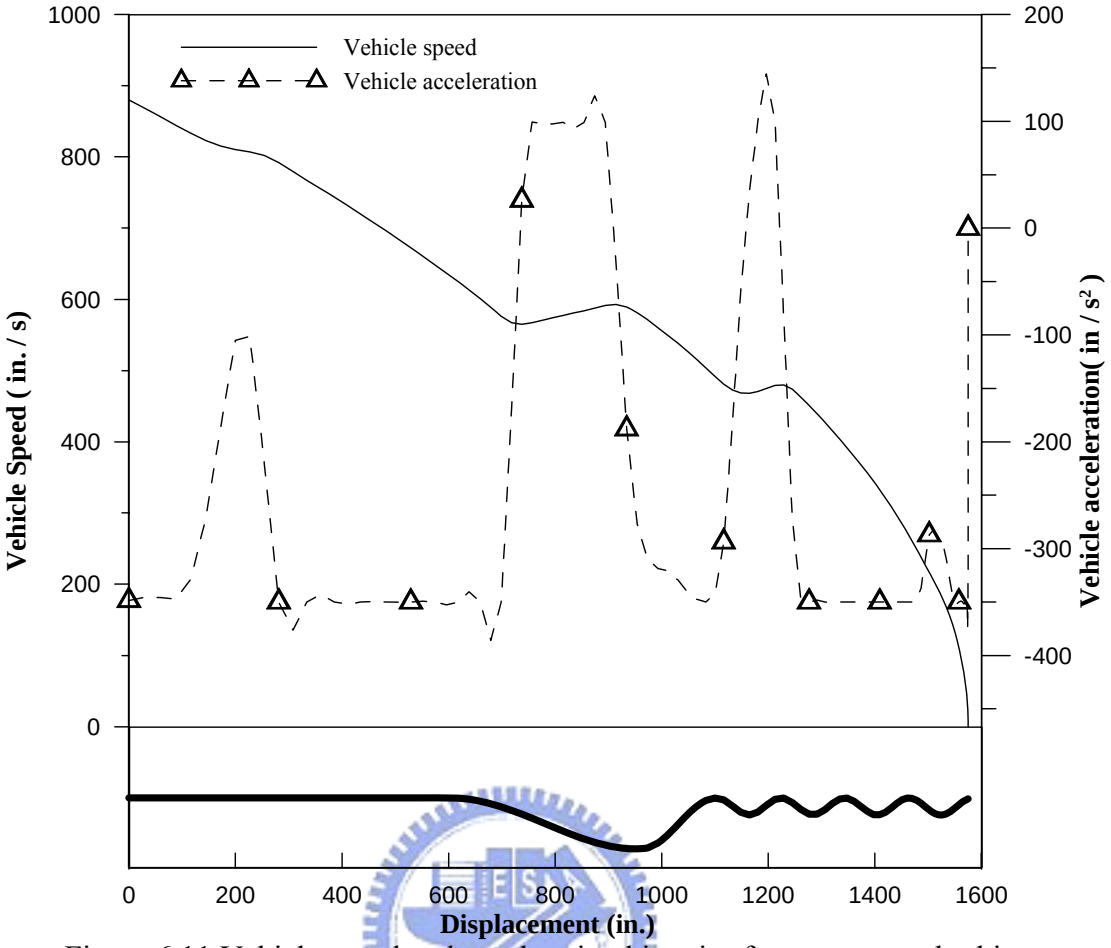


Figure 6.11 Vehicle speed and acceleration histories for emergency braking.

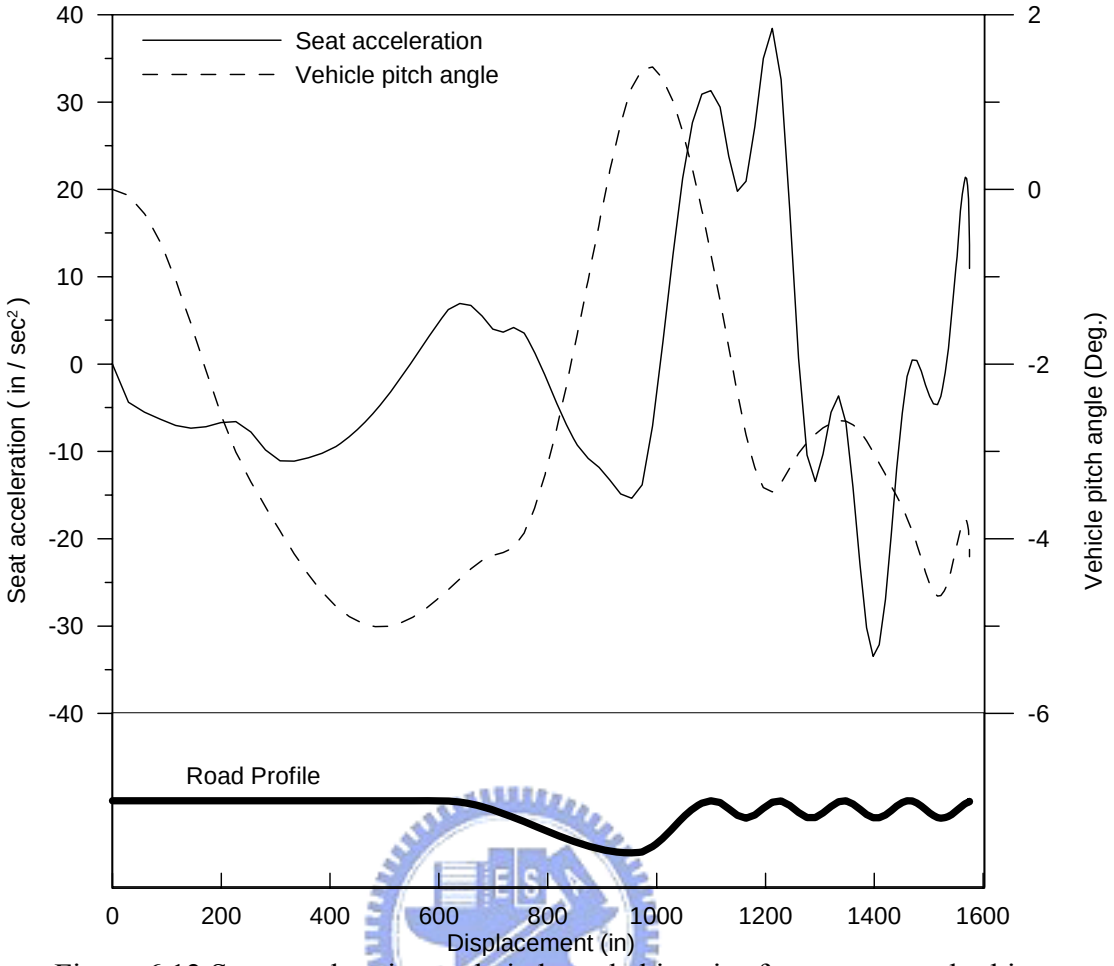


Figure 6.12 Seat acceleration and pitch angle histories for emergency braking.

CHAPTER 7
CONCLUSIONS AND FUTURE STUDY

7.1 Concluding Remarks

This study has introduced and compared two typical methods – the indirect and direct approaches – usually used to solve optimal control problems. Because most of practical control problems are described by strongly nonlinear differential equations that are difficult to solve by indirect methods, direct methods have been widely studied in the recent literature and are also adopted in this dissertation. In spite of extensive use of direct and indirect methods to solve optimal control problems, engineers still expend much effort on reformulating problems and implementing corresponding programs for different control problems. Therefore, the first objective of this dissertation was to develop a convenient solver and provide a systematic computational procedure capable of assisting engineers and students easily solve their dynamic system design problems.

To this end, a computational AOCP method, one kind of sequential direct method, has been developed and presented herein. Subsequently, it was implemented and coupled with many robust numerical schemes to develop a general OCP solver. Besides outlining the theoretical and numerical foundations involved in the proposed solver, the discussion detailed the OCP solver implementation, including the dynamic constraint treatments, ODE solver, sensitivity analysis, and so forth. The *van der Pol* oscillator problem with three different terminal conditions and a highly nonlinear time-optimal control problem were used to illustrate and verify the stability and capability of the proposed solver. In these examples, different numerical schemes and different time intervals were applied to investigate the numerical schemes' effect on the validity of the solution and computational efficiency. The results indicate that the OCP solver coupled with the systematic procedure suggested in this study can truly facilitate the solving of engineering control problems in a systematic and

efficient way.

The other objective of this dissertation was to develop a novel method for solving discrete-valued optimal control problems. Most traditional methods focus on the continuous optimal control problems and fail when applied to a discrete-valued optimal control problem. One common type of such problems is the bang-bang type control problem arising from time-optimal control problems. When the controls are assumed to be of the bang-bang type, the time-optimal control problem becomes one of determining the TOCP switching times. Several methods for such determination have been studied extensively in the literature; however, these methods require that the number of switching times be known before their algorithms can be applied. As a result, they cannot meet practical situations in which the number of switching times is usually unknown before the control problem is solved. Therefore, to solve discrete-valued optimal control problems, this dissertation has focused on developing a computational method consisting of two phases: (a) the calculation of switching times using existing optimal control methods and (b) the use of the information obtained in the first phase to compute the discrete-valued control strategy. The proposed algorithm combines the proposed OCP solver with an enhanced branch-and-bound method. To demonstrate the proposed computational scheme, the study applied third-order and fourth-order systems and an F-8 fighter aircraft control problem considered in several pioneering studies. Comparing the results of this study with the results from the literature indicates that the proposed method provides a better solution and the accuracy of the terminal constraints is acceptable. Finally, the proposed solver and procedure were applied to two engineering applications: the flight level control problem and the vehicle suspension design problems.

7.2 Future Study

Future study will focus on two topics: one is to develop a web-based dynamic

optimization solver and another one is to improve the global convergence of proposed method. In terms of the former, although a convenient solver for dynamic optimization has been developed in this dissertation, constructing a turnkey system for solving dynamic system design problems is still a major problem for inexperienced engineers and students. In addition, because of the resource limitation including finances and laboratory facilities, it is difficult to provide each engineer with a turnkey system to solve his/her dynamic system design problem. The advent of the Internet and distributed computing technologies suggest that a Web-based optimization tool may provide a potential partial solution to this problem (Chu, 1999). A Web-based dynamic optimization solver would also be useful to engineering curriculums because students could share limited resource via the Internet. Many Internet techniques such as Web service, Simple Object Access Protocol (SOAP), DataSocket, and XML, could be applied to the development of such a solver, for which a conceptual flowchart is given in Figure 7.1. For this suggested model, the proposed OCP solver would serve as a numerical engine.

As regards improving global convergence, the convergency of the OCP solver depends on good initial guesses that speed up optimization problem convergence and produce high-precision solutions. However, it is difficult for the inexperienced to provide good initial guesses that lie within the convergence domain. Therefore, a module that assists with estimation of the optimal solution will be developed to help the novice making the proper initial guess.

It is expected that the contents of Chapters 2 through 5 will be the basis for addressing these two topics, which, because of modular programming techniques, can be effected by adding external modules into the proposed solver or replacing the original modules with new ones. This feature allows the proposed solver to be easily updated by state-of-the-art algorithms.

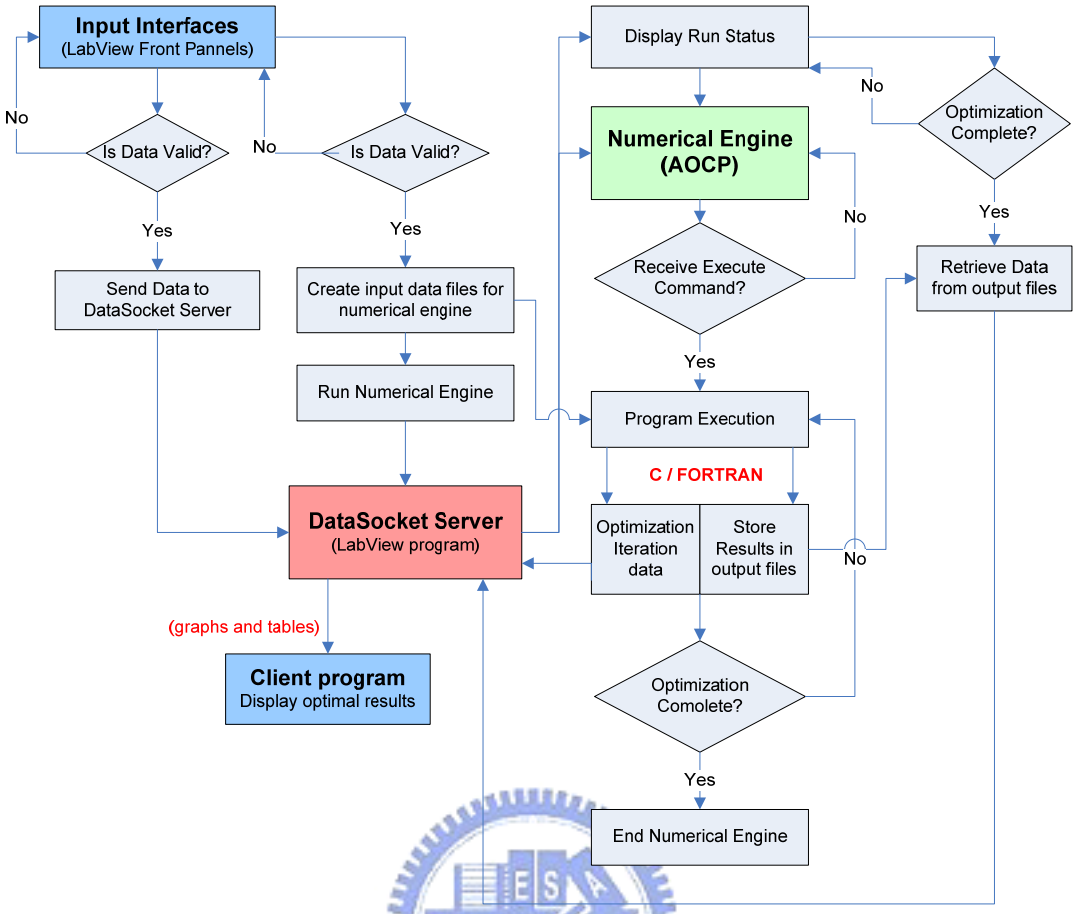


Figure 7.1 Conceptual flow chart of a web-based dynamic optimization solver.

REFERENCE

Arora, J. S., *Introduction to Optimum Design*, McGraw-Hill, 2nd Ed., 2004.

Banks, S.P., and Mhana, K.J., "Optimal Control and Stabilization of Nonlinear systems," IMA Journal of Mathematical Control and Information, Vol. 9, pp. 179-196, 1992.

Barclay, A., Gill, Ph.E., and Rosen, J.B., "SQP methods and their application to numerical optimal control," Report NA97-3, Department of Mathematics, University of California, San Diego, USA, 1997.

Belegundu, A.D., and Arora, J.S., "A Recursive Quadratic Programming Algorithm with Active Set Strategy for Optimal Design," International Journal for Numerical Methods in Engineering, Vol. 20, pp. 803-816. , 1984

Bellman, R., *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1957.

Bellman, R., Introduction to the Mathematical Theory of Control Processes, Vol. 2, Academic Press, New York, 1971.

Bellman, R., and Dreyfus, S., *Applied Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1962.

Bellman, R., and Dreyfus, R.E., *Dynamic Programming and Modern Control Therapy*, Academic Press, Orlando, Florida, 1977.

Bertsekas, D.P., Constrained Optimization and Lagrange Multiplier Methods, Academic Press, 1982.

Bertrand, R., and Epenoy, R., "New Smoothing Techniques for Solving Bang-Bang Optimal Control Problems – Numerical Results and Statistical Interpretation," Optimal Control Applications and Methods, Vol. 23, pp. 171-197, 2002.

Betts, J.T., and Huffmann, W.P., "Application of Sparse Nonlinear Programming to Trajectory Optimization," J. Guidance, Control and Dynamics, Vol. 15, pp. 198-206, 1992.

Betts, J.T., and Huffmann, W.P., “Path Constrained Trajectory Optimization Using Sparse Sequential Quadratic Programming,” J. Guidance, Control and Dynamics, Vol. 16, pp. 59-68, 1993.

Betts, J.T., “Very Low-Thrust Trajectory Optimization Using A Direct SQP Method,” Journal of Computational and Applied Mathematics, Vol. 120, pp. 27-40, 2000.

Betts, J.T., “Survey of Numerical Methods for Trajectory Optimization,” J. Guidance, Control and Dynamics, Vol. 21, No. 2, pp. 193-207, 1998.

Betts, J.T., *Practical Methods for Optimal Control Using Nonlinear Programming*, SIAM, Philadelphia, 2001.

Bock, H.G., “Numerical Solution of Nonlinear Multipoint Boundary Value Problems with Application to Optimal Control,” ZAMM, Vol. 58, pp. 407-409, 1978.

Boggs, P.T., and Tolle, J.W., *Sequential quadratic programming*, Acta Numerica, pp. 1-52, 1995.

Bryson, A.E. Jr., and Ho, Y.C., *Applied Optimal Control*, John Wiley & Sons, New York, 1975.

Bryson, A.E. Jr., and Ross, S.E., *Optimum rocket trajectories with aerodynamic drag*, Jet Propulsion, 1958.

Bulirsch, R., “Die Mehrzielmethode zur numerischen Losung von nichtlinearen Randwertproblemen und Aufgaben der optimalen Steuerung,” Report of the Carl-Cranz Gesellschaft, DLR, Oberpfa_enhofen, 1971.

Bullock, T.E., and Franklin, G.F., “A Second-order Feedback Method for Optimal Control computations,” IEEE Transactions on Automatic Control, Vol. 12, pp. 666-673, 1967.

Buskens, C., and Maurer, H., “SQP-Methods for Solving Optimal Control Problems with Control and State Constraints: Adjoint Variables, Sensitivity Analysis and Real-Time Control,” Journal of Computational and Applied Mathematics, Vol.120, pp. 85-108, 2000.

Cervantes, A., and Biegler, L.T., "Large-Scale DAE Optimization using Simultaneous Nonlinear Programming Formulations," AIChE Journal, Vol. 44, pp. 1038, 1998.

Chu, K.C., "The development of a web-based teaching system for engineering education," Eng. Sci. Educ. J. Vol. 8, No. 3, pp. 115-118, 1999.

Chyba, M., Leonard, N.E., and Sontag, E. D., "Singular Trajectories in Multi-input Time-Optimal Problems: Application To Controlled Mechanical Systems," Journal of Dynamical and Control Systems, Vol. 9, No. 1, pp. 103-129, 2003.

Cook, M.V., *Flight Dynamics Principles*, Wiley, New York, 1997.

Etkin, B., and Redi, L.D., *Dynamics of Flight: Stability and Control*, 3rd Ed., Wiley, New York, 1996.

Floudas, C.A., and Pardalos, P.M., *Recent Advances in Global Optimization*, Princeton University Press, 1992.

Gill, P.E., Murray, W., and Saunders, M.A., "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," SIAM Jounal of Optimization, Vol. 12, No. 4, pp. 979-1006, 2002.

Gillespie, T.D., *Fundamentals of Vehicle Dynamics*, Society of Automotive Engineers, 1992.

Goh, C.J., and Teo, K.L., "Control Parameterization: A Unified Approach to Optimal Control Problems with General Constraints," Automatica, Vol. 24, pp. 3-18, 1988.

Hansen, E.R., *Global Optimization Using Interval Analysis*, M. Dekker, New York, 1992.

Haug, E.J., and Arora, J.S., *Applied Optimal Design: Mechanical and Structural Systems*, John Wiley and Sons, New York, 1979.

Hock, W., and Schittkowski, K., *Lecture Notes in Economics and Mathematical Systems 187 - Test Examples for Non-linear Programming Codes*, Spring-Verlag, New York, 1980.

Horst, R., and Tuy, H., *Global optimization: Deterministic Approaches*, Springer-Verlag, Berlin, 1993.

Hribar, M.E., Large scale constrained optimization, Ph.D. Disertation, Northeasten University, 1996.

Hu, G.S., Ong, C.J., and Teo, C.L., "An Enhanced Transcribing Scheme for The Numerical Solution of A Class of Optimal Control Problems," Engineering Optimization, Vol. 34, No. 2, pp. 155-173, 2002.

Huang, C.H., "Integrating Analysis and Optimization Systems on X Window," Master Thesis in Mechanical Engineering, Department of Mechnaical Engineering, National Chiao-Tung Univ., Taiwan, R.O.C., 1994.

Huang, C.H., and Tseng, C.H., "Computational Algorithm for Solving A Class of Optimal Control Problems," IASTED International Conference on Modelling, Identification, and Control (MIC2003), Innsbruck, Austria, 2003, pp. 118-123.

Huang, C.H., and Tseng, C.H., "Numerical Approaches for Solving Dynamic System Design Problems: An Application to Flight Level Control Problem," Proceedings of the Fourth IASTED International Conference on Modelling, Simulation, and Optimization (MSO2004), Kauai, Hawaii, USA, 2004, pp. 49-54.

Huang, C.H., and Tseng, C.H., "A Convenient Solver for Solving Optimal Control Problems," Jounal of the Chinese Institute of Engineers, Vol. 28, pp. 727-733, 2005.

Huang, C.H., and Tseng, C.H., "A Two-Phase Computational Scheme for Solving Bang-Bang Control Problems," Optimization and Engineering, 2006. (Accepted)

Jaddu, H., "Direct Solution of Nonlinear Optimal Control Problems Using Quasilinearization and Chebyshev Polynominals," Journal of the Franklin Institute, Vol. 339, pp. 479-498, 2002.

Jaddu, H., and Shimemura, E., "Computational Method Based on State Parameterization for Solving Constrained Nonlinear Optimal Control Problems," International Journal of Systems Science, Vol. 30, No. 3, pp. 275-282, 1999.

Jalili, N., and Esmailzadeh, E., “Optimum Active Vehicle Suspensions With Actuator Time Delay,” ASME Trans. Journal of Dynamic Systems, Measurement, and Control, Vol. 123, pp. 54-61, 2001.

Kim, T.H., and Ha, I.J., “Time-Optimal Control of a Single-DOF Mechanical System with Friction,” IEEE Trans. Automat. Contr., Vol. 46, No. 5, pp.751-755, 2001.

Kirk, D.E., *Optimal Control Theory: An Introduction*, Prentice-Hall, 1970.

Kreyszig, E., *Advanced Engineering Mathematics*, 8th Edition, John Wiley & Sons, New York, 1999.

Kaya, C.Y., and Noakes, J.L., “Computations and Time-Optimal Controls,” Optimal Control Applications and Methods, Vol. 17, pp. 171-185, 1996.

Lee, H.W., Jennings, L.S., Teo, K.L., and Rehbock, V., “Control Parameterization Enhancing Technique for Time Optimal Control Problems,” Dynamic Systems and Applications, Vol. 6, pp. 243-262, 1997.

Lee, H.W.J., Teo, K.L., Rehbock, V., and Jennings, L.S., “Control Parameterization Enhancing Technique for Optimal Discrete-Valued Control Problems,” Automatica, Vol. 35, pp. 1401-1407, 1999.

Lin, S.Y., “Complete Decomposition Algorithm for Nonconvex Separable Optimization Problems and Applications,” Automatica, Vol. 28, pp. 249-254, 1992.

Lee, H.W.J., Teo, K.L., and Andrew E.B. Lim, “Sensor Scheduling in Continuous Time,” Automatica, Vol. 37, pp. 2017-2023, 2001.

Lucas, S.K., and Kaya, C.Y., “Switching-Time Computation for Bang-Bang Control Laws,” Proceedings of the 2001 American Control Conference, pp. 176-181, 2001.

Luenberger, D.G., *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, Reading, MA, 1984.

Lygeros, J., “Minimum Cost Optimal Control: An Application to Flight Level Tracking,”

IEEE 11th Mediterranean Conference on Control and Automation (MED'03), Rhodes, Greece, June 18-20, 2003.

Lygeros, J., Tomlin, C., and Sastry, S., "Controllers for Reachability Specifications for Hybrid Systems," Automatica, pp. 349-370, 1999.

Michalewicz, Z., Dasgupta, D., LeRiche, R.G., and Schoenauer, M., "Evolutionary Algorithms for Constrained Engineering Problems," Computers and Industrial Engineering Journal, Vol. 30, No.2, pp. 851-870, 1996.

Pinch, E.R., *Optimal Control and the Calculus Variations*, Oxford University Press Inc., New York, 1993.

Pytlak, R., "Numerical Methods for Optimal Control Problems with State Constraints," Lecture Notes in Mathematics 1707, Springer-Verlag, Berlin, 1999.

Polak, E., *Computation Method in Optimization*, New York and London: Academic Press, 1971.

Pontryagin, L.S., Boltyanskii, V.G., Gamkrelidze, R.V., and Mischenko, E.F., *The Mathematical Theory of Optimal Processes*, Wiley, 1962.

Powell, M.J.D., "The Convergence of Variable Metric Methods for Nonlinearly Constrained Optimization Calculations," in Nonlinear Programming, Vol. 3, edited by O.L. Mangasarian *et al*., Academic Press, New York, 1978.

Press, W., Teukolsky, S.A., Vetterling, W.T., and Flannery, B., *Numerical Recipes in C: The Art of Scientific Computing*, 2nd Edition, Cambridge, New York, 1992.

Pshenichny, B.N., "Algorithms for the General Problem of Mathematical Programming," Kibernetica, No.5, 1978.

Pytlak, R., "Numerical Methods for Optimal Control Problems with State Constraints," Lecture Notes in Mathematics 1707, Springer-Verlag, Berlin, 1999.

Sage, A.P., and White, C.C. III, *Optimum systems Control*, 2nd Edition, Prentic Hall, 1977.

Samaras, N.S., and Simaan, M.A., "Optimized Trajectroy Tracking Control of Multistage Dynamic Metal-Cooling Processes," IEEE Tran. Ind. Applicat., Vol. 27, No. 3, pp. 920-927, 2001.

Simakov, S.T., Kaya, C.Y., and Lucas, S.K., "Computations for Time-Optimal Bang-Bang Control Using A Lagrangian Formulation," 15th Triennial World Congress, Barcelona, Spain, 2002.

Steindl, A., and Troger, H., "Optimal Control of Deployment of a Tethered Subsatellite," Nonlinear Dynamics, Vol. 31, pp. 257–274, 2003.

Stoer, J., and Bulirsch, R., *Introduction to Numerical Analysis*, Springer, New York, 1980.

Stryk, O. and Bulirsch, R., "Direct and Indirect Methods for Trajectory Optimization," Annals Operation Research, Vol. 37, pp. 357-373, 1992.

Teo, K.L., Goh, C.J., and Wong, K.H., *A Unified Computational Approach to Optimal Control Problems*, John Wiley & Sons, New York, 1991.

Teo, K.L., and Wu, Z.S., *Computation Methods for Optimizing Distributed Systems*, Academic Press, Orlando, 1984.

Thanedar, P.B., Arora, J.S., and Tseng, C.H., "A Hybrid Optimization Method and Its Role in Computer-Aided Design," Comp. Struct., Vol. 23, pp. 305-314, 1986.

Tomlin, C., Lygeros, J., and Sastry, S., "Aerodynamic envelope protection using hybrid control," *Computer Vision, and Materials Science*, Cambridge University Press, New York, 1996.

Tseng, C.H., *Optimal Design for Dynamics and Control Using a Sequential Quadratic Programming Algorithm*, PhD dissertation, Department of Mechanical Engineering, Iowa University, 1987.

Tseng, C. H., and Arora, J. S., "On Implementation of Computational Algorithms of Optimal Design 1: Preliminary Investigation," International Journal for Numerical Methods in

Engineering, Vol. 26, pp.1365-1384, 1988a.

Tseng, C. H., and Arora, J. S., "On Implementation of Computational Algorithms of Optimal Design 2: Extensive Numerical Investigation," International Journal for Numerical Methods in Engineering, Vol. 26, pp.1385-1402, 1988b.

Tseng, C.H., and Liao, W.C., *Integrated Software for Multifunction Optimization*, Master Thesis in Mechanical Engineering, Department of Mechanical Engineering, National Chiao Tung Univ., R.O.C., 1990.

Tseng, C. H., Liao, W. C., and Yang, T. C., *MOST 1.1 User's Manual*. Technical Report No. AODL-93-01, Department of Mechanical Engineering, National Chiao Tung Univ., Taiwan, R.O.C., 1993.

Tseng, C.H., Wang, L.W., and Ling, S.F., "Enhancing Branch-And-Bound Method for Structural Optimization," Journal of Structural Engineering, Vol. 121, No. 5, pp. 831-837, 1995.

Volkwein, S., "Application of the Augmented Lagrangian-SQP Method to Optimal Control Problems for the Stationary Burgers Equation," Computational Optimization and Applications, Vol. 16, pp. 57–81, 2000.

Wu, S.T., "Time-Optimal Control and High-Gain Linear State Feedback," International Journal of Control, Vol. 72, No.9, pp. 764-772, 1999.

Wu, Z., *The Theory and Applications of Discrete Constrained Optimization Using Lagrange Multipliers*, PhD. Thesis in Computer Science, University of Illinois at Urbana-Champaign, 2000.

Xu, X., and Antsaklis, P. J., "Optimal Control of Switched Systems Based on Parameterization of the Switching Instants," IEEE Transactions on Automatic Control, Vol. 49, No. 1, 2004.

Yang, T.C., Tseng, C.H., and Huang, C.H., "An Interface Coupler for Finite Element Analysis

and Optimization," Proceedings of the 16th National Conference on Theoretical and Applied Mechanics, Society of Theoretical and Applied Mechanics of the Republic of China, Vol. 2, pp. 887-894., 1992.

AUTHOR'S PUBLICATION LIST

I. Referred Papers

1. C.H. Huang and C.H. Tseng, "An Integrated Two-Phase Scheme for Solving Bang-Bang Control Problems," Accepted for publication in Optimization and Engineering.

2. C.H. Huang and C.H. Tseng, "A Convenient Solver for Solving Optimal Control Problems," Journal of the Chinese Institute of Engineers, Vol. 28, pp. 727-733, 2005.

II. Conference Papers

1. C.H. Huang, and Tseng, C.H., "Numerical Approaches for Solving Dynamic System Design Problems: An Application to Flight Level Control Problem," Proceedings of the Fourth IASTED International Conference on Modelling, Simulation, and Optimization (MSO2004), Kauai, Hawaii, USA, 2004, pp. 49-54.

2. C.H. Huang, and Tseng, C.H., "Computational Algorithm for Solving A Class of Optimal Control Problems," IASTED International Conference on Modelling, Identification, and Control (MIC2003), Innsbruck, Austria, 2003, pp. 118-123.

3. 黃智宏，毛彥傑與曾錦煥，大學機械工程教育創意教學改進計畫之說明，工程創造力推動經驗交流研討會，第 107-113 頁，2003.

4. C.H. Huang, C.H. Tseng, and S.F. Ling, Integrating Analysis and Optimization Systems with Distributed Computing System," Proceedings of the 18th National Conference on Theoretical and Applied Mechanics, Society of Theoretical and Applied Mechanics of the Republic of China, Vol. 2, pp. 313-320, 1994.

5. T.C. Yang, C.H. Tseng, and C.H. Huang, "An Interface Coupler for Finite Element Analysis and Optimization", Proceedings of the 16th National Conference on Theoretical and Applied Mechanics, Keelung, December 1992.

6. S. Lin, C.H. Tseng, and C.H. Huang, "Integrating analysis and Optimum Design", Proceedings of the 16th National Conference on Theoretical and Applied Mechanics, Keelung, December 1992.

VITA

姓名：黃智宏 (Huang, Chih-Hung)

生日：民國 57 年 8 月 18 日

住址：台南縣新營市育德街 54 號

mailto: angushuang@ms93.url.com.tw

學歷：國立嘉義農業專科學校農業機械科

（民國 73 年 9 月至民國 78 年 6 月）

國立交通大學機械工程學系

（民國 78 年 9 月至民國 81 年 6 月）

國立交通大學機械工程研究所碩士班

（民國 81 年 9 月至民國 83 年 6 月）

國立交通大學機械工程研究所博士班

（民國 87 年 9 月至民國 95 年 5 月）