CRANFIELD UNIVERSITY

College of Defence Technology

Department of Aerospace, Power and Sensors

PhD Thesis

Academic Year 2005–2006

Subchan

Computational Optimal Control of the Terminal Bunt Manoeuvre

Supervisors

Dr. Rafał Żbikowski

Prof. Brian White

Acknowledgements

I would like to express my deepest gratitude to my supervisor Dr Rafał Żbikowski for his invaluable guidance, discussions and continuous encouragement during my PhD studies. His help and encouragement are unforgettable.

My sincerest thanks also go to my co-supervisor Professor Brian White for invaluable motivation and encouragement.

This work is an outgrowth of the project WSS/R4519 "Global and Multi-objective Optimisation of Missile Guidance" funded by the CRP Technology Group No. TG03 under the Pathfinder 1999 programme. I would like to acknowledge David East, formerly of DERA, who was instrumental in setting up the project; his gentlemanly support is gratefully acknowledged. John Cleminson of QinetiQ kindly suggested the terminal bunt problem, provided the defining data and shared his solution. His eager support and personal modesty is much appreciated.

I gratefully acknowledge the financial support of the Department of Aerospace, Power and Sensor, College of Defence Technology, Cranfield University.

I would like to thank the Indonesian society for their support and encouragement during my stay in England. Many thanks to my colleagues at the Heaviside Laboratories for making my stay at the Defence Academy a wonderful experience. I am most thankful to my colleagues and the staff at the Department of Aerospace, Power and Sensors for all assistance and support.

I would like to thank my parents, my brothers and sisters for their love, support and encouragement.

Last but not least, my sincerest thanks go to my wife Himmaty, for her support, encouragement and love throughout this difficult experience while raising our family. Also I would like to thank my daughters Ikha, Izzah, Wafa and Khansa for allowing me to spend so much time at the College.

Subchan Shrivenham, September 2005

Abstract

This work focuses on a study of missile guidance in the form of trajectory shaping of a generic cruise missile attacking a fixed target which must be struck from above. The problem is reinterpreted using optimal control theory resulting in two formulations: 1) minimum time-integrated altitude and 2) minimum flight time. Each formulation entails nonlinear, two-dimensional missile flight dynamics, boundary conditions and path constraints. Since the thus obtained optimal control problems do not admit analytical solutions, a recourse to computational optimal control is made. The focus here is on informed use of the tools of computational optimal control, rather than their development.

Each of the formulations is solved using a three-stage approach. In stage 1, the problem is discretised, effectively transforming it into a nonlinear programming problem, and hence suitable for approximate solution with the FORTRAN packages DIRCOL and NUDOCCCS. The results of this direct approach are used to discern the structure of the optimal solution, i.e. type of constraints active, time of their activation, switching and jump points. This qualitative analysis, employing the results of stage 1 and optimal control theory, constitutes stage 2. Finally, in stage 3, the insights of stage 2 are made precise by rigorous mathematical formulation of the relevant two-point boundary value problems (TPBVPs), using the appropriate theorems of optimal control theory. The TPBVPs obtained from this indirect approach are then solved using the FORTRAN package BNDSCO and the results compared with the appropriate solutions of stage 1.

For each formulation (minimum altitude and minimum time) the influence of boundary conditions on the structure of the optimal solution and the

performance index is investigated. The results are then interpreted from the operational and computational perspectives.

Software implementation employing DIRCOL, NUDOCCCS and BNDSCO, which produced the results, is described and documented.

Finally, some conclusions are drawn and recommendations made.

Contents

List of Figures

LIST OF FIGURES

Nomenclature

In general, the definitions below apply unless locally specified in the chapter or section.

Symbol	Description
A_1, A_2, A_3	coefficient of C_d
B_1, B_2	coefficient of C_l
C_1, C_2, C_3	coefficient of air density ρ
$\boldsymbol{f}$	dynamic equation vector
$\boldsymbol{x}, \boldsymbol{x}(t)$	state vector
$\boldsymbol{u}, \boldsymbol{u}(t)$	control vector
$\boldsymbol{\lambda}, \boldsymbol{\nu}$	Lagrange multipliers vector
$\boldsymbol{\psi}$	boundary constraint vector
$\boldsymbol{C}$	mixed inequality constraint
C_d	coefficient of drag
C_l	coefficient of lift
D	drag
g	gravitational constant
H	Hamiltonian
H^{af}	Hamiltonian for minimum altitude problem and no constraint active
H^{ac}	Hamiltonian for minimum altitude problem and all constraints active
H^{aa}	Hamiltonian for minimum altitude problem and altitude constraint active
H^{al}	Hamiltonian for minimum altitude problem and normal acceleration constraint active

Symbol	Description
H^{mtf}	Hamiltonian for minimum time problem and no constraint active
H^{mtc}	Hamiltonian for minimum time problem and all constraints active
H^{mta}	Hamiltonian for minimum time problem and altitude constraint active
H^{mtl}	Hamiltonian for minimum time problem and normal acceleration constraint
h	altitude
h_{min}	minimum altitude
J	performance index (objective function)
J_a	augmented performance index
$\mathcal{L}$	integrand of Lagrange cost term
L	lift
L_{max}	maximum normal acceleration
L_{min}	minimum normal acceleration
m	mass
$\boldsymbol{N}$	interior constraint
ρ	air density
$\boldsymbol{S}$	state inequality constraint
S_{ref}	reference area of the missile
T	thrust
T_{max}	maximum thrust
T_{min}	minimum thrust
t	time
t_0	initial time
t_f	terminal/final time
V	speed
α	angle of attack
δ	variation
γ	flight path angle
Φ	auxiliary function
ϕ	Mayer cost term
μ, ϑ	Lagrange multipliers or adjoint variables

Chapter 1

Introduction

Cruise missiles are guided weapons designed for atmospheric flight whose primary mission is precision strike of fixed targets. This can be achieved only by a judicious approach to guidance, navigation and control (GNC). Navigation is the process of establishing the missile's location. Based on the location, guidance produces the trajectory the missile should follow. Finally, control entails the use of the actuators, so that the missile indeed follows the desired trajectory.

This work deals with an approach to cruise missile guidance known as trajectory shaping. The essence of the approach is to compute an optimal trajectory together with the associated control demand. In other words, for given launch and strike conditions, find a missile trajectory which:

- hits the target in a pre-defined way
- shapes the missile's flight in an optimal fashion
- defines control demand for the optimal flight.

This setting leads naturally to expressing the guidance problem as an optimal control problem. Hence the solution approach for the trajectory shaping involves computational optimal control. This is a set of techniques which combines the theory of infinite dimensional optimisation with numerical methods of finite-dimensional optimisation and boundary value problem solvers. Both the optimal control theory and the numerical algorithms involved are rather non-trivial in nature, and their interaction

adds another layer of complexity. This work focuses on informed use of computational optimal control rather than development of either theory or numerics.

The theoretical and computational tools are used to elucidate the features of the special case of cruise missile trajectory shaping, the terminal bunt manoeuvre, defined in detail in Section 1.1 below. The tools are both powerful and complex. Their power gives insights into optimisation of the manoeuvre—operationally valuable knowledge. The complexity not only challenges the analyst, but uncovers the limitations of the approach and—crucially—elucidates the trade-offs between operationally desirable and computationally tractable. Hence the aims of this work were as follows:

- to formulate several operationally useful variants of trajectory shaping of the terminal bunt manoeuvre
- to analyse the formulations from the point of view of the solution structure, e.g. type of control demand, number and duration of active constraints etc.
- to compute the actual solutions
- to assess the results from the point of view of the analyst, i.e. insights offered and difficulties encountered.

The remainder of this introduction is structured as follows. Section 1.1 defines the terminal bunt manoeuvre and its variants considered later. Section 1.2 is a brief sketch of the historical context and salient features of computational optimal control. Section 1.3 is the outline of the thesis and Section 1.4 summarises its contributions.

1.1 Problem Formulation

Trajectory shaping of a missile is an advanced approach to missile guidance which aims at computing the whole trajectory in an optimal way. The approach underpins this work, focussing on the example of the bunt shaping problem for a cruise missile. The mission is to hit a fixed target while minimising the missile exposure to anti-air defences or minimising the flight time.

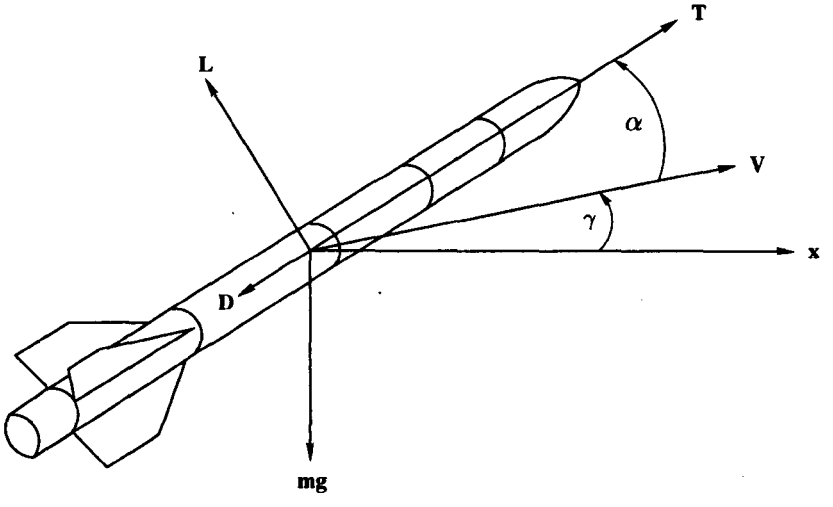


Figure 1.1: Definition of missile axes and angles

1. INTRODUCTION

The problem is to find the trajectory of a generic cruise missile from the assigned initial state to a final state either: 1) with the minimum altitude along the trajectory or 2) in minimum time. The first objective can be formulated by introducing the performance criterion:

$$J = \int_{t_0}^{t_f} h dt, \tag{1.1}$$

while the second objective is:

$$J = \int_{t_0}^{t_f} dt. \tag{1.2}$$

The performance criterion is subject to the equations of motion, which may be written as:

$$\dot{\gamma} = \frac{T-D}{mV}\sin\alpha + \frac{L}{mV}\cos\alpha - \frac{g\cos\gamma}{V} \tag{1.3a}$$

$$\dot{V} = \frac{T-D}{m}\cos\alpha - \frac{L}{m}\sin\alpha - g\sin\gamma \tag{1.3b}$$

$$\dot{x} = V\cos\gamma \tag{1.3c}$$

$$\dot{h} = V\sin\gamma \tag{1.3d}$$

where t is the actual time, $t_0 \leqslant t \leqslant t_f$ with t_0 as the initial time and t_f as the final time. The state variables are the flight path angle γ, speed V, horizontal position x and altitude h of the missile. The thrust magnitude T and the angle of attack α are the two control variables (see Figure 1.1). The aerodynamic forces D and L are functions of the altitude h, velocity V and angle of attack α. The following relationships have been assumed [29]:

Drag. The drag D is written in the form

$$D(h,V,\alpha) = \frac{1}{2}C_d\rho V^2 S_{ref} \tag{1.4}$$

$$C_d = A_1\alpha^2 + A_2\alpha + A_3 \tag{1.5}$$

Lift. The lift L is written in the form

$$L(h,V,\alpha) = \frac{1}{2}C_l\rho V^2 S_{ref} \tag{1.6}$$

$$C_l = B_1\alpha + B_2, \tag{1.7}$$

where ρ is air density given by

$$\rho = C_1 h^2 + C_2 h + C_3 \tag{1.8}$$

and S_{ref} is the reference area of the missile; m denotes the mass and g the gravitational constant.

Boundary conditions. The initial and final conditions for the four state variables are specified:

$$\begin{aligned}
\gamma(0) &= \gamma_0 && \text{(1.9a)}\\
\gamma(t_f) &= \gamma_{t_f} && \text{(1.9b)}\\
V(0) &= V_0 && \text{(1.9c)}\\
V(t_f) &= V_{t_f} && \text{(1.9d)}\\
x(0) &= x_0 && \text{(1.9e)}\\
x(t_f) &= x_{t_f} && \text{(1.9f)}\\
h(0) &= h_0 && \text{(1.9g)}\\
h(t_f) &= h_{t_f}. && \text{(1.9h)}
\end{aligned}$$

In addition, constraints are defined as follows:

- State path constraint

$$V_{min} \leqslant V \leqslant V_{max} \tag{1.10}$$

$$h_{min} \leqslant h \tag{1.11}$$

- Control path constraint

$$T_{min} \leqslant T \leqslant T_{max} \tag{1.12}$$

- Mixed state and control constraint (see equations (1.6)–(1.8))

$$L_{min} \leqslant \frac{L}{mg} \leqslant L_{max} \tag{1.13}$$

where L_{min} and L_{max} are normalised, see Table 1.1.

Table 1.1: Boundary data, constraint data and physical constants.

Quantity	Value	Unit	Quantity	Value	Unit
$V_{min_{t_f}}$	250	m/s	L_{min}	-4	
V_{min}	200	m/s	L_{max}	4	
V_{max}	310	m/s	A_1	-1.9431	
m	1005	kg	A_2	-0.1499	
g	9.81	m/s^2	A_3	0.2359	
S_{ref}	0.3376	m^2	B_1	21.9	
T_{min}	1000	N	B_2	0	
T_{max}	6000	N	C_1	$3.312 \cdot 10^{-9}$	kg m^{-5}
h_{min}	30	m	C_2	$-1.142 \cdot 10^{-4}$	kg m^{-4}
			C_3	1.224	kg m^{-3}

1.2 Historical Context of Computational Optimal Control

The history of optimal control problems could not be separated from the history of the calculus of variations. The history of optimal control reaches back to the famous brachistochrone problem which was proposed by the Swiss mathematician Johann Bernoulli in the seventeenth century, and might be formulated as an optimal control problem (see Bryson [20], Sussmann and Willems [102], Pesch and Bulirsch [78] and Sargent [93]). The problem is to find the quickest descent path between two points with different horizontal and vertical positions. In other words, the problem can be stated as an optimisation problem which is to find $y(x)$ that minimises the objective function:

$$J = \int_{x_0}^{x_f} \sqrt{1 + \frac{dy}{dx}^2}\, dx \tag{1.14}$$

subject to boundary conditions. The above calculus of variations problem can be converted into an optimal control problem by introducing $u = \frac{dy}{dx}$ as a control. Then, the problem is to find the control u that minimises the performance criterion:

$$J = \int_{x_0}^{x_f} \sqrt{1 + u^2}\, dx \tag{1.15}$$

subject to $\frac{dy}{dx} = u$ and the boundary conditions. Following their brilliant solution of the brachistochrone problem, Euler and Lagrange found a necessary condition for extremum of a functional which later known as Euler-Lagrange equation. The development of the extremum of functional became more sophisticated after Legendre, Clebsch, and Jacobi found further necessary conditions (the three necessary conditions of Euler/Lagrange, Legendre/Clebsch, and Jacobi were later proved to be sufficient for a weak local minimum), and, finally, Weierstrass and Carathéodory found, after Hilbert's contribution, sufficient conditions (for a strong local minimum).

A century later, in 1919, Goddard considered the calculus of variations as an important tool to analyse the performance of the rocket trajectory [30]. Subsequently, the variational formulation of the flight paths have been developed by Garfinkel [44, 45], Breakwell et al. [18, 17], Lawden [68] in the formulations of Bolza, Mayer and Lagrange type. The general theory of optimal control was developed by Breakwell [16], Hestenes [55], Pontryagin et al. [84]. The breakthrough, and consequently the birth of a new field in mathematics, optimal control, came with the proof of the maximum principle by Pontryagin, Boltyanskii and Gamkrelidze. Incidentally, their new necessary condition was firstly formulated by Hestenes; his proof, however, still in the context of calculus of variations and thus not as general as the one done by Pontryagin's group. This is because some optimal control problems may be transformed into problems of the calculus of variations, but even "simple" one, e.g. those with controls appearing linearly, cannot be transformed. Some classical books on optimal control are Bellman [5], Bryson and Ho [23], Berkovitz [7], Gamkrelidze [43]. Most early methods were based on finding an analytic solution that satisfied the maximum principle, or related conditions, rather than attempting a direct minimisation of the performance criterion of optimal control problem.

However, the pressing aerospace problems which arose from 1950s onwards did not have analytical solutions. Thus, while the theoretical necessary and sufficient conditions for optimal control were available, effective computation of solutions was still a challenge, compounded by the presence of constraints in real-life problems. The development of digital computer and reliable numerical methods transformed the situation and ushered the era of computational optimal control: a combination of optimal

control theory and the relevant numerics.

Initially, the focus was on approximating the underlying infinite-dimensional problem with a discretised, finite-dimensional version, thus obtaining a nonlinear programming formulation. This approach was given a strong theoretical impetus by the seminal results of Karush and Kuhn and Tucker [4] on optimality conditions for finite-dimensional constrained optimisation. Subsequently, several numerical methods were developed, among which the most important is sequential quadratic programming, or SQP. This method was developed further by many researchers, Powell [87], Gill et al. [48, 47]. Following the rapid development of the SQP methods, it became feasible to obtain numerical solutions of the optimal control problem by transforming the original problem to a nonlinear programming problem. This is done by discretising as state or/and control variables; the approach is known as the direct method.

Bulirsch [99] achieved a major breakthrough when he developed multiple shooting software BOUNDSOL (see Keller [61], Osborne [77]), which was applied successfully to solve several two-point boundary value problems. This enabled an alternative approach to computational optimal control, the indirect method. The essence of the method is first to use optimal control theory to derive the necessary conditions for optimality and then to solve the resulting two-point boundary value problem. Subsequently, BOUNDSOL was developed further by introducing a modified Newton method by Deuflhard [32, 31] and generalising the multiple shooting method for multi-point boundary value problems by Oberle (see the references cited in [76]) which improved convergence of the underlying multiple shooting method. The resulting software BNDSCO became a package of choice and has been used successfully to solve several optimal control problems via the indirect method.

In this work both approaches of computational optimal control, the direct and indirect method, are employed not only for comparison of their pros and cons, but also due to complementary insights into the solution they offer.

1.3 Outline of the Thesis

The remaining chapters of the thesis are organised as follows:

Chapter 2 presents an overview of general optimal control problems and reviews some numerical methods for solving the problems.

Section 2.1 focuses on the nonlinear optimal control problem formulation.

In Section 2.2 a detail of the variational approach is given. This section discusses the importance of constraints, in particular control constraint, mixed inequality constraint and pure state inequality constraint.

In Section 2.3 a nonlinear optimisation approach based on the Karush, Kuhn and Tucker theorem is considered.

In Section 2.4 numerical solution for the optimal control problem is presented. This section focuses on the direct method in 2.4.1 and the indirect method in 2.4.2.

In Section 2.5 summary and discussion are given.

Chapter 3 describes a detailed analysis of the optimal trajectory of a generic cruise missile attacking a fixed target where the target must be struck from above while minimising the missile exposure to anti-air defences.

In Section 3.1 the minimum altitude problem formulation is defined.

In Section 3.2 the computational results of the direct method are used for a qualitative analysis of the main features of the optimal trajectories and their dependence on several constraints.

In Section 3.3 the mathematical analysis based on the qualitative analysis is presented. This section begins with discussing a constraint on the thrust and is followed by path and mixed constraints.

Section 3.4 presents the indirect method approach. The co-state approximation issue is addressed in this section. This section is closed by some numerical solutions using the multiple shooting method package BNDSCO.

Finally, summary and discussion are given in Section 3.5.

Chapter 4 focuses on the optimal trajectories of a generic cruise missile attacking a fixed target in minimum time.

In Section 4.1 the problem formulation is defined for the time-optimal control of the terminal bunt manoeuvre.

Section 4.2 presents some computational results using the direct method and is followed by a qualitative analysis to reveal the structure of the solution.

Section 4.3 contains the mathematical analysis of the time-optimal control problem based on the qualitative analysis.

In Section 4.4 the numerical solutions using the multiple shooting package BNDSCO are obtained and compared with the results of DIRCOL and NUDOCCCS.

Finally, Section 4.5 gives summary and discussion.

Chapter 5 deals with software implementation of the terminal bunt problem using three different packages.

In Section 5.1 the DIRCOL implementation is given for the case of minimum time problem.

In Section 5.2 the minimum time problem is solved using NUDOCCCS.

In Section 5.3 the multiple shooting package BNDSCO implementation of the minimum time problem is shown.

Chapter 6 presents the conclusions of the thesis and recommendations for future work.

1.4 Thesis Contributions

As explained earlier in this chapter, this work focuses on informed use of computational optimal control for solving the terminal bunt manoeuvre, rather than development of either the underlying theory or the relevant numerics. In this context, the main contributions of this thesis are as follows:

- formulating trajectory shaping missile guidance as an optimal control problem for the case of terminal bunt manoeuvre
- devising two formulations of the problem:

  - minimum time-integrated altitude
  - minimum flight time

- proposing a three-stage hybrid approach to solve each of the problem formulations
  - stage 1: solution structure exploration using a direct method
  - stage 2: qualitative analysis of the solution obtained in stage 1, using optimal control
  - stage 3: mathematical formulation of the TPBVP based on the qualitative analysis of stage 2
- solving each of the problem formulations using the three-stage hybrid approach
  - stage 1: by using DIRCOL/NUDOCCCS solvers
  - stage 2: by using results of stage 1, understanding the underlying flight dynamics and employing optimal control theory
  - stage 3: by using optimal control theory and the BNDSCO solver
- analysing influence of boundary conditions on the structure of the optimal control solution of each problem formulation and the resulting values of the performance index
- interpreting the results from the operational and computational perspectives, pointing out the trade-offs between two
- using effectively DIRCOL, NUDOCCCS and BNDSCO and documenting their use.

Chapter 2

Optimal Control: Outline of the Theory and Computation

The main purpose of this chapter is to provide an overview of the aspects of optimal control necessary to analyse and solve the terminal bunt manoeuvre. The optimal control problem has been studied in many textbooks (see e.g. Pontryagin [84], Bellman [73], Athans and Falb [3], Kirk [63]. Leitmann [69, 70], Bryson and Ho [23], Lewis and Syrmos [72], Vinh [107], Betts [11], Naidu [75]) and survey papers (see e.g. Hartl, Sethi, and Vickson [53], Pesch [79, 80, 81, 82]). These sources have treated the optimal control problem in depth.

This chapter organised as follows. Section 2.1 concerns the general nonlinear optimal control problem formulation. As an example, a detail of the variational approach derivation is given in Section 2.2. This section discusses the important issue of constraints which are: control constraints in 2.2.1, mixed inequality constraints in 2.2.2 and pure state inequality constraints in 2.2.3. A nonlinear optimisation approach based on the Karush, Kuhn and Tucker theorem is considered in Section 2.3. Numerical solution for the optimal control problem is presented in Section 2.4. The direct method approach is considered in 2.4.1, followed by the indirect approach in 2.4.2. Finally, Section 2.5 presents summary and discussion.

2.1 The Optimal Control Problem

The problem is to find an admissible control $\boldsymbol{u}(t)$, which minimises the performance index:

$$\min_{\boldsymbol{u}\in\mathcal{U}} J = \phi[\boldsymbol{x}(t_f), t_f] + \int_{t_0}^{t_f} \mathcal{L}[\boldsymbol{x}(t), \boldsymbol{u}(t), t]dt \tag{2.1}$$

with respect to the state vector functions:

$$\mathcal{X} = \{\boldsymbol{x}\colon [0, t_f] \to \mathbb{R}^n |\ x_i, i = 1, \ldots, n, \text{piecewise continuously differentiable}\}, \tag{2.2}$$

and the control vector functions:

$$\mathcal{U} = \{\boldsymbol{u}\colon [0, t_f] \to U \subset \mathbb{R}^m |\ u_i, i = 1, \ldots, m, \text{piecewise continuous}\}, \tag{2.3}$$

subject to the following constraints:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \quad \boldsymbol{f}\colon \mathbb{R}^{n+m} \to \mathbb{R}^n \tag{2.4}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_0 \in \mathbb{R}^n \quad \boldsymbol{x}_0 \text{ known} \tag{2.5}$$

$$\boldsymbol{\psi}(\boldsymbol{x}(t_f), t_f) = 0 \in \mathbb{R}^p \quad \boldsymbol{\psi}\colon \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}^p,\ p \leqslant n,\ t_f \text{ unknown} \tag{2.6}$$

$$\boldsymbol{C}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leqslant 0 \in \mathbb{R}^q \quad \boldsymbol{C}\colon \mathbb{R}^{n+m} \to \mathbb{R}^q \tag{2.7}$$

$$\boldsymbol{S}(\boldsymbol{x}(t)) \leqslant 0 \in \mathbb{R}^s \quad \boldsymbol{S}\colon \mathbb{R}^n \to \mathbb{R}^s \tag{2.8}$$

The performance index describes a quantitative measure of the performance of the system over time. In aerospace problems, a typical performance index gives an appropriate measure of the quantities such as minimum fuel/energy, optimal time etc. Here $\phi\colon \mathbb{R}^{n+1} \to \mathbb{R}^1$ and $\mathcal{L}\colon \mathbb{R}^{n+m} \to \mathbb{R}^1$ are assumed to be sufficiently often continuously differentiable in all arguments. The type of performance index (2.1) is said to be in the Lagrange form when $\phi \equiv 0$ and in the Mayer form when $\mathcal{L} \equiv 0$ (see Oberle and Grimm [76]). Furthermore, it is in the linear Mayer form when $\mathcal{L} \equiv 0$ and ϕ is linear.

Minimising J with respect to the control function $\boldsymbol{u}$ must be accomplished in a way consistent with the dynamics of the system, whose performance is optimised. In other words, equation (2.4) is the first fundamental equality constraint. The optimal control $\boldsymbol{u}^*$, when substituted to (2.4), will produce the optimal state $\boldsymbol{x}^*$, while minimising J.

The optimal state $\boldsymbol{x}^*$ is further constrained by the boundary conditions (2.5) and (2.6): in our case the launch and strike conditions. These are point constraints, i.e. the act only at the selected points of the trajectory—t_0 and t_f—as opposed to the path constraints, valid for (t_0, t_f) and discussed below. It is a remarkable feature of optimal control problems that changes in the terminal conditions (2.6) may have a profound impact on the structure of the solution throughout the whole interval (t_0, t_f).

The optimal control $\boldsymbol{u}^*$ and optimal state $\boldsymbol{x}^*$ are subject to path constraints (2.7) and (2.8). Unlike boundary conditions (2.5)–(2.6), these conditions must be satisfied along the trajectory, i.e. on (t_0, t_f), which is a more demanding requirement than for the point constraints. In further contrast to (2.5)–(2.6), the path constraints are inequality constraints, making their analysis more involved. This is briefly discussed now, separately for (2.7) and (2.8).

In the case of (2.7), either (i) $\boldsymbol{C} = 0$ or (ii) $\boldsymbol{C} < 0$, and establishing the subintervals of (t_0, t_f) when (i) occurs is of fundamental importance. If the constraint is active, case (i), then (2.4) and (2.7) become a system of differential algebraic equations. Indeed, equation (2.7) then implicitly defines the state $\boldsymbol{x}$ as a function of control $\boldsymbol{u}$, effectively lowering the dimension of the original system of controlled ordinary differential equations (2.4). It is important to note that, when (2.7) is active, the algebraic relationship between $\boldsymbol{x}$ and $\boldsymbol{u}$ is, at least in principle, clear, provided that the assumptions of Implicit Function Theorem hold.

In the case of (2.8), again, either (i) $\boldsymbol{S} = 0$, or (ii) $\boldsymbol{S} < 0$, and the occurrence of (i) is the key issue. However, the situation is now more challenging compared with the previous one, $\boldsymbol{C} = 0$, because it is not explicit how $\boldsymbol{S} = 0$ constraints the choice of $\boldsymbol{u}$ and thus how to modify the search for optimal control. Various approaches are possible, and are discussed in Sections 2.2.3 and 3.4.2, but it should be noted that the presence of a pure state constraint (2.8) is always a challenge in the context of optimal control. By contrast, the mixed (state and control) constraint (2.7) is easier to deal with, due to the explicit presence of control $\boldsymbol{u}$.

Finally, it should be mentioned that (2.7) or (2.8) may be active on a subinterval of (t_0, t_f) or just at a point. In the former case, the constrained (active) subarc will be

characterised by the entry time t_1 and the exit time t_2 with $t_0 \leq t_1 < t_2 \leq t_f$. In the latter case, the subarc collapses to a single (touch) point, $t_1 = t_2$.

The functions appearing in (2.1)–(2.8) are assumed to be sufficiently continuously differentiable with respect to their arguments. Note that the definition of $\mathcal{U}$ allows discontinuities in controls and thus implies corners (cusps) in the states, so that $\mathcal{X}$ comprises *piecewise* smooth functions. This is a practical necessity, as many real-world applications of optimal control involve bang-bang type inputs.

Problem (2.1)–(2.8) is infinite-dimensional: its solution is not a finite vector of numbers, but a function. For a real-life application it is impossible to guess the optimal function, so a recourse to approximate methods is necessary. They attempt to find a finite-dimensional representation of the solution which is accurate at the nodes of the representation, has acceptable error between the nodes and converges to the true function as the number of nodes tends to infinity, if second order sufficient conditions hold.

There are two main approaches to the solution of the problem. The direct approach replaces the continuous time interval with a grid of discrete points, thus approximating it with a finite-dimensional problem, albeit of high dimension (hundreds of discretised variables). The indirect approach preserves the infinite-dimensional character of the task and uses the theory of optimal control to solve it. This is the focus of Section 2.4.2, while the direct approach is treated in Section 2.4.1.

2.2 Variational Approach to Problem Solution

The indirect approach to solution of the optimal control problem is based on a generalisation of the calculus of variations. Necessary conditions for an extremum are derived by considering the first variation of the performance index J with constraints adjoined in the manner of Lagrange. Since the setting is infinite-dimensional, the familiar Lagrange multipliers are now functions of time, $\boldsymbol{\lambda} = \boldsymbol{\lambda}(t)$, and are called co-states in analogy to the system state $\boldsymbol{x} = \boldsymbol{x}(t)$. While in the finite-dimensional case the multipliers are computed from algebraic equations, the co-states obey a differential equation.

The necessary conditions thus entail both the original differential equations of the underlying dynamical system and the associated adjoint differential equations of the co-states. The end result is a two-point boundary value problem (TPBVP) which is made up of the state and co-states equations together with the initial and terminal conditions.

The approach is called indirect, because the optimal control is found by solving the auxiliary TPBVP, rather than by a direct focus on the original problem.

Here, we consider the general nonlinear optimal control problem given by equations (2.1)–(2.6). The performance index is given in the Bolza form, so it contains a final cost function in addition to the general cost function. Introducing the Lagrange multiplier $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ and adjoining the dynamic equations and the boundary conditions to the performance index, we obtain the following augmented performance index:

$$\begin{aligned} J_a &= \phi[\boldsymbol{x}(t_f), t_f] + \boldsymbol{\nu}^T \boldsymbol{\psi}[\boldsymbol{x}(t_f), t_f] \\ &\quad + \int_{t_0}^{t_f} \left[\mathcal{L}[\boldsymbol{x}(t), \boldsymbol{u}(t), t] + \boldsymbol{\lambda}^T \left\{\boldsymbol{f}[\boldsymbol{x}(t), \boldsymbol{u}(t), t] - \dot{\boldsymbol{x}}\right\}\right] dt \end{aligned} \tag{2.9}$$

The first order necessary conditions can be derived by applying the variational approaches as follows:

$$\begin{aligned} \delta J_a &= \frac{\partial \phi}{\partial \boldsymbol{x}(t_f)} \delta \boldsymbol{x}_f + \frac{\partial \phi}{\partial t_f} \delta t_f + \delta \boldsymbol{\nu}^T \boldsymbol{\psi} + \boldsymbol{\nu}^T \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{x}(t_f)} \delta \boldsymbol{x}_f + \boldsymbol{\nu}^T \frac{\partial \boldsymbol{\psi}}{\partial t_f} \delta t_f \\ &\quad + (\mathcal{L} + \boldsymbol{\lambda}^T . (\boldsymbol{f} - \dot{\boldsymbol{x}}) \mid_{t=t_f} \delta t_f + (\mathcal{L} + \boldsymbol{\lambda}^T . (\boldsymbol{f} - \dot{\boldsymbol{x}}) \mid_{t=t_0} \delta t_0 \\ &\quad + \int_{t_0}^{t_f} \left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}} \delta \boldsymbol{x} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}} \delta \boldsymbol{u} + \delta \boldsymbol{\lambda}^T . (\boldsymbol{f} - \dot{\boldsymbol{x}}) \right. \\ &\quad \left. + \boldsymbol{\lambda}^T . \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \delta \boldsymbol{x} + \boldsymbol{\lambda}^T . \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \delta \boldsymbol{u} - \boldsymbol{\lambda}^T \delta \dot{\boldsymbol{x}} \right] dt \end{aligned} \tag{2.10}$$

Integrate by parts the last term of (2.10)

$$\int_{t_0}^{t_f} -\boldsymbol{\lambda}^T \delta \dot{\boldsymbol{x}} = -\boldsymbol{\lambda}^T(t_f) \delta \boldsymbol{x}(t_f) + \boldsymbol{\lambda}^T(t_0) \delta \boldsymbol{x}(t_0) + \int_{t_0}^{t_f} \dot{\boldsymbol{\lambda}}^T \delta \boldsymbol{x} dt \tag{2.11}$$

Since the final time t_f is free, the variation between the final state, $\delta \boldsymbol{x}_f$ and the state at the final time, $\delta \boldsymbol{x}(t_f)$ are different and can be defined as follows

$$\delta \boldsymbol{x}_f = \delta \boldsymbol{x}(t_f) + \dot{\boldsymbol{x}}(t_f) . \delta t_f \tag{2.12}$$

By substituting equations (2.11) and (2.12) into (2.10) we obtain

$$\begin{aligned}\delta J_a &= \left(\frac{\partial \phi}{\partial x(t_f)} + \nu^T \frac{\partial \psi}{\partial x(t_f)} - \lambda^T(t_f)\right)\delta x_f + \delta \nu^T \psi \\ &+ \left(\frac{\partial \phi}{\partial t_f} + \nu^T \frac{\partial \psi}{\partial t_f} + (\mathcal{L} + \lambda^T.(f - \dot{x}) + \lambda^T \dot{x}) \mid_{t=t_f}\right)\delta t_f \\ &+ (\mathcal{L} + \lambda^T.(f - \dot{x}) + \lambda^T \dot{x}) \mid_{t=t_0} + \lambda^T \delta x(t_0) \\ &+ \int_{t_0}^{t_f} \left[\left(\frac{\partial \mathcal{L}}{\partial x} + \lambda^T \frac{\partial f}{\partial x} + \dot{\lambda}^T\right)\delta x + \left(\frac{\partial \mathcal{L}}{\partial u} + \lambda \frac{\partial f}{\partial u}\right)\delta u \right. \\ &\left. + \delta \lambda^T (f - \dot{x})\right] dt \end{aligned} \tag{2.13}$$

The extremum of the functional J is obtained when the first variation δJ_a vanishes. Thus the necessary conditions can be established by setting the coefficients of the independent variations δx, δu, $\delta \lambda$ and $\delta \nu$ of (2.13) equal zero. The initial state $x(t_0)$ and initial time t_0 are given in this case, consequently $\delta x(t_0)$ and δt_0 are both zero. The boundary at the terminal conditions are given by the first and third component of (2.13). In summary, the necessary conditions for J to have an extremum value are

- State equation

$$\dot{x} = f[x(t), u(t), t] \tag{2.14}$$

- Co-state equation

$$-\dot{\lambda}^T = \frac{\partial \mathcal{L}}{\partial x} + \lambda^T \frac{\partial f}{\partial x} \tag{2.15}$$

- Stationarity condition

$$0 = \frac{\partial \mathcal{L}}{\partial u} + \lambda \frac{\partial f}{\partial u} \tag{2.16}$$

- Boundary condition

$$\left[\frac{\partial \phi}{\partial x} + \nu^T \frac{\partial \psi}{\partial x} - \lambda^T\right]_{t_f} \delta x_f + \left[\frac{\partial \phi}{\partial t_f} + \nu^T \frac{\partial \psi}{\partial t_f} + H\right]\delta t_f = 0 \tag{2.17}$$

where $H = \mathcal{L} + \lambda^T.f$

If the final time and the final state are both free, the boundary conditions (2.17) can be rewritten as

$$\begin{aligned} \boldsymbol{\lambda}(t_f)^T &= \frac{\partial \phi}{\partial \boldsymbol{x}(t_f)} + \boldsymbol{\nu}^T \frac{\partial \psi}{\partial \boldsymbol{x}(t_f)} \\ 0 &= \frac{\partial \phi}{\partial t_f} + \boldsymbol{\nu}^T \frac{\partial \psi}{\partial t_f} + H(t_f) \end{aligned}$$

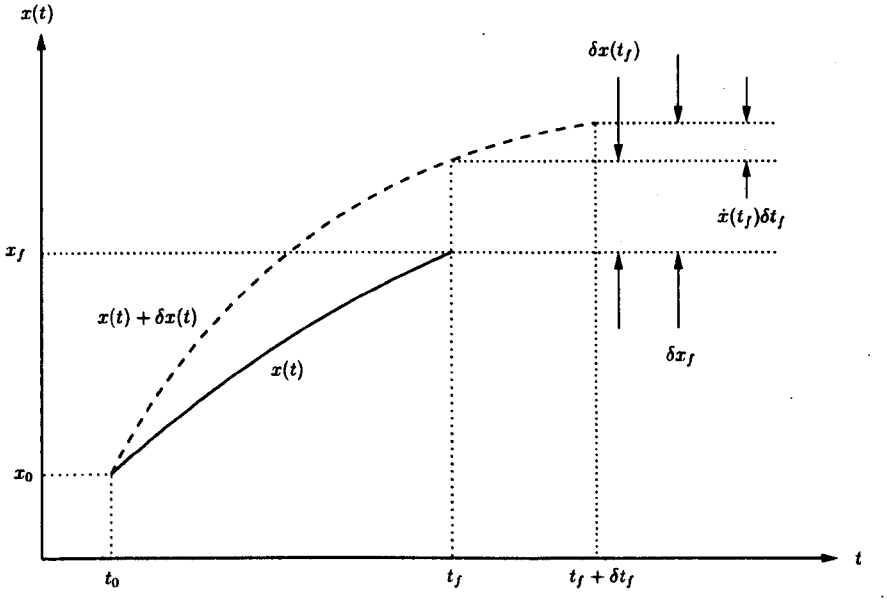


Figure 2.1: The difference between δx_f and $\delta x(t_f)$

Similarly, the derivation of the necessary conditions can be done by defining the Hamiltonian and the auxiliary function as follows

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) \triangleq \mathcal{L}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{\lambda}^T \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{2.18}$$

$$\Phi(\boldsymbol{x}, t, \boldsymbol{\nu}) \triangleq \phi(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{\nu}^T \psi(\boldsymbol{x}, \boldsymbol{u}) \tag{2.19}$$

Here λ: $[0, t_f] \rightarrow \mathbb{R}^n$ and $\nu \in \mathbb{R}^p$ denote Lagrange multipliers or adjoint variables. The following necessary conditions (see references [23], [82]) are obtained:

- differential equations of Euler-Lagrange

$$\dot{x}^T = H_\lambda = \frac{\partial H}{\partial \lambda} \tag{2.20}$$

$$\dot{\lambda}^T = -H_x = -\frac{\partial H}{\partial x} \tag{2.21}$$

- minimum principle

$$u = \arg\min_{u \in U} H(x, \lambda) \tag{2.22}$$

- transversality conditions

$$\lambda^T(t_f) = \Phi_x|_{t=t_f} \tag{2.23}$$

$$(\Phi_t + H)|_{t=t_f} = 0 \tag{2.24}$$

If u appears nonlinearly in H, the control function can eliminated as a function of x and λ. This can be obtained in most practical problems explicitly:

$$u = u(x, \lambda). \tag{2.25}$$

Otherwise, the solution can be computed iteratively from the implicit equation $H_u = 0$, provided that the assumptions of the Implicit Function Theorem hold, too. Note that, H_u and $H_{uu} > 0$ (positive definite) are sufficient conditions for the necessary condition of the minimum principle (2.22) to hold, if U is an open set. The latter condition $H_{uu} > 0$, respectively $H_{uu} \geq 0$ (positive semidefinite), is also known as the necessary condition of Legendre-Clebsch, respectively strengthened necessary condition of Legendre-Clebsch in the calculus of variations.

2.2.1 Control Constraints

In this case the constraint contains the control $u(t)$ only:

$$C(u(t)) \leqslant 0.$$

Therefore the constraint can be adjoined directly to the Hamiltonian by a Lagrange multiplier. If $\boldsymbol{u}$ appears linearly in H, firstly we assume that $m = 1$ and $U = [u_{min}, u_{max}]$. The equation (2.18) can be written in the form

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) = H_1(\boldsymbol{x}, \boldsymbol{\lambda}) + uH_2(\boldsymbol{x}, \boldsymbol{\lambda}). \tag{2.27}$$

H_u does not determine the optimal control solution. If the second term $H_2(\boldsymbol{x}, \boldsymbol{\lambda})$ does not vanish identically on subinterval $[t_{entry}, t_{exit}]$ of $[0, t_f]$ with $t_{entry} < t_{exit}$, the minimum principle yields

$$u = \begin{cases} u_{max} & \text{if } H_2 < 0 \\ u_{min} & \text{if } H_2 > 0 \end{cases}$$

H_2 is called the switching function associated with the control variable $\boldsymbol{u}$. However, if H_2 vanishes on a subinterval of $[0, t_f]$, the control variable u has a singular subarc. In this case the optimal control variable can be computed by successive differentiation of the switching function H_2 with respect to time t until the control variable appears explicitly (see Bryson and Ho [23, page 110]). The case for vector $\boldsymbol{u}$ is treated similarly, but may be more involved.

2.2.2 Mixed State-Control Inequality Constraints

In this section, the constraint includes the state and control variables:

$$\boldsymbol{C}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leqslant 0.$$

The mixed inequality constraint can be adjoined directly to the Hamiltonian as in the previous section. For simplicity, we assume that $m = q = 1$ and using the augmented Hamiltonian

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) := \mathcal{L}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{\lambda}^T \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) + \mu \boldsymbol{C}(\boldsymbol{x}, \boldsymbol{u}). \tag{2.29}$$

Necessary conditions for minimising the Hamiltonian then can be derived. The Lagrangian parameter μ is

$$\mu = \begin{cases} 0 & \text{if } \boldsymbol{C} < 0 \\ \mu \geqslant 0 & \text{if } \boldsymbol{C} = 0 \end{cases}$$

The Euler-Lagrange equations become

$$\lambda^T = -H_x = \begin{cases} \mathcal{L}_x - \lambda^T f_x & \text{if } C < 0 \\ \mathcal{L}_x - \lambda^T f_x - \mu C_x & \text{if } C = 0 \end{cases}$$

The control $u(t)$ along the constrained arc can be derived from the mixed constraints:

$$C(x,u) = 0 \text{ for all } t \text{ with } t_1 \leqslant t \leqslant t_2 \text{ and } t_1 < t_2, \tag{2.30}$$

the control variable can be represented by a function,

$$u = u(x) \tag{2.31}$$

if equation (2.30) can be uniquely solved for u. If $C_u \neq 0$, the multiplier μ is given by (2.22):

$$H_u \triangleq \mathcal{L}_u + \lambda^T f_u + \mu C_u. \tag{2.32}$$

2.2.3 State Inequality Constraints

We now summarise some results of optimal control theory for problems with a state variable inequality constraint (2.8) based on the Bryson's formulation. Consider now the following equation:

$$S(x(t)) \leqslant 0, \quad S : \mathbb{R}^n \to \mathbb{R}^s.$$

For simplicity, we us assume that $m = s = 1$ and that the constraint is active on a subinterval

$$S(x(t)) \equiv 0 \quad \text{for all} \quad t \in [t_1, t_2] \subset [0, t_f]. \tag{2.34}$$

We take successive total time derivatives of (2.8) and substitute $f(x(t), u(t))$, until we obtain explicit dependence on u. We obtain on $[t_1, t_2]$

$$S(x) \equiv 0, S^{(1)}(x) \equiv 0, \ldots, S^{(r-1)}(x) \equiv 0 \tag{2.35}$$

$$\text{with} \quad S^{(r)}(x,u) = 0. \tag{2.36}$$

If r is the smallest non-negative number such that (2.36) holds, r is called the order of the state constraint. Here $S^{(r)}(x,u)$ plays the role of $C(x,u)$ in (2.29) so that the Hamiltonian is

$$H(x,u,\lambda,\mu) = \mathcal{L} + \lambda^T f + \mu S^{(r)}. \tag{2.37}$$

Again for μ we have

$$\mu = \begin{cases} 0 & \text{if } S^{(r)} < 0 \\ \mu \geqslant 0 & \text{if } S^{(r)} \equiv 0 \end{cases}$$

The control $\boldsymbol{u}$ on the constrained arcs can be derived from (2.36) and μ from (2.22). The right-hand sides of the differential equations for the adjoint variables (2.21) are to be modified along $[t_1, t_2]$. In order to guarantee that not only (2.36) but also (2.35) is satisfied, we have to require that the so-called entry conditions are fulfilled:

$$N^T(\boldsymbol{x}(t_1), t_1) := (S(\boldsymbol{x}(t_1)), S^{(1)}(\boldsymbol{x}(t_1)), \ldots, S^{(r-1)}(\boldsymbol{x}(t_1)) = 0. \tag{2.38}$$

Therefore $\boldsymbol{\lambda}$ generally is discontinuous at t_1 and continuous at t_2. Sometimes boundary points occur instead of boundary arcs. If, for example, the order is $r = 2$, the following conditions hold:

$$S(\boldsymbol{x}(t_b)) = 0 \qquad S^{(1)}(\boldsymbol{x}(t_b)) = 0. \tag{2.39}$$

The first condition is regarded as an interior point condition and yields a possible discontinuity of $\boldsymbol{\lambda}$; the second condition determines t_b. Singular arcs are treated in a similar manner, leading to multipoint boundary value problems with jump conditions and switching functions.

2.3 Nonlinear Programming Approach to Solution

As seen from Section 2.2, the indirect approach entails a considerable amount of rather non-trivial theoretical conditions. The relevant conditions have to be applied judiciously in order to formulate a TPBVP appropriate to the optimal control problem in question. Then the resulting TPBVP has to be solved and the optimal control $\boldsymbol{u}^*$ calculated from the TPBVP solution including the optimal state $\boldsymbol{x}^*$ and co-state $\boldsymbol{\lambda}^*$.

An alternative, aimed at avoiding the above complications, is to discretise the original problem (2.1)–(2.8) and interpret the result as a finite-dimensional optimisation problem. This approximation will result in a nonlinear programming (NLP) problem with equality and inequality constraints, possibly of high dimension due to the fineness of discretisation grid. Hence we begin by recalling the basics of the NLP problem and associated necessary conditions for optimality.

Suppose we have the optimisation problem as follows:

$$\min \quad f(\boldsymbol{x}) \tag{2.40a}$$

$$\text{subject to} \quad \mathrm{g}_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m \tag{2.40b}$$

where the objective and the constraint functions (2.40) are assumed to be continuously differentiable. The problem is to find such a solution $\boldsymbol{x}^*$ that minimises the objective function, out of all possible solutions $\boldsymbol{x}$ that satisfy the constraints. Introduce the Lagrange function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\mu})$:

$$\mathcal{L}[\boldsymbol{x}, \boldsymbol{\mu}] = f(\boldsymbol{x}) + \sum_{i=1}^{m} \mu_i \mathrm{g}_i(\boldsymbol{x}), \quad i = 1, \ldots, m \tag{2.41}$$

where $\mu_i, i = 1, \ldots, m$ are known as the Lagrange multipliers.

The theorem of Karush, Kuhn and Tucker gives first order necessary conditions for a point $\boldsymbol{x}$ to be a local minima.

Theorem Karush–Kuhn–Tucker (KKT) necessary conditions: Given the optimisation problem (2.40), where $f(\boldsymbol{x}), \mathrm{g}_i(\boldsymbol{x}), i = 1, \ldots, m$ are differentiable. Let $\boldsymbol{x}^*$ be a point satisfying all constraints, and let $\nabla \mathrm{g}_i(\boldsymbol{x}^*), i = 1, \ldots, m$ be linearly independent at $\boldsymbol{x}^*$. If $\boldsymbol{x}^*$ is a local optimum of (2.40) then there is exists scalars $\mu_i^*, i = 1, \ldots, m$ such that

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{m} \mu_i^* \nabla \mathrm{g}_i(\boldsymbol{x}^*) = 0 \tag{2.42a}$$

$$\mu_i^* \mathrm{g}_i(\boldsymbol{x}^*) = 0, \quad i = 1, \ldots, m \tag{2.42b}$$

$$\mu_i^* \geq 0, \quad i = 1, \ldots, m \tag{2.42c}$$

The equation (2.42a) is equivalent to the equation $\nabla_x \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\mu}^*) = 0$; the scalars μ_i are called Lagrange multipliers. For a proof for the above theorem, see e.g. [4]. Let us focus on the Mayer type of the performance index of (2.1):

$$J = \phi[\boldsymbol{x}(t_f), t_f] \tag{2.43}$$

subject to dynamic equations

$$\dot{\boldsymbol{x}} = \boldsymbol{f}[\boldsymbol{x}(t), \boldsymbol{u}(t), t] \tag{2.44}$$

Consider the nonlinear programming problem as follows

$$\boldsymbol{Y}^T = [\boldsymbol{u}(t_1), \ldots, \boldsymbol{u}(t_N), \boldsymbol{x}(t_1), \ldots, \boldsymbol{x}(t_N)] \tag{2.45}$$

where: $t_0 = t_1 < t_2 < \ldots < t_N = t_f$, defining $h_d = t_f/N$ or $t_{k+1} = t_k + h_d$. The equation (2.44) can be approximated by the Euler method:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}[\boldsymbol{x}(t), \boldsymbol{u}(t), t] \approx \frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}(t_k)}{h_d} \tag{2.46}$$

for sufficiently small h_d. The optimal control problem (2.43)–(2.44) can be transformed as follows: The objective function (2.43) can be rewritten in discrete approach:

$$J = \phi[\boldsymbol{x}(t_N)] \tag{2.47}$$

The dynamic equation (2.46) can be rewritten as follows:

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}(t_k) - h_d \boldsymbol{f}[\boldsymbol{x}(t_k), \boldsymbol{u}(t_k), t_k] = 0 \tag{2.48}$$

and becomes an equality constraint. Thus the Lagrangian for the discrete optimal control above is:

$$\mathcal{L}[\boldsymbol{x}, \boldsymbol{\mu}] = \phi[\boldsymbol{x}(t_N)] + \sum_{i=1}^{m-1} \mu_i [\boldsymbol{x}_{k+1} - \boldsymbol{x}(t_k) - h_d \boldsymbol{f}[\boldsymbol{x}(t_k), \boldsymbol{u}(t_k), t_k]] \tag{2.49}$$

The KKT necessary conditions for the discrete approaches are:

$$\frac{\partial \mathcal{L}}{\partial \mu_k} = \boldsymbol{x}_{k+1} - \boldsymbol{x}(t_k) - h_d \boldsymbol{f}[\boldsymbol{x}(t_k), \boldsymbol{u}(t_k), t_k] = 0 \tag{2.50}$$

$$\frac{\partial \mathcal{L}}{\partial x_k} = (\mu_k - \mu_{k-1}) + h_d \mu_k^T \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_k} = 0 \tag{2.51}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_k} = h_d \mu_k^T \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_k} = 0 \tag{2.52}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_{t_N}} = \boldsymbol{\mu}_{t_N - 1} + \frac{\partial \phi}{\partial \boldsymbol{x}_{t_N}} = 0 \tag{2.53}$$

Equations (2.50)–(2.53) can be an estimator of the equations (2.14)–(2.18) by letting $N \rightarrow \infty$ and $h_d \rightarrow 0$. Thus the KKT necessary conditions can be used as an estimator of the optimal control necessary conditions.

2.4 Numerical Solution of the Optimal Control Problem

Numerical solution of the optimal control problem can be categorised into two main approaches. The first approach corresponds to the direct method which is based on discretisation of state and/or control variables over time, so that an NLP solver can be used. The second approach corresponds to the indirect method. The first step of this method is to formulate the appropriate TPBVP and the second step is to solve the TPBVP numerically.

2.4.1 Direct Method Approach

Direct methods are based on the transformation of the original optimal control problem into nonlinear programming (NLP) by discretising the state and/or control history and then solving the resulting NLP problem. A variety of direct methods has been developed and applied. Gradient algorithms were proposed by Kelley [62] and by Bryson and Denham [21]. Pytlak solved a state constrained optimal control problem using a gradient algorithms and applied it for some problems (see [86], [85]). Hargraves and Paris [52] reintroduced the direct transcription approach, by discretising the dynamic equations using a collocation method. A cubic polynomial is used to approximate the state variables and linear interpolation for the control variables. The collocation scheme was originally used by Dickmanns and Well [33] to solve TPBVPs. Seywald et al. introduced an approach based on the representation of the dynamical system in terms of differential inclusions. This method employs the concepts of hodograph space and attainable sets (see [67, 95, 94, 96]). Direct transcriptions have been presented in detail by many researchers, e.g., Betts et al. [11, 12, 13, 9, 8, 14, 10, 15], Enright and Conway [39, 38], Herman [54], Tang and Conway [103], Ross and Fahroo [41, 91, 90], Elnagar et al. [42, 35, 36, 37].

Based on the discretisation of the state and/or control, direct methods can be categorised into three different approaches.

The first approach is based on state and control variables parameterisation. Both the control and the state are discretised and then the resulting discretisation is solved using

an NLP solver. In Section 2.4.1.1, the direct collocation approach based on the full discretisation of the state and control is given and compared with partial discretisation in which just the control is discretised while the state is obtained recursively. In Section 2.4.1.2 the Legendre pseudospectral method is considered. In this method the state and control variables are approximated using the Lagrange interpolation polynomial.

The second approach is control parameterisation, so that the state and performance index can be solved by numerical integration. This approach is known as control parameterisation and will be discussed in Section 2.4.1.3. The idea of control parameterisation is to approximate the control variables and compute the state variables by integrating the state equations. The control variables can be approximated by choosing an appropriate function with finitely many unknown parameters. This method is presented by Rosenbrock and Storey [88], Hicks and Ray [56] as follows

$$u(t) = \sum_{i=0}^{n} a_i \phi_i(t), \tag{2.54}$$

where a_i denote unknown parameters and $\phi_i(t)$ are some polynomial functions. Hicks and Ray [56] reported the difficulty of these methods. Brusch [19] introduced piecewise polynomials for the control approach in equation (2.54). The modification can handle constraints efficiently. These methods can be found in many research papers and books, e.g. Teo, Jennings, Lee and Rehbock [104] have studied control parameterisation by introducing variable switching time into equivalent standard optimal control problems involving piecewise constant or piecewise linear control functions with prefixed switching times. Control parameterisation with direct shooting method has been studied and applied for mechanical multi-body systems by Gerdts [46] where the control is parameterised by the B-spline function. Further references include Goh and Teo [50], Teo, Goh and Wong [106], Teo and Wong [105], Kraft [65, 64].

The third approach is based on the state parameterisation only (see [97]). Jaddu and Shimemura [58] solved unconstrained nonlinear optimal control problems by transforming them into a sequence of quadratic programming problem and state parameterisation. They extended it for constrained nonlinear optimal control problems using the Chebyshev polynomials for the state parameterisation (see [59, 60]).

2.4.1.1 Direct Collocation Approach

The basic approach for solving optimal control problem by direct collocation approach is to transform the optimal control problem into sequence of nonlinear constrained optimisation problems by discretising of the state and/or control variables. Two approaches will be considered.

The first approach is based on the discretisation of both the state and control variables. The following derivation is mainly taken from von Stryk and Bulirsch [110].

The duration time of the optimal trajectory is divided into subinterval as follows:

$$t_0 = t_1 < t_2 < t_3 \ldots < t_k = t_f \tag{2.55}$$

The state and control variables at each node is $\boldsymbol{x}_j = \boldsymbol{x}(t_j)$ and $\boldsymbol{u}_j = \boldsymbol{u}(t_j)$, such that the state and control variables at the nodes are defined as nonlinear programming variables:

$$\boldsymbol{Y} = [\boldsymbol{u}(t_1), \ldots, \boldsymbol{u}(t_k), \boldsymbol{x}(t_1), \ldots, \boldsymbol{x}(t_k)]. \tag{2.56}$$

The controls are chosen as piecewise linear interpolating functions between $\boldsymbol{u}(t_j)$ and $\boldsymbol{u}(t_{j+1})$ for $t_j \leq t \leq t_{j+1}$ as follows:

$$\boldsymbol{u}_{app}(t) = \boldsymbol{u}(t_j) + \frac{t - t_j}{t_{j+1} - t_j}\left[\boldsymbol{u}(t_{j+1}) - \boldsymbol{u}(t_j)\right] \tag{2.57}$$

The value of the control variables at the centre is given by

$$\boldsymbol{u}(t_{c,j}) = \frac{\boldsymbol{u}(t_j) + \boldsymbol{u}(t_{j+1})}{2} \tag{2.58}$$

The piecewise linear interpolation is used to prepare for the possibility of discontinuous solutions in control.

The state variable $\boldsymbol{x}(t)$ is approximated by a continuously differentiable and piecewise Hermite-Simpson cubic polynomial between $\boldsymbol{x}(t_j)$ and $\boldsymbol{x}(t_{j+1})$ on the interval $t_j \leq t \leq t_{j+1}$ of length q_j :

$$\boldsymbol{x}_{app}(t) = \sum_{r=0}^{3} c_r^j \frac{t - t_j}{p_j} \tag{2.59}$$

$$
\begin{aligned}
c_0^j &= \boldsymbol{x}(t_j) \\
c_1^j &= q_j \boldsymbol{f}_j \\
c_2^j &= -3\boldsymbol{x}(t_j) - 2q_j \boldsymbol{f}_j + 3\boldsymbol{x}(t_{j+1}) - q_j \boldsymbol{f}_{j+1} \\
c_3^j &= 2\boldsymbol{x}(t_j) + q_j \boldsymbol{f}_j - 2\boldsymbol{x}(t_{j+1}) + q_j \boldsymbol{f}_{j+1}
\end{aligned}
$$

where

$$
\begin{aligned}
\boldsymbol{f}_j = \boldsymbol{f}(\boldsymbol{x}(t_j), \boldsymbol{u}(t_j), t_j), \quad q_j = t_{j+1} - t_j \\
t_j \leq t \leq t_{j+1}, j = 1, \ldots, k-1
\end{aligned}
$$

The value of the state variables at the centre point of the cubic approximation

$$
\boldsymbol{x}_{c,j} = \frac{\boldsymbol{x}(t_j) + \boldsymbol{x}(t_{j+1})}{2} + \frac{q(\boldsymbol{f}(t_j) + \boldsymbol{f}(t_{j+1}))}{8} \tag{2.60}
$$

and the derivative is

$$
\frac{d\boldsymbol{x}_{c,j}}{dt} = -\frac{3(\boldsymbol{x}(t_j) + \boldsymbol{x}(t_{j+1}))}{2q} - \frac{q(\boldsymbol{f}(t_j) + \boldsymbol{f}(t_{j+1}))}{4} \tag{2.61}
$$

In addition, the chosen interpolating polynomial for the state and control variables must satisfy the midpoint conditions for the differential equations as follows:

$$
\boldsymbol{f}[\boldsymbol{x}_{app}(t_{c,j}), \boldsymbol{u}_{app}(t_{c,j}), t_{c,j}] - \dot{\boldsymbol{x}}_{app}(t_{c,j}) = 0 \tag{2.62}
$$

The equations (2.1)–(2.8) in Section 2.1 now can be defined as a discretised problem as follows:

$$
\min f(\boldsymbol{Y}), \tag{2.63}
$$

subject to

$$
\begin{aligned}
\boldsymbol{f}(\boldsymbol{x}_{app}(t), \boldsymbol{u}_{app}(t), t) - \dot{\boldsymbol{x}}_{app} &= 0 && (2.64) \\
\boldsymbol{x}_{app}(t_1) - \boldsymbol{x}_1 &= 0 && (2.65) \\
\boldsymbol{\psi}(\boldsymbol{x}_{app}(t_k), t_k) &= 0 && (2.66) \\
\boldsymbol{C}(\boldsymbol{x}_{app}(t), u_{app}(t), t) &\leq 0 && (2.67) \\
\boldsymbol{S}(\boldsymbol{x}_{app}(t), t) &\leq 0 && (2.68)
\end{aligned}
$$

where x_{app}, u_{app} are the approximation of the state and control, constituting Y in (2.63). This above discretisation approach has been implemented in the DIRCOL package which employed the sequential quadratic programming method SNOPT by Gill et al. [49, 47].

In contrast with the DIRCOL approach, Büskens and Maurer [27] proposed to discretise the control only and use an NLP solver with respect to the discretised control only. The corresponding discretised state variables can be determined recursively using a numerical integration scheme (e.g. Euler, Heun, Runge-Kutta etc.). This approach has been implemented in the NUDOCCCS package [26]. NUDOCCCS has more flexibility in choosing the numerical method approach for both the control and state variables. For simple problems, low order (e.g. Euler) numerical integration of the state is sufficient, but for complex problems, especially when a pure state inequality constraint occurs, the numerical integration approach can be more advanced (e.g. Runge-Kutta).

One of the main advantages of DIRCOL and NUDOCCCS is that both packages provide an approximation for the co-state variables λ which can then be used as an initial guess in the indirect multiple shooting approach. Each of the packages uses a different approach to obtain the co-state variables.

In DIRCOL the co-state variables are derived as follows. Consider the equation (2.63)–(2.68) and define the Lagrangian equation as follows:

$$\begin{aligned} \mathcal{L}^f &= f(Y) + \Sigma_{i=1}^{k} \lambda (f(x_{app}(t), u_{app}(t), t) - \dot{x}_{app}) \\ &\quad + \kappa(x_{app}(t_1) - x_1) + \nu\psi(x_{app}(t_k), t_k) \\ &\quad + \varsigma \Sigma_{i=1}^{q} C(x_{app}(t), u_{app}(t), t) + \varrho \Sigma_{i=1}^{s} S(x_{app}(t), t) \end{aligned} \tag{2.69}$$

By using the necessary conditions and the Lagrange multiplier of the discretised equality and inequality constraints from the equation (2.69), the co-state variables can be approximated (see KKT necessary condition on equations (2.50)–(2.53)).

Büskens and Maurer employed a different way by using a recursive approach to compute the state variable. In this case, a discretised version of equation (2.69) can be solved recursively after optimal u and x were obtained (see [27, page 92]). Another

way of obtaining the co-state approximation is by exploiting the Lagrangian equation:

$$\begin{aligned}\mathcal{L}^p &= f(\boldsymbol{Y}) + \kappa(\boldsymbol{x}_{app}(t_1) - \boldsymbol{x}_1) + \nu\boldsymbol{\psi}(\boldsymbol{x}_{app}(t_k), t_k) \\ &\quad + \varsigma\Sigma_{i=1}^{q}\boldsymbol{C}(\boldsymbol{x}_{app}(t), \boldsymbol{u}_{app}(t), t) + \varrho\Sigma_{i=1}^{s}S(\boldsymbol{x}_{app}(t), t) \end{aligned} \tag{2.70}$$

and then determine the derivative of equation (2.70) with respect to the state

$$\boldsymbol{\lambda} = \mathcal{L}_{\boldsymbol{x}}^{p} \tag{2.71}$$

and use it to approximate the co-state variables. This approach has been implemented in NUDOCCCS and produces a more reliable and accurate approximation (see [27, pp. 92–93], [28]).

2.4.1.2 Pseudospectral Method for Optimal Control

Among the direct transcription method for optimal control problem is the Legendre pseudospectral method (see Benson [6], Elnagar, Kazemi and Razzaghi [42, 37], Fahroo and Ross [40, 41, 91, 90]). This method is based on the spectral collocation in which the trajectory for the state and control variables are approximated by the Nth degree Lagrange interpolating polynomial. The value of the variables at the interpolating nodes is the unknown coefficients which in this technique are the Legendre-Gauss-Lobatto points.

Consider the Legendre-Gauss-Lobatto (LGL) points, $t_i, i = 0, \ldots, N$ and distributed on the interval $\tau \in [-1, 1]$. These points can be given by $t_0 = -1$, $t_N = 1$ and for $1 \leq i \leq N-1$, t_i are the zeros of $\dot{\Lambda}_N$, which is the derivative of the Legendre polynomial, Λ_N. The transformation between the LGL domain $\tau \in [-1, 1]$ and the physical domain $t \in [t_0, t_f]$ can be defined by the following linear relations:

$$\tau = \frac{\tau_f - \tau_0}{2}t + \frac{\tau_f + \tau_0}{2} \tag{2.72}$$

The approximation for the state and control variables at the LGL points are given by the Nth degree Lagrange interpolating polynomial as follows:

$$\boldsymbol{x}(t) \approx \boldsymbol{X}(t) = \sum_{i=0}^{N} \boldsymbol{x}(t_i)L_i(t) \tag{2.73}$$

$$\boldsymbol{u}(t) \approx \boldsymbol{U}(t) = \sum_{i=0}^{N} \boldsymbol{u}(t_i)L_i(t) \tag{2.74}$$

where $L_i(t)$ are the Lagrange interpolating polynomial of order N and is defined by

$$L_i(t) = \frac{1}{N(N+1)\Lambda_N(t_i)} \frac{(t^2-1)\dot{\Lambda}_N(t)}{t-t_i} = \begin{cases} 1 & \text{if } l = k, \\ 0 & \text{if } l \neq k \end{cases} \tag{2.75}$$

The state approximation (2.73) for the dynamic equations must satisfy the condition of the exact derivative of (2.73) at the LGL points. The derivative of (2.73) is given by

$$\dot{\boldsymbol{x}}(t_k) \approx \dot{\boldsymbol{X}}(t_k) = \sum_{i=0}^{N} \boldsymbol{x}(t_i)\dot{L}_i(t_k) = \sum_{i=0}^{N} D_{ki}\boldsymbol{x}(t_i) \tag{2.76}$$

where $D_{ki} = \dot{L}_i(t_k)$ are the entries of the $(N+1) \times (N+1)$ pseudospectral Legendre derivative matrix and defined by [36]

$$D_{ki} = \begin{cases} \frac{\Lambda_N(t_k)}{\Lambda_N(t_l)} \cdot \frac{1}{t_k - t_l} & \text{if } l \neq k \\ -\frac{N(N+1)}{4} & \text{if } l = k = 0 \\ \frac{N(N+1)}{4} & \text{if } l = k = N \\ 0 & \text{otherwise} \end{cases} \tag{2.77}$$

The objective function 2.1 is discretised using the Gauss-Lobatto quadrature rule

$$\min J = \phi[\boldsymbol{X}(t_N), t_N] + \sum_{k=0}^{N} \mathcal{L}[\boldsymbol{X}(t_k), \boldsymbol{U}(t_k), t_k].w_k \tag{2.78}$$

where w_k are the LGL weights. The boundary conditions can be defined by the approximating of the state variables at $\boldsymbol{X}_1$ and $\boldsymbol{X}_N$:

$$\psi(\boldsymbol{X}_1, \boldsymbol{X}_N) = 0 \tag{2.79}$$

The optimal control problem now can be solved as the NLP problem by using an established NLP solver.

The pseudospectral method has been implemented in commercially available software DIDO [89] and is, in principle, capable of producing estimates of co-states. However, recent work by Benson [6] shows that this does not work for the pure state constraint case, even for the simple benchmark problem of Bryson [23], see Appendix A.

2.4.1.3 Direct Multiple Shooting

The basic idea of the direct multiple shooting method is to transform the original optimal control problem into nonlinear programming problem by coupling the control parameterisation with a multiple shooting discretisation of the state variables (see Keller [61], Stoer and Bulirsch [99], Ascher et al. [2]). The control can be approximated by piecewise functions and the state variables are approximated at the shooting nodes t_i (see Figure 2.2). The initial value $x(t_i)$ for the state variables at nodes t_i must be guessed. Then in each interval the state equations must be integrated individually from t_i to t_{i+1}. In addition, the continuity conditions (matching conditions) must be satisfied which require that on each differential nodes the values $x(t_{i+1})$ should equal the final value of the preceding trajectory.

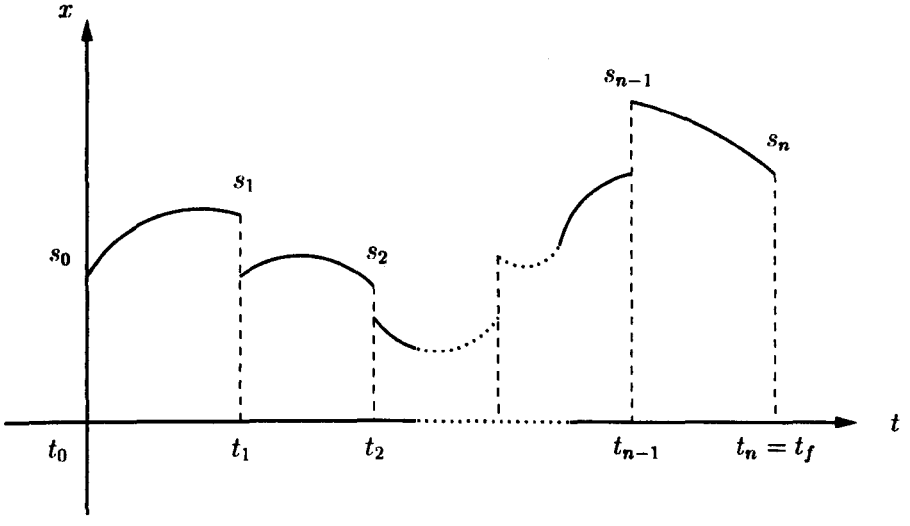


Figure 2.2: Multiple shooting

Consider now the following boundary value problem.

$$\dot{x} = f[x(t), u(t)], \quad r[x(t_0), x(t_f)] = 0 \tag{2.80}$$

The basic idea of the multiple shooting is to find simultaneously the values

$$s_i = x(t_i), \quad i = 1, \ldots, n, \tag{2.81}$$

for the solution of the boundary value problems (2.80) at the discretised nodes

$$t_0 < t_1 < t_2 < \ldots < t_n = t_f. \tag{2.82}$$

We assume that the discretisation nodes for the control parameterisation are the same as for the state parameterisation. Suppose $x[t; s_i, v_i]$ is the solution of the initial value problem:

$$\dot{x} = f[t, x, u(t, v_i)], \quad x(t_i) = s_i, \quad t \in [t_i, t_{i+1}]. \tag{2.83}$$

The problem now is to find the vector $s_i, i = 0, 1, \ldots, n$ and $v_i, i = 0, 1, \ldots, n-1$ such that the function $x(t)$ pieced together, continuously, by the following IVP solutions:

$$x(t) := x[t; s_i, v_i] \text{ for } t \in [t_i, t_{i+1}[, \quad i = 0, 1, \ldots n-1, \tag{2.84}$$

$$x(t_n) := s_n. \tag{2.85}$$

In addition, the boundary condition $r[x(t_0), x(t_f)] = 0$ must be satisfied by $x(t)$. Hence, the boundary value problem 2.80 is solved on the whole interval. Consider now the following equation $\boldsymbol{X}(s)$:

$$\boldsymbol{X}(s) = \begin{pmatrix} x[t_1; s_0, v_0] - s_1 \\ x[t_2; s_1, v_1] - s_2 \\ \vdots \\ x[t_n; s_{n-1}, v_{n-1}] - s_n \\ r[x(s_0), x(s_n)] \end{pmatrix} = 0 \tag{2.86}$$

where the unknown variables

$$s = \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \\ s_n \end{pmatrix} \tag{2.87}$$

must be found.

The optimal control problem now can be rewritten as an NLP problem.

$$\min J(s, v) = \sum_{i=0}^{n-1} J_i(s_i, v_i) \tag{2.88}$$

subject to

$$x[t_{i+1}; s_i, v_i] - s_{i+1} = 0, \quad i = 0, 1, \ldots, n-1 \tag{2.89}$$

$$r[x(s_0), x(s_n)] = 0 \tag{2.90}$$

The path constraints are transformed into vector inequality constraints at the multiple shooting nodes. The NLP problem result can then be solved by an established NLP solver.

2.4.2 Indirect Method Approach

Instead of using a direct approach as discussed in 2.4.1, the numerical method of solving a boundary value problem will be applied to solve the optimal control problem.

2.4.2.1 Multiple Shooting

The mathematical two-point boundary value problem (TPBVP) is best understood on the simple example of firing a shell. Given the initial barrel orientation and the initial shell speed, one can compute its trajectory and, in particular, the impact point. This is known as the initial value problem (IVP), as all we need to know is the starting point. The main observation is that for a given initial condition, the terminal condition (impact point) is uniquely determined, because it follows from integration of the known differential equation of motion.

However, if both the initial and terminal conditions are specified, then this is a TPBVP: the trajectory must be a solution of the defining differential equation, but must pass through prescribed points at both ends (boundaries). In the shell example, this means that we must find such a combination of barrel orientation and projectile speed that it indeed lands at the prescribed impact point.

This underpins the idea of the numerical method of shooting. It solves the TPBVP by repeated uses of readily available procedures (e.g. Runge-Kutta) for solving IVP. A guess of the initial point is made and the corresponding terminal point is computed. If it is not the prescribed one, the guess is optimally modified and serves as the starting point for the next use of an IVP solver. This process is repeated until convergence is obtained. The procedure illustrated in Figure 2.3 can be explained as follows. The

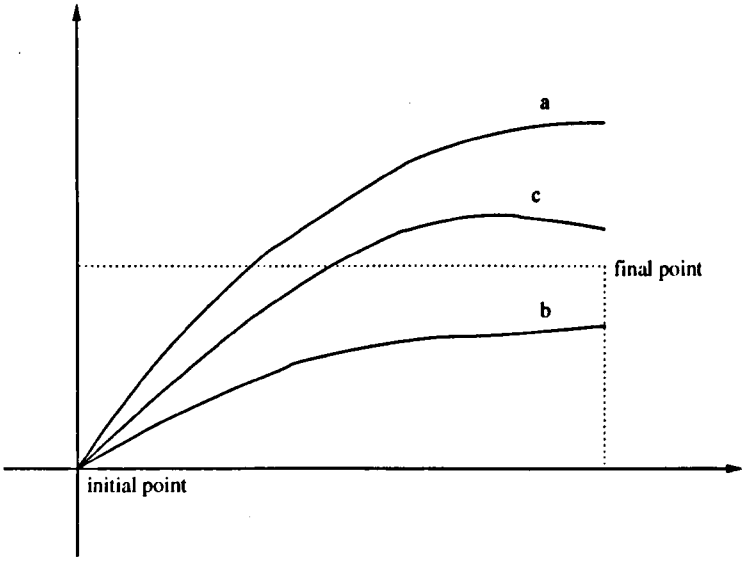


Figure 2.3: Shooting method procedure

initial point has two parameters: position (always at the origin) and speed (variable). Trajectory **a** clearly overshoots the prescribed terminal point, so the speed was modified to get **b** which now undershoots. Finally, **c** shows that systematic improvement can be attained.

The actual details for a second order equation $\ddot{x} = f(t, x, \dot{x})$ are given on Figure 2.4. The initial position $x_0 = x(t_0)$ is fixed and so is the terminal one $x_f = x(t_f)$. Thus the initial speed $\dot{x}(t_0)$ has to be iteratively modified until the end of the trajectory is within the desired accuracy ε.

The first guess $s^{(1)}$ of the initial speed $\dot{x}(t_0)$ is made to start the procedure and the corresponding initial value problem (IVP 1) is solved (block 1). The error X between the thus obtained terminal value $x(t_f; s^{(1)})$ and the desired one $x(t_f)$ is formed (block 2) and checked against the desired accuracy ε (block 3). If the accuracy requirement is met, the desired trajectory has been found; if not, then the guess of the initial speed must be improved.

The improvement is based on the idea that, ideally, the error X should be zero. In other words, we should try to find a value of the guess $s^{(i)}$ of the initial speed $\dot{x}(t_0)$

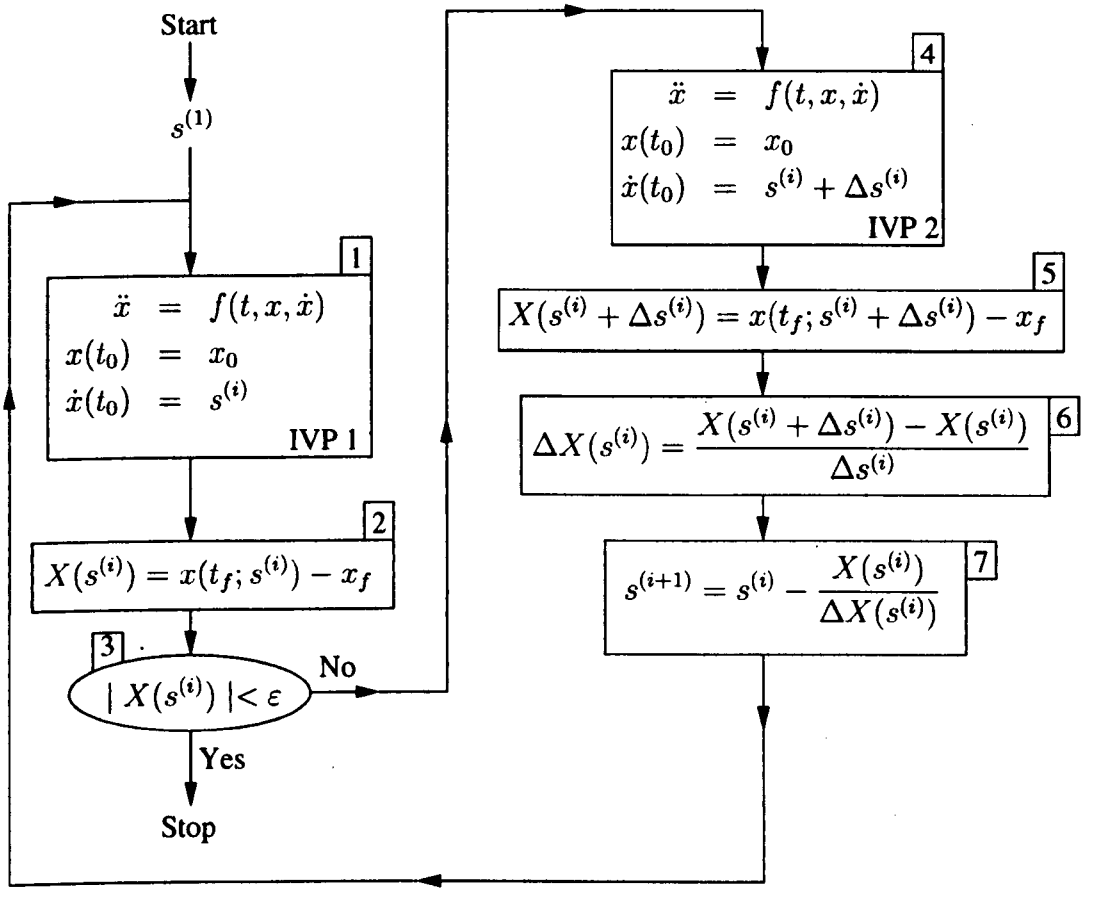


Figure 2.4: Shooting method flowchart

which yields $X(s^{(i)}) = x(t_f; s^{(i)}) - x_f = 0$. This is done by the well-known Newton procedure in block 7. The preceding blocks 4-6 perform the auxiliary computations: block 6 is the approximation ΔX of the derivative of X and needs the results of blocks 4 and 5. As a consequence, another IVP must be solved (block 4), so that the computation becomes more expensive.

The main drawback of the shooting method is the sensitivity of the initial guess, because of the use of Newton's iteration (block 7).

To overcome this problem, the trajectory must be split up into subintervals and apply the same shooting method for each subinterval which is the method of multiple shooting. The theoretical background for the multiple shooting is the same as direct multiple shooting in Section 2.4.1.3. However, the indirect multiple shooting solves the problem using a Newton iteration.

In a highly constrained optimal control problem, the jump and switching conditions on the co-state or control variables might occur. In order to handle those conditions, some new nodes must be inserted into subinterval. Consider the following boundary value problem (based on page 30 ref. [76]):

$$\dot{x}(t) = f_k(t, x(t)), \quad \xi_k \le t \le \xi_{k+1}, \quad 0 \le k \le s \tag{2.91a}$$

$$x(\xi_k^+) = h_k(\xi_k, x(\xi_k^-)), \quad \text{for} \quad k = 1, \ldots, s \tag{2.91b}$$

$$r_i(x(t_0), x(t_f)) = 0, \quad \text{for} \quad 1 \le i \le n_1 \tag{2.91c}$$

$$r_i(\xi_{k_i}, x(\xi_{k_i}^-)) = 0, \quad \text{for} \quad i = n_1 + 1, \ldots, n + s \tag{2.91d}$$

In the optimal control framework the equation (2.91a) represents state and co-state equations which are a piecewise smooth function and $\xi_i, i = 1, \ldots, s$ is a switching point. The equation (2.91b) is a jump condition at the switching point ξ_i. The boundary conditions at the initial and final time are described by equation (2.91c) and the condition at the switching point is given by equation (2.91d).

Suppose S_j are the initial guesses for the $x(t_i)$ and Ξ_j are the initial guesses for the switching points ξ_j. Let us define:

$$y(t) = \begin{bmatrix} x(t) \\ \xi \end{bmatrix} \tag{2.92}$$

and

$$\boldsymbol{Y}(t) = \begin{bmatrix} \boldsymbol{S} \\ \Xi \end{bmatrix} \tag{2.93}$$

The problem now is to find the solution of the IVP:

$$\dot{\boldsymbol{y}}(t) = \begin{bmatrix} \boldsymbol{f}(t, \boldsymbol{x}(t)) \\ 0 \end{bmatrix}, \quad t_j \leq t \leq t_{j+1}, \quad \boldsymbol{y}(t_j) = \boldsymbol{Y}_j, \quad j = 1, \ldots, n-1 \tag{2.94}$$

where $\boldsymbol{y}(t)$ consists of the switching point ξ and must be computed simultaneously in the numerical processes. A modified Newton method is used to determine $\boldsymbol{y}(t)$.

A professional version of the modified Newton algorithm, tailored to optimal control applications, has been implemented in FORTRAN and is available as the package BNDSCO (see Oberle and Grimm [76]). However, it should be emphasised that even the best TPBVP solver cannot overcome the fundamental problem of a narrow convergence interval inherent in TPBVP.

2.5 Summary and Discussion

This chapter presented an overview of the optimal control problem and its numerical solution. Real-life nonlinear optimal control problems cannot be solved analytically, in general, and must be solved numerically. Numerical solution of continuous optimal control problems can be categorised into two different approaches: 1) the direct and 2) the indirect method. Direct methods are based on the transformation of the original optimal control problem into a nonlinear programming (NLP) problem by discretising the state or/and control history and then solving the resulting problem using an NLP solver. The indirect method solves the optimal control problem by deriving the necessary conditions based Pontryagin's Minimum Principle.

In the indirect method the user must derive the appropriate equations for co-state variables, transversality and optimality conditions before the problem can be solved using a boundary value problem solver. Furthermore, the problem is more involved when the problem contains path constraints. The sequence of the constrained/unconstrained arcs must be guessed and the corresponding switching and jump conditions must be derived, which is a non-trivial task. Secondly, the narrow convergence of the multiple

shooting method must be considered. Finally, the co-state variables must be guessed which is a nonintuitive task because the variables do not have physical meaning

In contrast, the direct method is easy to implement because all it requires is a fairly straightforward discretisation of the original problem. But the accuracy of the direct method is less than that of the indirect method.

Chapter 3

Minimum Altitude Formulation

In Section 1.1 the problem formulation for the terminal bunt manoeuvre is given for the minimum time and minimum altitude problem. This chapter presents analysis and computation for the minimum altitude version of the terminal bunt manoeuvre. The cruise missile must hit the fixed target from above while minimising the missile exposure to anti-air defences. This means that the flight altitude should be as low as possible, but the impact must be achieved by a vertical dive. This leads to the generic trajectory shape where the missile initially flies straight and level at the minimum altitude. When it approaches the target, it must climb (nose up) to gain enough height for the final dive (nose down). This up-and-down terminal manoeuvre is known as the bunt, and establishing its optimal parameters is an example of trajectory shaping.

The quantity to be minimised is the integrated altitude, but this minimisation must take into account, inter alia, missile dynamics (manoeuvrability constraints of the platform), the final dive specifications and limits on controls (thrust and angle of attack).

This chapter is organised as follows. In Section 3.1 the problem formulation of the minimum altitude of the terminal bunt manoeuvre is given. The computational results are based on the DIRCOL package which are then used for a qualitative analysis of the main features of the optimal trajectories and their dependence on several constraints, as discussed in Section 3.2. The mathematical analysis based on the qualitative analysis is presented in Section 3.3. This section begins with discussing the constraint on the thrust, followed by path and mixed constraints. Section 3.4 focuses on the indirect method approach. The co-state approximation issue is considered in that section. This

section is concluded with some numerical solutions using the multiple shooting method package BNDSCO. Finally, Section 3.5 presents summary and discussion.

3.1 Minimum Altitude Problem

The objective function as given in Section 1.1 is to determine the trajectory of the generic cruise missile from an initial state to a final state with minimum altitude along the trajectory. The objective can be formulated by introducing the performance index:

$$J = \int_{t_0}^{t_f} h \quad dt. \tag{3.1}$$

This objective function is subject to the dynamic equations and some constraints as defined in Section 1.1.

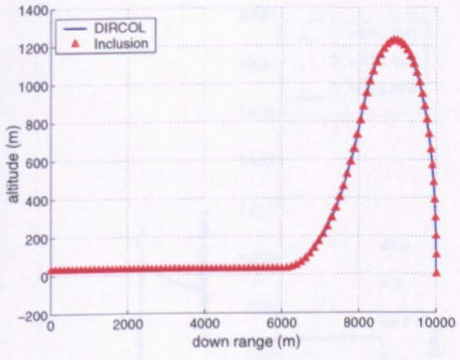
3.2 Qualitative Analysis

This section gives a qualitative discussion of the optimal trajectory of a cruise missile performing a bunt manoeuvre. Subchan et al. [101] presented a qualitative analysis for the terminal bunt manoeuvre based on Cleminson's result [29].

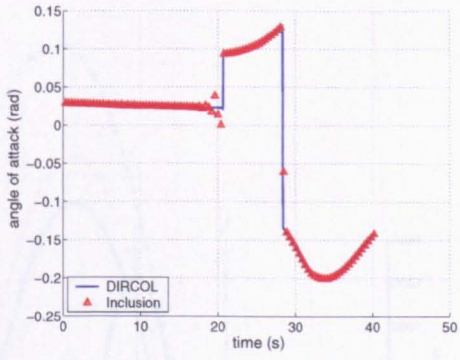
The computational results of the terminal bunt manoeuvre are obtained using a direct collocation method package DIRCOL by von Stryk [109] and then the resulting nonlinear programming problem solved using the SNOPT solver, which is based on sequential quadratic programming due to Gill et al. [47, 92]. The important feature of DIRCOL is that it provides an approximation for the co-state variables.

In this simulation the missile is assumed to be launched horizontally from the minimum altitude constraint $h_0 = 30$ m. The initial and final conditions can be given as follows:

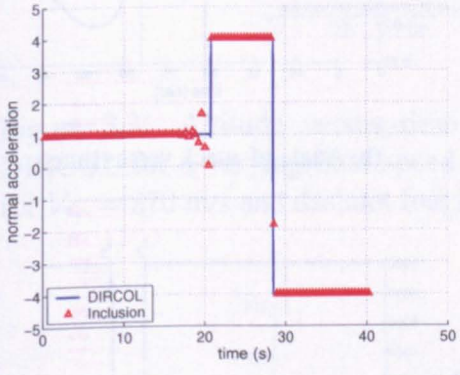
$$\begin{aligned}
\gamma_0 &= 0 \text{ deg}, & \gamma_{t_f} &= -90 \text{ deg} \\
V_0 &= 272 \text{ m/s}, & V_{t_f} &= 250, 270, 310 \text{ m/s} \\
x_0 &= 0 \text{ m}, & x_{t_f} &= 10000 \text{ m} \\
h_0 &= 30 \text{ m}, & h_{t_f} &= 0 \text{ m}.
\end{aligned}$$



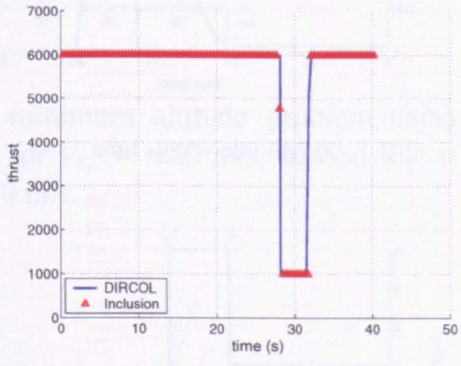
(a) Altitude versus downrange



(b) Angle of attack versus time



(c) Normal acceleration versus time

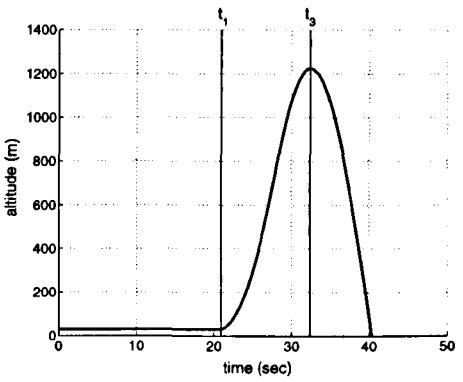


(d) Thrust versus time

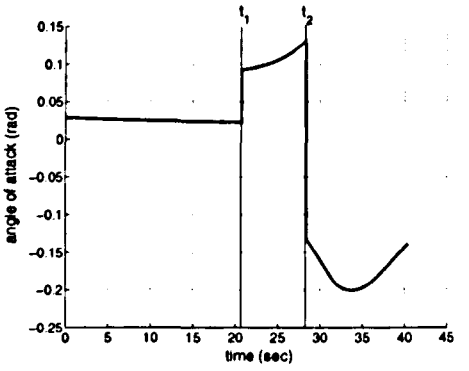
Figure 3.1: Comparison of DIRCOL and differential inclusion results for minimum altitude problem for final speed $V_{t_f} = 250$ m/s.

Figure 3.1 shows the comparison of the DIRCOL and differential inclusion results which are taken from [29]. It can be seen that the DIRCOL results give a more smooth solution on the control. Based on Figures 3.2–3.8, an attempt is made to identify characteristic arcs of the trajectory, classify them according to the constraints active on them, and suggest physical/mathematical explanations for the observed behaviour.

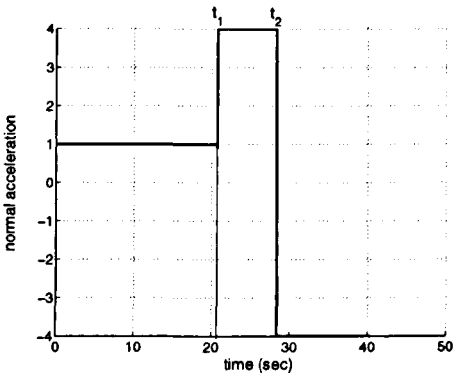
The trajectory is split into three subintervals: level flight, climbing and diving.



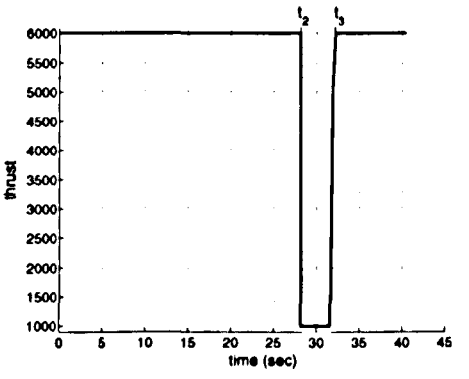
(a) Altitude versus time



(b) Angle of attack versus time



(c) Normal acceleration versus time



(d) Thrust versus time

Figure 3.2: Computational results for minimum altitude problem using DIRCOL for $V_f = 250$ m/s. t_1 is the time when the missile starts to climb, t_2 is the time when the thrust switches to minimum value and t_3 is the time when the missile starts to dive.

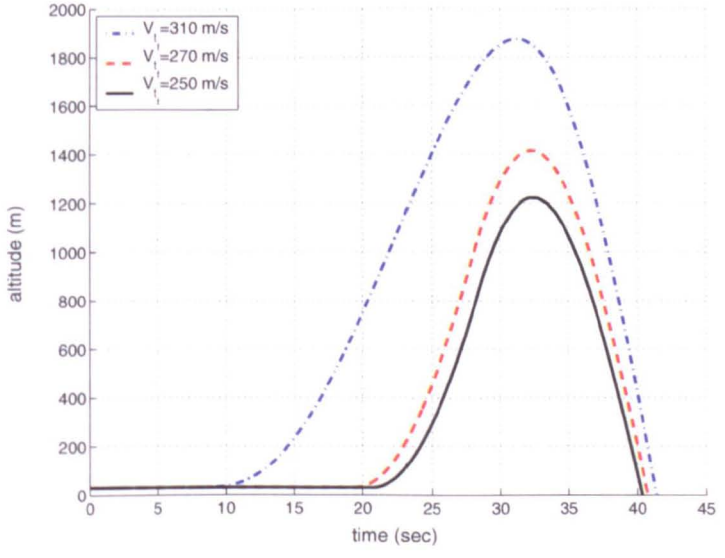


Figure 3.3: Altitude versus time histories for minimum altitude problem using DIRCOL for a varying final speed. Solid line is for $V_{t_f} = 250$ m/s, dashed line is for $V_{t_f} = 270$ m/s and dashdot line is for $V_{t_f} = 310$ m/s.

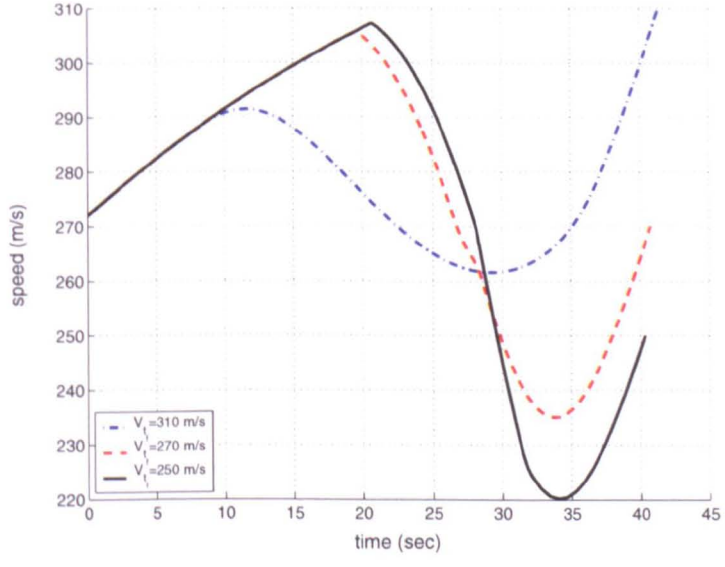


Figure 3.4: Speed versus time histories for minimum altitude problem using DIRCOL for a varying final speed. Solid line is for $V_{t_f} = 250$ m/s, dashed line is for $V_{t_f} = 270$ m/s and dashdot line is for $V_{t_f} = 310$ m/s.

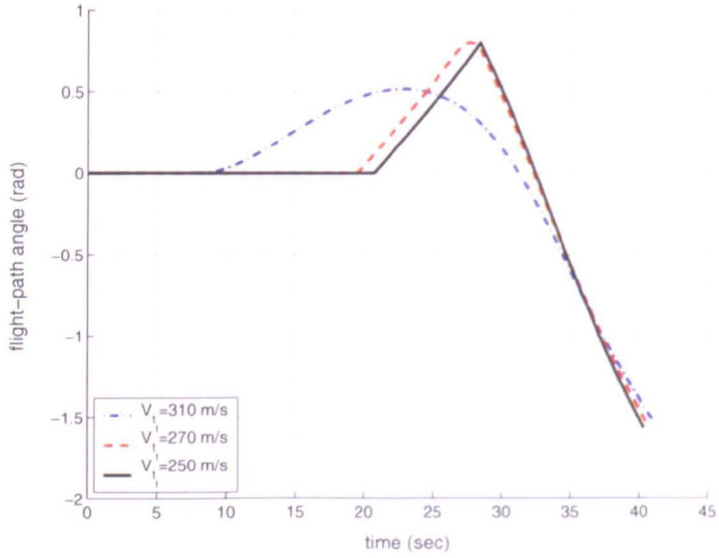


Figure 3.5: Flight-path angle versus time histories for minimum altitude problem using DIRCOL for a varying final speed. Solid line is for $V_{t_f} = 250$ m/s, dashed line is for $V_{t_f} = 270$ m/s and dashdot line is for $V_{t_f} = 310$ m/s.

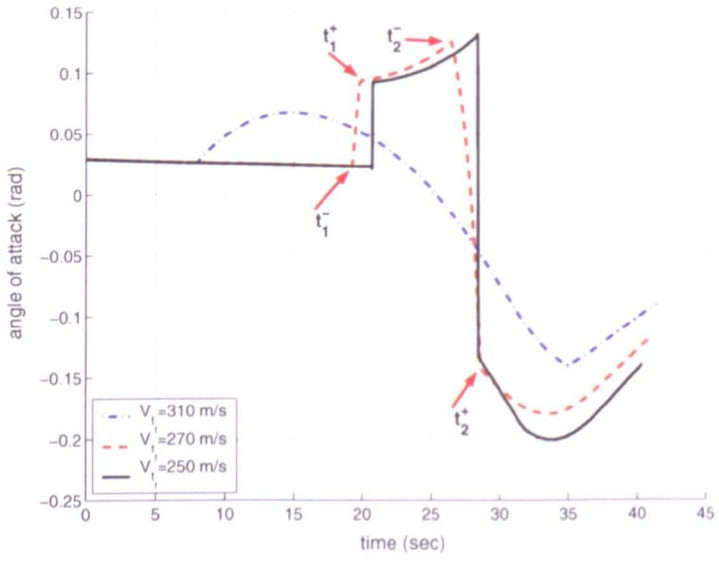


Figure 3.6: Angle of attack versus time histories for minimum altitude problem using DIRCOL for a varying final speed. Solid line is for $V_{t_f} = 250$ m/s, dashed line is for $V_{t_f} = 270$ m/s and dashdot line is for $V_{t_f} = 310$ m/s.

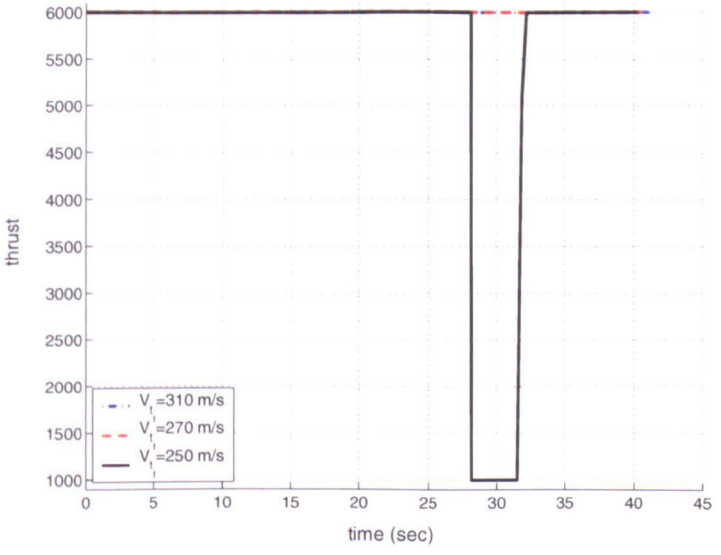


Figure 3.7: Thrust versus time histories for minimum altitude problem using DIRCOL for a varying final speed. Solid line is for $V_{t_f} = 250$ m/s, dashed line is for $V_{t_f} = 270$ m/s and dashdot line is for $V_{t_f} = 310$ m/s.

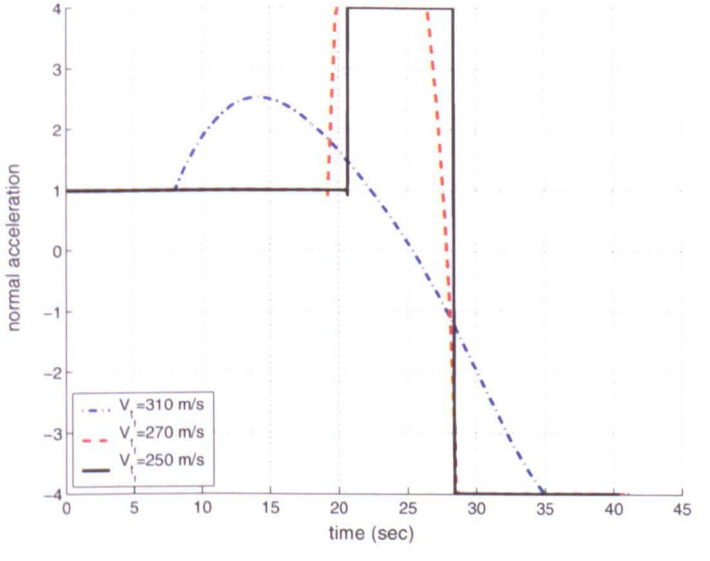


Figure 3.8: Normal acceleration versus time histories for minimum altitude problem using DIRCOL for a varying final speed. Solid line is for $V_{t_f} = 250$ m/s, dashed line is for $V_{t_f} = 270$ m/s and dashdot line is for $V_{t_f} = 310$ m/s.

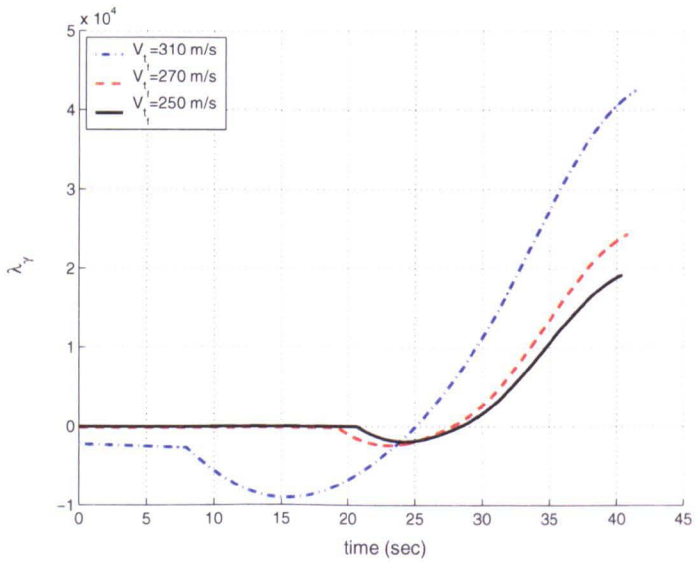


Figure 3.9: λ_γ versus time histories for minimum altitude problem using DIRCOL for a varying final speed. Solid line is for $V_{t_f} = 250$ m/s, dashed line is for $V_{t_f} = 270$ m/s and dashdot line is for $V_{t_f} = 310$ m/s.

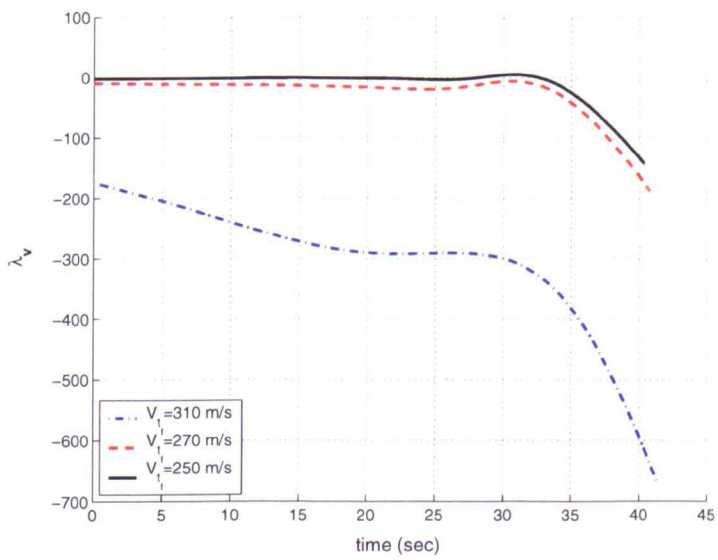


Figure 3.10: λ_V versus time histories for minimum altitude problem using DIRCOL for a varying final speed. Solid line is for $V_{t_f} = 250$ m/s, dashed line is for $V_{t_f} = 270$ m/s and dashdot line is for $V_{t_f} = 310$ m/s.

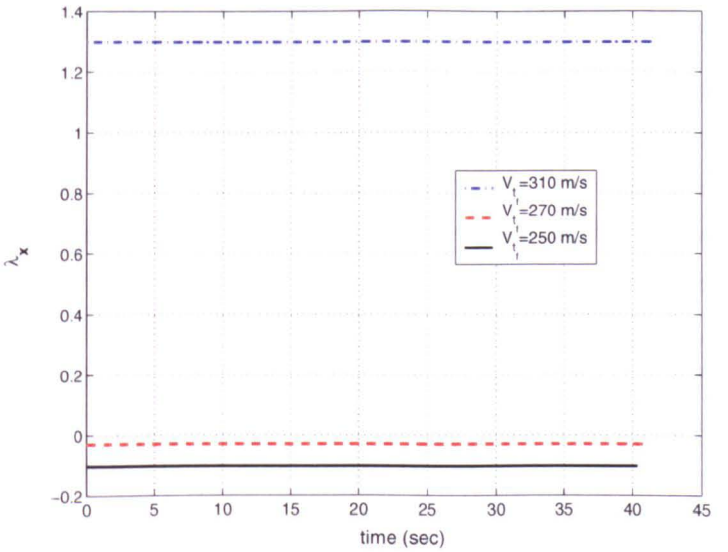


Figure 3.11: λ_x versus time histories for minimum altitude problem using DIRCOL for a varying final speed. Solid line is for $V_{t_f} = 250$ m/s, dashed line is for $V_{t_f} = 270$ m/s and dashdot line is for $V_{t_f} = 310$ m/s.

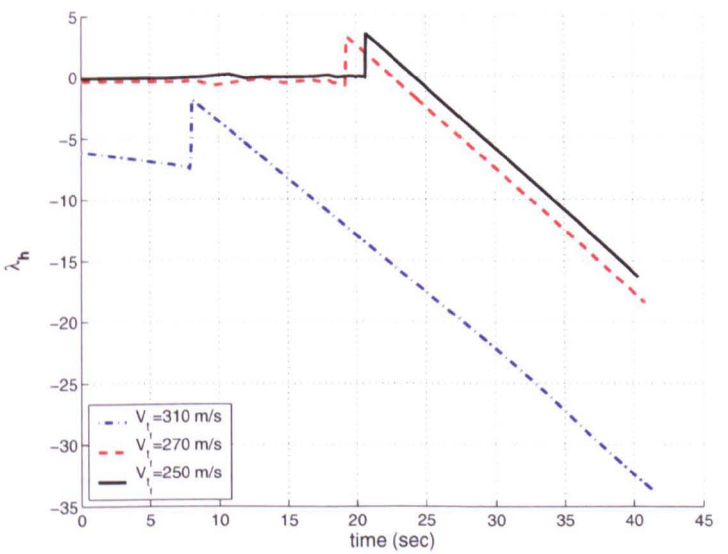


Figure 3.12: λ_h versus time histories for minimum altitude problem using DIRCOL for a varying final speed. Solid line is for $V_{t_f} = 250$ m/s, dashed line is for $V_{t_f} = 270$ m/s and dashdot line is for $V_{t_f} = 310$ m/s.

3.2.1 First arc (flight): minimum altitude flight $t_0 \leqslant t \leqslant t_1$

The thrust and altitude constraints are active directly at the start of the manoeuvre. In this case the altitude h of the missile remains constant on the minimum value (h_{min}) until the missile must start climbing while the thrust is on the maximum value. The flight time depends to the final-speed V_{t_f} (see Figure 3.3). In addition, the flight time is longer for the smaller final-speed in this case.

Equation (1.3d) equals zero during this flight because the altitude remains constant. It means the flight path angle γ equals zero because the velocity V is never equal to zero during flight. In addition, $\gamma(t) = 0$ for $t_0 \leqslant t \leqslant t_1$ (see Figure 3.2 for the definition of t_1, t_2 and t_3) causes the derivative of the flight path angle $\dot{\gamma}$ to be equal to zero. The dynamics equation (1.3) is therefore reduced as follows:

$$\dot{\gamma} = \frac{T-D}{m}\sin\alpha + \frac{L}{m}\cos\alpha - g = 0 \tag{3.2a}$$

$$\dot{V} = \frac{T-D}{m}\cos\alpha - \frac{L}{m}\sin\alpha \tag{3.2b}$$

$$\dot{x} = V \tag{3.2c}$$

$$\dot{h} = 0 \tag{3.2d}$$

We now consider the consequences of the right-hand side of equation (3.2a) being zero. This condition means that the normal acceleration L/m remains almost constant, because the angle of attack α is very small. The first term on the right-hand side of equation (3.2a) is small, because $\sin\alpha \approx \alpha \approx 0$ and we are left with $L/m \approx g$ due to $\cos\alpha \approx 1$.

During this time speed increases, because for small α

$$\dot{V} \approx \frac{T-D}{m} > 0, \quad \text{as } T > D.$$

This in turn means that the angle of attack α slowly decreases in accordance with equation (1.6) and in order to maintain L/m approximately be equal to g.

3.2.2 Second arc: climbing

This analysis mainly discusses the case for the final-speed 250 m/s because it exhibits switching in the thrust (see Figure 3.2d). The missile must climb eventually in order to

achieve the final condition of the flight path angle γ_{t_f}. This condition occurs between $t_1 \leqslant t \leqslant t_3$.

3.2.2.1 Climbing: full thrust & maximum normal acceleration ($t_1 \leqslant t \leqslant t_2$)

At the beginning of the climbing manoeuvre the thrust is on the maximum value. Since altitude above h_{min} is penalised, the climb occurs as late as possible, so must be done sharply and last as short as possible. Hence, at the beginning of ascent the angle of attack must increase to facilitate a rapid nose up motion and the thrust has the maximum value.

During this time, the normal acceleration is saturated on the maximum value L_{max} due to the jump of the angle of attack α. The speed keeps decreasing while the angle of attack α and altitude h increase. This arc ends at t_2 when the thrust switches to the minimum value T_{min}. During this time, the normal acceleration jumps to the minimum value L_{min} due to the jump angle of attack α.

3.2.2.2 Climbing: minimum thrust ($t_2 \leqslant t \leqslant t_3$)

While rapid climbing is necessary, the missile should also turn over to begin its dive as soon as possible, so that the excess of altitude (above h_{min}) is minimised. Thus, the thrust should soon be switched to its minimum value and at the same time the angle of attack should be decreased to negative values, further to promote pitching down.

From the computational results (see Figure 3.2d) it follows that the thrust is switched to the minimum value before turnover. This occurs approximately t_2 after firing. Immediately after the thrust is switched, the flight path angle γ decreases rapidly while the angle of attack α jumps. This causes the normal acceleration to jump, saturating on the minimum value L_{min}. When the normal acceleration is saturated on the minimum value, the angle of attack decreases further. At the same time the speed decreases and the altitude increases until the missile turns over.

The missile turns over when the flight path angle $\gamma = 0$ and $\dot{h} = 0$. At the same time the thrust switches back to the maximum value to facilitate rapid arrival on the target.

For the final-speed 270 m/s and 310 m/s the thrust does not exhibit any switching.

Table 3.1: Performance index and final time for the minimum altitude problem for different terminal speed using DIRCOL

Final speed V_{t_f} (m/s)	J	t_f (sec)	grid points
250	13563.3003	40.34764	176
270	16764.8403	40.75616	169
310	31562.5639	41.38049	169

3.2.3 Third arc: diving ($t_3 \leqslant t \leqslant t_f$)

The missile starts diving at approximately t_3 seconds. At the end of the manoeuvre the missile should hit the target with a certain speed V_{t_f}. The speed during turnover is smaller than final speed V_{t_f}, so the speed must increase and hence the thrust switches back to the maximum value for the case $V_{t_f} = 250$ m/s. It means the thrust will facilitate the missile's arrival on the target as soon as possible.

In this case the normal acceleration is on the minimum value. Obviously, the altitude goes down to reach the target ($\gamma < 0 \rightarrow \dot{h} < 0$, see equation (1.3d)), while the speed goes up to satisfy the terminal speed condition V_{t_f}. Finally, the missile satisfies the terminal condition of the manoeuvre approximately t_f after firing.

Table 3.1 shows that the objective function is bigger for the greater final-speed. The performance index for the case $V_{t_f} = 310$ m/s is twice of the case $V_{t_f} = 250$ m/s while the final-time is circa one minute different. The co-state approximation is given in Figures 3.9–3.12. It can be seen that the co-state approximation for λ_h has a jump at the exit of the pure state constraint (minimum altitude constraint).

3.3 Mathematical Analysis

The qualitative analysis of Section 3.2 is now made precise using optimal control theory.

3.3.1 Constrained on the Thrust Only

First, we investigate the minimum altitude problem when the initial and final conditions (1.9) are active and the control is constrained on thrust T only (1.12). Necessary

conditions for optimality can be determined by applying Pontryagin's Minimum Principle [84]. For this purpose, we first consider the following Hamiltonian:

$$\begin{aligned} H^{af} &= h + \frac{\lambda_\gamma}{V}\left[\frac{T-D}{m}\sin\alpha + \frac{L}{m}\cos\alpha - g\cos\gamma\right] \\ &\quad +\lambda_V\left[\frac{T-D}{m}\cos\alpha - \frac{L}{m}\sin\alpha - g\sin\gamma\right] \\ &\quad +\lambda_x V\cos\gamma + \lambda_h V\sin\gamma, \end{aligned} \tag{3.3}$$

where the co-state variables $\boldsymbol{\lambda} = (\lambda_\gamma, \lambda_V, \lambda_x, \lambda_h)$ have been adjoined to the dynamics system of equation (1.3). The co-state equations are determined by

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial H^{af}}{\partial \boldsymbol{x}}. \tag{3.4}$$

The component of co-state vector $\boldsymbol{\lambda}$ satisfying the preceding equations are:

$$\dot{\lambda}_\gamma = -\left\{\frac{\lambda_\gamma}{V}g\sin\gamma - \lambda_V g\cos\gamma - \lambda_x V\sin\gamma + \lambda_h V\cos\gamma\right\} \tag{3.5}$$

$$\begin{aligned} \dot{\lambda}_V &= -\left\{\lambda_\gamma\left[-\frac{T\sin\alpha}{V^2 m} - \frac{C_d\rho S_{ref}\sin\alpha}{2m} + \frac{C_l\rho S_{ref}\cos\alpha}{2m} + \frac{g}{V^2}\cos\gamma\right]\right. \\ &\quad \left. -\frac{\lambda_V\rho V S_{ref}}{m}\left[C_d\cos\alpha + C_l\sin\alpha\right] + \lambda_x\cos\gamma + \lambda_h\sin\gamma\right\} \end{aligned} \tag{3.6}$$

$$\dot{\lambda}_x = 0 \tag{3.7}$$

$$\begin{aligned} \dot{\lambda}_h &= -\left\{1 + \lambda_\gamma\left[-\frac{C_d V S_{ref}\sin\alpha\rho_h}{2m} + \frac{C_l V S_{ref}\cos\alpha\rho_h}{2m}\right]\right. \\ &\quad \left. +\lambda_V\left[-\frac{C_d V^2 S_{ref}\cos\alpha\rho_h}{2m} - \frac{C_l V^2 S_{ref}\sin\alpha\rho_h}{2m}\right]\right\}, \end{aligned} \tag{3.8}$$

where:

$$\rho_h = 2C_1 h + C_2$$

The optimal values of the control variables are generally to be determined from the Pontryagin's Minimum Principle. A necessary condition for optimal control is the

Minimum Principle

$$\min_{\boldsymbol{u}} H^{af}, \tag{3.9}$$

i.e. the Hamiltonian must be minimised with respect to the vector of controls $\boldsymbol{u}$. Applying (3.9) to (3.3) we obtain

$$\begin{aligned} H_\alpha^{af} &= (T - D + L_\alpha)\Big[\frac{\lambda_\gamma}{Vm}\cos\alpha - \frac{\lambda_V}{m}\sin\alpha\Big] \\ &\quad -(D_\alpha + L)\Big[\frac{\lambda_\gamma}{Vm}\sin\alpha + \frac{\lambda_V}{m}\cos\alpha\Big] = 0 \end{aligned} \tag{3.10}$$

with

$$\begin{aligned} L_\alpha &= \frac{1}{2}\rho V^2 S_{ref} B_1 \\ D_\alpha &= \frac{1}{2}\rho V^2 S_{ref}(2A_1\alpha + A_2) \end{aligned}$$

Since the control T appears linearly in the Hamiltonian, the condition $H_T^{af} = 0$ from (3.11) does not determine optimal thrust. Since T is bounded, the following provides the minimum of the Hamiltonian:

$$T = \begin{cases} T_{max} & \text{if } H_T^{af} < 0, \\ T_{sing} & \text{if } H_T^{af} = 0, \\ T_{min} & \text{if } H_T^{af} > 0. \end{cases}$$

with

$$H_T^{af} = \lambda_\gamma \frac{\sin\alpha}{Vm} + \lambda_V \frac{\cos\alpha}{m} \quad \text{(switching function)} \tag{3.11}$$

- **Case when T on the boundary ($T = T_{max}$ or $T = T_{min}$)**
  In this case α can be determined from:

$$\begin{aligned} H_\alpha^{af} &= \frac{T - D + L_\alpha}{m}\Big[\frac{\lambda_\gamma}{V}\cos\alpha - \lambda_V\sin\alpha\Big] \\ &\quad -\frac{D_\alpha + L}{m}\Big[\frac{\lambda_\gamma}{V}\sin\alpha + \lambda_V\cos\alpha\Big] = 0 \end{aligned}$$

  The value of α cannot be derived in closed form from (3.12), and must be obtained numerically. Note that D, D_α and L depend on α.

- **Case when $T = T_{sing}$ (singular control)**
  When the switching function H_T^{af} becomes zero in an interval $(t_1, t_2) \subset (t_0, t_f)$, the control corresponding to the magnitude of the thrust T is singular. In these circumstances, there are finite control variations of T which do not affect the value of the Hamiltonian.

  From Bryson and Ho [23, page 246], the singular arcs occur when:

$$H_{\boldsymbol{u}}^{af} = 0 \quad \text{and} \quad \det H_{\boldsymbol{uu}}^{af} = 0 \tag{3.12}$$

  Substituting (3.3) into (3.12) with component $\boldsymbol{u} = (T, \alpha)$ yields

$$H_T^{af} = \lambda_\gamma \frac{\sin\alpha}{Vm} + \lambda_V \frac{\cos\alpha}{m} = 0 \tag{3.13}$$

$$\begin{aligned} H_\alpha^{af} \;=\; & (T - D + D_\alpha)\Big[\frac{\lambda_\gamma}{Vm}\cos\alpha - \frac{\lambda_V}{m}\sin\alpha\Big] \\ & -(D_\alpha + L)\Big[\frac{\lambda_\gamma}{Vm}\sin\alpha + \frac{\lambda_V}{m}\cos\alpha\Big] = 0 \end{aligned} \tag{3.14}$$

$$\det H_{\boldsymbol{uu}}^{af} = 0 \iff \lambda_\gamma \frac{\cos\alpha}{Vm} - \lambda_V \frac{\sin\alpha}{m} = 0 \tag{3.15}$$

  Conditions (3.13)–(3.15) cannot be satisfied simultaneously, so we conclude that there are no singular arcs. However, jump discontinuities in the control T may appear if, at a time t, the switching function (3.11) changes sign.

The Hamiltonian is not an explicit function of time, so H^{af} is constant along the optimal trajectory.

3.3.2 Optimal Control With Path Constraints

In section 3.3.1 we derived necessary conditions for optimality by considering only the boundary conditions and thrust constraint. In this section the level of complexity is increased by considering some additional constraints as defined in Section 1.1.

The first state path constraint (1.10) can be split as $V_{min} - V \leqslant 0$ and $V - V_{max} \leqslant 0$. Both of them are of order 1, because $\dot{V}$ explicitly depends on the controls, see [23, pp. 99–100]. Since the speed constraint is not active during the manoeuvre in this

case, it will not be taken into account in the Hamiltonian (see Figure 3.4 page 45). The second path constraint (1.11) is of order 2 and the mixed state-control constraint (1.13) is split as $L_{min} - \frac{L}{mg} \leqslant 0$ and $\frac{L}{mg} - L_{max} \leqslant 0$, and L depends on the control explicitly. The Hamiltonian can be defined as follows:

$$H^{ac} = H^{af} + \mu_1 \left\{ -\frac{L}{mg} + L_{min} \right\} + \mu_2 \left\{ \frac{L}{mg} - L_{max} \right\} + \mu_3(\ddot{h})$$

The differential equations for co-state vector $\boldsymbol{\lambda} = (\lambda_\gamma, \lambda_V, \lambda_x, \lambda_h)$ can be written as

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial H^{ac}}{\partial \boldsymbol{x}}. \tag{3.16}$$

Since these equations are rather lengthy, they are omitted here. For the Lagrange multipliers $\mu_i = 1, \ldots, 5$, there must hold

$$\mu_i \begin{cases} = 0, & \text{if the associated constraint is not active;} \\ \geqslant 0, & \text{if the associated constraint is active.} \end{cases}$$

The necessary conditions are completed by deriving the junction conditions at the switching points $\bar{t}_i$ as follows [23, page 101]

$$H^{ac}(\bar{t}_i^-) = H^{ac}(\bar{t}_i^+) - \boldsymbol{\vartheta}^T \frac{\partial \boldsymbol{N}}{\partial t_i} \tag{3.17}$$

$$\boldsymbol{\lambda}^T(\bar{t}_i^-) = \boldsymbol{\lambda}^T(\bar{t}_i^+) + \boldsymbol{\vartheta}^T \frac{\partial \boldsymbol{N}}{\partial x} \tag{3.18}$$

which requires finding the additional multipliers $\boldsymbol{\vartheta}$.

3.3.3 First Arc: Minimum Altitude Flight

In this section we consider only the state path constraint $h_{min} \leqslant h$ and thrust control constraint (T is on the maximum value). In this case we assume that the missile launches at the initial altitude $h = h_{min}$. Therefore the constraints are active at the beginning of the manoeuvre directly. The constraint $h_{min} \leqslant h$ has no explicit dependence on the control variables, therefore we must take the time derivative on the

constraint until, finally, explicit dependence on the control does occur. Consider the following equations:

$$h - h_{min} = 0 \tag{3.19a}$$

$$\dot{h} = V\sin\gamma = 0 \quad \text{and} \quad V \neq 0$$

$$\Rightarrow \gamma(t) = 0 \quad \text{for} \quad t \in [t_0, t_1] \tag{3.19b}$$

$$\ddot{h} = \dot{V}\sin\gamma + V\dot{\gamma}\cos\gamma = 0$$

$$\Rightarrow \dot{\gamma}(t) = 0 \quad \text{for} \quad t \in [t_0, t_1] \tag{3.19c}$$

The controls appear explicitly after differentiating the constraint $h_{min} \leqslant h$ twice, therefore the order of the constraint is 2. Substituting equation (3.19) to the equation of motion (1.3) we obtain the following reduced state equations:

$$\dot{\gamma} = \frac{T-D}{m}\sin\alpha + \frac{L}{m}\cos\alpha - g = 0 \tag{3.20a}$$

$$\dot{V} = \frac{T-D}{m}\cos\alpha - \frac{L}{m}\sin\alpha \tag{3.20b}$$

$$\dot{x} = V \tag{3.20c}$$

$$\dot{h} = 0 \tag{3.20d}$$

The angle of attack α can be obtained numerically from equation (3.20a). Then substituting α to equation (3.20b) and (3.20c), these equations can be solved as an initial value problem (IVP). Thus we can find the first arc easily, but we do not know how long it will last. For this purpose we should formulate the appropriate boundary value problem (BVP) which involves finding co-state variables by defining the Hamiltonian as follows:

$$H^{aa} = H^{af} + \mu_3\left\{\dot{V}\sin\gamma + V\dot{\gamma}\cos\gamma\right\} \tag{3.21}$$

The components of co-state vector $\boldsymbol{\lambda}$ satisfying the preceding equations are:

$$\begin{aligned}
\dot{\lambda}_\gamma &= -\Big\{\frac{\lambda_\gamma}{V} g\sin\gamma - \lambda_V g\cos\gamma - \lambda_x V\sin\gamma + \lambda_h V\cos\gamma \\
&\quad +\mu_3\Big[\frac{d\dot{V}}{d\gamma}\sin\gamma + \dot{V}\cos\gamma + \frac{d\dot{\gamma}}{d\gamma}V\cos\gamma - \dot{\gamma}V\sin\gamma\Big]\Big\} \qquad (3.22)\\
\dot{\lambda}_V &= -\Big\{\lambda_\gamma\Big[-\frac{T\sin\alpha}{V^2 m} - \frac{C_d\rho S_{ref}\sin\alpha}{2m} + \frac{C_l\rho S_{ref}\cos\alpha}{2m} + \frac{g}{V^2}\cos\gamma\Big] \\
&\quad -\frac{\lambda_V\rho V S_{ref}}{m}\Big[C_d\cos\alpha + C_l\sin\alpha\Big] + \lambda_x\cos\gamma + \lambda_h\sin\gamma \\
&\quad +\mu_3\Big[\frac{d\dot{V}}{dV}\sin\gamma + \frac{d\dot{\gamma}}{dV}V\cos\gamma + \dot{\gamma}\cos\gamma\Big]\Big\} \qquad (3.23)\\
\dot{\lambda}_x &= 0 \qquad (3.24)\\
\dot{\lambda}_h &= -\Big\{1 - \frac{\lambda_\gamma V S_{ref}\rho_h}{2m}\Big[C_d\sin\alpha - C_l\cos\alpha\Big] \\
&\quad -\frac{\lambda_V V^2 S_{ref}\rho_h}{2m}\Big[C_d\cos\alpha + C_l\sin\alpha\Big] \\
&\quad +\mu_3\Big[\frac{d\dot{V}}{dh}\sin\gamma + \frac{d\dot{\gamma}}{dh}V\cos\gamma\Big]\Big\}, \qquad (3.25)
\end{aligned}$$

Using Pontryagin's Minimum Principle for (3.21) yields:

$$\begin{aligned}
H_\alpha^{aa} &= (T - D + L_\alpha)\Big[\frac{\lambda_\gamma}{Vm}\cos\alpha - \frac{\lambda_V}{m}\sin\alpha\Big] \\
&\quad -(D_\alpha + L)\Big[\frac{\lambda_\gamma}{Vm}\sin\alpha + \frac{\lambda_V}{m}\cos\alpha\Big] \\
&\quad +\mu_3\Big[\frac{d\dot{V}}{d\alpha}\sin\gamma + \frac{d\dot{\gamma}}{d\alpha}V\cos\gamma\Big] = 0.
\end{aligned}$$

The thrust is on the maximum values. From (3.19b)–(3.19c) it follows that $\gamma = 0$ and $\dot{\gamma} = 0$, so we obtain the following reduced equations:

- State equations:

$$\begin{aligned}
\dot{\gamma} &= \frac{T-D}{m}\sin\alpha + \frac{L}{m}\cos\alpha - g = 0 \\
\dot{V} &= \frac{T-D}{m}\cos\alpha - \frac{L}{m}\sin\alpha \\
\dot{x} &= V \\
\dot{h} &= 0
\end{aligned}$$

- Co-state equations:

$$\begin{aligned}
\dot{\lambda}_\gamma &= -\Big\{ -\lambda_V g + \lambda_h V + \mu_3\Big[\frac{T-D}{m}\cos\alpha - \frac{L}{m}\sin\alpha\Big]\Big\} \\
\dot{\lambda}_V &= -\Big\{(\lambda_\gamma + \mu_3 V)\Big[-\frac{C_d\rho S_{ref}\sin\alpha}{2m} - \frac{T\sin\alpha}{V^2 m} + \frac{C_l\rho S_{ref}\cos\alpha}{2m} + \frac{g}{V^2}\Big] \\
&\quad -\frac{\lambda_V\rho V S_{ref}}{m}\Big[C_d\cos\alpha + C_l\sin\alpha\Big] + \lambda_x\Big\} \\
\dot{\lambda}_x &= 0 \\
\dot{\lambda}_h &= -\Big\{1 + \frac{(\lambda_\gamma + \mu_3 V)V S_{ref}\rho_h}{2m}\Big[-C_d\sin\alpha + C_l\cos\alpha\Big] \\
&\quad -\frac{\lambda_V V^2 S_{ref}\rho_h}{2m}\Big[C_d\cos\alpha + C_l\sin\alpha\Big]\Big\}
\end{aligned}$$

- Optimality condition

$$\begin{aligned}
H_\alpha^{aa} &= (T - D + L_\alpha)\Big[\Big(\frac{\lambda_\gamma}{V} + \mu_3 V\Big)\frac{\cos\alpha}{m} - \frac{\lambda_V}{m}\sin\alpha\Big] \\
&\quad -(D_\alpha + L)\Big[\Big(\frac{\lambda_\gamma}{V} + \mu_3 V\Big)\frac{\sin\alpha}{m} + \frac{\lambda_V}{m}\cos\alpha\Big] = 0 \qquad (3.27)
\end{aligned}$$

The angle of attack α can be obtained from equation (3.20a). Lagrange multiplier μ_3 can be derived explicitly from equation (3.27) and substituted into state and co-state equations. Since we know the flight path angle and the altitude during this manoeuvre, the number of differential equations reduces to six. The main difficulty in solving this problem is that we do not know an initial guess for the co-state variables.

3.3.4 Second Arc: Climbing

In this analysis we focus on the case of final speed 250 m/s only and consider the thrust and normal acceleration constraints. From the qualitative analysis in Section 3.2, we know that the thrust control switches to the minimum value during climbing for final speed 250 m/s, therefore the switching function must change sign from negative to positive.

Consider mixed state-control inequality constraints as mentioned in equation (1.13) as follows:

$$L_{min} \leqslant \frac{L}{mg} \leqslant L_{max}$$

and L explicitly depends on the control α. The inclusion of the mixed constraints above leads to the augmented Hamiltonian:

$$H^{al} = H^{af} + \mu_1\Big(-\frac{L}{mg} + L_{min}\Big) + \mu_2\Big(\frac{L}{mg} - L_{max}\Big) \tag{3.29}$$

The right-hand side of the differential equations for the co-state equations are to be modified along subarcs of this second arc. Additionally, we have a necessary sign condition for the Lagrange parameter μ_i,

$$\mu_1 \begin{cases} = 0 & \text{on unconstrained subarcs} \\ = \frac{H_\alpha^{af} mg}{L_\alpha} & \text{on constrained subarcs} \end{cases}$$

and

$$\mu_2 \begin{cases} = 0 & \text{on unconstrained subarcs} \\ = -\frac{H_\alpha^{af} mg}{L_\alpha} & \text{on constrained subarcs} \end{cases}$$

The angle of attack α can be determined as follows:

- Optimality condition when normal acceleration is on the maximum value

$$\begin{aligned} H_\alpha^{al} = {} & (T - D + L_\alpha)\Big[\frac{\lambda_\gamma}{Vm}\cos\alpha - \frac{\lambda_V}{m}\sin\alpha\Big] \\ & -(D_\alpha + L)\Big[\frac{\lambda_\gamma}{Vm}\sin\alpha + \frac{\lambda_V}{m}\cos\alpha\Big] + \mu_2\Big(\frac{L_\alpha}{mg}\Big) = 0 \end{aligned} \tag{3.30}$$

- Optimality condition when normal acceleration is on the minimum value

$$\begin{aligned} H_{\alpha}^{al} &= (T - D + L_{\alpha})\Big[\frac{\lambda_{\gamma}}{Vm}\cos\alpha - \frac{\lambda_V}{m}\sin\alpha\Big] \\ &\quad -(D_{\alpha} + L)\Big[\frac{\lambda_{\gamma}}{Vm}\sin\alpha + \frac{\lambda_V}{m}\cos\alpha\Big] - \mu_1\Big(\frac{L_{\alpha}}{mg}\Big) = 0 \end{aligned} \tag{3.31}$$

When the normal acceleration constraint is active (L_{max}), the angle of attack can be determined from (1.6) as

$$\alpha = \frac{2mgL_{max} - B_2\rho V^2 S_{ref}}{B_1\rho V^2 S_{ref}}. \tag{3.32}$$

Equation (3.32) is valid until the normal acceleration and the thrust switch to minimum value (see Figures 3.2c–3.2d).

Below we summarise the results for the case when the normal acceleration is saturated on the maximum value (L_{max}).

- State equations:

$$\begin{aligned} \dot{\gamma} &= \left\{\frac{T-D}{m}\sin\alpha + \frac{L}{m}\cos\alpha - g\cos\gamma\right\}\frac{1}{V} \\ \dot{V} &= \frac{T-D}{m}\cos\alpha - \frac{L}{m}\sin\alpha - g\sin\gamma \\ \dot{x} &= V\cos\gamma \\ \dot{h} &= V\sin\gamma \end{aligned}$$

- Co-state equations:

$$\begin{aligned} \dot{\lambda}_{\gamma} &= -\left\{\frac{\lambda_{\gamma}}{V}g\sin\gamma - \lambda_V g\cos\gamma - \lambda_x V\sin\gamma + \lambda_h V\cos\gamma\right\} \\ \dot{\lambda}_V &= -\left\{\lambda_{\gamma}\left[-\frac{T\sin\alpha}{V^2 m} - \frac{C_d\rho S_{ref}\sin\alpha}{2m} + \frac{C_l\rho S_{ref}\cos\alpha}{2m} + \frac{g}{V^2}\cos\gamma\right]\right. \\ &\quad -\frac{\lambda_V\rho V S_{ref}}{m}\Big[C_d\cos\alpha + C_l\sin\alpha\Big] \\ &\quad \left.+\lambda_x\cos\gamma + \lambda_h\sin\gamma + \mu_2\Big[\frac{L_V}{mg}\Big]\right\} \end{aligned}$$

$$\begin{aligned}\dot{\lambda}_x &= 0\\ \dot{\lambda}_h &= -\Bigg\{1-\frac{\lambda_\gamma V S_{ref}\rho_h}{2m}\Big[C_d\sin\alpha - C_l\cos\alpha\Big]\\ &\quad -\frac{\lambda_V V^2 S_{ref}\rho_h}{2m}\Big[C_d\cos\alpha + C_l\sin\alpha\Big] + \mu_2\Big[\frac{L_h}{mg}\Big]\Bigg\}\end{aligned}$$

- Optimality condition

$$\begin{aligned}H_\alpha^{al} &= (T-D+L_\alpha)\Big[\frac{\lambda_\gamma}{Vm}\cos\alpha - \frac{\lambda_V}{m}\sin\alpha\Big]\\ &\quad -(D_\alpha + L)\Big[\frac{\lambda_\gamma}{Vm}\sin\alpha + \frac{\lambda_V}{m}\cos\alpha\Big] + \mu_2\Big(\frac{L_\alpha}{mg}\Big) = 0 \quad (3.34)\end{aligned}$$

where

$$\alpha = \frac{2mgL_{max} - B_2\rho V^2 S_{ref}}{B_1\rho V^2 S_{ref}} \quad (3.35)$$

and the thrust switches to the minimum value when H_T^{al} changes sign from negative to positive.

3.3.5 Third Arc: Diving

In this analysis we consider only the normal acceleration constraint. During this time the thrust is on the maximum value and normal acceleration is saturated on the minimum value. The angle of attack can be determined from (1.6) as follows:

$$\alpha = \frac{2mgL_{min} - B_2\rho V^2 S_{ref}}{B_1\rho V^2 S_{ref}}. \quad (3.36)$$

The Hamiltonian and co-state equations are nearly the same as in the previous section, therefore the derivation is omitted here. The equations can be summarised as follows:

- State equations:

$$\begin{aligned}\dot{\gamma} &= \Big\{\frac{T-D}{m}\sin\alpha + \frac{L}{m}\cos\alpha - g\cos\gamma\Big\}\frac{1}{V}\\ \dot{V} &= \frac{T-L}{m}\cos\alpha - \frac{L}{m}\sin\alpha - g\sin\gamma\\ \dot{x} &= V\cos\gamma\\ \dot{h} &= V\sin\gamma\end{aligned}$$

- Co-state equations:

$$\begin{aligned}
\dot{\lambda}_\gamma &= -\left\{\frac{\lambda_\gamma}{V} g\sin\gamma - \lambda_V g\cos\gamma - \lambda_x V\sin\gamma + \lambda_h V\cos\gamma\right\} \\
\dot{\lambda}_V &= -\left\{\lambda_\gamma\left[-\frac{T\sin\alpha}{V^2 m} - \frac{C_d\rho S_{ref}\sin\alpha}{2m} + \frac{C_l\rho S_{ref}\cos\alpha}{2m} + \frac{g}{V^2}\cos\gamma\right]\right. \\
&\quad -\frac{\lambda_V\rho V S_{ref}}{m}\left[C_d\cos\alpha + C_l\sin\alpha\right] \\
&\quad \left.+\lambda_x\cos\gamma + \lambda_h\sin\gamma - \mu_1\left[\frac{L_V}{mg}\right]\right\} \\
\dot{\lambda}_x &= 0 \\
\dot{\lambda}_h &= -\left\{1 - \frac{\lambda_\gamma V S_{ref}\rho_h}{2m}\left[C_d\sin\alpha - C_l\cos\alpha\right]\right. \\
&\quad \left.-\frac{\lambda_V V^2 S_{ref}\rho_h}{2m}\left[C_d\cos\alpha + C_l\sin\alpha\right] - \mu_1\left[\frac{L_h}{mg}\right]\right\}
\end{aligned}$$

- Optimality condition

$$\begin{aligned}
H_\alpha^{al} &= (T - D + L_\alpha)\left[\frac{\lambda_\gamma}{Vm}\cos\alpha - \frac{\lambda_V}{m}\sin\alpha\right] \\
&\quad -(D_\alpha + L)\left[\frac{\lambda_\gamma}{Vm}\sin\alpha + \frac{\lambda_V}{m}\cos\alpha\right] - \mu_1\left(\frac{L_\alpha}{mg}\right) = 0 \qquad (3.38)
\end{aligned}$$

where

$$\alpha = \frac{2mgL_{min} - B_2\rho V^2 S_{ref}}{B_1\rho V^2 S_{ref}} \qquad (3.39)$$

The schematic representation of the boundary value problem associated with the switching structure can be seen in Figures 3.23–3.27 on pp. 83–91.

3.4 Indirect Method Solution

BNDSCO is a software package developed by Oberle, for references see [76], which implements a multiple shooting algorithm (see Stoer and Bulirsch [99], Keller [61] and Osborne [77]) and is a reliable solver of Multi Point Boundary Value Problem

(MPBVP) with discontinuities, specially written for solving optimal control problems. However, it has the weakness of all shooting methods that it has a narrow domain of convergence. Therefore initial guesses for the state and co-state variables are crucial for successful computation, especially the co-state variable which has no physical interpretation. Moreover, the task becomes more difficult when the problem has pure state constraints.

3.4.1 Co-state Approximation

Von Stryk [108] shows that the co-state variable can be estimated by the necessary conditions of the discretised problem of the optimal control. He developed the DIRCOL package [109] based on a direct collocation method and it has been used for solving several real-life problems (see von Stryk and Bulirsch [110] and von Stryk and Schlemmer [111], see also section 2.4.1.1). Grimm and Markl [51] estimated the co-state variables using direct multiple shooting method. Their co-state approximation is accurate for the unconstrained problem while it does not work well for the constrained problem. Fahroo and Ross [40] proposed a Legendre pseudospectral method to estimate the co-state variables and presented an accurate estimator for the unconstrained problem. Benson [6] proposed a Gauss pseudospectral transcription to solve optimal control problem and use it to approximate co-state variables. Again the co-state approximation does not give good initial guess for the pure state constrained problem.

This section presents an example of the pure state constrained problem which is the first arc of the terminal bunt manoeuvre. In this example, Bryson's and Jacobson's formulation are compared and then the DIRCOL package is used to approximate the co-state variables.

Jacobson et al. [57] presented a direct adjoining of pure state constraint to the Hamiltonian while Bryson et al. [22] proposed an indirect adjoining of pure state constraints to the Hamiltonian. In Bryson's approach the pure state constraint is differentiated until u appears explicitly and then the resulting equation is adjoined to the Hamiltonian (see section 3.3.3). Consider now the following Bryson's formulation:

$$S = h - h_{min} = 0 \tag{3.40a}$$

$$S^{(1)} = \dot{h} = V \sin\gamma = 0 \quad \text{and} \quad V \neq 0$$

$$\Rightarrow \gamma(t) = 0 \quad \text{for} \quad t \in [t_0, t_1] \tag{3.40b}$$

$$S^{(2)} = \ddot{h} = \dot{V} \sin\gamma + V\dot{\gamma}\cos\gamma = 0$$

$$\Rightarrow \dot{\gamma}(t) = 0 \quad \text{for} \quad t \in [t_0, t_1] \tag{3.40c}$$

Thus the constraint is of second order, see section 3.3.3, as controls appear in $\dot{\gamma}$ and $\dot{V}$, see equations (1.3a) and (1.3b) on page 4. The Hamiltonian for Bryson's formulation can be defined as:

$$H^B = \boldsymbol{\lambda}^B \boldsymbol{f} + \mu S^{(2)} \tag{3.41}$$

In contrast to the Bryson's formulation, the Hamiltonian for Jacobson's formulation is given by

$$H^J = \boldsymbol{\lambda}^J \boldsymbol{f} + \nu S \tag{3.42}$$

Note that $\boldsymbol{\lambda}^B \neq \boldsymbol{\lambda}^J$, in general, because of different definitions of the Hamiltonian.

The direct method approach for optimal control mainly uses Jacobson's formulation in the derivation of the Karush-Kuhn-Tucker (KKT) necessary conditions (see section 2.2). Therefore the co-state estimation from the direct method is accurate for the problem having a mixed constraint while it may not work well for the problem having a pure state constraint.

Thus, in general, for a pure state constraint situation DIRCOL will compute $\boldsymbol{\lambda}_J$, while BNDSCO will need $\boldsymbol{\lambda}_B$. The following example gives some insights into the different co-state estimation for Bryson's and Jacobson's formulation using DIRCOL. In this example DIRCOL is implemented for the first arc only because the minimum altitude constraint is active in this arc. Both co-state estimation methods are then used as initial guesses for BNDSCO but it does not work well. Figures 3.14 and 3.16 show

DIRCOL solutions obtained using the following data:

$$\begin{aligned} \gamma_0 &= 0 \text{ deg}, & \gamma_{t_f} &= 0 \text{ deg} \\ V_0 &= 272 \text{ m/s}, & V_{t_f} &= 306.324004 \text{ m/s} \\ x_0 &= 0 \text{ m}, & x_{t_f} &= 5813.44774 \text{ m} \\ h_0 &= 30 \text{ m}, & h_{t_f} &= 30 \text{ m}. \end{aligned}$$

The following equation shows the differences in the co-state equations for Bryson's and Jacobson's formulation.

- state and co-state equations for the Bryson's formulation

$$\dot{\gamma} = \frac{T-D}{m}\sin\alpha + \frac{L}{m}\cos\alpha - g = 0 \tag{3.43}$$

$$\dot{V} = \frac{T-L}{m}\cos\alpha - \frac{L}{m}\sin\alpha \tag{3.44}$$

$$\dot{x} = V \tag{3.45}$$

$$\dot{h} = 0 \tag{3.46}$$

$$\dot{\lambda}_\gamma^B = -\left\{ -\lambda_V^B g + \lambda_h^B V + \mu\left[\frac{T-D}{m}\cos\alpha - \frac{L}{m}\sin\alpha\right]\right\} \tag{3.47}$$

$$\begin{aligned} \dot{\lambda}_V^B = -\Bigg\{ &(\lambda_\gamma^B + \mu V)\left[-\frac{C_d \rho S_{ref}\sin\alpha}{2m} - \frac{T\sin\alpha}{V^2 m} + \frac{C_l \rho S_{ref}\cos\alpha}{2m} + \frac{g}{V^2}\right] \\ &-\frac{\lambda_V^B \rho V S_{ref}}{m}\left[C_d\cos\alpha + C_l\sin\alpha\right] + \lambda_x^B\Bigg\} \end{aligned} \tag{3.48}$$

$$\dot{\lambda}_x^B = 0 \tag{3.49}$$

$$\begin{aligned} \dot{\lambda}_h^B = -\Bigg\{ &1 + \frac{(\lambda_\gamma^B + \mu V) V S_{ref}\rho_h}{2m}\left[-C_d\sin\alpha + C_l\cos\alpha\right] \\ &-\frac{\lambda_V^B V^2 S_{ref}\rho_h}{2m}\left[C_d\cos\alpha + C_l\sin\alpha\right]\Bigg\} \end{aligned} \tag{3.50}$$

- state and co-state equations for the Jacobson's formulation

$$\dot{\gamma} = \frac{T-D}{mV}\sin\alpha + \frac{L}{mV}\cos\alpha - \frac{g\cos\gamma}{V} \tag{3.51}$$

$$\dot{V} = \frac{T-D}{m}\cos\alpha - \frac{L}{m}\sin\alpha - g\sin\gamma \tag{3.52}$$

$$\dot{x} = V\cos\gamma \tag{3.53}$$

$$\dot{h} = V\sin\gamma \tag{3.54}$$

$$\dot{\lambda}_\gamma^J = -\left\{\frac{\lambda_\gamma^J}{V}g\sin\gamma - \lambda_V^J g\cos\gamma - \lambda_x^J V\sin\gamma + \lambda_h^J V\cos\gamma\right\} \tag{3.55}$$

$$\begin{aligned}\dot{\lambda}_V^J = &-\left\{\lambda_\gamma^J\left[-\frac{T\sin\alpha}{V^2 m} - \frac{C_d\rho S_{ref}\sin\alpha}{2m} + \frac{C_l\rho S_{ref}\cos\alpha}{2m} + \frac{g}{V^2}\cos\gamma\right]\right. \\ &\left. -\frac{\lambda_V^J\rho V S_{ref}}{m}\left[C_d\cos\alpha + C_l\sin\alpha\right] + \lambda_x^J\cos\gamma + \lambda_h^J\sin\gamma\right\}\end{aligned} \tag{3.56}$$

$$\dot{\lambda}_x^J = 0 \tag{3.57}$$

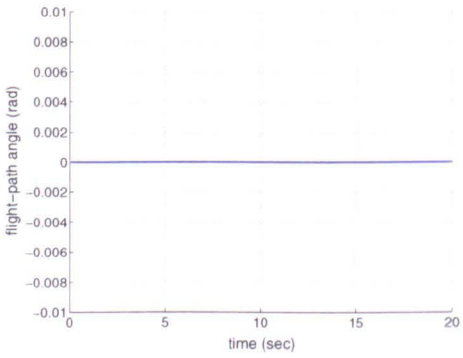
$$\begin{aligned}\dot{\lambda}_h^J = &-\left\{1 - \frac{\lambda_\gamma^J V S_{ref}\rho_h}{2m}\left[C_d\sin\alpha - C_l\cos\alpha\right]\right. \\ &\left. -\frac{\lambda_V^J V^2 S_{ref}\rho_h}{2m}\left[C_d\cos\alpha + C_l\sin\alpha\right] + \nu\right\}\end{aligned} \tag{3.58}$$

Figures 3.14 and 3.16 show that the computational results for the co-state variables are very different. It is obvious because the constraint which is adjoined to the Hamiltonian differs for both cases. However, the state variables give the same approximate solutions (see Figure 3.13 and 3.15).

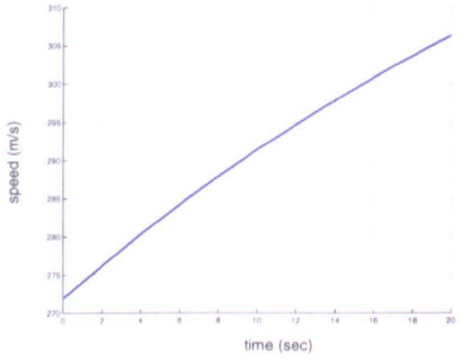
Note that, when solving the TPBVP corresponding to (3.51)–(3.58), finding ν in (3.58) does not lend itself to a systematic iterative procedure, as pointed out by Maurer and Gillesen [74, Page 111]). On the other hand, μ in (3.43)–(3.50) can be found readily using the conditions $\dot{\gamma} = 0$ and $H_\alpha^B = 0$.

3.4.2 Switching and Jump Conditions

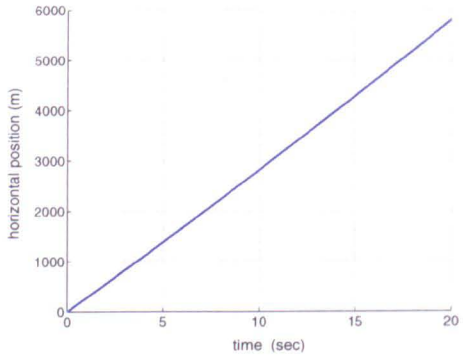
In the previous section we showed that Bryson's and Jacobson's formulation produce very different $\boldsymbol{\lambda}$, see Figures 3.14 and 3.16. In this section we compare the differences



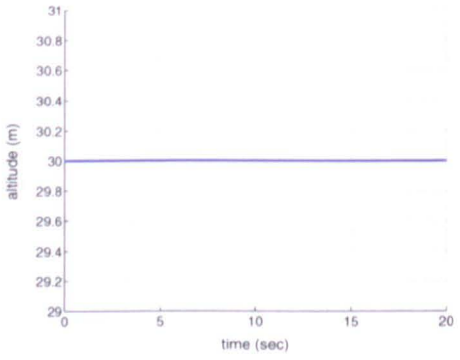
(a) Flight-path angle versus time



(b) Speed versus time

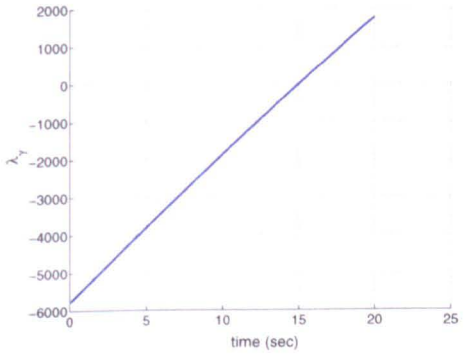


(c) Horizontal position versus time

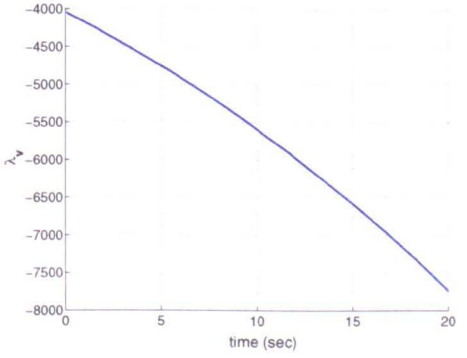


(d) Altitude versus time

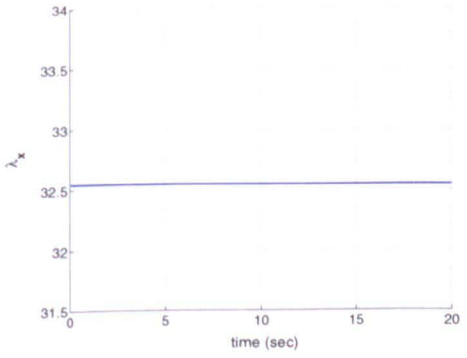
Figure 3.13: DIRCOL computational results for the state variables of the Bryson's formulation.



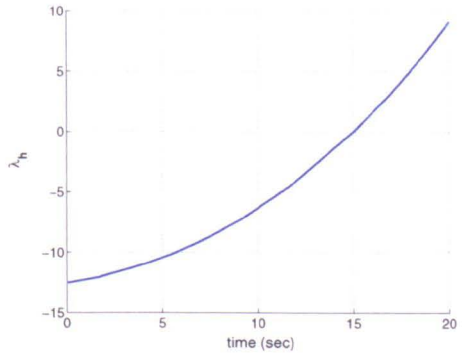
(a) λ_γ versus time



(b) λ_V versus time

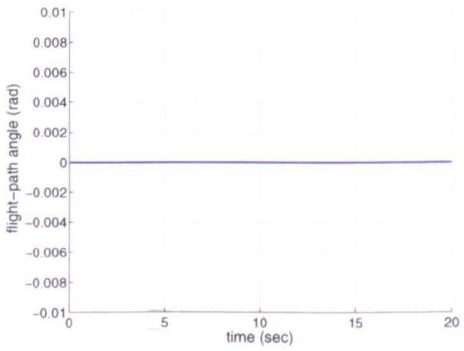


(c) λ_x versus time

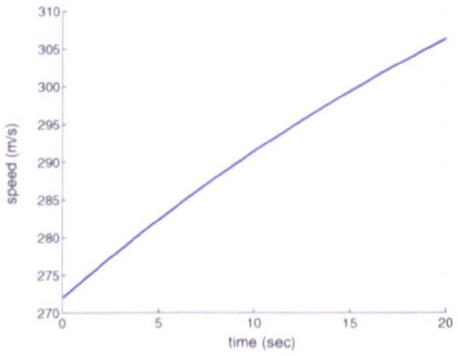


(d) λ_h versus time

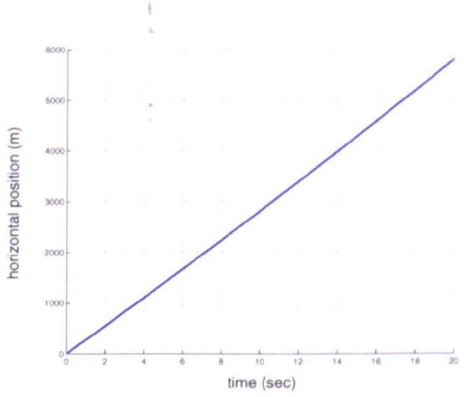
Figure 3.14: DIRCOL computational results for the co-state variables of the Bryson's formulation.



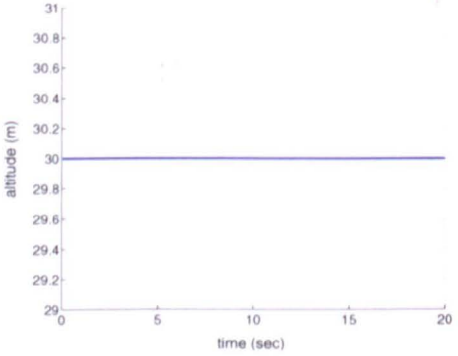
(a) Flight-path angle versus time



(b) Speed versus time

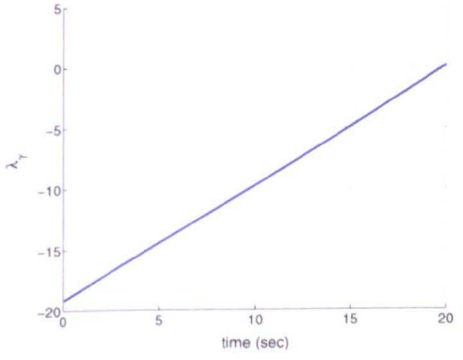


(c) Horizontal position versus time

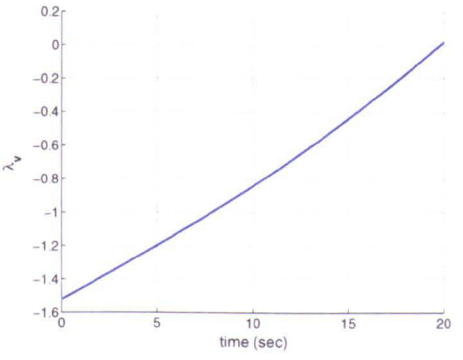


(d) Altitude versus time

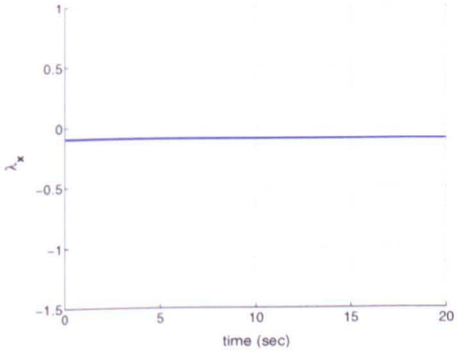
Figure 3.15: DIRCOL computational results for the state variables of the Jacobson's formulation.



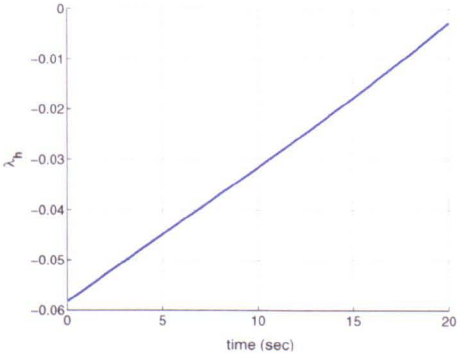
(a) λ_γ versus time



(b) λ_V versus time



(c) λ_x versus time



(d) λ_h versus time

Figure 3.16: DIRCOL computational results for the co-state variables of the Jacobson's formulation. Note the different magnitudes compared to Figure 3.14.

in switching and jump conditions for Bryson's and Jacobson's formulations, further to emphasise the consequences of different definitions of co-state variables $\boldsymbol{\lambda}$.

3.4.2.1 Bryson's Formulation

The switching and jump conditions at entry t_{en} and exit t_{ex} will be considered. The jump conditions at entry point can be derived from tangency constraint $P(\boldsymbol{x}) = 0$ as follows. Consider the following equations of Bryson's formulation

$$P(\boldsymbol{x}) = \begin{bmatrix} P_0(\boldsymbol{x}) \\ P_1(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} S(\boldsymbol{x}) \\ \dot{S}(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} h - h_{min} \\ V \sin\gamma \end{bmatrix} = 0 \tag{3.59}$$

If we assume the jump occurred at the entry point then the jump conditions are given by:

$$\boldsymbol{\lambda}^+ = \boldsymbol{\lambda}^- - N_{\boldsymbol{x}}^T \boldsymbol{\sigma} \tag{3.60}$$

with

$$P_{\boldsymbol{x}} = \nabla_{\boldsymbol{x}} N = \begin{bmatrix} (\nabla_{\boldsymbol{x}} P_0)^T \\ (\nabla_{\boldsymbol{x}} P_1)^T \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ V\cos\gamma & \sin\gamma & 0 & 0 \end{bmatrix} \tag{3.61}$$

Thus equation (3.60) can be rewritten as:

$$\begin{bmatrix} \lambda_\gamma(t_{en}^+) \\ \lambda_v(t_{en}^+) \\ \lambda_x(t_{en}^+) \\ \lambda_h(t_{en}^+) \end{bmatrix} = \begin{bmatrix} \lambda_\gamma(t_{en}^-) \\ \lambda_v(t_{en}^-) \\ \lambda_x(t_{en}^-) \\ \lambda_h(t_{en}^-) \end{bmatrix} - \begin{bmatrix} 0 & V\cos\gamma \\ 0 & \sin\gamma \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} \tag{3.62}$$

or

$$\lambda_\gamma(t_{en}^+) = \lambda_\gamma(t_{en}^-) - \sigma_2 V\cos\gamma \tag{3.63a}$$

$$\lambda_v(t_{en}^+) = \lambda_v(t_{en}^-) - \sigma_2 \sin\gamma \tag{3.63b}$$

$$\lambda_x(t_{en}^+) = \lambda_x(t_{en}^-) \tag{3.63c}$$

$$\lambda_h(t_{en}^+) = \lambda_h(t_{en}^-) - \sigma_1 \tag{3.63d}$$

By substituting $\gamma = 0$ to equation (3.63), the above equation can be reduced as follows:

$$\lambda_\gamma(t_{en}^+) = \lambda_\gamma(t_{en}^-) - \sigma_2 V \tag{3.64a}$$
$$\lambda_v(t_{en}^+) = \lambda_v(t_{en}^-) \tag{3.64b}$$
$$\lambda_x(t_{en}^+) = \lambda_x(t_{en}^-) \tag{3.64c}$$
$$\lambda_h(t_{en}^+) = \lambda_h(t_{en}^-) - \sigma_1 \tag{3.64d}$$

The switching conditions at the entry point are given by:

$$P(\boldsymbol{x}) = \begin{bmatrix} h - h_{min} \\ V \sin\gamma \end{bmatrix} = 0 \tag{3.65}$$

and

$$\begin{aligned} H_{t_{en}}^+ - H_{t_{en}}^- &= 0 \\ &\Rightarrow -\sigma_2 V\dot{\gamma} - \sigma_1 \dot{h} + \mu_3 \left[\dot{V}\sin\gamma + V\dot{\gamma}\cos\gamma\right] = 0 \end{aligned} \tag{3.66}$$

with

$$H_{t_{en}}^- = \lambda_\gamma^-\dot{\gamma} + \lambda_V^-\dot{V} + \lambda_x^-\dot{x} + \lambda_h^-\dot{h} \tag{3.67}$$
$$H_{t_{en}}^+ = \lambda_\gamma^+\dot{\gamma} + \lambda_V^+\dot{V} + \lambda_x^+\dot{x} + \lambda_h^+\dot{h} + \mu_3\left[\dot{V}\sin\gamma + V\dot{\gamma}\cos\gamma\right] \tag{3.68}$$

The switching conditions at exit point t_{ex} are:

$$\begin{aligned} H_{t_{ex}}^+ - H_{t_{ex}}^- &= 0 \\ &\Rightarrow \dot{V}\sin\gamma + V\dot{\gamma}\cos\gamma = 0 \end{aligned} \tag{3.69}$$

3.4.2.2 Kreindler's Remarks

Equation (3.19) shows that the altitude constraint is order 2 ($q = 2$). Based on Kreindler's remarks in [66, page 244], the constant multipliers σ_i are unique except possibly σ_{q-1}. If an arbitrary constant ξ is added to σ_{q-1} then the discontinuity occurs

at exit point by $-\xi\nabla_x P_1$. Consider now the following equations:

$$P_1 = V\sin\gamma \tag{3.70}$$

$$\nabla_x P_1 = \begin{bmatrix} V\cos\gamma \\ \sin\gamma \\ 0 \\ 0 \end{bmatrix} \tag{3.71}$$

Thus the jump conditions at entry and exit points can be derived as follows:

- Jump conditions at entry point

$$\lambda_\gamma(t_{en}^+) = \lambda_\gamma(t_{en}^-) - (\sigma_2+\xi)V\cos\gamma \tag{3.72a}$$

$$\lambda_v(t_{en}^+) = \lambda_v(t_{en}^-) - (\sigma_2+\xi)\sin\gamma \tag{3.72b}$$

$$\lambda_x(t_{en}^+) = \lambda_x(t_{en}^-) \tag{3.72c}$$

$$\lambda_h(t_{en}^+) = \lambda_h(t_{en}^-) - \sigma_1 \tag{3.72d}$$

- Jump conditions at exit point

$$\lambda_\gamma(t_{ex}^+) = \lambda_\gamma(t_{ex}^-) - \xi V\cos\gamma \tag{3.73a}$$

$$\lambda_v(t_{ex}^+) = \lambda_v(t_{ex}^-) - \xi\sin\gamma \tag{3.73b}$$

$$\lambda_x(t_{ex}^+) = \lambda_x(t_{ex}^-) \tag{3.73c}$$

$$\lambda_h(t_{ex}^+) = \lambda_h(t_{ex}^-) \tag{3.73d}$$

3.4.2.3 Necessary Conditions of Jacobson et al.

In this case the state constraint (3.19a) is adjoined to the Hamiltonian directly with multiplier function ν, $\nu \geqslant 0$ (see Jacobson and Lele [57] and Kreindler [66]). The necessary conditions can be derived by defining the Hamiltonian as follows:

$$H^{mas}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\lambda},\nu) = \boldsymbol{\lambda}^T f(\boldsymbol{x},\boldsymbol{u}) + \nu S(\boldsymbol{x}) \tag{3.74}$$

The jump conditions at entry and exit points can be derived as follows

$$\boldsymbol{\lambda}^+(t_i) = \boldsymbol{\lambda}^-(t_i) - \nu_i S_{\boldsymbol{x}} \tag{3.75}$$

where the scalar ν_i is nonnegative,

$$\nu_i \geqslant 0 \tag{3.76}$$

The jump conditions at entry point:

$$\begin{bmatrix} \lambda_\gamma(t_{en}^+) \\ \lambda_v(t_{en}^+) \\ \lambda_x(t_{en}^+) \\ \lambda_h(t_{en}^+) \end{bmatrix} = \begin{bmatrix} \lambda_\gamma(t_{en}^-) \\ \lambda_v(t_{en}^-) \\ \lambda_x(t_{en}^-) \\ \lambda_h(t_{en}^-) \end{bmatrix} - \begin{bmatrix} \nu_1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.77}$$

or

$$\lambda_\gamma(t_{en}^+) = \lambda_\gamma(t_{en}^-) \tag{3.78a}$$
$$\lambda_v(t_{en}^+) = \lambda_v(t_{en}^-) \tag{3.78b}$$
$$\lambda_x(t_{en}^+) = \lambda_x(t_{en}^-) \tag{3.78c}$$
$$\lambda_h(t_{en}^+) = \lambda_h(t_{en}^-) - \nu_1 \tag{3.78d}$$

The jump conditions at exit point:

$$\begin{bmatrix} \lambda_\gamma(t_{ex}^+) \\ \lambda_v(t_{ex}^+) \\ \lambda_x(t_{ex}^+) \\ \lambda_h(t_{ex}^+) \end{bmatrix} = \begin{bmatrix} \lambda_\gamma(t_{ex}^-) \\ \lambda_v(t_{ex}^-) \\ \lambda_x(t_{ex}^-) \\ \lambda_h(t_{ex}^-) \end{bmatrix} - \begin{bmatrix} \nu_2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.79}$$

or

$$\lambda_\gamma(t_{ex}^+) = \lambda_\gamma(t_{ex}^-) \tag{3.80a}$$
$$\lambda_v(t_{ex}^+) = \lambda_v(t_{ex}^-) \tag{3.80b}$$
$$\lambda_x(t_{ex}^+) = \lambda_x(t_{ex}^-) \tag{3.80c}$$
$$\lambda_h(t_{ex}^+) = \lambda_h(t_{ex}^-) - \nu_2 \tag{3.80d}$$

The jump conditions given by Jacobson et al. are consistent with the DIRCOL results (see Figure 3.12 on page 49). This is not surprising, because the constraints are adjoined directly in KKT necessary conditions.

3.4.3 Numerical Solution

The computational results for the terminal bunt problem were computed by the multiple shooting package BNDSCO [76]. The problem is split it up into two phases: the first phase is the level flight and the rest of the manoeuvre is the second phase. The time t_i at which the transition from phase 1 to 2 occurs must be determined as a part of the BVP problem. In the presented solution t_i was estimated from the direct method approximation. This is a sub-optimal solution obtained using the following data:

$$\begin{aligned} \gamma_0 &= 0 \text{ deg}, & \gamma_{t_f} &= -90 \text{ deg} \\ V_0 &= 272 \text{ m/s}, & V_{t_f} &= 310 \text{ m/s} \\ x_0 &= 0 \text{ m}, & x_{t_f} &= 10000 \text{ m} \\ h_0 &= 30 \text{ m}, & h_{t_f} &= 0 \text{ m}. \end{aligned}$$

where the intermediate time $t_i = 10.45$ sec. The minimum performance index is 40445.48347 m^2 and final time 41.4789 sec. In the first phase the minimum altitude h_{min} constraint is active. The missile starts to climb and then dive to reach the target by a bunt manoeuvre which is considered in the second phase. The minimum normal acceleration constraint is active during diving. Figures 3.17–3.21 show that the DIRCOL solutions for the state variables are close enough to the BNDSCO solutions. However, the co-state approximation does not work well in the first phase.

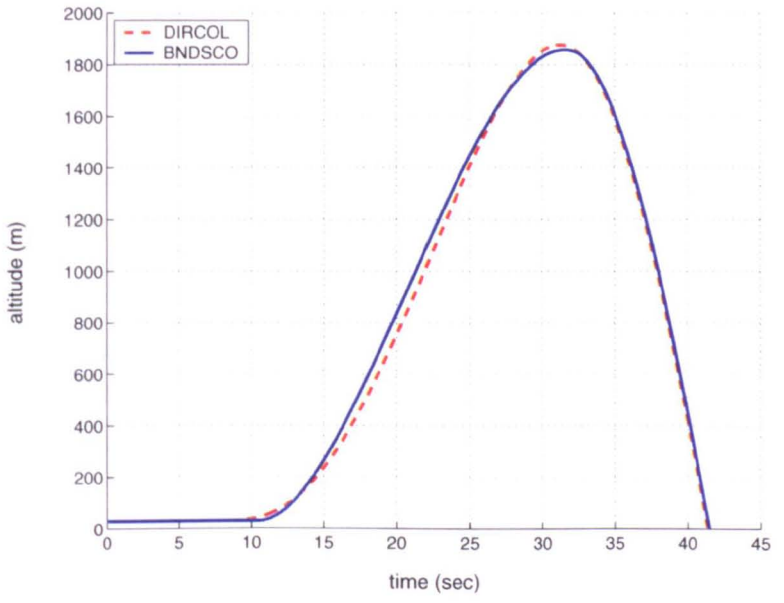


Figure 3.17: Altitude versus time histories using BNDSCO.

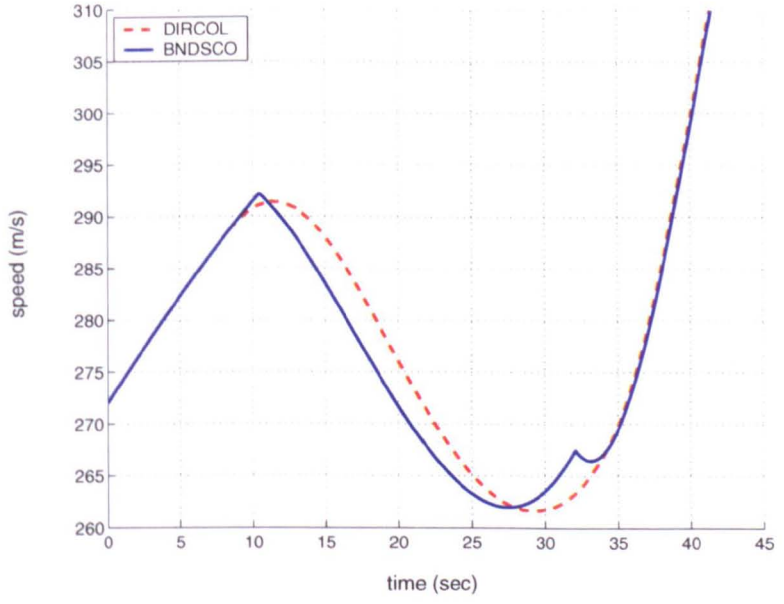


Figure 3.18: Speed versus time histories using BNDSCO.

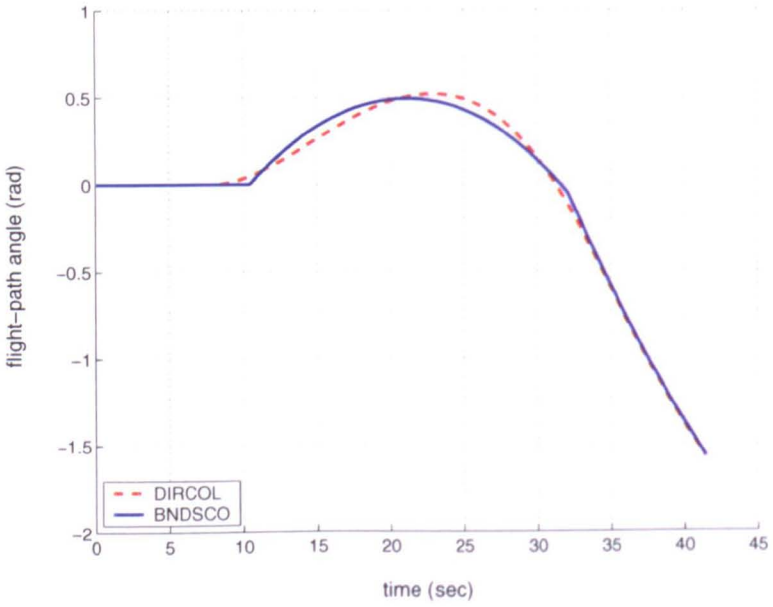


Figure 3.19: Flight-path angle versus time histories using BNDSCO.

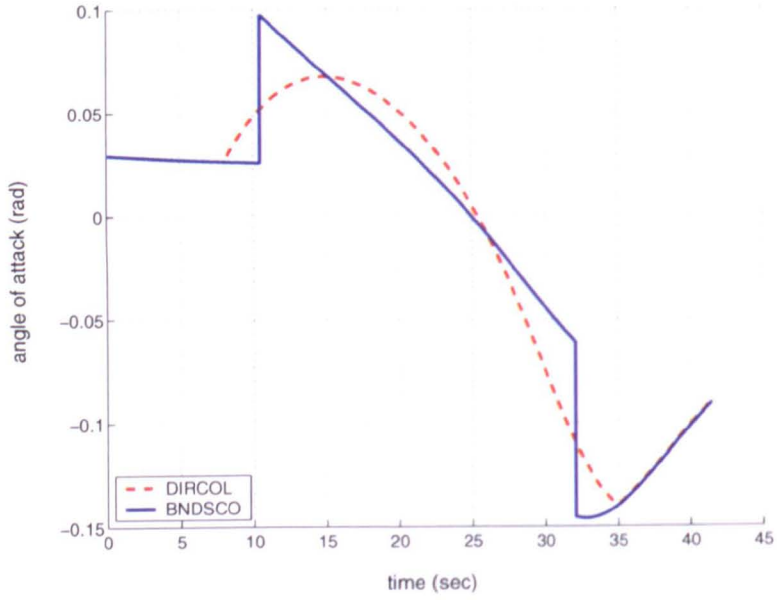


Figure 3.20: Angle of attack versus time histories using BNDSCO.

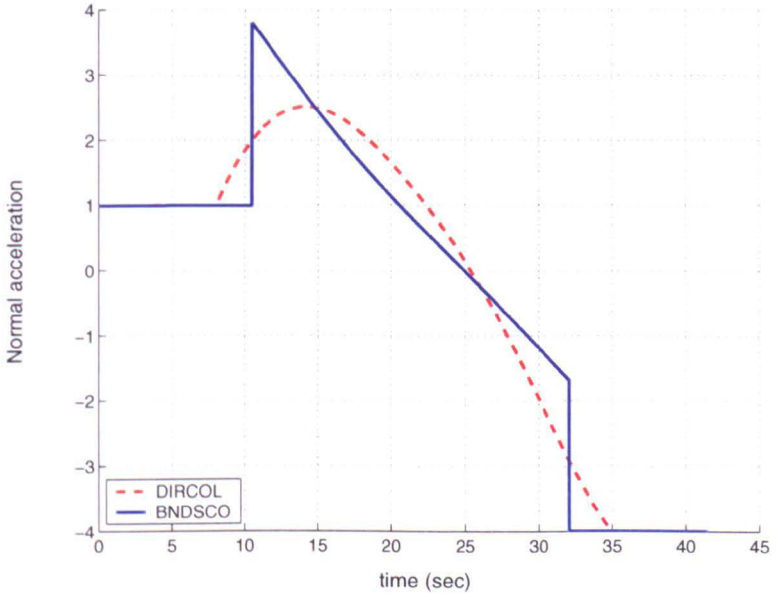


Figure 3.21: Normal acceleration versus time histories using BNDSCO.

3.5 Summary and Discussion

The study of computational results for the minimum altitude of terminal bunt manoeuvre with varying final speed is important from the operational viewpoint. Since the mission is to strike a fixed target while minimising the missile exposure to anti-air defences, one should consider both the type of target and the exposure of the missile during the manoeuvre. If the mission to strike a bunker, it is important to hit the target with the maximum capability of the missile. If the target's prosecution may lead to collateral damage, then a more measured impact is advisable, so that the final speed should be lower. It is always important to avoid anti-air defences during the manoeuvre, so optimal exposure must be taken into account. Based on the computational result using DIRCOL the optimal exposure for maximum speed 310 m/s is more than twice bigger compared to final speed 250 m/s, while the manoeuvre time is not much different (see Table 3.1, page 52). Hence, if the mission has a risk of collateral damage, the final speed 250 m/s or 270 m/s is a better choice than 310 m/s, also because the anti-air defences have lees time to intercept the missile (see Figure 3.3). While the

Table 3.2: Performance index and final time for the minimum altitude with varying initial altitude using DIRCOL

Initial altitude h_0 (m/s)	J	t_f (sec)	grid points
100	16894.4321	40.817911	140
200	17176.9987	40.945641	144
500	18423.3035	41.200737	139

final speed 310 m/s trajectory has higher exposure, it has a comparable flight time, but much higher terminal kinetic energy.

The above parametric study assumed that the missile launches from a ship, $h_0 =$ 30m. The second parametric study is by varying initial altitude of the missile. In this case the missile is assumed to be launched from an aircraft. In this case the missile is launched at 100m, 200m and 500m (see Figures B.1–B.6 on pp. 172–175). It can be seen from Table 3.2 that the influence of the initial altitude is not really significant for the performance index and final time.

Even though the direct collocation does not give an accurate solution, the numerical solutions give a starting point to analyse the performance of the missile during the bunt manoeuvre. The optimal trajectory of the manoeuvre can be split it up into three main arcs.

The first arc is level flight at the minimum altitude. The thrust is on the maximum value and the pure state constraint is active which is the minimum altitude constraint. The flight time is longer for the smaller final speed which means that the missile tries to climb as late as possible to gain enough power to do the bunt manoeuvre to satisfy the final speed. This arc is the most difficult one to compute because the pure state constraint is active. DIRCOL package can solve this arc and gives a good insight into the problem.

In the second arc the missile must climb in order to achieve the final condition. The thrust is still on the maximum value for some cases while for the case 250 m/s the thrust switches to the minimum value. The details of the switching structure of the equations and the constraints can be seen in Figure 3.23–3.27 on pp. 83–91. The normal constraint is active directly for the case 250 m/s and at the beginning of climbing

the normal acceleration is on the maximum value. The normal acceleration switches to the minimum value following the switching of the thrust to the minimum value. It is important to notice that the DIRCOL solutions produce a free arc (no constraint active except the thrust saturated on the maximum value) at the beginning of the climbing for the case 270 m/s and then saturated on the maximum value until the missile approaching to dive. Again for the same case 270 m/s, DIRCOL produces a free arc during this arc. Just before the missile turns over to dive, the normal acceleration constraint is active again and is saturated on the minimum value. The thrust is the only active constraint for the case 310 m/s in this arc.

The third arc is diving. Since the initial diving speed is lower than the final speed, the missile must gain the power to reach the target. Therefore the thrust is on the maximum value. It can be seen in Figure 3.23 that the thrust switches to the maximum value for the case 250 m/s. The normal acceleration is saturated on the minimum value for the case 250 m/s and 270 m/s, while for the case 310 m/s the minimum normal acceleration is active for just a few seconds after the missile starts to dive.

3.5.1 Comments on Switching Structure

An intriguing feature of the DIRCOL solutions in Figures 3.3–3.12 is discontinuous jumps in the angle of attack α, particularly for final speeds 250 m/s and 270 m/s, see Figure 3.6 on page 46. Both the thrust T and the angle of attack α are controls, but T enters the Hamiltonian linearly (thus allowing jumps via bang-bang control, see equation (3.11) on page 54), but α does so non-linearly. Thus, on free arcs (no constraints active), the optimal value of α must computed from the condition $H_\alpha = 0$, see equation (3.10) on page 54. However, for constrained arcs one should not use $H_\alpha = 0$, but an appropriate equality, corresponding to the constraint active. For example, if the normal acceleration is saturated at $L = L_{max}$, then optimal α is computed from equation (3.32) on page 61; similarly, if $L = L_{min}$ is active, optimal α is obtained from equation (3.36) on page 62.

Consider now the jumps in the angle of attack α in Figure 3.6, see also Figure 3.2. These jumps might be caused by multiplicity of solutions of α in the Hamiltonian, see Figure 3.22 on page 82. While there are three possible solutions of $H_\alpha = 0$, defining

optimal α for a free arc, only the solution closest to zero is physically meaningful, as the other two are approximately $-114.65°$ and $131.85°$, clearly infeasible values. Moreover, if the equation $H_\alpha = 0$ is solved at each time step using the Newton-Raphson method with an initial guess of α close to 0, the very steep slope will prevent the method from finding the outlying solutions. Thus, we may assume that jumps in the value of α, observed in Figure 3.6 for $V_{t_f} = 250$ m/s or $V_{t_f} = 270$ m/s, are not due to multiplicity of solutions of $H_\alpha = 0$. In other words, the fact that the Hamiltonian is not regular, does not affect the numerical solution for free arcs.

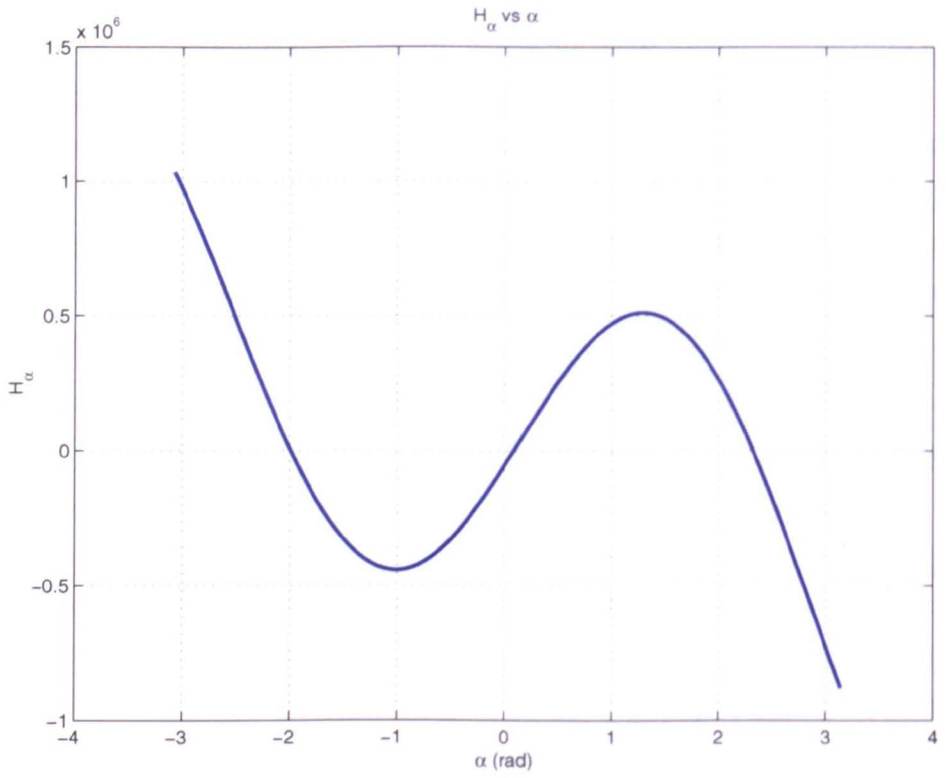


Figure 3.22: Hamiltonian (3.3) on page 53 is not regular, as the optimality condition $H_\alpha = 0$, equation (3.10) on page 54, has multiple solutions. The above plot of H_α versus α, revealing three possible values of optimal α, was obtained for $t = 10.45$ sec; similar curves were obtained for other times. Note a very steep slope in the vicinity of the middle solution (the one closest to zero): the slope is almost vertical, as the tangent is approximately 500,000.

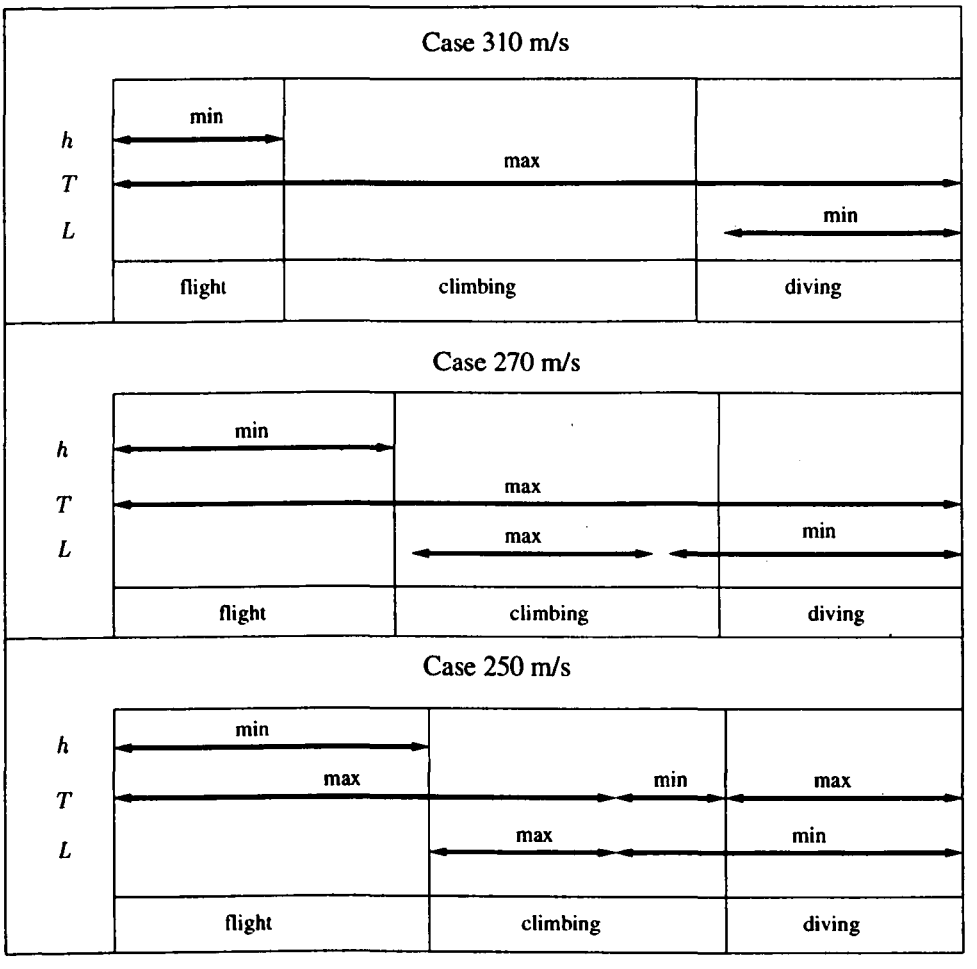


Figure 3.23: Switching structure of the minimum altitude for the terminal bunt manoeuvre.

In order to understand what does cause discontinuities in α in Figure 3.6 let us first investigate the case of final speed 310 m/s. Since the missile is launched at $h = h_{min}$, the first arc (level flight) is directly a constrained arc and therefore optimal α is computed not from $H_\alpha = 0$, but from equation (3.20a) on page 57. After 8.001305 sec, the first arc ends, as the missile starts climbing, thus beginning the second arc. Although the altitude increases, see Figure 3.3, the normal acceleration is not saturated during climbing, because the speed is not big enough, see Figure 3.4, to cause the normal acceleration to saturate, see Figure 3.8 and equations (1.6)–(1.8) on pp. 4–5. Thus, while the thrust is saturated on the maximum value, $T = T_{max}$, no other constraints are active, so that optimal α is computed from $H_\alpha = 0$, see equation (3.10) on page 54, where T_{max} should be substituted for T. From the point of view of calculating optimal α, this is a free arc (note that $T = T_{max}$ throughout). During climbing, the speed V decreases while the angle of attack α increases to facilitate climbing. While rapid climbing is necessary, the missile should also turn over to begin its dive as soon as possible, so that the excess of altitude is minimised. Therefore the angle of attack α starts decreasing, and reaches a negative value at time 25.21624 sec. During diving the speed V increases to satisfy the terminal final speed condition. This, in turn, causes the activation of minimum normal acceleration constraint L_{min} at time 34.91479 sec, marking the end of the free arc, which began at 8.001305 sec when the constrained first arc (level flight) ended, see Figure 3.3. The remainder of the trajectory is another constrained arc, with $L = L_{min}$, so that α is computed from equation (3.36) on page 62, see also Figure 3.23 on page 83.

In summary, for the case $V_{t_f} = 310$ m/s, the thrust T is saturated at T_{max} throughout the whole trajectory, and the trajectory starts with (i) a constrained arc ($h = h_{min}$), lasting from 0 to 8.001305 sec, followed by (ii) a free arc between 8.001305 and 34.91479 sec, and finishes with (iii) another constrained arc ($L = L_{min}$). As for optimal α, it is computed from equation (3.20a) on page 57 on (i), then from equation (3.10) on page 54 on (ii), and from equation (3.36) on page 62 for (iii). This results in α being a continuous function of time t, but with two points of non-smoothness, coinciding with the (i)→(ii) and (ii)→(iii) transitions, see Figure 3.6.

For the case of final speed 270 m/s the time of the level flight is longer than for the case of final speed 310 m/s. It means that the missile has a higher speed at the end of the first arc, and hence when it starts climbing with a rapidly increasing altitude the maximum normal acceleration constraint L_{max} is active, after a short free arc on $[t_1^-, t_1^+]$ (see Figure 3.8). At the start of climbing at $t_1^- = 19.26365$ sec, optimal α is computed from equation (3.10) until it hits the maximum normal acceleration at $t_1^+ = 19.58206$ sec. Then optimal α is computed from equation (3.32). Note that the activation of maximum normal acceleration is caused by the high speed of the missile (see equations (1.6)–(1.8)) while the thrust is $T = T_{max}$ throughout the whole trajectory. The angle of attack must decrease to facilitate the missile turnover at $t_2^- = 26.26862$ sec. While rapid decrease in α is needed, it soon causes the activation of minimum normal acceleration constraint L_{min} at $t_2^+ = 28.49747$ sec. It must be noted that the thrust is still on the maximum value. The angle of attack α is then computed from equation (3.36), see also Figure 3.23.

In summary, for the case $V_{t_f} = 270$ m/s, the thrust T is saturated at T_{max} throughout the whole trajectory. The trajectory starts with (i) a constrained arc ($h = h_{min}$), lasting from 0 to 19.26365 sec, followed by (ii) a short free arc between t_1^- and t_1^+, then by (iii) another constrained arc ($L = L_{max}$) from t_1^+ to t_2^-, then by (iv) another short free arc on $[t_2^-, t_2^+]$, and—finally—by (v) the last constrained arc ($L = L_{min}$). This results in α still being a continuous function of time t, but with steep slopes on the short intervals $[t_1^-, t_1^+]$ and $[t_2^-, t_2^+]$, and non-smoothness points at t_1^-, t_1^+, t_2^- and t_2^+. As for the case $V_{t_f} = 310$ m/s, non-smoothness points are due to joining of constrained and free arcs: (i)→(ii) at t_1^-, (ii)→(iii) at t_1^+, (iii)→(iv) at t_2^- and (iv)→(v) at t_2^+.

For the case of final speed 250 m/s the time of the level flight is longer than for the case of final speed 270 m/s. Now t_1^- and t_1^+ merge into one point t_1, because the missile has a very high speed at the end of level flight and the altitude increases rapidly, so that the constraint $L = L_{max}$ is activated immediately after the end of level flight, i.e. at $t_1 = 20.68605$ sec. Thus, optimal α is computed from equation (3.20a) to the left of t_1, and from equation (3.32) to the right, causing a jump in α; note that $T = T_{max}$ on both sides of t_1. The constrained arc ($L = L_{max}$) to the right of t_1 continues until speed V decreases enough for $L < L_{max}$ to become true, see Figure 3.4 and equations (1.6)–

(1.8) on page 4–5 (note that V dominates in equation (1.6)). When $L = L_{max}$ happens at time 20.68605 sec, two factors contribute to computation of the optimal solution at that point: 1) the need to decrease speed V further in order to meet the terminal condition $V_{t_f} = 250$ m/s, 2) occurrence of a free arc as on $[t_2^-, t_2^+]$ for $V_{t_f} = 270$ m/s. To achieve 1), optimal α should decrease rapidly towards negative values (to facilitate turnover), while satisfying $H_\alpha = 0$ according to 2). However, rapid decrease in α activates the $L = L_{min}$ constraint and the decrease is limited via equation (3.36). In the view of the arrested decrease in α, the only other way of facilitating the required turnover is by a more rapid decrease in speed V. This indeed is achieved by switching the thrust from T_{max} to T_{min} and holding it at T_{min} for a short period of time, see Figure 3.7 on page 47. Hence, the short free arc between t_2^- and t_2^+, seen for $V_{t_f} = 270$ m/s, collapses now to a point $t_2^- = t_2^+ = t_2 = 28.036943$ sec at which optimal α is computed from $H_\alpha = 0$, or equation (3.10). However, in that equation T changes from $T = T_{max}$ to the left of t_2 into $T = T_{min}$ to the right of t_2, thus effecting a jump in α at t_2. This discontinuity in α at t_2 immediately activates the $L = L_{min}$ constraint which remains active till t_f. Still before t_f, the thrust switches back to T_{max}, once its short-lasting lowering to T_{min} accomplished the necessary facilitation of the missile turnover, see also the switching function in Figure 3.24 on page 87 .

In summary, for the case $V_{t_f} = 250$ m/s, the switching structure of the case $V_{t_f} = 270$ m/s occurs in a limiting form. In other words, the free arcs $[t_1^-, t_1^+]$ and $[t_2^-, t_2^+]$ collapse each to a point: $t_1^- = t_1^+ = t_1$ and $t_2^- = t_2^+ = t_2$. In the latter case, a switch from T_{max} to T_{min} also happens to facilitate the missile turnover—the thrust was at T_{max} all time for $V_{t_f} = 270$ m/s. This results in α no longer being a continuous function of time t, but the causes of jumps at t_1 and t_2 are of different origin. In the case of t_1, the collapsed free arc does not show itself in the optimal solution: optimal α is computed from equation (3.20a) to the left of t_1 and from equation (3.32) to the right of t_1, while $T = T_{max}$ on both sides of t_1 (and at t_1). On the other hand, optimal α at t_2 is computed in a more subtle way. To the left of t_2 it is obtained from equation (3.32) and to the right of t_2 from equation (3.36), but its transition between these two values takes it through equation (3.10), where T jumps from T_{max} to T_{min}. Thus, the

jump in α at t_2 is caused by the jump in T, affecting it through the optimality equation for a (collapsed) free arc, $H_\alpha = 0$.

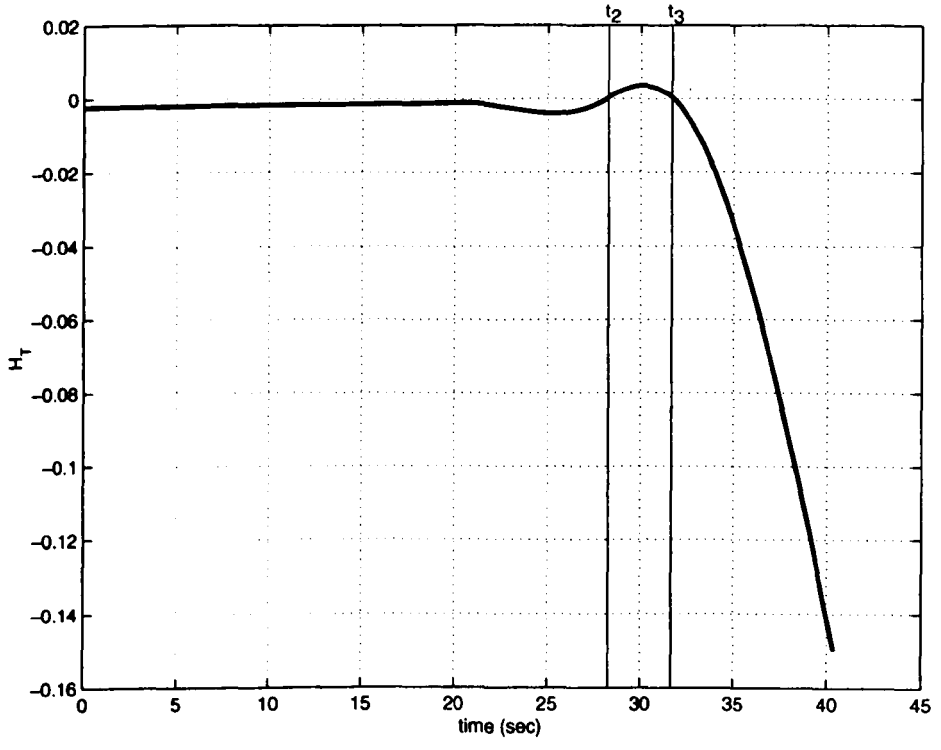


Figure 3.24: Switching function H_T versus time, see equation (3.11) on page 54

3.5.2 Comments on Implementation

DIRCOL worked very well for the whole trajectory, while NUDOCCCS had a convergence problem due to the pure state constraint activation. The direct method results were then used to feed a multiple shooting method as initial guesses for the state and co-state variables. It is well known that a multiple shooting requires a very good initial guess to start a Newton's iteration. Furthermore, for complex problems the jump and switching points must be guessed accurately. In this problem we fail to solve the terminal bunt problem with BNDSCO due to the pure state constraint. The initial guess from DIRCOL is based on Jacobson's formulation, therefore the pure state constraint is

adjoined directly. Since Jacobson's formulation may be extremely difficult to handle, Bryson's formulation is used to derive the necessary conditions. The jump condition of the DIRCOL solution is the same as the jump condition given by Jacobson's formulation. These can be seen on Figure 3.12 and equation (3.80).

Due to the difficulty to handle a pure state constraint, the suboptimal trajectory is given by splitting up the trajectory into two phases. The first phase employs the reduced equations on the state when the minimum altitude constraint is active and the rest of the manoeuvre is the second phase. At the second phase the minimum normal acceleration is active. Each phase is solved using BNDSCO, but the time when the first phase ends and the second phase begins is taken from the DIRCOL solution, thus the phases are joined in a suboptimal way. The suboptimal solutions have a slightly higher performance index compared to the DIRCOL solution.

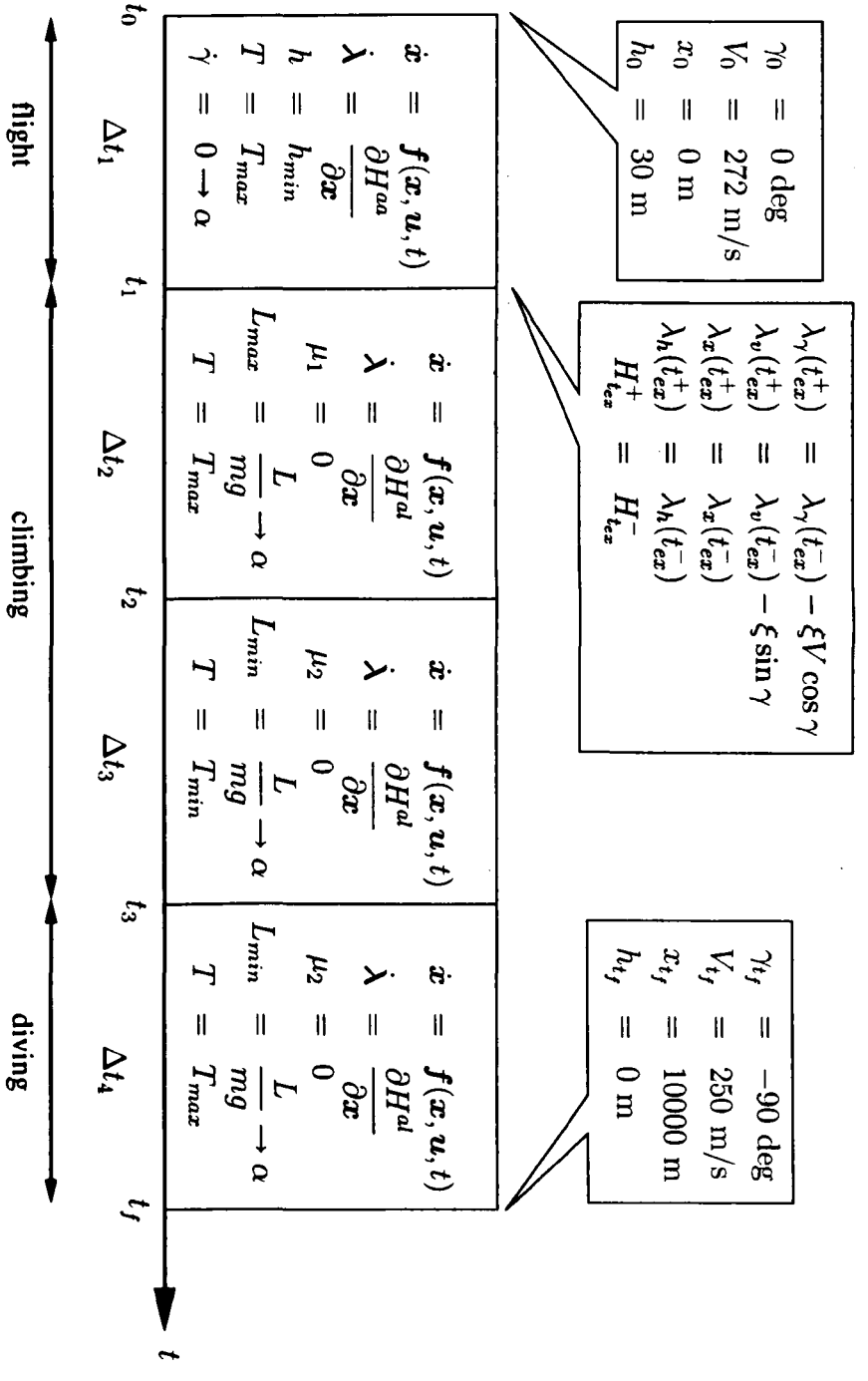


Figure 3.25: Schematic representation of the boundary value problem associated with the switching structure for the minimum altitude problem, case 250 m/s.

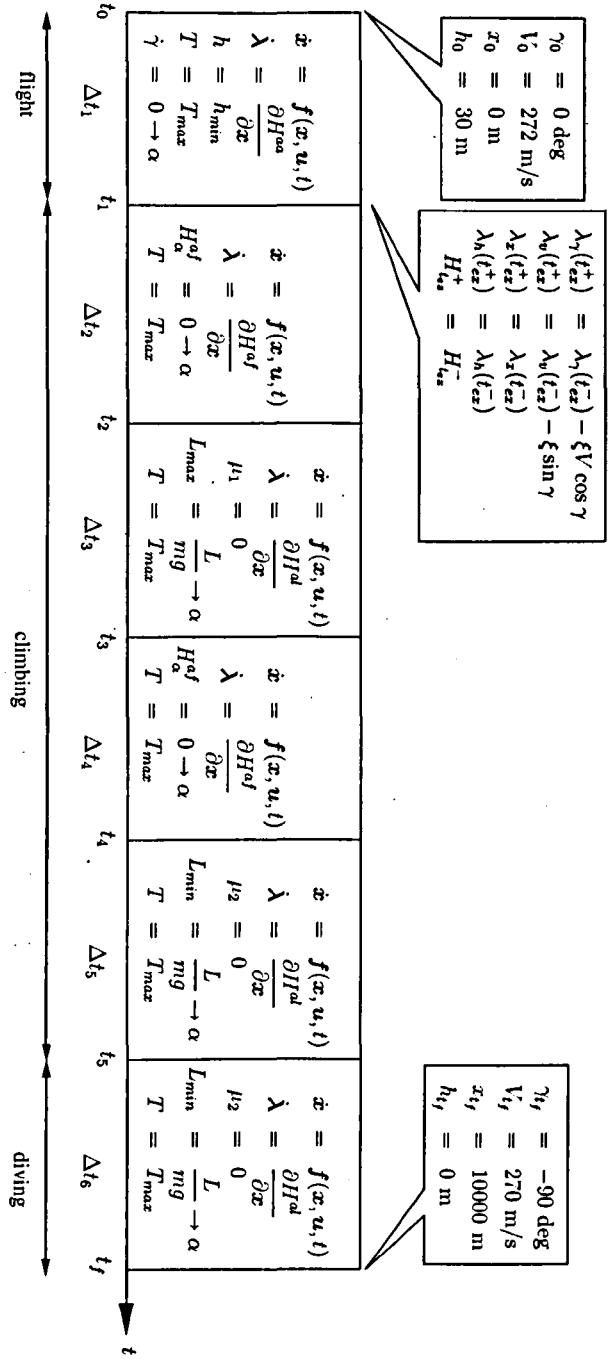


Figure 3.26: Schematic representation of the boundary value problem associated with the switching structure for the minimum altitude problem, case 270 m/s.

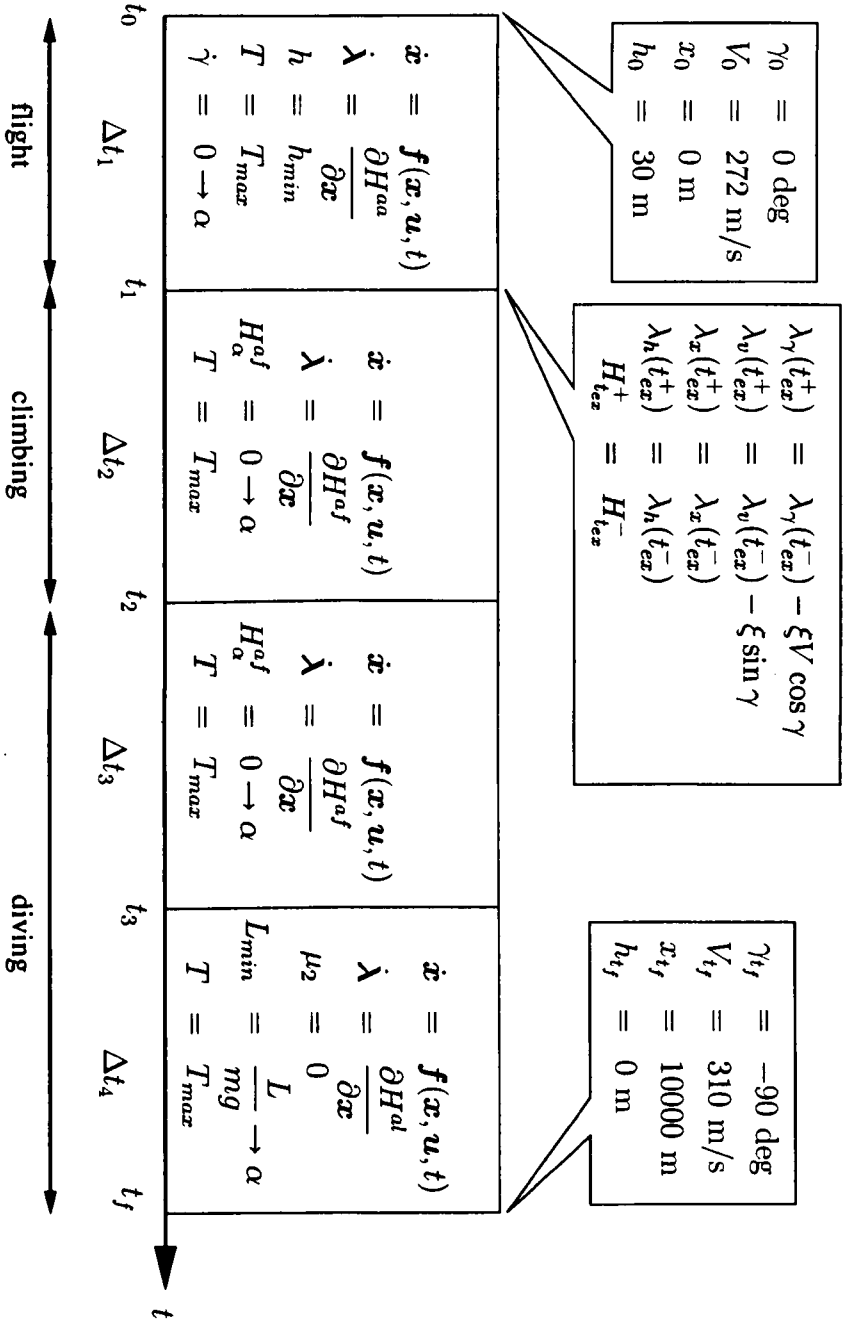


Figure 3.27: Schematic representation of the boundary value problem associated with the switching structure for the minimum altitude problem, case 310 m/s.

Chapter 4

Minimum-Time Formulation

This chapter focuses on the optimal trajectories of a generic cruise missile attacking a fixed target in minimum time [100]. The target must be struck from above, subject to missile dynamics and path constraints. The generic shape of the optimal trajectory is: level flight, climbing, dive; this combination of the three flight phases is called the bunt manoeuvre.

In chapter 3 we analysed and solved the terminal bunt manoeuvre of a generic cruise missile for which its exposure to anti-air defences was minimised. This resulted in a nonlinear optimal control problem for which time-integrated flight altitude was minimised. In this chapter we consider the same missile model, but we analyse and solve the terminal bunt manoeuvre for the fastest attack. This leads to a minimum-time optimal control problem which is solved in two complementary ways.

A direct approach based on a collocation method is used to reveal the structure of the optimal solution which is composed of several arcs, each of which can be identified by the corresponding manoeuvre executed and constraints active. The DIRCOL and NUDOCCCS packages used in the direct approach produce approximate solutions for both states and co-states.

The indirect approach is employed to derive optimality conditions based on Pontryagin's Minimum Principle. The resulting multi-point boundary value problem is then solved via multiple shooting with the BNDSCO package. The DIRCOL and NUDOCCCS results provide an initial guess for BNDSCO.

This chapter is organised as follows. In Section 4.1 the problem formulation is defined. Section 4.2 presents some computational results of the minimum-time problem using the direct collocation method package DIRCOL, followed by a qualitative analysis for the resulting optimal trajectory. Section 4.3 focuses on the mathematical analysis of the problem based on the qualitative analysis. Numerical results using BNDSCO package is presented in Section 4.4. BNDSCO, DIRCOL and NUDOCCCS results are compared in that section. Finally, summary and discussion is presented in Section 4.5.

4.1 Minimum Time Problem

In this section we consider the same missile model as defined in section 1.1 and the only difference is the objective function.

The problem is to find the trajectory of a generic cruise missile from the assigned initial state to a final state with the minimum-time along the trajectory. This problem can be formulated by introducing the performance criterion

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

4.2 Qualitative Analysis

This section gives a qualitative discussion of the optimal trajectory of a cruise missile performing a bunt manoeuvre.

The computational results are solved using a direct collocation method (DIRCOL) based on von Stryk [109]. Figures 4.1–4.6 shows the computational results using the following boundary conditions:

$$\begin{aligned}
\gamma_0 &= 0 \text{ deg}, & \gamma_{t_f} &= -90 \text{ deg} \\
V_0 &= 272 \text{ m/s}, & V_{t_f} &= 250, 270, 310 \text{ m/s} \\
x_0 &= 0 \text{ m}, & x_{t_f} &= 10000 \text{ m} \\
h_0 &= 30 \text{ m}, & h_{t_f} &= 0 \text{ m}.
\end{aligned}$$

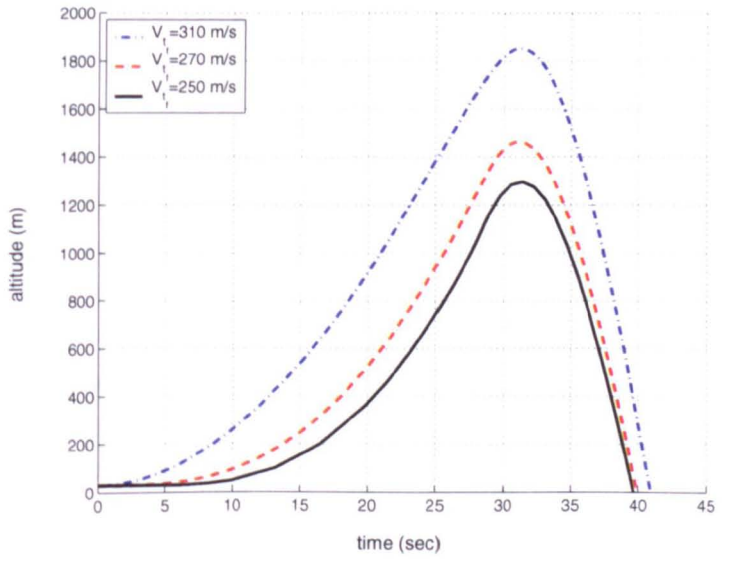


Figure 4.1: Altitude versus time histories for minimum time problem using DIRCOL for a varying final speed.

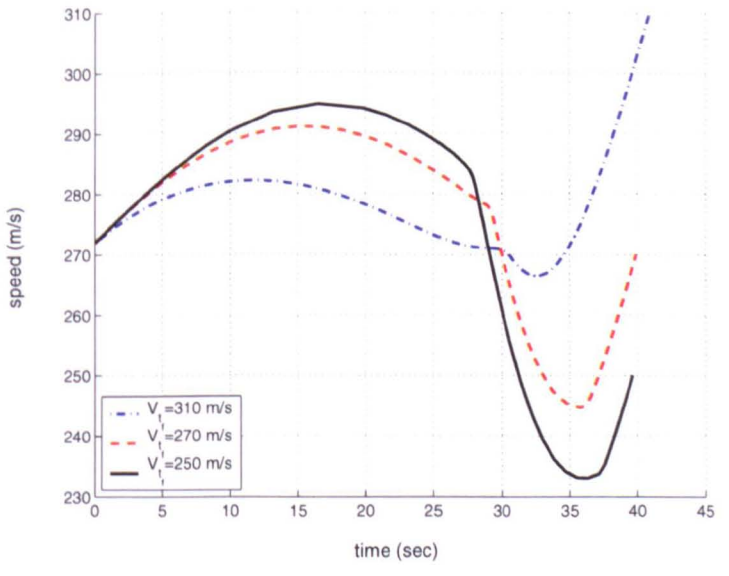


Figure 4.2: Speed versus time histories for minimum time problem using DIRCOL for a varying final speed.

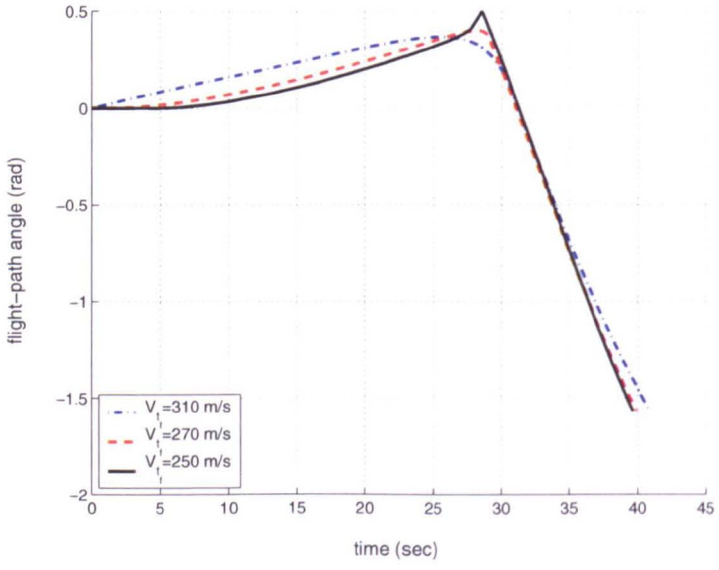


Figure 4.3: Flight-path angle versus time histories for minimum time problem using DIRCOL for a varying final speed.

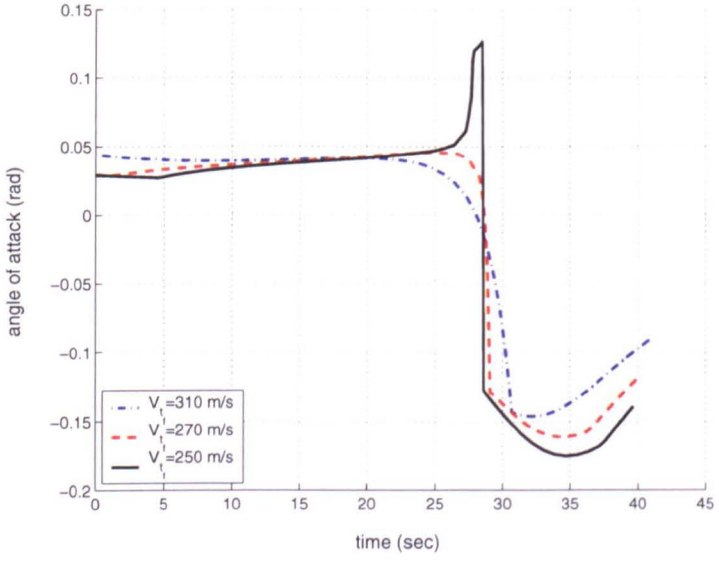


Figure 4.4: Angle of attack versus time histories for minimum time problem using DIRCOL for a varying final speed.

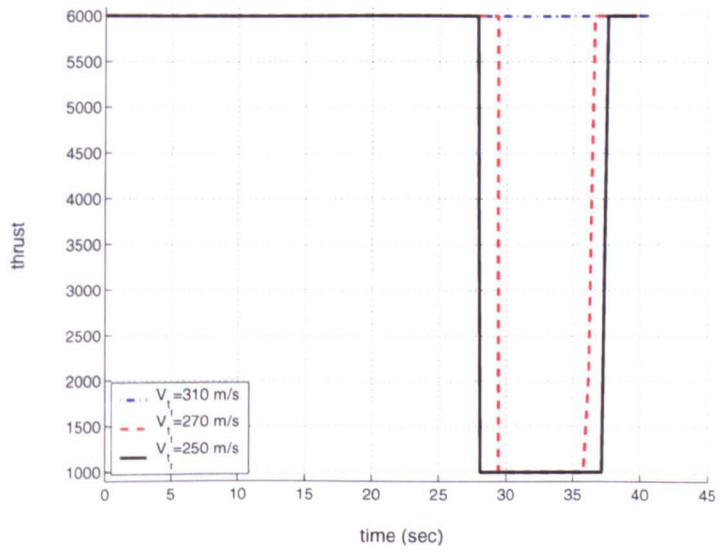


Figure 4.5: Thrust versus time histories for minimum time problem using DIRCOL for a varying final speed.

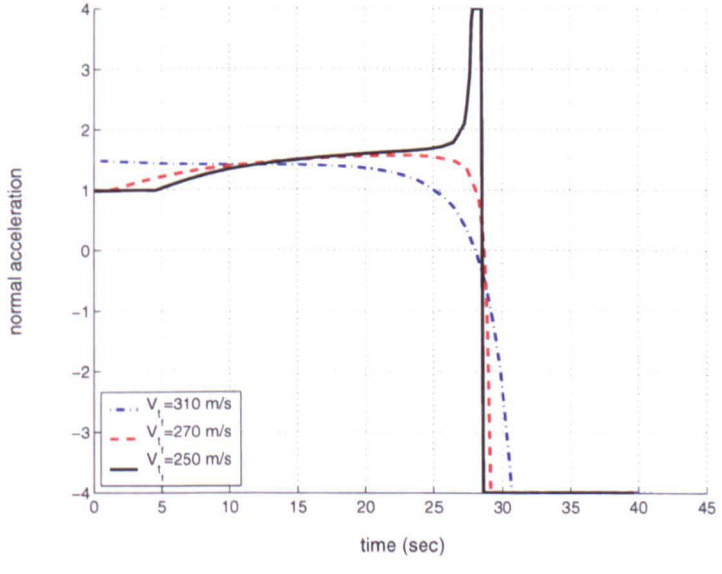


Figure 4.6: Normal acceleration versus time histories for minimum time problem using DIRCOL for a varying final speed.

Based on Figures 4.1–4.6, an attempt is made to identify characteristic arcs of the trajectory, classify them according to the constraints active on them, and suggest physical/mathematical explanations for the observed behaviour. In this analysis the missile is assumed to be launched horizontally from the minimum altitude constraint ($h_0 = 30$ m). The trajectory is split into three subintervals: level flight, climbing and diving.

4.2.1 First Arc (Flight): Minimum Altitude Flight

The missile flies at the minimum altitude with the thrust on the maximum value. Thus the thrust and altitude constraints are active directly at the start of the manoeuvre. In this case the altitude h of the missile remains constant on the minimum altitude (h_{min}) until the missile must start climbing. The cruise-flight time depends on the final speed V_{t_f} (see Figure 4.1).

Equation (1.3d) equals zero during this flight because the altitude remains constant. It means the flight path angle γ equals zero because the velocity V is never equal to zero during flight. Obviously, $\gamma(t) = 0$ causes the derivative of the flight path angle $\dot{\gamma}$ to be equal to zero. The dynamics equation (1.3) is therefore reduced as follows:

$$\dot{\gamma} = \frac{T-D}{m}\sin\alpha + \frac{L}{m}\cos\alpha - g = 0 \tag{4.2a}$$

$$\dot{V} = \frac{T-D}{m}\cos\alpha - \frac{L}{m}\sin\alpha \tag{4.2b}$$

$$\dot{x} = V \tag{4.2c}$$

$$\dot{h} = 0 \tag{4.2d}$$

We now consider the consequences of the right-hand side of equation (4.2a) being zero. This condition means that the normal acceleration L/m remains almost constant, because the angle of attack α is very small. The first term on the right hand side of equation (4.2a) is small, because $\sin\alpha \approx \alpha \approx 0$ and we are left with $L/m \approx g$ due to $\cos\alpha \approx 1$.

In this arc, the speed increases, because for small α

$$\dot{V} \approx \frac{T-D}{m} > 0, \quad \text{as } T > D.$$

This in turn means that the angle of attack α slowly decreases in accordance with equation (1.6) and in order to maintain L/m approximately be equal to g.

4.2.2 Second Arc (Climbing)

The missile climbs eventually in order to achieve the final condition of the flight-path γ_{t_f}. Figure 4.1 shows that the missile climbs directly at the beginning of launch for the case of final-speed $V_{t_f} = 310$. The thrust constraint is the only active constraints at the beginning of climbing. Although the missile needs the full power to reach the target as soon as possible, the missile must satisfy the final speed at the boundary conditions. Therefore the thrust switches to minimum value for the case of final-speed 250 m/s and 270 m/s when the missile nearly turns over.

At the end of climbing the angle of attack is rapidly decreasing, while for the case of final-speed 250 m/s the angle of attack is increasing and then decreasing rapidly. This makes the maximum normal acceleration constraint active for the case of final-speed 250 m/s. The minimum normal acceleration is active at the end of climbing because of rapidly decreasing angle of attack.

4.2.3 Third Arc (Diving)

The missile dives with the minimum thrust at the beginning of diving for the cases of final-speed 250 m/s and 270 m/s. Furthermore the missile must hit the target a certain value speed at the end of manoeuvre. In addition, the speed during the turnover is lower than the final speed. Therefore the speed must increase and hence the thrust switches back to the maximum value. It means the thrust will facilitate the missile's arrival on the target as soon as possible.

In this case the normal acceleration is still saturated on the minimum value. Obviously, the altitude goes down to reach the target ($\gamma < 0 \rightarrow \dot{h} < 0$, see equation (1.3d)), while the speed goes up to satisfy the terminal speed condition V_{t_f}. Finally, the missile satisfies the terminal condition of the manoeuvre approximately t_f after firing.

Table 4.1: Performance index for the minimum time problem using DIRCOL

Final speed V_{t_f} (m/s)	J (sec)
250	39.64711
270	39.90681
310	40.90780

4.3 Mathematical Analysis

This section describes mathematical analysis of the minimum-time terminal bunt problem by considering qualitative analysis results from Section 4.2. The basic premise of the analysis is to exploit the clearly identifiable arcs of the trajectory and obtain the full solution by piecing them together. The theoretical basis of this approach is Bellman's Optimality Principle [73, page118]: *Any piece of an optimal trajectory is optimal*, a result following easily from a proof by contradiction. While the analysis of trajectory is thus considerably simplified, establishing the length (duration) of each arc still requires formulation and solution of consistent Boundary Value Problems (BVPs).

In section 4.3.1 the problem with only thrust constraint is considered. Section 4.3.2 explains the derivations relevant to optimal control problems with path constraints. Section 4.3.3 presents the first arc of the bunt manoeuvre, which is constrained on the altitude. The climbing manoeuvre is described in section 4.3.4. The last part of the trajectory, the diving manoeuvre, is discussed in section 4.3.5.

4.3.1 Constrained on the Thrust Only

First, we investigate the problem when the initial and final conditions (1.9) are active and the control is constrained on thrust T only (1.12). Necessary conditions for optimality can be determined by applying Pontryagin's Minimum Principle [84]. For this

purpose, we first consider the Hamiltonian for the unconstrained case:

$$\begin{aligned} H^{mtf} &= 1+\frac{\lambda_\gamma}{V}\left[\frac{T-D}{m}\sin\alpha+\frac{L}{m}\cos\alpha-g\cos\gamma\right] \\ &\quad +\lambda_V\left[\frac{T-D}{m}\cos\alpha-\frac{L}{m}\sin\alpha-g\sin\gamma\right] \\ &\quad +\lambda_x V\cos\gamma+\lambda_h V\sin\gamma, \end{aligned} \tag{4.3}$$

where the co-states $\boldsymbol{\lambda}=(\lambda_\gamma,\lambda_V,\lambda_x,\lambda_h)$ have been adjoined to the dynamics system of equation (1.3). The co-states are determined by

$$\dot{\boldsymbol{\lambda}}=-\frac{\partial H^{mtf}}{\partial \boldsymbol{x}}. \tag{4.4}$$

The component of co-state vector $\boldsymbol{\lambda}$ satisfying the preceding equations are:

$$\dot{\lambda}_\gamma = -\left\{\frac{\lambda_\gamma}{V}g\sin\gamma-\lambda_V g\cos\gamma-\lambda_x V\sin\gamma+\lambda_h V\cos\gamma\right\} \tag{4.5}$$

$$\begin{aligned} \dot{\lambda}_V &= -\left\{\lambda_\gamma\left[-\frac{T\sin\alpha}{V^2 m}-\frac{C_d\rho S_{ref}\sin\alpha}{2m}+\frac{C_l\rho S_{ref}\cos\alpha}{2m}+\frac{g}{V^2}\cos\gamma\right]\right. \\ &\quad \left.-\frac{\lambda_V\rho V S_{ref}}{m}\left[C_d\cos\alpha+C_l\sin\alpha\right]+\lambda_x\cos\gamma+\lambda_h\sin\gamma\right\} \end{aligned} \tag{4.6}$$

$$\dot{\lambda}_x = 0 \tag{4.7}$$

$$\begin{aligned} \dot{\lambda}_h &= -\lambda_\gamma\left[-\frac{C_d V S_{ref}\sin\alpha\rho_h}{2m}+\frac{C_l V S_{ref}\cos\alpha\rho_h}{2m}\right] \\ &\quad +\lambda_V\left[\frac{C_d V^2 S_{ref}\cos\alpha\rho_h}{2m}+\frac{C_l V^2 S_{ref}\sin\alpha\rho_h}{2m}\right] \end{aligned} \tag{4.8}$$

where:

$$\rho_h = 2C_1 h + C_2$$

The optimal values of the control variables are generally to be determined from the Pontryagin's Minimum Principle. A necessary condition for optimal control is the Minimum Principle

$$\min_{\boldsymbol{u}} H^{mtf}, \tag{4.9}$$

i.e. the Hamiltonian must be minimised with respect to the vector of controls $\boldsymbol{u}$. Applying (4.9) to (4.3) we obtain

$$\begin{aligned} H_{\alpha}^{mtf} &= (T - D + L_{\alpha})\Big[\frac{\lambda_{\gamma}}{Vm}\cos\alpha - \frac{\lambda_V}{m}\sin\alpha\Big] \\ &\quad -(D_{\alpha} + L)\Big[\frac{\lambda_{\gamma}}{Vm}\sin\alpha + \frac{\lambda_V}{m}\cos\alpha\Big] = 0 \end{aligned} \tag{4.10}$$

Since the control T appears linearly in the Hamiltonian, the condition $H_T^{mtf} = 0$ does not determine optimal thrust. Since T is bounded, the following provides the minimum of the Hamiltonian:

$$T = \begin{cases} T_{max} & \text{if } H_T^{mtf} < 0, \\ T_{sing} & \text{if } H_T^{mtf} = 0, \\ T_{min} & \text{if } H_T^{mtf} > 0. \end{cases}$$

with

$$H_T^{mtf} = \lambda_{\gamma}\frac{\sin\alpha}{Vm} + \lambda_V\frac{\cos\alpha}{m} \quad \text{(switching function)} \tag{4.11}$$

- **Case when T on the boundary ($T = T_{max}$ or $T = T_{min}$)**
  In this case α can be determined from:

$$\begin{aligned} H_{\alpha}^{mtf} &= \frac{T - D + L_{\alpha}}{m}\Big[\frac{\lambda_{\gamma}}{V}\cos\alpha - \lambda_V\sin\alpha\Big] \\ &\quad -\frac{D_{\alpha} + L}{m}\Big[\frac{\lambda_{\gamma}}{V}\sin\alpha + \lambda_V\cos\alpha\Big] = 0 \end{aligned} \tag{4.12}$$

  The value of α cannot be derived in closed form from (4.12), and must be obtained numerically.

- **Case when $T = T_{sing}$ (singular control)**
  When the switching function H_T^{mtf} becomes zero in an interval $(t_1, t_2) \subset (t_0, t_f)$, the control corresponding to the magnitude of the thrust T is singular. In these circumstances, there are finite control variations of T which do not affect the value of the Hamiltonian.

  From Bryson & Ho [23, page 246], the singular arcs occur when:

$$H_{\boldsymbol{u}}^{mtf} = 0 \quad \text{and} \quad \det H_{\boldsymbol{uu}}^{mtf} = 0 \tag{4.13}$$

Substituting (4.3) into (4.13) with component $\boldsymbol{u} = (T, \alpha)$ yields

$$H_T^{mtf} = \lambda_\gamma \frac{\sin\alpha}{Vm} + \lambda_V \frac{\cos\alpha}{m} = 0 \tag{4.14}$$

$$\begin{aligned} H_\alpha^{mtf} &= (T - D + D_\alpha)\Big[\frac{\lambda_\gamma}{Vm}\cos\alpha - \frac{\lambda_V}{m}\sin\alpha\Big] \\ &\quad -(D_\alpha + L)\Big[\frac{\lambda_\gamma}{Vm}\sin\alpha + \frac{\lambda_V}{m}\cos\alpha\Big] = 0 \end{aligned} \tag{4.15}$$

$$\det H_{\boldsymbol{uu}}^{mtf} = 0 \iff \lambda_\gamma \frac{\cos\alpha}{Vm} - \lambda_V \frac{\sin\alpha}{m} = 0 \tag{4.16}$$

Conditions (4.14)–(4.16) cannot be satisfied simultaneously, so we conclude that there are no singular arcs. However, jump discontinuities in the control T may appear if, at a time t, the switching function (4.11) changes sign.

The Hamiltonian is not an explicit function of time, so H^{mtf} is constant along the optimal trajectory and must be equal zero because of minimum-time problem.

4.3.2 Optimal Control with Path Constraints

In section 4.3.1 we derived necessary conditions for optimality by considering only the initial and terminal conditions. In this section the level of complexity is increased by considering some additional constraints as defined in section 1.1.

The first state path constraint (1.10) can be split as $V_{min} - V \leqslant 0$ and $V - V_{max} \leqslant 0$. Both of them are of order 1, because $\dot{V}$ explicitly depends on the controls, see [23, pp. 99–100]. Since the speed constraint is not active during the manoeuvre in this case therefore it will not be taken into account in the Hamiltonian (see figure 4.2 page 95). The second path constraint (1.11) is of order 2 and the mixed state-control constraint (1.13) is split as $L_{min} - \frac{L}{mg} \leqslant 0$ and $\frac{L}{mg} - L_{max} \leqslant 0$ and $\frac{L}{mg}$ depends on the control explicitly. The Hamiltonian can be defined as follows:

$$H^{mtc} = H^{mtf} + \mu_1\left\{-\frac{L}{mg} + L_{min}\right\} + \mu_2\left\{\frac{L}{mg} - L_{max}\right\} + \mu_3(\ddot{h})$$

The differential equations for co-state vector $\boldsymbol{\lambda} = (\lambda_\gamma, \lambda_V, \lambda_x, \lambda_h)$ can be written as

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial H^{mtc}}{\partial \boldsymbol{x}}. \tag{4.17}$$

Since these equations are rather lengthy, they are omitted here. For the Lagrange multipliers $\mu_i = 1, \ldots, 5$, there must hold

$$\mu_i \begin{cases} = 0, & \text{if the associated constraint is not active;} \\ \geqslant 0, & \text{if the associated constraint is active.} \end{cases}$$

From section 4.2 we know that the state path constraint (1.10) is not active during the entire manoeuvre. Therefore in the following section we consider only altitude and normal acceleration constraints.

4.3.3 First Arc: Minimum Altitude Flight

In this analysis we consider only the state path constraint $h_{min} \leqslant h$ and thrust control constraint (T is on the maximum value). In this case we assume that the missile starts at the initial altitude $h = h_{min}$ and $T = T_{max}$. Therefore the constraints are active at the start of the manoeuvre directly. The constraint $h_{min} \leqslant h$ has no explicit dependence on the control variables, therefore we must take the time derivative on the constraint until, finally, explicit dependence on the control does occur. Consider the following equations:

$$h - h_{min} = 0 \tag{4.18a}$$

$$\dot{h} = V \sin\gamma = 0 \Rightarrow \gamma(t) = 0 \quad \text{for} \quad t \in [t_0, t_1] \tag{4.18b}$$

$$\ddot{h} = \dot{V} \sin\gamma + V\dot{\gamma}\cos\gamma = 0 \Rightarrow \dot{\gamma}(t) = 0 \quad \text{for} \quad t \in [t_0, t_1] \tag{4.18c}$$

The controls appear explicitly after differentiating the constraint $h_{min} \leqslant h$ twice, therefore the order of the constraint is 2. Substituting equation (4.18) in the equation of motion (1.3) we obtain the following reduced equations:

$$\dot{\gamma} = \frac{T - D}{m} \sin\alpha + \frac{L}{m} \cos\alpha - g = 0 \tag{4.19a}$$

$$\dot{V} = \frac{T - D}{m} \cos\alpha - \frac{L}{m} \sin\alpha \tag{4.19b}$$

$$\dot{x} = V \tag{4.19c}$$

$$\dot{h} = 0 \tag{4.19d}$$

The angle of attack α can be obtained numerically from equation (4.19a). Then substituting α to equation (4.19b) and (4.19c), these equations can be solved as an initial value problem (IVP). Thus we can find the first arc easily, but we do not know how long it will last. For this purpose we should formulate the appropriate boundary value problem (BVP) which involves finding co-state variables by defining the Hamiltonian as follows:

$$H^{mta} = H^{mtf} + \mu_3 \left\{ \dot{V} \sin\gamma + V\dot{\gamma}\cos\gamma \right\} \tag{4.20}$$

The appropriate co-state equations must be derived. The necessary conditions for optimality is given by

$$\begin{aligned} H_\alpha^{mta} &= (T - D + L_\alpha)\Big[\Big(\frac{\lambda_\gamma}{V} + \mu_3 V\Big)\frac{\cos\alpha}{m} - \frac{\lambda_V}{m}\sin\alpha\Big] \\ &\quad -(D_\alpha + L)\Big[\Big(\frac{\lambda_\gamma}{V} + \mu_3 V\Big)\frac{\sin\alpha}{m} + \frac{\lambda_V}{m}\cos\alpha\Big] = 0 \end{aligned} \tag{4.21}$$

The angle of attack α can be obtained from equation (4.19a) while the Lagrange multiplier μ_3 can be derived explicitly from equation (4.21) and substituted into state and co-state equations. Since we know the flight path angle and the altitude during this manoeuvre, the number of differential equations reduces to six.

4.3.4 Second Arc: Climbing

In this analysis we consider thrust and normal acceleration constraints. From the qualitative analysis, we know that the thrust control switches to the minimum value during climbing for the final speed 250 m/s and 270 m/s cases, therefore the switching function must change sign from negative to positive. In this section we do not derive optimality conditions for the "free" arc cases, because we can refer to section 4.3.1 for it.

Consider mixed state-control inequality constraints, as mentioned in equation (1.13):

$$L_{min} \leqslant \frac{L}{mg} \leqslant L_{max}$$

and L explicitly depends on the control α. The inclusion of the mixed constraints above leads to the augmented Hamiltonian:

$$H^{mtl} = H^{mtf} + \mu_1\Big(-\frac{L}{mg} + L_{min}\Big) + \mu_2\Big(\frac{L}{mg} - L_{max}\Big) \tag{4.23}$$

The right-hand side of the differential equations for the co-state equations are to be modified along subarcs of this second arc. Additionally, we have a necessary sign condition for the Lagrange parameter μ_i,

$$\mu_1 \begin{cases} = 0 & \text{on unconstrained subarcs} \\ = \frac{H_\alpha^{mtf}}{L_\alpha} mg & \text{on constrained subarcs} \end{cases}$$

and

$$\mu_2 \begin{cases} = 0 & \text{on unconstrained subarcs} \\ = -\frac{H_\alpha^{mtf}}{L_\alpha} mg & \text{on constrained subarcs} \end{cases}$$

where:

- optimality condition when normal acceleration is on the maximum value

$$\begin{aligned} H_\alpha^{mtl} &= (T - D + L_\alpha)\left[\frac{\lambda_\gamma}{Vm}\cos\alpha - \frac{\lambda_V}{m}\sin\alpha\right] \\ &\quad -(D_\alpha + L)\left[\frac{\lambda_\gamma}{Vm}\sin\alpha + \frac{\lambda_V}{m}\cos\alpha\right] + \mu_2\left(\frac{L_\alpha}{mg}\right) = 0 \end{aligned} \quad (4.24)$$

- optimality condition when normal acceleration is on the minimum value

$$\begin{aligned} H_\alpha^{mtl} &= (T - D + L_\alpha)\left[\frac{\lambda_\gamma}{Vm}\cos\alpha - \frac{\lambda_V}{m}\sin\alpha\right] \\ &\quad -(D_\alpha + L)\left[\frac{\lambda_\gamma}{Vm}\sin\alpha + \frac{\lambda_V}{m}\cos\alpha\right] - \mu_1\left(\frac{L_\alpha}{mg}\right) = 0 \end{aligned} \quad (4.25)$$

When the maximum normal acceleration L_{max} constraint is active (case 250 m/s), the angle of attack can be determined from (1.6) as

$$\alpha = \frac{2mgL_{max} - B_2\rho V^2 S_{ref}}{B_1\rho V^2 S_{ref}}. \quad (4.26)$$

Equation (4.26) is valid until equation (4.11) changes the sign to positive.

Below we summarise the results for the case when the normal acceleration is saturated on the maximum value (L_{max}).

- State equations:

$$\begin{aligned}
\dot{\gamma} &= \left\{\frac{T-D}{m}\sin\alpha + \frac{L}{m}\cos\alpha - g\cos\gamma\right\}\frac{1}{V} \\
\dot{V} &= \frac{T-D}{m}\cos\alpha - \frac{L}{m}\sin\alpha - g\sin\gamma \\
\dot{x} &= V\cos\gamma \\
\dot{h} &= V\sin\gamma
\end{aligned}$$

- Co-state equations:

$$\begin{aligned}
\dot{\lambda}_\gamma &= -\left\{\frac{\lambda_\gamma}{V}g\sin\gamma - \lambda_V g\cos\gamma - \lambda_x V\sin\gamma + \lambda_h V\cos\gamma\right\} \\
\dot{\lambda}_V &= -\left\{\lambda_\gamma\left[-\frac{T\sin\alpha}{V^2 m} - \frac{C_d\rho S_{ref}\sin\alpha}{2m} + \frac{C_l\rho S_{ref}\cos\alpha}{2m} + \frac{g}{V^2}\cos\gamma\right]\right. \\
&\quad -\frac{\lambda_V\rho V S_{ref}}{m}\left[C_d\cos\alpha + C_l\sin\alpha\right] \\
&\quad \left.+\lambda_x\cos\gamma + \lambda_h\sin\gamma + \mu_2\left[\frac{L_V}{mg}\right]\right\} \\
\dot{\lambda}_x &= 0 \\
\dot{\lambda}_h &= -\left\{\frac{\lambda_\gamma V S_{ref}\rho_h}{2m}\left[C_d\sin\alpha - C_l\cos\alpha\right]\right. \\
&\quad \left.-\frac{\lambda_V V^2 S_{ref}\rho_h}{2m}\left[C_d\cos\alpha + C_l\sin\alpha\right] + \mu_2\left[\frac{L_h}{mg}\right]\right\}
\end{aligned}$$

- Optimality condition

$$\begin{aligned}
H_\alpha^{mtl} &= (T - D + L_\alpha)\left[\frac{\lambda_\gamma}{Vm}\cos\alpha - \frac{\lambda_V}{m}\sin\alpha\right] \\
&\quad -(D_\alpha + L)\left[\frac{\lambda_\gamma}{Vm}\sin\alpha + \frac{\lambda_V}{m}\cos\alpha\right] + \mu_2\left[\frac{L_\alpha}{mg}\right] = 0 \quad (4.28)
\end{aligned}$$

where

$$\alpha = \frac{2mgL_{max} - B_2\rho V^2 S_{ref}}{B_1\rho V^2 S_{ref}} \quad (4.29)$$

and the thrust switches to the minimum value when H_T^{mtf} changes sign from negative to positive.

4.3.5 Third Arc: Diving

In this analysis we consider only the normal acceleration constraint. At the start of diving the thrust is on the minimum value and then switches back to the maximum value for the final speeds of 250 m/s and 270 m/s cases. In addition, normal acceleration is saturated on the minimum value. The angle of attack can be determined from (1.6) as follows:

$$\alpha = \frac{2mgL_{min} - B_2\rho V^2 S_{ref}}{B_1\rho V^2 S_{ref}}. \tag{4.30}$$

The Hamiltonian and co-state equations are nearly the same as in the previous section, therefore the derivation is omitted here. The equations can be summarised as follows:

- State equations:

$$\begin{aligned}
\dot{\gamma} &= \left\{\frac{T-D}{m}\sin\alpha + \frac{L}{m}\cos\alpha - g\cos\gamma\right\}\frac{1}{V} \\
\dot{V} &= \frac{T-L}{m}\cos\alpha - \frac{L}{m}\sin\alpha - g\sin\gamma \\
\dot{x} &= V\cos\gamma \\
\dot{h} &= V\sin\gamma
\end{aligned}$$

- Co-state equations:

$$\begin{aligned}
\dot{\lambda}_\gamma &= -\left\{\frac{\lambda_\gamma}{V}g\sin\gamma - \lambda_V g\cos\gamma - \lambda_x V\sin\gamma + \lambda_h V\cos\gamma\right\} \\
\dot{\lambda}_V &= -\left\{\lambda_\gamma\left[-\frac{T\sin\alpha}{V^2 m} - \frac{C_d\rho S_{ref}\sin\alpha}{2m} + \frac{C_l\rho S_{ref}\cos\alpha}{2m} + \frac{g}{V^2}\cos\gamma\right]\right. \\
&\quad \left. -\frac{\lambda_V\rho V S_{ref}}{m}\left[C_d\cos\alpha + C_l\sin\alpha\right] + \lambda_x\cos\gamma + \lambda_h\sin\gamma - \mu_1\left[\frac{L_V}{mg}\right]\right\} \\
\dot{\lambda}_x &= 0 \\
\dot{\lambda}_h &= -\left\{\frac{\lambda_\gamma V S_{ref}\rho_h}{2m}\left[C_d\sin\alpha - C_l\cos\alpha\right]\right. \\
&\quad \left. -\frac{\lambda_V V^2 S_{ref}\rho_h}{2m}\left[C_d\cos\alpha + C_l\sin\alpha\right] - \mu_1\left[\frac{L_h}{mg}\right]\right\}
\end{aligned}$$

- Optimality condition

$$H_{\alpha}^{mtl} = (T - D + L_{\alpha})\left[\frac{\lambda_{\gamma}}{Vm}\cos\alpha - \frac{\lambda_V}{m}\sin\alpha\right] - (D_{\alpha} + L)\left[\frac{\lambda_{\gamma}}{Vm}\sin\alpha + \frac{\lambda_V}{m}\cos\alpha\right] - \mu_1\left[\frac{L_{\alpha}}{mg}\right] = 0 \quad (4.32)$$

where

$$\alpha = \frac{2mgL_{min} - B_2\rho V^2 S_{ref}}{B_1\rho V^2 S_{ref}} \quad (4.33)$$

4.4 Indirect Method Solutions

The multi-point boundary value problem is solved by means of the multiple shooting code BNDSCO [76] and compared with the DIRCOL and NUDOCCCS results. The direct method results based on DIRCOL and NUDOCCCS packages give a good approximation for the state and co-state variables although the problem involves an active mixed state-control inequality constraint. Figures 4.7–4.15 show the computational results for BNDSCO, DIRCOL and NUDOCCCS results for the following boundary conditions.

The initial conditions are:

$$\begin{aligned} \gamma_0 &= 0 \text{ deg}, \\ V_0 &= 272 \text{ m/s}, \\ x_0 &= 0 \text{ m}, \\ h_0 &= 30 \text{ m}. \end{aligned}$$

The final conditions are:

$$\begin{aligned} \gamma_{t_f} &= -90 \text{ deg}, \\ V_{t_f} &= 310 \text{ m/s}, \\ x_{t_f} &= 10000 \text{ m}, \\ h_{t_f} &= 0 \text{ m}. \end{aligned}$$

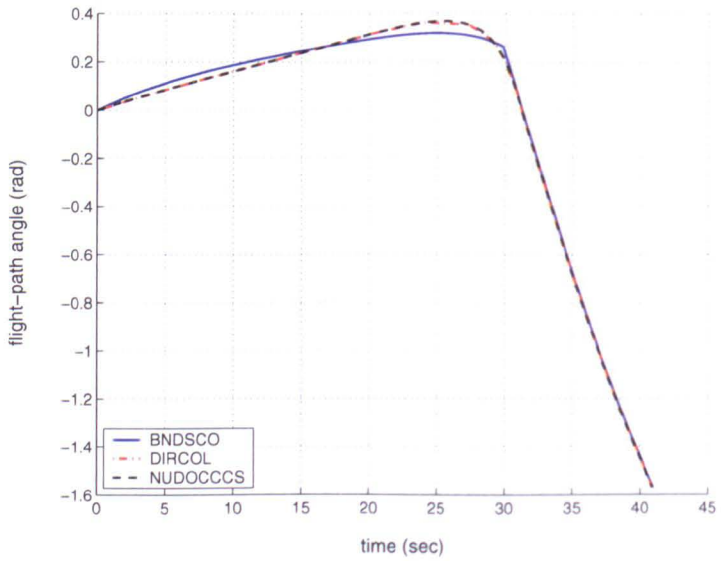


Figure 4.7: Comparison of BNDSCO, DIRCOL and NUDOCCCS results of the flight-path angle versus time, constrained minimum time problem.

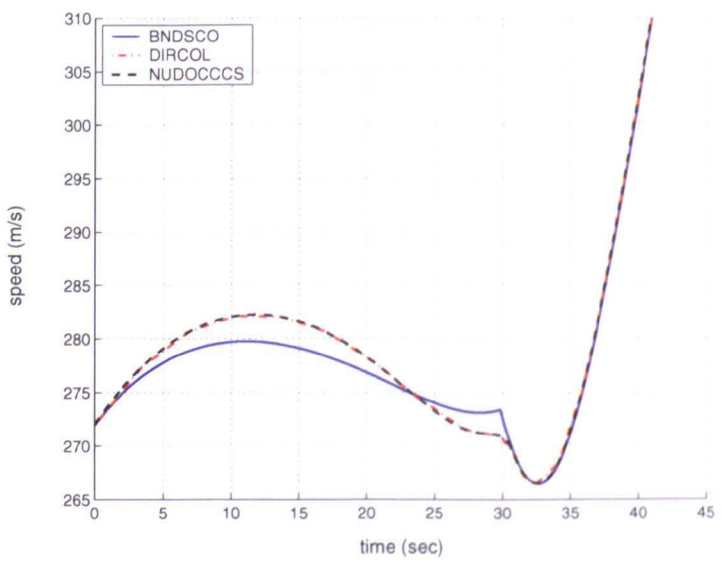


Figure 4.8: Comparison of BNDSCO, DIRCOL and NUDOCCCS results of the velocity versus time, constrained minimum time problem.

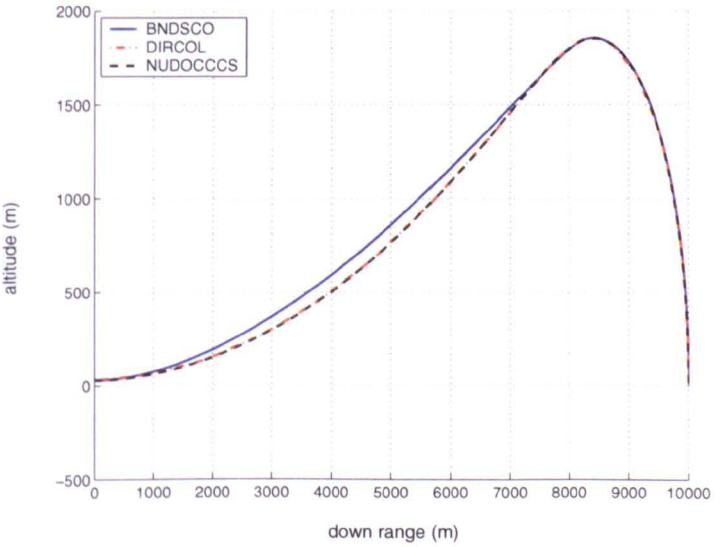


Figure 4.9: Comparison of BNDSCO, DIRCOL and NUDOCCCS results of the altitude versus down-range, constrained minimum time problem.

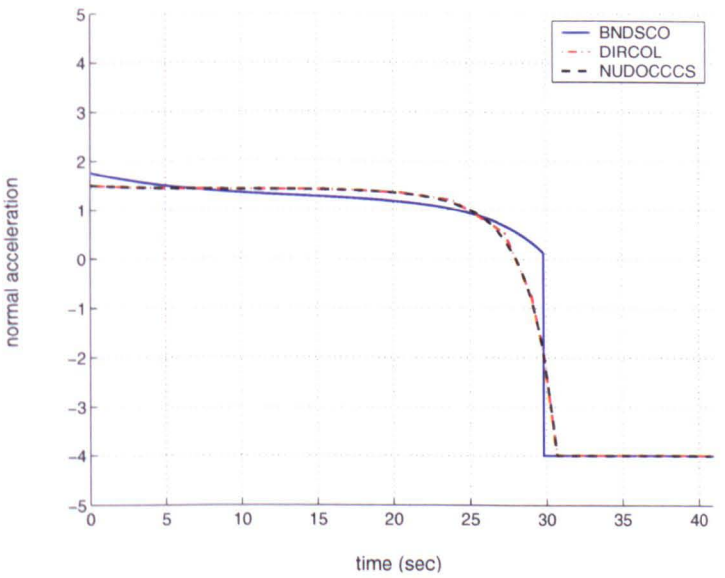


Figure 4.10: Comparison of BNDSCO, DIRCOL and NUDOCCCS results of the normal acceleration versus time, constrained minimum time problem.

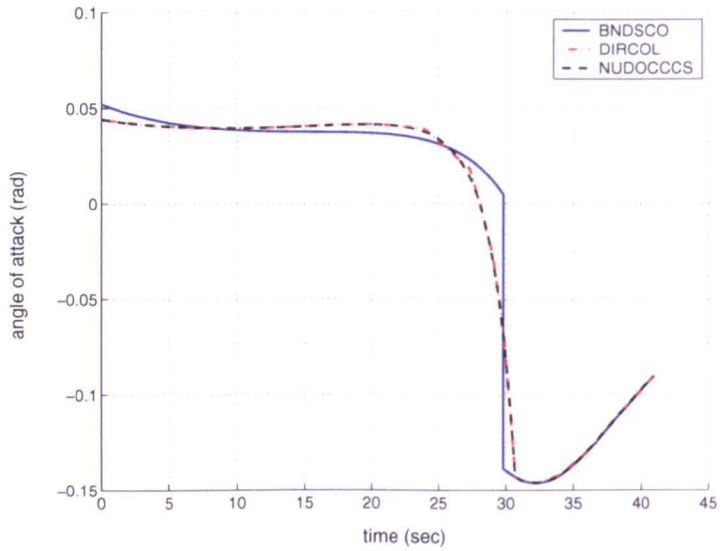


Figure 4.11: Comparison of BNDSCO, DIRCOL and NUDOCCCS results of the angle of attack versus time, constrained minimum time problem.

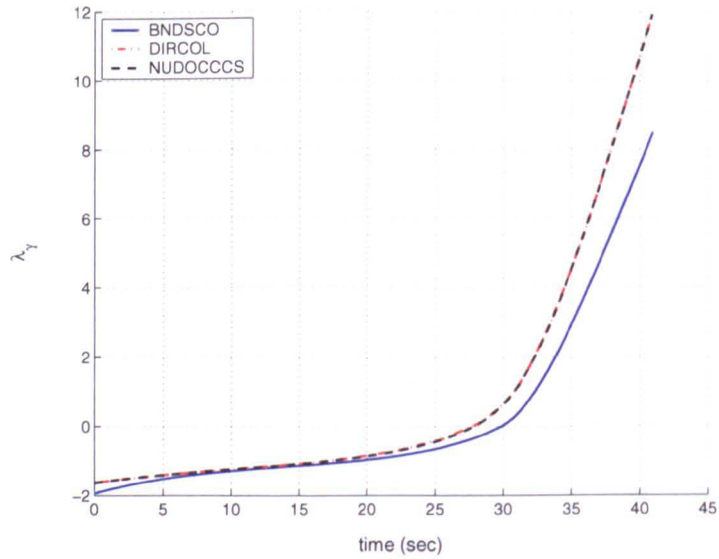


Figure 4.12: Comparison of BNDSCO, DIRCOL and NUDOCCCS results of the λ_γ versus time, constrained minimum time problem.

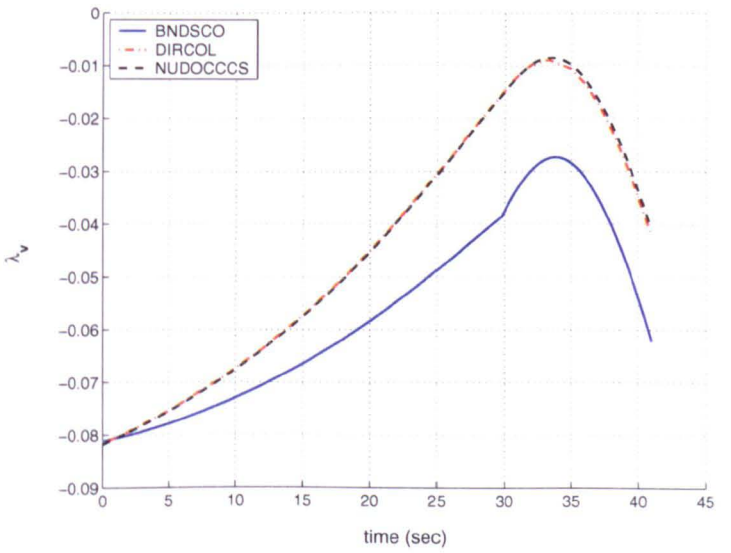


Figure 4.13: Comparison of BNDSCO, DIRCOL and NUDOCCCS results of the λ_V versus time, constrained minimum time problem.

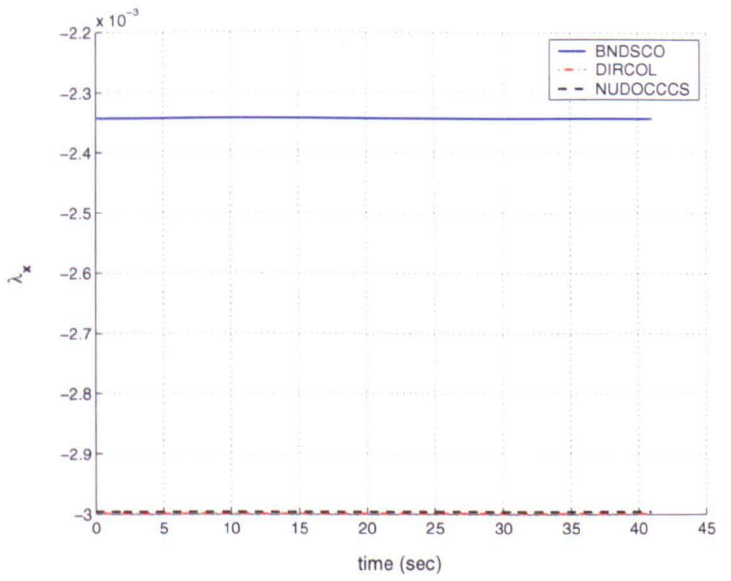


Figure 4.14: Comparison of BNDSCO, DIRCOL and NUDOCCCS results of the λ_x versus time, constrained minimum time problem.

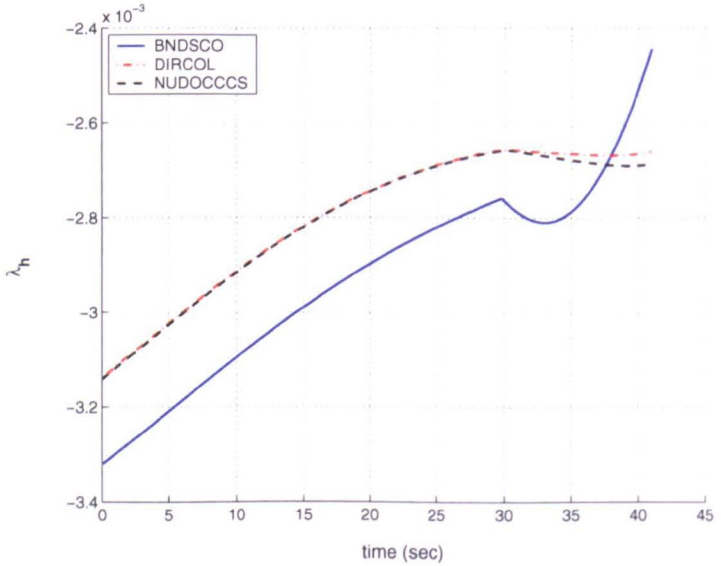


Figure 4.15: Comparison of BNDSCO, DIRCOL and NUDOCCCS results of the λ_h versus time, constrained minimum time problem.

4.5 Summary and Discussion

The purpose of this chapter was to find the fastest trajectory to strike a fixed target which must be hit from above. Firstly, the computational results were obtained by using direct method packages DIRCOL and NUDOCCCS for varying final speed. The computational results show that varying the final speed produces no significant differences in the final time (see Table 4.1). Furthermore, if we consider the minimum

Table 4.2: Performance index for the minimum time problem for the case of final speed $V_{t_f} = 310$ m/s

Software	J (sec)
DIRCOL	40.90780
NUDOCCCS	40.90739
BNDSCO	40.94762
	(Switching time 29.79072)

time only, then the maximum final speed 310 m/s will inflict the greatest damage on the target. But if we consider both the optimal time and minimum exposure during manoeuvre, then further analysis must be done.

The minimum time solution gives the flight time less than a second faster compared to the minimum altitude solution, while the exposure during the manoeuvre for the minimum altitude problem is smaller than the minimum time problem. Hence, the trade-off between both objectives (minimum time and minimum altitude) must be taken into account.

The generic trajectory for the minimum time problem has nearly the same performance as in the minimum altitude. Since we only optimise the time, the missile climbs earlier than in the minimum altitude problem for the same final speed. Thus the level flight arc only occurs for the case of final speed 250 m/s. For the cases of final speed 270 m/s and 310 m/s the missile climbs directly at the beginning of launch.

During climbing, the thrust is on the maximum value for the cases of final speed 310 m/s while for the cases of final speed 250 m/s and 270 m/s during climbing the thrust switches to the minimum value. The maximum normal acceleration constraints are active only for the case 250 m/s in the middle of climbing which occurs in a few seconds. The normal acceleration and the thrust then switches to the minimum value. For the case of final speed 270 m/s the thrust switches to the minimum value at the end of climbing followed by the normal acceleration switches to the minimum value.

At the start of diving, the minimum normal acceleration is active while the thrust is on the maximum value for the case 310 m/s. In the middle of diving for the cases of final speed 250 m/s and 270 m/s the thrust switches back to maximum value to gain enough power to achieve the final speed while the normal acceleration saturated on the minimum. The structure of the equations and switching time is given in Figure 4.16–4.19.

The computational results of the direct method and indirect method are compared for the case 310 m/s. In this case the minimum normal acceleration constraint is active during diving. DIRCOL and NUDOCCCS produce nearly the same trajectory for the state variables. While for the co-state variables, DIRCOL and NUDOCCCS produce nearly the same for the unconstrained arc, but not for the constrained arc. Furthermore,

both of them give a good initial guess for the BNDSCO. The minimum time of the indirect method is greater then in the direct method result. It is possible because in the direct method the constraints may not be accurately satisfied due to the approximation.

4.5.1 Comments on Switching Structure

Similarly to Section 3.5.1, the jumps in the angle of attack α and the switching structure of thrust T are investigated here. Consider first Figures 4.1–4.6 on pp. 95–97, where for the case $V_{t_f} = 310$ m/s the missile climbs directly. Although the altitude increases, see Figure 4.1 on page 95, and the angle of attack α is relatively constant, see Figure 4.4 on page 96, but the normal acceleration is not saturated during climbing, see Figure 4.6 on page 97. Thus, while the thrust is saturated at the maximum value, $T = T_{max}$, no other constraints are active, so that optimal α is obtained from $H_\alpha = 0$, see equation (4.10) on page 102, where T_{max} should be substituted for T. While rapid climbing is necessary, the missile should also turn over to begin its dive as soon as possible. Therefore the angle of attack α and speed V decreases to facilitate the missile turns over. The angle of attack α reaches a negative value at time 28.12411 sec, which causes the activation of minimum normal acceleration constraint ($L = L_{min}$) at time 30.68085 sec, marking the end of free arc. The remainder of the trajectory is constrained arc, with $L = L_{min}$, so that α is obtained from equation (4.30) on page 108.

In summary, for the case $V_{t_f} = 310$ m/s, the thrust T is saturated at T_{max} throughout the whole trajectory, and the trajectory starts with (i) a free arc, lasting from 0 to 30.68085 sec, and finishes with (ii) a constrained arc ($L = L_{min}$) from 30.68085 sec till t_f. As for optimal α, it is computed from equation (4.10) on (i), and from equation (4.30) for (ii). This results in α being a continuous function of time t, but with one point of non-smoothness, coinciding with the (i)→(ii) transition, see Figure 4.4.

Let us investigate the case $V_{t_f} = 270$ m/s. The missile is launched at $h = h_{min}$, the first arc (level flight) is directly a constrained arc and therefore optimal α is obtained not from $H_\alpha = 0$, but form equation (4.19a) on page 104. The first arc ends at time 1.662784 sec, as the missile starts climbing, thus beginning the second arc. Although the altitude increases, see Figure 4.1, the normal acceleration is not saturated during climbing. While rapid climbing is necessary, the missile should also turn over to begin

its dive as soon as possible. Therefore optimal α should decrease rapidly towards negative values (to facilitate turnover). However, rapid decrease in α via equation (4.10) is not sufficient for the required turnover, so it is helped by switching the thrust from T_{max} to T_{min}. The rapidly decreasing α immediately activates the $L = L_{min}$ constraint which remains active till t_f. Still before t_f, the thrust switches back to T_{max}, once it short-lasting lowering to T_{min} accomplished the necessary facilitation of the missile turnover.

In summary, for the case $V_{t_f} = 270$ m/s, the trajectory starts with (i) a constrained arc ($h = h_{min}$), lasting from 0 to 1.662784 sec, followed by (ii) a free arc between 1.662784 and 29.09871 sec, and finishes with (iii) another constrained arc ($L = L_{min}$). As for optimal α, it is computed from equation (4.19a) on (i), then from equation (4.10) on page 102 on (ii), and from equation (4.30) on page 108 for (iii). This results in α being a continuous function of time t, but with two points of non-smoothness, coinciding with the (i)→(ii) and (ii)→(iii) transitions, see Figure 4.4.

For the case $V_{t_f} = 250$ m/s, the time of the level flight is longer than for the case of final speed 270 m/s. The first arc (level flight) is directly a constrained arc ($h = h_{min}$) and therefore optimal α is computed from equation (4.19a). The first arc ends at time 4.955889 sec, as the missile starts climbing, thus beginning the second arc. Although the altitude increases, see Figure 4.1, and the speed is relatively big, the normal acceleration is not saturated until 27.77363 sec. The free arc ends when the maximum normal acceleration ($L = L_{max}$) is active. During the free arc α is obtained from equation (4.10). Then α is obtained from equation (4.26) on page 106 for the time when the maximum normal acceleration ($L = L_{max}$) is active. Just after that the thrust switches to minimum value to facilitate the missile turnover. Although the thrust switches, the normal acceleration is still saturated. However, to facilitate the turnover the angle of attack α decreases rapidly and reaches a negative value at time 28.59961 sec, which causes the normal acceleration to switch to saturate at the minimum value ($L = L_{min}$) at the same time, and remains active till t_f. At this time, α is computed from equation (4.30). Still before t_f, the thrust switches back to T_{max} to facilitate the speed V reaching the final condition.

In summary, for the case $V_{t_f} = 250$ m/s, the trajectory starts with (i) a constrained arc ($h = h_{min}$), lasting from 0 to 4.955889 sec, followed by (ii) a free arc between 4.955889 and 27.77363 sec, then by (iii) another short constrained arc ($L = L_{max}$) from 27.77363 to 28.59961 sec, while in between the thrust switches from T_{max} to T_{min}, and—finally—by (iv) the last constrained arc ($L = L_{min}$) again the switches from T_{min} to T_{max}. As a result, optimal α is no longer a continuous function of time, having a discontinuity at (iii)→(iv) transition. It also has two other points of non-smoothness: at the (i)→(ii) and (ii)→(iii) transitions.

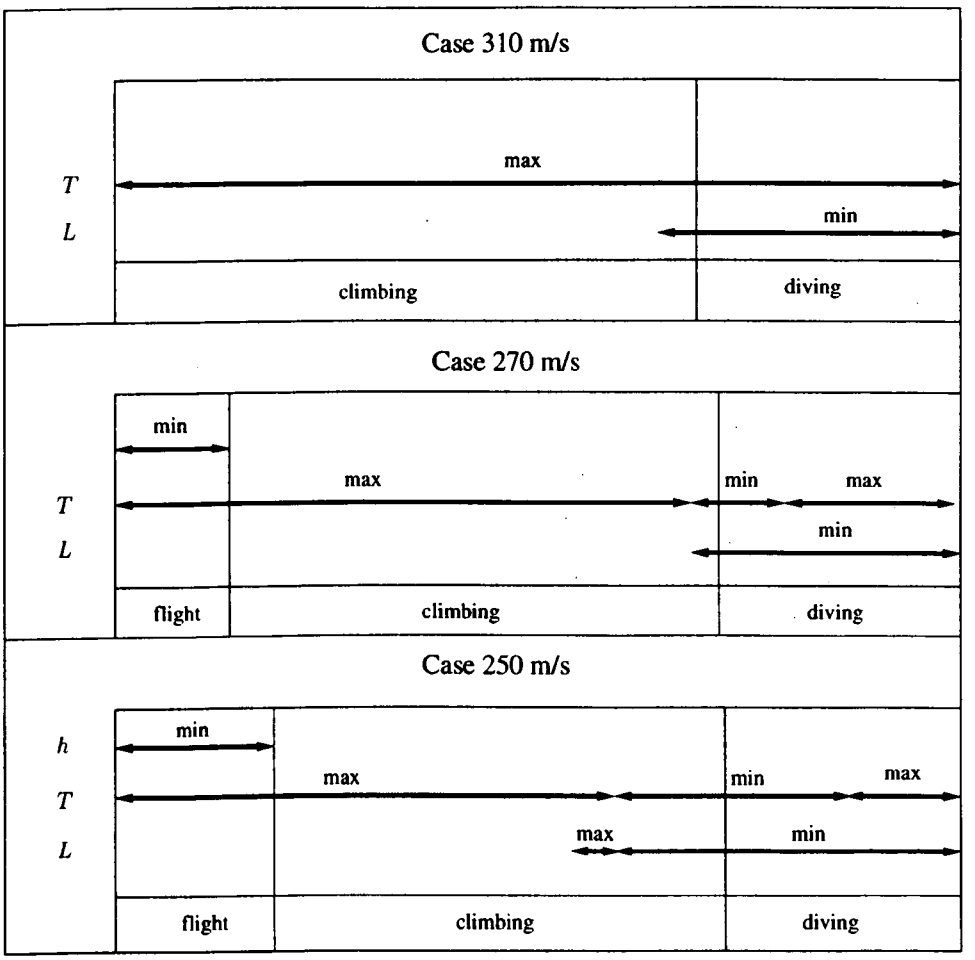


Figure 4.16: Switching structure of the minimum time for the terminal bunt manoeuvre.

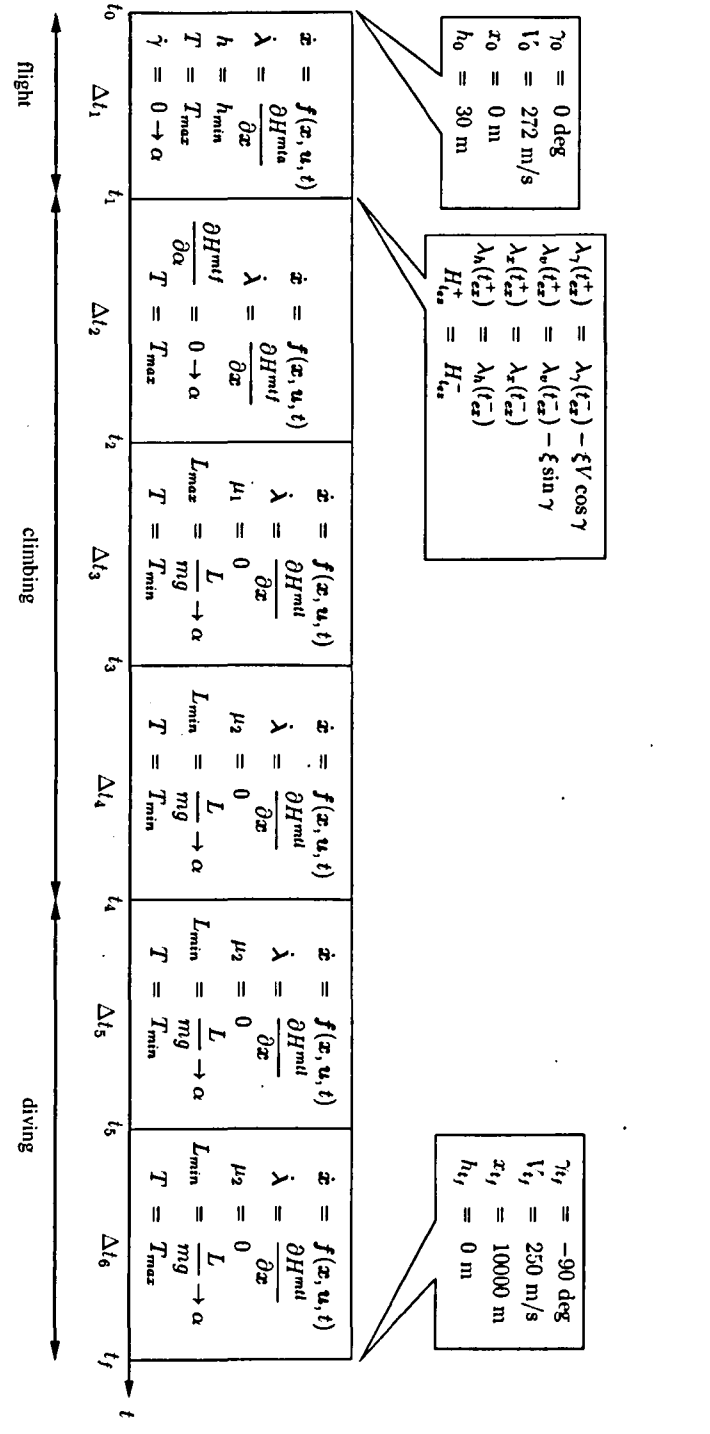


Figure 4.17: Schematic representation of the boundary value problem associated with the switching structure for the minimum time problem, case 250 m/s.

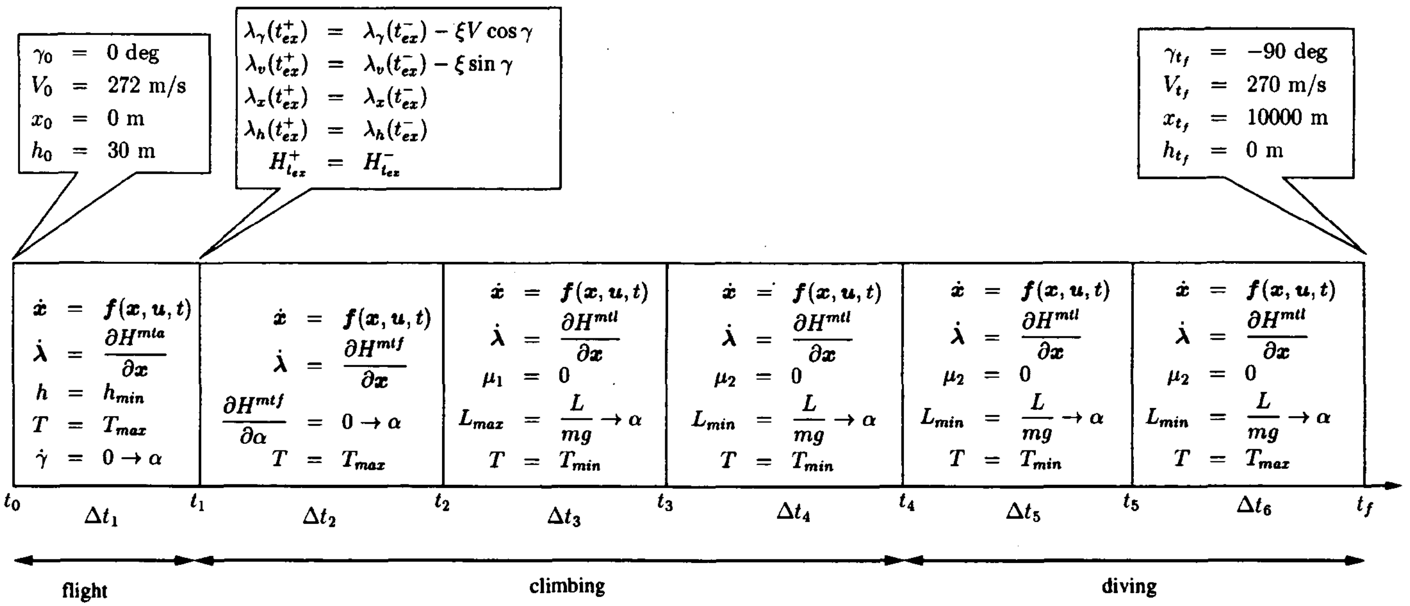


Figure 4.18: Schematic representation of the boundary value problem associated with the switching structure for the minimum time problem, case 270 m/s.

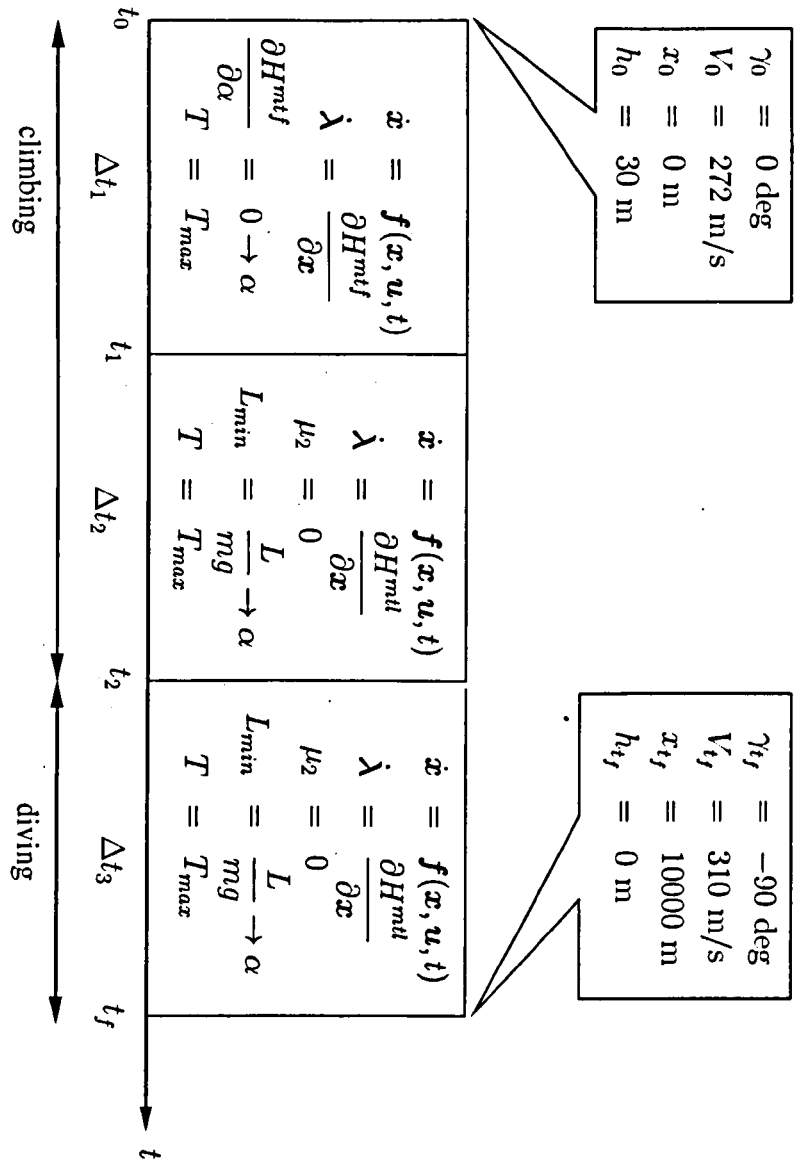


Figure 4.19: Schematic representation of the boundary value problem associated with the switching structure for the minimum time problem, case 310 m/s.

Chapter 5

Software Implementation

This chapter considers example software implementations performed in order to solve the terminal bunt manoeuvre as discussed in Chapter 3 and 4. The direct method is given by DIRCOL and NUDOCCCS packages, while for the indirect method is presented via the BNDSCO package. Section 5.1 focuses on the DIRCOL implementation, followed by the NUDOCCCS implementation in Section 5.2. The indirect method package BNDSCO is discussed in Section 5.3. The three packages illustrate the same problem which is the minimum time problem for the case of final speed 310 m/s.

5.1 DIRCOL implementation

DIRCOL (a Direct Collocation Method for the Numerical Solution of Optimal Control Problems) is a collection of subroutines in FORTRAN designed to solve optimal control problems of the systems described by first order differential equations subject to general equality or inequality constraints on the control and/or state variables [109].

The direct collocation methods transform the optimal control problem into sequence of nonlinear constrained optimisation problems (NLP) by discretising of the state and control variables. In DIRCOL, the controls are chosen as piecewise linear interpolating functions and the states are chosen as continuously differentiable and piecewise cubic function. The NLP results of the transformation are solved by the sequential quadratic programming method SNOPT [47]. One of the advantages of

DIRCOL is that it also computes the estimation of the adjoint variables (co-state) [108] (see Section 2.4.1.1).

This section describes in detail how the terminal bunt manoeuvre problem was implemented in DIRCOL by displaying and commenting on essential subroutines in the main program `user.f` and two input files `DATDIM` and `DATLIM`. This section does not attempt to explain how to implement the general optimal control problem in DIRCOL, as this is done in von Stryk's User's Guide for DIRCOL [109].

Terminal bunt manoeuvre for minimum time problem ($\int_{t_0}^{t_f} dt$) will be given as an illustration of how to implement terminal bunt problem in DIRCOL

5.1.1 Main program `user.f`

- Subroutine `DIRCOM`

  This subroutine defines basic data for the problem such as the gravitational constant g (`GRA`), the mass m (`AM`), the reference area of the missile S_{ref} (`SREF`), the polynomial coefficient for D (equation (1.5)) (`CA2, CA1, CA0`), the polynomial coefficient for L (equation (1.6)) (`CN0`) and the polynomial coefficient of air density ρ (`CRHO2, CRHO1, CRHO0`). The user can exploit `COMMON` block such as `/USRCOM/`, which may be shared by all problem dependent. The data of the terminal bunt problem can be described as:

```
      IMPLICIT NONE
C
C--------BEGIN---PROBLEM-----------------------------------------------
C
C*      COMMON /USRCOM/
C
C---------END----PROBLEM-----------------------------------------------
      DOUBLE PRECISION PI,AM,AGRA,SREF,CA2,CA1,CA0,CN0,CRHO2,CRHO1,CRHO0
C
      COMMON /USRCOM/  PI,AM,AGRA,SREF,CA2,CA1,CA0,CN0,CRHO2,CRHO1,CRHO0

C --- Konstanten aus der Referenz
      PI     = 3.141592653589793238D0
      AM     = 1005
      AGRA   = 9.8
      SREF   = 0.3376
      CA2    = -1.9431
      CA1    = -0.1499
      CA0    = 0.2359
      CN0    = 21.9
      CRHO2  = 3.312D-9
```

```
      CRHO1  = -1.142D-4
      CRHO0  = 1.224
C
      RETURN
      END
```

- Subroutine USRSTV

This subroutine contains the initial estimates of the state and control variable histories, the control parameters, and initial estimates of the events as provided by the user. The initial estimates for terminal bunt problem can be given as:

```
      IMPLICIT NONE
      INTEGER  IPHASE, NX, LU, IFAIL
      DOUBLE PRECISION
     +     TAU, X(NX), U(LU)
C
C--------BEGIN---PROBLEM-----------------------------------------------
C
      DOUBLE PRECISION PI,AM,AGRA,SREF,CA2,CA1,CA0,CN0,CRHO2,CRHO1,CRHO0
      DOUBLE PRECISION AGAM0, V0, X0, AH0, Z0, XF
C
      COMMON /USRCOM/  PI,AM,AGRA,SREF,CA2,CA1,CA0,CN0,CRHO2,CRHO1,CRHO0

      PARAMETER (X0 = 0.D0, AH0 = 30.D0, V0 = 270.0D0)
      PARAMETER (AGAM0   = 0.0D0, Z0 = 30.0D0, XF = 10000.0D0)
C
c      IF (IPHASE .GT. 0) THEN
C
C ------ Initial estimates of X(t) and U(t)
C
         X(1) = AGAM0
         X(2) = V0
         X(3) = X0   + TAU * (XF - X0)
         X(4) = AH0
         U(1) = 0.000D0
         U(2) = 6000.D0
C
C---------END----PROBLEM-------------------------------------------------
C
      RETURN
C --- End of subroutine USRSTV
      END
```

- Subroutine USROBJ

This subroutine contains the objective of the optimal control problem in the Mayer form (see Section 2.1). The parameter NR below specifies the required component of the objective in each phase. The objective function of terminal bunt problem can be defined as:

```
      IMPLICIT NONE
      INTEGER  NX, LU, LP, NR, IFAIL
      DOUBLE PRECISION
     +     ENR, XL(NX), UL(LU), P(LP), FOBJ, XR(NX), UR(LU), TF
C
C--------BEGIN---PROBLEM----------------------------------------------
C
      IF (NR .EQ. 1) THEN
         FOBJ = TF
      ELSE
         FOBJ = 0.0D0
      END IF
C
C---------END----PROBLEM----------------------------------------------
C
      RETURN
C --- End of subroutine USROBJ
      END
```

- Subroutine USRDEQ

  This subroutine provides the right-hand side of the dynamic equations:

```
      IMPLICIT NONE
      INTEGER  IPHASE, NX, LU, LP, IFAIL
      DOUBLE PRECISION
     +     X(NX), U(LU), P(LP), T, F(NX)
      DOUBLE PRECISION RHO,RHOV,A,AN,COSGAM,SINGAM,COSALP,SINALP,TMA,ANM
      DOUBLE PRECISION PI,AM,AGRA,SREF,CA2,CA1,CA0,CN0,CRHO2,CRHO1,CRHO0
C
      COMMON /USRCOM/  PI,AM,AGRA,SREF,CA2,CA1,CA0,CN0,CRHO2,CRHO1,CRHO0
C
C--------BEGIN---PROBLEM----------------------------------------------
C
      INTEGER  I
      INTRINSIC COS, SIN
C --- the air density
      RHO = CRHO2*X(4)**2+CRHO1*X(4)+CRHO0

C --- 0.5 RHO V^2 SREF
      RHOV = 0.5 * RHO * X(2)**2 * SREF
C
C --- the drag
      A  = (CA2 * U(1)**2 + CA1 * U(1) + CA0) * RHOV
C
C --- the lift
      AN = CN0 * RHOV * U(1)
C
C
C --- the differential equations
C     -------------------------
C
      COSGAM = COS(X(1))
      SINGAM = SIN(X(1))
      COSALP = COS(U(1))
      SINALP = SIN(U(1))
```

```
      TMA    = (U(2)-A)/AM
      ANM    = AN/AM

C
      F(1) = (TMA*SINALP + ANM*COSALP - AGRA*COSGAM)/X(2)
      F(2) = TMA*COSALP - ANM*SINALP - AGRA*SINGAM
      F(3) = X(2) * COSGAM
      F(4) = X(2) * SINGAM
C
C---------END----PROBLEM-------------------------------------------------
C
      RETURN
C --- End of subroutine USRDEQ
      END
```

- Subroutine USRNBC

  This subroutine describes the nonlinear boundary conditions of the problem. The parameter IKIND below specifies the type of boundary/switching conditions (explicit or implicit) that has to be computed. The parameter XR is the final conditions of the state variables that are defined by explicit boundary conditions for the terminal bunt problem. The following is the subroutine for terminal bunt problem:

```
      IMPLICIT NONE
C
      INTEGER  IKIND, NRNLN, NX, LU, LP, IFAIL
C**** REAL
      DOUBLE PRECISION
     +     XL(NX), XR(NX), UL(LU), UR(LU), P(LP), EL, ER, RB(NRNLN)
C
C--------BEGIN---PROBLEM----------------------------------
C
C     This problem doesn't have any
C     (nonlinear) implicit boundary/switching conditions
C
C     There are some explicit boundary conditions of the second kind.
      IF (IKIND .EQ. -1) THEN
         XR(1) =  -1.57D0
         XR(2) =  310.0D0
         XR(3) =  10000.0D0
         XR(4) =  0.0D0
      END IF
C
C---------END----PROBLEM-----------------------------------
C
      RETURN
C --- End of subroutine USRNBC
      END
```

- Subroutine `USRNIC`

  This subroutine provides the nonlinear inequality constraints of the problem. The following inequality constraints should be defined for terminal bunt problem:

$$200 \leq V \leq 310 \tag{5.1}$$

$$30 \leq h \tag{5.2}$$

$$1000 \leq T \leq 6000 \tag{5.3}$$

$$-4 \leq \frac{L}{mg} \leq 4 \tag{5.4}$$

  - Altitude constraint (5.2). In terminal bunt problem, the final condition of the altitude is $h_{t_f} = 0$. Therefore the altitude constraint is always contradictory to the final condition on the altitude. It is necessary to give a condition for the altitude constraint to overcome this problem.
  - Normal acceleration (5.4). The normal acceleration constraints can be rewritten as:

$$0 \leq 16 - \left[\frac{L}{mg}\right]^2 \tag{5.5}$$

    The normal acceleration now can be implemented as one constraint instead of two.
  - Air speed (5.1). The air speed constraints can be split into two inequality constraints:

$$0 \leq V - 200 \quad \text{and} \quad 0 \leq 310 - V \tag{5.6}$$

  The constraints on the thrust are not defined here, as they will be defined as lower and upper bounds in file `DATLIM`, see section 5.1.2.

  The following subroutine defines the nonlinear inequality constraints of the terminal bunt problem

```
      IMPLICIT NONE
C
      INTEGER  NGNLN, NX, LU, LP, IPHASE, NEEDG(NGNLN), IFAIL
      DOUBLE PRECISION
C**** REAL
     +     T, X(NX), U(LU), P(LP), G(NGNLN)
      DOUBLE PRECISION Rhok, ACCN
      DOUBLE PRECISION RHO,RHOV,A,AN,COSGAM,SINGAM,COSALP,SINALP,TMA,ANM
      DOUBLE PRECISION PI,AM,AGRA,SREF,CA2,CA1,CA0,CN1,CRHO2,CRHO1,CRHO0
C
      COMMON /USRCOM/  PI,AM,AGRA,SREF,CA2,CA1,CA0,CN1,CRHO2,CRHO1,CRHO0
C
C
C--------BEGIN---PROBLEM---------------------------------
C
      Rhok = CRHO2*X(4)*X(4)+CRHO1*X(4)+CRHO0
      ACCN = (21.9/2 * U(1)*Rhok*X(2)*X(2)*SREF)/(1005*9.8)
C --Normal Acceleration Constraint
      G(1) = 16 - ACCN*ACCN
C --Altitude Constraint
      IF (X(3).LE.7500.0D0) THEN
          G(2) = X(4)-30.0D0
      ENDIF
C
C---------END----PROBLEM---------------------------------
C
      RETURN
C --- End of subroutine USRNIC
      END
```

- Subroutine USRNEC

  This subroutine describes the nonlinear equality constraints of the problem. The terminal bunt problem does not have any equality constraint.

5.1.2 Input file DATLIM

The input file DATLIM is always needed to run DIRCOL. This file prescribes values at the initial time, final time, lower and upper bounds of the state and control variables data limits. In this file the lower and upper bounds of the final time are prescribed. The lower and upper bounds for the fixed final time can be defined by same values of the lower and upper bounds.

```
*************************************************************************
*                          file  DATLIM                                 *
* (prescribed values at initial time, final time and switching points, *
*         lower and upper bounds for all variables X, U, P, E)         *
*************************************************************************
*
*  the NX values of X(1) through X(NX) at E(1)=T0, E(M)=TF are
```

```
1 , 1, 0.00D0
1 , 1, 272.0D0
1 , 1, 0.0D0
1 , 1, 30.D0
1 , 0, 0.0D0
*  the LU values of U(1) through U(LU) at E(1)=T0, E(M)=TF are
0 , 0
0 , 0
*
* 1. switching point E(2):
* ----------------------
*  the NX values of X(1) through X(NX) at the switching point are
* ...
*  the LU values of U(1) through U(LU) at the switching point are
*
* 2. switching point E(3):
* ----------------------
*  the NX values of X(1) through X(NX) at the switching point are
* ...
*  the LU values of U(1) through U(LU) at the switching point are
* ...
*
*  the lower and upper bounds of the events E(2),...,E(M)=TF are
*  ------------------------------------------------------------
*    MIN       ,       MAX
     30.0D0    ,       50.0D0
*
*  1. phase:
* ---------
*  the NX lower and upper bounds of the state variables X are
*  X(I)MIN     ,     X(I)MAX
   -1.57D0     ,  1.57D0
   +200.0D0    ,  +310.0D0
   0.0D0       ,  +10000.0D0
   0.00D0      ,  +1900.00D0
   0.00D0      ,  +100000.00D0
*
*  the LU lower and upper bounds of the control variables U are
*  U(K)MIN     ,     U(K)MAX
  -0.3D0       ,  +0.3D0
  +1000.0D0    ,  +6000.0D0
*
*  2. Phase:
* ---------
*  the NX lower and upper bounds of the state variables X are
*  X(I)MIN     ,     X(I)MAX
*
*  the LU lower and upper bounds of the control variables U are
*  U(K)MIN     ,     U(K)MAX
*
*
*  the LP lower and upper bounds of the control parameters P are
*  ------------------------------------------------------------
*  P(K)MIN     ;     P(K)MAX
*
*
*234567890123456789012345678901234567890123456789012345678901234567890123456789012
```

5.1.3 Input file DATDIM

The input file DATDIM is always needed to run DIRCOL. This file prescribes the following block information:

- name of the problem
- type of simulation has to be performed by DIRCOL and the maximum number of iterations
- the major optimality tolerance of SNOPT
- the nonlinearity feasibility tolerance
- major print level
- iScale (the type of scaling) and iDiff (the type of finite difference approximations of nonzero derivatives
- the dimension of the state, control, and control parameter
- number of phases
- number of nonlinear implicit boundary constraints
- number of nonlinear inequality and equality constraints
- number of grid points
- grid point parameter
- starting values
- estimates of the adjoint variables and the switching structure
- name of the state variables
- name of the control variables
- definition of the constraints for the angle

- name of inequality constraints

```
*************************************************************************
*                          file  DATDIM                                 *
*       (Dimensions of the parameterized optimal control problem)       *
*************************************************************************
*
* NAME of the OPTIMAL CONTROL PROBLEM
* -----------------------------------
*23456789012345678901234567890* (<-- max. length of name)
Terminal Bunt Manoeuvre
*
* iAction:
* -------
*  - OPTIMIZATION using NPSOL ..................................... (0)
*  - a check of all dimensions of feasibility ..................... (1)
*  - a check of subroutines & computation of starting trajectory . (2)
* or computation of a FEASIBLE TRAJECTORY by
*  - objective min-max1 / use NPOPT ............................... (3)
*  - objective min-max1 / use NPSOL ............................... (4)
*  - objective min-max2 / use NPSOL ............................... (5)
* or actions involving SNOPT:
*  - OPTIMIZATION         using NPOPT ............................. (6)
*  - OPTIMIZATION         using SNOPT (dense Jacobian)............. (7)
*  - OPTIMIZATION         using SNOPT (sparse Jacobian)............ (8)
*  - FEASIBLE TRAJECTORY using SNOPT (sparse Jacobian)............ (9)
*
* iAction, MajItL = ?,?
*
* 0, -5
* 1
* 2, -1
* 4, -1
* 5, -1
* 6, -1
* 7, -1
*
 8, -11
*
* Optional SQP-Parameters:
* ------------------------
*  Optimality Tolerance             EPSOPT = ?
1.0E-9
*
*  Nonlinear Feasibility Tolerance  EPSNFT = ?
1.0E-9
*
*  Major Print Level (0, 5 or 10)          = ?
5
*
* which SCALINGS and DIFFERENCE APPROXIMATIONS are to be used:
* ----------------------------------------------------------
* iScale:
* ------
*  - automatic scaling (but for X, U, E in each phase the same)      (0)
*  - read scalings from file 'DATSKA'                                (1)
*  - use no scaling                                                  (2)
*  - automatic scaling (X, U, E in each phase different)             (3)
*  - automatic scaling (X, U in each phase the same, but E different)(4)
```

```
*
* iDiff:
* -----
*  - forward difference approximations of DIRCOL (default)  (0)
*  - internal difference approximation of NPSOL or SNOPT   (-1)
*
* iScale, iDiff = ?,?
0, -1
*
* NUMBER of STATE VARIABLES     ( NX ),
* ------ of CONTROL VARIABLES   ( LU ),
*        of CONTROL PARAMETERS  ( LP ),
* NX, LU, LP = ?
5, 2, 0
*
* NUMBER of PHASES  M1 = ?
* -----------------------
1
*
* NUMBERS of NONLINEAR IMPLICIT BOUNDARY CONSTRAINTS
* --------------------------------------------------
* NRNLN(1)
* ...
* NRNLN(M1)
0
*
* NUMBERS of NONLINEAR INEQUALITY and EQUALITY CONSTRAINTS
* --------------------------------------------------------
* in phases 1 through M1:
* NGNLN(1) , NHNLN(1)
* ...
* NGNLN(M1), NHNLN(M1)
*
2,0
*
* NUMBER of GRID POINTS in phases 1 through M1  ( NG(k) >= 3 ):
* -------------------------------------------------------------
* NG(1)
* ...
* NG(M1)
7
*
* GRID POINTs parameters:
* iStartGrid                  | iOptGrid (during optimization):
* ----------                  | --------
* (starting positions):       |  - fixed grid points           (0)
*  - equidistant          (0) |  - movable (collocation error) (1)
*  - as in file DATGIT    (1) |  - movable (variation)         (2)
*  - as Chebysev points    (2) |  - movable (no add. eq. cons.) (3)
*
* iStartGrid, iOptGrid = ?,?
0, 0
*
* STARTING VALUES of  X(t),  U(t),  P,  and  E:
* ---------------------------------------------
*  -  as specified in subroutine USRSTV                        (0)
*  -  as in files GDATX, GDATU (unchanged number of phases)    (1)
*  -  X, U, P as in files GDATX, GDATU and
*     E as specified in USRSTV (changed number of phases)      (2)
```

```
1
*
* ESTIMATES of the ADJOINT VARIABLES and of
* ----------------------------------------
* the SWITCHING STRUCTURES of state and control constraints
*  -  are NOT required    (0)
*  -  are required        (1)
1
*
* NAMES of the  NX  state variables:
* --------------------------------
* X(1)_Name
* ...
* X(NX)_Name
*2345678901234* (<-- max. length of name)
g(t)
s(t)
x(t)
h(t)
z(t)
*
* NAMES of the  LU  control variables:
* ----------------------------------
* U(1)_Name
* ...
* U(LU)_Name
*2345678901234* (<-- max. length of name)
    alpha(t)
    Thrust(t)
*
* the I-th STATE VARIABLE   (I = 1,.., NX) is an UNCONSTRAINED ANGLE
* and varies only in  [ -PI, PI [ : 1 (if yes) or 0 (if not)
*
1
0
0
0
0
*
* the K-th control variable (K = 1,.., LU) is an UNCONSTRAINED ANGLE
* and varies only in  [ -PI, PI [ : 1 (if yes) or 0 (if not)
*
1
0
*
* NAMES of the  NGNLN(1)  nonlinear INEQUALITY CONSTRAINTS of the 1-st phase:
* -----
* 1-st name
* ...
* NGNLN(1)-th name
*2345678901234* (<-- max. length of name)
*
16 - (N/Mg)^2
h - hmin
*
*
* NAMES of the  NGNLN(2)  nonlinear INEQUALITY CONSTRAINTS of the 2-nd phase:
* 1-st name
* ...
```

```
* NGNLN(1)-th name
*2345678901234* (<-- max. length of name)
*
*
*234567890123456789012345678901234567890123456789012345678901234567890123456789012
```

5.1.4 Grid Refinement and Dimension of DIRCOL

The grid refinement can be done either by editing the input file `DATGIT` or increasing the number of grid point at the input file `DATDIM`. The dimension of the internal DIRCOL might be changed by editing the file `dircol.h`. The dimension of the constraints and the grid can be edited in this file.

5.2 NUDOCCCS Implementation

NUDOCCCS (Numerical Discretisation method for Optimal Control problems with Constraints in Controls and States) is a collection of FORTRAN codes developed by Büskens [26] to solve optimal control problem by discretising the state and control variables (see Section 2.4.1.1). This section presents the subroutines which must be prepared before solving the problem using NUDOCCCS. The minimum time problem is given as an example.

5.2.1 Main Program

In the main program the user must define mainly:

- number of ordinary differential equations
- number of controls
- number of maximum grid points
- number of unknown initial values
- number of multiple shooting nodes

- number of constraints (state and mixed constraints)
- number of inequality constraints related to state and mixed constraints
- number of points equality, e.g. terminal conditions
- maximum precision
- order of the constraints
- initial guess for the control variables, etc.

```
C-----------------------------------------------------------------
C Bunt Manoeuvre Problem - Minimum Time Problem
C-----------------------------------------------------------------

      PROGRAM MAIN

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      PARAMETER(
     A   NDGL     = 5,          ! #ODE                        (>0)
     B   NSTEUER  = 2,          ! #CONTROLS                   (>0)
     C   NDISKRET = 201,        ! MAX # OF GRIDPOINTS         (>1)
     D   NUNBE    = 1,          ! #UNKNOWN INITIAL VALUES     (>0)
     E   NSTUETZ  = 1,          ! #MULTIPLE SHOOTING KNOTES   (>0)
     F   NNEBEN   = 1,          ! #STATE OR MIXED CONSTRAINTS
     G   NUGLNB   = 1,          ! #THEREOF INEQUALITY CONSTRAINTS
     H   NRAND    = 4,          ! #POINTEQUALITIES (E.G. TERMINAL CONDITIONS)
     I   NARTADJ  = 1,          ! TYPE OF APPROXIMATING ADJOINTS
     J   IPRINT   = 5,          ! PRINT LEVEL
     K   DEL1     = 1.0D-6,     ! FINITE DIFFERENCE FOR OPTIMIZER
     L   DEL2     = 1.0D-4,     ! FINITE DIFFERENCE FOR GRIDFIT
     M   EPS      = 1.0D-12,    ! HIGHEST REACHABLE PRECISION OF SOLUTION
     N   EPS2     = 1.0D-120,   ! SMOOTHING OPERATOR FOR TRUNCATION ERRORS
     O   EPS3     = 1.0d-6,     ! PRECISION OF GRIDREFINEMENT
     P   NZUSATZ  = NUNBE*NSTUETZ,
     Q   N        = (NDISKRET+2)*NSTEUER+NZUSATZ,
     R   M        = NDISKRET*NNEBEN+NRAND+NZUSATZ-NUNBE,
     S   ME       = M-NDISKRET*NUGLNB,
     T   MAX1M    = M)

      DIMENSION
     A  X(NDGL,NDISKRET),U(NSTEUER,NDISKRET+2),DFDU(N),
     B  G(MAX1M),T(NDISKRET),UNBE(NUNBE,NSTUETZ),
     C  UHELP(N),DCDU(MAX1M,N),BL(N+M),BU(N+M),
     D  WORK(3*N*N+2*N*M+21*N+22*M),IWORK(4*N+3*M),
     E  MSDGL(NUNBE),MSSTUETZ(NSTUETZ),IUSER(22+NUNBE+NSTUETZ),
     F  USER(10+7*NDGL+NNEBEN+NSTEUER+NDISKRET*(NDGL+NSTEUER+5)),
     G  ADJ(NDGL,NDISKRET),ADJH(NDGL),DISERR(NDISKRET),
     H  U2(NSTEUER,NDISKRET+2),X2(NDGL,NDISKRET+2),T2(2*NDISKRET),
     I  DSDXH(NNEBEN*NDGL+2*NDGL),DFDXH(NDGL*NDGL+2*NDGL),
     J  PDSDX(NDGL),PD2SD2X(NDGL,NDGL),PDFDX(NDGL,NDGL),
```

```
      K  CONORDER(NNEBEN+1),CONH(4*NNEBEN+1)

       COMMON/RK/rkeps,tol


C ORDER OF CONSTRAINTS --( OPTIONAL, MUST NOT BE SET)--------
c       conorder(1) = 2



C NUMBER OF DISCRETE POINTS -------------------------------
      write(*,*) 'NDISKRET at the beginning (e.g. 21):'
      read(*,*) ndis1 !#OF GRIDPOINTS AT THE BEGINNING  (>1)



C FIT DYNAMICAL DIMENSIONS --------------------------------
      N1         = (NDIS1+2)*NSTEUER+NZUSATZ
      M1         = NDIS1*NNEBEN+NRAND+NZUSATZ-NUNBE
      ME1        = M1-NDIS1*NUGLNB
      MAX1M1     = MAX(1,M1)

C AEQUIDISTANTE DISKRETISIERUNG---------------------------
C DISCRETIZATION OF TIME ----------------------------------
C HERE: EQUIDISTANT ---------------------------------------
      DO 104 I=1,NDIS1
         T(I) = 1.0d0/(NDIS1-1)*(I-1)
104.  CONTINUE



C RKEPS AND TOL ARE ONLY USED FOR NART=8,9,18,19,28,29 ----
C INITIAL STEPSIZE OF RKF ---------------------------------
C IF RKF=0 THEN RADAU5 SOLVER IS USED (COMPARE SUBROUTINE MAS)
      rkeps  = 0.0d0
C TOLERANCE OF RKF- OR DAE-SOLVER -------------------------
      tol    = 1.0d-8

C INITIAL PRECISION OF SOLUTION ---------------------------
C TENDS TO EPS IF GRIDFIT IS USED SEVERAL TIMES -----------
      epsgit = 1.0d-5


C STARTSCHAETZUNG DER STEUERUNGEN -------------------------
C INITIAL GUESS FOR CONTROL VARIABLES ---------------------
      do 202 i=1,ndis1
         u(1,i) = 0.0112d0
         u(2,i) = 6000.0d0
202   CONTINUE
C ONLY USED FOR CUBIC INTERPOLATION OF CONTROL ------------
      u(1,ndiskret+1) = 0.011d0
      u(2,ndiskret+2) = 6000.0d0


C STARTSCHAETZUNG DER FREIEN ANFANGSWERTE UND MEHRZIELKNOTEN
C HERE: NOT USED ------------------------------------------
      DO 300 I=1,NUNBE
```

```
        DO 300 J=1,NSTUETZ
          UNBE(1,J) = 25.0d0
300   CONTINUE


C --------TYPE OF INTERPOLATION, INTEGRATION, OPTIMIZATION---------
C --------NART=0..9 ARE FASTES AND WORK WELL FOR MOST PROBLEMS-----
C NART= 0:                    EULER INTEGRATION,      NO CONTROL INTERPOL.
C NART= 1:                     HEUN INTEGRATION,      NO CONTROL INTERPOL.
C NART= 2:        IMPR. POLY. EULER INTEGRATION, CONST. CONTROL INTERPOL.
C NART= 3:        IMPR. POLY. EULER INTEGRATION,   LIN. CONTROL INTERPOL.
C NART= 4:           RUKU 4 ENGLAND INTEGRATION,   LIN. CONTROL INTERPOL.
C NART= 5:           RUKU 5 ENGLAND INTEGRATION,   LIN. CONTROL INTERPOL.
C NART= 6:           RUKU 4 ENGLAND INTEGRATION,  CUBIC CONTROL INTERPOL.
C NART= 7:           RUKU 5 ENGLAND INTEGRATION,  CUBIC CONTROL INTERPOL.
C NART= 8: RKEPS>0: RKFEHLBERG 7/8 INTEGRATION, CONST. CONTROL INTERPOL.
C NART= 8: RKEPS=0: IMPLIC. RADAU5 INTEGRATION, CONST. CONTROL INTERPOL.
C NART= 9: RKEPS>0: RKFEHLBERG 7/8 INTEGRATION,   LIN. CONTROL INTERPOL.
C NART= 9: RKEPS=0: IMPLIC. RADAU5 INTEGRATION,   LIN. CONTROL INTERPOL.
C --------NART=10...19 ARE SLOWER BUT SAVER-----------------------
C NART=10: AS NART=0 BUT SLOWER AND WITH CENTRAL DIFFERENCES IF NECESSARY
C NART=11: AS NART=1 BUT SLOWER AND WITH CENTRAL DIFFERENCES IF NECESSARY
C NART=12: AS NART=2 BUT SLOWER AND WITH CENTRAL DIFFERENCES IF NECESSARY
C NART=13: AS NART=3 BUT SLOWER AND WITH CENTRAL DIFFERENCES IF NECESSARY
C NART=14: AS NART=4 BUT SLOWER AND WITH CENTRAL DIFFERENCES IF NECESSARY
C NART=15: AS NART=5 BUT SLOWER AND WITH CENTRAL DIFFERENCES IF NECESSARY
C NART=16: AS NART=6 BUT SLOWER AND WITH CENTRAL DIFFERENCES IF NECESSARY
C NART=17: AS NART=7 BUT SLOWER AND WITH CENTRAL DIFFERENCES IF NECESSARY
C NART=18: AS NART=8 BUT SLOWER AND WITH CENTRAL DIFFERENCES IF NECESSARY
C NART=19: AS NART=9 BUT SLOWER AND WITH CENTRAL DIFFERENCES IF NECESSARY
C --------NART=20...29 ARE MUCH SLOWER BUT MUCH MORE SAVE----------
C NART=20: AS NART=10 WITH ORIGINAL OPTIMIZERSETTINGS, PLUS DIFF. CHECKS
C NART=21: AS NART=11 WITH ORIGINAL OPTIMIZERSETTINGS, PLUS DIFF. CHECKS
C NART=22: AS NART=12 WITH ORIGINAL OPTIMIZERSETTINGS, PLUS DIFF. CHECKS
C NART=23: AS NART=13 WITH ORIGINAL OPTIMIZERSETTINGS, PLUS DIFF. CHECKS
C NART=24: AS NART=14 WITH ORIGINAL OPTIMIZERSETTINGS, PLUS DIFF. CHECKS
C NART=25: AS NART=15 WITH ORIGINAL OPTIMIZERSETTINGS, PLUS DIFF. CHECKS
C NART=26: AS NART=16 WITH ORIGINAL OPTIMIZERSETTINGS, PLUS DIFF. CHECKS
C NART=27: AS NART=17 WITH ORIGINAL OPTIMIZERSETTINGS, PLUS DIFF. CHECKS
C NART=28: AS NART=18 WITH ORIGINAL OPTIMIZERSETTINGS, PLUS DIFF. CHECKS
C NART=29: AS NART=19 WITH ORIGINAL OPTIMIZERSETTINGS, PLUS DIFF. CHECKS
      write(*,*) 'NART (e.g. 4):'
      read(*,*) nart                ! IN GENERAL BEST RESULTS WITH NART=4


 401  write(*,*) 'NDISKRET for next calculations '


C Set before each call to NUDOCCCS -------------------------
      iter  = 0                     ! NO. OF ITERATIONS
      ifail = -1                    ! ERROR MESSAGE


C START OPTIMIZATION ---------------------------------------
      CALL NUDOCCCS(NDGL,NSTEUER,NDIS1,NUNBE,NNEBEN,NUGLNB,NRAND,
     1 NZUSATZ,nart,N1,M1,ME1,MAX1M1,ITER,IFAIL,IPRINT,DEL1,EPSGIT,
     2   X,U,DFDU,FF,G,DCDU,BL,BU,T,UNBE,UHELP,
     3   NSTUETZ,MSDGL,MSSTUETZ,IWORK,WORK,IUSER,USER)
```

```
C POSTOPTIMAL CALCULATION OF ADJOINTS -----------------------
      CALL ADJUNG(NDGL,NSTEUER,NDIS1,NUNBE,NNEBEN,NUGLNB,NRAND,
     1    NZUSATZ,NART,NARTADJ,N1,M1,ME1,MAX1M1,ITER,IFAIL,IPRINT,DEL1,
     2    EPSGIT,X,U,DFDU,FF,G,DCDU,BL,BU,T,UNBE,UHELP,NSTUETZ,MSDGL,
     3    MSSTUETZ,IWORK,WORK,IUSER,USER,ADJ,DSDXH,DFDXH,ADJH)


C SAVE RESULTS ---------------------------------------------
      CALL AUSGABE(FF,x,adj,UHELP,T,G,NDGL,NSTEUER,NDIS1,NNEBEN,
     1    NRAND,N1,M1)
      write(*,*) 'State, adjoints and control saved.'


C ADAPTIVE AUTOMATIC GRIDREFINEMENT ------------------------------
1243   CALL GITTERFIT(NDGL,NSTEUER,NDISKRET,NUNBE,NNEBEN,NUGLNB,NRAND,
     1    NZUSATZ,NART,N1,M1,ME1,MAX1M1,ITER,IFAIL,IPRINT,DEL1,
     2    DEL2,EPSGIT,EPS,EPS3,X,U,DFDU,FF,G,DCDU,BL,BU,T,UNBE,UHELP,
     3    NSTUETZ,MSDGL,MSSTUETZ,IWORK,WORK,IUSER,USER,CONORDER,
     4    NDIS1,N1,M1,ME1,MAX1M1,DISERR,X2,U2,T2,pdsdx,pd2sd2x,pdfdx,
     5    conh,FINISH)

      WRITE(*,*) 'Take new grid and optimize new :<ENTER>'
      WRITE(*,*) '<CTRL/C> for termination ...'
      READ(*,*)
      GOTO 401

      STOP
      END
```

5.2.2 Subroutine MINFKT

This subroutine contains the objective function in the Mayer form (see Section 2.1).

```
      SUBROUTINE MINFKT(X,U,T,MIN,NDGL,NSTEUER,NDISKRET)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION MIN,LAGINT
      DIMENSION U(NSTEUER,NDISKRET),X(NDGL,NDISKRET),T(NDISKRET)


C OBJECTIVE: BOLZA FORMULATION ---------------------------
C HERE: Mayer-Formulation --------------------------------
      min   =  x(5,ndiskret)


      RETURN
      END
```

5.2.3 Subroutine INTEGRAL

This subroutine contains the objective function in the Lagrange form (see Section 2.1).

```
      SUBROUTINE INTEGRAL(INT,X,U,T,NDGL,NSTEUER)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION INT
      DIMENSION U(NSTEUER),X(NDGL)

C CAN BE USED FOR LAGRANGE-PART IN OBJECTIVE-------------
C BUT BETTER USE MAYER-FORMULATION -----------------------
c       INT=u(1)*u(1)+u(2)*u(2)+u(3)*u(3)

      RETURN
      END
```

5.2.4 Subroutine DGLSYS

This subroutine provides the right-hand side of the dynamic equations.

```
      SUBROUTINE DGLSYS(X,U,T,DX,NDGL,NSTEUER)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION PI,AM,AGRA,SREF,CA2,CA1,CA0,CN1,CRHO2,CRHO1,CRHO0
      DOUBLE PRECISION RHO,RHOV,A,AN,COSGAM,SINGAM,COSALP,SINALP,TMA,ANM
      DIMENSION X(NDGL),U(NSTEUER),DX(NDGL)
c
C --- Konstanten aus der Referenz
      PI     = 3.14159265358979323 8D0
      AM     = 1005.0d0
      AGRA   = 9.8D0
      SREF   = 0.3376D0
      CA2    = -1.9431D0
      CA1    = -0.1499D0
      CA0    = 0.2359
      CN1    = 21.9D0
      CRHO2  = 3.312D-9
      CRHO1  = -1.142D-4
      CRHO0  = 1.224D0
c
C RIGHT HAND SIDE OF DGL SYSTEM --------------------------
      RHO = CRHO2*x(4)**2+CRHO1*x(4)+CRHO0
c
C --- 0.5 RHO V^2 SREF
      RHOV = 0.5 * RHO * x(2)**2 * SREF
C
C --- the drag
      A   = (CA2 * u(1)**2 + CA1 * u(1) + CA0) * RHOV
C
C --- the lift
      AN = CN1 * RHOV * u(1)
C
C --- the differential equations
C
      COSGAM = dcos(x(1))
      SINGAM = dsin(x(1))
      COSALP = dcos(u(1))
      SINALP = dsin(u(1))
c
```

```
      TMA    = (u(2)-A)/AM
      ANM    = AN/AM
C
      dx(1) = x(5)*(TMA*SINALP + ANM*COSALP - AGRA*COSGAM)/x(2)
      dx(2) = x(5)*(TMA*COSALP - ANM*SINALP - AGRA*SINGAM)
      dx(3) = x(5)*x(2)*COSGAM
      dx(4) = x(5)*x(2)*SINGAM
      dx(5) = 0

      RETURN
      END
```

5.2.5 Subroutine ANFANGSW

This subroutine defines the unknown and known initial conditions.

```
      SUBROUTINE ANFANGSW(AWX,UNKNOWN,NDGL,NUNBE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION AWX(NDGL),UNKNOWN(NUNBE)

C INITIAL CONDITIONS ------------------------------------------------
      AWX(1) = 0.0d0
      AWX(2) = 272.0d0
      AWX(3) = 0.0d0
      AWX(4) = 30.0d0
      AWX(5) = unknown(1)
      RETURN
      END
```

5.2.6 Subroutine RANDBED

This subroutine provides the point equality constraints, i.e. boundary values.

```
      SUBROUTINE RANDBED(X,U,T,R,NDGL,NSTEUER,NDISKRET,NRAND)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET),R(NRAND)
      DIMENSION T(NDISKRET)

C POINTWISE EQUALITY CONSTRAINTS ----------------------------------
C HERE: TERMINAL CONDITIONS ---------------------------------------

      r(1) = X(1,ndiskret) + 1.57d0
      r(2) = X(2,ndiskret) - 310.0d0
      r(3) = X(3,ndiskret) - 10000.0d0
      r(4) = X(4,ndiskret) - 0.0d0

      RETURN
      END
```

5.2.7 Subroutine NEBENBED

This subroutine describes the state and mixed constraints, i.e. inequality path constraints.

```
      SUBROUTINE NEBENBED(X,U,T,CON,NDGL,NSTEUER,NNEBEN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),CON(NNEBEN),U(NSTEUER)

C CONSTRAINTS ----------------------------------------
C NO BOX CONSTRAINTS OF CONTROLS IN THIS ROUTINE------
      PI     = 3.14159265358979323 8D0
      AM     = 1005.0d0
      AGRA   = 9.8D0
      SREF   = 0.3376D0
      CA2    = -1.9431D0
      CA1    = -0.1499D0
      CA0    = 0.2359
      CN1    = 21.9D0
      CRHO2  = 3.312D-9
      CRHO1  = -1.142D-4
      CRHO0  = 1.224D0

      Rhok = CRHO2*X(4)*X(4)+CRHO1*X(4)+CRHO0
      ACCN = (21.9/2 * U(1)*Rhok*X(2)*X(2)*SREF)/(1005*9.8)

      CON(1) = 16.0D0 - ACCN*ACCN

      RETURN
      END
```

5.2.8 Subroutine CONBOXES

This subroutine prescribes the lower and upper bound for the control and constraint defined in the subroutine NEBENBED.

```
      SUBROUTINE CONBOXES(NSTEUER,NDISKRET,NNEBEN,BL,BU,BLCON,BUCON,T)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION BL(NSTEUER),BU(NSTEUER),BLCON(NNEBEN),BUCON(NNEBEN)

C LOWER AND UPPER BOUNDS OF THE CONTROL FUNCTIONS ----------------
C FOR THIS EXAMPLE NO CONTROL CONSTRAINTS ------------------------

      BL(1) = -0.3d0
      BU(1) =  0.3d0
      BL(2) =  1000.0d0
      BU(2) =  6000.0d0
```

```
C LOWER AND UPPER BOUNDS OF THE CONSTRAINTS IN SUBROUTINE NEBENBED-

      BLCON(1) = 0.0d0

      RETURN
      END
```

5.2.9 Subroutine MAS

This subroutine is used for the problem which contains algebraic equations.

```
      SUBROUTINE MAS(NDGL,AM,LMAS,RPAR,IPAR)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION AM(LMAS,NDGL)
      COMMON/MASS/IMAS,MLMAS,MUMAS,idx1dim,idx2dim,idx3dim,MLJAC,MUJAC
C-----------------------------------------------------------------------
C ONLY USED FOR NART=8,9,18,19,28,29 IF RKEPS=0 IN MAIN---------------
C-----------------------------------------------------------------------
C ALLOWS TO SOLVE SYSTEMS OF FORM (e.g. DAE SYSTEMS, TIME CONSUMING)
C
C                          M * x' = f(x,u,t)
C
C WITH CONSTANT (REGULAR OR SINGULAR) MATRIX M
C-----------------------------------------------------------------------
C     MAS          NAME OF SUBROUTINE COMPUTING THE MASS-
C                  MATRIX M.
C                  IF IMAS=0, THIS MATRIX IS ASSUMED TO BE THE IDENTITY
C                  MATRIX AND NEEDS NOT TO BE DEFINED;
C                  SUPPLY A DUMMY SUBROUTINE IN THIS CASE.
C                  IF IMAS=1, THE SUBROUTINE MAS IS OF THE FORM
C                     SUBROUTINE MAS(N,AM,LMAS,RPAR,IPAR)
C                     DOUBLE PRECISION AM(LMAS,N)
C                     AM(1,1)= ....
C                     IF (MLMAS.EQ.N) THE MASS-MATRIX IS STORED
C                     AS FULL MATRIX LIKE
C                          AM(I,J) = M(I,J)
C                     ELSE, THE MATRIX IS TAKEN AS BANDED AND STORED
C                     DIAGONAL-WISE AS
C                          AM(I-J+MUMAS+1,J) = M(I,J).
C
C     IMAS        GIVES INFORMATION ON THE MASS-MATRIX:
C                     IMAS=0: M IS SUPPOSED TO BE THE IDENTITY
C                        MATRIX, MAS IS NEVER CALLED.
C                     IMAS=1: MASS-MATRIX  IS SUPPLIED.
C
C     MLMAS        SWITCH FOR THE BANDED STRUCTURE OF THE MASS-MATRIX:
C                     MLMAS=N: THE FULL MATRIX CASE. THE LINEAR
C                        ALGEBRA IS DONE BY FULL-MATRIX GAUSS-ELIMINATION.
C                     0<=MLMAS<N: MLMAS IS THE LOWER BANDWITH OF THE
C                        MATRIX (>= NUMBER OF NON-ZERO DIAGONALS BELOW
C                        THE MAIN DIAGONAL).
C                  MLMAS IS SUPPOSED TO BE .LE. MLJAC.
C     MUMAS        UPPER BANDWITH OF MASS-MATRIX (>= NUMBER OF NON-
```

```
C                ZERO DIAGONALS ABOVE THE MAIN DIAGONAL).
C                NEED NOT BE DEFINED IF MLMAS=N.
C                MUMAS IS SUPPOSED TO BE .LE. MUJAC.
C     MLJAC      SWITCH FOR THE BANDED STRUCTURE OF THE JACOBIAN:
C                   MLJAC=N: JACOBIAN IS A FULL MATRIX. THE LINEAR
C                      ALGEBRA IS DONE BY FULL-MATRIX GAUSS-ELIMINATION.
C                   0<=MLJAC<N: MLJAC IS THE LOWER BANDWITH OF JACOBIAN
C                      MATRIX (>= NUMBER OF NON-ZERO DIAGONALS BELOW
C                      THE MAIN DIAGONAL).
C
C     MUJAC      UPPER BANDWITH OF JACOBIAN  MATRIX (>= NUMBER OF NON-
C                ZERO DIAGONALS ABOVE THE MAIN DIAGONAL).
C                NEED NOT BE DEFINED IF MLJAC=N.
C
C
C       THE FOLLOWING 3 PARAMETERS ARE IMPORTANT FOR
C       DIFFERENTIAL-ALGEBRAIC SYSTEMS OF INDEX > 1.
C       THE FUNCTION-SUBROUTINE SHOULD BE WRITTEN SUCH THAT
C       THE INDEX 1,2,3 VARIABLES APPEAR IN THIS ORDER.
C       IN ESTIMATING THE ERROR THE INDEX 2 VARIABLES ARE
C       MULTIPLIED BY H, THE INDEX 3 VARIABLES BY H**2.
C
C    IDX1DIM  DIMENSION OF THE INDEX 1 VARIABLES (MUST BE > 0). FOR
C             ODE'S THIS EQUALS THE DIMENSION OF THE SYSTEM.
C             DEFAULT IDX1DIM=NDGL.
C
C    IDX2DIM  DIMENSION OF THE INDEX 2 VARIABLES. DEFAULT IDX2DIM=0.
C
C    IDX3DIM  DIMENSION OF THE INDEX 3 VARIABLES. DEFAULT IDX3DIM=0.

c      IMAS   = 0
c      MLMAS  = ndgl
c      MUMAS  = ndgl
c      MLJAC  = ndgl
c      MUJAC  = ndgl

c      idx1dim = ndgl
c      idx2dim = 0
c      idx3dim = 0

c      do 200 i=1,ndgl
c        AM(i-i+MUMAS+1,i) = 1.0d0
c 200  continue

       RETURN
       END
```

5.3 BNDSCO Implementation

In this section, some important aspects of BNDSCO are discussed in the context of the indirect method implementation of the minimum time problem. Additionally, classical benchmark problem with a pure state constraint due to Bryson and Ho [23] is presented

in Appendix A to highlight other important details of BNDSCO implementation, not covered here.

The discussion begins with a trouble shooting list in Section 5.3.1 for BNDSCO implementation developed in the course of the present work, based on the BNDSCO manual and hard-won experience with the terminal bunt problem. With this aid in hand, the actual code for the minimum time formulation is given in Section 5.3.2.

5.3.1 Possible Sources of Error

- **Analytical**: wrong formulae and/or wrong data.
- **Numerical**: wrong accuracy, singular data (division by 0).
- **Coding**: wrong FORTRAN implementation of formulae/data.

5.3.1.1 Analytical Errors: A Discussion

Since the ultimate numerical tool for solving the underlying TPBVP is the BNDSCO package, it is prudent to formulate optimal control problem, and apply Pontryagin's Maximum Principle, in the way summarised in the BNDSCO manual [76]. However, one should remember that the DIRCOL and/or NUDOCCCS packages are also needed, in general, to generate an initial guess of the solution (especially co-states). Thus, the formulae must be consistent for both packages.

With the above in mind, there follows a list of possible analytical errors produced by a close study of the BNDSCO manual and experience with the terminal bunt problem; the relevant pages of the manual are given.

1. Bolza $\rightarrow$ Mayer; page 10
2. Boundary conditions for co-states λ, appropriate to the problem (unconstrained, free-time, state constraint only, etc.): pp. 12–16, 21–22, 24–25.
3. Jump points for co-state λ, pp. 22, 24–25.
4. Additional checks: consistency of Hamiltonian etc: page 50.

5.3.1.2 Numerical Errors: A Discussion

BNDSCO is a reliable solver of MPBVP with discontinuities, specially written for optimal control problems. However, it has the weakness of all shooting methods that it has a narrow domain of convergence. In other words, if the data are inaccurate or finite precision arithmetic errors accumulate, it will fail. Hence, the list below reflects these premises.

1. BNDSCO requires that time instants (solution nodes) are different from switching points; page 31.

2. Solution nodes need not be equidistant: concentrate them where rapid changes are expected; page 47.

3. Accuracy of integration: play with the parameters

   - Highly accurate solution of IVP required; page 42.
   - Tolerance in integration TOL; page 42.
   - Maximum number of iterations ITMAX; page 40.

4. Accuracy of Newton's method (inside BNDSCO): play with the EPMACH parameter; page 49.

5. Scaling of boundary conditions (stored in variable W of subroutine R): all components should be of the same order; pp. 52–53.

5.3.1.3 Coding Errors: A Discussion

The first check in this category is obvious, i.e. whether the properly derived formulae and data (constants and initial guesses) were coded correctly into FORTRAN. Assuming that this indeed has been done, the list below addresses more subtle possibilities.

1. The whole of BNDSCO is written in double precision, so all variables, constants and data must be in that format, using directive DBLE, when necessary.

2. The all-important initial guesses are not passed through a COMMON, but via the array PAR. The entries of PAR and their subsequent use should be checked carefully; pp. 40, 49.

3. Integer parameter J and L of subroutine F must be consistent with each other; page 48.

5.3.2 BNDSCO Code

5.3.2.1 Main Program

```
C ***********************************************************************
C     Minimum time of terminal bunt manoeuvre,
C
C
C     Properly working for unconstrained minimum time problem
C     GTF     =  -1.57D0
C     V0      =  272.0D0
C     VTF     =  310.0D0
C     H0      =  30.0D0
C     XF      =  10000.0D0
C     HF      =  0.0D0
C
C ***********************************************************************
C
      IMPLICIT  DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MMAX=150,MSMAX=140,MMS=170,NMS=170,NP=96,
     *           NDW=NMS*(120+MMAX*(120+6*NMS)),NDIW=115*NMS)
      DIMENSION X(MMS),XS(MSMAX),Y(NMS,MMS),WORK(NDW)
      DIMENSION TI(NP),GI(NP),VI(NP),PI(NP),AI(NP),
     *          TC(NP),CG(NP),CV(NP),CP(NP),CA(NP)

      INTEGER   JS(MMS,MSMAX),IWORK(NDIW)
      EXTERNAL F, R, DIFSYB
      COMMON /OUT/  DALP
      COMMON /ITEIL/ ITEIL
      COMMON /PARA1/ B1,B2,B3,E1,E2,E3,D1,AM,T,GR,SREF,TM,
     +               HGM,HV,HXP,HH,ALP
c

C     Initial guesses
      OPEN(3,FILE='test1.dat',STATUS='old')
c     OPEN(5,FILE='gnuad310.m',STATUS='old')

C     Result file's name

      OPEN(6,FILE='nmin.txt',STATUS='UNKNOWN')
      OPEN(7,FILE='nminp.dat',STATUS='UNKNOWN')
C
C     Reading the initial guesses
      DO I=1,NP
```

```
        READ(3,*)TI(I),GI(I),VI(I),PI(I),AI(I),CG(I),CV(I),CP(I),CA(I)
      enddo
C-----------------------------------------------------------------
c     Parameters
c
      B1     =  3.312D-9
      B2     = -1.142D-4
      B3     =  1.224D0
      E1     = -1.9431D0
      E2     = -0.1499D0
      E3     =  0.2359D0
      D1     =  21.9D0
      AM     =  1005.D0
      SREF   =  0.3376D0
      T      =  6000.0D0
      GR     =  9.8D0
      TM     =  1.0D0/(2.0D0*AM)
C
      N      =  9
      M      =  NP
      TOL    =  1.D-06
      KS     =  1
      MS     =  1
      XS(1)  =  0.786818979D0

      NFILE  =  6
      WRITE(6,1000)
C
C --- Iniatial trajectory
C
      DO 100 K=1,M
        X(K)    =  FLOAT(K-1)/FLOAT(M-1)
        Y(1,K)  =  GI(K)
        Y(2,K)  =  VI(K)
        Y(3,K)  =  PI(K)
        Y(4,K)  =  AI(K)
        Y(5,K)  =  CG(K)
        Y(6,K)  =  CV(K)
        Y(7,K)  =  CP(K)
        Y(8,K)  =  CA(K)
        Y(9,K)  =  40.9D0
 100  CONTINUE
C
      KP     =  0
      ITMAX  =  20
      CALL BNDSCO(F,R,DIFSYB,X,XS,Y,WORK,IWORK,JS,N,M,MS,
     *     KS,TOL,ITMAX,KP,MMAX,MSMAX,MMS,NMS,NDW,NDIW,NFILE)
      IF(KP.LT.0) GOTO 900
C
      KP     =  0
      IFEIN  =  5
      NPLOT  =  7
      IPLOT  =  1
      ICASE  =  1
      CALL AWP(DIFSYB,F,X,XS,Y,N,M,MS,
     *         JS,MMS,NMS,IFEIN,IPLOT,KP,NFILE,NPLOT,ICASE)
C
  900 CONTINUE
 1000 FORMAT(//' TERMINAL BUNT PROBLEM:')
```

```
      STOP
      END
```

5.3.2.2 Subroutine F

The equations of motion for each subarcs are defined in this subroutine.

```
C*********************************************************************
      SUBROUTINE F(X,Y,DY,J,L,JS,MMS)
C*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION Y(*),DY(*),AROOT(2)
      INTEGER JS(MMS,*)
    COMMON /OUT/  DALP
      COMMON /ITEIL/ ITEIL
      COMMON /PARA1/ B1,B2,B3,E1,E2,E3,D1,AM,T,GR,SREF,TM,
     +               HGM,HV,HXP,HH,ALP
C ---------------------------------
      GM    = Y(1)
      V     = Y(2)
      XP    = Y(3)
      H     = Y(4)
      GL    = Y(5)
      VL    = Y(6)
      XL    = Y(7)
      HL    = Y(8)
      TF    = Y(9)
*-----------------------------------------------------
C     CONSTRAINED PART
*-----------------------------------------------------
        IF (JS(J,1).GT.0) THEN
C---------------------------------
          V2    = V*V
          RHO   = B1*H*H+B2*H+B3
          DRHO  = 2*B1*H+B2
          ALNN  = -8*AM*GR/(21.9*RHO*V2*SREF)
          dalp  = alnn
C *******************************************
C     determine mu_1
C *******************************************
          A      = ((E1*ALNN+E2)*ALNN+E3)*0.5*RHO*V2*SREF
          AA     = (E1*2*ALNN+E2)*0.5*RHO*V2*SREF
          AN     = D1*ALNN*0.5*RHO*V2*SREF
          ANA    = D1*0.5*RHO*V2*SREF
          CGU    = DCOS(ALNN)
          SGU    = DSIN(ALNN)
          COMP1  = GL*CGU/(V*AM)-VL*SGU/AM
          COMP2  = GL*SGU/(V*AM)+VL*CGU/AM
          FAU    = (T-A+ANA)*COMP1 - (AA+AN)*COMP2
          CONC    = -(21.9*RHO*V2*SREF)/(2*AM*GR)
          CMU1    = -FAU/CONC
C *******************************************
C     end determine mu_2
C *******************************************
          CG    = DCOS(GM)
          SG    = DSIN(GM)
```

```
*-------------------------------------------------
*      DYNAMIC EQUATIONS
*-------------------------------------------------
           DY(1) = TF*(((T-A)*SGU+AN*CGU)/AM-GR*CG)/V
           DY(2) = TF*(((T-A)*CGU-AN*SGU)/AM-GR*SG)
           DY(3) = TF*V*CG
           DY(4) = TF*V*SG
           DY(5) = -TF*(GL*GR*SG/V-VL*GR*CG-XL*V*SG+HL*V*CG)
*-------------------------------------------------
           VLC1  = GL*(-T*SA/(V2*AM)-(ACOEF*SA-D1*ALP*CA)*RHO*SREF*TM
     +          +GR*CG/V2)
           VLC2  = -VL*(ACOEF*CA+D1*ALP*SA)*RHO*V*SREF/AM
           VLC3  = -CMU1*(21.9*ALNN*RHO*V*SREF)/(AM*GR)
*-------------------------------------------------
           DY(6) = -TF*(VLC1+VLC2+XL*CG+HL*SG+VLC3)
           DY(7) = 0.D0
*-------------------------------------------------
           DRHO  = 2*H*B1+B2
           HLC1  = GL*(-ACOEF*SA+D1*ALP*CA)*V*SREF*DRHO*TM
           HLC2  = -VL*(ACOEF*CA+D1*ALP*SA)*V2*SREF*DRHO*TM
           HLC3  = -CMU1*(21.9*ALNN*0.5*DRHO*V2*SREF)/(AM*GR)
           DY(8) = -TF*(HLC1+HLC2+HLC3)
           DY(9) = 0.D0
*-------------------------------------------------
*-------------------------------------------------
         Else
C-------------------------------------------------
C      UNCONSTRAINED PART
C-------------------------------------------------
           RHO   = B1*H*H+B2*H+B3
           DRHO  = 2*B1*H+B2
           GALP  = 0.02
           V2    = V*V

c
c----- Newton Raphson -----------------
c
           DO 12 I=1,100
             A      = ((E1*GALP+E2)*GALP+E3)*0.5*RHO*V2*SREF
             AA     = (E1*2*GALP+E2)*0.5*RHO*V2*SREF
             AN     = D1*GALP*0.5*RHO*V2*SREF
             ANA     = D1*0.5*RHO*V2*SREF
             CGU    = DCOS(GALP)
             SGU    = DSIN(GALP)
             COMP1  = GL*CGU/(V*AM)-VL*SGU/AM
             COMP2  = GL*SGU/(V*AM)+VL*CGU/AM
             FA     = (T-A+ANA)*COMP1 - (AA+AN)*COMP2
**************************************************
             AAPANX = (E1*2+D1)*0.5*RHO*V2*SREF
**************************************************
***      Derivative FA wrt alpha                  *******
**************************************************
             FAP    = -AA *COMP1 - (AA+AN)*COMP1-(T-A+ANA)*COMP2-AAPANX*COMP2
             XRN    = GALP-FA/FAP
             ceck   = abs(XRN-GALP)
             toln    = 1.0E-9
             if (ceck.le.toln) goto 10
             GALP = XRN
 12        enddo
```

```
 10       ALP    = GALP
          DALP   = ALP
          CA     = DCOS(ALP)
          SA     = DSIN(ALP)
          ACOEF  = (E1*ALP+E2)*ALP+E3
          ALP2   = ALP*ALP
C ---------------------------------
          AEQ    = ACOEF*0.5*RHO*V2*SREF
          ANO    = D1*ALP*0.5*RHO*V2*SREF
          CG     = DCOS(GM)
          SG     = DSIN(GM)
*-------------------------------------------------
*     DYNAMIC EQUATIONS
*-------------------------------------------------
          DY(1) = TF*(((T-AEQ)*SA+ANO*CA)/AM-GR*CG)/V
          DY(2) = TF*(((T-AEQ)*CA-ANO*SA)/AM-GR*SG)
          DY(3) = TF*V*CG
          DY(4) = TF*V*SG
          DY(5) = -TF*(GL*GR*SG/V-VL*GR*CG-XL*V*SG+HL*V*CG)
*-------------------------------------------------
          VLC1  = GL*(-T*SA/(V2*AM)-(ACOEF*SA-D1*ALP*CA)*RHO*SREF*TM
     +         +GR*CG/V2)
          VLC2  = -VL*(ACOEF*CA+D1*ALP*SA)*RHO*V*SREF/AM
*-------------------------------------------------
          DY(6) = -TF*(VLC1+VLC2+XL*CG+HL*SG)
          DY(7) = 0.D0
*-------------------------------------------------
          DRHO  = 2*H*B1+B2
          HLC1  = GL*(-ACOEF*SA+D1*ALP*CA)*V*SREF*DRHO*TM
          HLC2  = -VL*(ACOEF*CA+D1*ALP*SA)*V2*SREF*DRHO*TM
          DY(8) = -TF*(HLC1+HLC2)
          DY(9) = 0.D0
      endif
*-------------------------------------------------
      RETURN
      END
```

5.3.2.3 Subroutine R

Subroutine R defines the boundary, jump and switching conditions.

```
C*********************************************************************
      SUBROUTINE R(YA,YB,ZZ,W,NYA,NSK,J,L,LS,JS,MMS)
C*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION YA(*),YB(*),ZZ(*),W(*)
      INTEGER NYA(*),NSK(*)
      INTEGER JS(MMS,*)
      COMMON /ITEIL/ ITEIL
      COMMON /OUT/  DALP
      COMMON /PARA1/ B1,B2,B3,E1,E2,E3,D1,AM,T,GR,SREF,TM,
     +               HGM,HV,HXP,HH,ALP
C
      GTF    =  -1.57D0
      V0     =  272.0D0
      VTF    =  310.0D0
```

```
      H0     =  30.0D0
      XF     =  10000.0D0
      HF     =  0.0D0
C
C---- Boundary conditions
C
      W(1)   =  YA(1)
      W(2)   =  YA(2)/V0 - 1.D0
      W(3)   =  YA(3)
      W(4)   =  YA(4)/H0 - 1.D0
      W(5)   =  YB(1)/GTF - 1.D0
      W(6)   =  YB(2)/VTF - 1.D0
      W(7)   =  YB(3)/XF - 1.D0
      W(8)   =  YB(4)
C
      NYA(1) =  1
      NYA(2) =  2
      NYA(3) =  3
      NYA(4) =  4
C*****************************************
C   Hamiltonian at final time
C*****************************************
      YB2    = YB(2)*YB(2)
      ROI    = B1*YB(4)*YB(4)+B2*YB(4)+B3
      DROI   = 2*B1*YB(4)+B2
      ALPI   = -8*AM*GR/(21.9*ROI*YB2*SREF)
C *********************************************
C     determine mu_1
C *********************************************
      A      = ((E1*ALPI+E2)*ALPI+E3)*0.5*ROI*YB2*SREF
      AA     = (E1*2*ALPI+E2)*0.5*ROI*YB2*SREF
      AN     = D1*ALPI*0.5*ROI*YB2*SREF
      ANA    = D1*0.5*ROI*YB2*SREF
      CGU    = DCOS(ALPI)
      SGU    = DSIN(ALPI)
      COMP1  = YB(5)*CGU/(YB(2)*AM)-YB(6)*SGU/AM
      COMP2  = YB(5)*SGU/(YB(2)*AM)+YB(6)*CGU/AM
      FAU    = (T-A+ANA)*COMP1 - (AA+AN)*COMP2
      CONC   = -(21.9*ROI*YB2*SREF)/(2*AM*GR)
      CMU1   = -FAU/CONC
C *********************************************
C     end determine mu_1
C *********************************************
C
      CAI     = DCOS(ALPI)
      SAI     = DSIN(ALPI)
      ACOEFI  = (E1*ALPI+E2)*ALPI+E3
C
      ATFRI  = ACOEFI*0.5*ROI*YB2*SREF
      ANTFI  = D1*ALPI*0.5*ROI*YB2*SREF
      SGTFI  = DSIN(YB(1))
      CGTFI  = DCOS(YB(1))
      HGMI   = (((T-ATFRI)*SAI+ANTFI*CAI)/AM-GR*CGTFI)/YB(2)
      HVRI   = ((T-ATFRI)*CAI-ANTFI*SAI)/AM-GR*SGTFI
      HXPRI  = YB(2)*CGTFI
      HHRI   = YB(2)*SGTFI
      COMU   = -CMU1*((21.9*ALPI*0.5*DROI*YB2*SREF)/(AM*GR)+4.D0)
      CONA   = YB(5)*HGMI+YB(6)*HVRI+YB(7)*HXPRI+YB(8)*HHRI+COMU+1
C
```

```
C***********************************************
C   Determine alpha using Newton at Initial
C***********************************************
      YA2    = YA(2)*YA(2)
      RHO    = B1*YA(4)*YA(4)+B2*YA(4)+B3
      DRHO   = 2*B1*YA(4)+B2
      GALP   = 0.02
      DO 12 I=1,100
        A       = (-0.328*GALP*GALP-0.0253*GALP+0.03982)*RHO*YA2
        AA      = (-0.328*2*GALP-0.0253)*RHO*YA2
        AN      = 3.69672*GALP*RHO*YA2
        ANA     = 3.69672*RHO*YA2
        CGU     = DCOS(GALP)
        SGU     = DSIN(GALP)
        COMP1   = YA(5)*CGU/(YA(2)*AM)-YA(6)*SGU/AM
        COMP2   = YA(5)*SGU/(YA(2)*AM)+YA(6)*CGU/AM
        FA      = (T-A+ANA)*COMP1 - (AA+AN)*COMP2
*************************************************************************
        AX      = (-0.656*GALP-0.0253)*RHO*YA2
        AAPANX  = 3.04072*RHO*YA2
*************************************************************************
***                      {derivative FA wrt alpha}                    *******
*************************************************************************
        FAP     = -AX *COMP1 - (AA+AN)*COMP1-(T-A+ANA)*COMP2-AAPANX*COMP2
        XRN     = GALP-FA/FAP
        ceck    = abs(XRN-GALP)
        toln     = 1.0E-9
        if (ceck.le.toln) goto 10
        GALP = XRN
 12   enddo

 10   ALPB     = GALP
C***************************************
c   Hamiltonian at initial time
C***************************************
      CAR      = DCOS(ALPB)
      SAR      = DSIN(ALPB)
      ACOEFR = (E1*ALPB+E2)*ALPB+E3

      ATFR    = ACOEFR*0.5*RHO*YA(2)*YA(2)*SREF
      ANTFR   = D1*ALP*0.5*RHO*YA(2)*YA(2)*SREF
      SGTFR   = DSIN(YA(1))
      CGTFR   = DCOS(YA(1))
      HGMR    = (((T-ATFR)*SAR+ANTFR*CAR)/AM-GR*CGTFR)/YA(2)
      HVR     = ((T-ATFR)*CAR-ANTFR*SAR)/AM-GR*SGTFR
      HXPR    = YA(2)*CGTFR
      HHR     = YA(2)*SGTFR
      CONB    = YA(5)*HGMR+YA(6)*HVR+YA(7)*HXPR+YA(8)*HHR+1

      W(9)    = CONA-CONB
C*****************************************************************
      ZZ2      = ZZ(2)*ZZ(2)
      ZRHO     = B1*ZZ(4)*ZZ(4)+B2*ZZ(4)+B3
      ALPI     = -8*AM*GR/(21.9*ZRHO*ZZ2*SREF)
      W(10)    = -(21.9*ALPI*0.5*ZRHO*ZZ2*SREF)/(AM*GR)-4.0D0
      NSK(10)  = 1
C
      RETURN
      END
```

Chapter 6

Conclusions and Recommendations

This thesis presented a study of the optimal trajectory of a generic cruise missile attacking a fixed target where the target must be struck from above, subject to missile dynamics and path constraints. Two objective functions were considered. The first objective was to minimise the exposure of the missile to anti-air defences. This resulted in a nonlinear optimal control problem where time-integrated flight altitude was minimised. The second objective was to attack the target in the fastest possible time. This led to a time-optimal control problem. The generic shape of the optimal trajectory was: level flight, climbing, diving; this combination of the three flight phases is called the bunt manoeuvre.

6.1 Three-stage Manual Hybrid Approach

In this work a combination of a direct and an indirect approach (and the relevant codes) was used resulting, in effect, in a hybrid approach. The main direct solver, DIRCOL, was used to discern the solution structure, including characteristic subarcs, constraints' activation and switching times. Whenever possible, DIRCOL results were compared with those of another direct solver, NUDOCCCS. Both codes produce initial guesses for the co-state, an essential feature to enable subsequent use of the BNDSCO code for TPBVP.

The DIRCOL/NUDOCCCS approximate solution was used to aid the qualitative analysis of the optimal trajectory and to initialise BNDSCO. The hybridisation was done manually, i.e. DIRCOL/NUDOCCCS was run first, the results analysed to help formulate an appropriate TPBVP, and then the results were fed to BNDSCO as an initial guess.

There are two main reasons for opting for the manual hybrid approach.

Firstly, the three-stage approach:

- direct solution (NLP via DIRCOL/NUDOCCCS)
- analysis (optimal control theory, TPBVP formulation)
- indirect solution (TPBVP solution via BNDSCO)

offers valuable insights into the problem, its solution structure, the role of constraints and boundary conditions. The focus of this work is trajectory shaping, i.e. not just computing a bunt manoeuvre, but exploring a family of terminal bunt problems. Thus, insights into the influence of constraints and boundary conditions on the solution structure (e.g. the number of switching points, the number of constraints active, duration of their activation) is of significant operational and engineering importance.

Secondly, trajectory shaping of the bunt manoeuvre naturally leads to a pure state constraint formulation, a difficult type of optimal control problem. The arising difficulties can be handled—if not fully resolved—due to the gradual progression of the three-stage, manual hybridisation approach.

6.2 Generating an Initial Guess: Homotopy

The main difficulty in the indirect approach is due to finding a converging initial guess for the state and co-state variables. In some problems, inserting new shooting points might help, but for complex problems this strategy may not work. In addition, continuation, or homotopy, method can be employed to overcome these difficulties (see Allgower and Georg [1]).

The homotopy methods solve the original problem via the solution of the "family" problem. It means that the original problem is embedded into a family of problems characterised by a parameter. However, the selected parameter is the crucial aspect in this method. If the selected parameter is physically appropriate and mathematically convenient, then the family of problems will include the original problem. The parameter is then used to find the solution of the original problem by correcting gradually the parameter until the required solution is achieved. The progress is done by using the previous solution as an initial guess for the next problem, starting with a parameter value for which the corresponding problem is tractable.

For a highly constrained optimal control problem, Bulirsch [99, chapter 7, page 563] has warned that the parameter must be an intrinsic parameter to the problem, as opposed to an artificial parameter. If a simple or arbitrary parameter that has nothing to do with the problem is used, one may not succeed to solve the problem. A natural parameter which is related to the problem may give a good starting solution. In addition, one may need more than one natural parameter to be used to find the solution for the original problem (see Steindl [98]).

Ehtamo et al. [34] proposed a final time as a natural parameter to solve the minimum time problems. They converted the minimum time optimal control problem into a sequence of terminal cost minimisation by fixing the terminal time. The starting point is to use a small value for the fixed final time and then by using an appropriate search procedure, gradually, the final time for the original problem is found. They applied these continuation methods for the minimum time flight of an aircraft.

Bulirsch et al. [25] and Pesch [81] proposed a combination of the Chebyshev and Bolza functional as the objective function to find the solution of the abort landing in windshear problem. Both functionals are originally derived form the minimax optimal control problem, therefore they are related naturally (see Bulirsch et al [24, page 4]). They solved the unconstrained problem and then gradually the constraints were activated until the original problem was solved.

Three different formulations might be employed to overcome the problem considered in this thesis. Firstly, consider the minimum altitude problem in Chapter 3. We can begin by solving the unconstrained problem, which means that we neglect con-

straints especially the difficult state constraint $h > h_{min}$, and retain only a constraint on the thrust T. Thus, the missile immediately dives and "hits" the ground (or "goes underground", followed by climbing and—finally—finished by diving to reach the target. The next step is finding a touch point ($h = h_{touch}$) on the minimum altitude ($\dot{h} = 0$, $\ddot{h} > 0$) on the unconstrained trajectory from the previous step. This way, the difficult state constraint $h > h_{min}$ is put back into the problem formulation, albeit with lower h_{min} than required, i.e. with $h = h_{touch}$. This introduction of a touch point results in computation similar to the unconstrained case, but a new switching point must be introduced. The following steps consist in gradual "lifting" the minimum altitude constraint from h_{touch}, activating a boundary arc starting from a short boundary arc close to the touch point ($h = h_{touch} + \Delta h$), and—finally—by increasing Δh the problem can be solved for a fully constrained formulation.

Another approach would be to retain all the constraints, but to shorten the distance between the launch point and the target. Then the down-range is squeezed so that the missile directly climbs, which avoids the activation of the minimum altitude constraint. The next step is to increase the down-range gradually, activating the constraint at a point, a short arc etc, until the original problem is solved. Effectively, this simply is homotopy on $x(t_f)$, one terminal condition.

Finally, a new objective function can be introduced which is a combination of both cases, minimum altitude and minimum time. Consider the following new objective function

$$J = \left[\frac{(1-\varepsilon)}{\Omega} t_f + \frac{\varepsilon}{\Omega} \int_0^{t_f} h dt \right] \tag{6.1}$$

where Ω has a big value and $0 \leq \varepsilon \leq 1$. This approach starts by solving the minimum time problem where the minimum altitude constraint is not active ($\varepsilon = 0$). Then the original problem is solved by gradually increasing ε.

6.3 Pure State Constraint and Multiobjective Formulation

As explained in Chapter 1, a guidance strategy based on trajectory shaping lends itself to optimal control interpretation. In the context of a cruise missile required to perform the terminal bunt manoeuvre the resulting optimal control leads naturally to pure state constraints. The presence of such constraints is a source of significant difficulties for the indirect method approach to solution, as explained in Section 2.1 and illustrated in Section 3.4. Indeed, even the direct method solver NUDOCCCS had convergence problems in that case, so it was not possible to use its approximate solution to compare it with that of DIRCOL and use it to initialise BNDSCO. Theoretically, there are a few ways of dealing with the pure state constraint, see e.g. Section 3.4.2, but none of them offers a magic bullet and, besides, the theoretical results must be numerically practical which is not always the case (see Maurer and Gillesen [74, Page 111]).

Thus, pure state constraints are best avoided, but can they be—they arise naturally for the terminal bunt problem? This leads to the issue of problem formulation.

From the user (operational) point of view, it is natural to impose inequality constraints on state variables—it is intuitively clear and practically desirable to limit the missile altitude and speed. It is also transparent to have a simple performance index, e.g. penalise altitude or flight time only, for it is obvious what it means, and its value is meaningful for the user. Moreover, if a parametric study is conducted, say, by varying the terminal speed, the resulting changes in the performance index are easy to comprehend.

From the analyst (computational) point of view, it is more desirable to have as few constraints as possible, particularly if they are path constraints and especially if a pure state constraint is involved. On the other hand, the performance index can be complex, because this can be handled easily.

How, then, can one practically reconcile the user's preference for many constraints and a simple performance index with the analyst's desire for the converse?

The key observation is that, for a given problem, the user would benefit more from *establishing* bounds on the state variables rather than *imposing* them a priori. Indeed

for all the cases in this study the missile speed constraints are never activated, but must be included when the relevant TPBVP is formulated. Therefore, it would be perhaps more desirable to make finding bounds on the states a part of the optimisation process at the outset. How can this be done? Including states in the performance index in order to minimise then together with, say, flight time is not satisfactory, for the resulting combination of disparate variables produces a scalar measure of performance lacking transparency (what would be the units and meaning of such a performance index?). However, a vector performance measure preserves transparency by having separate entries for each variable of interest thus allowing natural units and avoiding mixing of incommensurable variables.

The resulting multiobjective formulation [71, 83] can be handled computationally and has at its core partial ordering of the vector performance index. The solution is expressed through a Pareto set which contains trade-off points only. For example, if there are two objectives to be minimised simultaneously, say, the flight time t_f and the time-integrated square deviation from minimum altitude $\delta = \int (h - h_{min})^2 dt$, then the Pareto set will be comprised of pairs (t_f^*, δ^*) such that any further minimisation of t_f^* would increase (worsen) δ^* and conversely. Analysing the pairs (t_f^*, δ^*) would give the user insights into inherent trade-offs between the duration of the attack and the exposure to anti-air defences. Computationally, replacing the state constraint $h \geqslant h_{min}$ with the second objective will simplify problem solution considerably. If δ^* solutions result in $h < h_{min}$ subarcs, the integrand $(h - h_{min})^2$ can be modified to include a suitable barrier function for penalising the $h < h_{min}$ solution more than the $h \geq h_{min}$ ones.

6.4 Summary and Discussion

A three-stage hybrid approach was used to explore the terminal bunt problem. The main benefit of the approach is that it produces valuable insights into the problem by forcing the user to understand the structure of the boundary value problem and the corresponding switching structure of the optimal trajectory. However, the occurrence of pure state inequality constraints caused difficulties during computation.

To overcome the above problem a homotopy method might be implemented to generate an initial guess for the state and co-state variables in the indirect multiple shooting. But finding the family of the problem is a non-trivial task.

As an alternative multiple objective optimal control might be considered. This method might be used to reinterpret the pure state constraint as an entry in the vector performance index. However, solving the resulting multiobjective optimal control problem adds another layer of complexity. Also, practice may be needed to understand and interpret multiobjective results.

6. CONCLUSIONS AND RECOMMENDATIONS

Appendix A

BNDSCO Benchmark Example

This BNDSCO benchmark example is taken from Bryson and Ho [23] and is worked out in detail below as a complement to the implementation issues covered in Section 5.3. Consider the following state variable inequality constraint problem. Let x and v define the position and velocity of a particle in rectilinear motion. Find the control history $a(t), 0 \leq t \leq 1$, such that the objective function

$$\mathcal{J} := \int_0^1 \frac{1}{2} a^2(t) dt \tag{A.1}$$

is minimised subject to the following constraints:

- dynamic equation

$$\dot{x} = v \tag{A.2}$$

$$\dot{v} = a \tag{A.3}$$

- boundary conditions

$$x(0) = x(1) = 0 \tag{A.4}$$

$$v(0) = -v(1) = 1, \tag{A.5}$$

- state inequality constraint

$$x(t) \leq l \quad \text{for} \quad 0 \leq t \leq 1. \tag{A.6}$$

A.1 Analytic Solution

By introducing an additional state variable z, the problem can be transformed into a Mayer problem as follows:

$$\min \mathcal{J} := z(1) \tag{A.7}$$

subject to the constraints

- dynamic equation

$$\dot{x} = v \tag{A.8a}$$
$$\dot{v} = a \tag{A.8b}$$
$$\dot{z} = \frac{1}{2}a^2 \tag{A.8c}$$

- boundary conditions

$$x(0) = x(1) = 0, \tag{A.9}$$
$$v(0) = -v(1) = 1, \tag{A.10}$$
$$z(0) = 0 \tag{A.11}$$

and the state inequality constraint (A.6) remains the same. Following the BNDSCO manual notation [76], let $y := (x, v, z)$ and the Hamiltonian can be defined by

$$H = \lambda_x v + \lambda_v a + \frac{1}{2}\lambda_z a^2. \tag{A.12}$$

The co-state equations can be given by

$$\dot{\lambda} = -H_y^T = \begin{bmatrix} \lambda_x \\ \lambda_v \\ \lambda_z \end{bmatrix} = \begin{bmatrix} 0 \\ -\lambda_x \\ 0 \end{bmatrix} \tag{A.13}$$

The stationary condition is given by

$$\frac{\partial H}{\partial a} = \lambda_v + \lambda_z a = 0 \rightarrow a = \frac{-\lambda_v}{\lambda_z}. \tag{A.14}$$

From transversality condition we obtain

$$\lambda_z(T) = \frac{\partial \Phi}{\partial z(T)} = 1. \tag{A.15}$$

Substituting equation (A.15) to equation (A.14) gives $a = -\lambda_v$. Consider now the case when the state inequality constraint $S = x - l$ is active. The first and second total time derivative of S yield

$$S = x - l \tag{A.16}$$

$$\dot{S} = \dot{x} = v \tag{A.17}$$

$$\ddot{S} = \dot{v} = a \tag{A.18}$$

Thus, the state inequality constraint is a second-order constraint. The solution can be a touch point or a constrained arc depending on the value of l. In this example, the constrained arc problem will be discussed.

Based on the second order state constraints (A.16), the Hamiltonian is given by

$$H^{svic} = \lambda_x v + \lambda_v a + \frac{1}{2}\lambda_z a^2 + \mu a. \tag{A.19}$$

The stationary condition yields:

- unconstrained

$$\left.\begin{array}{c} H_a^{svic} = 0 \\ \mu = 0 \end{array}\right\} \quad a = -\frac{\lambda_v}{\lambda_z}, \quad \mu = 0 \tag{A.20}$$

- constrained

$$\left.\begin{array}{c} H_a^{svic} = 0 \\ \ddot{S} = 0 \end{array}\right\} \quad a = 0, \quad \mu = -\lambda_v \tag{A.21}$$

At the beginning of the constrained arc, we obtain

$$N(x(t_1)) = \begin{bmatrix} x - l \\ v \end{bmatrix} = 0 \tag{A.22}$$

Based on the equation 20, p. 21 of [76], the discontinuity at the jump conditions at the entry point t_1 is given

$$\lambda(t_1^+)^T = \lambda(t_1^-)^T - \sigma^T N_y = \lambda(t_1^-)^T - [\sigma_1, \sigma_2, 0] \tag{A.23}$$

The Hamiltonian at the both switching points is continuous

$$H(t_i^+) = H(t_i^-), \quad i = 1, 2 \tag{A.24}$$

A.1.1 Unconstrained or Free Arc ($l \geq 1/4$)

In this case the constraint $x \leq l$ is not active along the optimal trajectory. Based on equations (A.8) and (A.13) the differential equations can be given by

$$\dot{x} = v \quad \rightarrow \quad \text{DY(1) = Y(2)} \tag{A.25a}$$

$$\dot{v} = a = -\lambda_v \quad \rightarrow \quad \text{DY(2) = -Y(4)} \tag{A.25b}$$

$$\dot{\lambda}_x = 0 \quad \rightarrow \quad \text{DY(3) = 0.0D0} \tag{A.25c}$$

$$\dot{\lambda}_v = -\lambda_x \quad \rightarrow \quad \text{DY(4) = -Y(3)} \tag{A.25d}$$

The above equations (A.25) correspond to equation (2.91a) and must be implemented in **subroutine F**. The initial and final conditions are given by

$$x(0) = 0 \quad \rightarrow \quad \text{W(1) = YA(1) - X0} \quad \rightarrow \quad \text{NYA(1) = 1} \tag{A.26a}$$

$$x(1) = 0 \quad \rightarrow \quad \text{W(3) = YB(1) - XF} \tag{A.26b}$$

$$v(0) = 1 \quad \rightarrow \quad \text{W(2) = YA(2) - V0} \quad \rightarrow \quad \text{NYA(2) = 2} \tag{A.26c}$$

$$v(1) = -1 \quad \rightarrow \quad \text{W(4) = YB(2) - VF} \tag{A.26d}$$

The equations (A.26) correspond to equation (2.91c) and must be implemented in **subroutine R** and the corresponding initial conditions have to be marked by defining **NYA** (see page 51 [76] for more detail).

The following are the notation based on equation (2.91). The state and co-state variables are ($N = 4$):

$$y = \begin{bmatrix} x \\ v \\ \lambda_x \\ \lambda_v \end{bmatrix} \begin{matrix} \rightarrow y_1 \\ \rightarrow y_2 \\ \rightarrow y_3 \\ \rightarrow y_4 \end{matrix} \tag{A.27}$$

The boundary conditions can be defined as

$$r(y(0), y(1)) = \begin{bmatrix} x(0) - 0 \\ v(0) - 1 \\ x(1) - 0 \\ v(1) + 0 \end{bmatrix} \begin{matrix} \rightarrow r_1 \\ \rightarrow r_2 \\ \rightarrow r_3 \\ \rightarrow r_4 \end{matrix} \begin{matrix} \rightarrow \text{W(1)} \\ \rightarrow \text{W(2)} \\ \rightarrow \text{W(3)} \\ \rightarrow \text{W(4)} \end{matrix} \tag{A.28}$$

where r_1, r_2, r_3 and r_4 are initial and final conditions.

A.1.2 Touch Point Case ($1/6 \leq l \leq 1/4$)

In this case the optimal trajectory for x touches the constraint at only one point. The differential equations are:

$$\dot{x} = v \quad \rightarrow \quad \text{DY(1) = Y(2)} \tag{A.29a}$$

$$\dot{v} = a = -\lambda_v \quad \rightarrow \quad \text{DY(2) = -Y(4)} \tag{A.29b}$$

$$\dot{\lambda}_x = 0 \quad \rightarrow \quad \text{DY(3) = 0.0D0} \tag{A.29c}$$

$$\dot{\lambda}_v = -\lambda_x \quad \rightarrow \quad \text{DY(4) = -Y(3)} \tag{A.29d}$$

$$\dot{l}_0 = 0 \quad \rightarrow \quad \text{DY(5) = 0.0D0} \tag{A.29e}$$

The boundary conditions are:

$$x(0) = 0 \quad \rightarrow \quad \text{W(1) = YA(1) - X0} \quad \rightarrow \quad \text{NYA(1) = 1} \tag{A.30a}$$

$$x(1) = 0 \quad \rightarrow \quad \text{W(3) = YB(1) - XF} \tag{A.30b}$$

$$v(0) = 1 \quad \rightarrow \quad \text{W(2) = YA(2) - V0} \quad \rightarrow \quad \text{NYA(2) = 2} \tag{A.30c}$$

$$v(1) = -1 \quad \rightarrow \quad \text{W(4) = YB(2) - VF} \tag{A.30d}$$

Switching and jump conditions at touch point t_b are:

$$x(t_b) = l \quad \rightarrow \text{W(5) = ZZ(1) / PAR(1) - 1.D0} \quad \rightarrow \text{NSK(5) = 1} \tag{A.31a}$$

$$v(t_b) = 0 \quad \rightarrow \text{W(6) = ZZ(2)} \quad \rightarrow \text{NSK(6) = 1} \tag{A.31b}$$

$$\lambda_x(t_b^+) = \lambda_x(t_b^-) - l_0 \quad \rightarrow \text{ZZ(3) = ZZ(3) - ZZ(5)} \tag{A.31c}$$

The equations (A.29) correspond to equation (2.91a) and must be placed in **subroutine F** while the equations (A.30) and (A.31) are in **subroutine R**. The equations (A.30) correspond to equation (2.91c) while equations (A.31a)-(A.31b) correspond to equation (2.91d). The jump condition in equation (A.31c) corresponds to equation (2.91b). In this case the user must prescribe **NYA** and **NSK** accordingly. The following are the notation based on the equation (2.91). The state and co-state variables are ($N = 5$):

$$y = \begin{bmatrix} x \\ v \\ \lambda_x \\ \lambda_v \\ l_0 \end{bmatrix} \begin{matrix} \rightarrow y_1 \\ \rightarrow y_2 \\ \rightarrow y_3 \\ \rightarrow y_4 \\ \rightarrow y_5 \end{matrix} \rightarrow \xi_1 \tag{A.32}$$

The boundary conditions can be defined as

$$r(y(0),y(1)) = \begin{bmatrix} x(0)-0 \\ v(0)-1 \\ x(1)-0 \\ v(1)+0 \\ x(t_b)-l \\ v(t_b)-0 \end{bmatrix} \begin{matrix} \rightarrow r_1 & \rightarrow \text{W(1)} \\ \rightarrow r_2 & \rightarrow \text{W(2)} \\ \rightarrow r_3 & \rightarrow \text{W(3)} \\ \rightarrow r_4 & \rightarrow \text{W(4)} \\ \rightarrow r_5 & \rightarrow \text{W(5)} \\ \rightarrow r_6 & \rightarrow \text{W(6)} \end{matrix} \tag{A.33}$$

where r_1, r_2, r_3 and r_4 are initial and final conditions while r_5 and r_6 are for switching conditions at touch point t_b.

A.1.3 Constrained Arc Case ($0 \leq l \leq 1/6$)

The constraint $x \leq l$ is active in the optimal trajectory. The constrained arc is active on the state variable x for a finite time $t_1 \leq t \leq t_2$. The differential equations can be written as

$$\dot{x} = v \rightarrow \text{DY(1) = Y(2)} \tag{A.34a}$$
$$\dot{v} = a = -\lambda_v \rightarrow \text{DY(2) = -Y(4)} \tag{A.34b}$$
$$\dot{\lambda}_x = 0 \rightarrow \text{DY(3) = 0.0D0} \tag{A.34c}$$
$$\dot{\lambda}_v = -\lambda_x \rightarrow \text{DY(4) = -Y(3)} \tag{A.34d}$$
$$\dot{\sigma}_1 = 0 \rightarrow \text{DY(5) = 0.0D0} \tag{A.34e}$$
$$\dot{\sigma}_2 = 0 \rightarrow \text{DY(6) = 0.0D0} \tag{A.34f}$$

The above equations (A.34) correspond to equation (2.91a) and must be put into **subroutine F**. The initial and final conditions based on equations (2.91c) can be given by

$$x(0) = 0 \rightarrow \text{W(1) = YA(1) - X0} \rightarrow \text{NYA(1) = 1} \tag{A.35a}$$
$$x(1) = 0 \rightarrow \text{W(3) = YB(1) - XF} \tag{A.35b}$$
$$v(0) = 1 \rightarrow \text{W(2) = YA(2) - V0} \rightarrow \text{NYA(2) = 2} \tag{A.35c}$$
$$v(1) = -1 \rightarrow \text{W(4) = YB(2) - VF} \tag{A.35d}$$

The switching and jump conditions at t_1 are:

$$\begin{aligned}
x(t_1) &= l \quad \rightarrow \text{W(5)} = \text{ZZ(1)} / \text{PAR(1)} - 1.\text{D0} \rightarrow \text{NSK(5)} = 1 \quad \text{(A.36a)} \\
v(t_1) &= 0 \quad \rightarrow \text{W(6)} = \text{ZZ(2)} \rightarrow \text{NSK(6)} = 1 \quad \text{(A.36b)} \\
\lambda_v(t_1^-) &= 0 \quad \rightarrow \text{W(7)} = \text{ZZ(4)} \rightarrow \text{NSK(7)} = 1 \quad \text{(A.36c)} \\
\lambda_x(t_1^+) &= \lambda_x(t_1^-) - \sigma_1 \rightarrow \text{ZZ(3)} = \text{ZZ(3)} - \text{ZZ(5)} \quad \text{(A.36d)} \\
\lambda_v(t_1^+) &= \lambda_v(t_1^-) - \sigma_2 \rightarrow \text{ZZ(4)} = \text{ZZ(4)} - \text{ZZ(6)} \quad \text{(A.36e)}
\end{aligned}$$

and the switching conditions at t_2 are:

$$\lambda_v(t_2) = 0 \quad \rightarrow \quad \text{W(8)} = \text{ZZ(4)} \quad \rightarrow \quad \text{NSK(8)} = 2 \qquad \text{(A.37)}$$

Equations (A.36a)–(A.36c) and (A.37) correspond to equation (2.91d), while equations (A.36d)–(A.36e) correspond to equation (2.91b). Equations (A.35), (A.36) and (A.37) must be placed in **subroutine R**. The initial conditions, switching point and jump conditions must be prescribed in **NYA** and **NSK**.

The following are the notation based on the equation (2.91). The state and co-state variables are ($N = 6$):

$$y = \begin{bmatrix} x \\ v \\ \lambda_x \\ \lambda_v \\ \sigma_1 \\ \sigma_2 \end{bmatrix} \begin{matrix} \rightarrow y_1 \\ \rightarrow y_2 \\ \rightarrow y_3 \\ \rightarrow y_4 \\ \rightarrow y_5 \quad \rightarrow \xi_1 \\ \rightarrow y_6 \quad \rightarrow \xi_2 \end{matrix} \qquad \text{(A.38)}$$

The boundary conditions can be defined as

$$r(y(0), y(1)) = \begin{bmatrix} x(0) - 0 \\ v(0) - 1 \\ x(0) - 0 \\ v(0) + 0 \\ x(t_1) - l \\ v(t_1) - 0 \\ \lambda_v(t_1^-) - 0 \\ \lambda_v(t_2) - 0 \end{bmatrix} \begin{matrix} \rightarrow r_1 \quad \rightarrow \text{W(1)} \\ \rightarrow r_2 \quad \rightarrow \text{W(2)} \\ \rightarrow r_3 \quad \rightarrow \text{W(3)} \\ \rightarrow r_4 \quad \rightarrow \text{W(4)} \\ \rightarrow r_5 \quad \rightarrow \text{W(5)} \\ \rightarrow r_6 \quad \rightarrow \text{W(6)} \\ \rightarrow r_7 \quad \rightarrow \text{W(7)} \\ \rightarrow r_8 \quad \rightarrow \text{W(8)} \end{matrix} \qquad \text{(A.39)}$$

where r_1, r_2, r_3 and r_4 are initial and final conditions while r_5, r_6, r_7 and r_8 are for switching conditions at t_1 and t_2.

A. BNDSCO BENCHMARK EXAMPLE

Appendix B

Computational Results

B.1 Initial Altitude Above h_{min}

The DIRCOL implementation for the missile launches above the minimum altitude h_{min} is presented. Figure B.1 shows that the missile dives directly after launching, so that the excess of the altitude is minimised. The diving manoeuvre is ended when the missile hits the minimum altitude h_{min}. The missile then flies on the minimum altitude until it has to start climbing in order to hit the target from above. Then the same structure of the optimal trajectory will follow as explained in Section 3.2. In this simulation the missile is assumed to be launched horizontally ($\gamma_0 = 0$) by varying the initial altitude h_0. The initial and final conditions can be given as follows.

The initial conditions are:

$$\begin{aligned} \gamma_0 &= 0 \text{ deg}, \\ V_0 &= 272 \text{ m/s}, \\ x_0 &= 0 \text{ m}, \\ h_0 &= 100, 200, 500 \text{ m}. \end{aligned}$$

The final conditions are:

$$\begin{aligned} \gamma_{t_f} &= -90 \text{ deg}, \\ V_{t_f} &= 270 \text{ m/s}, \\ x_{t_f} &= 10000 \text{ m}, \\ h_{t_f} &= 0 \text{ m}. \end{aligned}$$

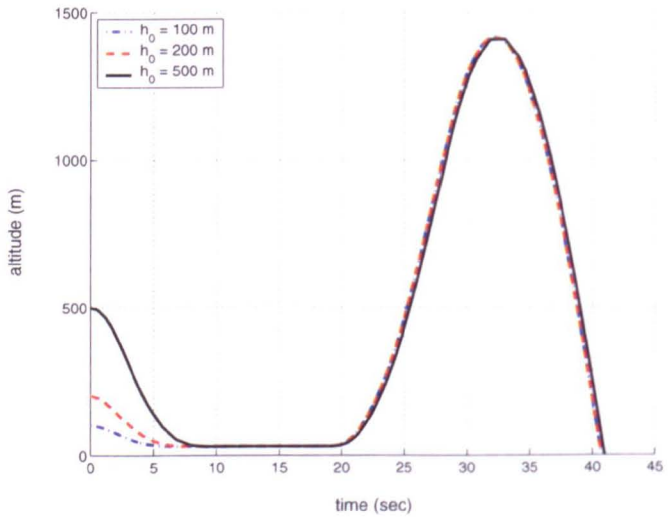


Figure B.1: Altitude versus time histories for minimum altitude problem using DIRCOL for a varying initial altitude.

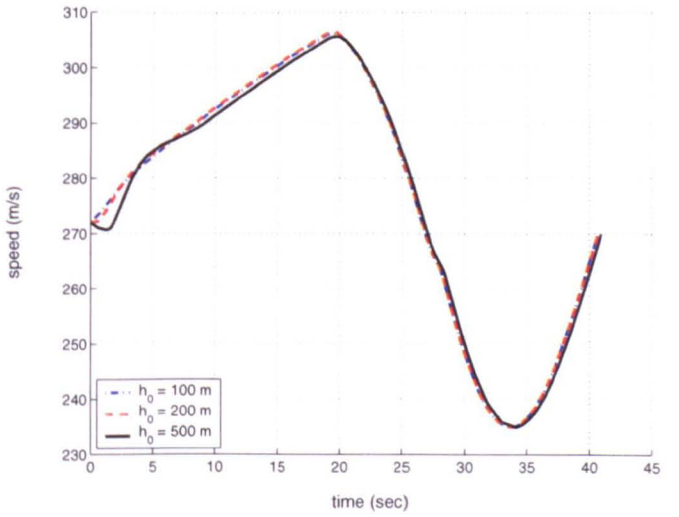


Figure B.2: Speed versus time histories for minimum altitude problem using DIRCOL for a varying initial altitude.

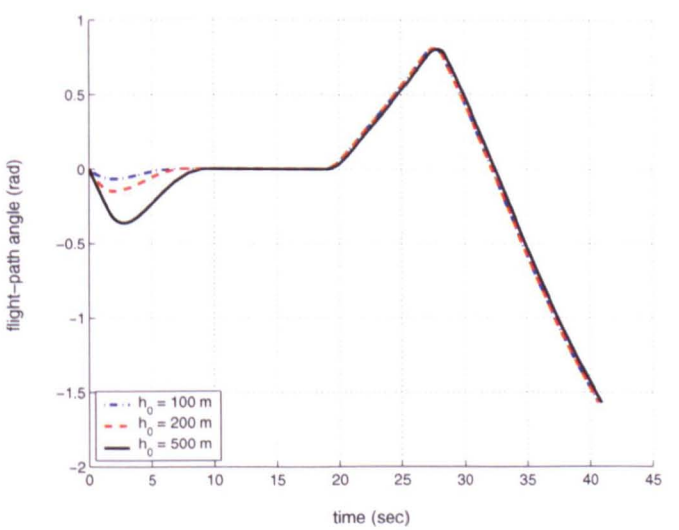


Figure B.3: Flight-path angle versus time histories for minimum altitude problem using DIRCOL for a varying initial altitude.

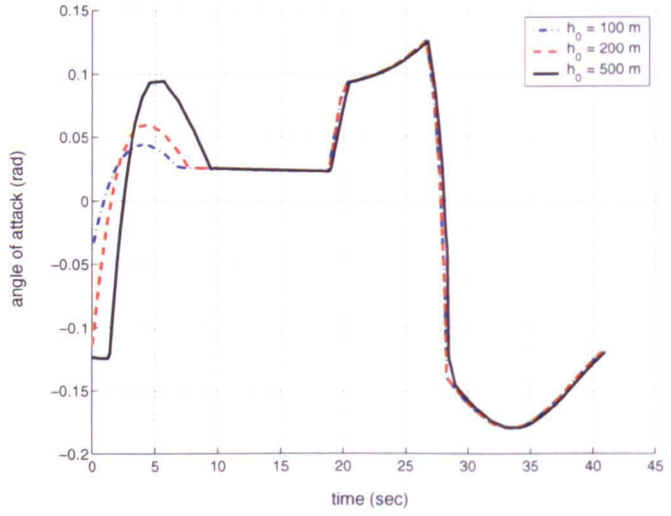


Figure B.4: Angle of attack versus time histories for minimum altitude problem using DIRCOL for a varying initial altitude.

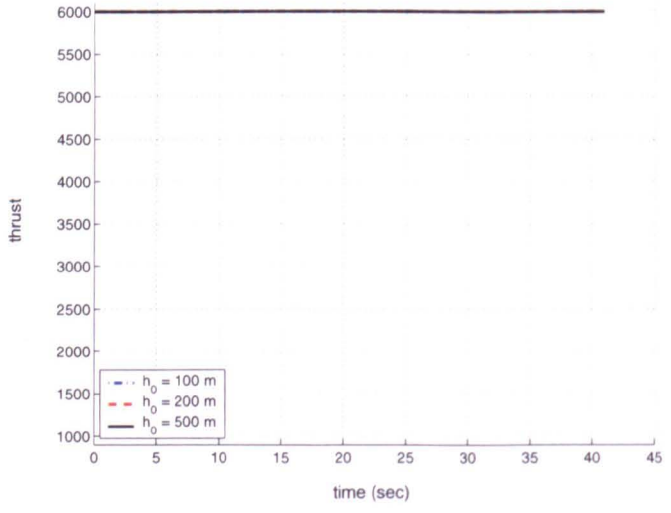


Figure B.5: Thrust versus time histories for minimum altitude problem using DIRCOL for a varying initial altitude.

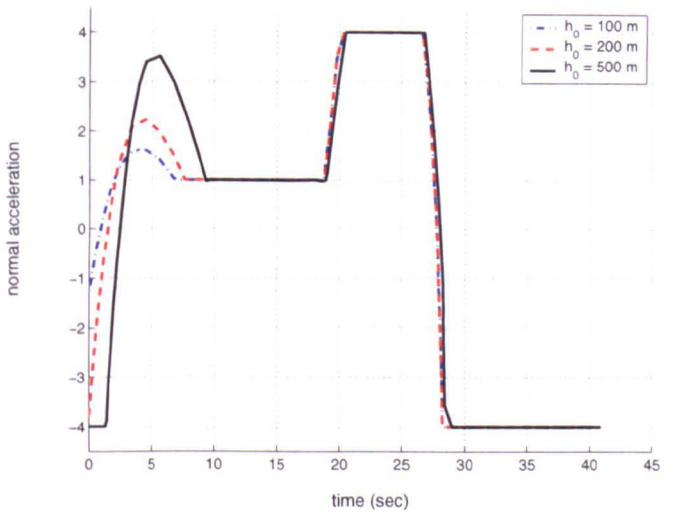


Figure B.6: Normal acceleration versus time histories for minimum altitude problem using DIRCOL for a varying initial altitude.

References

[1] E.L. Allgower and K. Georg. *Numerical Continuation Methods: An Introduction*. Springer Verlag, New York, 1990.

[2] U.M. Ascher, R.M.M. Mattheij, and R.D. Russel. *Numerical Solution of Boundary Value Problem for Ordinary Differential Equations*. SIAM, Philadelphia, 1995.

[3] M. Athans and P.L. Falb. *Optimal Control: An Introduction to Theory and Its Application*. McGraw-Hill Book Company, New York, 1966.

[4] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty. *Nonlinear Programming: Theory and Algorithms*. Wiley-Interscience, New York, 2 edition, 1993.

[5] R. Bellman. *Dynamics Programming*. University Press, Princeton, NJ, 1957.

[6] D. Benson. *A Gauss Pseudospectral Transcription for Optimal Control*. PhD thesis, Massachusetts Institute Technology, February, 2005.

[7] L.D. Berkovitz. *Optimal Control Theory*. Springer-Verlag, New York, 1974.

[8] J.T. Betts. Issues in the direct transcription of optimal control problems to sparse nonlinear programs. In R. Bulirsch and D. Kraft, editors, *Computational Optimal Control*, volume 115 of *International Series of Numerical Mathematics*, pages 3–17. Birkhäuser Verlag, Basel, 1994.

[9] J.T. Betts. Trajectory optimization using sparse sequential quadratic programming. In R. Bulirsch, D. Kraft, J. Stoer, and K.H. Well, editors, *Optimal Control: Calculus of Variations, Optimal Control Theory and Numerical Methods*, volume 111 of *International Series of Numerical Mathematics*, pages 115–128. Birkhäuser Verlag, Basel, 1994.

[10] J.T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

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

[12] J.T. Betts and W.P. Huffman. Trajectory optimization on a parallel processor. *Journal of Guidance, Control, and Dynamics*, 14(2):431–439, 1991.

[13] J.T. Betts and W.P. Huffman. Application of sparse nonlinear programming to trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 15(1):198–206, 1992.

[14] J.T. Betts and W.P. Huffman. Path-constrained trajectory optimization using sparse sequential quadratic programming. *Journal of Guidance, Control, and Dynamics*, 16(1):59–68, 1993.

[15] J.T. Betts and W.P. Huffman. Large scale parameter estimation using sparse nonlinear programming methods. *SIAM Journal on Optimization*, 14(1):223–244, 2003.

[16] J.V. Breakwell. The optimization of trajectories. *Journal of the Society for Industrial and Applied Mathematics*, 7(2):215–247, 1959.

[17] J.V. Breakwell and J.F. Dixon. Minimum-fuel rocket trajectories involving intermediate-thrust arcs. *Journal of Optimization Theory Applications*, 17:465–479, 1975.

[18] J.V. Breakwell, J.L. Speyer, and A.E. Bryson. Optimization and control of nonlinear systems using the second variation. *SIAM Journal on Control*, 1(2):193–223, 1963.

[19] R. Brusch. A nonlinear programming approach to space shuttle trajectory optimization. *Journal of Optimization Theory and Applications*, 13:94–118, 1974.

[20] A.E. Bryson. Optimal control 1950 to 1985. *IEEE Control Systems Magazine*, 16(3):26–33, 1996.

[21] A.E. Bryson and W.F. Denham. A steepest ascent method for solving optimum programming problems. *ASME Journal of Applied Mechanics*, Series E:247–257, 1962.

[22] A.E. Bryson, W.F. Denham, and S.E. Dreyfus. Optimal programming problems with inequality constraints 1: Necessary conditions for extremal solutions. *AIAA Journal*, 1(11):2544–2550, 1963.

[23] A.E. Bryson and Y.C. Ho. *Applied Optimal Control. Optimization, Estimation, and Control. Revised Printing*. Hemisphere Publishing Corporation, New York, 1975.

[24] R. Bulirsch, F. Montrone, and H. J. Pesch. Abort landing in the presence of windshear as a minimax optimal control problem, Part 1: Necessary conditions. *Journal of Optimization Theory and Applications*, 70(1):1–23, 1991.

[25] R. Bulirsch, F. Montrone, and H. J. Pesch. Abort landing in the presence of windshear as a minimax optimal control problem, Part 2: Multiple shooting and homotopy. *Journal of Optimization Theory and Applications*, 70(2):223–254, 1991.

[26] C. Büskens. Lösung optimaler Steuerprozesse. Lösung adjungierter Variablen. Automatische Gitterpunktsanpassung, NUDOCCCS. Technical report, Westfälischen Wilhelms–Universität Münster, 1996.

[27] C. Büskens and H.Maurer. 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*, 120(1–2):85–108, 2000.

[28] K. Chudej, C. Büskens, and T. Graf. Solution of a hard flight path optimization problem by different optimization codes. In M. Breuer, F. Durst, and C. Zenger, editors, *High Performance Scientific and Engineering Computing*, pages 289–296. Springer, 2001.

[29] J. R. Cleminson, D. R. Cooke, and P. G. Earwicker. *Robust Missile Guidance: Final progress report, Unpublished DERA Report, 1999.*

[30] D. Darling. *The Complete Book of Spaceflight from Apollo 1 to Zero Grafity.* John Wiley and Sons, New York, 2002.

[31] P. Deuflhard and G. Bader. Multiple shooting techniques revisited. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations*, volume 2 of *Progress in Scientific Computing*, pages 74–94. Birkhäuser, Boston, 1983.

[32] P. Deuflhard, H.J. Pesch, and P. Rentrop. A modified continuation method for the numerical solution of nonlinear two-point boundary value problems by shooting techniques. *Numerische Mathematik*, 26:327–343, 1976.

[33] E.D. Dickmanns and K.H. Well. Approximate solution of optimal control problems using third order Hermite polynomial functions. In *Proceedings of the IFIP Technical Conference*, pages 158–166. Springer-Verlag, 1974.

[34] H. Ehtamo, T. Raivio, and R.P. Hämäläinen. A continuation method for minimum time problems. Technical report, System Analysis Laboratory, Helsinki University of Technology, March 2000.

[35] G.M. Elnagar and M. Razzaghi. A collocation-type method for linear quadratic optimal control problems. *Optimal Control Applications & Methods*, 18(3):227–235, 1997.

[36] G.N. Elnagar. State-control spectral Chebyshev parameterization for linearly constrained quadratic optimal control problems. *Journal of Computational and Applied Mathematics*, 79(1):19–40, 1997.

[37] G.N. Elnagar and M. A. Kazemi. Pseudospectral Chebyshev optimal control of constrained nonlinear dynamical systems. *Computational Optimization and Applications*, 11(2):195–217, 1998.

[38] P.J. Enright and B.A. Conway. Discrete approximations to optimal trajectories using direct transcription and nonlinear programming. *Journal of Guidance, Control, and Dynamics*, 15(4):994–1002, 1991.

[39] P.J. Enright and B.A. Conway. Optimal finite-thrust spacecraft trajectories using collocation and nonlinear programming. *Journal of Guidance, Control, and Dynamics*, 14(5):981–985, 1991.

[40] F. Fahroo and I. M. Ross. Costate estimation by a Legendre pseudospectral method. *Journal of Guidance, Control, and Dynamics*, 24(2):270–277, 2001.

[41] F. Fahroo and I. M. Ross. Direct trajectory optimization pseudospectral method. *Journal of Guidance, Control, and Dynamics*, 25(1):160–166, 2002.

[42] M. A. Kazemi G. Elnagar and M. Razzaghi. The pseudospectral Legendre method for discretizing optimal control problems. *IEEE Transactions on Automatic Control*, 40(10):1793–1796, 1995.

[43] R.V. Gamkrelidze. *Principles of Optimal Control Theory.* Plenum Press, New York, 1978.

[44] B. Garfinkel. Minimal problems in airplane performance. *Quarterly of Applied Mathematics*, 9(2), 1951.

[45] B. Garfinkel. A solution of the Goddard problem. *SIAM Journal on Control*, 1(3):349–368, 1963.

[46] M. Gerdts. Optimal control and real-time optimization of mechanical multi-body systems. *Zeitschrift für Angewandte Mathematik und Mechanik*, 83(10):705–719, 2003.

[47] P. E. Gill, W. Murray, and M. A. Saunders. Large-scale SQP methods and their application in trajectory optimization. In R. Bulirsch and D. Kraft, editors, *Computational Optimal Control*, volume 115 of *International Series of Numerical Mathematics*, pages 29–42. Birkhäuser, Basel/Boston/Berlin, 1993.

[48] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, London, 1981.

[49] P.E. Gill, W. Murray, and M.H. Wright. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12(4):979–1006, 2002.

[50] C.J. Goh and K. L. Teo. Control parametrization: A unified approach to optimal control problems with general constraints. *Automatica*, 24(1):3–18, 1988.

[51] W. Grimm and A. Markl. Adjoint estimation from a direct multiple shooting method. *Journal of Optimization Theory and Applications*, 92(2):263–283, 1997.

[52] C. R. Hargraves and S. W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal Guidance, Control, and Dynamics*, 10(4):338–342, 1987.

[53] R.F. Hartl, S.P. Sethi, and R.G. Vickson. A survey of maximum principles for optimal control problems with state constraints. *SIAM Review*, 37(2):181–218, 1995.

[54] A.L. Herman and B.A. Conway. Direct optimization using collocation based on higher-order Gauss-Lobatto quadrature rules. *Journal of Guidance, Control, and Dynamics*, 19(3):592–599, 1996.

[55] M.R. Hestenes. *Calculus of Variations and Optimal Control Theory*. Wiley, New York, 1966.

[56] G.H. Hicks and W.H. Ray. Approximation methods for optimal control synthesis. *The Canadian Journal of Chemical Engineering*, 49:522–528, 1971.

[57] D.H. Jacobson, M.M. Lele, and J.L. Speyer. New necessary conditions optimality for control problems with state-variable inequality constrains. *Journal of Mathematical Analysis and Applications*, 35:255–284, 1971.

[58] H. Jaddu and E. Shimemura. Solution of nonlinear optimal control problem using Chebyshev polynomial. In *Proceedings of the 2nd Asian Control Conference*, volume 1, pages 471–420, Seoul, Korea, 1997.

[59] H. Jaddu and E. Shimemura. Computational methods based on the state parameterization for solving constrained optimal control problems. *International Journal of Systems Science*, 30(3):275–282, 1999.

[60] H. Jaddu and E. Shimemura. Computational of optimal control trajectories using Chebyshev polynomials: Parametrization, and quadratic programming. *Optimal Control, Applications and Methods*, 20(1):21–42, 1999.

[61] H.B. Keller. *Numerical Methods for Two-Point Boundary Value Problems*. Dover Publications, New York, 1992.

[62] H.J Kelley. Methods of gradients. In G. Leitmann, editor, *Optimization Techniques with Application to Aerospace Systems*, volume 5 of *Mathematics in Science and Engineering*, pages 206–254. Academic press, New York, 1962.

[63] D.E. Kirk. *Optimal Control Theory*. Englewood Cliffs, New Jersey, 1970.

[64] D. Kraft. On converting optimal control problems into nonlinear programming problems. In K. Schittkowski, editor, *Computational Mathematical Programming*, volume 15 of *NATO ASI Series. Series F: Computer and System Science*, pages 261–280, Berlin, 1985. Springer-Verlag.

[65] D. Kraft. Algoritm 733: TOMP-Fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software*, 20(3):262–281, 1994.

[66] E. Kreindler. Additional necessary conditions for optimal control with state-variable inequality constraints. *Journal of Optimization Theory and Applications*, 38(2):241–250, 1982.

[67] R.R. Kumar and H. Seywald. Fuel-optimal station keeping via differential inclusion. *Journal of Guidance, Control, and Dynamics*, 18(5):1156–1162, 1995.

[68] D.F. Lawden. *Optimal Trajectories for Space Navigation.* Butterworths, London, 1963.

[69] G. Leitmann. *Optimization Techniques.* Academic, London, 1962.

[70] G. Leitmann. *The Calculus of Variations and Optimal Control.* Plenum Press, New York, 1981.

[71] R.R. Levary. Optimal control problems with multiple goal objective. *Optimal Control Applications & Methods*, 7(2):201–207, 1986.

[72] F.L. Lewis and V.L. Syrmos. *Optimal Control.* John Wiley & Sons, inc, New York/Chicester/Brisbane/Toronto/Singapore, 2nd edition, 1995.

[73] J. Macki and A. Strauss. *Introduction to Optimal Control Theory.* Springer-Verlag, New York/Heidelberg/Berlin, 1982.

[74] H. Maurer and W. Gillessen. Application of multiple shooting to the numerical solution of optimal control problems with bounded state variables. *Computing*, 15:105–126, 1975.

[75] D.S. Naidu. *Optimal Control Systems.* CRC Press, Florida, 2002.

[76] H.J. Oberle and W. Grimm. BNDSCO - a program for the numerical solution of optimal control problems. Technical Report DLR IB 515-89-22, Institute for Flight Systems Dynamics, DLR, Oberpfaffenhofen, Germany, 1989.

[77] M.R. Osborne. On shooting methods for boundary value problems. *Journal of Mathematical Analysis and Applications*, 27:417–433, 1969.

[78] H. J. Pesch and R. Bulirsch. The maximum principle, Bellman's equation, and Caratheodory's work. *Journal of Optimization Theory and Applications*, 80(2):199–225, 1994.

[79] H.J. Pesch. Real-time computation of feedback controls for constrained optimal control problems, Part 1: Neighboring extremals. *Optimal Control Applications & Methods*, 10(2):129–145, 1989.

[80] H.J. Pesch. Real-time computation of feedback controls for constrained optimal control problems, Part 2: A correction method based on multiple shooting. *Optimal Control Applications & Methods*, 10(2):147–171, 1989.

[81] H.J. Pesch. Offline and online computational of optimal trajectories in the aerospace field. In A. Miele and A. Salvetti, editors, *Applied Mathematics in Aerospace Science and Engineering*, Proceedings of a Meeting on Applied Mathematics in Aerospace Field, pages 165–220. Plenum Press, New York, 1991.

[82] H.J. Pesch. A practical guide to the solution of real-life optimal control problems. *Control and Cybernetics*, 23(1 and 2):7–60, 1994.

[83] L.A. Petrosyan. Strongly dynamically stable optimality principles in multicriteria optimal control problems. *Journal of Computer Systems Sciences International*, 32(2):146–150, 1994.

[84] L.S. Pontryagin, V.G. Boltyanskii, R.V. Gamkrelidze, and E.F. Mishchenko. *The Mathematical Theory of Optimal Processes*. John Wiley & Sons, New York, 1962.

[85] R. Pytlak. Runge-Kutta based procedure for the optimal control of differential-algebraic equations. *Journal of Optimization Theory and Applications*, 97(3):675–705, 1998.

[86] R. Pytlak. *Numerical Methods for Optimal Control Problems with State Constraints*, volume 1707 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, Berlin, 1999.

[87] J.B. Rosen, O.L. Mangasarian, and K. Ritter. *Nonlinear Programming*, chapter A new algorithm for unconstrained optimization. Academic Press, New York, 1970.

[88] H.H. Rosenbrock and C. Storey. *Computational Techniques for Chemical Engineers*. Pergamon, Oxford, United Kingdom, 1966.

[89] I.M. Ross and F. Fahroo. User's manual for DIDO 2002: A MATLAB application package for solving optimal control problems. Technical Report AA-02-002, Department of Aerospace and Astronautics, Naval Postgraduate School, Monterey, California, 2002.

[90] I.M. Ross and F. Fahroo. Legendre pseudospectral approximations of optimal control problems. In Mingqing Xiao Wei Kang, Carlos Borges, editor, *New Trends in Nonlinear Dynamics and Control and their Applications*, volume 295 of *Lecture Notes in Control and Information Sciences*, pages 327–343. Springer-Verlag Heidelberg, 2004.

[91] I.M. Ross and F. Fahroo. Pseudospectral knotting methods for solving optimal control problems. *Journal of Guidance, Control, and Dynamics*, 27(3):397–405, 2004.

[92] R. W. H. Sargent. The development of the SQP algorithm for nonlinear programming. In L.T. Biegler, T.F. Coleman, A.R. Conn, and F.N. Santosa, editors, *Large-scale Optimization with Application. Part II: Optimal Design and Control*, volume 93 of *The IMA Volumes in Mathematics and Its Applications*, pages 1–19. Springer-Verlag, 1997.

[93] R. W. H. Sargent. Optimal control. *Journal of Computational and Applied Mathematics*, 124(1-2):361–371, 2000.

[94] H. Seywald. Trajectory optimization based on differential inclusion. *Journal of Guidance, Control, and Dynamics*, 17(3):480–487, 1994.

[95] H. Seywald, E. M. Cliff, and K.H. Well. Range optimal trajectories for an aircraft flying in the vertical plane. *Journal of Guidance, Control, and Dynamics*, 17(2):389–398, 1994.

[96] H. Seywald and R.R. Kumar. Some recent developments in computational optimal control. *IMA Volumes in Mathematics and its Applications*, 93:203–234, 1997.

[97] H.R. Sirisena and F.S. Chou. State parameterization approach to the solution of optimal control problems. *Optimal Control Applications & Methods*, 2:289–298, 1981.

[98] A. Steindl and H. Troger. Optimal control of deployment of a tethered subsatellite. *Nonlinear Dynamics*, 31(3):257–274, 2003.

[99] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer, New York, 3rd edition, 2002.

[100] S. Subchan and R. Żbikowski. Minimum-time optimal trajectories for the terminal bunt problem. In *Proceedings AIAA Guidance, Navigation and Control Conference and Exhibition*, San Francisco, California, 2005. Paper AIAA 2005–5968.

[101] S. Subchan, R. Żbikowski, and J.R. Cleminson. Optimal trajectory for the terminal bunt problem: An analysis by the indirect method. In *Proceedings AIAA Guidance, Navigation and Control Conference and Exhibition*, volume 4, pages 3055–3065, Austin, Texas, 2003.

[102] H.J. Sussmann and J.C. Willems. 300 years of optimal control: From the brachystochrone to the maximum principle. *IEEE Control Systems Magazine*, 17(3):32–44, 1997.

[103] S. Tang and B.A. Conway. Optimization of low-thrust interplanetary trajectories using collocation and nonlinear programming. *Journal of Guidance, Control, and Dynamics*, 18(3):599–604, 1995.

[104] K. L. Teo, L.S. Jennings, H.W.J. Lee, and V. Rehbock. The control parameterization enhancing transform for constrained optimal control problems. *Journal of the Australian Mathematical Society Series B*, 40(3):314–335, 1999.

[105] K. L. Teo and K. H. Wong. Nonlinearly constrained optimal control problems. *Journal of the Australian Mathematical Society Series B*, 33(4):517–530, 1992.

[106] K.L. Teo, C. Goh, and K. Wong. *A Unified Computational Approach to Optimal Control Problems*. Longman Scientific and Technical, England, 1991.

[107] N.X. Vinh. *Optimal Trajectories in Atmospheric Flight*. Elsevier Scientific Publishing Company, Amsterdam, 1981.

[108] O. von Stryk. Numerical solution of optimal control problems by direct collocation. In R. Bulirsch, A. Miele, J. Stoer, and K.H. Well, editors, *Optimal Control, Calculus of Variations, Optimal Control Theory and Numerical Methods*, volume 111 of *International Series of Numerical Mathematics*, pages 129–143. Birkhäuser Verlag, Basel, 1993.

[109] O. von Stryk. *User's guide for DIRCOL - a direct collocation method for the numerical solution of optimal control problems*. Technische Universität Darmstad, November 1999.

[110] O. von Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37(1-4):357–373, 1992.

[111] O. von Stryk and M. Schlemmer. Optimal control of the industrial robot manutec r3. In R. Bulirsch and D. Kraft, editors, *Computational Optimal Control*, volume 115 of *International Series of Numerical Mathematics*, pages 367–382. Birkhäuser, Basel, 1994.