The Pennsylvania State University
The Graduate School
Department of Aerospace Engineering

APPLICATION OF A HOMOTOPY METHOD

TO LOW-THRUST TRAJECTORY OPTIMIZATION

A Thesis in
Aerospace Engineering

by

Chenmeng Tu

© 2008 Chenmeng Tu

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2008
The thesis of Chenmeng Tu was reviewed and approved* by the following:

Robert G. Melton
Professor of Aerospace Engineering and Director of undergraduate studies
Thesis Adviser

David B. Spencer
Associate Professor of Aerospace Engineering

George A. Lesieutre
Professor of Aerospace Engineering
Head of the Department of Aerospace Engineering

*Signatures are on file with the Graduate School.
Continuous optimization methods, historically classified into direct and indirect method categories, have demonstrated notable success in continuous-thrust trajectory optimization. An optimization problem is transformed into a nonlinear programming problem (NLP) in a direct method or a multiple-point boundary value problem (MPBVP) in an indirect method. The solution of the resulting NLP or MPBVP is obtained by numerical computation. However, the class of extremely low-thrust trajectory optimization problems creates a challenge for the direct use of those methods. Very low thrust acceleration levels (<1 mN/kg) in these problems result in lengthy trajectories and subsequent large optimization problems to solve. The sizes of these problems raise the concern of convergence and computing time. Thus, an accurate initial estimate of the solution (i.e. initial guess) becomes highly valued to reduce the search effort. However, such an accurate initial guess may be unavailable or difficult to construct.

A homotopy optimization method obtains the solution of one problem from a known solution from another problem. In its application to trajectory optimization, the optimal trajectory of one mission scenario can be derived from a known optimal trajectory of a similar scenario, which creates a means of generating initial guess for successive, more complex problems. This thesis develops a homotopy method which modifies the level of thrust produced by the propulsion system of a spacecraft from a low value to a higher value and forms a chain of problems linking the original low-thrust problem to a modified higher-thrust problem. The optimal trajectory of this higher-thrust problem is first obtained by using a direct method. The modified higher level of the thrust results in a shorter mission trajectory which in turn means a smaller NLP that is easier to solve at first. The obtained solution then serves as an initial guess for the next neighboring
problem (lower thrust level) in the problem chain. The computer program implementing the
homotopy method successively loads the result from the previous higher-thrust problem as the
initial guess and optimizes the next lower-thrust problem in the chain by using the same direct
method. After traversing through the problem chain from one end to another, the optimal
trajectory of the original low-thrust problem can be obtained. This method has high probability of
achieving convergence. In fact, if any two values of the thrusts of two neighboring problems in
the chain are close enough, the method ensures probability-one (i.e. definite) convergence to a
locally optimal trajectory (local optima). We apply this method to a realistic force three-
dimensional escape transfer problem which is modified from the model of SMART-1 or “Small
Missions for Advanced Research in Technology” of the ESA scientific program. A
straightforward way of optimizing this problem using a direct method requires direct handling of
a huge NLP. By using the homotopy method, we start by solving a small NLP and eventually
obtain a locally optimal minimum-time escape transfer trajectory. We discuss the direction of
future development to enable the homotopy optimization method to search for the global
minimum.
# TABLE OF CONTENTS

LIST OF FIGURES ................................................................................................................................................. vi

LIST OF TABLES ..................................................................................................................................................... vii

ACKNOWLEDGEMENTS ........................................................................................................................................... viii

Chapter 1  Introduction ........................................................................................................................................ 1

1.1 Trajectory Optimization ................................................................................................................................. 1
1.2 Motivation and Objectives ............................................................................................................................. 2
1.3 Thesis Overview ............................................................................................................................................... 4

Chapter 2  Low-Thrust Trajectory Optimization .................................................................................................. 5

2.1 The Necessary Condition of Optimality for an Optimal Control Problem .................................................. 5
2.2 Direct and Indirect Methods .......................................................................................................................... 8
2.3 Deterministic and Stochastic Search Techniques ........................................................................................... 10

Chapter 3  Homotopy Optimization Method ...................................................................................................... 12

3.1 General Homotopy Method .......................................................................................................................... 12
3.2 Application to Trajectory Optimization ......................................................................................................... 14
3.3 Thrust Modification Method ........................................................................................................................ 16
  3.3.1 Basic Algorithm ...................................................................................................................................... 16
  3.3.2 Numerical Optimization Method ........................................................................................................... 18

Chapter 4  Application to a Two-Dimensional Two-Body Escape Problem .......................................................... 21

4.1 Dynamics and Boundary Conditions ............................................................................................................. 21
4.2 Optimization Program .................................................................................................................................... 22
4.3 Results ......................................................................................................................................................... 23

Chapter 5  Application to a Realistic Force Three-Dimensional Escape Problem ............................................ 30

5.1 Dynamics and Boundary Conditions ............................................................................................................. 30
  5.1.1 Equations of Motion Expressed in Modified Equinoctial Elements ..................................................... 31
  5.1.2 The Calculation of Perturbations ........................................................................................................... 33
  5.1.3 Boundary conditions ............................................................................................................................... 36
  5.1.4 Other Constraints and the Objective function ....................................................................................... 37
5.2 Optimization Program .................................................................................................................................... 38
5.3 Results ......................................................................................................................................................... 38

Chapter 6  Conclusions and Recommendations for Future Research ............................................................... 49

References ......................................................................................................................................................... 51
LIST OF FIGURES

Figure 2-1: Illustration of the DCNLP method ................................................................. 10
Figure 4-1: The initial result from optimization of TA = 0.6 LU/TU^2 problem ................. 24
Figure 4-2: The refined result from optimization of TA = 0.6 LU/TU^2 problem ................. 24
Figure 4-3: The result from optimization of TA = 0.5 LU/TU^2 problem .......................... 25
Figure 4-4: The result from optimization of TA = 0.4 LU/TU^2 problem .......................... 25
Figure 4-5: The result from optimization of TA = 0.3 LU/TU^2 problem .......................... 26
Figure 4-6: The result from optimization of TA = 0.2 LU/TU^2 problem .......................... 26
Figure 4-7: The result from optimization of TA = 0.1 LU/TU^2 problem .......................... 27
Figure 4-8: The interpolated escape transfer trajectory of TA = 0.1 LU/TU^2 problem ....... 27
Figure 4-9: Optimal Control of Thrust Angle (ϕ) of TA = 0.1 LU/TU^2 problem .......... 28
Figure 4-10: Value of Hamiltonian function of the candidate optimal trajectory of TA = 0.1 LU/TU^2 problem ................................................................. 29
Figure 5-1: Optimized escape transfer trajectory in ECI frame ........................................ 39
Figure 5-2: Equinoctial element p during transfer ......................................................... 40
Figure 5-3: Equinoctial element f during transfer ......................................................... 40
Figure 5-4: Equinoctial element g during transfer ......................................................... 40
Figure 5-5: Equinoctial element h during transfer ......................................................... 41
Figure 5-6: Equinoctial element k during transfer ......................................................... 41
Figure 5-7: Equinoctial element L during transfer ......................................................... 41
Figure 5-8: Eccentricity during transfer ........................................................................... 42
Figure 5-9: Inclination during escape .............................................................................. 42
Figure 5-10: Argument of Perigee during transfer ......................................................... 43
Figure 5-11: Right ascension of ascending node during transfer .................................... 43
Figure 5-12: True anomaly during transfer.................................................................44
Figure 5-13: Optimal control history component $U_r$ during transfer ......................44
Figure 5-14: Optimal control history component $U_{th}$ during transfer .......................45
Figure 5-15: Optimal control history component $U_h$ during transfer .........................45
Figure 5-16: Optimal Control of Thrust Angle ($\phi$) during transfer .............................46
Figure 5-17: The time value of Hamiltonian function of the new solution ......................47
LIST OF TABLES

Table 3-1: Basic Thrust Modification Algorithm ................................................................. 17

Table 4-1: Optimization process of a 2D escape problem .................................................... 23

Table 5-1: Initial values of classical orbital elements ............................................................ 36
I would like to thank Dr. Robert Melton, my advisor, for all his guidance and support during my research and the completion of this thesis. I would also like to thank Dr. Spencer for his careful review of this thesis.
Chapter 1

Introduction

Space missions utilizing low-thrust propulsion systems, such as solar-electric systems have been investigated and developed for decades. Compared to traditional high-thrust propulsion technologies (e.g. chemical propulsion), these advanced low-thrust propulsion systems are capable of using high specific impulse ($I_{sp}$) propellant or even can function without any propellant (such as solar sail technology). Spacecraft equipped with these propulsion systems turn out to be more fuel efficient which in turn allow a high spacecraft payload ratio. This fuel-efficiency is exchanged for a much longer flight time due to the low thrust level and therefore low thrust acceleration.

1.1 Trajectory Optimization

An important task in the preliminary stage of space mission design is the optimization of possible mission trajectories. Trajectory optimization is the process of obtaining a trajectory that minimizes or maximizes some measures of mission performance subject to a set of mission constraints. Typically, measures of mission performance consist of the flight time, propellant consumption or a mixed performance measure calculated from time and propellant consumption simultaneously. Mission constraints consist of the differential equations describing the spacecraft’s movement, boundary conditions of state of the spacecraft and any other restrictions on paths of the state or control over time.
The essence of the optimization process is to find the set of functions of control variables that result in the optimal trajectory. In missions with a high-thrust spacecraft, thrust arcs are short, so the controls are typically modeled on isolated and singular points which justify the usage of discrete optimization theory to obtain the optimal control for the mission. In contrast, a low-thrust spacecraft keeps its propulsion system operating over a significant part of the mission time. Control variables in those missions must be modeled as continuous functions whose optimizations are based on more complicated continuous time methods.

Multiple continuous optimization methods have been proposed and tested on different types of trajectory optimization problems. Those methods historically fall into two categories: direct and indirect methods. Indirect methods derive the necessary condition for optimality using calculus of variations and form a multiple-point boundary value problem (MPBVP). Direct methods, on the other hand, discretize and approximate a continuous problem with a parameter optimization problem and result in a nonlinear programming problem (NLP).

1.2 Motivation and Objectives

A solution obtained by indirect methods exactly satisfies the necessary condition of optimality so that can represent the optimal trajectory with superior accuracy. But sensitivity problems associated with indirect methods make them hard to implement. Direct methods discretize the original problem and generate an approximate solution to the problem. The advantage of direct methods is their numerical robustness which becomes a decisive feature favoring direct methods over indirect methods. For general problems, direct methods converge fast to the solutions without a demand of a high-accuracy initial estimate of these solutions. This feature has even been tested by using physically unreasonable initial guesses [19]. But as the size
of the problem enlarges, besides the exponentially increased computing time, the effect of the accuracy of the initial guess on the performance of these methods becomes obvious. In low-thrust trajectory optimization problems, perturbations often dominate the thrust and make mission trajectories extremely long. The transformed NLP’s for those problems are typically very huge. For direct handling of those huge NLP’s, a highly accurate initial guesses is critical. If these initial guesses are not available, they must be carefully constructed before the optimization. In practice, existing solutions of similar problems can be modified to meet the demand occasionally. For a new problem, an initial guess is generally derived analytically by exploiting optimal control theories. However, convergence is not guaranteed by using these initial guesses.

A homotopy optimization method obtains the solution of one problem from a known solution from a closed related problem. The optimal trajectory of one mission scenario can be derived from a known optimal trajectory of another scenario, which eliminates the necessity of the construction of the initial guess for the current scenario. We develop a homotopy method which modifies the level of thrust produced by the propulsion system of a spacecraft from a low value to a higher value and forms a chain of problems linking the original low-thrust problem to a modified higher-thrust problem. The optimal trajectory of this higher-thrust problem is first obtained by using a direct method. The modified higher level of the thrust results in a shorter mission trajectory which in turn means a smaller NLP that is easier to solve at first. The solution obtained then serves as an initial guess for the next neighboring problem in the problem chain. The computer program implementing the homotopy method successively loads the result from the previous higher-thrust problem as the initial guess and optimizes the next lower-thrust problem in the chain by using the same direct method. After traversing through the problem chain from one end to the other, the optimal trajectory of the original low-thrust problem can be obtained.
We demonstrate that if two values of the thrust of two neighboring problems in the chain are close enough, the method ensures with high-probability convergence to a locally optimal trajectory (local optimizer). The primary objective of this thesis is to validate the features of the strategy by applying it to a low-thrust trajectory optimization problem which has lengthy trajectory and various perturbation effects. The second objective is to discuss the issue of local vs. global optimality and give a possible direction for development of a global homotopy method.

1.3 Thesis Overview

Chapter 2 derives the necessary condition of optimality for a solution of an optimal control problem and makes a comparison between direct and indirect methods. Chapter 3 outlines the homotopy optimization method and in particular, the thrust modification method used in this thesis. Chapter 4 describes the application of the method to a two-dimensional two-body escape transfer problem. Chapter 5 describes the application to a realistic force three-dimensional escape transfer problem. Chapter 6 summaries the work and gives a brief discussion of the development of a global homotopy optimization method.
Chapter 2

Low-Thrust Trajectory Optimization

Analytical solutions can be obtained only for some classical optimal control problems and special weakly nonlinear low dimensional systems by satisfying the necessary and sufficient conditions of optimality. Low-thrust trajectory problems are described by strongly nonlinear differential equations and generally depend on numerical methods for their solutions.

2.1 The Necessary Condition of Optimality for an Optimal Control Problem

Let \( x(t) \in X \subseteq \mathbb{R}^n \) being a function of an n-dimensional vector, \( u(t) \in U \subseteq \mathbb{R}^m \) being a function of an m-dimensional vector. An optimal control problem is to find a control function \( u^*(t) \) which satisfies a set of nonlinear differential equations

\[
\dot{x}(t) = f(x(t),u(t),t)
\] (2.1)

in a time interval \( t \in [t_0,t_f] \) and boundary conditions

\[
\psi(x(t_0),x(t_f),t_0,t_f) = 0
\] (2.2)

so that a cost function

\[
J = \phi(x(t_f),t_f) + \int_{t_0}^{t_f} \varphi(x(t),u(t),t)dt
\] (2.3)

is minimized, where \( \phi \) and \( \varphi \) are the terminal and accumulated cost function, respectively. It is more convenient to transform the constrained problem into the unconstrained problem. Then the Lagrange form of the cost function is constructed as
\[ J^L = \phi(x(t_f),t_f) + v^T \psi(x(t_f),t_f) + \int_{t_0}^{t_f} (\varphi(x(t),u(t),t) + \lambda^T (f(x(t),u(t),t) - \dot{x}(t))) dt \]  
(2.4)

where \( v \) and \( \lambda \) are two co-vectors associated with boundary and dynamic constrains. Define the Hamiltonian function

\[ H = \varphi(x(t),u(t),t) + \lambda^T f(x(t),u(t),t) \]  
(2.5)

\( J^L \) can be rewritten as

\[ J^L = \phi(x(t_f),t_f) + v^T \psi(x(t_f),t_f) + \int_{t_0}^{t_f} (H - \lambda^T \dot{x}(t)) dt \]  
(2.6)

Follow the analysis of Pontryagin [2], the necessary condition of optimality is derived from computing the variation of \( J^L \). After reduction and rearrangement, the variation of \( J^L \) takes the following form

\[ \delta J^L = \left[ \frac{\partial \phi}{\partial x} + v^T \frac{\partial \varphi}{\partial x} - \lambda \right] \delta x + \left[ \frac{\partial \phi}{\partial t} + v^T \frac{\partial \psi}{\partial t} + H \right] \delta t \]

\[ + \int_{t_0}^{t_f} \left[ \left( \frac{\partial H}{\partial x} + \lambda \right) \delta x + \left( \frac{\partial H}{\partial u} - \dot{x} \right) \delta u \right] dt \]  
(2.7)

The necessary condition of optimality requires \( \delta J^L = 0 \), which is satisfied by enforcing the coefficients of the variation terms to equal zero. Then the necessary conditions of optimality are

\[ \dot{\lambda} = -\frac{\partial H}{\partial x} \]  
(2.8)

\[ \frac{\partial H}{\partial u} = 0 \]  
(2.9)

\[ \frac{\partial H}{\partial \lambda} = \dot{x} \]  
(2.10)

\[ \left[ \frac{\partial \phi}{\partial x} + v^T \frac{\partial \psi}{\partial x} - \lambda \right]_{ij} = 0 \]  
(2.11)

\[ \left[ \frac{\partial \phi}{\partial t} + v^T \frac{\partial \psi}{\partial t} + H \right]_{ij} = 0 \]  
(2.12)
Equation (2.10) is automatically satisfied since \( \frac{\partial H}{\partial \lambda} = f = \dot{x}. \) Thus the necessary conditions for optimality are to satisfy the co-state differential Eq. (2.8), the optimal control Eq. (2.9) and the two boundary conditions on terminal time according to Eqs. (2.11-2.12). Second order necessary conditions of optimality can be satisfied by making the matrix \( \frac{\partial^2 H}{\partial u^2} \) positive semidefinite. Thus a full necessary condition of optimality changes the optimal control Eq. (2.9) into

\[
\frac{\partial H}{\partial u} = 0, \quad \frac{\partial^2 H}{\partial u^2} \geq 0
\]  

(2.13)

For a minimum-time problem described by differential equations without explicit presence of time, the Hamiltonian function of the solution takes a special value. The first time derivative of the Hamiltonian is expanded as:

\[
\dot{H}(x(t),u(t),\lambda,t) = (\frac{\partial H}{\partial x})^T \dot{x} + (\frac{\partial H}{\partial u})^T \dot{u} + (\frac{\partial H}{\partial \lambda})^T \dot{\lambda} + \frac{\partial H}{\partial t}
\]  

(2.14)

When rewritten using Eq. (2.1) and Eq. (2.5), the derivative takes the form

\[
\dot{H} = [\frac{\partial H}{\partial x} + \dot{\lambda}]^T f + (\frac{\partial H}{\partial u})^T \dot{u} + \frac{\partial H}{\partial t}
\]  

(2.15)

If the necessary condition is satisfied by a candidate solution, Eqs. (2.8-2.9) imply that the first two terms in the right-hand side of Eq. (2.15) vanish. If the optimization problem minimizes the terminal cost (i.e. \( \varphi = 0 \)) and there is no explicit time \( t \) appearing in the differential equation of dynamics (i.e. \( \dot{x}(t) = f(x(t),u(t)) \)) , the third term also vanishes which implies that the Hamiltonian function has a constant value over time. Furthermore, if the problem is a minimum–time problem (i.e. \( \phi = t_f \) ), examining the boundary condition in Eq. (2.12) yields
\[ H_{|_{t_f}} = -\frac{\partial \phi}{\partial t}_{|_{t_f}} = -1 \]  

(2.16)

which in turn means

\[ H \equiv -1 \]  

(2.17)

This is a very important conclusion for checking the optimality for special minimum-time problems. We use this conclusion to check the optimality of results for the minimum-time problems treated in this thesis.

### 2.2 Direct and Indirect Methods

Indirect methods depend on the necessary conditions of optimality derived in the previous section. The differential equations of state, Eq. (2.1) and co-state, Eq. (2.8)

\[ \dot{x} = f(x,u,t), \quad \dot{\lambda} = -\frac{\partial H(x,u,\lambda,t)}{\partial x} \]  

(2.18)

plus boundary conditions of state on \( t_0 \) and \( t_f \) yield a two-point boundary value problem (TPBVP). If there are additional constraints in some interior points, the BVP then becomes a multiple point boundary value problem (MPBVP). The optimal control \( u \) used in Eq. (2.18) is derived by minimizing the Hamiltonian function with respect to the control vector as described in Eq. (2.13). The unknown boundary value is the initial value of the co-state \( \lambda(t_0) \). Then the optimization task becomes: choosing the boundary value \( \lambda(t_0) \) so that the cost function \( J \) in Eq. (2.3) is minimized, an indirect way to treat the original problem. Indirect methods are attractive in that they can obtain solutions with superior accuracy. But as discussed in the literature [19,20], the co-state equation is extremely sensitive to the variation of \( \lambda(t_0) \), thus the successful application of indirect methods depends heavily on the accuracy of the initial estimate of the \( \lambda(t_0) \). But the
co-state vector actually is just a set of Lagrange multipliers and generally has no physical meaning. This makes indirect methods quite difficult to get started. In practice, indirect methods are best used in the last stage of a multi-phase optimization process in which an approximate solution is first obtained by using direct methods. But the process of converting from the approximate solution from direct methods to a form suitable for the indirect method is tedious and also requires calculating the associated initial co-state value.

Direct methods discretize the optimal control problem over the time interval and directly choose the values of states or controls at these discrete points of time (nodes) to minimize the cost function. The original problem is then transformed into a nonlinear programming problem (NLP). This can be done by two approaches. The first approach is to choose the values of the controls at the nodes only and numerically integrate the differential equations of dynamics to obtain the values of the cost function. This method is called direct-shooting. The problem with this approach is the cost of integrating the differential equations. The alternative approach is called the direct collocation method with nonlinear programming problem (DCNLP) which chooses values of both the controls and states at the nodes. In DCNLP, the trajectory and the controls are approximated by continuous polynomials interpolated from the chosen values of states and controls. A defect vector \( d \) is calculated at the center point between two neighboring nodes

\[
d = f(x_c, u_c) - \dot{x}_c
\]  

(2.19)

where \( f \) is the dynamic function as in Eq. (2.1), \( x_c \) and \( u_c \) are the state and control vector at the center point and \( \dot{x}_c \) is the vector of time derivatives of the polynomials approximating the states at the center point. By enforcing \( d \) sufficiently small, the differential equations of dynamics are implicitly integrated. This method is illustrated in Fig 2-1.
Traditionally the polynomials used are local polynomials that approximate the trajectory piecewise. Hargraves and Paris [3] used linear interpolation for the control and a third-order Hermite interpolation for the states. Higher-order collocation methods were studied by Herman and Spencer [4]. The concept of using globally orthogonal polynomials has gained in importance and received increasing attention in the literature [5,6,7,8]. In general, direct methods have larger convergence space and are easier to start. The drawback is their relatively lower accuracy compared with indirect methods.

**2.3 Deterministic and Stochastic Search Techniques**

When a problem is transformed into an MVBVP or an NLP, the task left is to choose the set of optimization parameters that reduce the value of the cost function to a minimum point. How to search this set of parameters falls into two categories: deterministic and stochastic search techniques. Deterministic techniques use local gradient information. They choose a direction in search space and calculate a step length that deterministically decreases the value of the cost function in each iteration and terminates at a point when no further decrease can be found. As is
widely known, a trajectory optimization problem may have many local minima. A local minimum is a point that satisfies the necessary condition of optimality as derived in section 2.1. The global minimum is the local minimum that has the lowest value of the cost function. By using local gradient information, deterministic techniques tend to find the local minimum that is closest to the start point of the search. Stochastic search techniques, primarily implemented by the so called evolutionary algorithms, randomly select a population of candidate solutions in the search space and evaluate them. “Good features” of those candidates are selected out and combined to form a set of new candidates. After being randomly mutated, these candidates generate new populations with the intent of producing better candidates. The algorithm will iterate this process of generating new populations until the values of the cost function of those candidates have been reduced to a satisfactory value. Stochastic search techniques have a fairly high probability to enclose the global optimum. However, the convergence of using stochastic search techniques is slow compared to using gradient-based, deterministic techniques.
Chapter 3

Homotopy Optimization Method

A homotopy optimization method obtains the solution of one problem from a known solution from another problem. In its application to the trajectory optimization, the optimal trajectory of one mission scenario can be derived from a known optimal trajectory of another scenario, which eliminates the necessity of the construction of the initial guess for the current mission. Small modification on parameters of one optimal control problem results in a different but similar problem. If the modification is small enough, the optimal controls for these two problems are nearly identical. Then a known solution of one problem becomes an ideal starting point to the search of the solution for another problem. Continuously small modifications could lead the original problem to a totally new problem through a path depending on how the parameters of the problem are modified. The known solution of the original problem “traces” through this path and converges to the solution of the new one.

3.1 General Homotopy Method

Homotopy methods are also referred to as continuation methods and have been effectively used for solving systems of nonlinear equations [9] and nonlinear optimization problems [10]. To solve a system of nonlinear equations \( f^1 (x), f^1 : \mathbb{R}^n, x \in \mathbb{R}^n \), given \( x^*_0 \) is a known zero for another system \( f^0 (x) \), i.e. \( f^0 (x^*_0) = 0 \), we define the homotopy variable \( s \in [0,1] \subset \mathbb{R} \) and construct a homotopy function depending on \( s \)

\[
H(x, s) = sf^1 (x) + (1 - s)f^0 (x)
\] (3.1)
which satisfies

\[ H(x,0) = f^0(x), \quad H(x,1) = f^1(x) \]  

(3.2)

The known zero for \( H \) is \((x_0^*,0)\). If another zero \((x_1^*,1)\) can be found, then the original system of equation is solved with \( f^1(x_1^*) = 0 \). To find \((x_1^*,1)\), we discretize the interval of \( s \) into

\[ 0 = s_1 < s_2 < \ldots < s_n = 1 \]  

(3.3)

Starting with the known zero \((x^*(s_1),s_1) = (x_0^*,0)\), we obtain the zero \((x^*(s_i),s_i)\) of \( H \) by using an iterative method with the initial point \((x^*(s_{i-1}),s_{i-1})\). A string of zeros is then traced with the last one \((x^*(s_n),s_n) = (x_1^*,1)\) the desired solution.

Now consider its application to an optimal control problem defined in section 2.1. Let \( X_0^* = (x^*(t),u^*(t),t_0^*,t_f^*) \) be a known solution for an optimal control problem \( C_0 \) subject to dynamic constraints

\[ \dot{x}(t) = f^0(x(t),u(t),t) \]  

(3.4)

and boundary conditions

\[ \psi^0(x(t_0),x(t_f),t_0,t_f) = 0 \]  

(3.5)

and that minimizes the cost function \( J \) as given in Eq. (2.4). Our goal is to solve another optimal control problem \( C_1 \) subject to dynamic constraints

\[ \dot{x}(t) = f^1(x(t),u(t),t) \]  

(3.6)

and boundary conditions

\[ \psi^1(x(t_0),x(t_f),t_0,t_f) = 0 \]  

(3.7)

and that minimizes the same cost function \( J \). We define again the homotopy variable \( s \in [0,1] \subset \mathbb{R} \). Then we construct the \( s \)-dependent homotopy dynamic constraint
\[ F(x(t), u(t), t, s) = sf^1(x(s), u(s), t) + (1 - s)f^0(x(s), u(s), t) \quad (3.8) \]

and homotopy boundary constraint

\[ \Psi(x(t_0), x(t_1), t_0, t_1, s) = s \psi^1(x(t_0), x(t_1), t_0, t_1) + (1 - s)\psi^0(x(t_0), x(t_1), t_0, t_1) \quad (3.9) \]

The parameter-dependent homotopy optimal control problem \( C(s) \) is defined as finding an optimal control \( u^*(t) \) which satisfies the homotopy dynamic constraints

\[ \dot{x}(t) = F(x(t), u(t), t, s) \quad (3.10) \]

and the homotopy boundary conditions

\[ \Psi(x(t_0), x(t_f), t_0, t_f, s) = 0 \quad (3.11) \]

and that minimizes the cost function \( J \). We discretize the interval of \( s \) again as in Eq. (3.3). The solution of \( C(0) \) is known as \( X^*(0) = X^*_0 \). If the solution \( X^*_1 = X^*(1) \) can be found, then the optimization problem \( C_1 \) is solved. To find \( X^*(1) \), starting with the known solution \( X^*(0) \), we obtain the solution \( X^*(s_i) \) of \( C(s_i) \) by using an numerical optimization method with the solution \( X^*(s_{i-1}) \) of \( C(s_{i-1}) \) as the initial guess. A string of minima is traced by optimizing a chain of problems with the last minimum \( X^*(1) \) being the desired solution.

### 3.2 Application to Trajectory Optimization

As seen in the previous section, the problem with a known solution and the problem with the unknown solution must have similar structures in the dynamics and boundary conditions. If not, the dynamics and constraints can not be added in Eqs. (3.8-3.9). For trajectory optimization, this means two problems must have the same dimension in states and controls with identical coordinates or set of orbital elements used. Because of these constraints, instead of relying on
solutions of solved problems, it is better to directly construct a new problem from the original one by modifying some mission parameters. This can be done by two approaches: modifying the parameters of the propulsion system or modifying the terminal state of the spacecraft. By modifying, we can construct a new mission (problem) which needs a shorter mission trajectory and fewer maneuvers during the mission. For example, for a mission using a solar-electric propulsion system, we can increase its maximum level of thrust to a much higher level that results in a high-, but still continuous-thrust, problem. For a solar-sail mission, this can be done by expanding the area of the solar sail to increase the characteristic acceleration of the sail. Meanwhile, modifying the terminal state can also lead to a shorter trajectory. Consider a transfer problem between two circular orbits with radii \( R_1 = 1 \text{ LU} \) and \( R_2 = 3 \text{ LU} \) in canonical length unit (canonical units are defined in Section 4.1). We can construct another transfer problem with radii of circular orbits set to \( R_1 = 1 \text{ LU} \) and \( R_2 = 1.5 \text{ LU} \) and use the result of this problem to start the homotopy process.

At first glance, this method looks cumbersome and inefficient because we have to solve many problems instead of one. But by taking advantage of the very closeness of two adjacent optimal problems in the problem chain, we limit the searching effort for each optimization by using a start point that is already very close to the minimum. If the discretization of the homotopy variable \( s \) in its interval produces a fairly large number of subintervals, the method will trace a near-continuous path from the solution of the modified problem to the desired solution of the original problem and ensure convergence with high-probability. The number of subintervals used is decided on a balance between limiting the number of optimizations and limiting the search effort for a single optimization.
As mention in Section 2.3, a trajectory optimization problem may have many local minima. A local minimum is a point that satisfies the necessary condition of optimality as derived in Section 2.1. The global minimum is determined by comparing the values of the cost function of all the local minima. Deterministic techniques search a direction that sufficiently decreases the value of the cost function in a single iteration and terminates when a local minimum is found. Thus these techniques find a single local minimum that tends to be close to the start point of the search. A homotopy method combined with a numerical method using these techniques leads to the convergence to a single local minimum and is called a local homotopy method. Ensuring that the initial solution is a global optimum does not guarantee the path-tracing to the global optimum of the desired problem. To help the convergence to the global minimum, stochastic optimization techniques such as Genetic Algorithms must be introduced to the homotopy methods. Global homotopy optimization methods are not used in the current study but are discussed in the final chapter.

### 3.3 Thrust Modification Method

As discussed in the last section, there are two approaches to modify a trajectory optimization problem to a new problem. Modification can be done to the parameters of the propulsion system or to the terminal state of the spacecraft. The first is the method we use in the thesis. Since the problem that we propose to solve uses a solar-electric propulsion system, we use a thrust modification method specifically.
3.3.1 Basic Algorithm

Suppose we need to optimize the trajectory for a low-thrust mission using a solar-electric propulsion system. The corresponding optimization problem is denoted $C_L$. The maximum thrust level of the propulsion system is $T_L$. Then we modify the value of the maximum thrust to a higher value $T_H$ to form a high-thrust problem $C_H$. By using the homotopy variable $s$ we defined in section 2.1, we construct the $s$-dependent homotopy problem $C(s)$ which has a $s$-dependent maximum thrust level as

$$T(s) = sT_L + (1-s)T_H$$  \hspace{1cm} (3.12)

Define the step length variable $\Delta_k, k = 0,1,2...n-1$ which satisfies

$$\Delta_k > 0, \sum_{k=0}^{n-1} \Delta_k = 1$$  \hspace{1cm} (3.13)

Then the basic algorithm is the following:

Table 3-1: Basic Thrust Modification Algorithm

Optimize: Problem $C_H$ with an arbitrary selected initial guess using a local optimization method

Input: $X(0) = X_H$, the obtained local minimum of $C_H$;

The step length variable $\Delta_k, k = 0,1,2...n-1$.

Initialize: homotopy variable $s_k = 0$;

for $k = 1...n$

$$s_k = s_{k-1} + \Delta_k$$

$$T(s_k) = s_kT_L + (1-s_k)T_H$$

Use the local minimization method, starting with $X(s_{k-1})$ to compute
the solution \( X(s_k) \).

end

Output \( X(1) = X_L \)

3.3.2 Numerical Optimization Method

The homotopy method relies on one of the numerical methods to optimize a particular program. We choose a direct method called Legendre Pseudospectral Method and its implementation in the software package DIDO to optimize a particular problem.

Legendre Pseudospectral Method

As discussed in section 2.2. In a direct method, the optimal control problem is first discretized into a parameter optimization problem. The resulting nonlinear programming problem (NLP) is then solved numerically. The discretization to a parameter optimization problem can be done in one of two ways: parameterization of the control variables only and parameterization of both control and state variables. A Legendre Pseudospectral Method uses the second method to transform an optimal control problem. Instead of using locally piecewise-continuous polynomials as the interpolating functions between prescribed nodes in DCNLP, Pseudospectral Methods (PS) use global orthogonal polynomials such as Legendre and Chebyshev polynomials for the approximation of the control and state variables. The PS method implemented in our homotopy method is called the Legendre Pseudospectral Method which is based on representing the states and controls as continuous functions whose values are specified at special locations known as
Legendre-Gauss-Lobatto (LGL) quadrature nodes $\tau_i$, which are distributed over the interval $\tau \in [-1,1]$. If there are totally $N$ nodes are used for the interpolation, the locations of those nodes are the zeros of the derivative of the Legendre polynomial of degree $N$, $L_N$. Equation (3.14) is used to shift the LGL nodes from the computational domain $\tau \in [-1,1]$ to the physical time domain $t \in [t_0,t_f]$.

$$t(l) = \frac{(t_f - t_0)\tau(l) + (t_f + t_0)}{2}, \ 1 \leq l \leq N$$ \hspace{1cm} (3.14)

The state and control functions are then approximated by

$$x(t(\tau)) \approx \sum_{l=0}^{N} x_l \phi_l(\tau)$$ \hspace{1cm} (3.15)

$$u(t(\tau)) \approx \sum_{l=0}^{N} u_l \phi_l(\tau)$$ \hspace{1cm} (3.16)

where

$$\phi_l(\tau) = \frac{1}{N(N+1)L_N(\tau_i)} \frac{(\tau^2 - 1)L_N(\tau)}{\tau - \tau_i}$$ \hspace{1cm} (3.17)

The differential equations of dynamics are approximated by a set of algebraic equations

$$\frac{t_f - t_0}{2} f(x_k,u_k) - \sum_{l=0}^{N} \phi_l(\tau_k) x_l = 0$$ \hspace{1cm} (3.18)

In summary, there are three major differences between the Legendre Pseudospectral Method (LPS) and traditional direct collocation methods. First, LPS uses global polynomials based on the information of states and controls at all the nodes while approximating polynomials in DCNLP are based on information of states and controls at neighboring nodes. Second, the locations of the nodes in LPS are special locations which are the roots of the derivative of the Legendre polynomial while the locations in DCNLP are relatively arbitrary. Three, the dynamics
constraints are enforced at nodes in LPS while they are enforced at the center points between nodes in DCNLP. In practice, LPS can achieve the same accuracy solutions by using a much lower number of nodes than DCNLP. Detailed explanations and mathematical formulations for Legendre Pseudospectral method can be found in references [11,12].

The DIDO Software Package

DIDO is a MATLAB software package first developed by Ross [13] in 1998 to implement the Legendre Pseudospectral Method. It is capable of solving a broad class of smooth and nonsmooth hybrid optimal control problems. There are two features that make DIDO a convenient tool to use in our homotopy method. First, running DIDO just requires calling a MATLAB function. Since using a homotopy method needs repeatedly calling optimization tools to optimize a chain of problems, a single line to execute DIDO makes the overall coding clean and intuitive. Second, DIDO can begin an optimization by selecting an arbitrary initial guess itself. This saves our effort to input an initial guess to start the whole optimization process. Details on DIDO usage can be found in the DIDO user’s manual [13].
Chapter 4

Application to a Two-Dimensional Two Body Escape Problem

This method is first tested on a two-dimensional two body escape transfer problem. The spacecraft is originally in a circular orbit. The goal is to transfer the spacecraft to an escape trajectory in minimum flight time. The spacecraft has a low-thrust propulsion system which produces a fixed level of thrust. We model the mass of the spacecraft to be constant during the flight for the purpose of simplification. This is reasonable for a relatively massive spacecraft using a high specific impulsive system. We seek the optimal direction of the thrust leading to the minimum-time transfer.

4.1 Dynamics and Boundary Conditions

The dynamics model we use in this problem is a two-dimensional two-body model which has no perturbations from other sources. Canonical units are used exclusively in the problem. The length unit (LU) and the time unit (TU) are related, thus making the gravitational parameter of the central gravitational body $\mu = 4\pi^2$. Because we fix the value of thrust produced by the propulsion system and the mass of the spacecraft, the spacecraft will have a constant value of thrust acceleration (TA) throughout the flight. The control variable is the direction of the thrust. The equations of motion, in two-dimensional polar form are

\[ \dot{r} = v_r \]  

(4.1)

\[ \dot{\theta} = \frac{v_r}{r} \]  

(4.2)
\[
\dot{\gamma}_r = \frac{v_r^2}{r} - \frac{\mu}{r^2} + a_r \sin(\phi) 
\] (4.3)

\[
\dot{\gamma}_s = \frac{-v_r v_s}{r} + a_r \cos(\phi) 
\] (4.4)

where \( r \) is the radial distance from the gravitational center, \( \dot{\theta} \) is the time rate-of-change of true anomaly, \( v_r \) is the radial-velocity component, \( v_s \) is the transverse velocity component and \( \phi \) is the thrust control angle measured from the transverse velocity vector to the thrust acceleration vector with positive values in the clockwise direction. \( a_r \) is the constant value of thrust acceleration. In our problem, we set \( a_r = 0.1 \) LU/TU\(^2\).

At reference epoch \( (t_0) \), the spacecraft is in a circular orbit with the radius \( R=1 \) LU. The terminal state is determined by calculating the specific orbit energy

\[
\zeta = \frac{v_r^2 + v_s^2}{2} - \frac{\mu}{r} 
\] (4.5)

If the spacecraft reaches a escape trajectory, \( \zeta = 0 \). The cost function for a minimum-time problem is just \( J = t_f \).

### 4.2 Optimization Program

A computer program to optimize this problem based on the algorithm in Table 3-1. Since we need to use DIDO, the program was written in MATLAB. According to the algorithm, we need to specify two things to start the optimization: the modified high-thrust program and a set of homotopy variable step lengths. To specify the modified problem, we just need to input the modified value of the thrust \( T_H \). But as seen in Eqs. (4.3-4.4), we can directly input the modified
value of the thrust acceleration (TA) since the mass of the spacecraft is constant. We modify the value of the TA to $a_T = 0.6 \text{ LU}/\text{TU}^2$. We adjust the step lengths of the homotopy variable $s$ so that a chain of 6 problems, having values of TA equal to 0.6, 0.5, 0.4, 0.3, 0.2, and 0.1 respectively, will be solved. The choice of a proper set of step length is problem dependent. In fact, the last two step lengths in the set we use are too long as can be seen in the jumps of computing times in solving the last two problems in Table 4-1. However, this two-body problem is a relatively simple problem so that we can obtain the final result with a “coarse” discretization. The step lengths we use to solve the realistic force three-dimensional problem are much smaller as can be seen in Section 5.2.

4.3. Results

The program was run on a Pentium-4 machine with 3GHz CPU and 2GB memory and solves the whole chain of problems with a time of 685 s, roughly 7 minutes.

Table 4-1 Optimization process of a 2D escape problem

<table>
<thead>
<tr>
<th>Solved Optimization problems</th>
<th>Node number used</th>
<th>Computing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial optimization: TA = 0.6 LU/TU$^2$</td>
<td>20</td>
<td>80s</td>
</tr>
<tr>
<td>Result refinement: TA = 0.6 LU/TU$^2$</td>
<td>50</td>
<td>12s</td>
</tr>
<tr>
<td>Optimization: TA = 0.5 LU/TU$^2$</td>
<td>60</td>
<td>12s</td>
</tr>
<tr>
<td>Optimization: TA = 0.4 LU/TU$^2$</td>
<td>80</td>
<td>30s</td>
</tr>
<tr>
<td>Optimization: TA = 0.3 LU/TU$^2$</td>
<td>100</td>
<td>58s</td>
</tr>
<tr>
<td>Optimization: TA = 0.2 LU/TU$^2$</td>
<td>120</td>
<td>211s</td>
</tr>
<tr>
<td>Optimization: TA = 0.1 LU/TU$^2$</td>
<td>150</td>
<td>282s</td>
</tr>
</tbody>
</table>
As seen in Table 4-1, there are two jumps in the computing time in the last two steps because of a “coarse” discretization. But it still outperforms the traditional optimization strategy. A direct optimization of the original problem consumes 20-50 minutes in the same machine with different arbitrarily selected initial guesses. The obtained solutions from each optimization in Table 4-1 are shown below:

Fig. 4-1 The initial result from optimization of TA = 0.6 LU/TU^2 problem

Fig. 4-2 The refined result from optimization of TA = 0.6 LU/TU^2 problem
Fig. 4-3 The result from optimization of $TA = 0.5 \ LU/TU^2$ problem

Fig. 4-4 The result from optimization of $TA = 0.4 \ LU/TU^2$ problem
Fig. 4-5 The result from optimization of $TA = 0.3$ LU/TU^2 problem

Fig. 4-6 The result from optimization of $TA = 0.2$ LU/TU^2 problem
Fig. 4-7 The result from optimization of TA = 0.1 LU/TU² problem

Figure 4-7 shows the desired result of the original TA = 0.1 LU/TU² problem. A smooth trajectory was obtained by interpolating the trajectory between the nodes. The interpolated trajectory is shown in Fig. 4-8:

Fig. 4-8 The interpolated escape transfer trajectory of TA = 0.1 LU/TU² problem
All the units used in Fig. 4-8 are canonical units. The minimum flight-time is 51.2648 TU. The spacecraft swirls around the central body for roughly 15 revolutions before reaching the escape trajectory. The obtained optimal control of the direction of thrust is shown in Fig. 4-9.

![Control History](image)

**Fig. 4-9** Optimal control of thrust angle ($\phi$) of $TA = 0.1$ LU/TU$^2$ problem

Checking the optimality of the result can be done by examining the value of its Hamiltonian. As derived in Section 2.1, for a minimum-time optimal control problem with no explicit presence of time in the dynamics, the Hamiltonian of an optimal solution is constant over time and strictly equal to -1. To get the value of the Hamiltonian of a result, the co-state differential equation needs to be integrated. However, DIDO can automatically generate the Hamiltonian information without explicitly integrating the co-state equation. This is done by implementing the so-called covector mapping theorem [14,15]. The time value of the Hamiltonian function of the result we obtained is shown in Fig. 4-10.
Fig. 4-10 Value of Hamiltonian function of the candidate optimal trajectory of $TA = 0.1 \text{ LU/TU}^2$ problem.

Figure 4-10 shows that the result approximately satisfies the necessary condition of optimality and thus is approximately a local minimum. The accuracy of the result can be improved further by refining in an indirect method.
Chapter 5

Application to a Realistic Force Three-Dimensional Escape Problem

In this chapter, we describe the application of our thrust modification method to a realistic force three-dimensional escape problem. The optimization problem is derived from a real mission called SMART-1 or ‘Small Missions for Advanced Research in Technology’ of the ESA scientific program [1]. The goal of the mission is to insert a spacecraft with a solar-electric propulsion system into a lunar orbit that is polar and elliptic and has its perilune above the south pole. The spacecraft is deployed from an Ariane-5 into an elliptic Earth-centered parking orbit. The mission was successfully fulfilled in 2003. The exact optimization problem based on this mission is treated by Schoenmaekers, Horas, and Pulido [1] and Betts and Erb [16]. The problem treated here is a modified one. Our modified optimization problem is to obtain an optimal trajectory to insert the spacecraft into an escape trajectory from Earth, an optimal escape problem instead of the original Earth-Moon transfer problem.

5.1 Dynamics and Boundary Conditions

A spacecraft in the real space environment is subject to the gravity of the central body, the thrust force and various perturbation forces. The space environment near the Earth typically includes the perturbation forces from the Earth oblateness effect, gravities of secondary bodies, solar radiation pressure and atmospheric drag. In our particular problem, the most dominant perturbation forces are the gravitation of the Moon and the Earth oblateness effect. These forces are modeled in our dynamics as well as the gravity of the Earth and the thrust force. In the process of escaping the Earth, the trajectory of the spacecraft goes from an elliptic orbit to an
parabolic orbit. The set of classical orbital elements is not applicable since the semimajor axis will go to infinity when the spacecraft approaches the parabolic orbit. Therefore we use a set of modified equinoctial orbital elements which is described by Betts [16,17] based on a derivation by Walker et al [18].

5.1.1 Equations of Motion Expressed in Modified Equinoctial Elements

The state variables of the spacecraft consist of two parts,

\[ z^T = [y^T, m] \]  \hspace{1cm} (5.1)

where \( y \) is the vector of equinoctial coordinates given as

\[ y = [p, f, g, h, k, L]^T \]  \hspace{1cm} (5.2)

and \( m \) is the mass of the spacecraft. The control variables

\[ u^T = [u_r, u_\theta, u_\phi] \]  \hspace{1cm} (5.3)

describes the orientation of the thrust vector in a rotating radial frame defined in Eq. (5.17). Then the dynamics of the system can be described as,

\[ \dot{y} = A(y)\Delta + b \]  \hspace{1cm} (5.4)

\[ \dot{m} = -T / C \]  \hspace{1cm} (5.5)

\[ \|u\| = 1 \]  \hspace{1cm} (5.6)

where \( A \) is a \( y \)-dependent matrix that is defined in Eq. (5.7). \( \Delta \) is the vector of perturbation and is defined in Eq. (5.16). \( b \) is a vector that is defined in Eq. (5.8). \( T \) is the value of the thrust force produced by the propulsion system and \( C \) is the exhaust velocity of the thruster. In our particular program, \( T = 73.19 \) mN and \( C = 16.434 \) km/s. The matrix of the equinoctial dynamics \( A \) is given as
\[
A = \begin{bmatrix}
0 & \frac{2\nu}{L} \sqrt{\frac{\mu}{\nu}} & 0 \\
\sqrt{\frac{\mu}{\nu}} \sin L & \sqrt{\frac{\mu}{\nu}} \{ (\xi + 1) \cos L + f \} & -\sqrt{\frac{\mu}{\nu}} \{ h \sin L - k \cos L \} \\
-\sqrt{\frac{\mu}{\nu}} \cos L & \sqrt{\frac{\mu}{\nu}} \{ (\xi + 1) \sin L + g \} & \sqrt{\frac{\mu}{\nu}} \frac{e^2 \cos L}{2\nu} \\
0 & 0 & \sqrt{\frac{\mu}{\nu}} \frac{e^2 \sin L}{2\nu} \\
0 & 0 & \sqrt{\frac{\mu}{\nu}} \frac{e^2 \sin L}{2\nu} \\
0 & 0 & \sqrt{\frac{\mu}{\nu}} \{ h \sin L - k \cos L \}
\end{bmatrix}
\]

(5.7)

and the vector \( b \) is

\[
b^T = \begin{bmatrix}
0 & 0 & 0 & 0 & \sqrt{\frac{\mu}{P}} \left( \frac{\xi}{P} \right)^2
\end{bmatrix}
\]

(5.8)

where

\[
\xi = 1 + f \cos L + g \sin L
\]

(5.9)

\[
r = \frac{P}{\xi}
\]

(5.10)

\[
\alpha^2 = h^2 - k^2
\]

(5.11)

\[
\chi = \sqrt{h^2 + k^2}
\]

(5.12)

\[
s^2 = 1 + \chi^2
\]

(5.13)

The equinoctial coordinates \( y \) are related to the Cartesian state according to the following expression

\[
r(y) = \begin{bmatrix}
\frac{r}{s^2} \left( \cos L + \alpha^2 \cos L + 2hk \sin L \right) \\
\frac{r}{s^2} \left( \sin L - \alpha^2 \sin L + 2hk \cos L \right) \\
\frac{2\nu}{s^2} (h \sin L - k \cos L)
\end{bmatrix}
\]

(5.14)

\[
v(y) = \begin{bmatrix}
-\frac{1}{s^2} \sqrt{\frac{\mu}{P}} (\sin L + \alpha^2 \sin L - 2hk \cos L + g - 2fhk + \alpha^2 g) \\
-\frac{1}{s^2} \sqrt{\frac{\mu}{P}} (-\cos L + \alpha^2 \cos L + 2hk \sin L - f + 2ghk + \alpha^2 f) \\
\frac{2\nu}{s^2} \sqrt{\frac{\mu}{P}} (h \cos L + k \sin L + fh + gk)
\end{bmatrix}
\]

(5.15)
The vector of perturbation $\Delta$ is given by

$$\Delta = \Delta_g + \Delta_q + \Delta_T$$  \hspace{1cm} (5.16)

where $\Delta_g$ is the perturbation effect from the Earth oblateness effect, $\Delta_q$ is the perturbation from the secondary body (the Moon) and $\Delta_T$ is just the thrust acceleration. They are expressed in a rotating radial frame whose principal axes are defined by:

$$Q = \begin{bmatrix} i_r & i_\theta & i_h \end{bmatrix} = \begin{bmatrix} \frac{r}{\|F\|} & \frac{(r \times \mathbf{V}) \times F}{\|F\| \|F\|} & \frac{r \times \mathbf{V}}{\|F\| \|F\|} \end{bmatrix}$$ \hspace{1cm} (5.17)

### 5.1.2 The Calculation of Perturbations

**Earth oblateness effect**

Consider the $J_2$ effect only, $\Delta_g$ is given by

$$\Delta_g = \Delta_{J_2} = [\Delta_{J_{2r}}, \Delta_{J_{2\theta}}, \Delta_{J_{2h}}]^T$$ \hspace{1cm} (5.18)

where the three components are

$$\Delta_{J_{2r}} = -\frac{3 \mu J_2 R_e^2}{2r^4} \left[ 1 - \frac{12 (h \sin L - k \cos L)^2}{(1 + h^2 + k^2)^2} \right]$$ \hspace{1cm} (5.19)

$$\Delta_{J_{2\theta}} = -\frac{12 \mu J_2 R_e^2}{r^4} \left[ \frac{(h \sin L - k \cos L)(h \cos L + k \sin L)}{(1 + h^2 + k^2)^2} \right]$$ \hspace{1cm} (5.20)

$$\Delta_{J_{2h}} = -\frac{6 \mu J_2 R_e^2}{r^4} \left[ \frac{(1 - h^2 - k^2)(h \sin L - k \cos L)}{(1 + h^2 + k^2)^2} \right]$$ \hspace{1cm} (5.21)
where \( \mu \) is the gravitational parameter of the Earth and has a value \( \mu = 398600 \text{ km}^3/\text{s}^2 \). \( R_e \) is the Earth’s equinoctial radius and has a value \( R_e = 6378 \text{ km} \). For the Earth, \( J_2 = 1.0827 \times 10^{-3} \).

**Secondary body gravitation as perturbation**

Generally, a secondary body gravitation as perturbation is calculated as

\[
\delta \mathbf{q} = - \sum_{j=1}^{n} \mu_j \left[ \frac{\mathbf{d}_j}{d_j^2} + \frac{\mathbf{s}_j}{s_j^3} \right]
\]

where \( s_j \) is the vector from the primary body to a secondary body \( j \), with gravitational constant \( \mu_j \), and \( d_j \) is the vector from the secondary body \( j \) to the spacecraft.

According to Battin [21], this calculation is prone to cancelation because of the significantly different size of the terms which can lead to large round off errors. Instead, he suggests defining the function

\[
F(q_k) = q_k \left[ \frac{3 + 3q_k + q_k^2}{1 + (\sqrt{1 + q_k})^3} \right]
\]

(5.23)

where

\[
q_k = \frac{r^T(r - 2s_k)}{s_k^Ts_k}
\]

(5.24)

and finally,

\[
\delta \mathbf{q} = - \sum_{k=1}^{n} \frac{\mu_k}{\dot{q}_k^3} [\mathbf{r} + F(q_k)s_k]
\]

(5.25)

To obtain the acceleration of the perturbing bodies in the rotating radial frame,
\[ \Delta_q = Q^T \dot{q} \]  \hspace{1cm} (5.26)

where \( Q \) is defined in Eq. (5.17)

To calculate the secondary body perturbing effect, the locations of other celestial bodies with respect to the Earth are needed. This could be done by using an ephemeris, for example, the Jet Propulsion Laboratory (JPL) Ephemeris, DE 405 [24]. Because we include the secondary body gravitation of the Moon, we obtain the coordinates of the Moon in the Earth Centered Inertial (ECI) frame for a whole year which can cover the entire flight time.

**Thrust acceleration**

The thrust acceleration is defined by

\[ \Delta_r = \frac{T}{m} u \]  \hspace{1cm} (5.27)

where \( T \) is the maximum thrust of the thruster as defined in Eq. (5.5) and \( m \) is the mass of the spacecraft. The direction of the thrust acceleration vector, which is defined by the control vector \( u^T = [u_\tau, u_\phi, u_\lambda] \), can be arbitrary in space but has a unit length path constraint defined by Eq. (5.6).
5.1.3 Boundary conditions

Initial condition

The original mission nominal launch window opens on December 20, 2002 at 23:18:48. For our problem, the reference epoch \( t_0 \) is defined at midnight on June 20th, 2008, which corresponds to a Julian date of 2454632.5. The initial mass of the spacecraft is 350 kg. The solar-electric propulsion system utilizes a single PPS-1350 Hall-plasma thruster with a force of 73.19 mN and an exhaust velocity of 16.434 km/s. The parameters of the geostationary transfer orbit (GTO) established after deployment from an Ariane-5 expressed in classical orbital elements are in summarized in Table 5-1.

Table 5-1 Initial values of classical orbital elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semimajor axis (km)</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>24661.144</td>
</tr>
<tr>
<td>Eccentricity</td>
<td>e</td>
</tr>
<tr>
<td></td>
<td>0.716227925</td>
</tr>
<tr>
<td>Inclination (deg)</td>
<td>i</td>
</tr>
<tr>
<td></td>
<td>7.0</td>
</tr>
<tr>
<td>Argument of perigee (deg)</td>
<td>( \omega )</td>
</tr>
<tr>
<td></td>
<td>178</td>
</tr>
<tr>
<td>Right ascension of the ascending node (deg)</td>
<td>( \Omega )</td>
</tr>
</tbody>
</table>

The initial values of the modified equinoctial orbital elements can be calculated from classical orbital elements by the following formulas

\[
p = a(1 - e^2)
\]

\[
f = e \cos(\omega + \Omega)
\]

\[
g = e \sin(\omega + \Omega)
\]

\[
h = \tan\left(\frac{i}{2}\right) \cos(\Omega)
\]
\[ k = \tan\left(\frac{i}{2}\right) \sin(\Omega) \quad (5.32) \]

\[ L = \Omega + \omega + \theta \quad (5.33) \]

**Final condition**

One way of determining if the spacecraft reaches the escape trajectory is by calculating the specific orbit energy as we do in the two-dimensional problem. But this method needs the value of the radial distance and the velocity of the spacecraft which is hard to compute from the modified equinoctial orbital elements as seen in Eqs. (5.14-5.15). A more convenient way is to compute the eccentricity from the equinoctial elements

\[ e = \sqrt{f^2 + g^2} \quad (5.34) \]

For an escape trajectory, \( e \geq 1 \).

**5.1.4 Other Constraints and the Cost function**

Beside the boundary constrains, there is a path constraint in the control vector in Eq. (5.6). The thrust force \( T \) of the thruster in our problem is set to be fixed and always be the maximum level that the thruster can produce. This reduces one dimension of the control. In fact, although a real thruster can vary the thrust level during the flight, since our problem is a minimum-time problem with the cost function \( J = t_f \), the thruster will always set its thrust at the maximum value during the flight to save time.
5.2 Optimization Program

The computer program written for this problem is similar as the one used in the two-dimensional problem and is also based on the algorithm in Table 3-1. The dynamics of the problem are fully described in Section 5.1. The two things that we need to specify to start the optimization are the modified high-thrust problem and a set of homotopy variable step lengths. To specify the modified problem, we need to input the modified value of the thrust \( T_H \). The modified value we use is 103 times the original thrust \( T_L \): \( T_H = 103 \times T_L = 7.541 \) N. This level of thrust enables the spacecraft to reach the escape trajectory with a much shorter trajectory. We discretize the interval of the homotopy variable \( s \) into 95 subintervals. We input the set of step lengths which is based on our discretization. Then the optimization program can start to run and solve a chain of 96 problems.

5.3 Results

This program was run on a Pentium-4 machine with a 3GHz CPU and 2GB memory. The first problem where the thrust force is 7.541 N is first solved in DIDO using 80 nodes for about 20 minutes with an arbitrary initial guess. For each later refinement or the optimization of a new problem, the computing time ranges from minutes to hours. The last problem (the real problem) is optimized by using a node number of 550. The result is interpolated between nodes to obtain a smooth trajectory. The whole optimization process lasted for several days.
The real thrust of the propulsion system is 73.19 mN which produces an initial thrust acceleration of 0.2091 mN/kg. The spacecraft needs to undergo a total number of 40-50 revolutions around the Earth to transfer to the escape trajectory. The minimized flight time during the transfer is 16155761.9 s which is roughly 187 days. The final mass of the spacecraft is 274.3 kg. The fuel consumption is then 75.7 kg, which is 21.6% of the initial mass. The result in the Earth Centered Inertial (ECI) frame is shown in Fig. 5-1:

![Optimized escape transfer trajectory in ECI frame](image)

As can be seen from Fig. 5-9, the motion is not coplanar. This is because of the perturbation of the gravity of the Moon instead of any out-of-plane maneuver. As seen in Fig. 5-15, the optimal control component $U_h$ is strictly zero over time which means there is no out-of-plane maneuver. The effect of the perturbation of the Moon on inclination of the orbit becomes more obvious when the spacecraft approaches the escape trajectory. The changes of each of the modified equinoctial elements are shown from Fig. 5-2 to Fig. 5-7.
Fig. 5-2 Equinoctial element p during transfer

Fig. 5-3 Equinoctial element f during transfer

Fig. 5-4 Equinoctial element g during transfer
Fig. 5-5 Equinoctial element h during transfer

Fig. 5-6 Equinoctial element k during transfer

Fig. 5-7 Equinoctial element L during transfer
The classical orbital elements are computed from the equinoctial elements and are shown in Fig. 5-8 to Fig. 5-12. The semimajor axis $a$ will go to infinity when the spacecraft approaches the escape trajectory and it is not computed.

![Eccentricity during transfer](image)

**Fig. 5-8 Eccentricity during transfer**

The eccentricity $e$ is shown in Fig. 5-8. The change in eccentricity indicates that the orbit of the spacecraft gradually switch from the highly elliptic GTO orbit to a circular-like orbit in the beginning phase. When it reaches some critical point of time, the shape of the orbit switches from circular-like to elliptic rapidly and finally to parabolic.

![Inclination during transfer](image)

**Fig. 5-9 Inclination during transfer**
Figure 5-9 shows the change in inclination. This change is due to the perturbation of the Moon’s gravity. This effect becomes more obvious when the spacecraft is getting away from the Earth.

Figure 5-10 and 5-11 show the change in argument of perigee and right ascension of ascending node. The thrust, the Earth oblateness effect and the Moon’s gravity all play roles in these changes.

![Figure 5-10 Argument of Perigee during transfer](image1)

![Figure 5-11 Right ascension of ascending node during transfer](image2)
The change in true anomaly is shown in Fig. 5-12.

![Graph showing change in true anomaly](image1)

Fig. 5-12 True anomaly during escape

The control histories consist of three components in a rotation radial frame that is defined by Eq. (5.17). The component $U_r$ is the projection of the unit control vector on the axis directed along the radial position vector of the spacecraft, $U_h$ is the projection on the axis directed along the normal vector of the current orbital plane and $U_{th}$ is the projection on the axis directed along the transverse velocity vector. They are shown in Fig. 5-13 to Fig. 5-15.

![Graph showing control history components](image2)

Fig. 5-13 Optimal control history component $U_r$
Figure 5-15 indicates there is no out-of-plane maneuver during the transfer. We can derive the value of thrust angle (\( \phi \)), which is measured from the transverse velocity vector to the thrust vector with positive values in the clockwise direction. This value is a better way of displaying the control of the direction of the thrust. \( \phi \) can be computed by

\[
\phi = \tan^{-1} \left( \frac{U}{U_{th}} \right)
\]  

(5.35)
Figure 5-16 shows the change of $\phi$ during the transfer.

Checking the optimality of the solution of this problem is not straightforward as we do in the two-dimensional problem. The problem has a path constraint on control as defined in Eq. (5.6). Thus the necessary condition falls outside the derivation in section 2.1. However, as indicated in Fig. 5-15, there is zero value for the component in $U_{th}$. Then we can reduce the dimension of the control to one and use thrust angle $\phi$ as the only control variable. The control vector $u$ is then changed into

$$u^T = [u_r, u_{\phi}, u_h] = [\sin(\phi), \cos(\phi), 0] \quad (5.36)$$

The problem then has no path constraint. The problem is re-optimized directly by using DIDO and the result displayed in Fig. 5.2 – Fig. 5.7 and Fig. 5-16 as the initial guess. The solution obtained in this optimization nearly coincides with the original solution. The minimum flight time is $16155737.5$ s compared with the original result $16155761.9$ s. The tiny difference in flight time indicates that two solutions are approximately the same. Because the new problem has no
constraint, the Hamiltonian information obtained in DIDO can be used to indicate optimality. The time value of Hamiltonian of the new solution is shown in Fig. 5-17

![Graph showing Hamiltonian function](image)

**Fig. 5-17** The time value of Hamilton function of the new solution

Figure 5-17 indicates that the solution obtained from optimizing the one control variable problem is near-optimal. Because the solution is approximately the same as the solution obtained from optimizing the original problem, the original solution is then verified to be an approximate local minimum.

The optimal trajectory we obtained is a one-burn optimal trajectory. The thruster functions from the start of the mission to the end and always produces the maximum-thrust. The flight time is minimized but not the fuel consumption. Actually, it is more fuel-efficient to use many short duration burns instead of one long duration burn and a varied thrust level. The penalty for the enhanced fuel efficiency is an increase in the mission time. In fact, the most fuel-efficient strategy is to use an infinite number of infinitesimal thrust arcs with infinitesimal thrust which results in an infinite flight time. This is not practical of course. However, a practical and most-
fuel efficient-maneuver can be optimized by fixing the flight time and results in a minimum-fuel problem.

Although we solve an escape transfer problem instead of the original Earth-Moon transfer problem, it is still interesting to examine the results computed by others for the Earth-Moon transfer problem. Betts [16] solved the problem by using his SOCS software. He computed the problem on a DEC Alpha machine with 8 Gbytes of memory in 35 hours. He optimized both the two-burn minimum-fuel and minimum-time problem. In the two-burn minimum-time problem, the mission time is 198.384 days and the spacecraft consumes 75.335 kg propellant. Schoenmaekers [1], et al. used a pragmatic engineering approach without applying optimization techniques. The transfer time is 17 months and the fuel demand ranges from 54.3 kg to 60.9 kg depending on the launch date.
Chapter 6

Conclusions and Recommendations for Future Research

We develop a homotopy optimization method which solves a complicated trajectory optimization problem from the solution of another simple problem. In particular, we develop the thrust modification method which solves a low-thrust trajectory optimization problem from the solution of a relatively high-thrust problem. The link between the high-thrust problem and the original low-thrust problem is a chain of problems with a gradually decreasing thrust level. The optimization program obtains the solution of a low-thrust problem by first solving its counterpart high-thrust problem and then traversing through the problem chain. The method enables a straightforward means to optimize a complex and large-scale low-thrust problem without purposely constructing the initial guess. The method is first applied on a two-dimensional two body escape problem to find the minimum transfer time, which is 51.2648 TU. The optimality of the solution is checked by the direct examination of the Hamiltonian information. The time value of the Hamiltonian indicates the solution is a local optimum. The method is then applied to a realistic force three-dimensional problem based on the model of SMART-1. We obtained the solution by solving a chain of 96 optimization problems. For a one burn minimum-time optimal trajectory, the flight time is 16155761.9 s which is roughly 187 days. The fuel consumption is 75.7 kg and is 21.6% of the initial mass of the spacecraft. The check of optimality is done in an indirect way. The result is compared with another result that can be directly checked against optimality. The comparison shows that the two results are approximately the same. The original result was thus verified to be a local optimum.
As discussed in section 3.2, homotopy method combined with deterministic optimization techniques can help lead to the convergence of a local optimum. To help lead to the convergence to the global minimum, stochastic optimization techniques must be applied to enclose the global minimum. Whereas a local homotopy method generates a sequence of local optima that converges to a local optimum of the final problem, a global homotopy method generate a sequence of ensembles of local optima that converges to an ensemble of local optima of the final problem. A global optimization algorithm, such as Genetic Algorithm, can be used to generate an ensemble of local optima for the next problem based on the ensemble of optima found for the last problem. But since the number of optima in an ensemble will grow exponentially as the optimization process continues, limiting the number of optima in an ensemble is necessary. However, by doing this, the probability of convergence to the global optimum is reduced. The extent of limiting the number in an ensemble is the balance between computing time and the result-quality requirement. Thus, whether the real global optimum can be found depends heavily on the performances of the search algorithms. Contrary to the possible probability-one convergence to a local optimum in a local homotopy method, the convergence to the global optimum can not be guaranteed in anyway in a global homotopy method. This is not only because a global optimization algorithm may fail to enclose the global optimum in a single optimization but also because traces all the existing paths of optimum for the previous problem doesn’t consider that new optimum may appear in some new regions in the search space as the problem evolves. The global optimum may actually comes from those regions thus has lower probability of being traced if the path-tracing starts from some search point that is far away from these regions in the search space. Although a global homotopy method can not guarantee finding a global optimum, a global optimization method can surely outperform a local homotopy method in that a collection of candidate solutions instead of a single solution is obtained for the user.
References


