SUCCESSIVE BACKWARD SWEEP METHODS FOR OPTIMAL CONTROL OF NONLINEAR SYSTEMS WITH CONSTRAINTS

A Dissertation

by

DONGHYURN CHO

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Srinivas Rao Vadali
Committee Members,	Shankar P. Bhattacharyya
	Raktim Bhattacharya
	John Hurtado
Head of Department,	Rodney Bowersox

August 2013

Major Subject: Aerospace Engineering

ABSTRACT

Continuous and discrete-time Successive Backward Sweep (SBS) methods for solving nonlinear optimal control problems involving terminal and control constraints are proposed in this dissertation. They closely resemble the Neighboring Extremals and Differential Dynamic Programming algorithms, which are based on the successive solutions to a series of linear control problems with quadratic performance indices. The SBS methods are relatively insensitive to the initial guesses of the state and control histories, which are not required to satisfy the system dynamics. Hessian modifications are utilized, especially for non-convex problems, to avoid singularities during the backward integration of the gain equations. The SBS method requires the satisfaction of the Jacobi no-conjugate point condition and hence, produces optimal solutions. The standard implementation of the SBS method for continuous-time systems incurs terminal boundary condition errors due to an algorithmic singularity as well as numerical inaccuracies in the computation of the gain matrices. Alternatives for boundary error reduction are proposed, notably the aiming point and the switching between two forms of the sweep expansion formulae. Modification of the sweep formula expands the domain of convergence of the SBS method and allows for a rigorous testing for the existence of conjugate points.

Numerical accuracy of the continuous-time formulation of the optimal control problem can be improved with the use of symplectic integrators, which generally are implicit schemes in time. A time-explicit group preserving method based on the Magnus

series representation of the state transition is implemented in the SBS setting and is shown to outperform a non-symplectic integrator of the same order.

Discrete-time formulations of the optimal control problem, directly accounting for a specific time-stepping method, lead to consistent systems of equations, whose solutions satisfy the boundary conditions of the discretized problem accurately. In this regard, the second-order, implicit mid-point averaging scheme, a symplectic integrator, is adapted for use with the SBS method. The performance of the mid-point averaging scheme is compared with other methods of equal and higher-order non-symplectic schemes to show its advantages. The SBS method is augmented with a homotopy-continuation procedure to isolate and regulate certain nonlinear effects for difficult problems, in order to extend its domain of convergence. The discrete-time SBS method is also extended to solve problems where the controls are approximated to be impulsive and to handle waypoint constraints as well.

A variety of highly nonlinear optimal control problems involving orbit transfer, atmospheric reentry, and the restricted three-body problem are treated to demonstrate the performance of the methods developed in this dissertation.

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude and thanks to my advisor Dr. Srinivas R . Vadali. This dissertation would not be possible without his guidance, patience and profound knowledge. He always showed me a new direction whenever I needed it. I heartily thank him for his effort in directing me in my research.

I also would like to thank Dr. Shankar P. Bhattacharyya, Dr. Raktim Bhattacharya and Dr. John. E. Hurtado who served as my committee members. Also I would like to thank Dr. John L. Junkins, who taught me celestial mechanics and estimation courses.

Thanks also go to my friends and colleagues and the faculty and staff of the Department of Aerospace Engineering for making my time at Texas A&M University a great experience.

Finally, thanks to my mother and father for their encouragement and to my wife for her patience and love.

NOMENCLATURE

AP	Aiming Point
CP	Continuation Parameter
DDP	Differential Dynamic Programming
HDDP	Hybrid Differential Dynamic Programming
LQ	Linear Quadratic
MI	Magnus Integrator
MSBS	Modified Successive Backward Sweep
OCP	Optimal Control Problem
ODV	Optimal Descent Vector
RK 5^{th}	The 5^{th} Order Runge Kutta Algorithm
RTBP	Restricted Three Body Problem
SBS	Successive Backward Sweep
SDRE	State Dependent Riccati Equation
SF	Switching Function
STM	State Transition Matrix
TPBVP	Two Point Boundary Value Problem

TABLE OF CONTENTS

Page

ABSTRACT ii

ACKNOWLEDGEMENTS iv

NOMENCLATURE v

TABLE OF CONTENTS vi

LIST OF FIGURES ix

LIST OF TABLES xvi

1. INTRODUCTION 1

    1.1 Overview 5

2. CONTINUOUS-TIME FORMULATION OF THE SBS METHOD 8

    2.1 Continuous-time nonlinear optimal control 8
    2.2 Sufficient conditions and neighboring extremal paths 10
    2.3 The SBS method 14
    2.4 Input update and Hessian modification 16
    2.5 Evaluation of the performance index 18
    2.6 Methods for determining ν 19
        2.6.1 Pseudoinverse method 19
        2.6.2 Aiming point method 20
    2.7 The modified SBS method 22
    2.8 The SBS method with control constraints 24
    2.9 Numerical examples 27
        2.9.1 Example 2-1: A 1-D Nonlinear problem without state and control constraint 27
        2.9.2 Example 2-2: A 2-D nonlinear problem with final state constraints 31
        2.9.3 Example 2-3: A 1-D classical example 34
        2.9.4 Example 2-4: Earth to Mars orbit transfer problem with control constraints 35

Page

3. The SBS METHOD INCORPORATING A MAGNUS INTEGRATOR 42

3.1 The Magnus integrator 42
3.2 The SBS method using a Magnus integrator 45
3.3 The modified SBS method using a Magnus integrator 46
3.4 Numerical examples 47
3.4.1 Example 3-1: A 2-D nonlinear problem with final state constraints 47
3.4.2 Example 3-2: A hypersensitive problem 50
3.4.3 Example 3-3: The atmospheric reentry problem 51

4. DISCRETE TIME FORMULATION OF THE SBS METHOD 61

4.1 The discretized SBS method 61
4.2 The discretized SBS method for free final time problems 67
4.3 The discretized SBS method with impulsive control 72
4.4 The discretized SBS method with a waypoint scheme 75
4.5 Numerical examples 77
4.5.1 Example 4-1: A 2-D nonlinear problem with final state constraints 77
4.5.2 Example 4-2: A 2-D nonlinear problem with free final time 84
4.5.3 Example 4-3: A 2-D linear problem with final state constraints and bounded inputs 86
4.5.4 Example 4-4: A hypersensitive problem by using the waypoint scheme 89
4.5.5 Example 4-5: A linear problem with impulsive control 90

5. THE SBS METHOD AUGMENTED WITH A HOMOTOPY ALGORITHM. 93

5.1 Numerical examples 98
5.1.1 Example 5-1: A simple two point boundary value problem without control inputs 98
5.1.2 Example 5-2: The Earth to Mars orbit transfer problem 104
5.1.3 Example 5-3: Formulation of the orbit transfer problem using orbital elements 109

6. THE RESTRICTED THREE BODY PROBLEM (RTBP) 115

7. CONCLUSIONS 134

Page

REFERENCES 136

APPENDIX A 142

APPENDIX B 144

APPENDIX C 146

LIST OF FIGURES

FIGURES Page

2.1 Aiming point vector 20

2.2 State history for Example2-1..... 29

2.3 Input history for Example 2-1 29

2.4 Costate history for Example2-1..... 30

2.5 The cost value history for Example2-1 30

2.6 State history for Example2-2..... 32

2.7 Input history for Example 2-2 32

2.8 Costate history for Example2-2..... 33

2.9 The cost convergence vs. iterations for Example2-2 33

2.10 Input history for Example2-4 (Open-Loop)..... 40

2.11 Input history for Example2-4 (SBS-$\nu(t_0)$) 40

2.12 Input history for Example2-4 (SBS-Pseudoinverse)..... 41

2.13 Input history for Example2-4 (SBS-AP $\tilde{x}_f$)..... 41

3.1 State history for Example3-1..... 49

3.2 Input history for Example3-1 49

3.3 Costate history for Example 3-1..... 49

3.4 The cost value history for Example3-1 49

3.5 State history for Example3-2..... 51

3.6 Costate history for Example 3-2..... 51

Page

3.7 Altitude history for Example3-3 55

3.8 Longitude history for Example3-3 55

3.9 Latitude history for Example3-3 56

3.10 Velocity history for Example3-3 56

3.11 Flight path angle history for Example3-3 56

3.12 Heading angle history for Example3-3 56

3.13 The input history for Example 3-3 57

3.14 Cost convergence vs iterations for Example3-3 57

3.15 The terminal constraint violation $\|x(T) - x_f\|$ for Example 3-3 57

3.16 Neighboring reentry trajectories for Example3-3 (MSBS-MI) 59

4.1 The time index and time step of impulsive control 73

4.2 x_1 & x_2 Trajectory ($n = 5$, Euler forward method) 79

4.3 x_1 & x_2 Trajectory ($n = 5$, Euler backward method) 79

4.4 x_1 & x_2 Trajectory ($n = 5$, Trapezoidal rule) 79

4.5 x_1 & x_2 Trajectory ($n = 5$, Simpson's rule) 80

4.6 x_1 & x_2 Trajectory ($n = 5$, Midpoint rule) 80

4.7 λ_1 & λ_2 Trajectory ($n = 5$, Euler forward method) 80

4.8 λ_1 & λ_2 Trajectory ($n = 5$, Euler backward method) 80

4.9 λ_1 & λ_2 Trajectory ($n = 5$, Trapezoidal rule) 80

4.10 λ_1 & λ_2 Trajectory ($n = 5$, Simpson's rule) 80

Page

4.11 λ_1 & λ_2 Trajectory ($n=5$, Midpoint rule) 80

4.12 x_1 & x_2 Trajectory ($n=10$, Euler forward method) 81

4.13 x_1 & x_2 Trajectory ($n=10$, Euler backward method)..... 81

4.14 x_1 & x_2 Trajectory ($n=10$, Trapezoidal rule) 81

4.15 x_1 & x_2 Trajectory ($n=10$, Simpson's rule)..... 81

4.16 x_1 & x_2 Trajectory ($n=10$, Midpoint rule)..... 81

4.17 λ_1 & λ_2 Trajectory ($n=10$, Euler forward method) 81

4.18 λ_1 & λ_2 Trajectory ($n=10$, Euler backward method) 81

4.19 λ_1 & λ_2 Trajectory ($n=10$, Trapezoidal rule) 81

4.20 λ_1 & λ_2 Trajectory ($n=10$, Simpson's rule) 82

4.21 λ_1 & λ_2 Trajectory ($n=10$, Midpoint rule) 82

4.22 State history for Example4-2..... 85

4.23 Input history for Example4-2 85

4.24 Costate history for Example4-2..... 85

4.25 The cost value for Example4-2 85

4.26 Input history for Example4-3 (SBS- $\varepsilon = 1e-0, 1e-1, 1e-2$) 88

4.27 Input history for Example4-3 (open-loop) 88

4.28 State history for Example4-4..... 89

4.29 Input history for Example4-4 89

4.30 State trajectory for Example4-5 (n_{imp} =2) 91

Page

4.31 State trajectory for Example4-5 (n_{imp} =5) 91

4.32 State trajectory for Example4-5 (n_{imp} =10) 92

4.33 State trajectory for Example4-5 (n_{imp} =100) 92

5.1 The SBS method augmented by a homotopy algorithm 97

5.2 State history for Example5-1(SBS-homotopy) 102

5.3 State error history for Example5-1 $e = |x - x_a|$ (SBS-homotopy) 102

5.4 State history for Example5-1(ODV) 102

5.5 State error history for Example5-1 $e = |x - x_a|$ (ODV) 102

5.6 Trajectory for Example 5-2 (case 1) 106

5.7 Trajectory for Example 5-2 (case 2) 106

5.8 Trajectory for Example 5-2 (case 3) 107

5.9 Trajectory for Example 5-2 (naive initial guess) 108

5.10 Trajectory for Example 5-2 (case 1) 108

5.11 Trajectory for Example 5-2 (case 2) 108

5.12 Trajectory for Example 5-3 ($t_f = 15$ hours) 112

5.13 Trajectory for Example 5-3 ($t_f = 100$ hours) 112

5.14 Trajectory for Example 5-3 ($t_f = 500$ hours) 112

5.15 Trajectory for Example 5-3 ($t_f = 1000$ hours) 112

5.16 Input history for Example 5-3 ($t_f = 15$ hours) 113

Page

5.17 Input history for Example 5-3 ($t_f = 100$ hours)..... 113

5.18 Input history for Example 5-3 ($t_f = 500$ hours)..... 113

5.19 Input history for Example 5-3 ($t_f = 1000$ hours)..... 113

6.1 Trajectory1 for RTBP..... 120

6.2 Trajectory2 for RTBP..... 120

6.3 Trajectory3 for RTBP..... 120

6.4 Trajectory4 for RTB..... 120

6.5 Variation of the Jacobi Integral (Trajectory1)..... 121

6.6 Variation of the Jacobi Integral (Trajectory2)..... 121

6.7 Variation of the Jacobi Integral (Trajectory3)..... 121

6.8 Variation of the Jacobi Integral (Trajectory4)..... 121

6.9 Variation of $\det(\phi_{11})$ (Trajectory 1) 123

6.10 Variation of $\det(\phi_{12})$ (Trajectory 1) 123

6.11 Variation of $\det(\phi_{11})$ (Trajectory 2) 123

6.12 Variation of $\det(\phi_{12})$ (Trajectory 2) 123

6.13 Variation of $\det(\phi_{11})$ (Trajectory 3) 124

6.14 Variation of $\det(\phi_{12})$ (Trajectory 3) 124

6.15 Variation of $\det(\phi_{11})$ (Trajectory 4) 124

6.16 Variation of $\det(\phi_{12})$ (Trajectory 4) 124

Page

6.17 Condition # of ϕ_{11} (Trajectory1) 126

6.18 Condition # of ϕ_{12} (Trajectory1) 126

6.19 Condition # of ϕ_{11} (Trajectory2) 126

6.20 Condition # of ϕ_{12} (Trajectory2) 126

6.21 Condition # of ϕ_{11} (Trajectory3) 127

6.22 Condition # of ϕ_{12} (Trajectory3) 127

6.23 Condition # of ϕ_{11} (Trajectory4) 127

6.24 Condition # of ϕ_{12} (Trajectory4) 127

6.25 Trajectory (10days-free final velocity) 129

6.26 Trajectory (20days-free final velocity) 129

6.27 Trajectory (30days-free final velocity) 129

6.28 Trajectory (40days-free final velocity) 129

6.29 Trajectory (50days-free final velocity) 129

6.30 Trajectory (60days-free final velocity) 129

6.31 Jacobi constant (10days-free final velocity)..... 129

6.32 Jacobi constant (20days-free final velocity)..... 129

6.33 Jacobi constant (30days-free final velocity)..... 129

6.34 Jacobi constant (40days-free final velocity)..... 130

6.35 Jacobi constant (50days-free final velocity)..... 130

6.36 Jacobi constant (60days-free final velocity)..... 130

Page

6.37 Trajectory (10days-fixed final velocity) 131

6.38 Trajectory (20days-fixed final velocity) 131

6.39 Trajectory (30days-fixed final velocity) 131

6.40 Trajectory (40days-fixed final velocity) 131

6.41 Trajectory (50days-fixed final velocity) 131

6.42 Trajectory (60days-fixed final velocity) 131

6.43 Jacobi constant (10days-fixed final velocity)..... 132

6.44 Jacobi constant (20days-fixed final velocity)..... 132

6.45 Jacobi constant (30days-fixed final velocity)..... 132

6.46 Jacobi constant (40days-fixed final velocity)..... 132

6.47 Jacobi constant (50days-fixed final velocity)..... 132

6.48 Jacobi constant (60days-fixed final velocity)..... 132

LIST OF TABLES

TABLE		Page
2.1	Simulation conditions for Example 2-1	28
2.2	The results of simulation for Example 2-1	29
2.3	Simulation conditions for Example 2-2	32
2.4	The results of simulation for Example 2-2	32
2.5	Simulation parameters for Example 2-4	37
2.6	The boundary conditions for Example 2-4	37
2.7	Simulation results for Example 2-4	39
3.1	Simulation conditions for Example 3-1	48
3.2	The results of simulation for Example 3-1	48
3.3	Simulation conditions for Example 3-2	50
3.4	Simulation parameters for Example 3-3	53
3.5	Boundary conditions for Example 3-3	54
3.6	Simulation results for Example 3-3	55
3.7	Boundary satisfaction accuracy for Example 3-3	58
3.8	The average error in the final velocity for Example 3-3	60
4.1	Simulation results for Example 4-1	78
4.2	The results of $\varepsilon(x_i) = ch^{q(x_i)}$ between n=50 and n=100	83
4.3	The results of simulation for Example 4-2	85
4.4	Simulation conditions for Example 4-3	87

Page

4.5 The results of simulation for Example 4-3 88

4.6 Simulation conditions for Example 4-5 91

5.1 Simulation conditions for Example 5-1 101

5.2 The converged maximum error for Example 5-1 103

5.3 Boundary conditions for Example 5-2 105

5.4 Simulation conditions for Example 5-3 106

5.5 The initial costate and cost value for multiple minimum trajectories 107

5.6 Boundary conditions for Example 5-4 111

5.7 Simulation conditions and orbital parameters for Example 5-4 111

6.1 Non-dimensional parameters for the RTBP 116

6.2 Simulation conditions for the RTBP 119

6.3 The simulation results for the four local minimum solutions 122

6.4 The magnitude of $\det(\phi_{12})$ at $\det(\phi_{11}) = 0$ 125

6.5 The cost values at each final time (the free final velocity) 130

6.6 The cost values at each final time (the fixed final velocity) 132

1. INTRODUCTION

Over the last fifty years, many direct, indirect, and hybrid methods have been developed to solve optimal control problems (OCP) involving nonlinear dynamical systems with constraints [1-2]. Direct methods, such as collocation and transcription, convert the dynamic optimal control problem into a parameter optimization problem, involving discretized states and controls. The discretized problems are solved with the help of available nonlinear programming software. Indirect methods convert the OCP into a multi-point boundary value problem, involving the costate (adjoint) vector, by the application of Pontryagin's principle. A hybrid method is part direct and part indirect, i.e., the objective function is directly optimized, without the explicit satisfaction of one or more necessary conditions. Often, these methods deliver extremals that may not be optimal. A check for optimality is performed by testing a solution against the second-order necessary and sufficient conditions.

Several time-discretization approaches have also been proposed to convert optimal control problems into optimization problems, which are solved using nonlinear programming approaches. Alternatively, an OCP can be directly formulated in discrete-time using the discrete minimum principle [3]. Highly nonlinear optimal control problems such as those involving atmospheric reentry, interplanetary orbit transfer, and the restricted three-body problem (RTBP) prove challenging due to their sensitivity to initial guesses of the costate or control variables and the lack of continuity of solutions with respect to initial conditions and system parameters. These types of problem are

easily solved using any of the discretization approaches.

Methods based on the second variation enforce the necessary and sufficient conditions for optimality. Jacobson developed the second-order differential dynamic programming (DDP) method [4-5] to solve continuous and discrete-time, constrained nonlinear optimal control problems. In this approach, the principles of dynamic programming are applied to a converging sequence of perturbed problems, involving a linearized state model and a quadratic approximation of the performance index (LQ). Subsequently, the DDP approach has been extensively developed and applied, leading to the modified DDP [6] and the hybrid differential dynamic programming [7] (HDDP) methods. The DDP method shares many similarities with the method of neighboring extremals [8-14]. The neighboring extremal problem can be solved either by using a state transition matrix (STM) approach or the sweep method. The SBS method solves a sequence of neighboring extremal problems to achieve a solution to a nonlinear OCP by using the sweep method. The sweep method is based on the solution to a Riccati equation, which plays an important role in checking for the satisfaction of the sufficient condition for optimality. This method is the focus of study in this dissertation.

The suboptimal state dependent Riccati equation (SDRE) method [15-16] has found many applications in the aerospace field. This method, dependent on the factorization of a nonlinear system into a non-unique pseudo-linear form, has been applied to the solution of optimal control problems [17] by using a successive approximating sequence method. More recently, this approach has been extended to problems involving terminal constraints by Parsley and Sharma [18]. However, the

solutions obtained by these methods do not necessarily satisfy the sufficient conditions for optimality and they are not applicable to problems which require accurate satisfaction of terminal and control constraints. When applicable, the SDRE approach does provide a constructive method to determine an initial control approximation.

The success of the SBS method depends on the proper construction of the quadratic approximation of the original nonlinear problem, i.e., the Hessian for each sub-problem. At the beginning of the solution process, especially for a poor initial guesses of the state and control variables, a second-order approximation may result in large corrections, invalidating the linearization process and requiring the need for a step-size control algorithm. Hence, a first-order algorithm is recommended far from the optimal; the second-order terms accelerate convergence near the optimal solution. In this thesis, solutions to sensitive problems are obtained by initially neglecting some of the second-order contributions from the Hamiltonian to the LQ sub-problems.

In the continuous-time setting, the optimal feedback control determined for the linearized problem by the sweep method results in a boundary condition error, due to the inaccuracy of the numerical integration method and its effect on the solution to the terminal constraint Lagrange multiplier vector. Furthermore, the solution to the Riccati equation is not necessarily bounded, even in the absence of conjugate points. A simple method for applying the check for satisfaction of the Jacobi condition for nonlinear OCPs has been presented by Jo and Prussing [19]. Mereau and Powers [20] also present a specialized treatment of conjugate points for the LQ problem with terminal constraints. As suggested by these and other works related to the sweep method, it is beneficial to

use an alternate representation, requiring a reduced number of gains, for the costate vector as a function of the current state and the terminal constraint specifications. Furthermore, the Jacobi, no-conjugate point sufficient condition can be directly checked by following this approach. The method is adapted to develop a robust numerical method that can accommodate poor initial guesses for the state and control variables and satisfy the terminal constraints accurately. Sequential solutions of LQ problem are at the heart of the SBS method.

Methods based on the Magnus series expansion [21] generate high accuracy solutions to linear, time-varying differential equations, compared to those obtained from non-symplectic Runge-Kutta integrators of the same order. The Magnus series expansion represents the STM as a product of matrix exponentials, to a prescribed order of accuracy. The exponential of a the Hamiltonian matrix is symplectic, The Magnus series representation preserves the symplectic property of the STM for Hamiltonian systems even under truncation. The higher accuracy provided by the exponential integrators can be traded for larger step sizes of integration and smaller computation times.

Applying the optimal control necessary conditions to the discrete-time system has an advantage of minimizing boundary condition errors, since the dynamics of the integrator can be accounted for in the discretization scheme. As mentioned before, of special interest for the study of Hamiltonian systems are the symplectic integrators, which preserve certain constants of integration over a long duration during a numerical simulation. In general, symplectic schemes are time-implicit. However, for linear systems, they can be expressed as time-explicit schemes, provided an average control is

defined to make the system causal. The implicit midpoint rule has been used extensively for the study of discrete dynamical systems and optimal control [22]. Fortunately, since the SBS method progresses by successive linearization, the time-explicit form of the midpoint rule can be adopted.

The midpoint rule is also applied to problems involving impulsive controls [23]. Furthermore, it has been extended to handle waypoint constraints as well. The waypoint scheme allows one to deal with sensitive problems and long simulation times. Continuation or homotopy methods can also alleviate the sensitivity issues associated with many optimal control problems. A homotopy approach [24-25] in the context of the SBS method is developed to increase its convergence robustness. This method is of particular use for problems with multiple extrema of two body problems. It is possible to isolate for certain problems, the global minimum, as a continuation parameter is varied slowly. A variety of highly nonlinear examples are considered to demonstrate the performance of the SBS method, including atmospheric reentry and orbit transfer problems for two-and three-body problems.

The contents of the sections following are summarized in the following.

1.1 Overview

◦ Section 2

This section presents the continuous-time formulation of a standard optimal control problem as well as the first and second order necessary conditions. The neighboring extremal problem is discussed and the importance of its STM is highlighted

in connection with the check for conjugate points. The SBS method is presented next. Methods for limiting the input updates and Hessian modification are described. Several methods of handling singularities in the calculation of the terminal constraint Lagrange multipliers are also discussed. An alternate form of the gain equations is derived to facilitate the conjugate point condition check. This modification reduces the number of gains required to be computed over a large segment of the trajectory.

◦ Section 3

The adaptation of a 4^{th} order Magnus series representation scheme for the state transition matrix of the linearized problem under the standard and modified SBS applications is presented in this section. The accuracy offered by the Magnus series representation of the STM is evaluated.

◦ Section 4

Discrete-time formulations of the optimal control problems using the explicit midpoint rule is presented in this section. The application of the SBS method is presented for this and other time-explicit numerical integration schemes and performance comparisons are provided. The SBS method is extended to solve problems for which the control is a series of impulses. Finally, the waypoint scheme is developed to reduce convergence sensitivity associated with highly nonlinear problems.

◦ Section 5

The homotopy based SBS method is developed as another approach to solve long-duration, sensitive optimal control problems. The continuation parameter gradually modulates the nonlinearity in the system. This approach is able to navigate around

multiple local minima to seek the optimal solution for an orbit transfer problem.

◦ Section 6

The low thrust transfer orbit from L1 and L2 liberation point in the Earth-moon system is considered. A modification to the performance index using the Jacobi constant is proposed as a means to improve convergence of the SBS method and reduce sensitivity for this class of problems. Multiple solutions are obtained to a liberation point transfer problem and the optimality of these trajectories is investigated by using the conjugate point check.

◦ Section 7

This section presents a summary of the results obtained and the concluding remarks.

2. CONTINUOUS-TIME FORMULATION OF THE SBS METHOD*

This section presents the continuous-time formulation of a standard optimal control problem, including the treatment of first and second-order necessary conditions. The features of the SBS method for solving continuous-time nonlinear optimal problems with terminal constraints directly follow from these developments.

2.1 Continuous-time nonlinear optimal control

A large class of continuous-time nonlinear optimal problems are formulated as follows [8]:

Determine the control input to minimize:

$$J = \phi[x(T)] + \int_{t_0}^{T} q(x,u,t)dt \tag{2.1}$$

Subject to:

$$\dot{x} = f(x,u,t) \tag{2.2}$$

$$\psi[x(T)] = \psi_f \tag{2.3}$$

where J is the performance index, x is the vector of state, u is a vector of control inputs, $\phi[x(T)]$ is the terminal penalty function, $f(x,u,t)$ is the vector of system dynamics, $\psi[x(T)]$ is a vector function of the final state, and ψ_f is a constant. It is

*Part of this section is reprinted with permission from "The Successive Backward Sweep Method for Optimal Control of Nonlinear Systems with Constraints" by Cho,D.H, Vadali,S.R, 2013. Advances in the Astronautical Sciences, Volume 147, pp 163-183, Copyright [2013] by American Astronautical Society

assumed that the final time, T, is fixed and the terminal constraint vector is regular, i.e., its Jacobian is of full rank. The necessary and sufficient conditions for optimality are compactly presented in terms of the control Hamiltonian, defined as:

$$H(x,u,\lambda,t) = q(x,u,t) + \lambda^T f(x,u,t) \tag{2.4}$$

where and λ is the costate vector. The first-order necessary conditions can be obtained from the Hamiltonian as follows:

$$\dot{x} = H_\lambda = f \tag{2.5}$$

$$\dot{\lambda} = -H_x = q_x + f_x^{\ T}\lambda \tag{2.6}$$

$$u = \arg\min(H) \tag{2.7}$$

and the transversality condition

$$\lambda(T) = \phi_{x(T)} + \psi_{x(T)}^T \nu \tag{2.8}$$

where ν is a vector of Lagrange multipliers and the subscripts represent partial derivatives, e.g., H_λ is the gradient of H with respect to λ. In the absence of control constraints, to first order, Eq. (2.7) is equivalent to

$$0 = H_u = q_u + f_u^T \lambda \tag{2.9}$$

2.2 Sufficient conditions and neighboring extremal paths

A solution satisfying the necessary conditions of Eqs. (2.5-2.8) has to be tested further for the satisfaction of the sufficient conditions for the existence of neighboring extremal paths in a weak sense (allowing small variations in the states and controls). As shown in reference [8], these are 1) the strengthened Legendre-Clebsch condition $H_{uu}(t) > 0$, 2) normality condition, 3) the Jacobi no-conjugate point condition. These three conditions are related to the existence of neighboring extremal paths, which can be obtained from a linearization of Eqs. (2.5-2.8). The linearized system is governed by the Hessian matrix entries, $H_{xx}, H_{xu}, H_{x\lambda},$ and H_{uu}. Along the neighboring extremal paths, the costate and final boundary condition deviations can be expressed as follows:

$$\delta\lambda(t) = S\delta x(t) + P\delta\nu \tag{2.10}$$

$$\delta\psi_f = P^T\delta x(t) + W\delta\nu \tag{2.11}$$

where $\delta x(t), \delta\lambda(t)$ are the perturbed state and costate vectors, $\delta\nu$ is the perturbed Lagrange multiplier, and $\delta\psi_f$ is the terminal constraint deviation vector. The Jacobi condition is

$$\overline{S} \triangleq S - PW^{-1}P^T \quad \text{finite for} \quad t_0 \le t < t_f \tag{2.12}$$

The Jacobi condition can be related to the elements of the STM $\phi = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix}$,

defining the backward-time relationship obtained by back integration.

$$\begin{bmatrix} \delta x(t) \\ \delta\lambda(t) \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix} \begin{bmatrix} \delta x(t_f) \\ \delta\lambda(t_f) \end{bmatrix} \tag{2.13}$$

where $\phi(t_f) = \begin{bmatrix} \phi_{11}(t_f) & \phi_{12}(t_f) \\ \phi_{21}(t_f) & \phi_{22}(t_f) \end{bmatrix} = I$, and I is the identity matrix of appropriate

dimension. Written out in scalar form, these relationships are

$$\delta x(t) = \phi_{11}\delta x(t_f) + \phi_{12}\delta\lambda(t_f) \tag{2.14}$$

$$\delta\lambda(t) = \phi_{21}\delta x(t_f) + \phi_{22}\delta\lambda(t_f) \tag{2.15}$$

Equation (2.14) results in

$$\delta x(t_f) = \phi_{11}^{-1}[\delta x(t) - \phi_{12}\delta\lambda(t_f)] \tag{2.16}$$

Substituting Eq. (2.16) into Eq. (2.15), one obtains the result

$$\delta\lambda(t) = \phi_{21}\phi_{11}^{-1}\delta x(t) + (\phi_{22} - \phi_{21}\phi_{11}^{-1}\phi_{12})\delta\lambda(t_f) \tag{2.17}$$

Focusing for simplicity on linear terminal constraints of the type $x(t_f) = x_f$, it can be shown that $\delta\nu = \delta\lambda(t_f)$. Equations (2.10) and (2.17) show that

$$S = \phi_{21}\phi_{11}^{-1} \tag{2.18}$$

$$P = \phi_{22} - \phi_{21}\phi_{11}^{-1}\phi_{12} \tag{2.19}$$

$$W = -\phi_{11}^{-1}\phi_{12} \tag{2.20}$$

Equation (2.10) also can be expressed in terms of $\delta x(t)$ and $\delta x(t_f)$ by the use of Eq. (2.11), since for linear terminal constraints, $\delta\psi_f = \delta x(t_f)$. Thus, an alternate representation for $\delta\lambda(t)$ is obtained as follows:

$$\delta\lambda(t) = (S - PW^{-1}P^T)\delta x(t) + PW^{-1}\delta x(t_f) = \bar{S}\delta x(t) + \bar{P}\delta x(t_f) \tag{2.21}$$

where

$$\bar{S} = S - PW^{-1}P^T \tag{2.22}$$

$$\bar{P} = PW^{-1} \tag{2.23}$$

Equation (2.21) can be used to determine the neighboring feedback control, if the gains $\bar{S}$ and $\bar{P}$ are directly available. In that case, there is no need to compute W and its inverse.

It also can be shown from Eqs. (2.14-2.15) that

$$\delta\lambda(t_f) = \phi_{12}^{-1}[\delta x(t) - \phi_{11}\delta x(t_f)] \tag{2.24}$$

and

$$\delta\lambda(t) = \phi_{22}\phi_{12}^{-1}\delta x(t) + (\phi_{21} - \phi_{22}\phi_{12}^{-1}\phi_{11})\delta x(t_f) \tag{2.25}$$

The gain matrices $\bar{S}$ and $\bar{P}$ can also be expressed in terms of the elements of STM by comparing Eqs. (2.21) and (2.25):

$$\bar{S} = \phi_{22}\phi_{12}^{-1} \tag{2.26}$$

$$\bar{P} = \phi_{21} - \phi_{22}\phi_{12}^{-1}\phi_{11} \tag{2.27}$$

The Jacobi condition, Eq. (2.12), requires that the matrix $\bar{S}$ remain finite in the interval $t_0 \le t < t_f$. This condition is satisfied if ϕ_{12} remains nonsingular. Since $\bar{S}$ cannot be calculated near t_f from Eq. (2.26), due to a singularity, it must be indirectly computed from Eq. (2.22) over a finite time period, during the back integration. It is important to note that even though $S(t)$ may become unbounded at a particular time, the matrix $\bar{S}$ can remain finite. Hence, a switch from Eq. (2.22) to Eq. (2.26) can be made as soon as the conditioning of the matrix ϕ_{12} allows it. Equation (2.18) shows that $S(t)$ becomes unbounded if ϕ_{11} is singular. Occurrence of singularities ϕ_{12} in $t_0 \le t \le t_f$ indicates the existence of conjugate points.

2.3 The SBS method

The SBS method proceeds with a linearization of the nonlinear Hamiltonian system and boundary conditions, given by Eqs. (2.5-2.8), about a nominal reference trajectory:

$$\dot{x} = H_{\lambda x} x + H_{\lambda u} u + \alpha \tag{2.28}$$

$$\dot{\lambda} = -(H_{xx} x + H_{xu} u + H_{x\lambda} \lambda + \beta) \tag{2.29}$$

$$H_u = 0 = H_{ux} x + H_{uu} u + H_{u\lambda} \lambda + \gamma \tag{2.30}$$

$$\lambda(T) = [\phi_{\bar{x}\bar{x}} + (\bar{\nu}^T \psi_{\bar{x}})_{\bar{x}}]\Big|_{t=T} x(T) + \psi^T_{\bar{x}(T)} \nu + [\phi_{\bar{x}} - (\phi_{\bar{x}\bar{x}} + (\bar{\nu}^T \psi_{\bar{x}})_{\bar{x}})\bar{x}]\Big|_{t=T} \tag{2.31}$$

where

$$\alpha = H_\lambda - H_{\lambda x}\bar{x} - H_{\lambda u}\bar{u} \tag{2.32}$$

$$\beta = H_x - H_{xx}\bar{x} - H_{xu}\bar{u} - H_{x\lambda}\bar{\lambda} \tag{2.33}$$

$$\gamma = H_u - H_{ux}\bar{x} - H_{uu}\bar{u} - H_{u\lambda}\bar{\lambda} \tag{2.34}$$

and $\bar{x}, \bar{u}, \bar{\lambda}$ indicate the nominal state, input, and costate, respectively and α, β, γ represent the nonlinearity of the state, input and costate. The control input u can be calculated from Eq. (2.30) as follows:

$$u = -H_{uu}^{-1}(H_{ux} x + H_{u\lambda} \lambda + \gamma) \tag{2.35}$$

The backward sweep method is based on the costate and linearized terminal constraint Eq. (2.3) representations of the form:

$$\lambda = Sx + P\nu + V \tag{2.36}$$

and

$$\psi(\bar{x}(T)) + \psi_{\bar{x}(T)}[x(T) - \bar{x}(T)] - \psi_f = 0 = P^T x + N + W\nu - \psi_f \tag{2.37}$$

Equation (2.36) is substituted into Eq. (2.29) to obtain the following differential equations for the gains. The terminal boundary conditions of the gains can be determined from Eqs. (2.31) and (2.37).

$$\dot{S} + SH_{\lambda x} + H_{x\lambda}S - (H_{xu} + SH_{\lambda u})H_{uu}^{-1}(H_{ux} + H_{u\lambda}S) + H_{xx} = 0, \quad S(T) = [\phi_{\bar{x}\bar{x}} + (\bar{\nu}^T\psi_{\bar{x}})_{\bar{x}}]\Big|_{t=T} \tag{2.38}$$

$$\dot{P} - [(H_{xu} + SH_{\lambda u})H_{uu}^{-1}H_{u\lambda} - H_{x\lambda}]P = 0, \quad P(T) = \psi_{\bar{x}(T)}^T \tag{2.39}$$

$$\dot{V} - [(H_{xu} + SH_{\lambda u})H_{uu}^{-1}H_{u\lambda} - H_{x\lambda}]V - (H_{xu} + SH_{\lambda u})H_{uu}^{-1}\gamma + S\alpha + \beta = 0, \quad V(T) = [\phi_{\bar{x}} - (\phi_{\bar{x}\bar{x}} + (\bar{\nu}^T\psi_{\bar{x}})_{\bar{x}})\bar{x}]\Big|_{t=T} \tag{2.40}$$

$$\dot{W} - P^T H_{\lambda u}H_{uu}^{-1}H_{u\lambda}P = 0, \quad W(T) = 0 \tag{2.41}$$

$$\dot{N} + P^T[\alpha - H_{\lambda u}H_{uu}^{-1}(\gamma + H_{u\lambda}V)] = 0, \quad N(T) = (\psi(\bar{x}) - \psi_{\bar{x}}\bar{x})\Big|_{t=T} \tag{2.42}$$

Equation (2.37) shows that ν can be computed at any time instant at which the matrix

W is nonsingular.

2.4 Input update and Hessian modification

In order to maintain the validity of the LQ approximation, and a weak control variation, the control update must be appropriately small in magnitude. This is achieved by introducing a new parameter a_u and the update rule

$$u = a_u u^* + (1 - a_u)\bar{u}\,, \qquad 0 \le a_u \le 1 \tag{2.43}$$

where u^* is the current computed input and $\bar{u}$ is the nominal input. The parameter a_u can be determined to minimize a cost function through a line search. A sufficient condition for the existence of a positive definite solution to the modified Riccati Eq. (2.38) is that H_{xx} be positive definite along the trajectory. During the initial convergence process, the nature of this matrix proves important and a method to shape it is discussed. A special modification to the performance index is proposed as follows:

$$J = \frac{1}{2}\int_0^T [x^T (Q + F(\bar{x}))x + u^T R u]dt \tag{2.44}$$

$$F(x) = a_{fx}(x^T x I - x x^T) \tag{2.45}$$

where a_{fx} is a non-negative coefficient related with F, a positive semi-definite matrix

function evaluated on the reference trajectory. The matrix F satisfies the property

$$F(x)x = 0 \tag{2.46}$$

Hence, F does not affect the performance index when $x = \bar{x}$, i.e., as the converged state is approached. The modification automatically vanishes upon convergence of the process.

One of the goals of this research is to develop a simple nonlinear optimal method that is relatively insensitive to initial guesses. For poor initial solutions, the high-order nonlinearity resulting from f and q often causes convergence difficulties. Therefore, terms involving the second partial derivatives of H, arising from the terms involving λ are neglected during the initial stages of computation:

$$H_{xx} = q_{xx} + (f_x^T \lambda)_x \tag{2.47}$$

$$H_{xu} = q_{xu} + (f_x^T \lambda)_u \tag{2.48}$$

$$H_{ux} = q_{ux} + (f_u^T \lambda)_x \tag{2.49}$$

$$H_{uu} = q_{uu} + (f_u^T \lambda)_u \tag{2.50}$$

In most cases, the second order terms are reinstated after sufficient progress toward reductions in the values of the performance index and constraint violations are achieved. For problems posed in the Mayer form, with $q_{uu} = 0$, a small correction term

is added to make $H_{uu} > 0$. For some highly nonlinear problems, a modification of the simple form is applied:

$$H_{xx} = H_{xx0} + C_{hxx} I \tag{2.51}$$

$$H_{uu} = H_{uu0} + C_{huu} I \tag{2.52}$$

where H_{xx0} and H_{uu0} are the elements of the original Hessian matrix and C_{hxx} and C_{huu} are the positive scalars, introduced to satisfy convexity conditions and achieve convergence from arbitrary initial guesses.

2.5 Evaluation of the performance index

Two different forms of the performance index are introduced to verify the accuracy of the converged solutions. For a quadratic index of the form

$$J_1 = \frac{1}{2}\int_0^T (x^T Q x + u^T R u) dt \tag{2.53}$$

associated with the linearized system of Eq. (2.28) and linear terminal constraint $x(T) = 0$, it is easy to verify that if the state-costate differential equations and the boundary conditions are satisfied, the cost-to-go function can be written as:

$$J_2(t) = \frac{1}{2} x^T(t)\lambda(t) + \frac{1}{2}\int_t^T \alpha^T \lambda dt \tag{2.54}$$

where α is defined by Eq. (2.32). Note that for systems which are linear in x and u to begin with, $\alpha = 0$. The derivation of Equation (2.54) is presented in Appendix A.

One way to evaluate the degree of sub-optimality of a solution is to determine the difference between the two indices. If the converged values of the two are the same, it can be concluded that an obtained solution has indeed been obtained, without having to verify the result via an open-loop solution. Numerical experiments show that J_2 is sensitive to integration error much more so than is J_1. Therefore, the difference between the two indices also indicates the degree of integration error.

2.6 Methods for determining ν

The Lagrange multiplier ν for each sub-problem is a constant vector, and it can be computed at any time other than the final time. However, due to numerical inaccuracies, it varies along the trajectory, if evaluated from Eq. (2.37). Typically, it is calculated from Eq. (2.37) at the initial time. It has been observed by numerical experiments that it is better to continuously update ν using any one of the following methods.

2.6.1. Pseudoinverse method

A pseudoinverse solution to the following equations is sought at each time

instant:

$$W\nu + Px + N = ax_f + b\bar{x}(T); \quad a+b=1; \qquad a=1 \tag{2.55}$$

where a and b are parameters with ideal values, $a=1$ and $b=0$. Equations (2.55) can be expressed in matrix form as:

$$\begin{bmatrix} W(t) & -x_f & -\bar{x}(T) \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \nu \\ a \\ b \end{bmatrix} = \begin{bmatrix} -P(t)x - N(t) \\ 1 \\ 1 \end{bmatrix} \tag{2.56}$$

The pseudoinverse method is not applicable near the final time. In the results provided in this method, the update for $\nu(t)$ is carried out partially over a domain, e.g., $t \in [t_0, 0.9T]$ and the value of $\nu(0.9T)$ is used in the region $t \in [0.9T, T]$.

2.6.2 Aiming point method

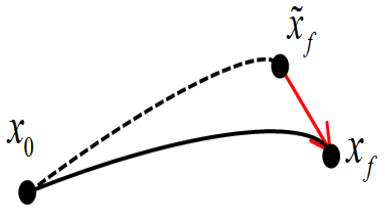


Figure 2.1: Aiming point vector

Figure 2.1 shows the relationship between the aiming point $\tilde{x}_f$ and the terminal

boundary point x_f. The aiming point is initially unknown, but is iteratively adjusted as in the augmented Lagrangian method or the method of multipliers [26]. For a finite scalar terminal penalty weight, there exists an aiming point which leads to the satisfaction of the terminal constraint. For the case of a quadratic terminal penalty function with a unit scalar weight, the multiplier ν is related to the miss distance from an aiming point through the transversality condition:

$$\lambda(T) = \nu = x(T) - \tilde{x}_f \tag{2.57}$$

In the aiming point method, $\tilde{x}_f$ is determined at each iteration, instead of ν. At the end of the solution process, ν satisfies the relationship: $\lambda(T) = \nu = x_f - \tilde{x}_f$. The sweep expansion for λ is a modified version of Eq. (2.36):

$$\lambda = Sx + P\tilde{x}_f + V \tag{2.58}$$

The substitution of Eq. (2.58) in Eq. (2.29) results in the following gain equations, some of which have been obtained previously.

$$\dot{S} + SH_{\lambda x} + H_{x\lambda}S - (H_{xu} + SH_{\lambda u}){H_{uu}}^{-1}(H_{ux} + H_{u\lambda}S) + H_{xx} = 0, \qquad S(T) = I \tag{2.59}$$

$$\dot{P} - [(H_{xu} + SH_{\lambda u}){H_{uu}}^{-1}H_{u\lambda} - H_{x\lambda}]P = 0, \qquad P(T) = -I \tag{2.60}$$

$$\dot{V} - [(H_{xu} + SH_{\lambda u}){H_{uu}}^{-1}H_{u\lambda} - H_{x\lambda}]V - (H_{xu} + SH_{\lambda u}){H_{uu}}^{-1}\gamma + S\alpha + \beta = 0 \quad V(T) = 0 \tag{2.61}$$

$$\dot{W} + P^T H_{\lambda u} H_{uu}^{-1} H_{u\lambda} P = 0\,, \qquad W(T) = 0 \quad (2.62)$$

$$\dot{N} - P^T [\alpha - H_{\lambda u} H_{uu}^{-1} (\gamma + H_{u\lambda} V)] = 0 \qquad N(T) = 0 \quad (2.63)$$

Except for the boundary conditions on Eq. (2.60), Eqs. (2.38-2.40) and Eqs. (2.59-2.61) are respectively very similar. Differences between Eqs. (2.41-2.42) and Eqs. (2.62-2.63), respectively, are limited to changes in the signs of certain expressions. The aiming vector $\tilde{x}_f$ can be calculated as:

$$\tilde{x}_f = W^{-1}[x_f + P^T x(t) - N] \quad (2.64)$$

Equation (2.64) is applicable for computing $\tilde{x}_f$ in a domain excluding the end point. Equation (2.64) shows that the aiming point is generally not the terminal boundary point.

2.7 The modified SBS method

This method implements the ideas presented in section 2.2 to eliminate the need for computing ν over a significant portion of the trajectory. ν is a constant vector and it can be computed at any time other than the final time by using Eq. (2.37) as follows:

$$\nu = -W^{-1}(P^T x + N - \psi_f) \quad (2.65)$$

Equation (2.65) is substituted into Eq. (2.36) to obtain

$$\lambda = (S - PW^{-1}P^T)x + PW^{-1}\psi_f + V - PW^{-1}N \tag{2.66}$$

Equation (2.66) can be represented in an alternate form as follows:

$$\lambda = \bar{S}x + \bar{P}\psi_f + \bar{V} \tag{2.67}$$

where

$$\bar{V} = V - PW^{-1}N \tag{2.68}$$

Equation (2.67), when substituted into Eq. (2.29), results in the following gain equations, which are similar to those obtained previously in Eqs. (2.38-2.40).

$$\dot{\bar{S}} + \bar{S}H_{\lambda x} + H_{x\lambda}\bar{S} - (H_{xu} + \bar{S}H_{\lambda u})H_{uu}^{-1}(H_{ux} + H_{u\lambda}\bar{S}) + H_{xx} = 0 \tag{2.69}$$

$$\dot{\bar{P}} - [(H_{xu} + SH_{\lambda u})H_{uu}^{-1}H_{u\lambda} - H_{x\lambda}]\bar{P} = 0 \tag{2.70}$$

$$\dot{\bar{V}} - [(H_{xu} + SH_{\lambda u})H_{uu}^{-1}H_{u\lambda} - H_{x\lambda}]\bar{V} - (H_{xu} + \bar{S}H_{\lambda u})H_{uu}^{-1}\gamma + \bar{S}\alpha + \beta = 0 \tag{2.71}$$

The advantage of switching to the modified sweep formulation is that only three gain equations (2.69-2.71) need be integrated, after the switch is made. Furthermore, the Jacobi, no-conjugate point sufficient condition, which requires that $\bar{S}(t)$ be finite along the trajectory, can be checked directly. However, since $\bar{S}(T)$ cannot be evaluated due to the singular boundary condition $W(T) = 0$, Eqs. (2.69-2.71) are applied over the domain,

$t \in [t_0, 0.9T]$ and the original gain differential equations (2.38-2.42) are used in the region $t \in [0.9T, T]$. The values of $S(t), P(t), V(t), W(t), N(t)$ serve to determine $\bar{S}(t), \bar{P}(t), \bar{V}(t)$ of Eqs. (2.22-2.23) and Eq.(2.68) at $t_1 = 0.9T$.

2.8 The SBS method with control constraints

The SBS method with bounded control inputs can also be applied to problems with control constraints. For each control, two possibilities have to be considered, depending on the control constraint being inactive or active.

$$u = -H_{uu}^{-1}(H_{ux}x + H_{u\lambda}\lambda + \gamma)$$

(control constraint inactive, same as Eq. (2.35))

$$u = \pm u_m \text{ (control constraint active)} \qquad (2.72)$$

Equations (2.35) and Eq. (2.72) can be combined into a single expression as follows:

$$u = P_c u_m - \pi\{H_{uu}^{-1}\}(H_{ux}x + H_{u\lambda}\lambda + \gamma) \qquad (2.73)$$

where P_c is a diagonal matrix with entries of ones or zeros, one for a saturated control and a zero otherwise. The operator $\pi\{\cdot\}$ introduces zeros along the row corresponding to a saturated control. For example, in the case of two inputs, if u_1 is saturated, P_c and π are written as follows:

$$P_c = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \pi\{\cdot\} = \begin{bmatrix} 0 & 0 \\ X & X \end{bmatrix} \tag{2.74}$$

where X is used to indicate the unchanged entries of its argument.

If the control input u_2 is saturated, P_c and π are written as

$$P_c = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad \pi\{\cdot\} = \begin{bmatrix} X & X \\ 0 & 0 \end{bmatrix} \tag{2.75}$$

If the control input u_1 and u_2 are saturated, P_c and π are

$$P_c = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \pi\{\cdot\} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.76}$$

Therefore, the general form of the SBS method can handle optimal problems with bounded control inputs. Equations (2.38-2.42) are generalized to account for control constraints as follows:

$$\dot{S} + SH_{\lambda x} + H_{x\lambda}S - (H_{xu} + SH_{\lambda u})\pi\{H_{uu}{}^{-1}\}(H_{ux} + H_{u\lambda}S) + H_{xx} = 0$$
$$S(T) = [\phi_{\bar{x}\bar{x}} + (\bar{\nu}^T \psi_{\bar{x}})_{\bar{x}}]\Big|_{t=T} \tag{2.77}$$

$$\dot{P} - [(H_{xu} + SH_{\lambda u})\pi\{H_{uu}{}^{-1}\}H_{u\lambda} - H_{x\lambda}]P = 0, \qquad P(T) = \psi^T_{\bar{x}(T)} \tag{2.78}$$

$$\dot{V} - [(H_{xu} + SH_{\lambda u})\pi\{H_{uu}^{-1}\}H_{u\lambda} - H_{x\lambda}]V - (H_{xu} + SH_{\lambda u})\pi H_{uu}^{-1}\gamma$$
$$+S\alpha + \beta + (H_{xu} + SH_{\lambda u})P_c u_m = 0 \qquad V(T) = [\phi_{\bar{x}} - (\phi_{\bar{x}\bar{x}} + (\bar{\nu}^T \psi_{\bar{x}})_{\bar{x}})\bar{x}]\Big|_{t=T} \quad (2.79)$$

$$\dot{W} - P^T H_{\lambda u}\pi\{H_{uu}^{-1}\}H_{u\lambda}P = 0, \qquad W(T) = 0 \quad (2.80)$$

$$\dot{N} + P^T[\alpha + H_{\lambda u}P_c u_m - H_{\lambda u}\pi\{H_{uu}^{-1}\}(\gamma + H_{u\lambda}V)] = 0, \; N(T) = (\psi(\bar{x}) - \psi_{\bar{x}}\bar{x})\big|_{t=T} \quad (2.81)$$

The gain differential equations for the aiming point method with bounded control inputs are expressed (instead of Eqs. (2.59-2.63)) as:

$$\dot{S} + SH_{\lambda x} + H_{x\lambda}S - (H_{xu} + SH_{\lambda u})\pi\{H_{uu}^{-1}\}(H_{ux} + H_{u\lambda}S) + H_{xx} = 0, \; S(T) = I \quad (2.82)$$

$$\dot{P} - [(H_{xu} + SH_{\lambda u})\pi\{H_{uu}^{-1}\}H_{u\lambda} - H_{x\lambda}]P = 0, \qquad P(T) = -I \quad (2.83)$$

$$\dot{V} - [(H_{xu} + SH_{\lambda u})\pi\{H_{uu}^{-1}\}H_{u\lambda} - H_{x\lambda}]V - (H_{xu} + SH_{\lambda u})\pi H_{uu}^{-1}\gamma$$
$$+S\alpha + \beta + (H_{xu} + SH_{\lambda u})P_c u_m = 0 \qquad V(T) = 0 \quad (2.84)$$

$$\dot{W} + P^T H_{\lambda u}\pi\{H_{uu}^{-1}\}H_{u\lambda}P = 0, \qquad W(T) = 0 \quad (2.85)$$

$$\dot{N} - P^T[\alpha + H_{\lambda u}P_c u_m - H_{\lambda u}\pi\{H_{uu}^{-1}\}(\gamma + H_{u\lambda}V)] = 0 \qquad N(T) = 0 \quad (2.86)$$

Finally, the gain differential equations of the modified SBS method with bounded control inputs can also be expressed (instead of Eqs. (2.69-2.71)) as follows:

$$\dot{\bar{S}} + \bar{S}H_{\lambda x} + H_{x\lambda}\bar{S} - (H_{xu} + \bar{S}H_{\lambda u})\pi\{H_{uu}^{-1}\}(H_{ux} + H_{u\lambda}\bar{S}) + H_{xx} = 0 \quad (2.87)$$

$$\dot{\bar{P}} - [(H_{xu} + \bar{S}H_{\lambda u})\pi\{H_{uu}^{-1}\}H_{u\lambda} - H_{x\lambda}]\bar{P} = 0, \quad (2.88)$$

$$\bar{V} - [(H_{xu} + \bar{S}H_{\lambda u})\pi\{H_{uu}{}^{-1}\}H_{u\lambda} - H_{x\lambda}]\bar{V} - (H_{xu} + \bar{S}H_{\lambda u})\pi H_{uu}{}^{-1}\gamma$$
$$+\bar{S}\alpha + \beta + (H_{xu} + \bar{S}H_{\lambda u})P_c u_m = 0 \tag{2.89}$$

2.9 Numerical examples

Several examples are presented to illustrate the performance of the SBS method for solving continuous-time OCPs. All the examples are simulated by using the 5th order RK (Fixed-step, 5th order part of the Runge-Kutta-Fehlberg method) algorithm [27].

2.9.1 Example 2-1: A 1-D Nonlinear problem without state and control constraints

This example was introduced by Jacobson [4] and it uses the hyperbolic tangent function to limit the effect of the control in the system. The system dynamics and cost function are:

$$\dot{x} = -0.2x + \tanh(u) \tag{2.90}$$

$$J = 10x^2(T) + \int_0^{0.5} (10x^2 + u^2)dt \tag{2.91}$$

The given initial state is $x(t_0) = 5$. The Hamiltonian for this problem is

$$H(x,u,\lambda,t) = 10x^2 + u^2 + \lambda(-0.2x + 10\tanh(u)) \tag{2.92}$$

The partial derivatives H_u and H_{uu} are

$$H_u = 2u + 10\lambda(1-\tanh^2(u)) = 0 \tag{2.93}$$

$$\lambda = -\frac{u}{5(1-\tanh^2(u))} \tag{2.94}$$

$$\begin{aligned} H_{uu} &= 2 - 20\lambda \tanh(u)(1-\tanh^2(u)) \\ &= 2 + 4u\tanh(u) > 0 \end{aligned} \tag{2.95}$$

Since $H_u = 0$ and the convexity condition $H_{uu} > 0$ is satisfied, the Hamiltonian is indeed minimized. Therefore, the optimal solution can be determined by using the SBS method. Simulation results are presented for the following conditions:

Table 2.1: Simulation conditions for Example 2-1

Variable	Value
Number of data points (n)	150/250/500/1000/5000
Input correction (a_u)	0.5

The nominal state and control are arbitrarily chosen to be $\bar{x} = 1$ and $\bar{u} = 1$. These choices are inconsistent with respect to Eq. (2.90). The simulation results are summarized in Table 2.2. Figs. 2.2-2.5 are simulated in n=5000.

Table 2.2: The results of simulation for Example 2-1

Method	n	λ_0	$x(T)$	J
Open-loop	5000	23.9866	0.0354	J_1=41.5953
SBS Method	150	24.0491	0.0387	J_1=41.6036, J_2=41.4307
SBS Method	250	23.9864	0.0349	J_1=41.5954, J_2=41.4906
SBS Method	500	23.9865	0.0351	J_1=41.5953, J_2=41.5431
SBS Method	1000	23.9866	0.0353	J_1=41.5953, J_2=41.5692
SBS Method	5000	23.9866	0.0354	J_1=41.5953, J_2=41.5901

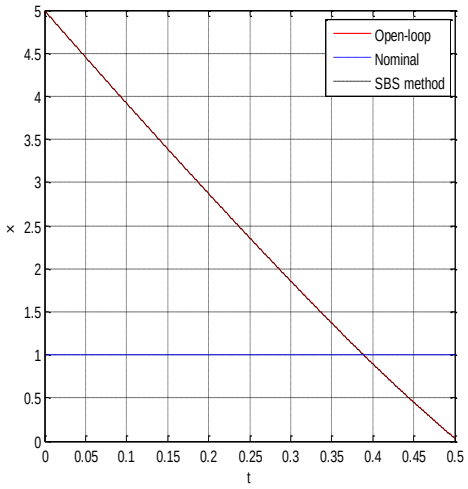


Figure 2.2: State history for Example2-1

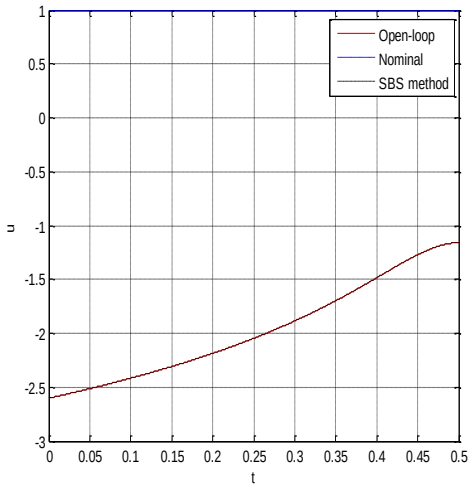
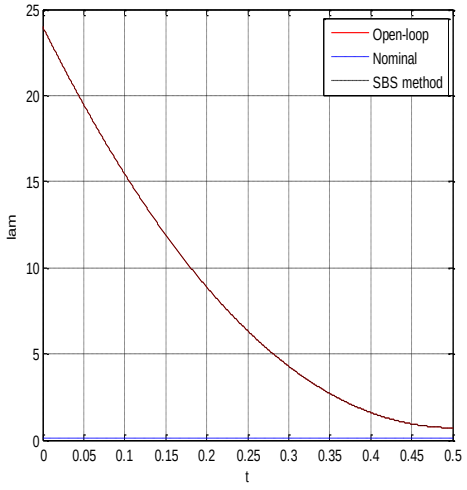


Figure 2.3: Input history for Example 2-1



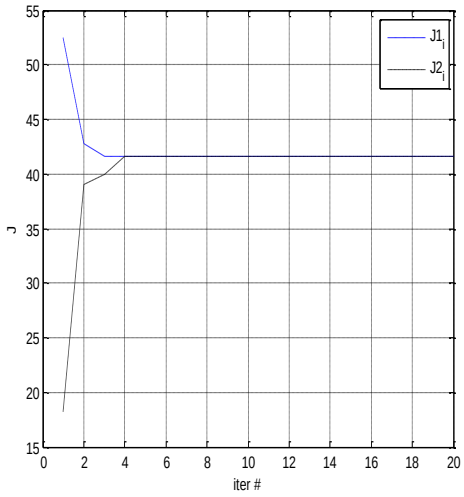


Figure 2.4: Costate history for Example 2-1 Figure 2.5: The cost value history for Example2-1

In Figs. 2.2-2.4, the red lines indicate the results of an open-loop solution, the blue dotted lines indicate the nominal trajectory variables, and the black dotted lines indicate the result obtained from the SBS method. The open-loop solution is obtained by using a single-shooting method initialized by the initial costate resulting from the SBS method. Figures 2.2-2.4 clearly show that the solutions from the shooting and SBS method agree. Figure 2.5 shows that the values of J_1 and J_2 differ significantly during the initial iterations, but converge on the 4th iteration. This indicates that the obtained solution is indeed an optimal solution.

However, if the number of data points is decreased from n=5000 to n=150, the numerical integration error is increased because of the large time step used. In this case, the discrepancy between J_1 and J_2 in Table 2.2 is indicative of the integration error. Table 2.2 shows that J_2 is more sensitive to numerical integration error than is J_1.

2.9.2 Example 2-2: A 2-D nonlinear problem with final state constraints

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_1 + x_1^{\,5} + u \end{aligned} \tag{2.96}$$

$$J = \frac{1}{2}\int_0^5 (x_2^{\,2} + u^2)dt \tag{2.97}$$

The given initial and final states are $x(t_0) = [1 \quad 1]$ and $x(T) = [0.5 \quad 0.5]$. This problem involves a highly nonlinear term (5^{th} order) and, unless the initial guess is excellent, the solution process requires a modification of the H_{xx} term to achieve convergence. The nominal trajectory is defined by arbitrary choices for the state, control, and costate histories: $\bar{x} = [1 \quad 1]$, $\bar{u} = 0.1$, and $\bar{\lambda} = [0.1 \quad 0.1]$. Since the terminal constraint is linear, $\nu = \lambda(T)$. In this situation, convergence can be achieved by using the F modification matrix, defined by Eq.(2.45). The simulation conditions are provided in Table 2.3.

Table 2.3: Simulation conditions for Example 2-2

Variable	Value
Number of data points (n)	500
Input correction (α_u)	1
Iteration number	20
The coefficients of F (a_{fx})	5/10/50/100

The results of simulation are summarized in Table 2.4.

Table 2.4: The results of simulation for Example 2-2

Method	λ_0	$\lvert x(T)-x_f \rvert$	J
Shooting	(25.044,9.420)	0.000	8.801
SBS Method	(25.059,9.410)	0.002	8.803

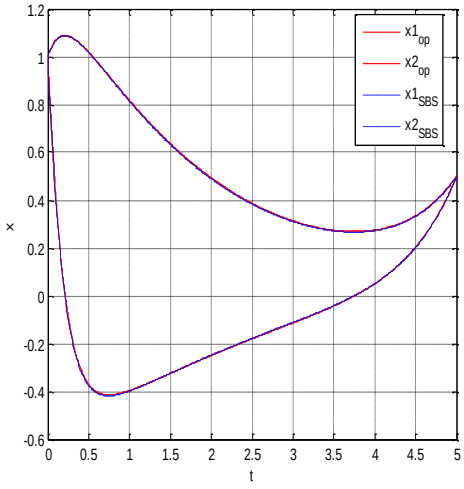


Figure 2.6: State history for Example2-2

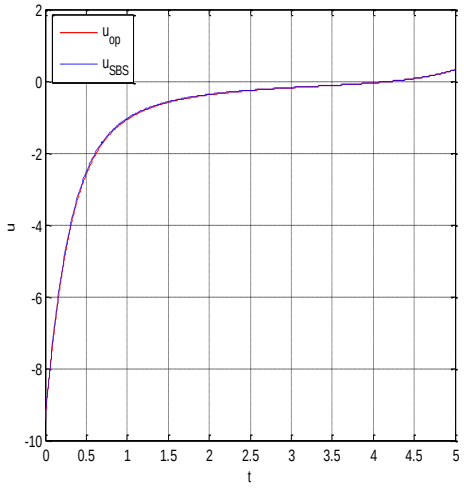


Figure 2.7: Input history for Example 2-2

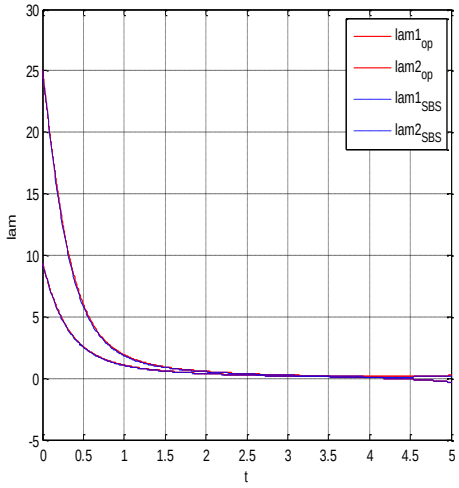


Figure 2.8: Costate history for Example2-2

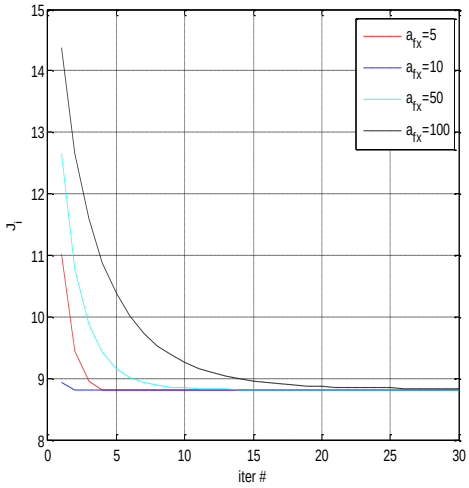


Figure 2.9: The cost convergence vs. iterations for Example2-2

In Figs. 2.6-2.8, the red lines indicate the results of the open-loop solution, the blue dotted lines indicate the nominal variables, and the black dotted lines indicate the results of the SBS method. The open-loop solution is obtained by using a single-shooting method based on the costate values obtained from the SBS method. Figures 2.6-2.8 show that the open-loop results match the respective SBS solutions exactly. Figure 2.9 shows the convergence history of J with respect to the number of iterations, as the parameter a_{fx} is varied. If the value of a_{fx} is too large, the convergence rate slows down.

2.9.3 Example 2-3: A 1-D classical example

This example serves to review the well-known result that an unbounded solution to the Riccati equation is not indicative of the presence of a conjugate point for a terminally-constrained OCP. Consider the OCP given by.

$$\dot{x} = u, \quad x(t_0) = x(\pi) = 0 \tag{2.98}$$

$$\text{Min} : J = \frac{1}{2}\int_0^{\pi} (u^2 - x^2) dt \tag{2.99}$$

This problem has been extensively treated in the literature, e.g., Bryson and Ho [8]. The solution to the problem by the application of the first-order necessary conditions leads to the following differential equation:

$$\ddot{x}(t) + x(t) = 0 \tag{2.100}$$

The Hamiltonian for this problem satisfies the convexity condition, $H_{uu} > 0$. Eq. (2.100) is satisfied by a family of solutions, given by

$$x(t) = a\sin(t) \tag{2.101}$$

where *a* is any constant, indicating that the initial time point is conjugate to the final time point. The cost *J*, on any solution to Eq. (2.100) is zero. The analytical solutions to the

gains are:

$$S(t) = -\tan(\pi - t);\ P(t) = \sec(\pi - t);\ W(t) = -\tan(\pi - t) \tag{2.102}$$

Equations (2.102) show that $S(t)$ is unbounded at $t = \frac{\pi}{2}$. However, $\bar{S}(t)$ is bounded for $0 < t < \pi$:

$$\bar{S}(t) = S(t) - P(t)W^{-1}(t)P^{T}(t) = \cot(\pi - t) \tag{2.103}$$

Therefore, back integration using the modified SBS method can proceed without encountering a singularity for $0 < t < \pi$, whereas the original SBS method cannot be used while approaching the point $t = \frac{\pi}{2}$.

2.9.4 Example 2-4: Earth to Mars orbit transfer problem with control constraints

The Earth to Mars orbit transfer problem with terminal constraints and a bounded input is considered. The difficulty with the solution to this problem stems from the bang-off-bang nature of the control and the lack of knowledge of the switching structure a priori. This example taken from Ref.[7] is presented to compare the performance of the SBS method implemented with several options for computing ν or $\tilde{x}_f$. The object of this problem is to minimize the final mass with a thrust-limited engine. The system

dynamic equations [7] are

$$\dot{x} = V_x \tag{2.104}$$

$$\dot{y} = V_y \tag{2.105}$$

$$\dot{z} = V_z \tag{2.106}$$

$$\dot{V}_x = -\mu_s \frac{x}{r^3} + \frac{T_x}{m} \tag{2.107}$$

$$\dot{V}_y = -\mu_s \frac{y}{r^3} + \frac{T_y}{m} \tag{2.108}$$

$$\dot{V}_z = -\mu_s \frac{z}{r^3} + \frac{T_z}{m} \tag{2.109}$$

$$\dot{m} = -\frac{T}{g_0 I_{sp}} \tag{2.110}$$

where x, y, z are the positions of vehicle, V_x, V_y, V_z are velocities of vehicle, μ_s is a gravitational constant, I_{sp} is the specific impulse, T is thrust, which can be expressed as follows:

$$T = \sqrt{T_x^{\,2} + T_y^{\,2} + T_z^{\,2}} \tag{2.111}$$

When the thrust is saturated T_x, T_y, T_z are defined as:

$$T_x = \frac{T_{\max} T_x}{\sqrt{T_x^2 + T_y^2 + T_z^2}}, \quad T_y = \frac{T_{\max} T_y}{\sqrt{T_x^2 + T_y^2 + T_z^2}}, \quad T_z = \frac{T_{\max} T_z}{\sqrt{T_x^2 + T_y^2 + T_z^2}} \qquad (2.112)$$

The parameters used in simulation are presented in the Table 2.5. The boundary conditions are represented in the Table 2.6.

Table 2.5: Simulation parameters for Example 2-4

Variables	Value
Maximum Thrust($T_{\max}$)	$0.5N$
Specific Impulse (I_{sp})	$2000s$
Initial mass (m_0)	$1000kg$
Time of flight(t_f)	$348.795days$

Table 2.6: The boundary conditions for Example 2-4

Variables	Initial values	Final values
x	-140,699,693(km)	-172,682,023(km)
y	-51,614,428(km)	176,959,469(km)
z	980(km)	7,948,912(km)
V_x	9.774596(km/s)	-16.427384(km/s)
V_y	-28.07828(km/s)	-14.860506(km/s)
V_z	$-4.337725*10^{-4}$(km/s)	$9.21486*10^{-4}$(km/s)

In this problem, the maximum thrust input is limited to 0.5N. Two different forms of control smoothing are used to deal with the discontinuous thrust. In the first approach, used to determine the open-loop solution, the input is approximated by using the exponential function [28].

$$T = \frac{T_{\max}}{1+\exp(SF/\varepsilon_1)} \tag{2.113}$$

where SF indicates a switching function and ε_1 is a continuation parameter. For the application of the SBS method, the problem formulation is modified by defining the performance index as follows:

$$J = -m(t_f) + \frac{1}{2}\varepsilon_2 \int_0^{t_f} (T_x^{\,2} + T_y^{\,2} + T_z^{\,2})dt \tag{2.114}$$

where ε_2 is a continuation parameter. The integration time step size is $\Delta t = t_f / 300$. The results obtained for the orbit transfer problem are presented in Table 2.7 for various combinations of methods and continuation parameter values.

Table 2.7: Simulation results for Example 2-4

Method	ε_1 or ε_2	$m(T)$	$\lvert x(T)-x_f \rvert$
Open-loop (ε_1)	$1e-0$	532.721	0.000
	$1e-1$	596.122	0.000
	$1e-2$	603.765	0.000
	$1e-3$	603.974	0.000
SBS method (ε_2) (ν evaluated at t_0)	$1e-0$	560.939	0.006
	$1e-1$	574.719	0.007
	$1e-2$	618.284	0.024
	$1e-3$	644.554	0.062
	$1e-4$	650.672	0.091
SBS method (ε_2) (pseudoinverse method for ν)	$1e-0$	561.417	0.008
	$1e-1$	575.386	0.008
	$1e-2$	613.814	0.011
	$1e-3$	624.023	0.023
	$1e-4$	624.487	0.022
SBS AP method (ε_2) (determination of $\tilde{x}_f$)	$1e-0$	561.423	0.008
	$1e-1$	575.703	0.008
	$1e-2$	613.992	0.012
	$1e-3$	623.093	0.022
	$1e-4$	624.934	0.023

The results presented in Table 2.7 are obtained without considering the 2nd order terms in the Hamiltonian. In this simulation, a relatively large time step size is used to reduce the simulation time. In this case, if the 2nd order terms are considered, the controls exhibit significant roughness. Furthermore, if ν is evaluated only at the initial

time, then relatively large boundary condition errors result. Table 2.7 shows that the boundary condition errors can be reduced by using the pseudoinverse and AP methods. The ν values are evaluated for an update region $t \in [t_0, 0.5T]$.

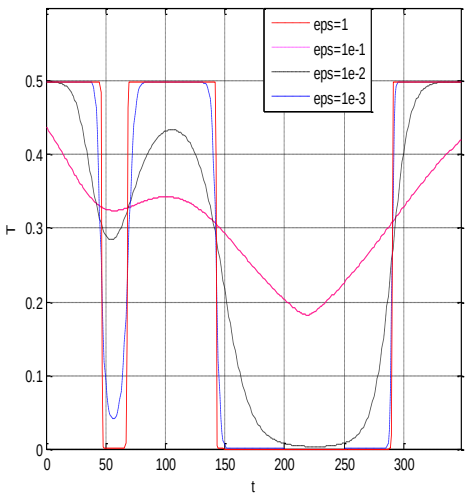


Figure 2.10: Input history for Example2-4 (Open-Loop)

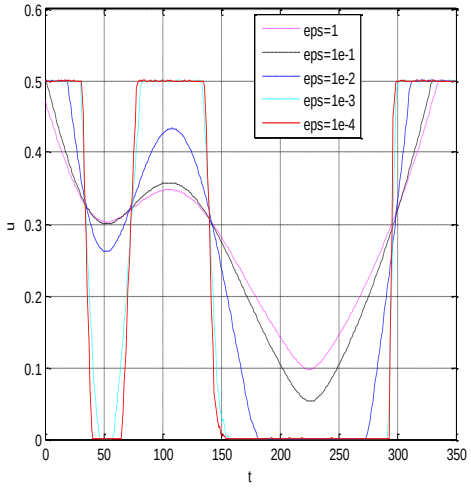


Figure 2.11: Input history for Example2-4 (SBS-$\nu(t_0)$)

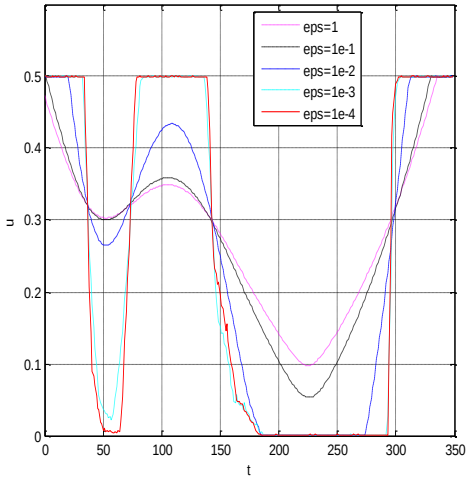


Figure 2.12: Input history for Example2-4 (SBS-Pseudoinverse)

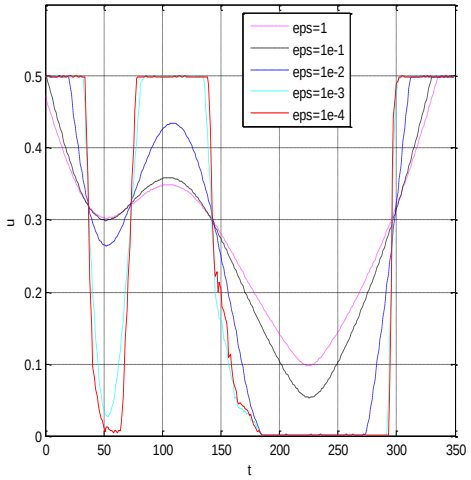


Figure 2.13: Input history for Example2-4 (SBS-AP $\tilde{x}_f$)

Figures 2.10-2.13 show that the continuation process has the ability to converge toward the exact solution. Figure 2.11 shows the input history if ν is evaluated only at the initial time. The sharp switching structure cannot be captured unless the integration step-size is reduced further. However, these converged solutions are good approximating solutions. If the SBS method switches to a multiple-shooting method near the end of iterative process, the true optimal solution can be easily obtained. Starting from the solution of Fig. 2.11, for $\varepsilon_2 = 1\times10^{-4}$, the optimal solution can be found after 38 iterations using multiple-shooting. For the remaining two methods(the pseudoinverse and AP methods), the optimal solution can be obtained in less than 30 iterations.

3. THE SBS METHOD INCORPORATING A MAGNUS INTEGRATOR*

In this section, the adaptation of the SBS method for use with a Magnus integrator is described.

3.1 The Magnus integrator

The Magnus integrator [21] is an exponential integrator, obtained by using the Magnus series expansion for the solution to a linear, time-varying matrix differential equation. This integrator is more accurate, for a given step size, than a polynomial integrator, e.g., the non-symplectic Runge Kutta method of the same order. Therefore, the boundary error resulting from numerical inaccuracies can be reduced with this approach. Equations (2.28-2.29) are represented to eliminate the control by substituting Eq. (2.30) as follows:

$$\dot{x} = \bar{A}x - \bar{B}\lambda + d_1 \tag{3.1}$$

$$\dot{\lambda} = -\bar{Q}x - \bar{A}^T\lambda - d_2 \tag{3.2}$$

where

$$\bar{A} = H_{\lambda}x - H_{\lambda u}H_{uu}{}^{-1}H_{ux} \tag{3.3}$$

$$\bar{B} = H_{\lambda u}H_{uu}{}^{-1}H_{u\lambda} \tag{3.4}$$

*Part of this section is reprinted with permission from "The Successive Backward Sweep Method for Optimal Control of Nonlinear Systems with Constraints" by Cho,D.H, Vadali,S.R, 2013. Advances in the Astronautical Sciences, Volume 147, pp 163-183, Copyright [2013] by American Astronautical Society

$$\bar{Q} = H_{xx} - H_{xu} H_{uu}^{-1} H_{ux} \tag{3.5}$$

$$d_1 = \alpha - H_{\lambda u} H_{uu}^{-1} \gamma \tag{3.6}$$

$$d_2 = \beta - H_{xu} H_{uu}^{-1} \gamma \tag{3.7}$$

Equations. (3.1-3.2) can be expressed in the form of a non-homogeneous linear matrix differential equation

$$\begin{bmatrix} \dot{x} \\ \dot{\lambda} \end{bmatrix} = \begin{bmatrix} \bar{A} & -\bar{B} \\ -\bar{Q} & -\bar{A}^T \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} + \begin{bmatrix} d_1 \\ -d_2 \end{bmatrix} \tag{3.8}$$

Equation (3.8) can be converted into the homogeneous equation by following arrangement [29]:

$$\begin{bmatrix} \dot{x} \\ \dot{\lambda} \\ 0 \end{bmatrix} = \begin{bmatrix} \bar{A} & -\bar{B} & d_1 \\ -\bar{Q} & -\bar{A}^T & -d_2 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \\ 1 \end{bmatrix} \tag{3.9}$$

Equation (3.9) is represented as

$$\dot{Y} = A(t, \bar{Y}) Y \tag{3.10}$$

where $Y = [x, \lambda, 1]^T$ and $A(t, \bar{Y})$ is a time-varying matrix of appropriate dimensions. The

solution to Eq. (3.10) is expressed as an explicit exponential mapping in discrete-time by using the nonlinear Magnus expansion as follows:

$$Y_{k+1} = e^{\Omega(h)} Y_k \tag{3.11}$$

In this research, the 4th order Magnus expansion for $\Omega(h)$ is calculated sequentially [30], with $s \in [0,\ h]$ as a dummy variable, representing time:

Order 1:

$$\Omega^{[1]}(s) = sA(t_k, \bar{Y}_k) + O(h^2) \tag{3.12}$$

Order 2:

$$\Omega^{[2]}(s) = \frac{s}{2}[A(t_k, \bar{Y}_k) + A(t_k + s, e^{\Omega^{[1]}(s)} \bar{Y}_k)] + O(h^3) \tag{3.13}$$

Order 3:

$$\begin{aligned} \Omega^{[3]}(s) = {} & \frac{s}{6}[A(t_k, \bar{Y}_k) + 4A_2(s/2) + A_2(s)] - \frac{s}{3}[\Omega^{[2]}(s/2), A_2(s/2)] \\ & - \frac{s}{12}[\Omega^{[2]}(s), A_2(s)] + O(h^4) \end{aligned} \tag{3.14}$$

Order 4:

$$\begin{aligned} \Omega^{[4]}(s) = {} & \frac{s}{6} A(t_k, \bar{Y}_k) + \frac{4s}{6} A_3(s/2) - \frac{s}{3}[\Omega^{[3]}(s/2), A_3(s/2)] \\ & + \frac{s}{18}[\Omega^{[3]}(s/2), [\Omega^{[3]}(s/2), A_3(s/2)]] + \frac{s}{6} A_3(s) \\ & - \frac{s}{12}[\Omega^{[3]}(s), A_3(s)] + \frac{s}{72}[\Omega^{[3]}(s), [\Omega^{[3]}(s), A_3(s)]] \end{aligned} \tag{3.15}$$

where

$$A_2(s) = A(t_k + s, e^{\Omega^{[2]}(s)}\bar{Y}_k), \quad A_3(s) = A(t_k + s, e^{\Omega^{[3]}(s)}\bar{Y}_k)$$

Substitution of Eqs. (3.12-3.14) into Eq. (3.15) leads to the following discrete-time mapping of Eq. (3.9):

$$\begin{bmatrix} x_{k+1} \\ \lambda_{k+1} \\ 0 \end{bmatrix} = \begin{bmatrix} F_{11} & F_{12} & D_{11} \\ F_{21} & F_{22} & D_{22} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ \lambda_k \\ 1 \end{bmatrix} \tag{3.16}$$

Equation (3.16) can be solved in partitioned form, as required for the application of the SBS method.

3.2 The SBS method using a Magnus integrator

Equation (3.16) can be solved in partitioned form, with λ_k assumed as in Eq. (2.36):

$$\lambda_k = S_k x_k + P_k \nu + V_k \tag{3.17}$$

The gain difference equations can be obtained by substituting Equation (3.17) into Equation (3.16):

$$S_k = (F_{22} - S_{k+1}F_{12})^{-1}(S_{k+1}F_{11} - F_{21}), \qquad S_n = [\phi_{\bar{x}\bar{x}} + (\bar{\nu}^T \psi_{\bar{x}})_{\bar{x}}]\Big|_{k=n} \quad (3.18)$$

$$P_k = (F_{22} - S_{k+1}F_{12})^{-1} P_{k+1}, \qquad P_n = \psi_{\bar{x}_n}^T \quad (3.19)$$

$$V_k = (F_{22} - S_{k+1}F_{12})^{-1}(V_{k+1} + S_{k+1}D_{11} - D_{22}), \quad V_n = [\phi_{\bar{x}} - (\phi_{\bar{x}\bar{x}} + (\bar{\nu}^T \psi_{\bar{x}})_{\bar{x}})\bar{x}]\Big|_{k=n} \quad (3.20)$$

$$W_k = W_{k+1} + P_{k+1}{}^T F_{12} P_k, \qquad W_n = 0 \quad (3.21)$$

$$N_k = N_{k+1} + P_{k+1}{}^T (F_{12}V_k + D_{11}), \qquad N_n = (\psi(\bar{x}) - \psi_{\bar{x}}\bar{x})\Big|_{k=n} \quad (3.22)$$

3.3 The modified SBS method using a Magnus integrator

In contrast to Eq. (3.17), the modified form of the gain equation is used over a restricted domain

$$\lambda_k = \bar{S}_k x_k + \bar{P}_k \psi_f + \bar{V}_k \quad [0 \le k \le n_1 - 1] \qquad (3.23)$$

where n is the index representing the final time and n_1 is the time index where the gains are switched during the sweep operation. Equation (3.17) is used for $[n_1 \le k \le n-1]$. The process for determining the gain difference equations is exactly the same as mentioned previously, and the results are

For $[0 \le k \le n_1 - 1]$

$$\bar{S}_k = (F_{22} - \bar{S}_{k+1}F_{12})^{-1}(\bar{S}_{k+1}F_{11} - F_{21}), \qquad \bar{S}_{n_1} = S_{n_1} - P_{n_1} W_{n_1}{}^{-1} P_{n_1} \quad (3.24)$$

$$\bar{P}_k = (F_{22} - \bar{S}_{k+1}F_{12})^{-1}\bar{P}_{k+1}, \qquad \bar{P}_{n_1} = P_{n_1}W_{n_1}^{-1} \quad (3.25)$$

$$\bar{V}_k = (F_{22} - \bar{S}_{k+1}F_{12})^{-1}(\bar{V}_{k+1} + \bar{S}_{k+1}D_{11} - D_{22}), \qquad \bar{V}_{n_1} = V_{n_1} - P_{n_1}W_{n_1}^{-1}N_{n_1} \quad (3.26)$$

The gain differential equations for $[n_1 \le k \le n-1]$ are defined as in Eqs. (3.18-3.22). The three gain Eqs. (3.24-3.26), are respectively identical in form to Eqs. (3.18-3.20). The boundary conditions for Eqs. (3.24-3.26) are determined by using the gain values obtained from Eqs.(3.18-3.22) at the switching point.

3.4 Numerical examples

Three numerical examples are presented to demonstrate the performance of the Magnus series-based modified SBS method. These examples are simulated by using the 5^{th} order Runge-Kutta algorithm as well as the 4^{th}-order Magnus integrator. The focus is on comparing the achieved accuracies in satisfying the terminal boundary conditions.

3.4.1 Example 3-1: A 2-D nonlinear problem with final state constraints

The problem of Example 2-2 is used to compare the performance of the modified SBS method. For this problem, the simple modification of Eq. (2.51) is applied: C_{hxx} is set to 5 at the beginning of the iterations and it is subsequently reduced to 0. The simulation conditions are provided in Table 3.1.

Table 3.1: Simulation conditions for Example 3-1

Variable	Value
Number of data points (n)	500/50
Number of iterations	20
Input correction (α_u)	1
Switching time to the modified SBS method (t_1)	$0.9T$

Table 3.2 presents the data on the boundary error norms for the various integration methods and procedures for calculating ν. The boundary error for each method is an indicator of its numerical accuracy.

Table 3.2: The results of simulation for Example 3-1

Method	Integrator	n	$x(T)-x_f$	Computation Time (normalized)
Original SBS; ν is evaluated at t_0	RK-5th	500	0.008	1
	MI	500	1.405e-15	9.93
	MI	50	1.435e-14	1.07
The modified SBS method	RK-5th	500	1.581e-6	2.43
	MI	500	1.554e-14	11.57
	MI	50	1.688e-13	1.07

* RK-5th : the 5th order Runge-Kutta algorithm , MI : Magnus Integrator

Regardless of the method used to solve for ν, the accuracy of the 4th order Magnus integrator is significantly higher than that of the RK-5th integrator, for the values of n considered. Moreover, the computational time required for the Magnus method can

be reduced by selecting larger step sizes. Even for the case of $n = 50$, the Magnus integrator is highly accurate, while its computational burden is nearly the same as that of the RK-5th integrator. The modified SBS method with the RK-5th integrator reduces the boundary error significantly. Figures 3.1-3.4 show the results of simulation by using the modified SBS method with the Magnus integrator for $n = 50$.

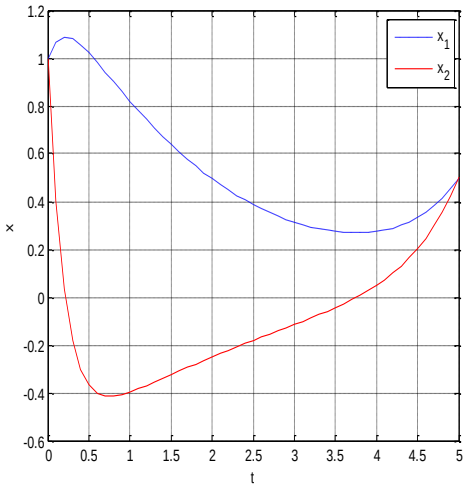


Figure 3.1: State history for Example3-1

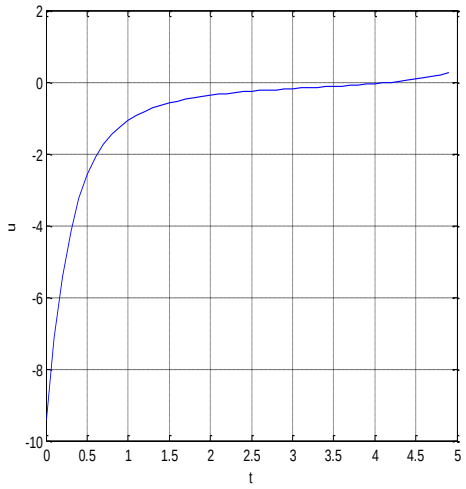


Figure 3.2: Input history for Example3-1

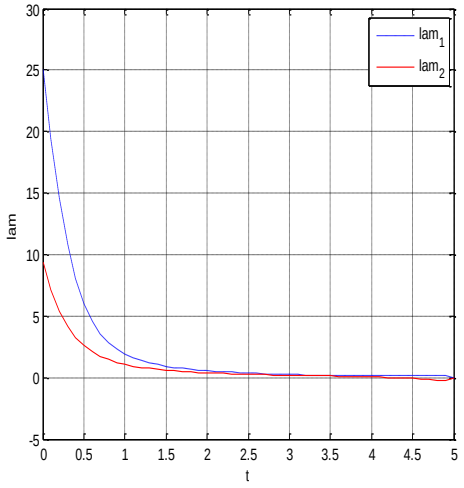


Figure 3.3: Costate history for Example3-1

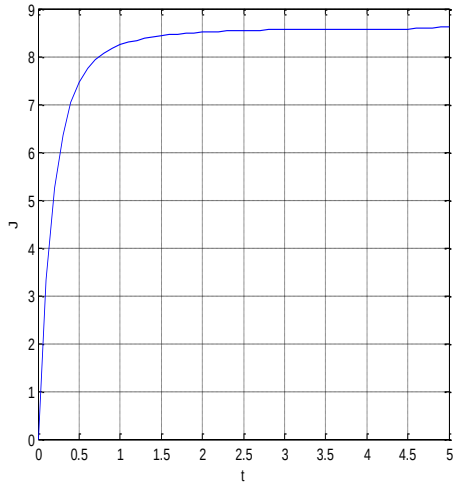


Figure 3.4: The cost value history for Example3-1

3.4.2 Example 3-2: A hypersensitive problem

Consider an example of a hypersensitive problem [31]. This problem is difficult to solve by using an indirect method. The system dynamics and cost function are

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_1 - {x_1}^3 + u \end{aligned} \tag{3.27}$$

$$J = \frac{1}{2}\int_0^{40} ({x_1}^2 + {x_2}^2 + u^2)dt \tag{3.28}$$

The given initial and final states are $x(t_0) = [1 \quad 0]$ and $x(T) = [0.75 \quad 0]$. The nominal trajectory variable histories are selected as for the previous example: $x_n = [1 \quad 1]$, $u_n = 0.1,\ \lambda_n = [0.1 \quad 0.1]$. The simulation parameters are provided in the Table 3.3.

Table 3.3: Simulation conditions for Example 3-2

Variable	Value
Data points (n)	500
Iteration number	10
Input correction (α_u)	1
Switching time to the modified SBS method (t_1)	$0.9T$

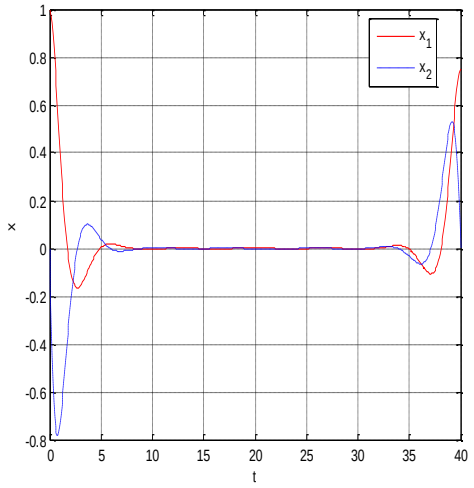


Figure 3.5: State history for Example3-2

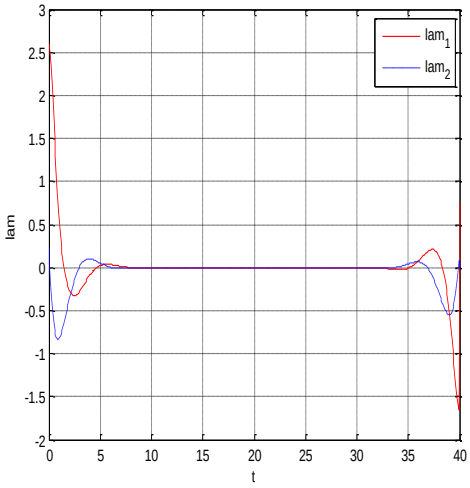


Figure 3.6: Costate history for Example 3-2

Figures 3.5-3.6 show the state and costate history of Example 3-2. The standard SBS method with the RK-5th integrator fails to produce a converged solution even after several attempts to modify the Hessian. However, the solution of this problem can be easily obtained when the modified SBS method with the Magnus integrator is used. The boundary error norm of the converged solution is $\Psi(T) = 8.623e-15$.

3.4.3 Example 3-3: The atmospheric reentry problem

Atmospheric reentry is a well-known example of a highly nonlinear problem. The system dynamic model is given by [32-33] the following equations:

$$\dot{r} = V \sin\gamma \tag{3.29}$$

$$\dot{\theta} = \frac{V \cos\gamma \cos\varphi}{r \cos\phi} \tag{3.30}$$

$$\dot{\phi} = \frac{V \cos\gamma \sin\varphi}{r} \tag{3.31}$$

$$\dot{V} = -\frac{\mu}{r^2}\sin\gamma - D \tag{3.32}$$

$$\dot{\gamma} = -\frac{\mu\cos\gamma}{r^2 V} + \frac{V}{r}\cos\gamma + \frac{L}{V}\cos\beta \tag{3.33}$$

$$\dot{\varphi} = -\frac{V}{r}\frac{\cos\gamma\cos\varphi\sin\phi}{\cos\phi} - \frac{L}{V}\frac{\sin\beta}{\cos\gamma} \tag{3.34}$$

where r is the position of vehicle, θ is the longitude, ϕ is the latitude, V is the velocity of vehicle, γ is the flight path angle, φ is the heading angle, μ is a gravitational constant and β is the control bank angle. The performance index to be minimized is a weighted combination of the aerodynamic loading and connective heating:

$$J = \int_0^T [\sqrt{L^2 + D^2} + \varepsilon\sqrt{\rho}V^3]dt \tag{3.35}$$

where ε is chosen to reflect equal weightage on the two factors. The density of atmosphere is defined as:

$$\rho = \rho_0 e^{-k(r - kr_e)} \tag{3.36}$$

where ρ_0 is the density at sea level, k is an atmospheric scale height, r_e is the radius of the Earth. The lift and drag forces per unit mass are represented as:

$$L = \frac{1}{2} C_l \rho S V^2 \tag{3.37}$$

$$D = \frac{1}{2} C_d \rho S V^2 \tag{3.38}$$

where C_l is the lift coefficient, C_d is the drag coefficient, and S is the normalized reference area. The parameters used in the simulations are presented in the Table 4. Although the distance unit for the data shown in Table 3.4 is the foot, the unit of distance used in the simulations is the mile, and the value of ε accounts for this normalization.

Table 3.4: Simulation parameters for Example 3-3

Variable	Value
Density (ρ_0)	$2.7*10^{-3} slug / ft^3$
Atmospheric scale height (k)	$4.2*10^{-5} / ft$
Gravitational constant (μ)	$1.4077*10^{16} ft^3 / s^2$
Lift coefficient (C_l)	0.35
Drag coefficient (C_d)	1.3
Reference area (S)	$0.3752 ft^2 / slug$
Scaling factor (ε)	$1.0538*10^{-6}$

The boundary conditions at initial and final points are given in Table 3.5.

Table 3.5: Boundary conditions for Example 3-3

Variables	Initial values	Final values
Time (t)	0(s)	390(s)
Altitude (r)	400,000(ft)	-
Longitude (θ)	0(rad)	0.330(rad)
Latitude (ϕ)	0(rad)	-0.025(rad)
Velocity (V)	36,000(ft/s)	2,640(ft/s)
Flight path angle (γ)	-6.5(deg)	-
Heading angle (φ)	0(deg)	-

For this problem, the number of integration data points selected is $n = 500$. The nominal trajectory is obtained by integrating the equations of motion with a constant bank angle $\beta = 0.8$. The costates are also selected to be constants over the entire trajectory: $\bar{\lambda} = [0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1]^T$. This problem is challenging because H_{uu} can become zero at isolated instants of time and hence, a perturbation factor of 1 is added to H_{uu}, which is subsequently reduced to $1e-7$. The convergence criterion is $\left\| x(T) - x_f \right\| < 1e-8$. For the control and costate guesses selected, the standard SBS gain back propagation is carried up to the switch point $t_1 = 0.1T$ initially, because of the conditioning of the gain matrices. After about 30 iterations, the switch time is moved to $t_1 = 0.9T$. The simulation results are shown in Table 3.6.

Table 3.6: Simulation results for Example 3-3

Variables	Initial values	Desired values	Final values
Altitude (r)	400,000(ft)	-	105,800(ft)
Longitude (θ)	0(rad)	0.33(rad)	0.330(rad)
Latitude (ϕ)	0(rad)	-0.025(rad)	-0.025(rad)
Velocity (V)	36,000(ft/s)	2,640(ft/s)	2,640(ft/s)
Flight path angle (γ)	-6.5(deg)	-	-13.33(deg)
Heading angle (φ)	0(deg)	-	-20.89(deg)
Cost (J)	-	-	13.33

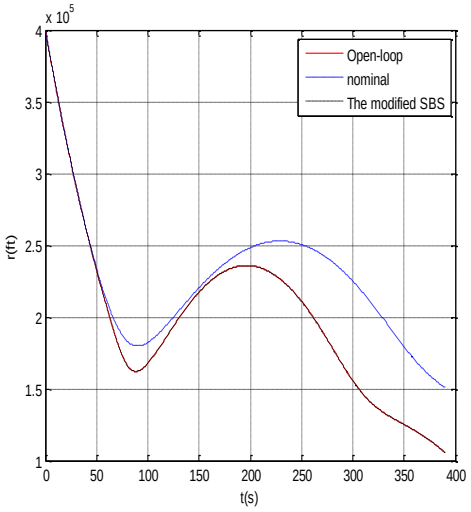


Figure 3.7: Altitude history for Example3-3

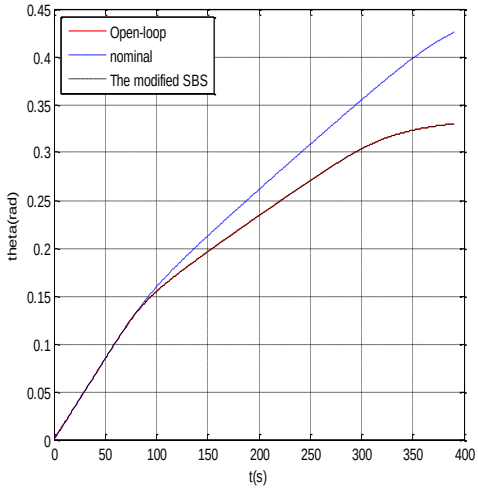


Figure 3.8: Longitude history for Example3-3

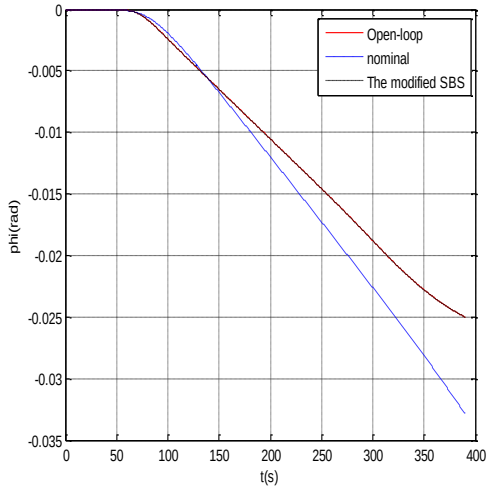


Figure 3.9: Latitude history for Example3-3

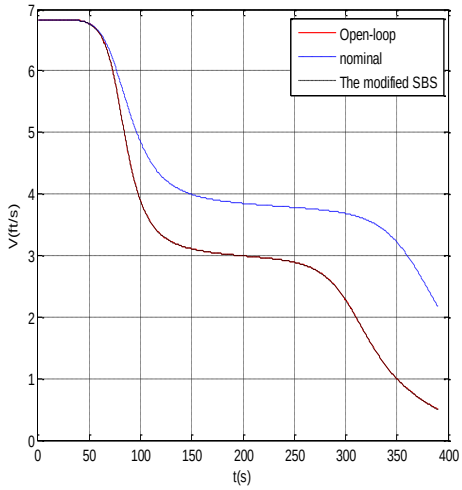


Figure 3.10: Velocity history for Example3-3

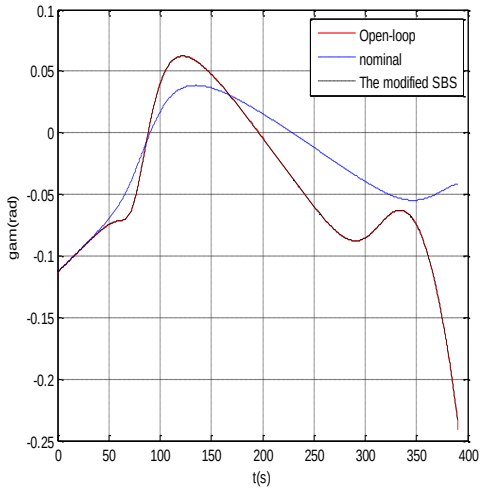


Figure 3.11: Flight path angle history for Example3-3

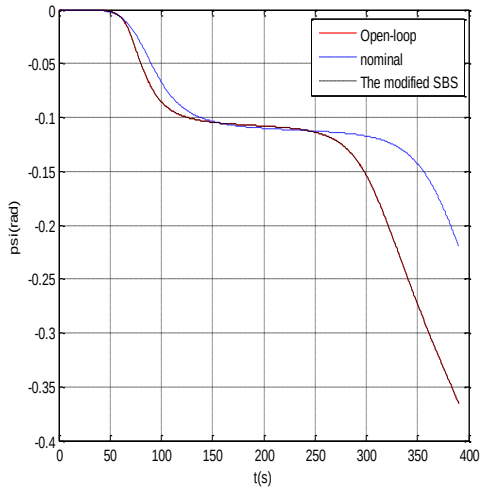


Figure 3.12: Heading angle history for Example3-3

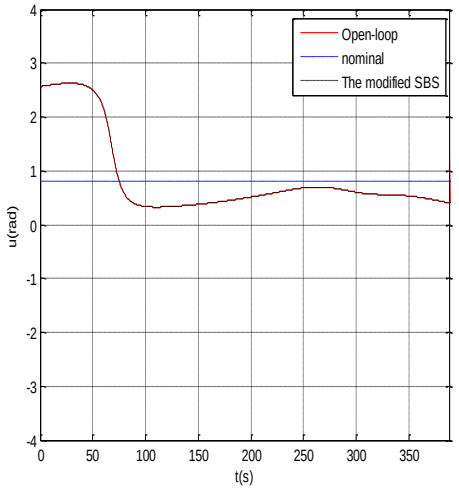


Figure 3.13: The input history for Example 3-3

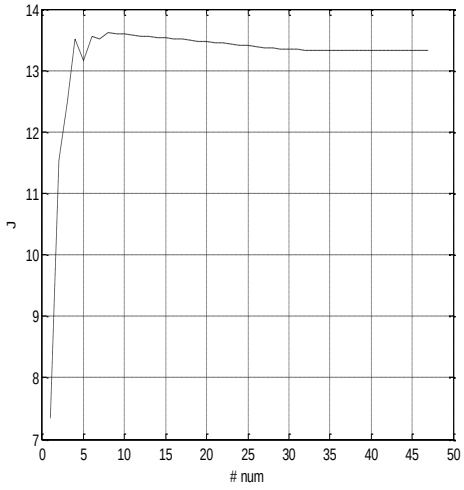


Figure 3.14: Cost convergence vs. iterations for Example3-3

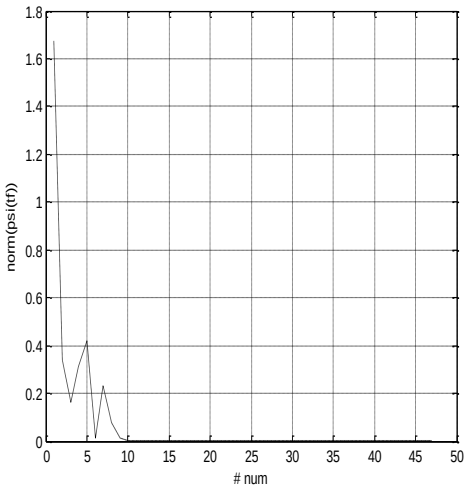


Figure 3.15: The terminal constraint violation $\left\| x(T) - x_f \right\|$ for Example 3-3

Figures 3.7-3.14 show three sets of data for the states, control, and the objective function. These correspond to: 1) the nominal trajectory, 2) results of using a shooting method (Matlab-fsolve), and 3) the modified SBS method with the Magnus integrator

(MSBS-MI). The initial costates used for the shooting method were obtained from MSBS-MI. The results from the two methods match very well and are indistinguishable to the scales of the figures. Figure 3.14 shows the convergence history for the performance index and Fig. 3.15 shows that for the constraint satisfaction error. Both of the indicators converge rapidly during the initial iterations. MSBS-MI obtains the least boundary condition error.

Table 3.7: Boundary satisfaction accuracy for Example 3-3

Method	Integrator	Iteration #	n	$x(T)-x_f$	Simulation Time (normalized)
ν is evaluated at t_0	RK-5th	200	1,300	0.01	1
	MI	47	1,300	4.895e-10	0.68
	MI	47	500	2.712e-9	0.25
The modified SBS method	RK-5th	200	1,300	4.985e-5	1.26
	MI	49	1,300	1.049e-11	0.75
	MI	47	500	2.744e-11	0.26

Table 3.7 shows that the computation burden of the Magnus integrator can be traded for larger step sizes. Either of the SBS methods used with the RK-5th integrator requires nearly 200 iterations to find a converged solution. However, MSBS-MI finds a high accuracy solution within 50 iterations. Note that the computation times can be significantly lower for more accurate initial guesses.

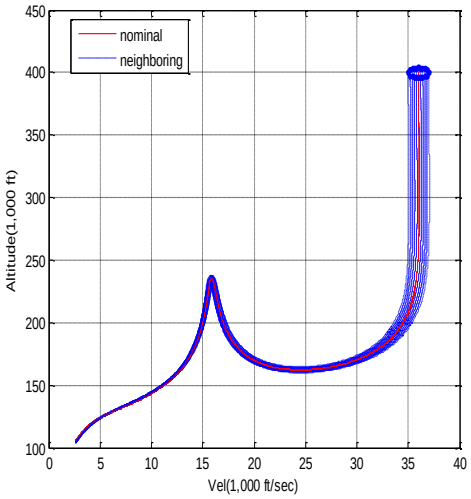


Figure 3.16: Neighboring reentry trajectories for Example 3-3 (MSBS-MI)

One advantage of the SBS method is that it produces the feedback gains required for perturbation guidance. The gains computed for the nominal trajectory can be used for computing control solutions for trajectories with perturbed initial conditions. Figure 3.16 shows the sensitivity when small perturbations are applied to the initial altitude and velocity, centered on their nominal values. The red line indicates the nominal trajectory obtained by MSBS-MI. The perturbations are $\pm 4,000\, ft$ in the initial altitude and $\pm 1,000\, ft/\sec$ in the initial velocity. For each method, the perturbed trajectories are propagated using the same integrator as used for computing the nominal trajectory and the gains. The gain values stored at evenly spaced time points are held constant between the nodes, during the integrations of the perturbed trajectories.

Table 3.8: The average error in the final velocity for Example 3-3

Method	Integrator	n	The average error in the final velocity (ft/sec)
ν is evaluated at t_0	RK-5th	1,300	$84.14\ ft/\sec$
	MI	1,300	$28.29\ ft/\sec$
	MI	500	$27.93\ ft/\sec$
The modified SBS method	RK-5th	1,300	$13.38\ ft/\sec$
	MI	1,300	$0.70\ ft/\sec$
	MI	500	$0.76\ ft/\sec$

The data of Table 3.8 indicate that MSBS-MI provides accurate satisfaction of the terminal boundary conditions for the class of perturbations considered, compared to the unmodified SBS method as well as the MSBS-RK-5th methods. An increase of the step size or alternatively, a reduction in the number of data points from 1300 to 500 (for a fixed final time) shows only a slight increase in the average terminal error in the final velocity.

4. DISCRETE TIME FORMULATION OF THE SBS METHOD

Coupling of the discretization scheme to the derivation the necessary conditions has distinct advantages, reduction in the terminal boundary error being the foremost. The SBS method can be applied in two ways, leading to the same results. The first application is to linearize a continuous-time system and then convert it to a discretized system. The second is to apply the discretization and then linearize the discrete time necessary conditions. The former approach is followed in this section.

4.1 The discretized SBS method

To begin with, the linearized, continuous-time necessary conditions for the control constraint problem are given by Eq. (3.8) and the associated boundary conditions. Equation (3.8) is reproduced in this section

$$\begin{bmatrix} \dot{x} \\ \dot{\lambda} \end{bmatrix} = \begin{bmatrix} \bar{A} & -\bar{B} \\ -\bar{Q} & -\bar{A}^T \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} + \begin{bmatrix} d_1 \\ -d_2 \end{bmatrix} \tag{3.8}$$

Equation (3.8) can be expressed in the simple form

$$\dot{X} = FX + D \tag{4.1}$$

where X indicates $X = [x\ \lambda]'$. The respective entries of Eq. (3.8) are modified from

those in Eqs. (3.3-3.7) because of the control constraint.

$$\bar{A} = H_{\lambda x} - H_{\lambda u}\pi\{H_{uu}^{-1}\}H_{ux} \tag{4.2}$$

$$\bar{B} = H_{\lambda u}\pi\{H_{uu}^{-1}\}H_{u\lambda} \tag{4.3}$$

$$\bar{Q} = H_{xx} - H_{xu}\pi\{H_{uu}^{-1}\}H_{ux} \tag{4.4}$$

$$d_1 = \alpha + H_{\lambda u}(P_c u_m - \pi\{H_{uu}^{-1}\}\gamma) \tag{4.5}$$

$$d_2 = \beta + H_{xu}(P_c u_m - \pi\{H_{uu}^{-1}\}\gamma) \tag{4.6}$$

With all the variables as defined previously, these equations can be expressed in a matrix form as in Eq. (4.1), which can be solved by any discretization of choice. In this section, six discretization schemes are used. The selected discretization schemes are 1) Euler forward method, 2) Euler backward method, 3) The midpoint rule, 4) Trapezoidal rule, 5) Simpson's rule, and 6) The discretization by using RK 4th [34]. Among these methods, the first five schemes are presented in the following and the algorithm of discretization by using the RK 4th method is described in Appendix B.

The left hand side of Eq. (4.1) commonly is discretized as follows:

$$\dot{X} = \frac{X_{k+1} - X_k}{\Delta} \tag{4.7}$$

where Δ is a time step ($\Delta = t_{k+1} - t_k$). The right hand side of Eq. (4.1) is discretized by

selecting one of the following schemes

Euler forward method:

$$x = x_k, \quad \lambda = \lambda_{k+1} \tag{4.8}$$

Euler backward method:

$$x = x_{k+1}, \quad \lambda = \lambda_k \tag{4.9}$$

The implicit midpoint rule:

$$x = \frac{x_k + x_{k+1}}{2}, \quad \lambda = \frac{\lambda_k + \lambda_{k+1}}{2} \tag{4.10}$$

Trapezoidal Rule:

$$FX + D = \frac{(F_k X_k + D_k) + (F_{k+1} X_{k+1} + D_{k+1})}{2} \tag{4.11}$$

Simpson's Rule:

$$Simpson\ Rule = \frac{2}{3}(Midpoint\ averaged\ rule) + \frac{1}{3}(Trapezoidal\ rule) \tag{4.12}$$

The second-order implicit midpoint rule of Eq. (4.10) provides many advantages. It is symmetric and is also a symplectic scheme, preserving constants of integration to second order. However, since it is an implicit scheme, it is applied in an explicit form in

the SBS approach by introducing an averaged input $\bar{u}$. For example, the system

$$\dot{x} = u \tag{4.13}$$

can be discretized by using the midpoint rule as follows:

$$\frac{x_{k+1} - x_k}{\Delta} = \frac{u_{k+1} + u_k}{2} \tag{4.14}$$

Equation (4.14) is not causal, a requirement for the use of the SBS method. However, Eq. (4.14) can be made causal by defining an average control $\bar{u}_k$ such that

$$\bar{u}_k = \frac{u_{k+1} + u_k}{2} \tag{4.15}$$

The SBS method is fully developed for use with the midpoint discretization scheme in the following. Equation (3.8) can be converted into a linear discrete-time system by substituting Eq. (4.10) into Eq. (3.8) as follows:

$$\begin{bmatrix} 2I - \Delta\bar{A} & \Delta\bar{B} \\ -\Delta\bar{Q} & 2I - \Delta\bar{A}^T \end{bmatrix} \begin{bmatrix} x_{k+1} \\ \lambda_k \end{bmatrix} = \begin{bmatrix} 2I + \Delta\bar{A} & -\Delta\bar{B} \\ \Delta\bar{Q} & 2I + \Delta\bar{A}^T \end{bmatrix} \begin{bmatrix} x_k \\ \lambda_{k+1} \end{bmatrix} + \begin{bmatrix} 2\Delta d_1 \\ 2\Delta d_2 \end{bmatrix} \tag{4.16}$$

Note that for the discrete-time approximation by the midpoint rule, λ_{k+1} appears on the

right hand side of Eq. (4.16). The state and costate equations in the discrete time system can be easily obtained by using the inverse transformation in Eq. (4.16).

$$\begin{bmatrix} x_{k+1} \\ \lambda_k \end{bmatrix} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{11}^{\ T} \end{bmatrix} \begin{bmatrix} x_k \\ \lambda_{k+1} \end{bmatrix} + \begin{bmatrix} D_{11} \\ D_{21} \end{bmatrix} \tag{4.17}$$

where

$$F = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{11}^{\ T} \end{bmatrix} = \begin{bmatrix} 2I - \Delta\bar{A} & \Delta\bar{B} \\ -\Delta\bar{Q} & 2I - \Delta\bar{A}^T \end{bmatrix}^{-1} \begin{bmatrix} 2I + \Delta\bar{A} & -\Delta\bar{B} \\ \Delta\bar{Q} & 2I + \Delta\bar{A}^T \end{bmatrix} \tag{4.18}$$

$$D = \begin{bmatrix} D_{11} \\ D_{21} \end{bmatrix} = \begin{bmatrix} 2I - \Delta\bar{A} & \Delta\bar{B} \\ -\Delta\bar{Q} & 2I - \Delta\bar{A}^T \end{bmatrix}^{-1} \begin{bmatrix} 2\Delta d_1 \\ 2\Delta d_2 \end{bmatrix} \tag{4.19}$$

Equation (4.17) can be solved in partitioned form by substituting the expression λ_k as given by Eq. (3.17). The differential gain equations of the discretized SBS method are obtained as follows:

$$S_k = F_{21} + F_{11}^{\ T} S_{k+1} K F_{11}, \qquad S_n = [\phi_{\bar{x}\bar{x}} + (\bar{v}^T \psi_{\bar{x}})_{\bar{x}}]\big|_{k=n} \tag{4.20}$$

$$P_k = F_{11}^{\ T} (I + S_{k+1} K F_{12}) P_{k+1}, \qquad P_n = \psi_{\bar{x}_n}^T \tag{4.21}$$

$$V_k = F_{11}^{\ T} (I + S_{k+1} K F_{12}) V_{k+1} + D_{21} + F_{11}^{\ T} S_{k+1} K D_{11}, V_n = [\phi_{\bar{x}} - (\phi_{\bar{x}\bar{x}} + (\bar{v}^T \psi_{\bar{x}})_{\bar{x}})\bar{x}]\big|_{k=n} \tag{4.22}$$

$$W_k = W_{k+1} + P_{k+1}^{\ T} K F_{12} P_{k+1}, \qquad W_n = 0 \tag{4.23}$$

$$N_k = N_{k+1} + P_{k+1}^{\ T} K (F_{12} V_{k+1} + D_{11}), \qquad N_n = (\psi(\bar{x}) - \psi_{\bar{x}} \bar{x})\big|_{k=n} \tag{4.24}$$

where

$$K = (I - F_{12}S_{k+1})^{-1} \tag{4.25}$$

The discretized input variable can be obtained by using the stationary condition. The stationarity condition in the discrete-time system is represented as:

$$H_{ux}(\frac{x_k + x_{k+1}}{2}) + H_{uu}\bar{u}_k + H_{u\lambda}(\frac{\lambda_k + \lambda_{k+1}}{2}) + \gamma_k = 0 \tag{4.26}$$

The discretized control input can be obtained by substituting the Eq. (4.17) and Eq. (3.17) into Eq. (4.26).

$$\bar{u}_k = -\frac{1}{2}H_{uu}{}^{-1}[(H_{ux} + K_u F_{11})x_k + (H_{u\lambda} + K_u F_{12})(P_{k+1}\nu + V_{k+1}) + H_{u\lambda}\lambda_k + K_u D_{11} + 2\gamma] \tag{4.27}$$

where

$$K_u = (H_{ux} + H_{u\lambda}S_{k+1})K \tag{4.28}$$

4.2 The discretized SBS method for free final time problems

The SBS method can also be extended to solve free final time problems. A free final time problem can be converted into a fixed final time problem by defining a nondimensional time [35] as follows:

$$\tau = \frac{t}{t_f} \quad (0 \le \tau \le 1) \tag{4.29}$$

The final time can be considered as a trivial state variable. The linearized necessary conditions in the continuous-time domain can be expressed as follows:

$$x' = H_{\lambda x}x + H_{\lambda u}u + H_{\lambda b}b + \alpha \tag{4.30}$$

$$\lambda' = -(H_{xx}x + H_{xu}u + H_{x\lambda}\lambda + H_{xb}b + \beta) \tag{4.31}$$

$$\mu' = -(H_{bx}x + H_{bu}u + H_{b\lambda}\lambda + H_{bb}b + \gamma) \tag{4.32}$$

$$u = P_c u_m - \pi\{H_{uu}^{-1}\}(H_{ux}x + H_{u\lambda}\lambda + H_{ub}b + \delta) \tag{4.33}$$

where μ is the costate corresponding to t_f, designated by the new variable b, and $'$ indicates the derivative with respect to τ

$$\alpha = H_{\lambda} - H_{\lambda x}\bar{x} - H_{\lambda u}\bar{u} - H_{\lambda b}\bar{b} \tag{4.34}$$

$$\beta = H_{x} - H_{xx}\bar{x} - H_{xu}\bar{u} - H_{x\lambda}\bar{\lambda} - H_{xb}\bar{b} \tag{4.35}$$

$$\gamma = H_b - H_{bx}\bar{x} - H_{bu}\bar{u} - H_{b\lambda}\bar{\lambda} - H_{bb}\bar{b} \tag{4.36}$$

$$\delta = H_u - H_{ux}\bar{x} - H_{uu}\bar{u} - H_{u\lambda}\bar{\lambda} - H_{ub}\bar{b} \tag{4.37}$$

where $\bar{b}$ indicates the nominal value of b, the final time. Substitution of Eq. (4.33) into Eqs. (4.30-4.32) results in the state and costate equations:

$$x' = \bar{A}x - H_{\lambda u}\pi\{H_{uu}^{-1}\}H_{u\lambda}\lambda + \bar{B}b + d_1 \tag{4.38}$$

$$\lambda' = -(\bar{Q}x + \bar{A}^T\lambda + \bar{C}b + d_2) \tag{4.39}$$

$$\mu' = -(\bar{C}^T x + \bar{B}^T\lambda + \bar{D}b + d_3) \tag{4.40}$$

where

$$\bar{A} = H_{\lambda x} - H_{\lambda u}\pi\{H_{uu}^{-1}\}H_{ux} \tag{4.41}$$

$$\bar{B} = H_{\lambda b} - H_{\lambda u}\pi\{H_{uu}^{-1}\}H_{ub} \tag{4.42}$$

$$\bar{C} = H_{xb} - H_{xu}\pi\{H_{uu}^{-1}\}H_{ub} \tag{4.43}$$

$$\bar{D} = H_{bb} - H_{bu}\pi\{H_{uu}^{-1}\}H_{ub} \tag{4.44}$$

$$\bar{Q} = H_{xx} - H_{xu}\pi\{H_{uu}^{-1}\}H_{ux} \tag{4.45}$$

$$d_1 = \alpha + H_{\lambda u}(P_c u_m - \pi\{H_{uu}^{-1}\}\delta) \tag{4.46}$$

$$d_2 = \beta + H_{xu}(P_c u_m - \pi\{H_{uu}^{-1}\}\delta) \tag{4.47}$$

$$d_3 = \gamma + H_{bu}(P_c u_m - \pi\{H_{uu}^{-1}\}\delta) \tag{4.48}$$

Equations (4.38-4.40) are discretized by the explicit midpoint rule, Eq. (4.10) and Eq. (4.15), resulting in the matrix form

$$\begin{bmatrix} 2I-\Delta\bar{A} & \Delta H_{\lambda u}\pi\{H_{uu}^{-1}\}H_{u\lambda} & 0 \\ -\Delta\bar{Q} & 2I-\Delta\bar{A}^T & 0 \\ -\Delta\bar{C}^T & -\Delta\bar{B}^T & 2 \end{bmatrix}\begin{bmatrix} x_{k+1} \\ \lambda_k \\ \mu_k \end{bmatrix}=\begin{bmatrix} 2I+\Delta\bar{A} & -\Delta H_{\lambda u}\pi\{H_{uu}^{-1}\}H_{u\lambda} & 0 \\ \Delta\bar{Q} & 2I+\Delta\bar{A}^T & 0 \\ \Delta\bar{C}^T & \Delta\bar{B}^T & 2 \end{bmatrix}\begin{bmatrix} x_k \\ \lambda_{k+1} \\ \mu_{k+1} \end{bmatrix}+\begin{bmatrix} 2\Delta\bar{B} \\ 2\Delta\bar{C} \\ 2\Delta\bar{D} \end{bmatrix}b+\begin{bmatrix} 2\Delta d_1 \\ 2\Delta d_2 \\ 2\Delta d_3 \end{bmatrix} \quad (4.49)$$

After performing the matrix inverse, Eq. (4.49) is transformed into

$$\begin{bmatrix} x_{k+1} \\ \lambda_k \\ \mu_k \end{bmatrix}=\begin{bmatrix} F_{11} & F_{12} & 0 \\ F_{21} & F_{11}{}^T & 0 \\ F_{31} & F_{32} & 1 \end{bmatrix}\begin{bmatrix} x_k \\ \lambda_{k+1} \\ \mu_{k+1} \end{bmatrix}+\begin{bmatrix} C_{11} \\ C_{21} \\ C_{31} \end{bmatrix}b+\begin{bmatrix} D_{11} \\ D_{21} \\ D_{31} \end{bmatrix} \quad (4.50)$$

where

$$F=\begin{bmatrix} F_{11} & F_{12} & 0 \\ F_{21} & F_{11}{}^T & 0 \\ F_{31} & F_{32} & 1 \end{bmatrix}=\begin{bmatrix} 2I-\Delta\bar{A} & \Delta H_{\lambda u}\pi\{H_{uu}^{-1}\}H_{u\lambda} & 0 \\ -\Delta\bar{Q} & 2I-\Delta\bar{A}^T & 0 \\ -\Delta\bar{C}^T & -\Delta\bar{B}^T & 2 \end{bmatrix}^{-1}\begin{bmatrix} 2I+\Delta\bar{A} & -\Delta H_{\lambda u}\pi\{H_{uu}^{-1}\}H_{u\lambda} & 0 \\ \Delta\bar{Q} & 2I+\Delta\bar{A}^T & 0 \\ \Delta\bar{C}^T & \Delta\bar{B}^T & 2 \end{bmatrix} \quad (4.51)$$

$$C=\begin{bmatrix} C_{11} \\ C_{21} \\ C_{31} \end{bmatrix}=\begin{bmatrix} 2I-\Delta\bar{A} & \Delta H_{\lambda u}\pi\{H_{uu}^{-1}\}H_{u\lambda} & 0 \\ -\Delta\bar{Q} & 2I-\Delta\bar{A}^T & 0 \\ -\Delta\bar{C}^T & -\Delta\bar{B}^T & 2 \end{bmatrix}^{-1}\begin{bmatrix} 2\Delta\bar{B} \\ 2\Delta\bar{C} \\ 2\Delta\bar{D} \end{bmatrix} \quad (4.52)$$

$$D=\begin{bmatrix} D_{11} \\ D_{21} \\ D_{31} \end{bmatrix}=\begin{bmatrix} 2I-\Delta\bar{A} & \Delta H_{\lambda u}\pi\{H_{uu}^{-1}\}H_{u\lambda} & 0 \\ -\Delta\bar{Q} & 2I-\Delta\bar{A}^T & 0 \\ -\Delta\bar{C}^T & -\Delta\bar{B}^T & 2 \end{bmatrix}^{-1}\begin{bmatrix} 2\Delta d_1 \\ 2\Delta d_2 \\ 2\Delta d_3 \end{bmatrix} \quad (4.53)$$

The gain differential equations are obtained by assuming λ_k as in in Eq. (3.17):

$$\lambda_k = S_k x_k + P_k \nu + M_k b + V_k \tag{4.54}$$

Boundary values of gain matrices can be determined from Eqs. (4.55-4.57).

$$\lambda_n = \phi_{xx} x_n + \psi_{x_n}{}^T \nu + \Omega_{x_n}{}^T b = S_n x_n + P_n \nu + M_n b + V_n \tag{4.55}$$

$$\psi_n = \psi_{x_n} x_n + \psi_b b = P_n^T x_n + W_n \nu + G_n b + N_n \tag{4.56}$$

$$\Omega_n = M_n{}^T x_n + G_n{}^T \nu + K_n b + Z_n \tag{4.57}$$

where

$$\Omega_n = \phi_b + \psi_b{}^T \nu + H_n \tag{4.58}$$

Finally, the equations for propagating the gain matrices are obtained by substituting Eq. (4.54) into Eq. (4.50) as follows:

$$S_k = F_{21} + K_h F_{11}, \qquad S_n = [\phi_{\bar{x}\bar{x}} + (\bar{\nu}^T \psi_{\bar{x}})_{\bar{x}} + (\bar{b}^T \Omega_{\bar{x}})_{\bar{x}}]\Big|_{k=n} \tag{4.59}$$

$$P_k = F_{11}{}^T K_g P_{k+1}, \qquad P_n = \psi_{\bar{x}_n}^T \tag{4.60}$$

$$M_k = F_{11}{}^T K_g M_{k+1} + K_h C_{11} + C_{21} \qquad , \; M_n = \Omega_{\bar{x}_n}^T \tag{4.61}$$

$$V_k = F_{11}{}^T K_g V_{k+1} + K_h D_{11} + D_{21}, \;\; V_n = [\phi_{\bar{x}} - (\phi_{\bar{x}\bar{x}} + (\bar{\nu}^T \psi_{\bar{x}})_{\bar{x}} + (\bar{b}^T \Omega_{\bar{x}})_{\bar{x}})\bar{x}]\Big|_{k=n} \tag{4.62}$$

$$W_k = W_{k+1} + P_{k+1}{}^T K F_{12} P_{k+1}, \qquad W_n = 0 \quad (4.63)$$

$$G_k = G_{k+1} + P_{k+1}{}^T K (F_{12} M_{k+1} + C_{11}), \qquad G_n = \psi_b \quad (4.64)$$

$$N_k = N_{k+1} + P_{k+1}{}^T K (F_{12} V_{k+1} + D_{11}), \qquad N_n = (\psi - \psi_{\bar{x}_n} \bar{x}_n - \psi_b b)\Big|_{k=n} \quad (4.65)$$

$$K_k = K_i M_{k+1} + K_{k+1} + C_{31} + K_m C_{11}, \qquad K_n = \Omega_b \quad (4.66)$$

$$Z_k = K_i V_{k+1} + Z_{k+1} + D_{31} + K_m D_{11}, \qquad Z_n = (\Omega - \Omega_{\bar{x}_n} \bar{x}_n - \Omega_b b)\Big|_{k=n} \quad (4.67)$$

where

$$K_g = I + S_{k+1} K F_{12} \quad (4.68)$$

$$K_h = F_{11}{}^T S_{k+1} K \quad (4.69)$$

$$K_i = F_{32} + K_m F_{12} \quad (4.70)$$

$$K_m = (F_{32} S_{k+1} + M_{k+1}{}^T) K \quad (4.71)$$

The stationarity condition for this problem (midpoint rule equivalent to Eq. 4.33) is

$$H_{ux}\left(\frac{x_k + x_{k+1}}{2}\right) + H_{uu} \bar{u}_k + H_{u\lambda}\left(\frac{\lambda_k + \lambda_{k+1}}{2}\right) + H_{ub} b + \delta_k = 0 \quad (4.72)$$

The averaged control variable can be obtained by substituting in Eq. (4.72), the first of Eqs. (4.50) for x_{k+1}, and the equivalent of Eq. (4.54) for λ_{k+1}.

$$\begin{aligned}\bar{u}_k = -\frac{1}{2} H_{uu}^{-1}[&(H_{ux} + K_j F_{11}) x_k + (K_j F_{12} P_{k+1} + H_{u\lambda} P_{k+1}) \nu \\ &+ [2H_{ub} + K_j (F_{12} M_{k+1} + C_{11}) + H_{u\lambda} M_{k+1}] b + 2\delta \\ &+ K_j (F_{12} v_{k+1} + D_{11}) + H_{u\lambda} v_{k+1} + H_{u\lambda} \lambda_k]\end{aligned} \tag{4.73}$$

where

$$K_j = (H_{ux} + H_{u\lambda} S_{k+1}) K \tag{4.74}$$

4.3 The discretized SBS method with impulsive control

For an autonomous linear system given by

$$\dot{x} = Ax + Bu \tag{4.75}$$

with $u(t)$ assumed to be piecewise constant, with discontinuities only at times $k\Delta$ (a zero-order hold), the discrete-time system can be obtained as follows:

$$x_{k+1} = e^{A\Delta} x_k + \int_{k\Delta}^{(k+1)\Delta} e^{A((k+1)\Delta - \tau)} B d\tau u_k \tag{4.76}$$

On the other hand, if the input $u(t)$ is a series of impulses acting at times $t_k = k\Delta$, the discrete-time system is

$$x_{k+1} = e^{A\Delta} x_k + e^{A\Delta} B \upsilon_k \tag{4.77}$$

where

$$\upsilon_k = \int_{k\Delta}^{(k+1)\Delta} u(\tau)d\tau \tag{4.78}$$

Since the continuous-time, linearized system obtained at each iteration of the SBS method is non-autonomous, other forms of discretization than those discussed previously have to be employed. The explicit midpoint rule can be applied to a linear system of the form

$$\dot{x} = Ax + Bu + \alpha \tag{4.79}$$

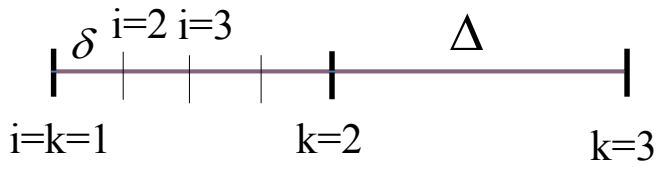


Figure 4.1: The time index and time step of impulsive control

with the impulse application scheme shown in Fig. 4.1. The impulse application times are not necessarily synchronized with the discretization steps. In Figure 4.1, the indices i and k, respectively indicate the data points and the impulse application times; Δ is the time step between two successive impulses and several smaller propagation steps of size δ are used in-between. Between two successive propagation steps, the discrete-time system matrices are

$$A_i = (2I - A\delta)^{-1}(2I + A\delta) \tag{4.80}$$

$$B_i = (2I - A\delta)^{-1}2B \tag{4.81}$$

$$D_i = (2I - A\delta)^{-1}2\delta\alpha \tag{4.82}$$

The discretized equation for stepping between consecutive impulse application times is

$$x_{k+1} = \bar{A}_k x_k + \bar{B}_k \upsilon_k + \bar{d}_k \tag{4.83}$$

where

$$\bar{A}_k = A_n \cdot A_{n-1} \cdots A_1 \tag{4.84}$$

$$\bar{B}_k = A_n \cdot A_{n-1} \cdots A_2 B_k \tag{4.85}$$

$$\bar{d}_k = (A_n \cdot A_{n-1} \cdots A_2)D_1 + (A_n \cdot A_{n-1} \cdots A_3)D_2 + \cdots A_n D_{n-1} + D_n \tag{4.86}$$

and n is the number of data points between $[k, k+1]$. The optimal impulse vector can be obtained by using the stationarity condition:

$$\upsilon_k = -K_{imp}{}^{-1}[H_{ux}x_k + H_{u\lambda}(S_{k+1}\bar{A}_k x_k + P_{k+1}\nu + \upsilon_{k+1}) + \gamma_k + H_{u\lambda}S_{k+1}\alpha_k] \tag{4.87}$$

where

$$K_{imp} = H_{uu} + H_{u\lambda}S_{k+1}\bar{B}_k \tag{4.88}$$

4.4 The discretized SBS method with a waypoint scheme

Generally, highly nonlinear OCPs posed over long time intervals are difficult to solve due to numerical sensitivity issues. Introduction of waypoints ameliorates the situation by restricting the integration domain in each segment or stage. If the waypoint (x,t) corresponds to a point on the optimal trajectory and the original problem admits continuous costates [36], then they should not exhibit discontinuities at the segment boundaries. If however, the waypoints are arbitrarily prescribed by the initial guess, then the costate jumps can be used as error indicators to be corrected to produce the optimal waypoint locations. In this section, the main idea is developed for the case of two waypoints. Without loss of generality, this procedure can be easily extended to the problem with any number of waypoints.

The waypoint (x,t) locations chosen as initial guesses are refined iteratively to produce the converged trajectory. In the following, it is assumed that the waypoint time is fixed, but its spatial coordinates are free to be selected. The total cost is the sum of the cost values incurred on each segment.

$$J = J_1 + J_2 + J_3 \tag{4.89}$$

where J_i is the cost for the ith segment. The segment costs are represented as:

$$J_1 = \frac{1}{2} x_0{}^T S_0 x_0 + x_0{}^T P_0 \nu_1 + x_0{}^T V_0 + \frac{1}{2} \nu_1{}^T W_0 \nu_1 + N_0{}^T \nu_1 - x_1{}^T \nu_1 \tag{4.90}$$

$$J_2 = \frac{1}{2} x_1^T S_1 x_1 + x_1^T P_1 \nu_2 + x_1^T V_1 + \frac{1}{2} \nu_2^T W_1 \nu_2 + N_1^T \nu_2 - x_2^T \nu_2 \tag{4.91}$$

$$J_3 = \frac{1}{2} x_2^T S_2 x_2 + x_2^T P_2 \nu_3 + x_2^T V_2 + \frac{1}{2} \nu_3^T W_2 \nu_3 + N_2^T \nu_3 - x_f^T \nu_3 \tag{4.92}$$

where

$$\nu_1 = W_0^{-1}[x_1 - P_0^T x_0 - N_0] \tag{4.93}$$

$$\nu_2 = W_1^{-1}[x_2 - P_1^T x_1 - N_1] \tag{4.94}$$

$$\nu_3 = W_2^{-1}[x_f - P_2^T x_2 - N_2] \tag{4.95}$$

The condition that the costates have to be continuous at a waypoint can be expressed as follows [36]:

$$\frac{\partial J}{\partial x_1} = \frac{\partial J_1}{\partial x_1} + \frac{\partial J_2}{\partial x_1} = 0 \tag{4.96}$$

$$\frac{\partial J}{\partial x_2} = \frac{\partial J_2}{\partial x_2} + \frac{\partial J_3}{\partial x_2} = 0 \tag{4.97}$$

Developed fully, Eqs. (4.96-4.97) can be expressed in the matrix form

$$\begin{bmatrix} W_0^{-1} - S_1 + P_1 W_1^{-1} P_1^T & -P_1 W_1^{-1} \\ -W_1^{-1} P_1^T & W_1^{-1} - S_2 + P_2 W_2^{-1} P_2^T \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} W_0^{-1}(P_0^T x_0 + N_0) - P_1 W_1^{-1} N_1 + V_1 \\ W_1^{-1} N_1 + P_2 W_2^{-1}(x_f - N_2) + V_2 \end{bmatrix} \tag{4.98}$$

Hence, the optimal waypoints also can be obtained by using the inverse transformation

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{wp} = \begin{bmatrix} W_0^{-1} - S_1 + P_1 W_1^{-1} P_1^T & -P_1 W_1^{-1} \\ -W_1^{-1} P_1^T & W_1^{-1} - S_2 + P_2 W_2^{-1} P_2^T \end{bmatrix}^{-1} \begin{bmatrix} W_0^{-1}(P_0^T x_0 + N_0) - P_1 W_1^{-1} N_1 + V_1 \\ W_1^{-1} N_1 + P_2 W_2^{-1}(x_f - N_2) + V_2 \end{bmatrix} \tag{4.99}$$

4.5 Numerical examples

4.5.1 Example 4-1: A 2-D nonlinear problem with final state constraints

Example 2-2 is used in this section to benchmark the performances of the discretized methods considered in this section. The previous formulations, Example 2-2 and Example 3-1, required Hessian modifications. However, the discretized SBS method can easily find a converged solution even for a relatively large time step size, without requiring a modification of the Hessian. The discretization methods of Eqs. (4.8-4.12) and also the RK-4th method are compared. The simulation conditions are the same as in Example 2-2. The convergence tolerance for the total cost used to determine the number of iterations is $1e-3$.

Table 4.1: Simulation results for the Example 4-1

Method	n	Error x_1 (Max)	Error x_2 (Max)	Iteration #	J
Euler forward method	5	2.1036	3.2123	14	414.82
	10	0.4785	1.3280	5	22.53
	50	0.0497	0.0813	5	10.05
	100	0.0239	0.0342	5	9.40
Euler backward method	5	0.3534	0.1601	6	1.32
	10	0.2250	0.0464	5	3.86
	50	0.0465	0.0380	5	7.70
	100	0.0232	0.0240	5	8.24
RK 4^{th} order	5	X	X	X	X
	10	X	X	X	X
	50	0.0026	0.0080	5	8.84
	100	0.0007	0.0020	5	8.81
Trapezoidal rule	5	0.3341	0.9838	45	9.22
	10	0.0976	0.1555	8	11.09
	50	0.0027	0.0101	5	8.83
	100	0.0007	0.0026	5	8.81
Simpson's rule	5	0.2425	0.7903	7	8.63
	10	0.0722	0.2572	5	9.06
	50	0.0027	0.0107	5	8.81
	100	0.0007	0.0027	5	8.80
The explicit midpoint rule	5	0.2492	0.7835	5	8.08
	10	0.0648	0.2869	5	8.62
	50	0.0026	0.0111	5	8.79
	100	0.0006	0.0028	5	8.80

X indicates no converged solution

The results of the numerical experiments are summarized in Table 4.1. As expected, the first order Euler forward/backward methods have the maximum error for a given step size. Trapezoidal rule requires the largest number of iterations for n = 5. An interesting pattern is noticed for the cost resulting from the two methods. For the Euler forward method, the cost converges from above, with a decrease in the discretization step size and the opposite trend is observed for the backward Euler method. The error for the midpoint rule is quite insensitive to step size variation. On the other hand, the RK 4th order method cannot find a converged solution for large step sizes (n=5,10).

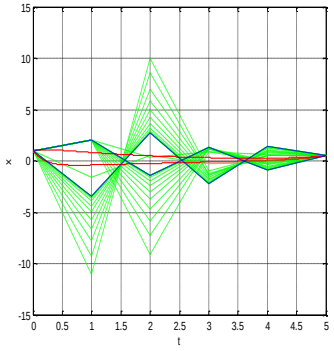


Figure 4.2: x_1 & x_2 Trajectory ($n = 5$, Euler forward method)

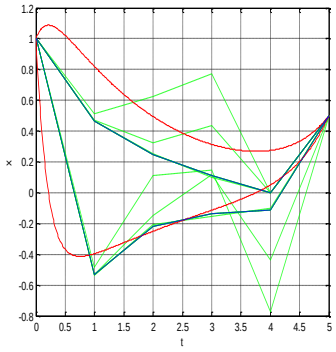


Figure 4.3: x_1 & x_2 Trajectory ($n = 5$, Euler backward method)

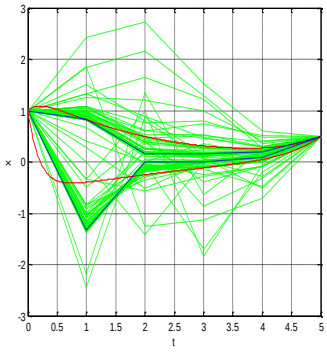


Figure 4.4: x_1 & x_2 Trajectory ($n = 5$, Trapezoidal rule)

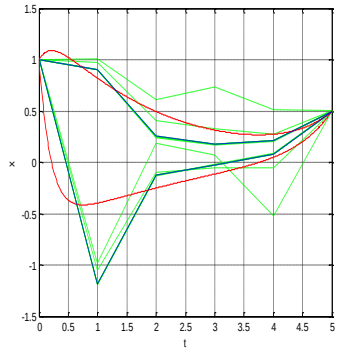


Figure 4.5: x_1 & x_2 Trajectory ($n = 5$, Simpson's rule)

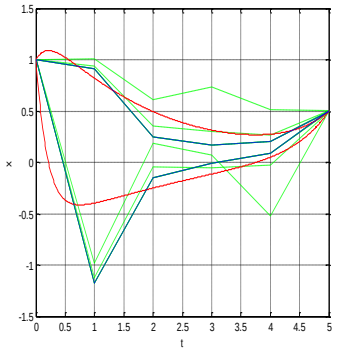


Figure 4.6: x_1 & x_2 Trajectory ($n = 5$, Midpoint rule)

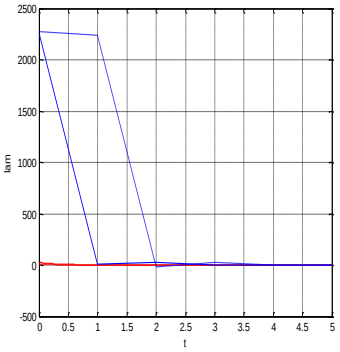


Figure 4.7: λ_1 & λ_2 Trajectory ($n = 5$, Euler forward method)

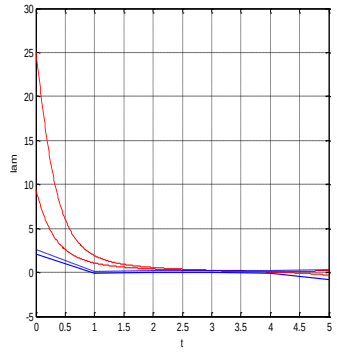


Figure 4.8: λ_1 & λ_2 Trajectory ($n = 5$, Euler backward method)

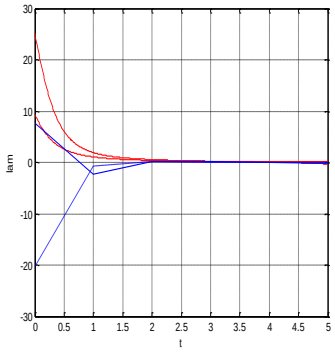


Figure 4.9: λ_1 & λ_2 Trajectory ($n = 5$, Trapezoidal rule)

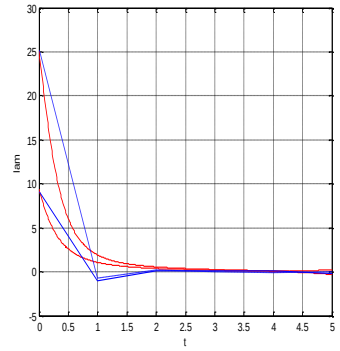


Figure 4.10: λ_1 & λ_2 Trajectory ($n = 5$, Simpson's rule)

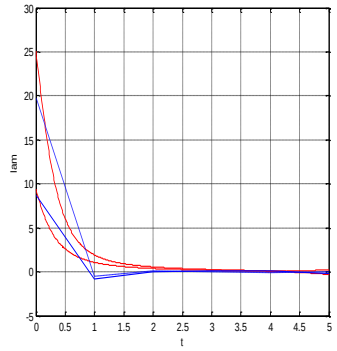
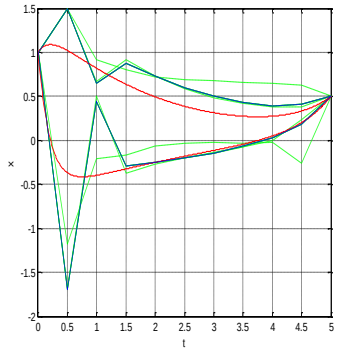
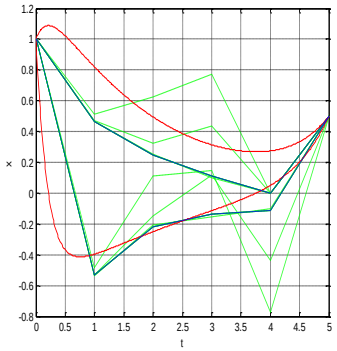


Figure 4.11: λ_1 & λ_2 Trajectory ($n = 5$, Midpoint rule)





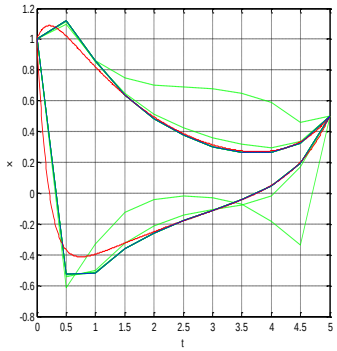
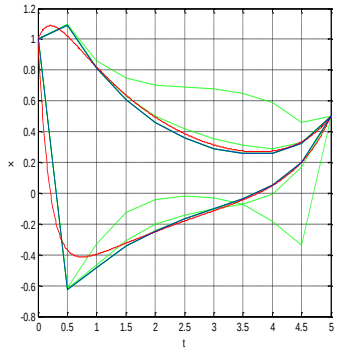


Figure 4.12: $x_1 \& x_2$ Trajectory ($n = 10$, Euler forward method)

Figure 4.13: $x_1 \& x_2$ Trajectory ($n = 10$, Euler backward method)

Figure 4.14: $x_1 \& x_2$ Trajectory ($n = 10$, Trapezoidal rule)



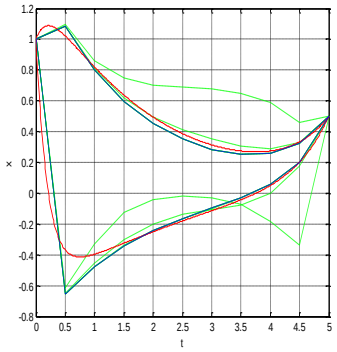
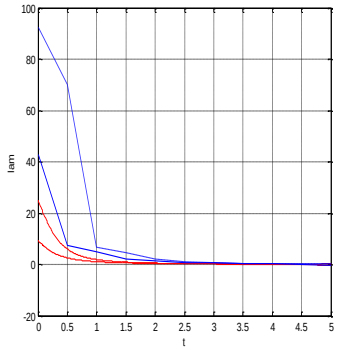
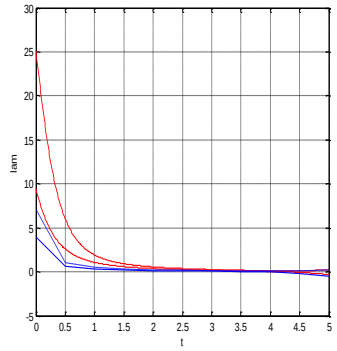


Figure 4.15: $x_1 \& x_2$ Trajectory ($n = 10$, Simpson's rule)

Figure 4.16: $x_1 \& x_2$ Trajectory ($n = 10$, Midpoint rule)





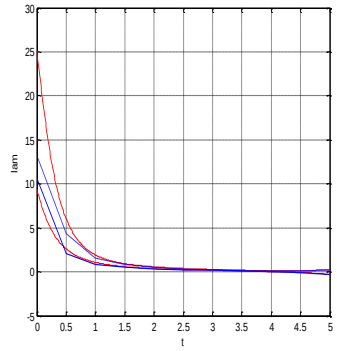


Figure 4.17: $\lambda_1 \& \lambda_2$ Trajectory ($n = 10$, Euler forward method)

Figure 4.18: $\lambda_1 \& \lambda_2$ Trajectory ($n = 10$, Euler backward method)

Figure 4.19: $\lambda_1 \& \lambda_2$ Trajectory ($n = 10$, Trapezoidal rule)

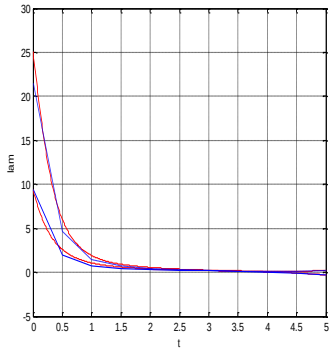


Figure 4.20: λ_1 & λ_2 Trajectory ($n = 10$, Simpson's rule)

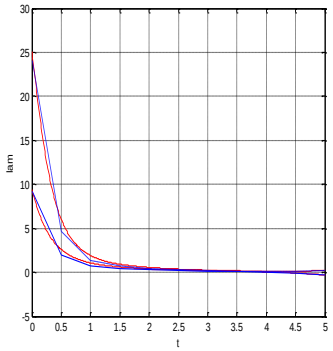


Figure 4.21: λ_1 & λ_2 Trajectory ($n = 10$, Midpoint rule)

Results for discretizations with n = 5 and n = 10 are shown in Figs. 4.2-4.11 and Figs, 4.12-4.21, respectively. In each of Figs. 4.2-4.6 and 4.12-4.16, the red lines indicates the open-loop trajectory variables, the blue lines indicate the corresponding converged solutions, and the green lines indicate the intermediate solutions obtained during the iteration process. Figures 4.7-4.11 and 4.17-4.21 show only the converged SBS and the corresponding open-loop solutions. The open-loop solutions result from a shooting method (Matlab, *fsolve*) with a small time step of 0.01. Figures 4.2-4.21 and the data of Table 4.1 show that the explicit midpoint rule is the best performer among all the methods considered. As seen from Fig. 4.21, the costate convergence for the explicit midpoint rule, at n=10, is remarkable.

A least squares fit of the state error of the form ch^q [37] is produced for the various methods considered by varying the step size between n=50 and n=100. The results are summarized for the exponent, q, for each state in Table 4.2.

Table 4.2: The results of $\varepsilon(x_i) = ch^{q(x_i)}$ between n=50 and n=100

Method	$q(x_1)$	$q(x_2)$	Average q
Euler forward method	1.057	1.249	1.153
Euler backward method	1.001	0.663	0.832
RK 4th order	1.893	2	1.947
Trapezoidal rule	1.947	1.957	1.952
Simpson's rule	1.947	1.987	1.967
The explicit midpoint rule	2.148	1.988	2.068

The data from Table 4.2 clearly show the expected behavior for all the methods except for the Simpson's rule and RK 4th method, for which the convergence rate is quadratic and not quartic. Overall, the results of Table 4.1 and 4.2 show that the explicit midpoint rule has performed better than the other methods on the example considered.

4.5.2 Example 4-2: A 2-D nonlinear problem with free final time

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_1 + x_1^5 + u \end{aligned} \tag{4.100}$$

$$J = \frac{1}{2}\int_0^{t_f} (x_2^2 + u^2) dt \tag{4.101}$$

Example 4-1 is considered again, but with a free final time. A nondimensional time variable τ is defined over the domain between 0 and 1, to convert the problem into one with a fixed final time. The transformed system and the performance index can be expressed in the τ domain by introducing a new parameter b as follows:

$$\begin{aligned} x_1' &= bx_2 \\ x_2' &= b(x_1 + x_1^5 + u) \end{aligned} \tag{4.102}$$

$$J = \frac{1}{2}\int_0^1 b(x_2^2 + u^2) d\tau \tag{4.103}$$

Figures 4.22-4.25 show that the simulation results of 2D nonlinear problem with free final time.

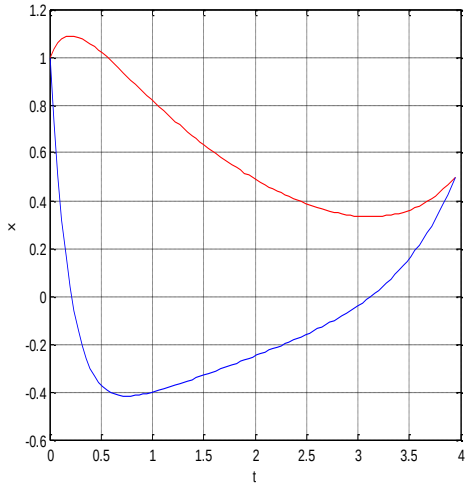


Figure 4.22: State history for Example4-2

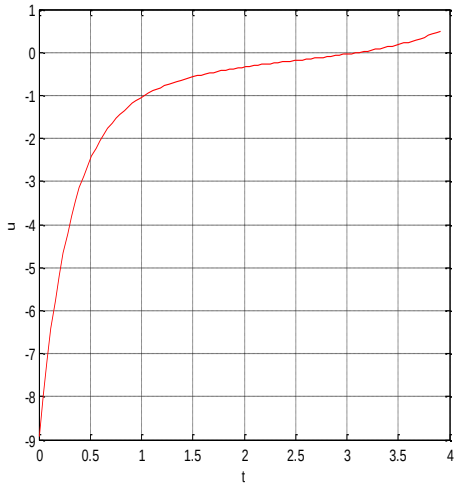


Figure 4.23: Input history for Example4-2

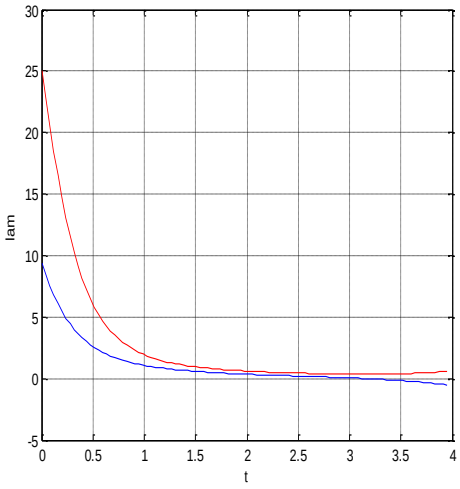


Figure 4.24: Costate history for Example4-2

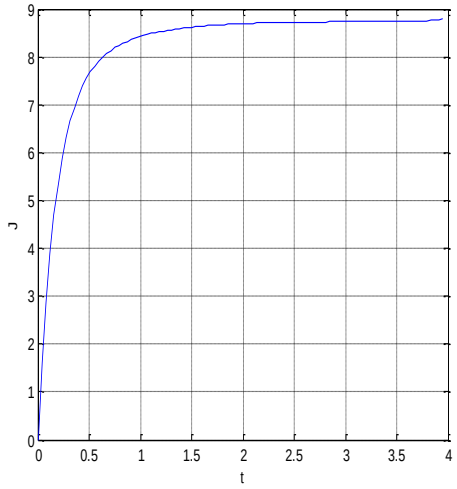


Figure 4.25: The cost value for Example4-2

Table 4.3: The results of simulation for Example 4-2

Method	J	t_f (s)
SBS Method (t_f free)	8.7846	3.95

The results of SBS method for the free final time problem are shown in Table 4.3. The obtained optimal final time t_f is 3.95s. The number of iterations required for convergence is 10.

4.5.3 Example 4-3: A 2-D linear problem with final state constraints and bounded input

This problem is selected to investigate the performance of the SBS method with bounded inputs. It was introduced by Jacobson [5] and it involves a simple linear system with a control magnitude constraint. The system dynamics and performance index are as follows:

$$\begin{aligned} \dot{x}_1 &= -0.5x_1 + 5x_2 \\ \dot{x}_2 &= -5x_1 - 0.5x_2 + u \\ \dot{x}_3 &= -0.6x_3 + 10x_4 \\ \dot{x}_4 &= -10x_3 - 0.6x_4 + u, \qquad \|u\| \le 1 \end{aligned} \tag{4.104}$$

$$J = x_4(T) \tag{4.105}$$

The Hamiltonian for this problem depends linearly on the control. The given initial and final states are $x(t_0) = [10, 10, 10, 10]$ and $x(T) = [x_1(T), x_2(T), x_3(T)] = [2.3, 2.4, 1.5]$. The explicit midpoint rule is applied to solve this problem. The problem is regularized by redefining the performance index as

$$J = x_4(T) + \frac{1}{2}\varepsilon \int_0^{2.5} u^2 dt \tag{4.106}$$

where ε is a small smoothing parameter. The input approaches a bang-bang profile as the value of ε is reduced. A new variable P is introduced to define the boundary satisfaction error as follows:

$$P = (x_1(T) - x_{f1})^2 + (x_2(T) - x_{f2})^2 + (x_3(T) - x_{f3})^2 \tag{4.107}$$

The simulation conditions used for this example are summarized in Table 4.4.

Table 4.4: Simulation conditions for Example 4-3

Variable	Value
Number of data points (n)	300
Input correction (a_u)	0.5

The nominal trajectory is given by the inconsistent choices for the state and control: $\bar{x} = 1$ and $\bar{u} = 0$. The open-loop solution is obtained by using a multiple-shooting method based on the costate and switching times obtained from the SBS method. The simulation results are summarized in Table 4.5.

Table 4.5: The results of simulation for Example 4-3

Method	ε	P	J
SBS method (The explicit midpoint rule)	1e-0	1.97e-27	2.435
	1e-1	2.03e-25	2.321
	1e-2	3.87e-28	2.315

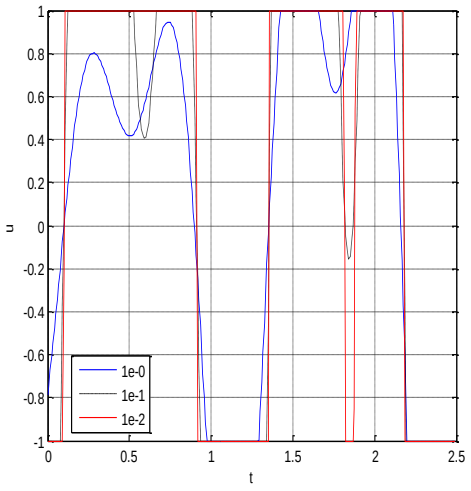


Figure 4.26: Input history for Example4-3 (SBS- $\varepsilon = 1e-0, 1e-1, 1e-2$)

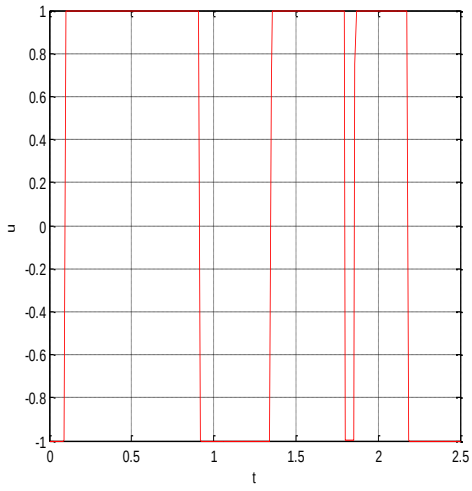


Figure 4.27: Input history for Example4-3 (open-loop)

Figure 4.26 shows the control input profiles obtained for different choices of ε. These results converge to the open-loop control profile shown in Fig. 4.27 as the smoothing parameter value is reduced.

4.5.4 Example 4-4: A hypersensitive problem by using the waypoint scheme

The so called hypersensitive problem is difficult to solve by a shooting method because it involves a highly nonlinear system and a long time duration. The waypoints scheme proves useful for solving this problem. Only two waypoints are sufficient to produce a converged solution from relatively poor initial guesses. The system dynamics and cost function are as for Example 3-2. The nominal trajectory is also the same as used for Example 3-2.

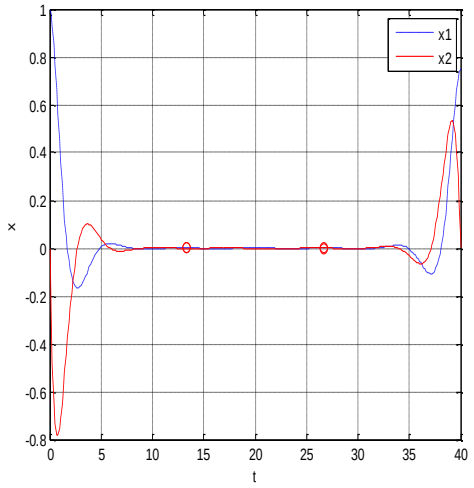


Figure 4.28: State history for Example4-4

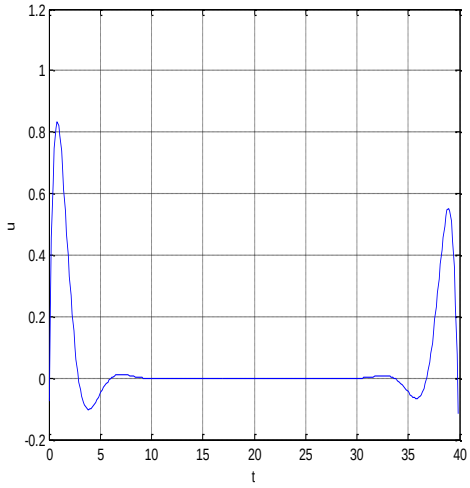


Figure 4.29: Input history for Example4-4

The obtained waypoints and optimal cost value are as follows.

Waypoint 1 : $x_{1wp} = 1\text{e-}004 * [1.443 \;\; -1.563]$

Waypoint 2 : $x_{2wp} = 1\text{e-}004 * [1.123 \;\; 0.978]$

where x_{1wp} is the position of the first waypoint and x_{2wp} is the position of the second waypoint. Analytical solutions for the waypoint locations are at the origin of the state space. The total cost value is $J = 1.689$. Figures 4.28 and 4.29 show the presence of boundary layers, involving rapid changes in the state and control variables.

4.5.5 Example 4-5: A linear problem with impulsive control

A simple nonlinear problem is solved by using the impulse approximation. The system dynamics and cost function [38] are as follows:

.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u \end{aligned} \tag{4.108}$$

$$J = \frac{1}{2}\int_0^3 u^2 dt \tag{4.109}$$

The given initial and final states are $x(t_0) = [1 \quad 0]$ and $x(T) = [0 \quad 0]$. The nominal trajectory variables are arbitrarily selected as $\bar{x} = [0 \quad 0]$, $\bar{u} = 0.1$, $\bar{\lambda} = [0.1 \quad 0.1]$. The simulation conditions are summarized in Table 4.6.

Table 4.6: Simulation conditions for Example 4-5

Variable	Value
The number of impulse (n_{imp})	2/5/10/100
The data points between impulses	100
Iteration number	10
Input correction (α_u)	1

This example is solved for a various number of impulses, n_{imp} : (n_{imp} =2, 5, 10, 100).

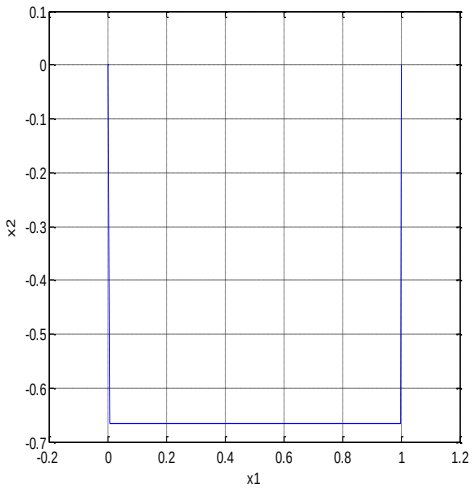


Figure 4.30: State trajectory for Example4-5 (n_{imp} =2)

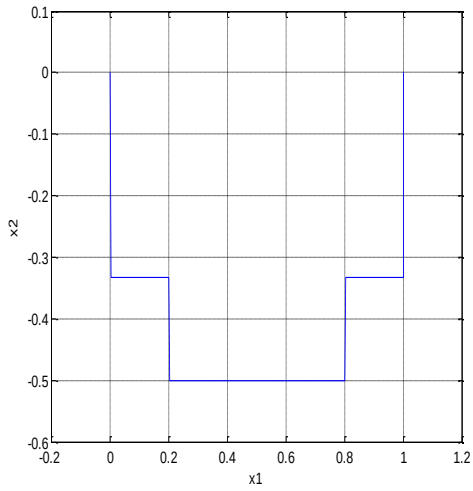


Figure 4.31: State trajectory for Example4-5 (n_{imp} =5)

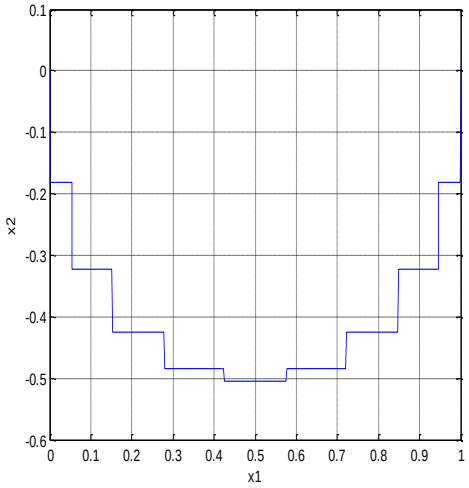


Figure 4.32: State trajectory for Example4-5 (n_{imp} =10)

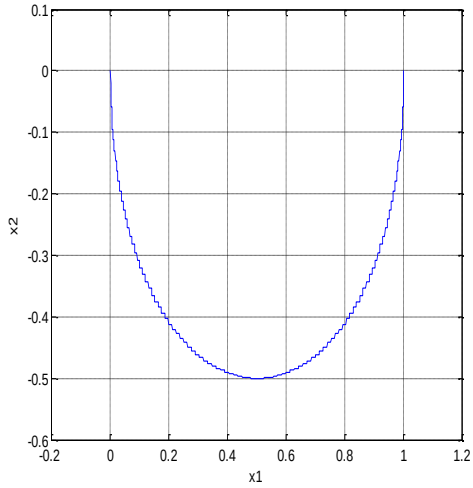


Figure 4.33: State trajectory for Example4-5 (n_{imp} =100)

Fig. 4.30-4.33 show the x_1, x_2 phase plots for the various numbers of impulses. As the number of impulses increases, e.g., the 100 impulse case of Fig. 4.33, the solution approaches that for the optimal continuous control.

5. THE SBS METHOD AUGMENTED WITH A HOMOTOPY ALGORITHM

The homotopy algorithm provides a robust method for determining optimal control, in some cases the global minimum solution, as a continuation parameter is varied gradually to regulate the contributions of the nonlinear terms. In this section, the SBS method is augmented with a homotopy algorithm. The approach is effective for highly nonlinear problems with multiple locally optimal solutions. The system dynamics is represented as:

$$\dot{x} = Ax + \varepsilon f(x) + Bu + d \qquad (5.1)$$
(linear) (nonlinear)

where ε is continuation parameter , Ax is the linear part of system and $f(x)$ is the nonlinear part of system. The continuation parameter in Eq. (5.1) can regulate the nonlinear contributions to the system. The parameter $\varepsilon \in [0,1]$ is gradually increased from 0 to 1 in an automatic fashion. The homotopy algorithm is used to solve for the unknown parameters through an embedding [39] defined as:

$$F[z(s), s] = 0 \qquad (5.2)$$

where s is the independent variable and the sweep equation error function F and the unknown vector z are defined as

$$F = P^T x + W\nu + N - x_f = 0 \tag{5.3}$$

$$z = [\varepsilon \quad \nu] \tag{5.4}$$

The nonlinear function F is constructed such that it satisfies Eq. (5.3) exactly when $\varepsilon = 0$. The solution to the problem is obtained for $\varepsilon = 1$, by requiring that the unknown vector z satisfy the differential equation as a function of s:

$$\frac{dF[z(s), s]}{ds} = 0 \tag{5.5}$$

The substitution of Eq. (5.3) into Eq. (5.5) results in

$$\left(\frac{\partial P^T}{\partial \varepsilon} x + \frac{\partial W}{\partial \varepsilon} \nu + \frac{\partial N}{\partial \varepsilon} \right) \frac{\partial \varepsilon}{\partial s} + W \frac{\partial \nu}{\partial s} = 0 \tag{5.6}$$

Equation (5.6) can be expressed in matrix form as follows:

$$\left[\frac{\partial P^T}{\partial \varepsilon} x + \frac{\partial W}{\partial \varepsilon} \nu + \frac{\partial N}{\partial \varepsilon} \quad W \right] \begin{bmatrix} \frac{\partial \varepsilon}{\partial s} \\ \frac{\partial \nu}{\partial s} \end{bmatrix} = 0 \tag{5.7}$$

Equation (5.7) enables continuous updates of the values of ε and ν by integrating along the curve defined by s. In this section, the homotopy algorithm is

applied in the regular SBS method and it can easily be extended to the modified SBS method. The sensitivities required for solving Eq. (5.7) are those of the sweep method gains with respect to ε. These sensitivities can be propagated by simultaneously integrating in backward time, the following equations and the gain matrices of the sweep method, Eqs. (2.38-2.42):

$$\frac{d}{dt}\left(\frac{\partial S}{\partial \varepsilon}\right) = -\frac{\partial S}{\partial \varepsilon} H_{\lambda x} - S\frac{\partial H_{\lambda x}}{\partial \varepsilon} - \frac{\partial H_{x\lambda}}{\partial \varepsilon} S - H_{x\lambda}\frac{\partial S}{\partial \varepsilon} + \frac{\partial S}{\partial \varepsilon} H_{\lambda u} H_{uu}^{-1}(H_{ux} + H_{u\lambda} S) + (H_{xu} + S H_{\lambda u}) H_{uu}^{-1} H_{u\lambda}\frac{\partial S}{\partial \varepsilon} - \frac{\partial H_{xx}}{\partial \varepsilon}, \quad \frac{\partial S}{\partial \varepsilon}(T) = 0 \tag{5.8}$$

$$\frac{d}{dt}\left(\frac{\partial P}{\partial \varepsilon}\right) = \left(\frac{\partial S}{\partial \varepsilon} H_{\lambda u} H_{uu}^{-1} H_{u\lambda} - \frac{\partial H_{x\lambda}}{\partial \varepsilon}\right) P + [(H_{xu} + S H_{\lambda u}) H_{uu}^{-1} H_{u\lambda} - H_{x\lambda}]\frac{\partial P}{\partial \varepsilon}, \quad \frac{\partial P}{\partial \varepsilon}(T) = 0 \tag{5.9}$$

$$\frac{d}{dt}\left(\frac{\partial V}{\partial \varepsilon}\right) = \left(\frac{\partial S}{\partial \varepsilon} H_{\lambda u} H_{uu}^{-1} H_{u\lambda} - \frac{\partial H_{x\lambda}}{\partial \varepsilon}\right) V + [(H_{xu} + S H_{\lambda u}) H_{uu}^{-1} H_{u\lambda} - H_{x\lambda}]\frac{\partial V}{\partial \varepsilon} + \frac{\partial S}{\partial \varepsilon} H_{\lambda u} H_{uu}^{-1}\gamma - \frac{\partial S}{\partial \varepsilon}\alpha - S\frac{\partial \alpha}{\partial \varepsilon} + \frac{\partial \beta}{\partial \varepsilon}, \quad \frac{\partial V}{\partial \varepsilon}(T) = 0 \tag{5.10}$$

$$\frac{d}{dt}\left(\frac{\partial W}{\partial \varepsilon}\right) = \frac{\partial P^T}{\partial \varepsilon} H_{\lambda u} H_{uu}^{-1} H_{u\lambda} P + P^T H_{\lambda u} H_{uu}^{-1} H_{u\lambda}\frac{\partial P}{\partial \varepsilon}, \quad \frac{\partial W}{\partial \varepsilon}(T) = 0 \tag{5.11}$$

$$\frac{d}{dt}\left(\frac{\partial N}{\partial \varepsilon}\right) = \frac{\partial P^T}{\partial \varepsilon}[H_{\lambda u} H_{uu}^{-1}(H_{u\lambda} V + \gamma) - \alpha] + P^T(H_{\lambda u} H_{uu}^{-1} H_{u\lambda}\frac{\partial V}{\partial \varepsilon} - \frac{\partial \alpha}{\partial \varepsilon}), \quad \frac{\partial N}{\partial \varepsilon}(T) = 0 \tag{5.12}$$

where

$$\frac{\partial \alpha}{\partial \varepsilon} = H_{\lambda \varepsilon} - \frac{\partial H_{\lambda x}}{\partial \varepsilon} x \tag{5.13}$$

$$\frac{\partial \beta}{\partial \varepsilon} = H_{x\varepsilon} - \frac{\partial H_{xx}}{\partial \varepsilon} x - \frac{\partial H_{x\lambda}}{\partial \varepsilon}\lambda \tag{5.14}$$

The Lagrange multiplier ν is updated by using the Eq. (2.65), instead using Eq. (5.7) for reducing the integrating error. The Hessian modification can also be applied, if necessary, to improve further the robustness of the method when the initial guesses are poor.

The coefficients of Eqs. (2.51-2.52) are selected such that the convergence rate is adequate, as functions of ε :

$$C_{hxx} = \left| C_{hxx0}(1-\varepsilon)^n \right| \tag{5.15}$$

$$C_{huu} = \left| C_{huu0}(1-\varepsilon)^n \right| \tag{5.16}$$

where C_{hxx0}, C_{huu0} are the initial values of coefficients and n is a constant that determines the convergence rate. The coefficients C_{hxx} and C_{huu} gradually vanish as ε changes from 0 and 1. The value of n depends on the nonlinearity of system. Generally, n=2 or 3 has worked well for the examples considered. A flowchart for the homotopy-SBS algorithm is depicted in Figure 5.1.

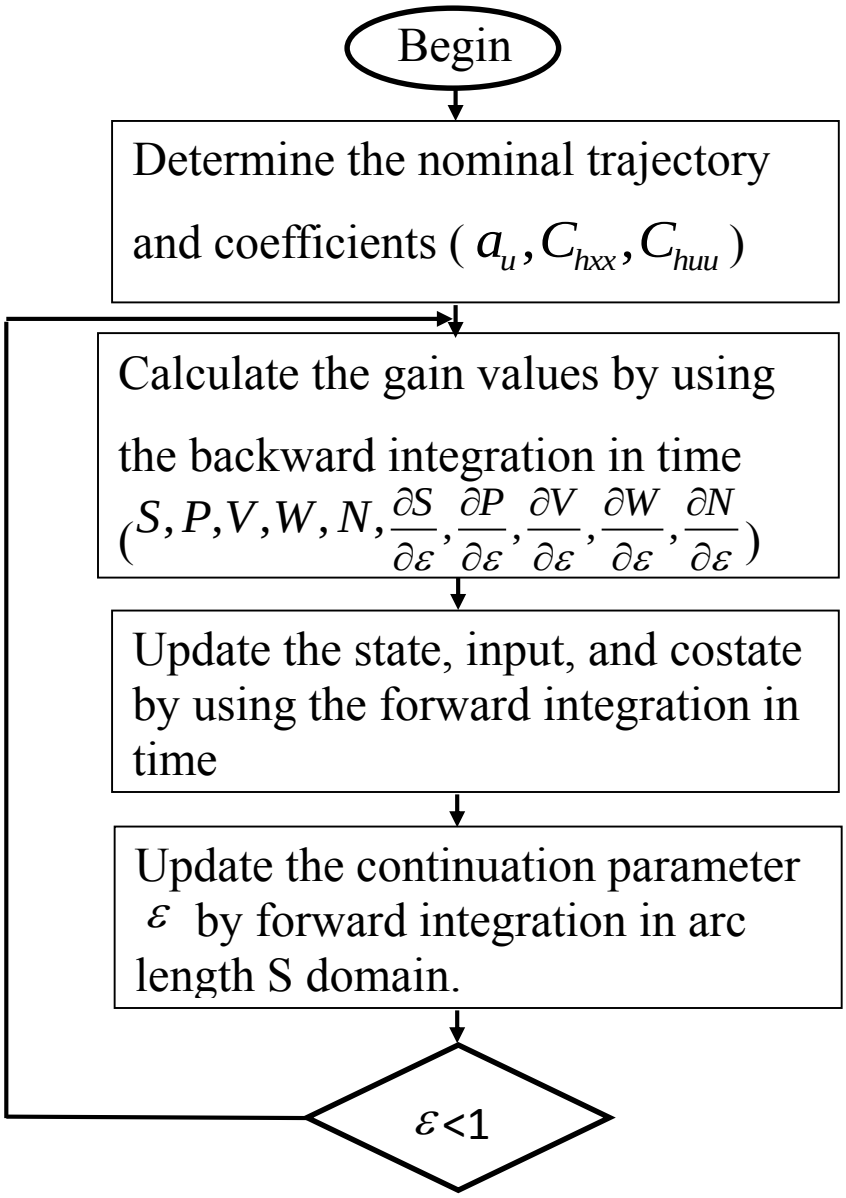


Figure 5.1: The SBS method augmented by a homotopy algorithm

5.1 Numerical examples

To illustrate the performance of the homotopy-augmented SBS method, three highly nonlinear problems having multiple solutions are considered.

5.1.1 Example 5-1: A simple two point boundary value problem without control inputs

This is a TPBVP without a control input. This type of nonlinear problem also can also be solved by using the SBS method. This problem has been treated in Ref. [40] in connection with another approach, called the optimal descent vector (ODV) method. The ODV algorithm is described in Appendix C. The given system dynamic model is

$$\ddot{x} = \frac{3}{2}x^2 \tag{5.17}$$

and the initial and final states are, respectively, $x(t_0) = 4$ and $x(T) = 1$ at $T = 1$. The continuation parameter ε is applied to the nonlinear term in Eq. (5.17):

$$\ddot{x} = \frac{3}{2}\varepsilon x^2 \tag{5.18}$$

Equation (5.18) is linearized by using the Taylor series expansion as follows:

$$\ddot{x} = (3\varepsilon\bar{x})x - \frac{3}{2}\varepsilon\bar{x}^2 = Ax + d \tag{5.19}$$

where

$$f = \frac{3}{2}\varepsilon x^2 \tag{5.20}$$

$$A = 3\varepsilon\bar{x} \tag{5.21}$$

$$d = f - Ax = -\frac{3}{2}\varepsilon\bar{x}^2 \tag{5.22}$$

Equation (5.19) can also be represented in the state space form by defining $x_1 = x$ and $x_2 = \dot{x}$.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= Ax_1 + d \end{aligned} \tag{5.23}$$

The variable $\dot{x}$ can be considered to be the costate for the application of the SBS method:

$$\dot{x} = Sx + P\dot{x}(t_f) + V \tag{5.24}$$

$$P^T x + W\dot{x}(t_f) + N - x_f = 0 \tag{5.25}$$

Since the boundary condition $\dot{x}(t_f)$ is linear, Eq. (2.64) can be expressed as

$$\dot{x}(t_f) = -W^{-1}(P^T x + N - x_f) \tag{5.26}$$

The gain differential equation can be derived by using Eq. (5.23) and Eqs. (5.24-5.25):

$$\dot{S} = -S^T S + A \qquad S(T) = 0 \tag{5.27}$$

$$\dot{P} = -SP \qquad P(T) = I \tag{5.28}$$

$$\dot{V} = -SV + d \qquad V(T) = 0 \tag{5.29}$$

$$\dot{W} = -P^T P \qquad W(T) = 0 \tag{5.30}$$

$$\dot{N} = -P^T V \qquad N(T) = 0 \tag{5.31}$$

The gain differential equation with respect to ε also can be obtained by using the backward integration to Eqs. (5.27-5.31).

$$\frac{d}{dt}\left(\frac{\partial S}{\partial \varepsilon}\right) = -\left(\frac{\partial S}{\partial \varepsilon}\right)S - S^T\left(\frac{\partial S}{\partial \varepsilon}\right) + \left(\frac{\partial A}{\partial \varepsilon}\right), \qquad \frac{\partial S(T)}{\partial \varepsilon} = 0 \tag{5.32}$$

$$\frac{d}{dt}\left(\frac{\partial P}{\partial \varepsilon}\right) = -\left(\frac{\partial S}{\partial \varepsilon}\right)P - S\left(\frac{\partial P}{\partial \varepsilon}\right), \qquad \frac{\partial P(T)}{\partial \varepsilon} = 0 \tag{5.33}$$

$$\frac{d}{dt}\left(\frac{\partial V}{\partial \varepsilon}\right) = -\left(\frac{\partial S}{\partial \varepsilon}\right)V - S\left(\frac{\partial V}{\partial \varepsilon}\right) + \left(\frac{\partial d}{\partial \varepsilon}\right), \qquad \frac{\partial V(T)}{\partial \varepsilon} = 0 \tag{5.34}$$

$$\frac{d}{dt}\left(\frac{\partial W}{\partial \varepsilon}\right) = -\left(\frac{\partial P}{\partial \varepsilon}\right)^T P - P^T\left(\frac{\partial P}{\partial \varepsilon}\right), \qquad \frac{\partial W(T)}{\partial \varepsilon} = 0 \tag{5.35}$$

$$\frac{d}{dt}\left(\frac{\partial N}{\partial \varepsilon}\right) = -\left(\frac{\partial P}{\partial \varepsilon}\right)^T V - P^T\left(\frac{\partial V}{\partial \varepsilon}\right), \qquad \frac{\partial N(T)}{\partial \varepsilon} = 0 \tag{5.36}$$

where

$$A_\varepsilon = 3\bar{x} \tag{5.37}$$

$$d_\varepsilon = -\frac{3}{2}\bar{x}^2 \tag{5.38}$$

Simulation conditions are summarized in Table 5.1.

Table 5.1: Simulation conditions for Example 5-1

Variable	Value
Number of data points in time domain (n)	10
Number of data points in arc length domain (n_a)	10
Input correction (a_u)	1
Iteration number	5

The nominal trajectory is arbitrary chosen: $\bar{x} = 1$. The exact analytical solution is represented as follows:

$$x(t) = \frac{4}{(1+t)^2} \tag{5.39}$$

The analytical solution is used to check the performance of the SBS and ODV methods.

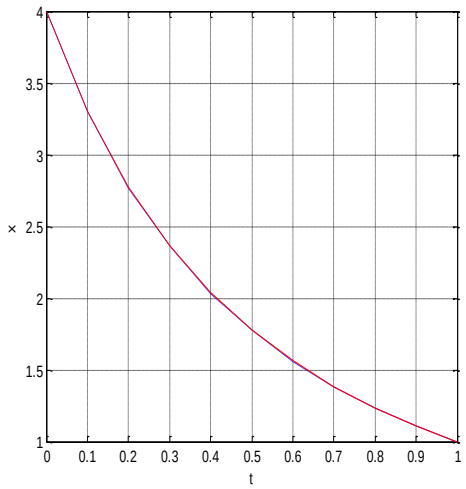


Figure 5.2: State history for Example5-1 (SBS-homotopy)

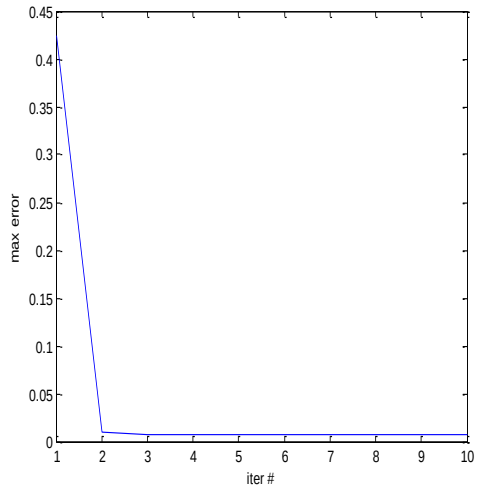


Figure 5.3: State error history for Example5-1 $e = |x - x_a|$ (SBS-homotopy)

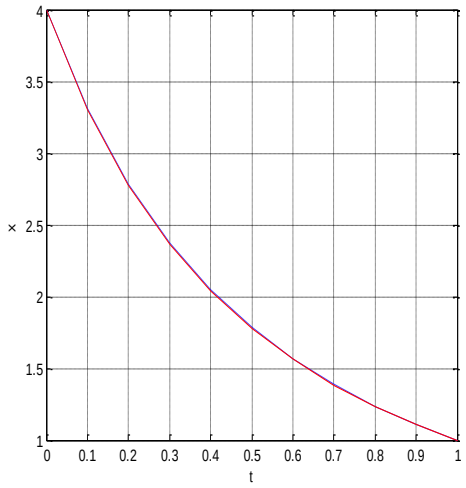


Figure 5.4: State history for Example5-1 (ODV)

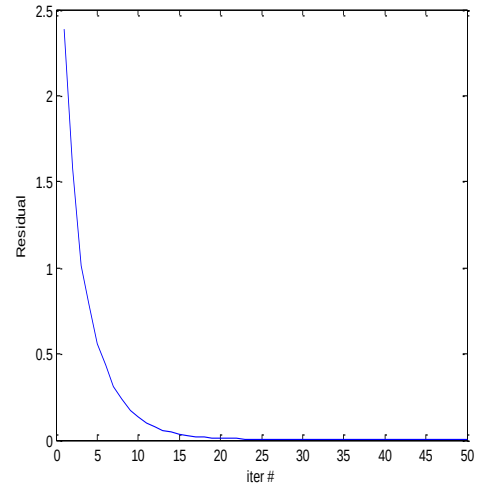


Figure 5.5: State error history for Example5-1 $e = |x - x_a|$ (ODV)

In Figs. 5.2 and 5.4, the red lines indicate the analytical solutions and the blue dotted lines, respectively, indicate the results of the SBS and ODV methods. Figures 5.3 and 5.5, respectively, show the maximum error between the analytical solution and the results of the SBS-midpoint and the ODV methods. The ODV implementation uses the central difference scheme for the numerical solution of Eq. (5.18). The SBS-homotopy method finds the converged solution within the 3 iterations, while the ODV method requires 20 iterations. The maximum errors between the solutions obtained from each method and the analytical solution are shown in Table 5.2

Table 5.2: The converged maximum error for Example 5-1

Method	$\max(\lvert x - x_a \rvert)$
Optimal descent vector	0.0047
SBS Method with continuation parameter	0.0068

where x_a is the state obtained from the analytical solution.

5.1.2 Example 5-2: The Earth to Mars orbit transfer problem

This example is the classical minimum-fuel, coplanar Earth-Mars orbit transfer problem. This problem has multiple local optimal solutions. Among these multiple solutions, the global minimum solution can be obtained by using the SBS-homotopy method for a large number of initial guesses. The simulation is performed in a heliocentric reference frame and system dynamics is represented in the polar coordinates [33] as follows:

$$\dot{r} = u \tag{5.40}$$

$$\dot{u} = \frac{v^2}{r} - \frac{\mu}{r^2} + u_r \tag{5.41}$$

$$\dot{v} = -\frac{uv}{r} + u_\theta \tag{5.42}$$

$$\dot{\theta} = \frac{v}{r} \tag{5.43}$$

where r is the radial distance from the sun, u is the radial velocity, v is the tangential velocity, θ is the angular displacement, u_r is the radial thrust , u_θ is the tangential thrust and μ is the gravitational constant. In this example, canonical units are selected such that $\mu = 1$. The canonical unit of time for the heliocentric system is $1\ TU$ $= 58.132821$ days.

The performance index is defined as follows:

$$J = \frac{1}{2}\int_0^{t_f} (u_r{}^2 + u_\theta{}^2)dt \tag{5.44}$$

The terminal constraint is given as follows:

$$\psi(t_f) = \begin{bmatrix} r(t_f) - r_f \\ u(t_f) - u_f \\ v(t_f) - v_f \end{bmatrix} \tag{5.45}$$

where r_f, u_f, v_f are final desired states. The boundary conditions are summarized in the Table 5.3.

Table 5.3: Boundary conditions for Example 5-2

Variables	Initial value	Final value
Time	0	3.0964 TU
r	1	$r_f = 1.524$
u	0	0
v	1	$\sqrt{\mu / r_f}$
θ	0	free
ε	0.5	1

The simulation conditions are summarized in Table 5.4.

Table 5.4: Simulation conditions for Example 5-3

Variable	Value
Number of data points in time domain (n)	100
Number of data points in arc length domain (n_a)	20
Input correction (α_u)	1
Iteration number	50
The coefficients of hessian matrix (C_{hxx})	1

As previously mentioned, this problem has multiple local minimum solutions, three of which are shown in Figures. 5.6-5.8. The solutions for Cases 1, 2, and 3 were obtained by a random selection of the guesses of the initial costates. The SBS method also converges to the same solutions from a close neighborhood, indicating that each trajectory is a local minimum solution.

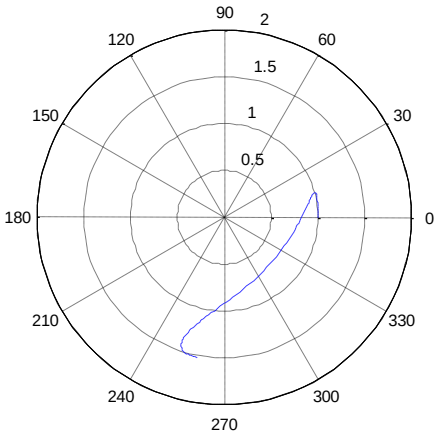


Figure 5.6: Trajectory for Example 5-2 (case 1)

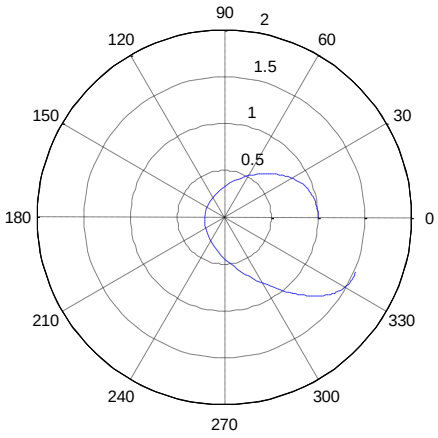


Figure 5.7: Trajectory for Example 5-2 (case 2)

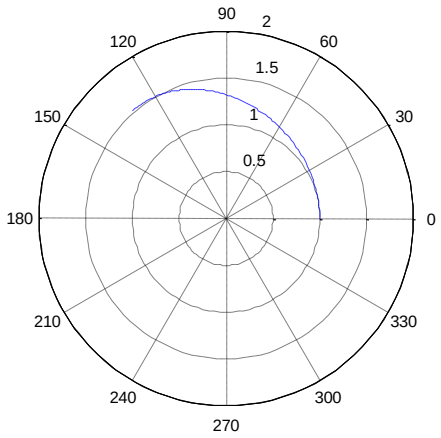


Figure 5.8: Trajectory for Example 5-2 (case 3)

The performance indices returned by the three local minimum solutions are given in Table 5.5.

Table 5.5: The initial costate and cost value for multiple minimum trajectories

Method	Case	λ_0	Cost value (J)
Open-loop (shooting method)	Case1	(-0.680,-0.084,1.639,0)	3.408
	Case2	(1.200,0.672,0.984,0)	0.567
	Case3	(-0.264,-0.154,-0.257,0)	0.040

It can be easily confirmed from the data in Table 5.5 that the trajectory of Case 3 is the global minimum solution for the case of continuous control. The converged initial costate, λ_0 for each case is shown in Table 5.5. Since λ_0 is also the sensitivity of the cost to the initial state, the trajectory corresponding to Case 3 is the least sensitive among the

three solutions, to changes in the initial values of r and v. The naive initial guess used for obtaining the solution depicted in Fig. 5.9 is $\bar{x} = x_0,\ \bar{u} = 0.1,\ \bar{\lambda} = 0.1$. Moreover, the solution of Case 3 is consistently obtained by the SBS-homotopy method, even if the converged solutions for Cases 1 and 2 are used as starting guess. The results of simulation are shown in Figs. 5.9-5.11.

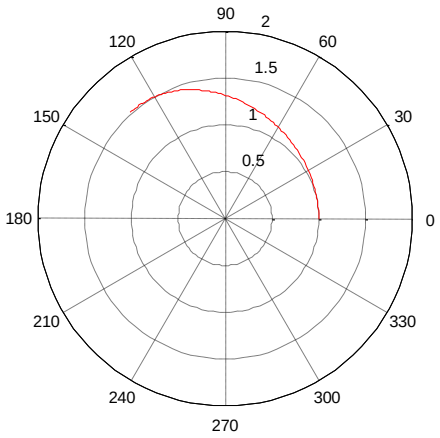


Figure 5.9: Trajectory for Example 5-2 (naive initial guess)

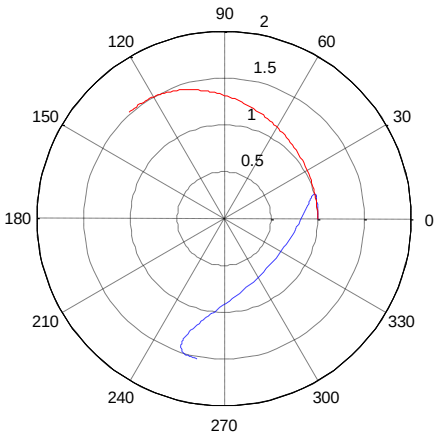


Figure 5.10: Trajectory for Example 5-2 (case 1)

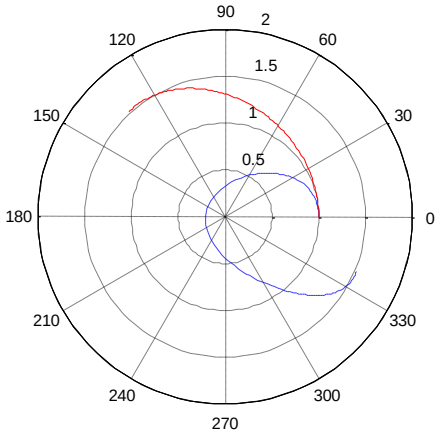


Figure 5.11: Trajectory for Example 5-2 (case 2)

In Figs. 5.9-5.11, the blue line indicates the initial nominal trajectory and the red line, the converged optimal trajectory obtained by the SBS-homotopy algorithm.

5.1.3 Example 5-3: Formulation of the orbit transfer problem using orbital elements

A long duration, minimum fuel transfer orbit from an eccentric initial orbit to a geostationary orbit is considered. This problem, taken from Ref. 40, demonstrates the performance of the SBS-homotopy algorithm. In Ref. 40, it was originally solved with a shooting method, augmented with a homotopy scheme. The formulation uses orbital elements, which also helps in sensitivity reduction. The chosen osculating orbital elements for the planar problem [41] are

$$x = [p, e_x, e_y, L] \tag{5.46}$$

where p is the semi-latus rectum, e is the eccentricity vector and L is the true longitude. The Gauss'equations for the orbital elements are

$$\dot{p} = \sqrt{\frac{p}{\mu}} \frac{2p}{w} u_2 \tag{5.47}$$

$$\dot{e}_x = \sqrt{\frac{p}{\mu}} [\sin(L) u_1 + [\cos(L) + (e_x + \cos(L))/w] u_2] \tag{5.48}$$

$$\dot{e}_y = \sqrt{\frac{p}{\mu}}[-\cos(L)u_1 + [\sin(L) + (e_y + \sin(L))/w]u_2] \tag{5.49}$$

$$\dot{L} = \sqrt{\frac{\mu}{p}}\frac{w^2}{p} \tag{5.50}$$

where

$$w = 1 + e_x\cos(L) + e_y\sin(L) \tag{5.51}$$

The transformations from the space of the orbital elements into the equatorial-plane, Cartesian position and velocity coordinates are

$$x = \frac{p}{w}\cos(L) \tag{5.52}$$

$$y = \frac{p}{w}\sin(L) \tag{5.53}$$

$$\dot{x} = -\sqrt{\frac{\mu}{p}}(e_y + \sin(L)) \tag{5.54}$$

$$\dot{y} = -\sqrt{\frac{\mu}{p}}(e_x + \cos(L)) \tag{5.55}$$

The performance index is defined as:

$$J = \frac{1}{2}\int_0^{t_f}({u_1}^2 + {u_2}^2)dt \tag{5.56}$$

The boundary conditions are summarized in Table 5.6:

Table 5.6: Boundary conditions for Example 5-4

Variables	Initial values	Final values
p	11.625Mm	41.625Mm
e_x	0.75	0
e_y	0	0
L	π	free

The physical parameters and simulation conditions are represented in the Table 5.7.

Table 5.7: Simulation conditions and orbital parameters for Example 5-4

Variable	Value
Number of data points in time domain (n)	3000
Number of data points in arc length domain (n_a)	100
Input correction (α_u)	1
Iteration number	100
The coefficients of Hessian matrix (C_{hxx})	1
The gravitational constant (μ)	1565.862 Mm^3h^{-2}
The initial mass (m)	1500kg

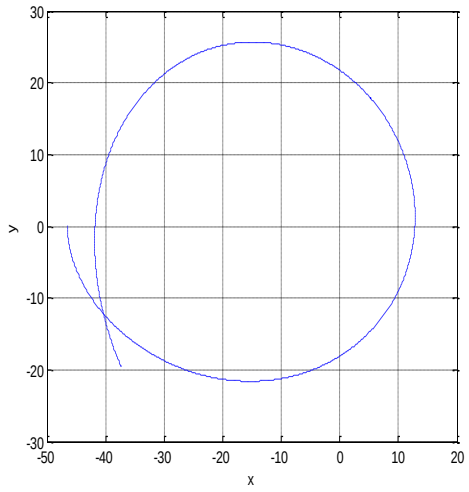


Figure 5.12: Trajectory for Example 5-3 ($t_f = 15$ hours)

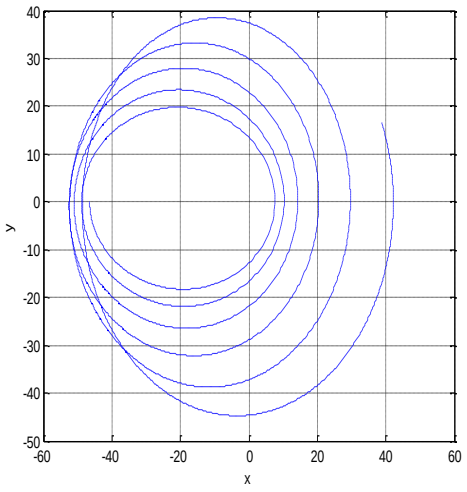


Figure 5.13: Trajectory for Example 5-3 ($t_f = 100$ hours)

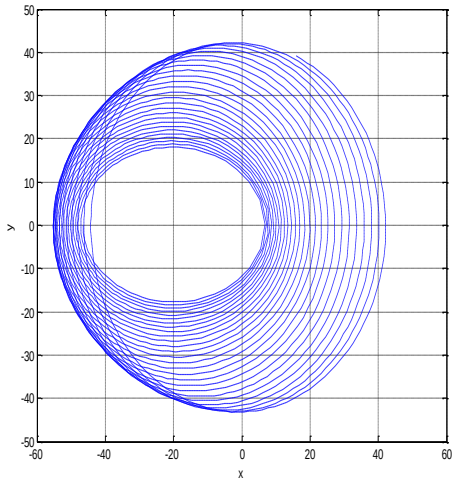


Figure 5.14: Trajectory for Example 5-3 ($t_f = 500$ hours)

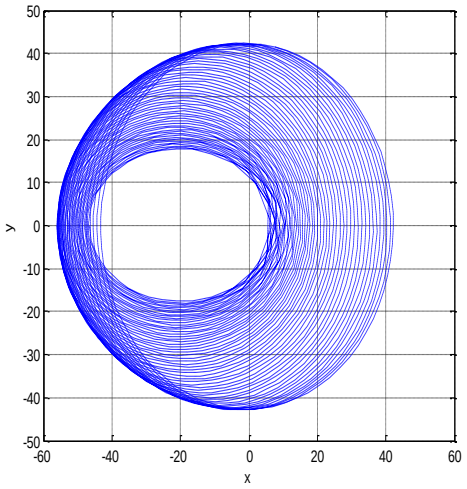


Figure 5.15: Trajectory for Example 5-3 ($t_f = 1000$ hours)

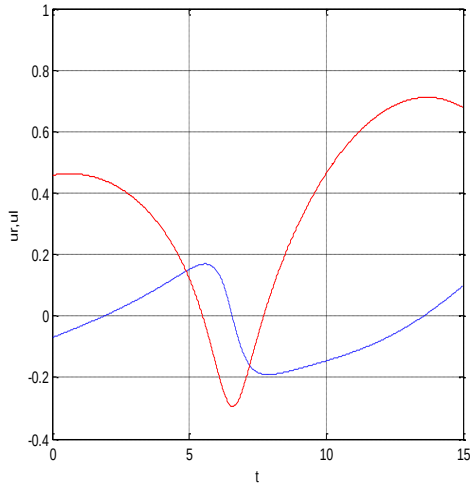


Figure 5.16: Input history for Example 5-3 ($t_f = 15$ hours)

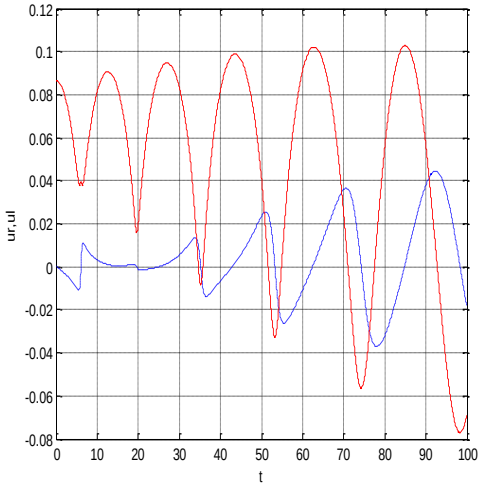


Figure 5.17: Input history for Example 5-3 ($t_f = 100$ hours)

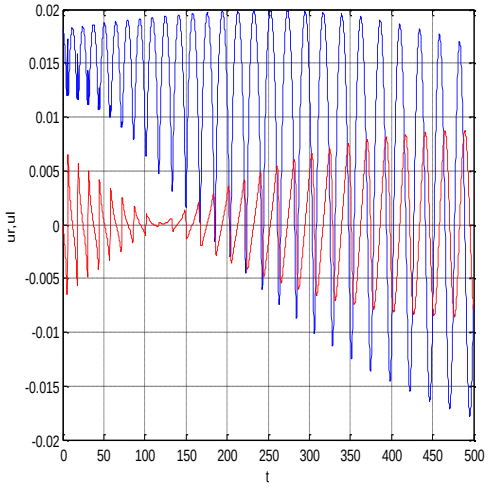


Figure 5.18: Input history for Example 5-3 ($t_f = 500$ hours)

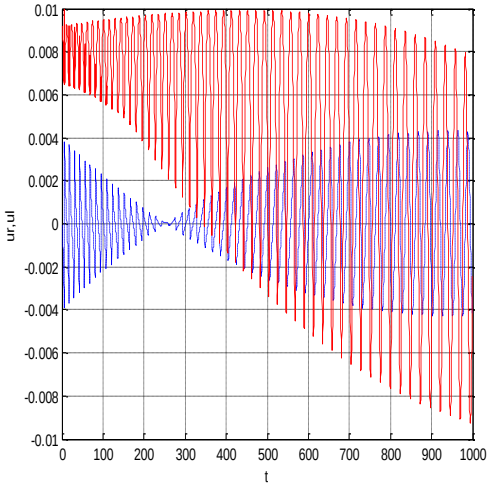


Figure 5.19: Input history for Example 5-3 ($t_f = 1000$ hours)

Figure 5.12 shows the optimal trajectory for $t_f = 15$ hours, with the arbitrary initial nominal trajectory variables selected as: $\bar{x} = x_0,\ \bar{u} = 0.1,\ \bar{\lambda} = 0.1$. The trajectories of Figs. 5.13-5.15 are sequentially obtained from the previous converged optimal solution, starting with the solution of Fig. 5.12. The SBS-homotopy algorithm is used with Hessian modification, as indicated by Eq. (5.15). The corresponding control profiles are shown in Figs. 5.16-5.19. This problem is difficult to solve by the standard SBS method, even in conjunction with the waypoint scheme.

These examples demonstrate that the SBS-homotopy algorithm is an effective method for solving problems of low-thrust trajectory optimization. Often, these problems have multiple local minima. The SBS-homotopy method enables one to gradually seek a global minimum for certain classes of problems.

6. THE RESTRICTED THREE BODY PROBLEM (RTBP)

In this section, an optimal low-thrust transfer orbit from the L1 to L2 liberation point in the Earth-moon system is considered. The RTBP [42] is well known to have rich dynamics, involving chaotic regimes and bifurcations. Hence, any gradient-based method will encounter convergence difficulties on certain classes of problems of the RTBP, unless a root tracing method is employed. Convergence can be achieved by neglecting second-order derivatives of the Hamiltonian during the initial stages of the iteration process. Unfortunately, this seemingly reasonable approach, in some cases, results in convergence on a solution which may not be optimal. Therefore, a method to constrain the solution to a certain neighborhood of the phase space is required. This goal is achieved by adding a penalty on the variation of the Jacobi constant in the performance index, especially for long transfer times. In this section, multiple solutions to the same problem, satisfying the first-order conditions, are investigated for local optimality using the no-conjugate point check. The elements of the STM of the linearized system obtained with reference to a nominal optimal trajectory play an important role in this determination.

Non-dimensional parameters are introduced to simulate the problem. Characteristic parameters [43] in the Earth-moon system are summarized in Table 6.1.

Table 6.1: Non-dimensional parameters for RTBP

Quantity	Value
Earth mass (M_e)	5.974*1e+24 (kg)
Lunar mass (M_m)	7.348*1e+22 (kg)
The gravitational constant (μ)	$M_m/(M_e+M_m)$
Earth moon distance (l^*)	3.844*1e+5 (km)
Characteristic time (t^*)	3.752*1e+5 (sec)

This problem is to find the low thrust trajectory between L1 and L2 liberation points in the Earth-moon system. The locations of thee equilibrium points can be easily calculated in a synodic coordinate system, rotating with the Earth-moon line, given the parameter μ. The position of moon in this coordinate system is $(x, y, z) = (0.9878, 0, 0)$ and the position of Earth is $(x, y, z) = (0.0122, 0, 0)$. The non-dimensional position of L1 and L2 liberation points for the Earth-moon system are as follows:

L1 point : $(x, y, z) = (0.83691531, 0, 0)$

L2 point : $(x, y, z) = (1.15568202, 0, 0)$

The system dynamics of the RTBP is represented as:

$$\dot{x} = \upsilon_x \tag{6.1}$$

$$\dot{y} = \upsilon_y \tag{6.2}$$

$$\dot{z} = \upsilon_z \tag{6.3}$$

$$\dot{\upsilon}_x = 2\upsilon_y + \frac{\partial \Omega_3}{\partial x} + T_x \tag{6.4}$$

$$\dot{\upsilon}_y = -2\upsilon_x + \frac{\partial \Omega_3}{\partial y} + T_y \tag{6.5}$$

$$\dot{\upsilon}_z = \frac{\partial \Omega_3}{\partial z} + T_z \tag{6.6}$$

where, $x, y, z, \upsilon_x, \upsilon_y, \upsilon_z$ are the Cartesian position and velocity coordinates of a satellite with respect to the Earth-moon barycenter and the potential

$$\Omega_3(x, y, z, \mu) = \frac{1}{2}(x^2 + y^2) + \frac{1-\mu}{r_1} + \frac{\mu}{r_2} + \frac{1}{2}\mu(1-\mu) \tag{6.7}$$

$$r_1^2 = (x+\mu)^2 + y^2 + z^2 \tag{6.8}$$

$$r_2^2 = (x+\mu-1)^2 + y^2 + z^2 \tag{6.9}$$

The performance index considered in this section is

$$J_2 = \frac{1}{2}\int_{t_0}^{t_f} (T_x^{\,2} + T_y^{\,2} + T_z^{\,2})dt \tag{6.10}$$

As mentioned previously, a neighboring problem is considered by adding a penalty, based on the Jacobi constant, in Eq. (6.10). The Jacobi constant is

$$c = \frac{1}{2}(\dot{x}^2 + \dot{y}^2 + \dot{z}^2) - \frac{1}{2}(x^2 + y^2) - \frac{1-\mu}{r_1} - \frac{\mu}{r_2} \tag{6.11}$$

The Jacobi constant is an integral of motion along coasting arcs. The penalty function is formulated, depending on whether the final velocity is free or required to be zero, as follows:

Free final velocity

$$J = J_1 + J_2 = \frac{1}{2}\int_{t_0}^{t_f} [Q_j (c - c_0)^2 + T_x^{\,2} + T_y^{\,2} + T_z^{\,2}]dt \tag{6.12}$$

Zero final velocity

$$J = J_1 + J_2 = \frac{1}{2}\int_{t_0}^{t_f} [Q_j (\dot{c} - c_{sl})^2 + T_x^{\,2} + T_y^{\,2} + T_z^{\,2}]dt \tag{6.13}$$

where, J_1 is the cost value related with the Jacobi constant, c_0 is the Jacobi constant at the initial point, c_f is the Jacobi constant at the final point, Q_j is a weighting factor, $\dot{c}$ is the time derivative of the Jacobi constant, and c_{sl} is the slope of Jacobi constant between an initial and a final point. The Jacobi constant c is considered as an additional state when the suboptimal method is applied to the RTBP. The variation of the Jacobi constant along thrusting arcs is given by

$$\dot{c} = \upsilon_x T_x + \upsilon_y T_y + \upsilon_z T_z \tag{6.14}$$

$$c_{sl} = (c_f - c_0)/t_f \tag{6.15}$$

The simulation conditions are summarized in the Table 6.2.

Table 6.2: Simulation conditions for RTBP

Quantity	Value
Number of data points (n)	1000
Input correction (α_u)	1
Iteration number	200
The coefficient of hessian modification (C_{hxx0})	1

Four solutions to the L1-L2 optimal transfer have been obtained for a transfer time of 10 days, by using a shooting method. The Jacobi constant penalty function is not required for this transfer. The four solutions are shown in Figs. 6.1-6.4 and the respective variations in the Jacobi constant are shown in Figs. 6.5-6.8.

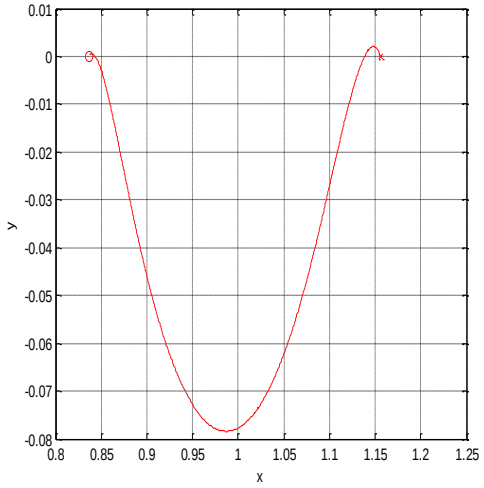


Figure 6.1: Trajectory1 for RTBP

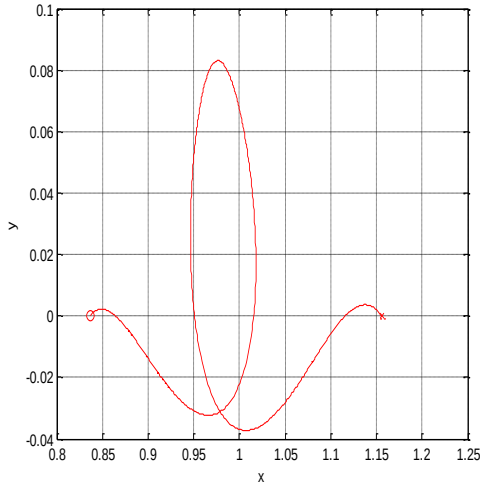


Figure 6.2: Trajectory2 for RTBP

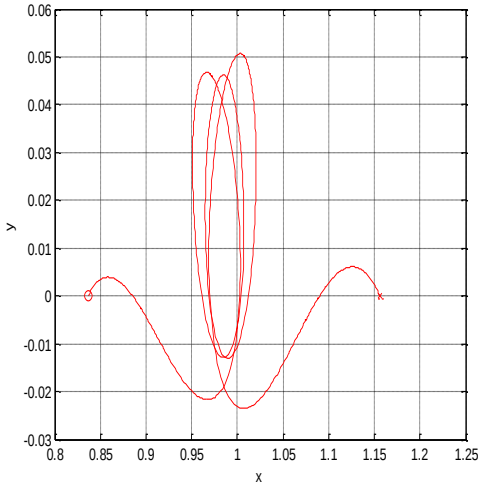


Figure 6.3: Trajectory3 for RTBP

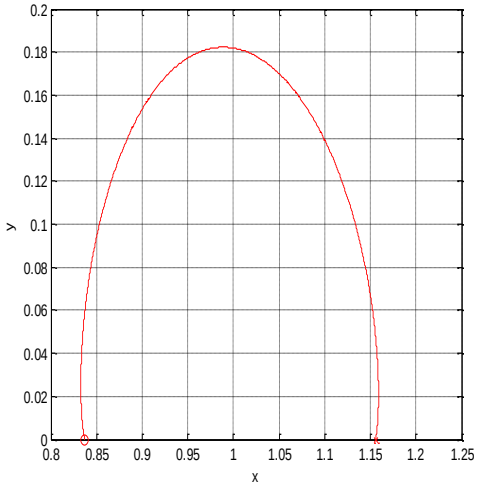


Figure 6.4: Trajectory4 for RTBP

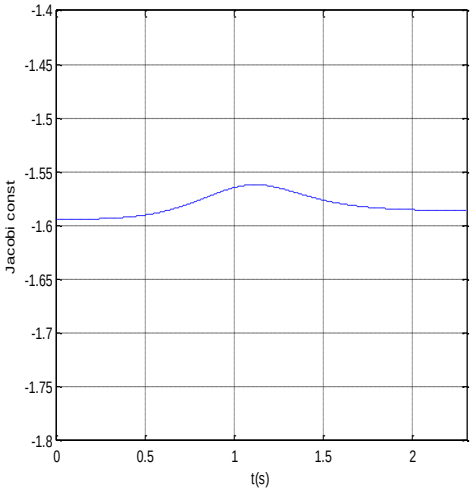


Figure 6.5: Variation of the Jacobi integral (Trajectory1)

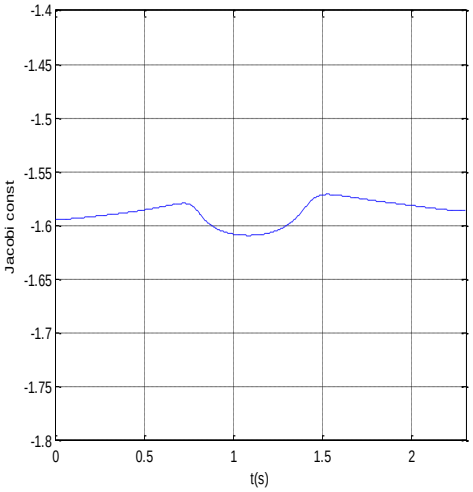


Figure 6.6: Variation of the Jacobi integral (Trajectory2)

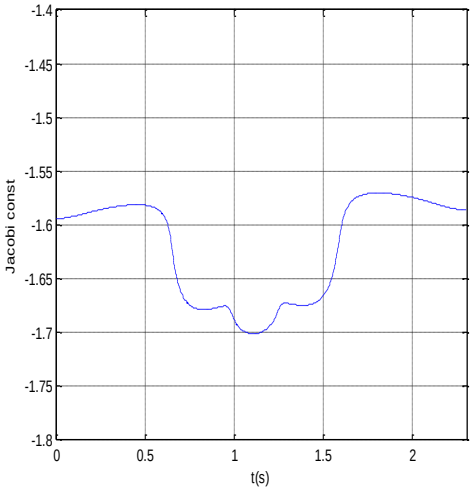


Figure 6.7: Variation of the Jacobi integral (Trajectory3)

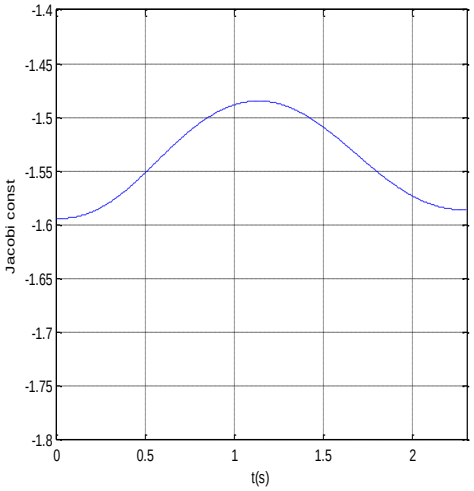


Figure 6.8: Variation of the Jacobi integral (Trajectory4)

The simulation results of the four local minimum solutions are presented in Table 6.3.

Table 6.3: The simulation results for the four local minimum solutions

Trajectory	Initial costate ($\lambda(t_0)$)	J (cost)
Trajectory 1	(-0.384 ,-0.395, 0,-0.163, -0.102, 0)	0.041
Trajectory 2	(-1.549 ,-0.164, 0,-0.419, -0.214, 0)	0.058
Trajectory 3	(-3.017 ,-0.232, 0,-0.721, -0.361, 0)	0.194
Trajectory 4	(-0.021 , 0.770, 0, 0.260, -0.419, 0)	0.286

The Jacobi constant variations are significantly larger for the two higher-cost solutions, it being the most gentle for the lowest cost trajectory, Fig. 6.1.

The modified SBS method is applied to these four solutions to check for the satisfaction of the second order conditions. Trajectories 1, 2, and 4 show the existence of neighboring extremal trajectories, thus proving that conjugate points do not exist for them. Trajectory 3, however, fails to produce a neighboring extremal trajectory. All the four trajectories are further tested for the no-conjugate point sufficient condition using the sub-matrices of the STM, as described in Section 2.2. Figures 6.9-6.16 show the traces of $\det(\phi_{11})$ and $\det(\phi_{12})$ along the four trajectories.

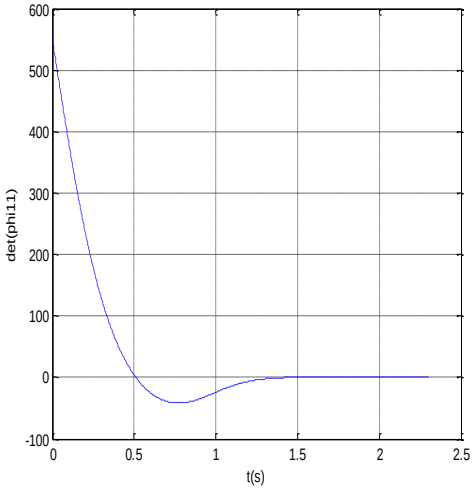


Figure 6.9: Variation of $\det(\phi_{11})$ (Trajectory1)

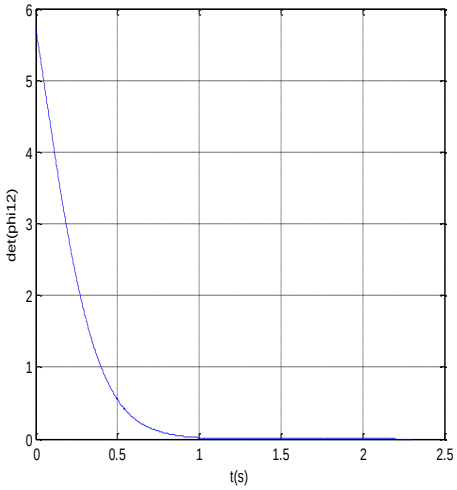


Figure 6.10: Variation of $\det(\phi_{12})$ (Trajectory1)

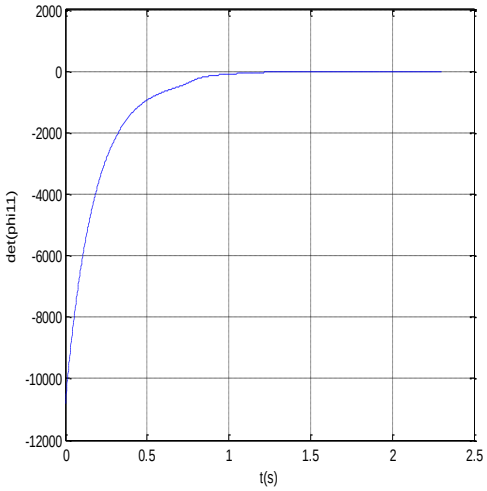


Figure 6.11: Variation of $\det(\phi_{11})$ (Trajectory2)

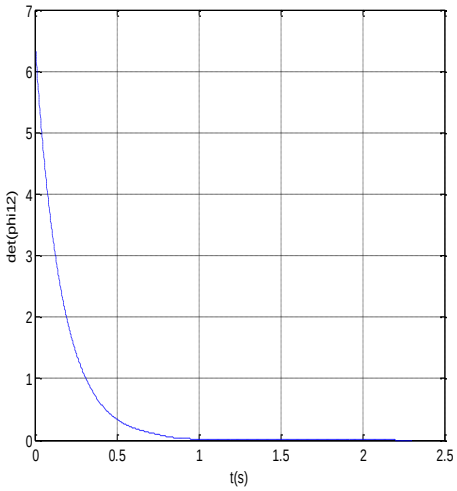


Figure 6.12: Variation of $\det(\phi_{12})$ (Trajectory2)

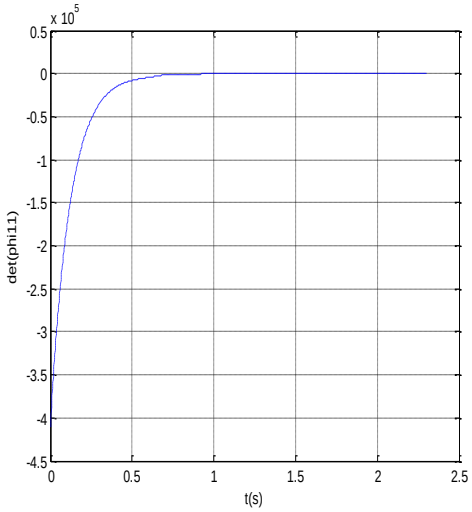


Figure 6.13: Variation of $\det(\phi_{11})$ (Trajectory3)

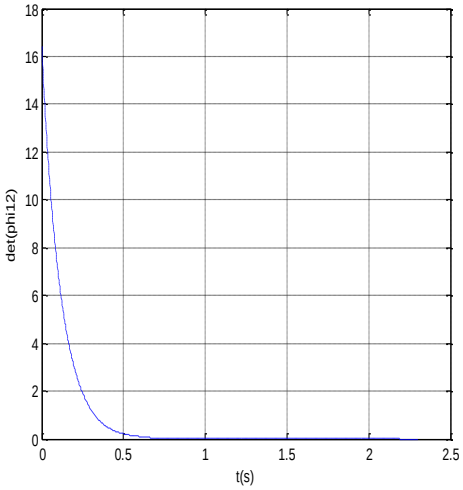


Figure 6.14: Variation of $\det(\phi_{12})$ (Trajectory3)

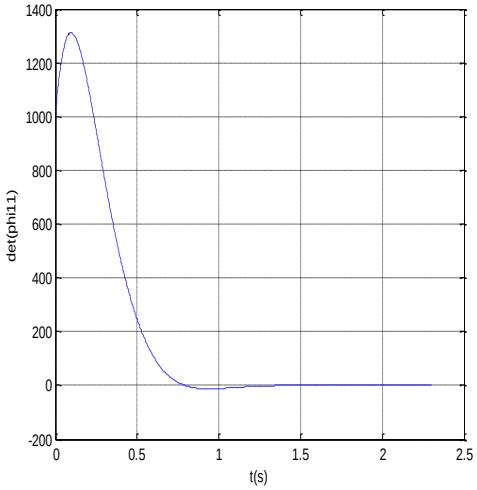


Figure 6.15: Variation of $\det(\phi_{11})$ (Trajectory4)

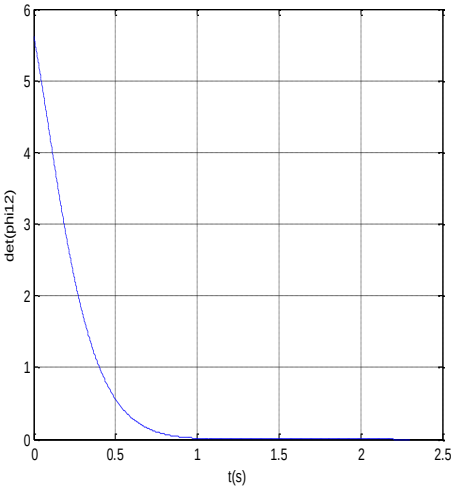


Figure 6.16: Variation of $\det(\phi_{12})$ (Trajectory4)

Data from these figures indicate that $\det(\phi_{12}) > 0$ at all points except the final time. However, $\det(\phi_{12})$ values are quite small on large segments of the trajectories for all the cases. There are two singular points for ϕ_{11} ($\det(\phi_{11}) = 0$) for trajectories 1 and 4, and only one each for trajectories 2 and 3. Hence, a time point can be found when Eq. (2.26) can be used for computing $\bar{S}$. Therefore, $\det(\phi_{12})$ is evaluated when the magnitude of $\det(\phi_{11})$ becomes to zero for the first time during the back integration. These results are shown in Table 6.4.

Table 6.4: The magnitude of $\det(\phi_{12})$ at $\det(\phi_{11}) = 0$

Type of trajectory	Time t_1 ($\det(\phi_{11}) = 0$)	$\det(\phi_{12})$
Trajectory 1	1.4497	8.941e-5
Trajectory 2	1.6156	6.632e-6
Trajectory 3	1.6988	1.415e-6
Trajectory 4	1.3904	1.946e-4

The results presented in Table 6.4 show that the value of $\det(\phi_{12})$ at the critical time is the smallest for trajectory 3. Such a low value of $\det(\phi_{12})$ prevents the initiation of the modified SBS method. As a further check, condition numbers of ϕ_{11} and ϕ_{12} are plotted in Figs. 6.17-6.24.

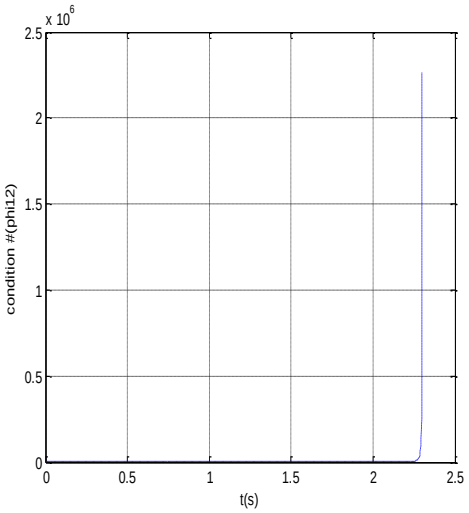


Figure 6.17: Condition # of ϕ_{11} (Trajectory1)

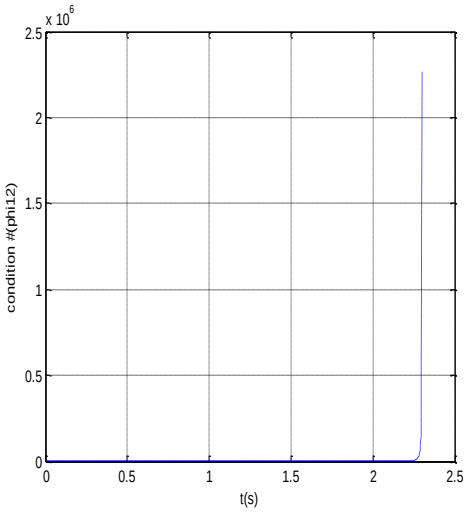


Figure 6.18: Condition # of ϕ_{12} (Trajectory1)

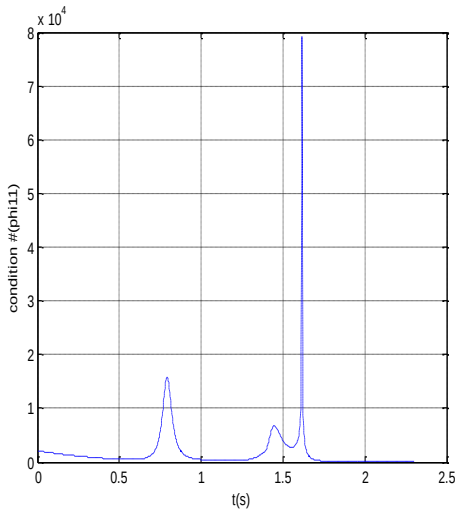


Figure 6.19: Condition # of ϕ_{11} (Trajectory2)

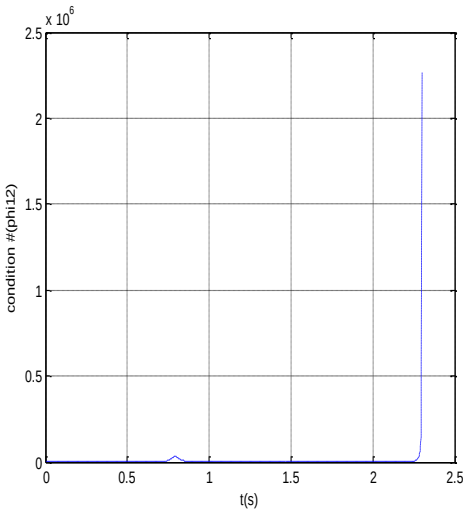


Figure 6.20: Condition # of ϕ_{12} (Trajectory2)

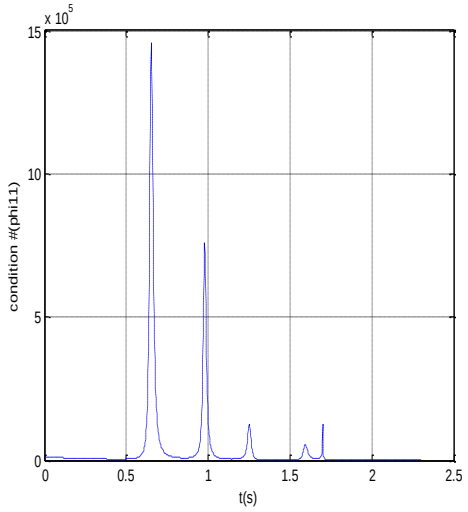


Figure 6.21: Condition # of ϕ_{11} (Trajectory3)

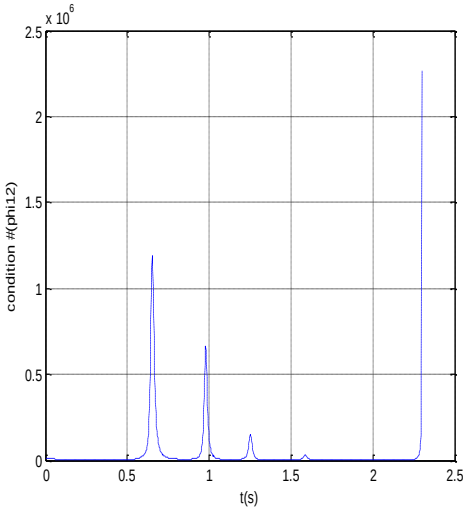


Figure 6.22: Condition # of ϕ_{12} (Trajectory3)

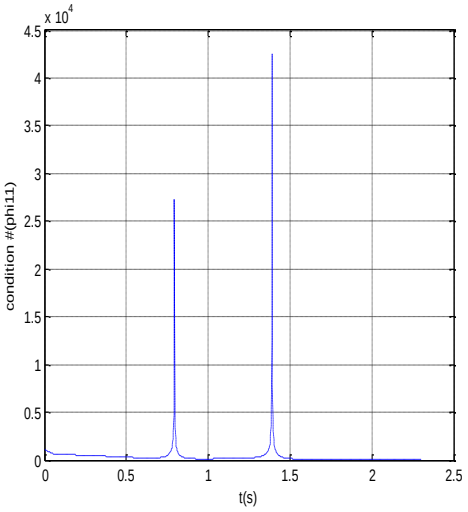


Figure 6.23: Condition # of ϕ_{11} (Trajectory4)

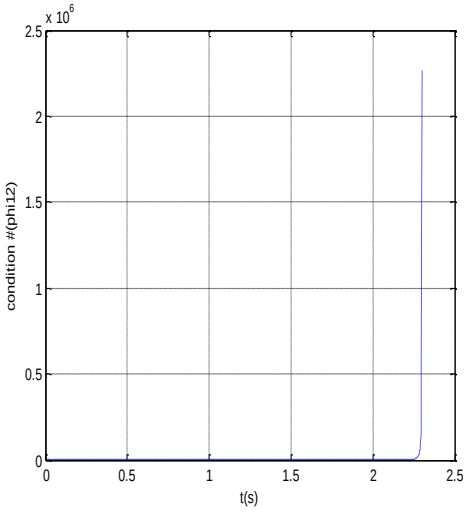


Figure 6.24: Condition # of ϕ_{12} (Trajectory4)

As mentioned in Section 2, ϕ_{11} must remain nonsingular for the gain S to remain finite. Similarly, ϕ_{12} must be nonsingular for $\bar{S}$ to remain finite. The Jacobi no-conjugate-point condition requires that $\bar{S}$ remain finite along the trajectory, except at the final time. A comparison of Figs. 6.18, 4.20, 6.22, and 6.24 shows that the condition number of ϕ_{12} along trajectory 3 is significantly higher, indicating that it is ill-conditioned. Hence, a conjugate point exists on trajectory 3. This is the reason for the lack of neighboring extremals for this trajectory and the failure of the SBS method.

Trajectory 1 is extended for longer durations with the inclusion of the penalty function based on the Jacobi constant. The converged solution for the 10-day optimal trajectory is used to initiate a sequential process to determine trajectories for increasing durations. The converged solution for one problem is used as the initial guess for the next problem in the sequence. The midpoint-SBS method is used to determine L1-L2 optimal solutions with free as well as zero final velocities. Figures 6.25-6.30 show the free-final-velocity trajectories obtained for final times ranging from 10days to 60days. The corresponding Jacobi constant variations are shown in Figs. 6.31-6.36.

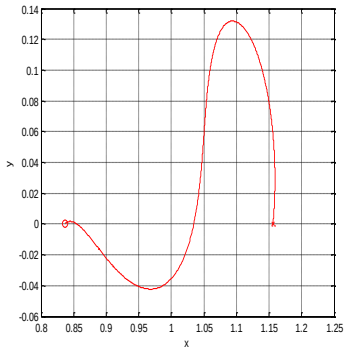


Figure 6.25: Trajectory (10days-free final velocity)

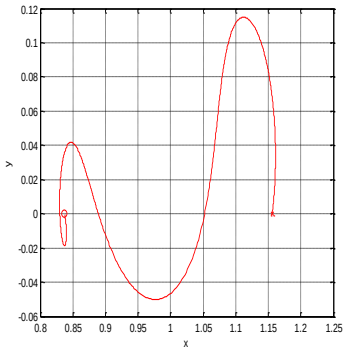


Figure 6.26: Trajectory (20days-free final velocity)

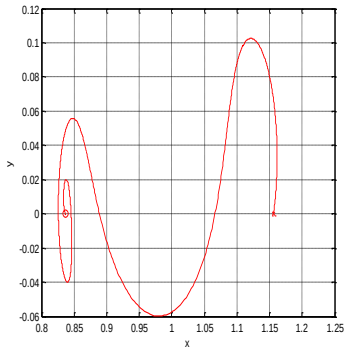


Figure 6.27: Trajectory (30days-free final velocity)

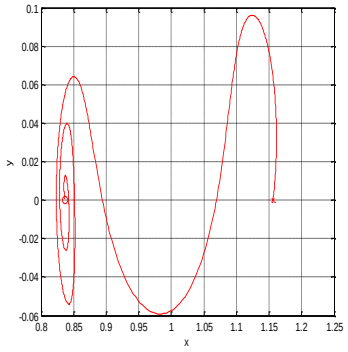


Figure 6.28: Trajectory (40days-free final velocity)

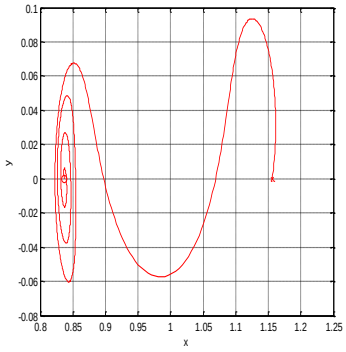


Figure 6.29: Trajectory (50days-free final velocity)

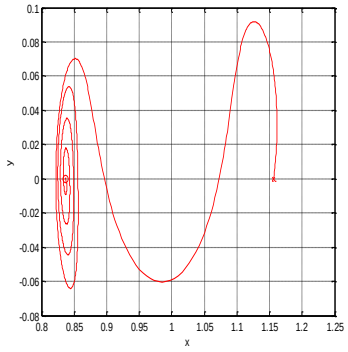


Figure 6.30: Trajectory (60days-free final velocity)

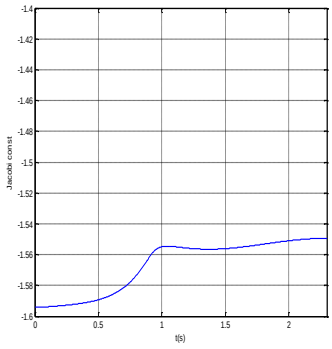


Figure 6.31: Jacobi constant (10days-free final velocity)

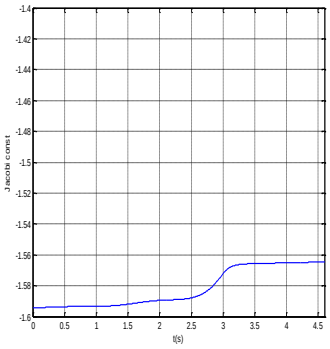


Figure 6.32: Jacobi constant (20days-free final velocity)

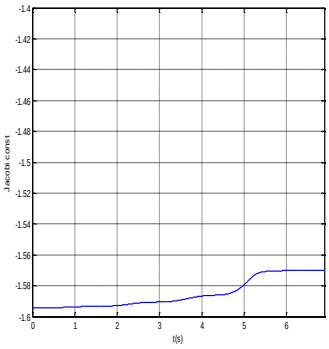


Figure 6.33: Jacobi constant (30days-free final velocity)

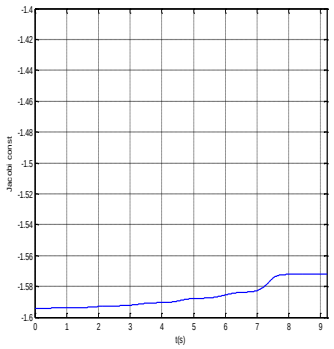


Figure 6.34: Jacobi constant (40days-free final velocity)

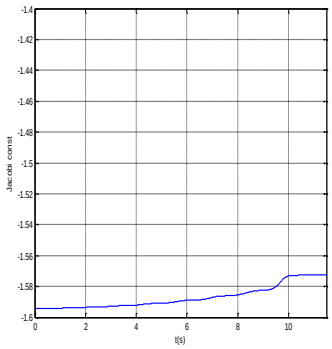


Figure 6.35: Jacobi constant (50days-free final velocity)

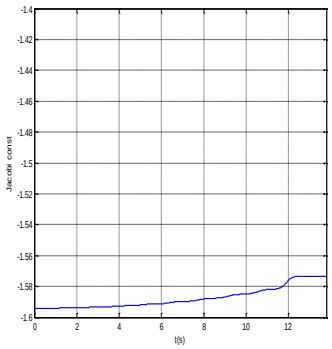


Figure 6.36: Jacobi constant (60days-free final velocity)

Table 6.5: The cost values at each final time (the free final velocity)

J (Cost)	10 days	20 days	30 days	40 days	50 days	60 days
J_1	0.0012	0.0007	0.0006	0.0006	0.0006	0.0007
J_2	0.0217	0.0068	0.0045	0.0032	0.0026	0.0022
J	0.0229	0.0075	0.0051	0.0038	0.0032	0.0029

Cost values for each final time are shown the Table 6.5. As given by Eqs. (6.10) and (6.12-13), the total cost (J) is a combination of the penalty function (J_1) and the control effort (J_2). The results indicate that there is little change in J_1 for trajectories with final times greater than 20 days. The control cost J_2 steadily decreases with increasing final times. The trajectories in Figs. 6.25-6.30 show increased spiraling about the L1 point as the final time is increased. There are no significant changes in the trajectories near L2. The reason for this behavior is seen from the performance index of Eq. (6.12), which penalizes deviations in the Jacobi constant from that for L1. Figures

6.31-6.36 show that the temporal variations of the Jacobi constant diminish as the final time increases.

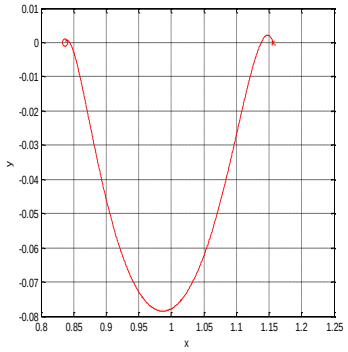


Figure 6.37: Trajectory (10days-fixed final velocity)

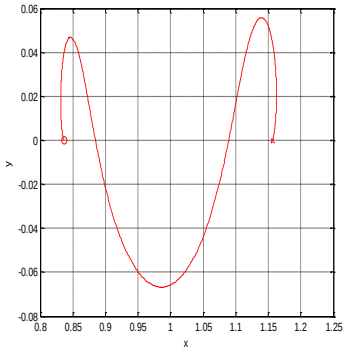


Figure 6.38: Trajectory (20days-fixed final velocity)

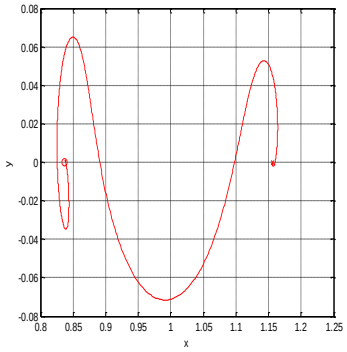


Figure 6.39: Trajectory (30days-fixed final velocity)

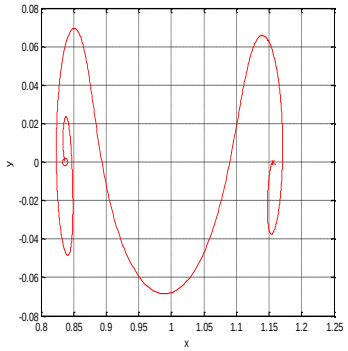


Figure 6.40: Trajectory (40days-fixed final velocity)

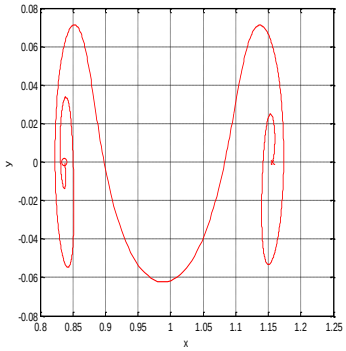


Figure 6.41: Trajectory (50days-fixed final velocity)

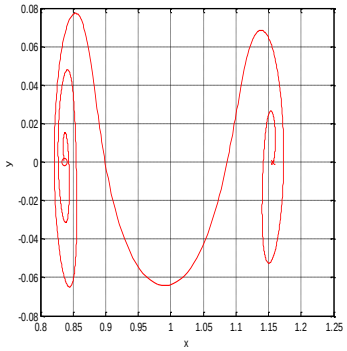


Figure 6.42: Trajectory (60days-fixed final velocity)

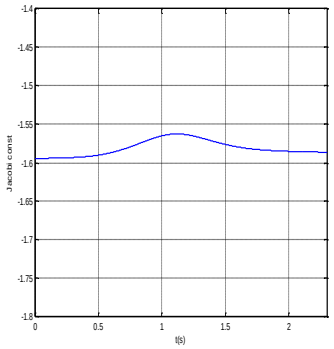


Figure 6.43: Jacobi constant (10days-fixed final velocity)

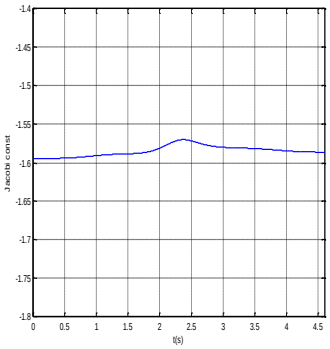


Figure 6.44: Jacobi constant (20days-fixed final velocity)

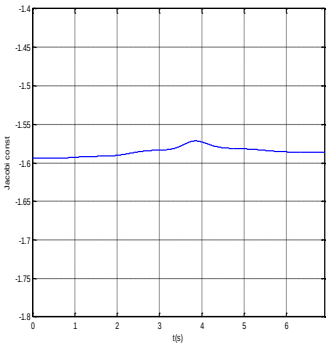


Figure 6.45: Jacobi constant (30days-fixed final velocity)

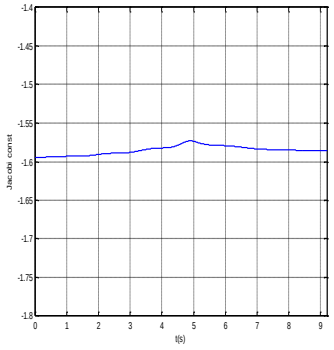


Figure 6.46: Jacobi constant (40days-fixed final velocity)

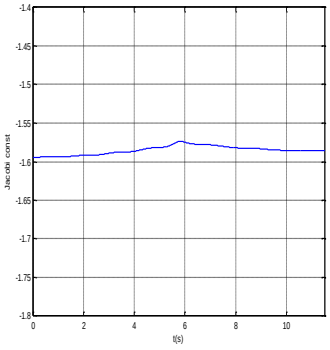


Figure 6.47: Jacobi constant (50days-fixed final velocity)

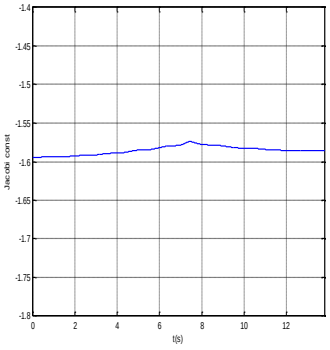


Figure 6.48: Jacobi constant (60days-fixed final velocity)

Table 6.6: The cost values at each final time (the fixed final velocity)

J (Cost)	10 days	20 days	30 days	40 days	50 days	60 days
J_1	0.0011	0.0004	0.0002	0.0001	0.0001	0.0001
J_2	0.0413	0.0185	0.0131	0.0090	0.0076	0.0062
J	0.0424	0.0189	0.0133	0.0091	0.0077	0.0063

Figures 6.37-6.42 and Figs. 6.43-6.48 show the trajectories and the corresponding Jacobi constant histories for increasing final times for the case of zero final velocity. The results for the cost values for each final time selected are shown in the Table 6.6. Unlike the free-final-velocity trajectories, the zero-final-velocity trajectories show symmetrical spiral motion near the two liberation points. For both the cases, there is a gradual deformation of the trajectories as the final time is increased. Such is not the case without the use of the Jacobi constant penalty function. As shown by Eq. (6.14), the variation of the Jacobi constant on a thrusting trajectory decreases with the thrust level. Figs. 6.43-6.48 show a reflection of this property, since an increase in the final time reduces the thrust level required.

7. CONCLUSIONS

This dissertation presents a variety of continuous and discrete-time Successive Backward Sweep (SBS) methods to solve nonlinear optimal control problems, without the need for accurate or consistent initial guesses. Several methods of handling algorithmic and numerical singularities are proposed to increase the domain of convergence of these methods. An aiming point method has been developed to improve terminal constraint satisfaction, especially for arbitrary initial guesses for the states and controls. The use of a 4^{th} order Magnus integrator in conjunction with the continuous-time SBS method is shown to improve terminal constraint satisfaction error over that from a non-symplectic integrator of the same order. The Magnus integrator allows for a trade of reduced computational time for larger step sizes, without a significant reduction in accuracy.

A modified SBS method is developed by incorporating a change in the sweep equations that directly check for conjugate points in the theory of the calculus of variations. This approach eliminates the need for computing the terminal Lagrange multiplier over a significant portion of the trajectory and improves the constraint satisfaction accuracy. This modification allows for a wider domain of application of the SBS method and provides a verification of the sufficient condition for optimality.

Formulating the optimal control problem in the discrete-time domain has several advantages, such as the incorporation of the time stepping scheme into the derivation of consistent necessary conditions for optimality and accurate satisfaction of the boundary

condition. The time-explicit form of the midpoint rule has been extensively employed in this work in conjunction with the SBS method. This second-order scheme is shown to have the same error characteristics as that of a 4^{th} order non-symplectic scheme on several test problems. The SBS method has been extended to solve optimal control problems with impulsive controls. A waypoint scheme has also been proposed to solve highly sensitive problems.

The SBS method is augmented with a homotopy-continuation procedure to isolate and regulate certain nonlinear effects in difficult problems in order to extend its domain of convergence. In certain problems of orbit transfer in the two-body setting, the homotopy approach is able to find global minimum solutions.

The application of the modified SBS method to evaluate the optimality of transfer trajectories between liberation points of the restricted three-body problem has produced interesting results. Multiple solutions are obtained for the same optimal control problem and the modified SBS and STM methods are applied to these solutions to check the no-conjugate point condition. Of the four solutions found for a particular example, three are determined to be locally optimal. An attempt is also made to characterize these multiple solutions based on their Jacobi constant variations. The Jacobi constant penalty function can also be used to augment the performance index for long duration trajectories to obtain suboptimal solutions with reduced sensitivity.

REFERENCES

[1] J. T.Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd Ed., SIAM, Philadelphia, PA, 2010.

[2] B. A. Conway, Ed., *Spacecraft Trajectory Optimization*, Cambridge University Press, New York, NY, 2010.

[3] J. E. Marsden and M. West, "Discrete mechanics and variational integrators," *Acta Numerica,* vol. 10, pp. 357-514, 2001.

[4] D. H. Jacobson, "New second order and first order algorithms for determining optimal control: A differential dynamic programming approach," *Journal of Optimization Theory and Applications,* vol. 2, pp. 411-440, 1968.

[5] D. H. Jacobson, "Differential dynamic programming methods for solving bang-bang control problems," *IEEE Transactions on Automatic Control,* vol. 13, pp. 661-675, 1968.

[6] C. Colombo, "*Optimal trajectory design for interception and deflection of Near Earth Objects*," PhD Thesis, University of Glasgow, Scotland, 2010.

[7] G. Lantoine, "*A methodology for robust optimization of low-thrust trajectories in multi-body environments*," PhD Thesis, Georgia Institute of Technology, GA, 2010.

[8] A. E. Bryson and Y.C. Ho, *Applied optimal control: optimization, estimation, and control*, Taylor & Francis Group, Levittown, PA, 1975.

[9] C. W. Merriam, *Optimization theory and the design of feedback control systems*, McGraw-Hill New York, NY, 1964.

[10] S. K. Mitter, "Paper II: Successive approximation methods for the solution of optimal control problems," *Automatica (Journal of IFAC),* vol. 3, pp. 135-149, 1966.

[11] S. R. Mc Reynolds and A. E. Bryson Jr, "A successive sweep method for solving optimal programming problems," DTIC Document, 1965.

[12] H. J. Kelley, R. E. Kopp, and H. G. Moyer, "A trajectory optimization technique based upon the theory of the second variation," *Astronaut Aeronaut,* vol. 14, pp. 559-582, 1964.

[13] T. Bullock and G. Franklin, "A second-order feedback method for optimal control computations," *Automatic Control, IEEE Transactions,* vol. 12, pp. 666-673, 1967.

[14] P. Dyer and S. R. McReynolds, *The computation and theory of optimal control* vol. 177: Academic Press New York, NY, 1970.

[15] R. A. Hull, J. R. Cloutier, C. P. Mracek, and D. T. Stansbery, "State-dependent Riccati equation solution of the toy nonlinear optimal control problem," in *American Control Conference,* pp. 1658-1662, 1998.

[16] J. R. Cloutier, "State-dependent Riccati equation techniques: an overview," in *American Control Conference,* pp. 932-936, 1997.

[17] S. Banks and K. Dinesh, "Approximate optimal control and stability of nonlinear finite-and infinite-dimensional systems," *Annals of Operations Research,* vol. 98, pp. 19-44, 2000.

[18] J. Parsley and R. Sharma, "Near-optimal Feedback Guidance for and Accurate Lunar Landing," AAS 11-166, 2012.

[19] J.W. Jo and J. E. Prussing, "Procedure for applying second-order conditions in optimal control problems," *Journal of Guidance, Control, and Dynamics,* vol. 23, pp. 241-250, 2000.

[20] P. M. Mereau and W. F. Powers, "Conjugate Point Properties for Linear Quadratic Problems," *Journal of Mathematical Analysis and Applications,* vol. 55, pp. 418-433, 1976.

[21] S. Blanes and E. Ponsoda, "Time averaging and exponential integrators for non-homogeneous linear IVPs and BVPs," *Applied Numerical Mathematics,* vol. 62, pp. 875-894, 2012.

[22] A. Moore, "*Discrete Mechanics and Optimal Control for Space Trajectory Design*," PhD Thesis, California Institute of Technology, CA, 2011.

[23] R. Gao, X. Gao, and J. Liu, "On optimal control problems of a class of impulsive switching systems with terminal states constraints," *Nonlinear analysis,* vol. 73, pp. 1940-1951, 2010.

[24] L. T. Watson and R. T. Haftka, "Modern homotopy methods in optimization," *Computer Methods in Applied Mechanics and Engineering,* vol. 74, pp. 289-305, 1989.

[25] X. Bai, J. D. Turner, and J. L. Junkins, "Bang-bang control design by combing pseudospectral method with a novel homotopy algorithm," in *AIAA Guidance, Navigation and Control Conference,* no. AIAA-2009-5955.

[26] K.L. Teo, "A unified computational approach to optimal control problems," in *Proceedings of the first world congress on World congress of nonlinear analysts,* vol. *III,* pp. 2763-2774, 1996.

[27] R. L. Burden and J. D. Faires, *Numerical Analysis,* 6th Edition, Brooks-Cole Publishing Co., Pacific Grove, CA, 1997.

[28] R. Bertrand, "New smoothing techniques for solving bang-bang optimal control problems-numerical results and statistical interpretation," *Optimal Control Applications & Methods,* vol. 23, pp. 171-197, 2002.

[29] F. Casas and A. Iserles, "Explicit Magnus expansions for nonlinear equations," *Journal of Physics A: Mathematical and General,* vol. 39, pp. 5445-5461, 2006.

[30] B. Kanat, "*Numerical solution of highly oscillatory differential equations by Magnus series method,*" MS Thesis, İzmir Institute of Technology, İzmir, 2006.

[31] A. Rao and K. Mease, "Dichotomic basis approach to solving hyper-sensitive optimal control problems," *Automatica,* vol. 35, pp. 633-642, 1999.

[32] W. E. Williamson Jr, "*Optimal three dimensional reentry trajectories for Apollo-type vehicles*," PhD Thesis, University of Texas at Austin, TX, 1970.

[33] C. M. McCrate, "*Higher-order methods for determining optimal controls and their sensitivities*," MS Thesis, Texas A&M University, TX, 2010.

[34] C. R. Dohrmann, G. R. Eisler, and R. D. Robinett, "Dynamic programming approach for burnout-to-apogee guidance of precision munitions," *Journal of Guidance, Control, and Dynamics*, vol. 19, pp. 340-346, 1996.

[35] D. G. Hull, *Optimal control theory for applications*: Springer Verlag, New York, NY, 2003.

[36] R. Sharma, S. Sharma, and J. Vadali, "Optimal Nonlinear Feedback Control Design Using a Waypoint Method," *Journal of Guidance, Control, and Dynamics*, vol. 34, pp. 698-705, 2011.

[37] W. W. Hager, "Rates of convergence for discrete approximations to unconstrained control problems," *SIAM Journal on Numerical Analysis*, vol. 13, pp. 449-472, 1976.

[38] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*: Wiley, New Jersey, NJ, 2012.

[39] S. K. Rahimian, F. Jalali, J. Seader, and R. E. White, "A new homotopy for seeking all real roots of a nonlinear equation," *Computers & Chemical Engineering*, vol. 35, pp. 403-411, 2011.

[40] C.S. Liu, H.-H. Dai, and S. N. Atluri, "A Further Study on Using x= λ [αR+ βP](P= F-R (F. R)/|| R|| 2) and x= λ [αF+ βP*](P*= R-F (F. R)/|| F|| 2) in Iteratively Solving the Nonlinear System of Algebraic Equations F (x)= 0," *Computer Modeling in Engineering and Sciences,* vol. 81, pp. 195-228, 2011.

[41] J.B. Caillau and J. Noailles, "Sensitivity analysis for time optimal orbit transfer," *Optimization,* vol. 49, pp. 327-350, 2001.

[42] H. Curtis, *Orbital mechanics for engineering students*: Academic Press, New York, NY, 2009.

[43] J. R. Stuart, "*Fuel-optimal, Low-Thrust Transfers Between Liberation Point Orbits*," MS Thesis, Purdue university, IN, 2011.

APPENDIX A

THE DERIVATION OF ALTERNATE FORMS OF THE PERFORMANCE INDEX

Note that $H_{\lambda x} = f_x$ and $H_{\lambda u} = f_u$ in Eq.(2.28). Consider the dynamic system and cost-to-go, given as:

$$\dot{x} = f_x x + f_u u + \alpha, \qquad J = \frac{1}{2}\int_t^{t_f} (x^T Q x + u^T R u) dt$$

where $\alpha = f - f_x \bar{x} - f_u \bar{u}$ and f & all partials are evaluated on a nominal trajectory. The Hamiltonian is defined as:

$$H = \frac{1}{2}(x^T Q x + u^T R u) + \lambda^T (f_x x + f_u u + \alpha)$$

$$H_u = Ru + f_u{}^T \lambda = 0 \rightarrow u = -R^{-1} f_u{}^T \lambda$$

$$\dot{x} = f_x x - f_u R^{-1} f_u{}^T \lambda + \alpha$$

$$\dot{\lambda} = -Qx - f_x{}^T \lambda$$

It follows that

$$\frac{d}{dt}(\frac{1}{2}x^T\lambda)=\frac{1}{2}x^T\dot{\lambda}+\frac{1}{2}\lambda^T\dot{x}=-\frac{1}{2}x^TQx+\frac{1}{2}\lambda^T\alpha-\frac{1}{2}\lambda^T f_u R^{-1}{f_u}^T\lambda$$

$$\frac{d}{dt}(\frac{1}{2}x^T\lambda)-\frac{1}{2}\lambda^T\alpha=-\frac{1}{2}(x^TQx+\lambda^T f_u R^{-1}{f_u}^T\lambda)=-\frac{d}{dt}J$$

Hence, upon integration

$$J(t)=\frac{1}{2}x^T(t)\lambda(t)-\frac{1}{2}x^T(T)\lambda(T)+\frac{1}{2}\int_t^T\alpha^T\lambda dt$$

Under the $x(T)=0$ assumption

$$\therefore J(t)=\frac{1}{2}x^T(t)\lambda(t)+\frac{1}{2}\int_t^T\alpha^T\lambda dt$$

APPENDIX B

THE DISCRETIZATION BY USING THE RK-4TH ORDER METHOD [Ref. 31]

$$\dot{x} = f(x,u,t)$$

$$x_{k+1} = g_k(x_k,u_k)$$

The state transition and control influence matrices are

$$A_k = \frac{\partial g_k}{\partial x_k}, \quad B_k = \frac{\partial g_k}{\partial u_k}$$

The formula of Runge-Kutta 4th order method is

$$g_k = x_k + \frac{(k_1 + 2k_2 + 2k_3 + k_4)}{6}$$

where

$$k_1 = hf(x_k,u_k,t_k)$$

$$k_2 = hf(x_k + k_1/2, u_k, t_k + h/2)$$

$$k_3 = hf(x_k + k_2/2, u_k, t_k + h/2)$$

$$k_4 = hf(x_k + k_3, u_k, t_k + h)$$

and h is the time span of discrete time system. Hence,

$$A_k = I + \frac{1}{6}(\frac{\partial k_1}{\partial x_k} + 2\frac{\partial k_2}{\partial x_k} + 2\frac{\partial k_3}{\partial x_k} + \frac{\partial k_4}{\partial x_k}), \quad B_k = \frac{1}{6}(\frac{\partial k_1}{\partial u_k} + 2\frac{\partial k_2}{\partial u_k} + 2\frac{\partial k_3}{\partial u_k} + \frac{\partial k_4}{\partial u_k})$$

where I is the identity matrix. Furthermore,

$$\frac{\partial k_1}{\partial x_k} = h\frac{\partial f}{\partial x_k}, \quad \frac{\partial k_1}{\partial u_k} = h\frac{\partial f}{\partial u_k}$$

$$\frac{\partial k_2}{\partial x_k} = h\frac{\partial f}{\partial x_k}(I + \frac{1}{2}\frac{\partial k_1}{\partial x_k}), \quad \frac{\partial k_2}{\partial u_k} = h(\frac{\partial f}{\partial u_k} + \frac{1}{2}\frac{\partial f}{\partial x_k}\frac{\partial k_1}{\partial u_k})$$

$$\frac{\partial k_3}{\partial x_k} = h\frac{\partial f}{\partial x_k}(I + \frac{1}{2}\frac{\partial k_2}{\partial x_k}), \quad \frac{\partial k_3}{\partial u_k} = h(\frac{\partial f}{\partial u_k} + \frac{1}{2}\frac{\partial f}{\partial x_k}\frac{\partial k_2}{\partial u_k})$$

$$\frac{\partial k_4}{\partial x_k} = h\frac{\partial f}{\partial x_k}(I + \frac{\partial k_3}{\partial x_k}), \quad \frac{\partial k_4}{\partial u_k} = h(\frac{\partial f}{\partial u_k} + \frac{\partial f}{\partial x_k}\frac{\partial k_3}{\partial u_k})$$

APPENDIX C

THE ALGORITHM OF OPTIMAL DESCENT VECTOR [Ref. 39]

The ODV algorithm for Example (5-1) is based on a central difference scheme for discretizing F_i. Algorithmically, the ODV method is

$$F_i = \frac{1}{\Delta t^2}(x_{i+1} - 2x_i + x_{i-1}) = \frac{3}{2}x_i^2$$

$$R_k = B_k^T F_k$$

$$P_k = F_k - \frac{R_k \cdot F_k}{\|R_k\|^2} R_k$$

$$v_1^k = B_k R_k, \quad v_2^k = B_k P_k$$

$$w_k = \frac{[v_1^k, F_k, v_2^k] \cdot v_1^k}{[v_2^k, F_k, v_1^k] \cdot v_2^k}$$

$$\alpha_k = \frac{\|R_k\|^2}{\|R_k\|^2 - w_k F_k \cdot R_k}, \quad \beta_k = \frac{w_k \|R_k\|^2}{\|R_k\|^2 - w_k F_k \cdot R_k}$$

$$u_k = \alpha_k R_k + \beta_k P_k, \quad v_k = \alpha_k v_1^k + \beta_k v_2^k$$

$$x_{k+1} = x_k - (1-\gamma)\frac{F_k \cdot v_k}{\|v_k\|^2} u_k$$

where i is the number of data points , k is the number of iterations, $0 \le \gamma < 1$ and

$B = \frac{\partial F}{\partial x}$. Note that $[a,b,c] = (a \cdot b)c - (c \cdot b)a$