SENSITIVITY METHODS APPLIED TO ORBITAL PURSUIT-EVASION

A Dissertation

by

WILLIAM THOMAS HAFER

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

DOCTOR OF PHILOSOPHY

August 2014

Major Subject: Aerospace Engineering

ABSTRACT

In this work, sensitivity methods are examined as a means to solve and analyze the problem of orbital pursuit-evasion (PE). Orbital PE is a two-sided spacecraft trajectory optimization problem characterized by high dimensionality and nonlinearity. Modern methods for solving problems of this sort employ generic, computationally intensive techniques, including random search methods such as the genetic algorithm; collocation methods based on discrete approximation; or combinations of these methods. The advantages of these methods are relatively high degrees of robustness, straight-forward implementation, and ease of handling state and control constraints. Yet we note the disadvantages: chiefly high computation load, as well as absence of insight into the problem, and accuracy of the result. Sensitivity methods provide corresponding strengths in each of these areas. We present novel sensitivity analysis techniques that may be useful in other optimization problems featuring high dimensionality, nonlinearity, and/or state and control constraints. The techniques shown include a novel solution method; a computationally efficient feedback control technique; a means of sketching barrier surfaces; and the use of hybrid one-sided/two-sided controllers for sophisticated emergent behavior. We also introduce a new formulation of the problem incorporating a minimum-altitude constraint, and we make an initial investigation of a sensitivity-based method of handling state constraints. Overall, our results suggest that sensitivity methods can provide useful augmentation to techniques that rely more heavily upon computational power, and may be particularly valuable for implementation in an onboard control algorithm.

ACKNOWLEDGEMENTS

I thank my advisor, Dr. Helen Reed, for supporting me in her lab throughout my studies, and for providing an example of academic as well as professional leadership that I have learned much from in my time here.

I thank Dr. Khanh Pham of the Air Force Research Laboratory, for providing me the opportunity to begin this research.

I thank the members of my committee: Dr. John Hurtado, who has been a fixture not only as an excellent professor, but as a source of positive energy and enthusiasm throughout my graduate school career; Dr. Srinivas Vadali, for his patient teachings of the challenging subject matter of optimal control, and for his firm standard of academic achievement; and Dr. Jennifer Welch, for helping out with last-minute committee requirements, and asking me useful questions about my research.

I thank Dr. James Turner for his assistance in defining the initial steps of this research during the summer we both spent in Albuquerque, New Mexico.

Finally, I thank my parents, for not only raising me until I was a competent adult, but supporting me in my endeavors even afterwards.

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

CHAPTER I

INTRODUCTION AND LITERATURE REVIEW

The focus that has emerged from this research is the development of new techniques using sensitivity methods that add value to existing optimization techniques. However, such a study should consider a variety of challenging optimization problems, while here, we only consider one, that of spacecraft orbital pursuit-evasion (PE). The reason is that this effort was motivated by the desire to solve the orbital PE problem in the course of the author's summer internship at the Air Force Research Laboratory (AFRL), and our efforts have focused on delivering useful results for that problem. Indeed, we only present one other optimal control problem, the Breakwell problem, for reasons that we will describe shortly. First, we describe the orbital PE problem.

In this problem, optimal trajectories are sought for a pursuer and an evader spacecraft in an orbital dynamics environment. (Figure 1) The pursuer wishes to intercept the evader, while the evader wishes to avoid or prolong interception. The basic problem is to find simultaneously optimal (saddle-point) trajectories for the pursuer and the evader.

The interest in the problem comes from several questions. Two questions that may arise in a military setting are: What are optimal spacecraft pursuit maneuvers, and correspondingly, optimal evasive maneuvers? However, non-adversarial situations can also be considered. Sometimes the interception or avoidance of an object that is passive, yet of unknown dynamic properties, is modeled as adversarial so as to capture worstcase scenarios. Thus, a trajectory for a spacecraft to avoid an incoming piece of orbital debris might also be modeled as a PE game.

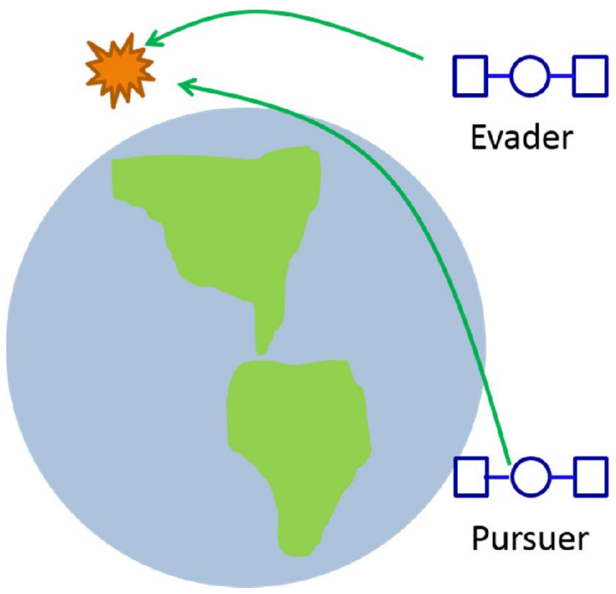


Figure 1 Orbital pursuit-evasion

The problem is characterized by high dimensionality and non-linearity, two hallmarks of a general class of challenging problems in modern control. The usual methods for solving problems of this sort employ generic, computationally intensive techniques. These include random search methods such as the genetic algorithm; interpolation methods such as collocation; or combinations of these methods. The advantages of these methods are relatively high degrees of robustness, straight-forward implementation, and ease of handling state and control constraints. But these methods also have disadvantages: chiefly high computation load, as well as absence of insight into the problem, and the sensitivity and accuracy of the result. We have been intrigued by the ability of sensitivity methods in some cases to solve these types of problems with great speed. An additional benefit is a strong verification of the optimality of the solution. And the methods lend themselves well to several interesting methods that involve continuously updating the solution in real time, as will be discussed.

Sensitivity methods, specifically the computation of trajectory sensitivity matrices to examine local sensitivities, are well-known. The essence of the technique is described in [1], and a thorough treatment is given in [2].

Our approach differs from many perturbation studies because rather than using the sensitivity matrices to examine small perturbations to a solution, we perform the process iteratively to find the solution to a problem whose conditions that may be arbitrarily different from the conditions we began with. In other words, we merge the sensitivity matrix method with a homotopy method. In pursuing this approach, we followed the footsteps of Kim et al [3], who applied the problem to a failed-reaction-wheel attitude control analysis. Prior to Kim, a general suggestion regarding the method was made by Majji et al. [4]

The contributions of this dissertation are:

1. A novel homotopy-based solution technique for the orbital PE problem (without altitude constraint).
2. A homotopy-based feedback control technique that is particularly attractive for an onboard controller. This feedback solution requires one initial solution,

but does not require a mesh of pre-computed solutions throughout the subsequent trajectory space as was employed in a prior method. [5]

3. A means of quickly locating and sketching barrier trajectories in the vicinity of a trajectory of interest. Previously, barrier trajectories in orbital PE had only been found in the region of linear approximation, in [6].
4. A general sensitivity-based method for obtaining the solution to an optimal control problem with state inequality constraints, by beginning with the solution to the unconstrained problem.
5. A reformulation of the orbital PE problem to rigorously enforce a minimum-altitude constraint.
6. Definition and implementation of hybrid one-sided/two-sided controllers that easily allow sophisticated, useful behaviors to emerge.

Regarding the third item, barrier trajectories are limiting trajectories that arise in differential games. [7] The union of barrier trajectories is a barrier surface, and at this surface, the solution of the differential game is discontinuous. The barrier surface has important practical implications for determining conditions under which interception may become impossible, or impossible within a certain time. For this reason, the ability to sketch the barrier surface in a computationally efficient manner is valuable in analysis, and may have uses in online control as well.

The fourth and fifth items involve an application of the theory of optimal control with state variable inequality constraints to the orbital PE problem. Previously, the optimal control formulation of the problem had not included state constraints (for example, in [8] and [9]). A minimum altitude constraint was imposed in post-processing. The constrained orbital PE formulation and the sensitivity-based method for finding a constrained solution, both shown here, could be used to obtain an altitude-constrained solution. However, although we show the sensitivity method for a simpler problem, namely the Breakwell problem, its complete implementation is complex, and remains an item for future work. However, a partial implementation reveals a rather wandering quality of the homotopy function—behavior that must be examined carefully before this method could be considered for online control.

The sixth item is a concept for how a differential (two-sided) optimizing controller can be usefully employed on a spacecraft. In practice, the spacecraft will need to spend most of its time in a one-sided control mode, i.e. performing station-keeping or some other routine operation. The spacecraft will need a mechanism for switching to a two-sided control only if an adversarial situation with another spacecraft manifests. We demonstrate how using the differential game value function as a switching criterion gives rise to useful control behaviors in this regard.

Here, we comment on relevant prior methods, and their relationship to the methods of this paper. A prior work by Pontani and Conway [8] solved the same problem using a two-step solution method consisting of a genetic algorithm (GA) followed by a semi-direct method, where the 'direct' refers to Direct Collocation by Nonlinear Programming (DCNLP). The semi-direct method is abbreviated semi-DCNLP. To aid discussion of the present work, Pontani and Conway's two-step method is diagrammed in Figure 2:

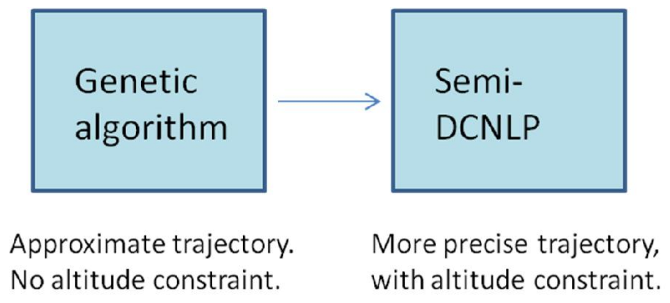

Genetic algorithm
Semi-DCNLP
Approximate trajectory. No altitude constraint.
More precise trajectory, with altitude constraint.

Figure 2 Method employed in prior orbital PE solution

Pontani and Conway's genetic algorithm does not enforce trajectory constraints, while the semi-DCNLP post-processor does. In cases where the altitude constraint is not active in the optimal solution, our homotopy solution, when successful, obtains a solution satisfying the same properties as the GA/DCNLP solver of [8]. In cases where the altitude constraint is active, the homotopy method could only be used in place of the GA. The time savings over the GA is still significant, as the GA described computes 50,000 trajectories. The number of trajectories run during the homotopy is variable, but in solving case 3 from [8], 222 trajectories were run, some of which could be removed in a streamlined code. These trajectories included sensitivity computation, and so should not be compared one-to-one to the GA trajectories, but even still, the speed-up is significant.

The remaining contributions listed earlier leverage the computational efficiency of the sensitivity method to achieve other purposes useful in the context of analysis or real-time control. However, the applicability of the homotopy solution is limited by the behavior of the homotopy function, in ways that will be described here. A useful question for further study would be to know a priori whether or not a homotopy method will be successful. In general, however, we have found that the homotopy method can extend the region of success beyond the linear approximation region, but may not have as large a range as the GA/semi-DCNLP method.

The remainder of this dissertation is organized as follows. In the next section, we provide review relevant literature. In Chapter II, we present the orbital PE problem mathematically, and describe the method of computing a sensitivity matrix for an orbital PE trajectory. In Chapter III, we present the homotopy solution technique, as well as a feedback control technique. As mentioned, the homotopy solution does not have global convergence, and so both successful and unsuccessful cases are shown. In Chapter IV, we describe the barrier surface in the context of this problem, and we show the connection between the barrier surface and failure of the homotopy method. We then present a means of quickly locating and sketching barrier surfaces, in the context of a useful application. In Chapter V, we present a sensitivity method for solving an optimal control problem with state variable inequality constraints, given a solution to the unconstrained problem. We demonstrate this method with the Breakwell problem, and we show a formulation of the orbital PE problem including a minimum altitude inequality constraint. We show a partial implementation of the sensitivity method to the orbital PE minimum altitude constraint, which suggests challenges with the approach, but still suggests that the full implementation may be worthwhile. In Chapter VI, we demonstrate new ideas for hybrid spacecraft controllers that use the game value function as a switching criterion, taking advantage of sensitivity methods in this process. In Chapter VII we make suggestions for future work, and in Chapter VIII we provide conclusions.

Literature review

We divide the literature review into three main topics: PE games in general; orbital PE games; and relevant numerical methods.

Pursuit-evasion games

In 1965, Rufus Isaacs provided the first seminal work on the subject of differential games. [7] The definition and analysis of barrier trajectories was one of the most important components of his work. Isaacs showed a means of identifying barrier trajectories at the terminal surface. Then propagating barrier trajectories backward in time, he was able to sketch the barrier surface in games whose barrier surface was relatively simple. For games of arbitrarily high dimension, it is only possible to visualize local, lower-dimensional sub-manifolds of the barrier surface. Furthermore, locating the barrier surface is more difficult in such games; we overcome this problem here using a sensitivity method.

Missile guidance has been a major application for differential games. Here, we will only cite a few recent examples. In 2001, Turetsky and Shinar compared two linear missile guidance games, "linear quadratic differential game" and a "norm differential game", the latter using the norm of the terminal separation vector for the cost function. [10] In 2003, Shima, Shinar and Weiss developed a differential games guidance law compensating for the effects of estimation delay. [11] In 2010, Trottemont, Scherer, Weiss and Vermeulen developed robust differential games control laws accounting for disturbances and model uncertainties. [12]

Each of these missile guidance studies used some or all of the following assumptions: linearization along the initial line of sight, planar geometry, and absence of gravity. These assumptions are not used in the fully nonlinear orbital PE problem, although some of them have been employed employed to obtain simplified solutions.

Recently, in his 2009 Masters Thesis, Johnson makes a case study of two well-known differential games, the Target Guarding problem and the Homicidal Chauffeur. [13] These problems are too simple to represent realistic engineering applications, yet still their solutions are not straight-forward. Johnson shows various sophisticated techniques, such as a boundary value problem solver, a decomposition method, and a neural network. None of these techniques could be guaranteed to provide the correct solution for the problems considered, given arbitrary initial conditions.

The take-away is that while the differential game framework can be used to *define* problems that are realistic and compelling, finding the solution is often a challenge for problems of even modest complexity.

Orbital pursuit evasion

Compared to the studies of missile guidance, orbital PE studies comprise a much smaller body of literature. The trend over the years has been to include in the model more of the full nonlinear dynamical environment.

In 1967, Wong considered orbital PE in a constant-gravity environment, i.e. unchanging in both magnitude and direction. In this environment, he finds that saddle-point solutions of pursuer and evader thrust directions will be both constant in time, and identical. (This is similar to the result in a gravity-less environment which we use here as the starting point to a continuation solution.) Wong considered the present definition of the game, as well as a variant based on minimizing terminal miss distance. [14]

In 1975, Anderson and Grazier considered orbital PE linearized about a circular orbit. They considered a set non-zero terminal distance as sufficient for interception, but this is not considered to be a fundamental difference. They focused on barrier trajectories, which is also a focus of this work. [6]

In 1981, Kelley, Cliffe and Lutze examined a different form of the game considering impulsive control. This study focused on cases involving a simplified, near-circular orbit. [15]

In 1988, Menon, Calise and Leung also considered a different form of the game. They begin with a general gravity model, a general actuator model, and more complex cost function, and consider a controller employing feedback linearization. [16]

In 2009, Pontani and Conway presented solutions to full non-linear problem in its present definition. Their solutions were obtained using a genetic algorithm pre-processor followed by a semi-DCNLP (Direct Collocation NonLinear Programming) solver. [8] Some of the cases presented in that study will also be examined here.

In 2011, Shen et al. also provided solutions to the full non-linear problem. They obtained solutions using the TOMLAB commercial optimization software, whose numerical techniques are proprietary. [9]

In 2013, Ghosh and Conway provided a sub-optimal feedback implementation. This was done by creating a mesh of solution points around a nominal trajectory, and then finding solutions for an arbitrary trajectory by extrapolation. It was necessary for the trajectory to remain within the confines of the pre-computed mesh. [5] Here, we will show a feedback control method that does not have this limitation.

Finally, the development of sensitivity matrices for orbital PE and some solutions obtained using this method were first presented by the present author in [17].

Numerical methods

In this section, we provide references on homotopy methods, and on the use of sensitivity methods in constrained optimal solutions.

Homotopy methods are covered thoroughly in the 1990 text by Allgower and Georg. [18] The predictor-corrector method discussed there is the method we used. Another reference that relates the topic to optimal control is [19]. The method is sometimes called Davidenko's method.

In 1991, Bullirsch, Montrone and Pesch used a homotopy method to find optimal trajectories for an aircraft abort landing in the presence of wind shear. [20] In 2006, Gergaud and Haberkorn solve a minimum-fuel orbit transfer problem, by performing a homotopy from a minimum-energy solution. [21] Their solution is notable because they begin by verifying the expected convergence of the homotopy function—a property that will not apply globally in our work.

In 2014, Kim applied a homotopy method to find optimal attitude maneuvers for an under-actuated spacecraft. [3] Kim's method for generating sensitivity matrices is the same as we used, except that we must compute additional sensitivities arising from the fact that the orbital PE problem is final-time-free.

In 2009, Majji et al. described the general idea of using sensitivity matrix computations as a generic solution technique for two-point boundary value problems. [4] The technique for generating sensitivity matrices for a trajectory is described by Schaub and Junkins in their textbook. [1]

In fields outside of engineering, in 1978, Chow et al. proposed a method of creating a homotopy function for a variety of problems, whose solution curve is guaranteed to be smooth. [22] This method is of interest for the present problem, but represents a fundamentally different type of homotopy, and has not been applied to date. Also, in 1999, Eaves and Schmedders demonstrated homotopy methods for solving economic systems, in which they selected homotopy functions that were found by inspection to be smooth. [23]

We note that while verifying smoothness and continuity properties of a homotopy function in a theoretical math environment is readily done, performing the same verification for a homotopy function defined using the natural parameters of an engineering problem is often more difficult. The study by Gergaud and Haberkorn is the only example we have seen of this accomplishment. In the present problem, we show that the homotopy function is *not* always continuous, and relate this finding to the theory of barrier surfaces in differential games. We suggest that the general problem of proving (or disproving) the smoothness of a homotopy function constructed using the natural parameters of an engineering problem may be an area for future research.

Finally, we cite references related to applying sensitivity techniques to the solution of an optimal control problem with state variable constraints. The constrained optimal control formulation used here was given by Bryson and Ho. [24] The question of obtaining a sensitivity matrix for an optimal trajectory with state/control inequality constraints has been considered. Two relevant examples are [25], which provides a numerical method for computing perturbation solutions in the vicinity of a constrained optimal trajectory; and [26], which focuses on providing a proof of solution differentiability, and also provides a numerical example. While both of these references encompass many of the techniques employed in our method, we provide a separate development that narrows the focus and clearly presents the means of forming the sensitivity matrices required for our purpose.

Another notable reference is the work of Hiskens and Pai [27], in which he provides a general extension of sensitivity methods to the case of hybrid dynamical systems. Although not commonly thought of as such, the constrained optimal control solution is a hybrid dynamical system, and many of our techniques are also developed by Hiskens and Pai, and we found their rigorous development to be very helpful. Again, however, the formulation we develop is more useful for the state-constrained optimal control problem we wish to consider.

CHAPTER II

PURSUIT-EVASION TRAJECTORIES AND SENSITIVITY MATRICES

We begin this chapter with a generic description of optimal trajectories in PE games, followed by a development specific to orbital PE. We then develop the method of sensitivity matrices, which will be applied in all our subsequent analyses.

Optimal trajectories in PE games

Here, we define the general PE game problem, and the characteristics of its optimal trajectories. The foundation for the differential game theory presented here is that of Isaacs [7], although we use much notation from Bryson and Ho [24].

We have a state variable $\boldsymbol{x}$ which here will always be the composite of pursuer and evader states: $\boldsymbol{x} = [\boldsymbol{x}_P{}^T \quad \boldsymbol{x}_E{}^T]^T, \boldsymbol{x}_P, \boldsymbol{x}_E \in R^n$. The subscripts $_P$ and $_E$ indicate, for any variable, its association with the pursuer or evader, respectively. Also, the subscript $_i$ will indicate applicability to either player. Here, we only consider separable differential games, meaning those in which state dynamics, costate dynamics, and the Hamiltonian can be divided into separate portions associated with the pursuer and the evader.

The state dynamics are:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = [\boldsymbol{f}_P(\boldsymbol{x}_P, \boldsymbol{u}_P)^T \quad \boldsymbol{f}_E(\boldsymbol{x}_E, \boldsymbol{u}_E)^T]^T$$

Again, we have $\boldsymbol{u} = [\boldsymbol{u}_P{}^T \quad \boldsymbol{u}_E{}^T]^T$, where $\boldsymbol{u}_P, \boldsymbol{u}_E \in R^c, 0 < c \leq n$, are the controls of the pursuer and evader, respectively.

There is a cost function $J \in R^+$, here defined as a terminal cost:

$$J = \Phi\big(\boldsymbol{x}(t_f), t_f\big)$$

The pursuer seeks to minimize J, and the evader seeks to maximize it. The game ends when the state reaches a terminal surface $\boldsymbol{\Psi} \in R^s, 0 < s \leq n$:

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

Along the lines of optimal control theory, we introduce costate variables $\boldsymbol{\Lambda} = [\boldsymbol{\Lambda}_P{}^T \quad \boldsymbol{\Lambda}_E{}^T]^T$ of the same dimension as the state $\boldsymbol{x}$. The co-state variables are the gradient of the optimal cost-to-go function, $\boldsymbol{\Lambda} = \frac{\partial J_{go}}{\partial \boldsymbol{x}}$, where $J_{go}(\boldsymbol{x}(t), t)$ is the remaining cost that will be incurred under optimal play proceeding from states $\boldsymbol{x}(t)$ at time t. (The function J_{go} is, of course, unknown.)

We form the Hamiltonian:

$$H = \sum_{i \in \{\boldsymbol{P}, \boldsymbol{E}\}} \boldsymbol{\Lambda}_i{}^T \boldsymbol{f}_i(\boldsymbol{x}_i, \boldsymbol{u}_i)$$

The optimal controls $\boldsymbol{u}_P^*$ and $\boldsymbol{u}_E^*$ are those that at each instant satisfy the minimax solution of the Hamiltonian:

$$(\boldsymbol{u}_P^*, \boldsymbol{u}_E^*) = \arg\left(\min_{\boldsymbol{u}_P} \max_{\boldsymbol{u}_E} H\right)$$

In this case, the minimax solution becomes:

$$\boldsymbol{u}_P^* = \arg\min_{\boldsymbol{u}_P} \boldsymbol{\Lambda}_P{}^T \boldsymbol{f}_P$$

$$\boldsymbol{u}_E^* = \arg\max_{\boldsymbol{u}_E} \boldsymbol{\Lambda}_E{}^T \boldsymbol{f}_E$$

From optimal control theory, the costate dynamics are:

$$\dot{\boldsymbol{\Lambda}} = -H_x = -\boldsymbol{f}_x{}^T \boldsymbol{\Lambda}$$

Again from optimal control theory, the constraints at the final time, known as the transversality conditions, are:

$$\Phi_x + (\boldsymbol{\Psi}^T \boldsymbol{v})_x - \boldsymbol{\Lambda}(t_f) = \boldsymbol{0}$$

$$\Phi_T + (\boldsymbol{\Psi}^T \boldsymbol{v})_t + H(t_f) = 0$$

We now give the specific formulation of the orbital PE game.

The orbital PE game formulation

Here, we show the system dynamics for pursuer and evader and the optimality criteria, and develop the optimal control problem leading to a two-point boundary value problem (TPBVP). Our formulation is similar to that of Pontani and Conway [8], although we use Cartesian coordinates, while they use orbital coordinates. To keep state variables at similar orders of magnitude, we use canonical units, where the Earth radius (6378.165 km) is one distance unit (DU), and the time unit (TU) is chosen as 806.8 s, making the gravitational parameter $\mu = 1 \frac{DU^3}{TU^2}$.

In the orbital PE game formulation, the state variables $\boldsymbol{x}_P, \boldsymbol{x}_E$ consist of three-dimensional position and velocity for each player, as shown below.

$$\boldsymbol{x}_i = [\boldsymbol{r}_i{}^T \ \boldsymbol{v}_i{}^T]^T , \qquad \boldsymbol{r}_i, \boldsymbol{v}_i \in R^3$$

The state vector dynamics are shown below. An inverse-r-squared gravity model is used, with gravity parameter $\mu = 1$ corresponding to full gravity with canonical units. Additionally, pursuer and evader both have thrust always on, with constant thrust-to-mass ratio T_{m_i}. The spacecraft's controls are the orientation of a unit vector $\hat{\boldsymbol{u}}_i \in R^3$ defined by the pair of angles $\{\alpha_i, \beta_i\}$. The spacecraft's thrust vector $\boldsymbol{u}_i$ is then $\hat{\boldsymbol{u}}_i$ multiplied by the scalar magnitude T_{m_i} associated with player i.

$$\boldsymbol{f}_i(\boldsymbol{x}_i) = \left[\boldsymbol{v}_i{}^T \ \left(-\frac{\mu}{(\boldsymbol{r}_i{}^T \boldsymbol{r}_i)}\right) \boldsymbol{r}_i + \boldsymbol{u}_i\right]^T$$

$$\boldsymbol{u}_i = \boldsymbol{T}_{m_i}\hat{\boldsymbol{u}}_i(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i) \;\; ; \;\; \hat{\boldsymbol{u}}_i{}^T\hat{\boldsymbol{u}}_i = 1$$

The cost is defined simply as the interception time:

$$J = t_f$$

The condition $T_{m_P} > T_{m_E}$ is sufficient to ensure interception in finite time, and is maintained in all cases studied here.

In this formulation, the pursuer (evader) controls minimizing (maximizing) the Hamiltonian are found by $\boldsymbol{u}_P^* = -\frac{\boldsymbol{\Lambda}_{v_P}}{|\boldsymbol{\Lambda}_{v_P}|}$ and $\boldsymbol{u}_E^* = \frac{\boldsymbol{\Lambda}_{v_E}}{|\boldsymbol{\Lambda}_{v_E}|}$. $\boldsymbol{\Lambda}_{v_i}$ denotes the costates corresponding to velocity of player i. The costates corresponding to position are denoted $\boldsymbol{\Lambda}_{r_i}$, and $\boldsymbol{\Lambda}_i = \left[\boldsymbol{\Lambda}_{r_i}{}^T \;\; \boldsymbol{\Lambda}_{v_i}{}^T\right]^T$.

The costate dynamics are:

$$\dot{\boldsymbol{\Lambda}}_{r_i} = -H_{r_i} = -G(\boldsymbol{r}_i)^T\boldsymbol{\Lambda}_{v_i}$$

$$\dot{\boldsymbol{\Lambda}}_{v_i} = -H_{v_i} = -\boldsymbol{\Lambda}_{r_i}$$

with

$$G(\boldsymbol{r}_i) = \frac{\partial}{\partial \boldsymbol{r}_i}\left[-\frac{\mu}{(\boldsymbol{r}_i^T\boldsymbol{r}_i)^{3/2}}\boldsymbol{r}_i\right] = -\frac{\mu}{(\boldsymbol{r}^T\boldsymbol{r})^{\frac{3}{2}}}I + \frac{3\mu}{(\boldsymbol{r}^T\boldsymbol{r})^{\frac{5}{2}}}(\boldsymbol{r}_i\boldsymbol{r}_i^T)$$

From the interception constraint and the transversality conditions, we obtain the following final-time constraints:

$$\boldsymbol{\Psi}(\boldsymbol{x}_P(t_f), \boldsymbol{x}_E(t_f), t_f) = \boldsymbol{r}_P(t_f) - \boldsymbol{r}_E(t_f) = \boldsymbol{0}$$

$$\boldsymbol{\Lambda}_{r_P}(t_f) + \boldsymbol{\Lambda}_{r_E}(t_f) = 0 \;\; ; \;\; \boldsymbol{\Lambda}_{v_P}(t_f) = 0 \;\; ; \;\; \boldsymbol{\Lambda}_{v_E}(t_f) = 0 \;\; ; \;\; H(t_f) + 1 = 0$$

The conditions $\boldsymbol{\Lambda}_{r_P}(t_f) + \boldsymbol{\Lambda}_{r_E}(t_f) = 0$ are condensed from the conditions $\boldsymbol{\Lambda}_{r_P}(t_f) = \boldsymbol{\nu}$ and $\boldsymbol{\Lambda}_{r_E}(t_f) = -\boldsymbol{\nu}$ stemming from the standard augmented cost function including the term $\boldsymbol{\Psi}^T\boldsymbol{\nu}$. The parameter $\boldsymbol{\nu}$ is not otherwise needed in the solution.

These developments result in a TPBVP where the unknowns are the initial values of the costates, $\boldsymbol{\Lambda}_i(t_0)$, as well as the final time, t_f, for a total of 13 unknowns. The 13 constraints at the final time are obtained from the interception constraint and the three final-time constraints on the costates (each of dimension 3), and the scalar Hamiltonian constraint. This TPBVP, which we refer to as TPBVP (*), is shown in Table 1.

Table 1 Formulation of the orbital PE problem TPBVP (*)

Given at time t_0:	Unknown at time t_0:		Constraints at t_f:	
$\boldsymbol{x}_P(t_0) = \boldsymbol{x}_{P_0}$	$\boldsymbol{\Lambda}_{P_0}$	(6)	$\boldsymbol{r}_P(t_f) - \boldsymbol{r}_E(t_f) = \boldsymbol{0}$	(3)
	$\boldsymbol{\Lambda}_{E_0}$	(6)	$\boldsymbol{\Lambda}_{r_P}(t_f) + \boldsymbol{\Lambda}_{r_E}(t_f) = \boldsymbol{0}$	(3)
$\boldsymbol{x}_E(t_0) = \boldsymbol{x}_{E_0}$	t_f	$\underline{(1)}$	$\boldsymbol{\Lambda}_{v_P}(t_f) = \boldsymbol{0}$	(3)
		(= 13)	$\boldsymbol{\Lambda}_{v_E}(t_f) = \boldsymbol{0}$	(3)
			$H(t_f) + 1 = 0$	$\underline{(1)}$
				(13)

We now discuss the formulation of sensitivity matrices for an orbital PE trajectory. The motivation is to enable local gradient search methods to find the values of $\boldsymbol{\Lambda}_P(t_0)$, $\boldsymbol{\Lambda}_E(t_0)$ and t_f that solve TPBVP (*).

Forming sensitivity matrices for the orbital PE problem

In this section, we discuss the computation of sensitivity matrices for the orbital PE problem. The goal is to compute a constraint sensitivity matrix, which is a matrix containing the sensitivities of the solution, namely the constraint equations of TPBVP (*), to a specified set of parameters. Since the constraints are functions of the states and other parameters, the first step is to compute a state sensitivity matrix. We describe next the steps to compute a state sensitivity matrix, and from there a constraint sensitivity matrix. When there is only one parameter of interest, we will refer to a state sensitivity vector and a constraint sensitivity vector.

As mentioned earlier, this problem is separable: pursuer and evader dynamics are independent. For this reason, we also compute separate state sensitivity matrices for the pursuer and evader, and use them both to compute a single constraint sensitivity matrix. We now discuss how to compute a state sensitivity matrix for either the pursuer or evader.

Suppose we have a set of parameters pertinent to the problem. They may include initial conditions; thrust and gravity parameters; or any other relevant parameters. Let there be a total of N parameters denoted by $a_j, j \in \{1,3,\dots,N\}$, and a vector formed from the parameters, $\boldsymbol{a} = [a_1,\dots,a_N]^T$. We wish to know the sensitivities to these parameters of the player (pursuer or evader) states at the final time, t_f. Since the costates are included in the terminal conditions, we will seek the sensitivities of the costates as well. This information would be captured in the matrix:

$$\frac{\partial \boldsymbol{Y}_i(t_f)}{\partial \boldsymbol{a}} = \left.\frac{\partial \boldsymbol{Y}_i(t)}{\partial \boldsymbol{a}}\right|_{t_f}$$

where $\boldsymbol{Y}_i(t)$ is a vector of player states and costates at any time t:

$$\boldsymbol{Y_i(t)} = [\boldsymbol{x}_i(t)^T \; \boldsymbol{\Lambda}_i(t)^T]^T$$

If there is only one parameter of interest (for example, gravity field strength), then $\boldsymbol{a}$ has only one element, and $\frac{\partial \boldsymbol{Y_i}}{\partial \boldsymbol{a}}$ is a column vector. We will refer to this quantity as a state sensitivity vector.

A technique exists for computing such a matrix for a system with general (linear or nonlinear) dynamics. We describe the technique as given by Schaub and Junkins [1], and as applied to an optimal control problem by Kim et al [3]. However, the present problem is final-time-free, so its constraint sensitivities matrix requires some information beyond that strictly contained in the state sensitivities matrix: specifically, time rates of change of states and costates at the final time, as will be shown.

The method is to solve the differential equation that the state sensitivity matrix satisfies. This differential equation is:

$$\frac{d}{dt}\left(\frac{\partial \boldsymbol{Y}_i}{\partial a}\right) = \frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{Y}_i}\frac{\partial \boldsymbol{Y}_i}{\partial a} + \frac{\partial \boldsymbol{F_i}}{\partial a}$$

$$\boldsymbol{F}_i(\boldsymbol{Y}_i) = \frac{d}{dt}(\boldsymbol{Y}_i) = \left[\dot{\boldsymbol{x}}_i{}^T \quad \dot{\boldsymbol{\Lambda}}_i{}^T\right]^T$$

where $\frac{\partial \boldsymbol{F_i}}{\partial \boldsymbol{Y}_i}$ is the Jacobian matrix of the state and costate dynamics with respect to the states and costates, also referred to as the dynamics sensitivities matrix. The computation of the elements of this matrix is shown in the Appendix. $\frac{\partial \boldsymbol{Y}_i(t_0)}{\partial \boldsymbol{a}}$ is dependent on the parameters $\boldsymbol{a}$.

The following steps are needed to compute a state sensitivity matrix for a given trajectory. The initial conditions of the state sensitivity matrix must be computed. If a parameter a_j does not affect the initial conditions, then $\frac{\partial \boldsymbol{Y}_i(t_0)}{\partial a_j} = \mathbf{0}$. In some cases, the parameters are the initial conditions, or are related to the initial conditions. For example, let $a_j = \boldsymbol{r}_{1_i}(t_0)$, the first coordinate of player i's initial state. Then, $\frac{\partial \boldsymbol{Y}_i(t_0)}{\partial a_j} =$ $[1 \quad \mathbf{0}_{1\times 11}]^T$.

This leads to consideration of the state transition matrix, $\frac{\partial \boldsymbol{Y}_i}{\partial \boldsymbol{Y}_{0_i}}$. This is the state sensitivity matrix when the parameter vector is the vector of initial conditions, $\boldsymbol{a} = \boldsymbol{Y}_i(t_0) = \boldsymbol{Y}_{0_i} = \left[\mathbf{x}_{i_0}{}^T \quad \boldsymbol{\Lambda}_{i_0}{}^T\right]^T$, the initial state and costate values. The differential equation satisfied by the state transition matrix is:

$$\frac{d}{dt}\left(\frac{\partial \boldsymbol{Y}_i}{\partial \boldsymbol{Y}_{0_i}}\right) = \frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{Y}_i}\frac{\partial \boldsymbol{Y}_i}{\partial \boldsymbol{Y}_{0_i}} \quad ; \quad \frac{\partial \boldsymbol{Y}_i(t_0)}{\partial \boldsymbol{Y}_{0_i}} = I \tag{1}$$

The procedure to compute a state sensitivity matrix for a given set of initial conditions ($\boldsymbol{Y}_{0_i}$) is to initialize and propagate the trajectory and, simultaneously, the differential equation (1). The dynamics sensitivity matrix must be computed at each time step.

What remains is to compute a single constraint sensitivity matrix using the state sensitivity matrices of both the pursuer and evader. The constraints are the constraint equations of TPBVP (*).

We consider a parameter vector $\boldsymbol{U}$, which consists of the unknown variables in TPBVP (*):

$$\boldsymbol{U} = \left[\boldsymbol{\Lambda}_P(t_0)^T \quad \boldsymbol{\Lambda}_E(t_0)^T \quad t_f\right]^T$$

We also define a vector function $\boldsymbol{S}(\boldsymbol{U})$ which consists of the values of the TPBVP (*) constraint equations at the final time, given the values of $\boldsymbol{U}$:

$$\boldsymbol{S}(\boldsymbol{U}) = \left[(\boldsymbol{r}_P - \boldsymbol{r}_E)^T \quad \left(\boldsymbol{\Lambda}_{r_P} + \boldsymbol{\Lambda}_{r_E}\right)^T \quad {\boldsymbol{\Lambda}_{v_P}}^T \quad {\boldsymbol{\Lambda}_{v_E}}^T \quad H + 1\right]\Big|_{t_f}^{T} \mathit{given}\,\boldsymbol{U}$$

We seek the matrix describing the sensitivities of the TPBVP (*) constraints to the unknowns. This matrix, denoted $\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}$, is written in full as:

$$\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} = \begin{bmatrix} \frac{\partial \boldsymbol{r}_P}{\partial \boldsymbol{\Lambda}_{P_0}} - \frac{\partial \boldsymbol{r}_E}{\partial \boldsymbol{\Lambda}_{P_0}} & \frac{\partial \boldsymbol{r}_P}{\partial \boldsymbol{\Lambda}_{E_0}} - \frac{\partial \boldsymbol{r}_E}{\partial \boldsymbol{\Lambda}_{E_0}} & \frac{\partial \boldsymbol{r}_P}{\partial t_f} - \frac{\partial \boldsymbol{r}_E}{\partial t_f} \\ \frac{\partial \boldsymbol{\Lambda}_{r_P}}{\partial \boldsymbol{\Lambda}_{P_0}} + \frac{\partial \boldsymbol{\Lambda}_{r_E}}{\partial \boldsymbol{\Lambda}_{P_0}} & \frac{\partial \boldsymbol{\Lambda}_{r_P}}{\partial \boldsymbol{\Lambda}_{E_0}} + \frac{\partial \boldsymbol{\Lambda}_{r_E}}{\partial \boldsymbol{\Lambda}_{E_0}} & \frac{\partial \boldsymbol{\Lambda}_{r_P}}{\partial t_f} + \frac{\partial \boldsymbol{\Lambda}_{r_E}}{\partial t_f} \\ \frac{\partial \boldsymbol{\Lambda}_{v_P}}{\partial \boldsymbol{\Lambda}_{P_0}} & \frac{\partial \boldsymbol{\Lambda}_{v_P}}{\partial \boldsymbol{\Lambda}_{E_0}} & \frac{\partial \boldsymbol{\Lambda}_{v_P}}{\partial t_f} \\ \frac{\partial \boldsymbol{\Lambda}_{v_E}}{\partial \boldsymbol{\Lambda}_{P_0}} & \frac{\partial \boldsymbol{\Lambda}_{v_E}}{\partial \boldsymbol{\Lambda}_{E_0}} & \frac{\partial \boldsymbol{\Lambda}_{v_E}}{\partial t_f} \\ \frac{\partial H}{\partial \boldsymbol{\Lambda}_{P_0}} & \frac{\partial H}{\partial \boldsymbol{\Lambda}_{E_0}} & \frac{\partial H}{\partial t_f} \end{bmatrix}\Bigg|_{t_f}$$

Except for the right-most column and bottom row, the elements of $\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}$ are computed from elements of the state transition matrices $\frac{\partial \boldsymbol{Y}_i}{\partial \boldsymbol{Y}_{0_i}}$ of each player. Below are formulas for the remaining elements.

$$\frac{\partial \boldsymbol{r}_i}{\partial t_f} = \dot{\boldsymbol{r}}_i|_{t_f} \ ; \ \frac{\partial \boldsymbol{\Lambda}_i}{\partial t_f} = \dot{\boldsymbol{\Lambda}}_i\Big|_{t_f}$$

$$\frac{\partial H}{\partial \boldsymbol{\Lambda}_{i_0}} = \frac{\partial \boldsymbol{\Lambda}_i}{\partial \boldsymbol{\Lambda}_{i_0}}^T \boldsymbol{f}_i + \boldsymbol{\Lambda}_i^T \left(\frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{Y}_i} \frac{\partial \boldsymbol{Y}_i}{\partial \boldsymbol{\Lambda}_{i_0}} \right) \Bigg|_{t_f}$$

$$\frac{\partial H}{\partial t_f} = \sum\nolimits_{i \in \{P,E\}} \left[\dot{\boldsymbol{\Lambda}}_i^T \boldsymbol{f}_i + \boldsymbol{\Lambda}_i^T \left(\frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{Y}_i} \boldsymbol{F}_i \right) \right] \Bigg|_{t_f}$$

Finally, consider a constraint sensitivity vector with regard to a single parameter a that is not an initial state. This will be relevant for the homotopy method. We require the constraint sensitivity vector with respect to a:

$$\frac{\partial \boldsymbol{S}}{\partial a} = \left[\frac{\partial \boldsymbol{r}_P}{\partial a} - \frac{\partial \boldsymbol{r}_E}{\partial a} \quad \frac{\partial \boldsymbol{\Lambda}_{r_P}}{\partial a} + \frac{\partial \boldsymbol{\Lambda}_{r_E}}{\partial a} \quad \frac{\partial \boldsymbol{\Lambda}_{v_P}}{\partial a} \quad \frac{\partial \boldsymbol{\Lambda}_{v_E}}{\partial a} \quad \frac{\partial H}{\partial a} \right] \Bigg|_{t_f}$$

All elements of this column vector except for the bottom element are computed using the state sensitivity vectors. The formula for the bottom element is:

$$\frac{\partial H}{\partial a} = \sum\nolimits_{i \in \{P,E\}} \left[\frac{\partial \boldsymbol{\Lambda}_i}{\partial a}^T \boldsymbol{f}_i + \boldsymbol{\Lambda}_i^T \left(\frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{Y}_i} \frac{\partial \boldsymbol{Y}_i}{\partial a} + \frac{\partial \boldsymbol{f}_i}{\partial a} \right) \right] \Bigg|_{t_f}$$

This concludes the discussion on forming state sensitivity matrices and constraint sensitivity matrices from parameters of interest, which may be initial conditions of the pursuer or evader, or other parameters. These computational tools will be relied upon in the remainder of this work.

We next present an original solution to the orbital PE problem enabled by these techniques.

CHAPTER III

A HOMOTOPY SOLUTION TO ORBITAL PURSUIT-EVASION

In this chapter, we present a homotopy method for solving the orbital PE problem. The method involves performing a homotopy in the gravity parameter of the problem. In later chapters, we will perform homotopies in other quantities, for purposes other than obtaining an initial solution.

The notation here is used only in this introductory section, although it is suggestive of the notation we will use in the orbital PE problem.

The idea of a homotopy method in solving a difficult mathematical problem is a three-step process. First, a simpler problem is found that can be related to the more difficult problem by parameter variation. For instance, given the root-finding problem $\boldsymbol{S}_1(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b} + \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0},\ \boldsymbol{S}_1, \boldsymbol{x} \in R^m$, the simpler problem might be $\boldsymbol{S}_0(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b} = \boldsymbol{0}, \boldsymbol{S}_0, \boldsymbol{x} \in R^m$.

Then, a homotopy function is written in the variables of interest and an additional variable, referred to as the homotopy parameter: $\boldsymbol{S}(\boldsymbol{x}, \varepsilon) = A\boldsymbol{x} + \boldsymbol{b} + \varepsilon\boldsymbol{g}(\boldsymbol{x}), \varepsilon \in [0,1]$.

Considered over the space $R^m \times [0,1]$, the solution set $\boldsymbol{S}(\boldsymbol{x}, \varepsilon) = \boldsymbol{0}$ is a one-dimensional curve, provided that the matrix $\begin{bmatrix} \frac{\partial S}{\partial x} & \frac{\partial S}{\partial \varepsilon} \end{bmatrix}$ is full rank and $\boldsymbol{S}_0(\boldsymbol{x}) = \boldsymbol{0}$ has a unique solution. The third step is to trace this curve from the easily obtained solution of $\boldsymbol{S}(\boldsymbol{x}, 0) = \boldsymbol{S}_0(\boldsymbol{x}) = \boldsymbol{0}$ to the more difficult solution of $\boldsymbol{S}(\boldsymbol{x}, 1) = \boldsymbol{S}_1(\boldsymbol{x}) = \boldsymbol{0}$.

Homotopy curve-tracing has been studied extensively. A detailed discussion of the method including development of sufficient conditions for a continuous, wellbehaved solution curve is given in the text by Allgower and Georg [18]. This reference also provides curve-tracing algorithms that we have generally followed in this work.

The “simple” problem used to begin our homotopy method is the solution to the “orbital” PE problem with the effects of gravity removed. This is equivalent to defining the homotopy parameter as $\varepsilon = \mu$, the gravitational field strength parameter, and then tracing the homotopy solution from $\varepsilon = 0$, with no gravity effects, to $\varepsilon = 1$, corresponding to the full gravity.

Without the effects of gravity, the problem dynamics are greatly simplified and allow an analytical solution. We note that Wong [14] in 1967 considered orbital PE with gravity acting in a constant direction. The methods of solving the problem for gravity in a constant direction and for zero gravity are very similar. In fact, Wong’s method might be applied to our process with some advantage, in the sense that a constant-gravity field may be a better choice of initial solution than zero gravity field; we have not yet pursued this idea.

The analytical solution to the zero-gravity case amounts to a cannonball trajectory problem. We place the mathematical details in the Appendix. Here, we describe qualitatively how the solution is found.

Zero-gravity solution

In the zero-gravity case, both pursuer and evader thrust directions are constant. Given initial starting conditions (position and velocity), optimal behavior is for pursuer and evader to select a thrust direction initially, and proceed in that direction until interception.

Pursuer and evader thrust directions are identical. Essentially, the evader runs "away" while the pursuer runs "toward", from start until interception. This simple relationship is due to the gravity-free dynamics.

These statements are confirmed by verifying that state and costate trajectories constructed based upon them satisfy the problem's optimality conditions.

We obtain the zero-gravity solution by first solving a one-player problem where the pursuer, from a given starting position and velocity, selects a constant thrust direction so as to arrive at the origin in minimum time. It happens that the two-player, pursuer/evader solution is a relative-motion case of the one-player solution. If the following function gives constant thrust angles to drive the pursuer to the origin in minimum time in the one-player game:

$$[\alpha, \beta] = \text{drive\_to\_origin\_min\_time}\left(\boldsymbol{r}_P(t_0), \boldsymbol{v}_P(t_0), T_{m_P}\right)$$

then the same function, supplied with relative motion arguments of the same form, will give the optimal control angles for pursuer and evader, which are identical:

$$[\alpha, \beta] = \text{drive\_to\_origin\_min\_time}\left(\boldsymbol{r}_P(t_0) - \boldsymbol{r}_E(t_0), \boldsymbol{v}_P(t_0) - \boldsymbol{v}_E(t_0), T_{m_P} - T_{m_E}\right)$$

From the controls, the pursuer and evader trajectories and the time of interception t_f are easily computed. The corresponding costates are also needed for the homotopy function. Obtaining the costates from the controls is the inverse of obtaining controls from costates.

We now describe the homotopy function relating the zero-gravity solution to the solution in gravity.

Homotopy function in gravity

For a given orbital PE problem specified by pursuer and evader initial states, we have a zero-gravity solution consisting of the costate initial values, and the final time t_f. Let $\boldsymbol{U}$ be the 13x1 vector containing these unknowns, defined in the previous chapter.

Now let $\boldsymbol{S}_0(\boldsymbol{U})$ be the values of the constraint functions of TPBVP (*) in the zero-gravity system, and let $\boldsymbol{S}_1(\boldsymbol{U})$ produce the same values in the full-gravity system:

$$\boldsymbol{S}_0(\boldsymbol{U}) = \left[(\boldsymbol{r}_P - \boldsymbol{r}_E)^T \quad \left(\boldsymbol{\Lambda}_{r_P} + \boldsymbol{\Lambda}_{r_E}\right)^T \quad {\boldsymbol{\Lambda}_{v_P}}^T \quad {\boldsymbol{\Lambda}_{v_E}}^T \quad H+1\right]\Big|_{t_f}^T \; given\, \mu = 0$$

$$\boldsymbol{S}_1(\boldsymbol{U}) = \left[(\boldsymbol{r}_P - \boldsymbol{r}_E)^T \quad \left(\boldsymbol{\Lambda}_{r_P} + \boldsymbol{\Lambda}_{r_E}\right)^T \quad {\boldsymbol{\Lambda}_{v_P}}^T \quad {\boldsymbol{\Lambda}_{v_E}}^T \quad H+1\right]\Big|_{t_f}^T \; given\, \mu = 1$$

Let the solution satisfying $\boldsymbol{S}_0(\boldsymbol{U}) = \boldsymbol{0}$ be $\boldsymbol{U}_0$, the zero-gravity solution which we can obtain analytically. Let the solution satisfying $\boldsymbol{S}_1(\boldsymbol{U}) = \boldsymbol{0}$ be $\boldsymbol{U}_1$, which is not known. Now let $\boldsymbol{S}(\boldsymbol{U}, \varepsilon)$ be a homotopy function in the vector of unknowns $\boldsymbol{U}$ and an additional homotopy parameter ε. Further, let $\boldsymbol{S}(\boldsymbol{U}, 0) = \boldsymbol{S}_0(\boldsymbol{U})$ and $\boldsymbol{S}(\boldsymbol{U}, 1) = \boldsymbol{S}_1(\boldsymbol{U})$. In other words, $\boldsymbol{S}$ is a homotopy function, where the homotopy in this case is in the gravity parameter μ:

$$\boldsymbol{S}(\boldsymbol{U}, \varepsilon) = \left[(\boldsymbol{r}_P - \boldsymbol{r}_E)^T \quad \left(\boldsymbol{\Lambda}_{r_P} + \boldsymbol{\Lambda}_{r_E}\right)^T \quad {\boldsymbol{\Lambda}_{v_P}}^T \quad {\boldsymbol{\Lambda}_{v_E}}^T \quad H+1\right]\Big|_{t_f}^T \; given\, \mu = \varepsilon$$

Our goal is now to trace the solution curve $\boldsymbol{S}(\boldsymbol{U}, 0) = \boldsymbol{0}$, beginning at the known point $\{\boldsymbol{U}_0, 0\}$ and ending at $\{\boldsymbol{U}_1, 1\}$.

Homotopy curve-tracing requires that the solution curve on the interval from start point to end point be smooth, continuous, and of finite length. The latter condition is violated in some instances in this problem, as will be shown. Given these conditions, tracing the curve is done by obtaining the tangent vector to the Jacobian matrix of $\boldsymbol{S}$. A sufficient condition for local smoothness of the solution curve is that this matrix be full rank.

We trace the curve with a predictor-corrector method, where the integration step is performed in the direction of the tangent vector. A Newtonian correction step is performed after each integration step, again using the Jacobian matrix (re-evaluated at the new location) to perform a linear correction.

The predictor step is in essence a numerical integration step along the solution curve. One way to parameterize the solution curve is in the parameter ε, with the curve being $\boldsymbol{U}(\varepsilon) = \boldsymbol{U} | \boldsymbol{S}(\boldsymbol{U}, \varepsilon) = \boldsymbol{0}$ over $\varepsilon \in [0,1]$. This parameterization will not be convenient, however, if the curve doubles back on itself over $\varepsilon \in [0,1]$. Therefore, we consider a parameterization in arc length s of the solution curve, $\boldsymbol{U}(s) = \boldsymbol{U} | \boldsymbol{S}(\boldsymbol{U}(s), \varepsilon(s)) = \boldsymbol{0}$ over $s \in [0, s_1]$ where $\boldsymbol{U}(0) = \boldsymbol{U}_0$ and $\boldsymbol{U}(s_1) = \boldsymbol{U}_1$. The solution then consists of solving the integral:

$$\begin{bmatrix} \boldsymbol{U}_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}_0 \\ 0 \end{bmatrix} + \int_0^{s_1} \hat{\boldsymbol{t}}(A)\, ds \tag{2}$$

$\hat{\boldsymbol{t}}(A)$ is the unit vector spanning the null space of the Jacobian matrix $A = \frac{\partial \boldsymbol{S}}{\partial [\boldsymbol{U} \quad \varepsilon]} = \begin{bmatrix} \frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} & \frac{\partial \boldsymbol{S}}{\partial \varepsilon} \end{bmatrix}$, which must be full rank. Note that the last element of $\hat{\boldsymbol{t}}$ corresponds to the increment of the homotopy parameter ε.

$\hat{\boldsymbol{t}}$ is specified up to sign, so a method for choosing the sign of $\hat{\boldsymbol{t}}$ is needed. We choose the sign initially such that $\hat{\boldsymbol{t}}$ points in the direction of increasing ε. Upon this choice, the sign of the determinant $\begin{vmatrix} A \\ \hat{\boldsymbol{t}}^T \end{vmatrix}$ is evaluated, and future selections of $\hat{\boldsymbol{t}}$ are made so as to hold this sign constant. (The sign of $\begin{vmatrix} A \\ \hat{\boldsymbol{t}}^T \end{vmatrix}$ will flip in the presence of a bifurcation point. This effect has rarely been observed in this problem, but means of accounting for bifurcation points are shown in [18].) One remaining caution is that the initial choice of $\hat{\boldsymbol{t}}$ in the direction of increasing ε is only a guess. If it is not successful, the curve can be followed in the other direction.

The numerical integration of equation (2) involves selecting an integration step size. Heuristics to adjust the step size are discussed in [18]. The main wrinkle is the use of a corrector step after each integration step. The corrector step leverages the fact that after an integration step is performed, violation of the optimality constraints can be checked at the new solution coordinates, and local corrections performed. This contrasts with standard numerical integration, where no check on the results is possible and so integration step size must be kept very small.

The corrector step is:

$$\begin{bmatrix} \boldsymbol{U}_{i+1} \\ \varepsilon_{i+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}_i \\ \varepsilon_i \end{bmatrix} - A^+ \boldsymbol{S}(\boldsymbol{U}_i, \varepsilon_i)$$

where A^+ is the Moore-Penrose pseudo-inverse of the Jacobian matrix A. A is re-computed each time it is applied in a prediction or correction step. The correction step is repeated until the constraint violations $\boldsymbol{S}(\boldsymbol{U}_i, \varepsilon_i)$ fall below a selected numerical error threshold.

Since the upper integration limit s_1 (the arc length required to traverse the solution) in equation (2) is not known, the integration is simply stopped when $\varepsilon = 1$ is reached.

Note that the value of ε can change during the correction step. This can result in the correction over-shooting ε, and so a more sophisticated control loop that allows backward integration is required.

The following is an alternate correction step which holds ε constant, which we have sometimes found preferable:

$$\boldsymbol{U}_{i+1} = \boldsymbol{U}_i - \left.\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}^{-1}\right|_{\boldsymbol{U}_i} \boldsymbol{S}(\boldsymbol{U}_i, \varepsilon)$$

The bottom line is that the combination of integration-and-linear-correction allows much larger integration steps to be taken, compared to numerical integration without correction.

To specify the gravity homotopy, we must identify $\frac{\partial \boldsymbol{Y}_i(t_0)}{\partial \varepsilon}$ and $\frac{\partial \boldsymbol{F}_i}{\partial \varepsilon}$:

$$\frac{\partial \boldsymbol{Y}_i(t_0)}{\partial \varepsilon} = \boldsymbol{0}$$

$$\frac{\partial \boldsymbol{F}_i}{\partial \varepsilon} = \left[\boldsymbol{0} \quad -\frac{\boldsymbol{r}_i^T}{(\boldsymbol{r}_i^T \boldsymbol{r}_i)^{-3/2}} \quad -\left[\frac{G(\boldsymbol{r}_i)^T}{\mu} \boldsymbol{\Lambda}_{v_i}\right]^T \quad \boldsymbol{0}\right]^T$$

Results

In this section, we present several results obtained from the homotopy method. For the sake of comparison, all of the results are based on orbital PE trajectories presented by Pontani and Conway in [8]. This study is hereafter referred to as PC09.

1. Cases 1, 2 and 3 from PC09. These cases are solved successfully by the homotopy method, except that in cases 1 and 2, the altitude constraint is violated. Thus, these solutions would still require a DCNLP step. The altitude constraint is inactive in Case 3, and so the solution provided by the homotopy method is sufficient.
2. PC09 case 4. The homotopy method fails in this case because it encounters a barrier surface *during the homotopy progression*. This barrier surface is associated with the orbital PE problem at partial gravity (the level reached at the point where the homotopy failed), and not the full-gravity problem. This topic will be examined in Chapter IV.

PC09 cases 1, 2, 3

Here, we present the solutions of the homotopy method for PC09 cases 1, 2 and 3. The homotopy method is capable of solving each of these cases. All of the cases are Low Earth Orbit (LEO) scenarios.

In PC09 cases 1 and 2, the altitude constraint is violated in the homotopy solution, and so the resemblance to the PC09 solution is not exact. In PC09 case 3, the altitude constraint is inactive, and so the homotopy solution and the PC09 solution satisfy the same optimality conditions. (The trajectories are in fact similar but not identical. This may be a case where multiple unique solutions exist; we have also seen this in cases where the homotopy function doubles back on itself.)

Case 1 is a tail-chase scenario including out-of-plane motion components. Initial conditions for this case are shown in Table 2. The homotopy solution trajectory and the PC09 trajectory are shown in Figure 3.

Figure 4 plots the unknown variables, the costates and the value of the final time, over the course of each homotopy. Figure 4 also shows the gravity-less trajectory computed as an input to the homotopy procedure.

Table 2 Initial conditions corresponding to PC09 case 1

	Initial position (x,y,z)			Initial velocity (x,y,z)			Thrust/mass
	DU			DU/TU			DU/TU^2 (g)
Pursuer	1.016	0.179	0	-0.148	0.840	0.492	0.1
Evader	0.822	0.608	0.136	-0.459	0.454	0.743	0.05

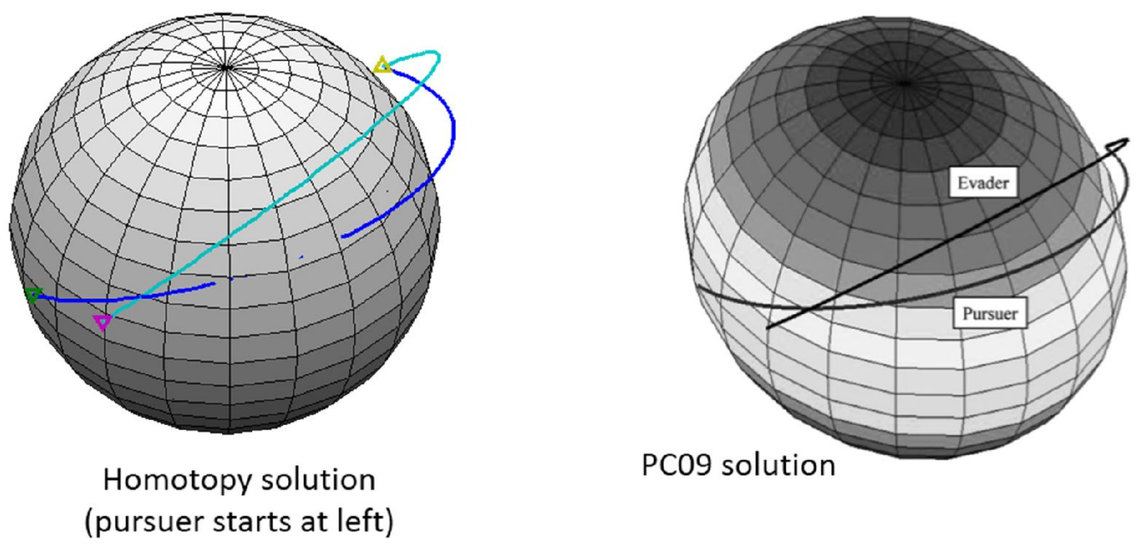


Figure 3 PC09 case 1: homotopy solution trajectory (left), and PC09 solution from prior study (right)

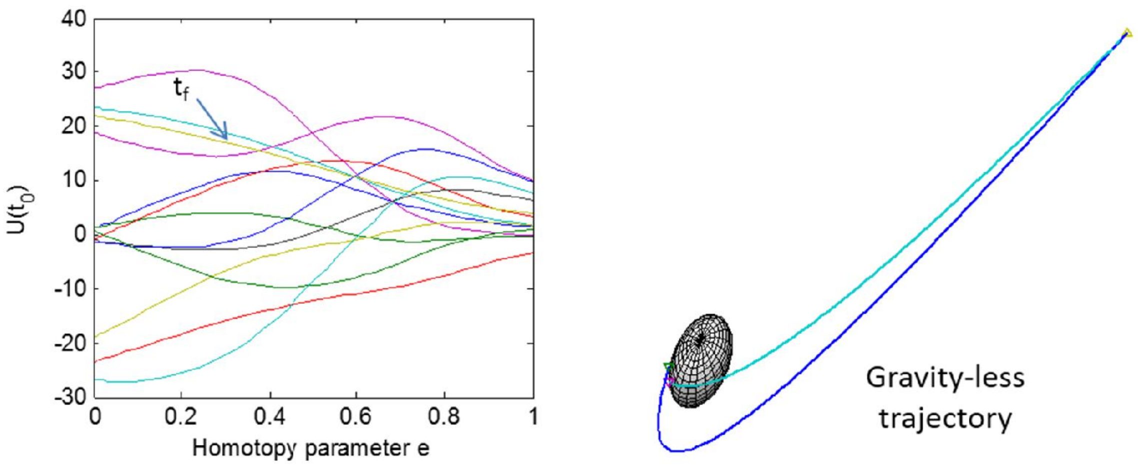


Figure 4 PC09 case 1: gravity homotopy initial costate values, and final time, vs. homotopy parameter ε (left), and gravity-less trajectory (right).

The next case shown in Case 2 from PC09. This case again involves a tail-chase with out-of-plane motion, this time with even less initial separation than in Case 1. Again, the altitude constraint is violated in the homotopy solution. Also, the interception in the PC09 trajectory occurs sooner than in the homotopy trajectory. This is surprising, since the altitude constraint is active for the pursuer and would be expected to increase interception time. One possible explanation is that the PC09 solution identified a second, shorter trajectory that satisfied optimality conditions, since such solutions need not be unique. Another cause might be inaccuracy in the PC09 solution, particularly the DCNLP post-processor which is known to provide no guarantee of optimality [28], and whose accuracy depends on factors such as the particular discretization used [29]. This question is not resolved at this time, but we discuss comparisons with the DCNLP solutions further in Chapter VII.

Initial conditions for Case 2 are shown in Table 3. The trajectory obtained by the homotopy, and the PC09 trajectory, are shown in Figure 5. The evolution of the unknown costates and final time is plotted in Figure 6, along with the gravity-less trajectory.

In this case, the homotopy solution passes very close to a barrier surface, but does not cross the barrier surface. This causes the costates to grow large, but then to settle again. This phenomenon will be discussed more in Chapter IV.

Table 3 Initial conditions corresponding to PC09 case 2

	Initial position (x,y,z)			Initial velocity (x,y,z)			Thrust/mass
	DU			DU/TU			DU/TU^2 (g)
Pursuer	1.016	0.179	0	-0.148	0.840	0.492	0.1
Evader	0.969	0.353	0	-0.292	0.801	0.492	0.05

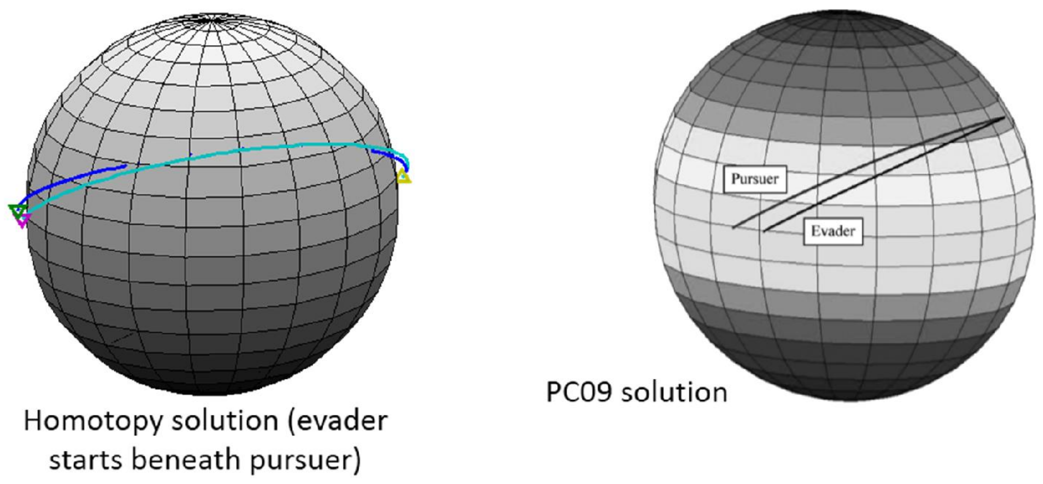
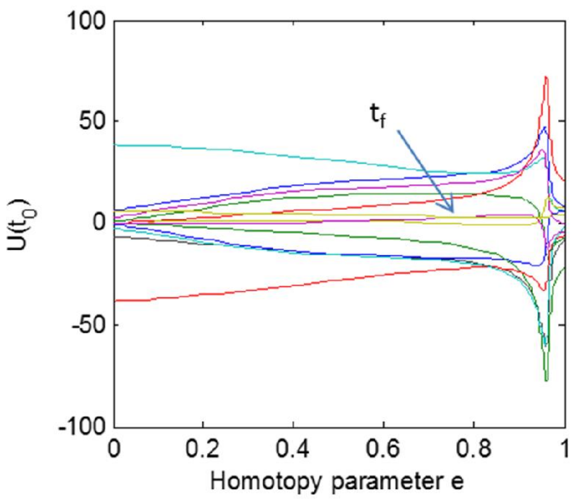


Figure 5 PC09 case 2: homotopy solution trajectory (left), and PC09 solution from prior study (right)



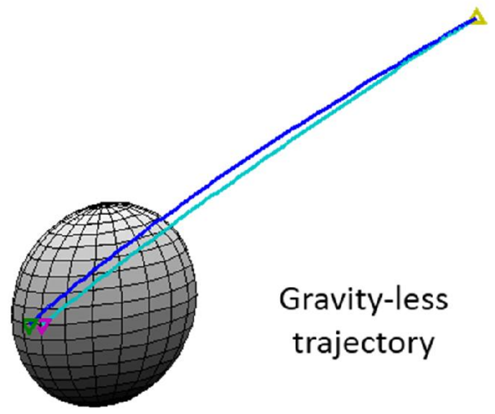


Figure 6 PC09 case 2: gravity homotopy initial costate values, and final time, vs. homotopy parameter ε (left), and gravity-less trajectory (right).

The next case shown is PC09 case 3. In this case, the altitude constraint is inactive. The homotopy and PC09 solutions appear similar, but in fact the terminal altitude is higher in the homotopy solution. It may be that both unique trajectory solutions satisfy the saddle-point optimality criteria, or that the DCNLP method does not converge with the same accuracy as the sensitivity method.

Table 4 shows the initial conditions for Case 3. Figure 7 shows the solution trajectories, and Figure 8 shows the unknown-value solution curves. The players' altitude histories for the homotopy solution and the PC09 study are shown in Figure 9.

Table 4 Initial conditions corresponding to PC09 case 3

	Initial position (x,y,z)			Initial velocity (x,y,z)			Thrust/mass
	DU			DU/TU			DU/TU^2 (g)
Pursuer	0.085	-0.953	-0.550	0.931	0.133	-0.017	0.1
Evader	-0.924	0.249	-0.550	0.293	0.894	-0.017	0.05

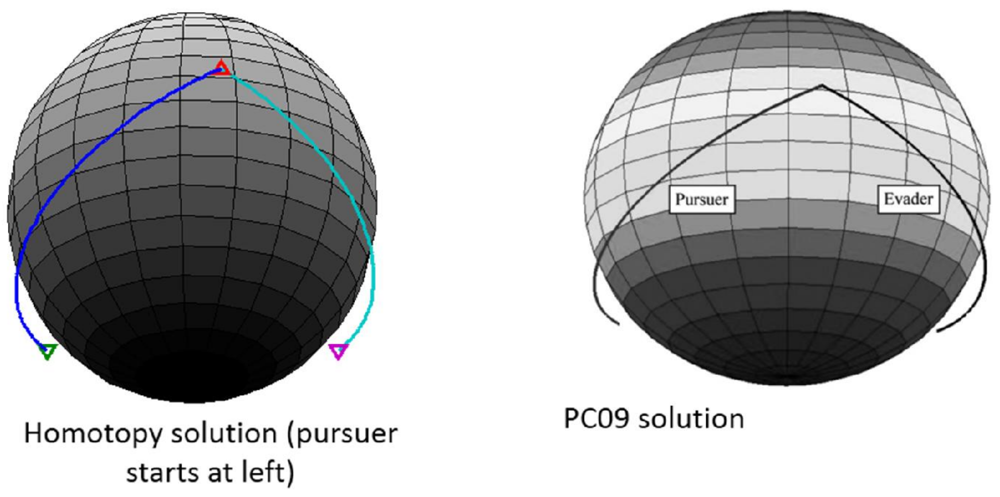
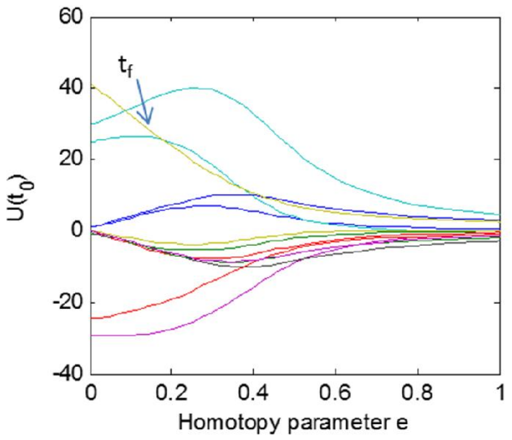


Figure 7 PC09 case 3: homotopy solution trajectory (left), and PC09 solution from prior study (right)



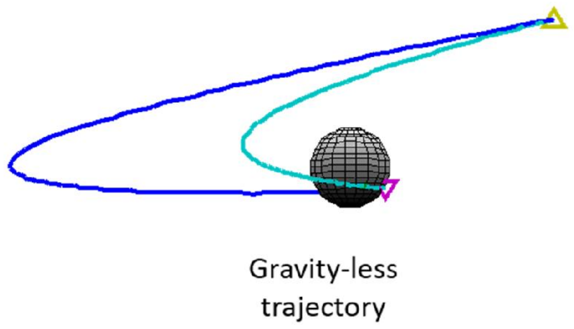
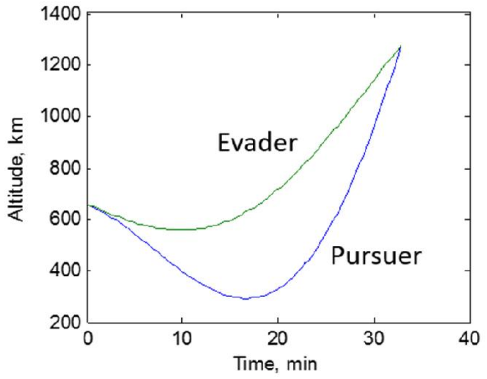


Figure 8 PC09 case 3: gravity homotopy initial costate values, and final time, vs. homotopy parameter ε (left), and gravity-less trajectory (right).



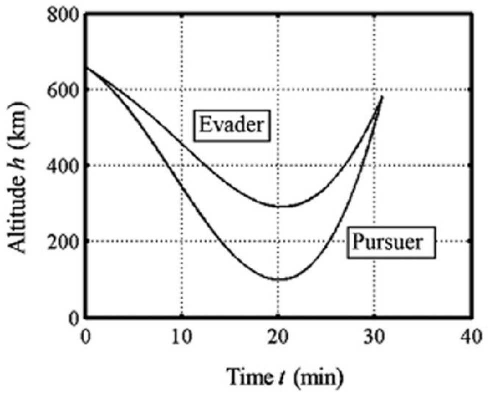


Figure 9 PC09 case 3: altitude histories for homotopy solution (left) and PC09 study (right)

Data related to computation time are shown in Table 5. The number of trajectory calls made by the homotopy solvers are shown, as are the CPU times for the solutions obtained using MATLAB's *cputime* function. The right-most column for the GA refers only to the task of running 50,000 trajectories, which is a subset of the total GA computation load. Two details are noted. First, a trajectory call for the GA does not compute sensitivities, while a trajectory call for the homotopy method does. Second, CPU times for the GA are directly comparable to those from PC09 case 3, because they were run with the same operating system (Windows XP). Cases 1 and 2 were recorded using Windows 7, for which CPU time per trajectory was slower than for Windows XP.

Case 3 obtains a greater than 200x improvement over a GA in both trajectory calls, and CPU time. Case 2 is the slowest case for the homotopy method. This is because the homotopy solution traversal passes near (though not through) a barrier trajectory, causing the curve tracing to proceed less efficiently.

Table 5 Computation results for PC09 cases 1-3, compared to genetic algorithm

	PC09 cases solved with homotopy method:			GA from [8]
	Case 1	Case 2	Case 3	
Trajectory calls	204	628	197	50,000
CPU time	152.3 s*	485.7 s*	42.2 s†	Trajectories only: 10,410.8 s†
* Desktop PC, 3.40 GHz Pentium processor running Windows 7 with 2 GB of RAM † Same hardware, Windows XP				

PC09 case 4

We examine the homotopy method operating on the scenario from PC09 case 4. The homotopy method is not successful in this case. We will discuss the failure mechanism here, and further in the next chapter.

As in case 3, PC09 case 4 involves the spacecraft approaching each other, rather than in a tailchase. The initial conditions are shown in Table 6.

Figure 10 shows the PC09 trajectory. Figure 11 shows the gravity-less trajectory, and the homotopy solution curve, which exhibits singularity-like behavior—the values abruptly explode toward infinity—while traversing from the zero-gravity solution to the full-gravity solution.

Table 6 Initial conditions corresponding to PC09 case 4

	Initial position (x,y,z)			Initial velocity (x,y,z)			Thrust/mass
	DU			DU/TU			DU/TU^2 (g)
Pursuer	-1.047	0.185	0	0.146	0.827	0.485	0.1
Evader	0.960	0.769	0.108	-0.519	0.586	0.443	0.05

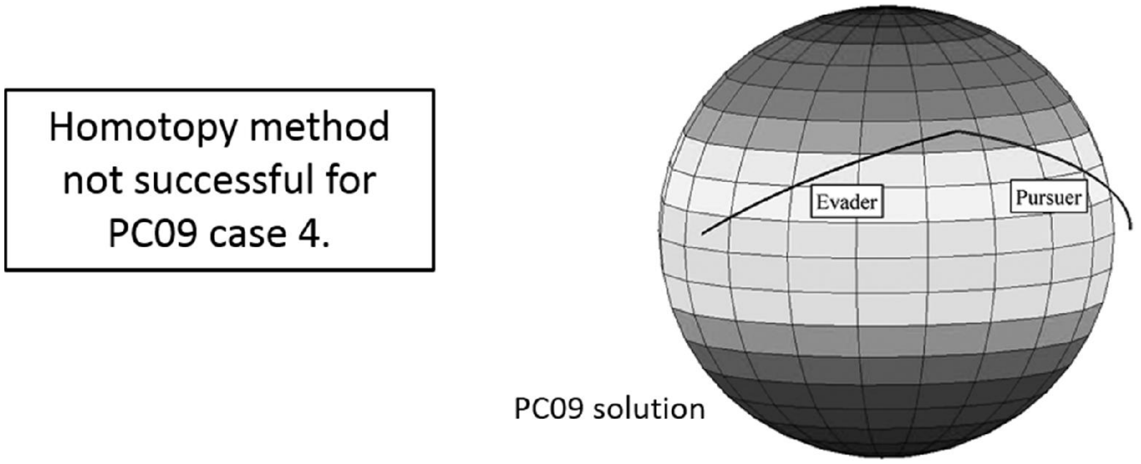


Figure 10 PC09 case 4: homotopy solution unsuccessful for this case. PC09 solution from prior study shown at right.

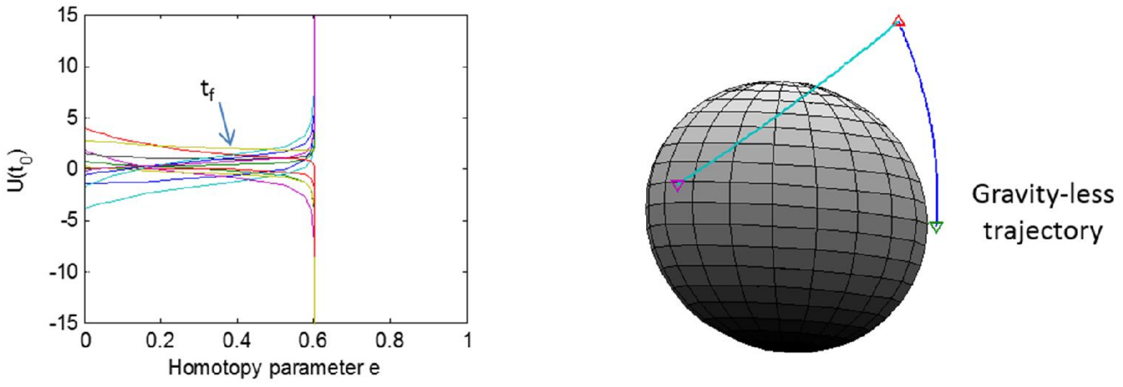


Figure 11 PC09 case 4: gravity homotopy initial costate values, and final time, vs. homotopy parameter ε, illustrating singularity behavior. (left) Gravity-less trajectory (right).

To preface our further examination of this situation in the next chapter, we make the following observation. For a given trajectory, multiplying all of the costate initial values by an identical positive scalar value does not change the trajectory. This is because the optimal controls are selected from the direction of the velocity costates, but not their magnitude; and the time rates of change of the velocity costates are proportional to the position costates, and vice versa. Hence, scaling them both results in identical costate trajectories apart from the scaling factor, and the scaling factor does not affect the controls.

Therefore, as the homotopy solution "blows up" as seen in PC09 case 4, the trajectory in this region is not changing at all.

However, the magnitude of the costates does affect the final-time Hamiltonian constraint. In fact, it is the enforcement of this constraint that is driving the costates to higher and higher values, without significantly changing the trajectory. In the next chapter, we will identify this cause, both from a numerical and theoretical standpoint.

CHAPTER IV

BARRIER SURFACES, TRACING THE BARRIER, AND FEEDBACK

As we saw in the previous chapter, in some cases the homotopy solution fails. This occurs when the costates appear to encounter a "singularity", in the sense that they grow infinitely large without changing the solution appreciably, and the homotopy cannot proceed. In this chapter, we will see that this is due to the solution encountering a barrier surface in the differential game associated with the system defined by the homotopy at that point (i.e. not the full-gravity system). It may be possible in theory to avoid encountering barrier surfaces by the proper choice of a homotopy function, but this is difficult in practice, and so is at present a limitation of the homotopy technique. However, our focus will turn to finding means of efficiently locating and tracing barrier surfaces, a task which has its own useful applications. We will also demonstrate a feedback control mechanism that, to operate robustly, must take into account barrier surfaces.

Barrier surfaces

The barrier surface in differential games was identified by Isaacs based on the observation that not all portions of the terminal surface are usable, meaning, can be reached in optimal play. The usable part of the terminal surface (called the "usable part") must involve instantaneous dynamics in which the pursuer is catching up to the evader, while the unusable part consists of points in which the evader is catching up to the pursuer: clearly, the latter would not be encountered in optimal play.

The mathematical condition corresponding to this requirement is found by constructing a vector $\hat{n}$ that is normal to the terminal surface. The magnitude of this vector is not important, but its sign is: it must point outward toward the region in which the game is played. [6] The condition for a usable terminal state is:

$$\boldsymbol{f} \cdot \boldsymbol{n} < 0 \tag{3}$$

This is a penetration condition: it states that the total (pursuer and evader) system dynamics must be in the direction of penetration of the terminal surface. This can be further understood for a separable game by separating the terminal surface into $\boldsymbol{n} = [\boldsymbol{n}_P{}^T \quad \boldsymbol{n}_E{}^T]^T$. Since the pursuer will seek to penetrate the terminal surface and the evader will seek to avoid it, equation (3) becomes $\boldsymbol{f}_P \cdot \boldsymbol{n}_P + \boldsymbol{f}_E \cdot \boldsymbol{n}_E < 0$ or $\boldsymbol{f}_P \cdot \boldsymbol{n}_P < -\boldsymbol{f}_E \cdot \boldsymbol{n}_E$, with the interpretation that on the usable part of the terminal surface, the pursuer's ability to penetrate the terminal surface must be greater than the evader's ability to avoid it.

Points on the terminal surface that satisfy the opposite inequality $\boldsymbol{f} \cdot \boldsymbol{n} > 0$ constitute the unusable part of the terminal surface. Points for which $\boldsymbol{f} \cdot \boldsymbol{n} = 0$ separate the usable and unusable parts, and the locus of these points on the terminal surface is called the boundary of the usable part (BUP).

Trajectories that terminate on the BUP are called barrier trajectories, and the locus of barrier trajectories is called the barrier surface. The barrier surface is not crossed in optimal play. In some games, the barrier surface may separate a region where interception is possible, from one where it is impossible. In orbital PE with $T_{m_P} > T_{m_E}$, interception is always possible given sufficient time; however, the barrier separates regions of continuous solution space. If we trace a set of solutions across a barrier surface, we will see that interception time jumps discontinuously when the barrier surface is crossed.

Equation (3) has a convenient property for games that have no terminal cost, or whose terminal cost is not a function of the states: for example, the present game. In such games, the transversality condition requires the costate vectors are normal to the terminal surface at the final time, and, being the gradient of the cost-to-go with respect to the states, point "in the direction in which the game is played," i.e. in the direction of the approach of the players' state vectors to the terminal surface. On an optimal solution trajectory, the costate vector at the final time is a valid choice for $\boldsymbol{n}$ in equation (3).

In other words, the vector $\boldsymbol{\Lambda} = [\boldsymbol{\Lambda}_P{}^T \quad \boldsymbol{\Lambda}_E{}^T]^T$ at the final time can be used in the identification of points on the BUP, and thus of barrier trajectories. This is true in general for games that involve a terminal surface, and a terminal cost that is not a function of the final states. A similar statement is made in [30].

In fact, in the orbital PE problem we have the Hamiltonian as $H = \boldsymbol{\Lambda}_P{}^T \boldsymbol{f}_P + \boldsymbol{\Lambda}_E{}^T \boldsymbol{f}_E$. At the final time, this is precisely the criterion for the usable/unusable part. Anderson and Grazier, in their orbital PE study, defined a "barrier Hamiltonian" in this way. [6] In this game, it coincides with the definition of the Hamiltonian in an optimal interception trajectory not on the barrier. Thus, on a barrier trajectory, we have $H = 0$; and not just at the final time but throughout, since $\dot{H} = 0$ as well. (The contradiction with the transversality constraint $H(t_f) = -1$ supports the notion that barrier trajectories should be seen as distinct from optimal interception trajectories.)

It is useful to consider the barrier Hamiltonian in light of the other final-time constraints: $\boldsymbol{r}_P = \boldsymbol{r}_E$, $\boldsymbol{\Lambda}_{r_P} = -\boldsymbol{\Lambda}_{r_E}$, $\boldsymbol{\Lambda}_{v_P} = \boldsymbol{\Lambda}_{v_E} = \boldsymbol{0}$. Given these constraints, the Hamiltonian at final time is simply:

$$H(t_f) = \sum\nolimits_{i \in \{P,E\}} \boldsymbol{\Lambda}_i{}^T \boldsymbol{f}_i \Big|_{t_f} = \boldsymbol{\Lambda}_{r_P}{}^T (\boldsymbol{v}_P - \boldsymbol{v}_E)\big|_{t_f} \triangleq \boldsymbol{\Lambda}_{r_P}{}^T \Delta \boldsymbol{v}\big|_{t_f}$$

i.e. the dot product of the relative velocity and the pursuer position costate vector at final time. If these vectors are normal to each other, we have $H = 0$, a barrier trajectory.

Now recall, as discussed in the last chapter, that the costates can be scaled to arbitrary magnitude without changing the trajectory. This suggests an explanation for the behavior of the homotopy function in the failed case: the solution was approaching a barrier trajectory, and the vectors $\Delta \boldsymbol{v}(t_f)$ and $\boldsymbol{\Lambda}_{r_P}(t_f)$ were becoming normal to each other. As they did so, the homotopy predictor-corrector maintained the optimality requirement $H(t_f) = \boldsymbol{\Lambda}_{r_P}{}^T \Delta \boldsymbol{v}\big|_{t_f} = -1$ by scaling the costates to increasingly large magnitudes. However, the solution could not progress beyond the barrier trajectory. Not only would the costates have to be scaled to infinity first, but on the other side of the barrier, the solution is in general not continuous.

The fact that both $\Delta \boldsymbol{v}(t_f)$ and $\boldsymbol{\Lambda}_{r_P}(t_f)$ can take arbitrary magnitude motivates the consideration of a normalized value:

$$B = \frac{\boldsymbol{\Lambda}_{r_P}{}^T \Delta \boldsymbol{v}}{|\boldsymbol{\Lambda}_{r_P}| \cdot |\Delta \boldsymbol{v}|}\Bigg|_{t_f}$$

B will be called the barrier indicator for this problem. In principle, it can take values from -1 to 1. But along optimal interception trajectories, its range will be [-1,0), with a value of zero on a barrier trajectory.

To emphasize these points, we provide an example in a simplified dynamical setting. Our goal will not be to derive the mathematics of the problem in detail, but simply to demonstrate the features of the barrier phenomenon.

Example with simplified dynamics

Consider the problem of a single "player" that, starting from some initial state, must be driven to the origin in minimum time, in a zero-gravity environment. In other words, this is the gravity-less orbital PE problem, with the evader fixed at the origin. The scenario is shown in Figure 12. As in orbital PE, the player has a constant-magnitude acceleration with controllable direction. The player starts with an initial velocity in the x- and y-directions, and the goal is to choose an angle of acceleration to send the player back to the origin in minimum time.

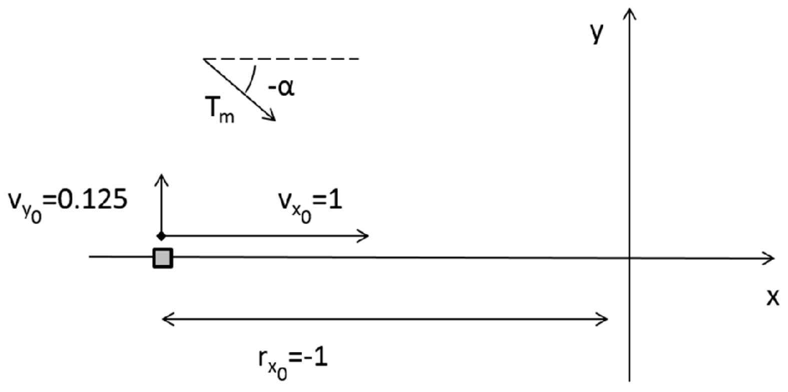


Figure 12 Simplified scenario: drive one player to origin in minimum time.

We examine how the optimal trajectory varies with the magnitude of the acceleration available to the player—in essence, we perform a homotopy in acceleration magnitude. Since this problem is exactly the one-player, gravity-less equivalent of the orbital PE problem, we formulate the problem (costates, Hamiltonian, final-time conditions) analogously. If we perform the homotopy, varying thrust downward from 0.3 to 0.1, we see exactly the same singularity behavior occurring as we saw in orbital PE. (Figure 13) Moreover, we see that the "pursuer" position costate vector becomes normal to the "relative velocity" (in this one-player game, the absolute velocity) at the singularity, as evidenced by the barrier indicator going to zero.

$$B \approx \begin{cases} -0.98, & (a) \\ -0.36, & (b) \\ -0.02 \; to \; 0^-, & (c) \end{cases}$$

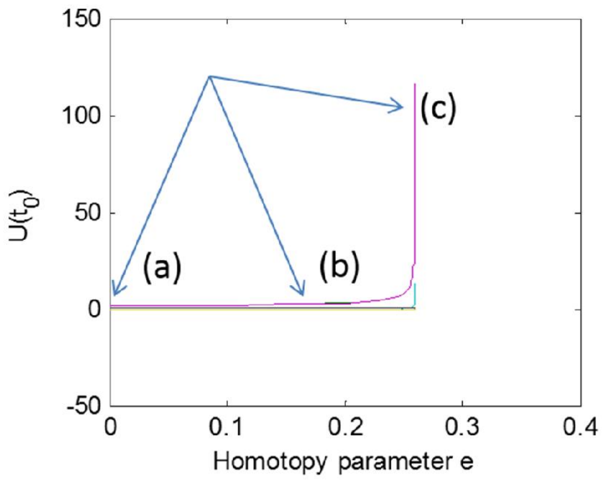


Figure 13 Homotopy in the simplified problem, exhibiting singularity behavior associated with the barrier indicator approaching zero

To explain this behavior, consider the progression of trajectory solutions shown in Figure 14. It is clear that given the player's initial velocity in the y-direction, some minimum acceleration magnitude is required to return the player to the x-axis before the y-axis is crossed. For acceleration magnitude less than this, the player must use an entirely different trajectory in which it shoots past the y-axis once, then is gradually halted and returned.

The controls, which correspond to the unknown costate values, are also discontinuous at the barrier. This can be seen in Figure 15, which plots $\alpha(t_0) = \alpha$, the only non-trivial control angle in this planar motion example, as thrust magnitude is varied.

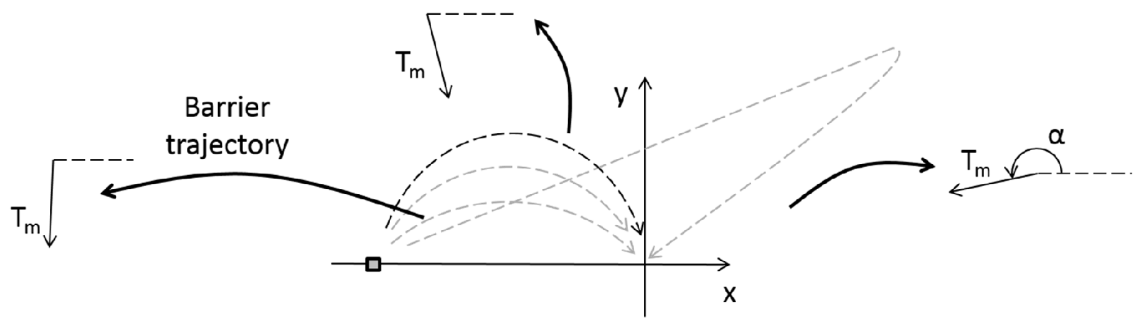


Figure 14 As thrust is reduced, trajectory solution crosses barrier, making a discontinuous jump

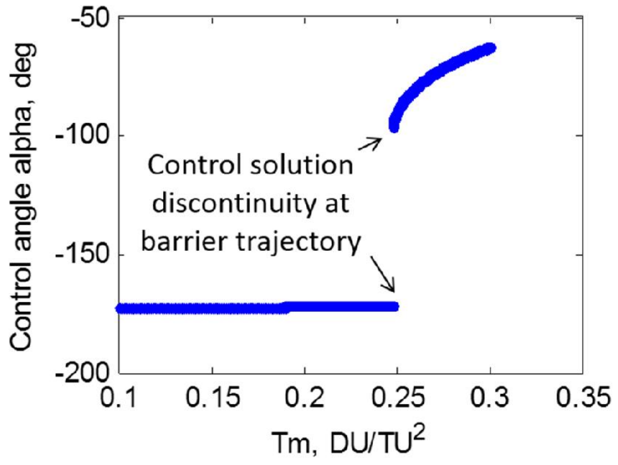


Figure 15 Control angle alpha for simplified example, exhibiting discontinuity at barrier trajectory

The significance of this example is summarized as follows:

1. The optimal trajectory solution is not always continuous in the problem parameters.
2. Discontinuities occur at a barrier trajectories, identified by the normalized indicator B going to zero at the final time.
3. Proximity to the barrier trajectory corresponds to the observed singularity behavior of the homotopy solution.

In light of these points, we now re-examine the solution curve of PC09 case 4.

PC09 case 4 barrier indicator

Figure 16 again shows the homotopy solution curve for PC09 case 4. Here, the values of the barrier indicator are marked at various points of the curve. In the singularity region, the barrier indicator goes toward 0^-, as expected.

$$B \approx \begin{cases} -0.78, & (a) \\ -0.89, & (b) \\ -0.05 \; to \; 0^{-}, & (c) \end{cases}$$

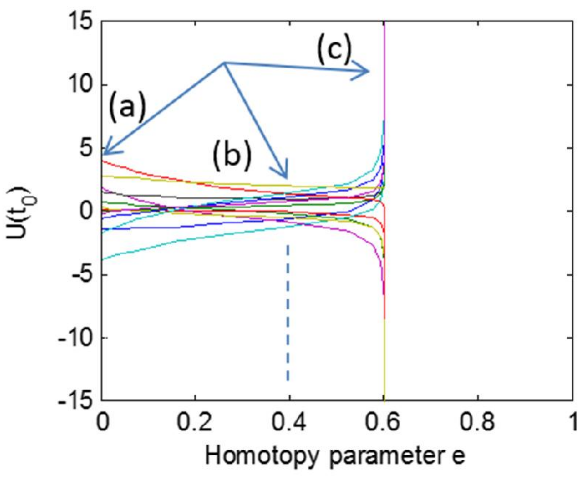


Figure 16 PC09 case 4 homotopy solution curve, with barrier indicator values marked

Discussion

Gradient-based traversal as employed in the sensitivity method cannot proceed across a barrier surface. We clarify that the barrier surface here means a barrier surface for the system at the point where the homotopy failed, not for the full-gravity system. For example, for the gravity homotopy shown in Figure 16, the homotopy failed at approximately $\mu = 0.6$, and so the barrier surface was associated with the given orbital PE problem at 60% of full gravity.

When a barrier surface is encountered, the problem must be solved anew for conditions immediately on the other side of the barrier, if possible. If it is in the initial homotopy solution that a barrier surface is encountered, the only recourse would be to use a different homotopy function in the solution, one whose initial solution starts out in the same continuous region of solution space as the desired solution. We will later discuss how during feedback control, barrier surfaces are often crossed due to sub-optimal play, and that often a new homotopy solution can be obtained in these cases. In general, however, a homotopy method that is robust to the barrier surface problem—i.e. that can reliably find a traversal path to the desired solution that does not cross a barrier surface—is difficult to obtain. Mathematical homotopies such as the Chow-Yorke method [22] may also be considered, but we have not yet examined this.

Alternatively, a genetic algorithm/collocation-type method could be used to obtain the solution on the other side of the barrier. Implementing one of these techniques to complement the homotopy method is viewed as a useful future avenue in this research. In the vicinity of a barrier surface, however, even techniques such as random search and collocation should be used with care. This is discussed further in Chapter VII.

Tracing the barrier surface

Having identified the means by which the barrier surface interacts with the homotopy solution method, we now consider whether a homotopy method can be used to find and trace barrier surfaces, or rather portions of it. In this problem, the combined pursuer and evader state space is of dimension $n = 12$, and the barrier surface is always of dimension $n - 1$. Therefore, we can only visually illustrate lower-dimensional submanifolds of the 11-dimensional barrier surface.

To motivate the question, consider the case shown in Figure 17. A pursuer must intercept an evader, starting from a fixed initial condition. For the evader, initial conditions within a certain range of values are of interest.

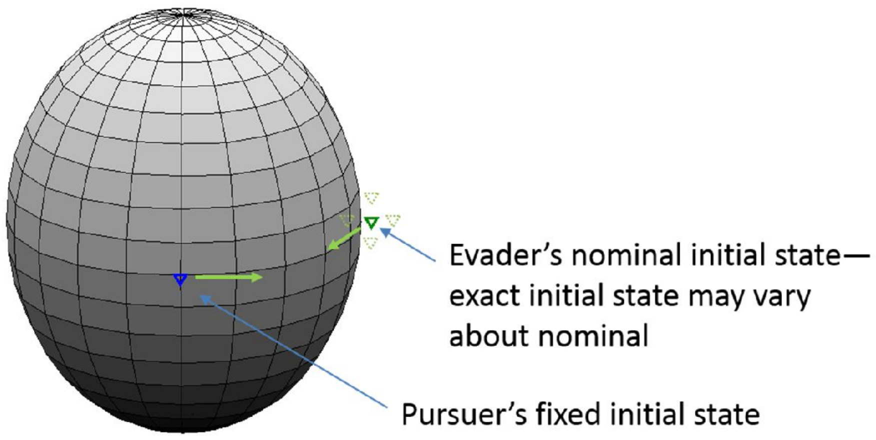

Evader's nominal initial state—
exact initial state may vary
about nominal
Pursuer's fixed initial state

Figure 17 Motivating scenario for problem of tracing barrier surface

This question can be considered with reference to barrier trajectories. The question is, given a solution to the interception at the nominal evader position, how far can the initial condition be perturbed in any parameter of interest—e.g. position and velocity components—before a barrier trajectory is reached? The assumption is that outside the barrier surface, a useful (first-pass) interception is not possible. This assumption is discussed further at the end of this section, but is expected to be valid for this scenario. The result is that if the region of possible evader initial conditions lies entirely within the space limited by the barrier trajectories, then all conditions can be handled by a single fixed pursuer. If this is not the case, then different plans, such as an array of pursuers sufficient to cover all the evader initial conditions, may need to be considered.

So the goal is to identify the locus of barrier trajectories that lie around the nominal evader initial condition. In this analysis, we have selected the evader nominal initial condition to lie on the x-z axis, and we will illustrate the method by examining variations in these two dimensions; we will not examine variations in the y-direction, or in the any of initial velocity components.

Starting conditions of pursuer and evader are shown in Table 7.

Table 7 Pursuer and (nominal) evader initial conditions for barrier sketching problem

	Initial position (x,y,z)			Initial velocity (x,y,z)			Thrust/mass
	DU			DU/TU			DU/TU^2 (g)
Pursuer	0	-1.0627	0	0.9700	0	0	0.1
Evader	1.0627	0	0	0	-0.9700	0	0.05

Before proceeding with the sensitivity technique for tracing the barrier surface, we describe an underlying technique that we will use extensively: a homotopy in the system states. It is a fairly straight-forward concept given the homotopy mechanism established in the previous chapter.

Homotopy in the system states

By the term "homotopy in the system states", we refer to a homotopy method in which the aspect of the problem being varied is the initial conditions of the pursuer and/or evader. We use this technique in a number of ways:

- To traverse a search space of possible starting conditions; applications include barrier trajectory identification. The sensitivity method is useful here because a large number of solutions are to be examined, which would take prohibitively long with a black-box solution method.
- To update a trajectory solution as the spacecraft trajectories evolve in time; applications include feedback and hybrid controller implementation (discussed later). Here, the solution is required in real time, which is likely possible with the sensitivity method and not with black-box methods.

The homotopy approach is straight-forward. We begin with a solution at a set of pursuer and evader initial conditions: $\boldsymbol{x}_P(t_0) = \boldsymbol{x}_{P_0}^{\boldsymbol{0}}$, $\boldsymbol{x}_E(t_0) = \boldsymbol{x}_{E_0}^{\boldsymbol{0}}$, and we are interested in the solution at a different set of initial conditions $\boldsymbol{x}_P(t_0) = \boldsymbol{x}_{P_0}^{\boldsymbol{1}}$, $\boldsymbol{x}_E(t_0) = \boldsymbol{x}_{E_0}^{\boldsymbol{1}}$. We let the homotopy parameter $\varepsilon = 0$ correspond to the solution with initial conditions $\{\boldsymbol{x}_{P_0}^{\boldsymbol{0}}, \boldsymbol{x}_{P_0}^{\boldsymbol{0}}\}$, and $\varepsilon = 1$ correspond to $\{\boldsymbol{x}_{P_0}^{\boldsymbol{1}}, \boldsymbol{x}_{P_0}^{\boldsymbol{1}}\}$. The sensitivity parameters unique to this homotopy are:

$$\frac{\partial \boldsymbol{Y}_i(t_0)}{\partial \varepsilon} = \begin{bmatrix} \boldsymbol{x}_{i_0}^{\boldsymbol{1}} - \boldsymbol{x}_{i_0}^{\boldsymbol{0}} \\ \boldsymbol{0} \end{bmatrix}$$

$$\frac{\partial \boldsymbol{F}_i}{\partial \varepsilon} = \boldsymbol{0}$$

Sensitivity method for tracing the barrier surface

We now define the problem as follows: given the evader starting condition, search the subset of state space consisting of variations in the x- and z-position dimensions until an enclosing manifold of barrier surface trajectories has been identified. (Figure 18) The restriction to variations in the x- and z-axes is made here for simplicity.

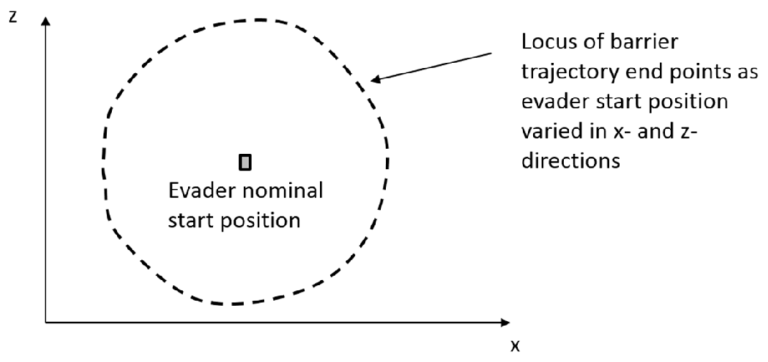

z
x
Locus of barrier trajectory end points as evader start position varied in x- and z-directions
Evader nominal start position

Figure 18 Illustration of barrier trajectories sought.

The task, then, is to methodically traverse the space depicted in Figure 18 until the location of the envelope of barrier trajectories is adequately defined. The means of traversing the space is a homotopy in the initial position of the evader.

This leaves the question of how best to identify a barrier trajectory. Clearly, we could identify a barrier trajectory as the point when the homotopy in evader initial state fails to make further progress. But this method is undesirable. The homotopy progression never actually reaches the barrier trajectory, and the predictor-corrector mechanism for optimal interception becomes exceedingly inefficient in the vicinity of the barrier.

Instead, it is better to search specifically for the barrier trajectory, at least when it is nearby. This is done by discarding the transversality constraint $H(t_f) + 1 = 0$, which is not satisfied on a barrier trajectory, and replacing it with a constraint on the barrier indicator: $B = 0$. All that remains is to develop sensitivity equations for the barrier indicator. Our method for doing this, which makes maximum use of existing sensitivity computations, is shown in the Appendix. When in a vicinity near to the barrier surface, this method allows a barrier trajectory to be identified quickly, and with arbitrary precision.

We offer a few remarks on selecting a homotopy parameter. In general, it may be desirable to vary, for example, the starting altitude of the evader, while maintaining a constraint on the other starting parameters such that a circular orbit initial condition is maintained. This would require more complex sensitivity equations with respect to the homotopy parameter, but this should not be too difficult a problem. Alternatively, a homotopy accomplishing the above could be broken down into small steps involving variation of only a single initial condition parameter at each step.

Here, for simplicity, we have not imposed any requirement such as a circular orbit initial condition, and so homotopies in the x- and z-position are all that is needed.

The barrier search procedure implemented here is as follows. From the evader initial condition, a vertical homotopy should be used until a point near the barrier surface in the vertical direction is reached. Here, this was done manually; it could also be identified by a threshold of the barrier indicator, e.g. -0.1. At this point, a control loop is entered in which three steps are performed. A horizontal homotopy is performed until the barrier indicator is at a threshold value, e.g. -0.05. Then a horizontal homotopy using the barrier constraint and sensitivities is used, which results in the identification of the barrier trajectory at that vertical location. This trajectory is stored, and starting at the end point of the horizontal interception homotopy (not the barrier homotopy), a small vertical homotopy of a pre-set magnitude, in the direction back toward the nominal condition, is performed. These three steps are repeated until the vertical coordinate of the nominal position is re-achieved. The result is the identification of a closely-spaced set of barrier trajectories in one quadrant of the desired search space. The analogous process is then repeated in the other three quadrants.

The idea is to use the interception homotopy where it is fast, yet not to switch to the barrier homotopy until close enough to the barrier to assure convergence. The barrier indicator values used may be tuned to this end, but the values mentioned above have been used to satisfaction.

This traversal sequence is illustrated in Figure 19.

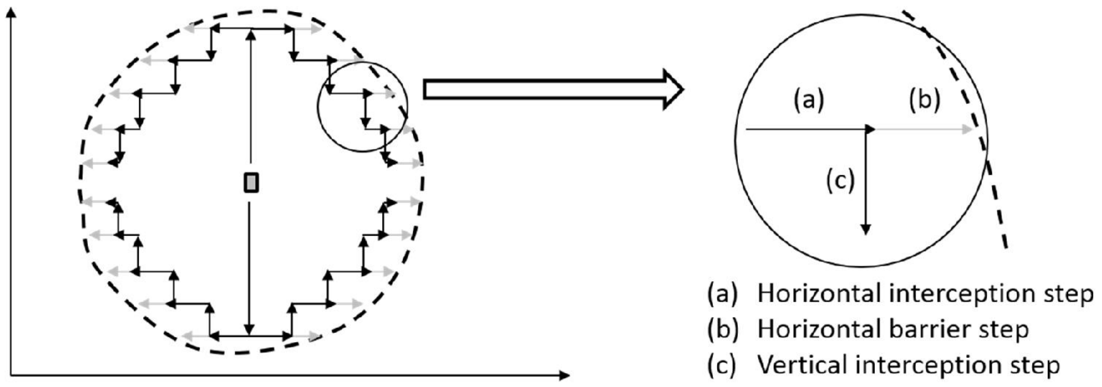

(a)
(b)
(c)
(a) Horizontal interception step
(b) Horizontal barrier step
(c) Vertical interception step

Figure 19 A simple traversal and barrier search process

Figure 20 presents the barrier trajectories obtained in this manner, for the pursuer initial conditions and the evader nominal initial conditions of Table 7.

Furthermore, the method of locating barrier surfaces described here was also implemented for a simple gravity-less dynamics case for which the barrier location could be ascertained analytically. This exercise provided a positive check on the validity of the barrier search method. The barrier location exercise for a simplified gravity-less case, and the analytical check, are shown in the Appendix.

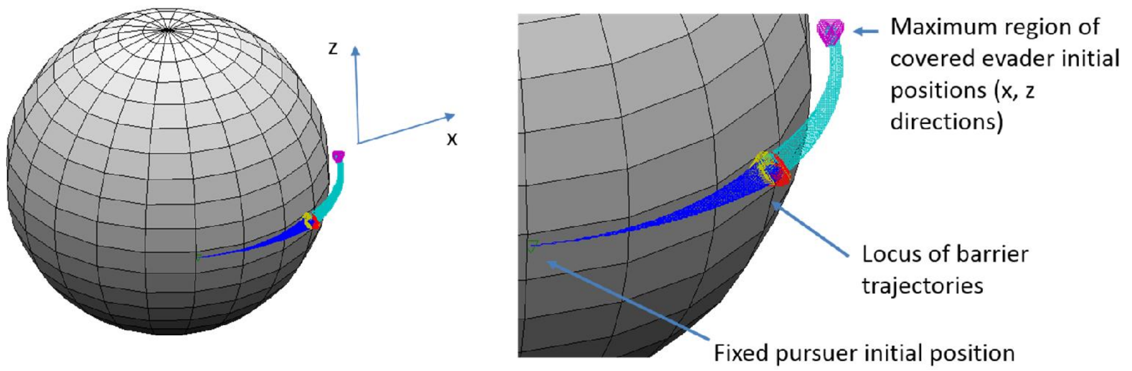

z
x
Maximum region of covered evader initial positions (x, z directions)
Locus of barrier trajectories
Fixed pursuer initial position

Figure 20 Barrier trajectories and associated evader initial conditions obtained in the vicinity of the nominal point

An assumption in the present analysis

In the present analysis, we assume that the first barrier surface encountered represents the surface beyond which an interception on first pass is impossible. (i.e. interception beyond that surface would involve both players meeting again on the opposite side of the planet; in practice, this would be a failed interception.) In fact, the only thing certain at a barrier trajectory is that the value of the game—the interception time—is discontinuous. Given the relatively simple head-on approach of this scenario, it seems likely that the first barrier surface is the only barrier surface for the first-pass interception. The verification, however, would be to use a suitable solution method, such as a random search or collocation technique, to solve for solutions outside the barrier surface, perhaps tailored to look for solutions on either a first pass or second pass.

If optimal first-pass solutions outside the first barrier surface were encountered, then the sensitivity method described here could be applied again to find the next barrier surface, until the limiting barrier surface is reached.

Feedback solution taking into account the barrier

Here, we present a feedback control mechanism. It is included in this chapter because when using optimal feedback control, barrier surfaces may be encountered as a result of sub-optimal play by the opposing player. This is discussed further at the end of this section.

In the feedback control method shown, at each time step a homotopy in the pursuer and evader states is performed, from their states at the previous time step to their present state. This allows one player, for example the pursuer, to obtain optimal controls at each time step given arbitrary behavior of the other player. An initial solution must be available to begin the method.

Here we show a feedback control implementation for the pursuer controls enabled by a homotopy method.

The initial conditions are those of PC09 case 3. However, the evader will use an arbitrary sub-optimal control, rather than the optimal control. The pursuer will react by identifying an updated optimal control based on the evader's state at each time step. The pursuer optimal control is obtained by performing a homotopy in the system states, from the control solution in the states of the previous time step, to the present states.

The conditions for this case are shown in Table 8. Figure 21 overlays the optimal trajectories computed by the pursuer at various points along the trajectory. The pursuer follows each new trajectory for only one time step, before computing another update.

As expected, interception time is reduced when the evader plays sub-optimally, from 2.443 TU (32 min, 51 sec) optimally (i.e. in the PC09 case 3 homotopy solution shown in Chapter III), to 2.312 TU (31 min, 5 sec) in the sub-optimal case shown here.

Table 8 Initial conditions for feedback implementation based on PC09 case 3

	Initial position, initial velocity, thrust/mass	Controls
Pursuer	PC09 case 3 conditions	Optimal
Evader	PC09 case 3 conditions	α=-130^0, β=110°

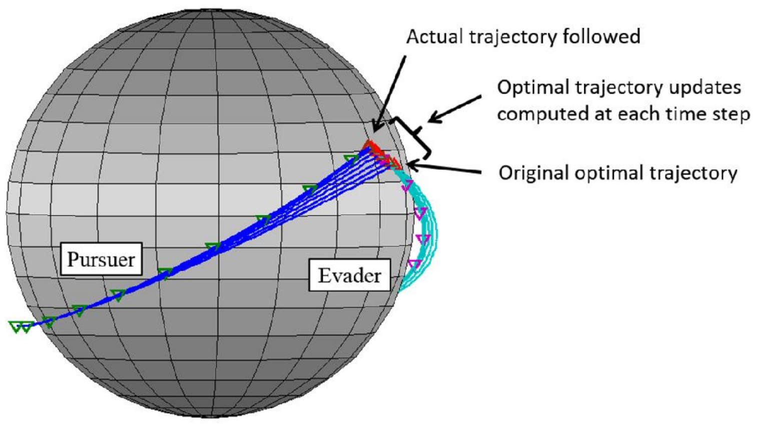

Actual trajectory followed
Optimal trajectory updates computed at each time step
Original optimal trajectory
Pursuer
Evader

Figure 21 Orbital PE feedback solutions.

Role of barrier surfaces in the feedback control method

During two-sided optimal control, a barrier surface is never crossed. However, in the optimal feedback control scenario, barrier surfaces can be crossed when the opposing player plays sub-optimally. This means that a method of obtaining updated controls using a homotopy from the previous time step must be supplemented with a means of finding a new solution from scratch, when a barrier surface is crossed. (The sensitivity numerical technique fails when crossing a barrier surface, regardless of the direction in which the surface is crossed.)

In the example shown, a barrier surface is in fact crossed a little less than three time steps before interception. In this case, it is possible to simply employ the initial-solution homotopy method again to find a new solution, although this method is not always successful. This method is employed here, and the feedback control continues to interception without incident.

We also note that the expected discontinuity in the game value, i.e. interception time, was not observed in association with this barrier crossing. This could be because the discontinuity in this case is small; or perhaps because on the other side of the barrier surface, the control solution continuous with the controls used before the barrier still satisfied local optimality conditions, and was the solution obtained. In other words, the pursuer may not have taken advantage of the evader's sub-optimal play to the fullest extent. Making the controller select the globally best choice from among multiple locally optimal solutions is a question for further study.

CHAPTER V

SENSITIVITIES FOR STATE-VARIABLE INEQUALITY CONSTRAINTS

In this chapter, we consider a solution to the problem encountered in Chapter III: orbital PE solutions generated by the homotopy method are not subject to the minimum altitude constraint. The altitude constraint is a state variable inequality constraint. In this chapter, we show a method for obtaining an optimal control solution subject to inequality constraints, from a solution to the unconstrained problem. The application of the method to the orbital PE problem is quite complex, and a full implementation has not been done here. Here, we show a partial implementation, which will suggest challenges in the approach, but also suggest that the full implementation may be of interest. Before proceeding to the orbital PE problem, however, we will demonstrate the method in full using the Breakwell problem, which is a common benchmark problem involving state-variable inequality constraints.

Our approach is, again, a sensitivity method. We use the fact that the unconstrained optimal trajectory is identical to a constrained optimal trajectory whose point of constraint happens to lie upon the unconstrained optimal path. Thus, we in fact have a constrained trajectory; however, the constraint has the wrong value. The problem is now susceptible, in principle, to a sensitivity technique by which the constraint is successively perturbed until the desired value is reached.

In the remainder of this chapter we cover the following topics. We first give a formulation for an optimal control problem subject to state variable inequality constraints. We then identify sensitivity matrices pertinent to this problem. We then apply the method to solve the Breakwell problem.

Next, we present a formulation of the orbital PE problem in which the minimum altitude constraint is rigorously incorporated. We then develop sensitivity matrices sufficient to handle the first two waypoint constraints of the problem, and we perform a partial homotopy solution which allows us certain useful insights into the homotopy solution to be followed in the fully constrained sensitivity method.

An optimal control problem subject to state variable inequality constraints

Here, we introduce the formulation of Bryson and Ho [24] for the optimal control problem subject to state variable inequality constraints. Consider the unconstrained optimal control problem described at the beginning of Chapter II. We now require that the state variables satisfy an inequality constraint:

$$C(\boldsymbol{x}) \leq 0$$

For simplicity, we do not consider constraints that depend explicitly on time. Further, let the constraint be second-order, meaning that the controls appear in the second derivative of the constraint with respect to time.

We enforce this inequality constraint in the solution by re-casting it as two tangency constraints, acting at time t_1^- when the constraint surface is reached (but before jump conditions are applied, as discussed next), and a control inequality constraint for the duration of contact with the constraint surface:

$$C\big(\boldsymbol{x}(t_1^-)\big) = 0$$

$$\dot{C}\big(\boldsymbol{x}(t_1^-), \boldsymbol{v}(t_1^-)\big) = 0$$

$$\ddot{C}(\boldsymbol{x}(t), \boldsymbol{v}(t), \boldsymbol{u}(t)) \leq 0 \text{ while } C = 0$$

Here, we assume a single activation of the constraint. The following method could, in principle, be applied multiple times to enforce multiple activation points of the constraint.

Note that C, $\dot{C}$ and $\ddot{C}$ are always evaluated with the arguments shown above. Since this is the case, we will simply write, e.g., C to denote $C(\boldsymbol{x}(t_1^-))$, etc.

Since it is not known when the state variables reach the constraint C, the variable t_1 is free. Therefore, the Hamiltonian is constant at t_1:

$$H({t_1}^-) = H({t_1}^+)$$

The costates at time t_1 will satisfy a jump condition:

$$\boldsymbol{\Lambda}({t_1}^-) = \boldsymbol{\Lambda}({t_1}^+) + \left(\begin{bmatrix} C \\ \dot{C} \end{bmatrix}^T \boldsymbol{\alpha}\right)_x = \boldsymbol{\Lambda}({t_1}^+) + \begin{bmatrix} C_x \\ \dot{C}_x \end{bmatrix}^T \boldsymbol{\alpha}$$

where $\boldsymbol{\alpha} = [\alpha_1 \quad \alpha_2]^T$ is unknown.

The control inequality constraint is enforced as follows. We append to the Hamiltonian the control constraint, multiplied by a Lagrange vector η (normally the symbol μ is used, but in this work that symbol is used for the gravity parameter):

$$H = \boldsymbol{\Lambda}^T \boldsymbol{f} + \eta \ddot{C}$$

(Note that we do not consider problems with an integral cost.) We find the control $\boldsymbol{u}$ as:

$$H_u = \boldsymbol{\Lambda}^T \boldsymbol{f}_u + \eta \ddot{C}_u = 0 \tag{4}$$

The Lagrange vector η takes its value as follows:

$$\eta \begin{cases} = 0, & \ddot{C} < 0 \\ \geq 0, & \ddot{C} = 0 \text{ and } C = 0 \end{cases} \tag{5}$$

On the constraint surface, when $\ddot{C} < 0$ would be violated for $\eta = 0$, η takes the required value such that, with controls selected from equation (4) given η, $\ddot{C} = 0$. In equation (5), the second condition is often only stated as $\ddot{C} = 0$. We add the requirement $C = 0$ to clarify that $\ddot{C}$ will not be constrained when the states do not lie on the constraint surface.

C is considered here to be a single scalar constraint. However, it can in general be a vector of several constraints. If the number of constraints of C is less than the degrees of freedom of the control, then C in general will constrain but not define the control, and so the control is selected from equation (4), after computing η in equation (5).

The final change from the unconstrained problem is in the propagation of the costates:

$$\dot{\boldsymbol{\Lambda}} = -H_x = -\boldsymbol{\Lambda}^T \boldsymbol{f}_x - \eta \ddot{C}_x \tag{6}$$

The result is again a TPBPV, with unknowns increased by three: the jump parameters $\boldsymbol{\alpha} = [\alpha_1 \quad \alpha_2]^T$ and the jump time t_1. The three additional constraints are the two tangency requirements $C = 0$, $\dot{C} = 0$, and $H(t_1{}^-) = H(t_1{}^+)$.

Development of a sensitivity technique

As mentioned earlier, references [25], [26], and [27] provide results similar to those we will present. Pesch [25] and Malanowski and Maurer [26] considered sensitivity techniques for inequality constrained problems, but without regard to a homotopy technique. Similarly, Hiskens and Pai [27] considered sensitivity methods applied to hybrid systems. However, none of these works provide the sensitivity equations we will require.

Our goal is to determine sensitivity equations for the additional parameters $\boldsymbol{\alpha} = [\alpha_1 \quad \alpha_2]^T$ and t_1 resulting from the state variable inequality constraint, as well as the three additional TPBVP constraints on the costates and Hamiltonian.

Specifically, we require the following terms:

$$\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} = \begin{bmatrix} \frac{\partial \boldsymbol{S}'}{\partial \boldsymbol{U}'} & \frac{\partial \boldsymbol{S}'}{\partial \boldsymbol{\alpha}} & \frac{\partial \boldsymbol{S}'}{\partial t_1} \\ \frac{\partial C}{\partial \boldsymbol{U}'} & \frac{\partial C}{\partial \boldsymbol{\alpha}} & \frac{\partial C}{\partial t_1} \\ \frac{\partial \dot{C}}{\partial \boldsymbol{U}'} & \frac{\partial \dot{C}}{\partial \boldsymbol{\alpha}} & \frac{\partial \dot{C}}{\partial t_1} \\ \frac{\partial \Delta H}{\partial \boldsymbol{U}'} & \frac{\partial \Delta H}{\partial \boldsymbol{\alpha}} & \frac{\partial \Delta H}{\partial t_1} \end{bmatrix}, \qquad \Delta H \triangleq H({t_1}^-) - H({t_1}^+)$$

Here, $\boldsymbol{S}'$ is the vector of optimality constraints of the unconstrained problem. Similarly, $\boldsymbol{U}'$ is a vector of unknowns of the unconstrained problem. $\boldsymbol{S}$ and $\boldsymbol{U}$ are the augmented constraints and unknowns for the constrained problem (since "constraints" can now refer to optimality constraints or state inequality constraints, we will rely upon context to distinguish the two):

$$\boldsymbol{S} = \left[\boldsymbol{S}'^T \quad C \quad \dot{C} \quad \Delta H_{t_1}\right]^T$$

$$\boldsymbol{U} = [\boldsymbol{U}'^T \quad \alpha_1 \quad \alpha_2 \quad t_1]^T$$

$\boldsymbol{S}$ is a shooting function employing the dynamics of the constrained problem, e.g. equations (4) and (6).

The methods already covered are used to find the sensitivies $\frac{\partial \boldsymbol{S'}}{\partial \boldsymbol{U'}}$ from the unconstrained problem, with one exception. Sensitivities to the parameter η must now also be computed. This is discussed in the next section.

Sensitivities of the new constraints C, $\dot{C}$ are problem-specific, yet straight-forward in principle. For the sensitivities of ΔH, the idea is to write $H(t_1{}^{-}) - H(t_1{}^{+})$ only in terms of states and costates at $t_1{}^{-}$ and jump parameters $\boldsymbol{\alpha}$, and take the difference of these two terms. Sensitivity equations can be formed from the result. The examples will provide illustration.

We examine the sensitivities to new unknowns $\boldsymbol{\alpha}$ and t_1, and treatment of the jump condition at t_1. This aspect of the problem is general.

Along the lines of previous notation, let a vector $\boldsymbol{Y}$ include the states and costates, $\boldsymbol{Y} = [\boldsymbol{x}^T \quad \boldsymbol{\Lambda}^T]^T$. The influence of $\boldsymbol{\alpha}$ and t_1 will be zero before t_1 is reached. With this in mind, our goal is to find:

- $\frac{\partial \boldsymbol{Y}(t_1{}^{+})}{\partial \boldsymbol{Y}_0}$, the state transition matrix after the jump given its value $\frac{\partial \boldsymbol{Y}(t_1{}^{-})}{\partial \boldsymbol{Y}_0}$ before the jump;
- $\frac{\partial \boldsymbol{Y}(t_1{}^{+})}{\partial \boldsymbol{\alpha}}$, the sensitivity of the states immediately after the jump to the jump parameters; and,
- $\left.\frac{\partial \boldsymbol{Y}(t)}{\partial t_1}\right|_{t_1{}^{+}}$, where $\frac{\partial \boldsymbol{Y}(t)}{\partial t_1}$ is the perturbation to the states at time $t > t_1$ resulting from a perturbation to the switch time t_1. This notation will be explained shortly.

From these quantities, we can compute sensitivities of C, $\dot{C}$ and ΔH, the constraints at time t_1. These quantities will subsequently be propagated from ${t_1}^+$ to t_f according to equation (1), i.e.:

$$\frac{d}{dt}\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{a}} = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{Y}}\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{a}} \quad , \quad \boldsymbol{a} \in \{\boldsymbol{Y}_0, \boldsymbol{\alpha}, t_1\} \tag{7}$$

where $\boldsymbol{F}$ is a vector of time derivatives of $\boldsymbol{Y}$, and $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{Y}}$ is found with normal methods. The dynamics sensitivity matrix (or vector) $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{a}} = \boldsymbol{0}$ for all quantities $\boldsymbol{a}$ above.

With this, we can compute sensitivities of the final-time constraints.

To compute $\frac{\partial \boldsymbol{Y}({t_1}^+)}{\partial \boldsymbol{\alpha}}$ and $\frac{\partial \boldsymbol{Y}({t_1}^+)}{\partial \boldsymbol{Y}_0}$, we write $\boldsymbol{Y}({t_1}^+)$ as:

$$\boldsymbol{Y}({t_1}^+) = \boldsymbol{Y}({t_1}^-) + \boldsymbol{Y}_{jump} = \boldsymbol{Y}({t_1}^-) + \begin{bmatrix} \boldsymbol{0} \\ -\begin{bmatrix} C_x \\ \dot{C}_x \end{bmatrix}^T \boldsymbol{\alpha} \end{bmatrix}$$

Then the computation of $\frac{\partial \boldsymbol{Y}({t_1}^+)}{\partial \boldsymbol{\alpha}}$ is straight-forward:

$$\frac{\partial \boldsymbol{Y}({t_1}^+)}{\partial \boldsymbol{\alpha}} = \frac{\partial}{\partial \boldsymbol{\alpha}}\left[\boldsymbol{Y}({t_1}^-) + \boldsymbol{Y}_{jump}\right] = \frac{\partial}{\partial \boldsymbol{\alpha}}\begin{bmatrix} \boldsymbol{0} \\ -\begin{bmatrix} C_x \\ \dot{C}_x \end{bmatrix}^T \boldsymbol{\alpha} \end{bmatrix} = -\begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ C_x^T & \dot{C}_x^T \end{bmatrix}$$

Similarly,

$$\frac{\partial \boldsymbol{Y}({t_1}^+)}{\partial \boldsymbol{Y}_0} = \frac{\partial}{\partial \boldsymbol{Y}_0}\left[\boldsymbol{Y}({t_1}^-) + \boldsymbol{Y}_{jump}\right] = \frac{\partial \boldsymbol{Y}({t_1}^-)}{\partial \boldsymbol{Y}_0} + \frac{\partial}{\partial \boldsymbol{Y}({t_1}^-)}\begin{pmatrix} \boldsymbol{0} \\ -\begin{bmatrix} C_x \\ \dot{C}_x \end{bmatrix}^T \boldsymbol{\alpha} \end{pmatrix}\frac{\partial \boldsymbol{Y}({t_1}^-)}{\partial \boldsymbol{Y}_0}$$

or

$$\frac{\partial \boldsymbol{Y}({t_1}^+)}{\partial \boldsymbol{Y}_0} = \left[I + \frac{\partial}{\partial \boldsymbol{Y}} \begin{pmatrix} \boldsymbol{0} \\ -\begin{bmatrix} C_x \\ \dot{C}_x \end{bmatrix}^T \boldsymbol{\alpha} \end{pmatrix} \right] \frac{\partial \boldsymbol{Y}({t_1}^-)}{\partial \boldsymbol{Y}_0} \tag{8}$$

Now we consider $\frac{\partial \boldsymbol{Y}(t)}{\partial t_1}$, which is propagated from the initial condition at t_1 of $\left.\frac{\partial \boldsymbol{Y}(t)}{\partial t_1}\right|_{{t_1}^+}$. We do *not* wish to denote $\left.\frac{\partial \boldsymbol{Y}(t)}{\partial t_1}\right|_{{t_1}^+} = \frac{\partial \boldsymbol{Y}({t_1}^+)}{\partial t_1} = \frac{\partial}{\partial t_1}\left[\boldsymbol{Y}({t_1}^-) + \boldsymbol{Y}_{jump}\right]$. The quantity we ultimately seek, $\frac{\partial \boldsymbol{Y}(t_f)}{\partial t_1}$, represents the state perturbations at the final time resulting from a change to t_1, the time when the altered dynamics begin. A better definition is:

$$\left.\frac{\partial \boldsymbol{Y}(t)}{\partial t_1}\right|_{{t_1}^+} = \lim_{t \to {t_1}^+} \frac{\partial \boldsymbol{Y}(t)}{\partial t_1}$$

With this, we denote the initial condition at t_1^+ of $\frac{\partial \boldsymbol{Y}(t)}{\partial t_1}$, the state perturbation at time $t > t_1$ due to a change in the time t_1 of the onset of altered dynamics.

To determine this initial value, we consider the variations involved. We first consider the states, then the costates. The change in the states due to incremental change in t_1 is shown in Figure 22.

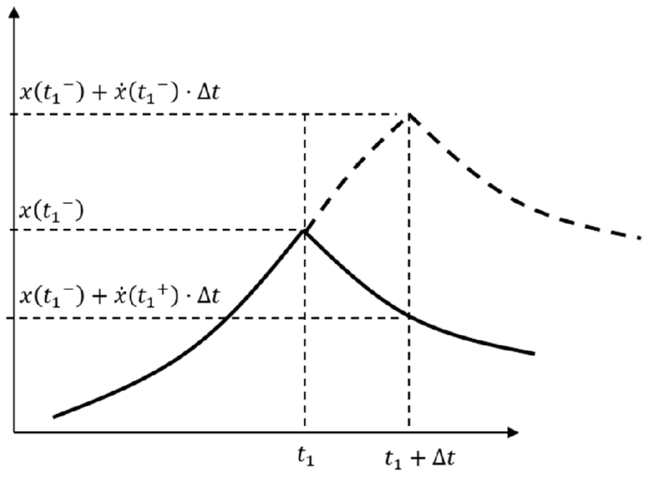
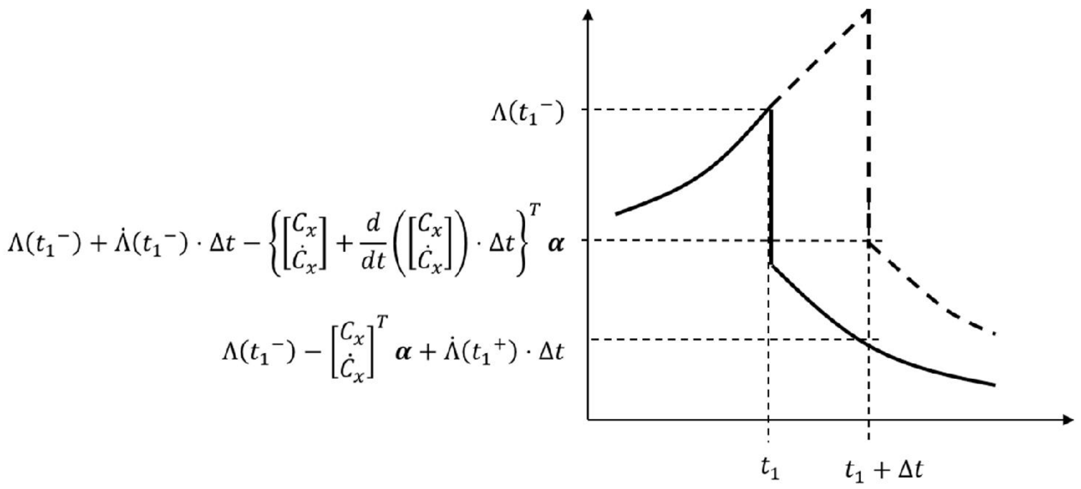

$x(t_1^-) + \dot{x}(t_1^-) \cdot \Delta t$
$x(t_1^-)$
$x(t_1^-) + \dot{x}(t_1^+) \cdot \Delta t$
t_1
$t_1 + \Delta t$

Figure 22 Illustration of change in states due to incremental change in t1

We have:

$$\frac{\partial \boldsymbol{x}(t_1^+)}{\partial t_1} = \lim_{\Delta t \to 0} \frac{\boldsymbol{x}(t_1^-) + \dot{\boldsymbol{x}}(t_1^-) \cdot \Delta t - [\boldsymbol{x}(t_1^-) + \dot{\boldsymbol{x}}(t_1^+) \cdot \Delta t]}{\Delta t} = \dot{\boldsymbol{x}}(t_1^-) - \dot{\boldsymbol{x}}(t_1^+)$$

For the costates, we must also account for the jump condition at t_1. In general, the magnitude of the jump will change as t_1 changes. This effect is illustrated for the costates in Figure 23.



$\Lambda(t_1{}^-)$
$\Lambda(t_1{}^-) + \dot{\Lambda}(t_1{}^-)\cdot\Delta t - \left\{\begin{bmatrix}C_x\\ \dot{C}_x\end{bmatrix} + \frac{d}{dt}\left(\begin{bmatrix}C_x\\ \dot{C}_x\end{bmatrix}\right)\cdot\Delta t\right\}^T \boldsymbol{\alpha}$
$\Lambda(t_1{}^-) - \begin{bmatrix}C_x\\ \dot{C}_x\end{bmatrix}^T \boldsymbol{\alpha} + \dot{\Lambda}(t_1{}^+)\cdot\Delta t$
t_1
$t_1 + \Delta t$

Figure 23 Illustration of change in costates due to incremental change in t1

By taking the limit in the same manner as for the states, we obtain:

$$\frac{\partial \boldsymbol{\Lambda}(t_1{}^+)}{\partial t_1} = \dot{\boldsymbol{\Lambda}}(t_1{}^-) - \frac{d}{dt}\left(\begin{bmatrix}C_x\\ \dot{C}_x\end{bmatrix}\right)^T \boldsymbol{\alpha} - \dot{\boldsymbol{\Lambda}}(t_1{}^+)$$

where the time rate of change of the costate jump magnitudes also appears.

In addition to the normal processes for computing sensitivities already illustrated, these are the additional quantities required to formulate sensitivity matrices for the optimal control solution subject to inequality constraints. However, one further complication remains, regarding the Lagrange vector η.

Sensitivities to the Lagrange vector η

For any parameter that is a function of η, its sensitivities must be evaluated conditionally: if $\eta > 0$, all sensitivities due to η must be included. η is a function of other parameters—values of the states, costates, etc—and so sensitivities due to η are included using the chain rule of differentiation. However, if $\eta = 0$, then sensitivities due to η should not be included. In general, η appears in the Hamiltonian, the state dynamics and the costate dynamics, so any sensitivity computation involving any of these quantities must be computed conditionally based on η.

Application to the Breakwell problem

The Breakwell problem is a simple example of an optimal control problem subject to state variable inequality constraints. It is frequently used as a demonstration problem in this context. For example, references [28] and [31] solve the Breakwell problem to demonstrate, respectively, direct collocation and Legendre pseudospectral approximation. The problem has a convenient analytical solution, which is presented in Bryson and Ho. [24]

The problem is to minimize the control energy while satisfying state variable initial and terminal conditions, subject to an inequality constraint. The states are scalar position and velocity. Specifically:

$$\dot{x} = v \quad ; \quad \dot{v} = u$$

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

$$\mathrm{x}(0) = 0 \quad ; \quad \mathrm{v}(0) = 1 \quad ; \quad \mathrm{x}(1) = 0 \quad ; \quad \mathrm{v}(1) = -1$$

$$C(x) = x - h \leq 0$$

where h is a parameter of the problem.

The method we propose here requires, in general, that we be given a solution to the optimal control problem with the state constraint removed. In this regard, the Breakwell problem without the state constraint is a straightforward Linear-Quadratic

Regulator (LQR) problem, for which solutions are known. But the analytical solution to the Breakwell problem is known in even simpler form. It is known that for $h \geq 0.25$, the constraint is inactive: the unconstrained solution and the constrained solution are identical. So we take as our starting case, the solution for $h = 0.25$. In the more general setting of, for example, a state-variable-constrained LQR problem, we would begin by solving the unconstrained LQR problem.

We have:

$$\dot{C} = v$$

$$\ddot{C} = u$$

$$H = \frac{1}{2}u^2 + \Lambda_x v + \Lambda_v u + \eta\ddot{C}$$

$$\dot{\Lambda}_x = 0$$

$$\dot{\Lambda}_V = -\Lambda_x$$

From $H_u = 0$, we have:

$$u^* = -\Lambda_v - \eta$$

$$\eta \begin{cases} = 0, & \ddot{C} < 0 \\ \geq 0, & \ddot{C} = 0 \text{ and } C = 0 \end{cases}$$

When $\ddot{C} \leq 0$ would be violated on the constraint surface, we select η such that $\ddot{C} = u = 0$, namely $\eta = -\Lambda_v$.

The jump conditions are:

$$\Lambda_x(t_1\text{-}) = \Lambda_x(t_1{}^+) + \alpha_1$$

$$\Lambda_v(t_1\text{-}) = \Lambda_v(t_1{}^+) + \alpha_2$$

The constraints are:

$$C = x(t_1) - h = 0$$

$$\dot{C} = v(t_1) = 0$$

$$\Delta H = H(t_1{}^{-}) - H(t_1{}^{+}) = 0$$

The unknowns $\boldsymbol{U}$ and constraints $\boldsymbol{S}$ are:

$$\boldsymbol{U} = [\Lambda_x(0) \quad \Lambda_v(0) \quad \alpha_1 \quad \alpha_1 \quad t_1]^T$$

$$\boldsymbol{S} = [x(1) \quad \mathrm{v}(1) \quad C \quad \dot{C} \quad \Delta H]^T$$

The analytical solution

Before proceeding, let us examine aspects of the analytical solution. From the solution at a given value of h, we can plot trajectories of the unknowns $\boldsymbol{U}$ vs. h. This is done in Figure 24.

The plots reveal that a change in the behavior of the unknowns with respect to h occurs at the point $h = \frac{1}{6}$. This is a known aspect of the analytical solution corresponding to the point when, instead of impinging on the constraint surface only instantaneously, the trajectory begins to ride the constraint for a finite time. Trajectory solutions at and around this point are shown in Figure 25.

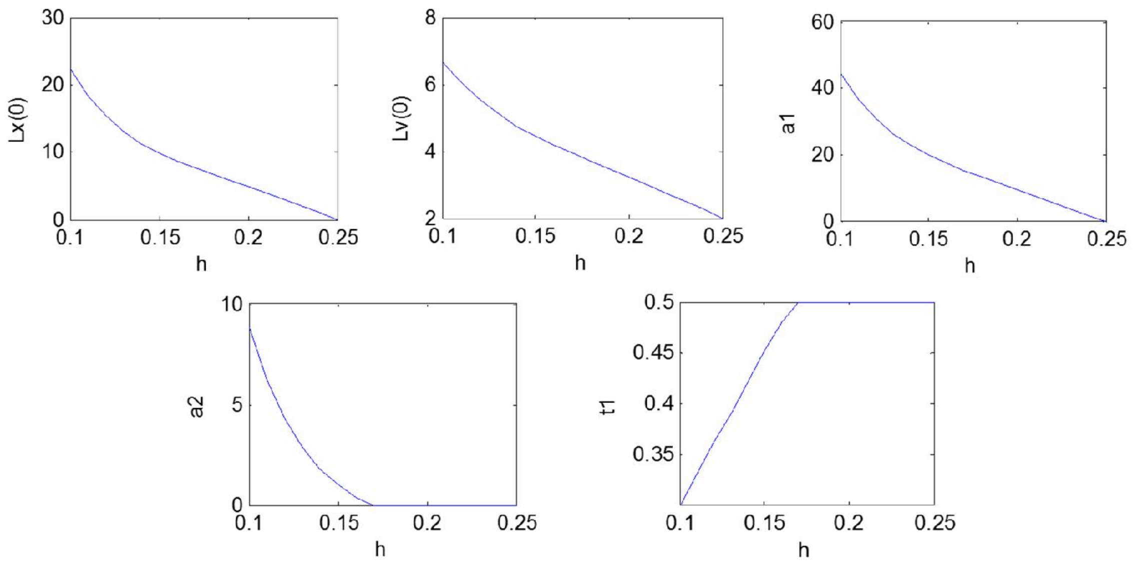


Figure 24 Shooting function unknowns vs. constraint *h* for Breakwell problem, from analytical solution

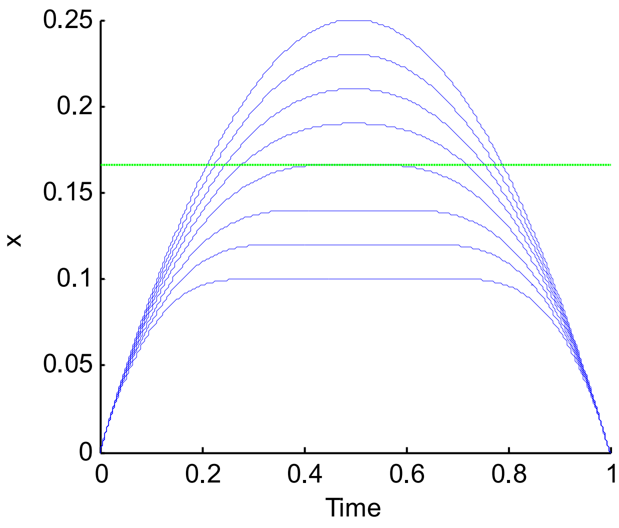


Figure 25 Breakwell state variable solutions for various constraint values, with the constraint h=1/6 marked at which the state begins to ride the constraint surface.

We expect to see a similar behavior in the orbital PE problem, so we consider now whether a homotopy solution is robust in the vicinity of this point. We suggest that it is, because while the derivatives of the unknowns with respect to the homotopy parameter (which will be related to the constraint h) do not exist at the point $h = \frac{1}{6}$, they do exist at all points surrounding it. Assuming local convergence properties, a gradient-following method should be able to traverse this point. For the Breakwell problem, we will show that this is so. (For the orbital PE problem, this point has not yet been tested, for reasons that will be shown later.)

Breakwell sensitivity computations

For the sensitivity solution, we begin by taking the analytical solution at $h = 0.25$, the point where the constrained and unconstrained solutions coincide. This solution, in terms of the unconstrained problem, is:

$$\Lambda_x(0) = 0 \quad ; \quad \Lambda_v(0) = 2$$

We extend the solution to the constrained setting by selecting correct values for the additional unknowns α_1, α_2, t_1:

$$\alpha_1 = 0 \quad ; \quad \alpha_2 = 0 \quad ; \quad t_1 = 0.5$$

α_1 and α_2 will always be zero when beginning with the unconstrained solution, while t_1 should be selected as the time when the states maximize the constraint function. The starting value of the constraint parameter, if not already known, is selected so that the constraint function takes the maximum admissible value of zero at t_1.

We now perform a homotopy in the constraint parameter h. Beginning with $h = 0.25$, we select a target value of $h = 0.1$.

We require a sensitivity matrix $\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}$ and sensitivity vector $\frac{\partial \boldsymbol{S}}{\partial \varepsilon}$, ε a homotopy parameter relating to the constraint parameter h. Specifically, we vary ε from zero to one, with $h(\varepsilon) = 0.25 - 0.15\varepsilon$. For the sensitivity vector, the only element of $\boldsymbol{S}$ affected by ε is C:

$$\frac{\partial \boldsymbol{S}}{\partial \varepsilon} = \frac{\partial \boldsymbol{S}}{\partial h}\frac{\partial h}{\partial \varepsilon} = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \end{bmatrix}^T \frac{\partial h}{\partial \varepsilon} = \begin{bmatrix} 0 & 0 & 0.15 & 0 & 0 \end{bmatrix}^T$$

For the sensitivity matrix, we have:

$$\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} = \begin{bmatrix} \frac{\partial x(1)}{\partial \Lambda_{x_0}} & \frac{\partial x(1)}{\partial \Lambda_{v_0}} & \frac{\partial x(1)}{\partial \alpha_1} & \frac{\partial x(1)}{\partial \alpha_2} & \frac{\partial x(1)}{\partial t_1} \\ \frac{\partial v(1)}{\partial \Lambda_{x_0}} & \frac{\partial v(1)}{\partial \Lambda_{v_0}} & \frac{\partial v(1)}{\partial \alpha_1} & \frac{\partial v(1)}{\partial \alpha_2} & \frac{\partial v(1)}{\partial t_1} \\ \frac{\partial C}{\partial \Lambda_{x_0}} & \frac{\partial C}{\partial \Lambda_{v_0}} & \frac{\partial C}{\partial \alpha_1} & \frac{\partial C}{\partial \alpha_2} & \frac{\partial C}{\partial t_1} \\ \frac{\partial \dot{C}}{\partial \Lambda_{x_0}} & \frac{\partial \dot{C}}{\partial \Lambda_{v_0}} & \frac{\partial \dot{C}}{\partial \alpha_1} & \frac{\partial \dot{C}}{\partial \alpha_2} & \frac{\partial \dot{C}}{\partial t_1} \\ \frac{\partial \Delta H}{\partial \Lambda_{x_0}} & \frac{\partial \Delta H}{\partial \Lambda_{v_0}} & \frac{\partial \Delta H}{\partial \alpha_1} & \frac{\partial \Delta H}{\partial \alpha_2} & \frac{\partial \Delta H}{\partial t_1} \end{bmatrix}$$

The upper-left 2x2 elements of this matrix are elements of the state transition matrix. They are computed in the manner presented in Chapter II, by initializing a state transition matrix $\frac{\partial \boldsymbol{Y}(0)}{\partial \boldsymbol{Y}_0} = I$ and propagating with equation (1) with $\frac{\partial \boldsymbol{F}}{\partial \varepsilon} = \boldsymbol{0}$. We compute $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{Y}}$ conditionally:

$$\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{Y}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial \dot{v}}{\partial \Lambda_v} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}, \quad \frac{\partial \dot{v}}{\partial \Lambda_v} = \begin{cases} 0, & \eta > 0 \\ -1, & \eta = 0 \end{cases}$$

This expresses the fact that if $\eta > 0$, then η is chosen so that $u = 0$ regardless of Λ_v, which would otherwise set the controls.

The upper-right 2x3 elements of $\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}$ are elements of the parameter sensitivity matrices $\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{\alpha}}$ and $\left.\frac{\partial \boldsymbol{Y}}{\partial t_1}\right|_{{t_1}^+}$ at final time. These are initialized at time t_1 as:

$$\frac{\partial \boldsymbol{Y}({t_1}^+)}{\partial \boldsymbol{\alpha}} = \begin{bmatrix} 0_{2\times2} \\ I_{2\times2} \end{bmatrix}$$

$$\left.\frac{\partial \boldsymbol{Y}}{\partial t_1}\right|_{{t_1}^+} = \begin{bmatrix} 0 & \frac{\partial v({t_1}^+)}{\partial t_1} & 0 & -\alpha_1 \end{bmatrix}^T, \qquad \frac{\partial v({t_1}^+)}{\partial t_1} = \begin{cases} -\Lambda_v({t_1}^-), & \eta > 0 \\ -\alpha_2, & \eta = 0 \end{cases}$$

and subsequently propagated using equation (7). The conditional effect of η is that if control is being suppressed to zero, $\frac{\partial v({t_1}^+)}{\partial t_1}$ is equal to the entire control at ${t_1}^-$. Otherwise, $\frac{\partial v({t_1}^+)}{\partial t_1}$ is only based on the jump parameter α_2.

Also, referring to equation (8), note that here we have $\frac{\partial}{\partial \boldsymbol{Y}({t_1}^-)}\left(\begin{matrix} \boldsymbol{0} \\ -\begin{bmatrix} C_x \\ \dot{C}_x \end{bmatrix}^T \boldsymbol{\alpha} \end{matrix} \right) = \boldsymbol{0}$, so there is no jump in the state transition matrix at t_1.

At time ${t_1}^+$, we also compute the values of the shooting function C, $\dot{C}$, and ΔH, and the sensitivities of these parameters. C and $\dot{C}$ are evaluated from the states at t_1. Their sensitivities at t_1 are:

$$\frac{\partial C}{\partial \boldsymbol{U}} = \begin{bmatrix} \frac{\partial C}{\partial \Lambda_{x_0}} & \frac{\partial C}{\partial \Lambda_{v_0}} & \frac{\partial C}{\partial \alpha_1} & \frac{\partial C}{\partial \alpha_2} & \frac{\partial C}{\partial t_1} \end{bmatrix} = \begin{bmatrix} \frac{\partial x({t_1}^-)}{\partial \Lambda_{x_0}} & \frac{\partial x({t_1}^-)}{\partial \Lambda_{v_0}} & 0 & 0 & v(t_1) \end{bmatrix}$$

$$\frac{\partial \dot{C}}{\partial \boldsymbol{U}} = \begin{bmatrix} \frac{\partial \dot{C}}{\partial \Lambda_{x_0}} & \frac{\partial \dot{C}}{\partial \Lambda_{v_0}} & \frac{\partial \dot{C}}{\partial \alpha_1} & \frac{\partial \dot{C}}{\partial \alpha_2} & \frac{\partial \dot{C}}{\partial t_1} \end{bmatrix} = \begin{bmatrix} \frac{\partial v({t_1}^-)}{\partial \Lambda_{x_0}} & \frac{\partial v({t_1}^-)}{\partial \Lambda_{v_0}} & 0 & 0 & u({t_1}^-) \end{bmatrix}$$

where $\frac{\partial x(t_1{}^-)}{\partial \Lambda_{x_0}}, \frac{\partial x(t_1{}^-)}{\partial \Lambda_{v_0}}, \frac{\partial v(t_1{}^-)}{\partial \Lambda_{x_0}}$, and $\frac{\partial v(t_1{}^-)}{\partial \Lambda_{v_0}}$ are obtained from the state transition matrix at $t_1{}^-$.

To compute ΔH sensitivities, we write the change in the Hamiltonian entirely in terms of states at $t_1{}^-$, and jump parameters. We have:

$$\Delta H = \begin{cases} -\frac{1}{2}\Lambda_v(t_1{}^-)^2 + \alpha_1 v, & \eta > 0 \\ \Lambda_v(t_1{}^-) + \frac{1}{2}\alpha_2{}^2 + \alpha_1 v, & \eta = 0 \end{cases}$$

The sensitivities of ΔH are:

$$\frac{\partial \Delta H}{\partial \boldsymbol{U}} = \begin{bmatrix} \frac{\partial \Delta H}{\partial \Lambda_{x_0}} & \frac{\partial \Delta H}{\partial \Lambda_{v_0}} & \frac{\partial \Delta H}{\partial \alpha_1} & \frac{\partial \Delta H}{\partial \alpha_2} & \frac{\partial \Delta H}{\partial t_1} \end{bmatrix}$$

To obtain the first two elements, we first require the sensitivity of ΔH to the states and costates $\boldsymbol{Y} = [x \quad v \quad \Lambda_x \quad \Lambda_v]^T$ at $t_1{}^-$ (note that all states referred to here, as well as the state transition matrix, are taken at time $t_1{}^-$):

$$\frac{\partial \Delta H}{\partial \boldsymbol{Y}} = \begin{bmatrix} 0 & \alpha_1 & 0 & \frac{\partial \Delta H}{\partial \Lambda_v} \end{bmatrix}, \qquad \frac{\partial \Delta H}{\partial \Lambda_v} = \begin{cases} -\Lambda_v, & \eta > 0 \\ -\alpha_2, & \eta = 0 \end{cases}$$

Employing the state transition matrix at $t_1{}^-$, we then have:

$$\frac{\partial \Delta H}{\partial \boldsymbol{U}} = \begin{bmatrix} \frac{\partial \Delta H}{\partial \boldsymbol{Y}}\frac{\partial \mathbf{Y}}{\partial \Lambda_{x_0}} & \frac{\partial \Delta H}{\partial \boldsymbol{Y}}\frac{\partial \mathbf{Y}}{\partial \Lambda_{v_0}} & v & \frac{\partial \Delta H}{\partial \alpha_2} & \frac{\partial \Delta H}{\partial t_1} \end{bmatrix}$$

with

$$\frac{\partial \Delta H}{\partial \alpha_2} = \begin{cases} 0, & \eta > 0 \\ -\Lambda_v + \alpha_2 & \eta = 0 \end{cases}$$

and

$$\frac{\partial \Delta H}{\partial t_1} = \begin{cases} \Lambda_v \Lambda_x + \alpha_1 u(t_1{}^-), & \eta > 0 \\ \alpha_2 \Lambda_x + \alpha_1 u(t_1{}^-), & \eta = 0 \end{cases}$$

Breakwell homotopy results

We perform the homotopy in the Breakwell problem beginning with a solution at the constraint value $h = 0.25$, corresponding to the unconstrained solution, and proceeding to $h = 0.1$. The homotopy predictor-corrector tracks the solution to the end-point in 17 steps, although this is a function of the step size and step adaptations chosen. Figure 26 plots the solution for unknowns $U = [\Lambda_x(0) \quad \Lambda_v(0) \quad \alpha_1 \quad \alpha_2 \quad t_1]^T$ obtained along the curve, overlaid on the analytical solution curves; the homotopy solutions track the analytical solutions.

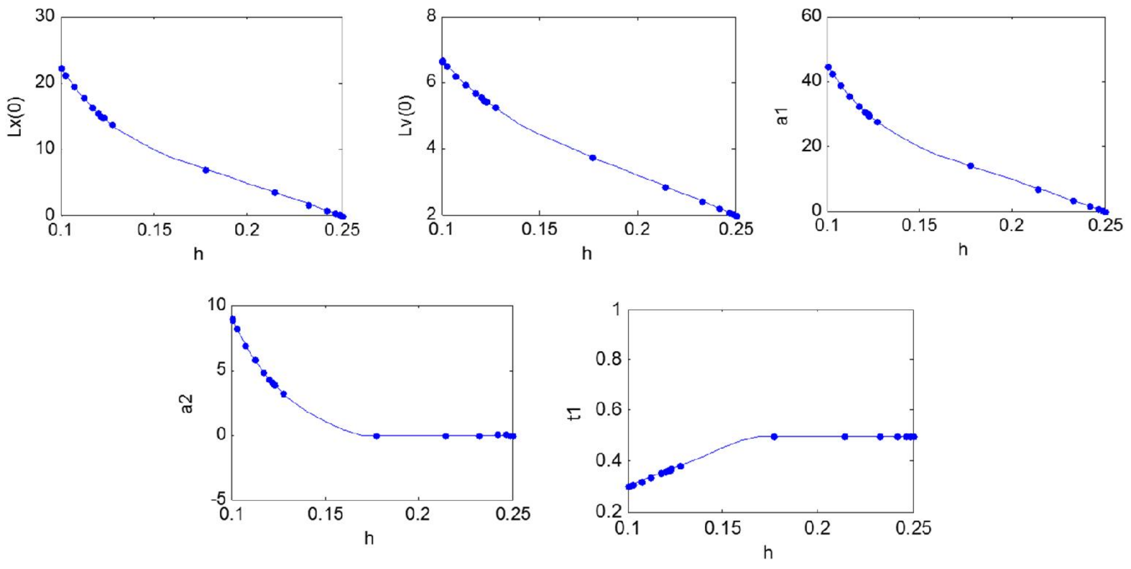


Figure 26 Solutions for the Breakwell unknowns obtained along the homotopy path (dots), overlaid on the analytical solution values (solid lines)

Figure 26 also indicates features of the step length corrector. Bearing in mind that the homotopy begins at the right-hand side of each plot, the step size is initially increased because the system is quite regular. In this regime, the homotopy solver only requires integration (predictor) steps, with no correction steps. However, at the value $h = \frac{1}{6}$, the behavior of the system changes. This is the point where the trajectory must begin to ride along the constraint surface for finite time. The integrator in fact takes such a large step that it soars past this value. A check of the homotopy diagnostics reveals that seven Newtonian corrections are needed at this step, compared to only one correction at subsequent steps. However, the step length adapter greatly reduces the step size for the remaining steps. All of this indicates healthy functioning of the predictor-corrector and step length adapter. However, initial step size and step length correction parameters can be adjusted as needed.

The state, costate and control trajectories generated by the homotopy solution for the point $h = 0.1$ are shown in Figure 27, overlaid over the analytical solutions. Given that accurate values for the unknowns were obtained, it is a foregone conclusion that the trajectories match.

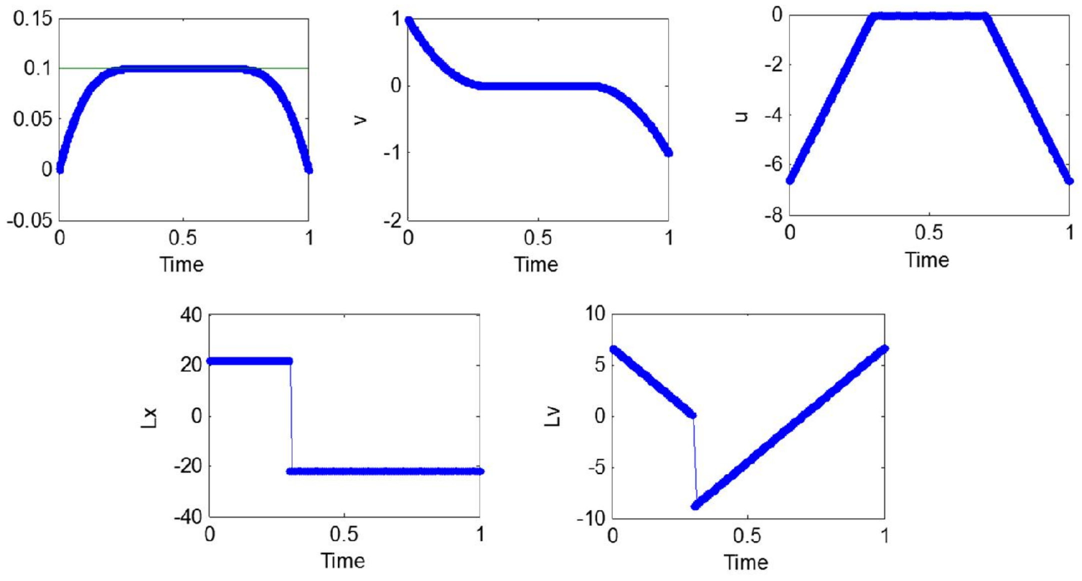


Figure 27 State, costate and control trajectories for the homotopy solution at *h=0.1*, overlaid over the analytical solutions.

TPBVP formulation of the pursuer altitude-constrained orbital PE problem

In this section, we reformulate the orbital PE problem as an optimal control problem subject to state variable inequality constraints. Specifically, we impose the constraint that pursuer altitude must not fall below a certain minimum value. Although our treatment is for the pursuer only for illustration, the technique could be equally well applied to either or both players.

Since we only consider the pursuer here, we will drop the subscript $_P$ in this section.

We take the state-variable inequality constraint:

$$C(\boldsymbol{r}) = \frac{1}{2}(h^2 - \boldsymbol{r}^T\boldsymbol{r}) \leq 0$$

with h a specified minimum altitude, measured from the center of the Earth.

C is a second-order inequality constraint, as we see from taking two time derivatives:

$$\dot{C}(\boldsymbol{r},\boldsymbol{v}) = -\boldsymbol{r}^T\boldsymbol{v}$$

$$\ddot{C}(\boldsymbol{r},\boldsymbol{v},\boldsymbol{u}) = -\boldsymbol{r}^T\left[-\frac{\mu\boldsymbol{r}}{(\boldsymbol{r}^T\boldsymbol{r})^{\frac{3}{2}}}+\boldsymbol{u}\right]-\boldsymbol{v}^T\boldsymbol{v} = -\boldsymbol{r}^T\boldsymbol{u}-\boldsymbol{v}^T\boldsymbol{v}+\frac{\mu}{(\boldsymbol{r}^T\boldsymbol{r})^{1/2}} \tag{9}$$

$\dot{C}$ is a constraint on the component of velocity penetrating the altitude constraint surface. $\ddot{C} = 0$ is a constraint requiring that the combined effects of control effort and gravitational force are exactly sufficient to keep the spacecraft on a circular orbit, i.e. constant altitude. Given controls of bounded magnitude, this constraint will not always be satisfiable; but since the altitude-constrained results from PC09 suggest that solutions exist for those cases.

To enforce the state variable inequality constraint over the trajectory, assuming a single incursion on the constraint surface, we interpret C and $\dot{C}$ as equality constraints at some free intermediate time t_1. We then apply the jump conditions to the costate and evaluate $\ddot{C}$ at t_1^+. If $\ddot{C}\big(u(t_1^+,\eta=0)\big) > 0$, we select a value for a Lagrange vector $\eta > 0$ such that $\ddot{C}\big(u(t_1^+,\ddot{\eta} > 0)\big) = 0$. η is selected based on minimizing the Hamiltonian with the constraint term incorporated.

$$H = \boldsymbol{\Lambda}^T\boldsymbol{f} + \boldsymbol{\Lambda}_E^T\boldsymbol{f}_E + \eta\ddot{C} = \boldsymbol{\Lambda}^T\boldsymbol{f} + \boldsymbol{\Lambda}_E^T\boldsymbol{f}_E + \eta\left[-\boldsymbol{r}^T\boldsymbol{u}-\boldsymbol{v}^T\boldsymbol{v}+\frac{\mu}{(\boldsymbol{r}^T\boldsymbol{r})^{1/2}}\right]$$

Pursuer controls are selected to minimize the Hamiltonian, for which terms that do not include pursuer controls can be dropped:

$$\boldsymbol{u}^* = \arg\min_{\alpha,\beta} \boldsymbol{\Lambda}_v^T \boldsymbol{u}(\alpha,\beta) - \eta \boldsymbol{r}^T \boldsymbol{u}(\alpha,\beta) = \arg\min_{\alpha,\beta} (\boldsymbol{\Lambda}_v - \eta \boldsymbol{r})^T \boldsymbol{u}(\alpha,\beta)$$

where (α,β) are the angles that set the constant-magnitude vector $\boldsymbol{u}$.

Absent the altitude constraint, the optimal controls were $\boldsymbol{u}^* = \arg\min_{\alpha,\beta} \boldsymbol{\Lambda}_v^T \boldsymbol{u}(\alpha,\beta)$, which resulted in $\boldsymbol{u}^*$ pointing opposite to $\boldsymbol{\Lambda}_v$. While on the altitude constraint surface, we select $\boldsymbol{u}^*$ to point opposite to $\boldsymbol{\Lambda}_v - \eta \boldsymbol{r}$. Specifically, we have:

$$\boldsymbol{u}^* = -T_m \frac{\boldsymbol{\Lambda}_v - \eta \boldsymbol{r}}{|\boldsymbol{\Lambda}_v - \eta \boldsymbol{r}|} = T_m \frac{(-\boldsymbol{\Lambda}_v + \eta \boldsymbol{r})}{|-\boldsymbol{\Lambda}_v + \eta \boldsymbol{r}|} \tag{10}$$

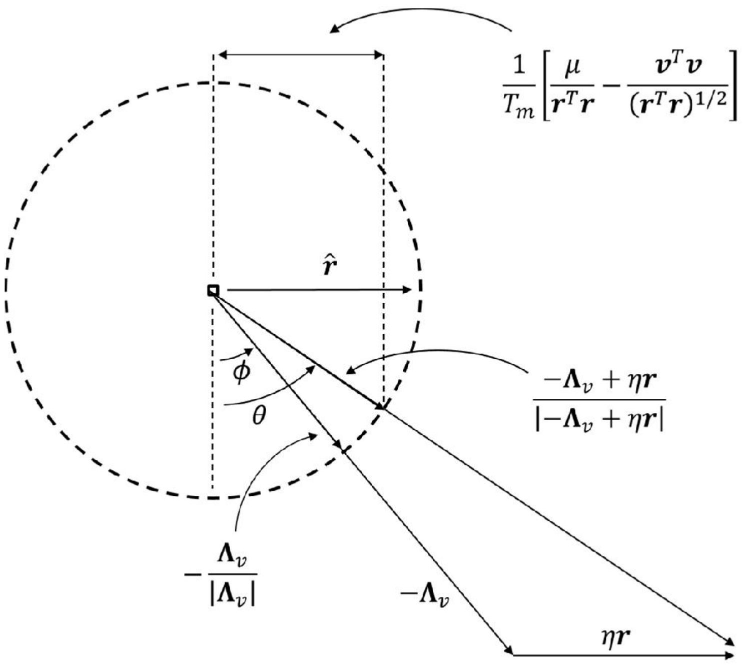
With this, we must find the value of η such that $\ddot{C} = 0$. Substituting equation (10) into equation (9), we have:

$$\ddot{C} = -\boldsymbol{r}^T \left(T_m \frac{(-\boldsymbol{\Lambda}_v + \eta \boldsymbol{r})}{|-\boldsymbol{\Lambda}_v + \eta \boldsymbol{r}|} \right) - \boldsymbol{v}^T \boldsymbol{v} + \frac{\mu}{(\boldsymbol{r}^T \boldsymbol{r})^{\frac{1}{2}}} = 0 \tag{11}$$

It is helpful to consider this requirement in terms of the relationship between the unit vector $\frac{-\boldsymbol{\Lambda}_v + \eta \boldsymbol{r}}{|-\boldsymbol{\Lambda}_v + \eta \boldsymbol{r}|}$, and a unit vector $\hat{\boldsymbol{r}}$ pointing in the direction $\boldsymbol{r}$. To this end, we divide equation (11) by $(T_m \cdot |\boldsymbol{r}|)$, and re-arrange:

$$\hat{\boldsymbol{r}}^T \left(\frac{-\boldsymbol{\Lambda}_v + \eta \boldsymbol{r}}{|-\boldsymbol{\Lambda}_v + \eta \boldsymbol{r}|} \right) = \frac{1}{T_m} \left[\frac{\mu}{\boldsymbol{r}^T \boldsymbol{r}} - \frac{\boldsymbol{v}^T \boldsymbol{v}}{(\boldsymbol{r}^T \boldsymbol{r})^{1/2}} \right] \tag{12}$$

We see that the projection of the unit vector $\frac{-\boldsymbol{\Lambda}_v + \eta \boldsymbol{r}}{|-\boldsymbol{\Lambda}_v + \eta \boldsymbol{r}|}$ in the radial direction must be equal to the quantity on the right-hand side of equation (12). This situation is illustrated in Figure 28.



$\frac{1}{T_m}\left[\frac{\mu}{\boldsymbol{r}^T\boldsymbol{r}} - \frac{\boldsymbol{v}^T\boldsymbol{v}}{(\boldsymbol{r}^T\boldsymbol{r})^{1/2}}\right]$
$\hat{\boldsymbol{r}}$
ϕ
θ
$\frac{-\boldsymbol{\Lambda}_v + \eta\boldsymbol{r}}{|-\boldsymbol{\Lambda}_v + \eta\boldsymbol{r}|}$
$-\frac{\boldsymbol{\Lambda}_v}{|\boldsymbol{\Lambda}_v|}$
$-\boldsymbol{\Lambda}_v$
$\eta\boldsymbol{r}$

Figure 28 Contribution of constraint Lagrange vector η to the constrained control

Since the controls are of fixed magnitude, for the constraint to be achievable we must have:

$$-1 \leq \frac{1}{T_m}\left[\frac{\mu}{\boldsymbol{r}^T\boldsymbol{r}} - \frac{\boldsymbol{v}^T\boldsymbol{v}}{(\boldsymbol{r}^T\boldsymbol{r})^{1/2}}\right] \leq 1$$

We solve for η as follows. With reference to Figure 28, we must choose η to rotate the unit control vector from an angle ϕ to an angle θ with respect to the plane normal to $\boldsymbol{r}$. Note that because we are only concerned with the radial projection of the control vector, we are not concerned with quadrant ambiguity. We have:

$$\sin\phi = -\frac{\boldsymbol{\Lambda}_v^T\boldsymbol{r}}{|\boldsymbol{\Lambda}_v| \cdot |\boldsymbol{r}|}$$

$$\cos\phi = (1 - \sin^2\phi)^{1/2}$$

$$\sin\theta = \frac{1}{T_m}\left[\frac{\mu}{\boldsymbol{r}^T\boldsymbol{r}} - \frac{\boldsymbol{v}^T\boldsymbol{v}}{(\boldsymbol{r}^T\boldsymbol{r})^{1/2}}\right]$$

$$\cos\theta = (1 - \sin^2\theta)^{1/2}$$

We then set the tangent of the angle between the vector $(-\boldsymbol{\Lambda}_v + \eta\boldsymbol{r})$ and the plane normal to $\boldsymbol{r}$ equal to $\tan\theta = \sin\theta\cos^{-1}\theta$:

$$\frac{-\boldsymbol{\Lambda}_v^T\frac{\boldsymbol{r}}{|\boldsymbol{r}|} + \eta|\boldsymbol{r}|}{(\Lambda_v^T\Lambda_v)^{1/2}\cos\phi} = \sin\theta\cos^{-1}\theta$$

Rearranging, we have:

$$\eta = (\boldsymbol{\Lambda}_v^T\boldsymbol{r})(\boldsymbol{r}^T\boldsymbol{r})^{-1} + (\boldsymbol{\Lambda}_v^T\boldsymbol{\Lambda}_v)^{1/2}(\boldsymbol{r}^T\boldsymbol{r})^{-1/2}\cos\phi\sin\theta\cos^{-1}\theta$$

The remaining modification to the system is to account for the constraint in the costate dynamics:

$$\dot{\boldsymbol{\Lambda}}_r = -H_r = -G(\boldsymbol{r})^T\boldsymbol{\Lambda}_v - \eta\ddot{C}_r$$

$$\dot{\boldsymbol{\Lambda}}_v = -H_v = -\boldsymbol{\Lambda}_r - \eta\ddot{C}_v$$

with

$$\ddot{C}_r = -\boldsymbol{u} - \mu(\boldsymbol{r}^T\boldsymbol{r})^{-3/2}\boldsymbol{r}$$

$$\ddot{C}_v = -2\boldsymbol{v}$$

A few altitude-constrained results

The sensitivity computations for the complete altitude-constrained solution have not been implemented, but sensitivities have been implemented for position and velocity waypoint constraints. This allows us to examine altitude constrained solutions up to the point when the pursuer must ride along the constraint surface for finite time. The reason for this is as follows.

Consider the Breakwell problem. When the state constraint is first applied to the unconstrained solution, the jump parameters are $\alpha_1 = 0$ and $\alpha_2 = 0$. As the constraint is then adjusted, the solution initially only contacts the constraint surface for an instant. In this region, the jump parameters are $\alpha_1 \neq 0, \alpha_2 = 0$. The fact that $\alpha_2 = 0$ means that this constraint is not necessary. The point when this behavior changes to one where the state rides along the constraint surface for finite time is precisely the point when jump parameter α_2 becomes other than zero.

When α_2 becomes other than zero, we must distinguish between a solution that enforces the acceleration-level constraint $\ddot{C} = 0$ via the Lagrange vector η, and one that does not. However, as long as $\alpha_2 = 0$, these two solutions are the same, and they are also the same as the solution to the system with only the position-level constraint. Both the velocity-level constraint and the control inequality constraint are not active.

These different trajectories are illustrated in Figure 29, for general trajectory shapes that are roughly indicative of common orbital PE pursuer altitude trajectories.

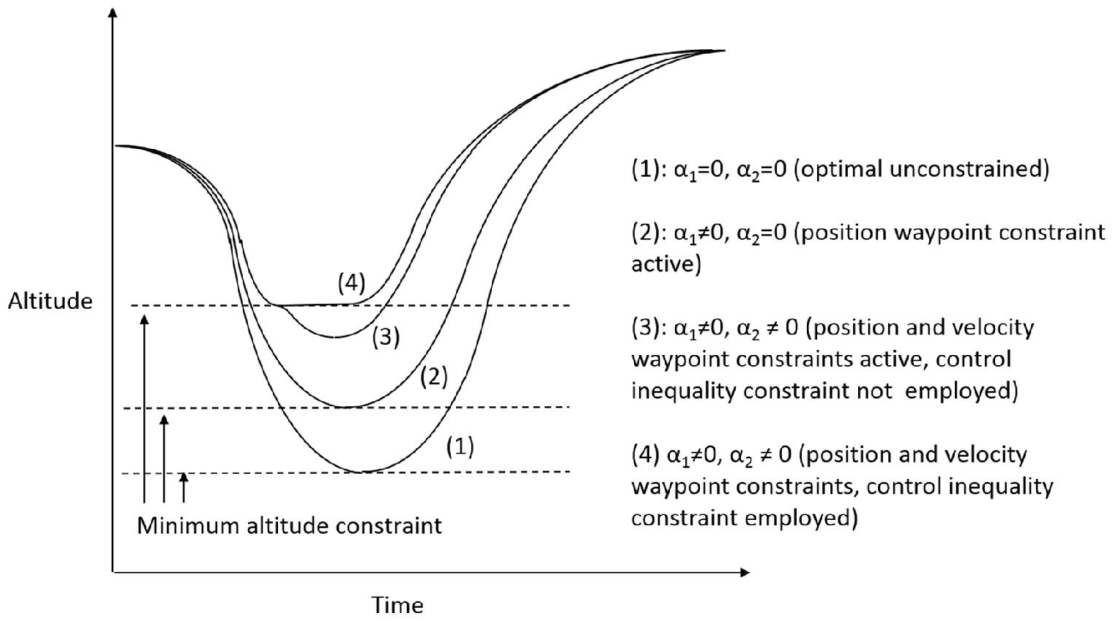

Altitude
(1): α1=0, α2=0 (optimal unconstrained)
(2): α1≠0, α2=0 (position waypoint constraint active)
(3): α1≠0, α2 ≠ 0 (position and velocity waypoint constraints active, control inequality constraint not employed)
(4) α1≠0, α2 ≠ 0 (position and velocity waypoint constraints, control inequality constraint employed)
(4)
(3)
(2)
(1)
Minimum altitude constraint
Time

Figure 29 Notional trajectory types corresponding to various sets of active constraints.

Our goal in this section is to perform a homotopy in an altitude constraint that will allow the solution of trajectory types (1) and (2) of Figure 29. For this, we must only expand our constraint sensitivity matrix $\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}}$ to include waypoint constraints $C = 0$ and $\dot{C} = 0$, and the differential Hamiltonian constraint ΔH, and sensitivities to the unknown parameters α_1, α_2, t_1. The sensitivity computations for these parameters are shown in the Appendix. We do not incorporate the constraint Lagrange vector η, but will only proceed in regions where it is not active, i.e. where trajectories correspond to types (1) and (2) of Figure 29.

We return to PC09 case 1 from the Pontani and Conway study, in which the altitude constraint was violated in our earlier homotopy solution. Using the methods described in this chapter, except for the sensitivities to eta, we can perform a homotopy in the altitude parameter h, seeking to raise the altitude constraint to the desired level. Homotopy solutions will represent valid solutions to the complete altitude-constrained problem as long as $\alpha_2 = 0$.

The minimum altitude attained in the unconstrained solution to PC case 1, which becomes the altitude constraint at the beginning of the homotopy, is 0.9839 DU, or 102.6474 km below the Earth’s surface. This corresponds to the value $\varepsilon = 0$ of the homotopy parameter. $\varepsilon = 1$ corresponds to the desired altitude constraint of 1.0157 DU, or 100 km above the Earth’s surface.

The homotopy solution curve obtained in this manner is shown in Figure 30. We comment on the result.

In homotopy solutions obtained so far, the problem encountered was that the unknown costate values blew up to infinitely large magnitude, due to encountering a barrier surface. Here, no barrier surface is encountered. Yet we see another phenomenon that is, although not fatal, at least undesirable: the homotopy solution curve turns away from the desired direction, and wanders remarkable far in the other direction: all the way to $\varepsilon < -7$, meaning it has traversed seven times as far in the wrong direction as it needed to traverse in the correct direction. When ε is negative, the homotopy traversal is bringing the altitude constraint down, rather than up. (It is the use of equation (2) as the homotopy traversal mechanism that enables the curve in this example to be traced as ε changes direction and becomes negative.)

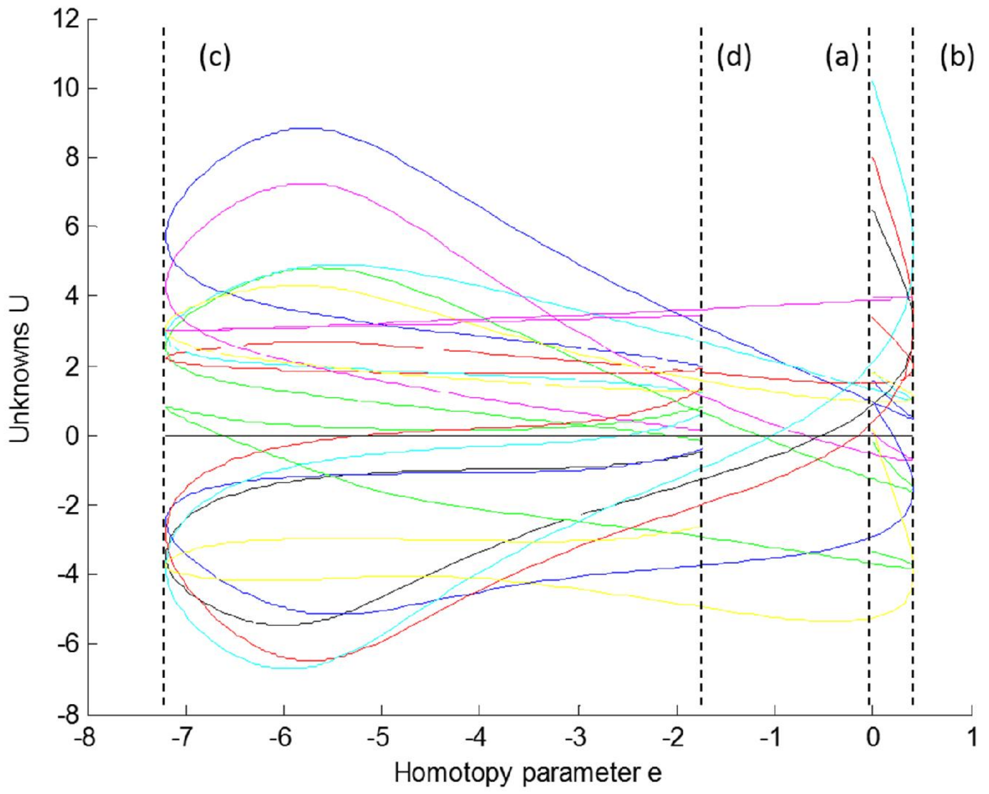


Figure 30 Altitude-raising homotopy solution curve for PC09 case 1, stopping at point where jump parameter α_2 departs from zero. Points are identified for further discussion.

The four points marked on the curve in Figure 30 are the starting location at $\varepsilon = 0$, two locations where the curve changes direction, and the last location where $\alpha_2 \approx 0$, at which point the curve stops. The altitude trajectories for these four points are shown in Figure 31. These plots illustrate that the homotopy is, in fact, computing solutions with lower altitude waypoint constraints than the unconstrained PC01 homotopy solution.

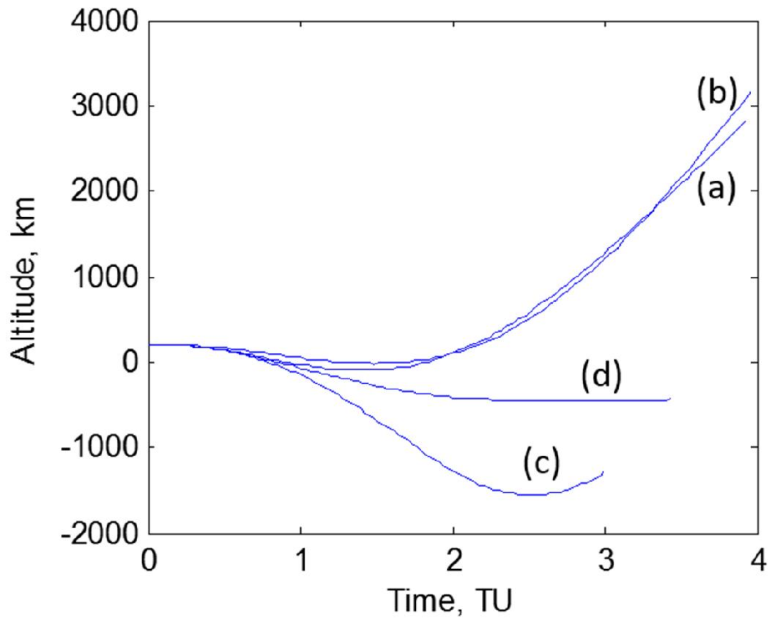


Figure 31 Pursuer altitude trajectories for the four marked points of the PC01 altitude constraint homotopy solution

Discussion

The altitude constraint homotopy has not yet failed, in this example. With the implementation of sensitivities for the Lagrange vector η, the curve-tracing could continue where it leaves off at point (3) in the previous figures, and there is no reason to think it could not eventually be traced to the desired solution at $\varepsilon = 1$. However, these curves also make clear that there is no reason to think it will proceed there in a direct manner. With careful step size adaptation methods, curve-tracing may be done quickly, even for long arc lengths. But the seemingly variable nature of the curves here offers less certainty in the success and the speed of the method than is desired.

Sensitivities in η should be implemented before a conclusion is reached on this question. But this is clearly a reminder that the direct progression of a homotopy solution curve from $\varepsilon = 0$ to $\varepsilon = 1$ should not be assumed. In addition to the problems posed by barrier surfaces, arbitrarily long solution traversals are another reason why when sensitivity methods are used, a backup method, such as random search and collocation, should also be in place.

CHAPTER VI

HYBRID ONE-SIDED/TWO-SIDED CONTROLLERS

In this chapter, we examine an idea that has arisen from this work, of a hybrid spacecraft controller employing both one-sided and two-sided controls. One-sided control refers to an optimal control selected to achieve an objective in the absence of an adversary. A station-keeping control is the most obvious example, as are orbit transfers, pointing operations, and so on. Two-sided control refers to optimal control in a differential game setting, as examined in this work.

The motivation for a hybrid one-sided/two-sided controller is that generally, spacecraft are not deployed simply to perform in a differential game. The evader spacecraft in particular is considered as a spacecraft performing a primary mission, until it is threatened by a pursuer spacecraft. However, spacecraft PE studies to date have not considered the interplay of spacecraft objectives. This question is addressed by a hybrid one-sided/two-sided controller.

The foremost question is how a spacecraft can decide when to switch between its one-sided control and two-sided controls. A natural answer is that the switching function of the hybrid controller should be based on the present value of the differential game. The value of a differential game is equivalent to the cost-to-go, which in turn is a measure of the imminence of interception under optimal play. For example, in the orbital PE game considered here, the value of the game is the interception time under optimal play. This would clearly be a relevant factor in a spacecraft's control strategy.

To use the value of the differential game as a control switching function requires that the value of the differential game be known, and this is done by solving the differential game. Thus, this hybrid control scheme requires a means of solving the differential game online rapidly, at each interval at which a control decision is required.

To accomplish this, we employ a method similar to the feedback control method shown earlier. Given an initial solution at the start of a trajectory—which is assumed to have been obtained either online or offline by any available method—we then perform homotopies in the system states to update the game value function as the trajectories of the pursuer and evader spacecraft evolve. The difference between the hybrid controller and the optimal PE feedback controller is that while the hybrid control will solve the differential game at every time step, it will not necessarily use the controls so obtained. Depending on the value of the differential game, it may disregard the optimal PE controls and simply employ its one-sided controls, such as station-keeping, etc. But provided the hybrid controller was structured correctly, this choice would be made in such a way that the spacecraft's goals with respect to the differential game were not threatened.

Novel control behaviors can be achieved in this hybrid control framework. Here, we present two control behaviors that may have useful applications: surreptitious control, and thermostat control. Both can be interpreted for the pursuer and evader roles, but here we demonstrate surreptitious control for the pursuer, and thermostat controls employed simultaneously by the evader and pursuer.

For illustration, we consider surreptitious control from the pursuer's perspective. Suppose that rather than proceeding directly on an optimal interception trajectory toward an evader, a pursuer wished to sneak up on the evader in some way. Perhaps the evader's maneuver capabilities were actually greater than the pursuer's, and so surprise was required; or there was some other motivation to avoid being perceived as a threat for as long as possible.

By employing hybrid one-sided/two-sided control, a pursuer can readily takes advantage of passive orbital motion to reduce the game value (the optimal interception time) whenever possible, and only employs active maneuver when passive maneuver would result in increasing the game value.

We describe thermostat controls as implemented by an evader. (Although the evader now has some primary mission other than pursuit/evasion, for simplicity we still call it an evader.) Suppose an evader detects a nearby spacecraft whose intention is uncertain. The evader could employ a hybrid switching function which allowed primary mission controls to be performed up until the game value—time to optimal interception—dropped below a certain value. At that point, the evader would take an optimal evasive maneuver, but only until the game value was increased back up to a certain safe threshold. At this point, the evader could continue its primary mission, until the next time its two-sided controls were triggered.

Each of these two controls modes gives rise to a distinct trajectory of the game value function in time, as opposed to the pure differential game value function which, in an interception scenario, decreases monotonously from game start until interception.

Qualitative depictions of the game value function trajectories for the pure differential game and the surreptitious and thermostat controls presented here are shown in Figure 32.

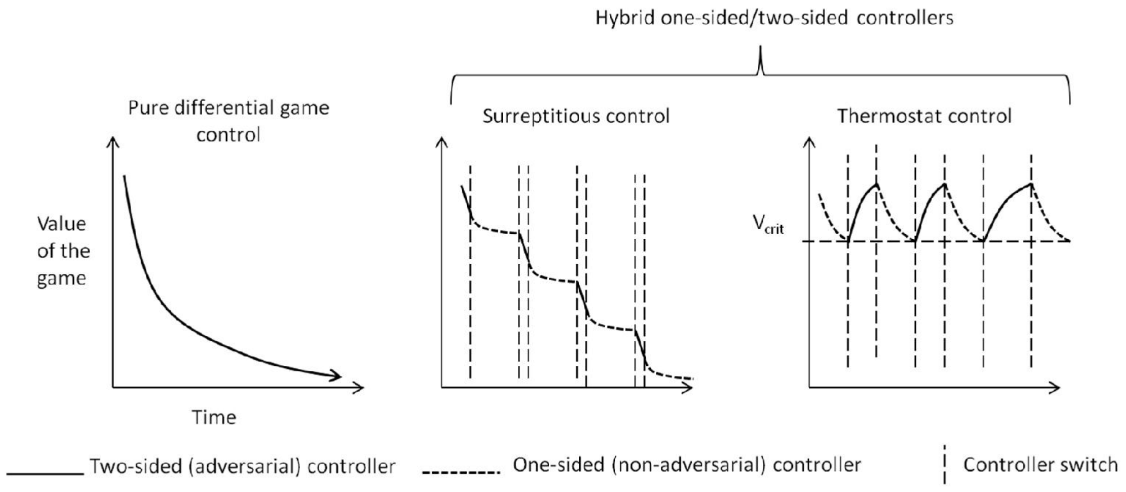


Figure 32 Illustrative behavior of the value function during a pure differential game (left), a pursuer surreptitious control game (center), and an evader thermostat control (right).

In the next sections, we elaborate on the surreptitious control and thermostat control, and demonstrate their numerical implementation.

Surreptitious control

We define the pursuer surreptitious controller as follows. Let $\boldsymbol{T}(\boldsymbol{x})$ be a strategy of the pursuer given game states $\boldsymbol{x}$. $\boldsymbol{u}_{1s}^{*}(\boldsymbol{x})$ and $\boldsymbol{u}_{2s}^{*}(\boldsymbol{x})$ are optimal one-sided and two-sided pursuer controls, respectively. $V(\boldsymbol{x})$ is the value of the differential game, and

$\dot{V}(\boldsymbol{u}, \boldsymbol{u}_E) = V_x \dot{\boldsymbol{x}}(\boldsymbol{u}, \boldsymbol{u}_E)$ is the time rate of change of V under controls $\boldsymbol{u}$ of the pursuer, and the (assumed) controls $\boldsymbol{u}_E$ of the evader.

The solution to the differential game provides V. In orbital PE, V is interception time, t_f. V_x, the gradient in state space of the game value V, is nothing other than the value of the costates $\boldsymbol{\Lambda}$ at the present time. Therefore, the solution to the differential game gives us both V and V_x.

The pursuer hybrid control strategy in its simplest form is then:

$$\boldsymbol{T}(\boldsymbol{x}) = \begin{cases} \boldsymbol{u}_{1s}^*(\boldsymbol{x}), & \dot{V}(\boldsymbol{u}_{1s}^*, \boldsymbol{u}_E) < 0 \\ \boldsymbol{u}_{2s}^*(\boldsymbol{x}), & \text{otherwise} \end{cases}$$

However, our previous findings suggest a complication which can occur in practice. The pursuer must also avoid crossing a barrier surface during passive control. This issue is particularly acute given that the pursuer will only update its control in discrete times steps, and so selected controls are applied blindly for some finite duration after the evaluation of $\dot{V}$.

To guard against crossing the barrier during this time, it is desirable to also consider proximity to the barrier surface. The barrier indicator identified in Chapter IV, which is obtained as a result of solving the differential game, is useful for this purpose. Based on barrier surface considerations, the surreptitious control criterion is expanded to:

$$\boldsymbol{T}(\boldsymbol{x}) = \begin{cases} \boldsymbol{u}_{1s}^*(\boldsymbol{x}) & \dot{V}(\boldsymbol{u}_{1s}^*, \boldsymbol{u}_E) < 0 \text{ or } B > B_{crit} \text{ or } V < V_{term} \\ \boldsymbol{u}_{2s}^*(\boldsymbol{x}) & \text{otherwise} \end{cases}$$

with B the barrier indicator of Chapter IV. B_{crit} is selected such that even with the finite sampling rate of the controller, the pursuer is unlikely to cross the barrier surface between updates. This condition can probably not be guaranteed without adding further sophistication, but it can likely be made fairly robust. The final criterion, $V < V_{term}$ reflects the fact that in the terminal stages of the game, the size of one time step compared to the remaining game time is magnified. In the terminal stages, experience shows that the pursuer must either increase its time sampling rate, or use only two-sided control to ensure interception. The latter method is used here, with $V_{term} = 0.3\ TU$.

Furthermore, to implement this control, the pursuer must make an assumption about evader control. This leads to the question: At each time step, what evader maneuver does the pursuer assume in computing $\dot{V}$?

One workable answer is, the pursuer will always assume the evader is about to take an optimal evasive maneuver. But different assumptions can also be used—a decision not to use the same assumption uniformly can even be justified. For example, the pursuer could compute differential game solutions based on optimal evader maneuver, but assume evader maneuver at the present time step based on observations—i.e. if the evader is not maneuvering, assume that will continue. In a word, the pursuer's goal is to continually reduce the evader's best possible game value, but the pursuer will use evader controls based on observations to accurately compute the present rate of change of game value.

Here, for simplicity, we assume optimal evader maneuver in all cases. The evader will not actually maneuver, but the pursuer will perform all computations on the assumption that the evader is about to optimally maneuver. This is a conservative pursuer strategy.

We now present a numerical implementation of surreptitious control. The initial conditions are shown in Table 9. The scenario is a tailchase at rather close initial separation, but with the orbital planes of pursuer and evader not entirely aligned.

Table 9 Initial conditions for hybrid surreptitious control example

	Initial position (x,y,z)			Initial velocity (x,y,z)			Thrust/mass
	DU			DU/TU			DU/TU^2 (g)
Pursuer	1.063	0	0	0	0.970	0	0.1
Evader	1.043	0.184	0.093	-0.168	0.952	0.085	0 (0.05[*])
[*] Value outside of parentheses used by evader. Value inside parentheses used in pursuer's optimal game computations.							

We show the following results. Figure 33 shows two trajectories. In the first, the pursuer uses optimal interception controls at every time step (assuming the evader will use optimal avoidance controls), while the evader does not maneuver. In the second trajectory, the pursuer uses surreptitious controls. The deviation from the pursuer's orbit is seen to be much more noticeable in the first trajectory.

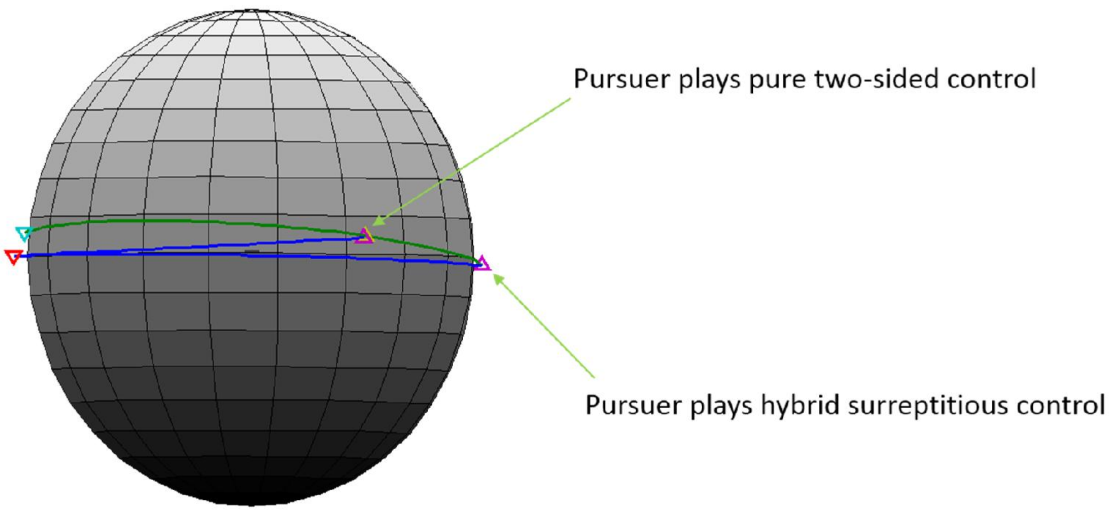

Pursuer plays pure two-sided control
Pursuer plays hybrid surreptitious control

Figure 33 Trajectories resulting from pure two-sided control vs. hybrid surreptitious control by the pursuer

Figure 34 shows the value function of the differential game over time, for the pursuer's pure two-sided control and surreptitious control. It can be seen that the time of interception is extended in surreptitious control, and the pursuer employs two-sided control approximately every other time step. In theory, the game value should still monotonically decrease in surreptitious control, but the effect of holding a control over a finite time step sometimes results in small increases in the game function.

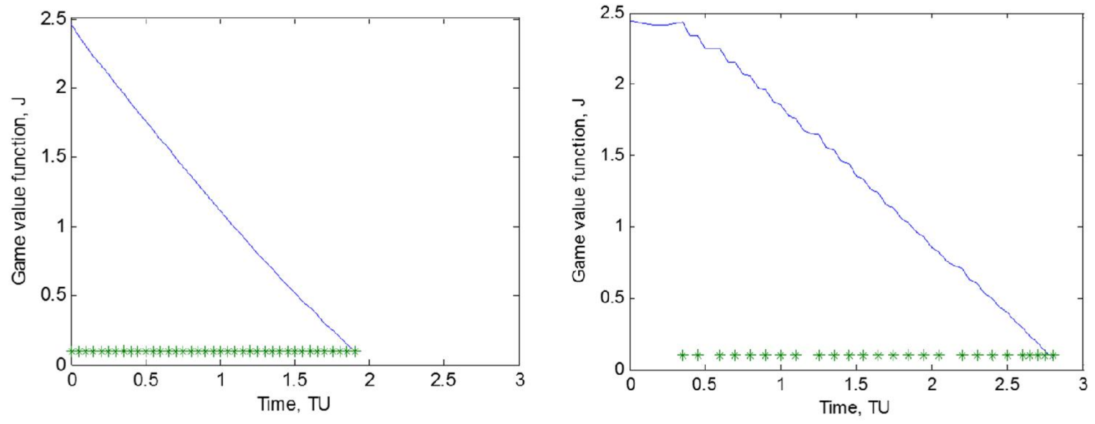


Figure 34 Game value vs. time, for pure two-sided control and surreptitious control. Asterisks denote time steps when pursuer used two-sided control.

Figure 35 shows the value of the barrier indicator for the two cases. The rapidity with which barrier surfaces approach in the terminal portion of the game is evident from the spike in the barrier indicator around 2.2 TU.

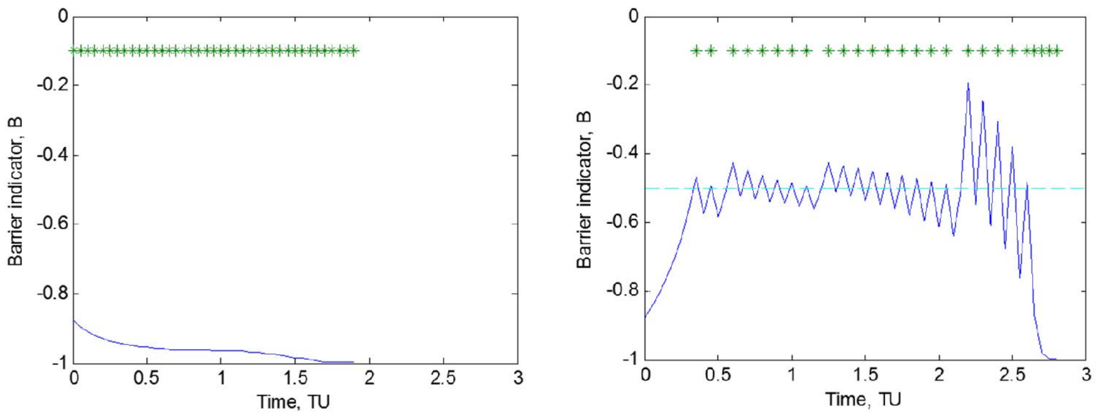


Figure 35 Barrier indicator vs. time, for pure two-sided control and hybrid surreptitious control. Asterisks indicate time steps where pursuer used two-sided control.

Thermostat control

Thermostat control can be implemented by an evader, to ensure that any intruders do not achieve an interception game value below a certain level; or by a pursuer, to "shadow an evader at a distance", without initiating interception.

Thermostat controls are simpler than surreptitious controls, and do not require an assumption about the other player's present controls, apart from the assumption about acceleration capability used in the differential game.

In thermostat controls, some critical threshold for the game value is established in advance. If the game value falls beneath this threshold (for the evader) or above it (for the pursuer), the player will employ two-sided controls until the game value is again on the desired side of the threshold.

Shown for the evader, the thermostat control strategy is:

$$\boldsymbol{T}_E(\boldsymbol{x}) = \begin{cases} \boldsymbol{u}_{2S_E}^*, & V < V_{crit_E} \\ \boldsymbol{u}_{1S_E}^* & \text{otherwise} \end{cases}$$

with $\boldsymbol{u}_{1S_E}^*$, $\boldsymbol{u}_{2S_E}^*$ and V_{crit_E} being the evader's optimal one-sided and two-sided controls, and choice of game value threshold, respectively. Corresponding notation will be used for the pursuer.

To demonstrate the effects clearly, we will show thermostat controls implemented by the pursuer and the evader simultaneously. For the pursuer, it is also necessary to manage the barrier surface, in a manner similar to that shown in surreptitious pursuit (there may be scenarios where the evader ought to manage a barrier surface, but these have not been encountered yet):

$$\boldsymbol{T}_P(\boldsymbol{x}) = \begin{cases} \boldsymbol{u}^*_{2s_P}, & V < V_{crit_P} \text{ or } B > B_{crit} \\ \boldsymbol{u}^*_{1s_P}, & \text{otherwise} \end{cases}$$

Our findings suggest that often, a barrier surface is eventually encountered that requires the pursuer to either avoid it by selecting optimal controls until interception, or cross the barrier surface. Initial conditions for the example shown were chosen somewhat carefully, so that this point did not occur in the time span of the example. This was done so that the effects of both players' thermostat controls would be readily apparent.

Initial conditions for the example are shown in Table 10. The pursuer uses $V_{crit_P} = 2.2\ TU$ and the evader uses $V_{crit_E} = 2.0\ TU$.

Table 10 Initial conditions for hybrid thermostat control example

	Initial position (x,y,z)			Initial velocity (x,y,z)			Thrust/mass
	DU			DU/TU			DU/TU^2 (g)
Pursuer	1.063	0	0	0	0.970	0	0.1
Evader	1.046	0.184	0.046	-0.168	0.954	0.042	0.05

For comparison, trajectory solutions for both purely passive control and thermostat control are shown in Figure 36. The trajectory results are not much different, because we have selected a fairly stable trajectory. A visible difference is that the pursuer finishes closer to the evader in the thermostat control case, which is consistent with the observation that the game value obtained with passive controls violated the pursuer's control threshold. The important point is that the thermostat controls by both pursuer and evader are effective in maintaining the game value. Game value vs. time is shown in Figure 37, for passive control and thermostat control.

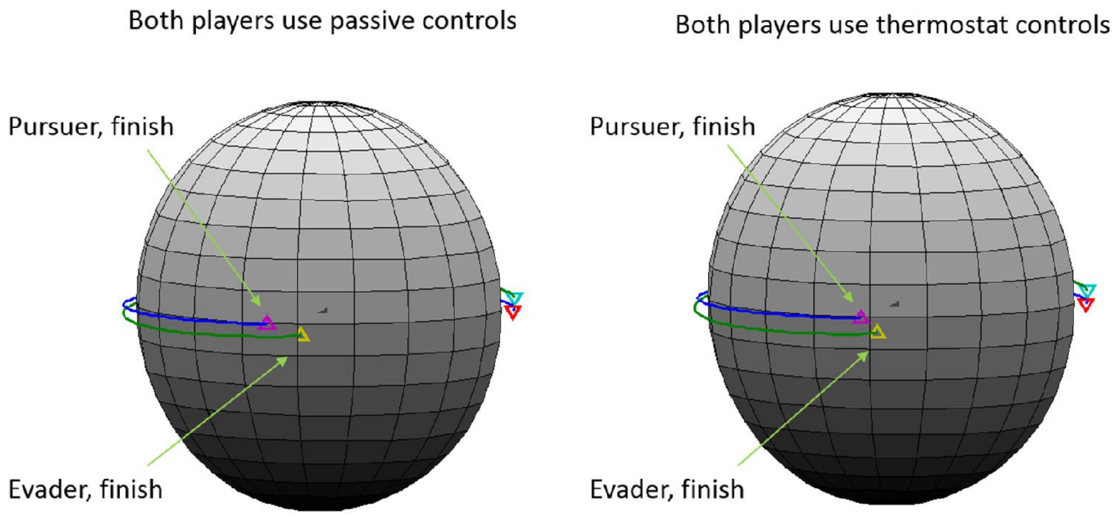

Both players use passive controls
Pursuer, finish
Evader, finish
Both players use thermostat controls
Pursuer, finish
Evader, finish

Figure 36 Trajectory solutions for thermostat initial conditions, with both players using passive control (left) and thermostat control (right)

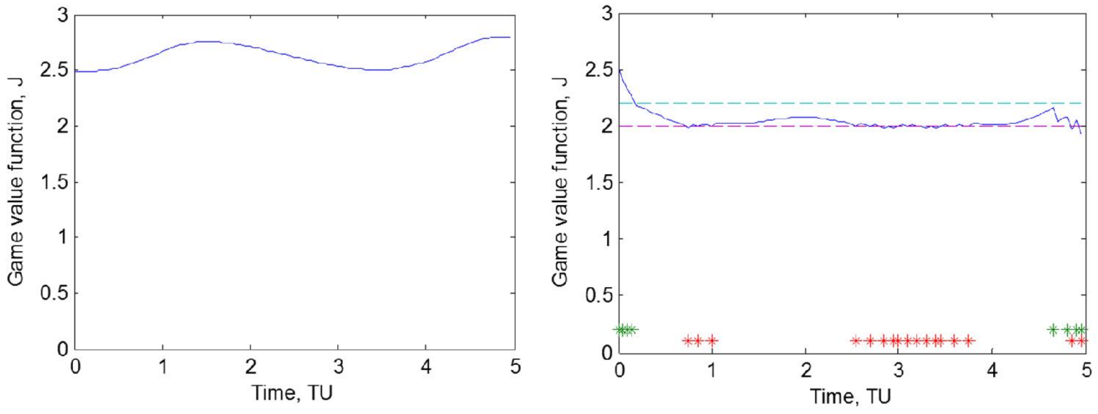


Figure 37 Game value vs. time, with zero control (left) and both players performing thermostat control (right). The asterisks indicate two-sided control for the pursuer (higher) and the evader (lower).

The barrier indicator vs. time is shown in Figure 38. The evader's controls during mid-trajectory drive proximity to the barrier surface such that late in the trajectory, the pursuer must respond to the barrier indicator, rather than the game value.

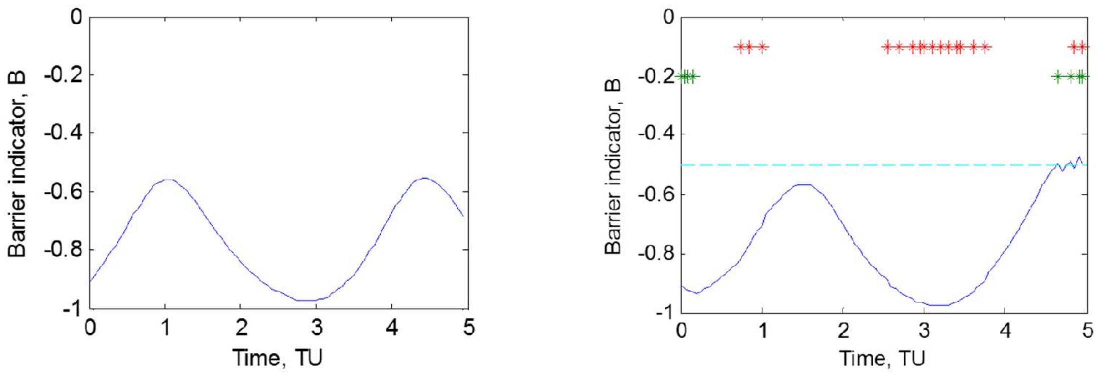


Figure 38 Barrier indicator vs. time, with zero control (left) and both players performing thermostat control (right). The asterisks indicate two-sided control for the pursuer (higher) and the evader (lower).

Discussion

The fundamental concept we present here is a hybrid one-sided/two-sided optimal controller where the switching function is based upon the value of the differential game. The notion is grounded in the application of differential game-based control to systems which have a primary, non-adversarial mission apart from the differential game; for example, a spacecraft that must perform a mission on orbit, such as communications, while also avoiding threats as they arise. The use of a hybrid one-sided/two-sided controller is a natural idea for such a system, and the idea of using the value of the game as a switching function seems further in line with one’s intuition about how such a system should operate.

Yet having identified the concept, it seems possible that even a system performing a solely adversarial mission, such as a spacecraft focused solely on pursuit, may find use for such a hybrid control scheme. This is because the hybrid control allows the adversarial system to mask its intentions, by selecting passive or one-sided controls in such a way that its adversarial objective is advanced.

Despite the limitations in the sensitivity method observed in our research efforts, we feel they may have a use in the hybrid controller, which relies upon very fast updating of the game value function. However, we expect that high-powered techniques such as random search and collocation methods will be required as well, to achieve system robustness. Management of the barrier surface, for example by increasing the sophistication of the techniques used by the pursuer here, may also be a useful area for further study.

CHAPTER VII

FUTURE WORK

A future work topic suggested by these examinations is the identification of computational tools and methods suitable to the various problems identified here: generating an initial solution, incorporating altitude constraints, and computing rapid updates to the solution, for the purposes of feedback control or hybrid one-sided/two-sided control. We provide the following comments on the assortment of solution methods that may be applicable to this problem, while acknowledging that the author has only strong first-hand experience with sensitivity methods, limited experience with genetic algorithms, and no first-hand experience with collocation methods.

First, sensitivity methods are not sufficiently robust to act as a stand-alone solution technique. Their principal shortcoming is the inability to cross discontinuous regions of solution space. A secondary liability is the fact that the length of the homotopy solution curve cannot be known in advance, and it may not proceed directly to the desired solution, which adds to computation time. However, their great speed advantage in many cases may merit their use, if a way can be determined to have other solution techniques ready when sensitivity methods are not suitable.

One notion that may be useful is a sensitivity-based random search. In this technique, an initial solution is generated at random, and a gradient-based search is then performed using that solution as a starting point. In general, if the initial solution that is in the same region of the solution space as the correct solution (i.e. the two are not separated by barrier surfaces) and shooting function error in that locale is positive definite about the true solution, the correct solution will be found by the gradient search. If not, the process is repeated with a new random initial solution. This process could be combined with a genetic algorithm, and it might be found that a speed-up resulted as compared to the genetic algorithm by itself.

Third, barrier surfaces are fatal for sensitivity-based methods, but they are also expected to be problematic for genetic algorithms as well as genetic algorithm-collocation methods. If a state space configuration is considered that happens to lie just on the other side of a barrier surface (the side corresponding to longer interception time), the genetic algorithm may tend to furnish solutions that sub-optimally cross the barrier surface, since its accuracy in general is low. In this event, the genetic algorithm may arrive at a much shorter, but incorrect, trajectory solution that is discontinuous from the true trajectory solution, due to the barrier surface. If the genetic algorithm passed its inaccurate solution to a collocation method, it is likely that the collocation method would have no way of obtaining the correct trajectory, and instead would either converge on a sub-optimal solution, or indicate failure. If the former occurred, the consequence would be (for example) a computed interception that would actually result in a near miss.

In light of these comments, we feel there are useful possibilities afforded by the combination of sensitivity methods, random search methods, and collocation methods:

- Find robust ways of moving between random search, collocation, and sensitivity solutions that take advantage of the speed of the sensitivity method, where possible. (See the next point.)

- Use of sensitivity methods to "look forward" and identify upcoming barrier surfaces. Use this information to either avoid the barrier surfaces, or to begin the computational search for a solution beyond the approaching barrier surface.
- Use sensitivity methods to validate the accuracy of the collocation solution, since a sensitivity method has the most rapid convergence rate of the methods, for a sub-optimal solution in the vicinity of an optimal solution. If there is no nearby optimal solution, the sensitivity method will quickly identify this, which will motivate the search for a new solution.

It is hoped that these methods may lead toward the real-time implementation of optimal control strategies that would ordinarily have only been applied off-line or for analysis purposes. It seems likely, however, that a synthesis of computational tools, rather than a single silver bullet, will be required.

CHAPTER VIII

CONCLUSIONS

In this work, we have provided a broad examination of the application of sensitivity methods to the orbital PE problem. In so doing, we have made several new contributions. We have developed an original sensitivity-based solution to the orbital PE problem, and a feedback method that overcomes the limitation accepted by prior researchers, of having to compute off-line a mesh of solutions in the entire space in which the differential game is presumed to be played.

We have demonstrated the importance of barrier surfaces in feedback control, which had been previously overlooked. We have shown a means of identifying a barrier surface, that affords at least some prior notice that the barrier surface is approaching.

We have developed a method for introducing state variable inequality constraints with a sensitivity method, and demonstrated it with the Breakwell problem. Fully implementing the method remains to be done, but our initial studies show that while the method may yet be successful for the orbital PE problem, it seems to be more subject to undesired behavior of the homotopy function than the other applications we have examined. Whether this method is practical for orbital PE remains to be seen.

Finally, we have presented the idea of hybrid one-sided/two-sided optimal controllers. These controllers are motivated by consideration of systems that must spend most of their time performing a mission involving one-sided optimization, while being on the lookout for an adversarial situation to arise, at which point they must switch to a two-sided optimal control to deal with the adversary. Spacecraft are certainly one example of such a system, but there ought to be others as well. We have shown how monitoring the game value function provides a useful means of accomplishing this hybrid switching, and that as a result, behaviors that seem intuitively useful can be achieved. This control implementation does not require sensitivity methods as opposed to any other solution technique, but it does require some means to rapidly update the game solution, for which sensitivity methods are well suited. We are excited about this line of research, and we feel further investigation is warranted both for application to spacecraft and other classes of systems.

These findings and contributions are summarized in Table 11.

Table 11 Summary of research contributions and benefits

Contribution	Improvement	Impact
Orbital PE initial solution via gravity homotopy	Less computation time than random-search methods	Online control possible
Barrier location method	Prior barrier studies limited to linear dynamics	Interception regions can be found for complex pursuit/evasion systems
Real-time control by initial conditions homotopy	Removes mesh of pre-computed solutions	Feedback controller more robust to adversary behavior
Hybrid one-sided/two-sided controllers	Integrate PE game information with spacecraft primary mission	Spacecraft can perform primary mission and avoid adversaries

Finally, a theme of this work has been to determine when sensitivity methods can offer benefits in comparison with classes of "black box" methods, such as random search and collocation methods, that employ no problem-specific knowledge and instead rely upon computational power. These two classes of methods have very complimentary sets of advantages and disadvantages. Random search and collocation methods, especially when used together, provide a general solution capability that can handle a wide range of initial condition scenarios, and can handle minimum-altitude constraints with little difficulty. On the other hand, these methods are computationally intensive, likely too much so for an onboard controller.

Here, we have shown that sensitivity methods are extremely fast—when they work. The possibility that is of interest to us now is the combination of both classes of methods, in a controller that employs sensitivity methods whenever possible, but intelligently manages its computational strategy, as well as perhaps its control strategy, to ensure that a new solution obtained from search and collocation methods is available when needed. In other words, the controller is as fast as possible, while still being robust.

Of course, it is true that computing power will continue to increase onboard spacecraft as on other platforms. Perhaps it will one day be acceptable to run a genetic algorithm of the sort applied to this problem in a real-time controller. (Indeed, the GA is well suited to a parallel computing approach.) However, our view is that engineering problems have a way of using up all available computing power. In concrete terms, the state space will be increased, multiple players will be considered, and so on. We suggest that sensitivity methods, with their fundamental speed advantages, will continue to be relevant to the solution of challenging engineering problems involving nonlinearity and high dimensionality.

REFERENCES

[1] Schaub, H., and Junkins, J. L., *Analytical Mechanics of Space Systems*, 1st ed., AIAA, Reston, VA, 2003, pp. 530-533.

[2] Rosenwasser, E., and Yusupov, R., *Sensitivity of Automatic Control Systems*, Chapman & Hall/CRC, Boca Raton, FL, 2000.

[3] Kim, D., Turner, J. D., and Junkins, J. L., "Optimal Actuator Failure Control Using a Homotopy Method", *Journal of Guidance, Control, and Dynamics*, 2014 (Accepted)

[4] Majji, M., Turner, J. D., and Junkins, J. L., "Solution of Two-Point Boundary-Value Problems using Lagrange Implicit Function Theorem", *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 5, 2009, pp. 1684–1687.

[5] Ghosh, P., Conway, B. A., "Near-Optimal Feedback Strategies Synthesized Using a Spatial Statistical Approach", *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 4, 2013, pp. 905-919.

[6] Anderson, G. M., and Grazier, V. W., "A Closed-Form Solution for the Barrier in Pursuit-Evasion Problems Between Two Low Thrust Orbital Spacecraft and Its Application," *Aerospace Sciences Meeting*, AIAA, New York, Jan. 1975.

[7] Isaacs, R., *Differential Games*, Wiley, New York, 1965

[8] Pontani, M. and Conway, B. A., "Numerical Solution of the Three-Dimensional Orbital Pursuit-Evasion Game", *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 2, 2009, pp. 474-487.

[9] Shen, E., Pham, K., Blasch, E., Chen, H., and Chen, G., “Pursuit-Evasion Orbital Game for Satellite Interception and Collision Avoidance,” *Proceedings of SPIE*, Vol. 8044, *Sensors and Systems for Space Applications IV*, 80440B, 2011. doi: 10.1117/12.882903

[10] Turetsky, V., and Shinar, J., “Missile Guidance Laws Based on Pursuit-Evasion Game Formulations”, *Automatica*, Vol. 39, 2003, pp. 607–618. doi:10.1016/S0005-1098(02)00273-X

[11] Shima, T., Shinar, J., and Weiss, H., “New Interceptor Guidance Law Integrating Time Varying and Estimation Delay Models”, *Journal of Guidance, Control, and Dynamics*, Vol. 26, No. 2, 2003, pp. 295–303.

[12] Trottemant, E. J., Scherer, C. W., Weiss, M., and Vermeulen, A., “Robust Missile Feedback Control Strategies”, *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 6, 2010, pp. 1837–1846. doi:10.2514/1.48844

[13] Johnson, P. A., “Numerical Solution Methods for Differential Game Problems”, M.S. Thesis, Dept. of Aeronautics and Astronautics, Massachusetts Inst. Of Tech., Cambridge, MA, June 2009

[14] Wong, R. E., “Some Aerospace Differential Games”, *Journal of Spacecraft and Rockets*, Vol. 4, No. 11, 1967, pp. 1460-1465.

[15] Kelley, H. J., Cliff, E. M., and Lutze, F. H., “Pursuit–Evasion in Orbit,” *Journal of the Astronautical Sciences*, Vol. 29, No. 3, 1981, pp. 277–288.

[16] Menon, P. K. A., Calise, A. J., and Leung, S. K. M., “Guidance laws for spacecraft pursuit-evasion and rendezvous”, *AIAA Guidance Navigation and Control Conference*, 88-4134-CP, Minneapolis, MN, 1988. doi: 10.2514/MGNC88

[17] Hafer, W., Reed, H., Turner, J., and Pham, K., “Initial and Feedback Solutions for Orbital Pursuit-Evasion Using a Homotopy Method,” *American Astronautical Society Guidance and Control Conference*, No. AAS 14-056, Vol. 151, American Astronomical Society, San Diego, CA, 2014, pp. 259-270.

[18] Allgower, E., and Georg, K., *Numerical Continuation Methods: An Introduction*, Springer-Verlag, Berlin, 1990

[19] Zhulin, S. S., “Homotopy Method for Finding Extremals in Optimal Control Problems”, *Differential Equations*, Vol. 43, No. 11, 2007, pp. 1495–1504.

[20] Bulirsch, R., Montrone, F., and Pesch, H. J., “Abort Landing in the Presence of Windshear as a Minimax Optimal Control Problem, Part 2: Multiple Shooting and Homotopy,” *Journal of Optimization Theory and Applications*, Vol. 70, No. 2, 1991, pp. 223–254.

[21] Gergaud, J., and Martinon, P., “Homotopy Method for Minimum Consumption Orbit Transfer Problem”, *Control, Optimisation and Calculus of Variations*, Vol. 12, No. 2, 2006, pp. 294–310.

[22] Chow, S. N., Mallet-Paret, J., and Yorke, J. A.,“Finding Zeros of Maps: Homotopy Methods That Are Constructive with Probability One,” *Mathematics of Computation*, Vol. 32, No. 143, 1978, pp. 887–899. doi:10.1090/S0025-5718-1978-0492046-9

[23] Eaves, B. C., and K. Schmedders, “General Equilibrium Models and Homotopy Methods,” *Journal of Economic Dynamics and Control*, Vol. 23, No. 9-10, 1999, 1249–79.

[24] Bryson, A. E., and Ho, Y. C., *Applied Optimal Control*, Blaisdell Publishing Company, Waltham, MA, 1969

[25] Pesch, H. J., “Real-Time Computation of Feedback Controls for Constrained Optimal Control Problems, Part 1: Neighboring Extremals,” *Optimal Control Applications and Methods*, Vol. 10, No. 2, 1989, pp. 129–145.

[26] Malanowski, K., and Maurer, H., “Sensitivity Analysis for Parametric Control Problems with Control State Constraints”, *Computational Optimization and Applications*, Vol. 5, No. 3, 1996, pp. 253–283.

[27] Hiskens, I. A., and Pai, M. A., “Trajectory Sensitivity Analysis of Hybrid Systems”, *IEEE Transactions on Circuits and Systems I*, Vol. 47, No. 2, 2000, pp. 204-220

[28] Ross, I. M., and Fahroo, F., “Legendre Pseudospectral Approximations of Optimal Control Problems”, *Lecture Notes in Control and Information Sciences*, Vol. 295, Springer–Verlag, New York, 2004, pp. 327–342.

[29] Conway, B. A., “A Survey of Methods Available for the Numerical Optimization of Continuous Dynamical Systems”, *Journal of Optimization Theory and Applications*, Vol. 152, No. 2, 2012, pp. 271–306.

[30] Cawdery, P. H. “Toward the Definition of Escape and Capture Regions for a Two Aircraft Pursuit-Evasion Game”, M.S. Thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, June 1973.

[31] von Stryk, O., “Numerical Solution of Optimal Control Problems by Direct Collocation”, *Optimal Control*, edited by R. Bulirsch, A. Miele, J. Stoer, and K. H. Well, Vol. 111, International Series in Numerical Mathematics, Birkhauser, Basel, Germany, 1993, pp. 129–143.

APPENDIX

Solutions and computations not necessary to the main discussion are included here.

Jacobian of state/costate dynamics matrix (unconstrained problem)

The dynamics vector for either player is below.

$$\boldsymbol{F}_i = \frac{d}{dt}(\boldsymbol{Y}_i) = \begin{bmatrix} \boldsymbol{v}_i^T & \left(-\frac{\mu}{(\boldsymbol{r}_i^T\boldsymbol{r}_i)^{3/2}}\boldsymbol{r}_i + \boldsymbol{u}_i\right)^T & -\left[G(\boldsymbol{r}_i)^T\boldsymbol{\Lambda}_{v_i}\right]^T & -\boldsymbol{\Lambda}_{r_i}^{\mathrm{T}} \end{bmatrix}^T$$

The Jacobian of the dynamics vector is below. Each entry is a 3x3 block.

$$\frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{Y}_i} = \begin{bmatrix} \boldsymbol{0} & I & \boldsymbol{0} & \boldsymbol{0} \\ G(\boldsymbol{r}) & \boldsymbol{0} & \boldsymbol{0} & \dfrac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{\Lambda}_{v_i}} \\ -\dfrac{\partial}{\partial \boldsymbol{r}}\left[G(\boldsymbol{r}_i)^T\boldsymbol{\Lambda}_{v_i}\right] & \boldsymbol{0} & \boldsymbol{0} & -G(\boldsymbol{r}_i) \\ \boldsymbol{0} & \boldsymbol{0} & -I & \boldsymbol{0} \end{bmatrix}$$

$$\frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{\Lambda}_{v_i}} = \pm T_{m_i}\left[\left(\boldsymbol{\Lambda}_{v_i}^T\boldsymbol{\Lambda}_{v_i}\right)^{-1/2} I - \left(\boldsymbol{\Lambda}_{v_i}^T\boldsymbol{\Lambda}_{v_i}\right)^{-3/2}\left(\boldsymbol{\Lambda}_{v_i}\boldsymbol{\Lambda}_{v_i}^T\right)\right] \quad ; \quad \pm = \begin{cases} -, & \text{pursuer} \\ +, & \text{evader} \end{cases}$$

$$\frac{\partial}{\partial \boldsymbol{r}_i}\left[G(\boldsymbol{r}_i)^T\boldsymbol{\Lambda}_{v_i}\right] = 3\mu\left\{\left[-5\frac{(\boldsymbol{r}_i\boldsymbol{r}_i^T)}{(\boldsymbol{r}_i^T\boldsymbol{r}_i)^{\frac{7}{2}}} + \frac{I}{(\boldsymbol{r}_i^T\boldsymbol{r}_i)^{\frac{5}{2}}}\right]\left(\boldsymbol{r}_i^T\boldsymbol{\Lambda}_{v_i}\right) + \frac{(\boldsymbol{r}_i\boldsymbol{\Lambda}_{v_i}^T + \boldsymbol{\Lambda}_{v_i}\boldsymbol{r}_i^T)}{(\boldsymbol{r}_i^T\boldsymbol{r}_i)^{5/2}}\right\}$$

Gravity-less solution

The one-player, zero-gravity solution, in which an optimal (constant) control direction is sought to drive the player to the origin in minimum time, is found as follows. As stated earlier, the two-player zero-gravity solution is found with the same mechanism, since a relative motion frame can be used in the zero-gravity system. Here, we drop the player-specific notation, and use notation for a single player that must be driven to the origin.

The method begins with a coordinate system transformation placing the player (pursuer) on the positive x-axis of the transformed coordinate system, with the player's velocity vector lying in the x-y plane. With $\boldsymbol{r}$ and $\boldsymbol{v}$ the player's initial position and velocity, the trans-formation matrix is obtained as:

$$\hat{\boldsymbol{r}} = \frac{\boldsymbol{r}}{|\boldsymbol{r}|} \quad ; \quad \hat{\boldsymbol{v}} = \frac{\boldsymbol{v}}{|\boldsymbol{v}|}$$

$$\hat{\boldsymbol{t}}_1 = \hat{\boldsymbol{r}} \quad ; \quad \hat{\boldsymbol{t}}_3 = \frac{\hat{\boldsymbol{r}} \times \hat{\boldsymbol{v}}}{|\hat{\boldsymbol{r}} \times \hat{\boldsymbol{v}}|} \quad ; \quad \hat{\boldsymbol{t}}_2 = \hat{\boldsymbol{t}}_3 \times \hat{\boldsymbol{t}}_1 \quad ; \quad |\hat{\boldsymbol{r}} \times \hat{\boldsymbol{v}}| \neq 0$$

$$[C] = [\hat{\boldsymbol{t}}_1 \quad \hat{\boldsymbol{t}}_2 \quad \hat{\boldsymbol{t}}_3]$$

where $[\boldsymbol{a}]_F = [C][\boldsymbol{a}]_N$ for all vectors $\boldsymbol{a}$, with $_N$ denoting the original frame and $_F$ the resultant frame.

If $|\hat{\boldsymbol{r}} \times \hat{\boldsymbol{v}}| = 0$, the player's initial motion is in a line, and the optimal control is trivial.

In this frame, since initial motion is in the x-y plane, thrust to bring the player to the origin will clearly also act in the x-y plane. An optimal angle γ for thrust direction must be chosen, as shown in Figure 39.

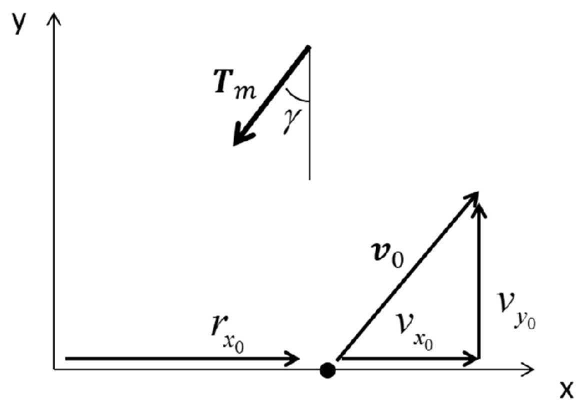


Figure 39 Zero-gravity, one-player motion geometry in transformed frame, at initial time

With reference to the quantities in Figure 39, the equations of motion for the player are:

$$r_x(t,\gamma) = r_{x_0} + v_{x_0}t + \frac{1}{2}(T_m \sin\gamma)t^2 \quad ; \quad r_y(t,\gamma) = v_{y_0}t - \frac{1}{2}(T_m \cos\gamma)t^2$$

We wish to select a thrust angle γ such that the player is returned to the x-axis and the y-axis at the same time, t_f. We pick γ to satisfy:

$$r_x(t_f,\gamma) = 0 \quad ; \quad r_y(t_f,\gamma) = 0$$

Substituting this constraint into the equations of motion at t_f yields:

$$v_{y_0}^2 \sin\gamma + v_{x_0}v_{y_0}\cos\gamma + \frac{1}{2}r_{x_0}T_{m_p}\cos^2\gamma = 0$$

This equation can be expanded with $\sin\gamma = \pm\sqrt{1-\cos^2\gamma}$ and solved as a quartic equation in $\cos\gamma$. Of as many as eight possible solutions (four from the quartic solution, each with one possible cosine ambiguity), solutions with imaginary content, and solutions with high position error with respect to the origin at time $t_f = \frac{2v_{y_0}}{T_m \cos\gamma}$ are discarded. Sometimes two solutions may remain, in which case the solution with lowest t_f is selected.

This yields the required control in the transformed frame. The reverse frame transformation on the thrust vector is then performed to yield controls in the original frame.

Recreating the costates

To complete the zero-gravity solution, we must reproduce the trajectories of the costates, using the trajectories and controls obtained above. As discussed earlier, $\boldsymbol{u}_i^*$ points in the direction of $\pm\boldsymbol{\Lambda}_{v_i}$, with (-) for pursuer and (+) for evader. And in the zero gravity case, $\boldsymbol{u}_P^* = \boldsymbol{u}_E^*$. This gives the directions of $\Lambda_{v_i}(t_0) = \Lambda_{v_i}(t)$. The costate dynamics and final-time conditions are:

$$\dot{\boldsymbol{\Lambda}}_{r_i} = \mathbf{0} \;\; ; \;\; \dot{\boldsymbol{\Lambda}}_{v_i} = -\boldsymbol{\Lambda}_{r_i} \;\; ; \;\; \boldsymbol{\Lambda}_{v_i}(t_f) = \mathbf{0}$$

The position costates $\boldsymbol{\Lambda}_{r_i}$ are constant, and must be chosen to drive the velocity costates to zero at the final time. This defines all the costates up to a positive scale parameter, which is constrained by:

$$H(t_f) + 1 = 0 \;\; ; \;\; \dot{H} = 0$$

We can enforce the Hamiltonian constraint at the initial time to find suitable costate initial values.

Sensitivity equations for barrier indicator

To make maximum use of existing sensitivity computations, when driving the barrier indicator B to zero we instead employ a correction-based method to drive the final-time Hamiltonian H to zero. However, we rescale the costates at each correction step, so that $H = 0$ can only be achieved by making $\boldsymbol{\Lambda}_{r_P}$ normal to $\Delta\boldsymbol{v}$.

The barrier is located using a Newtonian correction method; i.e. a corrector step is used without the predictor step. The barrier shooting function $\boldsymbol{S}_b$ is:

$$\boldsymbol{S}_b = \begin{bmatrix} \boldsymbol{r}_P - \boldsymbol{r}_E & \boldsymbol{\Lambda}_{r_P} + \boldsymbol{\Lambda}_{r_E} & \boldsymbol{\Lambda}_{v_P} & \boldsymbol{\Lambda}_{v_E} & H \end{bmatrix}\Big|_{t_f}^{T}$$

which is identical to the optimal interception shooting function $\boldsymbol{S}$ except for the last entry being H rather than $H + 1$. The unknowns are $\boldsymbol{U}_b = \begin{bmatrix} \boldsymbol{U} \\ r_{1E_0} \end{bmatrix}$, where $\boldsymbol{U}$ is the vector of unknowns for optimal interception, and r_{1E_0} is the x-component of the evader position. r_{1E_0} is used because we have chosen this to be the element we vary as we search for the nearest barrier trajectory. The correction elements at each correction step are:

$$\boldsymbol{U}_{b_{i+1}} = \boldsymbol{U}_{b_i} - A^{+}\boldsymbol{S}_b\left(\boldsymbol{U}_{b_i}\right)$$

$$A = \begin{bmatrix} \dfrac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} & \dfrac{\partial \boldsymbol{S}}{\partial r_{1E_0}} \end{bmatrix}$$

$$\frac{\partial \boldsymbol{S}}{\partial r_{1E_0}} = \begin{bmatrix} -\dfrac{\partial \boldsymbol{r}_E^T}{\partial r_{1E_0}} & \dfrac{\partial \Lambda_{r_E}^T}{\partial r_{1E_0}} & \boldsymbol{0} & \dfrac{\partial \Lambda_{v_E}^T}{\partial r_{1E_0}} & \dfrac{\partial H}{\partial r_{1E_0}} \end{bmatrix}^T$$

$$\frac{\partial H}{\partial r_{1E_0}} = \frac{\partial \boldsymbol{\Lambda}_E}{\partial r_{1E_0}}^T \boldsymbol{f}_E + \boldsymbol{\Lambda}_E^T \frac{\partial \boldsymbol{f}_E}{\partial \boldsymbol{Y}_E}\frac{\partial \boldsymbol{Y}_E}{\partial r_{1E_0}}$$

Barrier trajectory identification for simplified case

Here, the sensitivity method for locating barrier trajectories is applied to a system without gravity, and a fixed evader (thus, a one-player or one-sided optimal control problem). Since the evader is stationary, it will be referred to as the target. The barrier is also verified analytically, to provide a check on the validity of the sensitivity method.

In the problem, the pursuer is initially heading toward the target in the y-direction. The initial conditions are shown in Table 12. Barrier trajectories are sought as target position is varied in the x- and z-directions.

Table 12 Initial conditions for hybrid surreptitious control example

	Initial position (x,y,z)			Initial velocity (x,y,z)			Thrust/mass
	DU			DU/TU			DU/TU^2 (g)
Pursuer	1	0	0	0	1	0	0.1
Target	1	1	0	0	0	0	0*
* Target is fixed: a one-sided control problem.							

The resultant barrier trajectories are shown in Figure 40. The maximum excursion in the x-direction achievable by the pursuer, which corresponds to the right-most barrier trajectory, is between $0.050\ DU$ and $0.051\ DU$. (Note that the pursuer starts at 1.00 DU). The exact excursion of this barrier trajectory will now be found analytically to be consistent with the above.

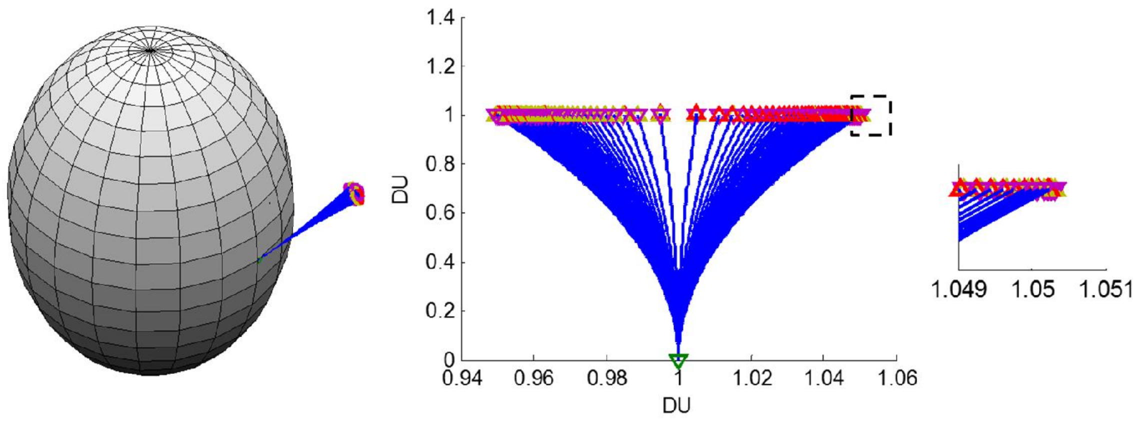


Figure 40 Barrier trajectories for simplified, gravity-less case: left, in planet view; center, plotted in x-y plane; right, zoomed in on largest excursion in (+x) direction.

The analytical solution is based on a variational argument. The barrier trajectory in the (+x) direction represents the furthest point in the (+x) direction the pursuer can reach by selection of optimal controls. Referring to the geometry in Figure 41, the pursuer must select a control angle ϕ to maximize position excursion in the x-direction, $r_x(t_f)$. t_f is the time t when excursion in the y-direction $r_y(t)$ equals r_{y_0}, the initial separation from the target.

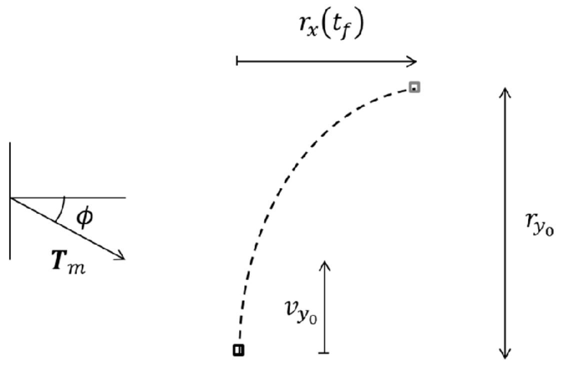


Figure 41 Geometry for barrier analytical study in gravity-less case

The excursion in the y-direction at t_f is found as:

$$r_y(t_f) = v_{y_0} t_f - \frac{1}{2} T_m sin\phi t_f^2 = r_{y_0}$$

which gives t_f as:

$$t_f = \frac{v_{y_0} - \sqrt{v_{y_0}^2 - 2T_m r_{y_0} sin\phi}}{T_m sin\phi}$$

where the smaller of the two times, representing the first arrival at the target's y-coordinate, is chosen.

The excursion in the x-direction is then:

$$r_x(t_f) = \frac{1}{2} T_m cos\phi t_f^2 = \frac{1}{2} T_m cos\phi \left(\frac{v_{y_0} - \sqrt{v_{y_0}^2 - 2T_m r_{y_0} sin\phi}}{T_m sin\phi} \right)^2$$

Values of ϕ are sought for which $\frac{\partial r_x(t_f)}{\partial \phi} = 0$. Manipulation gives:

$$\frac{\partial r_x(t_f)}{\partial \phi} = \left(\frac{-1}{2T_m sin\phi} - \frac{\cos^2\phi}{T_m \sin^3\phi}\right)\left[v_{y_0}^2 - 2v_{y_0} rad(\phi)^{\frac{1}{2}} + rad(\phi)\right]$$

$$- \frac{cos\phi}{T_m \sin^2\phi}\left(1 - \frac{v_{y_0}}{rad(\phi)^{\frac{1}{2}}}\right)\left(T_m r_{y_0} cos\phi\right)$$

with $rad(\phi) = v_{y_0}^2 - 2T_m r_{y_0} sin\phi$.

For the present problem, parameter values are $r_{y_0} = 1\,DU$, $v_{y_0} = 1\,DU/TU$, $T_m = 0.1\,DU/TU^2$. $\frac{\partial r_x(t_f)}{\partial \phi} = 0$ is approximately obtained for $\phi = 0.10007\,rad$. This control angle results in $r_x(t_f) = 0.0503\,DU$. This is in agreement with the results from the sensitivity approach to locating the barrier shown above.

Sensitivity equations for altitude constraints

This section deals with sensitivities for the state- and velocity-level waypoint constrained orbital PE problem. Beyond the constraints and unknowns of the original problem, we add unknowns α_1, α_2, t_1, waypoint constraints $C = 0$ and $\dot{C} = 0$ at time t_1 (free), and the constant Hamiltonian constraint $\Delta H = 0$.

We have augmented unknowns vector $\boldsymbol{U}_a$ and shooting function $\boldsymbol{S}_a$, given the previous unknowns and shooting function $\boldsymbol{U}$ and $\boldsymbol{S}$:

$$\boldsymbol{U}_a = [\boldsymbol{U}^T \quad \alpha_1 \quad \alpha_2 \quad t_1]^T$$

$$\boldsymbol{S}_a = [\boldsymbol{S}^T \quad C \quad \dot{C} \quad \Delta H]^T$$

We must find the following quantities:

$$\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{\alpha}}, \frac{\partial \boldsymbol{S}}{\partial t_1}$$

$$\frac{\partial C}{\partial \boldsymbol{U}_a} = \begin{bmatrix} \frac{\partial C}{\partial \boldsymbol{\Lambda}_{P_0}} & \frac{\partial C}{\partial \boldsymbol{\Lambda}_{E_0}} & \frac{\partial C}{\partial t_f} & \frac{\partial C}{\partial \boldsymbol{\alpha}} & \frac{\partial C}{\partial t_1} \end{bmatrix}$$

$$\frac{\partial \dot{C}}{\partial \boldsymbol{U}_a} = \begin{bmatrix} \frac{\partial \dot{C}}{\partial \boldsymbol{\Lambda}_{P_0}} & \frac{\partial \dot{C}}{\partial \boldsymbol{\Lambda}_{E_0}} & \frac{\partial \dot{C}}{\partial t_f} & \frac{\partial \dot{C}}{\partial \boldsymbol{\alpha}} & \frac{\partial \dot{C}}{\partial t_1} \end{bmatrix}$$

$$\frac{\partial \Delta H}{\partial \boldsymbol{U}_a} = \begin{bmatrix} \frac{\partial \Delta H}{\partial \boldsymbol{\Lambda}_{P_0}} & \frac{\partial \Delta H}{\partial \boldsymbol{\Lambda}_{E_0}} & \frac{\partial \Delta H}{\partial t_f} & \frac{\partial \Delta H}{\partial \boldsymbol{\alpha}} & \frac{\partial \Delta H}{\partial t_1} \end{bmatrix}$$

For $\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{\alpha}}$ and $\frac{\partial \boldsymbol{S}}{\partial t_1}$, all elements of $\boldsymbol{S}$ are trivially formed from elements of the state sensitivity matrices $\frac{\partial \boldsymbol{Y}_i}{\partial \boldsymbol{\alpha}}$ and $\frac{\partial \boldsymbol{Y}_i}{\partial t_1}$, except for $\frac{\partial H(t_f)}{\partial \boldsymbol{\alpha}}$ and $\frac{\partial H(t_f)}{\partial t_1}$. These are found as (with all sensitivities and dynamics evaluated at time t_f):

$$\frac{\partial H(t_f)}{\partial \boldsymbol{\alpha}} = \frac{\partial H(t_f)}{\partial \boldsymbol{Y}_P}\frac{\partial \boldsymbol{Y}_P}{\partial \boldsymbol{\alpha}} \quad ; \quad \frac{\partial H(t_f)}{\partial t_1} = \frac{\partial H(t_f)}{\partial \boldsymbol{Y}_P}\frac{\partial \boldsymbol{Y}_P}{\partial t_1}$$

with

$$\frac{\partial H(t_f)}{\partial \boldsymbol{Y}_P} = \Lambda_P^T \frac{\partial \boldsymbol{f}_P}{\partial \boldsymbol{Y}_P} + [\boldsymbol{0} \quad \boldsymbol{f}_P^T]$$

Note that sensitivities of the evader states and costates to $\boldsymbol{\alpha}$ and t_1 are zero.

This leaves the sensitivities of C, $\dot{C}$ ΔH, all evaluated at t_1. The sensitivities of each of these quantities to evader costates, and final time t_f, are zero. For the rest, we have:

$$\frac{\partial C}{\partial \boldsymbol{\Lambda}_{P_0}} = \frac{\partial C}{\partial \boldsymbol{Y}_P}\frac{\partial \boldsymbol{Y}_P}{\partial \boldsymbol{\Lambda}_{P_0}} \quad ; \quad \frac{\partial C}{\partial \boldsymbol{\alpha}} = \frac{\partial C}{\partial \boldsymbol{Y}_P}\frac{\partial \boldsymbol{Y}_P}{\partial \boldsymbol{\alpha}} \quad ; \quad \frac{\partial C}{\partial t_1} = \frac{\partial C}{\partial \boldsymbol{Y}_P}\dot{\boldsymbol{Y}}_P(t_1^-)$$

$$\frac{\partial \dot{C}}{\partial \boldsymbol{\Lambda}_{P_0}} = \frac{\partial \dot{C}}{\partial \boldsymbol{Y}_P}\frac{\partial \boldsymbol{Y}_P}{\partial \boldsymbol{\Lambda}_{P_0}} \quad ; \quad \frac{\partial \dot{C}}{\partial \boldsymbol{\alpha}} = \frac{\partial \dot{C}}{\partial \boldsymbol{Y}_P}\frac{\partial \boldsymbol{Y}_P}{\partial \boldsymbol{\alpha}} \quad ; \quad \frac{\partial \dot{C}}{\partial t_1} = \frac{\partial \dot{C}}{\partial \boldsymbol{Y}_P}\dot{\boldsymbol{Y}}_P(t_1^-)$$

with

$$\frac{\partial C}{\partial \boldsymbol{Y}_P} = [\boldsymbol{r}^T \quad \boldsymbol{0} \quad \boldsymbol{0} \quad \boldsymbol{0}]$$

$$\frac{\partial \dot{C}}{\partial \boldsymbol{Y}_P} = [\boldsymbol{r}^T \quad \boldsymbol{v}^T \quad \boldsymbol{0} \quad \boldsymbol{0}]$$

For ΔH, first we compute $\Delta H = H(t_1^-) - H(t_1^+)$, with the result:

$$\Delta H = -T_m(\boldsymbol{\Lambda}_v^T\boldsymbol{\Lambda}_v)^{\frac{1}{2}} + T_m[\boldsymbol{V}^T\boldsymbol{V}]^{\frac{1}{2}} + \alpha_1\boldsymbol{r}^T\boldsymbol{v} + \alpha_2[\boldsymbol{v}^T\boldsymbol{v} + \mu(\boldsymbol{r}^T\boldsymbol{r})^{-1/2}]$$

with $\boldsymbol{V} = \boldsymbol{\Lambda}_v - \alpha_2\boldsymbol{r}$. Then:

$$\frac{\partial \Delta H}{\partial \boldsymbol{r}} = \alpha_1\boldsymbol{v}^T + \alpha_2\mu(\boldsymbol{r}^T\boldsymbol{r})^{-3/2}\boldsymbol{r}^T - \alpha_2 T_m(\boldsymbol{V}^T\boldsymbol{V})^{-1/2}\boldsymbol{V}^T$$

$$\frac{\partial \Delta H}{\partial \boldsymbol{v}} = \alpha_1\boldsymbol{r}^T + 2\alpha_2\boldsymbol{v}^T$$

$$\frac{\partial \Delta H}{\partial \boldsymbol{\Lambda}_v} = -T_m(\boldsymbol{\Lambda}_v^T\boldsymbol{\Lambda}_v)^{-\frac{1}{2}}\boldsymbol{\Lambda}_v^T + T_m(\boldsymbol{V}^T\boldsymbol{V})^{-\frac{1}{2}}\boldsymbol{V}^T$$

$$\frac{\partial \Delta H}{\partial \boldsymbol{Y}_P} = \begin{bmatrix} \frac{\partial \Delta H}{\partial \boldsymbol{r}} & \frac{\partial \Delta H}{\partial \boldsymbol{v}} & \boldsymbol{0} & \frac{\partial \Delta H}{\partial \boldsymbol{\Lambda}_v} \end{bmatrix}$$

This allows two of our desired results:

$$\frac{\partial \Delta H}{\partial \boldsymbol{\Lambda}_{P_0}} = \frac{\partial \Delta H}{\partial \boldsymbol{Y}_P}\frac{\partial \boldsymbol{Y}_P}{\partial \boldsymbol{\Lambda}_{P_0}} \quad ; \quad \frac{\partial \Delta H}{\partial t_1} = \frac{\partial \Delta H}{\partial \boldsymbol{Y}_P}\dot{\boldsymbol{Y}}_P(t_1^-)$$

For $\frac{\partial \Delta H}{\partial \boldsymbol{\alpha}}$, we must explicitly evaluate the partial derivative, which gives:

$$\frac{\partial \Delta H}{\partial \boldsymbol{\alpha}} = \left[\boldsymbol{r}^T\boldsymbol{v} \quad \boldsymbol{v}^T\boldsymbol{v} - \mu(\boldsymbol{r}^T\boldsymbol{r})^{-1/2} - T_m(\boldsymbol{V}^T\boldsymbol{V})^{-\frac{1}{2}}\boldsymbol{V}^T\boldsymbol{r}\right]$$

These sensitivities, and the underlying system, account for the waypoint constraints $C = 0$ and $\dot{C} = 0$ at t_1. The system as discussed here does not apply, and the sensitivities do not account, for the control constraint $\ddot{C} \leq 0$ for finite time along the constraint surface, if it is violated. To extend the system in that regard, the constraint $\ddot{C}$

should be evaluated at and after t_1, and if necessary, the Lagrange parameter $\eta > 0$ and an adjusted control should be computed. The dynamics of the costates should be adjusted accordingly for as long as the constraint is active, i.e. $\eta > 0$. As for sensitivities, the sensitivities of ΔH must now take into account the influence of the parameter η, since η is a function of the states $\boldsymbol{Y}_P(t_1^-)$ and the jump parameters $\boldsymbol{\alpha}$. The initialization of $\left.\frac{\partial \boldsymbol{Y}_P(t)}{\partial t}\right|_{t_1^+}$ is also affected. Also, the dynamics sensitivities matrix $\frac{\partial \boldsymbol{F}_P}{\partial \boldsymbol{Y}_P}$ must incorporate sensitivities to η for as long as $\eta > 0$.