OPTIMAL PURSUIT/EVASION SPACECRAFT TRAJECTORIES IN THE HILL REFERENCE FRAME

BY

JOAN MARIE STUPIK

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Aerospace Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Adviser:

Professor Bruce Conway

ABSTRACT

The pursuit/evasion game for two spacecraft is significantly simplified by being described with a linearized system of equations of motion. A pursuit/evasion game is also known as a "minimax" problem, because the objective of the pursuer is to minimize the time to capture, while the evader's goal is to maximize the time to capture. In this work, the minimax problem is solved in the Hill-Clohessy-Wiltshire (HCW) reference frame with Earth as the central body and each spacecraft uses continuous low-thrust propulsion. The thrust-pointing direction is the control for each spacecraft. Each vehicle has a finite specific impulse and therefore the mass of each vehicle decreases as propellant is consumed. Optimal closed-loop controllers were also sought, but due to the nature of the minimax problem, creating a traditional closed-loop feedback control is difficult. A closed-loop controller has been developed using a method called kriging. Results for both open and closed-loop trajectories are reported for a significant range of initial conditions and for different thrust accelerations of the two spacecraft.

To Dad, Mom, Annie, Grandpa, and Grandma, for their love and continued support.

ACKNOWLEDGMENTS

I would like to acknowledge Dr. Bruce Conway, who gave me guidance, advice, and ideas, as well as Mauro Pontani, who made the problem tractable. Thank you also to Pradipto Ghosh, without whom the kriging would have remained a mystery. I also want to acknowledge Donald Ellison, best office-mate in the world. Thank you also to all my friends who gave me emotional support: Justine, Marianne, Bindu, and Mallory.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

$\boldsymbol{u}$	Control vector
f	Cost function
$\boldsymbol{x}$	State vector
x	Scalar component of state vector
y	Scalar component of state vector
z	Scalar component of state vector
ω	Angular velocity
μ	Gravitation parameter
a	Semimajor axis
$\boldsymbol{a}$	Thrust acceleration vector
p	Pursuing vehicle
e	Evading vehicle
$\boldsymbol{\psi}$	Terminal constraint vector
$\boldsymbol{g}$	Constraint vector
$\boldsymbol{\lambda}$	Adjoint variable vector
H	Hamiltonian
L	Lagrangian
t_f	Final time
α	In-plane control angle
β	Out-of-plane control angle

$\boldsymbol{r}$	Particle position
$\boldsymbol{v}$	Particle "velocity"
i	Iteration number
j	Current generation
k	Particle number
S	Set of training locations
w_i	Kriging weights

CHAPTER 1

INTRODUCTION

1.1 Pursuit/Evasion Game Definition and Motivation

A pursuit/evasion game is an optimization problem in which there are two players that have conflicting goals. Perhaps the best known example of such a game is the "homicidal chauffeur problem". This is one of the simplest pursuit/evasion games and was introduced by Rufus Issacs [1] in 1965. It describes a runner attempting to escape capture from a car (with higher speed but less mobility) in active pursuit. In this problem, both the runner and the car are optimizing time until capture, but the runner wants to maximize that time while the car is trying to minimize that time. Hence, this type of problem is often called a "minimax" problem.

The minimax problem considered in this work is a differential game between two spacecraft in orbit around Earth. The motion of the spacecraft is described in relative coordinates, specifically in the Hill-Clohessy-Wiltshire reference frame. An interception is the goal in this case, i.e. the positions at the final time will match (but the velocities need not). In terms of the goals, this problem is similar to the homicidal chauffeur example - the pursuing vehicle is minimizing the time until interception and the evading vehicle is maximizing the time until interception.

In fact, the evader would like to drive the interception time to infinity, if possible. To guarantee the existence of a solution to the game in this work, the pursuer was assigned a thrust sufficiently higher than that of the evader to ensure that the evader cannot escape.

There are a number of potential applications for this analysis. First, it could be used to simulate interaction between hostile intelligence satellites. The second and more subtle application is to give an estimate for a "worst-case scenario" in some maneuver in which there is uncertainty in one player's

motion. For example, consider a satellite with a sporadic and uncontrolled thruster and the pursuing repair satellite. The longest possible time until capture occurs in the case in which the rogue satellite is actively avoiding capture, i.e. the pursuit/evasion game. Knowledge of this worst-case scenario could provide design bounds on the vehicles and controller software used for such a repair mission.

1.2 One-Sided Optimization

The simplest optimization problem is one-sided, meaning that there is only one player that is trying to optimize some cost. The problem may be constrained or unconstrained. An example of unconstrained optimization in orbit mechanics is to find the thrust profile to maximize the final energy in a given time. This problem is unconstrained because there are no requirements for a feasible solution.

A problem where the goal is to find the thrust history from Earth to rendezvous with Mars at an unspecified time is a constrained problem. In order to be a feasible solution, the spacecraft must have matched the position and velocity of Mars at the unspecified final time. These requirements restrict the possible choices for the thrust time history.

In an optimization problem, the "cost function", or "objective function", is the function that is being optimized (minimized or maximized). In order to achieve the optimal value, some variables called the "controls" or "decision variables" are defined. The optimizing process varies these controls in order to find the optimal solution. In trajectory optimization, the control variables are usually the thrust pointing angles and/or the thrust magnitude. By denoting the objective function as $f(x)$, the optimal control variable vector $\boldsymbol{u}$ can be written as

$$\boldsymbol{u}^T = \arg\min_{\boldsymbol{u}} f(x) \tag{1.1}$$

This means that the optimal choice of $\boldsymbol{u}$ causes the smallest $f(x)$ out of all other possible choices for $\boldsymbol{u}$.

1.3 Differential Game

1.3.1 Two-Sided Optimization

The differential game problem considered in this work is a not a one-sided optimization, but instead is a two-sided optimization. A two-sided problem has two players that are each optimizing the same cost function, but with different goals. This means that although the cost function is the same, one player wants to minimize that cost while the other player wants to maximize it.

The optimal solution to a differential game is the solution that, if changed, would cause one of the player's situations to worsen, i.e. change contrary to that player's goal. This is referred to as a saddle point solution.

To illustrate the concept of a saddle point solution, consider the 3D plot of $z = x^2 - y^2$ in Figure 1.1. This figure's shape demonstrates why the solution point is known as a saddle point.

In this case the objective is simply z. The point $(x, y, z) = (0, 0, 0)$ is the saddle point. Note that in the x direction (along the $x - z$ plane), the saddle point is the minimum and in the y direction (along the $y - z$ plane), it is the maximum. Imagine the x and y coordinates to be two players, with the x player attempting to minimize the objective and the y player maximizing the objective. If the x player chooses a different solution, i.e. a different point on the x-axis, the new z value is increased - contrary to the goal of the x player.

In Section 1.2, the $\boldsymbol{u}$ vector represented the set of controls for the one-player problem. For a two-sided problem, each player has its own set of controls. Therefore, an expression for the control variables similar to eqn. (1.1) can be written as

$$\begin{bmatrix} \boldsymbol{u_1} & \boldsymbol{u_2} \end{bmatrix}^T = \arg \min_{\boldsymbol{u_1}} \max_{\boldsymbol{u_2}} f(x) \tag{1.2}$$

1.3.2 Pursuit/Evasion Game

A common type of differential game is a pursuit/evasion game. The objective function is the final maneuver time, or time to capture. Following the

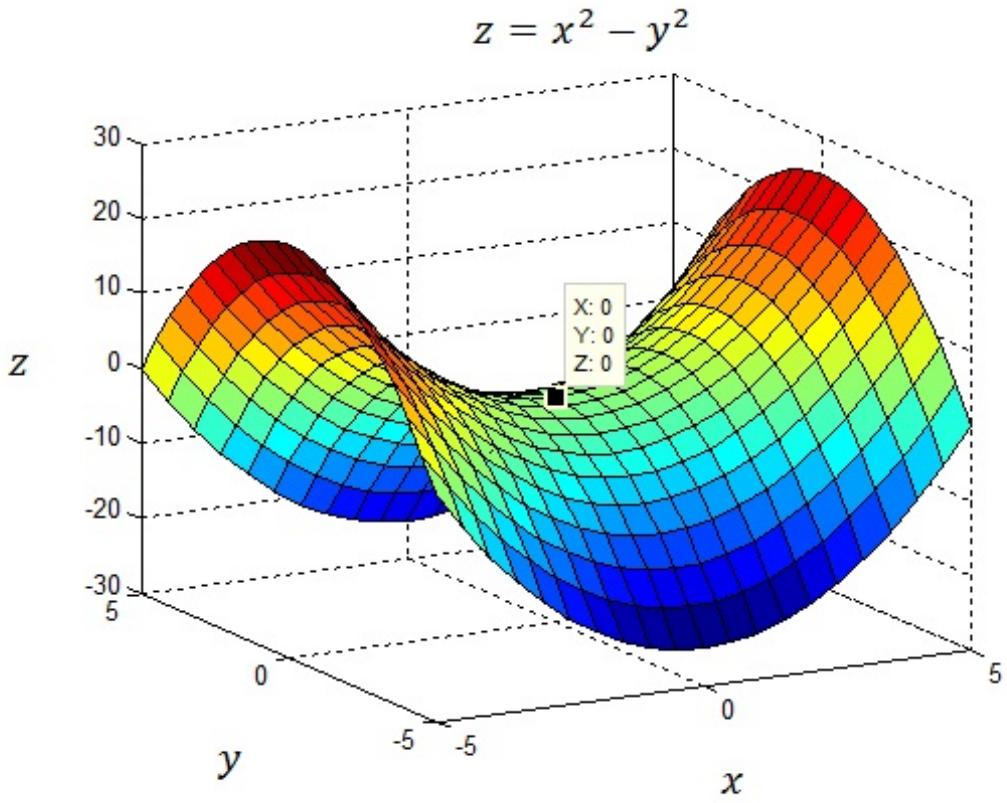

Figure 1.1: Illustrating a Function with a Saddle Point

discussion in Section 1.3.1, the players have conflicting goals and each has its own set of active controls. The pursuer is the vehicle that is minimizing the time to capture and the evader is maximizing the final time (or driving it to infinity if possible). As stated previously, the saddle-point solution is such that if a player deviates from its optimal strategy, that player's situation would worsen. In the pursuit/evasion case, this means that

1. if the pursuer changed its control history, the time to capture would be increased, and

2. if the evader changed its control history, the time to capture would be decreased.

For the pursuit/evasion game between spacecraft in this work, the vehicles are assumed to have only continuous low-thrust capability. The continuous thrust complicates the solution of the equations of motion and necessitates the use of a numerical solver.

1.4 Numerical Methods

The differential game between two satellites described in this work is solved in two ways. Each solution uses a different numerical solver. The two types of solutions are differentiated by the method of calculating the controls at each instant. The first solution, the open-loop solution, uses only the initial states to solve for an expression for the entire control-time history. The second (closed-loop) solution uses a feedback controller to find the controls as functions of the states at any time.

The open-loop problem is derived using optimal control theory. This derivation yields a set of necessary conditions that must be satisfied by a local extremum (the complete derivation is given in Chapter 3). Some of the necessary conditions are differential equations and some are terminal boundary conditions. The differential equations describe the evolution of new parameters conjugate to the states called the Lagrange multipliers, doubling the size of the problem. These necessary conditions constitute a two-point boundary value problem (TPBVP), with some known initial conditions and some known terminal conditions. The open-loop pursuit/evasion problem was solved with particle swarm optimization (PSO). PSO is a type of

heuristic, stochastic optimizer [2, 3] that has several benefits over the more traditional methods for solving TPBVPs. One drawback, however, is in computation time.

This large computation time is the motivation for creating a computationally cheap closed-loop solver. The nature of the differential game makes a feedback controller difficult to design. The efficiency of some methods of feedback control depends on variations in the expected state remaining small, but this problem requires the capability to adapt to large errors in state. A method called "kriging" [4, 5, 6], which originated in geostatistics, applied to a static problem, has been applied to a dynamic feedback control problem in orbit mechanics by Ghosh and Conway [7]. Kriging yielded a successful method of closed-loop control. The kriging process involves interpolation between a pre-computed group of open-loop optimal trajectories.

PSO and the kriging procedure and application will be described comprehensively in Chapters 4 and 5 respectively.

1.5 Previous Work

There are not many extant numerical solutions of pursuit/evasion games for the spacecraft vs. spacecraft case. Menon and Calise [8] developed an effective guidance strategy for two spacecraft involved in a three dimensional pursuit/evasion game. By using feedback linearization and a quadratic objective function, established methods for the solution of LQ problems are available and the optimal feedback laws may be found in closed form.

Horie [9] solved a coplanar orbital pursuit/evasion game using the semi-direct method with collocation developed by Horie and Conway [10, 11]. That numerical method is quite different from what is used in this work; the most significant difference being that it employs the analytical necessary conditions for (and thus must find Lagrange multipliers for) only one of the two players. It also employs Earth-centered polar coordinates while this work takes advantage of the simplification resulting from using coordinates referenced to the local HCW frame. Pontani and Conway [12] solved a more sophisticated 3D version of this problem, but also used the semi-direct optimization method and Earth-centered coordinates.

1.6 Thesis Outline

Chapter 2 discusses the Hill-Clohessy-Wiltshire equations of motion that relate the relative positions of each vehicle to a ghost point in circular orbit. In Chapter 3, the analytical necessary and sufficient conditions for optimality for the pursuit/evasion problem are derived. Chapter 3 ends with a summary of all the equations, knowns, and unknowns that characterize the two-point boundary value (TPBVP) problem. Chapter 4 contains a detailed description of Particle Swarm Optimization (PSO), how it is applied to the TPBVP from Chapter 3, and some sample solutions. The closed-loop problem is discussed in Chapter 5, including a description of kriging and some sample solutions. Chapter 6 contains a summary of the work and suggestions for future work.

CHAPTER 2

GOVERNING EQUATIONS

2.1 Hill-Clohessy-Wiltshire Frame

In order to model the differential game, the equations of motion must be developed for the system. This work is considering a pursuit/evasion problem in Earth orbit with the two spacecraft in relatively close proximity. These conditions allow for the use of a relative reference frame and relative equations of motion that are much simpler than those of the full model. The relative frame used here is the Hill-Clohessy-Wiltshire (HCW) frame.

The Hill-Clohessy-Wiltshire (HCW) frame describes the location of an object in space relative to another (moving) object in a circular orbit, where each vehicle is assumed to have a point mass. The equations of motion that come from the HCW frame are accurate as long as the objects are not very far apart. The HCW frame is shown in Figure 2.1. The shaded gray area is the circular orbit, viewed at an angle in three-dimentional space. The blue coordinate system shows Earth's inertial frame. The purple coordinate system is the HCW reference frame, with red vector $\bar{\boldsymbol{r}}$ locating an object in the HCW frame. The orange vector r^* locates the origin of the HCW frame relative to Earth.

The reference object, at the origin of the reference frame, continues to move in the same circular orbit. The movement of the second object is described in terms of relative distance from the origin in the HCW reference frame.

In this work, there are two objects in the HCW frame: the pursuer and the evader. The origin of the reference frame moves as a fictitious satellite in an unperturbed, circular orbit would move. The states of both vehicles are described relative to this imaginary object in a circular orbit.

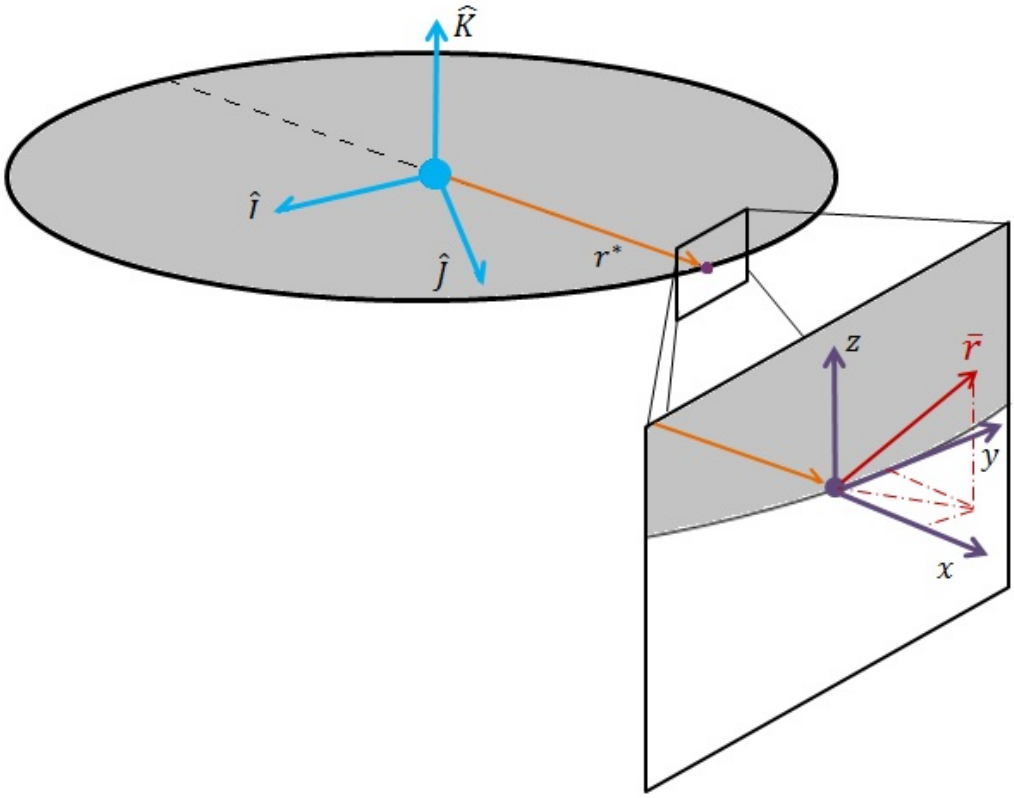

Figure 2.1: Hill-Clohessy-Wiltshire (HCW) Frame

2.2 Equations of Motion

The HCW equations are:

$$\ddot{x} - 2\omega\dot{y} - 3\omega^2 x = a_x \tag{2.1a}$$

$$\ddot{y} + 2\omega\dot{x} = a_y \tag{2.1b}$$

$$\ddot{z} + \omega^2 x = a_z \tag{2.1c}$$

where

$$\omega = \sqrt{\frac{\mu}{a^3}}$$

where μ is the gravitation parameter of the primary body, a is the semimajor axis of the circular orbit, and a_x, a_y and a_z are the components of applied thrust acceleration in the x, y, and z directions.

2.3 Equations of Motion in Matrix Form

The governing equations may be rewritten in matrix form. First, a state vector $\boldsymbol{x}$ is defined as

$$\begin{aligned}\boldsymbol{x} &= \begin{bmatrix} x & y & z & v_x & v_y & v_z \end{bmatrix}^T \\ &= \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}^T\end{aligned}$$

With this state vector and the thrust acceleration vector, $\boldsymbol{a}$, the equations of motion from Section 2.2 can be written in state-space form [13] as

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{a} \tag{2.2}$$

where

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3\omega^2 & 0 & 0 & 0 & 2\omega & 0 \\ 0 & 0 & 0 & -2\omega & 0 & 0 \\ 0 & 0 & -\omega^2 & 0 & 0 & 0 \end{bmatrix} \tag{2.3}$$

and

$$\boldsymbol{a} = \begin{bmatrix} 0 & 0 & 0 & a_x & a_y & a_z \end{bmatrix}^T \tag{2.4}$$

CHAPTER 3

NECESSARY CONDITIONS FOR SADDLE POINT EQUILIBRIUM

3.1 General Optimization Equations

In order to simulate the differential game, the problem must be fully defined. The approach used to define the system in this work was to use optimal control theory to derive the necessary and sufficient conditions for an optimal game. A feasible solution of the resulting equations will be a critical point for the optimization problem. Before deriving the equations for the differential game, a brief summary of the general optimization equations is given.

Consider a general nonlinear optimization problem, i.e.

$$\begin{aligned} &Minimize\ f(\boldsymbol{x}, \boldsymbol{u}) \\ &subject\ to \\ &\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{x}}) = \boldsymbol{0} \\ &with\ terminal\ constraints\ \boldsymbol{\psi}[\boldsymbol{x}(t_f), t_f] = \boldsymbol{0} \end{aligned} \tag{3.1}$$

where f($\boldsymbol{x}, \boldsymbol{u}$) is the function to be optimized (also known as the cost function), $\boldsymbol{g}$ is a vector describing the system dynamics, and $\boldsymbol{\psi}$ contains any terminal constraints as functions of the final time and/or the states, $\boldsymbol{x}$, at the final time.

The scalar function defined as the Lagrangian, L, of the problem (3.1) comes from the cost function, f. The cost function is of the form

$$f = \phi[\boldsymbol{x}(t_f), t_f] + \int_{t_0}^{t_f} L\, dt$$

where $\boldsymbol{\phi}$ is a function of the final states and the final time. The cost function has two components: $\boldsymbol{\phi}$ is a cost assigned only at the final time, while the integral of L is an accumulated cost over the entire time interval.

Define the Hamiltonian, H, as

$$H = L + \boldsymbol{\lambda}^T \boldsymbol{g} \tag{3.2}$$

where $\boldsymbol{\lambda}$ is a vector of additional unknown, time-dependent variables called Lagrange multipliers (also known as adjoint or costate variables). Each adjoint variable corresponds to a constraint in $\boldsymbol{g}$, so the number of Lagrange multipliers is equal to the number of constraints.

To simplify the result, define a function Φ as

$$\Phi = \phi + \boldsymbol{\nu}^{\boldsymbol{T}} \boldsymbol{\psi} \tag{3.3}$$

where $\boldsymbol{\nu}$ is a vector of unknown constants (additional Lagrange multipliers).

With these functions defined, the necessary conditions (NCs) for optimality for the problem defined in eqn. (3.1) are [14]:

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

with

$$\boldsymbol{\lambda}(t_f) = \frac{\partial \Phi}{\partial \boldsymbol{x}} \tag{3.5}$$

and

$$\frac{\partial H}{\partial \boldsymbol{u}} = 0 \tag{3.6}$$

Eqn. (3.6) is called the "optimality condition".

These are the NCs for a general problem with an objective of any form. The expressions for ϕ and L will vary based on the problem objective and user specifications. The objective of interest here is to minimize the final time. To achieve a cost function for this objective, i.e.

$$f = t_f$$

there are a few possible choices for sets of ϕ and L. For example, one choice is $\phi = 0$ and $L = 1$. The set used in this work is $\phi = t_f$ and $L = 0$, because this option simplifies the Hamiltonian. Using these values for an objective of

minimizing final time, eqn. (3.2) becomes

$$H = \boldsymbol{\lambda}^T \boldsymbol{g} \tag{3.7}$$

The sufficient condition (SC) for optimality for eqn. (3.1) is

$$\frac{\partial H^2}{\partial \boldsymbol{u} \partial \boldsymbol{u}} \geq 0 \tag{3.8}$$

i.e. the Hessian of the Hamiltonian must be positive semi-definite. It should be noted that this sufficient condition applies to problems where the objective is minimization, such as in eqn. (3.1). In a maximizing problem, the cost function, Hamiltonian, and NCs remain the same, but the SC changes slightly. If system (3.1) were to maximize the objective, the SC would become

$$\frac{\partial H^2}{\partial \boldsymbol{u} \partial \boldsymbol{u}} \leq 0 \tag{3.9}$$

i.e. the Hessian of the Hamiltonian must be negative semi-definite.

3.2 Differential Game Optimization Equations

Section 3.1 discusses the general equations that can be applied to any problem. This section applies these conditions to the differential game in the HCW reference frame, yielding the full analytical problem.

3.2.1 Controls

The first step in defining the pursuit/evasion analytical problem is to define the controls, represented by the symbol $\boldsymbol{u}$ in Section 3.1. The controls are the values that the optimizer adjusts to find the optimal solution. In this minimax problem, the controls for each player are two angles that orient the thrust acceleration vector in the Hill-Clohessy-Wiltshire frame. Angle α represents the in-plane angle and β represents the out-of-plane angle. Figure 3.1 shows how the control angles orient the thrust acceleration vector $\boldsymbol{a}$ in the HCW reference frame. The out-of-plane component of the frame, the z coordinate, is shown in red, while the x and y plane is shown in purple.

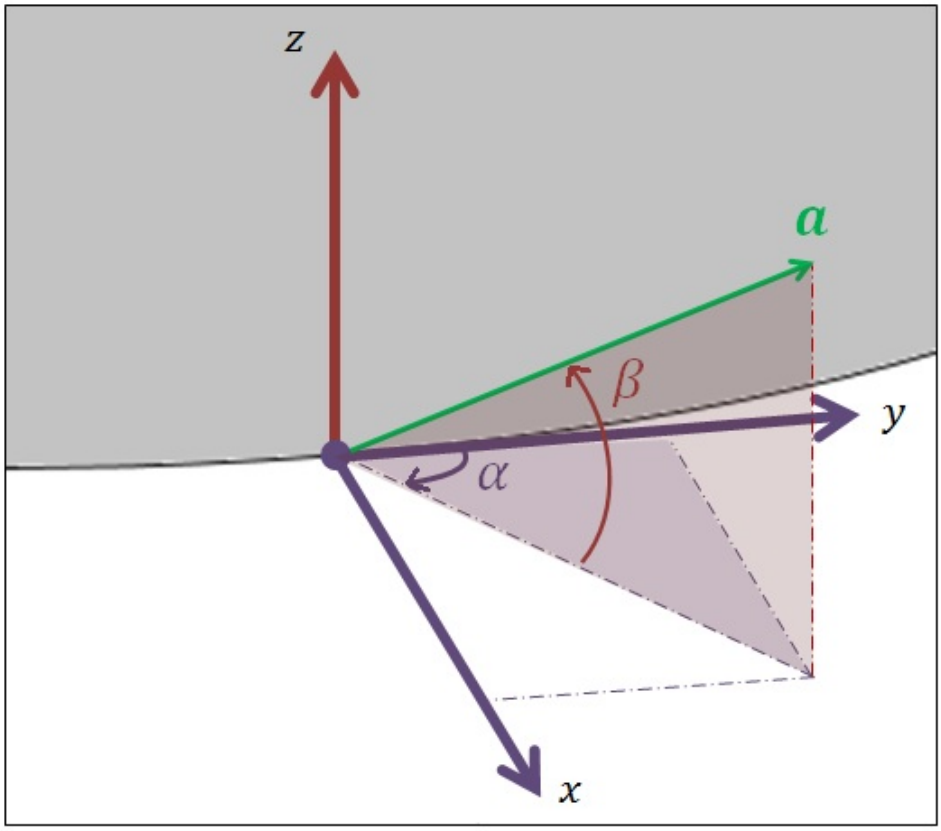

Figure 3.1: Control Angle Visualization

From geometry, the thrust acceleration vector components can be expressed in terms of the control angles as well as the thrust acceleration magnitude.

$$\boldsymbol{a} = \begin{bmatrix} \|a\| \sin\alpha \cos\beta & \|a\| \cos\alpha \cos\beta & \|a\| \sin\beta \end{bmatrix}^T \tag{3.10}$$

Substituting the values of the thrust acceleration vector into eqn. (2.1) yields the following equations of motion

$$\ddot{x} - 2\omega\dot{y} - 3\omega^2 x = \|a\| \sin\alpha \cos\beta \tag{3.11a}$$

$$\ddot{y} + 2\omega\dot{x} = \|a\| \cos\alpha \cos\beta \tag{3.11b}$$

$$\ddot{z} + \omega^2 x = \|a\| \sin\beta \tag{3.11c}$$

3.2.2 Pursuit/Evasion Problem

The rest of the pursuit/evasion problem definition can be derived from the general optimization equations in Section 3.1.

In the minimax problem, there are two objects in the HCW frame that are each subject to eqn. (2.2). Therefore, the total set of equations of motion for the pursuit/evasion problem is as follows

$$\dot{\boldsymbol{x}}_{\boldsymbol{p}} = \boldsymbol{A}\boldsymbol{x}_{\boldsymbol{p}} + \boldsymbol{a}_{\boldsymbol{p}} \tag{3.12a}$$

$$\dot{\boldsymbol{x}}_{\boldsymbol{e}} = \boldsymbol{A}\boldsymbol{x}_{\boldsymbol{e}} + \boldsymbol{a}_{\boldsymbol{e}} \tag{3.12b}$$

where subscripts $\boldsymbol{p}$ and $\boldsymbol{e}$ denote the pursuer and the evader, respectively, and the state vectors are defined as

$$\begin{aligned} \boldsymbol{x}_{\boldsymbol{p}} &= \begin{bmatrix} x_p & y_p & z_p & v_{x,p} & v_{y,p} & v_{z,p} \end{bmatrix}^T \\ &= \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}^T \end{aligned}$$

and

$$\begin{aligned} \boldsymbol{x}_{\boldsymbol{e}} &= \begin{bmatrix} x_e & y_e & z_e & v_{x,e} & v_{y,e} & v_{z,e} \end{bmatrix}^T \\ &= \begin{bmatrix} x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} \end{bmatrix}^T \end{aligned}$$

The minimax problem also has some terminal constraints, denoted by $\boldsymbol{\psi}$ in eqn. (3.1). The objective is achieved and the problem is completed when capture has occurred (i.e. the positions of the pursuer and the evader are equal and the time is the final time. This requirement is represented in terms of the terminal constraints

$$\boldsymbol{\psi} = \begin{bmatrix} x_{1f} - x_{7f} \\ x_{2f} - x_{8f} \\ x_{3f} - x_{9f} \end{bmatrix} = \mathbf{0} \tag{3.13}$$

This minimax problem is a two-sided optimization problem, meaning that the two players have conflicting goals (see Section 1.3.2). The pursuer's objective is to minimize the final time while the evader's objective is to maximize the final time. If the control vector $\boldsymbol{u}$ is defined as

$$\boldsymbol{u} = \begin{bmatrix} \alpha & \beta \end{bmatrix}^T$$

using the thrust pointing angles, then the total control vector can be written as a function of the conflicting goals as follows

$$\begin{bmatrix} \boldsymbol{u_p} & \boldsymbol{u_e} \end{bmatrix}^T = \arg\min_{\boldsymbol{u_p}} \max_{\boldsymbol{u_e}} t_f \tag{3.14}$$

By examining the equations of motion in eqn. (3.12), it is seen that the equations governing each vehicle are decoupled from the other. Due to this separability, eqn. (3.14) can be split into two parts

$$\boldsymbol{u_p} = \arg\min_{\boldsymbol{u_p}} t_f \tag{3.15a}$$

$$\boldsymbol{u_e} = \arg\max_{\boldsymbol{u_e}} t_f \tag{3.15b}$$

Pontryagin's minimum principle states that minimizing the Hamiltonian produces the same solution as minimizing the cost directly. Therefore, t_f in eqn. (3.15) can be replaced by the Hamiltonian H

$$\boldsymbol{u_p} = \arg\min_{\boldsymbol{u_p}} H \tag{3.16a}$$

$$\boldsymbol{u_e} = \arg\max_{\boldsymbol{u_e}} H \tag{3.16b}$$

To find the Hamiltonian for the pursuit/evasion problem, eqns. (3.7) and

eqn. (3.1) are combined to yield

$$H = \boldsymbol{\lambda_p}^T \dot{\boldsymbol{x}}_{\boldsymbol{p}} + \boldsymbol{\lambda_e}^T \dot{\boldsymbol{x}}_{\boldsymbol{e}} \tag{3.17}$$

where

$$\boldsymbol{\lambda_p} = \begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 & \lambda_4 & \lambda_5 & \lambda_6 \end{bmatrix}^T$$

and

$$\boldsymbol{\lambda_e} = \begin{bmatrix} \lambda_7 & \lambda_8 & \lambda_9 & \lambda_{10} & \lambda_{11} & \lambda_{12} \end{bmatrix}^T$$

Now that the controls and Hamiltonian are defined, the necessary conditions for the specific problem can be found. Applying eqn. (3.4) to the Hamiltonian (3.17) yields the equations for the evolution of the Lagrange multipliers

$$\dot{\boldsymbol{\lambda}}_{\boldsymbol{p}} = -\boldsymbol{A}^T \boldsymbol{\lambda_p} \tag{3.18a}$$

$$\dot{\boldsymbol{\lambda}}_{\boldsymbol{e}} = -\boldsymbol{A}^T \boldsymbol{\lambda_e} \tag{3.18b}$$

with boundary conditions at t_f given by eqn. (3.5):

$$\begin{aligned} \lambda_{1f} &= \nu_1 & \lambda_{7f} &= -\nu_1 \\ \lambda_{2f} &= \nu_2 & \lambda_{8f} &= -\nu_2 \\ \lambda_{3f} &= \nu_3 & \lambda_{9f} &= -\nu_3 \\ \lambda_{4f} &= 0 & \lambda_{10f} &= 0 \\ \lambda_{5f} &= 0 & \lambda_{11f} &= 0 \\ \lambda_{6f} &= 0 & \lambda_{12f} &= 0 \end{aligned}$$

These boundary conditions can also be written as:

$$
\begin{aligned}
\lambda_{1f} + \lambda_{7f} &= 0 \\
\lambda_{2f} + \lambda_{8f} &= 0 \\
\lambda_{3f} + \lambda_{9f} &= 0 \\
\lambda_{4f} &= 0 \\
\lambda_{5f} &= 0 \\
\lambda_{6f} &= 0 \\
\lambda_{10f} &= 0 \\
\lambda_{11f} &= 0 \\
\lambda_{12f} &= 0
\end{aligned}
\tag{3.19}
$$

or even more simply:

$$
\boldsymbol{\lambda_{pf}} + \boldsymbol{\lambda_{ef}} = \mathbf{0} \tag{3.20}
$$

The optimal controls can then be derived from the Hamiltonian in terms of the values of the Lagrange multipliers. This is done using the optimality condition eqn. (3.6) and the sufficient conditions in eqns. (3.8) and (3.9). For the pursuer, these become

$$
\sin\beta_p = -\frac{\lambda_6}{\sqrt{\lambda_4^2 + \lambda_5^2 + \lambda_6^2}} \tag{3.21a}
$$

$$
\cos\alpha_p = -\frac{\lambda_5}{\cos\beta_p\sqrt{\lambda_4^2 + \lambda_5^2 + \lambda_6^2}} \tag{3.21b}
$$

$$
\sin\alpha_p = -\frac{\lambda_4}{\cos\beta_p\sqrt{\lambda_4^2 + \lambda_5^2 + \lambda_6^2}} \tag{3.21c}
$$

and for the evader

$$\sin\beta_e = \frac{\lambda_{12}}{\sqrt{\lambda_{10}^2 + \lambda_{11}^2 + \lambda_{12}^2}} \tag{3.22a}$$

$$\cos\alpha_e = \frac{\lambda_{11}}{\cos\beta_e\sqrt{\lambda_{10}^2 + \lambda_{11}^2 + \lambda_{12}^2}} \tag{3.22b}$$

$$\sin\alpha_e = \frac{\lambda_{10}}{\cos\beta_e\sqrt{\lambda_{10}^2 + \lambda_{11}^2 + \lambda_{12}^2}} \tag{3.22c}$$

An interesting relationship exists between the control laws for the pursuer and evader - they are always equal, regardless of initial conditions. A simple proof is shown below.

From eqn. (3.27),

$$\boldsymbol{\lambda_p}(t) = \tilde{\boldsymbol{\Phi}}(t, t_0)\boldsymbol{\lambda_p}(t_0) \tag{3.23}$$

$$\boldsymbol{\lambda_e}(t) = \tilde{\boldsymbol{\Phi}}(t, t_0)\boldsymbol{\lambda_e}(t_0) \tag{3.24}$$

Using the boundary conditions provided in eqn. (3.20) and eqn. (3.27),

$$\tilde{\boldsymbol{\Phi}}(t, t_0)[\boldsymbol{\lambda_{p_0}} + \boldsymbol{\lambda_{e0}}] = \mathbf{0}$$

Under the assumption that $\tilde{\boldsymbol{\Phi}}$ is invertible

$$\boldsymbol{\lambda_{p_0}} + \boldsymbol{\lambda_{e0}} = \mathbf{0}$$

$$\boldsymbol{\lambda_{p_0}} = -\boldsymbol{\lambda_{e0}}$$

Eqn. (3.18) shows that the differential equations for both the pursuer and the evader evolve in the same way. If they begin equal and opposite, they

will remain equal and opposite at each instant in time

$$\begin{aligned}
\lambda_1 &= -\lambda_7 \\
\lambda_2 &= -\lambda_8 \\
\lambda_3 &= -\lambda_9 \\
\lambda_4 &= -\lambda_{10} \\
\lambda_5 &= -\lambda_{11} \\
\lambda_6 &= -\lambda_{12}
\end{aligned} \tag{3.25}$$

Substituting eqn. (3.25) into eqns. (3.21) and (3.22) yields

$$\begin{aligned}
\boldsymbol{\beta_p} &= \boldsymbol{\beta_e} \\
\boldsymbol{\alpha_p} &= \boldsymbol{\alpha_e}
\end{aligned} \tag{3.26}$$

3.2.3 Problem Simplification

Eqn. (3.18) describes the evolution of the Lagrange multipliers. It can be shown [13] to have a solution of the form

$$\boldsymbol{\lambda_i}(t) = \tilde{\boldsymbol{\Phi}}(t, t_0)\boldsymbol{\lambda_i}(t_0), \quad (i = p, e) \tag{3.27}$$

where, setting $\xi = t - t_0$

$$\tilde{\boldsymbol{\Phi}} = \begin{bmatrix}
4 - 3\cos\omega\xi & 6\omega\xi - 6\sin\omega\xi & 0 & -3\omega\sin\omega\xi & -6\omega(1-\cos\omega\xi) & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & \cos\omega\xi & 0 & 0 & \omega\sin\omega\xi \\
-\frac{\sin\omega\xi}{\omega} & -\frac{2(1-\cos\omega\xi)}{\omega} & 0 & \cos\omega\xi & 2\sin\omega\xi & 0 \\
\frac{2(1-\cos\omega\xi)}{\omega} & \frac{3\omega\xi - 4\sin\omega\xi}{\omega} & 0 & -2\sin\omega\xi & 4\cos\omega\xi - 3 & 0 \\
0 & 0 & -\frac{\sin\omega\xi}{\omega} & 0 & 0 & \cos\omega\xi
\end{bmatrix}$$

An additional simplification for this particular problem results from the nature of that analytical solution. Knowing $\tilde{\boldsymbol{\Phi}}$ and the boundary conditions from eqn. (3.19), it is possible to solve algebraically for 9 out of the 12 initial Lagrange multipliers in terms of the remaining 3 as well as the final time.

In other words, matrices $\boldsymbol{C}$ and $\boldsymbol{\Xi}$ can be found such that

$$\boldsymbol{C}\begin{bmatrix}\lambda_{1,0} & \lambda_{2,0} & \lambda_{3,0} & \mathbf{0}_{1x6}\end{bmatrix}^T = \boldsymbol{\Xi}\begin{bmatrix}\lambda_{4,0} & \lambda_{5,0} & \lambda_{6,0} & \lambda_{7,0} & \lambda_{8,0} & \lambda_{9,0} & \lambda_{10,0} & \lambda_{11,0} & \lambda_{12,0}\end{bmatrix}^T \tag{3.28}$$

If one defines $\boldsymbol{P}$ as

$$\boldsymbol{P} = \begin{bmatrix}\tilde{\boldsymbol{\Phi}}(t_f, t_0) & \mathbf{0} \\ \mathbf{0} & \tilde{\boldsymbol{\Phi}}(t_f, t_0)\end{bmatrix}$$

then $\boldsymbol{\Xi}$ can be found as

$$\boldsymbol{\Xi} = \begin{bmatrix}
P_{(1,4)} & P_{(1,5)} & P_{(1,6)} & P_{(7,7)} & P_{(7,8)} & P_{(7,9)} & P_{(7,10)} & P_{(7,11)} & P_{(7,12)} \\
P_{(2,4)} & P_{(2,5)} & P_{(2,6)} & P_{(8,7)} & P_{(8,8)} & P_{(8,9)} & P_{(8,10)} & P_{(8,11)} & P_{(8,12)} \\
P_{(3,4)} & P_{(3,5)} & P_{(3,6)} & P_{(9,7)} & P_{(9,8)} & P_{(9,9)} & P_{(9,10)} & P_{(9,11)} & P_{(9,12)} \\
P_{(4,4)} & P_{(4,5)} & P_{(4,6)} & 0 & 0 & 0 & 0 & 0 & 0 \\
P_{(5,4)} & P_{(5,5)} & P_{(5,6)} & 0 & 0 & 0 & 0 & 0 & 0 \\
P_{(6,4)} & P_{(6,5)} & P_{(6,6)} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & P_{(10,7)} & P_{(10,8)} & P_{(10,9)} & P_{(10,10)} & P_{(10,11)} & P_{(10,12)} \\
0 & 0 & 0 & P_{(11,7)} & P_{(11,8)} & P_{(11,9)} & P_{(11,10)} & P_{(11,11)} & P_{(11,12)} \\
0 & 0 & 0 & P_{(12,7)} & P_{(12,8)} & P_{(12,9)} & P_{(12,10)} & P_{(12,11)} & P_{(12,12)}
\end{bmatrix}$$

and

$$\boldsymbol{C} = \begin{bmatrix}
P_{(1,1)} & P_{(1,2)} & P_{(1,3)} & 0 & 0 & 0 & 0 & 0 & 0 \\
P_{(2,1)} & P_{(2,2)} & P_{(2,3)} & 0 & 0 & 0 & 0 & 0 & 0 \\
P_{(3,1)} & P_{(3,2)} & P_{(3,3)} & 0 & 0 & 0 & 0 & 0 & 0 \\
P_{(4,1)} & P_{(4,2)} & P_{(4,3)} & 0 & 0 & 0 & 0 & 0 & 0 \\
P_{(5,1)} & P_{(5,2)} & P_{(5,3)} & 0 & 0 & 0 & 0 & 0 & 0 \\
P_{(6,1)} & P_{(6,2)} & P_{(6,3)} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

With $\boldsymbol{C}$ and $\boldsymbol{\Xi}$ known (and a given value for ω), for any chosen values of $\lambda_{1,0}$, $\lambda_{2,0}$, $\lambda_{3,0}$, and t_f, the remaining initial Lagrange multipliers can be calculated with simple matrix multiplication. When solving the minimax problem with numerical methods, eqn. (3.28) allows the number of decision

variables to be reduced from thirteen to just four: the three initial Lagrange multipliers and the final time. This drastically simplifies the numerical solution. This simplification is only possible because the HCW reference frame was used.

3.3 Analytical Problem Summary

After applying the general optimization equations to the pursuit/evasion problem and using simplifications in Section 3.2, the final system to be solved consists of:

- 12 differential equations - the equations of motion from eqn. (3.12)
- 12 algebraic equations - Lagrange multipliers from eqn. (3.27) and (3.28)
- 13 known values - 1 initial time, 12 initial states
- 12 constraints - 12 final Lagrange multipliers from eqn (3.20)
- 4 unknown values - the final time and 3 Lagrange multipliers from eqn. (3.28)

All equations must be solved simultaneously, because the optimal controls (3.21) and (3.22) are functions of the Lagrange multipliers. By solving this system, the trajectories for the differential game are found.

While the HCW equations (2.1) have an analytical solution if the right-hand sides - a_x, a_y, and a_z - are zero (i.e. no external applied thrust), this work assumes that these values are never zero (i.e. constant external applied thrust). This assumption necessitates the use of a numerical solver.

CHAPTER 4

SOLUTION OF THE DIFFERENTIAL GAME

4.1 Introduction to Particle Swarm Optimization (PSO)

Section 3.3 contains the summary of the differential game problem. As stated in that section, the Lagrange multiplier time histories are able to be found analytically for each choice of the decision parameters (λ_{1_0}, λ_{2_0}, λ_{3_0}). However, the equations of motion require the controls of eqns. (3.21) and (3.22) and must be integrated numerically. A frequently used method for numerical optimization of such a continuous system subject to constraints is called a direct method. A direct method converts the differential equations into algebraic constraints (e.g. using collocation) and then the problem becomes a nonlinear programming (NLP) problem [15]. When using NLP solvers, an initial guess of all the parameters must be provided. This initial guess must be in the neighborhood of the optimal solution. Often, the appropriate local neighborhood for a parameter is not at all intuitive, making it extremely difficult to find solutions. Also, there is no guarantee of global optimality once a solution has been found. If the guess lies in the region near a local optimum, NLP will likely converge to that globally non-optimal solution. Thus another numerical solution method, more likely to locate the global minimum, known as particle swarm optimization (PSO) is used in this work.

Particle swarm optimization is a type of heuristic, stochastic optimizer [2, 3]. It is a population-based process, in this respect similar to some evolutionary computation methods such as the genetic algorithm. Evolutionary computation methods mimic various processes found in nature. For example, PSO is modeled after the behavior of a flock of birds searching for food. Other types of metaheuristics (e.g. genetic algorithms) follow the process of natural selection.

PSO creates an initial population of particles (or agents) that each represent a solution in an n-dimensional space, where n is the number of parameters to be optimized. This initial population of solutions is randomly generated within user-specified bounds. Each particle has a fitness associated with it. The fitness is a measure of how closely the particle matches the desired solution. In other words, it is a weighted sum of the constraints to be satisfied. The particles then begin taking a series of steps in specified directions in the solution space and eventually converge on the optimal solution. This process is described in the cartoon of Figure 4.1

The new position, $\boldsymbol{r}$ of each particle, k, after taking a step is given by

$$\boldsymbol{r_k}^{(j+1)}(i) = \boldsymbol{r_k}^{(j)}(i) + \boldsymbol{v_k}^{(j+1)}(i) \tag{4.1}$$

where j represents the state in the current generation and i represents the iteration number.

The direction that each particle steps is determined by a so-called "velocity" vector, denoted $\boldsymbol{v_k}$ in eqn. (4.1). This velocity vector does not have the units of velocity, but rather behaves as a displacement. The velocity vector has the form

$$\boldsymbol{v_k}^{(j+1)}(i) = w\boldsymbol{v_k}^{(j)}(i)+c_1R(0,1)[\boldsymbol{\Psi_k}^{(j)}(i)-\boldsymbol{r_k}^{(j)}(i)]+c_2R(0,1)[\boldsymbol{\rho_k}^{(j)}-\boldsymbol{r_k}^{(j)}(i)] \tag{4.2}$$

where w, c_1, c_2 are weights, $\boldsymbol{\Psi}$ is the individual particle's best location, $\boldsymbol{\rho}$ is the global best position, and $R(0,1)$ is a random variable between 0 and 1.

In other words, the velocity vector is a randomized, weighted sum of the vectors

- in the direction the particle was moving during the previous generation (called "inertial" component)
- pointing towards the individual particle's best fitness (called "cognitive" component)
- pointing towards the position of the best global fitness (called "social" component)

There are three common ways to address a particle that exceeds the bounds of the search space. The method used in this work is to "wrap" the search

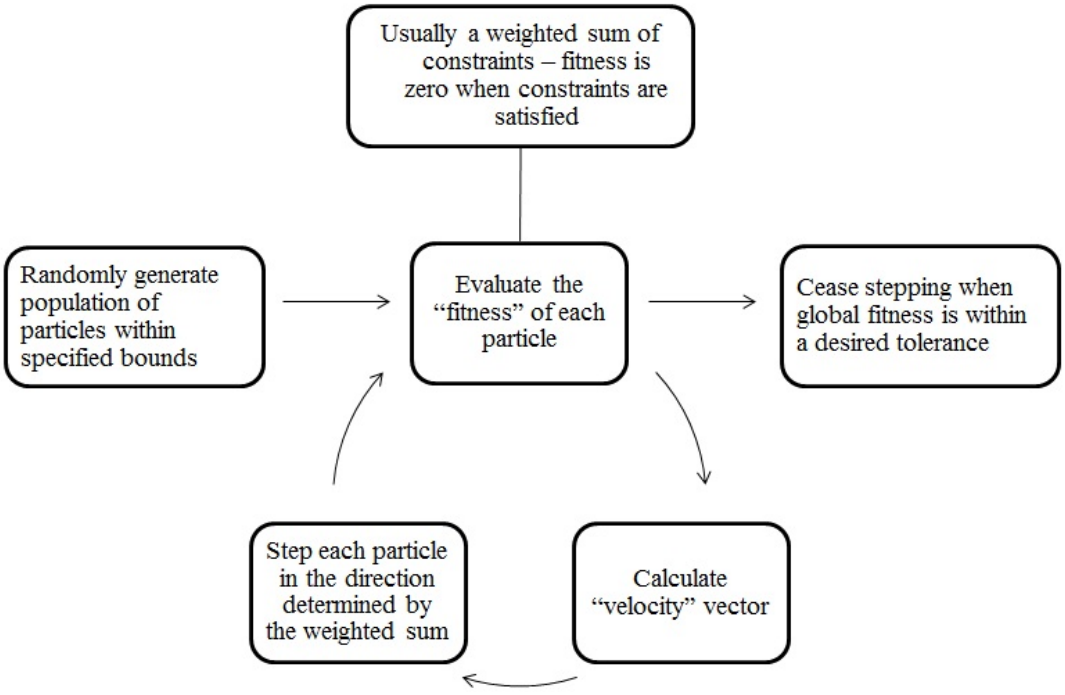

Figure 4.1: Illustration of PSO Algorithm

space i.e. if a particle reaches and will exceed an upper bound, the particle is automatically transported to the corresponding lower bound (as if the search space were wrapped such that the lower bound and upper bound touched). Another option is to reflect the particle from the edge of the search space, meaning that the current velocity is multiplied by -1. A third method is to halt the particle at the boundary, and allow the cognitive and social components to pull the particle away from the bound.

PSO has two significant benefits over NLP. First, PSO does not require an initial guess. The initial population is randomly generated. The user-specified values include the size of the population, the number of generations, and the bounds on the search space. Second, the global nature of PSO makes it much less likely to get "stuck" at a local, non-optimal solution. As they travel across the search space towards the current global best fitness, the particles often discover a new global best fitness on the way.

4.2 Application to the Differential Game Problem

Applying PSO to the minimax problem described in Section 3.3 is straightforward. As shown in Section 3.2.3, there are a total of four decision parameters: three are initial Lagrange multipliers and the fourth is the final time. For each particle, the fitness is determined by integrating the equations of motion with ode45 in MATLab, using the optimal controls (3.21) and (3.22), forward to that particle's final time and creating a weighted sum of the constraints. The constraints, in this case, state that the difference in positions of the vehicles at the final time is equal to zero. If the vehicles occupied the same point in space, the fitness would be equal to zero.

The weights for both the components of the velocity update vector and the fitness penalty function were determined largely by trial and error. The values used for the solutions in this work are listed in Table 4.1

Table 4.1: PSO Weights

w	c_1	c_2	Fitness Penalty
0.65	2	2	6378

4.3 Problem Description

A few simplifying assumptions were made about the system for the solution to the minimax problem. First, both vehicles were assumed to be point masses i.e. spacecraft attitude was not considered. Second, the propulsion systems are assumed ideal, with constant parameters. Unless otherwise stated, mass loss caused by propellant expulsion is accounted for. The mass loss is documented through the variation of the thrust acceleration magnitude with time as mass is expelled, i.e. the current mass is inversely proportional to the current acceleration which is given by:

$$a_{instantaneous} = \frac{a_{initial}}{1 - t(\frac{a_{initial}}{c})} \tag{4.3}$$

where c is the exhaust velocity.

The numerical method for solving the minimax problem requires several inputs, which are separated into two groups: the problem parameters and the initial conditions of the pursuer and evader. The problem parameters include the reference orbit to which the HCW frame is attached, thrust acceleration magnitudes of each vehicle, and effective exhaust velocity. Table 4.2 lists the values chosen for this study. All the results to follow use these parameters.

Table 4.2: Problem Parameters

Vehicle	Ref Orbit	Thrust Mag (N/kg)	Exhaust Vel (km/s)
Pursuer	GEO	0.0686	3
Evader	GEO	0.0343	3

4.4 Sample Solution: Open-Loop Case 1

Table 4.3 contains the initial conditions of both the pursuer and the evader for an example open-loop solution. Many solutions have been obtained for various initial conditions. This case with parameters as given in Table 4.2 and 4.3 is representative in complexity and degree of success obtained. Note that the evader begins its motion from the origin of the HCW reference frame.

The initial x-position of the pursuer (-38.93 km) is the difference in orbit altitude such that, if no maneuver were executed, would result in a drift,

i.e. a separation of the two vehicles in longitude, of $\frac{1}{2}$ degree per day. The numerical solution beginning from the initial conditions in Table 4.3 yields the optimal decision parameters shown in Table 4.4. The final positions of both vehicles are given in Table 4.5, where it can be seen that the terminal condition of interception is satisfied "exactly". The vehicles intercept to an accuracy on the order of centimeters.

Table 4.3: Initial Conditions for Open-Loop Case 1

Vehicle	x (km)	y (km)	z (km)	V_x (km/s)	V_y (km/s)	V_z (km/s)
Pursuer	-38.9328	-100	10	0	0	0
Evader	0	0	0	0	0	0

Table 4.4: Final Decision Variables for Open-Loop Case 1

$\lambda_{1,0}$	$\lambda_{2,0}$	$\lambda_{3,0}$	t_{final} (min)
-0.2382	-0.5704	0.0553	41.312

Table 4.5: Final Conditions for Open-Loop Case 1

Vehicle	x (km)	y (km)	z (km)
Pursuer	40.0557	97.8551	-9.6483
Evader	40.0557	97.8551	-9.6483
Difference	0	0	0

Figure 4.2 shows the optimal minimax trajectory in the HCW reference frame. The path is in 3D space. For clarity, the view of the path projected onto the XY plane is shown in Figure 4.3. The thrust pointing angle time histories are provided in Figure 4.4. It should be noted that the controls for the pursuer and the evader exactly overlap. This result was expected because of the analysis in Section 3.2.1.

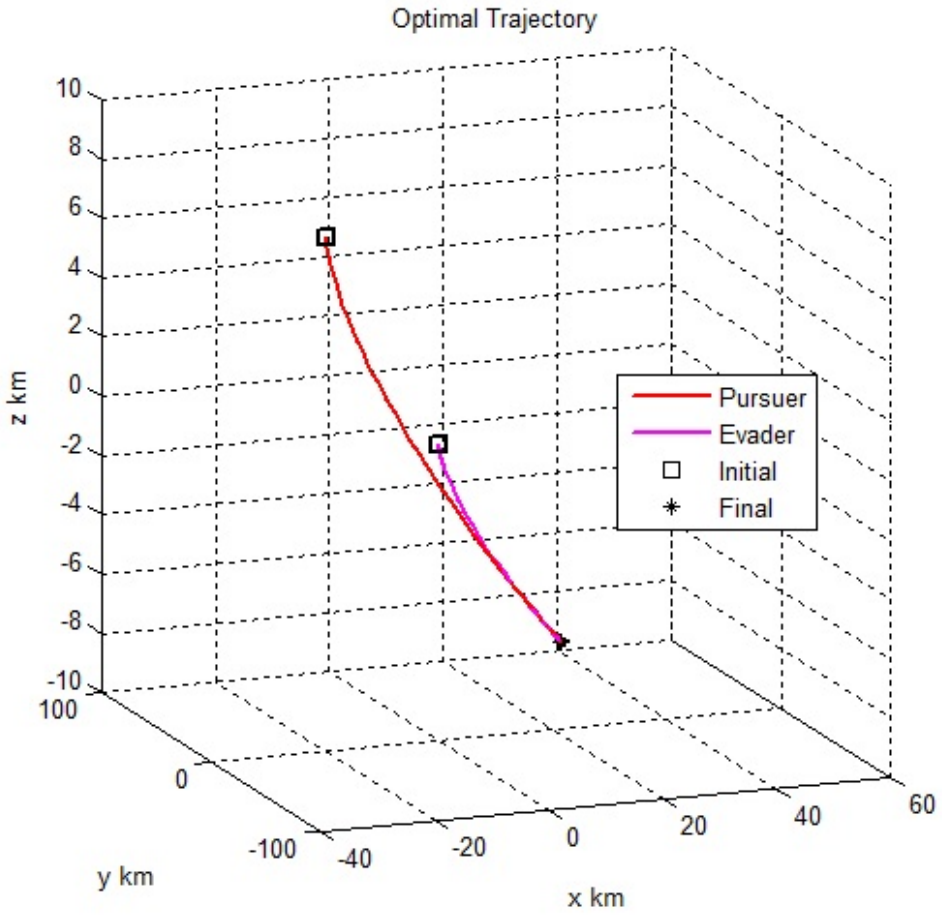

Figure 4.2: Optimal Pursuit/Evasion Trajectories in 3D Case 1

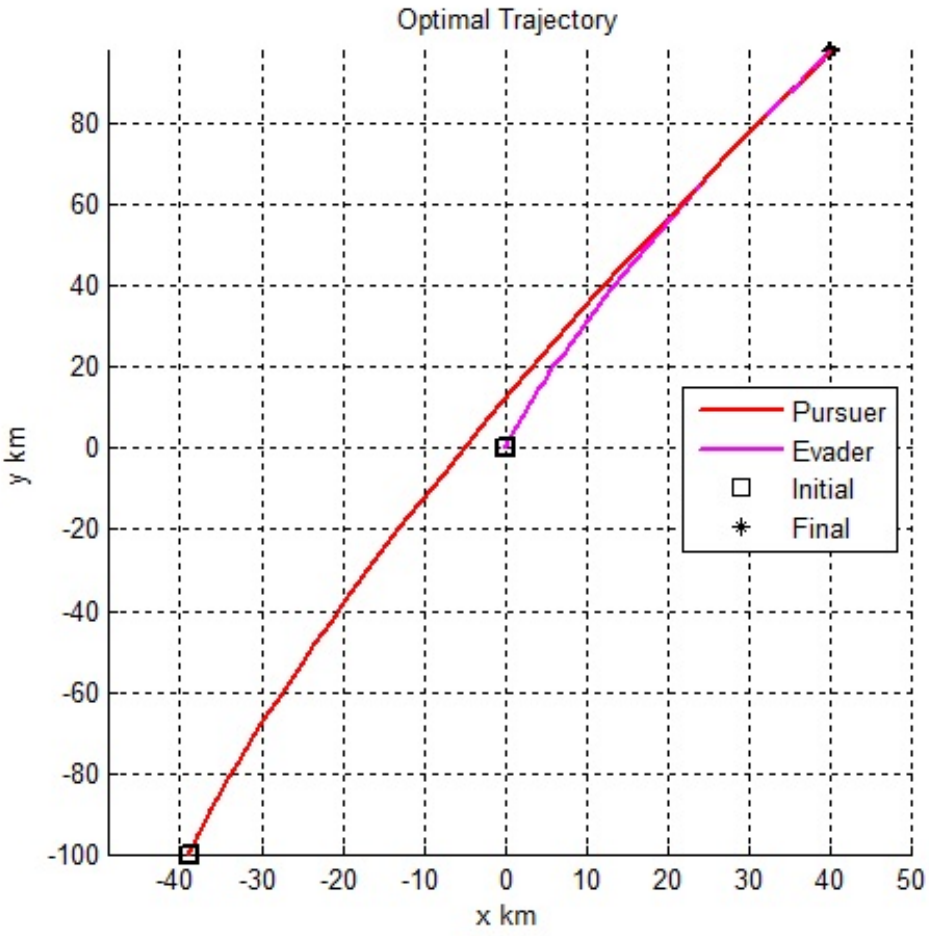

Figure 4.3: View of 3D Trajectory from XY Plane Case 1

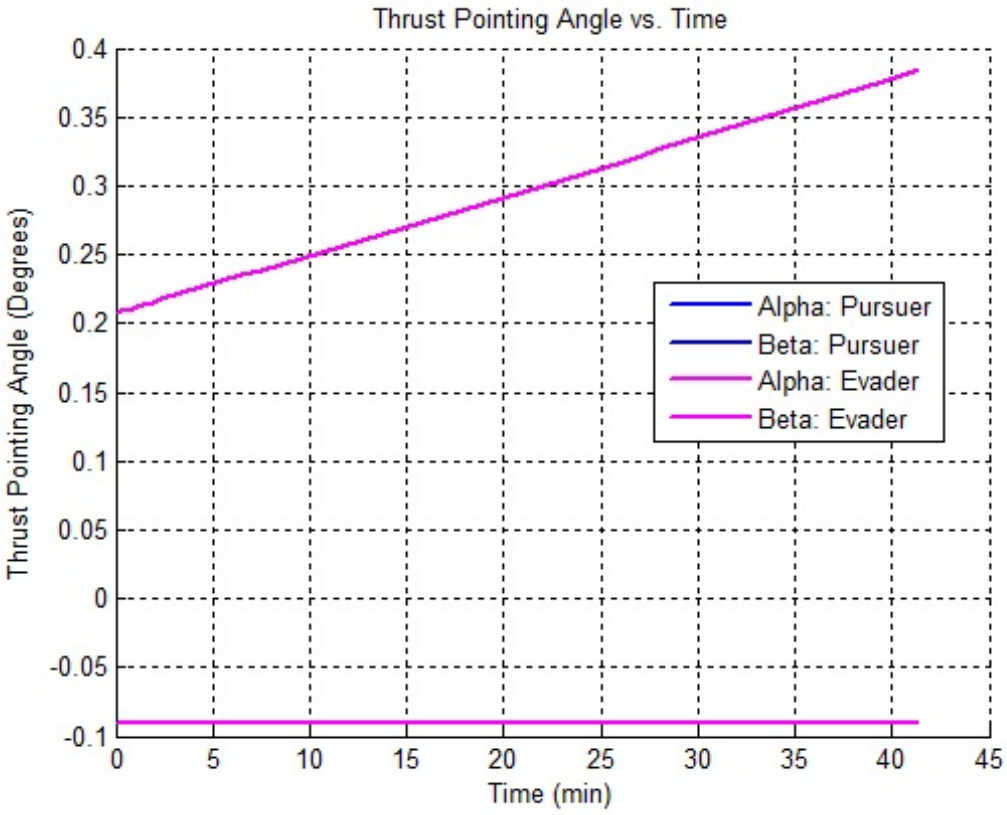

Figure 4.4: Control Time History for 3D Trajectory Case 1

4.5 Sample Solution: Open-Loop Case 2

A second solution was obtained to demonstrate the robustness of the numerical solver. In this Case 2, with initial conditions given in Table 4.6, the evader is displaced from the origin of the HCW reference frame (in Case 1 it begins from the origin, as indicated in Table 4.3). The decision variables chosen by the optimizer are listed in Table 4.7. The final states in Table 4.8 show that the PSO solver again finds an "exact" interception.

Table 4.6: Initial Conditions for Open-Loop Case 2

Vehicle	x (km)	y (km)	z (km)	V_x (km/s)	V_y (km/s)	V_z (km/s)
Pursuer	-35.9328	-101.5	5	0	0	0
Evader	-0.5	1	0.4	0	0	0

Table 4.7: Final Decision Variables for Open-Loop Case 2

$\lambda_{1,0}$	$\lambda_{2,0}$	$\lambda_{3,0}$	t_{final} (min)
-0.2405	-0.6494	0.0282	41.4430

Table 4.8: Final Conditions for Open-Loop Case 2

Vehicle	x (km)	y (km)	z (km)
Pursuer	35.9386	101.3223	-4.044
Evader	35.9386	101.3223	-4.044
Difference	0	0	0

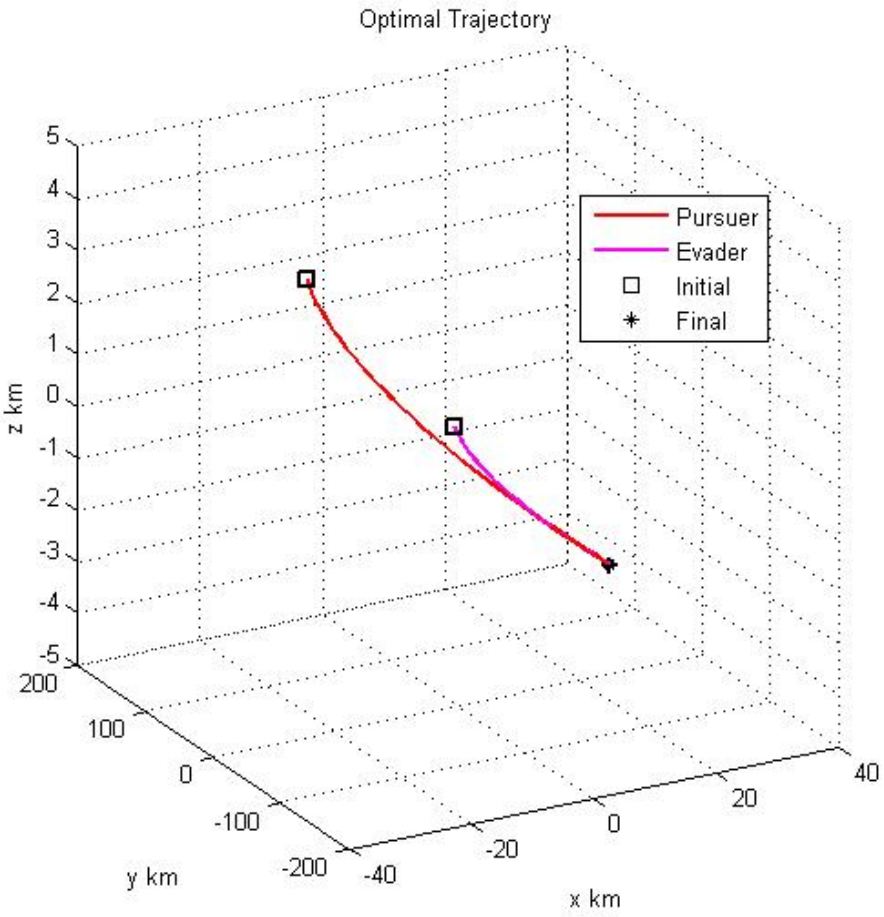

Figure 4.5: Optimal Pursuit/Evasion Trajectories in 3D Case 2

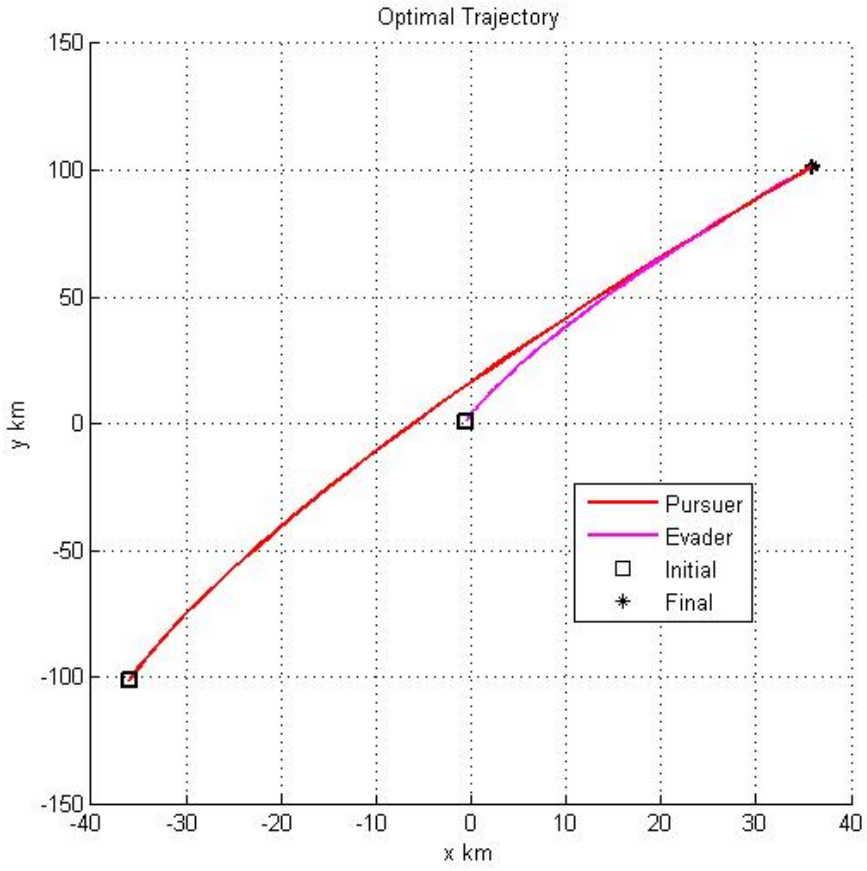

Figure 4.6: View of 3D Trajectory from XY Plane Case 2

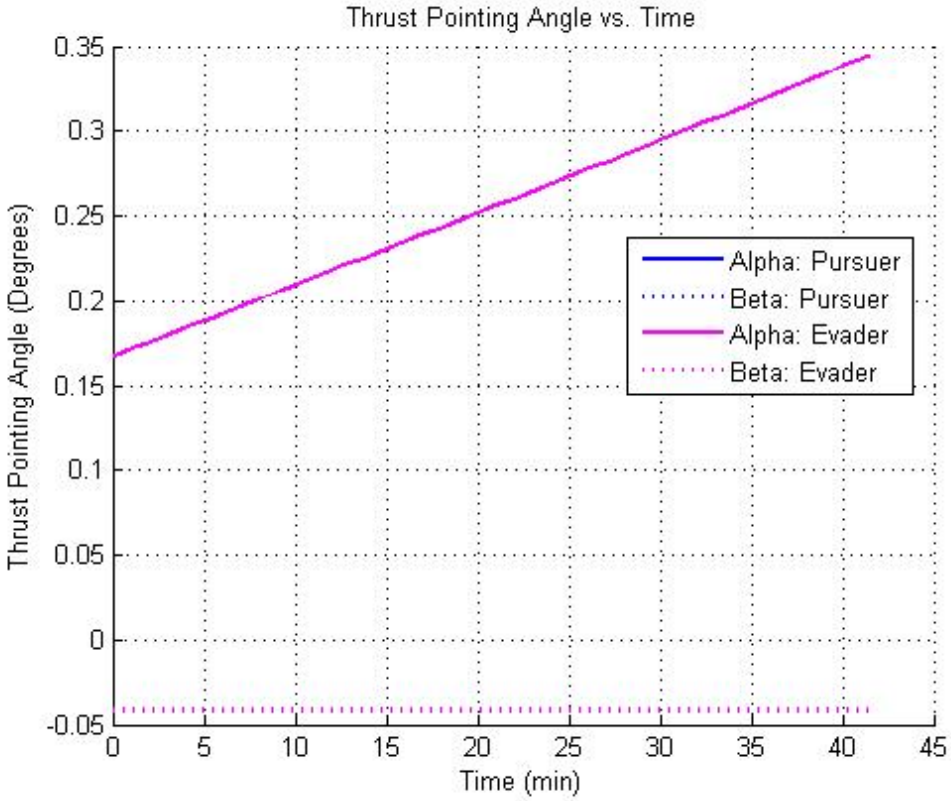

Figure 4.7: Control Time History for 3D Trajectory Case 2

4.6 Open-Loop Controller Summary

Particle swarm optimization has been shown to be an effective tool for solving this differential game. For the two cases presented, and for many other cases solved but not shown, PSO finds the optimal decision variables (three of the initial Lagrange multipliers and the final time) as well as the optimal control time-history. For each solution, interception is achieved almost exactly i.e. on the order of micrometers.

The CPU time for these solutions varies with the number of particles created in the search space and the number of iterations taken to achieve interception. For each computation, there were 175 (a number chosen by trial and error) PSO particles in the search space. The number of iterations typically varied between 150 and 800, with an average of approximately 300 iterations. For 175 particles, each iteration takes a CPU time of approximately 1.7 seconds. Therefore, the average CPU time of a single PSO calculation with 175 particles is 8.5 minutes, but can extend up to 30 minutes.

CHAPTER 5

SOLUTION FOR THE FEEDBACK CONTROLLER

5.1 Introduction to Kriging

While the PSO method provides a satisfactory method for open-loop optimal trajectory determination, computation time is sufficiently long as to make continued updates by recalculating new open-loop trajectories from intermediate points unfeasible in real-time. Instead, this work develops a method of closed-loop control to adapt to any changes in the initial state of either vehicle without having to recalculate the entire trajectory. Some typical methods of feedback control may not be applied to this problem due to the nature of the differential game as well as the potentially large magnitude of the changes in states. An alternative method, called kriging, has recently been applied to the field of optimal space trajectories [7]. This is the method of feedback control used for the minimax problem.

Kriging is a group of interpolation and extrapolation techniques for determining the value of a variable at an unknown location based on the known values of the variable at surrounding points, i.e. it was developed to be applied to a static system. It originated as a method used by geologists to map the location of natural resources. Ghosh and Conway have recently and for the first time applied kriging to a continuous dynamical system, an optimal spacecraft trajectory [7].

Kriging requires a set of known points and associated parameter values that are then used to interpolate values at unsampled points [4, 5, 6]. Consider p known "training locations" in a set S

$$S = \{x_1^*, x_2^*, ...x_p^*\}$$

and the corresponding known, optimal open-loop controls

$$\{u^*(x_1), u^*(x_2), ...u^*(x_p)\} = \{u^*_{s1}, u^*_{s2}, ...u^*_{sp}\}$$

Kriging calculates an estimation of control $\tilde{u}^*(x_0)$ corresponding to $x_0 \notin S$ but $x_0 \in ConvexHull(S)$. The control at this untrained location is given by

$$\tilde{u}^*(x_0) = \sum_{i=1}^{p} w_i(x_0) u^*_{si} \tag{5.1}$$

where the w_i are weights. In other words, the estimated control is a weighted linear combination of the observed controls from the known training locations. The kriging process calculates the optimal values for the weights w_i. The manner of optimally selecting these values is thoroughly described in the literature [4, 5, 6, 7].

In order to create a feedback controller for an optimal spacecraft trajectory with kriging, set of training locations is needed. This set consists of randomly perturbed initial conditions for the pursuit/evasion game. The strategy, then, is to use the procedure outlined in Chapter 4 for each initial condition to produce a group of optimal paths called a "field of extremals". The kriging code then takes this field as an input and creates a feedback controller that, given the states, provides the required control at each time.

5.2 Kriging Software

There are several open-source versions of kriging software. The software used in this work is a MATLab Toolbox for kriging called DACE [16]. Each time a controller is calculated from a field of extremals, there are several parameters that the user must specify. The regression model order, correlation model, and initial guesses for the correlation parameters are all values that must be specified. The accuracy of the controller is very sensitive to the choices of these values.

The regression model order can be chosen as a polynomial of order 0, 1, or 2.

The correlation model and correlation parameters determine the covariance matrix used to solve for values at untrained locations [17]. The possible choices for the correlation model are exponential, generalized exponential,

Gaussian, linear, spherical, and cubic spline. The correlation parameters are optimized by the kriging code, but the user-required initial guess of these correlation parameters can be anything (usually between 0 and 5 for this work). Upper and lower bounds must also be specified for the correlation parameter search space (in this work, between 10^{-5} and 10).

5.3 Problem Description

The problem description remains the almost the same as in Section 4.3. The difference for this problem is that the mass is assumed to be constant, for simplicity. This is a reasonable assumption, because in the the open-loop cases considered in this work, the final mass fractions of both vehicles remain above 94%.

5.4 Sample Solution: Kriging Case 1

Table 5.1 contains the initial conditions for both the pursuer and the evader for a nominal, unperturbed case. It is a 2D version of the example given in Section 4.4. All the results for the kriging are in 2D, so all out-of-plane positions and velocities are zero. Note that the evader begins its motion at the origin of the CW reference frame. Random perturbations of less than 5 km are made in the initial positions of the evader to create the field of 30 extremals shown in Figure 5.1.

Table 5.1: Nominal Initial Conditions Case 1

Vehicle	x (km)	y (km)	V_x (km/s)	V_y (km/s)
Pursuer	-38.9328	-100	0	0
Evader	0	0	0	0

The kriging parameters used to find an effective controller for this field of extremals are listen in Table 5.2. These values were determined primarily by trial and error.

To verify that kriging produces a nearly optimal trajectory, a path is begun from an arbitrary initial point and obtained using the feedback-controller

created by kriging. This path can then be compared with the optimal open-loop trajectory calculated from that same initial point.

Table 5.2: Kriging Calculation Parameters Case 1

Regression Order	1
Correlation Model	Exponential
Initial Guess Corr. Parameters	0.2
Lower Bound	10^{-5}
Upper Bound	10

The initial conditions used for the comparison of the resulting open-loop and closed-loop trajectories are in Table 5.3. For this perturbed case, the evader's starting position is moved a total of 2.25 km from its original position. The final positions for both the open-loop calculation and the closed-loop calculation are in Tables 5.4 and 5.5. A comparison of the final states shows that the kriging calculation not only achieves an accurate interception, but in fact finds almost the same trajectories as those found by the PSO solver.

Table 5.3: Initial Conditions for Comparison

Vehicle	x (km)	y (km)	V_x (km/s)	V_y (km/s)
Pursuer	-38.9328	-100	0	0
Evader	1.5	1.5	0	0

Table 5.4: Final Positions for Open-Loop (PSO) Optimal Trajectories with Final time 42.2136 minutes

Vehicle	x (km)	y (km)
Pursuer	44.0724	102.7363
Evader	44.0724	102.7363
Difference	0	0

Table 5.5: Final Positions for Closed-Loop (Kriging) Trajectories with Final time 42.2137 minutes

Vehicle	x (km)	y (km)
Pursuer	44.0730	102.7367
Evader	44.0727	102.7365
Difference	0.0002886	0.00022643

Figure 5.2 shows the optimal trajectories resulting from both PSO and kriging on the same plot for comparison. In this figure, the two solutions are indistinguishable. Figure 5.3 shows a magnified view of the interception points. Note that the axes require a very small range to capture the difference between the trajectories.

Figure 5.4 shows the control time histories resulting from PSO and kriging on the same plot. Again, the two solutions are indistinguishable at this view, so Figure 5.5 is provided to capture the difference in the controls at the interception point.

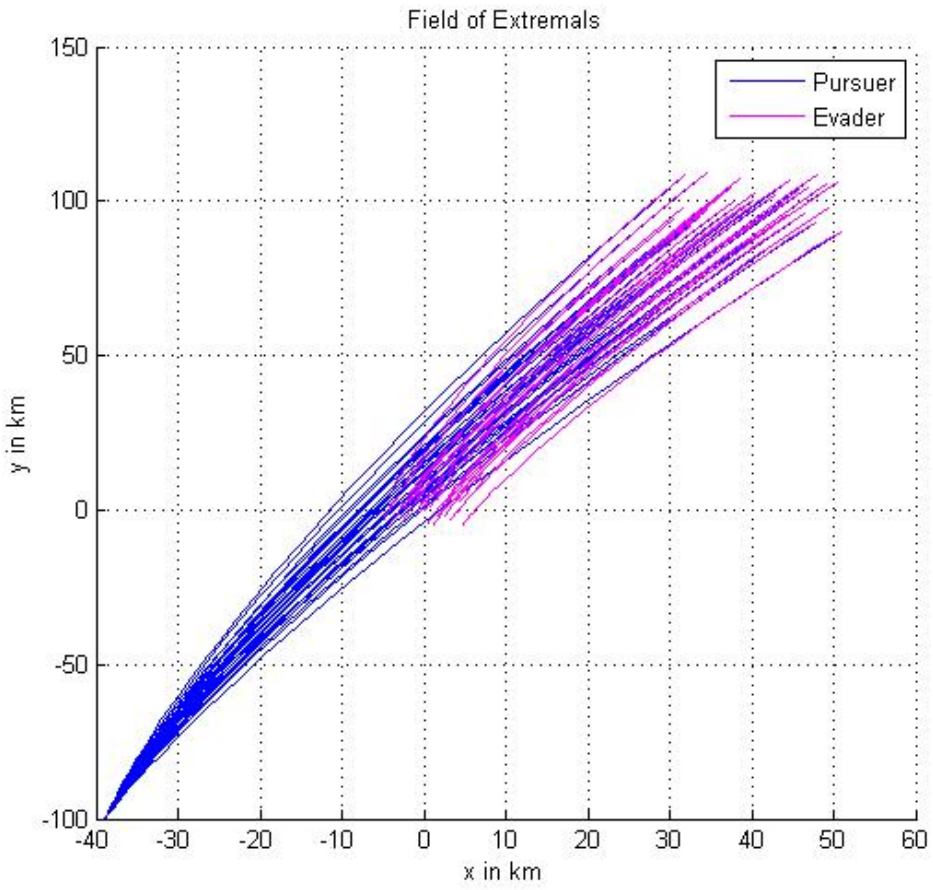

Figure 5.1: Field of Extremals - Perturbation in Evader Only

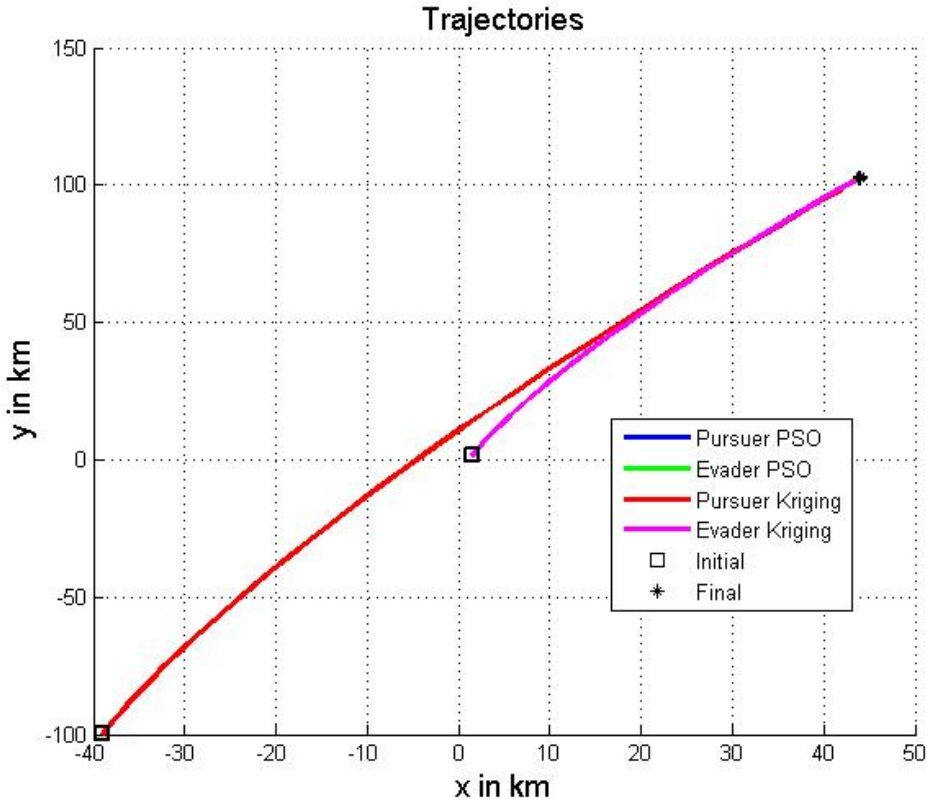

Figure 5.2: Comparing Trajectory Solutions from PSO and Kriging

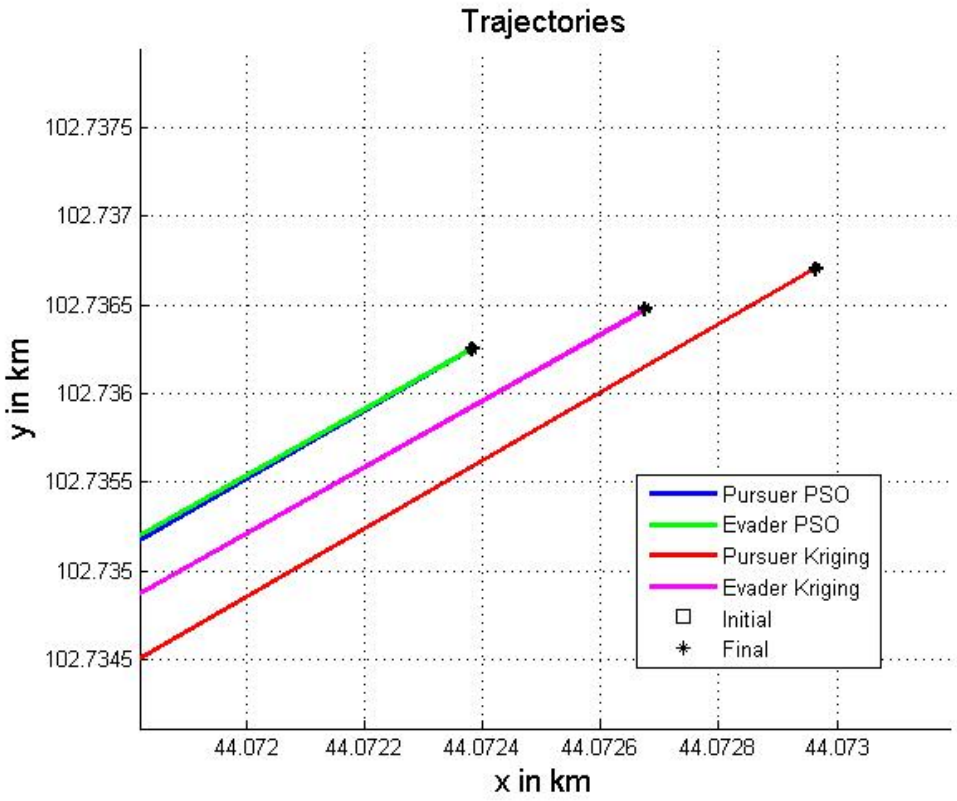

Figure 5.3: Magnified View Comparing Terminal Trajectory Solutions from PSO and Kriging

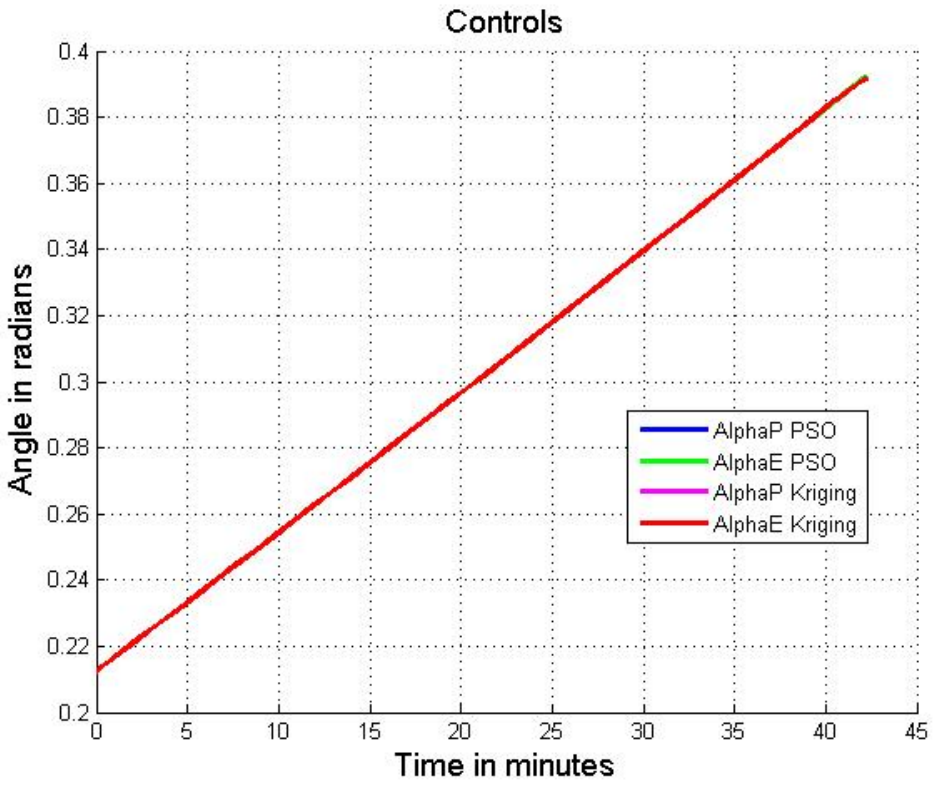

Figure 5.4: Comparing Control Time History from PSO and Kriging

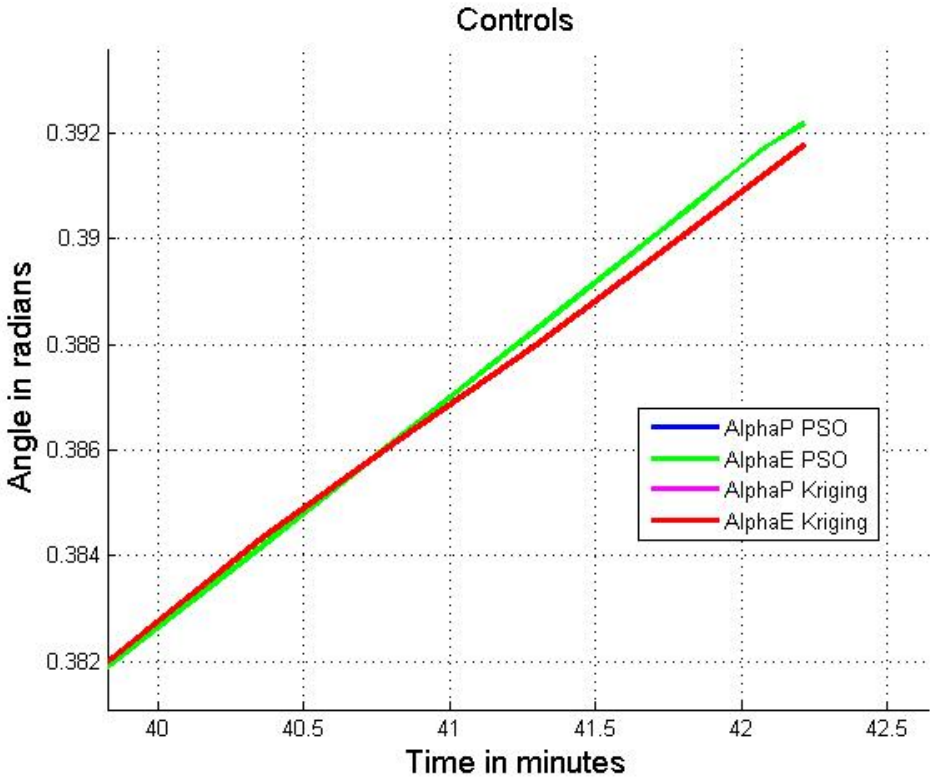

Figure 5.5: Magnified View Comparing Terminal Control Time History from PSO and Kriging

5.5 Sample Solution: Kriging Case 2

The second sample solution of kriging allows for perturbations in the initial positions of both the pursuer and the evader. Table 5.6 contains the initial conditions for both the pursuer and the evader for a nominal, unperturbed case. These nominal positions are the same as in Section 5.4. Once again, all the results are in 2D and the evader begins its motion at the origin of the CW reference frame. Random perturbations of less than 5 km are made in the initial positions of both the pursuer and the evader to create the field of 30 extremals shown in Figure 5.6.

Table 5.6: Nominal Initial Conditions Case 2

Vehicle	x (km)	y (km)	V_x (km/s)	V_y (km/s)
Pursuer	-38.9328	-100	0	0
Evader	0	0	0	0

The kriging parameters used to find an effective controller for this field of extremals are listen in Table 5.7. These values were determined primarily by trial and error.

Table 5.7: Kriging Calculation Parameters Case 2

Regression Order	2
Correlation Model	Exponential
Initial Guess Corr. Parameters	0.02
Lower Bound	10^{-5}
Upper Bound	2

The initial conditions chosen arbitrarily to test the success of the kriging in achieving an interception are listed in Table 5.8. The final positions of both the pursuer and the evader using the kriging feedback controller are shown in Table 5.9. The final time for this case is 41.8 minutes. For comparison, the final positions from the solution found by the PSO solver for the initial conditions in Table 5.8 are given in Table 5.10.

Table 5.8: Initial Conditions (Final time 41.8 minutes) Case 2

Vehicle	x (km)	y (km)	V_x (km/s)	V_y (km/s)
Pursuer	-34.7861	-99.9714	0	0
Evader	-1.1076	1.4647	0	0

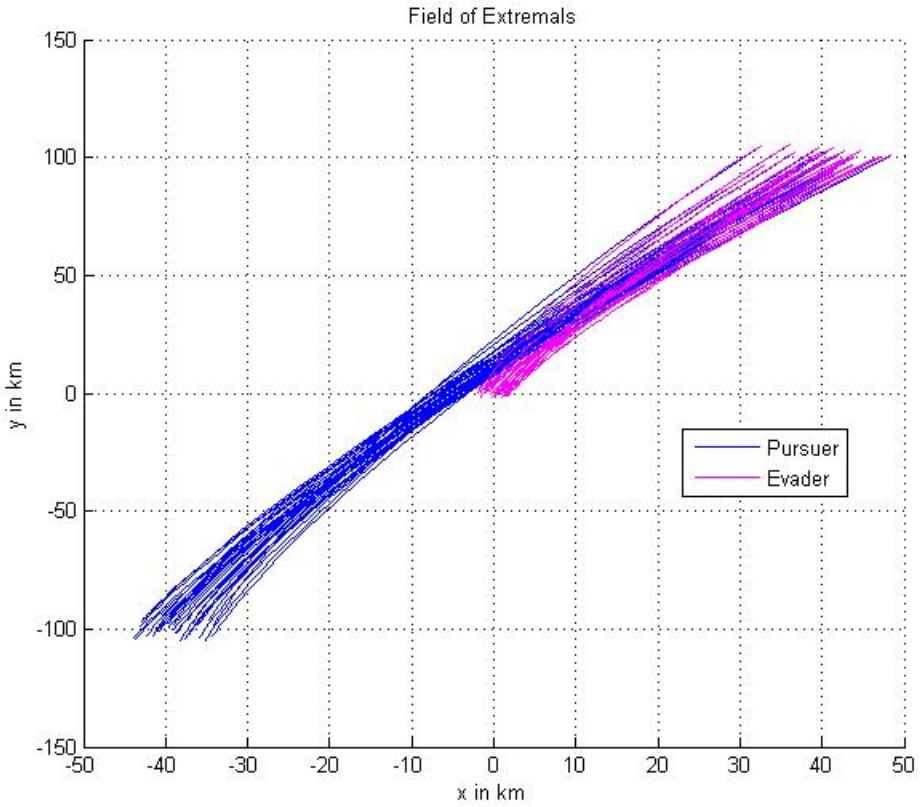

Figure 5.6: Field of Extremals - Perturbation in Both Players Initial Conditions

Table 5.9: Final Positions Kriging Case 2

Vehicle	x (km)	y (km)
Pursuer	34.1938	102.7025
Evader	34.1933	102.7027
Difference	0.000056	0.0000267

Table 5.10: Final Positions PSO (Final time 41.72 minutes) Case 2

Vehicle	x (km)	y (km)
Pursuer	34.194	102.703
Evader	34.194	102.703
Difference	0	0

As in Section 5.4, the kriging controller yields the same optimal trajectories as the open-loop PSO solver (starting from the same initial conditions) and achieves an accurate interception. The trajectories and control time histories are shown in Figures 5.7 and 5.8 respectively.

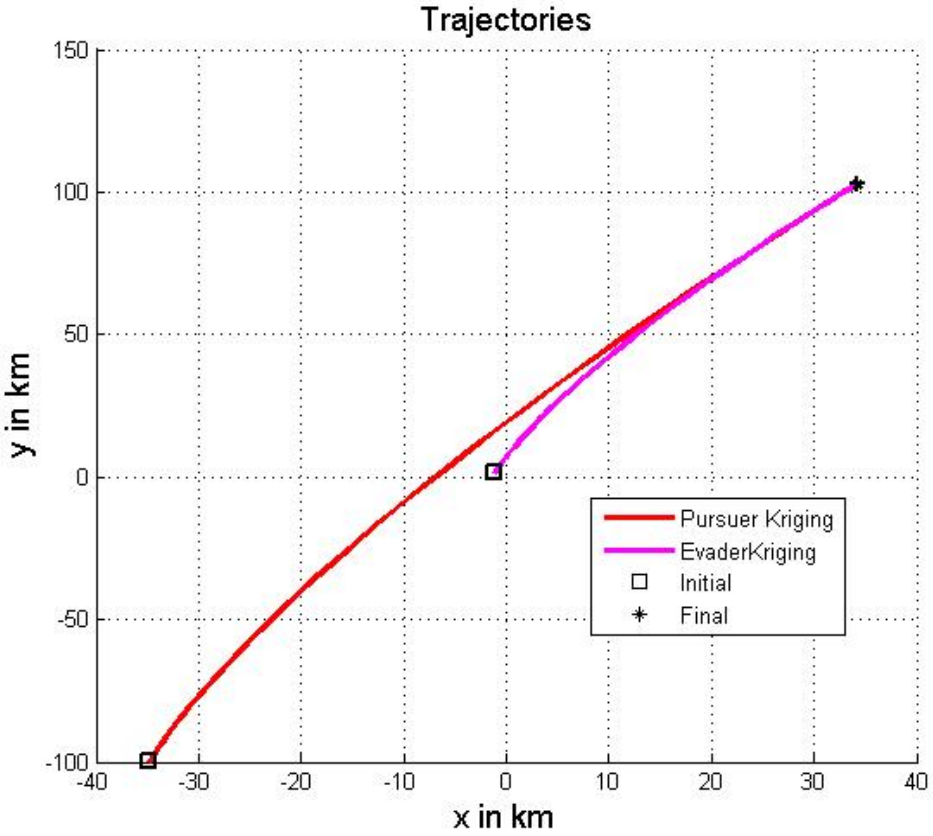

Figure 5.7: Kriging Pursuit/Evasion Trajectories Case 2

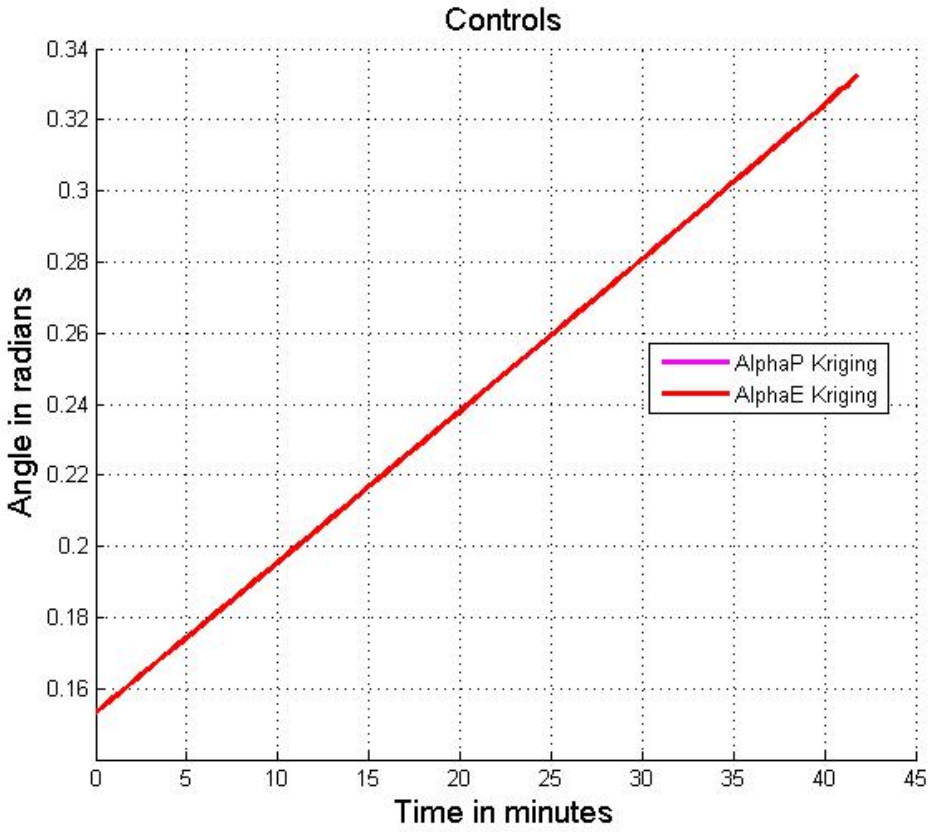

Figure 5.8: Control Time History for Kriging Case 2

5.6 Practical Limitations When Using Kriging

It seems reasonable that if the kriging calculations do not have enough data points, the interpolation will not be accurate. It is not intuitive, however, that kriging also breaks down if there are *too many* data points. In fact, when creating a controller with an average of 50 data points, the vehicles do not follow the optimal (open-loop) solution and do not achieve interception. With the same kriging parameters and initial conditions as in Tables 5.7 and 5.8, a controller created from 50 data points per extremal results in the final conditions shown in Table 5.11.

Table 5.11: Final Positions Kriging with Too Many Data Points

Vehicle	x (km)	y (km)
Pursuer	35.9491	106.7209
Evader	35.0875	104.7089
Difference	0.8616	2.0120

It was discovered that if there are too many data points, the correlation matrix used to calculate untrained states is too large for the kriging software to process accurately. The fields of extremals used to obtain the results in Sections 5.4 and 5.5 had an average of 50 data points per extremal after the initial calculation. Creating a controller directly from these fields gave poor results. The solution to this limitation is to use fewer data points via interpolation. The interpolation function "interp1" in MATLab was used to interpolate each state vs. time and each control vs. time individually. The default interpolation order (3) was used for each variable except for the controls, which used an interpolation order of 1 due to their linear nature. Using the interpolation, the number of data points was reduced to 32 for each extremal, yielding the results in Sections 5.4 and 5.5.

5.7 Kriging Summary

The interception errors for both examples of kriging, in Tables 5.5 and 5.9, are both larger than that of PSO, but still achieve interceptions to an accuracy of millimeters. This is a small loss of accuracy compared to the micrometers that the PSO achieves, but the CPU time gains outweigh the accuracy loss.

The a priori calculation of the field of extremals and the feedback controller takes approximately 4.5 hours. However, after the feedback controller is obtained, the CPU time of the kriging solution is an average of only 3 seconds. As discussed in Section 4.6, the average computation time of the PSO solution is on the order of 10 minutes.

CHAPTER 6

CONCLUSION

6.1 Summary

A differential game problem where one player's goal is to minimize the time until capture, while the other player wishes to maximize the time until capture, is called a minimax problem. In this work, the minimax problem between two satellites in the HCW reference frame is solved using Particle Swarm Optimization (PSO). PSO is a beneficial choice as a numerical solver because it does not require an initial guess to iterate upon. It is also a very simple and straightforward technique to implement, particularly as it requires no gradient information with respect to the system constraints or objective. PSO's global approach provides protection from the solver finding and remaining at a local minimum and makes it more likely the global minimum will be found. These benefits are balanced by a longer computation time than some of the more conventional numerical methods. If a mid-course correction were needed, a new solution directly from PSO would be impractical due to the computation cost. The method used in this paper, called kriging, provides an alternative to using PSO in real time. Instead, a closed-loop feedback controller is found a priori using a field of extremals. Using the feedback controller, the kriging result is both simple and fast to calculate - about 200 times faster than the PSO calculation. The speed of the kriging calculation implies that this method could be used as a real time controller. Together, PSO and kriging have been found to provide accurate solutions to both the open-loop and the closed-loop minimax problem.

6.2 Future Work

The results in this work are preliminary and there is much more exploratory work that remains to be done on the subject of applying the kriging method, originally intended for stationary systems, to dynamic systems. There are also several interesting cases with the pursuit/evasion problem that could be explored. Therefore, there are two distinct areas of focus for future work: kriging and the pursuit/evasion problem.

Kriging has only begun to be applied to dynamic systems, let alone orbit mechanics. There is still work to be done in verifying kriging's accuracy and fidelity. Also, there has been little research into applying kriging to problems with terminal constraints.

One major assumption that was made in this work was that the pursuer always had a thrust acceleration magnitude sufficiently higher than the evader, in order to ensure that capture does occur. Finding the situations with conditions such that the evader may escape capture altogether would be an important study if this methodology were ever used on real spacecraft.

REFERENCES

[1] R. Issacs, *Differential Games*. John Wiley and Sons, Inc., 1965.

[2] P. Ghosh and B. A. Conway, "A Direct Method for Trajectory Optimization Using the Particle Swarm Approach," 21 st AAS/AIAA Space Flight Mechanics Meeting, American Astronautical Society. New Orleans, 2011, paper 11-155.

[3] P. Ghosh and B. A. Conway, "Numerical Trajectory Optimization with Swarm Intelligence and Dynamic Assignment of Solution Structure," *Journal of Guidance Control and Dynamics*, vol. 35, no. 4, pp. 1178–1191, 2012.

[4] N. Cressie, "*Statistics for Spatial Data*," Wiley-Interscience, 1993.

[5] J. Martin and T. Simpson, "Use of Kriging Models to Approximate Deterministic Computer Models," *AIAA Journal*, vol. 43, no. 4, pp. 853–863, 2005.

[6] E. H. Issaks and R. M. Srivastave, "*An Introduction to Applied Geostatistics*," Oxford University Press, 1990.

[7] P. Ghosh and B. A. Conway, "Near-Optimal Feedback Strategies for Optimal Control and Pursuit-Evasion Games: A Spacial Statistical Approach," AIAA Guidence, Navigation, and Control Conference. Minneapolis, MN, August 2012.

[8] P. K. A. Menon and A. J. Calise, "Guidence Laws for Spacecraft Pursuit-Evasion and Rendezvous," AIAA Guidence, Navigation, and Control Conference. Minneapolis, MN, August 1988, technical Papers, Part 2 (A88-50160 21-08). Washington, DC, American Institute of Aeronautics and Astronautics. pp. 688–697.

[9] K. Horie and B. A. Conway, "Optimal Fighter Pursuit-Evasion Maneuvers Found via Two-Sided Optimization," *Journal of Guidance, Control and Dynamics*, vol. 29, no. 1, pp. 105–112, 2006.

[10] K. Horie, "Collocation with Nonlinear Programming for Two-Sided Flight Path Optimization," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2002.

[11] B. A. Conway and K. Horie, "A New Collocation-based Method for Solving Pursuit-Evasion (Differential Games) Problems," AAS/AIAA Astrodynamics Specialist Conference. Quebec City, Canada, August 2001, aSS Paper 01-445.

[12] M. Pontani and B. A. Conway, "Numerical Solution of the Three-Dimensional Orbit Pursuit Evasion Game," *Journal of Guidance Control and Dynamics*, vol. 32, no. 2, pp. 474–487, March-April 2009.

[13] M. Pontani and B. A. Conway, "Optimal Finite-Thrust Rendezvous Trajectories Found via Particle Swarm Algorithm."

[14] A. E. Bryson and Y. Ho, *Applied Optimal Control*. Taylor & Francis, LLC, 1975.

[15] B. A. Conway and S. W. Paris, "Spacecraft Trajectory Optimization Using Direct Transcription and Nonlinear Programming," in *Spacecraft Trajectory Optimization*, B. A. Conway, Ed. Cambridge University Press, 2010, ch. 3.

[16] H. B. Nielsen, "DACE: A MATLab Kriging Toolbox," Technical University of Denmark, 2007. [Online]. Available: http://www2.imm.dtu.dk/hbni/dace/

[17] S. r. N. Lophaven, H. B. Nielsen, and J. Sø ndergaard, *DACE: A MATLab Kriging Toolbox*.