AKSHAT AGARWAL, MIHIR DOSHI, KARAN IYER
FATIMA MEHDI, GLORIANE TONG
PROF. NAIMA HAMMOUD

DIFFERENTIAL GAMES AND THEIR APPLICATIONS

ANALYZING THE USE OF DIFFERENTIAL EQUATIONS IN GAME THEORY VIA THE PURSUIT-EVASION GAME

COURSE: ORDINARY DIFFERENTIAL EQUATIONS
NAIMA HAMMOUD
GROUP MEMBERS: AKSHAT AGARWAL – AA8161
MIHIR DOSHI – MDD421
KARAN IYER – KIC243
FATIMA MEHDI – YM1755
GLORIANE TONG – GET248

Abstract

This paper explores the intersection of differential equations and game theory in what is formally known as Differential Games. Game theory carries various applications in mathematics (as well as other social sciences) and is concerned with the analysis of strategies for dealing with competitive behavior. This paper deals with the pursuit-evasion game, in which the pursuer is trying to capture the evader, while the evader is trying to avoid capture. The general approach to a pursuit-evasion game will be explored in conjunction with specific examples.

Key Terms: Differential Games, Pursuit-Evasion, Zero Sum

Contents

List of Figures

List of Symbols

V_E	Evader's Velocity
V_P	Pursuer's Velocity
R	Pursuer's Turning Radius
u	Pursuer's Control (Scalar based on the minimum turning radius)
Ψ	Evader's control (Instantaneous bearing relative to the Pursuer)
φ	Bearing of Evader at Time of Capture
l	Capture Radius

1 Introduction

1.1 What is Game Theory - Anywhere

In 1944, mathematicians from Princeton, John van Neumann and Oskar Morganstern, co-authored a book entitled "The Theory of Games and Economic Behavior". While mathematics was used to compute problems and model trends, these authors claimed a new branch of mathematics was needed in order to properly describe economics, hence the creation of game theory.

In the world of evolutionary biology, changes in market signalling, and in the dynamics of bargaining, time is continuous. But despite whether or not time is continuous in mathematical models, differential equations can be applied. Game theory applications range from rock-paper-scissors to capture the flag and from surface-to-air missile trajectories to football plays. The common theme among all of these events is interdependence. Each player in any of these strategic situations chooses their course of action dependent on what the other players will do.

The most common examples of game theory in economics or political science include prisoner's dilemma, tragedy of the commons, matching pennies, and battle of the sexes all falling under categories of either cooperation or competition. Below are two baseline examples to set the tone of game theory.

Prisoner's Dilemma: Prisoner A and B have committed a crime. When the police question each one separately, whether or not a prisoner cooperates or defects affects the playoffs of both players. If one prisoner decides to cooperate with the police while the other prisoner does not, the cooperative prisoner will not receive any years of imprisonment, while the prisoner who does not cooperate gets 5 years in prison. If both prisoners cooperate with the police and confess, both will get 10 years in prison. If both prisoners defect and stay silent, both will get 1 year in prison. Neither prisoner knows how the other prisoner will act, but it is in each prisoner's best interest to cooperate, hoping that the other prisoner will defect. In this game, Nash equilibrium is reached when both prisoners defect, when Prisoner A defects and Prisoner B cooperates, or when Prisoner A cooperates and Prisoner B defects.

		Prisoner 2	
		Cooperate	Defect
Prisoner 1	Cooperate	-10, -10	-5, 0
	Defect	0, -5	-1, -1

Matching Pennies (each outcome sums up to zero, meaning the winner takes all and the loser takes all the cost): No matter the outcome of the game, one player will win the equal amount of what the other player loses.

		Player 2	
		Heads	Tails
Player 1	Heads	-1, 1	-3, 3
	Tails	0, 0	-2, -2

The concept of game theory is strategizing the best course of action based on all the possible actions other "players" will take. In a game, there consists of multiple players, the course of action the

players choose to take, and the payoffs that result. The strategy of the game requires players to take into account all possible actions of other players and choose the best choice for themselves. Depending on whether the game has complete or incomplete information, the players will know what the opponents will choose to play, and then proceed to make their decision based on that information set. Such games result in three types of outcomes: zero-sum (winner takes all and loser gets nothing), positive sum (both players gain) and negative sum (both players lose). Other more broad categories of game theory include simultaneous or sequential and cooperative or non-cooperative games. When all players reach an outcome, equilibrium is reached, and depending on the outcome, a specific term coined Nash equilibrium can be applied. As a background to game theory, Nash equilibrium is a state of equilibrium reached when both players are satisfied with the outcome and no player wants to change their outcome based on the other player's action. Specifically in our project, we will focus on zero-sum, cooperative, simultaneous games with complete information. But what happens when the actions of player A and player B are no longer static actions? What happens when the actions of players turn into a game of cat and mouse? When time is continuous and variables of motion start to depend on time, differential equations must be applied. This concept must incorporate the intersection of game theory and controlled systems (controlled by the players and modeled with differential equations).

1.2 Differential Games: The Pursuit Evasion Game - Isaacs, Chapter 1

A pursuit-evasion game as a finite extensive form is a generalization of a differential game. In other words, a pursuit-evasion game is an outline for differential games. A closer look at differential games and how they are relevant to game theory will be a concept explored all throughout this paper.

A dynamical system describes the changes a point will incur over time. This point is typically present in a geometric space. That being said, two definitions of time are classified. Those classifications being discrete time and continuous time. The two main Dynamical systems are Discrete time and Continuous time.

Discrete	Continuous
A. $t \in N$: time takes values in {0,1,2,...} B. Can be thought of as "steps" C. Examples: computers and digital systems	A. $t \in R \geq 0$: time takes values in $[0,\infty)$ B. Models of systems arising from (large-scale) physical phenomena C. Examples: airplanes, cars, room temperature, plants moving around the sun

Table 1: Discrete vs. Continuous Games

The two main dynamical systems, described in table 1 can be further illustrated by differential equations. Discrete-time Dynamical Systems are described by difference equations in the form:

$$x[t+1] = f(x[t]) \quad x(t) \in X$$

On the other hand, continuous-time Dynamical Systems are described by differential equations in the form:

$$\dot{x} = \frac{dx}{dt} = f(x[t]),$$

At this point, dynamical systems and the differential equations that represent them have been discussed. This discussion was intended to enhance mathematical skills since these differential equations will be used in pursuit evasion games.

A pursuit evasion game has at least two players.

Example 1: One Pursuer, One evader: An example of a simple pursuit evasion game can have one pursuer and one evader. To understand this better, let us explain an example where the pursuer is an airplane and the evader is a missile. The equations are created using dynamical systems.The goal behind this pursuit evasion game is for the airplane to optimally reach the missile and minimize the distance between them such that the Airplane can capture the missile. A dynamical system of equations would be of the form $\vec{x} = f(x, Player1Control, Player2Control)$ (the concept of controls will be elaborated on in further sections). This equation represents the position with the progression of time. A pursuit evasion game is varied depending on the game and number of pursuers and evaders. The most general case is n pursuers and n evaders, where n is the number of pursuers and evaders.

Although the simple pursuit evasion games above illustrate the concepts behind Pursuit Evasion games, it does not show the strong capabilities behind Pursuit Evasion Games. A more complex example, namely the Homicidal Chauffeur game, goes into detail.

1.2.1 A Real Life Example: Wargaming

As aforementioned, differential game theory was originally motivated by military efforts. While stationed inside a high security military base in Virginia, I (Gloriane) attended a course with other military personnel to learn about intelligence operations. As a part of our final exercise, we were tasked to split into two teams (countries) with the goal of capturing a port, which in turn, determines the winner of the battle. As shown in the diagram, the blue square and orange square represent the two opponents, the lines represent possible routes they can take, and the red circle symbolizes the target that must be captured. In order to win, either one country reaches the port and captures it first, or one team intercepts and "takes out" the opponent, also deeming a victory. As each country approaches its target, many factors must come into play such as platoon size, maneuvering ability in crowds during hours of work or prayer, width of roads and valleys, speed on incline surfaces, manpower, firepower, weather conditions, and much more. Differential games affect the speed factor for inclines, angle maneuvers, direction, and whether the country wants to intercept the other or capture the port. For any time that is chosen within certain bounds, the speed of how troops move will change depending on those factors aforementioned. On high inclines in hot weather with a low supply of water, troops may move slower and decide to change the trajectory towards the goal with a smaller distance. Whereas, the opposing country in hot weather going downhill may move quicker and decide to change their trajectory and target goal. The bottom line is that at any time during the operation, both "players" must choose an action that incorporates the use of differential equations with respect to time for speed.

2 Homicidal Chauffeur Game

2.1 The Question - Thesis

First, to define the parameters, there are state and control variables. Control variables determine the state variables, which must be known at any time, because the state variable changes as the game progresses. Since the state variable must be known at all times to all of the players, this is called a

complete information game.

The specific game we will be focusing on is the Homicidal Chauffeur, a variant of pursuit-evasion games. In it, a fast homicidal chauffeur (pursuer), though not very maneuverable, is trying to catch the evader in as little time as possible by getting him within a target radius l. In contrast, the evader, a slow runner (or a target) but highly maneuverable, is trying to escape from any contact of the pursuer for as long as possible.

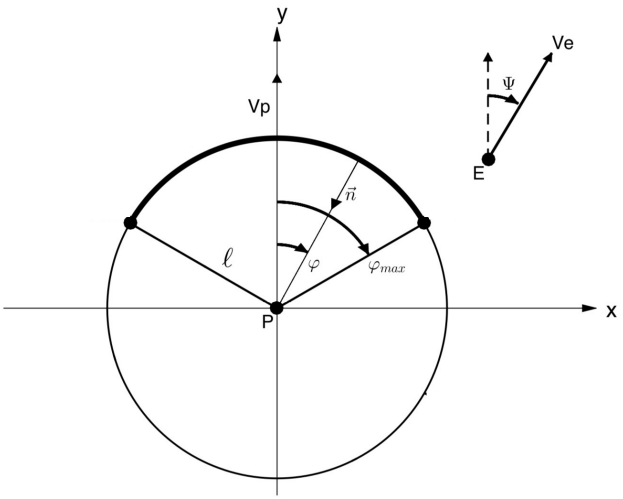

Figure 1: Capture Radius with Pursuer and Evader

The pursuer's velocity is V_P and its control u is a scalar that sets the minimal turning radius R of its turning circle. The Evader's velocity is V_E and its control, the scalar point Ψ is its instantaneous heading relative to the Pursuer's heading. The scalar φ is the bearing of the Evader at the time it is captured.

These conflicting objectives frame the question at hand: what variables must be controlled and strategies must be applied for the chauffeur to catch the runner at any infinite time? Similarly, under what conditions and strategies must the runner be able to evade the pursuer at any infinite time?

2.2 Defining the Reference Frame - Isaacs 2.1

Before we define our reference frame, we need to define control and state variables: control variables are, as the name suggests, under the control of the player at an instant. These are known to the player only. These, in turn, define the state variables, which are visible to all, and define the characteristics of the player's motion. This game takes place on a two-dimensional plane, but as we will see, the state variables occupy a 5-dimensional space. As we set up the pursuit-evasion game, there are a few salient points that must be kept in mind for defining our reference frame.

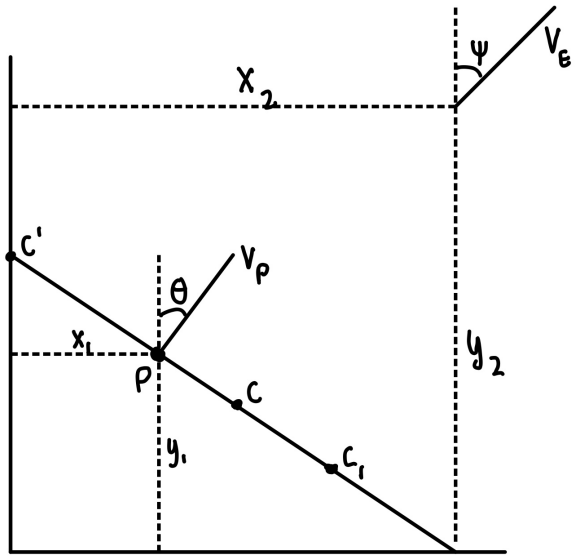

Figure 2: Preliminary Reference Frame

The pursuer moves with a fixed speed V_P and is subject to a restriction on steering at each instant, with a minimum radius of curvature R. It steers with an angle of θ, which is not a control variable in itself; it is dependent on the control variable u. Thus, it has three state variables: its "x"-position, "y"-position, and direction. These will be specified as x_1, y_1, and θ, respectively. We can set up the following differential equations for these variables:

$$\frac{dx_1}{dt} = V_P sin\theta, \quad \frac{dy_1}{dt} = V_P cos\theta, \quad \theta = \frac{V_P}{R}u$$

The evader moves with a fixed speed, V_E, which is slower than V_P, but has no restrictions on its steering at each instant. It steers with an angle of , a control variable. Thus, it has two state variables: its "x"-position and "y"-position, which will be represented as x_2 and y_2, respectively. We can set up the following differential equations for these variables:

$$\frac{dx_2}{dt} = V_E sin\Psi, \quad \frac{dy_2}{dt} = V_E cos\Psi,$$

In this situation, although the set of 5 equations presents a sound understanding of the situation, it may be difficult to computationally approach a system with 5 state variables such as this one. To help address this issue, we will define a "new set of coordinates". This will be referred to as the reduced space. Within this reduced space, we can use the coordinates x and y of the Evader's position as the sole state variables. The vertical axis will always be in the direction of the Pursuer's velocity. Such is now a "relative" frame, where the time derivatives $\frac{dy}{dt}$ and $\frac{dx}{dt}$ represent the relative change in position over time of the Evader relative to the Pursuer as the Evader's position changes.

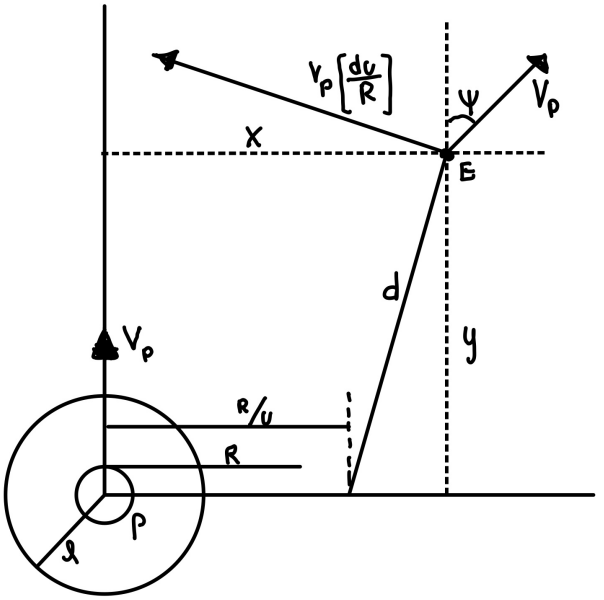

Figure 3: Reduced Space Coordinates

A vigorous geometric analysis, explained by Rufus Isaacs in *Differential Games: A Mathematical Theory with Application to Warfare and Pursuit, Control and Optimization* 2.2 provides a robust geometric analysis which results in the following reduced-space dynamic equations to model the motion of the system.

$$\dot{x} = \frac{dx}{dt} = -\frac{V_P}{R}yu + V_E sin(\Psi) \tag{1}$$

$$\dot{y} = \frac{dy}{dt} = \frac{V_P}{R}xu - V_P + V_E cos(\Psi) \tag{2}$$

2.3 The "Main Equation": minmax function - Isaacs 4.2

When working with a differential game such as the Homicidal Chauffeur game, we use a *minmax* strategy, where the goal is to make the best out of a worst-case situation. In his book, Rufus Isaacs derives the following function and names it the "*main equation*":

$$\max_u \min_\Psi [\overrightarrow{n} \cdot \overrightarrow{f}(x, y, u, \Psi)] = 0$$

where $\overrightarrow{n}$ is the inward pointing vector normal to the capture set, where the magnitude is the representation of the angle bearing of the captured evader, as follows:

$$\overrightarrow{n} = -(sin(\varphi), cos(\varphi))$$

and $\overrightarrow{f}$ represents the column vector of the x- and y- velocities previously established:

$$\overrightarrow{f}(x, y, u, \Psi) = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix}.$$

Note that in the case of a zero-sum game such as this one, where we observe the function at a fixed value of 0, the order with which we evaluate the minimum and maximum doesn't matter. In other words, it is equivalent to state the main equation as

$$\min_\Psi \max_u [\overrightarrow{n} \cdot \overrightarrow{f}(x, y, u, \Psi)] = 0.$$

3 Homicidal Chauffeur Exploration

3.1 Circular Capture Target

In section 2.1, we have obtained the motion dynamics for the problem at hand. To restate, the dynamic functions are:

$$\dot{x} = -\frac{V_P}{R} yu + V_E sin(\Psi) \tag{3.1}$$

$$\dot{y} = \frac{V_P}{R} xu - V_P + V_E cos(\Psi) \tag{3.2}$$

Such parameters are independent of the pursuer's capture set type. By applying the following non-dimensional parameters,:

$$x \to \frac{x}{R} \quad y \to \frac{y}{R} \quad t \to \frac{V_P}{R} t$$
$$l \to \frac{l}{R} \quad \mu \to \frac{V_E}{V_P}$$

Substituting the new parameters into the preliminary equations gives us two new equations:

$$\dot{x} = -yu + \mu sin(\Psi) \tag{3.3}$$

$$\dot{y} = xu - 1 + \mu cos(\Psi) \tag{3.4}$$

The initial conditions are:

$$x(0) = x_0 \ \ and \ \ y(0) = y_0, \quad where 0 \leq t \leq t_f$$

The above equations mimic the dynamics of the system. Suppose, in a simplified version of the game, the evader is steady/is not moving. The evaders velocity, V_E would be 0. Since μ is the ratio of the evader's velocity to the pursuers velocity, $\mu = 0$. In such a situation, the new dynamic equations can be rewritten as:

$$\dot{x} = -yu, \quad x(0) = x_0 \tag{3.5}$$

$$\dot{y} = xu - 1 \quad y(0) = y_0 \tag{3.6}$$

Based on the understanding of the "main equation" established in section 2.3, we can start to construct our *minmax* function:

$$\max_u \min_\Psi [\vec{n} \cdot \vec{f}(x, y, u, \Psi)] = 0 \tag{3.7}$$

The vector $\vec{n}$ is the inward-pointing normal to the capture set. In the Homicidal Chauffeur differential game with a circular capture set, becasue the Pursuer enforces penetration of the capture set, the normal to the capture set can be defined as:

$$\vec{n} = -(sin\varphi, cos\varphi) \tag{3.8}$$

$\vec{f}(x, y, u, \Psi)]$ is specified by a column vector with the two dynamic functions:

$$\vec{f} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -yu + \mu sin(\Psi) \\ xu - 1 + \mu cos(\Psi) \end{bmatrix} \tag{3.9}$$

We now substitute equations 3.8 and 3.9 into 3.7 and take a dot product to obtain our *minmax* function

$$\max_u \min_\Psi [-(sin\varphi, cos\varphi) \cdot \begin{bmatrix} -yu + \mu sin(\Psi) \\ xu - 1 + \mu cos(\Psi) \end{bmatrix}] = 0$$

$$\max_u \min_\Psi [sin\varphi(yu - \mu sin\psi) + cos\varphi(-xu - 1 + \mu cos\psi)] = 0$$

$$\max_u \min_\Psi [yu * sin\varphi - \mu sin\psi sin\varphi - xu * cos\varphi + cos\varphi + \mu cos\psi cos\varphi] = 0$$

$$\max_{u} \min_{\Psi} [yu * sin\varphi - xu * cos\varphi + cos\varphi + \mu cos\psi cos\varphi - \mu sin\psi sin\varphi] = 0$$

Set $u = 1$ (since we are trying to maximize it on the range [0, 1].

$$\max_{u} \min_{\Psi} [y * sin\varphi - x * cos\varphi + cos\varphi + \mu cos\psi cos\varphi - \mu sin\psi sin\varphi] = 0$$

We can remove all the terms with only φ as this is not a parameter in the *minmax* function.

$$y * sin\varphi - x * cos\varphi + cos\varphi + \min_{\Psi} [\mu cos\psi cos\varphi - \mu sin\psi sin\varphi] = 0$$

By factoring, we get

$$(1 - x)cos\varphi + y * sin\varphi + \mu \min_{\Psi} [\cos\psi cos\varphi + sin\psi sin\varphi] = 0 \tag{3.10}$$

This function can be converted into a *max* function by multiplying the objective function, $\mu \min_{\Psi} [\cos\psi cos\varphi + sin\psi sin\varphi]$, by -1.

$$(1 - x)cos\varphi + y * sin\varphi - \mu \max_{\Psi} [\cos\psi cos\varphi + sin\psi sin\varphi] = 0 \tag{3.11}$$

Recall that $cos(\alpha - \beta) = cos\alpha cos\beta + sin\alpha sin\beta$. We can thus rewrite 3.11 as

$$(1 - x)cos\varphi + y * sin\varphi - \mu \max_{\Psi} [\cos(\varphi - \psi)] = 0 \tag{3.12}$$

Since the capture set can be parametrized based on the following equations

$$x = l * sin\varphi \quad y = l * cos\varphi$$

Substituting these into 3.12 gives us:

$$(1 - lsin\varphi)cos\varphi lcos\varphi * sin\varphi - \mu \max_{\Psi} [\cos(\varphi - \psi)] = 0$$

$$cos\varphi - lsin\varphi cos\varphi + lcos\varphi sin\varphi - \mu \max_{\Psi} [\cos(\varphi - \psi)] = 0$$

$$cos\varphi - \mu \max_{\Psi} [\cos(\varphi - \psi)] = 0$$

At this point, it may be helpful to backtrack a bit and substitute μ for $\frac{V_E}{V_P}$

$$cos\varphi - \frac{V_E}{V_P} \max_{\Psi} [\cos(\varphi - \psi)] = 0$$

$$V_P cos\varphi - V_E \max_{\Psi} [\cos(\varphi - \psi)] = 0$$

Assuming that the evader will not travel backwards/towards the pursuer, we can confine its control ψ to the range $[\frac{-\pi}{2}, \frac{\pi}{2}]$. Since we are currently maximizing ψ, by substituting ψ for $\frac{\text{í}}{2}$, we get:

$$V_P cos\varphi - V_E \max_{\Psi} [sin\varphi] = 0$$

$$V_P cos\varphi - V_E sin\varphi = 0$$

Redistribution gives us

$$V_P cos\varphi = V_E sin\varphi$$

$$\frac{V_P}{V_E} = tan^{-1}\varphi$$

3.2 Linear Capture Range

In the previous section, we explore what happens if the pursuer has a circular capture range. For instance, assume that the pursuer's weapon can rotate in a circular path and has a range of length l. Now, let's look at a situation in which the pursuer has a linear capture range of length l as opposed to a circular one.

Equation 3.7 would remain the same. However, 3.8, the vector $\vec{n}$ representing the inward pointing normal, now becomes:

$$\vec{n} = -(1, 0) \tag{3}$$

By substituting into equation 3.7, we get:

$$\max_{u} \min_{\Psi}[-(sin\varphi, cos\varphi) \cdot \begin{bmatrix} -yu + \mu sin(\Psi) \\ xu - 1 + \mu cos(\Psi) \end{bmatrix}] = 0$$

$$\max_{u} \min_{\Psi}[-1(-yu + \mu sin\psi)]$$

Again, by setting $u = 1$ since we are maximizing it on the interval [-1, 1]

$$\min_{\Psi}[-1(-y + \mu sin\psi)]$$

Minimizing Ψ and setting it equal to $\frac{-\pi}{2}$, we get:

$$y - \mu = 0, \;\; where \;\; \mu = \frac{V_E}{V_P}$$

This yields a rather interesting result that in the situation of a linear capture target, the optimal occurs when y is

$$\frac{V_E}{V_P}$$

4 Numerical Solution

4.1 Methods and Initial Conditions

In this section, we will be trying to numerically solve a given Homicidal Chauffeur Problem using a method called the Boundary-Value Method (Johnson 2009), which uses a function called the Hamiltonian to convert the game into a boundary-value problem, which can be solved numerically in MATLAB. The initial conditions we have chosen to apply are:

$$x_1 = 0 \quad y_1 = 0 \quad x_2 = 5 \quad y_2 = 5 \quad \theta = 0 \tag{4.1}$$

We also set

$$V_p = 4 \quad V_e = 1 \quad u = 1 \quad l = 2 \tag{4.2}$$

4.2 Analytical Solution - Johnson 2.3

For the dynamics of the pursuer and evader, we use the 5 equations from the preliminary reference frame in 2.2 to form a vector

$$\dot{\vec{x}} = \begin{bmatrix} \dot{x_1} \\ \dot{y_1} \\ \theta \\ \dot{x_2} \\ \dot{y_2} \end{bmatrix} = \begin{bmatrix} V_P \cos\theta \\ V_P \sin\theta \\ u \\ V_E \cos\Psi \\ V_E \sin\Psi \end{bmatrix} \tag{4.3}$$

Note that we now consider angles from the horizontal for simplicity, and we are free to remove unnecessary parameters attached to the pursuer's control variable. We will also use the following condition ("terminal constraint") at the end of the game:

$$b = (x_1 - x_2)^2 + (y_1 - y_2)^2 - 4 = 0 \tag{4.4}$$

$4 = 2^2$, where 2 is the capture radius.
To solve this optimal control problem, we take the Hamiltonian defined as

$$\mathbf{H} = \lambda^T \dot{\mathbf{x}} + L \tag{4.5}$$

L is equal to zero in such an optimal control game. Finding the minimax of this optimal hamiltonian solves the game. Here, λ^T is defined by the Hamiltonian itself as:

$$\lambda^T = \frac{d\mathbf{H}}{d\mathbf{x}} \tag{4.6}$$

The components of $\lambda : \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$ can be thought of as Lagrange multipliers of the state equations, but are functions of time.

From the above, we get:

$$\mathbf{H} = \lambda_1 V_P cos\theta + \lambda_2 V_P sin\theta + \lambda_3 u + \lambda_4 V_E cos\Psi + \lambda_5 V_E sin\Psi \tag{4.7}$$

By differentiating, we find:

$$\dot{\lambda} = \begin{bmatrix} 0 \\ 0 \\ \lambda_1 * V_P sin\theta - \lambda_2 * V_P cos\theta \\ 0 \\ 0 \end{bmatrix} \tag{4.8}$$

Hence, all λ except λ_3 are constant.
From here, we can easily find the optimal play by the evader, which, to maximize, will merely be when the derivative of the Hamiltonian with respect to Ψ (its control variable) is 0. Differentiating, we have

$$\lambda_5 * V_E cos\Psi - \lambda_4 * V_E sin\Psi = 0 \tag{4.9}$$

This implies that

$$\Psi = tan^{-1} \frac{\lambda_5}{\lambda_4} \tag{4.10}$$

Such a value is a constant. We can now get the dynamics of the Evader by substituting the constant Ψ into its state equations. Below are the dynamics of the evader, which will be plotted in the MATLAB simulation:

$$\begin{bmatrix} \dot{x_2} \\ \dot{y_2} \end{bmatrix} = \begin{bmatrix} V_E * cos(tan^{-1} \frac{\lambda_5}{\lambda_4}) \\ V_E * sin(tan^{-1} \frac{\lambda_5}{\lambda_4}) \end{bmatrix} = V_E \begin{bmatrix} \frac{1}{\sqrt{1+(\frac{\lambda_5}{\lambda_4})^2}} \\ \frac{\frac{\lambda_5}{\lambda_4}}{\sqrt{1+(\frac{\lambda_5}{\lambda_4})^2}} \end{bmatrix} \tag{4.11}$$

For the pursuer, as u (the pursuer's control) is limited, we can analyze its minimization of the Hamiltonian. Clearly, it will be minimized when $\lambda_3 * u$ is minimum, which is -1 when $\lambda_3 > 0$, and 1 when $\lambda_3 < 0$. When $\lambda_3 = 0$, the pursuer does not need to turn, and the optimal play is 0. These rules can be used to program and plot the pursuer's dynamics when substituted in the state equations, based on Johnson's numerical solution (2009).

Furthermore, for the lagrange multipliers, we use the costate terminal condition, where

$$\dot{\lambda}^T = \frac{dJ}{d\dot{x}} + \alpha \frac{db}{d\dot{x}} \tag{4.12}$$

Here, α is a multiplier to be found and J is the cost function. As $\frac{dJ}{d\dot{x}}$ we can differentiate to get our final $\dot{\lambda}^T$.

$$\lambda_T(t_f) = \alpha \begin{bmatrix} 2(x_1(t_f) - x_2(t_f)) \\ 2(y_1(t_f) - y_2(t_f)) \\ 0 \\ 2(x_2(t_f) - x_1(t_f)) \\ 2(y_2(t_f) - y_1(t_f)) \end{bmatrix} \tag{4.13}$$

t_f is the final time.

We also need to consider the transversity condition for this free final time period [Joh09]:

$$\lambda_1 V_P cos\theta + \lambda_2 V_P sin\theta + \lambda_3(t_f)u + \lambda_4 V_E cos\Psi + \lambda_5 V_E sin\Psi + 1 = 0. \tag{4.14}$$

The above equations, initial conditions, and constraints will form the boundary of the boundary-value problem.

4.3 Implementation on MATLAB

We use MATLAB's extremely helpful bvp4c function to solve our boundary-value problem.

```
function homicidalChauffeurGameSolver

%%% First we will initialize and define important variables.

% Velocities:
vp = 4;
ve = 1;

% Pursuer's initial position and angle:
x1_initial = [0;0];
theta = 0;

%  Evader's inital position:
x2_initial = [5;5];

% Creating variables for the plot based on the above given initial arrays
x1_p = x1_initial(1);
x2_p = x2_initial(1);
theta_p =theta;
y1_p = x1_initial(2);
y2_p = x2_initial(2);
t_p = 0; %time

% Guess for alpha
alpha_initial = 0.25;

% The following are the initial guesses for the constant lambdas.

p1 = -4.5;
p2 = -4.5;

% Note that in our equations, p1=-p4 and p2=-p5, aiding us to make a
% guess that will accurately reflect the system.

p4 = 4.5;
p5 = 4.5;
```

```
% There must be a point when the pursuer must stop turning; if its heading
% is very close to the evader's [Johnson], we can set u=0 as the optimal play will no
% longer be to turn.

thresh = 0.05;

% First defining the system of differential equations for ALL variables,
% state and co-state (lambdas) and time. The function bvp4c solves taking
% this system into account.

function ode = ode_function(t,y,alpha)
    % State variables
    x1 = y(1);
    y1 = y(2);
    theta1 = y(3);
    x2 = y(4);
    y2 = y(5);
    % Lambdas
    p1 = y(6);
    p2 = y(7);
    p3 = y(8);
    p4 = y(9);
    p5 = y(10);
    % Time
    T = y(11);
    % Using the minimization principle for pursuer's control variable.
    u = -p3;
    if u > thresh
        u = 1;
    elseif u < -thresh
        u = -1;
    end
```

```
    %Stating the differentials for all the variables, which will be returned:

    x1dot = vp*cos(theta1);
    y1dot = vp*sin(theta1);
    theta1dot = u;
    x2dot = ve*(1/sqrt(1+(p5/p4)^2));
    y2dot = ve*(p5/p4)/sqrt(1+(p5/p4)^2);
    p1dot = 0;
    p2dot = 0;
    p3dot = p1*vp*sin(theta1) - p2*vp*cos(theta1);
    p4dot = 0;
    p5dot = 0;
    dt = 0;
    ode = T*[x1dot; y1dot; theta1dot; x2dot; y2dot; p1dot; p2dot; p3dot; p4dot; p5dot; dt];
end

% Now we define the boundary values based on information we know, which
% will also be fed into bvp4c:

function b = Boundary_condition(ya,yb,alpha)
    % lambda_3 is the 8th entry in our system.
    p3_bv = yb(8);
    u_bv = -p3_bv;
    if u_bv > thresh
        u_bv = 1;
    elseif u_bv < -thresh
        u_bv = -1;
    end
    % Defining these to save space in the final array element
    star1 = 1/sqrt(1+(yb(10)/yb(9))^2);
    star2 = (yb(10)/yb(9))/sqrt(1+(yb(10)/yb(9))^2);
    % The boundary conditions we use are:

    % Initial [signature call ya]: x1 (0) = 0, y1 (0) = 0, theta(0)=0,
    % x2 (0) = 0, y2 (0) = 0
```

```
    % Final [signature call yb]: lambda_1 (tf) = 2alpha(x1-x2),
    % lambda_2 (tf) = 2alpha(y1-y2), lambda_3 (tf) = 0, lambda_4 (tf) = 2alpha(x2-x1),
    % lambda_5 (tf) = 2alpha(y2-y1), our terminal constraint, and the
    % "transversality condition," formed by an "optimal Hamiltonian,"
    % which is of the form given in equation 4.14.

    % These are pasted into the array below, which is returned
    % by the function and used by bvp4c. Note that we consider 2*alpha as
    % alpha, and normalize our conditions for lambda_n.

    % Each element must be of the form of some function g(t)=0.

    b = [ya(1)-x1_initial(1);ya(2)-x1_initial(2); ya(3)-theta;
        ya(4)-x2_initial(1);ya(5)-x2_initial(2); ((sqrt((yb(5)-yb(2))^2 + (yb(4)-yb(1))^2)))-2;
        yb(6) - alpha*(yb(1)-yb(4))/((sqrt((yb(5)-yb(2))^2 + (yb(4)-yb(1))^2)));
        yb(7) - alpha*(yb(2)-yb(5))/((sqrt((yb(5)-yb(2))^2 + (yb(4)-yb(1))^2)));
        yb(8) - 0; yb(9) - alpha*(yb(4)-yb(1))/((sqrt((yb(5)-yb(2))^2 + (yb(4)-yb(1))^2)));
        yb(10)- alpha*(yb(5)-yb(2))/((sqrt((yb(5)-yb(2))^2 + (yb(4)-yb(1))^2)));
        yb(6)*vp*cos(yb(3))+yb(7)*vp*sin(yb(3))+yb(8)*u_bv+ yb(9)*ve*star1+yb(10)*ve*star2+1];
end

% The following space is needed for bvpinit preceding the bvp4c ODE
% function, which will solve this system of ODEs [of all parameters
% including time], as seen in Johnson's code and from MATLAB tutorials:

mesh = linspace(0,1,200);

% Initial guess for the ODE soltion, for every variable is done with
% "bvpinit," ([Joh09] and MATLAB tutorial.)

% The following value is an initial guess, using bvpinit.
solinit = bvpinit(mesh,@initial,alpha_initial);
```

```
function init = initial(a)
    initial_angle_guess = pi/6;
    time_guess  = 2;
    % Below, if the bearing of the evader is less than the upadated initial
    % guess, the program will guess a positive u (through lambda_3). Otherwise, knowing
    % the initial positions, u must be 0.
    if theta_p <  initial_angle_guess
        p3 = -1;
    else
        p3 = 0;
    end
    u_init = -p3;

    t = a*time_guess;
    dt = t - t_p;

    % We clarified the following relation earlier (4.3)
    theta1_d = u_init;
    theta1 = theta_p + dt*theta1_d;

    % From equation 4.3
    x1_d = vp*cos(theta1);
    y1_d = vp*sin(theta1);
    x2_d = ve*1/sqrt(1+(p5/p4)^2);
    y2_d = ve*1/sqrt(1+(p5/p4)^2);

    % Differential step
    x1 = x1_p + dt*x1_d;
    y1 = y1_p + dt*y1_d;
    x2 = x2_p + dt*x2_d;
    y2 = y2_p + dt*y2_d;

    % Function returns to bvp4c to be solved.
    init = [x1;y1;theta1;x2;y2;p1;p2;p3;p4;p5;time_guess];
```

```
    theta_p =theta1;
    x1_p = x1;
    y1_p = y1;
    x2_p = x2;
    y2_p = y2;
    t_p = t;
    end

% Feed solinit and the functions to bvp4c
sol = bvp4c(@ode_function,@Boundary_condition,solinit);

% As alpha is not a part of the system of ODEs but is nonetheless one of the
% unknown values we need, we can use sol.parameters (we specified alpha in
% solinit.
alpha = sol.parameters

% Plotting:

    hold on
    pursuer_position = sol.y(1:2,:);
    evader_position = sol.y(4:5,:);
    plot(evader_position(1, :),evader_position(2,:) ,'b--')
    plot(pursuer_position(1, :),pursuer_position(2,:) ,'r*-')
    plot(x2_initial(1),x2_initial(2),'bx');
    plot(x1_initial(1),x1_initial(2),'rd');
    legend('Evader Position','Pursuer Position','Evader Initial','Pursuer Initial');
    xlabel('x');ylabel('y');
    hold off

end
```

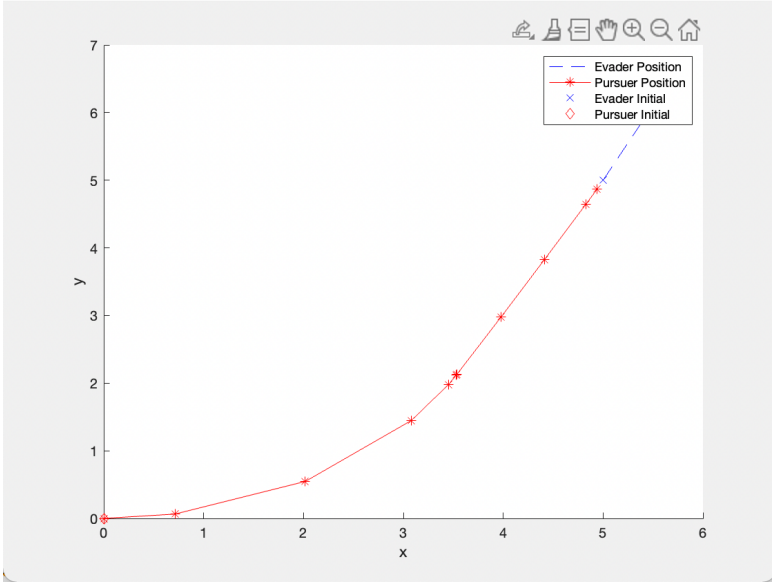

Figure 4: Plotting the Dynamical System

Here, we can the dynamics of the pursuer and the evader under optimal strategy, with the evader moving in a constant direction and with the pursuer turning until approximately reaching the same bearing as the evader, leading to the evader's inevitable capture due to speed difference.

5 Bibliography

References

[LL60] L.D. Landau and E.M. Lifshitz. *Mechanics*. English. Third. Vol. 1. Pergamon Press, 1960. ISBN: 0750628960. URL: https://cimec.org.ar/foswiki/pub/Main/Cimec/MecanicaRacional/84178116-Vol-1-Landau-Lifshitz-Mechanics-3Rd-Edition-197P.pdf.

[Ber92] Pierre Bernhard. "Differential Games : Lecture notes on the Isaacs-Breakwell Theory". PhD thesis. Cagliari, July 1992. URL: http://www-sop.inria.fr/members/Pierre.Bernhard/publications/ber92b.pdf.

[Isa99] Rufus Isaacs. *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Mineola, N.Y: Dover Publications, 1999. ISBN: 9780486406824.

[Li06] Dongxu Li. "MULTI-PLAYER PURSUIT-EVASION DIFFERENTIAL GAMES". English. PhD thesis. The Ohio State University, 2006. URL: https://etd.ohiolink.edu/apexprod/rws_etd/send_file/send?accession=osu1164738831&disposition=inline.

[Joh09] Philip Johnson. "Numerical Solution Methods for Differential Game Problems". English. PhD thesis. MASSACHUSETTS INSTITUTE OF TECHNOLOGY, June 2009. URL: https://core.ac.uk/download/pdf/4415133.pdf.

[BS15] Michael Brin and Garrett Stuck. *Introduction to dynamical systems*. Paperback edition with corrections. Cambridge, United Kingdom: Cambridge University Press, 2015. ISBN: 9781107538948.

[Coa17] Sean Coates. "AN INVESTIGATION OF THE HOMICIDAL CHAUFFEUR DIFFERENTIAL GAME". English. PhD thesis. Wright-Patterson Air Force Base, Ohio: AIR FORCE INSTITUTE OF TECHNOLOGY, Mar. 2017. URL: https://apps.dtic.mil/sti/pdfs/AD1054625.pdf.

[AD] HASSAN ALISHAH and PEDRO DUARTE. "HAMILTONIAN EVOLUTIONARY GAMES". English. URL: https://webpages.ciencias.ulisboa.pt/~pmduarte/Research/preprints/AD-HEG.pdf.

[BC] Olivier Bournez and Johanne Cohen. "Stochastic Learning of Equilibria in Games: The Ordinary Differential Equation Method". English. Centre National de la Recherche Scientifique, Laboratoire PRiSM Universit´e de Versailles 45, avenue des Etats-Unis, F-78000 Versailles. URL: https://www.lri.fr/~jcohen/documents/enseignement/Diff.pdf.

[WGP] Isaac Weintraub, Eloy Garcia, and Meir Pachter. English. In: *An Introduction to Pursuit-evasion Differential Games*.