ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE MADRID

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

DEVELOPMENT OF AN APPLICATION BASED ON DYNAMIC GAME THEORY FOR DESIGNING OPTIMAL TRAJECTORIES OF TWO UAVS IN A JAMMING PROBLEM

2016

JUAN PARRAS MORAL

Development of an application based on dynamic game theory for designing optimal trajectories of two UAVs in a jamming problem

Alumno: Juan Parras Moral

Tutor: Santiago Zazo Bello

Departamento de Señales, Sistemas y Radiocomunicaciones

Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

Keywords: Pursuit-evasion games, zero-sum games, Isaacs equations, mobile networks, UAVs

Resumen: El presente trabajo se divide en dos partes. En primer lugar, se realiza un estudio introductorio a la teoría de juegos diferenciales, prestando especial atención a una herramienta muy poco empleada en la ingeniería como son las ecuaciones de Isaacs. Se relacionan con la ecuación de Bellman y el principio del máximo de Pontryagin, mostrando que son una simplificación del segundo, y se estudia su campo de aplicación, haciendo especial énfasis en las condiciones que permiten resolver un juego empleando estas ecuaciones.

En segundo lugar, se estudia una aplicación concreta de esta herramienta a las telecomunicaciones, mediante el diseño de las trayectorias óptimas de dos drones en una situación de interferencia. Así, se plantea un juego donde un dron trata de comunicarse con estaciones base, que pueden ser fijas o móviles, y otro trata de interferir esa comunicación. El elemento a diseñar es la trayectoria óptima de cada uno de los drones, teniendo en cuenta las restricciones dinámicas de los mismos.

Para plantear y resolver esta situación, en primer lugar se estudia la capacidad de comunicaciones que se tiene en el entorno descrito, que a continuación es simplificada. A partir de aquí se exploran dos soluciones diferentes para obtener las trayectorias óptimas: por un lado, se plantea el juego en términos de persecución-evasión y se resuelve el problema, obteniendo los controles óptimos empleando para ello diferentes enfoques.

Por otro lado, se procede a plantear el problema como un juego de suma cero en términos de capacidad de comunicaciones y este problema es resuelto empleando las ecuaciones de Isaacs. Una vez que se obtienen los controles óptimos, es posible determinar las trayectorias de cada uno de los drones que resuelven el problema planteado.

Finalmente, se comparan ambas soluciones al problema, destacando las ventajas e inconvenientes de cada una de ellas y comparando las trayectorias a las que se llegan, todo lo cual repercutiría en el diseño real de un sistema que implementase alguna de estas soluciones para hacer frente al problema de interferencias planteado.

Parte de este trabajo se ha publicado en el Workshop on Statistical Signal Processing (2016), bajo el ttulo de “A new approach for solving anti-jamming games in stochastic scenarios as pursuit-evasion games”.

Abstract

This work presents two different sections. First, an introduction to differential game theory is presented, paying special attention to Isaacs' equations, which are a tool not often used in engineering. They are related to Bellman equation and Pontryagin maximum principle, being a simplification of the second, and their field of application is studied, with special emphasis in the conditions that allow a game to be solved using these equations.

Secondly, a concrete application of this tool to communications is studied: the design of optimal trajectories of two UAVs in a jamming environment. Hence, a game is posed where one UAV tries to communicate with relay stations, that can be fixed or mobile, and the other UAV tries to jam that communication. The design target is the optimal trajectory of each UAV, taking into account their dynamical restrictions.

In order to pose and solve this game, first, the communications capacity in the described environment is studied and simplified. Two different solutions are studied in order to obtain optimal trajectories: on the one hand, a pursuit-evasion game is posed and solved, obtaining the optimal controls using different approaches.

On the other hand, the game is posed as a zero-sum game in terms of communications capacity and this problem is solved using Isaacs equations. With the optimal controls, it is possible to obtain the trajectories of each UAV that solve the problem posed.

Finally, both solutions to the problem are compared, highlighting both their advantages and drawbacks and comparing the resulting trajectories. All this would have an impact on the actual design of a system that would implement any of the proposed solutions to face the problem of jamming described.

Part of this work was published in the Workshop on Statistical Signal Processing (2016), under the title “A new approach for solving anti-jamming games in stochastic scenarios as pursuit-evasion games”.

Madrid, June 15, 2016

Contents

List of Figures

List of Tables

Acronyms

CDF	Cumulative Distribution Function
CLPS	Closed Loop Perfect State
HJB	Hamilton - Jacobi - Bellman
ME	Main Equation
ODE	Ordinary Differencial Equation
RPE	Retrogressive Path Equation
SINR	Signal to Interference and Noise Ratio
SNR	Signal to Noise Ratio
SOO	Simultaneous Optimistic Optimization
UAV	Unmaned Aerial Vehicle

Chapter 1

Introduction

In the last years, a large research has been done related to Unmanned Aerial Vehicles (UAVs), either in military or civil scenarios. In a formation, communication between vehicles must be wireless. Thus these links are vulnerable to jamming attacks. This is an area of research where different attack / defense strategies have been proposed. A wide variety of techniques are used, as spectral channel surfing and spatial positioning of the nodes [1], game theory tools [2], [3], [4], [5] or the use of a honey-pot node [6]. A general survey of jamming techniques is presented in [7].

In case that the jammer and communicating nodes are mobile, the attack can be modeled as a zero-sum, non-cooperative, differential game [8]. There are several tools dedicated to analyze this kind of games, especially for two player games [9], [10], [11], [12]. There are specific solutions for some multi-player games, such as [13], [14], [15], [16], [17]. The main tools used are the Hamilton-Jacobi-Bellman-Isaacs equations, which are difficult to solve to obtain an analytical solution. In some specific games, the game can be solved using only Isaacs equations [10], which greatly simplifies the analysis.

In this work, the case in which there is one UAV trying to communicate with relay nodes while another UAV which tries to jam the communications is modeled using differential game theory. The relays can be static or dynamic but their exact position is unknown. The main contribution of the master thesis is posing the problem in terms of optimizing capacity and under some hypotheses, approximating it as a pursuit-evasion game using Isaacs' tools. To the best of our knowledge, this has not been done yet and allows obtaining a new approach in which communications related problems can be solved using well known pursuit-evasion game tools.

Part of the present thesis was submitted and published on IEEE Workshop on Statistical Signal Processing (2016), held in Palma de Mallorca, under the title "A new approach for solving anti-jamming games in stochastic scenarios as pursuit-evasion games" [18].

In chapter 2, we give a brief introduction to differential game theory and present Isaacs equations. Then, in chapter 3, we describe the jamming problem that we pose and obtain the expression for total system capacity. After, in chapter 4, we solve the game posed in chapter 3 approximating it as a pursuit-evasion game. Next, in chapter 5, the capacity game is solved. Both game results are compared in chapter 6 and the main conclusions are outlined in chapter 7.

Chapter 2

General framework of differential games

2.1 Introduction to game theory

Game theory [19] is a branch of mathematics that deals with interactions among multiple decision makers, which are called players. Each one of the players has a preference or a target that she wants to obtain, which is represented as an objective function. A player tries to optimize her own objective function, which generally depends on the actions of other players, which means that a player cannot optimize her objective function independently of the rest of players.

Games can also be classified as cooperative or non-cooperative. The former case models how agents compete and cooperate as coalitions in order to create value, since they have common objectives, whereas the latter models the actions of agents trying to maximize their own objective function [20]. In non-cooperative games games, the solution concept that is used is a Nash equilibrium, named after the mathematician John Nash who introduced and proved this concept [21] [22]: a Nash equilibrium is such that none of the players can improve her payoff by a unilateral move. There might be different Nash solution points to a game.

Another classification of games is done in terms of the time [23]: if the game takes place instantaneously and not over a whole interval of time, it is called static. If it happens otherwise and hence, a player takes different decisions over time, the game is called dynamic. Also, in the latter case, the objective function of the players depend on a state which changes with time. In static games, the action of each player is a single choice, whereas in dynamic games, each player makes various actions, which are collected by her strategy, which is a function of time.

In the case of dynamic games, the time interval over which the game takes place can be finite, that is, $t \in [0, t_f]$ or infinite, when $t \in [0, \infty)$: that causes games to be of finite or infinite horizon. Also, it is possible that this time is discrete or continuous, in the second case, the game is usually called differential game.

A final classification of games is done in terms of the objective functions of the players [19]: if they can be made zero after appropriate positive scaling and/or translation that do not depend on the decision variables of the players (i.e., their actions or controls) the game is called zero-sum; and nonzero-sum otherwise.

Game theory can be seen as a generalization of optimization and control theory. The first deals with problems in which an agent tries to optimize an objective function, and hence, it can be seen as a particular case of a static game, with only one player involved. Game theory is also related to multi-objective optimization, in which there are more than one objective functions. Control theory, on the other hand, deals with obtaining a function of time that optimizes a functional, and hence, it can be seen as a particular case of differential games, with only one player involved. Thus, optimization, control theory and game theory are closely related, being the last a generalization of the first two.

2.2 Introduction to differential games

A differential N-player game, with N players and where $\mathscr{N} := \{1, .., N\}$ is the players set, has the following elements:

- A continuous time interval, $t \in [0, t_f]$, where t_f is the final time of the game. This interval denotes the duration of the evolution of the game, which can be finite in case that $t_f < \infty$ or infinite otherwise. In this work, we will study finite horizon games, and hence, the condition that $t_f < \infty$ will always be satisfied.

- A trajectory space, denoted by $\mathscr{S}$, which is an infinite set whose elements are the permissible state trajectories, denoted as $\{x(t), 0 \leq t \leq t_f\}$. For each fixed $t \in [0, t_f]$, $x(t) \in S^0$, where S^0 is a subset of a finite-dimensional vector space. The trajectories $x(t)$ describe the state of each player in each time instant.

- An action space for each of the N players, denoted by $\mathscr{U}^i$, which is an infinite set defined for each $i \in \mathscr{N}$. The elements of this set are the permissible controls of player i. There exists a set $S^i \subseteq \mathbb{R}^{m_i}$ $(i \in \mathscr{N})$ so that for each fixed $t \in [0, t_f]$, $u_i(t) \in S^i$. The controls will be functions of the time and the game solution searches for the optimal control function for each one of the players that drive the game to a Nash equilibrium situation.

- A differential equation, called the dynamics equation, which defines how the states vary with time as a function of the players controls, states and time. The solution to this equation describes the state trajectory of the game as a function of controls and initial state (i.e., x_0). Its form, hence, will be:

$$\frac{dx(t)}{dt} = f(t, x(t), u_1(t), ..., u_N(t)), x(0) = x_0 \tag{2.1}$$

- A set-valued function $\eta^i(t)$, which determines the information that is available to player i at time t. Depending on the information that is available to each player in each moment of the game, it is possible to classify games according to, at least, two different patterns [8]:

1. Open loop pattern, if $\eta^i(t) = \{x_0\}, t \in [0, t_f]$. The player can only access the initial state of the game
2. Closed loop perfect state information (CLPS), if $\eta^i(t) = x(s), \forall s \in [0, t]$. The player has access, in every stage of the game, to the current, past and initial states.

- Two functionals for each player, $G^i : S^0 \to \mathbb{R}, L^i : [0, t_f] \times S^0 \times S^1 \times ... \times S^N \to \mathbb{R}$, defined for each $i \in \mathscr{N}$, so that the cost functional of player i, denoted by $\pi^i(x(t), u_1(t), ..., u_N(t))$ is well defined. Its form is:

$$\pi^i(x(t), u_1(t), ..., u_N(t)) = \int_0^{t_f} L^i(t, x(t), u_1(t), ..., u_N(t))dt + G^i(x(t_f)) \quad (2.2)$$

  This cost functional is the objective function (functional, in this case) that each player seeks to optimize. L^i is called the running cost and G^i is the terminal cost, the former being the cost incurred while the game is being played -that is, when $t \in [0, t_f]$ -, and the latter being the cost that adds up in a particular terminal state.

2.3 Standard methods for solving differential games

In order to solve a differential game, the information structure $\eta^i(t)$ plays a key role in the solution procedure used [24, pp 22-32]. Mainly, two approaches are followed: the maximum principle of optimal control, developed by Pontryagin [25] is used to solve open loop games, whereas the principle of dynamic programming by Bellman [26] is used to solve closed loop, perfect state information games.

If the information structure follows an open loop pattern, it is satisfied for all players that $\eta^i(t) = \{x_0\}, t \in [0, t_f], i \in \mathscr{N}$. Each player can only access the initial state of the game and this information allows each player to know the optimal trajectories of the others. Hence, the controls become a function of initial state and time. The solution to this problem uses the maximum principle of Pontryagin and is characterized using the following theorem [24, pp 24-25]:

Theorem 1. *A set of strategies $\{u_i^*(t), for\ i \in N\}$ provides an open loop Nash equilibrium solution to the game in Section 2.2, being $\{x^*(t), t \in [0, t_f]\}$ the corresponding state trajectory, if there exist m costate functions $\Lambda^i(t) : [0, t_f] \to \mathbb{R}^m$, for $i \in N$, such that the following relations are satisfied:*

- $u_i^*(t) = \arg\max_{u_i}\{L^i(t, x^*(t), u_1^*(t), ..., u_N^*(t)) + \Lambda^i(t)f(t, x^*(t), u_1^*(t), ..., u_N^*(t))\}$
- $\dot{x}^*(t) = f(t, x^*(t), u_1^*(t), ..., u_N^*(t)),\ x^*(0) = x_0$
- $\dot{\Lambda}^i(t) = -\frac{\partial}{\partial x^*}\{L^i(t, x^*(t), u_1^*(t), ..., u_N^*(t)) + \Lambda^i(t)f(t, x^*(t), u_1^*(t), ..., u_N^*(t))\}$
- $\Lambda^i(t_f) = \frac{\partial}{\partial x^*}\{G^i(x^*(t_f))\}$

for $i \in N$

This theorem could also be used to obtain solutions under closed loop information structure, however, the partial derivative with respect to x in the costate equations would receive contributions from dependence of the others $n-1$ players' strategies on the current value of x, which complicates the solution. Another problem is that there would be, in general, an uncountable number of solutions, due to information non-uniqueness, if it were applied to games with closed loop information structure.

In order to avoid these issues, closed loop perfect state (CLPS) information structure is used, where the players have access to every past and present state of every player. The solution to this problem uses Bellman's dynamic programming principle and is characterized using the following theorem [24, p 28]:

Theorem 2. *A set of strategies $\{u_i^*(t), \text{for } i \in N\}$ provides a feedback Nash equilibrium solution to the game in Section 2.2, if there exist continuously differentiable functions $V^i(t,x) : [0,t_f] \times \mathbb{R}^m \to \mathbb{R}, i \in N$, satisfying the following set of partial differential equations:*

- $-\frac{\partial V^i(t,x)}{\partial t} = \max_{u_i}\{L^i(t,x^*(t),u_1^*(t),...,u_N^*(t))+\frac{\partial V^i(t,x)}{\partial x} f(t,x^*(t),u_1^*(t),...,u_N^*(t))\}$
- $V^i(t_f,x) = G^i(x)$

for $i \in N$

2.4 Pursuit-evasion games

Let us particularize the expressions in Section 2.2 for a two player, zero-sum, pursuit evasion game. Being two-player means that $\mathscr{N} := \{1,2\}$, that is, there are $N = 2$ players, called pursuer and evader respectively. Pursuer tries to catch evader, whereas evader seeks to flee from pursuer. Their controls will be called $\phi(t)$ and $\psi(t)$, and the dynamics equation will be provided by the concrete setup of the game. The state vector will be called $x(t)$, and the game set, which is the set of all allowed states, will be supposed to be independent of the player's decisions (otherwise, generalized Nash Equilibrium theory should be used). Both players will have the same cost functional with opposite sign, and hence, the rewards add up zero, thus, the game will be zero-sum. That means that both players have the same reward with opposite sign, and therefore, the gains of one player are the losses of the other. Hence, one player tries to maximize the objective or payoff function and the other tries to minimize it. This payoff function is given by the following functional, which comes from (2.2):

$$\pi\left(x(t),\phi(t),\psi(t)\right) = \int_0^{t_f} L\left(x(t),\phi(t),\psi(t)\right)dt + G\left(x(t_f)\right) \tag{2.3}$$

In a Pursuit-Evasion game, final and running cost are $G = 0$ and $L = 1$ respectively; and thus, the payoff function will be $\pi = t_f$. This final time takes place when pursuer catches the evader, and it is often called capture time. Therefore, pursuer will try to minimize the capture time and evader will try to maximize it. Substituting in (2.3) it yields the following payoff function:

$$\pi[x(t),\phi(t),\psi(t)] = t_f \tag{2.4}$$

The game outcome obtained if both players implement their optimal strategy will be called value function $V(x) = \pi[x(t), \phi^*(t), \psi^*(t)]$, where ϕ^* denotes the optimum value of ϕ and ψ^* is the optimum value of ψ, for any state $x(t)$ in the state space. The gradient of the value function will be denoted as ∇V. Since the nature of the equilibrium searched is Nash, that means that any unilateral deviation of a player from its optimal strategy causes that the other player could obtain a better payoff - i.e, a pursuer not following her optimum strategy will allow t_f to increase, and the other way around also applies. Lastly, the concrete setup of the system will provide the dynamic equation, which will be expressed in the following form: $\dot{x} = f\left(x(t), \phi(t), \psi(t)\right)$.

Finally, a key element of the solution procedure is the Hamiltonian, which is built using the dynamics equation, the gradient of the value function and the running cost of the game as follows:

$$H(x, \nabla V, \phi, \psi) = \nabla V^T f(x, \phi, \psi) + L(x, \phi, \psi) = \nabla V^T f(x, \phi, \psi) + 1 \qquad (2.5)$$

where ∇V^T is the transposed of the vector ∇V.

2.5 Isaacs' approach

Apart from the methods described in Section 2.3, another approach can be used to solve certain kind of games: Isaacs' equations [10]. This method can be used to solve open loop games, which satisfy the following conditions:

- The game is two players, zero-sum, pursuit-evasion type. Being a pursuit evasion game implies that final time is free (i.e., to be optimized), but this condition can be relaxed [10, p. 34].
- The Hamiltonian is separable on its controls [10, p. 35].

If these hypotheses are satisfied, the Hamiltonian satisfies the following conditions along the optimal trajectories (where ϕ^* and ψ^* are the optimal controls of each of the two players):

1. $H(x, \nabla V, \phi, \psi^*) \leq H(x, \nabla V, \phi^*, \psi^*) \leq H(x, \nabla V, \phi^*, \psi)$
2. $H(x, \nabla V, \phi^*, \psi^*) = 0$

The first condition means that any unilateral deviation by the pursuer leads to a smaller Hamiltonian value (and any unilateral deviation by the evader leads to a larger Hamiltonian value), which is the Nash equilibrium definition. The second condition means that, when both players use their optimal controls, the Hamiltonian is zero.

The method used by Isaacs has the following steps, where ϕ will be the pursuer's control and ψ the evader's control:

- First, the system states must be defined, and a dynamics equation that relates states with controls must be obtained. This dynamics equation will have the following form:

$$\frac{dx(t)}{dt} = f(x(t), \phi(t), \psi(t)) \qquad (2.6)$$

- Secondly, the Hamiltonian must be built and optimized. This is done using Isaacs "Main Equation 1", which is the Hamiltonian:

$$\max_{\psi} \min_{\phi} \sum_{i} V_{x_i} f_i + L = 0 \tag{2.7}$$

  where V_{x_i} stands for the partial derivative, that is, $V_{x_i} = \frac{\partial V}{\partial x_i}$ and f_i is the i-th component of $f(x(t), \phi(t), \psi(t))$ (2.6).

  This expression must be posed and solved in order to obtain the optimal controls, which after being substituted into the Hamiltonian lead to the optimal Hamiltonian, denoted by H^*.

- Thirdly, the optimal trajectories are obtained using a backward procedure in which the Retrogressive Path Equations (RPE) play a key role. These equations are a function of retro-time τ, which is the time-to-go, obtained using the following variable change:

$$\tau = t_f - t \tag{2.8}$$

  where t_f is the termination time of the game. Intuitively, τ is a backward time: it goes from final time t_f until initial time $t = 0$. Hence, initial conditions in τ will be final conditions in time and the other way around.

  There will be two different RPEs equations. The first kind depend on the states and are obtained from the dynamics equation (2.6). These RPEs have the following form:

$$\frac{dx(t)}{dt} = f(x(t), \phi(t), \psi(t)) = -\frac{dx(\tau)}{d\tau} = \overset{\circ}{x}(\tau) \tag{2.9}$$

  where $\overset{\circ}{x}$ denotes the derivative of x with respect to retro time τ and $x(\tau) = x(t)|_{t=\tau}$. That means that these RPE are obtained changing the sign of the dynamic equation.

  The second kind of RPEs depend on the gradient of the value function. Along the optimal trajectory, the following adjoint equation, holds:

$$\frac{d}{dt}\nabla V[\mathbf{x}(t)] = -\frac{\partial}{\partial x} H(\mathbf{x}, \nabla V, \phi^*, \psi^*) \tag{2.10}$$

  Using the variable change (2.8), the adjoint equation becomes:

$$\frac{d}{d\tau}\nabla V[\mathbf{x}(\tau)] = \frac{\partial}{\partial x} H(\mathbf{x}, \nabla V, \phi^*, \psi^*) \tag{2.11}$$

  Hence, the RPEs related to the gradient are also related to the left hand side of the "Main Equation" (ME) (2.7), according to this expression [10, p. 82]:

$$\overset{\circ}{V_k} = \frac{\partial H}{\partial x_k} = \frac{\partial ME}{\partial x_k} \tag{2.12}$$

  where x_k refers to the states.

- In order to solve the RPEs, initial conditions in retro-time are needed. The terminal surface is defined as a manifold, denoted by h, which is parametrized using $n-1$ variables (where n is the number of states). Each of these variables will be called $s_i, i \in 1, ..., n-1$. These will be initial conditions in τ (in time t, they are final condition), and they are obtained using the following expression:

$$\frac{\partial G}{\partial s_k} = \sum_i V_{x_i} \frac{\partial h}{\partial s_k} \tag{2.13}$$

  where G is the final cost of the game considered, h the terminal manifold and s_k the variables used to describe this manifold. It is important to recall that the conditions obtained with this procedure will be final conditions in time, but initial conditions in retro-time τ, as can be derived from (2.8). This will give place to a problem when solving these equations: whereas the RPEs equations will be a function of final conditions in time, we will only have initial conditions in time.

- Once that final conditions in time are obtained, the RPEs are integrated in order to find out the optimal trajectories and the optimal controls for the posed game. However, as we showed before, these trajectories will be function of final time conditions, but we only know initial time conditions. In order to solve this issue, the final time t_f must be obtained in order to obtain a system of equations that may allow us to obtain these final conditions in time from the initial ones. In doing this, the following vectorial identity is used, where s are the final conditions, initial state x_0 are the initial conditions and T are the trajectories obtained after integrating the RPEs. The solutions of this equation system are the final conditions, depending on initial ones; by substituting these values on the trajectories equations, the dependency on initial conditions appears.

$$T(\tau, s) = T(t_f - t, s) = T(t_f, s) = x_0 \tag{2.14}$$

2.6 Comparison of Isaacs with Bellman and Pontryagin approaches

Isaacs' method described above is closely related to Pontryagin approach to solve games. If we compare Theorem 1 with Isaacs equations, it is possible to see that the first point of the Theorem corresponds to Isaacs' Main Equation 1 (2.7), the second one is the dynamics equation as appears in (2.6) and the third point is the adjoint equation which Isaacs includes in (2.10). Pontryagin uses costate functions, that he calls $\Lambda(t)$, which can be identified with the gradient of the value function ∇V that Isaacs uses. Also, the final conditions on costate functions from Pontryagin and gradient of the value function that Isaacs used are obtained through partial derivatives of the final cost, as in (2.13) and the fourth point of Theorem 1.

Hence, it is possible to see that Isaacs equations are actually a particularization of Pontryagin's method, for the concrete case that the game is zero-sum, two players and that controls are separable. Thus, it can be used to obtain open loop solution to games that fall into this category.

Isaacs method is also related to Bellman method. The latter allows us to solve games using feedback information structures, at the cost of solving a partial differential equation. The former allows us to solve open loop games - in order to avoid information non-uniqueness-, where ordinary differential equations are to be solved [27]. Let us start from Hamilton - Jacobi - Bellman equation (HJB), which comes from the first point in Theorem 2, using the definition of Hamiltonian from (2.5) and states that:

$$H^* + \frac{\partial V}{\partial t} = 0 \tag{2.15}$$

Isaacs' main equation [10, p. 67] can be seen as a particular case, when $\frac{\partial V}{\partial t} = 0$ and, hence, $H^* = 0$. Also, the following additional conditions must be satisfied:

- The game is two players, zero-sum, pursuit-evasion type. Being a pursuit evasion game implies that final time is free (i.e., to be optimized), but this condition can be relaxed [10, p. 34].

- The Hamiltonian is separable on its controls [10, p. 35].

Thus, if V, the game value function, does not depend explicitly on time, and these conditions are satisfied, Isaacs approach becomes also a particularization of Bellman equation (as it was expected: even the basis of their equations, Isaacs' "Tenet of transition" [10] and Bellman's "Principle of optimality" [26], are very similar). This condition is also satisfied, according to [28, p. 36], when the optimal control problem that is being solved is time-invariant and the final time is free, i.e., needs to be optimized. This is extended to differential games [8, p 223]: a game is time invariant if time does not appear explicitly as a variable in dynamics equation, running and terminal costs and termination condition. In that case, partial derivative of value function with respect to time will be zero. This will be the case of all the games studied in this work.

However, Pontryagin is not used to solve closed loop games due to that it complicates the partial derivatives and there are an uncountable number of solutions due to information non-uniqueness (Section 2.3). These drawbacks would also affect Isaacs equations, hence, they are usually only employed to solve open loop games.

Yet, as it is described in [8, pp 345-350], the solutions to some pursuit-evasion games are usually first obtained in open-loop strategies and the synthesized to feedback strategies, provided that both exists. Hence, in pursuit-evasion games, open-loop and feedback solutions are related. Bellman approach provides a sufficiency condition for saddle-point strategies, but his main drawback is that the value function V is generally not known ahead of time. In order to overcome this, Pontryagin method is used in order to obtain a set of necessary conditions for an open loop representation of the feedback solution: if both open loop and feedback equilibria exist, Pontryagin will lead to the desired solution. Hence, in these games, it is usual obtaining an open loop representation of the solution, which then can be synthesized to obtain the feedback strategy. This is the main contribution of Isaacs method: obtaining open loop solution for games that fall into the category of pursuit-evasion, thus providing a simpler method than Bellman's equation.

Chapter 3

Problem description

3.1 Capacity approximation

In this chapter, we pose a capacity game and solve it using an approximation. Let us suppose that there are two UAVs and a high number of relays, which can be static or dynamic. The communicator tries to communicate with the relays, whereas the jammer tries to jam this communication. Thus, both players have opposite objectives and, hence, a zero-sum game between them is posed.

The total capacity in this scenario can be computed as the sum of the different capacities at each relay. Considering a free space propagation model, orthogonal modulation and using Shannon's Capacity formula, the total capacity per bandwidth unit of the system depends on the SINR as follows:

$$C_t = \sum_{i=1}^{N} \log_2(1 + \mathrm{SINR}_i) = \sum_{i=1}^{N} \log_2 \left(1 + \frac{\frac{P_c}{d_{c,ri}^2}}{N_0 + \frac{P_j}{d_{j,ri}^2}} \right) \tag{3.1}$$

In the expression before, P_c and P_j are the communicator and the jammer transmission fixed power, respectively; $d_{c,ri}$ and $d_{j,ri}$ are the euclidean distances between the communicator or the jammer and relay i, respectively, considering that there are N relays; and N_0 is the noise floor power. In order to optimize the expression above, it would be necessary to know the position of each relay in every time instant (and their dynamics if they were mobile).

If there is no knowledge about relays positions, a different approach is required. Let us suppose that relays and UAVs move in the $\mathbb{R}^3$ Cartesian space, thus, in every time instant, the position is defined by the vector (x, y, z). Let us assume that both UAVs move on the same plane (i.e., they have constant Z-coordinate), and that all mobile relays also move on the same plane. Let us define ϵ as the distance between the plane of relays and the UAVs plane. This situation is shown in Figure 3.1. Hence, the total capacity equation becomes:

$$C_t = \sum_{i=1}^{N} \log_2 \left(1 + \frac{\frac{P_c}{(x_c - x_{r,i})^2 + (y_c - y_{r,i})^2 + \epsilon^2}}{N_0 + \frac{P_j}{(x_j - x_{r,i})^2 + (y_j - y_{r,i})^2 + \epsilon^2}} \right) \tag{3.2}$$

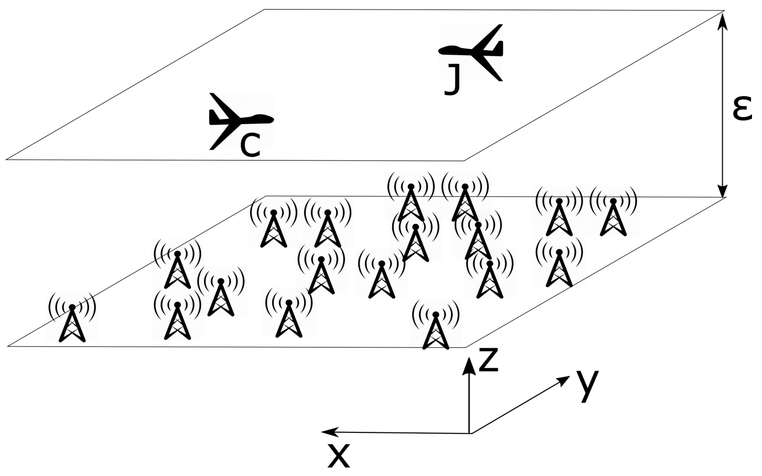


Fig. 3.1. Problem situation: there is a z-constant plane where UAVs move and a relay plane. The distance between planes is ϵ.

Assuming that $\frac{P_j}{d^2_{j,ri}} \gg N_0$, the SINR can be approached by the SIR. If the relays positions in the plane are considered to be a random vector $S = (S_x, S_y)$, with arbitrary probability density function $p_i(S_{x,i}, S_{y,i})$, the game payoff can be computed as the mathematical expectation of the SIR as follows:

$$\mathbb{E}\{C_t(S_x, S_y)\} = \int\int \sum_{i=1}^{N} \log_2 \left(1 + \frac{P_c}{P_j}\frac{d^2_{j,ri}(S)}{d^2_{c,ri}(S)}\right) p_i(S_{x,i}, S_{y,i}) dS_i \tag{3.3}$$

where $dS_i = dS_{y,i}dS_{x,i}$, and $d^2_{c,ri}(S) = (x_c - S_{x,i})^2 + (y_c - S_{y,i})^2 + \epsilon^2$ and $d^2_{j,ri}(S) = (x_j - S_{x,i})^2 + (y_j - S_{y,i})^2 + \epsilon^2$ are, respectively, the distance between the communicator or the jammer and relay i, whose plane-coordinates are $(S_{x,i}, S_{y,i})$. If the random variables S_i are considered to be independent and identically distributed (i.i.d.) and assuming that relays follow a uniform distribution in the interval $[-D, D]$ in coordinates X and Y, the expression in (3.3) becomes:

$$\mathbb{E}\{C_t(S_x, S_y)\} = \int_{-D}^{D}\int_{-D}^{D} N \log_2 \left(1 + \frac{P_c}{P_j}\frac{d^2_{j,r}(S)}{d^2_{c,r}(S)}\right) \frac{1}{4D^2} dS \tag{3.4}$$

where dS_i becomes $dS = dS_y dS_x$, and $d^2_{c,ri}(S)$ and $d^2_{j,ri}(S)$ become $d^2_{c,r}(S) = (x_c - S_x)^2 + (y_c - S_y)^2 + \epsilon^2$ and $d^2_{j,r}(S) = (x_j - S_x)^2 + (y_j - S_y)^2 + \epsilon^2$, respectively. The expression in (3.4) is hard to solve analytically, hence, we will use an approximation. Integrating (3.4) with respect to S_x and simplifying the results, considering that the jammer and evader are far from the region borders and that ϵ is small compared to the region size, which means that $D \gg |x_c|, D \gg |x_j|, D \gg \epsilon$, we can simplify and obtain:

$$\begin{aligned}\mathbb{E}\{C_t(S_x|S_y)\} &= \int_{-D}^{D} N\log_2\left(1+\frac{P_c}{P_j}\frac{d^2_{j,r}(S_x,S_y)}{d^2_{c,r}(S_x,S_y)}\right)\frac{1}{2D}dS_x \\ &\approx N\left(\log_2\left(1+\frac{P_c}{P_j}\right)-\frac{\sqrt{(y_c-s_y)^2}\pi}{D\log(2)}\right. \\ &\left.+\frac{\pi\sqrt{\left(\frac{P_j}{P_c}+1\right)\left((y_j-s_y)^2+\frac{P_j}{P_c}(y_c-s_y)^2\right)+\frac{P_j}{P_c}(x_c-x_j)^2}}{D\left(\frac{P_j}{P_c}+1\right)\log(2)}\right)\end{aligned} \tag{3.5}$$

If this approximation is integrated with respect to S_y and simplified using the same hypotheses than before but with respect to the Y coordinate ($D \gg |y_c|, D \gg |y_j|$), we can simplify the expression in (3.4) to:

$$\hat{\mathbb{E}}\{C_t(S_x,S_y)\} \approx N\left(\log_2\left(1+\frac{P_c}{P_j}\right)+\frac{\frac{P_j}{P_c}r\,\text{arcsinh}\left(\frac{D\left(1+\frac{P_j}{P_c}\right)}{\sqrt{\frac{P_j}{P_c}r}}\right)}{2D^2\left(1+\frac{P_j}{P_c}\right)^2\log(2)}\right) \tag{3.6}$$

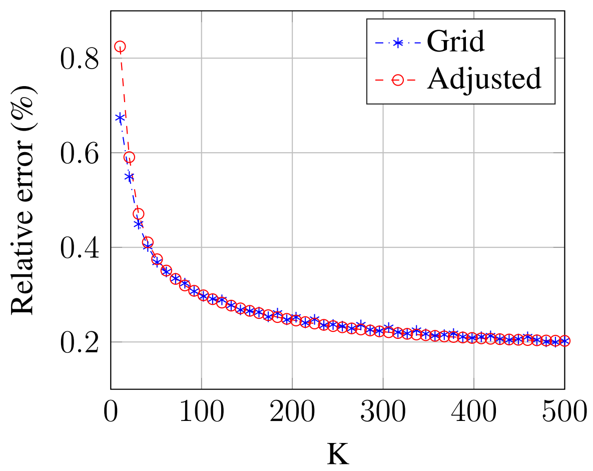
where $r = (y_c - y_j)^2 + (x_c - x_j)^2$. Hence, the capacity depends on r, the squared norm of the vector pointing from the communicator to the jammer: the bigger this norm is, the bigger capacity the system will have. Thus, the jammer wants to minimize capacity and that means trying to be spatially close to the communicator, whereas the communicator tries to maximize capacity and that means being spatially as far as possible from the jammer.

In order to estimate the performance of the capacity approximation in (3.6), a simulation has been run. Distributing $N = 100$ relays uniformly over a square of side $2D = 200$, the communicator has been placed on $(x_c, y_c) = (2, 1)$ and the jammer was placed along the main diagonal ($x_j = y_j$), with $\epsilon = 1$ and $P_c = P_j = 1$. We computed and averaged the empirical capacity for 100 realizations.The main conclusion we get is that once we can neglect border effects, if we are working in sufficiently big regions, this approximation is accurate.

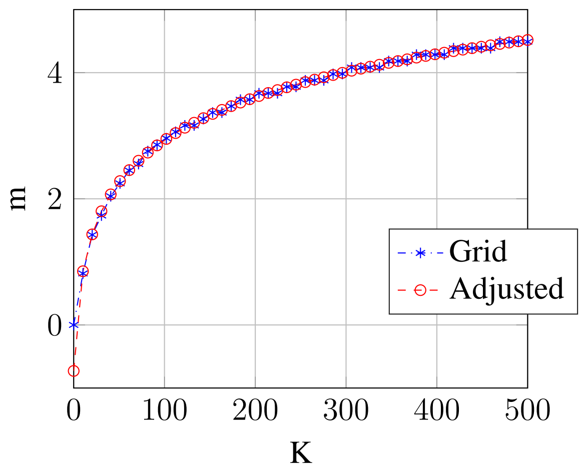
3.2 Hyperbolic arcsine linearization

The expression (3.6) can be further simplified linearizing the hyperbolic arcsine term. In order to do so, let us consider the following expression:

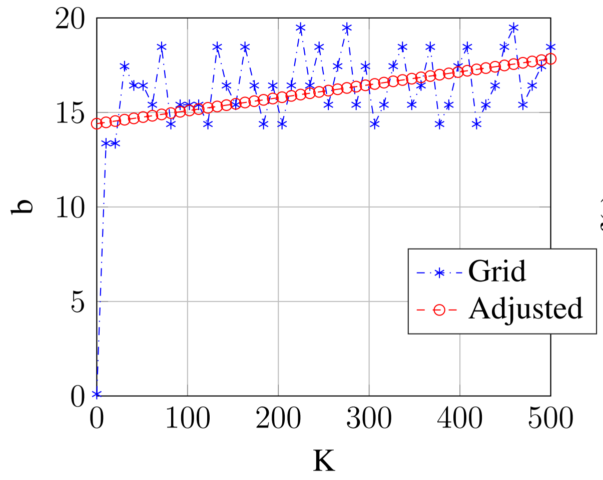
$$g_1(r) = r\,\text{arcsinh}\left(\frac{K}{\sqrt{r}}\right) \tag{3.7}$$



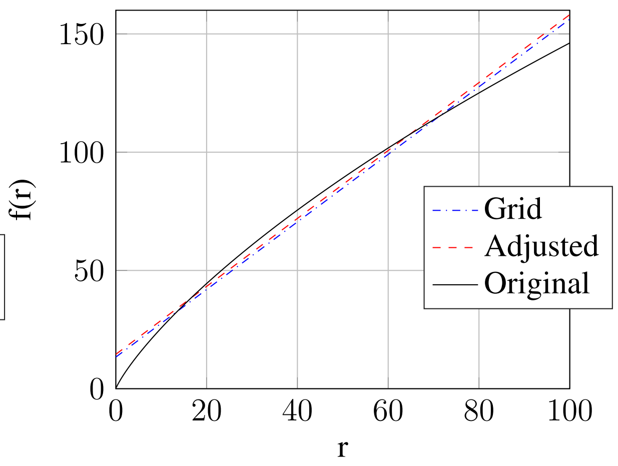
(a) Relative error.



(b) Comparison of m.



(c) Comparison of b.



(c) Approximation examples, K=20

Fig. 3.2. Results of hyperbolic arcsine linearization. The relative error is always below 0.01, using the expression in (3.12). The adjustement of m and b in (3.11) are also plot: m adjusts very good, b adjusts worse. An example of the original function g_1 (3.7) and the adjusted using (3.9) when $K = 20$. Grid stands for the optimal parameters (slope and intercept of the line) obtained via grid search, while adjusted stands for the parameters (slope and intercept) obtained using expressions in (3.11)

where K is a constant, that, in (3.6), corresponds to the following:

$$K = \frac{D\left(1 + \frac{P_j}{P_c}\right)}{\sqrt{\frac{P_j}{P_c}}} \tag{3.8}$$

We want to fit this function using a linear expression, that is:

$$g_2(r) = mr + b \tag{3.9}$$

where m is the slope of the line and b is the intercept. In order to approximate this function, we must obtain the optimal parameters m and b that satisfy the following optimization problem:

$$\min_{m,b} \int_0^D (g_2(r) - g_1(r))^2 \, dr = \min_{m,b} \int_0^D \left(mr + b - r \operatorname{arcsinh}\left(\frac{K}{\sqrt{r}}\right)\right)^2 dr \tag{3.10}$$

That is, we want to minimize the squared error between the original function and the fit, considering that the distance between players r is between 0 and D.

Since the expression in (3.10) is unsolvable using classical methods, a numerical approach will be used instead. We use a three-dimensional grid over the constant K and the parameters b and m in order to obtain the optimal duples (m, b) for each value of K. The grid will have 50 points in each dimension. The parameters ranges are $K \in [0, 500]$, $m \in [0, 5]$ and $b \in [-5, 45]$.

For each grid point, the last integral in (3.10) is numerically solved and the squared error value is stored. After computing all errors, for each value of K, the pair of (m, b) which minimizes the error is searched and considered to be the optimum one. Thus, we obtain a table, where for each value of K correspond a value of m and b.

This correspondence can also be adjusted. Adjusting in the least squared sense, the following expressions are obtained, which are represented in Figure 3.2:

$$\begin{aligned} m(K) &= \log(0.1824K + 0.4823) \\ b(K) &= 0.0069K + 14.4070 \end{aligned} \tag{3.11}$$

Finally, the relative error between the hyperbolic arcsine expressions and the exact function is computed, using the following expression:

$$\zeta = \frac{\sqrt{\int_0^D \left(m(K)r + b(K) - r \operatorname{arcsinh}\left(\frac{K}{\sqrt{r}}\right)\right)^2 dr}}{\int_0^D r \operatorname{arcsinh}\left(\frac{K}{\sqrt{r}}\right) dr} \tag{3.12}$$

where $m(K)$ and $b(K)$ are the ones from (3.11).

The relative error obtained, that can be observed in Figure 3.2, is always inferior to 1% and is monotone decreasing with K, that is, higher values of K yield lower relative errors to the linear approximation used. Hence, applying the expressions in (3.11), (3.9) and (3.8) to simplify (3.6) yields the following simplified expression for the capacity:

$$
\begin{aligned}
&\hat{\mathbb{E}}\{C_t(S_x,S_y)\} \\
&\approx N\left(\log_2\left(1+\frac{P_c}{P_j}\right)+\frac{\frac{P_j}{P_c}}{2D^2\left(1+\frac{P_j}{P_c}\right)^2\log(2)}\left(rm(K)+b\right)\right) \\
&= N\log_2\left(1+\frac{P_c}{P_j}\right)+\frac{N\frac{P_j}{P_c}}{2D^2\left(1+\frac{P_j}{P_c}\right)^2\log(2)}\Big(r\left(\log(0.1824K+0.4823)\right) \\
&\qquad +0.0069K+14.4070\Big) \\
&= N\log_2\left(1+\frac{P_c}{P_j}\right)+N\frac{\frac{P_j}{P_c}\left(0.0069K+14.4070\right)}{2D^2\left(1+\frac{P_j}{P_c}\right)^2\log(2)} \\
&\qquad +\frac{Nr\frac{P_j}{P_c}\left(\log(0.1824K+0.4823)\right)}{2D^2\left(1+\frac{P_j}{P_c}\right)^2\log(2)} \\
&= N\log_2\left(1+\frac{P_c}{P_j}\right)+N\frac{\frac{P_j}{P_c}\left(0.0069\frac{D\left(1+\frac{P_j}{P_c}\right)}{\sqrt{\frac{P_j}{P_c}}}+14.4070\right)}{2D^2\left(1+\frac{P_j}{P_c}\right)^2\log(2)} \\
&\qquad +\frac{Nr\frac{P_j}{P_c}\left(\log\left(0.1824\frac{D\left(1+\frac{P_j}{P_c}\right)}{\sqrt{\frac{P_j}{P_c}}}+0.4823\right)\right)}{2D^2\left(1+\frac{P_j}{P_c}\right)^2\log(2)}
\end{aligned}
\tag{3.13}
$$

Hence, the capacity expectation is approximated as a linear function of the distance with the form:

$$\hat{\mathbb{E}}\{C_t(S_x,S_y)\}\approx Ar+B \tag{3.14}$$

whose slope and intercept are:

$$
\begin{aligned}
A &= N\left(\log_2\left(1+\frac{P_c}{P_j}\right)+\frac{\frac{P_j}{P_c}\left(0.0069\frac{D\left(1+\frac{P_j}{P_c}\right)}{\sqrt{\frac{P_j}{P_c}}}+14.4070\right)}{2D^2\left(1+\frac{P_j}{P_c}\right)^2\log(2)}\right) \\
B &= N\frac{\frac{P_j}{P_c}\left(\log\left(0.1824\frac{D\left(1+\frac{P_j}{P_c}\right)}{\sqrt{\frac{P_j}{P_c}}}+0.4823\right)\right)}{2D^2\left(1+\frac{P_j}{P_c}\right)^2\log(2)}
\end{aligned}
\tag{3.15}
$$

3.3 Conclusions

In this chapter, we pose and solve an ergodic capacity problem in terms of its mathematical expectation. Approximating the solution when communicator and jammer are far from the relay region borders, the capacity turns out to be a function of the distance between both UAVs (3.6), that can even be more simplified to a linear expression, (3.13). In order to solve the game and obtain the trajectories, we will use two different procedures in the following sections:

- The function obtained for the capacity (3.13) can be used as the game running cost: the players will try to minimize or maximize it. This will yield a coupled solution: controls will depend on player's positions.
- The solution to the capacity game involves that the jammer tries to be close to the communicator and the communicator tries to be far away from the jammer. This is also the idea in pursuit-evasion games, yet in this games, the payoff is not in terms of capacity, but in terms of capture time (Chapter 2), and hence, the running cost is $L = 1$ in these games. In this case, we are using a surrogate function approach.

The first approach gives the actual solution for the capacity game, whereas the second is just an approximation. However, the second one gives an open loop solution with closed expressions for the trajectory, which are easier to study, whereas the first gives an open loop, coupled solution which is harder to solve and study. Both approaches are analyzed and compared in the following sections.

Chapter 4

Pursuit-Evasion game of two UAVs

4.1 Introduction

In this chapter, the two-person, zero-sum, pursuit-evasion game that appears when approximating the problem described in Chapter 3 will be solved using Isaacs method, described in [10, ch. 4] as a pursuit-evasion game, with running cost $L = 1$. We consider each UAV to have a constant acceleration that will be F_p for the pursuer and F_e for the evader. A friction limit will be used, for the speed not to grow unbounded denoted by k_p and k_e for the pursuer and evader, respectively. Therefore, the maximum speed will be F/k. This setup is an extension to Isaacs "isotropic rocket" game [10, pp. 105-116], but considering that pursuer and evader have the same dynamics: constant acceleration and bounded speed.

4.2 Dynamics of the UAVs

Each player control variable will be their heading angle with respect to y-axis, which will be noted ϕ for the pursuer and ψ for the evader. Considering that there are eight states, which will be the position (x and y coordinate) and the velocities (u and v, which are the velocity components) of the pursuer and evader, the dynamics are:

$$\begin{pmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{u}_p \\ \dot{v}_p \\ \dot{x}_e \\ \dot{y}_e \\ \dot{u}_e \\ \dot{v}_e \end{pmatrix} = \begin{pmatrix} u_p \\ v_p \\ F_p \sin(\phi) - k_p u_p \\ F_p \cos(\phi) - k_p v_p \\ u_e \\ v_e \\ F_e \sin(\psi) - k_e u_e \\ F_e \cos(\psi) - k_e v_e \end{pmatrix} \tag{4.1}$$

4.3 Control optimization

Since this is a pursuit-evasion game, the final time is the reward and thus, running cost is $L = 1$ and final cost is $G = 0$. The Hamiltonian is built according to Isaacs "Main Equation" [10, p. 67], as it was shown in (2.7):

$$\max_{\psi} \min_{\phi} \sum_{i} V_{x_i} f_i + L = 0 \tag{4.2}$$

Substituting, it yields:

$$\begin{aligned}&\max_{\psi} \min_{\phi} V_{x_p} u_p + V_{y_p} v_p + V_{u_p}(F_p \sin(\phi) - k_p u_p) + V_{v_p}(F_p \cos(\phi) - k_p v_p) \\ &+ V_{x_e} u_e + V_{y_e} v_e + V_{u_e}(F_e \sin(\psi) - k_e u_e) + V_{v_e}(F_e \cos(\psi) - k_e v_e) + 1 = 0\end{aligned} \tag{4.3}$$

Taking into account that controls are separable we have:

$$\begin{aligned}1 &+ V_{x_p} u_p + V_{y_p} v_p + \min_{\phi} \Big(V_{u_p}(F_p \sin(\phi) - k_p u_p) + V_{v_p}(F_p \cos(\phi) - k_p v_p) \Big) \\ &+ V_{x_e} u_e + V_{y_e} v_e + \max_{\psi} \Big(V_{u_e}(F_e \sin(\psi) - k_e u_e) + V_{v_e}(F_e \cos(\psi) - k_e v_e) \Big) \\ &= 0\end{aligned} \tag{4.4}$$

The optimization problems in (4.4) can be solved using a Lemma provided by Isaacs [10, p. 43]:

Lemma 1. *If the following optimization problem is to be solved:*

$$\max_{\theta}[\min_{\theta}] \left(a \cos(\theta) + b \sin(\theta) \right)$$

and we define $\rho = \sqrt{a^2 + b^2} > 0$*, then the solutions to the maximization problem (brackets show results for the minimization problem) are:*

$$\begin{aligned}\cos(\theta) &= +[-]a/\rho \\ \sin(\theta) &= +[-]b/\rho \\ \max_{\theta}[\min_{\theta}] \left(a \cos(\theta) + bsen(\theta) \right) &= +[-]\rho\end{aligned}$$

Hence, from (4.4), considering that $\rho_p = \sqrt{V_{u_p}^2 + V_{v_p}^2}$ and $\rho_e = \sqrt{V_{u_e}^2 + V_{v_e}^2}$, using Lemma 1, the solutions to the minimization and maximization problems are:

$$\begin{aligned}\cos(\phi^*) &= -V_{v_p}/\rho_p \\ \sin(\phi^*) &= -V_{u_p}/\rho_p \\ \min_{\phi} \Big(V_{u_p} F_p \sin(\phi) + V_{v_p} F_p \cos(\phi) \Big) &= -\rho_p F_p \\ \cos(\psi^*) &= V_{v_e}/\rho_e \\ \sin(\psi^*) &= V_{u_e}/\rho_e \\ \max_{\psi} \Big(V_{u_e} F_e \sin(\psi) + V_{v_e} F_e \cos(\psi) \Big) &= \rho_e F_e\end{aligned} \tag{4.5}$$

Using this result, the Hamiltonian (4.4) becomes:

$$\begin{aligned}1+&V_{x_p}u_p + V_{y_p}v_p - \rho_p F_p - k_p(V_{v_p}v_p + V_{u_p}u_p)\\ &+V_{x_e}u_e + V_{y_e}v_e + \rho_e F_e - k_e(V_{v_e}v_e + V_{u_e}u_e) = 0\end{aligned} \tag{4.6}$$

4.4 Retrogressive Path Equations

The next step in Isaac's method is to obtain the Retrogressive Path Equations (RPE).There will be sixteen RPEs: one per state and another one per each component of the gradient of the value function, while in Isaacs original setup there were 12 [10, p. 107] . The equations that depend on the dynamics equation are obtained from (4.1) using the variable change from (2.8).

The eight equations that depend on the dynamics equation are the following:

$$\begin{pmatrix} \mathring{x}_p \\ \mathring{y}_p \\ \mathring{u}_p \\ \mathring{v}_p \\ \mathring{x}_e \\ \mathring{y}_e \\ \mathring{u}_e \\ \mathring{v}_e \end{pmatrix} = \begin{pmatrix} -u_p \\ -v_p \\ F_p\sin(\phi) + k_p u_p \\ F_p\cos(\phi) + k_p v_p \\ -u_e \\ -v_e \\ -F_e\sin(\psi) + k_e u_e \\ -F_e\cos(\psi) + k_e v_e \end{pmatrix} \tag{4.7}$$

were $\mathring{x}$ denotes derivative of x with respect to τ.The other eight RPEs are obtained from the gradient of the value function, using (2.12), and hence, these RPEs are obtained through derivation of the obtained Main Equation (4.6) with respect to each state variable. The resulting RPEs are:

$$\begin{pmatrix} \mathring{V}_{x_p} \\ \mathring{V}_{y_p} \\ \mathring{V}_{u_p} \\ \mathring{V}_{v_p} \\ \mathring{V}_{x_e} \\ \mathring{V}_{y_e} \\ \mathring{V}_{u_e} \\ \mathring{V}_{v_e} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ V_{x_p} - k_p V_{u_p} \\ V_{y_p} - k_p V_{v_p} \\ 0 \\ 0 \\ V_{x_e} - k_e V_{u_e} \\ V_{y_e} - k_e V_{v_e} \end{pmatrix} \tag{4.8}$$

4.5 Final conditions

In order to determine the final conditions, we must define the terminal surface (i.e., the surface where the pursuer captures the evader), which we will call h. By considering that the capture distance is l, the surface capture will be the ball whose center is the evader position: when the pursuer enters that ball, the game ends and capture occurs. Hence, the termination surface will be the sphere in which the distance between pursuer and evader equals l, the capture distance. It can be parametrized using $n-1$ variables (where n is the number of states) as follows, where we recall that s_i are the final conditions variables:

$$h = \begin{pmatrix} x_p \\ y_p \\ u_p \\ v_p \\ x_e \\ y_e \\ u_e \\ v_e \end{pmatrix} = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_1 + l\sin(s_5) \\ s_2 + l\cos(s_5) \\ s_6 \\ s_7 \end{pmatrix} \tag{4.9}$$

Using (4.9), it is possible to obtain the final conditions with (2.13), taking into account that in this game, the final cost G is zero (Chapter 2). These final conditions are:

$$\begin{pmatrix} V_{x_p} + V_{x_e} \\ V_{y_p} + V_{y_e} \\ V_{u_p} \\ V_{v_p} \\ V_{x_e} l\cos(s_5) - V_{y_e} l\sin(s_5) \\ V_{u_e} \\ V_{v_e} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{4.10}$$

We remark that these are final time conditions ($t = t_f$), but in retro time, they are initial conditions ($\tau = 0$), as it is implicit in the variable change done. Hence, the s_i are initial conditions in retro time, but final conditions in normal time.

In order to solve these equations, an auxiliary variable, named λ, will be used. From (4.10), the two first equations and the fifth show that, in the terminal sphere:

$$\begin{aligned} -V_{x_p} = V_{x_e} = \lambda\sin(s_5) \\ -V_{y_p} = V_{y_e} = \lambda\cos(s_5) \end{aligned} \tag{4.11}$$

That is, the four equations are related using an unknown value that we will identify with λ. Also, from the rest of equations in (4.10) and that $\rho_p = \sqrt{V_{u_p}^2 + V_{v_p}^2}$ and $\rho_e = \sqrt{V_{u_e}^2 + V_{v_e}^2}$, in the terminal manifold, $\rho_e = \rho_p = 0$. Substituting all that in (4.6) yields:

$$-\lambda s_3 \sin(s_5) - \lambda s_4 \cos(s_5) + \lambda s_6 \sin(s_5) + \lambda s_7 \cos(s_5) + 1 = 0 \tag{4.12}$$

Manipulating (4.12) gives the following result for λ:

$$\lambda = \frac{1}{(s_3 - s_6)\sin(s_5) + (s_4 - s_7)\cos(s_5)} \tag{4.13}$$

4.6 RPEs integration

Let us start integrating the equations in (4.8). The four equations for $V_{x_e}, V_{y_e}, V_{x_p}, V_{y_p}$ were already solved in (4.11), where it was shown that:

$$\begin{aligned} -V_{x_p} &= V_{x_e} = \lambda \sin(s_5) \\ -V_{y_p} &= V_{y_e} = \lambda \cos(s_5) \end{aligned} \tag{4.14}$$

where λ is defined in (4.13). The other four RPEs in (4.8) are solved by replacing the values of $V_{x_e}, V_{y_e}, V_{x_p}, V_{y_p}$ that are in (4.14) and using the initial conditions (in retro time) from (4.10). The integrated equations are:

$$\begin{aligned} Vu_p &= -\lambda \sin(s_5) \frac{1 - e^{-k_p \tau}}{k_p} \\ Vv_p &= -\lambda \cos(s_5) \frac{1 - e^{-k_p \tau}}{k_p} \\ Vu_e &= \lambda \sin(s_5) \frac{1 - e^{-k_e \tau}}{k_e} \\ Vv_e &= \lambda \cos(s_5) \frac{1 - e^{-k_e \tau}}{k_e} \end{aligned} \tag{4.15}$$

The optimal controls can be obtained now: since $\rho_p = \sqrt{V_{u_p}^2 + V_{v_p}^2}$ and $\rho_e = \sqrt{V_{u_e}^2 + V_{v_e}^2}$, substituting the values above yield:

$$\begin{aligned} \cos(\phi^*) &= \cos(s_5) \\ \sin(\phi^*) &= \sin(s_5) \\ \cos(\psi^*) &= \cos(s_5) \\ \sin(\psi^*) &= \sin(s_5) \end{aligned} \tag{4.16}$$

Hence, both optimal controls are constant and equal to both players.The same solution is obtained in the original setup [10, p. 109], though the dynamics are different in this setup.

Proceeding similarly, it is possible to integrate the left eight RPEs from (4.7). Using the values in (4.15) and the initial conditions (in retro time) from (4.9) yield:

$$
\begin{aligned}
u_p &= s_3 e^{k_p\tau} + F_p \sin(s_5)\frac{1-e^{k_p\tau}}{k_p}\\
v_p &= s_4 e^{k_p\tau} + F_p \cos(s_5)\frac{1-e^{k_p\tau}}{k_p}\\
x_p &= s_1 + s_3\frac{1-e^{k_p\tau}}{k_p} + F_p \sin(s_5)\frac{e^{k_p\tau}-1-k_p\tau}{k_p^2}\\
y_p &= s_2 + s_4\frac{1-e^{k_p\tau}}{k_p} + F_p \cos(s_5)\frac{e^{k_p\tau}-1-k_p\tau}{k_p^2}\\
u_e &= s_6 e^{k_e\tau} + F_e \sin(s_5)\frac{1-e^{k_e\tau}}{k_e}\\
v_e &= s_7 e^{k_e\tau} + F_e \cos(s_5)\frac{1-e^{k_e\tau}}{k_e}\\
x_e &= s_1 + l\sin(s_5) + s_6\frac{1-e^{k_e\tau}}{k_e} + F_e \sin(s_5)\frac{e^{k_e\tau}-1-k_e\tau}{k_e^2}\\
y_e &= s_2 + l\cos(s_5) + s_7\frac{1-e^{k_e\tau}}{k_e} + F_e \cos(s_5)\frac{e^{k_e\tau}-1-k_e\tau}{k_e^2}
\end{aligned}
\tag{4.17}
$$

4.7 Analytical solution to the system

The equations in (4.17) give the optimal trajectories for both players, depending on the parameters used to describe the terminal sphere and the retro time τ, which are unknown. Since initial conditions are known (i.e, initial positions and speeds of both players), it is possible to obtain these parameters by equaling the equations in (4.17) to the initial conditions and particularized to $t = 0$, that is, $\tau = t_f - t = t_f$.

This system, is nonlinear and trigonometric and may be hard to solve. To simplify its resolution, we apply the same procedure that Isaacs used [10, pp. 110-111]: the final time t_f is obtained from the initial conditions and game parameters by squaring and adding these two identities and by using that $\cos^2(\alpha) + \sin^2(\alpha) = 1$:

$$
\begin{aligned}
x_p - x_e - u_p\left(\frac{e^{-k_p\tau}-1}{k_p}\right) + u_e\left(\frac{e^{-k_e\tau}-1}{k_e}\right) &= \sin(s_5)Q(\tau)\\
y_p - y_e - v_p\left(\frac{e^{-k_p\tau}-1}{k_p}\right) + v_e\left(\frac{e^{-k_e\tau}-1}{k_e}\right) &= \cos(s_5)Q(\tau)
\end{aligned}
\tag{4.18}
$$

where

$$
Q(\tau) = \frac{F_e(e^{-k_e\tau}-1+k_e\tau)}{k_e^2} - l - \frac{F_p(e^{-k_p\tau}-1+k_p\tau)}{k_p^2}
\tag{4.19}
$$

The resulting expression, which is in (4.20), only depends on known initial conditions and game parameters and hence, it is a nonlinear function of τ. By solving for τ, that is, $g(\tau) = 0$, the τ obtained will be the final time of the game, that is, $\tau = t_f$.

Algorithm 1 Steps for the analytical approach

```
1: Obtain initial conditions and game parameters
2: Obtain final time using (4.20)
3: Solve the equation system in (4.17) using (2.14) to obtain final time conditions from
   initial ones
4: Compute optimal trajectories using final conditions obtained with (4.17)
```

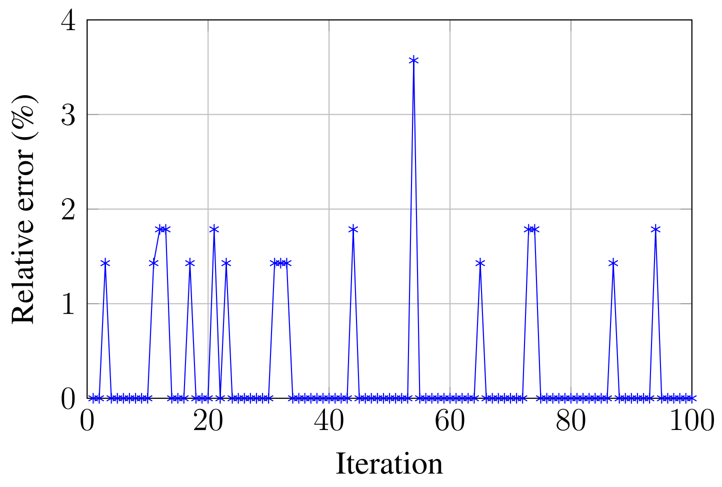


Fig. 4.1. Relative error between final time obtained using the analytical solution to the equations in (4.17) and (4.20) and the approach proposed to chose the optimal point for the cost function, based on using Runge-Kutta numerical method to solve differential equations and searching for the point where capture occurs that has the closest final heading angle to the one introduced as parameter. The error mean is 0.0029 and the standard deviation is of 0.0067, whereas the median is 0 and the maximum relative error obtained is 0.0357. In general, the method proposed works fine, there are only a few points which give a higher error, due to the fact that we are considering the optimum point to be the one where capture occurs and with smaller final heading angle difference.

$$\begin{aligned} g(\tau) = &\left(x_p - x_e - u_p\left(\frac{e^{-k_p\tau}-1}{k_p}\right) + u_e\left(\frac{e^{-k_e\tau}-1}{k_e}\right)\right)^2 + \\ &\left(y_p - y_e - v_p\left(\frac{e^{-k_p\tau}-1}{k_p}\right) + v_e\left(\frac{e^{-k_e\tau}-1}{k_e}\right)\right)^2 - Q(\tau)^2 \end{aligned} \tag{4.20}$$

Once that t_f has been obtained, it can be replaced in the system in (4.17). If this system is particularized for the initial time conditions, doing the following variable change: $w_1 = \cos(s_5), w_2 = \sin(s_5)$, yields a linear system which can be solved using standard techniques (recall that $w_1^2 + w_2^2 = 1$). An illustration of these steps is shown in Algorithm 1.

4.8 Optimization solution to the system

The technique proposed in the section before to solve the equations system (4.17) has a big drawback: due to the exponentials involved in the system, the solution is not always found by the computer. A different approach can be done in order to obtain the final conditions from the initials, based on searching an optimum of a cost function.

In order to perform this search, two different paths can be followed. The first one consists in performing a search over an eight-dimensional surface, where each of the dimensions correspond to a final condition (seven for the terminal sphere (4.9) and one for the final time t_f). The search must find the points where, in final time t_f, capture occurs and also, are congruent. With congruent, we mean that with the final conditions that correspond to the eight-dimensional surface, we would obtain the trajectories and check whether, in the final point of the trajectories, the final conditions - i.e., final speeds, positions and heading angle - correspond to the components of the point of the eight-dimensional surface. In other words, we would perform a search over this eight-dimensional surface, where a cost function would be used to test the deviation of the final conditions used as input to compute the trajectories and the actual final conditions obtained from the trajectories. This approach has the inconvenience that the search is not easy to do in this surface, due to high-dimensionality and that this surface is not smooth and non-convex.

The second approach consists in performing a search over a two dimensional surface. Since we know initial conditions of the game, the trajectories can be computed numerically using the expressions in (4.1). To do so, a Runge-Kutta method is used to solve the differential equations that control the dynamics of the UAVs and that allows us to obtain velocities and positions of each player in any time. Only two parameters are needed to obtain these trajectories: the final time t_f is required to integrate the differential equations and the final heading angle s_5 is required, since it is the control of both players.

Hence, in this case, a search over only two dimensions (t_f and s_5) is to be performed, in order to obtain the trajectories using a numerical ODE solver. After obtaining the trajectories, congruency is checked: if in final time, capture occurs and heading angle corresponds to s_5. If both conditions happen, then the point is a candidate to be a solution to the game - they give the final payoff, which is t_f, and the control for the players, which is s_5.

We implement this approach in order to obtain the game solution. The numerical ODE solver chosen is a Runge-Kutta one, based on Dormand-Prince (4,5) pair [29]. The duple (s_5, t_f) that is considered the solution is chosen as the duple where capture happens - that is, final distance between players is equal or smaller than capture distance l - and which has the smaller absolute error between the final heading angle obtained in the trajectories and the introduced a priori in the duple. The final heading angle can be obtained from (4.9) as:

$$\hat{s_5} = \arctan\left(\frac{x_{e,f} - x_{p,f}}{y_{e,f} - y_{p,f}}\right) \tag{4.21}$$

where $x_{e,f}$, $x_{p,f}$, $y_{e,f}$ and $y_{p,f}$ are the final points in the trajectories numerically obtained.

In order to validate these conditions to obtain the solution point, we run a simulation

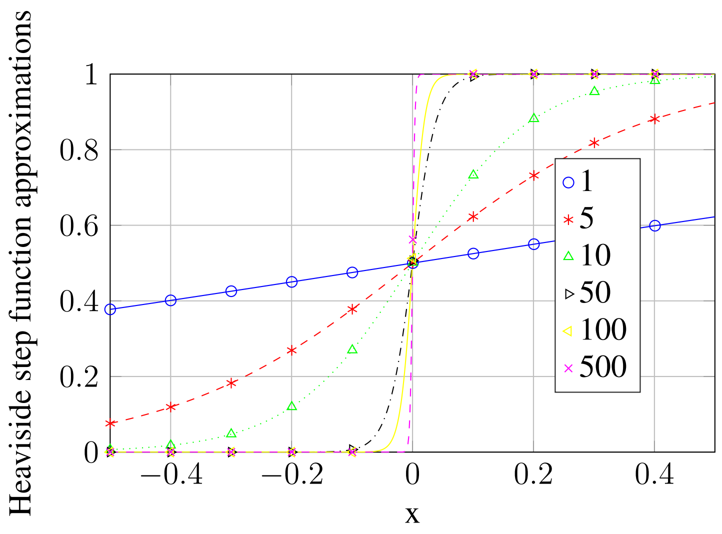


Fig. 4.2. Representation of the smooth approximation for Heaviside step function, corresponding to the expression $f(x) = \frac{1}{1+e^{-k_2 x}}$, for different values of k_2. It is possible to check how the transition in $x = 0$ becomes sharper as k_2 grows

where the game is solved using the approach described above. We choose random initial conditions, following a uniform distribution, with the following limits: $x_{e,0}, y_{e,0} \in [0, 10]$, $x_{p,0}, y_{p,0} \in [-10, 0]$, $u_{e,0}, v_{e,0} \in [0, 1]$, $u_{p,0}, v_{p,0} \in [-1, 0]$, $v_{max,p} \in [0, 2]$, $v_{max,e}, F_e, F_p \in [0, 1]$. The capture distance is $l = 1$. We define a random set of initial conditions and the game solution is obtained using the analytical method described above, where a linear system is solved. If this method fails to give a solution, another point is tried, until a valid point is obtained and we have the exact solution for the game - i.e., s_5 and t_f. Then, a two dimensional grid centered in the valid duple of (s_5, t_f) is computed. Each side of the grid is chosen to have 29 points, for computation speed reasons.

After, each point (s_5, t_f) of the grid is taken as an input duple for the solver that uses the Runge-Kutta method described above. This solver returns final distance between players and final heading angle (computed using (4.21)) obtained for that concrete input duple, and then, the solution point is taken as the point with minimum difference between the final heading angle used as input for the solver and the angle obtained using (4.21).

This procedure is computed 100 times and, since the game payoff is the capture time (recall that running cost $L = 1$), the relative error between the final time obtained using the Runge-Kutta solution and the exact solution obtained analytically is computed. A graph with this error as a function of the iteration is in Figure 4.1, and it validates our approach.

Finally, we need to put these conditions in a cost function: the solution point - the one with a lowest cost - must be the one that, among the points where capture happens, has smaller absolute error between the final heading angle introduced a priori and the one obtained using (4.21). This can be introduced in a cost function of the next form:

$$f_{c,1} = \frac{k_1}{1 + e^{-k_2(d_f - l)}} + k_3|s_5 - \hat{s_5}| \tag{4.22}$$

Algorithm 2 Steps for the optimization approach

```
1: Obtain initial conditions and game parameters
2: while Cost in (4.22) is greater than threshold do
3:     Guess a pair $(s_5, t_f)$
4:     Solve ODE system numerically from (4.1), using the $(s_5, t_f)$ pair guessed
5:     Obtain capture time and $\hat{s_5}$ from trajectories using (4.21)
6:     Compute cost for the pair $(s_5, t_f)$ using (4.22)
7: end while
8: The pair $(s_5, t_f)$ is correct: optimal trajectories are obtained by solving ODE system
   numerically from (4.1), using that $(s_5, t_f)$ pair
   {SOO is used in steps 2-7}
```

where k_1, k_2 and k_3 are constants, d_f is the final distance between players, computed using the trajectories values, l is capture distance, s_5 is the final heading angle supposed a priori and $\hat{s_5}$ is the final heading angle, computed with the trajectories using (4.21).

The first term is an analytic and smooth approximation for the Heaviside step function, when $k_1 = 1$. The parameter k_2 controls how sharp the transition will be in $d_f = l$: larger values of k_2 give a sharper transition, closer to the ideal but non-smooth step function, as it is possible to see in Figure 4.2.

For adequate values of the constants k_1, k_2 and k_3, it is possible to achieve the cost function that we need. If $d_f > l$, the exponential argument is negative and hence, small, so the first term is approximately k_1. If $k_1 > k_3|s_5 - \hat{s_5}|$, then, the value tends to be k_1. This is the case where capture does not occur.

If capture occurs, $d_f < l$ and hence, the exponential argument is positive. For sufficiently high values of k_2, the first term of the cost function vanishes and hence, the cost function tends to be $k_3|s_5 - \hat{s_5}|$. This means that, when capture occurs, the cost is proportional to the absolute error between heading angles, as we intended.

Hence, the cost function defined in (4.22) will be used for the two dimensional search proposed. We consider that the constants are $k_1 = 1$, $k_2 = 500$ and $k_3 = 1$. The non-convex algorithm Simultaneous Optimistic Optimization (SOO) [30] [31] is used in order to obtain the game solution - i.e., final heading angle, which is the control, and time of capture, which is the payoff of the game. An illustration of these steps is found in Algorithm 2.

4.9 Hybrid solution to the system

An intermediate approach between the analytical and the optimization methods proposed in the previous sections can also be defined. It consists in simplifying the two-dimensional optimization method by computing the right t_f using (4.19). Hence, in this case, we first obtain the final time analytically, by numerically solving (4.19) and afterwards, we perform a minimization of the cost function defined in (4.22) over the final heading angle s_5.

This approach needs less iterations of the optimization algorithm, and hence, it is faster, at the cost of having to solve numerically the expression shown in (4.19) in order to obtain the optimum final time. An illustration of these steps is found in Algorithm 3.

Algorithm 3 Steps for the hybrid approach

```
1: Obtain initial conditions and game parameters
2: Obtain final time using (4.20)
3: while Cost in (4.22) is greater than threshold do
4:     Guess a value for $s_5$
5:     Solve ODE system numerically from (4.1), using the $t_f$ computed and $s_5$ guessed
6:     Obtain capture time and $\hat{s_5}$ from trajectories using (4.21)
7:     Compute cost for the pair $(s_5, t_f)$ using (4.22)
8: end while
9: The pair $(s_5, t_f)$ is correct: optimal trajectories are obtained by solving ODE system
   numerically from (4.1), using that $(s_5, t_f)$ pair
   {SOO is used in steps 3-8}
```

4.10 Simulations

4.10.1 Simulation 1: Trajectories and capacity evolution

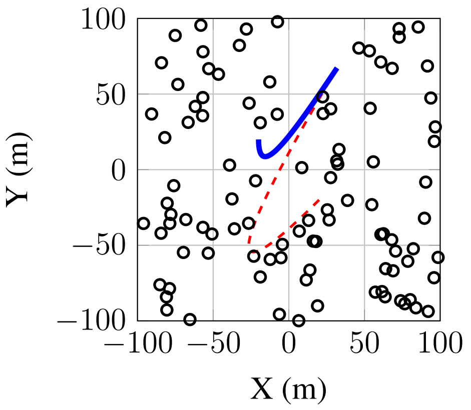
The first simulation uses the analytical solution approach in order to obtain trajectories and to show capacity evolution with time. The UAV parameters are $F_p = 0.05$, $k_p = 0.0125$ and, thus, $v_{max,p} = 4$; $F_e = 0.03$, $k_p = 0.03$ and hence, $v_{max,e} = 1$, $l = 0.1$. The relay region is a square, centered at the origin, whose side length is $2D = 200$. The relays are distributed randomly at each new simulation, following a uniform distribution in both X and Y coordinates. Initial conditions of both UAVs are randomly selected in each new simulation. A number of $N = 100$ static relays are considered, and $\epsilon = 1$.

Considering that the jammer and the communicator are in the same spatial position, from (3.4), we can see that the SIR will be the same in all relay nodes, and equal to $\frac{P_c}{P_j}$. Provided that there is a minimum SINR threshold for communication to succeed and if noise is much smaller than interference, then SINR can be approximated by SIR and if the jammer knows the communicator power, it can adjust its own in order to cause all nodes to fall below the SINR threshold and, hence, cause that total system capacity becomes zero. Considering that the communicator is at distance $\sqrt{\epsilon^2 + d_r^2}$ from a relay, and the jammer is at distance l from the communicator (therefore, in the frontier of the capture region), the SIR function is:

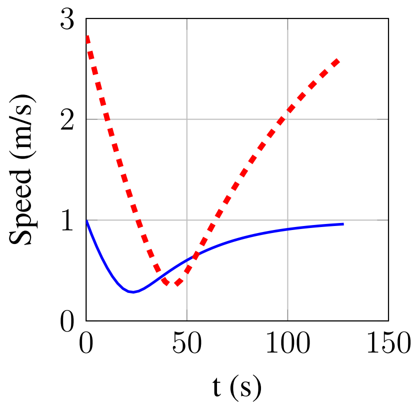
$$SIR = \frac{P_c}{P_j} \frac{(d_r + l)^2 + \epsilon^2}{\epsilon^2 + d_r^2} \tag{4.23}$$

It can be demonstrated that (4.23) has a maximum at:

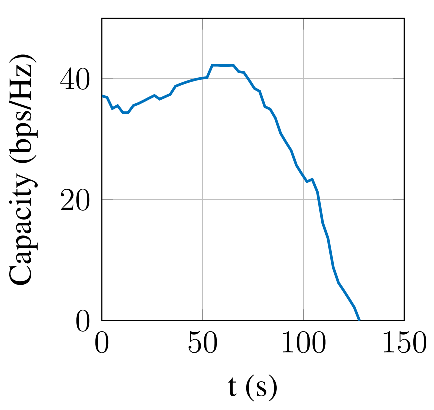
$$d_r = \frac{1}{2}\left(-l + \sqrt{l^2 + 4\epsilon^2}\right) \tag{4.24}$$



(a) Trajectories obtained.



(b) Evolution of the speeds.



(c) Evolution of the system capacity.

Fig. 4.3. Simulation 1 results. Dashed line is pursuer, continuous is evader, points are relays. Jumps in the capacity function are due to the minimum SNR threshold: as game evolves, more relays have an SNR below that limit and when their SNR surpass that threshold, their contribution to total capacity becomes zero

so the maximum SIR in the capture region, obtained by substituting this value in the SIR expression, is

$$SIR_{max} = \frac{P_c}{P_j}\left(1 + \frac{l^2 + l\sqrt{l^2 + 4\epsilon^2}}{2\epsilon^2}\right) \tag{4.25}$$

Equaling (4.25) to the minimum SIR in the receivers allows us to obtain the optimum jammer power. Hence, if the jamming power is considered to be much higher than noise and the minimum SINR is considered to be $SINR_{min} = 1$, with $P_c = 1$, the minimum jammer power will be $P_j = 1.11$. A noise floor of power $N_0 = 10^{-4}$ is considered.

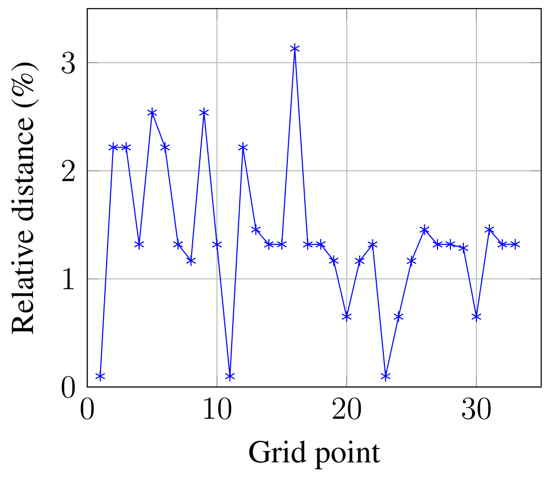
With these parameters, we simulated 100 UAVs optimal trajectories, using a time discretization of 50 points per trajectory. The capacity has been computed in each time step and the final results show that all simulations end up with capture and final capacity zero, so the jammer wins the game (as expected since the jammer is faster in this setup and its transmitted power is enough to cause all relay nodes SINR to fall below the threshold).

Figure 4.3 shows an example of trajectory on the plane, as well as the speed evolution of both players for initial conditions $(x_{p,0}, y_{p,0}) = (20, -20)$, $(x_{e,0}, y_{e,0}) = (-20, 20)$, $(u_{p,0}, v_{p,0}) = (-2, -2)$, $(u_{e,0}, v_{e,0}) = (0, -1)$.

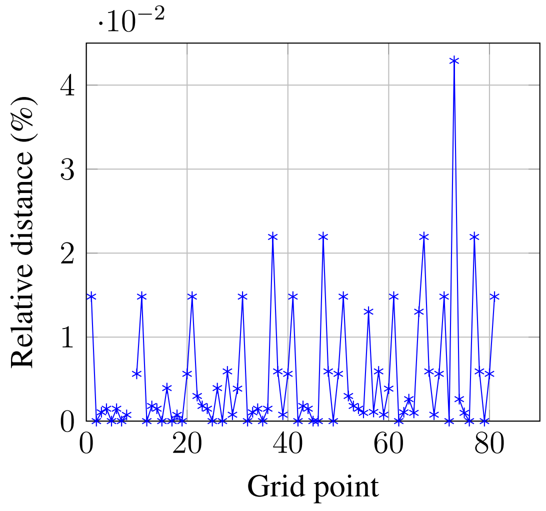
4.10.2 Simulation 2: Comparison between analytical, optimization and hybrid solution approaches

In this subsection, the three methods proposed in Subsections 4.7, 4.8 and 4.9 are implemented and compared. In order to do so, a grid has been defined over the initial position conditions, taking the following values: $x_{e,0}, y_{e,0} \in \{1, 6, 11\}$, $x_{p,0}, y_{p,0} \in \{-10, -5, 0\}$. Hence, the grid has 81 points. The rest of the parameters have the following values: $u_{e,0} = v_{e,0} = 1$, $u_{p,0} = v_{p,0} = -1$, $v_{max,e} = 1$, $v_{max,p} = 2$, $F_e = F_p = 1$, $l = 1$, $D = 100$, $N = 100$, $P_j = 1.11$ and $P_c = 1$ for a SINR threshold of $SINR_{min} = 1$, according to (4.25).

The non-convex optimization algorithm used [30] [31] in the optimization and hybrid methods stops when a certain number of iterations have been done, regardless of whether a solution was found or not. In order to test how this affects to solution obtaining, the algorithm has been run three times over the proposed grid, using a different number of iterations in each run. In the optimization method, the number of iterations chosen were $\{10^3, 10^4, 10^5\}$, whereas in the hybrid approach, the number of iterations chosen are $\{10^2, 10^3, 10^4\}$.



(a) Optimization approach.



(b) Hybrid approach.

Fig. 4.4. Relative distances between the results of the optimization and hybrid approach compared to the analytical solution of the system, from Simulation 2. Since a solution is not always found by all methods (see Table 4.1), some distances can not be computed.

In order to consider a solution valid in both hybrid and optimization approaches, a threshold has been defined over the cost function from (4.22). Since the cost will be smaller than one if and only if capture happens, the threshold taken was 0.9. Hence, if the point found by any of these two approaches after a certain number of iterations yield a cost smaller than this threshold, it is considered to be the valid optimum. A comparison of relative distance can be seen in Figure 4.4, where this relative distance has been computed as:

$$d_{rel} = \frac{||x_a - x_b||_2}{||x_a||_2} \tag{4.26}$$

where $||x||_2$ is the Euclidean norm of vector x, x_a is the solution vector that the analytical method provides - its two components are final heading angle and final time, $x_a = (t_f, s_5)$ and x_b is the solution vector that either optimization or hybrid method gives. Hence, this is a relative measure of how far are the solutions: a smaller value means that solutions found are close between the methods tested. In Figure 4.4 is is possible to see that for the hybrid method, this relative distance is always inferior to 0.05%, whereas for optimization approach, it is always below 3.5%.

Finally, Table 4.1 presents the results obtained with each method. It is possible to see that the hybrid method yields the highest number of solutions found, being able to find all the solutions for the proposed grid points. The second best solution is the analytical method, and the worse in number of solutions found is the optimization approach.

		Grid points where solution was found	%
Analytical approach		80	98.8
Optimization approach	10^3 iterations	9	11.1
	10^4 iterations	21	25.9
	10^5 iterations	33	40.7
Hybrid approach	10^2 iterations	59	72.8
	10^3 iterations	80	98.8
	10^4 iterations	81	100

Table 4.1. Comparison of analytical, optimization and hybrid approaches for finding the solutions to the game.

Comparing all the approaches, it is possible to see that the hybrid method yields better performance than optimization method thanks to having more information: knowing final time allows doing a search in only one dimension. The drawback is that it needs to solve a nonlinear expression for final time, but it achieves a solution with a smaller relative distance and it takes less iterations - which means less computation cost and time. Finally, analytical method is the fastest, but due to the nonlinearity of the system to be solved, a solution is not always achieved - in the proposed grid, though, that happened only once.

Chapter 5

Capacity game of two UAVs

5.1 Introduction

In this chapter, Isaacs' method, described in [10, ch. 4] will be used to solve an approximation of the capacity game described in Chapter 3. The running cost L will be considered to be linear (3.14):

$$L = A + Br$$

where $r = (y_c - y_j)^2 + (x_c - x_j)^2$, A and B are constants whose expression is obtained from (3.13) as it is shown in (3.15). The final cost G will be considered to be zero. As in Chapter 4, we consider each UAV to have a constant acceleration and a friction limit. Again, this setup is an extension to Isaacs "isotropic rocket" game [10, pp. 105-116], but considering that pursuer and evader have the same dynamics and not using a constant running cost.

5.2 Dynamics of the UAVs

We consider the player to have the same control variable as in the previous chapter, which will be their heading angle with respect to y-axis. Hence, there will be eight states, as in the previous case, and the dynamics of pursuer and evader are the same as in (4.1).

5.3 Control optimization

Building the Hamiltonian using Isaacs "Main Equation" [10, p. 67] yields:

$$\begin{aligned}\max_{\psi}\min_{\phi} & V_{x_p}u_p + V_{y_p}v_p + V_{u_p}(F_p\sin(\phi) - k_p u_p) + V_{v_p}(F_p\cos(\phi) - k_p v_p) \\ & + V_{x_e}u_e + V_{y_e}v_e + V_{u_e}(F_e\sin(\psi) - k_e u_e) + V_{v_e}(F_e\cos(\psi) - k_e v_e) \\ & + A + Br = 0\end{aligned} \tag{5.1}$$

Using that controls are separable allows:

$$\begin{aligned} A{+}Br &+ V_{x_p}u_p + V_{y_p}v_p \\ &+ \min_{\phi}\Big(V_{u_p}(F_p\sin(\phi) - k_pu_p) + V_{v_p}(F_p\cos(\phi) - k_pv_p)\Big) + V_{x_e}u_e + V_{y_e}v_e \\ &+ \max_{\psi}\Big(V_{u_e}(F_e\sin(\psi) - k_eu_e) + V_{v_e}(F_e\cos(\psi) - k_ev_e)\Big) = 0 \end{aligned} \tag{5.2}$$

The optimization problems in (5.2) is solved using the same approach as in Section 4.3, and the Hamiltonian (5.2) becomes:

$$\begin{aligned} A + Br{+}V_{x_p}u_p + V_{y_p}v_p - \rho_pF_p - k_p(V_{v_p}v_p + V_{u_p}u_p) \\ +V_{x_e}u_e + V_{y_e}v_e + \rho_eF_e - k_e(V_{v_e}v_e + V_{u_e}u_e) = 0 \end{aligned} \tag{5.3}$$

5.4 Retrogressive Path Equations

The sixteen Retrogressive Path Equations (RPE) are obtained using the same approach as in Section 4.4. The eight equations that depend on the dynamics equation are in (4.7). The other eight RPEs are obtained from the gradient of the value function, using (2.12), where ME stands again for "Main equation", and hence, these RPEs are obtained through derivation of the obtained Main Equation (5.3) with respect to each state variable. The resulting RPEs are:

$$\begin{pmatrix} \overset{\circ}{V}_{x_p} \\ \overset{\circ}{V}_{y_p} \\ \overset{\circ}{V}_{u_p} \\ \overset{\circ}{V}_{v_p} \\ \overset{\circ}{V}_{x_e} \\ \overset{\circ}{V}_{y_e} \\ \overset{\circ}{V}_{u_e} \\ \overset{\circ}{V}_{v_e} \end{pmatrix} = \begin{pmatrix} -2B(x_e - x_p) \\ -2B(y_e - y_p) \\ V_{x_p} - k_pV_{u_p} \\ V_{y_p} - k_pV_{v_p} \\ 2B(x_e - x_p) \\ 2B(y_e - y_p) \\ V_{x_e} - k_eV_{u_e} \\ V_{y_e} - k_eV_{v_e} \end{pmatrix} \tag{5.4}$$

Notice that these RPEs are different from the ones obtained in the case before, which can be seen in (4.8), because of using a different running cost.

5.5 Final conditions

As in Section 4.5, we consider that the capture distance is l and that the surface capture will be the ball whose center is the evader position and whose radius is l. Its parametrization can be found in (4.9). Using (4.9) and (2.13), taking into account that in this game, the final cost G is zero, the final conditions obtained are the same which were obtained in Section 4.5, which are the expressions in (4.10).

From (4.10), the two first equations and the fifth show that, in the terminal sphere, $-V_{x_p} = V_{x_e} = \lambda\sin(s_5)$ and $-V_{y_p} = V_{y_e} = \lambda\cos(s_5)$. Also, from the rest of equations in (4.10) and that $\rho_p = \sqrt{V_{u_p}^2 + V_{v_p}^2}$ and $\rho_e = \sqrt{V_{u_e}^2 + V_{v_e}^2}$, in the terminal manifold, $\rho_e = \rho_p = 0$. Substituting all that in (5.3) yields:

$$-\lambda s_3 \sin(s_5) - \lambda s_4 \cos(s_5) + \lambda s_6 \sin(s_5) + \lambda s_7 \cos(s_5) + A + Br = 0 \tag{5.5}$$

Manipulating (5.5) gives the following result for λ:

$$\lambda = \frac{A + B\left((y_c - y_j)^2 + (x_c - x_j)^2\right)}{(s_3 - s_6)\sin(s_5) + (s_4 - s_7)\cos(s_5)} \tag{5.6}$$

where the value of r was substituted. The expression in the denominator can be simplified: if final speeds of pursuer and evader are called, respectively, $v_{f,p}$ and $v_{f,e}$, we have that:

$$\begin{aligned} s_3 &= v_{f,p}\sin(s_5) \\ s_4 &= v_{f,p}\cos(s_5) \\ s_6 &= v_{f,e}\sin(s_5) \\ s_7 &= v_{f,e}\cos(s_5) \end{aligned} \tag{5.7}$$

Hence, replacing and manipulating in (5.6), taking into account that $\cos^2(s_5) + \sin^2(s_5) = 1$ yields the following expression for λ:

$$\lambda = \frac{A + B\left((y_c - y_j)^2 + (x_c - x_j)^2\right)}{v_{f,p} - v_{f_e}} \tag{5.8}$$

5.6 RPEs integration

Let us start integrating the equations in (5.4). The four equations for $V_{x_e}, V_{y_e}, V_{x_p}, V_{y_p}$ are solved using the initial condition found in the previous section and it yields:

$$\begin{aligned} -V_{x_p} = V_{x_e} &= \lambda\sin(s_5) - 2B\tau(x_p - x_e) \\ -V_{y_p} = V_{y_e} &= \lambda\cos(s_5) - 2B\tau(y_p - y_e) \end{aligned} \tag{5.9}$$

where λ is defined in (5.8). The other four RPEs in (5.4) are solved by replacing the values of $V_{x_e}, V_{y_e}, V_{x_p}, V_{y_p}$ that are in (5.9) and using the initial conditions (in retro time) from (4.10). The integrated equations are:

$$\begin{aligned} Vu_p &= \frac{1}{k_p^2}e^{-k_p\tau}\left(-2B\left(e^{k_p\tau}(k_p\tau - 1) + 1\right)(x_e - x_p) - k_p\lambda\left(e^{k_p\tau} - 1\right)\sin(s_5)\right) \\ Vv_p &= \frac{1}{k_p^2}e^{-k_p\tau}\left(-2B\left(e^{k_p\tau}(k_p\tau - 1) + 1\right)(y_e - y_p) - k_p\lambda\left(e^{k_p\tau} - 1\right)\cos(s_5)\right) \\ Vu_e &= \frac{1}{k_e^2}e^{-k_e\tau}\left(2B\left(e^{k_e\tau}(k_e\tau - 1) + 1\right)(x_e - x_p) + k_e\lambda\left(e^{k_e\tau} - 1\right)\sin(s_5)\right) \\ Vv_e &= \frac{1}{k_e^2}e^{-k_e\tau}\left(2B\left(e^{k_e\tau}(k_e\tau - 1) + 1\right)(y_e - y_p) + k_e\lambda\left(e^{k_e\tau} - 1\right)\cos(s_5)\right) \end{aligned} \tag{5.10}$$

It is possible to see that the expressions in (5.9) and (5.10) are more complex than their equivalents in the problem described in the previous chapter, which were (4.14) and (4.15). The optimal controls can be obtained now: since $\rho_p = \sqrt{V_{u_p}^2 + V_{v_p}^2}$ and $\rho_e = \sqrt{V_{u_e}^2 + V_{v_e}^2}$, substituting the values above yield the following expressions for the controls:

$$\begin{aligned}
\cos(\phi^*) &= \frac{A_\phi}{\sqrt{A_\phi^2 + B_\phi^2}} \\
\sin(\phi^*) &= \frac{B_\phi}{\sqrt{A_\phi^2 + B_\phi^2}} \\
\cos(\psi^*) &= \frac{A_\psi}{\sqrt{A_\psi^2 + B_\psi^2}} \\
\sin(\psi^*) &= \frac{B_\psi}{\sqrt{A_\psi^2 + B_\psi^2}}
\end{aligned} \tag{5.11}$$

where:

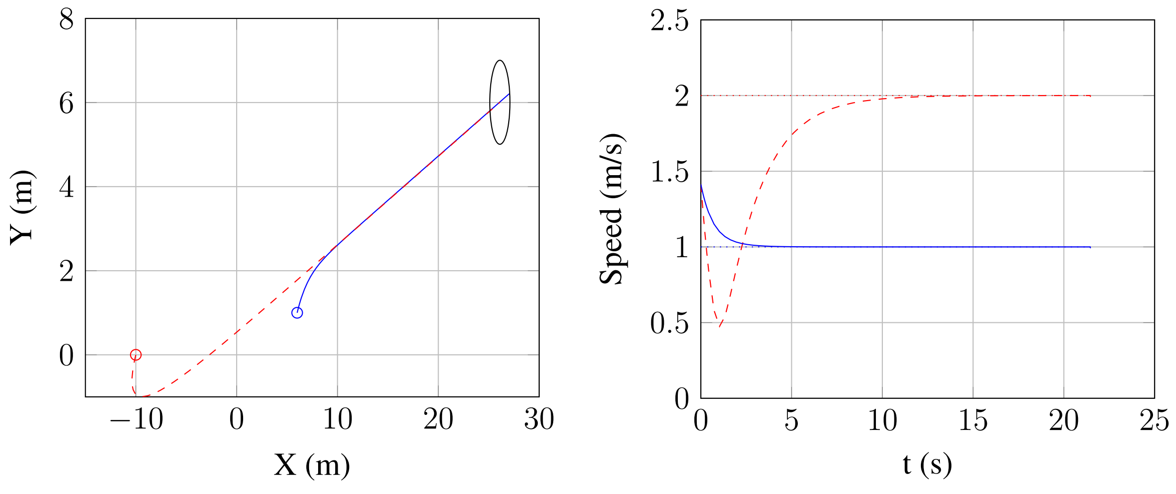
$$\begin{aligned}
A_\phi &= 2B\left(e^{k_p\tau}(k_p\tau - 1) + 1\right)(y_e - y_p) + k_p\lambda\left(e^{k_p\tau} - 1\right)\cos(s_5) \\
B_\phi &= 2B\left(e^{k_p\tau}(k_p\tau - 1) + 1\right)(x_e - x_p) + k_p\lambda\left(e^{k_p\tau} - 1\right)\sin(s_5) \\
A_\psi &= 2B\left(e^{k_e\tau}(k_e\tau - 1) + 1\right)(y_e - y_p) + k_e\lambda\left(e^{k_e\tau} - 1\right)\cos(s_5) \\
B_\psi &= 2B\left(e^{k_e\tau}(k_e\tau - 1) + 1\right)(x_e - x_p) + k_e\lambda\left(e^{k_e\tau} - 1\right)\sin(s_5)
\end{aligned} \tag{5.12}$$

It is possible to see that the optimal controls in (5.11) are neither constant nor equal for both players, as it happened in the problem in the previous chapter (see (4.16)). In this case, trajectories of both players are coupled, and the game is still open loop: optimal trajectories and controls, though coupled, can be obtained from initial conditions of the game.

The complex expressions for the controls in (5.11) causes that obtaining a closed expression for speeds and trajectories is pretty hard. Also, since the controls depend on λ and λ depends on the final conditions (5.8), if there are no closed expressions for the trajectories, the approach followed in Section 4.6 cannot be used to obtain the final conditions using the initial conditions. Hence, in order to solve this game, a similar approach to the one described in Section 4.8 will be used.

5.7 Simulation 3: Optimization approach solution to capacity game

In order to extend the approach proposed in Section 4.8 to this capacity game, the same grid used there for the initial conditions will be used here, that is, $x_{e,0}, y_{e,0} \in \{1, 6, 11\}$, $x_{p,0}, y_{p,0} \in \{-10, -5, 0\}$. The rest of the parameters have the following values: $u_{e,0} =$



(a) Positions (b) Speeds.

Fig. 5.1. Example of trajectories obtained for capacity game, without using Δv approximation (as in (5.14)). Initial grid conditions are $x_{e,0} = 6$, $y_{e,0} = 1$, $x_{p,0} = -10$, $y_{p,0} = 0$. The rest of parameters are described in Section 5.7. Dashed red line is the pursuer, whereas continuous blue line is the evader. The initial positions are marked in positions trajectory with a circle, and final capture region is the black circle centered in the jammer: capture happens when evader enters that region. In speeds plot, the point line is the speed limit for each player.

$v_{e,0} = 1$, $u_{p,0} = v_{p,0} = -1$, $v_{max,e} = 1$, $v_{max,p} = 2$, $F_e = F_p = 1$, $l = 1$, $D = 100$, $N = 100$, $P_j = 1.11$ and $P_c = 1$ for a SINR threshold of $SINR_{min} = 1$, according to (4.25).

The control equations in (5.11) will be used to numerically solve the system in (4.1) and hence, obtain the trajectories - that is, velocities and position of pursuer and evader, respectively. The numerical solver used is not the same that was described in Section 4.8, since the ODE system might become stiff and hence, a different method is required in order to be time-efficient. In this case, a variable-step, variable-order solver based on the numerical differentiation formulas of orders 1 to 5 is used, combined with Gear's method [32].

The non-convex optimization algorithm used will be the same that was used in previous chapter (SOO), but this time, the search will be performed over three dimensions, since there are three initial parameters to be obtained: final heading angle and final time (s_5 and t_f respectively) as in the game solved in Section 4.8, and the final difference of speeds, $v_{f,p} - v_{f_e}$, which is required to solve (5.8). The number of iterations chosen are $\{10^3, 10^4, 10^5\}$.

Finally, the cost function will be adapted from (4.22) and it will be the following:

$$f_{c2} = \frac{k_1}{1 + e^{-k_2(d_f - l)}} + k_3|s_5 - \hat{s_5}| + k_4|\Delta v_f - \hat{\Delta v_f}| \tag{5.13}$$

where the first two terms are the same than in (4.22) and the third one is due to the final difference of speeds, where Δv_f corresponds to the final difference of speeds obtained

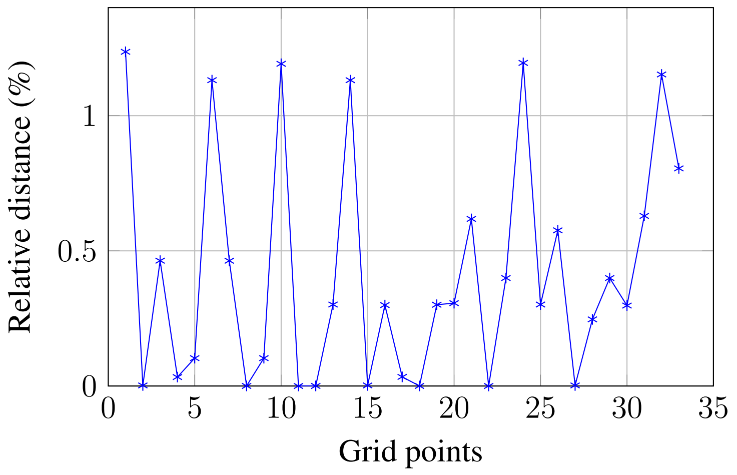


Fig. 5.2. Relative distance in final conditions triplets between optimization and $\hat{\Delta v}$ approach, computed using the expression in (5.15).

introduced a priori, whereas $\hat{\Delta v_f}$ corresponds to the final difference of speeds in the trajectories numerically obtained. Hence, this cost function tries to minimize the error between final heading angle and final difference of speeds, as well as adding a term if capture does not happen. In this simulation, $k_1 = k_3 = k_4 = 1$ and $k_2 = 500$. An illustration of the steps followed in this method can be found in Algorithm 4.

Algorithm 4 Steps for the optimization approach

```
1: Obtain initial conditions and game parameters
2: while Cost in (5.13) is greater than threshold do
3:     Guess a triple $(s_5, t_f, \Delta v)$
4:     Solve ODE system numerically from (4.1), using (5.11) and the $(s_5, t_f, \Delta v)$ triple
       guessed
5:     Obtain capture time, $\hat{s_5}$ and $\hat{\Delta v_f}$ from trajectories
6:     Compute cost for the triple $(s_5, t_f, \Delta v)$ using (5.13)
7: end while
8: The triple $(s_5, t_f, \Delta v)$ is correct: optimal trajectories are obtained by solving ODE
   system numerically from (4.1), using that $(s_5, t_f, \Delta v)$ triple
   {SOO is used in steps 2-7}
```

Also, an approximation of this method will be tested. Considering that final time t_f is sufficiently high for both players to be able to accelerate until they reach their speed limits, it is possible to approximate the final difference of speeds as follows:

$$\hat{\Delta v} = v_{max,p} - v_{max,e} \approx v_{f,p} - v_{f_e} \tag{5.14}$$

Using this approximation allows to reduce the dimensionality of the search to two dimensions, which means a smaller computational cost and time because we only search for final heading angle and final time (s_5 and t_f respectively) as in the game solved in

Section 4.8. The cost function used will be (5.13). Considering the final conditions triplet $(s_5, t_f, \Delta v)$, we define the relative distance as an error metric following this expression:

$$\frac{||\hat{\mathbf{x}} - \tilde{\mathbf{x}}||_2}{||\hat{\mathbf{x}}||_2} \tag{5.15}$$

where $||\mathbf{x}||_2$ denotes the Euclidean norm of vector $\mathbf{x}$, $\hat{\mathbf{x}}$ is the triplet of final conditions obtained with the optimization approach and $\tilde{\mathbf{x}}$ is the triplet of final conditions obtained with the $\hat{\Delta v}$ approximation, in which Δv follows the expression in (5.14). This relative distance can be observed in Figure 5.2 and we can see that this error is always smaller than 1.5%, hence, $\hat{\Delta v}$ approximation is validated. An illustration for the steps followed in this approximation can be found in Algorithm 5.

Algorithm 5 Steps for the $\hat{\Delta v}$ approximation approach

```
1: Obtain initial conditions and game parameters
2: Obtain an approximation of $\hat{\Delta v}$ using (5.14)
3: **while** Cost in (5.13) is greater than threshold **do**
4:     Guess a pair $(s_5, t_f)$
5:     Solve ODE system numerically from (4.1), using (5.11), the $(s_5, t_f)$ pair guessed
       and the $\hat{\Delta v}$ approximation
6:     Obtain capture time, $\hat{s_5}$ and $\hat{\Delta v_f}$ from trajectories
7:     Compute cost for the triple $(s_5, t_f, \Delta v)$ using (5.13)
8: **end while**
9: The triple $(s_5, t_f, \Delta v)$ is correct: optimal trajectories are obtained by solving ODE
   system numerically from (4.1), using that $(s_5, t_f, \Delta v)$ triple
   {SOO is used in steps 3-8}
```

The results obtained can be observed in Table 5.1, and are similar to the ones in Table 4.1 for the optimization approach. Since there are no closed expressions, we can not obtain relative distance measures. It is important to note that this game requires more iterations than the one in Table 4.1, and hence, the computational cost and time to solve this Capacity game increases with respect to the one in the previous chapter. It is possible to see also how the search over two dimensions yield more solutions with the same iterations, as expected: $\hat{\Delta v}$ approximation is less computationally costly. Finally, an example trajectory can be observed in Figure 5.1.

		Grid points where solution was found	%
Optimization approach	10^3 iterations	5	6.2
	10^4 iterations	7	8.7
	10^5 iterations	33	40.7
Optimization approach with $\hat{\Delta v}$ approximation	10^3 iterations	11	13.6
	10^4 iterations	44	54.3
	10^5 iterations	73	90.1

Table 5.1. Results obtained using optimization approach, with and without $\hat{\Delta v}$ approximation, for Capacity game.

Chapter 6

Comparison between games proposed

In Chapter 3 the main problem was posed: a UAV tries to communicate with some relays, whereas another UAV tries to jam that communication. The jamming was considered to be effective when both UAVs were within a certain distance. In order to solve the problem, the Shannon capacity of the system was computed and simplified to the expression in (3.14). Since the obtained capacity was a function of the distance, the following two approaches were followed in order to solve the problem:

- In chapter 4, a surrogate function approach was followed. Standard pursuit-evasion games pose the problem in terms of final time: the pursuer wants to minimize capture time, whereas the evader wants to maximize it. In other words, evader tries to be far from pursuer and pursuer tries to be close to jammer. Since capacity depends on the distance between players, a pursuit-evasion game whose running cost was $L = 1$ and whose payoff was the time of capture was posed and solved. The solution to this game was an open-loop, constant control, which was computed using three different approaches: an analytical, nonlinear expression, using non-convex optimization and using a hybrid method.

- In chapter 5, the game was solved in terms of capacity. The running cost was considered to be a linear function of the capacity, as in (3.14), and this game was solved. The solution to this game was an open-loop solution, which was solved using non-convex optimization, since no analytical expression was found for this game.

In this chapter, the trajectories and controls obtained in both approaches will be compared. Since the simulations done in the Sections before were run on the same grid of initial conditions for both games, it is straightforward to compare the results.

First, in Figure 6.1, it is possible to see two different examples of trajectories solved using different approaches for the same initial conditions, the first of them with very similar trajectories and the second is the trajectory with a high relative distance between trajectories. For game with running cost $L = 1$, the hybrid method is used, whereas for game with running cost $L = A + Br$, we use the optimization approach. We can compare the controls, the speeds and the trajectories on the plane. It is possible to see that, for the game with running cost $L = 1$, the controls are constant, whereas for the game with running cost $L = A + Br$, they are nearly constant. This small difference causes speeds and trajectories to be slightly different.

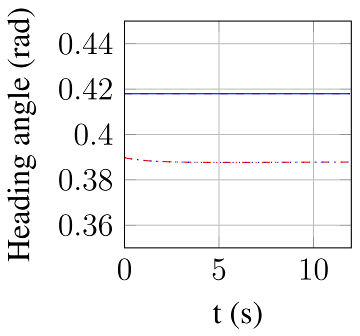
Secondly, a quantification of how much different the controls and trajectories are can be found in Table 6.1. The metric used is relative error in controls, which is computed as follows:

$$\frac{|\alpha_1 - \alpha_2|}{\alpha_1} \tag{6.1}$$

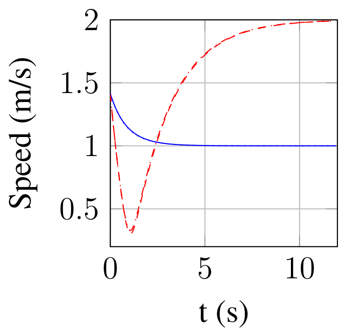
where α_1 is the heading angle in the case where running cost $L = A + Br$ and α_2 is the heading angle when $L = 1$. Since heading angle evolves with time in the first case, the relative error is computed along the whole trajectory for all the grid points of initial conditions on which both methods reach a solution, and this vector of relative errors is analyzed in Table 6.1. The methods compared are the hybrid method when $L = 1$ and for the case when $L = A + Br$, both the optimization approach and the $\hat{\Delta v}$ approximation are considered. In the first case, after computing the empirical cumulative distribution function (CDF), more than 90% of the errors are below 0.5%, whereas in the second case, more than 90% of the errors are below 1%. That means that it is possible to approach the second game by the first one, without getting a significant error.

	Mean (%)	Median (%)	Standard deviation (%)
Hybrid vs optimization approach	0.74	0.40	2.25
Hybrid vs $\hat{\Delta v}$ approximation	1.12	0.21	5.70

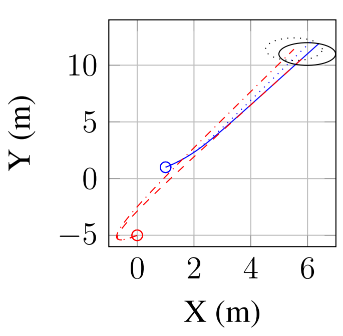
Table 6.1. Comparison of metrics over relative error in control, computed using (6.1).



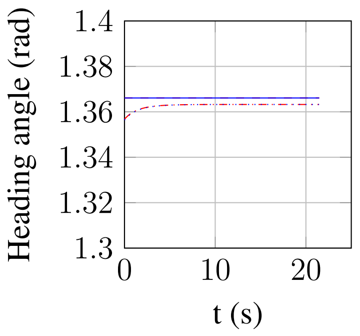
(a) Controls (case 1).



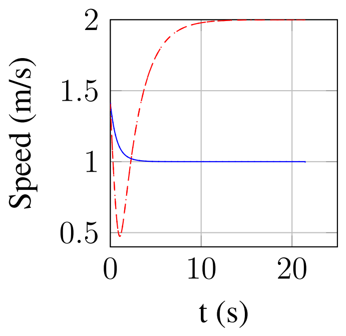
(b) Speeds (case 1).



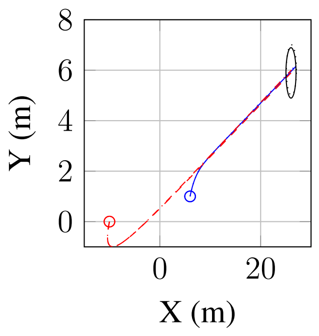
(b) Trajectories (case 1).



(a) Controls (case 2).



(b) Speeds (case 2).



(b) Trajectories (case 2).

Fig. 6.1. Comparison of controls, speeds and trajectories obtained for games with running cost $L = 1$ and $L = A + Br$. Initial grid conditions are $x_{e,0} = 1$, $y_{e,0} = 1$, $x_{p,0} = 0$, $y_{p,0} = -5$ for case 1 and $x_{e,0} = 6$, $y_{e,0} = 1$, $x_{p,0} = -10$, $y_{p,0} = 0$ for case 2. The rest of parameters are described in Section 5.7. Continuous blue line is evader and dashed red line is pursuer when $L = 1$, whereas dotted blue line is evader and dash-dot red line is pursuer when $L = A+Br$. It is possible to see that differences in control are small in case 2 and that means that trajectories are quite similar, but in case 1, the control differences are bigger and hence, trajectories vary more.

Chapter 7

Conclusions

We propose a new approach for solving games in stochastic scenarios, which consists in solving a pursuit-evasion game instead of a capacity one using an approximation. A concrete application to a jamming game has been studied.

The steps we have followed are the following:

- The communications maximum capacity has been computed in the environment we have posed. After obtaining it, we linearized this function in order to obtain a more tractable expression. We showed that, in our problem, maximum communications capacity depends linearly on the squared distance between players.

- The game was solved as a standard pursuit-evasion game, in which the payoff was the time of capture. In these games, the pursuer tries to capture evader as fast as possible, whereas evader tries to avoid capture as much time as possible. Since capture time is a function of distance, this might be seen as a surrogate function approach to our problem. This game was solved and the controls were obtained using three different approaches: solving a nonlinear system (analytical approach), performing an optimization over a non-convex, two-dimensional surface (optimization approach) and solving a non-linear equation and performing a non-convex optimization over a one-dimensional function (hybrid approach).

- The game was also solved considering that the total system capacity was the payoff, as a Zero-Sum game. This would be the exact solution to the game we posed. The solution has been obtained performing an optimization over a non-convex, three dimensional surface (optimization approach). It is possible, also, to approximate the solution if final time is high enough, so that the optimization has to be done only over a two-dimensional surface ($\hat{\Delta v}$ approximation).

- Both approaches were compared and it was shown that both yield very similar results, having a very small relative error. Hence, since the first game is faster and more efficient to solve, we have shown that it is possible to approximate accurately the exact solution by a simpler, faster to compute one, and yet be very accurate. Hence, the capacity game can be accurately approached as a standard pursuit-evasion one and be efficiently solved.

Chapter 8

Bibliography

8.1 References

[1] Wenyuan Xu, Timothy Wood, Wade Trappe, and Yanyong Zhang. Channel surfing and spatial retreats: defenses against wireless denial of service. In *Proceedings of the 3rd ACM workshop on Wireless security*, pages 80–89. ACM, 2004.

[2] Husheng Li and Zhu Han. Dogfight in spectrum: Combating primary user emulation attacks in cognitive radio systems, part i: Known channel statistics. *Wireless Communications, IEEE Transactions on*, 9(11):3566–3577, 2010.

[3] Husheng Li and Zhu Han. Dogfight in spectrum: combating primary user emulation attacks in cognitive radio systemspart ii: Unknown channel statistics. *Wireless Communications, IEEE Transactions on*, 10(1):274–283, 2011.

[4] Beibei Wang, Yongle Wu, KJ Liu, and T Charles Clancy. An anti-jamming stochastic game for cognitive radio networks. *Selected Areas in Communications, IEEE Journal on*, 29(4):877–889, 2011.

[5] Wenjing Wang, Shameek Bhattacharjee, Mainak Chatterjee, and Kevin Kwiat. Collaborative jamming and collaborative defense in cognitive radio networks. *Pervasive and Mobile Computing*, 9(4):572–587, 2013.

[6] Suman Bhunia, Xing Su, Shamik Sengupta, and Felisa Vázquez-Abad. Stochastic model for cognitive radio networks under jamming attacks and honeypot-based prevention. In *Distributed Computing and Networking*, pages 438–452. Springer, 2014.

[7] Konstantinos Pelechrinis, Marios Iliofotou, and Srikanth V Krishnamurthy. Denial of service attacks in wireless networks: The case of jammers. *Communications Surveys & Tutorials, IEEE*, 13(2):245–257, 2011.

[8] Tamer Basar and Geert Jan Olsder. *Dynamic noncooperative game theory*, volume 23. Siam, 1999.

[9] Joseph Lewin. *Differential games: theory and methods for solving game problems with singular surfaces*. Springer Science & Business Media, 2012.

[10] Rufus Isaacs. *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Corporation, 1999.

[11] Nikhil Karnad and Volkan Isler. Lion and man game in the presence of a circular obstacle. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5045–5050. IEEE, 2009.

[12] Jie Dong, Xu Zhang, and Xuemei Jia. Strategies of pursuit-evasion game based on improved potential field and differential game theory for mobile robots. In *Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2012 Second International Conference on*, pages 1452–1456. IEEE, 2012.

[13] AG Pashkov and SD Terekhov. A differential game of approach with two pursuers and one evader. *Journal of Optimization Theory and Applications*, 55(2):303–311, 1987.

[14] Sriram Shankaran, Dušan M Stipanović, and Claire J Tomlin. Collision avoidance strategies for a three-player game. In *Advances in Dynamic Games*, pages 253–271. Springer, 2011.

[15] Sourabh Bhattacharya, Seth Hutchinson, and Tamer Basar. Game-theoretic analysis of a visibility based pursuit-evasion game in the presence of obstacles. In *American Control Conference, 2009. ACC'09.*, pages 373–378. IEEE, 2009.

[16] Sourabh Bhattacharya and Tamer Basar. Game-theoretic analysis of an aerial jamming attack on a uav communication network. In *American Control Conference (ACC), 2010*, pages 818–823. IEEE, 2010.

[17] Sourabh Bhattacharya and Tamer Basar. Optimal strategies to evade jamming in heterogeneous mobile networks. In *Proceedings of the Workshop on Search and Pursuit-Evasion*, 2010.

[18] Juan Parras, Jorge Del Val, Santiago Zazo, Javier Zazo, and Sergio Valcarcel Macua. A new approach for solving anti-jamming games in stochastic scenarios as pursuit-evasion games. In *Statistical Signal Processing (SSP), 2016 IEEE Workshop on*. IEEE, 2016.

[19] Tamer Basar et al. Lecture notes on non-cooperative game theory. *Game Theory Module of the Graduate Program in Network Mathematics*, 2010.

[20] Olivier Chatain. Cooperative and non-cooperative game theory. 2014.

[21] John F Nash et al. Equilibrium points in n-person games. *Proc. Nat. Acad. Sci. USA*, 36(1):48–49, 1950.

[22] John Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

[23] Alberto Bressan. Noncooperative differential games. a tutorial. *Department of Mathematics, Penn State University*, 2010.

[24] David WK Yeung and Leon A Petrosjan. *Cooperative stochastic differential games*. Springer Science & Business Media, 2006.

[25] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. CRC Press, 1987.

[26] Richard Bellman. Dynamic programming princeton university press. *Princeton, NJ*, 1957.

[27] Joo P. Hespanha and Noncooperative Games. An introductory course in noncooperative game theory, available at http://www.ece.ucsb.edu/hespanha/ published, 2011.

[28] Hans P Geering. *Optimal control with engineering applications*, volume 113. Springer, 2007.

[29] John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

[30] R. Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *Advances in Neural Information Processing Systems 24 (NIPS)*, pages 783–791, Granada, Spain, Dec. 2011.

[31] R. Munos. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, 7(1), 2014.

[32] Lawrence F Shampine and Mark W Reichelt. The matlab ode suite. *SIAM journal on scientific computing*, 18(1):1–22, 1997.