2020

COMPUTATIONALLY EFFICIENT ALGORITHMS FOR OPTIMAL MOTION PLANNING AGAINST MULTI-DOMAIN SUPER SWARMS

Tsatsanifos, Theodoros

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

COMPUTATIONALLY EFFICIENT ALGORITHMS FOR OPTIMAL MOTION PLANNING AGAINST MULTI-DOMAIN SUPER SWARMS

by

Theodoros Tsatsanifos

June 2020

Thesis Advisor: Isaac I. Kaminer
Co-Advisor: Abram H. Clark IV

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 2020	3. REPORT TYPE AND DATES COVERED Master's thesis

4. TITLE AND SUBTITLE COMPUTATIONALLY EFFICIENT ALGORITHMS FOR OPTIMAL MOTION PLANNING AGAINST MULTI-DOMAIN SUPER SWARMS	5. FUNDING NUMBERS
6. AUTHOR(S) Theodoros Tsatsanifos	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	12b. DISTRIBUTION CODE A

13. ABSTRACT (maximum 200 words)

This thesis develops robust tactics for countering multi-domain super swarms. Previous research has provided tools for assessing adversarial swarms' internal cooperating strategies, quantifying risk based on swarm and weapons models, and generating optimal defender trajectories. In this research, we develop a simulation-based testbed for experimental validation of these strategies and a database of adversarial swarming models against which to test.

In this research, the aforementioned simulation-based testbed is examined from the perspective of computational efficiency. A significant computational advantage is obtained by replacing the Runge Kutta with the Verlet integration scheme frequently used in the molecular dynamics community. This almost equally numerically stable framework offers an impressive performance advantage.

Additionally, this research seeks to find the ideal balance between the deterministic nature of the dynamics of adversarial swarms and the requirement for a probabilistic approach in order to model the mutual attrition between opposing agents during combat situations. To achieve this end, several models of dynamics and attrition are introduced for optimal motion planning. Their outcomes are compared with a Monte Carlo simulation model, in which survivability is partially influenced by random number generation that aims to simulate the unpredictability exhibited in the real world.

14. SUBJECT TERMS xSwarm, optimal motion planning, swarm internal cooperation strategy	15. NUMBER OF PAGES 113
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

COMPUTATIONALLY EFFICIENT ALGORITHMS FOR OPTIMAL MOTION PLANNING AGAINST MULTI-DOMAIN SUPER SWARMS

Theodoros Tsatsanifos
Lieutenant, Hellenic Navy
BAS, Hellenic Naval Academy, 2008

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN APPLIED PHYSICS

and

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2020

Approved by: Isaac I. Kaminer
Advisor

Abram H. Clark IV
Co-Advisor

Kevin B. Smith
Chair, Department of Physics

Garth V. Hobson
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis develops robust tactics for countering multi-domain super swarms. Previous research has provided tools for assessing adversarial swarms' internal cooperating strategies, quantifying risk based on swarm and weapons models, and generating optimal defender trajectories. In this research, we develop a simulation-based testbed for experimental validation of these strategies and a database of adversarial swarming models against which to test.

In this research, the aforementioned simulation-based testbed is examined from the perspective of computational efficiency. A significant computational advantage is obtained by replacing the Runge Kutta with the Verlet integration scheme frequently used in the molecular dynamics community. This almost equally numerically stable framework offers an impressive performance advantage.

Additionally, this research seeks to find the ideal balance between the deterministic nature of the dynamics of adversarial swarms and the requirement for a probabilistic approach in order to model the mutual attrition between opposing agents during combat situations. To achieve this end, several models of dynamics and attrition are introduced for optimal motion planning. Their outcomes are compared with a Monte Carlo simulation model, in which survivability is partially influenced by random number generation that aims to simulate the unpredictability exhibited in the real world.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

FOV	Field of View
HVU	high value unit
MD	molecular dynamics
MD	Molecular Dynamics
NLP	non-linear problem
OCP	optimal control problem
ODE	Ordinary Differential Equations
UAV	Unmanned Autonomous Vehicle
VLAP	Virtual Leaders and Artificial Potentials

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First of all, I would like to express the deepest appreciation to my thesis advisors, Dr. Isaac Kaminer and Dr. Abe Clark, for their guidance and continuous support. They were always there for me whenever I asked them for help and always eager to support me with their expertise.

In addition, I feel very grateful for the collaboration that I had with Dr. Claire Walton, who provided me with valuable assistance and comments for this thesis.

Last but not least, I would like to express my very profound gratitude to my beloved wife, Eleni, who is currently carrying my upcoming daughter, Maria, for being my amazing companion and assistant during the fascinating journey of my studies. Thank you so much.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCING THE OPTIMAL MOTION PLANNING PROBLEM FOR COUNTER - SWARMING

A. AUTONOMY IN THE WARFARE OF TODAY AND TOMORROW

In 1954, the American mathematician Norbert Wiener, father of Cybernetics, in his article “Men, Machines and the World About” [1] argued that a machine is more likely to use better judgment in an emergency than a human would. A human who is not trained to deal with a specific emergency situation will almost certainly make the wrong decision when such an event arises.

In [1] Wiener wrote that the first industrial revolution, which changed our lives in all possible aspects, enabled us to replace human and animal power with the much superior strength and endurance of the machine. But he added, in 1954, a new industrial revolution was taking place that would replace less complex human judgment with the discrimination of the machine. This fact cannot be avoided; rather, it could used by humans to their advantage.

According to the U.S. Department of Defense (DOD) document, Unmanned Systems Roadmap (2007–2032) [2], the use of autonomous systems in the military domain is already established and going to increase continuously with the integration in operations of systems that have augmented capabilities made possible by the technological advancements of our time. That development is desirable because autonomous systems represent a relatively inexpensive way to project power with the minimum loss of human life. Moreover, as we may see in Figure 1, today autonomous systems are widely employed in every branch of the military and in every type of operation. Furthermore, the use of such systems is going to continue to grow in the near future.

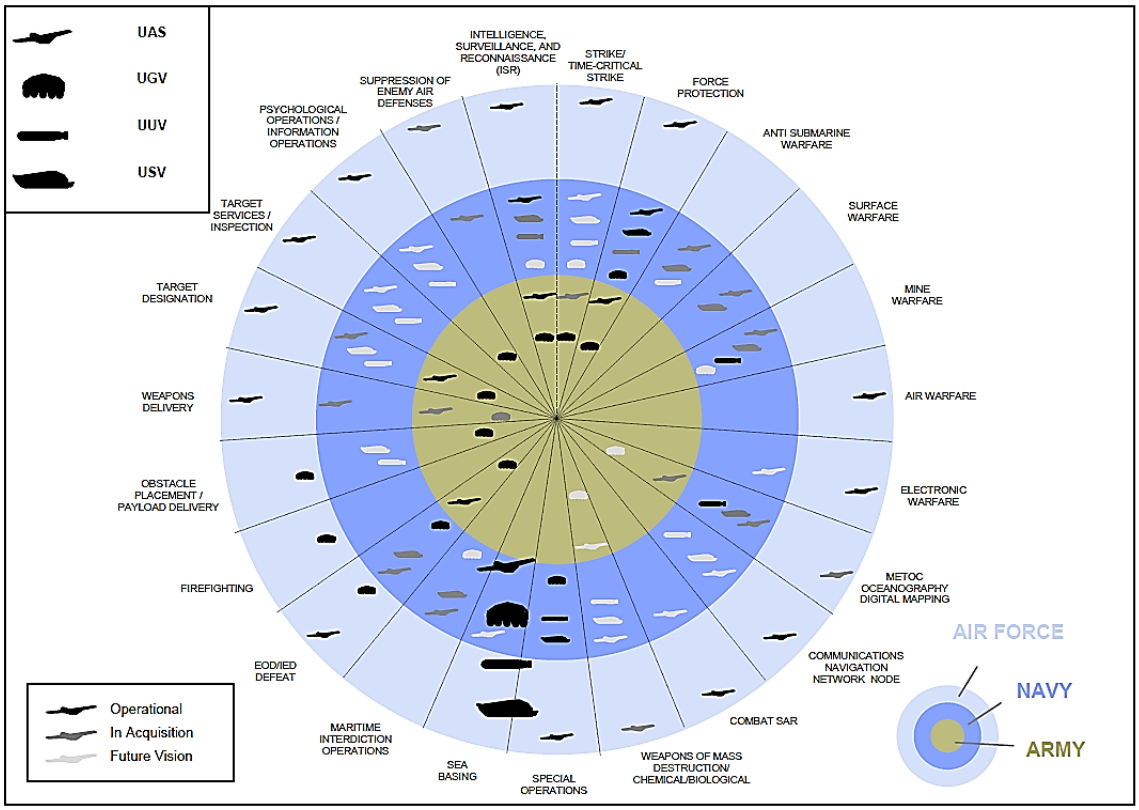

Figure 1. DOD unmanned systems, present and future roles. Source: [2].

B. INTRODUCING THE COUNTER-SWARMING PROBLEM

In this thesis research we are addressing the problem of defending against super swarms, consisting of approximately 500 autonomous aerial assets. The proposed solution utilizes a modeling framework from [3] that enables the efficient computation of numerical solutions for the task of trajectory generation.

In our simulation model, multiple opposing agents with pairwise interaction dynamics and a model of reciprocal attrition are combined with a cost function that encompasses a broad class of mission objectives. Our model output is trajectories that become the inputs in the high level controllers of the defending forces. These paths are optimized for combat situations with rapid fire rate, and multiple attacking and defending agents where the interaction forces matter for obstacle avoidance purposes, but have to be correctly constrained in order to avoid unrealistic solutions. Computationally efficient

algorithms are implemented in order to project our results in situations with a high number of agents inside the swarms.

In this thesis research report we are presenting the challenges of modeling the interaction forces between opposing swarms inside the aforementioned framework, and we are proposing revised interaction models that give more realistic solutions in our counter-swarm problem. Two sets of results are presented in Chapters IV and V for the following two different mission objectives:

1. High Value Unit Protection

The mission objective of High Value Unit (HVU) protection was initially introduced by the authors C. Walton, P. Lambrianides, I. Kaminer, J. Royset, and Q. Gong in 2016 in their journal article [3]. In this mission, an incoming attacking swarm has to be intercepted effectively by the defender forces before any agent of the attacking swarm can harm the HVU. In the results that we present in Chapter IV, the critical unit that we have to protect is a fixed asset with no self- defense capabilities, whereas the attackers are deemed dispensable swarm agents heading toward the HVU in a "kamikaze" like attack.

Each attacker has deterministic dynamics but its starting point varies in every scenario. In the first set of results they are approaching from a single threat direction. In the second set of results we address a much more challenging problem, because the threat directions are multiple. The attacking swarm has already encircled the HVU and the defenders have to spread towards all the thread directions in order to effectively protect the critical unit.

The goal of the scenario is to maximize the probability of HVU survival in the event of a large-scale swarm attack. We have to maximize this probability given the available defending forces' control inputs and constraints. There are two ways that the defenders can achieve a high survival probability for the HVU, and these approaches have been discussed in detail in [3].

The first way is by exploiting their weapons capabilities, such as fire rate, range, field of view, and dead sectors, which are included in the damage function. Consequently, the aim is the neutralization of attacker capabilities, which in turn decreases the destruction probability of the HVU.

The second way is through the defenders' capability to repulse the attackers away from the HVU. This herding technique exploits the collision avoidance algorithms that all potential-based swarming methods use as reactive motion planning. Herding strategies use this collision-avoidance path planning algorithm in order to protect the critical unit by guiding the attacking swarm away from the asset.

2. Air Superiority Operations

In most military plans air superiority is desired or even required for at least a specific window of time during the execution of military operations. Consequently, in this second scenario, we are launching our autonomous vehicles in an area controlled and patrolled by a large number of hostile drones and we are seeking the optimal trajectories for our autonomous systems to maximize inflicted attrition on the opposing forces and provide us the necessary conditions for air superiority.

Again, our mission objectives and constraints are captured in our cost function, which is now related with the probability of destruction of the hostile swarm.

C. INTRODUCING THE TRAJECTORY GENERATION PROCEDURE USING BERNSTEIN POLYNOMIALS AND BEZIER CURVES

The analytical form of the Optimal Control Problem (OCP) is used to **determine** the state vector $x(t)$ and the control vector $u(t)$ that **minimize** the cost function, expressed in Equation (1).

$$J = E(x(0), x(t_f)) + \int_0^{t_f} F(x(t), u(t))dt \tag{1}$$

where $E(x(0), x(t_f))$ is called the terminal cost and $\int_0^{t_f} F(x(t), u(t))dt$ is called the running cost. In our problem formulation we have running cost that in the HVU protection scenario corresponds to the probability of destruction of the HVU. In the case of the air superiority scenario, we still have a scalar cost, which corresponds to the average probability of survival of the opposing swarm. In other words, our gradient-based constraint optimizer in Matlab (function fmincon) may find the case of local minimum cost, and subsequently, the problem has to be formulated appropriately.

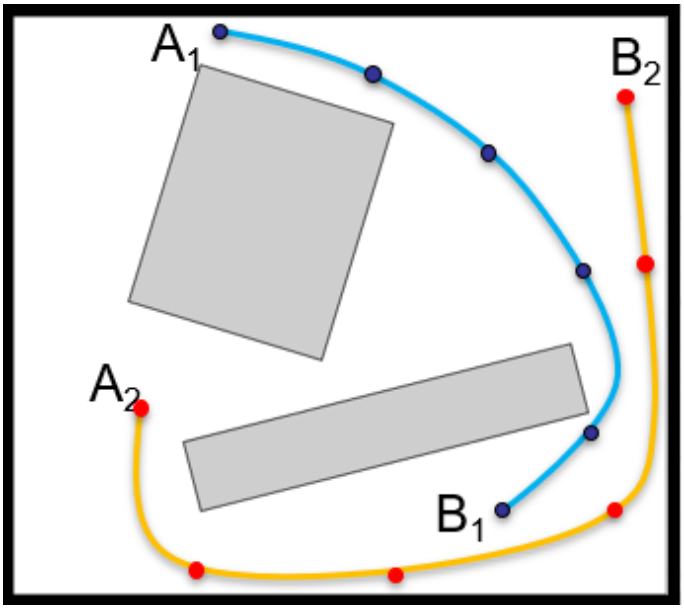

Figure 2. Discretization and interpolation of the non-linear optimal control problem. Source: [4].

The problem is **subject** to the system dynamics, equality, and inequality constraints, expressed in Equations (2)–(4).

$$\dot{x} = f(x(t), u(t)), \forall t \in [0, t_f] \tag{2}$$

$$e(x(0), x(t_f)) = 0 \tag{3}$$

$$h(x(t),u(t)) \le 0, \forall t \in [0,t_f] \tag{4}$$

In most cases, analytical solutions to such non-linear problems are not easy to find. Hence, in our case, we are implementing a numerical framework where we discretize the time interval, the state vector $x(t)$, and the control vector $u(t)$, and we exploit the numerical stability of Berstein polynomials and Berstein coefficients that are shown in Equations (5) and (6), in order to express the state vector $x(t)$ and the control vector $u(t)$ in terms of the Berstein coefficients.

Figure 2 shows how we want to solve numerically the typical optimal control problem of reaching endpoint B1 or B2 from an initial point A1 or A2. The optimal encompasses our desire to follow the shortest route while we have to avoid hitting obstacles. Consequently, we try to discretize the problem, find the optimal solution, and finally interpolate between time nodes.

A degree n Berstein polynomial is given by [4], [5]

$$x_N(t) = \sum_{k=0}^{N} c_k b_{k,N}(t) \tag{5}$$

where $b_{k,N}(t)$ are the basis of the Bernstein polynomial

$$b_{k,N} = \binom{N}{k} t^N (t_f - t)^{N-k}, t \in [0,t_f] \tag{6}$$

and $c_k \in \Re^3$ are the Bernstein coefficients.

In Figure 3 we may see how a Bezier curve is going to be created by six Bernstein coefficients. The curve is confined inside the convex hull that these six coefficients create.

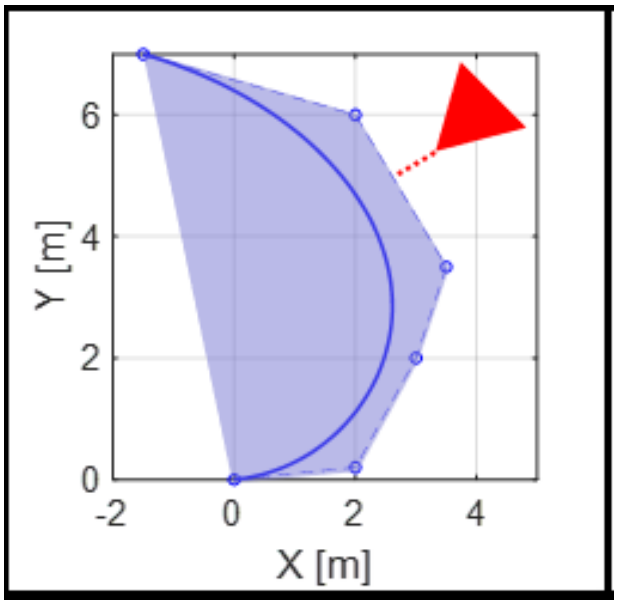

Figure 3. A Bernstein polynomial is contained within the convex hull defined by its Bernstein coefficients. Source: [6].

Consequently, the numerical framework of the problem, **asks** for the Berstein coefficients for the state c_k and the controls $c_{u,k}$, where $k = 0....N$ that **minimize** the cost function J and are **subject** to the following **constraints:**

$$\left\| \dot{x}_N(t_j) - f(x_N(t_j), u_N(t_j)) \right\| \leq N^{-\delta_p}, \forall j = 0,..., N \tag{7}$$

$$e(x_N(0), x_N(t_N)) = 0 \tag{8}$$

$$h(x_N(t_j), u_N(t_j) \leq N^{-\delta_p}, \forall j = 0,...., N \tag{9}$$

In Figure 4, the flow diagram of the required procedure is illustrated in order to replace the analytical OCP problem with a discretized version where a numerical solution through the Bezier coefficients is feasible.

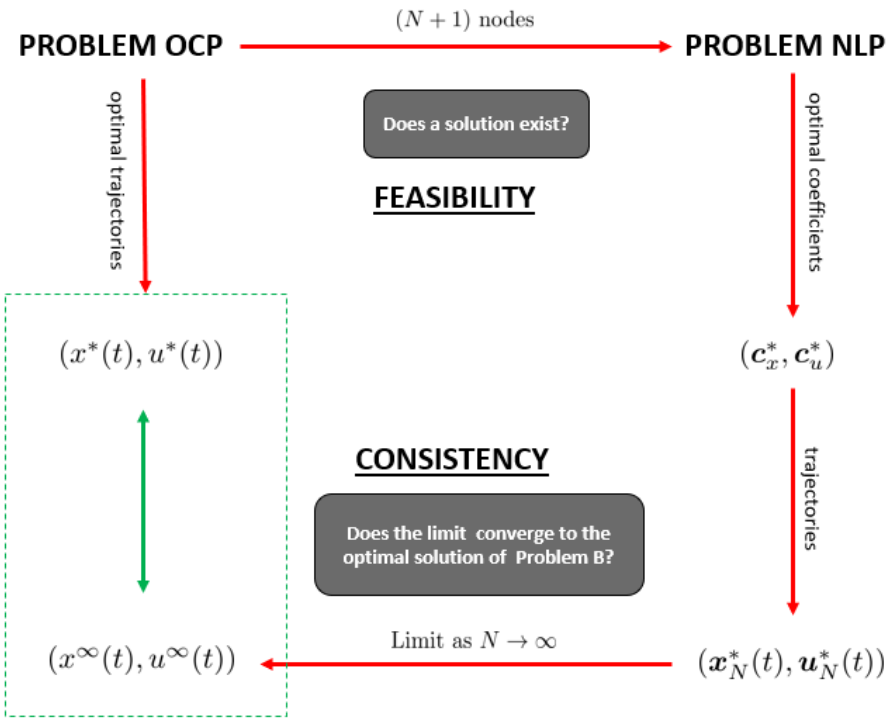

Figure 4. Flow diagram of the discretization of the original OCP in order to be solvable from our numerical tools. Source: [7].

The algorithm for the Bernstein polynomial approximation of the optimal trajectories solution is as follows:

- Discretize time into time nodes:

$$t_j = j\frac{t_f}{N}, j = 0.....N \tag{10}$$

- Apply Bernstein approximation of the state vector and the control vector [8]:

$$x(t) \approx x_N(t) = \sum_{k=0}^{N} c_k b_{k,N}(t) \tag{11}$$

$$u(t) \approx u_N(t) = \sum_{k=0}^{N} c_{u,k} b_{k,N}(t) \tag{12}$$

- Differentiate the state vector via the degree elevation matrix:

$$\dot{x} \approx \dot{x}_N(t) = \sum_{j=0}^{N-1}\left(\sum_{i=0}^{N} c_i D_{ij}\right) b_{j,N(t)} \tag{13}$$

$$\text{where } D = \begin{bmatrix} -\frac{N}{t_f} & 0 & \cdots & 0 \\ \frac{N}{t_f} & \ddots & \cdots & \vdots \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & -\frac{N}{t_f} \\ 0 & \cdots & \cdots & \frac{N}{t_f} \end{bmatrix} \tag{14}$$

With degree elevation, which is implemented through the matrix D, we obtain equivalent Bernstein polynomials of a higher order.

- Approximate the running cost:

$$\int_0^{t_f} F(x(t),u(t))dt \approx \sum_{i=0}^{N} w_i F(x_N(t_i),u_N(t_i)), w_i = \frac{t_f}{N+1} \tag{15}$$

THIS PAGE INTENTIONALLY LEFT BLANK

II. MODELING THE DYNAMICS AND MUTUAL ATTRITION FUNCTIONS OF A LARGE SCALE SWARM OF AUTONOMOUS SYSTEMS

A. ATTACKING SWARM DYNAMICS MODEL

Two dynamic swarming strategies have been chosen from the related literature for the formation of the attacking swarm, and they are both potential based. The potential function is defined as the sum of an attractive potential, pulling the swarm towards the HVU, and a repulsive potential, pushing the attacking swarm assets away from each other and from the defenders, for collision avoidance purposes. The notion of a virtual body is used in order to move the attacking swarm toward the HVU while keeping some form of cohesion. The virtual body consists of some fictional reference points known as virtual leaders. These reference points initiate forces that control the translational and rotational motion of the swarm and are proportional to the relative distance between each virtual leader and the swarm agent.

1. Virtual Body Artificial Potential

In this model, swarm agents track to a virtual leader (or leaders) inside a neighborhood of interaction, guiding their course while also reacting to intra-swarm forces of collision avoidance and group cohesion. The implemented control input is defined in Equations (16)–(17), and the notations for distances and forces are consistent with Figures 5 and 6.

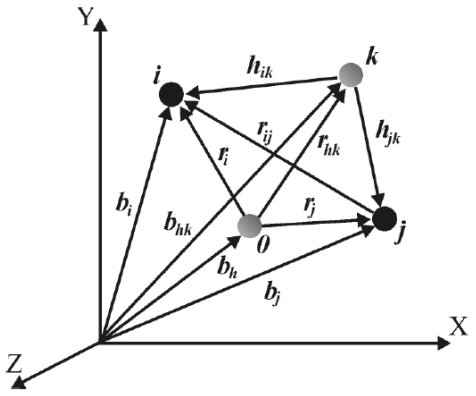

Figure 5. Model framework: Solid circles are vehicles and shaded circles are virtual leaders. Source: [9].

We are going to use Figures 5 and 6 to explain the driving forces of this model. In Figure 5 we have two vehicles, i and j, represented by solid circles, and two virtual leaders, 0 and k, represented by shaded circles. We also have two coordinate systems, an inertial frame with the XYZ axis depicted in Figure 5, and a moving frame with respect to an inertial observer that is stationary relative to the virtual leader, which is depicted as 0. The relative distance between vehicle i and virtual leader k is depicted as h_{ik} and the relative distance between i and j vehicles as r_{ij}. Let us assume that the distance between i attacker and w defender is s_{iw}. Then the control input is as follows:

$$u_i = \ddot{r}_i = -\sum_{j\neq i}^{N_{att}} \nabla_{x_i} V_I(x_{ij}) - \sum_{l=1}^{M_{lead}} \nabla_{x_i} V_h(h_{il}) - \sum_{w=1}^{N_{def}} \nabla_{x_i} V_d(s_{iw}) - F_i^{damp} \tag{16}$$

$$u_i = -\sum_{j\neq i}^{N_{att}} \frac{f_I(x_{ij})}{\|x_{ij}\|} x_{ij} - \sum_{l=1}^{M_{lead}} \frac{f_h(h_{il})}{\|h_{il}\|} h_{il} - \sum_{w=1}^{N_{def}} \frac{f_d(s_{iw})}{\|s_{iw}\|} s_{iw} - K^{damp}\dot{x}_i \tag{17}$$

where $\sum_{j\neq i}^{N_{att}} \frac{f_I(x_{ij})}{\|x_{ij}\|} x_{ij}$ corresponds to the sum of intra-swarm forces that i attacker is accepting from the rest $N_{attackers} - 1$ that are inside the neighborhood of interaction

$x_{ij} \in [0, d_1)$. This force, according to Figure 6, is repulsive when $\|x_{ij}\| \leq d_0$ and attractive when $d_0 < \|x_{ij}\| \leq d_1$.

$\sum_{l=1}^{M_{lead}} \frac{f_h(h_{il})}{\|h_{il}\|} h_{il}$ corresponds to the sum of the virtual leaders' interaction with the i attacker, which again are inside a tuned neighborhood of interaction $h_{il} \in [0, h_1)$. This force is driving the i attacker cohesively with the rest of the attacking swarm toward the HVU. This force, according to Figure 6, is repulsive when $\|h_{il}\| \leq h_0$ and attractive when $h_0 < \|h_{ij}\| \leq h_1$.

$\sum_{w=1}^{N_{def}} \frac{f_d(s_{iw})}{\|s_{iw}\|} s_{iw}$ represents the summation of repulsive forces generated from $N_{defenders}$ toward the i attacker in order to avoid collision with the defense agents inside a defined neighborhood of interaction $s_{iw} \in [0, s_1)$. This is always a repulsive force because it is designed for reactive obstacle avoidance purposes with respect to the swarm agents' sensors. As a result, this force is repulsive when $\|s_{iw}\| \leq s_0$ and has zero magnitude when $\|s_{iw}\| > s_0$. In Figure 23 in page 39, the purely repulsive force due to the defense agents is depicted.

Finally, F_i^{damp} is the velocity dampening term that is used for greater stability.

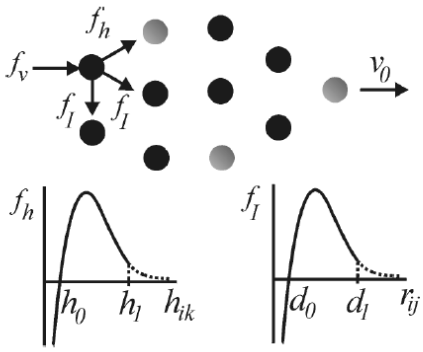

Figure 6. Generated forces in a swarm of eight vehicles and three virtual leaders, due to virtual forces f_h and intra-swarm forces f_I. Source: [9].

In Figure 6 we see how the magnitude of virtual forces f_h and intra-swarm forces f_I is changing with respect to the relative distance h_{il} between the i attacker and the l virtual leader, and the relative distance x_{ij} between the i attacker and the j attacker, accordingly. Namely, we are observing that a repulsive force from f_I is generated when $x_{ij} \in [0, d_0)$, and an attractive is generated when $x_{ij} \in [d_0, d_1]$. Additionally, there is a cutoff relative distance between the i attacker and the j attacker d_1, which is related with the end of the neighborhood of interaction. A relative distance greater than d_1 means that two agents are not interacting with each other. With exactly the same reasoning we can understand how the f_h force is repulsive when $h_{il} \in [0, h_0)$, attractive when $h_{il} \in [h_0, h_1)$, and has no effect for relative distances greater than h_1.

Figure 7 shows that the system of two vehicles and one or two virtual leaders will find an equilibrium stationing where the summation of potentials is minimal. Additionally, we see in two dimensions the advantage of having two virtual leaders instead of one when changing the orientation of the swarm.

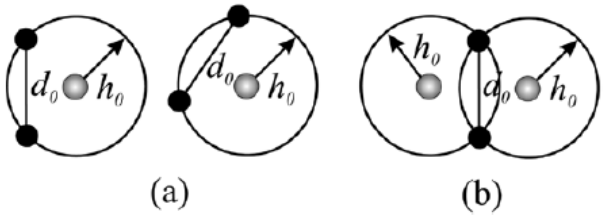

Figure 7. Set of solutions that minimize the total potential in a two dimensional swarm of two vehicles. (a) With one virtual leader there is a family of solutions (two are shown). (b) With two virtual leaders the orientation of the group can be altered appropriately. Source: [10].

2. Reynolds' Rule-Based Model

The original Reynolds model was published in 1987 [11] and it aimed to construct a behavioral animation model for a team of animals, like a swarm of birds, by summing the results of the actions of each individual animal as a reaction of the own local perception of the environment. Hence their flocking formation is not a priori defined; rather, it results from the summation of local rules. In other words, a flock is just the aggregate of the interactions between the behaviors of every individual bird, and if we introduce a computational framework that adds all these individual behaviors for every time step, we could predict a range of activities including path planning.

This research in [11] is not only useful for the simulation and the motion planning of a flock of birds but also for swarms of autonomous systems. The Reynolds' rule-based distributed model that we implement has five competing forces which take into account each time step in order to determine each agent's acceleration. In turn, that will drive its kinetics equation. The control equation is as follows:

$$u_i = \ddot{r}_i = -\sum_{j \neq i}^{N_{att}} (f_{al} + f_{coh} + f_{sep}) - \sum_{l=1}^{M_{lead}} \frac{f_h(h_{il})}{\|h_{il}\|} h_{il} - \sum_{w=1}^{N_{def}} \frac{f_d(s_{iw})}{\|s_{iw}\|} s_{iw} - K^{damp} \dot{x}_i \qquad (18)$$

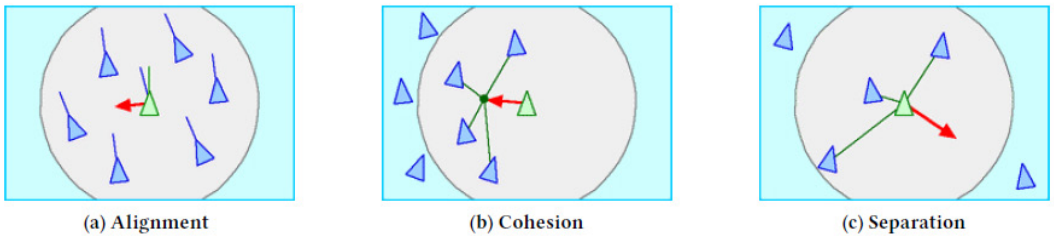

Figure 8. Interaction rules for flocking behavior in Reynolds' Model. Source: [12].

In Equation (18), we observe that each agent updates its acceleration at each time step by considering the following six components:

Alignment: Agents alter their heading in order to align their orientation with the average heading of their neighborhood.

$$f_{al} = -w_{al}\left[\dot{x}_i - \frac{1}{N_{neigh}}\sum_{j\in N_{neigh}} \dot{x}_j\right] \tag{19}$$

Cohesion: Vehicles move toward the centroid of the agents located inside their neighborhood [12].

$$f_{coh} = -w_{coh}\left[x_i - \frac{1}{N_{neigh}}\sum_{j\in N_{neigh}} x_j\right] \tag{20}$$

As we may observe in Figure 8, the robustness of a herd-like formation is dependent on the cohesion and alignment forces, as shown in Equations (19) and (20).

Separation: The assets inside a neighborhood of interaction use a collision avoidance algorithm that guarantees a minimum safety distance.

$$f_{sep} = -w_{sep}\frac{1}{N_{neigh}}\sum_{j\in N_{neigh}} \frac{x_j - x_i}{\left|x_j - x_i\right|} \tag{21}$$

The separation rule prevents crowding and collisions.

Virtual leader(s) interaction: This corresponds to the sum of the virtual leaders' interaction with the i attacker located inside the neighborhood of interaction $h_{il} \in [0, h_1)$. This force is driving the i attacker cohesively with the rest of the attacking swarm toward the HVU.

$$f_{lead} = \sum_{l=1}^{M_{lead}} \frac{f_h(h_{il})}{\|h_{il}\|} h_{il} \tag{22}$$

Collision avoidance with swarm intruders: This represents the summation of repulsive forces generated from $N_{defenders}$ toward the i attacker in order to avoid collision inside the defined neighborhood of interaction $s_{iw} \in [0, s_1)$.

$$f_{\text{int}rud} = \sum_{w=1}^{N_{def}} \frac{f_d(s_{iw})}{\|s_{iw}\|} s_{iw} \tag{23}$$

Velocity dampening: This is a dampening term used for greater stability.

$$F_i^{damp} = K^{damp} \dot{x}_i \tag{24}$$

B. MUTUAL ATTRITION MODEL

In the large scale simulations we execute, there are some couple hundred antagonistic vehicles from each side that are not stationary but are moving toward an objective. Consequently, the mutual attrition model is a very crucial part of our solution to correctly predict the outcome of the confrontation. The concept of mutual attrition is analyzed in [3] where a damage function with specific distribution characteristics (see Figures 9 and 10) is used to track the probability that defender k is destroyed by a shot from attacker l, and vice versa. Figure 9 illustrates how the damage functions are used for observing the inflicted mutual attrition.

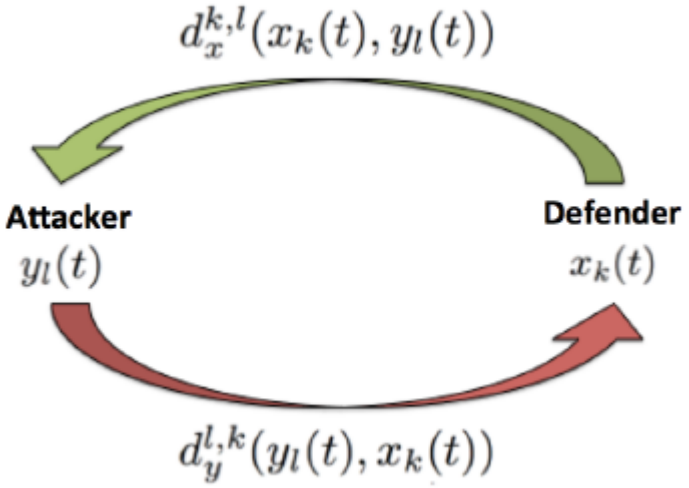

Figure 9. Mutual attrition derived from the damage function. Source: [3].

In [3] the authors examine the parameters we have to take into account in order to model our damage function appropriately. That is because in reality there are weapons that are very successful in their effective range, and their lethality decreases from this range up to the maximum range. Additionally, due to the building architecture of the launching platform, there may be some dead sectors, such as superstructures in a warship, where the platform cannot launch an attack. Moreover, the majority of aerial platforms use missiles that exploit the operating speed of the launching platform in order to reduce in size. Such a missile will not need a booster motor for the initial flight phase but only a sustainer motor for the inertial flight. Consequently, the dimensions of such a missile are significantly reduced and the payload of the aerial vehicle is increased. Hence, in these situations, the field of view (FOV) of the weapon must be mirrored in the death rate functions because these assets can launch their attack only toward specific directions, and this will have a huge impact in the strategy that they follow to approach the enemy. Figure 10 illustrates the shape of the damage function when a Poisson distribution is used or when FOV limitations are applied.

A final consideration for modeling the damage function is the fact that eventually we aim to find numerical solutions in an optimal control problem. Consequently, smooth functions where the gradients are continuous is desired for better numerical performance.

In Figure 11 we see the difference between a maximum range limit damage function versus the smooth and hence preferable Poisson scan model.

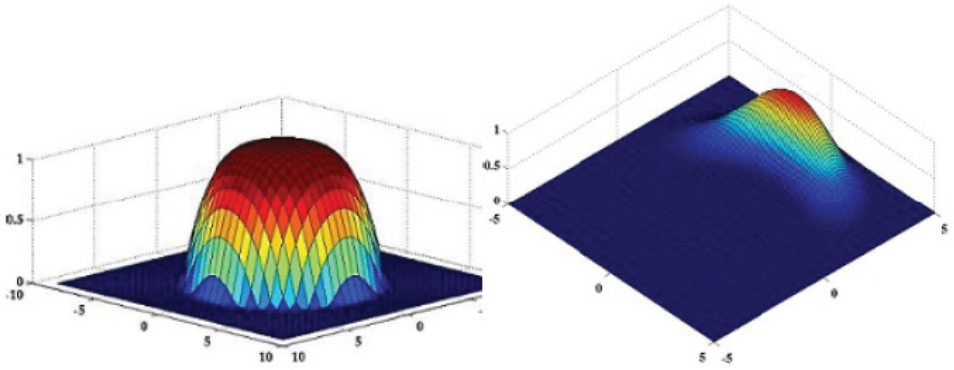

Figure 10. Damage functions: Poisson scan model (left), Resulting angularly decaying function reflecting FOV limitations (right). Source: [3].

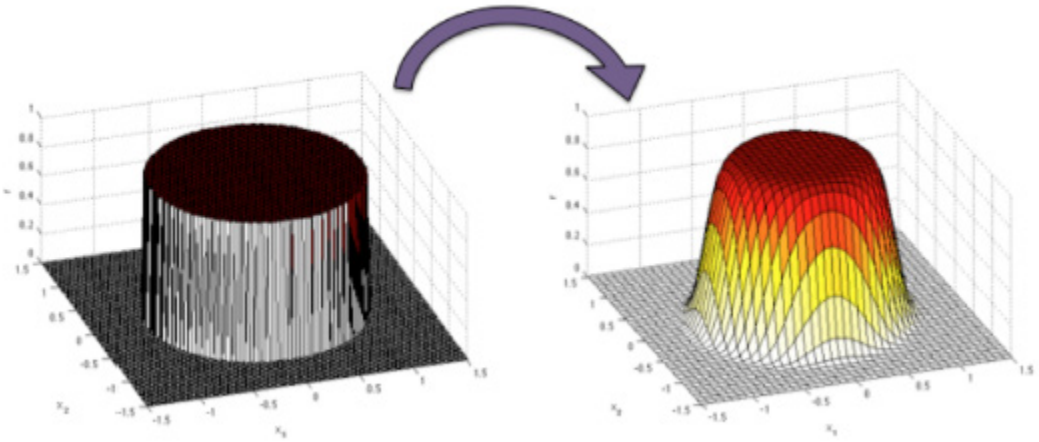

Figure 11. Maximum range limit (Left). Smoothed using the Poisson scan model (Right). Source: [3].

The damage functions of our model follow the Poisson scan model. The attrition dynamics take the following form:

$$d_{ik}^{att} = \lambda_d \Phi_d \left(\frac{F_d - \alpha_d \left[r_i - s_k \right]^2}{\sigma_d} \right) \tag{24}$$

where d_{ik}^{att} is the damage function of the i attacker due to defender k

$$d_{ki}^{def} = \lambda_a \Phi_a \left(\frac{F_a - a_a[r_i - s_k]^2}{\sigma_a}\right), \tag{25}$$

d_{ki}^{def} is the "death rate" of the defender k due to i attacker

$$d_i^{hvu} = \lambda_a \Phi_\alpha \left(\frac{F_a - a_a[r_i - r_{hvu}]^2}{\sigma_a}\right), \tag{26}$$

and d_i^{hvu} is the "death rate" of the HVU due to i attacker.

The parameters λ, F, σ and α are modeling weapons characteristics such as range, fire rate, and inflicted damage and can be manipulated to alter the steepness of the damage function in order to represent correctly the weapons capabilities over distance.

Probability of agent survival can be modeled based on the aggregate number of hits it takes to incapacitate the agent. In this thesis research we use two different systems of ODEs for our two attrition models, the Weighted and the Threshold models.

1. Weighted Attrition Model

In the following system of equations (27)–(29), we compute the probabilities of survival. $Q_i(t)$ is the survival probability of the attacker i, $P_k^d(t)$ is the survival probability of defender k, $P(t)$ is the survival probability of the HVU and d_{ik}^{att} is the damage efficiency of each of the k defenders towards the attacker i.

Consequently, $d_{ik}^{att} P_k^d(t-dt)dt$ corresponds to the probability that defender k, destroys attacker i during a timestep *dt.* Thus $\prod_k^{N_{def}} (1 - \left[d_{ik}^{att} P_k^d(t-dt) \right] dt)$ represents the probability that i attacker would survive during a timestep *dt,* whereas $1 - \prod_k^{N_{def}} (1 - \left[d_{ik}^{att} P_k^d(t-dt) \right] dt)$ is the probability that the i attacker would be destroyed during the timestep *dt.*

This quantity, multiplied by the current survival probability $Q_i(t-dt)$, $Q_i(t-dt)\left[1-\prod_{k}^{N_{def}}(1-\left[d_{ik}^{att}P_k^d(t-dt)\right]dt)\right]$, should be subtracted from your current survival probability $Q_i(t-dt)$, in order to get i attacker survival probability at the next timestep, as follows:

$$Q_i(t)=Q_i(t-dt)\left\{\prod_{k}^{N_{def}}(1-\left[d_{ik}^{att}P_k^d(t-dt)\right]dt)\right\} \tag{27}$$

As we see in Equation (27), the damage efficiency d_{ik}^{att} of each of the k defenders towards the attacker i is weighted by each defender probability of survival. Hence a defender with low probability of survival, has small contribution to the attrition of attacker i.

With the same reasoning as above, we may derive the survival probability of defender k at time t, $P_k^d(t)$:

$$P_k^d(t)=P_k^d(t-dt)\left\{\prod_{i}^{N_{att}}(1-\left[d_{ki}^{def}Q_i(t-dt)\right]dt)\right\} \tag{28}$$

As we see in Equation (28), the damage efficiency d_{ki}^{def} of each of the $N_{attackers}$ towards the defender k is weighted by each attacker probability of survival. Hence an attacker with low probability of survival, has small contribution to the attrition of defender k.

$$P(t)=P(t-dt)\left\{\prod_{k}^{N_{att}}(1-\left[d_{k}^{hvu}Q_k(t-dt)\right]dt)\right\} \tag{29}$$

In Equation (29) we track the probability of survival of HVU based on the attrition function of all the attackers. Again we tax their destructive capabilities with their own probability of survival.

Initial conditions are set as $Q_i(0)=P(0)=P_k^d(0)=1,\forall i,k$.

2. Attrition Model with Thresholds on Survival Probabilities

This is a model introduced for the first time in this thesis, and its usefulness is shown in Chapter III, where we introduce the new proposed models. In this model we use the following system of equations:

$$Q_i(t) = Q_i(t-dt)\left\{\prod_k^{N_{def}}(1-d_{ik}^{att}dt)\right\}, \forall P_k^d(t) \geq 50\% \tag{30}$$

$$P_k^d(t) = P_k^d(t-dt)\left\{\prod_i^{N_{att}}(1-d_{ki}^{def}dt)\right\}, \forall Q_i(t) \geq 50\% \tag{31}$$

$$P(t) = P(t-dt)\left\{\prod_k^{N_{att}}(1-d_k^{hvu}dt)\right\}, \forall Q_i(t) \geq 50\% \tag{32}$$

Consequently, the only difference with respect to the previous model is that now we do not tax the damage functions with the probability of survival of each asset. Nevertheless, as soon as someone's survival probability drops below 50%, we assume that this asset is more likely to have been incapacitated, and as a result, it will not contribute to the destruction of its adversaries.

C. EVALUATING THE PERFORMANCE OF THE TRAJECTORY OPTIMIZATION ALGORITHM COMPARED WITH INTUITION CONCERNING THE STATIONING OF THE DEFENDING VEHICLES

Now that we have introduced the fundamental parts of our algorithm, we want to show the efficiency of our method in terms of cost measurements. Our problem is constrained in such a way that the comparison of a single number, the scalar cost, can indicate whether it is advantageous to use the generated trajectories or not.

In Figures 12 and 13 we have at our disposal a force of 25 defenders tasked to protect an HVU against a swarm of 100 attackers. Although outnumbered, the defenders have a 50% larger weapons range as well as double the fire rate with respect to the attackers. This range and fire rate advantage is what our optimization framework exploits in order to define the defenders motion planning. In Figures 12 and 13 we initially

identify the trajectories of the attackers with the red color and the trajectories of the defenders with a cyan color. As the scenario is executed, however, we color code the followed paths with the survival probability of each agent. As a result, we may observe spatially the mutual attrition of the antagonistic agents. In both scenarios the defenders are stronger and suffer fewer losses; however, in the unoptimized scenario, some of the attacking agents manage to penetrate the defenders' zone and destroy the HVU.

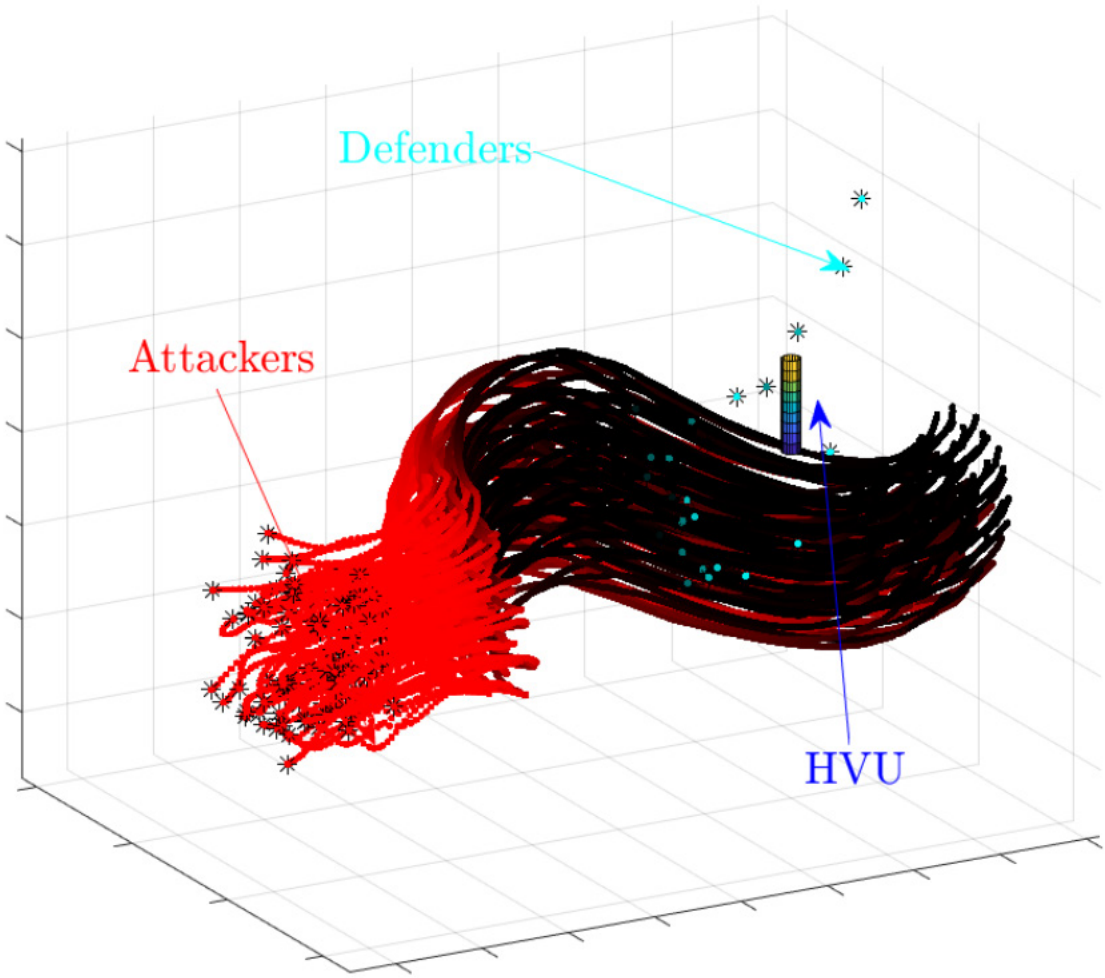

Figure 12. 25 defenders with unoptimized trajectories but with superior weapons failing to protect the HVU from 100 attackers.

In Figure 14 we compare the outcome of the optimized versus the unoptimized scenario in terms of survival probabilities. According to Figure 14, the defenders with the

optimized trajectories successfully protect the HVU, whereas in the unoptimized scenario they fail. Moreover, in the optimized scenario we observe that the attacking swarm is completely incapacitated by the end of the first quarter of the simulation. By contrast, in the unoptimized scenario we see that they manage to maintain a low survival probability by the end of the scenario. Last but not least, in the optimized scenario we observe higher mean survival probability for the defenders. Although this is not our primary mission objective because we consider the defending agents as dispensable, it is always desirable to minimize the attrition that our forces are going to suffer.

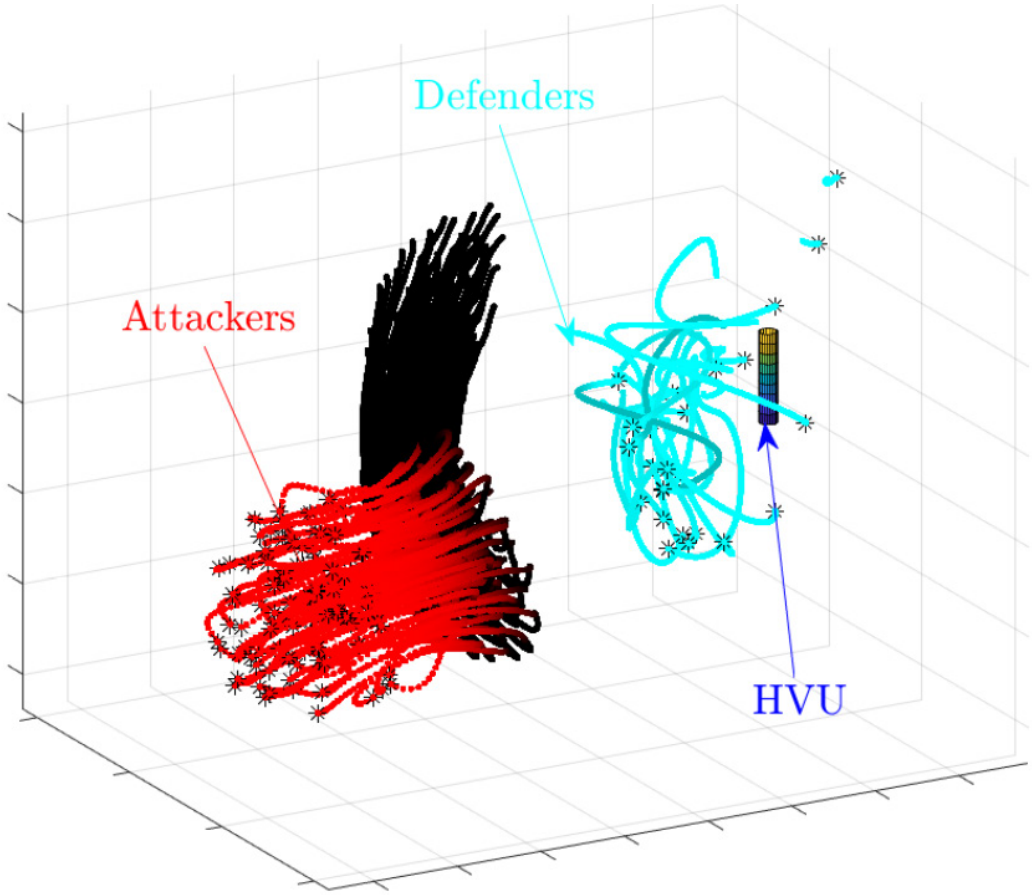

Figure 13. 25 defenders with optimized trajectories and superior weapons protect the HVU effectively from 100 attackers.

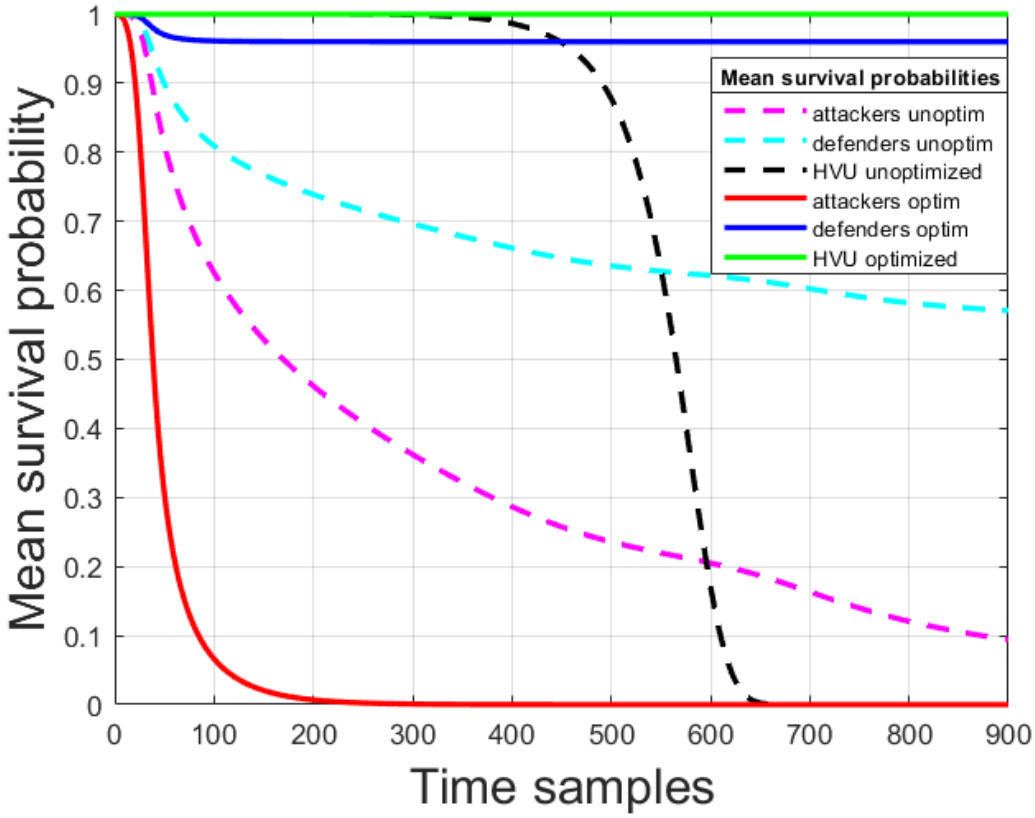

Figure 14. Comparison of survival probabilities of the optimized and unoptimized scenarios of the 100 attackers versus the 25 stronger defenders.

In Figure 15 we present a collection of some characteristic snapshots of the confrontations that we just analyzed with optimized trajectories for the defenders. Agents with reduced survival probability start to fade until they become completely invisible for 0% survival probability.

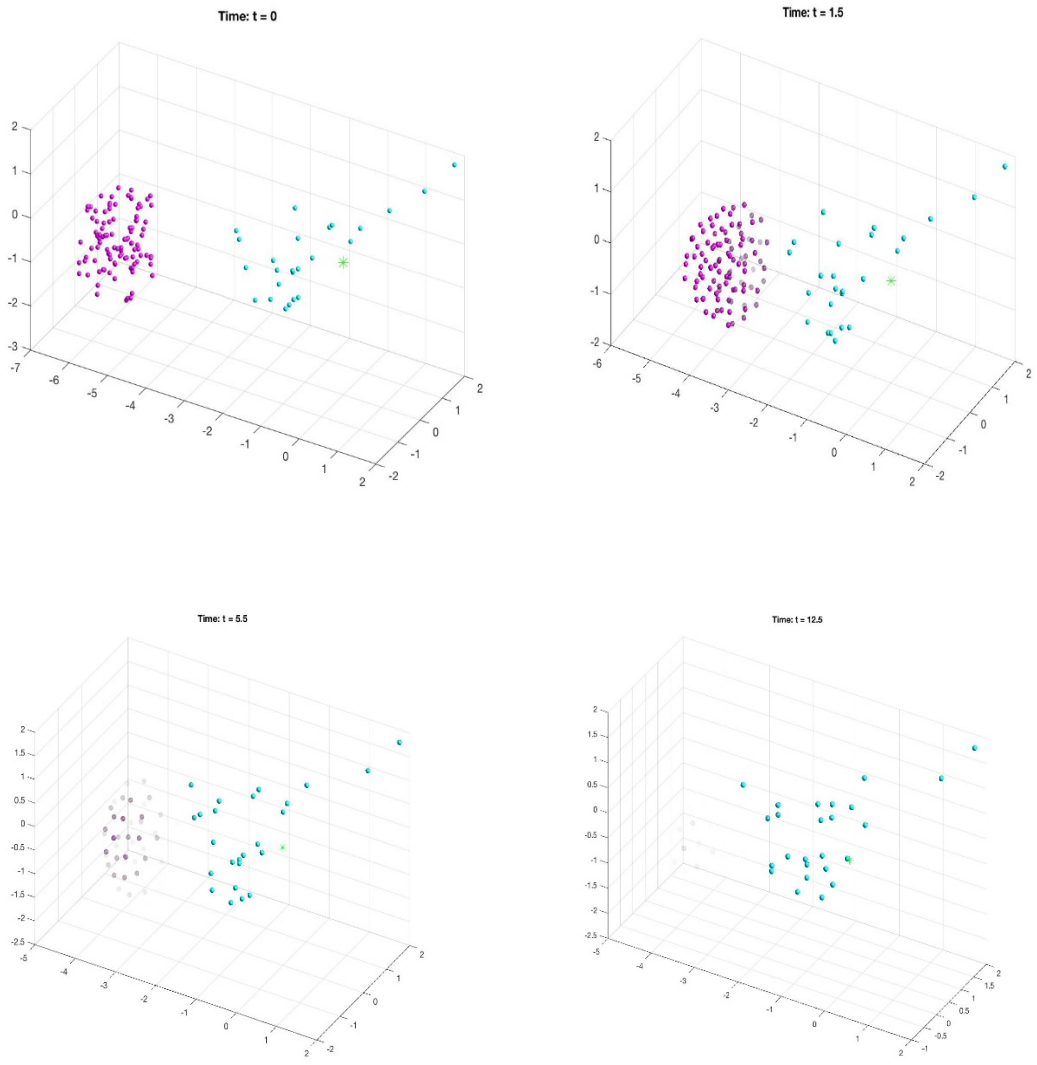

Figure 15. Snapshots from the optimized scenario of the 100 attackers versus the 25 stronger defenders.

D. CONTRIBUTION OF MOLECULAR DYNAMICS ALGORITHMS TO THE COMPUTATIONAL EFFECTIVENESS OF OUR FRAMEWORK

We can easily observe that the modeling of the dynamics of the attacking swarm in order to calculate its equations of motion is the most computationally expensive part of the algorithm. All these antagonistic forces that we detailed in section A of the current chapter contribute to the aggregate acceleration of each individual asset and must be accurately calculated for each time step and for each asset. It is proven that a simple Euler integration scheme is ineffective for accurately tracking the dynamics of such

sophisticated systems. Consequently, a fourth order Runge Kutta integration method with fixed time-step size was originally used. This integration method, although accurate in the calculation of the dynamics and the equations of motion of the attacking swarm, lacks computational effectiveness.

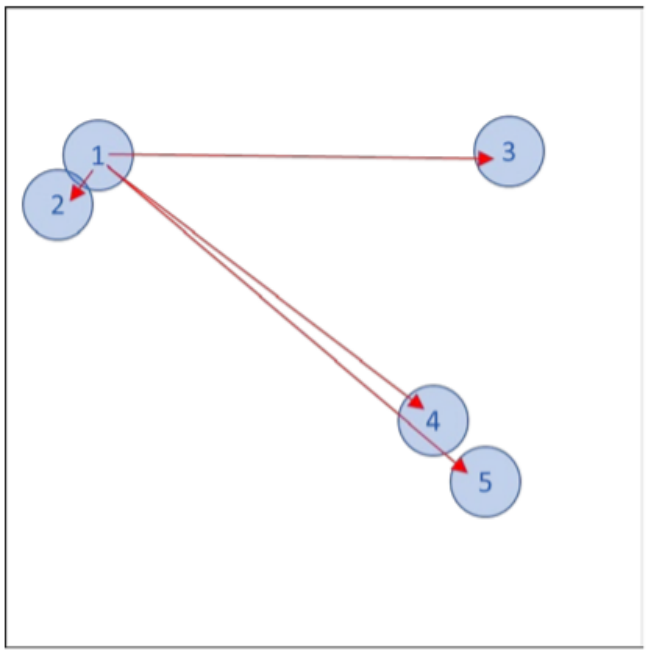

Figure 16. Molecular dynamics demonstration compares particle 1 to particles 2, 3, 4, and 5 to determine whether their inflicted interaction is affecting its trajectory. Source: [13].

Molecular dynamics (MD) is a fascinating domain in physics, where the motion of molecules is simulated based on the generated forces due to interatomic potentials. These forces define the molecular trajectories by simply applying Newton's second law. As we see in [13], MD is used to predict the damage inflicted from a ballistic impact on a soil target. As we may easily understand, during the impact, large non-linear forces are exerted, and as a result, the existing analytical models will not be able to accurately track the interaction forces for at least this important initial stage.

As a result, the MD community has adopted numerical methods to execute simulations that track the forces between the soil particles and then use the Verlet

integration algorithms to compute the velocity and the position of each particle. Verlet integration is widely used in this community because this algorithm provides accurate results with much better performance in comparison to Runge-Kutta integration.

The reason for this computational advantage is that Runge-Kutta has to calculate the forces four times every time-step, whereas Verlet only once. In fact, LAMMPS, the MD code that is primarily used for research nowadays, uses exclusively Verlet integration [14].

Additionally, we could easily see the similarity in the nature of these contact forces with the interaction forces that come into play in our counter-swarm simulations. This observation allowed us to adopt their computationally efficient techniques. As a result, a faster and more robust model was adopted that allowed us to execute simulations with an unprecedented number of agents.

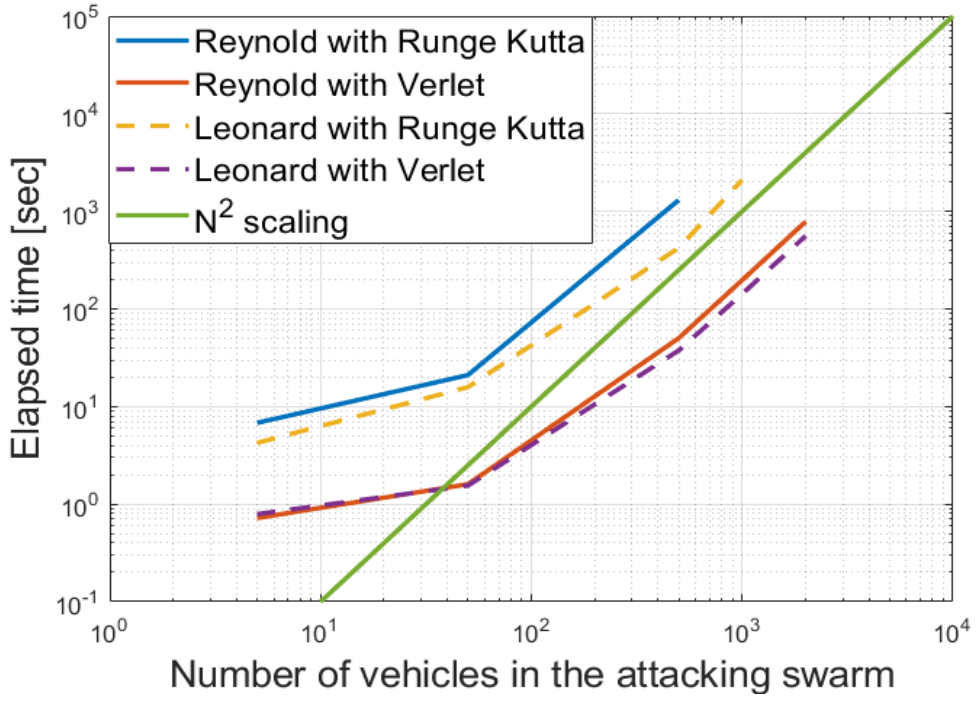

Figure 17. Computational superiority of Verlet versus Runge Kutta integration framework and comparison with $O(N^2)$ logarithmic scaling.

The velocity Verlet algorithm for computing the velocity and position of each particle from the equation of motion is shown as:

$$r_i(t+dt) = r_i(t) + v_i(t)dt + \frac{1}{2}a_i(t)dt^2 \tag{33}$$

$$v_i(t+dt) = v_i(t) + \frac{1}{2}\left[a_i(t) + a_i(t+dt)\right]dt \tag{34}$$

In Figure 13 and Tables 1 and 2 we observe a logarithmic comparison of the originally implemented Runge Kutta integration scheme versus the Verlet integration algorithm, which basically shows us that we have obtained an approximately 14.7 times faster code for Leonard dynamics and a **26.01 times faster** code for Reynolds dynamics.

Table 1. Computational effectiveness comparison of the Virtual Body Artificial Potential dynamics model with Verlet integration versus original model with Runge Kutta integration.

# of agents in the attacking swarm	**Elapsed time [sec] (Runge Kutta)**	**Elapsed time [sec] (Verlet)**	**Computational gain based on elapsed time division**
5	4.27	0.79	5.4
50	15.8	1.55	10.2
500	421.78	37.55	11.23
1000	2095	142.56	**14.7**
2000	-	559.7	

As mentioned previously, both our optimal motion planning problem and the MD problem use behavioral rules for computing potentials generated from interactions. Consequently, a significant computational advantage was gained by integrating this scheme into our problem formulation.

Additionally, other parts of the code were improved from the aspect of efficiency. Namely, the improved algorithm checks the neighbors interactions only once and this check is taken into account for both agents. Moreover, unnecessary computations like in the occasion where two agents are outside the neighborhood of interactions are avoided. Last but not least, MATLAB commands that are proven to be computationally expensive where replaced with faster scripts.

Table 2. Computational effectiveness comparison of the Reynolds dynamics model with Verlet integration versus original model with Runge Kutta integration.

# of agents in the attacking swarm	**Elapsed time [sec] (Runge Kutta)**	**Elapsed time [sec] (Verlet)**	**Computational gain based on elapsed time division**
5	6.81	0.72	9.46
50	21.01	1.6	13.13
500	1304.84	50.16	**26.01**
1000	-	197.4	-
2000	-	783.76	-

III. THE GHOST-HERDING PROBLEM AND THE PROPOSED INTERACTION AND ATTRITION MODELS

A. THE GHOST-HERDING PROBLEM — GENERATION OF NON-PHYSICAL SOLUTIONS

We have defined our problem in such a way that the Optimization Toolbox will generate the optimal trajectories for the defenders according to the mission objectives that are incorporated into the cost function, as discussed in Chapter I. Namely, we have accounted for two mission objectives, the protection of an HVU and the acquisition of air superiority by maximizing the adversary swarm attrition. Nevertheless, we have to constrain the optimization problem properly such that we avoid unrealistic solutions.

In Figures 18–22 we present an analysis that we have made based on the optimization results from a large scale simulation where 200 defenders protect the HVU from an attacking swarm of 2066 agents, which is the world's largest unmanned aerial vehicle (UAV) swarm that has ever been airborne [14]. In Figure 18 we see that the solution we predict is very optimistic. Namely, the survival probability of the HVU is as high as 96.3%.

Yet, in Figures 19 and 20 we see that half of the defenders die in the first quarter of the simulation. Moreover, the remaining defenders die by the end of the simulation. By contrast, as far as the attacking swarm is concerned, we observe some attrition during the first quarter of the scenario, but their mean survival probability stabilizes at 90% thereafter. Finally, we see in Figure 19 that by the end of the scenario all defenders are dead and about 1870 attackers are still operational.

So far we have applied some of the most widely known dynamics and attrition models that have been published (see List of References and Chapter II). With that in mind, we see that the swarm of the 2066 assets is approaching cohesively toward the HVU, making a collision avoidance maneuver in front of the defenders that are following the trajectories the optimal motion planning algorithm has computed.

Although from a mathematical point of view, the solver has found a correct solution for the mission of HVU protection, this local minimum has no physical meaning. The conducted analysis shows us that the Optimization Toolbox uses defenders that are already dead as a shield for the protection of the HVU.

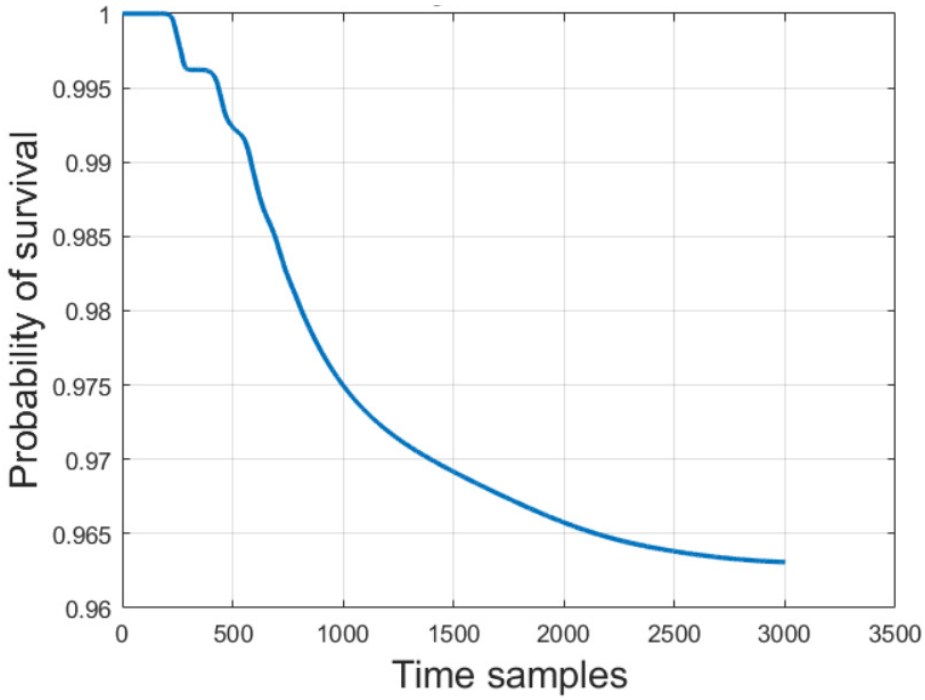

Figure 18. HVU survival probability during a 3000 time sample simulation.

In other words, defender trajectories are chosen such that the collision avoidance algorithm will create repulsive forces for the attackers. Namely, the dead defenders create a repulsive barrier that has no real meaning. In Figures 19 and 20 we observe that although for most of the simulation all the defenders are dead and no further attrition occurs between attackers and defenders, the attackers do not succeed in penetrating the barrier that the dead defenders are applying.

In Figure 22 we have chosen snapshots of the scenario for a ghost-herding problem demonstration. We see the positions of all the agents in the scenario according to their respective color codes. In the beginning of the scenario, when the opposing agents are at a distance with respect to their weapons capabilities, the attackers are magenta

colored whereas the defenders are shown in cyan, as their survival probability is close to 100%.

As the scenario advances, the attacking swarm approaches the HVU, but the defenders intercept their path. At that time, the color coding indicates the attrition of the assets, where black signifies 0% survival probability and the in-between colors represent how likely an asset is to be operational.

According to Figure 22, we see that the ghost-herding phenomenon starts to appear after the first 600 time samples from the beginning of the scenario consisting of 3000 total time samples.

Referring to Figure 21 we see the ratios of mean survival probabilities as $\frac{Def}{Att}$ and $\frac{Att}{Def}$. It is clear that since the opposing forces have the same weapons characteristics, the attrition of the defenders due to multiple attackers would be much more significant.

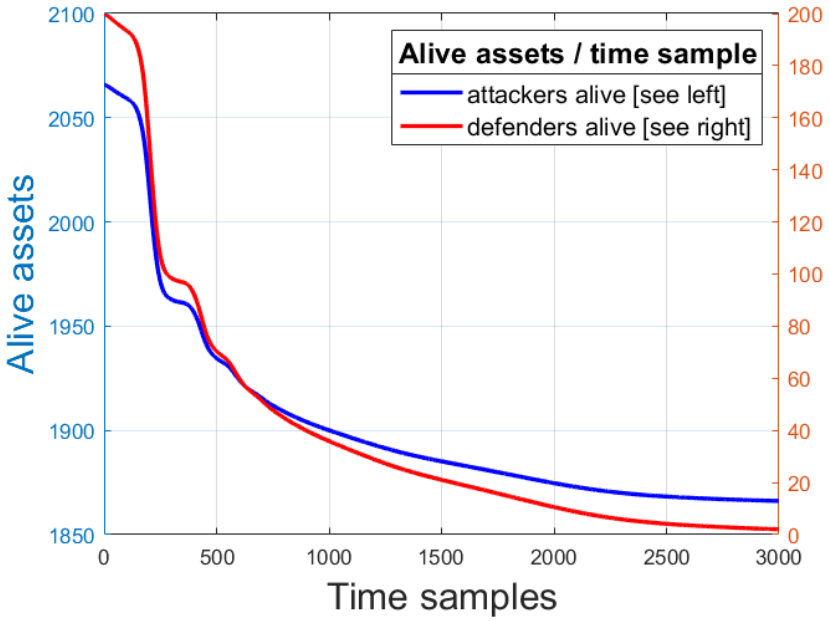

Figure 19. Total number of live defenders and attackers for the ghost-herding scenario of an attacking swarm of 2066 agents versus a defending force of 200 agents with the same kinetics and weapons capabilities.

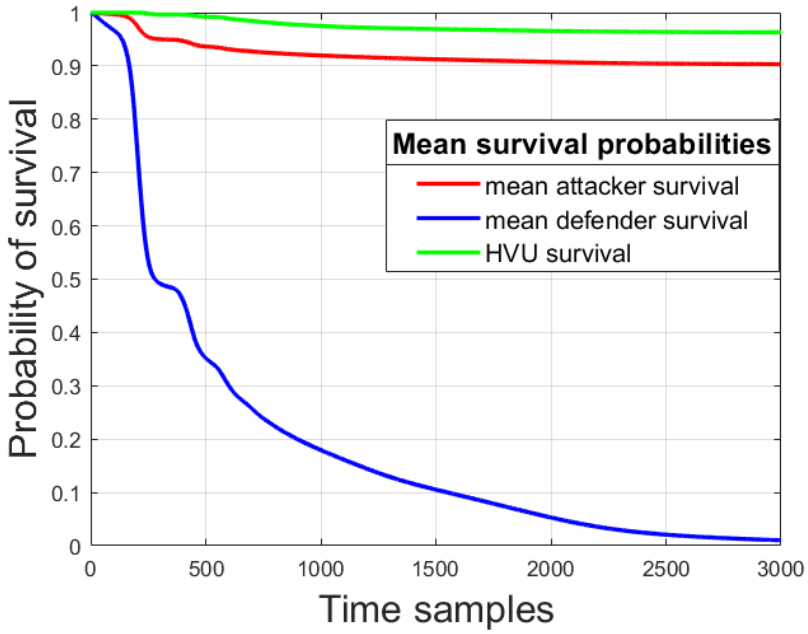

Figure 20. Mean survival probability for defenders, attackers, and HVU for the ghost-herding scenario of 2066 attackers versus 200 defenders.

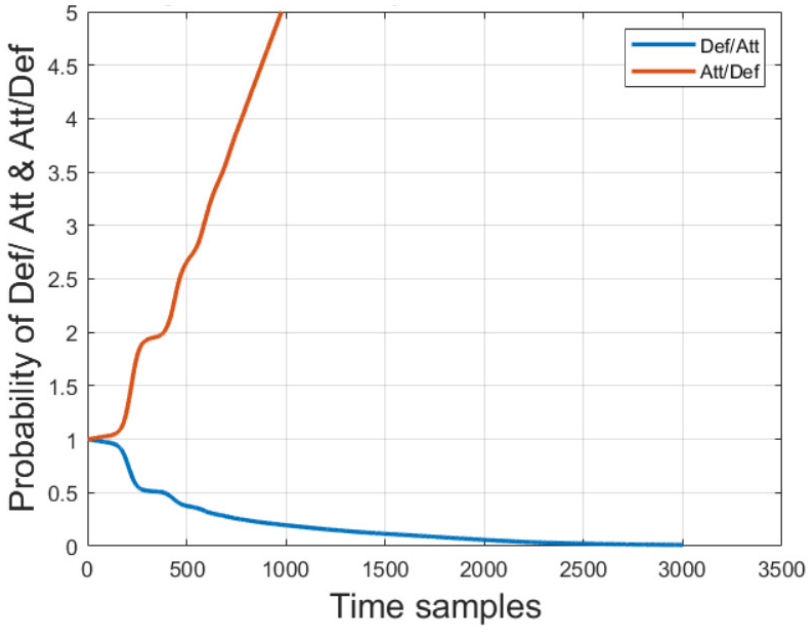

Figure 21. Ratios of mean survival probabilities $\frac{Def}{Att}$ and $\frac{Att}{Def}$ for the ghost-herding scenario of 2066 attackers versus 200 defenders.

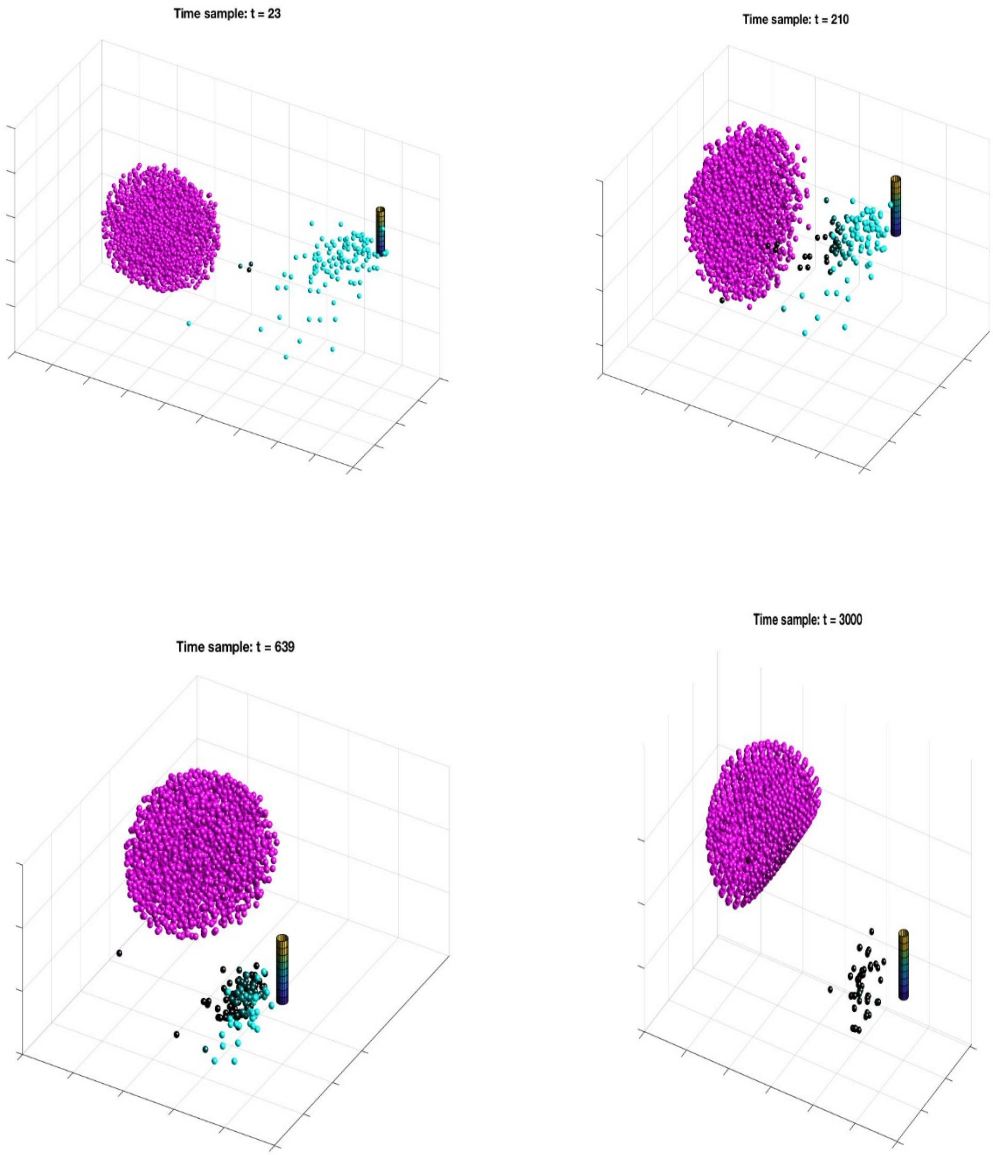

Figure 22. Snapshots of the ghost-herding scenario of 2066 attackers (Guinness world record) versus 200 defenders at time samples: 23, 210, 639 and 3000.

In both Figure 23 and in Figure 6 we see how steep the slope of the repulsive force is when an attacker is approaching a defender closer than the minimum distance. This minimum distance and the overwhelming generated repulsive force have, of course, a deterministic nature. They are associated with the collision avoidance algorithm of the attacking swarm but they are certainly an important parameter of the simulation because

they associate the deterministic nature of the exerted forces with the probabilistic approach of the whole scenario.

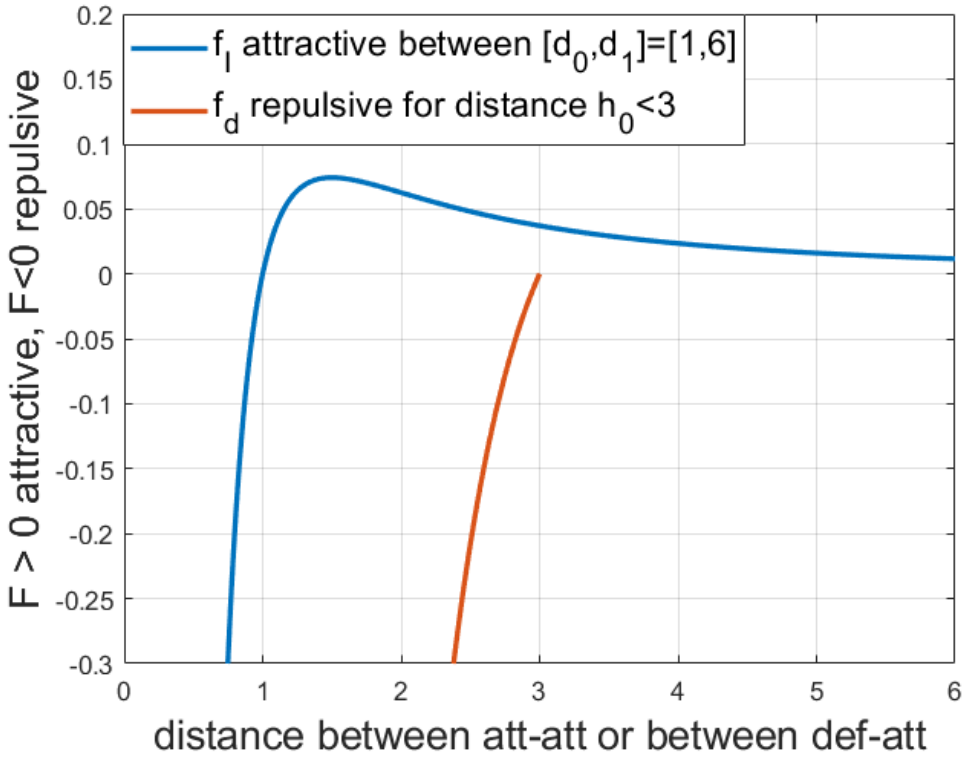

Figure 23. Magnitude of the intra-swarm forces and collision avoidance forces, due to defenders, with respect to the relative distance.

B. PROPOSED INTERACTION AND ATTRITION MODELS FOR OPTIMIZATION

To deal with the ghost-herding problem just described, we created models that correlate the deterministic nature of the dynamics with the survival probabilities of the attacking swarm, the defender forces, and the HVU.

Consequently, we created three respective models, two of them for optimization purposes and one for verification of the correctness of our results. For optimization purposes we introduce the models that we call:

- Weighted Dynamics and Attrition Model
- Threshold Dynamics and Attrition Model

For analysis and verification of the results we introduce the model that we have named:

- Monte Carlo Dynamics and Attrition Model

1. Dynamics and Attrition Model "Weighted" with Survival Probabilities

The weighted model correlates both the magnitude of the interaction forces and the lethality of the weapons of each asset with their survivability. As a result, the gradient-based optimizer that we are using can no longer employ defenders that are already incapacitated. Furthermore, it also cannot use assets with low probability of survival because their contribution to the attacking swarm attrition is very small.

a. Weighted Dynamics Model

The driving equations of the dynamics models detailed in Chapter II take the following form:

(1) Virtual Body Artificial Potential

$$u_i = \ddot{r}_i = -\left\{ \sum_{l=1}^{M_{lead}} \nabla_{x_i} V_h(h_{il}) + F_i^{damp} \right\} - \sum_{j \neq i}^{N_{att}} Q_j \nabla_{x_i} V_I(x_{ij}) - \sum_{k=1}^{N_{def}} \left[P_k^d \nabla_{x_i} V_d(s_{ik}) \right] \tag{35}$$

$$u_i = -\left\{ \sum_{l=1}^{M_{lead}} \frac{f_h(h_{il})}{\|h_{il}\|} h_{il} + K^{damp} \dot{x}_i \right\} - \sum_{j \neq i}^{N_{att}} Q_j \frac{f_I(x_{ij})}{\|x_{ij}\|} x_{ij} - \sum_{k=1}^{N_{def}} \left[P_k^d \frac{f_d(s_{ik})}{\|s_{ik}\|} s_{ik} \right] \tag{36}$$

As we see in Equations (35) and (36), our control law for the i attacker still consists of the intra-swarm forces, the virtual leader interaction, the repulsive forces from the defenders, and the dampening term. However, we now weight these forces with the survival probability of the agent that is the cause of this interaction.

The sum of intra-swarm forces that i attacker is accepting from the rest $N_{attackers}-1$ that are inside the neighborhood of interaction $x_{ij} \in [0, d_1)$ is taxed by the survival probability of each attacker j, respectively.

The repulsive forces generated from $N_{defenders}$ toward the i attacker in order to avoid collision with the defense agents inside the neighborhood of interaction $s_{iw} \in [0, s_1)$ are taxed with the survival probability of each individual defender.

(2) Reynold's Rule-based Model

$$u_i = \ddot{r}_i = -\left\{ \sum_{l=1}^{M_{lead}} \frac{f_h(h_{il})}{\|h_{il}\|} h_{il} + K^{damp} \dot{x}_i \right\} - \sum_{j \neq i}^{N_{att}} Q_j (f_{al} + f_{coh} + f_{sep}) - \sum_{k=1}^{N_{def}} \left[P_k^d \frac{f_d(s_{iw})}{\|s_{iw}\|} s_{iw} \right] \quad (37)$$

In Equation (37) we see the modified control law that we propose for the i attacker defined by the six forces we analyzed in Chapter II for the Reynold's dynamics model. Again the alignment, cohesion, and separation forces are weighted from the probability of the corresponding swarm agent that causes the interaction. Finally, defender forces are stronger when the defenders have high survival probabilities.

b. Weighted Attrition Model

For reciprocal attrition, in this case we use the model presented in [3].

2. Dynamics and Attrition Models Correlated with a Survival Probability Cutoff "Threshold"

The weighted model just described is able to tackle the problem of unrealistic solutions due to the fact that dead defenders are acting as a repulsive barrier. One could argue, however, that although it is an improved version, it is also a non-physical model because, and especially for aerial assets, you would either expect them to be fully functional and operational or completely destroyed and no longer operational. Further, you might predict that an asset, due to its proximity with opposing forces and due to the range and lethality of their weapons, would have only a 15% probability of being operational. In other words, this would mean that if you run the same scenario 100 times,

you would expect this asset to be operational at this specific time instant only in 15 iterations. It would, however, be either fully operational or completely ineffective, and this statement is what you should expect for both its interaction forces and its weapons.

This very reasonable statement was our driver for creating the threshold model, in which we do not weight the probability of survival with the interaction forces and the weapons effectiveness at all. Instead, we are using the 50% survival probability as a cutoff switch. This threshold and the survival probability of an agent define whether the model will take into account its interaction and weapons contribution.

a. Threshold Dynamics Model

Having introduced the rules of this model, we now give the driving forces of the control law as follows:

(1) Virtual Body Artificial Potential

$$u_i = -\sum_{j\neq i}^{N_{att}}\left[W_j^{\mathrm{I}}\frac{f_I(x_{ij})}{\|x_{ij}\|}x_{ij}\right] - \sum_{k=1}^{N_{def}}\left[W_k^d\frac{f_d(s_{ik})}{\|s_{ik}\|}s_{ik}\right] - \left[\sum_{l=1}^{M_{lead}}W_l^h\frac{f_h(h_{il})}{\|h_{il}\|}h_{il}\right] - K^{damp}\dot{x}_i$$

$$\left\{\begin{array}{l} W_j^I = 1\forall Q_j \in [0.5,1.0], W_j^I = 0\forall Q_j \in [0,0.5) \\ W_k^d = 1\forall P_k^d \in [0.5,1.0], W_k^d = 0\forall P_k^d \in [0,0.5) \\ W_l^h = 1\forall Q_i \in [0.5,1.0], W_l^h = 0\forall Q_i \in [0,0.5) \end{array}\right\} \tag{38}$$

The probability acts as a switch that neutralizes an asset instantaneously when its survival probability falls below 50 %. Namely, as soon as the destruction is more probable than the survival, we neutralize the asset.

(2) Reynold's Rule-based Model

$$u_i = \ddot{r}_i = -\sum_{j\neq i}^{N_{att}}\left[W_j^I\left(f_{al}+f_{coh}+f_{sep}\right)\right] - \sum_{k=1}^{N_{def}}\left[W_k^d\frac{f_d(s_{ik})}{\|s_{ik}\|}s_{ik}\right] - \left[\sum_{l=1}^{M_{lead}}W_l^h\frac{f_h(h_{il})}{\|h_{il}\|}h_{il}\right] - K^{damp}\dot{x}_i$$

$$\left\{\begin{array}{l} W_j^I = 1\forall Q_j \in [0.5,1.0], W_j^I = 0\forall Q_j \in [0,0.5) \\ W_k^d = 1\forall P_k^d \in [0.5,1.0], W_k^d = 0\forall P_k^d \in [0,0.5) \\ W_l^h = 1\forall Q_i \in [0.5,1.0], W_l^h = 0\forall Q_i \in [0,0.5) \end{array}\right\} \tag{39}$$

The same notion of a cutoff switch is applied in the Reynold's dynamics model as well.

b. Threshold Attrition Model

For reciprocal attrition we use the model introduced in Chapter II.

C. MONTE CARLO SIMULATION MODEL FOR ANALYSIS

Finally, in our last model, the Monte Carlo model, the fundamental difference is that we replace the survival probability approach for the defenders and the attackers with a binary state of being alive and operational or dead and out of the simulation. Although we can still use this approach for the HVU as well, we keep the previous probabilistic scheme for this unit. For better results during optimization, we would like a relatively smooth probabilistic approach for the HVU rather than an unpredictable approach.

a. Monte Carlo Attrition Model

The binary survival probabilities are updated via random number generation at each time step in the following way. The probability p_i^{att} for the i attacker to die due to all defenders k during a given time step with duration *dt* is given by:

$$p_i^{att}(t) = 1 - \prod_{k=1}^{N_{def}} (1 - d_{ik}^{att} W_k^{def}(t - dt)dt) \tag{43}$$

Similarly, the probability p_k^{def} for the defender k to die due to all attackers i during a given time step with duration dt is given by:

$$p_k^{def}(t) = 1 - \prod_{i}^{N_{att}} (1 - d_{ki}^{def} W_i^{att}(t - dt)dt) \tag{44}$$

Then, a random number X_i or Y_k is generated with a uniform distribution between 0 and 1 for each attacker i and defender k, respectively. We change W_i^{att} and W_k^{def} that now represent the binary state of the i attacker and the k defender, respectively, from 1 to 0, from alive to dead, when these random numbers are smaller

than the death probability. Otherwise, when these random numbers are bigger than the death probability, it would mean that this agent will survive the current time step and will be queried again during the next iteration. The mathematical statement is as follows:

$$X_i(t) < p_i^{att}(t) \rightarrow W_i^{att}(t) = 0 \tag{45}$$

$$X_i(t) \geq p_i^{att}(t) \rightarrow W_i^{att}(t) = W_i^{att}(t - dt) \tag{46}$$

$$Y_k(t) < p_k^{def}(t) \rightarrow W_k^{def}(t) = 0 \tag{47}$$

$$Y_k(t) \geq p_k^{def}(t) \rightarrow W_k^{def}(t) = W_k^{def}(t - dt) \tag{48}$$

The HVU survival probability could be calculated in this way as well, but for optimization we use a continuous approach, where P(t) is initialized with P(0) = 1 and then integrated in the same way as shown in the weighted attrition model in Chapter II and in [3] and [15]. Namely:

$$P(t) = P(t - dt)\left\{1 - \left[1 - \prod_{k}^{N_{att}} (1 - \left[d_k^{hvu} W_k^{att}(t - dt)\right] dt)\right]\right\} \tag{49}$$

The only difference with Equation (49) is that now W_k^{att} is the binary state of the i attacker. Consequently, only the live attackers contribute to the attrition of the HVU.

b. Monte Carlo Dynamics Model

Our model is based on random number generation that aims to simulate the unpredictability often exhibited in the real world. We are using a dynamics framework similar to the one analyzed in Chapter II, but because the survival probabilities have been replaced from the binary state condition, we have the following equations for the control law of the i attacker:

(1) Virtual Body Artificial Potential

$$u_i = \ddot{r}_i = -W_i^{att}\left\{\sum_{l=1}^{M_{lead}} \nabla_{x_i} V_h(h_{il}) + F_i^{damp} + \sum_{j \neq i}^{N_{att}} W_j^{att} \nabla_{x_i} V_I(x_{ij}) + \sum_{k=1}^{N_{def}} \left[W_k^{def} \nabla_{x_i} V_d(s_{ik})\right]\right\} \tag{40}$$

$$u_i = -W_i^{att}\left\{\sum_{l=1}^{M_{lead}}\left[\frac{f_h(h_{il})}{\|h_{il}\|}h_{il}\right] + K^{damp}\dot{x}_i + \sum_{j\neq i}^{N_{att}}\left[W_j^{att}\frac{f_1(x_{ij})}{\|x_{ij}\|}x_{ij}\right] + \sum_{k=1}^{N_{def}}\left[W_k^{def}\frac{f_d(s_{ik})}{\|s_{ik}\|}s_{ik}\right]\right\} \quad (41)$$

where $W_i^{att}(t), W_k^{def}(t) \in \{0,1\}$. Hence, for an interaction between attackers or between attacker and defender to be exerted, both agents must be stated as alive.

(2) Reynold's Rule-based Model

$$u_i = \ddot{r}_i = -W_i^{att}\left\{\sum_{l=1}^{M_{lead}}\left[\frac{f_h(h_{il})}{\|h_{il}\|}h_{il}\right] + K^{damp}\dot{x}_i + \sum_{j\neq i}^{N_{att}}\left[W_j^{att}(f_{al}+f_{coh}+f_{sep})\right] + \sum_{k=1}^{N_{def}}\left[W_k^{def}\frac{f_d(s_{ik})}{\|s_{ik}\|}s_{ik}\right]\right\} \quad (42)$$

where $W_i^{att}(t), W_k^{def}(t) \in \{0,1\}$. Hence, for an interaction between attackers or between attacker and defender to be exerted, both agents must be stated as alive.

IV. OPTIMIZATION RESULTS AND ANALYSIS WITH MISSION OBJECTIVE TO MINIMIZE HVU DESTRUCTION PROBABILITY

A. LOCAL AND GLOBAL MINIMUM SOLUTIONS FOR OPTIMIZATION PROBLEMS SPANNING A CONFIGURATION SPACE OF INFINITE DIMENSIONS

The main idea for our optimal motion planning problem is to use an objective function, in this chapter the HVU survival probability, and by computing its gradients we seek to find the ideal defender trajectories.

At this point we have to underline the importance of the initial estimate. The gradient based algorithm will drive the solution towards the direction of the gradient's steepest descent and it will stop when one or more of the following conditions are met:

- a local minimum is found
- a sufficiently flat area, within some tolerance defined in advance, of the configuration space is found
- the objective goal is met
- the maximum number of evaluation iterations is reached

Consequently the initial estimate is important to be an educated guess, based on our intuition or based on Operational Research techniques rather than a random guess, in order to drive the algorithm faster to a local minimum. When a local minimum is achieved, it will be analyzed in order to find out whether it is a descent local minimum that we could exploit or it is different than our intuition tells us that the global minimum should be.

In Chapter III we introduced the problem we called ghost-herding via a simulation in which the world's biggest-ever simultaneous UAV swarm [14], 2066 agents, was attacking our HVU, which was protected by 200 defenders. In such a large scale problem, our state vector, the set of data carrying the necessary information for every time-step to conduct our simulation, consists of the position and the velocity in 3-dimensional

coordinates of the attacking swarm, as well as the survival probability of all the assets (attackers, defenders, and HVU). Consequently, we end up for the aforementioned simulation with an enormous configuration space that has 14,662 variables.

As we may understand, a problem with so many variables may have thousands of local minimum solutions and a global minimum that is very computationally expensive to be found. We have to take into account that this application of optimal motion planning for the defenders is currently a robust analysis tool but the ultimate goal is to eventually reach real-time solutions.

Consequently, the larger the scale of our problem, the more important a good initial estimate will become, in order to end up with a local minimum close enough to the optimal solution, which is almost impossible to find.

In the rest of this chapter we demonstrate some results of our motion planning optimization algorithm for different initial conditions and different weapons characteristics that would indicate the ability of our framework to deal with a wide range of scenarios and simulation characteristics.

B. ELIMINATING THE GHOST-HERDING PROBLEM WITH OUR PROPOSED MODELS

We are going to revisit the scenario of the 2066 attackers versus the 200 defenders analyzed in Chapter III in order to demonstrate the ability of our proposed models to correlate effectively the dynamics of the antagonistic agents with the attrition models that are tracking the inflicted damage.

In Figures 24–26 we are tracking the survival probability of the HVU as well as the number of live defenders and attackers for each individual model. In Figure 24 we see that both the weighted and the threshold model are completely aligned with the Monte Carlo model that we use as a reference to validate our solution, whereas we see that the original code fails to predict the destruction of the HVU.

The Monte Carlo simulation is executed 200 times (ω=200) and the computed outcomes are averaged each time-step, in order to obtain unbias results from the random numbers generation procedure.

In Figure 25 we observe that the weighted model is closer to the Monte Carlo prediction concerning the live defenders for each time step. Nevertheless, both the weighted and the threshold models make perfect sense when combined with Figure 26. In this figure we see that around the 300th time sample, the confrontation of the antagonistic forces takes place. Despite the fact that the defenders in this scenario have superior weapons, they are heavily outnumbered and they all die around the 350th time sample, together with the HVU, due to the fact that there are no defenders left to protect it. Additionally, from this time sample and until the end of the scenario, we also see that the attackers suffer no more losses.

On the other hand, in this scenario we see that the ghost-herding model not only fails to predict the HVU survival probability but it also gives us inconclusive information concerning the inflicted attrition between opposing forces. Namely, we see in Figures 27 and 28 that according to the ghost-herding model, at the end of the scenario 1000 attackers and 60 defenders are still alive.

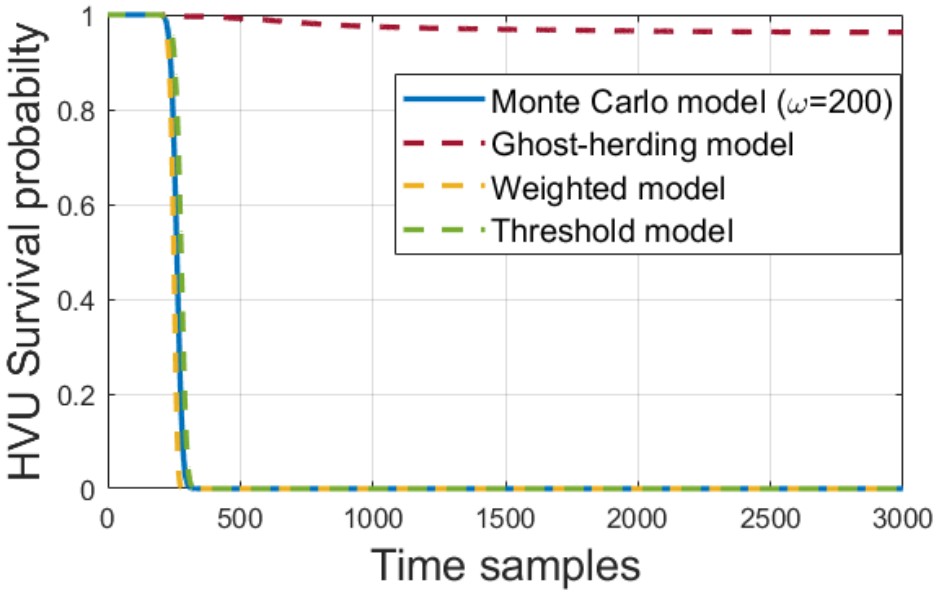

Figure 24. HVU survival probability comparison between multiple models for the scenario of 2066 attackers versus 200 defenders with superior weapons.

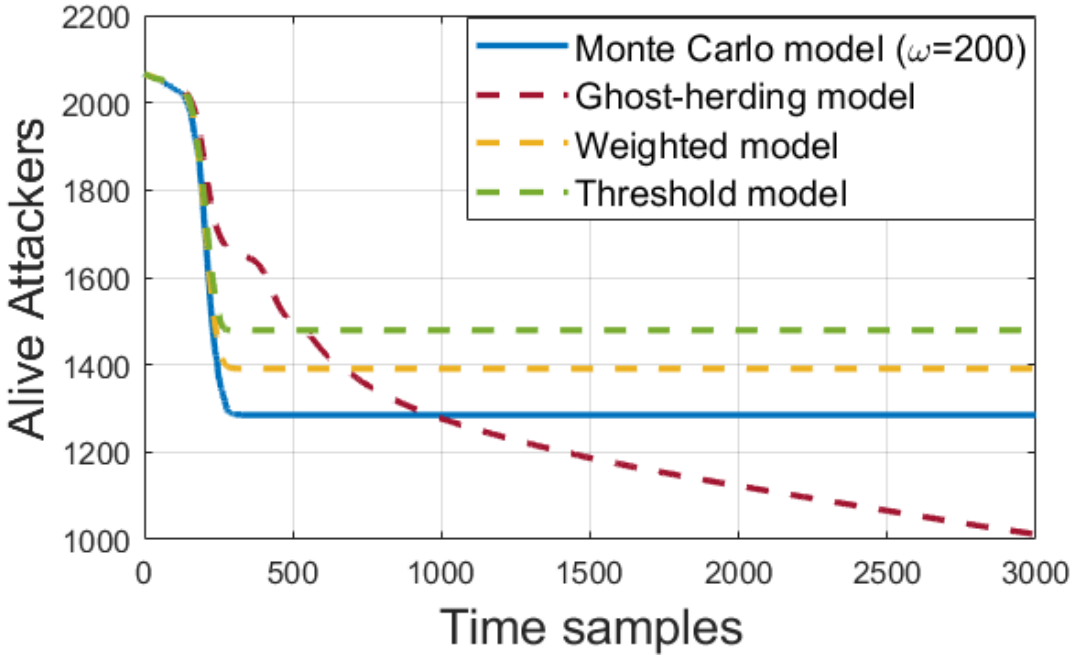

Figure 25. Number of live attackers for each time step for the different models introduced for the scenario of 2066 attackers versus 200 defenders with superior weapons.

This trend that we see in Figures 27 and 28 can be explained by examining the snapshots of the scenario depicted in Figure 29. In Figure 29 we see snapshots of the 3-dimensional representation of the scenario according to the ghost-herding model. Both attackers and defenders are color coded with their survival probability, where black means incapacitated and magenta and cyan mean operational. From the snapshots we see again the implications of having the dynamics of the model completely uncorrelated with the attrition model.

From the snapshots in Figure 29 we are able to observe that after the first 1000 time samples, multiple front layers of defenders and attackers have already been incapacitated. Since the attrition and the dynamics models are uncorrelated, these layers of incapacitated agents block the physical continuation of the scenario. As a result, we may see in Figures 25 and 26 that not only is the prediction of the HVU survival probability wrong, but also the survivability of both attackers and defenders is miscalculated.

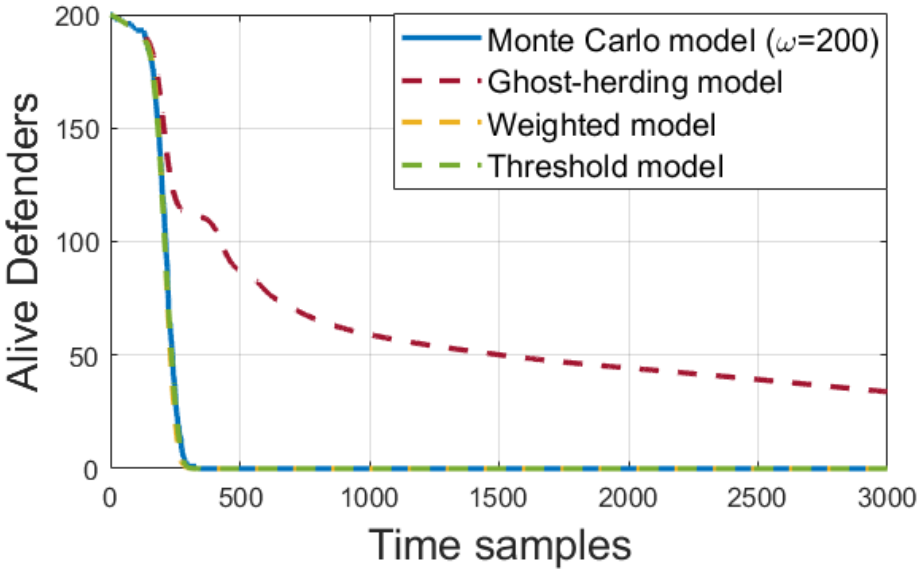

Figure 26. Number of live defenders for each time step for the different models introduced for the scenario of 2066 attackers versus 200 defenders with superior weapons.

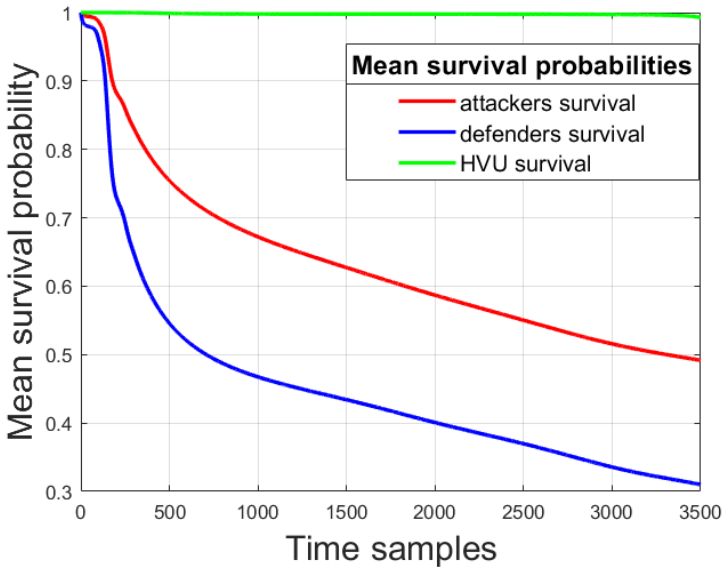

Figure 27. Mean survival probabilities for attackers, defenders, and HVU using the ill-performed ghost-herding model.

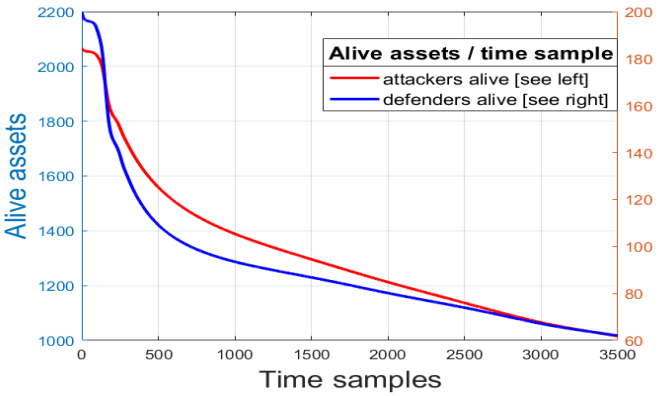

Figure 28. Number of live attackers and defenders for each time step, using the ill-performed ghost-herding model.

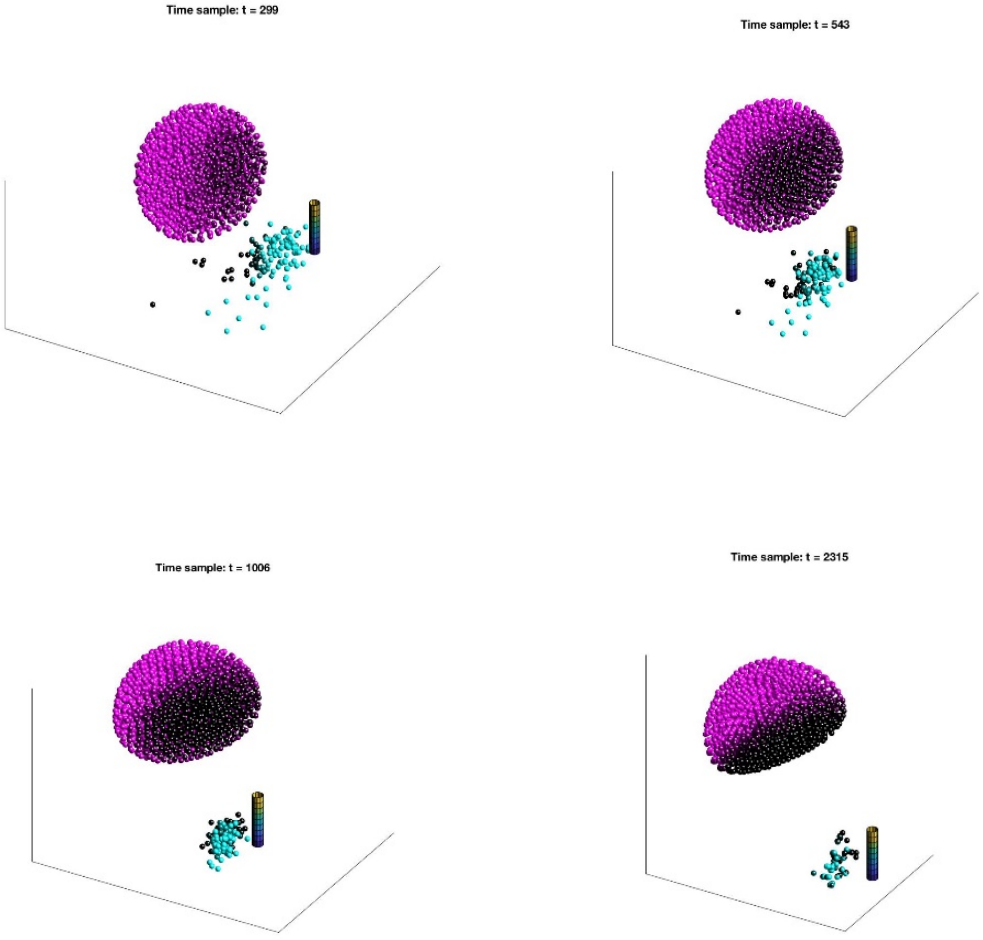

Figure 29. Snapshots for the scenario of 2066 attackers versus 200 defenders with superior weapons, performed with the ghost-herding model.

C. COMPARING THE PERFORMANCE OF THE PROPOSED MODELS BY COMPUTING THE NUMBER OF DEFENDERS REQUIRED TO DEFEND THE HVU FROM A SWARM ATTACK

In Section B of this chapter we compared the results derived from our proposed models that deal effectively with the correlation between dynamics and attrition. We used an example of 2066 attackers and 200 defenders, and we observed that the weighted and the threshold models derived almost the same results. Yet, one could say that our example had disproportional distribution of forces and the result was almost certain.

In Figures 30 and 31, we compare the optimization results for a confrontation between an attacking swarm of 50 agents and a defending force in order to compute how many defenders are required to effectively protect an HVU from such an assault, according to the different models.

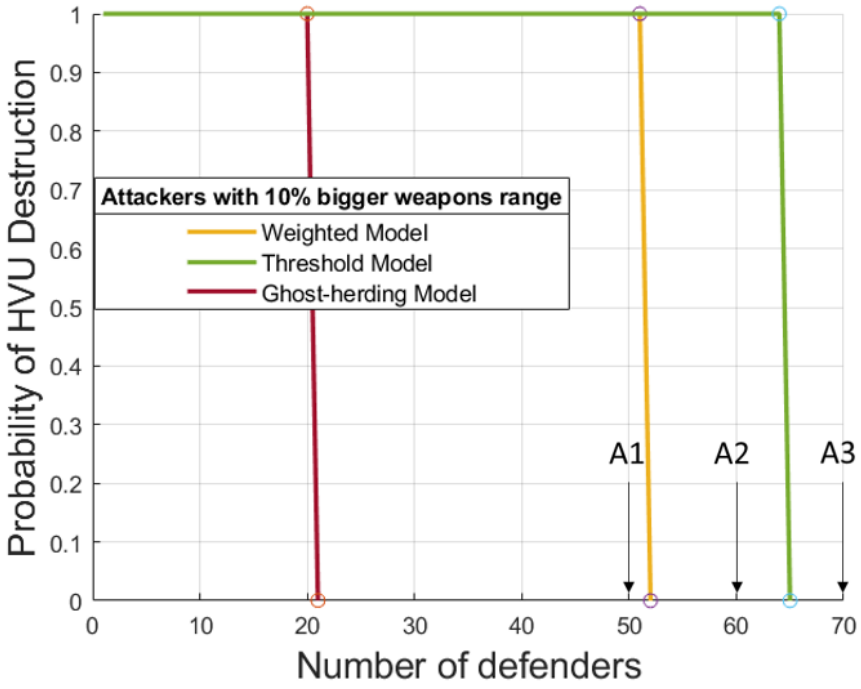

Figure 30. Number of defenders required to effectively protect an HVU from a 50-agent swarm attack with 10% longer weapons range than the defenders.

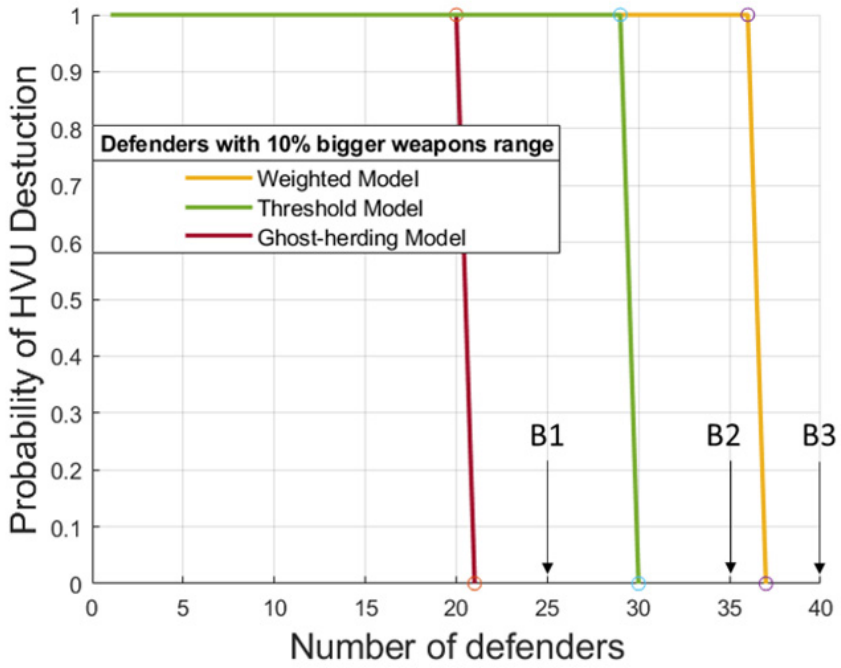

Figure 31. Number of defenders, with 10% longer weapons range than the attackers, required to effectively protect an HVU from a 50-agent swarm attack.

In Figure 30, the 50 agents of the attacking swarm have a 10% bigger range with respect to the defenders. The optimization results of this problem show us firstly that the ill-performing ghost-herding model differs significantly from the newly introduced models and is computing that only 20 defenders are enough. Additionally, we observe from Figure 30 that the weighted model estimates that 52 defenders are enough, whereas the threshold model gives a more pessimistic estimate requiring 65 defenders.

On the other hand, in Figure 31, the defending agents are those that have a 10% bigger range with respect to the attackers. In this simulation, though, the threshold model is more optimistic compared to the weighted model because, according to the weighted model computations, only 30 defenders are required versus 37 that the threshold model predicts. Interestingly, we see that in both scenarios, the estimate of the ghost-herding model is almost the same, either 20 or 21 defenders. This is another proof that the ghost-herding model gives non-physical results due to the uncorrelated attrition and dynamics models.

D. ANALYSIS OF THE OPTIMIZATION RESULTS

In order to analyze the derived optimization results, we refer again to the two scenarios introduced in Section C of this chapter, where a confrontation with a swarm of 50 agents was examined from both the perspective of stronger attackers and stronger defenders. Our analysis focuses on multiple checkpoints that are depicted in Figures 30 and 31.

These checkpoints (A1, B1) represent points in the plots predicting that the number of defenders is not sufficient with respect to our mission objective, according to both the weighted and the threshold models. Additionally, we examine the checkpoints (A3, B3) where the number of defenders is satisfactory for the HVU's protection according to both aforementioned models. On the other hand, we investigate the checkpoints (A2, B2) where both the weighted and the threshold models contradict each other. In all the aforementioned cases we use the Monte Carlo model as a reference to cross check our results.

The Monte Carlo simulation is executed 200 times (ω=200) and the computed outcomes are averaged each time-step, in order to obtain unbias results from the random numbers generation procedure.

1. Checkpoints A1–B1: Not Enough Defenders for HVU Protection

In Figures 32 and 33 we perform an analysis for the Checkpoints A1 and B1 derived from the optimization results presented via Figures 30 and 31. In Figure 32 we verify that the 50 defending agents are insufficient to prevent the HVU's destruction from the stronger 50 agents of the attacking swarm. In Figure 30 we saw that the threshold model was the most pessimistic for this scenario. This is the reason why in Figure 32 we see that according to this model, not only is the HVU destroyed more quickly, but also fewer defenders and more attackers remain operational. As a result, both models do capture the outcome of the confrontation, but the weighted model does that with higher fidelity.

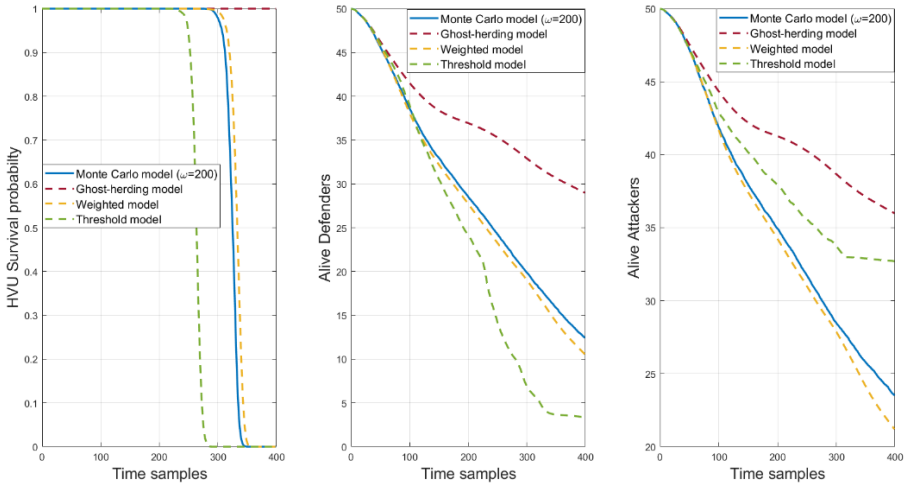

Figure 32. Checkpoint A1 (stronger attackers) analysis: Weighted and Threshold models align with the Monte Carlo estimation of HVU destruction.

Figure 33 is also in accordance with the optimization predictions of Figure 31. Even though the defenders are stronger now, 25 of them seem to be insufficient to achieve the goal of HVU protection. Both the weighted and the threshold models capture the outcome of the confrontation. Again the weighted model is closer to the Monte Carlo prediction, especially with respect to the mean attacker survivability.

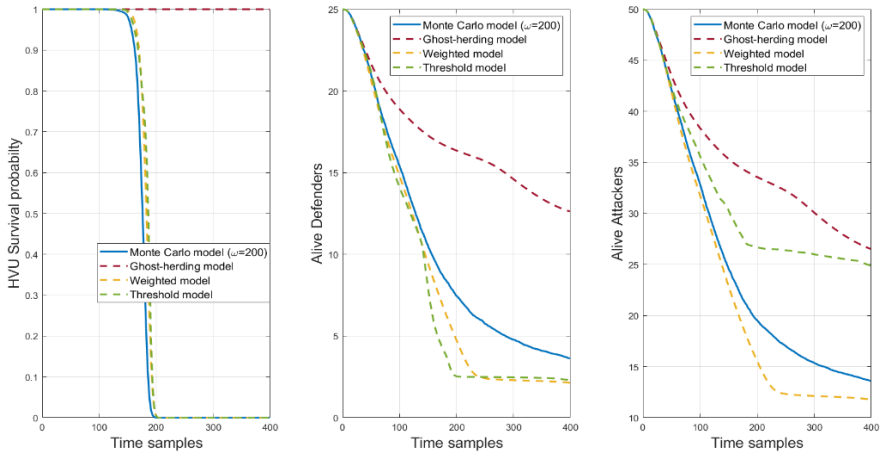

Figure 33. Checkpoint B1 (stronger defenders) analysis: Weighted and Threshold models align with the Monte Carlo estimation of HVU destruction.

2. Checkpoints A2–B2: Differentiation between Weighted and Threshold Models

Checkpoints A2 and B2 represent the gray area between the models. This uncertainty area where, according to Figures 30 and 31, the weighted and the threshold models predict different outcomes for the confrontation is what we are going to examine in this section.

According to both the scenarios of stronger attackers and stronger defenders, Figures 34 and 35, respectively, we may verify again the close fidelity of the weighted model with respect to the Monte Carlo simulation of the real outcome. In fact, we see that in the scenario of stronger attackers depicted in Figure 34, the 60 defenders are sufficient for the HVU's protection. The threshold model, however, predicts the opposite.

Moreover, in the scenario of stronger defenders depicted in Figure 35, for the confrontation between 50 attackers and 35 defenders we would have to increase the defending numbers to avoid the HVU's destruction. According to the threshold model, however, the HVU's survival could succeed even with this force of 35 defenders.

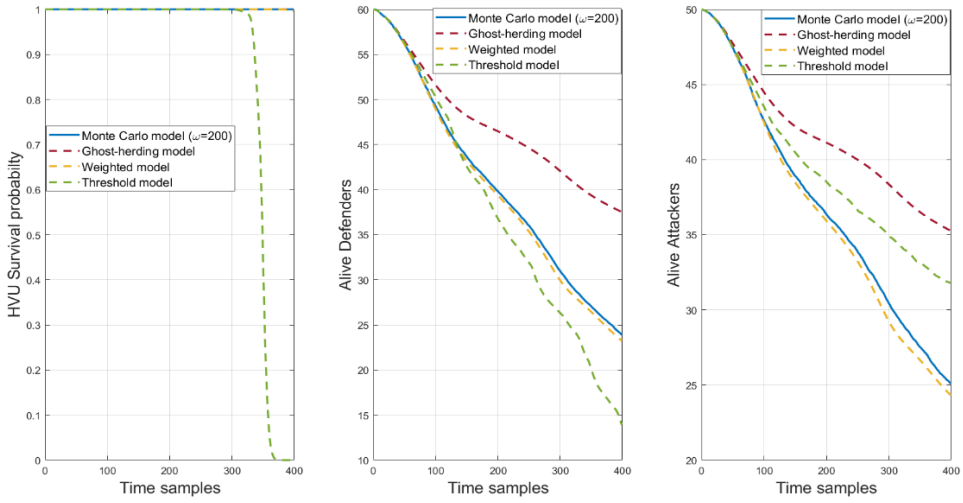

Figure 34. Checkpoint A2 (stronger attackers) analysis: Threshold model fails to predict HVU survival.

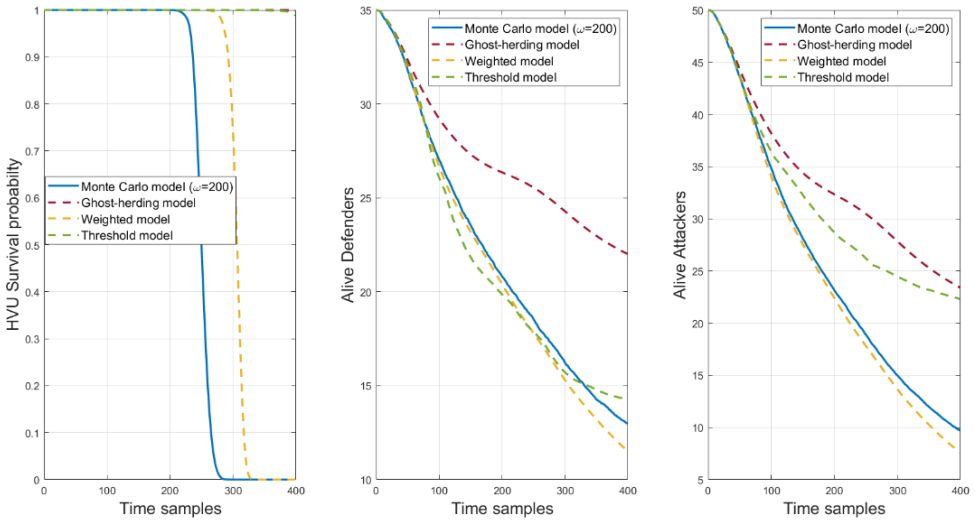

Figure 35. Checkpoint B2 (stronger defenders) analysis: Threshold model fails to predict HVU destruction.

3. Checkpoints A3–B3: Sufficient

Last but not least, we have to examine the Checkpoints A3 and B3 where the two optimization models, according to Figures 30 and 31, predict that our mission objective will be accomplished.

In Figures 36 and 37, we execute the analysis of the aforementioned checkpoints. As we may see, the results for both the scenarios of stronger attackers and stronger defenders verify the optimization prediction and the HVU's survival are achieved.

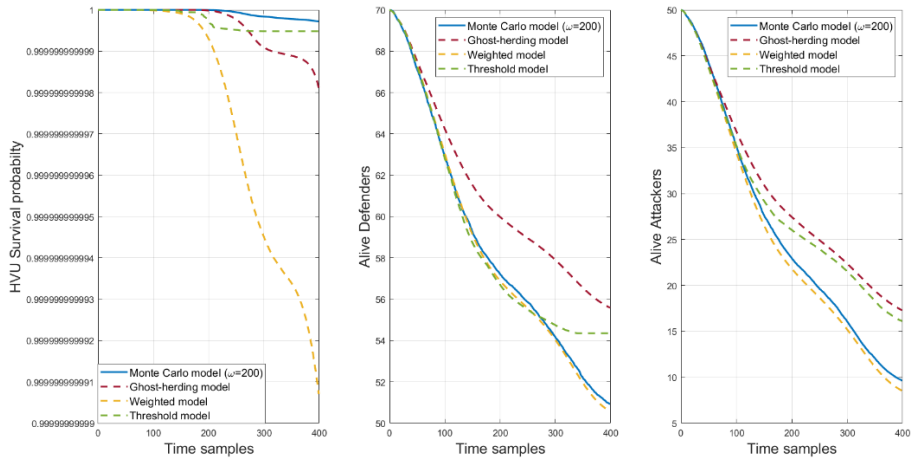

Figure 36. Checkpoint A3 (stronger attackers) analysis: Weighted and Threshold models align with the Monte Carlo estimation of HVU survival.

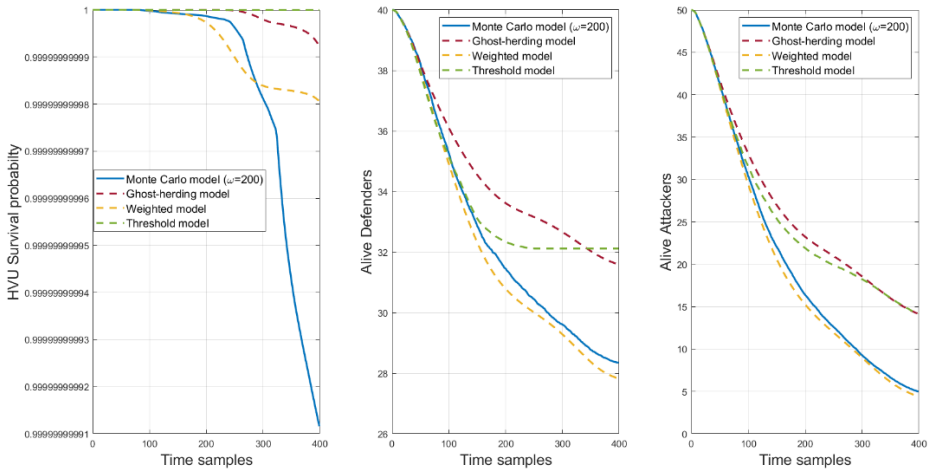

Figure 37. Checkpoint B3 (stronger defenders) analysis: Weighted and Threshold models align with the Monte Carlo estimation of HVU survival.

E. PERIPHERAL THREAT AXIS

A less practical but really challenging scenario is to defend an HVU asset when the attacking swarm has already encircled the defending forces. This means that we cannot focus our defending forces to a specific threat direction, rather we have to station the defenders accordingly, such that adequate peripheral protection is achieved.

In Figures 38–41, we examine this new scenario by optimizing the trajectories of 50 defenders with much superior weapons, double the range and double the fire rate, in order to deal with an assault from a swarm of 500 agents aiming to destroy the HVU.

In Figure 38 we see the followed trajectory of both the attackers and the defenders, color coded with their survival probabilities. Firstly, we observe that in such a confrontation where the threat axis is peripheral, the optimum solution for the fewer defenders that have superior weapons is to remain very close to the HVU. Secondly, we see that the defenders are effectively dealing with the threat because we see that when the attackers are close enough to launch their attack, they are moving around in a spherical pattern find potential weak points. However, by that time their path is depicted as black, which means that their survival probability is too low and they have been incapacitated.

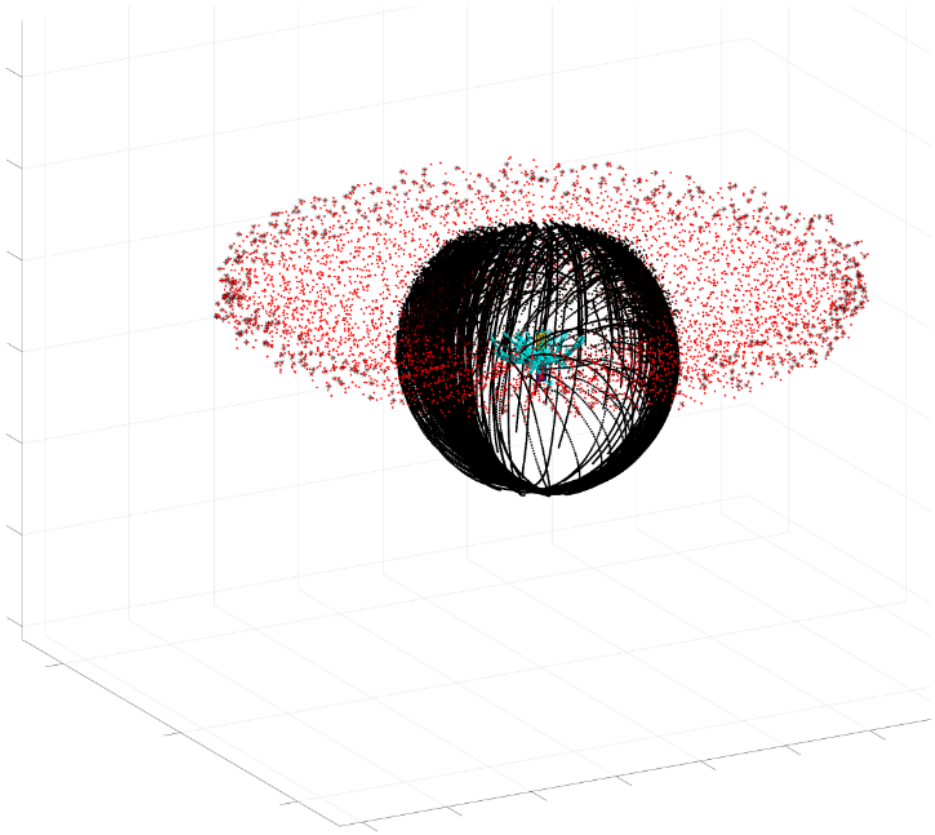

Figure 38. 500 attackers approaching from peripheral directions to destroy the HVU protected by 50 defenders with much superior weapons (double range and fire rate).

Figures 39 and 40 verify the assumptions made from Figure 38, because we observe in both figures that all the attackers are incapacitated by the end of the first 10% of the simulation, after a short period of mutual attrition, whereas almost 33 defenders remain operational by the end of the scenario.

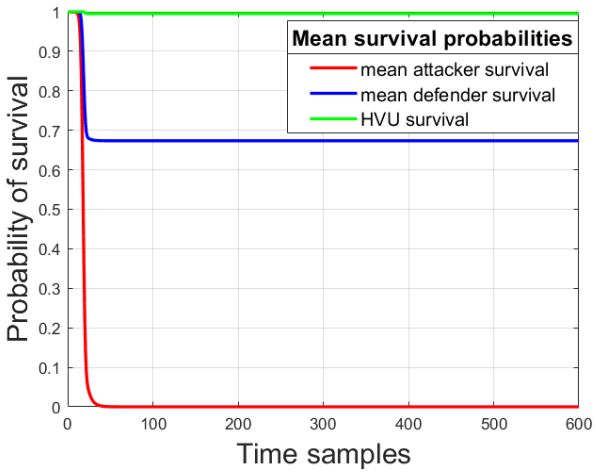

Figure 39. Mean survival probabilities for the scenario of the peripheral threat of 500 attackers facing 50 much stronger defenders.

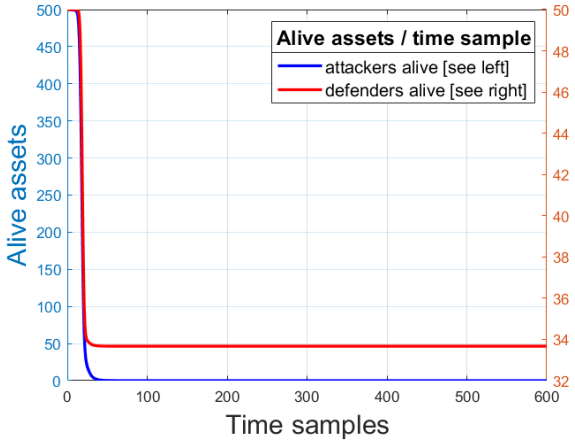

Figure 40. Mean number of live attackers and defenders for the scenario of the peripheral threat of 500 attackers facing 50 much stronger defenders.

In Figure 41, we execute a comparison analysis of the three different optimization models spoken of throughout this paper with respect to the Monte Carlo reference model.

Hence, we may see that all models are in close fidelity with the reference model—not only for tracking the HVU's survival probability but also for estimating the survivability of the rest of the antagonistic agents.

It is clear from this scenario, as well as from the previously examined scenarios, that the non-physical results of the ghost-herding model appear only in situations where the defenders would inevitably lose if they rely only on their weapons. On the other hand, in scenarios where the defenders are in sufficient numbers or are equipped with the appropriate weapons, the original ghost-herding model with the uncorrelated attrition and dynamics models would converge with the rest of the models.

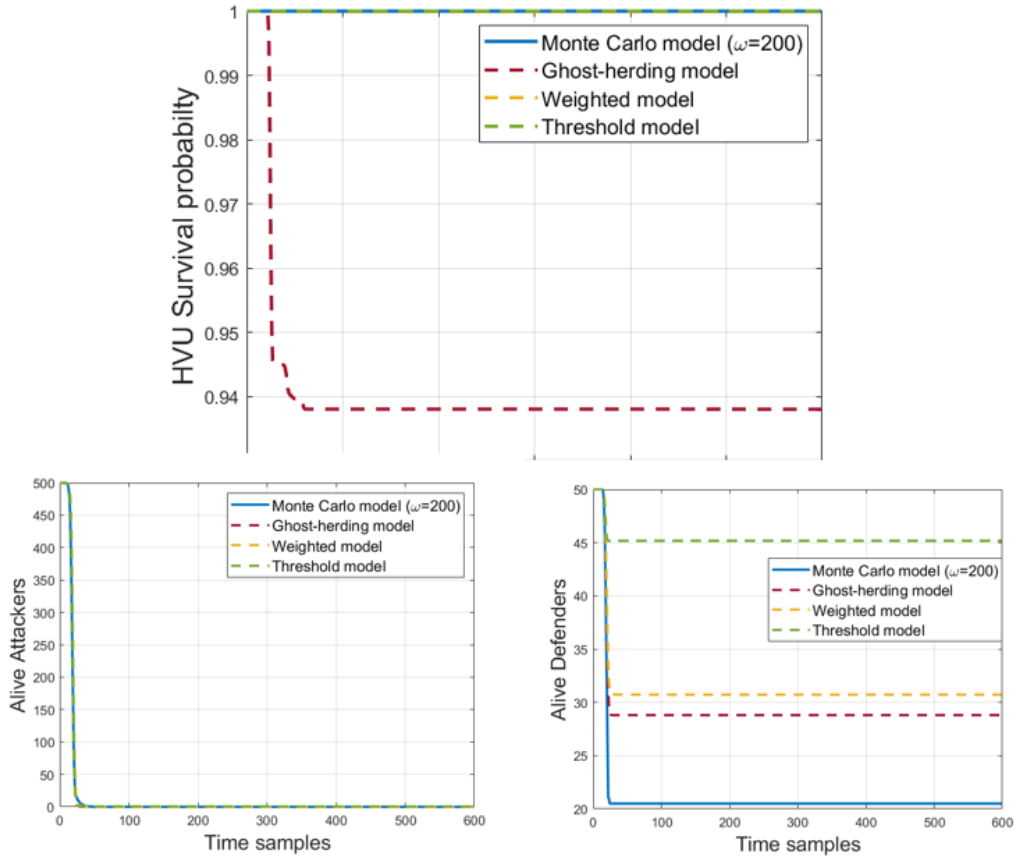

Figure 41. Comparison analysis of the four models for the confrontation between the 500 attackers approaching peripherally and the 50 much stronger defenders.

V. OPTIMIZATION RESULTS WITH RESPECT TO THE MISSION OBJECTIVE OF AIR SUPERIORITY

A. COMPARING OPTIMIZATION RESULTS FOR A CONFRONTATION WITH THE SAME CONDITIONS BUT DIFFERENT COST FUNCTIONS (HVU PROTECTION — AIR SUPERIORITY)

In this research, we have used the probability of an HVU's destruction as a mission objective for our motion planning problem. This HVU was a stationary unit with no self-defense capabilities. In this chapter, we introduce another mission objective to demonstrate the wide usability of our framework.

Specifically, we introduce a new cost function appropriate for achieving the mission objective of air superiority. The objective function of the mean attacker survival probability must be minimized to achieve our mission by causing the greatest possible attrition among the attackers. For consistency with the previous chapters, we regard the defending autonomous systems as dispensables means in order to achieve our goal.

In Figures 42–45, we compare a confrontation of 100 attackers versus a force of 25 defenders with double the fire rate and a 25% larger weapons range. In Figure 42 we see the trajectories of both the attackers and the defenders color-coded according to their survival probability. The defenders' path planning is computed with respect to the mission objective of HVU protection.

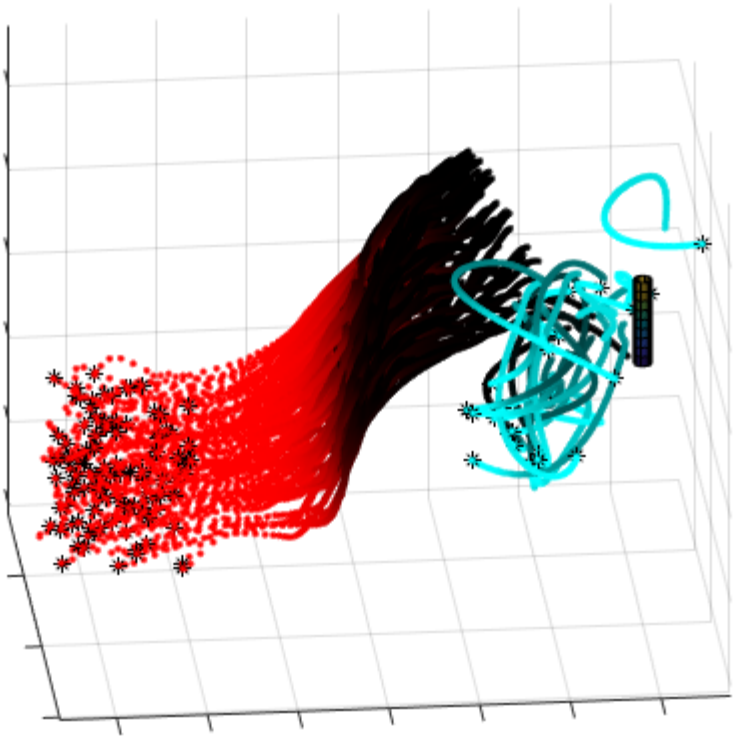

Figure 42. 100 attackers versus 25 defenders with superior weapons — HVU protection objective.

In addition to Figure 42, we can see in Figure 43 the color-coded trajectories of all the agents of the previous scenario. The initial conditions, the dynamics characteristics, and the weapons capabilities are exactly the same as before. The only difference is that on this occasion the defender trajectories are computed with respect to the new mission objective of air superiority acquisition.

The main difference we can observe between these two scenarios is that in Figure 42 the defenders have a more conservative motion. They intercept the attackers only when the attackers approach the HVU; the defenders block the attackers' way to the HVU and its destruction, and only then do they fight each other. In Figure 43, on the other hand, we see a more aggressive behavior on the part of the defenders. Now there is no HVU to be protected, and as a result, they are guided toward a head-on confrontation in order to achieve the minimization of the cost function as quickly as possible.

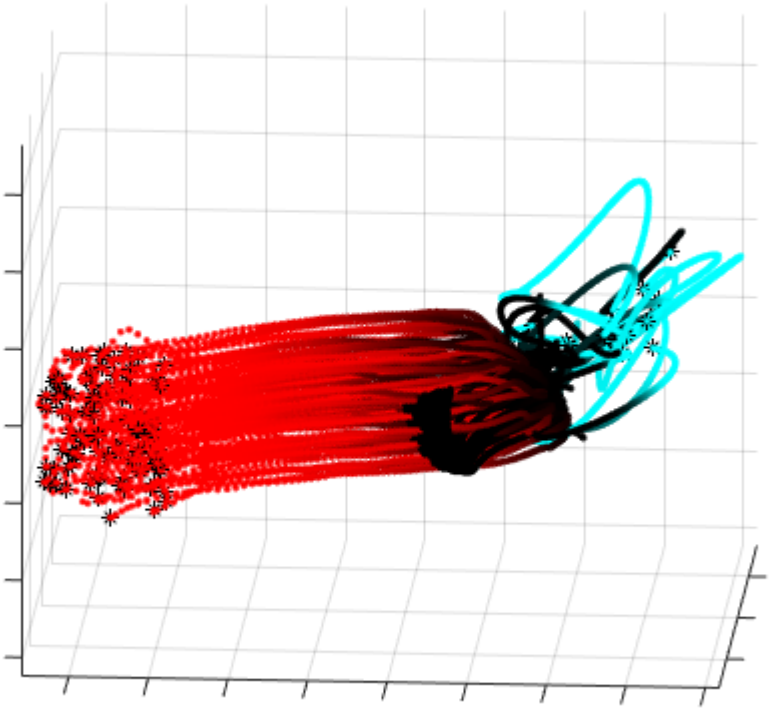

Figure 43. 100 attackers versus 25 defenders with superior weapons — Air superiority objective.

In Figures 44 and 45, the comparison of the two different scenarios is extended to the analysis of the survival probabilities and the survivability of all the agents. The take-away from these two figures is the following. Because the new objective function of air-superiority drives the defenders more aggressively to maximize the attrition of the opposing swarm, the cost that corresponds to the mean survival probability of the defenders decreases more quickly. This happens, however, with the sacrifice of more defenders with respect to the scenario of HVU protection. Namely, we see in Figure 44 that the mean defender survival probability for the air-superiority objective, at the end of the scenario, is 19% whereas in the HVU protection scenario it is 51%.

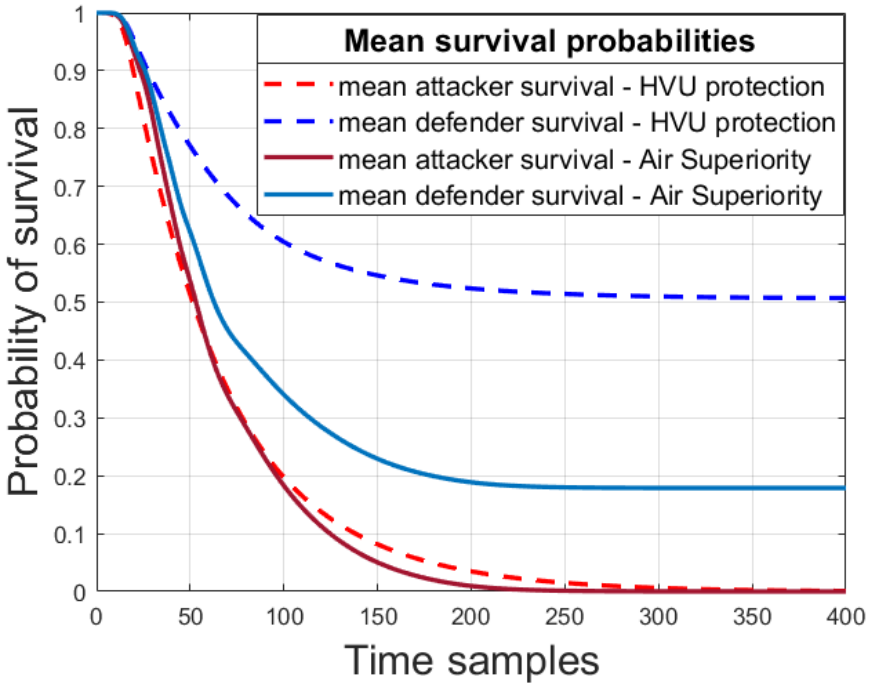

Figure 44. Mean survival probabilities comparison between the two different mission objectives for the scenarios with 100 attackers and 25 defenders with superior weapons.

In sheer numbers we see in Figure 45 that this mean probability difference corresponds to seven more defenders at the end of the HVU protection scenario. This difference makes sense since we regard the defender autonomous systems indispensable, and consequently, they do not influence the cost. If we had to consider their survivability as well, the head-on confrontation would not be the best solution since it does not exploit in the best way the extended range of the defenders' weapons.

Of course, this argument would not be correct if we had not extended this range for our forces, since we would like to drive the defenders as quickly as possible to a region where the extended weapons range advantage of the attackers would not matter anymore.

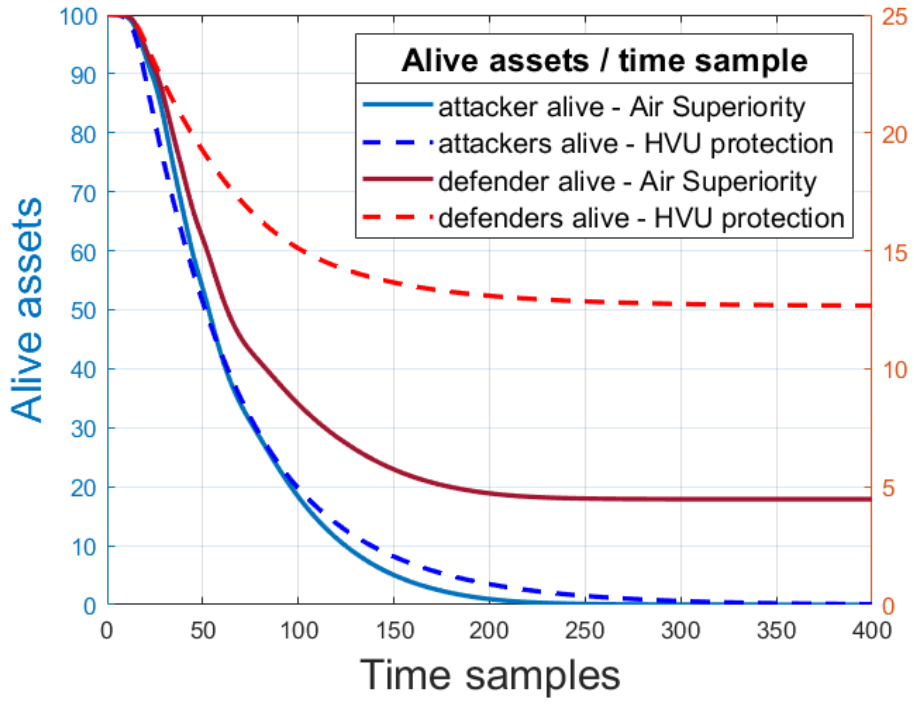

Figure 45. Mean survivability comparison between the two different mission objectives for the scenarios with 100 attackers and 25 defenders with superior weapons.

B. COMPARING THE PERFORMANCE OF THE PROPOSED MODELS BY COMPUTING THE NUMBER OF DEFENDERS REQUIRED TO OBTAIN AIR — SUPERIORITY

In Figures 46 and 47, we compare the optimization results for a confrontation between an attacking swarm of 50 agents and a defending force in order to compute how many defenders are required to achieve air superiority, according to the different models.

In Figure 46, the defending agents have a 10% bigger range with respect to the attackers. On the other hand, in Figure 47, the attacking swarm agents have a 10% bigger range with respect to the defenders. In both scenarios, though, the optimization results show us the same trend. Firstly, the ill-performing ghost-herding model is converging with the newly introduced models as the number of antagonistic agents that are interacting with each other is relatively low, and hence, the correlation between dynamics and attrition is not so important.

On the other hand, for a defending force of 20 or more agents, according to Figure 46, or for a defending force of 25 agents or more, according to Figure 47, we enter into the ghost-herding problem region. In this region the interaction of the antagonistic forces plays an important role in the outcome of the scenario. Consequently, these forces have to be correlated with the survivability of the agents in order to obtain a reliable computation of the final cost of the simulation.

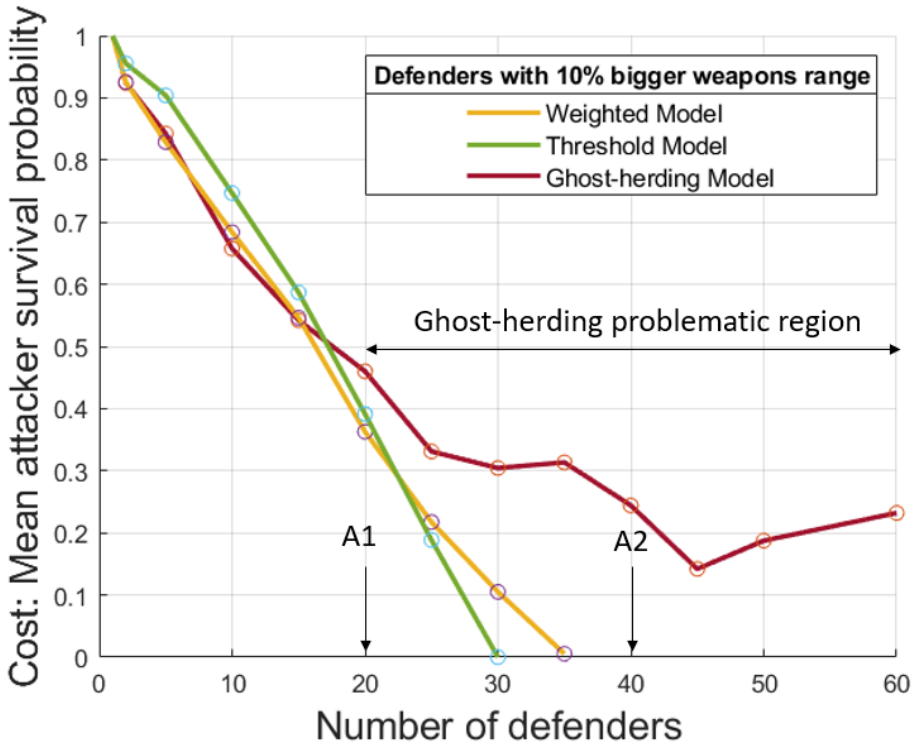

Figure 46. Number of defenders with 10% more extended weapons range than the attackers. Defenders must challenge a 50-agent swarm and obtain air superiority.

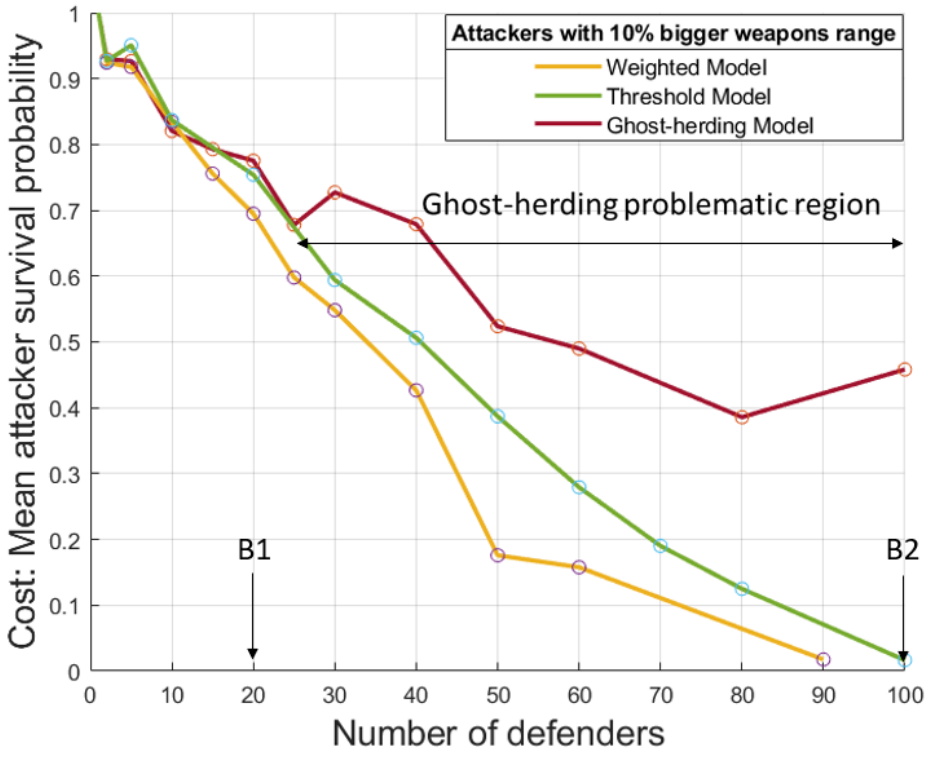

Figure 47. Number of defenders with 10% shorter weapons range than the attackers. Defenders must challenge a 50-agent swarm and obtain air superiority.

C. ANALYSIS OF THE OPTIMIZATION RESULTS

To analyze the derived optimization results, we refer again to the two scenarios introduced in Section B of this chapter, where a confrontation with a swarm of 50 agents was examined from both the perspective of the stronger attackers and the stronger defenders. Our analysis focuses on the checkpoints depicted in Figures 46 and 47.

These checkpoints (A1, B1) represent points in the plots predicting that the interaction forces do not influence the final cost of the scenario, and consequently, the ghost-herding model converges to both weighted and threshold models. On the other hand, we investigate the checkpoints (A2, B2) where the number of agents is such that the antagonistic forces do influence the final cost of the scenario. In all the aforementioned cases, we use the Monte Carlo model as a reference to cross check our results.

The Monte Carlo simulation is executed 200 times (ω=200) and the computed outcomes are averaged each time-step, in order to obtain unbias results from the random numbers generation procedure.

1. Checkpoints A1–B1: Models' Converging Region

In Figures 48 and 49 we perform an analysis for the Checkpoints A1 and B1 derived from the optimization results presented via Figures 46 and 47. In Figure 48 we see the analysis of a confrontation between 50 attackers and 20 defenders that have a 10% longer weapons range than the attackers. The objective of the scenario now is to minimize the attackers' survivability. Figure 48 shows us that all models converge not only with respect to the cost represented by the mean attackers' survivability but also with respect to the mean defenders' survival probability.

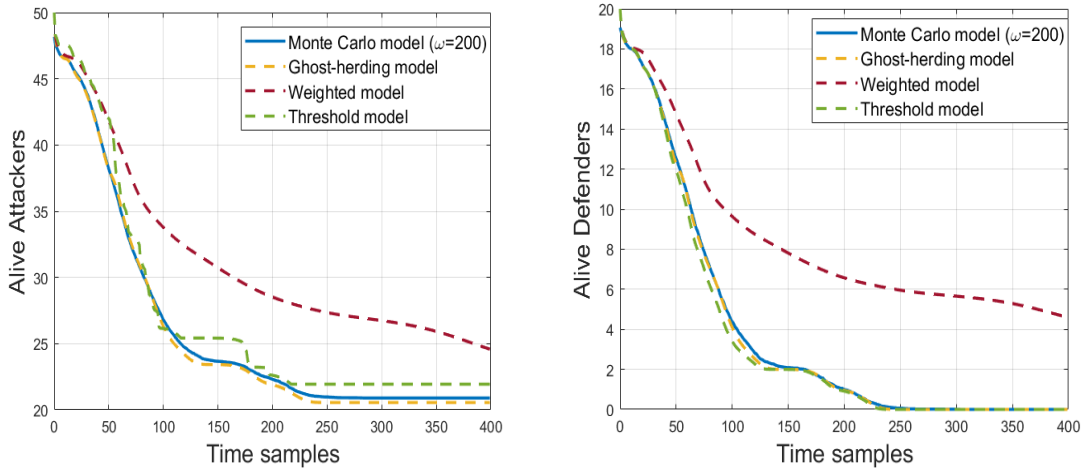

Figure 48. Checkpoint A1 (20 stronger defenders) analysis: Ghost-herding model converges to the other models.

Figure 49 also presents a simulation in which the ghost-herding problem does not differ significantly from the other models. Figure 49 also examines a confrontation between 50 attackers and 20 defenders, but now the 10% longer weapons range corresponds to the attackers.

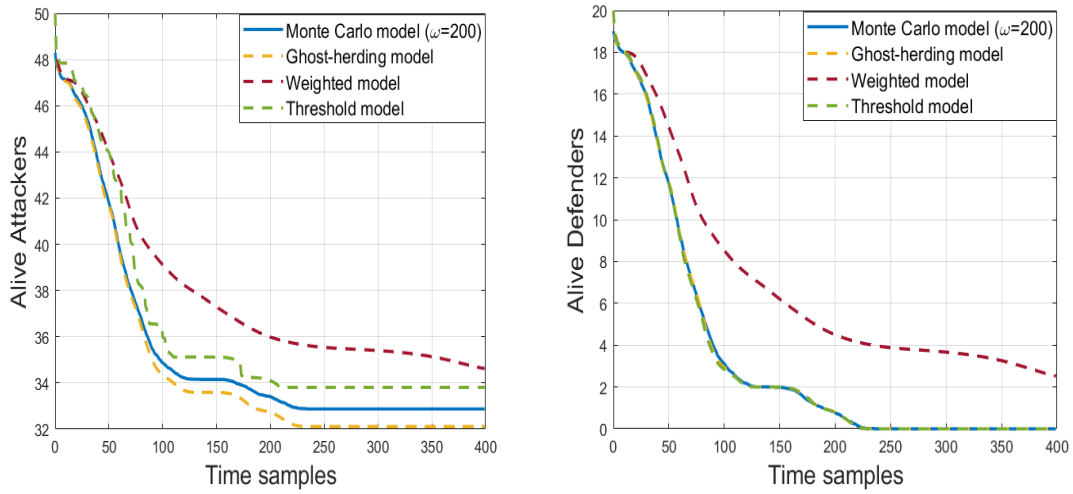

Figure 49. Checkpoint B1 (20 weaker defenders) analysis: Ghost-herding model converges to the other models.

2. Checkpoints A2–B2: Ghost-Herding Problem Region

Figure 50 depicts the Checkpoint A2 where 40 defenders with 10% longer weapons range eliminate an attacking swarm of 50 agents. In this simulation, the number of agents participating is such that we see that the problem of uncorrelated dynamics and attrition comes into play.

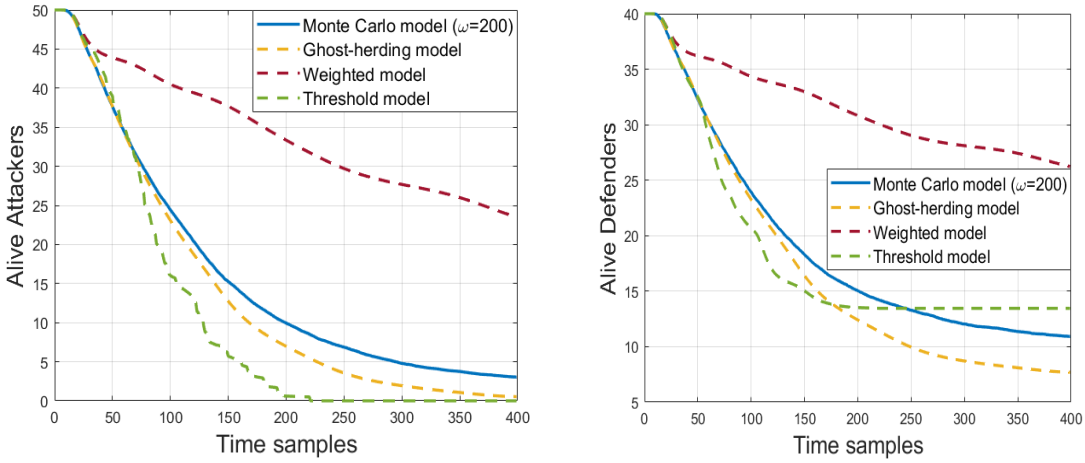

Figure 50. Checkpoint A2 (40 stronger defenders) analysis: Ghost-herding model differs with respect to the other models.

Consequently, agents are incapacitated and removed from the scenario for all models except from the ghost-herding model. In this model they remain in the field only to generate attractive and repulsive forces, since their weapons have been waived by the attrition model.

Figure 51 repeats the demonstration of the ghost-herding problem but this time for a confrontation involving 50 attackers versus 100 defenders. This time the attackers have the 10% range superiority in weapons capability.

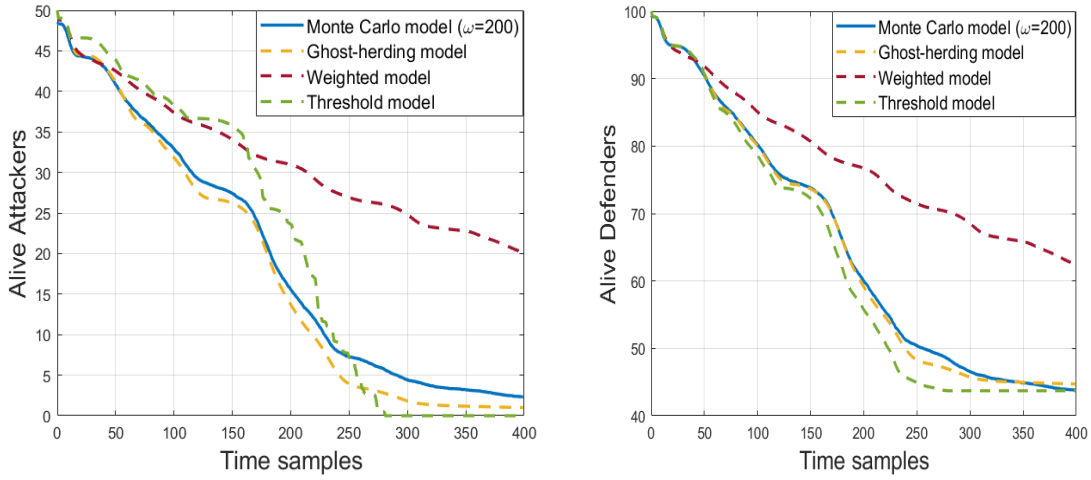

Figure 51. Checkpoint B2 (100 weaker defenders) analysis: Ghost-herding model differs with respect to the other models.

VI. CONCLUSION

The main goal of this thesis was to address a challenging issue of the optimal motion planning problem for autonomous systems facing large-scale antagonistic swarms. This issue is related to the nature of the models that come into play in such a framework. Namely, we seek to find the appropriate balance between the deterministic nature of the dynamics, which define the controlling forces for all the agents, and the probabilistic nature of the weapons engagements.

First, in Chapters I and II we introduced the framework used nowadays for optimal control. In Chapter III we demonstrated the issue we dubbed the ghost-herding problem in the original framework. Additionally, in Chapter III we presented our proposed models for dealing with this challenge. We introduced two models for optimization purposes, the weighted and the threshold models, as well as one model for analysis purposes, the Monte Carlo model.

The Monte Carlo simulation model was used as a reference during the analysis sections of this thesis. In this model, survivability is partially influenced by random number generation aimed at simulating the unpredictability that the real world exhibits. The demonstration and the analysis of our models were presented in Chapters IV and V for a number of different swarm confrontations. In Chapter IV, our cost was represented by our goal to protect an HVU unit whereas in Chapter V we used the objective function of minimizing the survival rate of the mean attackers.

Last but not least, the aspect of computational efficiency that allowed us to execute large-scale swarm simulations was addressed in Chapter II. A significant computational advantage was gained by replacing the Runge Kutta with the Verlet integration scheme frequently used in molecular dynamics. This almost equally numerically stable framework offers an impressive performance advantage.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. MATLAB FILES

A. COMPUTATIONALLY IMPROVED DYNAMICS / ATTRITION MODELS

1. Monte Carlo Dynamics and Attrition Model

a. Leonard Dynamics

```
function [yout,tot_surv_att,tot_surv_def,tot_surv_HVU,att_surv,def_surv] =
new_ode4_Leon_MonteCarlo(t0,dt,tf,Cd,N_attackers, N_defenders, N_Bezier, PARAMETERS,x,i)
seed = i;
rng(seed);
Patt_death = zeros(size((tf-t0)/dt));
Patt_rand = Patt_death;
Pdef_death = Patt_death;
Pdef_rand = Patt_death;
tot_surv_att = Patt_death;
tot_surv_def = Patt_death;
att_lamda = PARAMETERS.ATTACKERWEAPON.lambda;
att_sigma = PARAMETERS.ATTACKERWEAPON.sigma;
t_count = 1;
yout = x;
epsilon = 0.01; %distance cut off for near zero
umax = PARAMETERS.SWARM.umax;
K = PARAMETERS.SWARM.K;
K_hvu = PARAMETERS.SWARM.K_hvu;
d0_att=PARAMETERS.SWARM.d0;
d1_att=PARAMETERS.SWARM.d1;
alpha_att=PARAMETERS.SWARM.alpha_i;
d0_def=PARAMETERS.SWARM.INTd0;
d1_def=PARAMETERS.SWARM.INTd1;
alpha_def=PARAMETERS.SWARM.alphaINT_i;
mass_att = ones(1,N_attackers);
follower_states = reshape(x(1:6*N_attackers),6,N_attackers);
x_att=follower_states(1,:);
y_att=follower_states(2,:);
z_att=follower_states(3,:);
vx_att=follower_states(4,:);
vy_att=follower_states(5,:);
vz_att=follower_states(6,:);
ax_att = 0;
ay_att = 0;
az_att = 0;
ax_att_old = ax_att;
ay_att_old = ay_att;
az_att_old = az_att;
Q = x(6*N_attackers+1:7*N_attackers); %probability of attacker survival
Pd = x(7*N_attackers+1:7*N_attackers+N_defenders); %prob of def surv
P = x(end); %prob of HVU surv
t = t0:dt:tf;
length_time = length(t);
def_surv = ones(length_time, N_defenders);
att_surv = ones(length_time, N_attackers);
HVU_surv = 1;
for t = t0 : dt : tf-dt
   % first step in Verlet integration
   x_att=x_att+vx_att*dt+ax_att_old.*dt.^2/2;
```

```
y_att=y_att+vy_att*dt+ay_att_old.*dt.^2/2;
z_att=z_att+vz_att*dt+az_att_old.*dt.^2/2;
BN_t = bernsteinMatrix_a2b(N_Bezier,t,t0,tf);
pd = (BN_t*Cd);
pd = reshape(pd,3,N_defenders); % pd - position of the defenders
x_def=pd(1,:);
y_def=pd(2,:);
z_def=pd(3,:);
pd = [x_def;y_def;z_def];
follower_states = [x_att;y_att;z_att;vx_att;vy_att;vz_att];
position = [follower_states(1:3,:) pd]; % position: 3-by-(n+m)
p_hvu = PARAMETERS.DEFENDER.p_hvu;
Fx = zeros(1,N_attackers);
Fy = Fx;
Fz = Fx;
for nn = 1:N_attackers
   if att_surv(t_count,nn)==1
      dx_hvu = x_att(nn)-p_hvu(1);
      dy_hvu = y_att(nn)-p_hvu(2);
      dz_hvu = z_att(nn)-p_hvu(3);
      dd_hvu = sqrt(dx_hvu.^2+dy_hvu.^2+dz_hvu.^2);
      Fx(nn) = Fx(nn) - K_hvu.*dx_hvu./dd_hvu;
      Fy(nn) = Fy(nn) - K_hvu.*dy_hvu./dd_hvu;
      Fz(nn) = Fz(nn) - K_hvu.*dz_hvu./dd_hvu;
      for mm = (nn+1):N_attackers
         if att_surv(t_count,mm)==1
            dx = x_att(nn) - x_att(mm);
            if dx < d1_att
               dy = y_att(nn) - y_att(mm);
               if dy < d1_att
                  dz = z_att(nn) - z_att(mm);
                  ddnm = max(sqrt(dx.^2+dy.^2+dz.^2),epsilon);
                  if ddnm < d1_att
                     F = (1./ddnm.^2).*alpha_att.*(1-d0_att./ddnm);%;
                     Fx(nn) = Fx(nn) - F.*dx;
                     Fx(mm) = Fx(mm) + F.*dx;% same
                     Fy(nn) = Fy(nn) - F.*dy;% add something there
                     Fy(mm) = Fy(mm) + F.*dy;% same
                     Fz(nn) = Fz(nn) - F.*dz;% add something there
                     Fz(mm) = Fz(mm) + F.*dz;% same
                  end
               end
            end
         end
      end
   end
end
% b. Velocity dampening forces
Fx = Fx - K.*vx_att;
Fy = Fy - K.*vy_att;
Fz = Fz - K.*vz_att;
% c. Defender forces computation
distance_2defenders = d1_def*ones(N_attackers, N_defenders);
for nn = 1:N_attackers
   if att_surv(t_count,nn)==1
      for mm = 1:N_defenders
         if def_surv(t_count,mm)==1
            dx = x_att(nn) - x_def(mm);
            if dx < d1_def
               dy = y_att(nn) - y_def(mm);
               if dy < d1_def
```

```
                dz = z_att(nn) - z_def(mm);
                distance_2defenders(nn,mm) = max(sqrt(dx.^2+dy.^2+dz.^2),epsilon);
                if distance_2defenders(nn,mm) < d1_def
                    F = (1./(distance_2defenders(nn,mm)).^2).*alpha_def.*(1-d0_def./distance_2defenders(nn,mm));
                    Fx(nn) = Fx(nn) - F.*dx;
                    Fy(nn) = Fy(nn) - F.*dy;
                    Fz(nn) = Fz(nn) - F.*dz;
                end
            end
        end
    end
  end
end
% Thrust limits
for i = 1:N_attackers
    if Fx(i) >= umax
        Fx(i) = umax;
    elseif Fx(i) <= -umax
        Fx(i) = -umax;
    end
    if Fy(i) >= umax
        Fy(i) = umax;
    elseif Fy(i) <= -umax
        Fy(i) = -umax;
    end
    if Fz(i) >= umax
        Fz(i) = umax;
    elseif Fz(i) <= -umax
        Fz(i) = -umax;
    end
end
%Second step in Verlet integration
vx_att = vx_att + (ax_att_old + ax_att).*dt/2;
vy_att = vy_att + (ay_att_old + ay_att).*dt/2;
vz_att = vz_att + (az_att_old + az_att).*dt/2;
ax_att_old = ax_att;
ay_att_old = ay_att;
az_att_old = az_att;
%%  Deterministic Approach for attrition
rand_def = rand(1,N_defenders);
rand_att = rand(1,N_attackers);
rand_HVU = rand(1,1);
P_death_att=(1-prod(1-rda2d'.*dt));
P_death_def=(1-prod(1-rdd2a.*dt));
P_death_HVU=(1-prod(1-ra2hvu.*dt));
att_surv(t_count:end,rand_att<P_death_att)=0;
def_surv(t_count:end,rand_def<P_death_def)=0;
HVU_surv(rand_HVU<P_death_HVU)=0;
Patt_death(t_count) = mean(P_death_att);
Patt_rand(t_count) = mean(rand_att);
Pdef_death(t_count) = mean(P_death_def);
Pdef_rand(t_count) = mean(rand_def);
tot_surv_att(t_count) = sum(att_surv(t_count,:));
tot_surv_def(t_count) = sum(def_surv(t_count,:));
tot_surv_HVU(t_count) = HVU_surv;
t_count = t_count+1;
dy = [reshape([x_att; y_att; z_att; vx_att; vy_att; vz_att],6*N_attackers,1)];
yout = [yout; dy];
end
```

b. Reynolds Dynamics

```
function [yout, Patt_death, Patt_rand,Pdef_death, Pdef_rand,tot_surv_att,tot_surv_def,tot_surv_HVU] =
new_ode4_Reyn_MonteCarlo(t0,dt,tf,Cd,N_attackers, N_defenders, N_Bezier, PARAMETERS,x,i)
seed = i;
rng(seed);
Patt_death = zeros(size((tf-t0)/dt));
Patt_rand = Patt_death;
Pdef_death = Patt_death;
Pdef_rand = Patt_death;
tot_surv_att = Patt_death;
tot_surv_def = Patt_death;
t_count = 1;
yout = x;
epsilon = 0.01; %distance cut off for near zero
umax = PARAMETERS.SWARM.umax;
K = PARAMETERS.SWARM.K;
K_hvu = PARAMETERS.SWARM.K_hvu;
d_alig=PARAMETERS.SWARM.N_size_al_F;
weight_alig = PARAMETERS.SWARM.w_al_F;
% d_cohes=PARAMETERS.SWARM.N_size_c_F;
weight_cohes=PARAMETERS.SWARM.w_c_F;
d_separ=PARAMETERS.SWARM.N_size_s_I;
weight_separ = PARAMETERS.SWARM.w_s_F;
weight_separ_intr = PARAMETERS.SWARM.w_s_I;
mass_att = ones(1,N_attackers);
follower_states = reshape(x(1:6*N_attackers),6,N_attackers);
x_att=follower_states(1,:);
y_att=follower_states(2,:);
z_att=follower_states(3,:);
vx_att=follower_states(4,:);
vy_att=follower_states(5,:);
vz_att=follower_states(6,:);
ax_att = 0;
ay_att = 0;
az_att = 0;
ax_att_old = ax_att;
ay_att_old = ay_att;
az_att_old = az_att;
Q = x(6*N_attackers+1:7*N_attackers);
Pd = x(7*N_attackers+1:7*N_attackers+N_defenders);
P = x(end);
def_surv = ones(1, N_defenders);
att_surv = ones(1, N_attackers);
HVU_surv = 1;
for t = t0 : dt : tf-dt
   % first step in Verlet integration
   x_att=x_att+vx_att*dt+ax_att_old.*dt.^2/2;
   y_att=y_att+vy_att*dt+ay_att_old.*dt.^2/2;
   z_att=z_att+vz_att*dt+az_att_old.*dt.^2/2;
   BN_t = bernsteinMatrix_a2b(N_Bezier,t,t0,tf);
   pd = (BN_t*Cd);
   pd = reshape(pd,3,N_defenders); % pd - position of the defenders
   follower_states = [x_att;y_att;z_att;vx_att;vy_att;vz_att];
   position = [follower_states(1:3,:) pd]; % position: 3-by-(n+m)
   velocity = [follower_states(4:6,:) vd]; % velocity: 3-by-(n+m)
   p_hvu = PARAMETERS.DEFENDER.p_hvu;
   Fx = zeros(1,N_attackers);
   Fy = Fx;
   Fz = Fx;
```

```
%Velocity dampening forces
Fx = Fx - K.*vx_att;
Fy = Fy - K.*vy_att;
Fz = Fz - K.*vz_att;
counter_al = zeros(1,N_attackers);
sumx_al = counter_al;
sumy_al = counter_al;
sumz_al = counter_al;
sumx_coh = counter_al;
sumy_coh = counter_al;
sumz_coh = counter_al;
counter_sep = zeros(1,N_attackers);
sumx_sep = counter_sep;
sumy_sep = counter_sep;
sumz_sep = counter_sep;
x_def=pd(1,:);
y_def=pd(2,:);
z_def=pd(3,:);
counter_sep_int = zeros(1,N_attackers);
sumx_sep_int = counter_sep_int;
sumy_sep_int = counter_sep_int;
sumz_sep_int = counter_sep_int;
distance_2defenders = d_separ*ones(N_attackers, N_defenders);
for nn = 1:N_attackers
   if att_surv(nn)==1
      dx_hvu = x_att(nn)-p_hvu(1);
      dy_hvu = y_att(nn)-p_hvu(2);
      dz_hvu = z_att(nn)-p_hvu(3);
      dd_hvu = sqrt(dx_hvu.^2+dy_hvu.^2+dz_hvu.^2);
      %Artificial Potential forces
      Fx(nn) = Fx(nn) - K_hvu.*dx_hvu./dd_hvu;
      Fy(nn) = Fy(nn) - K_hvu.*dy_hvu./dd_hvu;
      Fz(nn) = Fz(nn) - K_hvu.*dz_hvu./dd_hvu;
      for mm = (nn+1):N_attackers
         if att_surv(mm)==1
            dx = x_att(nn) - x_att(mm);
            if dx < d_alig
               dy = y_att(nn) - y_att(mm);
               if dy < d_alig
                  dz = z_att(nn) - z_att(mm);
                  ddnm = sqrt(dx.^2+dy.^2+dz.^2);
                  if (ddnm < d_separ && ddnm > epsilon)
                     counter_sep(nn) = counter_sep(nn) +1;
                     counter_sep(mm) = counter_sep(mm) +1;
                     sumx_sep(nn) = sumx_sep(nn) + (dx/ddnm);
                     sumx_sep(mm) = sumx_sep(mm) - (dx/ddnm);
                     sumy_sep(nn) = sumy_sep(nn) + (dy/ddnm);
                     sumy_sep(mm) = sumy_sep(mm) - (dy/ddnm);
                     sumz_sep(nn) = sumz_sep(nn) + (dz/ddnm);
                     sumz_sep(mm) = sumz_sep(mm) - (dz/ddnm);
                     counter_al(nn) = counter_al(nn) +1;
                     counter_al(mm) = counter_al(mm) +1;
                     sumx_al(nn) = sumx_al(nn) + vx_att(mm);
                     sumx_al(mm) = sumx_al(mm) + vx_att(nn);
                     sumy_al(nn) = sumy_al(nn) + vy_att(mm);
                     sumy_al(mm) = sumy_al(mm) + vy_att(nn);
                     sumz_al(nn) = sumz_al(nn) + vz_att(mm);
                     sumz_al(mm) = sumz_al(mm) + vz_att(nn);
                     sumx_coh(nn) = sumx_coh(nn) + x_att(mm);
                     sumx_coh(mm) = sumx_coh(mm) + x_att(nn);
                     sumy_coh(nn) = sumy_coh(nn) + y_att(mm);
```

```
                    sumy_coh(mm) = sumy_coh(mm) + y_att(nn);
                    sumz_coh(nn) = sumz_coh(nn) + z_att(mm);
                    sumz_coh(mm) = sumz_coh(mm) + z_att(nn);
                elseif (ddnm < d_alig  && ddnm > epsilon)
                    counter_al(nn) = counter_al(nn) +1;
                    counter_al(mm) = counter_al(mm) +1;
                    sumx_al(nn) = sumx_al(nn) + vx_att(mm);
                    sumx_al(mm) = sumx_al(mm) + vx_att(nn);
                    sumy_al(nn) = sumy_al(nn) + vy_att(mm);
                    sumy_al(mm) = sumy_al(mm) + vy_att(nn);
                    sumz_al(nn) = sumz_al(nn) + vz_att(mm);
                    sumz_al(mm) = sumz_al(mm) + vz_att(nn);
                    sumx_coh(nn) = sumx_coh(nn) + x_att(mm);
                    sumx_coh(mm) = sumx_coh(mm) + x_att(nn);
                    sumy_coh(nn) = sumy_coh(nn) + y_att(mm);
                    sumy_coh(mm) = sumy_coh(mm) + y_att(nn);
                    sumz_coh(nn) = sumz_coh(nn) + z_att(mm);
                    sumz_coh(mm) = sumz_coh(mm) + z_att(nn);

                end
            end
        end
    end
    if att_surv(mm)==1
        for mm = 1:N_defenders
            if def_surv(mm)==1
                dx = x_att(nn) - x_def(mm);
                if dx < d_separ
                    dy = y_att(nn) - y_def(mm);
                     if dy < d_separ
                        dz = z_att(nn) - z_def(mm);
                        distance_2defenders(nn,mm) = sqrt(dx.^2+dy.^2+dz.^2);
                        if (distance_2defenders(nn,mm) < d_separ  && distance_2defenders(nn,mm) > epsilon)
                            counter_sep_int(nn) = counter_sep_int(nn) +1;
                            sumx_sep_int(nn) = sumx_sep_int(nn) + (dx./distance_2defenders(nn,mm));
                            sumy_sep_int(nn) = sumy_sep_int(nn) + (dy./distance_2defenders(nn,mm));
                            sumz_sep_int(nn) = sumz_sep_int(nn) + (dz./distance_2defenders(nn,mm));
                        end
                    end
                end
            end
        end
    end
  end
end
for nn = 1:N_attackers
  if counter_al(nn) ==0
     counter_al(nn)=1;
  end
  if counter_sep(nn) ==0
     counter_sep(nn)=1;
  end
  if counter_sep_int(nn)==0
     counter_sep_int(nn)=1;
  end
end
%Alignment Forces &Cohesion Forces
Fx = Fx + mass_att.*(weight_alig.*(sumx_al./counter_al - vx_att) + weight_cohes.*(sumx_coh./counter_al - x_att));
Fy = Fy + mass_att.*(weight_alig.*(sumy_al./counter_al - vy_att) + weight_cohes.*(sumy_coh./counter_al - y_att));
Fz = Fz + mass_att.*(weight_alig.*(sumz_al./counter_al - vz_att) + weight_cohes.*(sumz_coh./counter_al - z_att));
```

```
    %Separation Forces
    Fx = Fx + mass_att.*(weight_separ.*(sumx_sep./counter_sep) +
weight_separ_intr.*(sumx_sep_int./counter_sep_int));
    Fy = Fy + mass_att.*(weight_separ.*(sumy_sep./counter_sep) +
weight_separ_intr.*(sumy_sep_int./counter_sep_int));
    Fz = Fz + mass_att.*(weight_separ.*(sumz_sep./counter_sep) +
weight_separ_intr.*(sumz_sep_int./counter_sep_int));
    % Thrust limits
    for i = 1:N_attackers
        if Fx(i) >= umax
            Fx(i) = umax;
        elseif Fx(i) <= -umax
            Fx(i) = -umax;
        end
        if Fy(i) >= umax
            Fy(i) = umax;
        elseif Fy(i) <= -umax
            Fy(i) = -umax;
        end
        if Fz(i) >= umax
            Fz(i) = umax;
        elseif Fz(i) <= -umax
            Fz(i) = -umax;
        end
    end
    ax_att = Fx./mass_att;
    ay_att = Fy./mass_att;
    az_att = Fz./mass_att;
    vx_att = vx_att + (ax_att_old + ax_att).*dt/2;
    vy_att = vy_att + (ay_att_old + ay_att).*dt/2;
    vz_att = vz_att + (az_att_old + az_att).*dt/2;
    ax_att_old = ax_att;
    ay_att_old = ay_att;
    az_att_old = az_att;
    dy = [reshape([x_att; y_att; z_att; vx_att; vy_att; vz_att],6*N_attackers,1);Q;Pd;P];
    yout = [yout; dy];
    rand_def = rand(1,N_defenders);
    rand_att = rand(1,N_attackers);
    rand_HVU = rand(1,1);
    P_death_att=(1-prod(1-rda2d'.*dt));
    P_death_def=(1-prod(1-rdd2a.*dt));
    P_death_HVU=(1-prod(1-ra2hvu.*dt));
    att_surv(rand_att<P_death_att)=0;
    def_surv(rand_def<P_death_def)=0;
    HVU_surv(rand_HVU<P_death_HVU)=0;
    Patt_death(t_count) = mean(P_death_att);
    Patt_rand(t_count) = mean(rand_att);
    Pdef_death(t_count) = mean(P_death_def);
    Pdef_rand(t_count) = mean(rand_def);
    tot_surv_att(t_count) = sum(att_surv);
    tot_surv_def(t_count) = sum(def_surv);
    tot_surv_HVU(t_count) = HVU_surv;
    t_count = t_count+1;
end
```

2. Weighted Dynamics and Attrition Model

a. Leonard Dynamics

```
function yout = new_ode4_Leon_WeightForces(t0,dt,tf,Cd,N_attackers, N_defenders, N_Bezier, PARAMETERS,x,~)
```

```
yout = x;
epsilon = 0.01; %distance cut off for near zero
umax = PARAMETERS.SWARM.umax;
K = PARAMETERS.SWARM.K;
K_hvu = PARAMETERS.SWARM.K_hvu;
d0_att=PARAMETERS.SWARM.d0;
d1_att=PARAMETERS.SWARM.d1;
alpha_att=PARAMETERS.SWARM.alpha_i;
d0_def=PARAMETERS.SWARM.INTd0;
d1_def=PARAMETERS.SWARM.INTd1;
alpha_def=PARAMETERS.SWARM.alphaINT_i;
mass_att = ones(1,N_attackers);
att_lamda = PARAMETERS.ATTACKERWEAPON.lambda;
att_sigma = PARAMETERS.ATTACKERWEAPON.sigma;
follower_states = reshape(x(1:6*N_attackers),6,N_attackers);
x_att=follower_states(1,:);
y_att=follower_states(2,:);
z_att=follower_states(3,:);
vx_att=follower_states(4,:);
vy_att=follower_states(5,:);
vz_att=follower_states(6,:);
ax_att = 0;
ay_att = 0;
az_att = 0;
ax_att_old = ax_att;
ay_att_old = ay_att;
az_att_old = az_att;
Q = x(6*N_attackers+1:7*N_attackers); %probability of attacker survival
Pd = x(7*N_attackers+1:7*N_attackers+N_defenders); %prob of def surv
P = x(end); %prob of HVU surv
for t = t0 : dt : tf-dt
 % first step in Verlet integration
   x_att=x_att+vx_att*dt+ax_att_old.*dt.^2/2;
   y_att=y_att+vy_att*dt+ay_att_old.*dt.^2/2;
   z_att=z_att+vz_att*dt+az_att_old.*dt.^2/2;
   BN_t = bernsteinMatrix_a2b(N_Bezier,t,t0,tf);
   pd = (BN_t*Cd);
   pd = reshape(pd,3,N_defenders);
   follower_states = [x_att;y_att;z_att;vx_att;vy_att;vz_att];
   position = [follower_states(1:3,:) pd];
   p_hvu = PARAMETERS.DEFENDER.p_hvu;
   Fx = zeros(1,N_attackers);
   Fy = Fx;
   Fz = Fx;
   for nn = 1:N_attackers
      dx_hvu = x_att(nn)-p_hvu(1);
      dy_hvu = y_att(nn)-p_hvu(2);
      dz_hvu = z_att(nn)-p_hvu(3);
      dd_hvu = sqrt(dx_hvu.^2+dy_hvu.^2+dz_hvu.^2);
      Fx(nn) = Fx(nn) - (K_hvu.*dx_hvu./dd_hvu + K.*vx_att(nn));
      Fy(nn) = Fy(nn) - (K_hvu.*dy_hvu./dd_hvu + K.*vy_att(nn));
      Fz(nn) = Fz(nn) - (K_hvu.*dz_hvu./dd_hvu + K.*vz_att(nn));
      for mm = (nn+1):N_attackers
         dx = x_att(nn) - x_att(mm);
         if dx < d1_att
            dy = y_att(nn) - y_att(mm);
            if dy < d1_att
               dz = z_att(nn) - z_att(mm);
               ddnm = max(sqrt(dx.^2+dy.^2+dz.^2),epsilon);
               if ddnm < d1_att
                  F = (1./ddnm.^2).*alpha_att.*(1-d0_att./ddnm);%;
```

```
                    Fx(nn) = Fx(nn) - Q(mm).*F.*dx;% weighted forces
                    Fx(mm) = Fx(mm) + Q(nn).*F.*dx;
                    Fy(nn) = Fy(nn) - Q(mm).*F.*dy;
                    Fy(mm) = Fy(mm) + Q(nn).*F.*dy;
                    Fz(nn) = Fz(nn) - Q(mm).*F.*dz;
                    Fz(mm) = Fz(mm) + Q(nn).*F.*dz;
                end
            end
        end
    end
end
x_def=pd(1,:);
y_def=pd(2,:);
z_def=pd(3,:);
distance_2defenders = d1_def*ones(N_attackers, N_defenders);
for nn = 1:N_attackers
    for mm = 1:N_defenders
        dx = x_att(nn) - x_def(mm);
        if dx < d1_def
            dy = y_att(nn) - y_def(mm);
            if dy < d1_def
                dz = z_att(nn) - z_def(mm);
                distance_2defenders(nn,mm) = sqrt(dx.^2+dy.^2+dz.^2);
                distance_2defenders(nn,mm) = max(sqrt(dx.^2+dy.^2+dz.^2),epsilon);
                if distance_2defenders(nn,mm) < d1_def
                    F = (1./(distance_2defenders(nn,mm)).^2).*alpha_def.*(1-d0_def./distance_2defenders(nn,mm));
                    Fx(nn) = Fx(nn) - Pd(mm).*F.*dx;
                    Fy(nn) = Fy(nn) - Pd(mm).*F.*dy;
                    Fz(nn) = Fz(nn) - Pd(mm).*F.*dz;
                end
            end
        end
    end
end
% Thrust limits
for i = 1:N_attackers
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if Fx(i) >= umax
        Fx(i) = umax;
    elseif Fx(i) <= -umax
        Fx(i) = -umax;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if Fy(i) >= umax
        Fy(i) = umax;
    elseif Fy(i) <= -umax
        Fy(i) = -umax;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if Fz(i) >= umax
        Fz(i) = umax;
    elseif Fz(i) <= -umax
        Fz(i) = -umax;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
%  compute probability of attacker/defender/hvu survival
rda2d = PARAMETERS.DEFENDERWEAPON.lambda*normcdf((PARAMETERS.DEFENDERWEAPON.F - 
PARAMETERS.DEFENDERWEAPON.a*distance_2defenders.^2)/PARAMETERS.DEFENDERWEAPON.sigma,0,
1);
Pdmat = repmat(Pd',[N_attackers 1]);
```

```
    Q = Q.*(1-(1-prod(1-(rda2d.*Pdmat)'.*dt)))';
    % compute probability of defender survival
    rdd2a = att_lamda*normcdf((PARAMETERS.ATTACKERWEAPON.F -
PARAMETERS.ATTACKERWEAPON.a*distance_2defenders.^2)/att_sigma,0,1);
    Qmat = repmat(Q,[1 N_defenders]);
    Pd = Pd.*(1-(1-prod(1-(rdd2a.*Qmat).*dt)))';
    % compute probability of HVU survival
    distance_2HVU_sq = (position(1,1:N_attackers) - PARAMETERS.DEFENDER.p_hvu(1)).^2 +
(position(2,1:N_attackers) - PARAMETERS.DEFENDER.p_hvu(2)).^2 + (position(3,1:N_attackers) -
PARAMETERS.DEFENDER.p_hvu(3)).^2;
    ra2hvu = att_lamda*normcdf((PARAMETERS.ATTACKERWEAPON.F -
PARAMETERS.ATTACKERWEAPON.a*distance_2HVU_sq)/att_sigma,0,1);
    P = P.*(1-(1-prod(1-ra2hvu.*dt)));
    ax_att = Fx./mass_att;
    ay_att = Fy./mass_att;
    az_att = Fz./mass_att;
    vx_att = vx_att + (ax_att_old + ax_att).*dt/2;
    vy_att = vy_att + (ay_att_old + ay_att).*dt/2;
    vz_att = vz_att + (az_att_old + az_att).*dt/2;
    ax_att_old = ax_att;
    ay_att_old = ay_att;
    az_att_old = az_att;
    dy = [reshape([x_att; y_att; z_att; vx_att; vy_att; vz_att],6*N_attackers,1);Q;Pd;P];
    yout = [yout; dy];
end
```

b. Reynolds Dynamics

```
function yout = new_ode4_Reyn_WeightForces(t0,dt,tf,Cd,N_attackers, N_defenders, N_Bezier, PARAMETERS,x)
yout = x;
epsilon = 0.01; %distance cut off for near zero
umax = PARAMETERS.SWARM.umax;
K = PARAMETERS.SWARM.K;
K_hvu = PARAMETERS.SWARM.K_hvu;
d_alig=PARAMETERS.SWARM.N_size_al_F;
weight_alig = PARAMETERS.SWARM.w_al_F;
% d_cohes=PARAMETERS.SWARM.N_size_c_F;
weight_cohes=PARAMETERS.SWARM.w_c_F;
d_separ=PARAMETERS.SWARM.N_size_s_I;
weight_separ = PARAMETERS.SWARM.w_s_F;
weight_separ_intr = PARAMETERS.SWARM.w_s_I;
mass_att = ones(1,N_attackers);
follower_states = reshape(x(1:6*N_attackers),6,N_attackers); %% x y z vx vy vz
x_att=follower_states(1,:);
y_att=follower_states(2,:);
z_att=follower_states(3,:);
vx_att=follower_states(4,:);
vy_att=follower_states(5,:);
vz_att=follower_states(6,:);
ax_att = 0;
ay_att = 0;
az_att = 0;
ax_att_old = ax_att;
ay_att_old = ay_att;
az_att_old = az_att;
Q = x(6*N_attackers+1:7*N_attackers); %probability of attacker survival
Pd = x(7*N_attackers+1:7*N_attackers+N_defenders); %prob of def surv
P = x(end);
for t = t0 : dt : tf-dt
    % first step in Verlet integration
```

```
x_att=x_att+vx_att*dt+ax_att_old.*dt.^2/2;
y_att=y_att+vy_att*dt+ay_att_old.*dt.^2/2;
z_att=z_att+vz_att*dt+az_att_old.*dt.^2/2;
BN_t = bernsteinMatrix_a2b(N_Bezier,t,t0,tf);
pd = (BN_t*Cd);
pd = reshape(pd,3,N_defenders); % pd - position of the defenders
follower_states = [x_att;y_att;z_att;vx_att;vy_att;vz_att];
position = [follower_states(1:3,:) pd]; % position: 3-by-(n+m)
velocity = [follower_states(4:6,:) vd]; % velocity: 3-by-(n+m)
p_hvu = PARAMETERS.DEFENDER.p_hvu;
Fx = zeros(1,N_attackers);
Fy = Fx;
Fz = Fx;
counter_al = zeros(1,N_attackers);
sumx_al = counter_al;
sumy_al = counter_al;
sumz_al = counter_al;
sumx_coh = counter_al;
sumy_coh = counter_al;
sumz_coh = counter_al;
counter_sep = zeros(1,N_attackers);
sumx_sep = counter_sep;
sumy_sep = counter_sep;
sumz_sep = counter_sep;
x_def=pd(1,:);
y_def=pd(2,:);
z_def=pd(3,:);
counter_sep_int = zeros(1,N_attackers);
sumx_sep_int = counter_sep_int;
sumy_sep_int = counter_sep_int;
sumz_sep_int = counter_sep_int;
distance_2defenders = d_separ*ones(N_attackers, N_defenders);
for nn = 1:N_attackers
    dx_hvu = x_att(nn)-p_hvu(1);
    dy_hvu = y_att(nn)-p_hvu(2);
    dz_hvu = z_att(nn)-p_hvu(3);
    dd_hvu = sqrt(dx_hvu.^2+dy_hvu.^2+dz_hvu.^2);
    %Artificial Potential forces
    Fx(nn) = Fx(nn) - (K_hvu.*dx_hvu./dd_hvu + K.*vx_att(nn));
    Fy(nn) = Fy(nn) - (K_hvu.*dy_hvu./dd_hvu + K.*vy_att(nn));
    Fz(nn) = Fz(nn) - (K_hvu.*dz_hvu./dd_hvu + K.*vz_att(nn));
    for mm = (nn+1):N_attackers
        dx = x_att(nn) - x_att(mm);
        if dx < d_alig
            dy = y_att(nn) - y_att(mm);
            if dy < d_alig
                dz = z_att(nn) - z_att(mm);
                ddnm = sqrt(dx.^2+dy.^2+dz.^2);
                if (ddnm < d_separ && ddnm > epsilon)
                    counter_sep(nn) = counter_sep(nn) +1;
                    counter_sep(mm) = counter_sep(mm) +1;
                    sumx_sep(nn) = sumx_sep(nn) + (dx/ddnm);
                    sumx_sep(mm) = sumx_sep(mm) - (dx/ddnm);
                    sumy_sep(nn) = sumy_sep(nn) + (dy/ddnm);
                    sumy_sep(mm) = sumy_sep(mm) - (dy/ddnm);
                    sumz_sep(nn) = sumz_sep(nn) + (dz/ddnm);
                    sumz_sep(mm) = sumz_sep(mm) - (dz/ddnm);
                    counter_al(nn) = counter_al(nn) +1;
                    counter_al(mm) = counter_al(mm) +1;
                    sumx_al(nn) = sumx_al(nn) + vx_att(mm);
                    sumx_al(mm) = sumx_al(mm) + vx_att(nn);
```

```
                sumy_al(nn) = sumy_al(nn) + vy_att(mm);
                sumy_al(mm) = sumy_al(mm) + vy_att(nn);
                sumz_al(nn) = sumz_al(nn) + vz_att(mm);
                sumz_al(mm) = sumz_al(mm) + vz_att(nn);
                sumx_coh(nn) = sumx_coh(nn) + x_att(mm);
                sumx_coh(mm) = sumx_coh(mm) + x_att(nn);
                sumy_coh(nn) = sumy_coh(nn) + y_att(mm);
                sumy_coh(mm) = sumy_coh(mm) + y_att(nn);
                sumz_coh(nn) = sumz_coh(nn) + z_att(mm);
                sumz_coh(mm) = sumz_coh(mm) + z_att(nn);
            elseif (ddnm < d_alig  && ddnm > epsilon)
                counter_al(nn) = counter_al(nn) +1;
                counter_al(mm) = counter_al(mm) +1;
                sumx_al(nn) = sumx_al(nn) + vx_att(mm);
                sumx_al(mm) = sumx_al(mm) + vx_att(nn);
                sumy_al(nn) = sumy_al(nn) + vy_att(mm);
                sumy_al(mm) = sumy_al(mm) + vy_att(nn);
                sumz_al(nn) = sumz_al(nn) + vz_att(mm);
                sumz_al(mm) = sumz_al(mm) + vz_att(nn);
                sumx_coh(nn) = sumx_coh(nn) + x_att(mm);
                sumx_coh(mm) = sumx_coh(mm) + x_att(nn);
                sumy_coh(nn) = sumy_coh(nn) + y_att(mm);
                sumy_coh(mm) = sumy_coh(mm) + y_att(nn);
                sumz_coh(nn) = sumz_coh(nn) + z_att(mm);
                sumz_coh(mm) = sumz_coh(mm) + z_att(nn);
            end
          end
        end
      end
      for mm = 1:N_defenders
        dx = x_att(nn) - x_def(mm);
         if dx < d_separ
          dy = y_att(nn) - y_def(mm);
          if dy < d_separ
            dz = z_att(nn) - z_def(mm);
            distance_2defenders(nn,mm) = sqrt(dx.^2+dy.^2+dz.^2);
            if (distance_2defenders(nn,mm) < d_separ  && distance_2defenders(nn,mm) > epsilon)
              counter_sep_int(nn) = counter_sep_int(nn) +1;
              sumx_sep_int(nn) = sumx_sep_int(nn) + Pd(mm).*(dx./distance_2defenders(nn,mm));
              sumy_sep_int(nn) = sumy_sep_int(nn) + Pd(mm).*(dy./distance_2defenders(nn,mm));
              sumz_sep_int(nn) = sumz_sep_int(nn) + Pd(mm).*(dz./distance_2defenders(nn,mm));
            end
          end
        end
      end
    end
    for nn = 1:N_attackers
      if counter_al(nn) ==0
        counter_al(nn)=1;
      end
      if counter_sep(nn) ==0
        counter_sep(nn)=1;
      end
      if counter_sep_int(nn)==0
        counter_sep_int(nn)=1;
      end
     %Alignment Forces &Cohesion Forces
      Fx = Fx +Q(nn).*mass_att.*(weight_separ.*(sumx_sep./counter_sep) + weight_alig.*(sumx_al./counter_al -
vx_att(nn)) + weight_cohes.*(sumx_coh./counter_al - x_att(nn)));
      Fy = Fy +Q(nn).*mass_att.*(weight_separ.*(sumy_sep./counter_sep) + weight_alig.*(sumy_al./counter_al -
vy_att(nn)) + weight_cohes.*(sumy_coh./counter_al - y_att(nn)));
```

```
        Fz = Fz +Q(nn).*mass_att.*(weight_separ.*(sumz_sep./counter_sep) + weight_alig.*(sumz_al./counter_al -
vz_att(nn)) + weight_cohes.*(sumz_coh./counter_al - z_att(nn)));
        %Separation Forces
        Fx = Fx + mass_att.*(weight_separ_intr.*(sumx_sep_int./counter_sep_int));
        Fy = Fy + mass_att.*(weight_separ_intr.*(sumy_sep_int./counter_sep_int));
        Fz = Fz + mass_att.*(weight_separ_intr.*(sumz_sep_int./counter_sep_int));
    end
    % Thrust limits
    for i = 1:N_attackers
        if Fx(i) >= umax
            Fx(i) = umax;
        elseif Fx(i) <= -umax
            Fx(i) = -umax;
        end
        if Fy(i) >= umax
            Fy(i) = umax;
        elseif Fy(i) <= -umax
            Fy(i) = -umax;
        end
        if Fz(i) >= umax
            Fz(i) = umax;
        elseif Fz(i) <= -umax
            Fz(i) = -umax;
        end
    end
    %  compute probability of attacker/defender/hvu survival
    rda2d = PARAMETERS.DEFENDERWEAPON.lambda*normcdf((PARAMETERS.DEFENDERWEAPON.F -
PARAMETERS.DEFENDERWEAPON.a*distance_2defenders.^2)/PARAMETERS.DEFENDERWEAPON.sigma,0,
1);
    Pdmat = repmat(Pd',[N_attackers 1]);
    Q = Q.*(1-(1-prod(1-(rda2d.*Pdmat)'.*dt)))';
    % compute probability of defender survival
    rdd2a = PARAMETERS.ATTACKERWEAPON.lambda*normcdf((PARAMETERS.ATTACKERWEAPON.F -
PARAMETERS.ATTACKERWEAPON.a*distance_2defenders.^2)/PARAMETERS.ATTACKERWEAPON.sigma,0,
1);
    Qmat = repmat(Q,[1 N_defenders]);
    Pd = Pd.*(1-(1-prod(1-(rdd2a.*Qmat).*dt)))';
    % compute probability of HVU survival
    distance_2HVU_sq = (position(1,1:N_attackers) - PARAMETERS.DEFENDER.p_hvu(1)).^2 +
(position(2,1:N_attackers) - PARAMETERS.DEFENDER.p_hvu(2)).^2 + (position(3,1:N_attackers) -
PARAMETERS.DEFENDER.p_hvu(3)).^2;
    ra2hvu = PARAMETERS.ATTACKERWEAPON.lambda*normcdf((PARAMETERS.ATTACKERWEAPON.F -
PARAMETERS.ATTACKERWEAPON.a*distance_2HVU_sq)/PARAMETERS.ATTACKERWEAPON.sigma,0,1);
    P = P.*(1-(1-prod(1-ra2hvu.*dt)));
    ax_att = Fx./mass_att;
    ay_att = Fy./mass_att;
    az_att = Fz./mass_att;
    vx_att = vx_att + (ax_att_old + ax_att).*dt/2;
    vy_att = vy_att + (ay_att_old + ay_att).*dt/2;
    vz_att = vz_att + (az_att_old + az_att).*dt/2;
    ax_att_old = ax_att;
    ay_att_old = ay_att;
    az_att_old = az_att;
    dy = [reshape([x_att; y_att; z_att; vx_att; vy_att; vz_att],6*N_attackers,1);Q;Pd;P];
    yout = [yout; dy];
end
```

3. Threshold Dynamics and Attrition Models

a. Leonard Dynamics

```
function yout = new_ode4_Leon_ThreshForces_v2(t0,dt,tf,Cd,N_attackers, N_defenders, N_Bezier,
PARAMETERS,x,~)
yout = x;
epsilon = 0.01; %distance cut off for near zero
umax = PARAMETERS.SWARM.umax;
K = PARAMETERS.SWARM.K;
K_hvu = PARAMETERS.SWARM.K_hvu;
d0_att=PARAMETERS.SWARM.d0;
d1_att=PARAMETERS.SWARM.d1;
alpha_att=PARAMETERS.SWARM.alpha_i;
d0_def=PARAMETERS.SWARM.INTd0;
d1_def=PARAMETERS.SWARM.INTd1;
alpha_def=PARAMETERS.SWARM.alphaINT_i;
mass_att = ones(1,N_attackers);
att_lamda = PARAMETERS.ATTACKERWEAPON.lambda;
att_sigma = PARAMETERS.ATTACKERWEAPON.sigma;
follower_states = reshape(x(1:6*N_attackers),6,N_attackers); %% x y z vx vy vz
x_att=follower_states(1,:);
y_att=follower_states(2,:);
z_att=follower_states(3,:);
vx_att=follower_states(4,:);
vy_att=follower_states(5,:);
vz_att=follower_states(6,:);
ax_att = 0;
ay_att = 0;
az_att = 0;
ax_att_old = ax_att;
ay_att_old = ay_att;
az_att_old = az_att;
Q = x(6*N_attackers+1:7*N_attackers); %probability of attacker survival
Pd = x(7*N_attackers+1:7*N_attackers+N_defenders); %prob of def surv
P = x(end); %prob of HVU surv
for t = t0 : dt : tf-dt
  % first step in Verlet integration
   x_att=x_att+vx_att*dt+ax_att_old.*dt.^2/2;
   y_att=y_att+vy_att*dt+ay_att_old.*dt.^2/2;
   z_att=z_att+vz_att*dt+az_att_old.*dt.^2/2;
   BN_t = bernsteinMatrix_a2b(N_Bezier,t,t0,tf);
   pd = (BN_t*Cd);
   pd = reshape(pd,3,N_defenders); % pd - position of the defenders
   follower_states = [x_att;y_att;z_att;vx_att;vy_att;vz_att];
   position = [follower_states(1:3,:) pd];
   p_hvu = PARAMETERS.DEFENDER.p_hvu;
   Fx = zeros(1,N_attackers);
   Fy = Fx;
   Fz = Fx;
   for nn = 1:N_attackers
      if Q(nn)>= 0.5
         dx_hvu = x_att(nn)-p_hvu(1);
         dy_hvu = y_att(nn)-p_hvu(2);
         dz_hvu = z_att(nn)-p_hvu(3);
         dd_hvu = sqrt(dx_hvu.^2+dy_hvu.^2+dz_hvu.^2);
         Fx(nn) = Fx(nn) - K_hvu.*dx_hvu./dd_hvu - K.*vx_att(nn);
         Fy(nn) = Fy(nn) - K_hvu.*dy_hvu./dd_hvu - K.*vy_att(nn);
         Fz(nn) = Fz(nn) - K_hvu.*dz_hvu./dd_hvu - K.*vz_att(nn);
      end
```

```
    for mm = (nn+1):N_attackers
        dx = x_att(nn) - x_att(mm);
        if dx < d1_att
            dy = y_att(nn) - y_att(mm);
            if dy < d1_att
                dz = z_att(nn) - z_att(mm);
                ddnm = max(sqrt(dx.^2+dy.^2+dz.^2),epsilon);
                if ddnm < d1_att
                    if Q(mm)>= 0.5
                        F = (1./ddnm.^2).*alpha_att.*(1-d0_att./ddnm);
                        Fx(nn) = Fx(nn) - F.*dx;
                        Fx(mm) = Fx(mm) + F.*dx;
                        Fy(nn) = Fy(nn) - F.*dy;
                        Fy(mm) = Fy(mm) + F.*dy;
                        Fz(nn) = Fz(nn) - F.*dz;
                        Fz(mm) = Fz(mm) + F.*dz;
                    end
                end
            end
        end
    end
end
% c. Defender forces computation
x_def=pd(1,:);
y_def=pd(2,:);
z_def=pd(3,:);
distance_2defenders = d1_def*ones(N_attackers, N_defenders);
for nn = 1:N_attackers
    for mm = 1:N_defenders
        dx = x_att(nn) - x_def(mm);
        if dx < d1_def
            dy = y_att(nn) - y_def(mm);
            if dy < d1_def
                dz = z_att(nn) - z_def(mm);
                distance_2defenders(nn,mm) = sqrt(dx.^2+dy.^2+dz.^2);
                distance_2defenders(nn,mm) = max(sqrt(dx.^2+dy.^2+dz.^2),epsilon);
                if distance_2defenders(nn,mm) < d1_def
                    if Pd(mm)>=0.5
                        F = (1./(distance_2defenders(nn,mm)).^2).*alpha_def.*(1-d0_def./distance_2defenders(nn,mm));
                        Fx(nn) = Fx(nn) - F.*dx;
                        Fy(nn) = Fy(nn) - F.*dy;
                        Fz(nn) = Fz(nn) - F.*dz;
                    end
                end
            end
        end
    end
end
% Thrust limits
for i = 1:N_attackers
    if Fx(i) >= umax
        Fx(i) = umax;
    elseif Fx(i) <= -umax
        Fx(i) = -umax;
    end
    if Fy(i) >= umax
        Fy(i) = umax;
    elseif Fy(i) <= -umax
        Fy(i) = -umax;
    end
    if Fz(i) >= umax
```

```
        Fz(i) = umax;
      elseif Fz(i) <= -umax
        Fz(i) = -umax;
      end
    end
    %THRESHOLD ATTRITION MODEL IMPLEMENTATION
%  compute probability of attacker/defender/hvu survival
    rda2d = PARAMETERS.DEFENDERWEAPON.lambda*normcdf((PARAMETERS.DEFENDERWEAPON.F -
PARAMETERS.DEFENDERWEAPON.a*distance_2defenders.^2)/PARAMETERS.DEFENDERWEAPON.sigma,0,
1);
   %  Pdmat = repmat(Pd',[N_attackers 1]);
   for mm = 1:N_defenders
      if Pd(mm)<0.5
        rda2d(:,mm) = 0;
      end
   end
    Q = Q.*(1-(1-prod(1-(rda2d)'.*dt)))';
    % compute probability of defender survival
    rdd2a = att_lamda*normcdf((PARAMETERS.ATTACKERWEAPON.F -
PARAMETERS.ATTACKERWEAPON.a*distance_2defenders.^2)/att_sigma,0,1);
    % Qmat = repmat(Q,[1 N_defenders]);
    for  nn = 1:N_attackers
      if Q(nn)< 0.5
        rdd2a(nn,:) = 0;
      end
    end
    Pd = Pd.*(1-(1-prod(1-(rdd2a).*dt)))';
    % compute probability of HVU survival
    distance_2HVU_sq = (position(1,1:N_attackers) - PARAMETERS.DEFENDER.p_hvu(1)).^2 +
(position(2,1:N_attackers) - PARAMETERS.DEFENDER.p_hvu(2)).^2 + (position(3,1:N_attackers) -
PARAMETERS.DEFENDER.p_hvu(3)).^2;
    ra2hvu = att_lamda*normcdf((PARAMETERS.ATTACKERWEAPON.F -
PARAMETERS.ATTACKERWEAPON.a*distance_2HVU_sq)/att_sigma,0,1);
    for  nn = 1:N_attackers
      if Q(nn)< 0.5
        ra2hvu(1,nn) = 0;
      end
    end
    P = P.*(1-(1-prod(1-ra2hvu.*dt)));
    ax_att = Fx./mass_att;
    ay_att = Fy./mass_att;
    az_att = Fz./mass_att;
    vx_att = vx_att + (ax_att_old + ax_att).*dt/2;
    vy_att = vy_att + (ay_att_old + ay_att).*dt/2;
    vz_att = vz_att + (az_att_old + az_att).*dt/2;
    ax_att_old = ax_att;
    ay_att_old = ay_att;
    az_att_old = az_att;
    dy = [reshape([x_att; y_att; z_att; vx_att; vy_att; vz_att],6*N_attackers,1);Q;Pd;P];
    yout = [yout; dy];
end
```

B. COST FUNCTIONS

1. HVU PROTECTION

```
function J = costFunc_singleswarm(x,x_init,t0,h,tf,N_Bezier, N_attackers, N_defenders,  N_omega, PARAMETERS)
Cd = reshape(x,[(N_Bezier+1),N_defenders*PARAMETERS.DEFENDER.Nx]);
P_new = zeros(1, N_omega);
parfor i = 1:N_omega
```

```
    x_real = PARAMETERS.ode_func(t0,h,tf,Cd,N_attackers, N_defenders, N_Bezier, PARAMETERS,x_init{i}, i);
    P_new(i) = x_real(end);
end
Pnew = (sum(1 - P_new)/N_omega);
J = Pnew;
end
```

2. AIR SUPERIORITY

```
function J = costFunc_min_att(x,x_init,t0,h,tf,N_Bezier, N_attackers, N_defenders,  N_omega, PARAMETERS)
Cd = reshape(x,[(N_Bezier+1),N_defenders*PARAMETERS.DEFENDER.Nx]);
length_time = (tf-t0)/h+1;
P_new = zeros(1, N_omega);
for i = 1:N_omega
    x_real = PARAMETERS.ode_func(t0,h,tf,Cd,N_attackers, N_defenders, N_Bezier, PARAMETERS,x_init{i}, i);
    temp1 = reshape(x_real,7*N_attackers+N_defenders+1,length_time);
    att_surv = temp1(6*N_attackers+1:7*N_attackers,:);
    P_new(i) = mean(att_surv(:,end));
end
Pnew = (sum(P_new)/N_omega);
J = Pnew;
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] N. Wiener, “Men, Machines, and the World about.,” N.Y. Int. Univ. Press, pp. 793–799, 1954.

[2] Department Of Defense, “Unmanned Systems Roadmap 2007–2032,” Washington, DC, USA, Technical, 2007. Accessed: Apr. 12, 2020. [Online]. Available: http://www.dtic.mil/docs/citations/ADA475002.

[3] C. Walton, P. Lambrianides, I. Kaminer, J. Royset, and Q. Gong, “Optimal motion planning in rapid-fire combat situations with attacker uncertainty,” *Nav. Res. Logist. NRL*, vol. 65, no. 2, pp. 101–119, Mar. 2018, doi: 10.1002/nav.21790.

[4] R. T. Farouki, “The Bernstein polynomial basis: A centennial retrospective,” *Comput. Aided Geom. Des.*, vol. 29, no. 6, pp. 379–419, 2012, doi: 10.1016/j.cagd.2012.03.001.

[5] V. Cichella, I. Kaminer, C. Walton, and N. Hovakimyan, “Optimal Motion Planning for Differentially Flat Systems Using Bernstein Approximation,” *IEEE Control Syst. Lett.*, vol. 2, no. 1, pp. 181–186, Jan. 2018, doi: 10.1109/LCSYS.2017.2778313.

[6] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, “A fast procedure for computing the distance between complex objects in three-dimensional space,” *IEEE J. Robot. Autom.*, vol. 4, no. 2, pp. 193–203, Apr. 1988, doi: 10.1109/56.2083.

[7] V. Cichella, I. Kaminer, C. Walton, N. Hovakimyan, and A. Pascoal, “Bernstein approximation of optimal control problems,” *arXiv.org*, 2018, Accessed: Apr. 12, 2020. [Online]. Available: http://search.proquest.com/docview/2158090969/?pq-origsite=primo.

[8] V. Cichella, I. Kaminer, C. Walton, N. Hovakimyan, and A. M. Pascoal, “Consistent approximation of optimal control problems using Bernstein polynomials,” in *2019 IEEE 58th Conference on Decision and Control (CDC)*, Dec. 2019, pp. 4292–4297, doi: 10.1109/CDC40024.2019.9029677.

[9] N. E. Leonard and E. Fiorelli, “Virtual leaders, artificial potentials and coordinated control of groups,” in *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, Dec. 2001, vol. 3, pp. 2968–2973 vol.3, doi: 10.1109/CDC.2001.980728.

[10] P. Ogren, E. Fiorelli, and N. E. Leonard, “Cooperative Control of Mobile Sensor Networks: Adaptive Gradient Climbing in a Distributed Environment,” *IEEE Trans. Autom. Control*, vol. 49, no. 8, pp. 1292–1302, Aug. 2004, doi: 10.1109/TAC.2004.832203.

[11] C. W. Reynolds, “Flocks, herds and schools: A distributed behavioral model,” *ACM SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 25–34, 1987, doi: 10.1145/37402.37406.

[12] U. Mehmood *et al.*, “Declarative vs Rule-based Control for Flocking Dynamics,” *ArXiv171010013 Cs*, Oct. 2017, Accessed: Apr. 10, 2020. [Online]. Available: http://arxiv.org/abs/1710.10013.

[13] N. Krizou, “Nontrivial Power-Law Scaling of Peak Forces during Granular Impact,” M.S. thesis, Monterey, CA; Naval Postgraduate School, 2019.

[14] “Most Unmanned Aerial Vehicles (UAVs) airborne simultaneously,” *Guinness World Records*. https://www.guinnessworldrecords.com/world-records/373319-most%C2%A0unmanned-aerial%C2%A0vehicles%C2%A0uavs-airborne%C2%A0simultaneously%C2%A05-kg-or-less (accessed Apr. 29, 2020).

[15] A. Washburn, *Combat Modeling*, 1st ed. 2009. New York, NY: Springer U.S., 2009.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California