Guidance and Control of a Spacecraft Swarm with Limited Knowledge

A. Ripoll Sanchez

Technische Universiteit Delft

TU Delft

Guidance and Control of a Spacecraft Swarm with Limited Knowledge

by

A. Ripoll Sanchez

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday April 30, 2019 at 9:30 AM.

Student number:	4631862	
Project duration:	July 17, 2018 – April 30, 2019	
Thesis committee:	Prof. Dr. E. K. A. Gill,	TU Delft, Committee Chair
	Dr. J. Guo,	TU Delft, Supervisor
	Dr. G.C.H.E. de Croon	TU Delft, External Member
	Ir. M. Coppola,	TU Delft, Supervisor

TU Delft

To Andrés and Antonio

Acknowledgements

The submission and presentation of this thesis is the conclusion of my personal journey to become an aerospace engineer. This has been my dream since I was a kid, and it has not been easy. Many challenges and struggles have been part of this journey. Upon the end of this journey, I would like to thank all those people that have been next to me and have helped me to achieve this dream.

This sophisticated nature of this project is the result of acquired knowledge and practical skills. This has been possible thanks to the strive for excellence and hard work put by the Space Engineering Department of the TU Delft. The MSc thesis would have never taken place without the help of my supervisors, Jian Guo, and Mario Coppola. I will like to thank Mario for all his advice, help, support, career orientation and willingness to be part of this project until the very end and bring it to fruition. I will also like to thank Jian for all his creativity, support and understanding in the worst phases of the project.

This journey would not have been the same either without the joy and tears, good moments and bad times with the people that surrounded me for this past two years. Thanks to Paula for bringing her unconditional support and helping me achieve my goals without losing my mental health. Also I would like to thank my peers through this journey: Eric, Jose, Pablo, Albert, Jack, Mauricio, María, Virginia, Alejandro, Dani, Lorenzo, José Ángel, Timo, Arthur, Rodrigo, Victoria, Tim, Alba, Adrian, Alex, Adri, Patricia Susana, Pablo R., Pablo B., Pedro, Marita, Dionis, Blanca, Jorge and many others. Thanks to our chats, your advice and your help this journey has been the most pleasant one, even if it was the hardest.

Finally, I would like to thank my family. To my sisters, Lidia and Cristina and, especially, thanks to my parents, Andrés and Beatriz. Thanks for all their support, both economically and morally. I could have never asked for a better role model. And a final thanks to my grandparents, for teaching me the value of knowledge and the love for science, and for being the inspiration that made me continue in the moments of doubt.

A. Ripoll Sanchez
Delft, April 2019

Preface

There is no doubt that the smallest spacecraft platforms (CubeSats, NanoSats and microsats) are revolutionizing the market. The possibilities of developing spacecraft through standard platforms, commercial of the shelf components and then reduced the cost of development has allowed new organizations to enter the space section. On top of that, traditional businesses have been benefited by these type of new spacecraft, as more technology demonstrations and riskier missions have been able to be developed through this technology.

Nevertheless, these spacecraft systems cannot compete with their larger counterparts due to their size limitations. The extra space will allow for larger spacecraft to accommodate more complex and precise instruments and payloads. Therefore, an individual CubeSat will never be able to compete with a larger system. Nevertheless, in the last two decades, the concept of using the combined power of these simple and cheap systems to perform a task only possible with larger spacecraft has been proposed. One of the most interesting possibilities among the use of these distributed space systems is the use of swarming. Swarming is a technique which consists of the combination of a large number of simple elements to allow for the emergence of properties that allow the whole system to act as more than just its parts. The benefits of spacecraft swarming are undeniable, as inherently swarming brings robustness, flexibility, and scalability to the system, all three properties extremely valued in space missions.

For the moment spacecraft swarming has been mostly based on generating formation patterns with a large degree of autonomy. Still, only a handful of techniques have been explored. This work aims to further advance the knowledge of spacecraft swarming techniques. To do so, the implementation of a swarming algorithm never tested in space will be studied The advantage of this algorithm is double. First, as all swarming techniques, it presents a certain degree of artificial intelligence that will allow the swarm to cope with unexpected situations. Second, by design, this algorithm requires really little knowledge of the swarm environment. This translates in less need for information processing between spacecraft and within each spacecraft. Since the application is focused on already spacecraft with limited capabilities, this is certainly advantageous.

This project will reflect the research work done at the TU Delft's department of Space Systems Engineering on the use of the mentioned algorithm. This project was carried as the master thesis of the author. The aim will be to determine the advantages, disadvantages, and range of application of the algorithm. The work will be specially focused small spacecraft with limited actuation capacity. To study the algorithm a simulation tool has been created to simulate the dynamics and control of a swarm of spacecraft. This tool has been developed under a certain set of models and assumptions which will be presented throughout the report. The tool will be used to test the algorithm with several relevant spacecraft configurations to have a full overview of the performance of the algorithm. Also, optimization techniques have been used to improve the performance and have a clear picture of the current capabilities of the swarming algorithm. The tool created has been designed in such a way that future users can also incorporate their own modules of control, swarming or spacecraft design and test new swarming techniques, transcending its use on this project.

To fully understand the work here presented, it is recommended that the reader possesses graduate-level knowledge on orbital dynamics and control, as well as some undergraduate notions of computer programming and artificial intelligence techniques. If the reader lacks knowledge on any of the mentioned areas, it is recommended to read the associated literature study referenced in the bibliography.

A. Ripoll Sanchez
Delft, April 2019

Contents

List of Figures

List of Tables

I

Introduction: Swarming in Space

1 Introduction

Clearly miniaturization and size reduction has become a trend in the spacecraft world. Small spacecraft, especially microspacecraft and smaller systems (less than 100kg) [36] have the advantage of allowing for a faster development and, therefore lower cost. A good example of these kind of systems are the CubeSats. Since its introduction in the 1990's by Twigs and Puig Suari, CubeSat have revolutionized the space industry by allowing small capital organizations to access the possibility of launching their own space mission [60]. CubeSats have not only changed the whole development cycle of a spacecraft mission by switching the concept from tailored components to the use of Commercial of the Shelf (COTS) and standardized components, but also pushed space science to make the most of miniaturized elements. Still, the smallest spacecrafts present the major disadvantage of not being able to compete against the larger size instruments on board of larger size spacecraft individually [96]. In order to fully exploit the capabilities of these technologies a leap forward to collaborative spacecraft missions is needed. This is why probably, the smaller the size of the spacecraft, the more missions on collaborative approaches [36]. Among the many options presented in literature for collaborative spacecraft missions [41] one of the most interesting for its application in small spacecraft technologies is swarming. Swarming is a technique inherited from robotics which consists in the collaboration of a set of relatively simple elements (from now on agents) in such a way that the behaviors and actions resulting from the collaboration allow the emergence of more complex behaviors and actions than the ones that the agents by themselves could generate [42].

Swarming allows for the exploiting of the commercial advantages of less capable spacecraf while adding on the mission its three intrinsic properties: robustness, flexibility and scalability [11]. Furthermore, the emergence of properties due to swarming allows for the increase in the capabilities of the overall mission while still keeping the simplicity of the agents of the swarm. All these advantages, plus some others such as the cost reduction associated with the mass production of the agents, have increased the interest in spacecraft swarming in the last two decades. Both concepts for large budget interplanetary missions [38, 83] as well as lower budget university founded projects [27, 41, 47] based on swarming have been proposed in the last two decades. But it seems that there is no consensus on how to implement the guidance and control algorithms that allow the swarming of the spacecraft. Whilst the most classical approaches such as behavioral control seem to be preferred, they still present some inherited problems in scalability and required knowledge that cannot easily be solved for spacecraft missions. On the other hand, approaches based on advanced techniques such as deep learning still present a lack of understanding that makes them unsafe to use in a mission [16]. There is still a clear need for a reliable swarming guidance and control algorithm for spacecraft missions.

The goal of this thesis project will be to present a mode to form patters with a swarm spacecraft in a realistic space dynamic setting. This chapter intends to present first a small review of the state of the art of the given topic so all the concepts that will be needed to fully understand both the report and the motivation of the project become familiar to the reader. Then a short introduction of the method of swarming presented in this report will be also given in slightly more depth. The next section will then present a selected study case that will be used throughout the report to obtain and work with

realistic constraints. Finally, the last section of this chapter will be dedicated to present the reader with the project statement and objectives of this project, which will be the formal presentation of the report goals and its justification.

1.1. Swarming in Spacecraft-State of the Art

The current state of the art of spacecraft swarming algorithms have been focused in designing algorithms that allow to establish a desired pattern through two major techniques, behavioral algorithms and machine learning based controllers. Additionally, in the last couple of years some interesting concepts with space application have been rising from the literature in swarm robotics such as the use of Finite State Machines (FSMs) [2] for the guidance and control of large swarms. Finally, there has been some interest in including the use of optimization techniques in the design of the swarm controllers, especially with the use of evolutionary algorithms. Even though this has not been popularized yet in spacecraft swarming, some interesting applications have been proposed in literature which lead to believe that evolutionary robotics will soon be an integral part of spacecraft swarming.

1.1.1. Behavioral Algorithms

Behavioral algorithms are the most common choice in swarm guidance and control design. It consists in the definition of a set of mathematical functions which generate a set of behaviors that every single agent follows. The combination of those behaviors and the interaction of the swarm agents with each other then gives rise to the more complex global behavior of the swarm. An easy example of this could be the design of a flight in a spacecraft swarm where all agents must stay close to a reference orbit without colliding. This could then be translated into a local gather behavior of each agent towards the reference orbit to keep the agent close to the reference orbit. Then a repel behavior could be added so each agent does not collide with its neighbors.

Within behavioral algorithms the most popular implementation in space is the subset called Artificial Potential Fields (APF). Due to their popularity some authors even define them as a different subset of controllers [76]. Nevertheless given how they fit the general description of behavioral controllers, this report will follow the classification in [11, 16] and consider them a subset of behavioral controllers. These controllers are based on defining an artificial potential function for each behavior required by the agents of the swarm in such a way that the required global behaviors of the swarm represent the minimums of the global potential of the agent (that is, including the real dynamics on top of the artificial). Then, by adding all these artificial potentials to the dynamic model implemented it is quite straight forward to obtain the required action by the agent, while at the same time the solution can adapt to the changing conditions of the environment and still present the desired global behavior. These control algorithms are especially popular in spacecraft swarming design, with multiple interesting variants proposed [12, 53, 72]. Also a similar subset of behavioral algorithms quite popular in spacecraft control are the virtual forces models, which instead of the artificial potential define fictitious forces to be added to the dynamics models [25]. In any case the advantage of using these kind of behavioral algorithms seems to be an ease in the design as the designer can draw inspiration from nature forces or potentials to generate the desired functions.

Although the theory is pretty straightforward, there are many ways to implement a behavioral control depending on how the different behaviors interact with each other. Three main architectures can be considered in this case: subsumption, cooperative and Null-State-Based (NSB).

- **Subsumption** control consists in creating a competition among the different behaviors, only choosing the winning behavior each time to be executed [6]. A regulator function dependent on the sensors inputs is in charge of selecting the applied behavior.
- **Cooperative** control consists in generating a sum of all behaviors, therefore taking into account all behaviors in each iteration of the controller [7, 8]. Usually this kind of control is implemented following Arkin's motor's scheme where a supervisor generates a sum of the different behaviors, based or not on the sensory inputs [6]. Opposite to subsumption control, this scheme of behavioral control does not need a clear definition of hierarchies between the different behaviors, but still

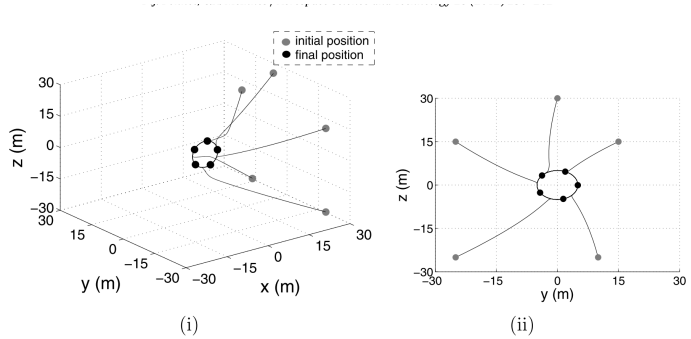


Figure 1.1: Example of the Working of Behavioral Algorithms from a Frontal and Top View from [12]

requires some hand tuning of the supervisor.

- **Null State-Based** control is proposed by [6] as a middle ground between the other two well established solutions. NSB allows for the existence of a hierarchy but at the same time does not force for only one behavior to be used. Instead NSB analyzes which behaviors are not generating conflicting actions in the state space and allows for the principal to be added with the non conflicting ones. In spacecraft swarming most of the behavioral control algorithms are artificial potential functions or virtual forces models with subsumption [72] or cooperative control [53] probably due to the fact that most of the behaviors are quite conflicting to fully exploit the NSB architectures.

Most designs mainly consist of a gather and a repel potential as they are the basis for pattern generation [52, 72]. Nevertheless, where many authors differ is in the different ways to add and improve on the swarm properties and in the design of these algorithms. In [52] a method is proposed to generate the manifold that leads from all possible final configurations to the velocity field that leads to them, reducing and automatizing more the design. Another interesting point that is not usually covered by the behavioral algorithms is how to address and plan the flexibility to, for example, change the pattern formed. In [74] it is proposed to address this problem by using bifurcating potentials, which vary the shapes by modifying the value of a certain pattern and allow for controlled transitions. Also to address this issue [25] proposes the control of the transitions by using Gabriel graphs [62] which avoid disconnections during the transitions.

Overall, these controllers seem to be an interesting option given the ease of design for the engineering teams, as there are already many functions to represent the different behaviors and inspiration from natural forces can be used to design new ones. Interesting results have been obtained proving both flexibility and automatization [52, 71] and scalability [74]. It is surprising thought that the loss of agents does not seem to be extremely studied in many of the articles. Nevertheless, the use of the behavioral algorithms still presents some major inherent drawbacks such as the recognized convergence in unexpected minimums [16, 53, 76]. This problem is likely to scale with the increase in size of the proposed swarms, and the different solutions proposed still seem to rely too much on human interaction or runs of many simulations in order to find the anomalies and hand tune for them [53, 76]. This, together with the need of large times tuning the artificial potentials to obtain the desired final patterns [6] pose two major areas of improvement for other techniques with respect to behavioral algorithm swarming.

1.1.2. Machine Learning Based Controllers

The second popular method to control a swarm comes from one of the current hot topics in academia, Machine Learning (ML). The classical definition of ML is the one from Mitchel [65]:

> *A computer program is said to learn from experience* ***E*** *with respect to some class of task* ***T*** *and performance measure* ***P****, if its performance at task* ***T****, measured by* ***P****, improves with experience* ***E***

Specifically in swarm control for spacecraft most ML controllers are based on a technique named Deep Learning (DL) which is used to generate a Neural Network (NN) controller. A NN is a set of logical

operations layered whose goal is to mimic a certain (not necessarily known) function g by learning a certain subset of parameters vector θ such that the mapping y = f(x,θ) fits best the function g [48]. The learning process will consist of an optimization of the parameters in such fashion that given a subset of examples a certain cost function (for example the norm squared) of the difference between y and g is minimized.

A controller for a swarm can be generated such that for each agent, giving its current state, a NN generates the action necessary to result in the desired pattern. This technique has extensively been proven in robotic swarming, especially in the field of evolutionary robotics, which tries to optimize the controller through the use of evolutionary algorithms [11]. Examples of these kind of controllers can be found in [16] [11] or [48]. By design NNs can easily model unknown non-linear dynamics. That is why in spacecraft swarming most applications of NN in controllers are found in the modeling of non-linear dynamics within the control unit [59, 107, 108]. Nevertheless, some interesting efforts have been presented by Izzo *et. al.* which use two neural networks to control both attitude and dynamics of three small spacecraft on board the International Space Station (ISS) [53, 54]. These efforts prove that these controllers can have interesting applications in space swarming.

NN controllers allow for an automatic design process, probably much more optimized than any which a human designer could do with behavioral algorithms in the same time span. Furthermore, their combination with evolutionary algorithms allows to find solutions closest to the absolute minimums[11]. They also allow to include non-linear effects in the system models which should produce better results in certain cases [108]. Nevertheless, they present a major disadvantage. As they are seen by humans as ”black-boxes” [16, 48] it is not easy, if possible at all, to understand what they are exactly doing, and therefore their behavior cannot be readily predicted. This, together with the fact that the learning process usually is not expected to take place during the mission due to computational requirements [11], generates a big risk as flying it on a mission will mean to have a controller not fully understood and only validated (and trained) by a simulation model on ground. Certainly NN controllers have a bright future ahead as they provide with a well tailored controller to the given data with minimal need for human interaction. Therefore, it is possible that when on-board computers reach the power necessary to train and refine models in space, this technique of swarm control will become popular. But, for the moment, using an NN will mean using a not fully understood element to manage a key subsystem such as the AOCS. This seems unlikely given the stringent requirements that usually govern space systems.

1.1.3. Other Techniques

The different inherent problems that NN and APF present, together with the peculiarities of the space environment seem to have moved researchers to also present some interesting swarming guidance and control algorithms as alternatives to the previous ones. These techniques usually come from either other areas of the space guidance and control [49, 55, 67] or other fields [44]. A more complete review of these and other swarming methods can be found in [78].

One of the most relevant groups of other techniques are Passive Relative Orbits (PROs) designts. These are a set of orbits predesigned to bound as much as possible the relative drift between agents. These require, once the swarm agents are placed in their designed orbits, a minimum amount of corrections easy to automatize or even command from ground. This technique is quite popular in formation flying [4], and has been introduced to swarms of spacecraft in through a minimization of the differential energy of the different agent orbits with a reference circular near-Earth orbit and only considering the effects of the J_2 spherical harmonic component of the gravity [67]. In the same line of thought [55] presents a new subset of conditions to ensure the minimization of the energy differences of the different orbits that escalate less quickly with the number of agents, allowing to ease the computational power requirement on the agents. This is coupled with the use of the Relative Orbital Elements (ROE) guidance and control scheme, proposed by D’Amico [33]. These methods have the advantage of being both originally designed for space, therefore the particularities of the spacecraft dynamics and environment are taken into account in the design of the algorithms. Nevertheless the results are hardly useful for other reference orbits or case studies. Furthermore, even though these examples are labeled as swarms, they certainly do not put much emphasis on interesting aspects of swarming such as how to

automatize them, how the swarm has knowledge of other agents, or how the swarm proves its flexibility, scalability and robustness.

Another interesting guide and control swarming method is presented in [45, 68, 69], the Satellite Assignment and Trajectory Optimization (SATO) algorithm. This algorithm joins a sequential convex programming [15] to optimize the trajectories of the swarm agents with an auction algorithm to select the final positions of each agent in each iteration of the algorithm. All this is integrated with a Model Predictive Control (MPC) controller. This algorithm optimizes the flight time, but also requires the capacity of recognizing the current and target positions as well as a full communication system within the swarm. It is also one of the few algorithms that seems to account for the massive lose of agents and for both homogeneous and heterogeneous swarms.

1.2. Swarming based on Probabilistic Finite State Machines

The previous section shows that the current state of the art techniques still have major improvement areas. NN and APF have been proven apt for space missions but still have some pitfalls as unexpected minimums, difficulties to generate complex behaviors, problems with scalability or lack of understanding. On the other hand *ad. hoc.* techniques still are too specific for certain cases or require large quantities of information and computing power to work. With this scenario it only seems logical to explore new solutions for spacecraft swarming.

One interesting method for swarming that has never been used in spacecraft is the use of FSMs. FSM can be defined as an algorithm that, given a set of limited states *Q*, and initial state, *I*, a final state, *F*, and an application, δ, a transition is generated such that $(F = \delta(I)) \in Q$. This artificial intelligence technique has been widely used in natural language processing [58] and even some of the initial attempts of neural networks clearly resemble FSMs [63]. In swarm control FSMs become an interesting solution as by creating a discretization in a state-space of the agents the necessary information to both generate and evaluate the state can be contained. Furthermore, being an artificial intelligence technique, it tries to reproduce the reasoning processes of an intelligent being [105], therefore it can cope with unexpected situations. This is extremely valuable for spacecraft as reduces both the operation cost and the need of continuous communication with the spacecraft. In the case of swarms the autonomy of the system is even more paramount as individually controlling the agents or adapting the whole system to changing conditions can generate an extreme increase in the ground station hours needed. Constellations such as Iridium or GlobalStar with tens of satellites already require hundreds of operators. Bearing in mind the low-cost idea behind nano-sats, which are the most likely realization in spacecraft of swarm agents, the cost associated with such a huge number of operations is simply unbearable. Therefore the adoption of AI solutions seems necessary to reduce these costs and allow for easier, cheaper and more efficient operation [89].

The use of PFSMs in swarm control is not new. Açıkmeşe and his collaborators [2, 3, 35, 39, 73] have been working on its use for a general purpose guidance system for years now. Their design is based on the view of the swarm from a probabilistic and density point of view, which allows for an ease in the control of extremely large swarms. Their idea is based on dividing the state space in sections or bins, possibly prepared for the task at hand (for example with a bigger bin density in interest areas) and then define a desired density distribution of the agents in said bins, such as the desired number of agents in each bin. A Markov chain is then generated to lead the transitions of the different agents from one bin to another as well as to keep track of the global state of the swarm locally. The idea is that for a large enough swarm the final density distribution will be really similar if not equal to the desired one. Therefore they use a probabilistic control method which instead of controlling the positions controls the propagation of the probability that an agent is in each of the bins. Given sufficient time and number of agents the difference between the desired distribution and the actual distribution will be minimal. As the Markov chain is generated with a Metropolis-Hastings locally in each agent, it allows for a decentralization of the algorithm. Also test have been performed for self-repair of the swarm as well as collision avoidance methods are implemented.

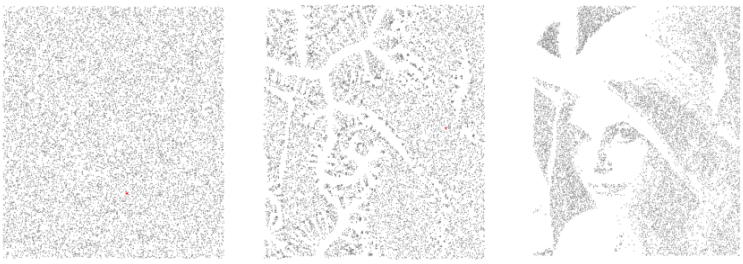


Figure 1.2: Açıkmeşe's algorithm example from [2]

Açıkmeşe 's work proves the interesting properties of using PFSMs as guidance methods for swarms. Nevertheless, some of the properties of following a probabilistic guidance approach as proposed can be counterproductive in space. It is mentioned that this algorithm works better on large swarms than in small swarms, which are better controlled by deterministic methods such as APF and NN. Nevertheless, a tentative spacecraft mission seems to be a middle-ground case in the size of the swarm. This technology is usually envisioned with small nano-sats [38] [45], whose launch is usually done as a piggy-back ride taking advantage of the space left in another lunch. It is not strange to consider that the swarm will vary in size, starting with a small number of elements and steadily growing as more and more launches are available. Furthermore, some of the considered systems such as CubeSats a standard, developed with a focus on enabling university and small organization projects. Therefore, this piggy-back ride launch strategy and the growth of the swarm with time and availability of resources, medium and variable size swarms should be something to consider. That is why this work will focus in the application of PFSMs with another algorithm, which has been named for this work Desired State Heuristic Algorithm (DESHA for now on) for the sake of brevity.

DESHA is a multipurpose swarming algorithm developed by Coppola *et. al.* [31] based on PFSMs developed with consideration for space and autonomous air vehicle uses. The idea behind DESHA is the definition of a desired global-state of the swarm such that it can be decomposed in a set of unique local states in such fashion that the only possible combination of said local states is the desired global state. Thanks to this the only state in which all elements of the swarm detect that they are in a desired state is when the global state is achieved. Meanwhile, a set of transitions are generated. Said transitions are encoded in the transition matrix Q which is created in such way that transitions which disconnect the swarm or transitions that lead to infinitely recursive movements are not allowed. Though originally the transitions in DESHA are generated with an equal probability, further developments on the theory [30] also have allowed methods for reducing the required number of steps to form the pattern through the use of evolutionary algorithms and reduction of allowed movements. The main advantage of this approach is the reduced knowledge required by the algorithm with respect to other methods which require global knowledge of the swarm such as Açıkmeşe 's, SATO or many of the APF presented. Given the scales up to which a swarm can grow, it seems unrealistic to pretend that hundreds of agents can communicate with each other to relay their current state in an acceptable amount of time. Another advantage of using DESHA versus Açıkmeşe 's approach lies in the more controlled collision avoidance by design included in it. Desired densities do not make the swarm avoid collisions, only make it less likely on low density areas since less agents will end up in them, but a second mechanism in the low level control of the agents needs to exist to avoid collisions if several agents are expected on a bin. By implementing DESHA, most collisions are avoided in the high level controller (DESHA itself). Given that a collision in spacecraft would be catastrophic both for the mission and other missions that might collide with the debris, this seems to support the use of the DESHA and other algorithms that consider the risk of collision at all times.

1.3. OLFAR, A Case Study in the Far Side of the Moon

In order to properly study the use of DESHA for spacecraft swarming it was considered interesting to find a study case which allows the reader to see one of the possible uses of spacecraft swarming. Also, this was interesting as a mental exercise to actually find real implementation problems rather

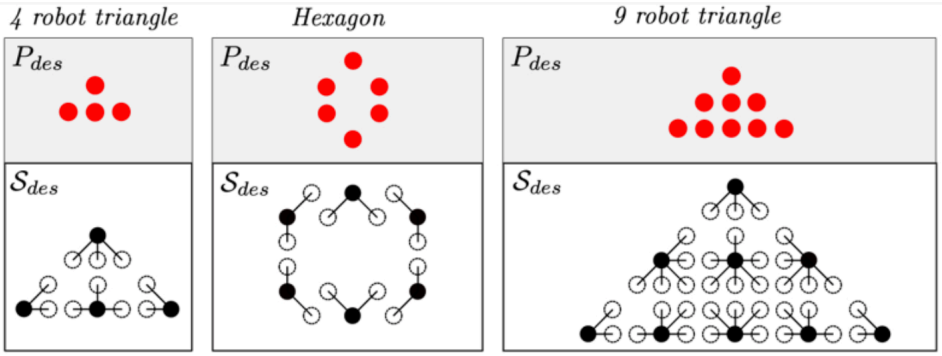


Figure 1.3: DESHA patterns and desired states. DESHA is capable of guiding the swarm to this pattern. From [31]

than just randomly defining a desired state in a random orbit and trying to obtain it. Although so far no real spacecraft swarming mission has been flown, there are several interesting concepts already developed, some of them with documentation developed. Examples of this are the APIES mission [38], the ANTS mission [32] or the SWIFT mission [27]. The case study selected is linked to one of the most interesting applications found by the scientific community to spacecraft swarms, distributed array space telescopes. In a nutshell, these consist in a set of small antennas whose combined effects allow to obtain observations of the same wavelengths as larger monolithic telescopes [34]. By using a swarm in orbit to develop this kind of distributed system it is possible to launch telescopes to receive these signals without the need of an extremely large antenna. The study mission selected is OLFAR [13, 19, 34, 94], a distributed array radio telescope mission to the far side of the Moon developed by the Dutch Universities of Technology. The advantage to send this radiotelescope to the far side of the Moon is the shielding effect that the Moon will have on Earth noise radiation.

From a practical perspective, this mission has been selected as the study case for this project thanks to all the available information published about it, which includes from tentative designs of the intersatellite links to a preliminary design of the orbit of the spacecraft. This allows to avoid taking too many assumptions on the mission which might end up in non-realistic proposals and at the same time generate conditions real enough to be published to the scientific community as a proposal. Nevertheless, the mission will not be the core of this project but rather a working reference frame. If needed, assumptions will be taken and, if so, they will be indicated.

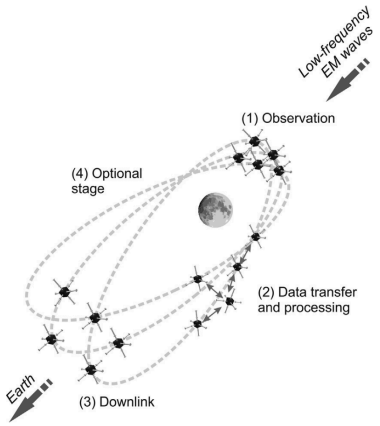


Fig. 1. OLFAR swarm on a lunar orbit.

Figure 1.4: OLFAR mission scheme. From [19]

1.4. Problem Statement

The main objective of this thesis report is to record the advances done during this project in order to improve on the current knowledge of the scientific community about spacecraft swarming. Ultimately, the whole thesis project aims to answer one un-answered scientific question whose resolution will enlarge the body of knowledge of the topic. Said question is:

Is it possible to control and guide a swarm of spacecraft using DESHA?

The main rationale behind the main question is to pose in a question form the main goal of the project, the analysis of the DESHA algorithm for spacecraft swarming. The project will try to answer if DESHA is suitable for space application also answering which kind of systems are suitable for the use of DESHA. Also the analysis of the limitations and performance of the algorithm will be performed in order to fully address the suitability of DESHA.

These goals will be formally presented in a set of three sub-questions, whose answer will provide input to the answer the main research question:

1. Does DESHA comply with the swarm properties of flexibility, scalability and robustness?
2. Does DESHA perform within reasonable limits in use of propellant and time with respect to the reference mission and the current state of the art of small spacecraft technology?
3. Does DESHA present any kind of pitfall or unexpected blockage?

The first sub-question will address the performance of DESHA by analyzing the properties that swarming brings to the space mission. In order for the algorithm to present that extra swarm intelligence that the algorithm should bring, these preoperties shoudl be present.

This analysis could be achieve both through quantitative and qualitative methods and will measure the performance of DESHA as a swarming algorithm. Therefore the scope of this sub-question will be both general as it will analyze DESHA as a swarming algorithm and specific as it will analyze its performance in a certain scenario.

The second sub-question will analyze specifically the extend in which DESHA is suitable for spacecraft swarming scenarios. The scope of the analysis will be limited to small spacecraft with real actuation capacity (i.e. microsatellites to nanosatellites). Smaller spacecraft such as chipsats or larger spacecraft such as medium and large size spacecrafts of several tons will not be considered. The first ones will be discarded as it is supposed that the current state of the art does not allow them to have active actuation capacities required by DESHA. Nevertheless, future developments might lead to these kind of spacecraft to achieve similar capacities as the ones considered in this report. In said case, the results obtained could be extrapolable to these systems. On the other hand, due to the cost of development of an spacecraft of more than 100kg and the launch costs, considering a swarm of tenths or thousands is regarded as just not feasible from the economic point of view. Therefore the actuation and possibilities of these systems will not be considered for this work.

Finally, the last sub-question will analyze the limitations found while applying DESHA to the space scenario. This sub-question will address possible unexpected situations that will occur while implementing and using the algorithm in space. Since this is a research project on a previously unexplored algorithm, it is not unlikely that the implementation of the algorithm in the space scenario will generate unexpected situations. This sub-question therefore will address the conclusions on unexpected situations of pitfalls found while implementing the algorithm in space.

Overall, the answering the three sub-questions will allow to address the suitability of DESHA for space applications, both benchmarking its performance and its suitability.

1.5. Structure of the Report

This report will be structured in four main parts. Part I will be this introduction. Part II will encompass Chapters 2 to 6 which will contain the methodologies used to study the use of DESHA in spacecraft

missions. Part III will encompass Chapters 7 to 9. On it the experiments performed with the developed methodology and the results obtained will be presented. Some initial discussions will also be presented in each of the experiments and as a conclusion at the end of each round of experiments. Finally Part IV will end the project with a set of discussions on the overall project, the conclusions, and future recommendations on the project. This structure is also shown in Figure 1.5.

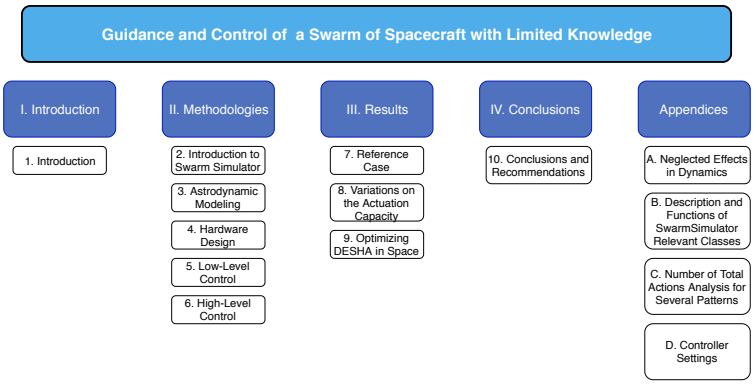


Figure 1.5: Report Structure

II

Methodologies: Swarm Simulator

Introduction to Methodologies

In this part of the report, the methodologies used to proceed with the study which will lead to the resolution of the research question are presented. The main methodology used are computational simulations. These will allow to proceed with a set of experiments which will determine the capacities of DESHA in spacecraft missions. This choice is mainly motivated by the time, available facilities and budget constraints of the thesis project. An accurate enough simulation allows, in the span of 7 months, to study if there are any issues in the implementation of DESHA in spacecraft missions and what are the advantages and pitfalls of it. At the same time the cost of the project is reduced to the student/supervisor hours needed and the computer resources required to run the simulation. Furthermore, it was seen in the literature review that this is the most common method to approach spacecraft swarm research [3, 68].

The simulation tool created and presented in this part of the report is called SwarmSimulator. The goal of this simulator double. On one side the simulator must be able to simulate all the physical aspects of the problem at hand. This means that it must be capable to simulate the spacecraft dynamics and main environmental effects affecting the swarm. Also, it must be able to reproduce the physical properties of the swarm agents and relevant hardware elements. On the other hand, it must be capable of incorporating the full guidance and control algorithm so that it works as a test bed for DESHA. It is expected that by achieving these two objectives the simulations can give sufficient confidence levels to the results obtained such that the scientific community regards them as valid.

Given that the main goal of this project is to generate a relevant contribution to the scientific community. To this end, it was preferred to develop the simulator in a prototyping language. The advantage of using a prototyping coding language instead of a language more oriented to hardware implementation or more optimized for faster run times, is that the writing and implementation of the algorithm is expected to take less time. This generates more time for testing and developing algorithms and solutions to encountered problems, which is more interesting from the scientific point of view. The language chosen was MatLab® given all the available documentation and tools included already with the license. It was also valued the possibility of transforming the code to other languages more suited for hardware implementation automatically.

This part of the report is organized in five chapters. The first one (Chapter 2) will present present a general overview of SwarmSimulator. This is expected to give the reader some context on the overall picture of the tool developed. The following chapters then will get into depth in the main four elements of the simulator. Chapter 3 will present the modelling of the dynamics and environmental effects used. Chapter 4 will present the modelling of the hardware used in the swarm. Chapter 5 will review the design of the low-level controller implemented. Finally Chapter 6 will present how the high-level controller (DESHA) has been implemented. It is expected that the reader will be able to fully understand the workings, assumptions and decisions taken after reading this part of the report Also the extend and limitations of the model of the methodologies used for the project.

2

Swarm Simulator Overview

SwarmSimulator is a specific software package developed for the testing and development of spacecraft swarms. Even though it has been mainly developed for the application of DESHA, it has been designed in such a way that other high-level and low-level controllers can be added in order to explore their properties. SwarmSimulator is designed based on a ”global-to-local” approach. The intention is to design a set of high-level programs easy to read to the coder which generate a set of calls to more specific modules, with slightly more complex functions. This pyramidal structure allows for future developers and users to be able to have a quick overview of the whole system and be able to get in-depth just in the parts of the code where there is a need for advancement or change. Said pyramidal structure can be appreciated in Figure 2.1.

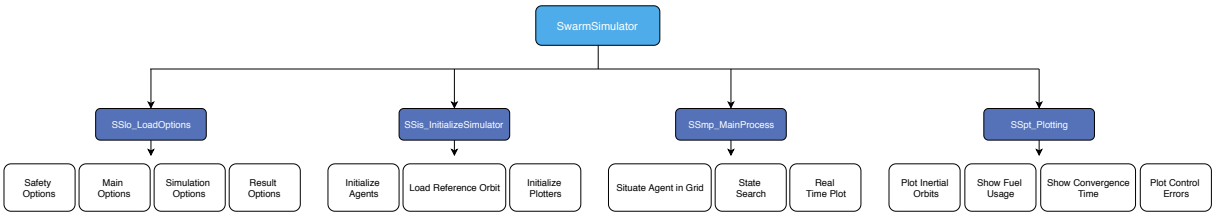


Figure 2.1: Swarm Simulator Functional Overview. It can be appreciated how the deeper layers of the simulator have more specialized functions than the upper ones.

SwarmSimulation is presented in this chapter. First in Section 2.1 a detailed view of the processes and elements composing SwarmSimulator will be shown. This section does not intend to work as a software manual. Therefore it will present the information from a functional point of view and technical details will be kept to the minimum. Then in Section 2.2 the outputs and goals of SwarmSimulator will be introduced briefly. Finally, SwarmSimulator has been designed with expansion capabilities as said before. In Section 2.3 these capabilities will be presented. The goal of this chapter is to let the reader have a full overview of the simulator. Thanks to this, the reader will have an overall context when the detailed models for the different elements of it are explained in the upcoming chapters.

2.1. SwarmSimulator – A Detailed Overview

A detailed overview of SwarmSimulator is presented in Figure 2.2 with a mix functional/programming approach. In this overview it can be seen both the internal working of each of the main processes of SwarmSimulator as well as the objects manipulated by the simulator.

SwarmSimulator is composed by four main processes:

- Loading of Options
- Initialization of the Simulator

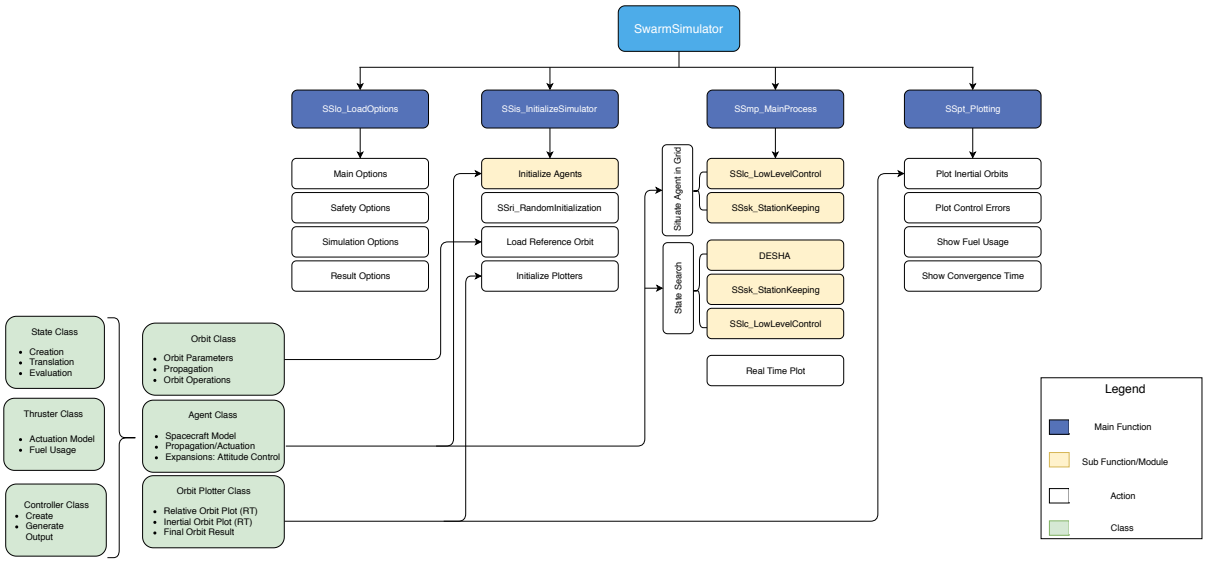


Figure 2.2: Detailed view of SwarmSimulator presenting both the different processes and task performed as well as the objects and specialized functions used.

- Main Process
- Results Processing

Each of these processes will manipulate a set of global variables and objects to finally output the results of the simulation. The global variables will define main options of the simulator, such as the size of the movements, the size of the swarm, the desired pattern, etc. These variables will be used to transfer the information through the whole simulator, making sure that all processes are working with the same parameters. With respect to the classes SwarmSimulator presents three main classes which define the objects modelled in the simulator:

- **Orbit**. Orbit objects contain the information and operation for the orbits used in the simulation. In this case this will be the reference orbit.
- **Agent**. The agent class will encompass all the properties that define the design of the agents of the swarm, as well as all the performed actions on them, including the control, collision avoidance, used propellant, definition of the state,etc. The class supports a series of subclasses which will define used elements of the agents and reduce the size of the main code for the class, such as thrusters or state-related operations.
- **Plotter**. This class will define all necessary objects to plot and analyze the results, both in real time during the simulation and after post-processing of the results.

Once the main elements that carry the simulator's information have been presented, the following subsections explicitly define how the main processes of SwarmSimulator work.

2.1.1. Loading of Options

SwarmSimulator has been defined as a versatile simulator, capable of carrying simulations with several patterns, several modes of high-level control, the use of optimized approaches, etc. The use and definition of all these options is done in this routine. For a better understanding, in an analogy with a Graphic User Interface (GUI), this is the code representation of an options tab in the simulator. In it, four main types of options, which might correspond to four sub-tabs, are loaded:

- **Main Options**. These refer to the characteristic options that will define this simulation. These include the type of pattern desired or the use of optimized approaches for example.

- **Simulation Times**. These options load time related parameters such as time-step used or total simulation time.
- **Safety Related Options**. This part will define options such as the inter-satellite safety distance or the station keeping tolerances of the simulation.
- **Output Options**. In this part the type of output plots and messages required are set. Options such as simulate in debugging mode or generate a log in the console of MatLab® are also set.

All the elements defined in this process are set as variables with the capacity to be accessed by any other main process in the simulator. These variables are supposed not to be modified during the simulation. So they will act as non-varying parameters of the simulation.

2.1.2. Initialization of the Simulator

Following the GUI analogy presented in Section 2.1.1, this process is a tentative set of tabs to define the elements of the simulator. This process initializes the different objects using to perform and store the operations used in the simulator. For example the swarm would be initialized in this process as a set of agent objects, whose positions and states would be latter manipulated to perform the simulation.

Four main sub-processes are performed in the initialization:

- **Agent Initialization**. The agents of the swarm are initialized for this simulation. Properties such as the design of their actuators, length of their movements or communications are set.
- **Random Positioning of the Agents**. The agents are set in a random initial state, but maintaining a connected topology. This will be essential for the use of DESHA (see Chapter 6) and it is done through a routine specifically designed for it.
- **Reference Orbit Initialization**. The reference orbit of the swarm is created. It is designed as a two body keplerian orbit. For more details on the propagation of this orbit see Appendix B.
- **Initialization of Plotters**. According to the options set, a series of figures are initialized (but not displayed) to plot in real time the simulation.

Most of the designs in this process are already set to a default value, although some changes such as the size of the swarm, the radius of communication or the distances moved by action can be changed here. The objects created in this process now have all the information to be manipulated by the simulation process. Unlike the options, these objects have some of their variables modified, such as the positions of the agents, by the main process of the simulator.

2.1.3. Main Process

As its name indicates, this is the process of the simulator that actually performs the simulation. This process can be subdivided into three sub-processes:

- **Situate Agents in Grid**. As it is explained in [31], DESHA is an algorithm that discretizes the space state, analyzes it and acts consequently. It improves the performance of the system if the agents are situated in points of this discretization of the state space. Thanks to this the errors for the discretization are reduced to only the control tolerance, as agents only move in points of the grid generated by the discretization. This part of the main process basically moves the agents, initialized already, to the closest point of the grid.
- **State Search**. The algorithm from [31] has been named Desired State Heuristic Algorithm (DESHA) as it performs a search for a set of desired local states. This is done in this sub-process, which use the high-level controller to generate the targets for the low-level controller to move the agents to the new desired locations. These positions are the steps in the search for the desired final state.

- **Real Time Plotting**. According to the options selected, the simulator plots the position of the agents, either relative to the reference orbit or an inertial reference frame (in this case centered in the Moon), for every time step of the simulation.

The two first sub-processes are performed thanks to the use of the high-level controller, the low-level controller and a routine to ensure the station keeping. This station keeping routine situates each agent in their current locations if the high-level controller has not generated any new movements for it in that time step. All these three actions are performed with their own functions and software modules.

When the main process is finished, the simulation is technically finished. The results of the movements of the agents according to the controllers used, the dynamics implemented and the agent design, are stored for their processing.

2.1.4. Results Processing

This process takes the results of the simulation and present them to the user, modifying them to obtain the desired presentation form. This process is composed of four sub-processes:

- **Plot Inertial Orbits**. The orbits of the agents, as well as the reference orbit, are plotted in an inertial reference frame centered in the Moon.
- **Low-Level Control Errors**. The errors of the low-level controller are plotted to analyze its performance.
- **Show Fuel Use**. The total fuel use and change in velocity needed are shown per agent.
- **Show Convergence**. The total number of actions, which will be called from now on convergence of the simulation (or convergence), are presented. This is the total number of moves requested by the high-level controller.

Also stored in the memory the history of the movements of the agents relative to the reference orbit is stored for possible uses and analysis needed. The execution of this process means the end of the simulation run. If multiple runs are needed, the results should be saved from the work space as the next run will erase all the existing information in the workspace.

2.2. SwarmSimulator Goals

In the end, it is expected that SwarmSimulator is capable of giving enough information to actually evaluate if the combination of the high-level controller (in this case DESHA) and the low-level controller selected for the given swarm is able to reach the requested state and maintain it during the simulation time. Furthermore, it is expected that DESHA can generate the following outputs:

- **Time of Convergence of the Swarm**. Understood as the time that the algorithms have required, under the selected dynamic and propagation model, to reach the desired state for all agents.
- **Propellant Used**. Understood as an estimation, based on the designed on-board agent and its hardware specifications, of the propellant used per agent to achieve and maintain, during the whole simulation time, the desired states.
- **Ability of the Swarm to Maintain the Desired State**. The results obtained with the simulator should allow evaluating if the control algorithms used will be able to, once reached, maintain the desired state.

SwarmSimulator also allows the user to give a real-time view of the movement and actions of the swarm. This permits a human supervisor to check for the proper working of collision avoidance mechanisms and avoidance of disconnections. Overall, the results obtained with the simulator shall allow answering, under the assumptions taken during the project, the research question and sub-questions presented in the first chapter.

2.3. SwarmSimulator – Possibilities for Expansion

SwarmSimulator has been designed with a modular philosophy to allow for the future study of new components, controllers, orbital dynamics and similar elements. Originally implemented in the simulator there are modules and functions for:

- **Attitude Control**. Attitude control functions are already designed (but empty) in the Agent class. Also a reaction wheel subclass is included in the Agent class with some properties and a constructor already designed.
- **New State Discretizations**. Three state shapes are included in SwarmSimulator (see Chatper 6), but new definitions can be included through the State subclass of the Agent class.
- **New Dynamic Models**. Adding dynamic functions in the dynamics folder and their corresponding propagator in the agent class will allow for the exploration of different relative dynamics.
- **New Controllers**. By varying the calls to the main or low level controls new controllers can be tested.

Furthermore, new modules, classes and functions can be defined and easily implemented in the SwarmSimulator flow by creating the desired function and adding the proper call within the processes of SwarmSimulator.

3

Astrodynamic Simulation

This chapter will present the models and assumptions taken to simulate the space dynamics and environment in SwarmSimulator. The astrodynamic model is composed of three elements, which represent the effects of the surrounding environment and its physics on the swarm. These are:

- **The Reference Scenario**. In the case of this report, the reference scenario will be the OLFAR mission [41]. This will determine which bodies are affecting the orbit of the swarm, which effects are dominant, the availability of communications with the ground station, the possibility of using space-born systems such as GNSS constellations, etc.
- **The Dynamic Model**. This will be the representation of the forces that affecting the agents of the swarm, as well as the reference systems used to express them.
- **The Propagation Model**. In order to evaluate the motion of the agents and the swarm under the effect of the dynamic forces, it will be necessary in some cases to integrate a set of differential equations through numerical methods. The selection of the method is vital as an error due to its use will be present.

This chapter will be dedicated to evaluate all these three elements, as well as the assumptions taken and the reasoning behind the suppression of the neglected effects and its consequences. First, in Section 3.1, an overview of the reference case will be given, this will give context to the reference orbit used as well as some peculiarities of the chosen scenario, such as the fact that the main body is the Moon instead of a more dominant body such as the Earth or the Sun. Then in Section 3.2 the set of assumptions taken for the design of the astrodynamic model will be stated and motivated. In Section 3.3 the dynamic models, that will be the main element of the astrodynamic simulation, are presented. Linked to this section, in Section 3.4 the consequences of the neglected environmental effects in the dynamics modelling will be shown. Finally in Section 3.5 some notes on how these models were implemented will be given.

3.1. OLFAR Context

OLFAR is a proposed space mission based on a swarm for radio-astronomy purposes where its operational time will be spent in the far side of the Moon. This poses a set of special characteristics that must be taken into account when simulating the dynamics and effects of the environment:

- **Lack of Access to the GNSS Systems**. Nowadays it is quite usual to find systems in LEO and even MEO orbits which make use of the GNSS constellation to obtain their positions and even velocities [37, 75]. Even though there is a clear intention to expand their use to higher orbits with the Service Volume Extension planned [10, 23], so far GNSS nominal working radius does not reach Moon orbits. Even though some proposals for the use of weak GNSS signals in such missions have been issued [61], the lack of real and tested data for these methods excludes their use for this application. This will entail that the agents will not have a constant signal that will allow

them to determine their global position. Their knowledge of their global position will be limited to Orbit Determination (OD) data from the ground.

- **Reduced Sphere of Influence**. The sphere of influence of a body is the volume around the main body where the orbital dynamics problem can be considered a two body problem. In the case of the Moon its sphere of influence has about 66183km of radius , as calculated in [98]. But also the calculations are less accurate than, for example, the values for planetary spheres of influence with respect to the Sun [98]. Therefore, if orbits below this altitude around the Moon are considered, the precision will be limited for a two body problem.
- **Eclipse Times with Respect to the Ground Station**. As the system is supposed to perform its scientific operations while it is on the dark side of the Moon this will imply that the swarm will not be able to communicate with the ground station at least during this time. Therefore the communication of information from the ground station will be limited.

In [34] two reference orbits are proposed for OLFAR in an attempt to create a free-drifting swarm with the specified baseline requirements. These orbits will be used during this project as reference orbits. The orbits are presented in Table 3.1. The first orbit is designed for better coverage with a smaller swarm (25 agents) while the second one accepts a higher number of agents (100) for the same coverage with a smaller drift. It is notable to say that in neither of these cases the atmospheric drag plays a role, as the atmosphere of the Moon is extremely tenuous and was not considered in [34], nor it will be during this project. The orbit choice will be performed in the results part of the report, based on the conclusions of other elements of the simulator (see Chapter 5)

Orbit	**a** (km)	**i** (rad)	**e**	Ω (rad)	ω (rad)	τ (s)	**Description**
Orbit 1	1937.4	0	0	N.A.	N.A.	0	Circular equatorial orbit of 200km of altitude.
Orbit 2	4737.4	0	0	N.A.	N.A.	0	Circular equatorial orbit of 3000km of altitude.

Table 3.1: Description of the two reference orbits proposed in [34]

3.2. Assumptions

Before studying how to model the environment effects a set of assumptions must be taken. These assumptions will be divided into those which affect the modeling of the environment and those which affect the swarm. The former will allow for both the simplification of the definition of the dynamics and the effects considered for modeling the environment while the latter will define constraints on some aspects of the swarm.

3.2.1. Assumptions on the Environment

1. **AS-EN-EN-001**: The dynamics will be considered a two body problem.
2. **AS-EN-EN-002**: The Moon and the agents will be considered point masses.
3. **AS-EN-EN-003**: No solar radiation or other perturbation effects will be considered.

Assumption AS-EN-EN-001 stands to reason as the two proposed orbits in [34] are within the sphere of influence of the Moon. Therefore, the dominant effect on the swarm will be the Moon by far. This will be further analyzed in 3.4. Assumption AS-EN-EN-002 will allow approximating the gravity effects of the Moon without having to consider density distribution of the bodies, as well as momentum, tides, and attitude. Assumption AS-EN-EN-003 follows the same idea. The perturbations will be neglected in order to simplify the dynamics. In Section 3.4 it will be shown that the omission of said effects will not be extremely relevant for obtaining a good dynamic model.

3.2.2. Assumptions on the Swarm

1. **AS-EN-SW-001**: The swarm will be able to communicate with the ground station once every orbit.
2. **AS-EN-SW-002**: The swarm will receive from the ground an updated version of the reference orbit once per orbit.
3. **AS-EN-SW-003**: The swarm will be able to receive the necessary solar input to have its batteries fully charged at all times.
4. **AS-EN-SW-004**: The swarm will be considered as undamaged by any environmental effects unless otherwise stated.

Assumption AS-EN-SW-001 and Assumption AS-EN-SW-002 ensure that the swarm will be able to have a sufficient model of its reference orbit at all times. The dynamic models simulating the movement of the swarm will be focused on relative dynamics around the reference orbit of the swarm. This means that the devised agents are expected to have the capacity to propagate their states with respect to a ground truth common to all, the reference orbit. The best way to ensure that this reference orbit is common to all is that the orbit, or at least a point of reference for all the swarms (such as a set of coordinates at a given time), is given to all the agents. The communication of this information to all agents of the swarm will be presented in Chapter 4.

Assumptions AS-EN-SW-003 and AS-EN-SW-004 basically state the fact that the swarm will not be damaged by aging effects or the environment unless otherwise stated. This will allow addressing the swarm in its operative form. Of course, further developments of this work should target agents affected by aging, with reduced actuation capabilities, etc. However for the current case, this is considered out of the scope.

3.3. Dynamic Models

Once the assumptions made have been analyzed, the next step will be to explain how the environment has been simulated within the software. In the case of the effects of the environment on the swarm, none of the assumptions generate any requirements for the codding. Assumption AS-EN-SW-001 only generates a need for the swarm to be able to autonomously operate on one orbit, while Assumptions AS-EN-SW-002 and AS-EN-SW-003 only relate to the fact that the swarm and its agents will be considered fully operative at all times unless otherwise stated. It is with the dynamics model where the environment plays a role in the calculations necessary to study the performance of DESHA in space missions.

In this section, the equations used to model the dynamics affecting the swarm will be explained.

3.3.1. Hill-Clohessy-Wiltshire Equations

Hill-Clohessy-Wiltshire (HCW) equations (presented as Equations 3.3.1) are one of the most, if not the most, common way to model the relative position and velocity of a spacecraft with respect to another one. They represent a particular case of the restricted three body problem under three assumptions (see derivation in [98]:

- The masses of two of the three bodies are negligible when compared to the third one, called the central body.
- One of the bodies (hereafter named the main body) will move in a circular Keplerian orbit around the central body.
- The separation between the two ”massless” bodies is much smaller than the radius of the orbit of the main body.

The advantage of using this approach is the fact that the equations provide a set of closed analytical solutions that allow determining the position and velocity of the secondary spacecraft with respect to the main spacecraft in a local-horizontal-local-vertical reference frame centered in the main body given

an initial state and the mean motion of the main orbit[98]). Therefore, if a virtual agent is considered to be always in the swarm's reference orbit, the movement of each of the agents of the swarm can be studied with these equations with respect to the reference orbit.

$$
\begin{aligned}
x &= x_0(4-3cos(nt)) + \frac{x_0}{n}sin(nt) + \frac{2\dot{y}_0}{n}(1-cos(nt)) + \frac{f_x}{n^2}(1-cos(nt)) + \frac{2f_y}{n^2}(nt+sin(nt)) \\
y &= y_0 - \frac{\dot{y}_0}{n}(3nt-4sin(nt)) - 6x_0(nt-sin(nt)) - 2\frac{x_0}{n}(1-con(nt)) - \frac{2f_x}{n^2} \\
z &= z_0 cos(nt) + \frac{\dot{z}_0}{n}sin(nt) + \frac{f_z}{n^2}(1-cos(nt)) \\
\dot{x} &= 3x_0 nsin(nt) + \dot{x}_0 cos(nt) + 2\dot{y}_0 sin(nt) + \frac{f_x}{n}sin(nt) + 2\frac{f_y}{n}(1-cos(nt)) \\
\dot{y} &= -\dot{y}_0(3-4cos(nt)) - 6x_0 n(1-cos(nt)) - 2\dot{x}_0 sin(nt) - \frac{2f_x}{n}(1-cos(nt)) - 2\frac{f_y}{n}(\frac{3}{2}nt - 2sin(nt)) \\
\dot{z} &= -z_0 nsin(nt) + \dot{z}_0 cos(nt) + \frac{f_x}{n}sin(nt)
\end{aligned}
\tag{3.1}
$$

In Equations 3.1 x_0, y_0, z_0 refer to the initial radial, initial along track and initial cross track relative position respectively, $\dot{x}_0, \dot{y}_0, \dot{z}_0$ refer to the initial radial, along track and cross track relative velocity, n is the mean motion of the orbit and t is the time since the initial condition. Since these equations do not require the use of any numerical methods to be solved, they can be directly implemented in SwarmSimulator to analyze the dynamics model. This allows for the propagation of the position of each agent with respect to the reference orbit at every time step of the simulation just by having the previous position. The reference frame used will be the classical one of the HCW equations. The X axis will be the radial direction of pointing outwards the main body. The Y axis will be the along track direction and the Z axis the cross track direction. The origin will be set in the main body, which in this case will be the reference position in its orbit of the swarm at that moment.

Verification of HCW Equations Implementation

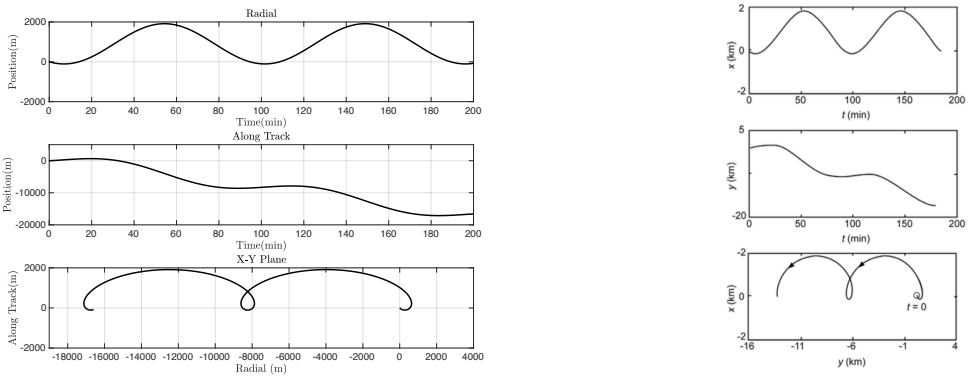
Equations 3.3.1 are quite popular in literature [80, 98], being a common choice for first analysis in spacecraft swarming and formation flying missions [74, 104]. Nevertheless, when implemented in SwarmSimulator, they need to be validated in order to ensure that they have been properly coded. To do so a simple analysis is done comparing a reference case (offered in [98]) to the solution with the dynamic model implementation of Equations 3.3.1 in SwarmSimulator. The results are presented in Figure3.1. In both cases, the motion of a secondary spacecraft originally at the same point as the main spacecraft but with a difference in velocity of -0.5m/s in the radial direction and a difference in position of 0.5m in the along track direction is presented. As it can be readily seen, both the results from the dynamics model implemented in SwarmSimulator (Figure 3.1a) and the ones in [98] (Figure 3.1b) fully coincide.

Equations 3.3.1 are further validated by following the example of the TU Delft's Astrodynamics Toolbox (TUDat) Unit Test [1]. The equations system implemented in SwarmSimulator is tested by inputting the same initial state as in the unit test for the *clohessyWiltshirePropagator* function. Said result is compared with the one presented from a reliable source (and peer reviewed, therefore can be considered reliable) in the literature [93]. In this case, the result of applying the HCW equations to a case of an Earth orbiting formation where the main spacecraft is at an altitude of 400km in a circular orbit and the secondary spacecraft has the initial state listed in Table 3.2 after 1800 seconds is known. The expected result, the obtained result and the difference between them are presented in Table 3.2.

The analysis provided by Table 3.2 seems to indicate that the implementation of the dynamic model is correct, therefore it can be safely used in the SwarmSimulator to model, under the assumptions and limitations related to the HCW equations, the dynamics of the swarm.

Limitations of the Model

The HCW equations are quite popular due to their simplicity, the fact that they do not require numerical methods to propagate the orbit of the formation and the fact that they allow for a quick exploration of



(a) (b)

Figure 3.1: Result Comparison of HCW. In Figure 3.1a the results obtained by the implemented model of the HCW in SwarmSimulator. In 3.1b the results obtained by Waker in [98] both for an orbit of 400km of altitude above the Earth being the secondary body in the same position of the primary body in the start of the propagation but with a difference in velocity of -0.5m/s in the radial direction and 0.5m in the along track direction.

Case	Radial	Along Track	Cross-Track Velocity
Initial Position (m)	45	37	12
Initial Velocity (m/s)	0.08	0.03	0.01
Expected Position (m)	380.65	-543.74	2.51
Expected Velocity (m/s)	0.15	-0.73	-0.02
Obtained Position (m)	380.65	-543.74	2.51
Obtained Velocity (m/s)	0.15	-0.73	-0.02
Difference in Position (m)	0	0	0
Difference in Velocity (m/s)	0	0	0

Table 3.2: Comparison between the results from a case from the literature used in the unit test of TUDat's *clohessyWiltshirePropagator* [1] with respect to the ones obtained with the implemented dynamics.

the main dynamics of the relative dynamics and rendezvous operations [98]. Nevertheless, it is good to remember that this is a linearization of an already specific problem (the restricted three body problem). This means that variations on the assumptions listed at the beginning of this subsection might cause large offsets between the results obtained here and the real trajectories of the spacecrafts. Slight modifications of the reference conditions assumed by the HCW model can generate long time errors which deviate the simulated results from the ones obtained with more accurate models, as shown in [51]. There it is proven that slight eccentricities already generate in few orbits offsets of kilometers. Therefore, this model only allows obtaining preliminary results on the behavior of the system. Still, as the goal of this work is a first evaluation on DESHA, the model is considered good enough.

3.4. Evaluation of Neglected Effects

Of course, the assumptions taken in Section 3.2 already set some constraints in the validity of the results. This section aims to roughly evaluate the effects neglected in the modeling of the dynamics both quantitatively and qualitatively.

3.4.1. Considering a Two Body Problem

Considering that the only gravitational influence comes from the main body, the Moon, obviously ignores the effect of other bodies that are affecting the dynamics of each agent. Even though the effects

of the main body are still bigger than any other since the whole swarm is within the sphere of influence of the Moon, there are other massive bodies that will affect the swarm. Here a small analysis of the order of magnitude of the forces neglected is presented and their ratio with respect to the force from the main body.

Setting an inertial reference frame whose axes are coincident with the ECI (Earth Centered Inertial) reference frame but whose origin is on the Moon, the force over each agent of the swarm can be described as in Equation 3.2 [98], where r_i, r_j are the distances from the agent of the swarm to the main body (the Moon) and the distance of the j^{th} body to the main body respectively, G is the universal constant of gravitation, and m_i, m_k, mj refers to the mass of the agent, main body and j^{th} body.

$$\frac{d^2\bar{r}_i}{dt^2} = -G\frac{m_i + m_k}{r_i^3}\bar{r}_i + G\sum_{j\neq i,k} m_j\left(\frac{\bar{r}_j - \bar{r}_i}{r_{ij}^3} - \frac{\bar{r}_j}{r_j^3}\right) \tag{3.2}$$

Considering that the most massive bodies affecting the Moon's orbit are the Earth and the Sun, it is only logical to analyze those as main perturbing bodies. This is done in Table 3.3:

Body	**Moon**	**Earth**	**Sun**
Force (N)	1.305	$2.7398 \cdot 10^{-5}$	$1.5478 \cdot 10^{-7}$
Ratio wrt. Total	99.9979	0.0021	$1.1857 \cdot 10^{-5}$

Table 3.3: Gravitational influence of the effect of the Moon, Sun and Earth on the swarm calculated with Equation 3.2 as calculated in Appendix A

Results presented in Table 3.3 correlate with the knowledge presented in literature [93, 98] which says that the Moon is within the sphere of influence of the Earth and that the Moon's sphere of influence is 60,000km in the Moon-Earth system. Therefore the orbit of the swarm is a perturbed orbit around the Moon. Furthermore, results show that the perturbations generated by the other possible massive bodies are small enough to be neglected without loss of generality.

3.4.2. Neglecting Perturbations

Besides the effects of other bodies, other perturbation effects have been neglected while modeling the dynamics. Following the approach presented in [98], there are three main effects besides other celestial bodies that can be considered in the agents of the swarm:

- Spherical Harmonics
- Atmospheric Drag
- Radiation Force

The spherical harmonics are actually not a physical effect (see AS-EN-SW-002). In a first approximation, bodies can be considered point masses. Nevertheless, the Moon, like any other celestial body, is not a perfect sphere of constant density. Rather than that, it is an amorphous shape whose density varies in all the geometry. Clearly, the gravity effects will change with this change in the distribution of the mass. This is addressed by modeling the body's potential with spherical harmonics. Besides the effects of dynamic movements inside the body (e.g if there are internal tectonics or tides), this approach better models the reality of the gravity effects of the Moon. According to the General Potential Theory used to model these effects, the gravitational potential of the effects of a body in a point outside the body can be modeled as in Equation 3.3.

$$U = -\frac{\mu}{r}\Big[1 - \sum_{n=2}^{\inf} J_n\left(\frac{R}{r}\right)^n P_n(sin\phi) + \sum_{n=2}^{\inf}\sum_{m=1}^{n} J_{n,m}\left(\frac{R}{r}\right)^n P_{n,m}(sin\phi)\left\{cos\left(m(\Delta - \Delta_{n,m})\right)\right\}\Big] \tag{3.3}$$

Where μ is the standard gravitational parameter, r is the distance from the center of the body to the affected point, ϕ is the latitude, Δ is the longitude (so r, ϕ, Δ are the spherical coordinates of the

considered point), R is the mean radius of the body, $P_{n,m}$ are the associated Legendre functions of the first kind of degree n and order m, P_n are the Legendre polynomials of order n,and J_n,$J_{n,m}$,and $\Delta_{n,m}$ are parameters of the model [98]. The first sum represents the zonal harmonics, i.e. longitudinal bands of mass distribution in the sphere, whereas the second one represents tesseral and sectorial harmonics, bands from pole to pole and square patterns all around the sphere. The perturbing acceleration is obtained with Equation 3.4.

$$f = -\nabla\left(U + \frac{\mu}{r}\right) \tag{3.4}$$

The full model of the harmonics for the gravity field of the Moon can be found in [82]. Although not exactly as oblate as the Earth, the most relevant harmonic of the Moon is the J_2 zonal harmonic, representing how oblate the sphere is. It is thus possible to calculate the perturbing acceleration of said harmonic component by substituting the J_2 term $\sum_{n=2}^{\inf} J_n \left(\frac{R}{r}\right)^n P_n(sin\phi)$ of Equation 3.3 in Equation 3.4. The necessary calculations are presented in Appendix A.

The second perturbation to take into account is the solar radiation pressure. This is the effect of the radiation, both incoming from the Sun and reflected by the Moon, on the spacecraft exerting a perturbation force. This effect is labeled as hard to model [93]. Since the only goal is to estimate its effect, the approximation presented in [98] will be used. Nevertheless, for more precise calculations in [93] a more comprehensive description of the effects of solar radiation pressure is presented.

Waker [98] approximates the solar radiation perturbing acceleration as:

$$f = -C_R \frac{WA}{Mc} \tag{3.5}$$

Where W is the energy flux, A is the reference area, M is the mass of the spacecraft, C_R is the reflectivity coefficient, and c is the speed of light in vacuum. An estimation of all these values is presented in Appendix A.

The result of the calculations presented in Appendix A to estimate the perturbing accelerations is presented in Table 3.4

Effect	**Estimated Value** (N)
J2 Effect	$6.384 \cdot 10^{-4}$
Radiation Force	$4.564 \cdot 10^{-8}$

Table 3.4: Estimation of the perturbation forces on the dynamics of the swarm as calculated in Appendix A

The results presented in Table 3.4 show that the most significant perturbation is the J_2 term of the gravity field. This makes sense as, as shown in Section 3.4.1 the gravitational attraction of the Moon is the main source of forces. The radiation pressure is a less dominant perturbation, but this was expected as, according to [98] the same force for the EchoI mission, a mission much more affected by it, was in the order of mm/s^2.

It is worth noting that one of the common perturbations to be taken into account with Earth orbits [93] has been omitted in this section, the atmospheric drag. As seen in [103], the atmosphere of the Moon is extremely tenuous. Since the effects of the atmospheric drag depend on the density of the atmosphere, it is not expected that at any orbit the effects of the atmospheric drag surpass those of already neglected effects such as solar radiation pressure or gravity harmonics.

3.4.3. Conclusions on the Neglected Effects

After the evaluation of the most relevant neglected effects, it seems clear that the assumptions taken are justifiable. No forces have been found to have an effect even close to the one of the main source force, the gravitational pull of the Moon. Therefore, the precision of the used model is expected to correlate well with reality. If further effects where to be modeled, it is recommended to start by taking into account the most dominant terms of the harmonic gravity field, as they seem to generate the biggest perturbations on the trajectory.

3.5. Note on Implementation

The modelling here presented is the one implemented in Swarm Simulator. A folder called Dynamic Models is included in the SwarmSimulator program. There, it is possible to define a new dynamic modelling by including a new file with a function defining the dynamics. By creating a set of functions that, given the necessary inputs for the model, output the state at the given time step. This will be later used by the simulator to propagate the dynamics. If the model has not an analytic solution, such as the HCW, a propagator will be required to integrate the dynamics. A set of interesting integration methods are presented in [93]. For more details on how the environmental model is inputted in the simulator, see Chapter 2.

4

Hardware Design

This chapter will be dedicated to the description of the agents and their included hardware. As in the previous chapters, the OLFAR mission's developed systems will be used as a reference point. The OLFAR mission is still under development, therefore many subsystems are still to be determined. No preliminary designs of the whole spacecraft exist in literature, only preliminary ideas for the models of few subsystems such as the antennas and the inter-satellite link systems [19, 20].

In cases where there is a lack of data, assumptions based on previous missions with small spacecraft and current state of the art technologies will be taken. Key subsystems for the GNC system of the spacecraft will be analyzed. A small review of the possible navigation systems will be included, although in this project it is considered out of the scope the simulation of navigation effects, as multiple solutions with sufficient accuracy are already available in the market.

4.1. Evaluation of the Elements to Analyze

The subsystems analyzed on this chapter are the ones that DESHA will need to be deployed and used to swarm the spacecraft. These can be summarized in the guidance, navigation and control of the spacecraft on top of the communication within and with the swarm and processing information. The identified subsystems that allow the spacecraft to develop these tasks are:

- **Actuators**: these are in charge of taking the control actions required by the GNC algorithms.
- **Inter-Satellite Links System**: in charge of ensuring the communication in between the agents and between the agents and the ground station. It will also take charge of the navigation side by determining the relative position of the spacecraft.
- **On-Board Computer**: in charge of receiving all the sensor's information, processing it and generating the output commands to the actuators.

The communication system and the inter-satellite link have been set together given their overlapping. In the following sections, each of these subsystems will be analyzed. In each section possible design options are offered, using solutions presented either in previous missions or from the COTS components currently offered. This does not mean that the final solution here presented is the one chosen for the reference mission nor that a mission using DESHA should include the selected components. Nevertheless, it will allow to simulate the performance of DESHA with variables equivalent to those of a real tentative mission. First, a list of all assumptions taken will be presented to give a full overview of the tentative capacities of the spacecraft assumed. Finally, a note on how the hardware models have been implemented in SwarmSimulator is presented.

4.2. Assumptions on the Hardware

This section presents a compendium of all the assumptions taken with respect to the hardware. Some of them will be used to ease the choice of a certain hardware. Others will flow down after the design

choice of the given subsystem. Those taken to ease the choice of the hardware or with general purpose will be explained at the end of this section. The ones flowing down from the analysis on design options or current status of the reference mission will be explained in their respective sections.

- **AS-HW-001** : The agents will be actuated in all three axis.
- **AS-HW-002** : The agents will always have the required attitude.
- **AS-HW-003** : The swarm agents will be located at the start of the analysis nearby the reference orbit.
- **AS-HW-004** : The swarm will be composed of equal agents.
- **AS-HW-005**: During the uplink of information at least one agent will be facing the Earth for enough time to uplink the reference orbit information.
- **AS-HW-006**: The inter-satellite link will be able to send all necessary information for the swarming algorithm.
- **AS-HW-007**: The inter-satellite link system will allow for ranging with a precision up to 1m.
- **AS-HW-008**: The inter-satellite link system will allow for ranging with angular precision up to 15°.
- **AS-HW-009**: The inter-satellite communications will be possible at all times during the mission.
- **AS-HW-010**: The agents will have all the necessary navigation information at all times.
- **AS-HW-011**: The on-board computer will be able to process, store and generate all necessary data.
- **AS-HW-012**: It is supposed that the thrust is always varied through the variation of the mass flow of the engine.
- **AS-HW-013**: It is supposed that the mass flow is constant (no need for a variation of the current to compensate for variations in the plasma flow)

Assumption AS-HW-003 is more of an operational assumption which will ease the choice of the propulsion options. It is expected that the design of the launch of the swarm to orbit will be such that the agents are transferred to the right orbit. It is true that in many cases these maneuvers imply a last impulse when the crossing between the transfer and the desired orbit coincide to equal the energy of the spacecraft to the one of the desired orbit. Nevertheless, in it this work it will be assumed that this has been done before the analysis. Therefore the thrust solutions analyzed should only be responsible of correction maneuvers and the relocation of agents.

Assumption AS-HW-004 is a basic design option in OLFAR. It will ensure the robustness of the system as there are no key elements unlike some other concepts [38]. Nevertheless, it will also put some challenges as everything will have to be done with unsophisticated agents.

Assumptions AS-HW-012 and AS-HW-013 are assumptions of the modeling for some engine calculations. They will be explained in Appendix B.

4.3. Actuators

Actuation systems translate the control actions generated in the controller into real movements. Usually they can be classified in force and momentum actions. The attitude actuation will not be considered in this work, as stated before, since the main goal is to test the capacities of DESHA for agent swarming and specially to generate desired formations. Therefore only thrust actuators will be analyzed.

In this section first the actuation needs and their consequences will be analyzed. Then the results of said analysis will be analyzed against the current state of the art technologies to see if it is realistic to demand said actuations to the spacecraft. Finally, a final conclusion on how the actuators will be modelled in the simulator will be presented.

4.3.1. Actuation Needs

As it will be shown in Chapter 6, the more complex the pattern required, the more number movements are required. This means that for the most complex patterns, an order of magnitude of 10^4 movements will be expected. Furthermore, since we are considering only small sized spacecraft (100kg or less), the storage available for propellants will be limited. This context leads to a need for extremely efficient thruster. Usually in rocket propulsion this is measured through the specific impulse(I_{sp}) which measures impulse (force integrated over time) generated per weight of the propellant used. Higher specific impulses will mean higher efficiency of the propellant use.

Usually the best specific impulses are obtained with electric thrusters, which reach specific impulses of thousands of seconds versus the usual tenths to hundredths of other systems [92]. Nevertheless, these come at a cost of an extremely low thrust force. This is not problematic. Unlike in launch where the thrusts has to be high to overcome other forces, in this case prolonged switched times allow for generating the maneuver although the thrust level is low. Nevertheless, the lower the thrust level, the longer the maneuvers will take. This together with the high number of maneuvers necessary in some cases (see Chapter 6) will generate extremely long times to achieve formation.

Finally, one last element to be considered in the selection of the actuators is the total mass of the system. In the end extremely heavy systems will require even higher levels of thrust to be moved at the same velocity than lighter systems. Obviously, the lighter the system, the faster it will move for the same level of thrust. Since the total mass of the system will depend on the design of the spacecraft, it is quite hard and useless to do any kind of estimation on this. Furthermore, as the goal of this work is to evaluate the suitability of DESHA for spacecraft swarming, it will be useless to constrain the analysis only to a certain level of mass for a spacecraft. To solve this, the mass variable will be coupled with the thrust level through the definition of the acceleration level of the system.

Therefore, the actuation needs of the system will be:

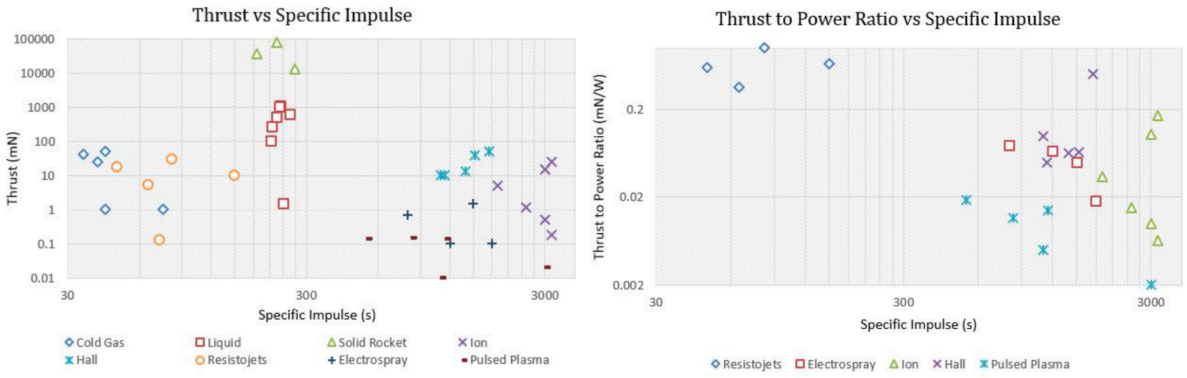
- High Level of I_{sp}. The higher the I_{sp} offered by the system the better, as it will reduce the required amount of propellant for the mission.
- Sufficient Level of Acceleration. The higher the acceleration attainable, the less time will the system require to achieve formation.

In the following subsection a small review on the current state of the art thrusters for the spacecrafts considered is presented. The idea will be to find which levels of I_{sp} and acceleration can be expected from the spacecraft. Specially CubeSat and nanosat platforms will be reviewed as they tend to use more standardized solutions through the COTS. Larger sized spacecraft considered, such as microspacecraft tend to offer *ad hoc* solutions developed for the mission. Therefore as far as the actuations required are within realistic limits, it is acceptable to assume that the required actuations will be attainable.

4.3.2. Current State of the Art of Actuators

Several reviews for COTS technologies on Cubesat and nanosat propulsion technologies are presented in the literature [81, 92]. When looking at the multiple options and characteristics, it is clear that the ones that best satisfy the need for a high I_{sp} are electric engines. In Figure 4.1 a comparison for different kinds of electric propulsion for Cubesat platforms is presented for reference. Even though some of these solutions are designed for CubeSats, due to their size and power budgets, the most powerful of them seem actually more fitted to larger spacecraft. In any case it is seen that these systems offer an extremely high specific impulse, meaning an extremely high efficiency. However this comes at the cost of a low thrust level. Therefore, the resulting acceleration will be low.

Other solutions in the literature trade I_{sp} for thrust. For example chemical engines achieve several hundreds of seconds of I_{sp} [81, 92, 106] and depending on the size they can achieve from tenths to several newtons of thrust. Another popular choice is the use of cold gas thrusters. The classical design releases pressurized gas from a tank to generate the thrust, achieving also thrust levels of newtons [106]. The COTS cold gas thrusters aim for a different approach, pressurizing the gas in its release to the vacuum of space to avoid having pressurized containers inside the spacecraft. Their designs also



(a) (b)

Figure 4.1: Survey of different types of electric thursters as presented in [92]. In Figure 4.1a a comparison between Thrust and I_{sp} of different electric thrusters is presented. In Figure 4.1b the same thrusters are presented, but this time the thrust to power ratio is compared with the specific impulse.

offer lower I_{sp} and, of course, due to their size, lower thrust levels (around milinewtons) [24].

Overall, the literature shows that levels of thrust ranges from milinewtons to several newton depending on the size of the propulsion system. On the side of the I_{sp} it is seen that levels between tenths of seconds to thousands of seconds are available, depending on the kind of propulsion selected.

4.3.3. Conclusions on Actuation

Several options for thrusters are available for the considered spacecrafts. In the previous section it was identified that thrusters with high I_{sp} will be necessary to achieve the required number of actions with reasonable amounts of propellant. Also, the higher the thrusts level, the better as it will allow to achieve the desired movements in the least time. Nevertheless, in order to fully explore the capabilities of DESHA a final choice or assumption on the thrust level will not be made. This is due to the fact that also the mission constraints will play a role in the selection. For example larger inter-satellite distances will generate a need for higher thrust levels to achieve the required number of movements in an acceptable amount of time. This might even come at the cost of lower I_{sp}. Instead, in the third part of this report an analysis on different acceleration levels (all coherent with the actuation capacities here presented) will be done. This will allow to see the possibilities of DESHA according to the tentative mission design. Also, analysis done based on real engines and tentative spacecraft will be done to analyze the propellant use of the mission.

Overall only three major assumptions will be taken with respect to the actuators of the spacecraft:

- **AS-HW-001**: The agents will be actuated in all three axes.
- **AS-HW-002**: The agents will always have the required attitude.
- **AS-HW-003**: The swarm will be located at the start of the analysis nearby the reference orbit.

The first assumption allows to generate control systems for each of the directions. This does not necessarily mean that one thruster in each direction is available as it can also be achieved with attitude changes and one thruster. This assumption nevertheless allows to simplify the analysis leaving the final decision on how many thrusters to include or how to achieve the thrusting in each direction to more specific uses of DESHA. The second assumption is taken to avoid the design of the attitude control, as this is deemed out of the scope of this project. The final assumption links with the fact that on this analysis only the operations of DESHA are taken into account. This means that only movements around the reference orbit will be taken into account, not transfer orbits.

4.4. Inter-Satellite Link/Communications System

The inter-satellite link system is responsible for three key functions to the swarm:

- Communications with the Ground Station
- Inter-Satellite Communications
- Ranging

This section will analyze how these three functions will affect DESHA. The needs of DESHA to satisfy these three functions will be analyzed in Section 4.4.1. Then, a review on systems for the considered spacecraft will be presented in Section 4.4.2 to analyze if current technologies allow to satisfy the needs presented in Section 4.4.1. Finally, in Section 4.4.3 the conclusions on the communication needs and assumptions taken will be presented.

4.4.1. Inte-Satellite Link/Communication Needs

Regarding the three functions presented in the introduction, the last two are specially interesting for DESHA. The possibility to know the state of the neighbouring agents is necessary for the algorithm. Ground communications is clearly necessary from the perspective of sending back the data retrieved by the swarm. However given the multiple possible missions for the swarm, no specific need is generated for DESHA out of it. In the context of the reference mission nevertheless, the need will be necessary to have a sufficiently powerful antenna to communicate back the swarm in the reference orbits presented in Chapter 3 with the ground station. Some work on the ground communications of the reference mission of this work is presented in [19] for further reference.

The inter-satellite communications between swarm agents is needed to be able to sense the state of each agent. DESHA requires to know the surrounding neighbourhood. Therefore, it is fundamental for the inter-satellite communication system to have an equal or similar range to the sensing distance required. This will be closely related with the state definitions presented in Chapter 6. A state definition requiring more positions to be sensed or more distant positions will require a larger communication range. Also a state definition requiring sensing in a single direction will ease the communication needs as compared with one that requires omni-directional sensing.

Related with inter-satellite communications will be the ranging between satellites. With the ranging also the sensing will taken into account. This means that the communication system is not only responsible of measuring the inter-satellite distance, but also of sensing the orientation with respect to a reference frame fixed in the agent. The needs for the precision of these measurements is again related with the definition of the inter-satellite distance and the state shape selected, as explained in the previous paragraph.

The inter-satellite distance can be assumed to be in the range of other formation flying missions with similar spacecraft. On the low level limit, it is possible to find close formation flying CubeSat based missions, with inter-satellite distances of the order of 50m (see [14]), although larger spacecraft aim for distances between hundreds of meters to kilometers [19].

The spacecraft need to be controlled in their position to avoid drifts and collisions. This means that the agents also need to be able to range themselves with others to maintain a control window narrower than this inter-satellite distance. This control window will be estimated in at least one order of magnitude below the inter-satellite distance. That is, in the lower limit the ranging will need to have an order of magnitude around 1m.

Finally, with respect to the sensing different agents around, this will require to generate a need in the precision of the sensing of the direction of arrival of the signal. Again, this will depend on the state shapes selected. For example, a state shape formed of a square, as the one presented in [31] or in Chapter 6 will require the agent to distinguish the direction of their neighbour in the plane with a precision shorter or equal than 22.5°. This is due to the fact that the states are separated equally each

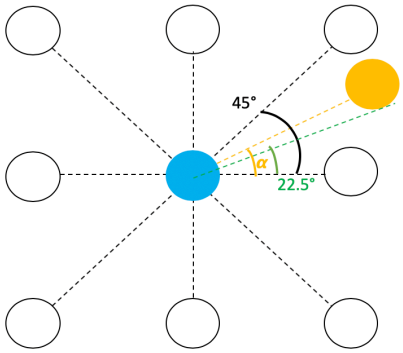


Figure 4.2: Representation of the angular accuracy for a square state shape. The blue agent has to difference the angles,α, with an accuracy equal or smaller than 22.5°to classify the yellow neighbour

45°. For the case of the hexagonal state presented in Chapter 6 this will be reduced to 15°. This is illustrated in Figure 4.2 for a square shaped state.

Therefore, the needs of the communication system will be:

- The inter-satellite communication range should be larger or equal to the maximum inter-satellite distance of the selected state shape.
- The inter-satellite communication system should be able to range agents with a precision of meters.
- The inter-satellite communication system should be able to sense the surrounding agents with a precision below 15°.

Additionally the communication system should be able to communicate with the ground station. This will be necessary to download the scientific data and perform housekeeping tasks and upload telecommands to the agents.

In the following section an analysis of the state of the art components and solutions of previous missions will be performed. This will allow to evaluate if these needs are possible to be satisfied by current technology for the considered spacecraft. As in Section 4.3 special attention will be focused to COTS solutions and CubeSat/Nanosat spacecraft, as the considered spacecraft spacecraft outside these categories tend to use *ad hoc* designs tailored for the mission.

4.4.2. State of the Art of Inte-Satellite Link/Communication Systems

In this section a review of the state of the art for communication systems of the considered spacecraft will be done. It will be analyzed if the needs presented are possible to be satisfied with current technologies. Even though these are not specific needs for DESHA, communications with the ground will also be analyzed first. Then inter-satellite links are analyzed to finish with the analysis of ranging.

Ground Communications

Ground communications between agents are quite advanced for the considered spacecraft. Current state of the art systems allow even CubeSat to actually communicate in deep space missions. For example the Lunar IceCube mission incorporates a high gain antenna to access NASA's Deep Space Network and communicate with the ground station [28] (see Figure 4.3). Therefore bigger spacecraft can also incorporate this solutions to actually communicate each agent with the ground station.

This will also allow the agents of the swarm to receive information from the ground, such as orbit determinations or housekeeping tasks.

Inter-Satellite Communication

On the inter-satellite communication side, also solutions are available even for the smallest systems considered. For example for the reference mission a system based on 6 patch antennas (one in each

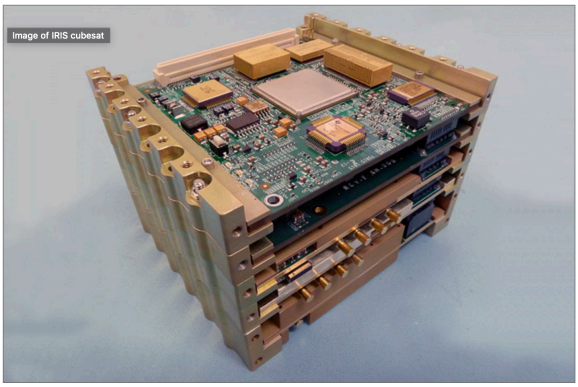


Figure 4.3: JPL's Iris X-band radio for CubeSats. With a size of half a CubeSat U and 35W of power consumption allows for CubeSats to be tracked and navigated with NASA's DSN. Image from [22]

side of the spacecraft) allows for a 90km range of communication between the agents [20]. In this work it is also noted that some missions might need to go silent due to legislation limitations. DESHA does need to maintain communication with the other agents at all times to ensure if the pattern needs to be maintained (unless the drift does not exceed the size of the control window during the no communication time). Therefore an assumption will be needed to take this into account.

Overall, the inter-satellite distances will not reach, in the projects found, more than a couple of kilometers. For the smallest systems considered, examples of communication ranges larger than this distance exist. It can be assumed that the same systems fit in larger spacecraft Therefore it can be said that there are technologies of the state of the art already achieve the needs of inter-satellite communications of DESHA.

Ranging and Angular Measurements

Finally for the ranging and angular measurement of inter-satellite distances several approaches exist. Examples in the literature) use differential GNSS systems to determine their relative position with great accuracy [14]s. Nevertheless, GNSS systems are not needed to be used with DESHA. The next most common approach is the use of radio-signals to determine the position of other agents [85] mimicking the use of GNSS systems.

With respect to the ranging several factors affect the measurement. For example the code used, the signal to noise ratio of the measurement, etc. Still, an approximation can be set with the measurement of the propagation time of the signal. A radio signal is sent from a neighbour to the agent with the time stamp of when it was sent. Since the propagation speed is known (the speed of light in vacuum), if the time elapsed between the sending and the arrival of the signal is computed, the distance between the agents can be estimated. The better the precision of the time measurement, the better the precision of the ranging. Since the speed of light (c) is of order $10^9 m/s$, then a precision of nanoseconds in the time measurement will be necessary to achieve distance measurements on the order of meters. Further details in the implementation of these systems can be found in [84]. Such precision also require a clock with both said nanosecond level precision and synchronization, as all clocks in the swarm need to have the same time up to nanosecond level. Nevertheless both of these are already achieved as presented in [77]. In this work, an algorithms to estimate the offset between clocks setting one of them as reference is presented, allowing for the synchronization of the whole system. Also, nowadays, there are lightweight atomic clocks designed even for extremely small platforms such as Nanosats [29]. These instruments allow for time measurements up to nanosecond levels, therefore achieving ranging according to the needs specified in Section 4.4.2.

Finally, according to literature, the use of Smart Antenna approaches, as those presented in [66] allow for precisely obtaining the direction of arrival of the signal by making use of the multiple antennas. As in any kind of the considered systems other works prove that several patch antennas can be added [20], this approach can be taken in all cases. Also the work of [95] shows that if instead of one patch

antenna the antenna is divided into two patches, the estimation in each antenna of the angle of arrival is possible with degree accuracy. Furthermore, estimations below these levels are possible with non-bi-dimensional antenna systems [70]. These techniques allow for precision of degrees. Therefore it is concluded that current state of the art techniques allow for satisfying the communication needs of DESHA in angular measurements.

4.4.3. Conclusions on Inte-Satellite Link/Communication

After the identification of the communication needs of DESHA, the review of the state of the art techniques and hardware available for small size spacecraft has shown that these needs are achievable with the current technology. Therefore, it is possible to make the following assumptions with respect to the communication of the agents of the swarm in this work:

- **AS-HW-005**: During the uplink of information at least one agent will be facing the Earth for enough time to uplink the reference orbit information.
- **AS-HW-006**: The inter-satellite link will be able to send all necessary information for the swarming algorithm.
- **AS-HW-007**: The inter-satellite link system will allow for ranging with a precision up to 1m.
- **AS-HW-008**: The inter-satellite link system will allow for ranging with angular precision up to 15°.
- **AS-HW-009**: The inter-satellite communications will be possible at all times during the mission.
- **AS-HW-010**: The agents will have all the necessary navigation information at all times.

Assumptions AS-HW-007 and AS-HW-008 will cover the ranging needs specified in Section 4.4.1. Assumption AS-HW-005 takes into account the ground communications. It is also taken since the reference orbit is also used in the simulator to save the current position of the agents. Although relative positioning between agents can also be used, since the reference orbit was needed for plotting purposes, it is also used for positioning. Then assumptions AS-HW-006 and AS-HW-009 are taken accordingly to the inter-satellite communication needs identified. Finally, in order to keep up with Assumption AS-HW-002, Assumption AS-HW-010 will be taken. Then it is supposed that the attitude is known and therefore always controlled and kept at all times.

Overall, the communication system allows for the transmission of the necessary information between agents and the generation of the agent state. These needs have been identified as attainable with current state of the art techniques for all kinds of spacecraft analyzed in this work.

4.5. On-Board Computing

The on-board computing is responsible with respect to DESHA of:

- Processing all the sensors information.
- Running the DESHA algorithm.
- Running the low-level control algorithms.
- Generating the necessary responses to use the actuators.

This section will analyze the needs for the communication system to perform all these tasks. In Section 4.5.1 the on-board computing needs will be analyzed. Then in Section 4.5.2 a review of the state of the art on-board computing systems for the considered spacecraft will be presented. Finally in Section 4.5.3 a set of conclusions on the on-board computing systems will be presented and the assumptions taken will be motivated.

4.5.1. On-Board Computing Needs

The on-board computing system is responsible of the processing the input information and the generating the required commands to the actuators. This task has to be done fast enough to be able to control the system. The spacecraft will be moving in orbit, many times drifting from the position. The control has to be fast enough to detect when the spacecraft is out of the control window designed and actuate it to restore its position. Also, each time that a control action is taken, the system will vary its state several times until achieving the desired new state. If the control system is not fast enough, these actions will lead to an oscillatory or divergent behaviour as control actions will not be taken to stabilize the agent in the desired position. Therefore the on-board computer will need to be fast enough to take sufficient control actions fast enough to control the dynamics of the system, leaving time for the actuators and sensors to generate the necessary inputs and outputs.

Of course the specific processing speed will vary depending on the final mission characteristics. Larger inter-satellite distances and control windows will mean that the speed required will be less as there will be more time available. On the other hand the larger the thrust available, the faster the dynamics of the movements and the faster the controller needs to be. Therefore there is not a clear specific need for the controller when using DESHA. Overall the main on-board computing system for DESHA can be expressed as:

- The on-board computer needs to process all the information and generate the necessary commands fast enough for the low-level controller and DESHA to control the spacecraft.

Of course more needs will be derived from the processing of the scientific information, the attitude control system, etc. but these are considered out of the scope of this project as they do not affect DESHA. In the following section a small review of the state of the art on-board computers will be given to analyze if the considered need is achievable with state of the art hardware. Also spin-ins from other fields will be presented given their possible relevance. The next section will be mostly focused again on the smallest systems considered (CubeSats and Nanosats). It is assumed that if these systems can incorporate technologies that allow the needs to be satisfied, then larger systems could also either mount the same systems or better achieve the same needs.

4.5.2. On-Board Computing State of the Art

Space processors and on-board computers are less powerful than the ones mounted in current computers. This is due, among many other factors, to the extra qualification needs that the space environment requires, such as the need for surviving constant radiation events. For example COTS for CubeSats and Nanosats mount processors with processing speeds around MHz and memories of MBs [21]. Nevertheless, previous missions have achieved formation flying between two agents by using several of these systems on board [14] to both process the information and control the formation.

Still the most interesting technology is yet to come. In the latest years telecommunication systems have been improving the performance of processors that fit on motherboards of small systems such as phones or tablets. Nowadays it is possible to find several processors with comparable capacity to the one of an average computer fitted in a small device [40]. It only seems logical that these technologies will be soon adapted also on the world of spacecraft. However, some of the performance might be lost while adapting these technologies to space. It is expected that in not too long all spacecraft will be equipped with on-board computers with similar performance to current smartphones.

Therefore, previous missions show that it is possible to run similar algorithms in the smallest systems considered. This leads to think that current state of the art technologies allow for the performance of all required tasks. Furthermore, spin-ins from other industries are expected, leading to an increase in the on-board computer capacities.

4.5.3. On-Board Computing Conclusions

In this section the needs of the on-board computing system have been analyzed and the state of the art of on-board computing technology has been shortly reviewed. The conclusion is that previous missions have achieved the identified needs with current state of the art technologies. On top of that it is expected

(a)

(b)

Figure 4.4: In 4.4a a modern on board computer for a CubeSat spacecraft from [21]. In 4.4b the motherboard of a modern smartphone from [40]. Certainly size and form of the components is similar.

that the capacities of on-board computers are going to be dramatically increased in the upcoming years.

It is true that the DESHA algorithm has never been tested in space before. Furthermore, to test it and find the true processing and memory needs of the algorithm the code should be programmed in embedded software languages such as C/C++ and the code optimized. This is considered out of the scope of this thesis project due to time constraints.

With all the above mentioned considerations the following assumption will be taken:

- **AS-HW-011**: The on-board computer will be able to process, store and generate all necessary data.

On top of that, it will be assumed that as far as the algorithm can run (for a single agent) in real time in a computer, the algorithm will be considered suitable for current or shortly upcoming on-board computers.

4.6. Note on Implementation

In this section the implementation in SwarmSimulator of the modeling of the agents and hardware presented in the previous sections will be reviewed. This does not intend to be a software manual for SwarmSimulator, but rather an introduction on how the implementation was done.

The hardware of the spacecraft has been modeled mainly through the definition of a class, the *Agent* class. This class encompasses all the values for the parameters determined by the hardware, such as the agent mass, the definition of the state used for DESHA (which links with discrete possible locations of the direction of arrival of the signal obtained with the antennas), etc. Most of the hardware design has been used implicitly in the assumptions taken on the model. For example, no limitations on angle of arrival of inter-satellite links have been considered, as the communication is full according to [20].

The only piece of hardware modeled more explicitly with respect to hardware considerations has been the thrusters. A subclass of *Agent* has been created called *Thruster* which includes all the information both of a catalogue of suitable thrusters (mostly the Busek's BIT family [90]) as well as the possibility of user defined thrusters. This class also includes a set of functions to calculate the use of propellant, the remaining propellant and the total ΔV used. More detail on the internal functions of the class thruster is available on Appendix B.

For navigation, the solution of having the reference orbit transmitted to all the agents has been used. This choice has been done due to the fact that for obtaining plots with the current orbit of each agent with respect to a Moon centered inertial reference frame the swarm reference orbit was necessary. Furthermore, this will link with the work presented in [34]. Nevertheless, this has been done without loss of generality, as with the relative localization option the only difference will be that the agent will not

store a variable with its current state at all times, but rather it will calculate with the proposed ranging methods the state of its neighbourhood. This has been implemented in SwarmSimulator through the *Reference Orbit* class, which calculates at the beginning of the simulation for each time step the coordinates in Moon inertial reference frame of the orbit with Kepler's equation [98]. The class also includes some minor functions to obtain derived values useful for the setting of some simulation parameters or calculations, such as the reference orbit mean motion or period. This class is public so each member of the swarm can access it during the simulation, imitating the fact that all the swarm will have a shared reference orbit.

Finally, two expansion options have been made available in the construction of the *Agent* class: reaction wheel and antennas. For both elements a class file was created which represents a subclass included in *Agent*. In future developments of Swarm Simulator it is possible to use the already defined elements of said classes as well as to include new elements. This will allow, for example, to expand the use of DESHA to attitude control; include radiation patterns for all the antennas and analyze the effects of the smart antenna approach in the definition of the discrete states of DESHA; include in the analysis models of the smart antenna approach and test them, etc. Nevertheless, since all these activities were out of the scope of this project and will not allow to answer the research question, the classes were not used.

5

Low-Level Control

DESHA is a high-level control algorithm. That means that it will generate as an output the necessary state transitions, that is, the new final state required for each agent. Nevertheless, the transition from the current to the new state of the agent will not be managed by DESHA. Therefore, for its proper implementation (presented in Chapter 6) first it will be necessary to have a low-level controller capable of controlling the spacecrafts in said transitions. This chapter presents the proposed solution for the low-level control.

In Section 5.1 the proposed approach and control theory will be reviewed, leading to the selection of a low-level controller. Then in Section 5.2 the tuning of the chosen low-level controller will be presented. In Section 5.3 the tuned controller is validated through a series of tests. Finally in Section 5.4 some notes on how this controller is implemented in Swarm Simulator are reviewed.

5.1. Proposed Approach to Low-Level Control

There are two main approaches to controllers: open loop controllers and closed-loop controllers. The main difference between one and the other is the inclusion of what is known as the feedback loop. In a closed-loop control after taking the control action, there is a measurement of the state of the system, which is fed again to the controller, so the controller has actually a feedback loop analyzing the result of its actions. Open loop control is a particular, much simpler, case of the closed-loop control where the feedback loop is removed [88].

In the case of spacecraft automatic control, and more in the case of control in formation flying, the use of closed-loop systems is the most current technique [4]. Specifically the use of Control Lyapunov Functions and Linear Quadratic Regulators is proposed in [4, 87]. Nevertheless, the latest developments point out Model Predictive Control [17, 68] or the use of simple techniques such as Q-Guidance Control [52, 71] for swarm low-level control.

In the end, the low-level controller will just be, at least in this first approach, a simple tool to perform state transitions. Therefore for this project a simple and easy-to-implement solution was preferred to more complex forms of control. The chosen controller for the low-level control was a Proportional Integral Derivative (PID) controller. This controller works in a similar, but simpler, way than the Linear Quadratic Regulator. It generates a response proportional to the error between the reference (end state) and the current state, its derivative and the integral of the error during the control process. The ratios to multiply the difference error is called the gain. The advantage of the use of the PID is that the system is easily understood, the response is just proportional to the error (and its derived magnitudes). Furthermore, its tuning can be done with a simple ”guess and check” method, although slightly more analytic methods are available in literature to tune the gains of the controller [9].

On the requirements side, a PID requires a linear dynamics system and a controllable plant. Thankfully, the presented dynamic model in Chapter 3 is linear. For the controlability analysis, the model can

be expressed in the form of a state space system (see [100]). This form is presented in Equations 5.1 and 5.2 where **X** is the state vector, A and B are matrices, **u** is the control vector, **y** is the output of the system, and C and D are matrices.

$$\dot{\mathbf{X}} = A\mathbf{X} + B\mathbf{u} \tag{5.1}$$

$$\dot{y} = C\mathbf{x} + D\mathbf{u} \tag{5.2}$$

In this case the state and the output are the same, therefore C is the identity matrix and D is a matrix full of zeros. For the rest of the matrices their values, according to the HCW equations are:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -3n^2 & 0 & 0 & 1 & 2n & 0 \\ 0 & 0 & 0 & 0 & -2n & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix} B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

$$u = \begin{bmatrix} f_x & f_y & f_z \end{bmatrix} \tag{5.4}$$

$$x = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix} \tag{5.5}$$

According to [100] a system is controlable if the rank of its controlability matrix (defined in Equation 5.6) is equal to the number of rows of A.

$$\begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} \tag{5.6}$$

MatLab® includes the functions *ctrl* and *rank* which allow to compute the controlability matrix and the rank of a matrix respectively. By performing this analysis with the state space system presented for the HCW equations the results show that the system is controlable. Therefore, for any initial state x_0 and desired state x_f there will be an output u that will transfer the state from the initial to the desired in a finite time.

Therefore it will be possible to design a response in the form of a PID control as presented in Equation 5.7 which will generate the desired movement. K_p, K_i, and K_d are vectors with the proportional, derivative and integral gains of the system. In the following section the method used for tuning these gains to obtain the desired responses will be presented

$$\mathbf{u} = K_p\mathbf{x} + K_d\dot{\mathbf{x}} + K_i \int \mathbf{x}dt \tag{5.7}$$

5.2. Tuning of the Controller

There are many methods in literature to design a PID control, from the bandwidth and damping rate selection presented in [100] to the use of genetic algorithms to fully optimize the design of the controller [99]. Also, there has been a rise in the use of automatic tuning methods [9] which allow the user to tune the desired controller by just defining the desired parameters of the controller such as rise time or overshoot.

In this case, a second effect has to be taken into account. There is a non-linearity in the system. The actuators of the system, as defined in Chapter 4, have a limit in the maximum thrust possible. This limits the control input. Therefore even if the PID is actually calculating the input, it will be necessary to take into account the effects of the saturation of the actuators.

As explained in [9] this real effect on the PID has the risk of generating an up-winding effect. This effect consist in an uncontrolled increase in the control output due to the accumulation of integral error. This is due to the fact that, although the controller is asking the system for a certain control input to reduce the error, the system is saturated, so the error increases at a faster rate than expected. A second source of risks of failure in the tuning comes from the scale of the control input versus the required new settling point. The use of the electric engine (which was deemed as interesting in Chapter

3) is extremely interesting for this mission as it offers an extremely large specific impulse and available velocity change. This will allow the high-level control to be able to achieve an extremely large amount of movements before the system runs out of fuel. Nevertheless, it comes at the cost of a very low actuation force. This will be reflected in the control input. On the other hand, the high-level controller will generate a new reference position to the low-level controller. This will be translated in an instantaneous change in the error, in the same shape as the response to a step. Given again the saturation of the low-level controller, integral error will even increase more as the errors accumulated will be bigger and bigger. Nevertheless, as noted in, [64], including the integral term will get rid of steady state errors.

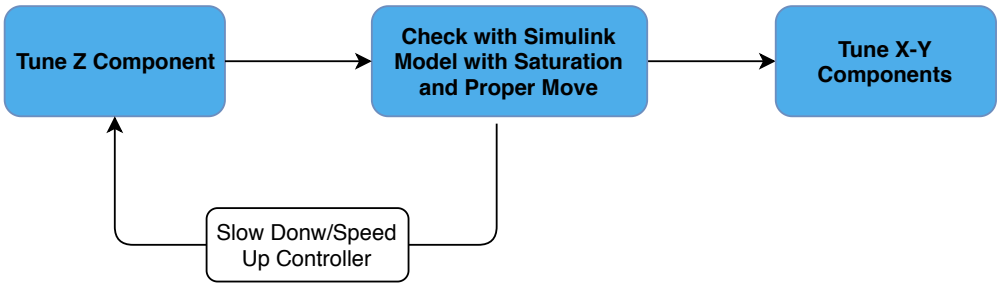


Figure 5.1: Flow diagram of the PID tuning strategy implemented

The final design to tune the controllers used in this report consisted in an iterative method depicted in Figure 5.1. It consists on four steps:

1. **Tune with PID Tuner App**. MatLab® includes a *pidtuner* application which allows both tune and preview a PID controller. The only limitation to this tool is the necessity of using a single-input single-output (SISO) controller. This is not the case of the HCW equations. The proposed approach is to use the z coordinate of the HCW equations, which is independent and can work as a SISO system, to be tuned with the PID Tunner App of MatLab®.

2. **Use Simulink® Model to Include the Saturation**. Simulink® allows to actually take into account the saturation of the controller in its PID Controller block. The solution of the previous step is inputted in a Simulink® model (presented in Figure 5.2) of the z coordinate with the controller. Then it will be evaluated if the designed gains are able to cope with the given movement and with the saturation of the actuator.

3. **Iterate Between Steps 1 and 2**. If the controller is not capable of controlling the system with the saturation, the controller is slowed down to reduce the wind up problem. Opposite to that, to speed up the controller, it will be sped up if it is possible.

4. **Tune X-Y Components**. Once the cross-track component has been tuned, the same solution is inputted for the radial and along track components. Then, with the low-level control routine from SwarmSimulator a series of tests moves are created to see if the controller can generate the desired state transition. If it is not the case, the gains are hand tuned according to [64] to obtain a better response.

The generated controllers will be validated. An example of the validation procedure will be presented in Section 5.3. To do so the error between the current and desired positions will be plotted, which will map the response of the controller for the system used in Chapter 7. This will allow to identify parameters such as rise time or the level of overshoot. Nevertheless, before that, the limits of the controller have to be explored so only possible moves are inputted to the controller. This will be done in Section 5.4.

5.3. Limitations of the Control

In ideal systems, the low-level controller will generate the right output to obtain the desired positioning required by the high-level controller. Nevertheless, the astrodynamic modelling done in Chapter 3 and

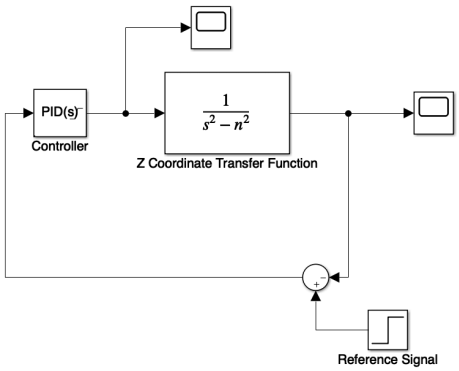


Figure 5.2: Simulink®model of the cross-track coordinate of the HCW equations for tuning.

Chapter 4 poses a set of real constraints to the possibilities of the low-level controller. The limits that these effects pose on the control problem are presented in this section.

In the end, the controller is mostly affected by the saturation of the actuators. The existence of a real limit on the maximum (in absolute value) that the controller can actuate the system generates a need for a tuned controller taking into account the possibility of wind up. It also poses a limit on the locations attainable by the controller.

An analysis on the HCW Equations from Chapter 3 proves that there will exist a certain movement with respect to the reference orbit for all agents not located within it. In other words, the agents will move with respect to another if they are not actuated. This is not totally true, as orbit designs exist where the agents will not drift apart, at least in a first analysis (see [4, 55]). Nevertheless, given the probabilistic nature of DESHA, it will be quite hard to only generate movements that only lead to non-drifting orbits. Furthermore, it might generate an over-constrained state-space for the high-level controller to only consider these orbits. This might lead to few designs of the swarm actually converging when using DESHA (see Chapter 6. If an unconstrained space is desired for the agents to move, the ability of the actuators to counteract these drifts will pose the constraints in the space available for the agents to move.

To illustrate this idea, the calculations for the case of OLFAR, its reference orbit, and the actuator BIT-3 used in Chapter 7 will be presented as an example. Given acceleration in the cross-track direction within the HCW problem can be computed with Equation 5.8 [98].

$$\ddot{z} = f_z - n^2 z \tag{5.8}$$

An agent is supposed to be left static at a certain point in the cross-track axis z_0. In order for the actuator to keep the agent static, the agent should be able to generate an acceleration capable of being equal to the drift force:

$$f_z = n^2 z \tag{5.9}$$

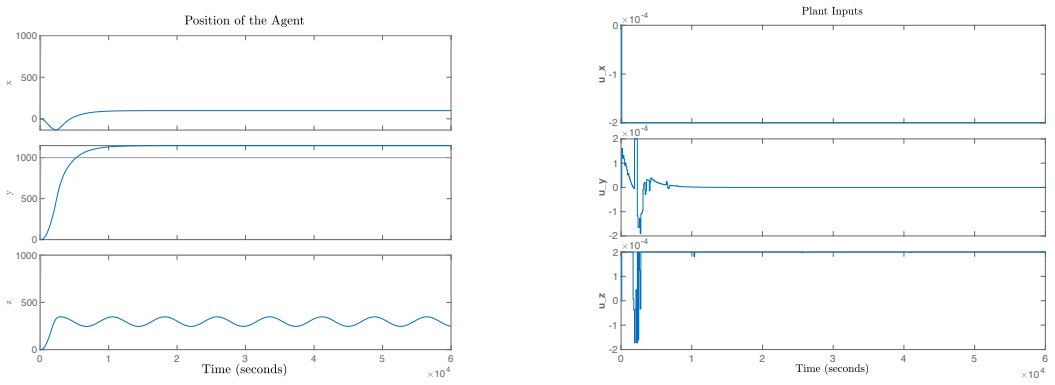
Therefore the maximum z coordinate achievable can be defined by Equation 5.10. As the equation is symmetric, the minimum achievable coordinate will be the same in absolute value, but with the opposite sign. This is reflected in Equation 5.10.

$$|z_{max}| = \frac{|f_{zmax}|}{n^2} \tag{5.10}$$

As the actuator presented in Chapter 7 has a maximum thrust of 0.0012 N and the design is a 6kg Cubesat, the maximum acceleration possible is computed as:

$$f_{zmax} = \frac{T_{max}}{M_{system}} = 0.0002 m/s^2 \tag{5.11}$$

Therefore, by applying this to Equation 5.10 the maximum controlable cross track coordinate will be $z_{max} = 296.82m$ for an orbit of 200km of altitude around the Moon (the worst case scenario of the two presented in Chapter 3). This was also verified by using MatLab®'s MPC Designer App. MPC is a type of controller which allows for taking into control input limitations and also looks for optimizing the movements and control actions [68]. Although MPC was deemed too complex for the current application, its designer toolbox is extremely useful to find the maximum when control input saturation is considered. A user case defining as variables to optimize only the positions and setting the limit of the acceleration as defined above is created. Then a step of 1000m was inputted as reference signal. As this is unattainable, the controller must stabilize in that position where the control effort equals the disturbance acceleration. Results are presented in Figure 5.3.



(a) (b)

Figure 5.3: Results with MPC Model on Saturated Controller and HCW Equations. In Figue 5.3a the position calculated with the control loop including the MPC model are presented. In Figure 5.3b the control inputs calculated with the controler are presented. The results show that there are maximums in the control of both the radial and cross track components.

In Figure 5.3 the control output is depicted. It is shown that, the cross-track component, as exposed analytically, saturates around 300m for the mean motion of the Orbit 2 presented in Chapter 2. Since the drift due to positioning is variable (see Chapter 3) this will probably explain the oscillations in the component. Therefore this tool can be used to obtain the control limits for along track and radial components, whose coupling complicates the analysis. In the case of the along track direction it seems that it is always to attain the required control. This is due to the fact that its acceleration is not dependent on the position of any component. Analyzing it as a second order linear system there is no restoration force in this coordinate, only damping, which leads to the offset. Nevertheless, it has a collateral effect on the radial component, so the overshoot of this component depends on the value of the desired final state of the along-track component. Therefore in the design this should be taken into account, as extremely large overshoots will actually lead to disconnections of the agents, which is not acceptable (see Chapter 6). The control limit for the radial component is around 100m. This is logical as, taking only into account the restoration force of the first HCW equation the formula to calculate the maximum radial control will be:

$$|x_{max}| = \frac{|f_{xmax}|}{n^2} = 98.94m \tag{5.12}$$

Just as predicted with the MPC Designer App.

With respect to the control input, displayed in Figure 5.3b it is possible to see that the control is limited to the designed maximum value. Also it is shown that in both cases where the desired value is not achieved the control input is set to the maximum for the whole time. On the other hand the along-track coordinate, the reference is achieved without any trouble and there is no need for more control input.

In conclusion, the given combination of dynamics modeling and actuators limits the capacities of the controller. This will inherently limit the attainable size of the swarm as agents situated in positions further away from the given limits will drift apart from the swarm. This could be modified by adopting different propulsion solutions, but as it will be exposed in Chapter 6, the high-level controller requires an extremely large amount of actions (movements) to achieve the desired formations. This means that

high changes in velocity will be needed. For swarms of larger sizes different actuators will be needed.

It is also concluded that the smaller the mean motion of the orbit, the larger the controlable area. That is why from now on only the orbit of 3000km in altitude presented in Chapter 3 will be used.

5.4. Limits of the Controller Verification

To verify all the findings in the previous section, a set of moves have been designed and their errors have been plotted. The movements have been designed taking into account the limitations presented in Section 5.4 too. To this end the reference orbit 1 will be used, as well as the system presented in the previous section, as it was used for the analysis already done.

The first test is presented in Figure 5.4. On it, an agent situated in the origin of the LVLH reference frame is given an instruction to move in the radial (Figure 5.4a), along-track (Figure 5.4b) and cross-track (Figure 5.4c) directions. In all cases the controller is capable of bringing the agent to the target without overshoot or oscillations.

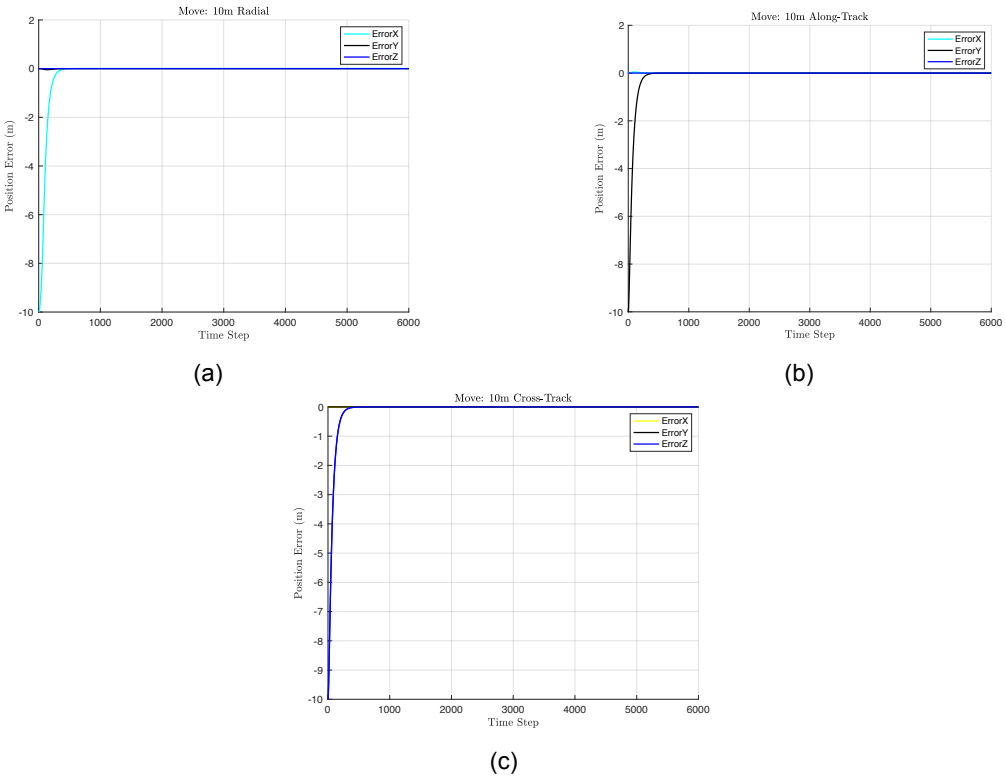


Figure 5.4: Low-level controller verification test. A movement of 10m from the origin has been inputted as the new reference in the radial 5.4a, along-track 5.4b and cross-track 5.4c

Once the short movements have been tested, the next step is to test longer movements close to the limits of the controlable area. To do this the same test is repeated, this time with a movement of 100m. The results are shown in Figure 5.5. In this case, being so close to the limit of the radial component, a noticeable overshoot appears in the radial direction.

Finally the system is brought to almost its limits as presented in Section 5.4 in Figure 5.6. It is shown how the radial direction presents extreme oscillations, while the cross-track and along track are able to attain the desired state with little or no overshoot.

Following this test of the limits calculated, in Figure 5.7 the agents are situated in their limit positions as the initial state. Then a small movement is requested for cross-track and radial directions. In the case of the along-track direction, an extreme initial position is given to perform the analysis. In Figure 5.7a it is shown how the radial direction explodes, whilst in Figure 5.7c it is shown how the cross-track components starts to oscillate. On the other hand in Figure 5.7b the along-track component is

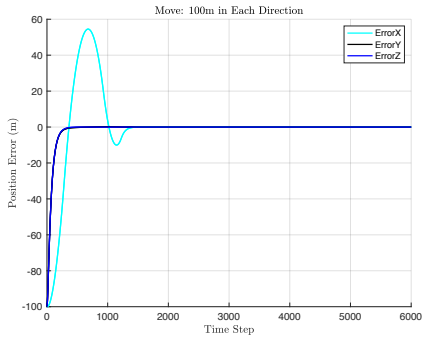


Figure 5.5: Long distance move. A movement of a 100m in each direction is requested.

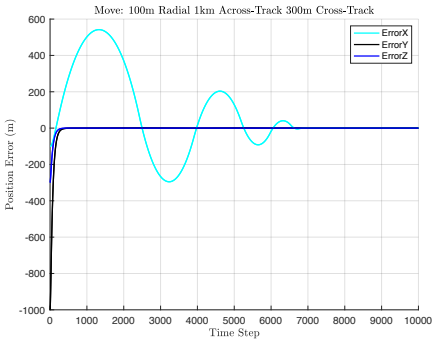
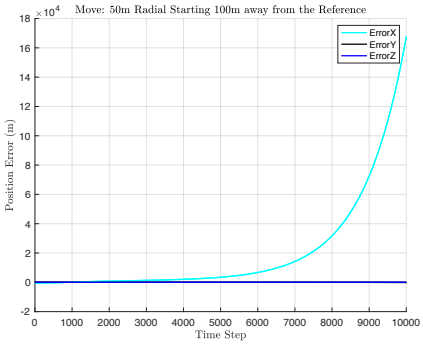
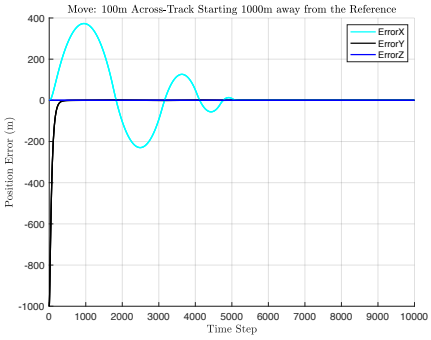


Figure 5.6: Low-level control of a combined maneuver close to the limits in the radial and cross-track directions.

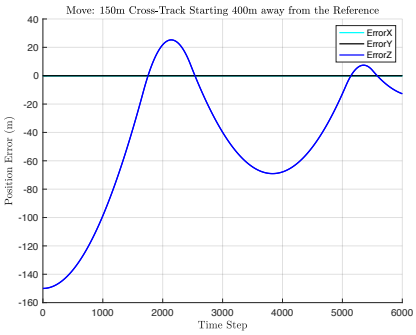
controlled, but there is a quite large overshoot which is not convenient.



(a)



(b)



(c)

Figure 5.7: Low-level control of a combined maneuver close to the limits in the radial and cross-track directions.

In conclusion, the low-level controller designed works as expected, and the limits of the control found in Section 5.3 are also verified when using the implemented low-level control moving the agents. The dynamics and actuators used pose a set of constraints on the system which must be taken into account in order to design the swarm and its controllers.

The Orbit 2 from Chapter 3 will be used as, for the least capable systems presented in the results part of this report, it will still allow for a controlable area of kilometers. All controllers used in this section will be then designed following the procedure presented in Section 5.2. To ensure their working a visualization of a move using the real time plots from SwarmSimulator will be done.

5.5. Note on Implementation

The low-level controller will be implemented through the controller class in SwarmSimulator. This class will be designed to include all the tuned parameters before and generate the given response. This class can be further developed to study the effect of using other controllers. Another perk of the implementation is the inclusion of a control window and an end-of-move window. The control window will avoid the agent to drift once it has reached the desired location whilst not constantly running the low level controller. As noted in Chapter 4, there is a 1m assumed resolution on the precision of the ranging method. This control window could be viewed as the effect of this resolution. Once the controller detects the agent stable in this 1m radius zone around the agent it basically detects that it is one the required point. Therefore it is switched off once it is detected that the agent has drifted (more than the given control window). This will explain why in the final appearance of the control error has oscillations of short period and small amplitude when it is supposed to be static. It means that the agent is oscillating, due to the dynamics, around its nominal position.

The high-level control window will be slightly bigger than the control window of the agent. Since the high-level control only generates commands for agents that it detect that are static. Depending on the position the agents will oscillate in their position and will need small corrections frequently. To allow the high-controller not to be affected by these oscillations and still operate with that agent, this larger control window for the high-level controller is set.

6

High-Level Control

The most key element of this research project is the DESHA swarming algorithm. In this chapter the implementation of DESHA in Swarm Simulator is presented. First in Section 6.1 a theoretical review of DESHA algorithm and its fundamentals, as presented in [31] is presented. Then in Section 6.2 a review of the adaptations done of DESHA in the context of this project is given.

6.1. Theoretical Background of DESHA

DESHA, as mentioned in Chapter II is the name given to the algorithm presented in [31] designed for the swarming. This algorithm goal is to set each agent local state within a group of desired local states. The union of said local states is such that the only possible emerging global state is the desired global state. DESHA has been designed to achieve formations in swarms, therefore most of its development has been focused on that. Nevertheless, it is understood that the final patterns and states could be identified with other variables, giving DESHA quite some versatility. Following a set of design rules it is ensured that reaching the pattern is possible and that the pattern is the unique patter capable of emerging from the union of all the local states.

The algorithm originally is designed taking into account a syste m with 10 constrains and 4 assumptions, listed in Table 6.1.

These assumptions and constrains are the framework where the algorithm was developed. Nevertheless, they do not have to always be true. Some of them might be violated without stopping the algorithm. For example the agents can communicate, the swarming algorithm can expand its possibilities by using this information (for example to double check some information), but even without active communication the algorithm is supposed to work. Furthermore, it is implied in [31] that the constraints might actually be to tight, and the performance of the algorithm could be improved by relaxing them. This should not be a problem, as it is explained in [31] that the algorithm works with these constrains as a reference frame, but they are not necessary. Actually, in the following sections it will be motivated that some of them will be violated by the swarm system designed.

DESHA tries to achieve the acquisition of the desired state keeping in mind its local philosophy and two key properties:

- **Safe** : Understood as without agent collisions or disconnections.
- **Live** : Understood as the fact that for any initial pattern P_0 the final global pattern P_{des} will eventually form provided that both patterns have connected topology.

To guarantee safety, the agents have to avoid disconnections and collisions. The proposed approach is to only allow only moves compliant with three conditions:

- Each agent only moves when no other agents (that it senses) move.
- Each agent only moves in directions where it does not sense any other agents.

Id	Type	Description
C1	Constraint	The robots are homogeneous (all entirely identical)
C2	Constraint	The robots are anonymous (they cannot sense each other's identity).
C3	Constraint	The robots are reactive (they only select an action based on their current state).
C4	Constraint	The robots are memoryless (they do not remember past states).
C5	Constraint	No robot can be a leader or seed.
C6	Constraint	The robots cannot communicate with each other.
C7	Constraint	The robots only have access to their local state.
C8	Constraint	The robots do not know their global position.
C9	Constraint	The robots exist in an unbounded space.
C10	Constraint	Each robot can only sense the relative location of its neighbors up to a short range.
A1	Assumption	The robots all have knowledge of a common direction (i.e., North).
A2	Assumption	The robots operate on a 2D plane.
A3	Assumption	When a robot senses the relative location of a neighbor, it can sense it with enough accuracy and update frequency to establish if a neighbor is moving or standing still (e.g., hovering).
A4	Assumption	P_0,the initial pattern formed by the robots, has a connected sensing topology.

Table 6.1: Constraints and Assumptions of the algorithm developed in [31]

- Each agent moves to locations where it will remain connected to its previous neighbourhood.

The first assumption allows to avoid collisions by the appearance of moving agents in the knowledge region of the agent. Ideally this assumption will also include that the agent is the only one to move in the entire swarm, but this without clock synchronization or global knowledge becomes impossible. The second is a trivial condition to avoid collisions. The third condition ensures that the local neighbourhoods stay connected, and since the initial topology was already globally connected, the swarm will remain connected.

Provided that the designed pattern is achievable while keeping the connectivity, there are two situations are identified that might prevent the P_{des} to form. The first one is the occurrence of deadlocks, *i.e.* situations where all the agents are blocked. The second one is the existence of livelock situations, situations where an agent is trapped in an infinite transition between a set of patterns. The identification of patterns that might include these kind of situations is vital to ensure that, with local knowledge, the global pattern is obtained.

Desired patterns complying with Theorem 1 will be free of livelock situations, whereas those complying with Lema 3 and Proposition 5 will be free of deadlocks (for proof see [31]).

Theorem 1
If the If the following conditions are satisfied:

1. P_{des} *is achievable,*
2. *a pattern in* $P \in P_{AS} \cup P_{des}$ *will always be reached from any other pattern* $P \notin P_{AS} \cup Pdes$*,*
3. G_S^1 *shows that any agent in any state* $s \in S_{active} \cap S_{simplicial}$ *can move to explore all open positions surrounding its neighbors (with the exception of when a loop is formed or when it enters a state* $s \in S_{static}$*),*
4. *in* G_S^3*, any agent in any state* $s \in S_{static}$ *only has outward edges toward states* $s \in S_{active}$ *(with the exception of a state that is fully surrounded along two or more perpendicular directions),*

then P_{des} *will always eventually be reached from any initial pattern* P_0.

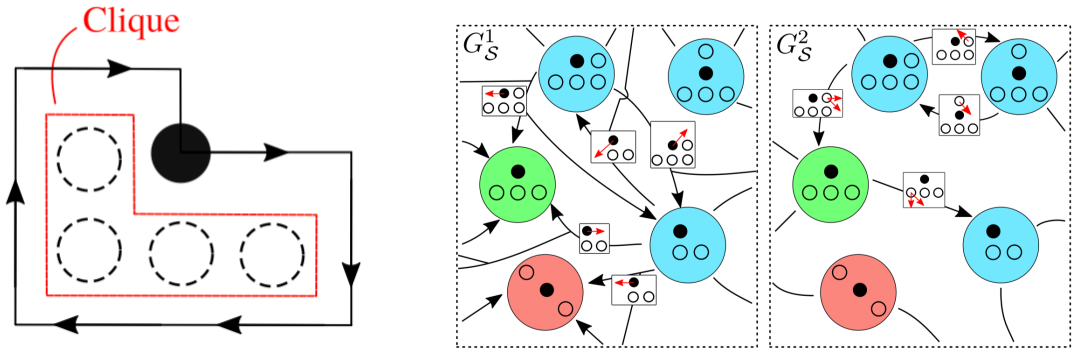
Lemma 3
If the following conditions are satisfied:

1. *for all states* $s \in S_{static} \cap S_{\neg simplicial} - s_{surrounded}$, *none of the cliques of each state can be formed only by agents that are in a state* $s \in S_{des} \cap S_{simplicial}$,
2. G_r^2 *shows that all static states with two neighbors will directly transition to an active state,*

then a pattern in $P \in P_{AS} \cup P_{des}$ *will always be reached from any other pattern* $P \notin P_{AS} \cup P_{des}$.

Proposition 5
If the conditions of Lemma 3 hold and $S_{des} \subseteq S_{simplicial} \cup s_{surrounded}$, *then all agents in a deadlock must be in a state* $s \in S_{des}$.

Where a clique is a defined as a connected set of agents and a simplicial a state not blocked for which its neighbours only form one clique. P_{AS} is all patterns where there is at least one agent active and simplicial, $S_{condition}$ refers to all states matching said condition. Finally G_S^1 is the graph where all vertices are the possible states represented by the move of an agent within the safe state space action and G_S^{2r} represents a graph where all vertices are the states reached by an agent due to the move of its neighbours within their local neighbourhood. The edges of said graphs represent the described transitions. For examples see Figure 6.1



(a) (b)

Figure 6.1: In Figure 6.1a an example of a simplicial and its associated clique is presented. In Figure 6.1b examples of G_S^1 and G_S^2. Images courtesy of [31]

By designing desired final patterns according to the above specified rules it is guaranteed that the final desired pattern will be achieved. These rules have to be encoded so that the transition from the initial to the final pattern can be done. To do this, it is proposed to use probabilistic finite state machines(PFSM).

A finite state machine is a computational tool that encodes the transitions between states of a system. That is, it is an element that evaluates the current state of the system and has encoded a transition to a new state. These transitions are generated with a previous algorithm (in this case DESHA). A finite state machine is by definition deterministic, by knowing the state, the transition is known, so given the initial state of the system, the final state of the system can be determined. This might be productive in some cases, but in the case of swarming, and more in the case of agents with local limited knowledge, a higher level of intelligence, understood as the capacity to cope with unexpected situations, is necessary.

That is why it is proposed in [31] to use stochastic policies. These are a regarded as a more general case of finite state machines [97] where the transitions are not determined. In this case, only probabilities are assigned to each of the possible transitions of the system given its current state. Then the state change is selected according to the given probabilities. In the case of DESHA, the PFSM will

encode, for each of the possible states, a probability to select among all the possible moves that match all the rules listed above (the safe state space) one. Their stochastic nature allows them to avoid stuck situations if, for example, something unexpected in the design blocks one transition. In Figure 6.2 a representation of all possible state transitions for a given pattern is presented.

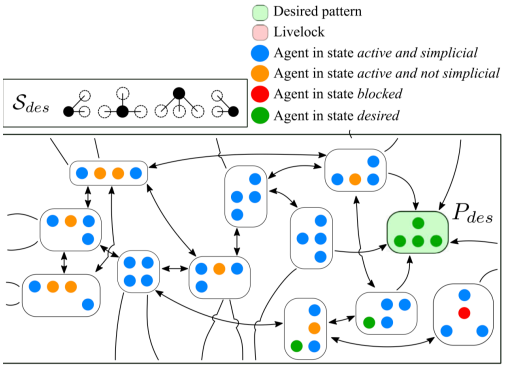


Figure 6.2: Transitions to achieve the desired pattern for four agents.

In this section DESHA, as presented in [31] has been presented. It is worth to note that in said article the authors point out that the algorithm presents some problems with the convergence of large scale swarms. The authors indicate that the relaxation of some of the conditions or the increase in the capabilities of the agents might improve said problems. Also some test are presented allowing to relax some of the lemmas and propositions presented. Interestingly enough, said relaxations actually allow for faster convergences. Nevertheless, given that said relaxations are not proven to always achieve the desired pattern.The aforementioned problems and modifications will be taken into account in the implementation of the algorithm in Swarm Simulator. In the following section, said implementation will be presented together with some modifications on the implemenation of the algorithm in Swarm Simulator.

6.2. Implementation of DESHA

DESHA was proven in [31] as a safe way to achieve formations with an extremely low requirement on knowledge. Furthermore, as any swarming algorithm, it incorporates elements of artificial intelligence as PFSMs [79] which allow the swarm to cope with unexpected situations and interactions with the environment. Nevertheless, in the context of the current project, some of the constraints presented either do not make sense or are not viable. This section will present the adaptation of the DESHA algorithm to the current project.

6.2.1. DESHA in Spacecraft Swarming

DESHA was designed with considering a swarm with a set of constraints and assumptions. Some of the constraints taken for the design will withstand when adapting the algorithm to spacecraft swarming, but others will actually be relaxed. Similarly, the assumptions taken for the design will have to be reviewed, as some differences might be found.

As noted in Chapter 4, communication both between agents and in between agents is already designed and expected. They are actually key to determine and sense the other spacecraft. Other space systems, such as the Galileo GNSS constellation, also incorporate these kind of systems to determine their relative state [43]. Therefore it is understood that at least for fairly distant systems, the communication, in the sense of a radio frequency system to allow sensing between the agents, will be permitted. Nevertheless, as indicated in [31], the generation of full communication protocols requires a large amount of time, therefore, the communication between agents will be kept to the minimum.

In the design of DESHA, the swarm was considered to have anonymous agents , that is, agents cannot recognize any kind of identification between their neighbours. For example, the OLFAR system already has designed communication protocols between the spacecrafts and the ground, and also

housekeeping task will be most probably necessary in each agent. It is expected that, in order for the ground station to keep track of the status of each element of the swarm, some kind of identification in each agent is available. Since this information will be probably an alphanumeric code, so not a really big package of information. Therefore it can be accepted that next to the radio signal to localize the agents, the identifier of the emitting agent can be included without any delay. This will lead to admit that the agents can know the identification code of their neighbours. Since the states, as presented in [31] are only arrays of integers, it could be also theorized that sending that information will not require too much effort either and will allow for the agents to enlarge by one order of magnitude their knowledge area. But following the philosophy of keeping the communication to the minimum this idea will be dismissed for the moment.

With respect to the constraint C8 of Table 6.1, the designed swarm actually does not have this constraint. As explained in Chapter 4, it was decided to implement the idea of the agents having access to the updated information of the reference orbit and the position of one of them, allowing the agents to estimate their relative and positions. Nevertheless, it was explained that this was done more with the idea of having in an efficient manner all the information for the data plotting within SwarmSimulator rather than a need for the operations of the swarm. Therefore, it will be considered that the agents will not have access to their global state. The effects of having the global position will probably render the algorithm quite useless, as each agent could just have access to the global point in space where it should go and go there. Only collision avoidance will be necessary to be implemented, but certainly the interactions between agents will be far less.

The rest of the constraints seem to be reasonable, and quite common in for most swarms, so the swarm considered will present them. Therefore it will be understood that they will hold also for the use of DESHA in spacecraft swarming.

With respect to the assumptions taken in [31] only three of them seem to hold when applying DESHA to a spacecraft swarm. Certainly there is not much problem in having a common reference direction, as star trackers are available to small satellite systems (see Chapter 4). Also, assumptions taken in this project also take into account the enough frequency and accuracy in the sensing and the initial connected topology (see Assumptions AS-HW-009 and AS-HW-003). Nevertheless, space is a 3-D environment. Theoretically all three directions of motion and rotations are viable, although it will be shown later that for this specific case some of the directions of motion are not interested. Therefore it will be necessary to allow for three dimensional capabilities in order fully represent the case of spacecraft swarming. Relaxing this assumption does not seem to be a problem in the proves and designs of the algorithm, it will just add some complexity to the problem. The only change will be that the agents will now need to know two reference directions (North and East for example). This extra complexity will make the states will grow in size as it will be shown in Section 6.2.3

6.2.2. Algorithm Overview

DESHA will be implemented in Swarm Simulator flowing the process depicted in Figure 6.3.

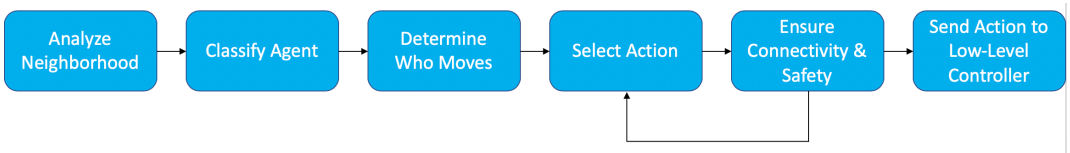


Figure 6.3: DESHA process with all identified steps of the algorithm.

The process of DESHA implies six steps:

1. **Analyze Neighbourhood**. The agent evaluates its neighbourhood and generates its current state with the position of its neighbours.

2. **Classify Agents**. The agent defines if he is active, and therefore ready to request a move, or static, either for being blocked or in a desired state.

3. **Determine Who Moves**. More than one agent might be active in the same neighbourhood. Nevertheless, in accordance with the safety principle of DESHA only one of them can move at the same time. If there is a certain frequency at which the high-level control runs in the system, it will be necessary to generate a negotiation between the agents to select which agent moves.
4. **Select Action**. The move to be performed is selected.
5. **Ensure Connectivity and Safety**: the move selected is evaluated to ensure that no disconnections are generated. If so, a new move is requested.
6. **Send Action to Low-Level Controller**: The selected action is set as the next target to be achieved by the low-level controller from Chapter 5.

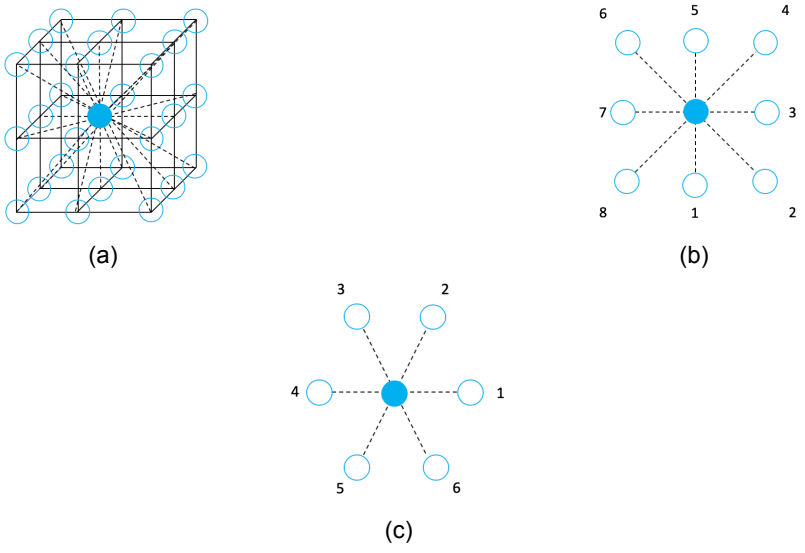
This process will have to be executed at the frequency of the high-level controller each time that the agent is not in move. How these actions are executed is explained in the following subsections.

6.2.3. Analyze Neighbourhood

In order to evaluate its state, each agent will have to sense its surroundings and evaluate in which positions of its state. By definition the states are discrete. This will imply that the algorithm will need to assign the positions to the discrete locations of the state encoded in the agent.

For this report, three possible states have been considered: one in 3-D and two in 2-D. The rationale behind this is to prove the use of DESHA in 3-D for the first possibility of the state. Later it will be shown that actually for spacecraft swarming 2-D patterns hold more interest. Therefore, in order to have a more efficient algorithm and runs in Swarm Simulator, a 2-D version of the 3-D patter will be encoded. Finally a second 2-D pattern was included since it was more convenient for achieving certain shapes. The three selected states are presented in be:

- Cubic State (see Figure 6.4a)
- Square State (see Figure 6.4b)
- Hexagonal State (see Figure 6.4c)



(a) (b) (c)

Figure 6.4: Representation of the three state shapes incorporated in SwarmSimulator. In Figure 6.4a the cubic state is presented. Its identifiers have been omitted for clarity. In Figure 6.4b the square state is presented with its identifiers. In Figure 6.4c the hexagonal state is presented with its identifiers.

An important criterion to select the states was considering only states capable of tessellation. This means the states will be shapes whose repetition can fill, without any free spaces, the entire space. The

choice of tessellating shapes is motivated by the fact of maintaining a coherent grid while discretizing the space. While evaluating the state, the importance of using tessellating is relative, as the agent will just simply assign to the nearest state the located neighbours. As far as they are further away than a safety radius, there will be no safety concerns. Nevertheless, the actions will be chosen to locations within the state. If a non tessellating shape is selected, the positions of the states of the neighbours and the states of the agent moving will not be coincident. Therefore, by the time the moving agent has reached its new location, this might be so distant from the neighbour states that the moving agent might have actually become disconnected from its previous neighbourhood. This can be solved by having a knowledge area much bigger than the state. Another problem related with having a non-tessellating state is the errors accumulated due to inconsistencies in the positioning of state nodes in neighbouring agents. In each move, some offset will be generated due to the difference in positions between the state node locations for the moving agent and its neighbours. Even if this does not cause disconnections, these errors will accumulate, forming much less precise patterns than with tessellating states. Since so many inconveniences arise from the use of non-tessellating states, and no advantages are perceived, only tessellating states will be used.

A graphical example of this is presented in Figure 6.5. As shown in Figure 6.5a, a transition with a tessellating state will, after the transition, present a correspondence in position of both states. On the other hand, not using a tessellating state, as in Figure 6.5b will end up with a lack of match between the states of the neighbouring agents. In the limit case, if the knowledge area of the agent is equal to its state, in Figure 6.5b a disconnection would have occurred.

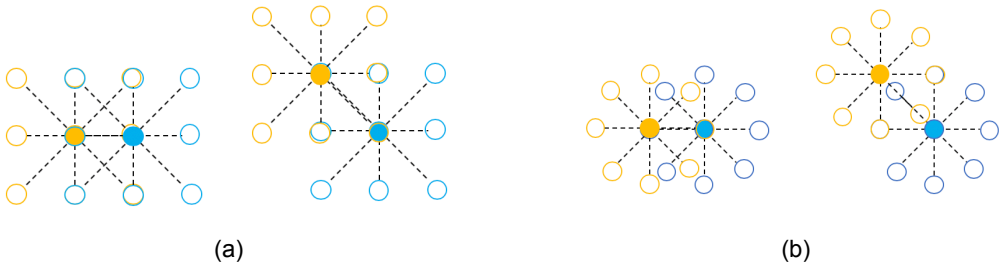


Figure 6.5: In Figure 6.5a a transition with a tessellating pattern (square) is presented. After the transition, the states of the neighbour and the moving agent still match in position. In Figure 6.5b the same transition is presented with a non-tessellating state. After the transition the states of both agents do not match, they will have to approximate the position of each other to the closest state.

All three states selected are tessellating states. The cube is the most basic tessellation pattern that covers all 3-D space. The square is just its equivalent in 2-D. The hexagon was selected for being not only tessellating, but also for giving access to other tessellating figures such as the triangle (hexagon all states filled) or rhombus.

The states will be represented mathematically by an array. 2-D states will be represented by a vector where a logical 1 will be set in the occupied states. The numbers in Figures 6.4b and 6.1b indicate the correspondence between the position and the index in the array. In the case of 3-D states a matrix will be used to represent the state. The azimuths will correspond with the rows of the matrix whereas the elevations will correspond with the elevations. The elevations will start in the bottom (South) direction of the cube and the azimuths will start in the right (East) direction of the cube.

6.2.4. Classify Agents

Once the state has been evaluated the algorithm has to be able to classify the agents. The agents will be classified in two groups: Active and Static. All agents which are not in a desired state and have any states empty will be considered, in this step of the algorithm, as free. To match the definition of active given in [31] still the connectivity of the move will have to be checked, but this will be done in a latter step of the process.

6.2.5. Determine Who Moves

As explained in Section 6.2.2, it might happen that two agents in the same neighbourhood decide to move simultaneously. This is more prone to happen in cases where there is some kind of synchronization in the whole swarm to run DESHA at the same time than in cases where each agent runs DESHA at its own frequency. Only the synchronized case will be studied in this work, as the system in devised in Chapter 4 all agents incorporate a Rubidium clock, stable enough to have a synchronization for the whole swarm to the $10^{-4}s$ level or below [77].

In the event that two agents find their neighbourhood quiet at the same time and both are active,i.e. ready to move, some sort of selection will be needed. Many approaches are possible, some of them might even decrease the number of moves necessary for the swarm to converge to the desired pattern. For example the agents could evaluate which of them is farther from reaching a desired state (numbers of filled positions missing to achieve the desired state). Then the agent with the state less similar to a desired state then would move. Another approach could be using the ALT3 and ALT4 proposed alterations in [31] where the moves towards more concentrated states are encouraged. Nevertheless, these might cause some patters to form and violate Theorem 1. Even if it is pointed out that these violations might not actually be so significant, as the subspace for moves that lead to livelocks and deadlocks seems to be really small, they will not be considered. In space one of the key principles given the lack of access to the system in many cases is safety. This will be violated if convergence could not be assured at all times. Therefore ALT3 and ALT4 implementations will not be considered for the moment.

As example of the selection by the pattern closes to the desired, in Figure 6.6 given the desired pattern for a 4 element triangle (see Figure 6.6a, and the current pattern in Figure 6.6b, the yellow agent will move. All three other agents has half or more of a desired pattern matching, but the bottom agent, which has only one third of a desired pattern matching.

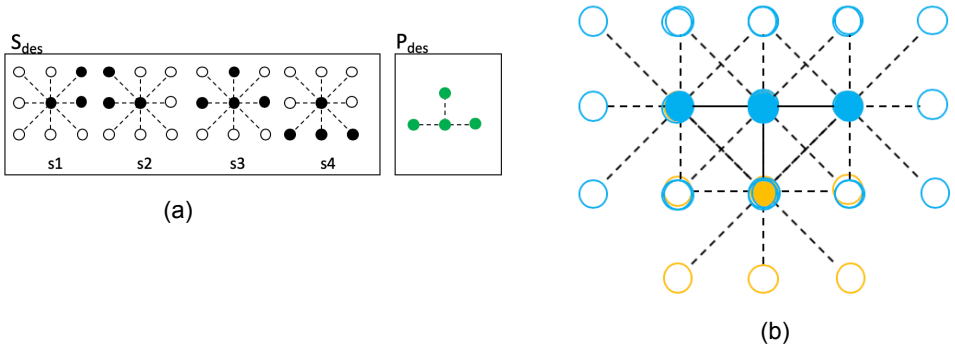


Figure 6.6: In Figure 6.6a the desired pattern for a four element triangle is presented along with the associated states. If a swarm with the patter of Figure 6.6b were to achieve this pattern, and the selection of the moving agent were to be done by moving the agent with the state less similar to a desired state, the agent moved will be the yellow one. All other agents have half or more of a desired state completed.

Exploring some of the presented possibilities might be interesting. Nevertheless,with the main research question in mind, the most practical approach is taken, at least on the first implementation of DESHA. This is to fully respect the original algorithm and its simplicity principle. Therefore all agents will have the same possibility to be the chosen to perform the move. To do so an number will be picked from a uniform distribution among each pair requesting the move simultaneously. The one with the biggest number will remain as active while the other will switch its state to static, as another agent in its neighbourhood is moving. This will be repeated until only one agent in the neighbourhood is active.

From the hardware perspective the implementation of these protocols seems relatively easy. As said before, the agents are allowed to know the identifier of their neighbours. Also, their ranging system is based on radio communications (see Chapter 3). They could add in said communication their identifiers, as said in Section 6.2.1 and, if active, a logical 1 in the message sent. Then active agents receiving messages with these flag from their neighbours could start the random selection process.

As several range measurements are done each time in order to gain precision [77], this addition of information should not significantly add time to the already established ranging process.

6.2.6. Select Action

Once the agent to move has been decided, the next step is to decide where to move. In [31] a PFSM is used to define the transitions of the agents. Usually a PFSM takes the form of a square matrix, where the index of each row indicates one of the possible states that the state shape defined can take and each column is one of the possible actions. These can be calculated with the possible selection of n elements, being n the number of elements of each state. This is calculated with the formula $2^n - 1$ where the state with all empty elements is the one subtracted. For example, in the square state shape, depicted in Figure 6.4b all possible states are $2^8 - 1 = 255$ and the possible actions are $n = 8$. Then the PFSM will take shape of a 255 by 8 matrix.

The probability of taking a given action, given a certain state, in each element of the matrix. This means that the agent in state identified by column 1 for example will have in element $(1, 1)$ the probability of moving to position 1 in the given state. Unsafe actions will have a 0 probability encoded. Also the do not move state can be encoded in an extra column. The rationale of this will be explained in Section 6.2.9.

The approach of using matrices to represent the PFSM is useful for states a with a relatively low amount of positions in each state. This can be acceptable for 2-D states. Nevertheless, when using 3-D states, the risk of an estate explosion is high. For example the presented cube state has 24 possible positions. This implies that the implementation will be have $2^24 - 1 = 16777215$ rows. Therefore it will be a $16777215x24$ matrix with 402653160 elements. From a practical point of view this is not efficient. The matrix will be probably encoded in a fixed memory in the agent. But loading and accessing this information each time that DESHA is run just to find a single element does not seem as the most efficient way to generate the moves. Furthermore, implementing this in SwarmSimulator, where not only one but tenths of agents have to be simulated at the same time will make an excessive use of the computational resources available. Finally, each state shape will need its own PFSM encoded and generated. This will require also a PFSM generator taking into account the rules of DESHA.

In the context of this project a different approach is proposed. In most cases, active agents not only have one but many positions available in their state. Initially the probabilities encoded in DESHA are uniform in all possible moves. Since the agents do not have any information on which moves are more interesting than others, a uniform probability is the logical choice to select the moves. Taking this into account, it will be more efficient then for 3-Dimensional states to simply randomly pick a movement from the empty elements in the state. If said movement passes a connectivity check, the movement is allowed. Else, a new movement is randomly picked.

The disadvantage of the implementation proposed is that, unlike pre-designed matrix PFSMs, the probabilities of the moves cannot be tuned. The choice is made without any previous knowledge each time. In Section 6.2.8 it will be shown that there is an interest in tuning the probabilities of the movements. Given their advantages and disadvantages, both methods of movement generation will be implemented in SwarmSimulator.

6.2.7. Connectivity Check

If the random move selection method presented in Section 6.2.6 is used, it is necessary to ensure that the selected move does not break the connected topology of the local neighbourhood. Several possibilities are available for analyzing the connectivity of the graph, as using pre-established routines to generate a mathematical graph object and through its adjacency matrix evaluate the connectivity. Nevertheless, the one chosen was what seems most obvious and simple, just follow the path. The designed routine just evaluates the state resultant from taking the move. Then, the central object of the graph is discarted and it is attempted to connect all the neighbours without moves larger than the knowledge distance of the agent. If this is possible the movement has maintained the neighbourhood connectivity, else, the movement is discarded. Both options are depicted in Figure 6.7.

Once the selected movement has been checked for connectivity, and has been approved, the in-

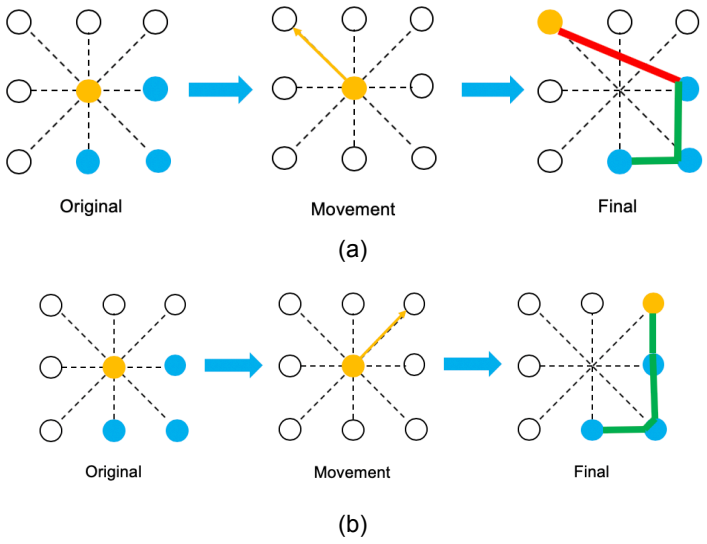


Figure 6.7: Connected and Unconnected Moves. In Figure 6.7a the move selected generates a final neighbourhood where the agent is further away than the knowledge distance of its original neighbours, the movement should be discarded. In Figure 6.7a the same configuration is repeated, but this time the move is within the knowledge area of one of the previous neighbours, so the move maintains local connectivity and its accepted.

formation of the new target will be sent to the low-level controller of Chatper 5, which will perform the maneuver.

6.2.8. Practical Considerations

On top of the presented adaptations for DESHA to the space environment and the context of this project

Space Discretization
DESHA Run Time
Safety Limits
Possible Patterns

6.2.9. DESHA Expansions: Optimization and Large Scale

It is already noted in [31] that there are convergence issues with DESHA when the complexity of the pattern required grows. Although only simple patterns could be researched in this project without lose of generality, one of the key properties of swarms is scalability. Certainly with the increase in the number of agents, complexity will increase and therefore one might say that there will be scalability issues. Therefore, in iterative improvements of the algorithm implementation some strategies were decided to reduce the incidence of this problem.

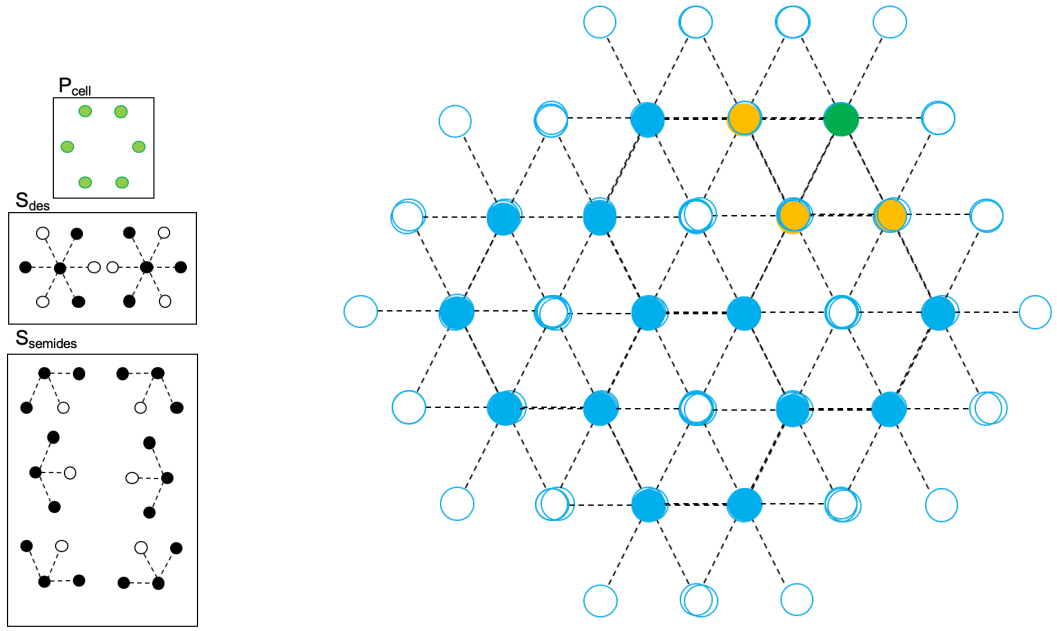
The first strategy to reduce the convergence times comes from [30]. In it a series of measures are taken to reduce optimize the time that DESHA needs to achieve the desired patterns:

- Reduce the size of the PFSM. This is done by seeing as static all agents moves that expect more surrounding elements that agents are on the swarm. Also, each state s is checked to see if it could be fully surrounded by static agents. If not, this pattern is moved to S_{static} as it is assured that at least one active agent is available. Only the case of simplicial states will be accepted as an exception to this.
- Optimize the PFSM. Once the PFSM has been reduced, the next action is to further reduce it by minimizing the norm of the matrix maintaining the conditions of the previous step and the subset S_{static}.
- Optimize the Probabilities. This will mean that the probabilities of the different moves will be optimized. Given the spurious nature of the algorithm, the chosen method is evolutionary algorithms, which will tune the probabilities by running many simulations of the convergence process of DESHA.

By generating PFSMs with this technique, according to [30] the number of agents necessary for the convergence of the swarm is greatly decreased.

Nevertheless, this was not the only policy applied to speed up the convergence of the algorithm. The context of a space swarm and of the reference case selected allows to think that large repetitive patterns will be of interest. Already the fact that the state shapes have been selected as tessellating shapes allows to think that a equally spaced coverage of a certain region can be achieved through these patterns. This could be interesting for example for creating a swarm to image in a fast manner a whole planet or give some kind of signal coverage to a large area. Nevertheless, making the whole swarm achieve a desired state seems as a complicated mission. Nevertheless, locally making a subset of agents converge to a simple shape has proven to be quite simple. The proposed idea is to create new category, $S_{semi-des}$ which already moves to static those agents whose state already covers a desired state. This is better explained through Figure 6.8.

In Figure 6.8a the desired states and the associated semi-desired states of an hexagonal cell pattern are presented. The semi-desired states are just the definition of the part of the state that is necessary for the agents to generate the cells of the tessellating pattern (in this case an hexagon). Any state matching the sequences represented in $S_{semidesired}$ will set the agent as static (this includes S_{des}. For example any agent matching with its east and south-west states full and its south-east state empty will match the top-left semi-desired state. Therefore it will stop moving. In Figure 6.8b this is represented. All blue agents have the states in S_{des}, their states fully match the original states desired for a hexagon tessellation. The yellow agents only match with semi-desired states, as the green agent does not match the desired pattern. With DESHA's original approach, both the green and the yellow agents will move, as none of them is fully in the desired state. Nevertheless, stands to reason that actually, only the green agent has to move, as all the others are already achieving the desired pattern (the local hexagon). With the semi-desired states, even though the yellow states do not comply with the fully desired state, the one making the failure of the pattern will be the one moving.



(a) (b)

Figure 6.8

This will allow for agents which are actually in the cell pattern already not to move, only moving those who need to form their own cell, reducing the actions to convergence. The reader might have realized that in the case of Figure 6.8b the green agent will not stop mooving. Another interesting aspect of tessellating patterns is that there is a required number of agents to ensure convergence. In this case, for every cell on top of the first one, three agents will be needed. This will be further explored

in the results part of the report.

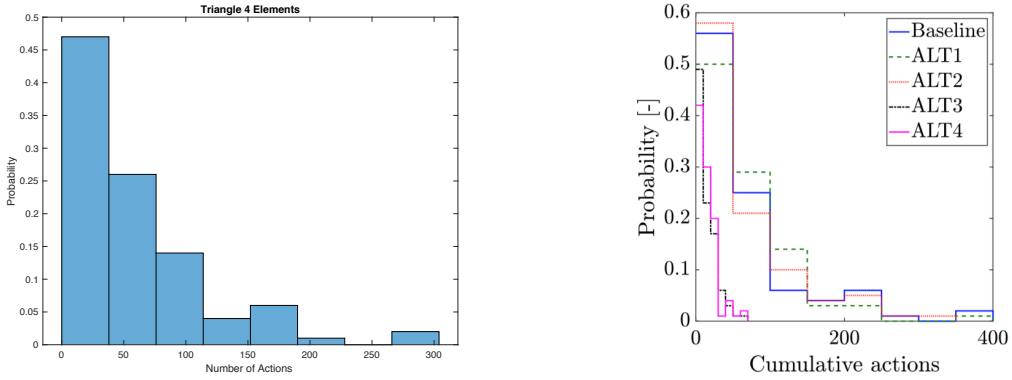
6.3. Verification of DESHA in SwarmSimulator

Once implemented in SwarmSimulator, it DESHA was tested to see if similar results were obtained as those presented in [30, 31]. To do so, a version of SwarmSimulator is made without the low-level controller and the environment simulator. The agents are randomly initialized in a connected topology. Then the agents are instantaneously move to their new locations according to DESHA. The number of actions needed for convergence are computed. As this is a probabilistic algorithm, obviously the actions for convergence vary depending on the simulation. That is why the simulations are run several times. The number of runs will depend on the target pattern to avoid extreme computational loads. The simulations performed are listed in Table 6.2.

Simulation ID	Target Pattern	Number of Runs	Notes
1	Equilateral Triangle 4 Elements	100	Using PFSM model from Secton 6.2.6
2	Equilateral Triangle 4 Elements	100	Using Optimized PFSM from [31]
3	Equilateral Triangle 9 Elements	25	Using PFSM model from Secton 6.2.6

Table 6.2: Simulations for DESHA Verification

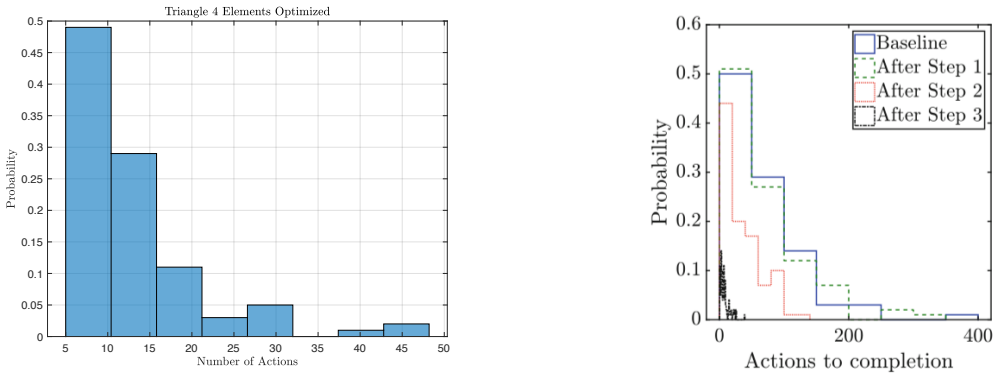
The results obtained are presented in Figures 6.9a, 6.10a and 6.11a . The results present the total number of actions required for the swarm to achieve the desired patterns (as listed in Table 6.2) against the normalized probability that the swarm will converge in said interval of actions. The probabilities have been calculated with the results of the listed simulations. These results are comparable with the ones from [30, 31] which use the same type of plots to present their results. These results are reproduced in Figures 6.9b,6.10b and 6.11b. The baseline for Figure 6.9b and Figure 6.11b will be used, as in these simulations not of the alternative approaches proposed in [31] have been taken due to the fact that they do not ensure convergence. The ”After Step 3” solution has been used for comparison in Figure 6.10b as the fully optimized PFSM has been used.



(a) (b)

Figure 6.9: DESHA Verification Results for 4 Element Equilateral Triangle. In Figure 6.9a the results for the simulations of an Equilateral Triangle of 4 Elements are presented. The results for the same pattern from [31] are presented in Figure 6.9b

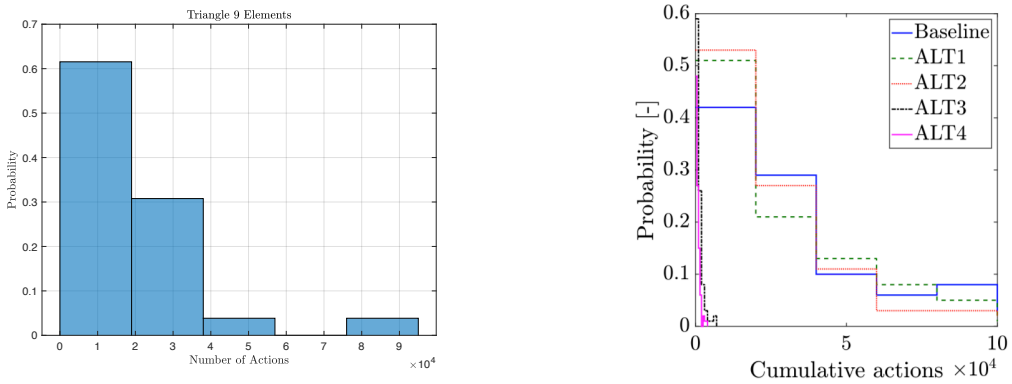
The results comparing Figure 6.9a and 6.9b show a similar tendency. The vast majority of the simulations are accumulated in the area between 0 and 100 total actions. It is true that in Figure 6.10a the scale goes up to 400 actions, but it is shown also there that those seem as residual runs, accumulating the vast majority of the results in the 0 to 200 actions area, just as in Figure 6.9a. This difference is most probably due to a larger bin size in the simulations in [31].



(a) (b)

Figure 6.10: DESHA Verification Results for 4 Element Equilateral Triangle with Optimization. In Figure 6.10a the results for the simulations of an Equilateral Triangle of 4 Elements are presented when the optimized PFSM from [30] is sued. The results for the same pattern from [30] are presented in Figure 6.10b

A similar trend is seen in between Figure 6.9b and 6.10b, where the optimized PFSM has been used to generate the movements. Again, probably due to a larger bin size, the results in [30] show residual runs with larger convergence times (around 100 total actions). Nevertheless, the bulk of the simulations show results coincident with the ones in Figure 6.5a. Also, as expressed in [30], a whole order of magnitude in total number of actions is decreased when applying optimzied PFSMs.



(a) (b)

Figure 6.11: DESHA Verification Results for 9 Element Equilateral Triangle. In Figure 6.11a the results for the simulations of an Equilateral Triangle of 9 Elements are presented. The results for the pattern from [31] are presented in Figure 6.11b.

Finally in Figure 6.11 the results for the Equilateral Triangle with 9 agents are presented. The same trend as in the other two cases is seen, similar distribution, the bulk of the simulations converge for the same number of total actions, but in the results from [31] there is a set of low probability runs with larger number of actions. This is probably due to a larger bin size, which increases the probability for non-usual cases with slower convergence.

Overall, it is verified that implementation of DESHA in SwarmSimulator works as expected, achieving the desired pattern and doing it in a similar number of actions as the ones obtained from the literature. Therefore the implementation of the algorithm is considered correct and the results obtained with it valid. This will be key to give validity to the results of the full simulation.

III

Results

Introduction to Results

This part of the project will present the experiments performed to asses the research questions presented in Chapter 1 and some associated discussions.

The performed experiments will be grouped in three chapters. Each chapter will try to address some aspects of the use of DESHA in space. The main goal of this analysis is to answer the research questions presented in Chapter 1

1. **Reference Case and Multiple Patterns**: The first chapter will address the capacity of achieving multiple patterns. The objective is to analyze the performance of the algorithm in achieving multiple patterns. The patterns will be selected in an increasing grade of complexity. The actuation capacity level selected will be the lowest considered. The inter-satellite distance will also be the minimum considered in this work. This will serve as a baseline to start analyzing on improvements with using systems with more acceleration levels achievable.
2. **Variation on Actuation Capacity**: This chapter will be focused on analyzing the effect of varying the actuation capacity of the system and studying its effect. This variations will also lead to an increase in the some capacities of the swarm, such as the controlable area. The new limits of the control will also be explored.
3. **Optimized DESHA**: In Chapter 6 it was shown that several approaches to make DESHA more effective for large swarms or to reduce the time to achieve formation were possible. This chapter will analyze their implementation on a spacecraft swarm.

In this work three levels of actuation are considered:

- Low Actuation Capacity: $10^{-4} m/s^2$
- Medium Actuation Capacity: $10^{-3} m/s^2$
- High Actuation Capacity: $10^{-2} m/s^2$

Of course these three options are not the only ones available in the market. With the further development of new technologies, both systems with higher and lower levels of thrust are expected to arise. Furthermore, systems which offer higher thrust force are available in the market nowadays. For example thrusters with newton level thrust forces are available for CubeSat platforms, whose order of magnitude of mass is around 10kg [92]. Nevertheless, these systems are prone to present lower levels of I_{sp}. As it has been noted in Chapter 6 and further expanded in Appendix C, many movements are expected. Therefore, it is not logical to expect that these kind of systems are adequate for their use with DESHA. On the other hand, smaller thrust levels are also available, with systems that reach thrust levels of micronewtons [81]. Nevertheless, as it will be shown in Chapter 7, the times to form the most complex patterns with the lowest acceleration level presented already reach extremely large times to achieve the pattern. Therefore, considering even slower systems will be also not realistic.

By performing these test it is expected that sufficient information to generate conclusions on the performance of DESHA in the space environment is attained.

Results of SwarmSimulator

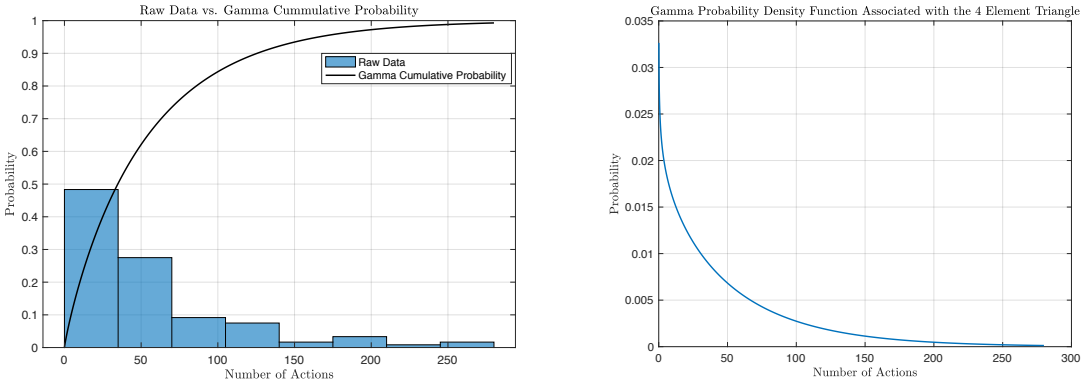
As explained in Chapter 6, DESHA has a probabilistic component. This means that the results will vary between simulations. In order to draw conclusions from the experiments, a set of metrics has to be developed. Four parameters will be measured in the experiments:

- **Time to Converge**: time to form the pattern.

- **Total Number of Actions**: Number of actions for the swarm to form the pattern.
- ΔV: Velocity increase necessary for the agents to form the pattern.
- **Remaining Fuel Per Agent**: Remaining fuel in the agents.

Several options were considered to present the data from the simulations. Plots were discarded, as four plots per simulation campaign, with two of them (the increase of velocity and fuel one) with one line per agent in each simulation seemed really confusing. Another way of representing the results was just to generate tables with all the raw data. But given that in some cases this meant tenths of columns and rows, it seemed not really clear. Since the idea is to explore the capabilities of DESHA, it was decided that three metrics, representative of the simulation campaign: minimum, maximum and average value. The minimum and the maximum give an idea of the both the upper and lower limit of the parameter measured during the simulation campaign. The average value intends to give the usual value of the parameter. Nevertheless, a look into the expected number of movements either from [31] or Appendix C shows that DESHA does not follow a normal distribution on simulation campaigns. A quick look to the expected number of actions shows that for any pattern the actions seem to increase up to a peak and then become more and more rare. Therefore, as the average gives the same weight to all the values, it will represent a larger value than the most probable values. Since this is a conservative approximation, the average will be still calculated. This way future users will have a conservative approach on the usual value expected for each parameter.

To give a sense of the distribution of the data, also the parameters of a probability distribution will be given. This probability distribution needs to match the distribution of the number of actions in DESHA, as all parameters are supposed to flow down from the movements. Note that this is not a deep probabilistic study about the statistical properties of DESHA, but rather just a quick fitting to give future tentative users an idea of how the measured variables are distributed. After analyzing several probability distributions, it was deemed that the best fit is the gamma distribution, which can be seen as a union of several normal distributions. Given the multiple factors that might affect the forming of the pattern, each one with their normal probability distribution, it make sense that this distribution was used.



(a) (b)

Figure 6.12: In 6.12a raw data is compared to the cumulative density function of the gamma probability estimated. In 6.12b the estimated probability density function for the 4 element triangle is presented.

This distribution was fitted using MatLab®'s *fitdist* to an initial test set such as the ones presented in Appendix C with 120 simulations of the DESHA movements to form a 4 element triangle. The result of this analysis is shown in Figure 6.12a. There the raw data histogram is plotted against the cumulative gamma probability density function whose parameters have been obtained with *fitdist*. It can be seen how the cumulative probability perfectly fits the first bar. And if the two first bars are added, the probability is nearly the same as with the cumulative probability (0.75). The histogram fits the cumulative probability as each of the histogram bars is the added probability of all the events that are in the range of the bar. The gamma probability function (probability of each event) is plotted, for this case, in Figure

6.12b.

For the time, actions, and mean ΔV of each simulation, the mean and standard deviation for the gamma probability density function associated with the data will be given as results. These will be obtained with MatLab®'s . This way the statistics offered are better fitted to the associated probability distribution. No probabilities will be plotted, as this will mean at least three plots per simulation campaign as this is considered not to add any more valuable information for the performance of DESHA. For the fuel remaining in each agent, nor these parameters nor the maximum will be given. This is due to the fact that in many cases some agents do not even move at all or perform few maneuvers. These leaves several agents with almost all the tank of propellant full, skewing the statistics offered. For the ΔV the maximum will be the maximum of all simulations performed in the campaign and the minimum will represent the minimum in the whole campaign of simulations. The distribution will be generated with the averages from each simulation. Additionally, the mean of the standard deviations from all agents of each simulation will be also given as the increase in velocity necessary for all agents tends to be the same (so it is, per simulation, closer to a normal distribution).

7

Low Actuation Capacity Results

Once SwarmSimulator has been built, a series of test are performed to see its ability to cope with the task of swarming a set of spacecraft. This chapter will present the analyis performed to evaluate the capabilities of DESHA on spacecraft pattern formation. Specifically this chapter will analyze a system with an acceleration level of $10^{-4}m/s^2$. This is the minimum level of actuation considered in this project and it will set the reference case for further advancements.

On top of being dedicated to set an initial point for the analysis, this chapter will be dedicated to test the performance under different patterns. To do so, four simulation campaigns will be designed to test patterns in an increasing order of complexity. Complexity will be understood as the necessary number of movements expected for the pattern to form. With these tests it is expected to establish the minimum capacities of the algorithm to form patterns in space. The results of this chapter will be compared with the ones obtained applying changes in the actuation capacity to evaluate the effect of having a faster system in Chapter 8.

This chapter will be organized in the following manner. First the test case will be designed, based on the capabilities of the spacecraft and low level control system. All decisions with respect to the selected test case will be motivated. This will be presented in Section 7.1. Once this has been achieved, a series of patterns will be tested, from simple to complex, to evaluate if DESHA is capable of swarming them without surpassing a maximum time. Finally some conclusions of these first results will be drawn in Section 7.6.

7.1. Test Case Set Up

The first simulation will take place with an acceleration level is $10^{-4}m/s^2$. This level of acceleration will be selected as the minimum considered in this work as it will be expected to generate already large times to form the pattern. Furthermore, in the case that in future analysis on the reference mission with the Orbit 1 were to be taken (See Chapter 3) the controlable area will be of the order of meters. The inter-satellite distance to be used will be in the order of tenths of meters whereas the control window of agent in its position is of meters. This is done again to maintain both the simulation time and the runtime of the simulation within acceptable tolerances. These distances are already low for pattern formation in space, although achievable[14]. Furthermore, the control window is in the minimum distance resolution considered (see Chapter 3). Therefore this will generate a test case that will analyze the limit capacities of the algorithm in space pattern formation.

In order to also have a point of link with reality the level of acceleration will be linked to an engine and a tentative platform. In this case Busek's engine BIT-3 [90] and a 6U Cubesat platform will be used. This will generate an acceleration level of $2 \cdot 10^{-4}m/s^2$ (assuming a mass of 6kg to the system, which adheres pretty well with the CubeSat standard [26]). This will actually be the configuration of the Lunar IceCube mission [28]. With this configuration also the ΔV and the propellant used will be calculated to give an order of magnitude in these two variables.

Figure 7.1: BIT-3 engine module [90] used in the Lunar IceCube mission [28], representative of the acceleration level considered.

In order to have comparable results between all patterns, a test scenario will be created. To create this test scenario two criteria will be taken into account:

- **Limitations of the System**. As noted in Chapter 5, the given engine limits and the orbital drifting limit the positions attainable by agents of the swarm. Furthermore, lower limits on relative distances attainable have also to be set to avoid collision events.
- **Computational Power Available**. The work will be performed with a laptop computer with an Intel®i5 processor and 8GB of available RAM. This means that for extremely intensive calculations it is quite possible that the system is incapable of performing the task in a reasonable amount of time. Therefore the simulations will be set in such way that each one runs in less than three hours in the computer.

With this two criteria in mind the parameters in Table 7.1 will be selected for this case.

Variable	Inter-Satellite Distance (m)	Simulation Time Step (s)	Controller Time Step (s)	DESHA Time Step (s)	Safety Distance (m)	State Tolerance (m)	Station Keeping Tolerance (m)
Value	50	50	50	1000	10	1	2

Table 7.1: Test Case Parameters

The first parameter to be chosen is the type of states used (3-D or 2-D). Although SwarmSimulator is designed and operative in 3-D cases, the possible moves are extremely large, this will increase even more the times for convergence of the different patterns, unless delimited,. Furthermore, as mentioned in [4], formation flying in the radial direction is not recommended as it will require much more control effort without any benefits. This last effect was also noticed while evaluating the control limits in Chapter 5, as the controlable area is smaller in the radial direction and the control presents higher overshoots. Therefore the Cross Track - Along Track plane will be selected to perform the simulations and the used states will be bi-dimensional.

The next selected criterion is the size of the discretized grid in which the algorithm will divide the space. Following the procedure presented in Chapter 5, the agents are expected to have a limit of hundreds of meters to move with this acceleration level . This means that the agents must lie in said window if they want to stay connected at all times. Another possibility could be to use passive relative orbit design and use the dynamics in space to keep the swarm together, but in this case all transitions should occur between said orbits. As this will further constrain the space, which might lead to non-convergence of the algorithm [31], this will not be the approach taken. As the whole swarm must lie and move in the controlable area, the size of the movements must be of at leas an order of magnitude less than the controlable area.

A third parameter more related with the computational limits is the time step used in the simulations. The time step is desired to be as large as possible to reduce the computational load in the equipment running SwarmSimulator. It must also be small enough to allow for representing properly the dynamics and control of the system. In the case at hand a time step of 50s is deemed a good trade-off between computational load and representation of the dynamics.

The low-level controller will run at the same rate as the propagation of the dynamics. This is selected as in reality, with current on-board computers, even faster frequencies would be possible. SO it is likely that for the continuous thrust approach taken, the controller will even run faster than the time step .

Running DESHA at every time step was proven counterproductive. As the algorithm runs each time that the agents are within tolerance of their desired positions, this caused that many times the algorithm was requesting moves to agents not settled. This caused instabilities and many collision risk events. In order to solve it, DESHA will make use of the synchronization of the swarm and with a certain frequency (see Section 4.4.1). The frequency is matched to the order of magnitude of the movements of the agents so that the high-level controller only runs when the agents are settled in their positions. Furthermore, this will also serve to support the collision avoidance strategy implemented. This is represented in Figure 7.2. It is based on sending back the agents to their previous locations in case two agents previously out of sight of each other pass too close to each other (for example because they are aiming for the same target location). In order to avoid disconnections, two DESHA calls cannot happen in more than the time it takes an agent to reach the collision avoidance distance with another agent and go back to the knowledge radius of its previous neighbourhood. Given that each movement is expected to require about 1000s with the current inter-satellite distance and acceleration level, DESHA will run at a frequency of 1000s to allow for this.

Figure 7.2: Collision avoidance strategy. Both active agents target the same position. They move towards it, but when they detect they are too close, one of them goes back to its original position whilst the other one finishes the maneuver.

As noted before, collision events might happen. The collision avoidance strategy is designed to avoid those. The distance at which it is triggered in this case is 10m. This is a small distance, but due to the small movements and the low thrust, it will be sufficient to slow down the agent and reverse its thrusting direction. Also, such a small distance was selected to ensure that diagonal moves (in between other agetns) were not stopped by the collision avoidance system.

In order to stabilize the agents around their position and avoid drifting two zones are defined. The state tolerance, which represents an sphere of radius 1m (the minimum resolution assumed on the ranging of the agents) around the desired position of the agent where the agent is said to be in the right position. The station keeping tolerance is a sphere of a slightly higher radius (in this case 2m was selected) where, if the controller detects that the agent has drifted that much, a small correction is performed to return the agent to its nominal position. The station keeping tolerance represents in the end the control window of the agent whilst the state tolerance is used by the low level controller to detect a movement as finished. In this case the control window is set to be of 2m and the state tolerance of 1m. This is one order of magnitude below the inter-satellite distance, so no collisions can happen in station keeping. In this case the narrow control window is expected to require many thrusting events

to maintain the position of the agent.

Finally the maximum run time is not fixed for all simulations. The maximum run time is set in each case trying to both allow for the computer simulation to run in less than 3h and at the same time to achieve the pattern desired. The expected time required to form the desired pattern can be estimated with Equation 7.1 where $t_{pattern}$ is the expected time to form the pattern; N_{moves} is the expected number of movements (form Appendix C and Chapter 6); and t_{move} is the time per movement of the simulation. In this case the movement time is estimated to be around 1000s.

$$t_{pattern} = N_{moves} \cdot t_{move} \tag{7.1}$$

7.2. Triangle of 4 Agents

One of the most simple patterns presented in [31] is the equilateral triangle formed by 4 agents with square state. This simple pattern was used to get a first feeling of the capabilities of the algorithm before advancing to more complex patterns such as the 9 element triangle or the hexagon.

The simulation campaign was set with the parameters presented in Table 7.2. The maximum simulation time was selected to be a million seconds, since, according to the results of Figure 6.9a the usual convergence of this pattern requires about 100-200 movements or less. Since every movement takes about 1000s, the expected time is about 10^5. Since collision avoidance maneuvers might take place, these have to be taken also into account by adding one order of magnitude to the maximum simulation time.

The choice of the knowledge radius is based on the maximum distance between two elements of the state. In this case this was the diagonal of a 50m square, so about 70.7m. Some extra distance is given to increase safety.

Parameter	Value
Number of Agents	4
Max Simulation Time (s)	$10^6 s$
Knowledge Radius (m)	81
Movement Radius (m)	50
Number of Simulations	10

Table 7.2: Set up for the simulation of an equilateral triangle made of four agents

The simulation was run 10 times to ensure an acceptable sample of results. The results that had converged in the first iteration by chance (so the random initialization algorithm had initialized randomly already in the desired pattern) were removed. This left 9 successful simulations. The results of said runs (average, minimums,etc.) are presented in Table 7.3. Only the minimum and average fuel remaining have been shown as in many cases one or more agents are static, being always the maximum close or equal to the initial propellant and therefore biasing the dispersion and maximum fuel.

The simulations results show some interesting data. The average time to converge was about 72h, being the maximum about 8.5 days, which is acceptable given the available actuation capabilities. Given that the average actions for convergence lies in the expected result from the data in [31], the results are accepted as valid. Both the fuel and ΔV used show that the actuators still have much actuation capability left. This is good as pattern reconfiguration or changes of pattern will be possible as well as other maneuvers.

Finally, in Figure 7.3 the orbits of the agents in the along track-cross track plane have been plotted with respect to time to see the evolution of the agent's movements.

Parameter	Min	Ave	Max	Std. Dev.	Mean Gamma	Std. Dev. Gamma
Time (s) to Converge	14300	259965	728400	n.a.	103990	99941
Total Number of Actions	2	62.2	165	n.a.	62.2	3.604
ΔV (m/s)	2.402	4.138	6.2397	1.718	4.1376	3.7765
Remaining Fuel per Agent (kg)	1.4914	1.4981	n.a.	n.a.	n.a	n.a.

Table 7.3: Results for forming a 4 agent equilateral triangle.

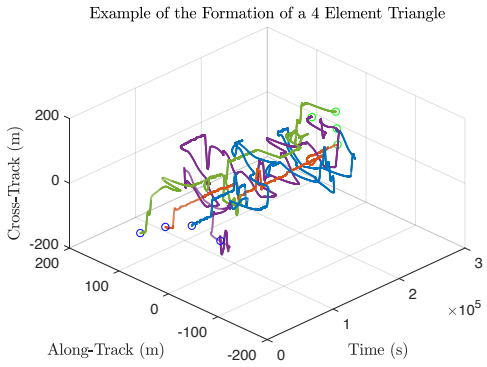


Figure 7.3: Orbits of the 4 agent triangle convergence. The initial and final positions have been plotted as blue and green dots respectively.

7.3. Line 6 Agent

The next simple pattern tested was a 6 agent line (in the cross track plane). In order to set the simulation campaign parameters, an analysis on the number of movements required for the convergence of the swarm is done as the ones performed in Section 6.3. The results for this analysis, as well as for all other patterns not presented in Section 6.3 but used from now on, are presented in Appendix C. The analysis shows that for a line composed of 6 elements the usual convergence lies within less than 1000 actions. Therefore a simulation time of about 10^6 should be sufficient for achieve convergence. With this in mind the parameters in Table 7.4

Parameter	Value
Number of Agents	6
Max Simulation Time (s)	$10^6 s$
Knowledge Radius (m)	81
Movement Radius (m)	50
Number of Simulations	15

Table 7.4: Set up for the simulation of a line made of six elements.

Simulations were run obtaining the results presented in Table 7.5.

The results presented in Table 7.5 show that the times to achieve the formation are higher than desired. What is expected to be a simple pattern needs about 192h to be formed (máximum time registered in this run). The number of actions correlate with those obtained in Appendix C. So these simulations are representative of what can be expected with larger simulation campaigns. On the bright

Parameter	Min	Ave	Max	Std. Dev.	Mean Gamma	Std. Dev. Gamma
Time (s) to Converge	135650	357910	702400	n.a.	357910	160610
Total Number of Actions	43	98.11	172	n.a.	98.11	40
ΔV (m/s)	4.04	5.147	6.425	n.a.	5.1465	2.2930
Remaining Fuel per Agent (kg)	1.4943	1.4976	n.a.	n.a.	n.a.	n.a.

Table 7.5: Results for forming a 6 agent line in the cross track direction.

side, the actuators still maintain much of their propellant.

For illustrative purposes the relative orbits in the cross-track along track plane have been plotted with respect to the time. The initial and final positions have been plotted as blue and green dots respectively. In the orbit it can be seen how the agents maintain their position and perform station keeping maneuvers (oscillations in between big moves). Also the overshoot of the controller in some actions can be seen in big step maneuvers (mostly on cross track maneuvers far from the reference orbit) Finally it can be seen that the connectivity is maintained during the whole process of forming the pattern. These results are presented in Figure 7.4.

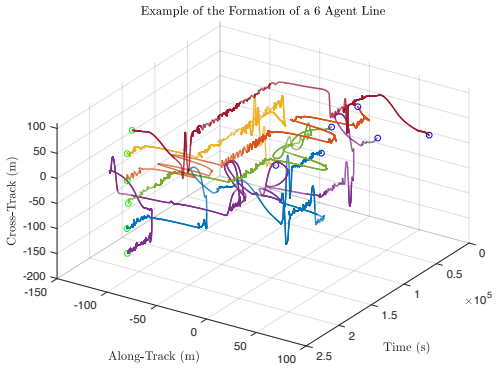


Figure 7.4: Orbits in the cross track-along track plane for the 6 agent line in the cross track direction. The initial and final positions have been plotted as blue and green dots respectively.

7.4. Hexagon 7 Elements

Growing in complexity, the next pattern to be tested was the 7 element hexagon. As in Section 7.3 an estimation of the actions necessary to form the pattern was done by simulating only DESHA movements. The results are presented in Appendix C. It was estimated that between 1000 and 2500 will be necessary to generate the desired pattern This translated in to an expected need of $10^7 s$ to 10^8 to ensure convergence as also collision events must be taken into account. As the hexagon state has the advantage that the distances are all equal between the agent (in the center) and all the other locations of the state. Therefore the knowledge radius will only need to be set slightly above the value of the movement radius to take into account the drifting in the static positioning. Since the hexagon is a complex pattern compared with others such as the triangle of 4 elements or lines of elements, the number of simulations allowed was lowered to reduce the computational load of the simulation campaign. Therefore the simulation campaign parameters were the ones presented in 7.6

The simulations were run, but the pattern did not form in any of them. Furthermore, upon inspection, it was seen that the number of movements performed in several runs did not go over the couple of hundreds movements. On top of that, the simulations run for more than the limit of 3h. It is clear that the system is too slow. It can not reach the required movements in the allocated time. As this

Parameter	Value
Number of Agents	7
Max Simulation Time (s)	$5 * 10^7 s$
Knowledge Radius (m)	55
Movement Radius (m)	50
Number of Simulations	6

Table 7.6: Set up for the simulation of a hexagon made of seven elements.

time is already 115 days, it is considered unreasonable to allocate more time for these simulations. Furthermore this will violate the second criterion showed on Section 7.1.

7.5. Triangle 9 Elements

Although the results obtained in Section 7.4 were quite discouraging, it was attempted to achieve the 9 agents equilateral pattern presented in [31]. The estimations presented in Chapter 6 show that this pattern requires the order of 10^4 actions to be achieved. This means that the simulation should be of about $10^8 s$ to take into account extra moves due to collision avoidance events. With this in mind the parameters of Table 7.7

Parameter	Value
Number of Agents	9
Max Simulation Time (s)	$10^8 s$
Knowledge Radius (m)	81
Movement Radius (m)	50
Number of Simulations	6

Table 7.7: Set up for the simulation of an equilateral triangle composed of 9 agents.

Again the test were run. But as in the case of Section 7.4 none of the test achieved the pattern before running for 6h. Again less actions than required were detected (in this case the order was thousands of actions). Single simulations were run in order to see if in any case the pattern was close to be achieved over the simulated time. The result of one of said simulations (which already surpassed by far the 3h limit run time) is presented in Figure 7.5. In said figure the final positions of the agents are plotted in blue while the targets for the low-level controller have been plotted in black.

It is shown that the swarm maintains the connectivity, and some of the positions are actually close to the desired pattern, but still many moves are expected to achieve the desired pattern. Some metrics of said simulation are presented in Table 7.8.

Parameter	Min	Ave	Max	Std. Deviation
ΔV (m/s)	98.2017	121.607	142.137	14.4
Remaining Fuel per Agent (kg)	1.4335	1.4431	1.454	0.007

Table 7.8: Results of a single simulation for a 9 agents equilateral triangle

As shown in Table 7.8 the results do not go over any limit of the system besides the simulation time. Nevertheless, putting this in perspective, going over the limit of $10^8 s$ means that after more than a year

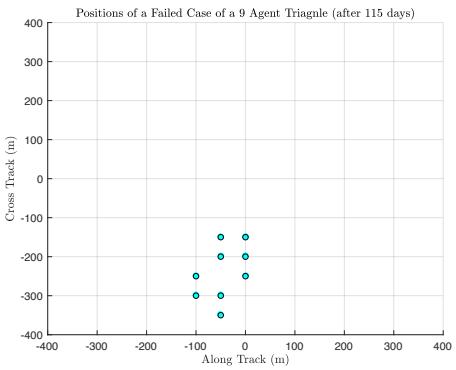


Figure 7.5: Final state of a single simulation for 9 agents equilateral triangle. In blue the agents' positions. In black the target positions for the low-level controller.

of operations the swarm still is looking to achieve the desired pattern. Taking into account that most CubeSat based missions have a lifespan lower than that [34], it is not realistic to accept these results as positive. Therefore it can be concluded that the present design of DESHA and the spacecraft cannot allow for the swarming of a pattern such a as 9 agent equilateral triangle in a reasonable amount of time.

7.6. Conclusion

After running the first simulation campaign and generating the test case scenario, the results show that it is not possible to achieve most of the patterns in a reasonable amount of time. In some cases the simulation limit time was overridden and full years of orbit were simulated. The results showed that even though the swarm still presented a connected topology, and no errors were found in the simulation, the agents still had performed really few movements. It is concluded that this was due to the extremely large times required by the actuators to achieve the next desired position, which had lead to an even larger time between each request made by DESHA. This showed that only patterns with really few movements will be achievable with the BIT-3 engine.

On the other hand simple patterns such as four agent triangle or six agents line were achieved with an acceptable amount of propellant left. But again even for these cases the times necessary for the formation acquisition surpassed the day.

Overall, it seems proven that the use of the minimum actuation level considered in this work only allows for the formation of relatively simple patterns. More complex patterns seem to require larger accelerations. Furthermore, the inter-satellite distance necessary for this acceleration level seems too small to ensure a safe formation flying. This is especially true for the largest systems considered, for example 50kg spacecraft. Again, larger acceleration capacities are expected to allow larger inter-satellite distances. This will be explored in Chapter 8.

8

Variations on the Actuation Capacity

The results for Chapter 7 showed that it is not possible to achieve, within an acceptable amount of time, the desired patterns if those require a large amount of movements. This is mostly due to the fact that the minimum acceleration level for the agents was considered. In this chapter the acceleration level will be varied in order to study the effect of the variation of this parameter.

Two possible effects can be derived of rising the acceleration level. First, the time for each movement is reduced, as the agent can change its position faster. Second, the agents can be controlled in further distances from the reference orbit without drifting away. This allows to have larger inter-satellite distances. The larger the inter-satellite distance, the larger the control window, the less thrusting events needed to perform station keeping and the safer the system.

The chapter will be organized as follows. In Section 8.1 the acceleration level will be risen one order of magnitude with respect to the reference case to test the improvements on time to form the pattern.. Then the set up of the test case will be presented in Section 8.1.1 for the medium acceleration level. After that, the patterns tested in Chapter 7 will be tested again in Section 8.1.2. Finally, a set of first conclusions will be drawn in Section 8.1.3. The same procedure will be followed for the high acceleration level in Section 8.2. In, Section 8.3 the inter-satellite distance is risen with both acceleration levels to achieve all the same pattern. Finally in Section 8.4 conclusions on the variations of acceleration level are drawn.

8.1. Medium Acceleration Level

The medium acceleration level considered in this work is $10^{-3} m/s^2$. This for example the acceleration level of a 50kg spacecraft with a thrusters of 0.05N maximum thrust. As in the reference case created in Chapter 7 a tentative system will be proposed to give some link with reality and estimate the ΔV and use of propellant. In this case a CubeSat platform of 10kg and the BIT-7 engine [56] are considered as analysis case. Even though the BIT-7 engine requires a higher power level than the BIT-3 presented in Chapter 7, it can be assumed that all the extra weight of the system compared with the system presented in Chapter 7 is dedicated to this extra power generation. Therefore, the propellant on board will be supposed to be equal to the one of BIT-3 (1.5kg)

A test case will be set for this system, with a maximum acceleration level of $1.1 \cdot 10^{-3} m/s^2$, in Section 8.1.1. Then the patterns analyzed in Chapter 7 will be analyzed again in 8.1.2 also maintaining the inter-satellite distance. Finally in Section 8.1.3 some small conclusions will be drawn related to the medium acceleration level systems.

8.1.1. Set Up of the Test Case

Once the low-level controller gains have been tuned for the new engine, the movement time is found to be 500s. Then the DESHA running time is reduced in this case to 500s. Also, to avoid that the low-level controller stops maneuvering the spacecraft before it is fully stabilized, the state tolerance is reduced.

Also to allow for having the agents more time ready to move according to DESHA, the station keeping tolerance is slightly risen. Therefore, the simulations with medium acceleration level will have the set up presented in Table 8.1.

Variable	Inter-Satellite Distance (m)	Simulation Time Step (s)	Controller Time Step (s)	DESHA Time Step (s)	Safety Distance (m)	State Tolerance (m)	Station Keeping Tolerance (m)
Value	50	10	10	700	20	5	1

Table 8.1: Test Case Parameters with Medium Acceleration Level.

8.1.2. Pattern Formation

This section will be dedicated to study again the formation of the patterns presented in Chapter 7. The extra acceleration should allow now to achieve the patterns in less time. Therefore it is expected a lower computational cost and time for convergence. Although it is now possible to increase the inter-satellite distance, it will be maintained at 50m during all these simulations to generate comparable results to Chapter 7.

Triangle of 4 Elements

The first pattern attempted will be the 4 element equilateral triangle. This pattern is the most simple one attempted in this report. Table 8.2 reflects the specifics of the simulation. The idea is to mimic the same simulation as the one in Section 7.2. The knowledge radius is slightly increased to maintain the connectivity due to the slightly larger overshoot with this level of acceleration. Also the safety distance is slightly increased.

Parameter	**Value**
Number of Agents	4
Max Simulation Time (s)	$1 * 10^6 s$
Knowledge Radius (m)	82
Movement Radius (m)	50
Number of Simulations	15

Table 8.2: Set up for the simulation of a triangle made of four elements.

The results of the simulation are presented in Table 8.3.

Parameter	**Min**	**Ave**	**Max**	**Std. Dev.**	**Mean Gamma**	**Std. Dev. Gamma**
Time (s) to Converge	25240	51420	96220	n.a.	35567	26194
Total Number of Actions	41	85	157	n.a.	58.8667	44.9762
ΔV (m/s)	5.6940	6.6347	7.8780	1.0554	4.6760	3.4381
Remaining Fuel per Agent (kg)	1.4932	1.4969	n.a.	n.a.	n.a.	n.a.

Table 8.3: Results for forming a 4 agent equilateral triangle.

As shown in Table 8.3, the time performance improves in a full order of magnitude. The maximum time for convergence is reduced from 8 days to slightly over a day. These numbers are actually quite

significant since in this simulation campaign the average number of movements and maximum number of movements required are almost equal. The increment in velocity used is about $2m/s$ bigger in all cases, with a smaller dispersion, most probably due to the bigger capacities of the actuator. Finally, the fuel used is of the same order of magnitude, proving again the interest of using highly efficient electric thrusters.

Overall it seems that the use of a medium acceleration capacity reduces, as expected, the time to form the triangle of 4 elments pattern. Furthermore, the proposed system used will be able to form the pattern maintaining still a lot of capacity to change the pattern or perform further maneuvers.

Line of 6 Elements
The second pattern attempted is the line of 6 elements. As shown in the analysis in Appendix C, this pattern is slightly more complex than the one presented in the previous section. Still this pattern was achievable by the low actuation capacity system of Chapter 7. As in the previous section, the specific set up for this simulation is copied from the equivalent one in Chapter 7 with minor changes.

Parameter	Value
Number of Agents	6
Max Simulation Time (s)	$1 * 10^6 s$
Knowledge Radius (m)	82
Movement Radius (m)	50
Number of Simulations	15

Table 8.4: Set up for the simulation of a line made of six elements.

The results of the simulation campaign presented in Table 8.4 are presented in Table 8.5.

Parameter	Min	Ave	Max	Std. Dev.	Mean Gamma	Std. Dev. Gamma
Time (s) to Converge	21210	117010	430830	n.a.	117010	96348
Total Number of Actions	32	232.8667	848	n.a.	232.8668	199.0184
ΔV (m/s)	10.9838	12.9633	15.5751	1.7518	12.9633	10.8749
Remaining Fuel per Agent (kg)	1.4729	1.4988	n.a.	n.a	n.a.	n.a.

Table 8.5: Results for forming a 6 agent line.

The results presented in Table 8.14 show that the improvement observed in the previous section with the equilateral triangle of four elements is also presented in this pattern. The number of actions obtained in the maximum are quite close to the maximum numbers expected. Still, the maximum time to form the pattern is close to half the one obtained in Section 7.3. The trend presented in the previous section with the necessary increase of velocity is also maintained. The increase in velocity necessary is larger. Finally the fuel remaining is almost the whole original tank. Overall, the expected trends of less time to form the pattern and higher increase of velocity seem to be maintained with the increase of acceleration.

Hexagon of 7 Elements
Given the good results obtained with the previous patterns, the pattern of an hexagon composed of 7 elements is attempted again. It is expected that the use of a higher acceleration level will allow the system to form the pattern within the simulation time. As done in previous sections, the simulation

setup is the same as the one presented in the equivalent section of Chapter 7. Only the knowledge radius is changed for accommodating the performance of the algorithm with the new acceleration level.

Parameter	Value
Number of Agents	7
Max Simulation Time (s)	$4 * 10^6 s$
Knowledge Radius (m)	55
Movement Radius (m)	50
Number of Simulations	10

Table 8.6: Set up for the simulation of a hexagon made of seven elements.

The increase in the actuation capacity lead, as expected, to a full campaign of successful simulations. The simulation results of the simulation campaign in Table 8.6 are presented in Table 8.7.

Parameter	Min	Ave	Max	Std. Dev.	Mean Gamma	Std. Dev. Gamma
Time (s) to Converge	20860	589220	2640890	n.a.	530340	781520
Total Number of Actions	43	1395.7	6259	n.a.	1256.1	1875
ΔV (m/s)	51.5486	61.3052	74.1338	8.0775	55.1784	81.8369
Remaining Fuel per Agent (kg)	1.3442	1.4717	n.a.	n.a.	n.a.	n.a.

Table 8.7: Results for forming a 7 agent hexagon.

The results presented in Table 8.7 are deemed representative enough, as the number of movements obtained correlates with the ones expected (see Appendix C). The simulation times are within tolerances, but for the worst case the simulation time reached a full month for the formation of the pattern. Nevertheless this is an extreme case, being the nominal one closer to the average (as seen in the analysis presented in Appendix C). The usual case is more simular to the average time of this simulation campaign, with around 1500 moves and about a week of configuration time. The increase in velocity required is close to one order of magnitude larger than the previous section. Analyzing all previous simulations with this acceleration level, it seems that each increase in complexity rises both the number of moves and the increase of velocity in around an order of magnitude. The remaining fuel is still high, proving once more the suitability of the system used for this application.

Finally, for illustrative purposes the relative orbits in the cross-track along track plane have been plotted with respect to the time. The initial and final positions have been plotted as blue and green dots respectively. In the orbit it can be seen how the agents maintain their position and perform station keeping maneuvers (oscillations in between larger movements). Opposite to what is shown in Chapter 7, these oscillations are not so noticeable given the larger number of movements and the scale of the figure. It can also be noticed the extreme number of movements to achieve the pattern, many of them not leading to a fast convergence. Finally it can be seen that the connectivity is maintained during the whole process of forming the pattern. These results are presented in Figure 7.4.

Overall, it is concluded that the use of a higher level of acceleration already allows for the formation of more complex patterns in acceptable times. The computational load of the simulation campaign is also acceptable, although slightly high, already close to the set limit.

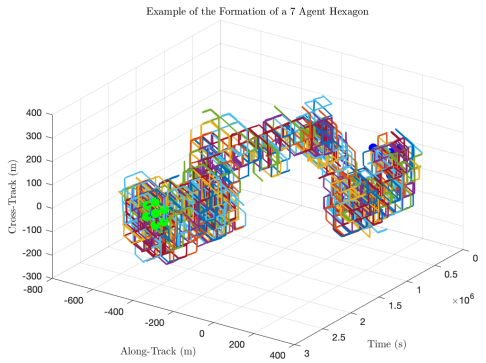


Figure 8.1: Orbits in the cross track-along track plane for the 7 agent hexagon in the cross track direction. The initial and final positions have been plotted as blue and green dots respectively.

Triangle of 9 Elements

The most complex pattern attempted in this report is the equilateral triangle composed of 9 elements. This pattern is presented also in [31] as one of the most complex patterns possible. Mostly this is due to the lack of symmetry of its different composing states and large number of agents. This pattern was not viable for the lowest actuation capacity, even if more than a year of simulation was allowed in Chapter 7. The computing time took more than 3 times what was allowed, still not achieving the required pattern. Nevertheless, the increase in actuation capacity, reducing the movement time, should allow this time to achieve the desired pattern.

The simulation campaign set up is presented in Table 8.8. As in the previous sections the knowledge radius has been accommodated to the new acceleration level (and its controller). Furthermore, previous experiences proved that establishing the run time at levels higher than $10^7 s$ would violate the maximum computing time. Therefore, the maximum allowed run time was set to this limit.

Parameter	Value
Number of Agents	9
Max Simulation Time (s)	$10^7 s$
Knowledge Radius (m)	83
Movement Radius (m)	50
Number of Simulations	3

Table 8.8: Set up for the simulation of a triangle made of nine elements.

The results of the simulation campaign presented in Table 8.8 are presented in 8.9. I has to be noted that originally the simulation campaign was set to be composed of six simulations. Nevertheless, each one of them required about 12 hours of computational time. It was considered worthy to relax the constraint in order to see if the swarm was able to form the pattern.

The results presented in Table 8.9 show that the results are representative, generating the pattern in the expected number of movements (see Appendix C). The required time to form the pattern in the three simulations performed oscillates between 29 days and 44 days.

As in previous cases, the increase in complexity implies an increase in the number of movements and, with the presented system, an increase in the required ΔV. It is remarkable that in this case the remaining fuel still goes over the 90% in the most requiring case. Nevertheless for the first time the maximum fuel use goes over 0.1kg. This shows that achieving this pattern is extremely intensive in actuator use. Overall it is shown that the formation of the triangle with 9 agents is an extremely requir-

Parameter	Min	Ave	Max	Std. Dev.	Mean Gamma	Std. Dev. Gamma
Time (s) to Converge	2506360	3025500	3834860	n.a.	3025500	561860
Total Number of Actions	7979	9570.3	12125	n.a.	9570.3	1765.7
ΔV (m/s)	262.5186	315.0813	395.1955	43.111	315.0821	57.2727
Remaining Fuel per Agent (kg)	1.2734	1.3548	n.a.	n.a.	n.a.	n.a.

Table 8.9: Results for forming a 9 agent triangle.

ing mostly due to the extreme number of movements required for forming the pattern. Even though the proposed system is perfectly capable of forming it, it requires a long time to form the pattern.

Overall, it is concluded that the acceleration level presented is capable of forming even the most complex patterns considered. Nevertheless the times required for the forming pattern and the runtime of the simulation reach the limits considered in this project. It is also worth noting that these limits are reached even when the inter-satellite distance is set to the minimum considered. The larger the amount of spacecraft considered, the better it is to have a large inter-satellite distance to avoid possible collisions. Nevertheless given the current results, doubling for example the inter-satellite distance will probably mean that several months will be required to form the final pattern.

Finally, as it has been done in previous sections, the movements of one of the simulations are plotted in 8.2. A quick look at the figure shows the extremely high number of movements required to achieve the pattern. Furthermore it is shown how the pattern is achieved, but the system has drifted away from the reference orbit. This is also counterproductive as the swarm will require higher impulses to avoid drifting away from the reference orbit. It is true that this drifting is a setback of using approaches that do not consider the global position. There are possible solutions to this, such as just removing the high-level control from one of the agents while the pattern is forming so the pattern will just form around it. Nevertheless this would violate the assumption that all the agents are equal. The fixed agent would act as an anchor of the swarm. But if something were to happen and the agent fails during the formation of the pattern, the system will drift. Therefore this approach would generate a swarm with key elements, and robustness will be diminished. Another option is to allow the agents to know their position with respect to the reference orbit at all times. This way a virtual agent could be created and fixed in the reference orbit without lose of robustness. However, the question of why not just letting the agents know their final position and send them there might arise, mostly for swarms with not an extremely large number of agents.

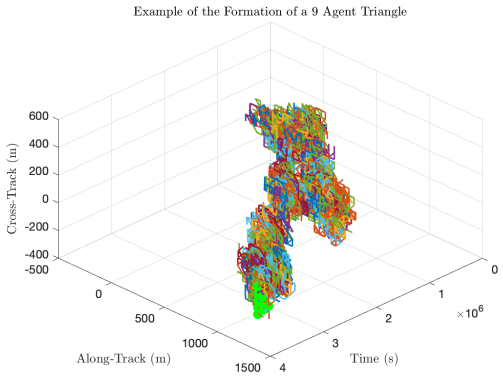


Figure 8.2: Orbits in the cross track-along track plane for the 9 agent triangle in the cross track direction. The initial and final positions have been plotted as blue and green dots respectively.

8.1.3. Conclusions on Medium Acceleration Level

The increase in acceleration level has been proved extremely interesting in the use of DESHA for space applications. The low level acceleration ($10^{-4}m/s^2$ proved itself useful only for the simplest patterns and the shortest inter-satellite distance. The medium acceleration level ($10^{-3}m/s^2$) has allowed to achieve all tested patterns with the minimum inter-satellite distance. Nevertheless, in the most complex cases, the simulations have been close to reach the established limits.

A tentative system has been proposed. The system proposed presents a good performance, being able to cope with all the required increment of velocity and still maintaining more than 90% of the propellant in most cases. Nevertheless if any of the assumptions taken for using this system results to be incorrect or not feasible, the data obtain will not be valid. What is certain is that given the high number of thrusting actions required, an extremely efficient thrusting system will be required to achieve the most complex patterns.

It is also interesting to analyze how in the most complex patters it has been possible to see how the dispersion of the number of movements required to achieve formation is of two orders of magnitude. This is a characteristic of the algorithm, which will be further discussed in the last part of this report. Nevertheless, it is already possible to see how the results are extremely dispersed due to this problematic.

For the most complex cases a higher inter-satellite distance is recommended. Nevertheless, this will mean that a higher time for forming the pattern will be required. As the current acceleration level was already reaching the established limits (or surpassing them) it will be interesting to analyze what would happen with a more capable system. This will be presented in the following section.

8.2. High Acceleration Level

The highest acceleration level considered in this work is $10^{-2}m/s^2$. This for example the acceleration level of a 100kg spacecraft with a thrusters of 0.1N maximum thrust. As in the previous cases, a real engine and a tentative platform are selected to give a link with reality and calculate the increment in velocity and propellant use. In this case a CubeSat platform of 10kg and the BGT-X5 engine [91] are considered as analysis case. The engine selected provides a higher level of thrust than the previous with an acceptable level of I_{sp} (hundreds of seconds). The main disadvantage is the fact that the propellant tank in the default version is of only 0.250kg. It will be assumed that the weight difference between this system and the one presented in Chapter 7 can be used to store more propellant. That will allow the system to perform the most demanding patterns such as the Triangle of 9 elements.

As it was done in the previous section, a test case will be set for this system, with a maximum acceleration level of $5 \cdot 10^{-2}m/s^2$, in Section 8.2.1. Then the patterns analyzed in Chapter 7 will be analyzed again in 8.2.2 also maintaining the inter-satellite distance. Finally in Section 8.2.3 some small conclusions will be drawn related to the high acceleration level systems.

8.2.1. Set Up of the Test Case

As in previous sections, the low-level controller is tuned for the use of the new engine and acceleration level. The time per movement is now decreased to the order of 100s, so DESHA is run at this rate too. Also, initial test seem to prove that it is possible to run the simulations at a faster rate (20s), most probably due to the higher breaking capacity of the system. The station keeping and state tolerances are set equal to the ones in Section 8.1 following the same rationale. These changes generate the set up presented in Table 8.10 which will be used throughout this section.

8.2.2. Pattern Formation

In this section the patterns attempted with the reference case in Chapter 7 are attempted again with the new acceleration level. It is expected that with this acceleration level, even the most complex patterns will be achieved without breaking any of the two criteria to design the reference case presented in Section 7.1. Nevertheless, the selected system presents lower efficiency. Therefore, the propellant use is expected to be more intensive than in previous cases.

Variable	Inter-Satellite Distance (m)	Simulation Time Step (s)	Controller Time Step (s)	DESHA Time Step (s)	Safety Distance (m)	State Tolerance (m)	Station Keeping Tolerance (m)
Value	50	20	20	100	10	5	1

Table 8.10: Test Case Parameters with High Acceleration Level.

Triangle of 4 Elements

As in previous cases, the first analysis done is the triangle made of four elements. This pattern will serve to benchmark the algorithm in the most simple cases with this acceleration level. This simulation can be compared with the ones in Section 7.2 and 8.1.2 to evaluate the improvements and drawbacks of using a higher acceleration level.

Parameter	**Value**
Number of Agents	4
Max Simulation Time (s)	$1 * 10^6 s$
Knowledge Radius (m)	82
Movement Radius (m)	50
Number of Simulations	15

Table 8.11: Set up for the simulation of a triangle made of four elements.

The results of the simulation presented in Table 8.13 are presented in Table 8.12.

Parameter	**Min**	**Ave**	**Max**	**Std. Dev.**	**Mean Gamma**	**Std. Dev. Gamma**
Time (s) to Converge	230	3515.3	15930	n.a.	3515.3	3751.6
Total Number of Actions	3	53.67	157	n.a.	53.6667	56.5296
ΔV (m/s)	11.2610	15.7013	19.4772	3.6650	15.7013	15.7013
Remaining Fuel per Agent (kg)	2.4509	2.8942	n.a.	n.a.	n.a.	n.a.

Table 8.12: Results for forming a 4 agent equilateral triangle.

Analyzing the results in Table 8.12 it can be seen that the number of movements are the ones expected (see Appendix C). Therefore, this simulation campaign can be considered representative. It is possible to see in this table how the maximum time has been reduced from a little bit over a day in Section 8.1.2 or the week in Section 7.2 to about 4h. Furthermore, the maximum number of movements is coincident with the one in Section 8.1.2. This allows for a one to one comparison between the two convergence times.

With respect to the performance of the proposed system, it is notable how the increase of velocity used has been one order of magnitude higher than the one in Section 8.1.2. This is due to the higher speeds achieved by the agent. Also the lower performance of the system shows that the remaining propellant is lower than in other cases. In the worst case only the 81% of the fuel is remaining. This might seem high, but in all other cases the agents were remaining 99% in the worst case.

Overall, as expected the increase in actuation capacity has improved time wise the performance of

the system. The pattern is formed in a few hours in the worst case analyzed. The proposed system is capable of forming the pattern without running out of fuel or requiring an extreme increment in velocity.

Line of 6 Elements
The next pattern attempted, growing in complexity, is the line of 6 elements. This simulation campaign will be comparable with the ones in Sections 7.3 and 8.1.2. As in the previous section, it is expected that a higher acceleration level will allow for a faster formation of the pattern. Nevertheless, the lower efficiency of the selected system will have a bigger effect on the propellant use. The simulation campaign designed is presented in Table 8.13.

Parameter	Value
Number of Agents	6
Max Simulation Time (s)	$1 * 10^6 s$
Knowledge Radius (m)	82
Movement Radius (m)	50
Number of Simulations	15

Table 8.13: Set up for the simulation of a line made of six elements.

The results of the simulation campaign presented in Table 8.13 are shown in Table 8.14.

Parameter	Min	Ave	Max	Std. Dev.	Mean Gamma	Std. Dev. Gamma
Time (s) to Converge	2330	16391	34760	n.a.	16391	12268
Total Number of Actions	38	287.5625	579	n.a	287.5625	220.6362
ΔV (m/s)	45.1435	58.0615	73.8474	10.5414	58.0615	43.9836
Remaining Fuel per Agent (kg)	2.0230	2.6129	n.a.	n.a.	n.a.	n.a.

Table 8.14: Results for forming a 6 agent line.

The results presented in Table 8.14 show that as expected the performance, measured in time to form the pattern, has been reduced. The results show the obtained number of movements is within the expected number (see Appendix C). It is true that the maximum number of movements is close to the maximum number of movements expected. Nevertheless, this will also allow to evaluate the full span of expected performances of the algorithm, as the minimum is also close to the minimum number of movements expected. The time to form the pattern has been reduced a whole order of magnitude compared with Section 8.1.2 from 4 days to less than 10h. Nevertheless, the minimum number of movements between Section 8.1.2 and this section is more comparable. The same reduction is observed in this category, from 32 movements in about 6h of Section 8.1.2 to less than an hour for 38 movements in this section.

The increment in velocity required for the proposed system has been multiplied about 5 times with respect to the one in Section 8.1.2. Nevertheless this comparison is not full as the propulsion systems used are quite different. Also the propellant use is more intensive in this case, again due to the worst efficiency of the engine used for this example. In this case, in the worst case, the remaining propellant is about 66% of the original capacity.

Overall it is concluded that, as expected, the trend that a higher acceleration capacity improves the time to form the pattern for the same number of movements and inter-satellite distance. Nevertheless, the selected system shows also a more intensive use of propellant, which reduces the capacity of the system to generate further patterns or maintain the position in later phases of the mission.

Hexagon of 7 Elements
This pattern was the first one out of reach of the reference case generated, with the lowest acceleration and inter-satellite distance considered. In Section 8.1.2 it was shown how an increase of acceleration level to the order of $10^{-3}m/s^2$ already allowed to achieve this pattern. Nevertheless, the high number of movements (see Appendix C) leaded to a need of a full month in the worst case to generate the pattern. This could be acceptable, as the nominal time of a mission, for example the reference case, is about a year [34]. Nevertheless, this is not the most complex pattern that it was set to be achieved. Therefore, it will be good to study if increasing the acceleration level would lead still to good results without extreme requirements in the propelling systems of the proposed spacecraft. The simulation campaign created for this is presented in Table 8.15.

Parameter	Value
Number of Agents	7
Max Simulation Time (s)	$4 * 10^6 s$
Knowledge Radius (m)	53
Movement Radius (m)	50
Number of Simulations	10

Table 8.15: Set up for the simulation of a hexagon made of seven elements.

The results of the simulation campaign presented in Table 8.15 are presented in Table 8.16.

Parameter	Min	Ave	Max	Std. Dev.	Mean Gamma	Std. Dev. Gamma
Time (s) to Converge	5400	130744	452180	n.a.	93334	12440
Total Number of Actions	27	1103.5	4432	n.a.	1014.6	1272.4
ΔV (m/s)	1.8288	206.77	905.69	28	206.11	247.2325
Remaining Fuel per Agent (kg)	-2.0489	1.7300	n.a.	n.a.	n.a.	n.a.

Table 8.16: Results for forming a 7 agent hexagon.

The results in Table 8.16 are considered valid as the number of movements obtained lie within the expected number of movements (Appendix C). As expected the time performance is greatly reduced from about a month to slightly more than 5 days to form the pattern in the worst case. It is true nevertheless that the number of movements has been a bit larger in the case in Section 8.1.2 than in this one.

The increase in velocity needed also has been risen about an order of magnitude due to the higher acceleration (and lower efficiency). It is good to see that on average the dispersion of the increase of velocity is quite low on average. This means that no agent is using extremely more propellant than other agents. Nevertheless, the system proposed already has trouble to create this formation. In the worst case the spacecraft will need about a 66% more of propellant than the one available. This case only happened in one of the simulations, being all the others within tolerances (even though one other

case is also close to running out to propellant). This nevertheless points out again one of the main problems of DESHA. As it is a probabilistic algorithm, it is never known when the algorithm is going to finally form the pattern.

Finally, for illustrative purposes, the movements of the swarm in one of the simulations are plotted in 8.3. The final positions of the agents have been plotted as greed dots whereas the initial ones are plotted as blue dots. It can be seen in the figure also the station keeping motion and the overshoots of the movements.

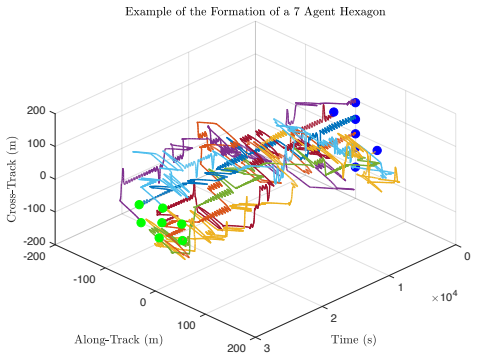


Figure 8.3: Orbits in the cross track-along track plane for the 7 agent hexagon in the cross track direction. The initial and final positions have been plotted as blue and green dots respectively.

Overall, it can be concluded that as expected the trend of rising the acceleration level allows for faster formation times. Nevertheless, this comes at the cost, at least with the proposed system, of in the most extreme cases running out of propellant. This is unacceptable being the formation acquisition only part of the operative side of the mission.

Triangle of 9 Elements

The most complex pattern considered in this work, the triangle of 9 agents, is presented in this section. This pattern was already achieved by using a $10^{-3} m/s^2$ acceleration level. Nevertheless, the required time to achieve the formation was already close to the maximum limit accepted. Furthermore, each simulation needed about 4 times more than the set limit to run. Both these reasons lead to think that studying the performance under a higher acceleration level might be beneficial. Still, in the previous section already some problems due to the less efficient propelling system selected were noticed for similar number of movements to the ones expected in this section (see Appendix C). Therefore, it is not clear if the results of these simulations will be fully positive. The simulation campaign set up is presented in Table 8.17. In this case, given the faster propagation step and the faster dynamics, the number of simulations has been risen. This is expected to allow for more general data and more probability to also have extreme cases included in the analysis.

Parameter	Value
Number of Agents	9
Max Simulation Time (s)	$1 * 10^7 s$
Knowledge Radius (m)	83
Movement Radius (m)	50
Number of Simulations	10

Table 8.17: Set up for the simulation of a triangle made of nine elements.

It is worth noting that, as expected, this simulation campaign run much more smoothly than the one in Section 8.1.2. The whole campaign was finished in less than the 3 hour computational limit set. The results of the campaign are presented in Table 8.18.

Parameter	Min	Ave	Max	Std. Dev.	Mean Gamma	Std. Dev. Gamma
Time (s) to Converge	39640	389604	1002100	n.a.	389600	350840
Total Number of Actions	444	4894.9	12836	n.a.	4894.9	4458
ΔV (m/s)	49.5	596.445	1659.7	66.264	596.4357	503.0787
Remaining Fuel per Agent (kg)	-4.9288	-0.3126	n.a.	n.a.	n.a.	n.a.

Table 8.18: Results for forming a 9 agent triangle.

As in all other cases, the number of movements correlates with the expected number of movements from Appendix C. Therefore it is concluded that the simulations are representative of what could be expected in future simulations. The results show a radical decrease in the required amount of time to achieve the pattern, being reduced from 44 days in Section 8.1.2 to 11 days in this case for the worst simulation. The number of movements required on the worst case are similar, therefore these two times are fully comparable.

As expected the performance of the thruster and system selected on the other hand is quite bad. On average the system runs out of fuel by a 10% of the tank value, being this number a deficit of 164.3% in the worst case. This again show how intensive DESHA is on complex patterns. This is also reflected in the fact that the maximum increase in velocity of the system already reaches the thousands of meters per second. Finally the dispersion is still kept low on average, showing that it is not expected that an agent or group of agents run out of fuel much faster than others.

Unlike in the case of the hexagon, in this case finding simulations where the agents have all run out of fuel is quite usual. Investigating the raw data it was seen that when the total number of moves of the swarm reaches about 4000, the agents tend to run out of fuel. This happened in roughly half of the simulations, showing that much more efficient systems are necessary for forming this pattern with this level of acceleration.

For illustrative purposes in Figure 8.4 the movements of the swarm have been plotted. The initial and final positions have been plotted as blue and green dots respectively. It has to be noted that this case happens to be the worst case scenario of the simulations permed. Still, it can be seen the extreme number of actions required to form the final pattern.

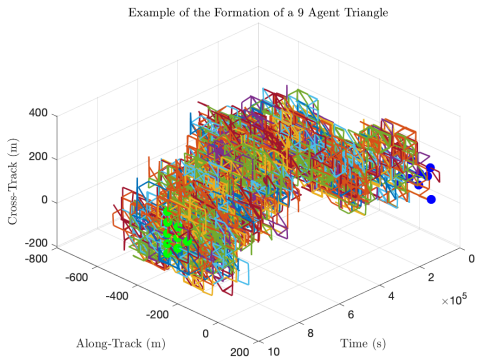


Figure 8.4: Orbits in the cross track-along track plane for the 9 agent triangle in the cross track direction. The initial and final positions have been plotted as blue and green dots respectively.

Overall it is concluded that an increase in the acceleration level of the spacecraft to this level is quite interesting, as formation times are reduced under the month level for this pattern. Nevertheless, it is mandatory to use extremely efficient propulsion systems, given the extremely high number of movements. The proposed system in this work as an example has shown to be not sufficient. The reviews found in the literature [56, 81] show that the kind of systems for achieving this acceleration level do not offer enough I_{sp} for CubeSats and NanoSats. It does not seem possible for larger systems either. Electric thrusters, which seems to be the only ones with enough I_{sp}, only offer up to 0.1N [46]. But the required power for these reaches the level 1kW or more in some cases [86]). The rise of new technologies or improvement on current ones might allow for the considered spacecraft to end up reaching the sufficient efficiency in the propellant use to make this acceleration level viable. But it seems that with the current technology only considering systems with an extremely large portion of their mass for propellant will allow to achieve this acceleration level to form complex patterns.

8.2.3. Conclusions on High Acceleration Level

The maximum acceleration level considered on this work has been $10^{-2}m/s^2$. The current section has evaluated the effects of using this acceleration level with the smallest inter-satellite distance considered in this work, 50m. The results shown that this increase in the acceleration level has reduced a full order of magnitude the time for achieving the pattern with respect to Section 8.1.

A system composed of the BGT-X5 thruster [56] and a 10kg spacecraft platform with 3kg of propellant. The most simple patterns are able to be achieved without using more than 50% of the fuel. Nevertheless, when it comes to reach patterns above 4000 movements the agents start running out of propellant. Therefore complex patterns will be not feasible with this proposed system. A more efficient system (I_{sp} of the order of 1000s) is necessary to be able to achieve this kind of patterns with this acceleration level or much more propellant on board the spacecraft. The current state of the art of thrusting technology does not seem to offer this kind of solutions yet. Nevertheless, given the fast development of small spacecraft technology it is not unlikely that in the near future systems with the required levels of thrust and specific impulse will become available.

8.3. Variations on Inter-Satellite Distance

The increase in acceleration has been tested in Section 8.1 and Section 8.2 to reduce the simulation time. Nevertheless, this increase of acceleration can also be used to generate formations with larger inter-satellite distances and larger control windows. As noted in Chapter 5, the larger the acceleration, the larger the area around the reference orbit where the spacecraft can be controlled without drifting away. Also the less time is needed to make these large movements. This is desirable as said systems are expected to be safer. This section will develop further into this.

To show the possibilities of said formations a single pattern is selected, as there is no interest in testing again if complex formations are achievable. Estimations on that information are already

available with Equation 7.1. That is why a medium complexity pattern (Hexagon of 7 Elements) is chosen to be the only pattern to be formed. This pattern will be tested for an inter-satellite distance in the order of hundreds of meters in the case of the medium acceleration system in Section 8.3.1. Then in Section 8.3.2 it will be tested in the order of kilometers. Some of these systems may allow for larger inter-satellite distances. But in order to avoid reaching extremely high fuel consumption levels the kilometer level test will be considered the maximum.

8.3.1. Medium Acceleration Level

With the medium acceleration level the system will be tested to form a pattern with 300m inter-satellite distance. Theoretically, the system can reach larger inter-satellite distances. Nevertheless, in order to maintain the simulation time below 3h this will be the selected distance. The set up of the simulation will be addapted to the new inter-satellite distance. This set up is presented in Table 8.19. The time step is maintained to the same level, mostly to avoid too much drift in the station keeping and in the proximities of the target location. The DESHA frequency was set to the estimated movement time of the spacecraft, 1000s. The safety distance is increased to 120m to allow the agents to react before a collision happens. Also the control window has been risen a whole order of magnitude, which is expected to reduce the needed thrusting events.

Variable	Inter-Satellite Distance (m)	Simulation Time Step (s)	Controller Time Step (s)	DESHA Time Step (s)	Safety Distance (m)	State Tolerance (m)	Station Keeping Tolerance (m)
Value	300	10	10	1000	120	1	50

Table 8.19: Test Case Parameters with Medium Acceleration Level and 300m Inter-Satellite Distance.

On top of these values, the specific set up of this simulation will be set up as presented in Table 8.20

Parameter	**Value**
Number of Agents	7
Max Simulation Time (s)	$4 * 10^6 s$
Knowledge Radius (m)	310
Movement Radius (m)	300
Number of Simulations	10

Table 8.20: Set up for the simulation of a hexagon made of seven elements.

The simulation campaign presented in Table 8.20 offered the results presented in Table 8.21. It is worth noting that one of the simulations was discarded due to the fact that randomly the swarm was located already in the desired pattern. Another one was discarded due to a lack of convergence after the whole simulation.

The results show that the number of movements is within the expected range (see Appendix C), therefore the simulation is accepted as valid. The results show that the convergence is achieved in about 8.6 days in the worst case. Compared with the average case in Section 8.1.2 which shows an equal number of movements, there is only an increment of 2 days (a third) on the time. This is probably explained due to the fact that the agents with longer distances expend more time in full acceleration. This means that the velocities reached will be higher, reducing the time between movements.

The proposed system has required in the worst case an increment of velocity of about 800m/s, which is high. This is also reflected on the propellant use, where for the first time the system has con-

Parameter	Min	Ave	Max	Std. Dev.	Mean Gamma	Std. Dev. Gamma
Time (s) to Converge	6800	166672	740420	n.a.	112270	228700
Total Number of Actions	13	401.8	1974	n.a.	311.3	430.45
ΔV (m/s)	0.0440	159.7846	804.2235	10.2824	125.36	169.95
Remaining Fuel per Agent (kg)	1.1320	1.4265	n.a.	n.a.	n.a.	n.a.

Table 8.21: Results for forming a 7 agent hexagon for 300m inter-satellite distance.

sumed more than a 20% of the propellant. This, again, is consequence of the longer thrusting arcs.

As it was done in previous sections, the movement of the agents is plotted, for one simulation, in Figure 8.5. The initial and final positions of the agents are plotted as blue and green dots respectively. It can be seen that the number of movements seems smaller than the one in Section 8.1.2. Also the increase in the inter-satellite distance is also fully noticeable. In this image also one of the interesting situations of DESHA is presented. Even though initially only one agent is missing to achieve the desired pattern, all agents end up moving until the final pattern is reached. This is a consequence of the local knowledge of DESHA.

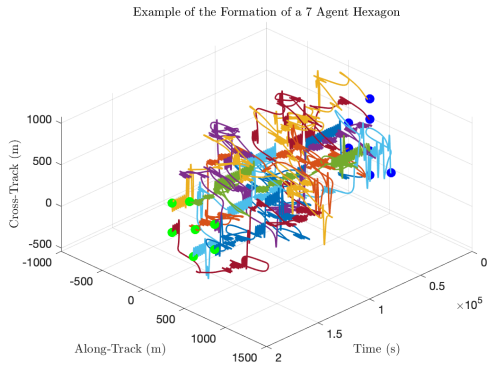


Figure 8.5: Orbits in the cross track-along track plane for the 7 agent hexagon in the cross track direction. The initial and final positions have been plotted as blue and green dots respectively.

As a test, a movement of 1km in cross track and along track directions was studied with this proposed acceleration level. The result is shown in Figure 8.6. It is shown how the movement will take about 3000s. This means about three times what it took to move 300m. Therefore it will be expected that in the worst case scenario the 7 agent hexagon pattern from Table 8.21 will form in the order of months. Also, the overshoot is also quite high (almost half a movement radius), so maybe it is not advisable to use this inter-satellite distance as a failure of the agent while it is moving will leave it too close to neighbouring agents.

Overall it can be concluded that the medium acceleration system can easily reach and form patterns of inter-satellite distances of hundreds of meters without trouble. Larger distances could be in theory achieved, but initial test show that the increase movement time will risk the system to go over extremely large computational times. Furthermore, the overshoot that larger movements produce on the system will also further reduce the frequency at which DESHA is run. Therefore, due to the lack of computational resources, these test were not conducted.

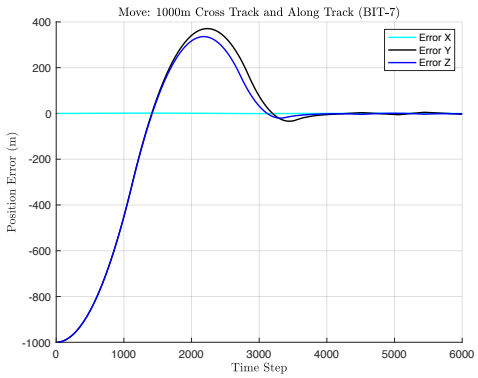


Figure 8.6: 1000m movement with medium acceleration level. The error in all three axis is plotted.

8.3.2. High Acceleration Level

The highest acceleration level proposed in this work is $10^{-2}m/s^2$. This should easily let the agents reach inter-satellite distances of several kilometers. Nevertheless, the proposed system in Section 8.2.2 already has suffered from lack of propelant with the pattern attempted in this section in the most extreme cases. To minimize the risk of this happening again, the distance attempted has been set to 1km. The simulation set up is presented in Table 8.22.

Variable	Inter-Satellite Distance (m)	Simulation Time Step (s)	Controller Time Step (s)	DESHA Time Step (s)	Safety Distance (m)	State Tolerance (m)	Station Keeping Tolerance (m)
Value	1000	20	20	800	300	1	100

Table 8.22: Test Case Parameters with High Acceleration Level and 1000m Inter-Satellite Distance.

The simulation time step has been set, as in previous cases, to 20s to accelerate the process. The time for the agents to reach their new targets is about 800s, so this is set as DESHA run time. The safety distance was risen to a third of the inter satellite distance. The state tolerance is set to 1m to ensure that the low level control works up to the end of the maneuver. The control window, taking advantage of the larger inter-satellite distance, is set to 100m.

The specific set up of this simulation campaign tries to mimic the one in Section 8.2.2 adapting it to this case. The set up is presented in Table 8.23.

Parameter	**Value**
Number of Agents	7
Max Simulation Time (s)	$4 * 10^6 s$
Knowledge Radius (m)	1300
Movement Radius (m)	1150
Number of Simulations	10

Table 8.23: Set up for the simulation of a hexagon made of seven elements.

The simulation campaign presented in Table 8.23 was run. The results obtained are presented in Table 8.24.

Parameter	Min	Ave	Max	Std. Dev.	Mean Gamma	Std. Dev. Gamma
Time (s) to Converge	24020	381978	747500	n.a.	381978	296140
Total Number of Actions	39	477.2	944	n.a.	477.2	357.3032
ΔV (m/s)	236.449	1858.5	2414.1	29.5162	1858.5	924.8424
Remaining Fuel per Agent (kg)	-6.9767	-5.2148	n.a.	n.a.	n.a.	n.a.

Table 8.24: Results for forming a 7 agent hexagon for 300m inter-satellite distance.

As in other cases, the number of movements is compared with the one in Appendix C to see if the simulation is representative enough. In this case it randomly happens that the result are within the accepted limits, but no cases of the maximum expected number of runs have happened. Nevertheless these result are deemed valid as they represent a usual case scenario. The results show that the agents are capable of forming the pattern in similar times to those of Section 8.3.1. The results can be compared with Section 8.2.2, where the same pattern was attempted with a smaller inter-satellite distance. In this case the results show that for more or less the same number of movement (see both minimums) the time to achieve the pattern is multiplied by almost 5.

With respect to the system presented as an example for this acceleration level, the results show that an extremely large increment of velocity is necessary. To achieve the pattern, on average, the agents will need about a 173.8% increment on average of the available propellant. In the worst case this will rise to a 232.55%. This will mean that, as in previous cases, complex patterns are out of reach of the proposed system due to a lack of propellant.

For illustrative purposes the movements of one of the simulations are plotted in Figure 8.7. As in previous occasions the initial and final positions have been plotted in blue and green respectively. On it is possible to appreciate the complexity required, as in previous cases, to achieve the pattern.

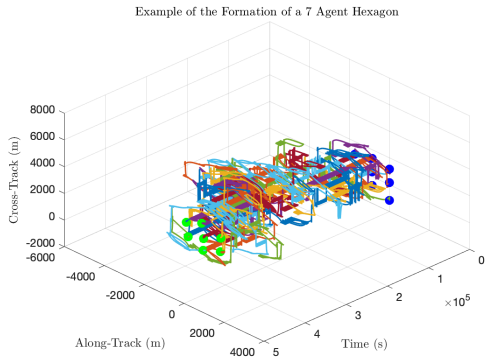


Figure 8.7: Orbits in the cross track-along track plane for the 7 agent hexagon in the cross track direction. The initial and final positions have been plotted as blue and green dots respectively.

Overall it is concluded that the acceleration level of $10^2 m/s^2$ allows for achieving a medium level pattern in a matter of days, even if the inter-satellite distance is of the order of kilometers. Nevertheless, the proposed system is unable to achieve this formation due to an (extreme) lack of propellant. As in the most complex cases of Section 8.2.2, either more efficient engines or systems with larger fuel tanks will be necessary to achieve these patterns at this acceleration level.

8.4. Conclusions on Variations

In this chapter the reference case presented in Chapter 7 has been modified. Two possible new levels of acceleration have been tested in the system: $10^{-3}m/s^2$ and $10^{-2}m/s^2$. These change has two possible translations in the system performance, either an increase in the inter-satellite system or a decrease in the time to create the pattern. Both options where studied by proposing two tentative systems with the given acceleration level. The one representing the $10^{-3}m/s^2$ will use a larger electric thruster as compared with the reference case. To compensate for the extra power needed, 4kg of mass will be added. The one representing the $10^{-2}m/s^2$ will use, in order to achieve the required acceleration level, a green propellant engine with a larger system to store the necessary propellant.

In both cases all patterns from Chapter 7 were tested with the same inter-satellite distance as the reference case to investigate the effects over the time to form the pattern. In the case of the mid-acceleration level, it was found that all patterns were achieved in less than a month and a half. All this without consuming more than half the propellant on board. Compared with the reference case, where the most complex patterns were not achievable even in a full year, this is a significant improvement.

The highest acceleration level the times reduced even further. The most complex patterns were achieved in a matter of days whereas the simplest ones were achieved in hours. That is significantly less than the days and years of the reference case. Nevertheless, the lower efficiency of the propeller used meant that in the most complex cases the agents started running out of propellant. Specifically, any time that an agent went over around 4000 moves (in total), it was shown that most of the agents were out of propellant.

In the case of the increase of the inter-satellite distance, only one pattern (the hexagon of 7 elements) was studied. In this case the differences between the pattern with the minimum inter-satellite distance considered (50m) and the tested ones. In order to avoid extremely large computing times the inter-satellite distances used in the case of the medium acceleration case is limited to 300m. In this case the pattern is achieved in all occasions, although the propellant use is much more intensive than in previous occasions (up to 24.55% of the fuel). The computing time was within boundaries, although it was seen that larger inter-satellite distances will end up in larger overshoot and about 5000s for a movement of 1km. This means that for the same pattern (which requires about 2000 movements to 6000 movements) the final time for formation will require up to two months to achieve the formation.

In the case of the high-acceleration system the maximum inter-satellite distance was set to 1km. This was done due to the identified problems with the propellant shortage on the medium and high complexity patterns. Therefore, in order to try to avoid running out of propellant the inter-satellite distance is not set to the maximum. Nevertheless, the results show that the propellant required is about a 200% more than what was initially available. Therefore, at least with the systems reviewed for this acceleration level, these kind of patterns become impossible unless the most part of the spacecraft is fully dedicated to propellant.

In conclusion, the agents increase of acceleration allows to achieve all patterns, no matter the complexity, presented in this work. The highest acceleration level considered, $10^{-2}m/s^2$, is proven to be extremely interesting to decrease the time to form the pattern. In the most complex case, the pattern was achieved in less than two weeks. Nevertheless, all patterns that require about more than 4000 moves for the case of 50m in inter-satellite links provoke that the proposed system runs out of propellant. This is especially worrying as a larger propellant tank than the original design of the manufacturer was supposed. The conclusion is that with the proposed system only simple patterns are achievable. Also it is not recommended to use with this system extremely large inter-satellite distance. It is true that the same propeller could be used in a larger system, where a larger mass of the system will be dedicated to propellant. This way the formation time will be reduced keeping the system with a relatively long inter-satellite distance.

On the other hand, it can be concluded that the use of the medium acceleration level considered offers a really interesting solution. The time to create the formation is high in the most complex cases, but not extremely high. For example for a one year mission, such as the reference scenario used in this

work [34], it will no go longer than a 16.6% of the whole operative time. Furthermore, the proposed system allows to maintain up to 75% of the fuel to perform further maneuvers or change the pattern formed.

Finally, it is worth noting that after the analysis done on the different systems proposed shows that the identified need of a really efficient thrusting system is confirmed. The longer the distances the longer or the more complex the patterns, the more thrusting events needed. It is worth noting that the maneuvers performed in this work have been focused on using a continuous thrust approach. With electrical thrusting engines this is more common, but with other types of thrusters it is more common to use a two impulse approach [98]. This means that first a thrusting event will set the agent in a trajectory that will intersect with the desired position and when this position is reached a second maneuver will be performed to stabilize the agent in that position. Then station keeping maneuvers can be performed. This approach might be attempted and studied to reduce the propellant use in future versions of the simulator.

9

Optimizing DESHA in Space

As seen in Chapter 7 and 8, one of the main problems of using DESHA to swarm spacecraft is the extremely large number of actions required for the algorithm to finally form the pattern. Furthermore, most of these actions are actually unnecessary or counter-productive to achieve the global pattern. Oftentimes the agents are close to form the pattern but, by choosing the wrong action, provoke a cascade of movements in the whole swarm reshuffling system. For example, in Figure 8.5 only one agent needs to be moved to achieve the pattern. Nevertheless, all other agents end up moving back and forth to the same position until the pattern is reached.

To solve this issue, two approaches will be taken. First, in Section 9.1 optimized PFSM from [30] will be used for the four equilateral triangles to reduce their actions to convergence. For the square and honeycomb patterns, the use of the large scale technique will be implemented in sections 9.2 and 9.3. The results will prove the extent of the improvements of the implementation of these two strategies.

9.1. Triangle 4 Elements

The first pattern attempted was the 4 element equilateral triangle using the optimized PFSM presented in [30]. The simulation campaign was run with the same parameters as in Section 7.2, to have a comparable environment. The actuator set was the BIT-3, the low acceleration actuator. This was done to have a comparison with the reference case presented in Section 7.2. The results of the simulation are presented in Table 9.1. It is worth noting that since the simulation times are expected to be shorter due to the smaller number of actions required (see Chapter 6), the simulation time was reduced to 10^5s. In this case 12 simulations were performed.

Parameter	Min	Ave	Max	Std. Dev.	Mean Gamma	Std. Dev. Gamma
Time (s) to Converge	7120	77950	277680	n.a.	51071	35591
Total Number of Actions	2	10.14	26	n.a.	10.4167	7.7361
ΔV (m/s)	$1.5{\cdot}10^{-3}$	$1.7{\cdot}10^{-3}$	$1.7{\cdot}10^{-3}$	$8.19{\cdot}10^{-5}$	0.0017	$4.5326{\cdot}10^{-5}$
Remaining Fuel per Agent (kg)	1.4987	1.4998	n.a.	n.a.	n.a.	n.a.

Table 9.1: Results for forming a 4 element equilateral triangle with optimized PFSM.

As expected, the use of an optimized PFSM shows a great improvement compared to the ones in Section 7.2. The time to form the pattern has been reduced a whole order of magnitude. Also, as expected, the actions to form the pattern lie now in the order of magnitude of tenths. Finally, the capacities of the actuator have not been overcome. The spacecraft still has plenty of propellant to perform

further maneuvers. Overall, the use of optimized PFSMs seems to be a good strategy to reduce the required number of actions to achieve the desired pattern. In this case, it reduced the convergence time by a whole order of magnitude . Therefore, in the future it is recommended to always optimize the PFSM to increase the effectiveness of the algorithm, no matter the acceleration level.

For illustrative purposes a sample orbit in the along track-cross track has also been plotted versus the time. The original and final states have been marked as blue and green dots respectively. The result is presented in Figure 9.1. It is possible to see how in a few movements all the agents have addopted a desired state.

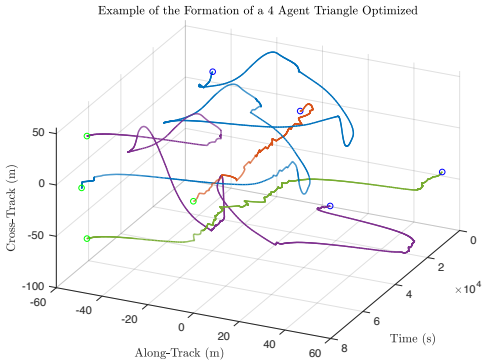


Figure 9.1: Orbits of the agents to form a 4 agent triagle in the along track-cross track plane. The original and final states have been marked as blue and green dots respectively.

9.2. Repeated 12 Agent Squares

The next strategy implemented was the semi-desired states presented in Chapter 6. In this case, a four agent square is the tessellating unit. The first square will act as kind of a seed, allowing any two pairs of agents that encounter next to each other to join the original square and grow the pattern.

To test this pattern the medium acceleration system with the BIT-7 actuator, will be used. This choice is motivated with the fact that this is the actuator and acceleration level with the best overall performance. The simulation set up will be very similar to the one used in Section 8.1.1. The time step was discovered to be stable when it was increased to 20 seconds, so it was decided to apply this time step to reduce the computational time. The simulation set up is presented in Table 9.2. The inter-satellite distance is kept at 50m just to maintain comparability with the original patterns. Of course, as it was proven in Section 8.3, this could be raised, at the cost of higher computational times to form the pattern.

Variable	Inter-Satellite Distance (m)	Simulation Time Step (s)	Controller Time Step (s)	DESHA Time Step (s)	Safety Distance (m)	State Tolerance (m)	Station Keeping Tolerance (m)
Value	50	20	20	500	10	5	1

Table 9.2: Set Up for Tessellating Squares Pattern.

The specific parameters of this simulation campaign are presented in Table 9.3. The selection of the amount of agents was difficult as it requires that no matter how they arrange, a set of squares is possible. For example 9 agents could not be used for this pattern as they could arrange either in a 9 by 9 square and converge or in two 4 by 4 squares, leaving one element alone. The simulation time was set based on estimations with the number of actions required as calculated in Chapter 6 (see results in Appendix C).

Parameter	Value
Number of Agents	12
Max Simulation Time (s)	$10^7 s$
Knowledge Radius (m)	82
Movement Radius (m)	50
Number of Simulations	10

Table 9.3: Set up for the simulation of a tessellating square pattern made of 12 agents.

The simulations were successful. The results of said simulations are presented in Table 9.4.

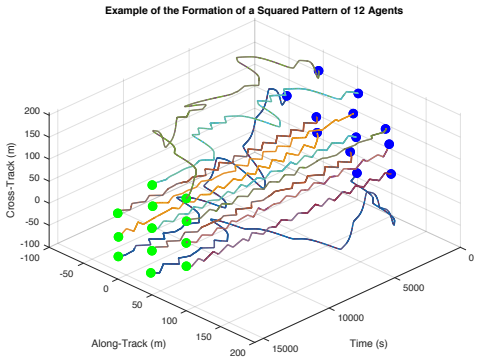


Figure 9.2: Orbits of the agents to form a 12 agent squared pattern in the along track-cross track plane. The original and final states have been marked as blue and green dots respectively.

Parameter	Min	Ave	Max	Std. Dev.	Mean Gamma	Std. Dev. Gamma
Time (s) to Converge	10860	32358	67540	n.a.	32358	16246
Total Number of Actions	21	41.9	65	n.a.	41.9	16.4437
ΔV (m/s)	0	3.3395	17.5115	1.5598	3.3395	1.4854
Remaining Fuel per Agent (kg)	1.4919	1.4985	n.a.	n.a.	n.a.	n.a.

Table 9.4: Results for forming a 12 agent squared pattern.

At a first sight, the results are by far the best ones obtained for a pattern of this size. The number of movements required is so low that in some cases the agents do not even have time to drift and need impulses. Nevertheless a close analysis of the relative orbits (plotted in Figure 9.2 shows that in reality this is not a really successful simulation campaign. As shown in the figure, most of the agents are actually in the desired states at the beginning of the simulation. This is due to the fact that the first thing agents do is range themselves and go to the closest point of the grid in which the high-level controller discretizes the space. As the selected pattern fully fills this space, all agents that are close by will already fill the desired state, being only left to move those that do not have enough neighbours nearby. These agents will act as simplicials around the clique formed by the already formed squares until they find each others and fix their positions. This method achieves the pattern in short time, and maintains the swarming algorithm. This means that in the event of lose of agents or addition these set of rules will allow them to reconfigure. Nevertheless, it seems more of a configuration by ranging than a proper pattern formation.

As it was done in the previous section, a sample orbit in the along track-cross track has also been plotted versus the time. The original and final states have been marked as blue and green dots respectively. The result is presented in Figure 9.2. Due to the few movements, it is also possible to see in this figure how the agents oscillate around their positions while they do not perform a movement. These oscillations correspond to the station keeping algorithm. Every time that the agents surpass their station keeping tolerance (control window), the low-level control is activated and sends back to their state tolerance position the agents.

9.3. Repeated 6 Element Hexagons

In order to test again the extreme scale approach, a final simulation was performed this time with a state shape that does not fill completely the state shape, the honeycomb. An analysis on the expected number of actions (see Appendix C for the results) already showed that this pattern will require a

large number of iterations. Therefore the BIT-7 propulsion system was selected as given its good performance in Chapter III after this number of movements.

Parameter	Value
Number of Agents	10
Max Simulation Time (s)	$10^7 s$
Knowledge Radius (m)	55
Movement Radius (m)	50
Number of Simulations	10

Table 9.5: Set up for the simulation of a tessellating honeycomb pattern made of 10 agents.

The simulation set up was the one presented in Table 9.5. The target pattern was a double hexagon, so the number of agents selected was 10. The simulation time was set to $10^7 s$ according to the estimations done with the number or required movements and the time per movement estimated also in previous simulations. The simulations results are presented in Table 9.6.

Parameter	Min	Ave	Max	Std. Dev.	Mean Gamma	Std. Dev. Gamma
Time (s) to Convergence	38280	1453990	3028720	n.a.	1440300	2196200
Total Number of Actions	80	3848.8	10060	n.a.	3848.8	4029.7
ΔV (m/s)	2.615	198.71	509.04	19.597	198.7132	199.4484
Remaining Fuel per Agent (kg)	1.26598	1.40837	n.a.	n.a.	n.a.	n.a.

Table 9.6: Results for forming a 10 agent honeycomb pattern.

The simulation results show a number of movements corresponding with the expected number. Therefore they are considered representative. The time to converge is about a month in the worst case. Therefore the complexity of this pattern is comparable with the 9 agent triangle. Cases with really fewer movements are also possible.

With the proposed system, the results are also positive, as expected. The required increment in velocity is quite high, with a use of propellant also really high, similar again to the 9 agent triangle. Initially it could be thought that the use of the semi-desired states would not be useful, as for a similarly complex pattern with the same number of agents, the results seem to be similar.

Nevertheless, in Figure 9.3 shows the advantage of the semi-desired states. On it the state of the swarm after 23920s and 94 movements is plotted. The agents have been plotted as blue dots, whereas their targets are plotted as black empty circles. It is possible to appreciate how after only one hour the first out of the two cells is already formed. This means that only after 94 actions in this example 60% of the swarm is already in position. Furthermore, this first cell will work as the first seed, which will allow to the next cell to be formed next to it. All the agents that are not in this first cell will work as simplicials around this first cell, which will be a clique. Therefore, even though part of the swarm is still reaching its final target, another part of it is operative. So for complex patterns and large number of agents, this will allow to start scientific operations with part of the swarm before the total swarm is formed. Furthermore, this part of the swarm will already be stable, not moving again with the random actions of the agents that are left to from the rest of the cells.

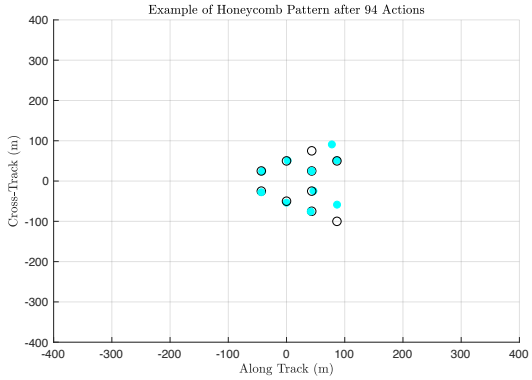


Figure 9.3: Middle step of the honeycomb pattern formation. The agents are plotted in blue while their next target is plotted in black. It is possible to appreciate that the first cell is already formed.

As in other cases, the relative orbits of the spacecraft are plotted in Figure 9.4. In it the complexity of forming the whole pattern is seen. Also the final and initial states of the agents are shown as green and blue dots respectively.

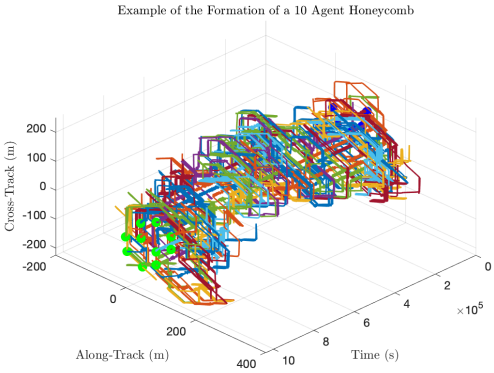


Figure 9.4: Orbits of the agents to form a 10 agent honeycomb pattern in the along track-cross track plane. The original and final states have been marked as blue and green dots respectively.

Finally, in order to show the power of this algorithm, a small test has been done with 28 agents. The simulation was set with the high-acceleration actuator and the shortest inter-satellite distance (see Section 8.2.2) to be able to cope with the computational load of the simulation in reasonable times. The variable of interest is the number of movements, as then this can be scaled with the information of the simulations in Chapters 7 and 8, and the previous sections of this chapter. The result of this simulation is presented in Figure 9.4. These results are obtained after 6954 iterations. This is a similar number to the one required for a triangle with 9 agents. Already 5 out of 8 cells expected are formed and the ones missing are mostly formed.

It has also been noted that in some cases stuck situations have been observed with two positions. The first one is when an agent is captured in the middle of a cell that is fully surrounded by other cells. The other one is when an agent is stuck in between two cells and only one more agent is active. In this case when the active agent reaches a position nearby the deadlock, the stuck agent reaches a semi-desired state position. This avoids the stuck agent from moving when it could, perpetuating the deadlock. Further study will be then necessary on which patterns are possible with the semi-desired. Nevertheless, these formations allow still for creating formations of extremely large number of agents partially in record times given their complexity and number of agents.

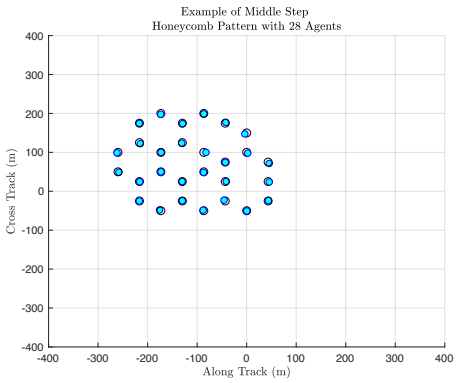


Figure 9.5: Orbits of the agents to form a 10 agent honeycomb pattern in the along track-cross track plane. The original and final states have been marked as blue and green dots respectively.

9.4. Conclusions on the Use of Optimization with DESHA

In this chapter, optimization techniques and large scale strategies were applied to the DESHA algorithm in order to reduce the required number of actions for convergence. The results seem to indicate that the use of PFSMs reduces also greatly the consumption of fuel and time for formation of the pattern, proving itself a good asset. Nevertheless, further testing with other optimized PFSMs will be necessary in order to ensure that these results are applicable to more complex patterns.

As for the use of large scale strategies, it has been proved that states that fully fill the space (i.e. the squared pattern) allow a fast convergence of the algorithm for large patterns. But it comes at the cost of generating the formation through ranging rather than forming actual complex patterns. On the other hand, when the tessellating pattern does not fully fill the state such as the honeycomb, the actions require rise. Nevertheless, it offers the advantage that the majority of the swarm forms in really few actions, being only the last cell the one that requires most of the actions to be formed. Still more research on semi-desired states is needed to find patterns that only can form the desired tessellating patters with these method without deadlocks or livelocks.

In conclusion, the use of both optimized PFSMs and large scale strategies proves that the the required number of actions to convergence can be lowered to acceptable limits. Nevertheless, further studies with different patterns and more optimal code approaches will be necessary to actually extend the final conclusions.

IV

Conclusions and Recommendations

10

Conclusion and Recommendations

In this work the use of the algorithm presented in [31] for swarming spacecraft has been evaluated trying to present a realistic scenario. To do so a simulation tool was designed which modelled both the implementation of algorithm and the necessary environment and subsystems were developed. This was presented in the second part of the report. Then, in the third part, a campaign of experiments was launched to evaluate the suitability of the algorithm for spacecraft applications. Initially a reference case was generated with the minimum acceleration and the minimum inter-satellite distance. This was used as a reference case to then evaluate the effects on increasing the actuation capacity of the system. Both the effects on the time to form the pattern and on the inter-satellite distance were evaluated. Finally, in order to improve the results obtained in previous sections, a set of optimized and special techniques were presented. These techniques allow to improve the algorithm's performance and cope with the most complex situations.

The final goal of this work was to answer the research question posed in Chapter 1 and its associated sub-questions. After the analysis done with the simulator, an answer has been obtained for all of them. The answers to the research sub-questions are presented in Section 10.1. Then, with said answers, a more elaborated answer to the research question is presented in Section 10.2. Finally, the simulation tool, as well as the obtained results, have opened an interesting line of investigation for upcoming researchers. Thus, in Section 10.3 a set of recommendations and future work are presented. These will give ideas and future lines of research that will probably increase the scientific output possible to be achieved with DESHA.

10.1. Research Sub-Questions

Three research sub-questions were presented on Chapter 1 to elaborate more specifically on the different facets of the research question. After the experiments performed, it is possible to answer all of them.

10.1.1. Sub-Question 1: Swarm Properties

The first sub-question was:

Does DESHA comply with the swarm properties of flexibility, scalability and robustness?

This question intended to first evaluate if DESHA, as an algorithm, complied with the swarm properties when it encountered a space scenario.

Flexibility

The property of flexibilty is clearly present. In the third part of the report multiple patterns with different sizes were tested. For the most part the algorithm was able to reach the final pattern with the set constraints. Pattern transitions are also possible depending on the system designed. For example, the two systems proposed for the medium and low acceleration levels in Chapters 7 and 8 remain with

most of their propellant tank by the time they achieve the formation. This means that re-shuffelling into another pattern is possible and, since in any case more than 25% of the propellant tank is used, the system will still maintain some actuation capacity.

It is true that the change of pattern will depend on the system selected. For example, the high acceleration system proposed would run out of fuel if a medium complexity formation was the original formation and then any other formation would be attempted. This puts into manifest again the key role of the design of the system. As it was noted in Chapter 4 already, systems with unefficient thrusting system will not be able to achieve the extraordinary number of movements that the most complex patterns require unless most of the spacecrafts are composed of fuel.

The same conclusion can be drawn about the inter-satellite distance. Multiple inter-satellite distances can be achieved with DESHA, mostly for the medium and high actuation capacity systems. Nevertheless, the larger the inter-satellite distance, the bigger the propellant consumption. This, together with the high number of movements that some patterns require, leads to a need for a system either with a really large propellant tank or a really efficient thruster.

Robustness

The robustness of the algorithm, at least when adapted to space, is not so clear. DESHA is an algorithm based on local knowledge. This means that the agents only know a local neighbourhood, not the whole swarm. This allows for a reduction of the communication needs of the swarm as compared with other algorithms [68]. Nevertheless, it also means that the system is not robust to the failure of some of their elements, at least in some cases.

When the pattern is formed, depending on the shape, it can happen that some of the elements become key. This is illustrated in Figure10.1. In Figure 10.1a the failure of an agent in a 9 Element Triangle will provoke that two of the agents suddenly do not have a desired state. This will translate in a cascade effect, where they will move the other agents of the swarm until eventually they start running out of fuel. This is due to the fact that the pattern specifically requires a number of agents. Another similar case is presented in Figure 10.1b. There, if an agent in the middle of the line were to fail, it would split the swarm in two, as the agents only communicate with their neighbours. In the second case the lack of knowledge of the agents of a further distance rather than their neighbourhood only leads them to disconnect. On the other hand, in the first case the fact that DESHA only stops shuffling the agents when all of the desired states are matched at least once leads to the infinite shuffling when an agent is lost.

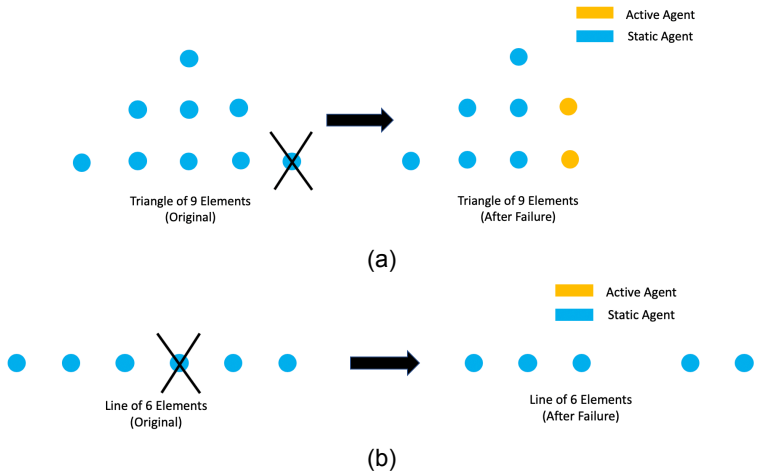


Figure 10.1: Different scenarios before and after the failure of an agent. In 10.1a the case of the failure of an agent in a 9 agent triangle leaves the pattern ready for an infinite shuffle as the desired global pattern is not possible. In 10.1b the result of a failure of an agent in the middle of the chain generates two independent and disconnected swarms.

These failures do not happen with all patterns, as shown in Figure 10.2. Some patterns such as repetitive square and hexagon patterns allow for a more robust solution. In these two cases, the failure of agents will only lead to a couple of more agents (in yellow in Figure 10.1b) to wonder around the cells until running out of fuel or a replacement agent arrives. This is due to the fact that both patterns are composed of a basic unit cell and do not care about the surroundings once the cell is achieved due to the semi-desired state approach (see Chapter 6). Also patterns based on rules rather than fixed states (such as having always two neighbours) present this property. These patterns were not used in this work as initial tests showed that they required really high numbers of movements to form a pattern (a pattern never formed in the test performed).

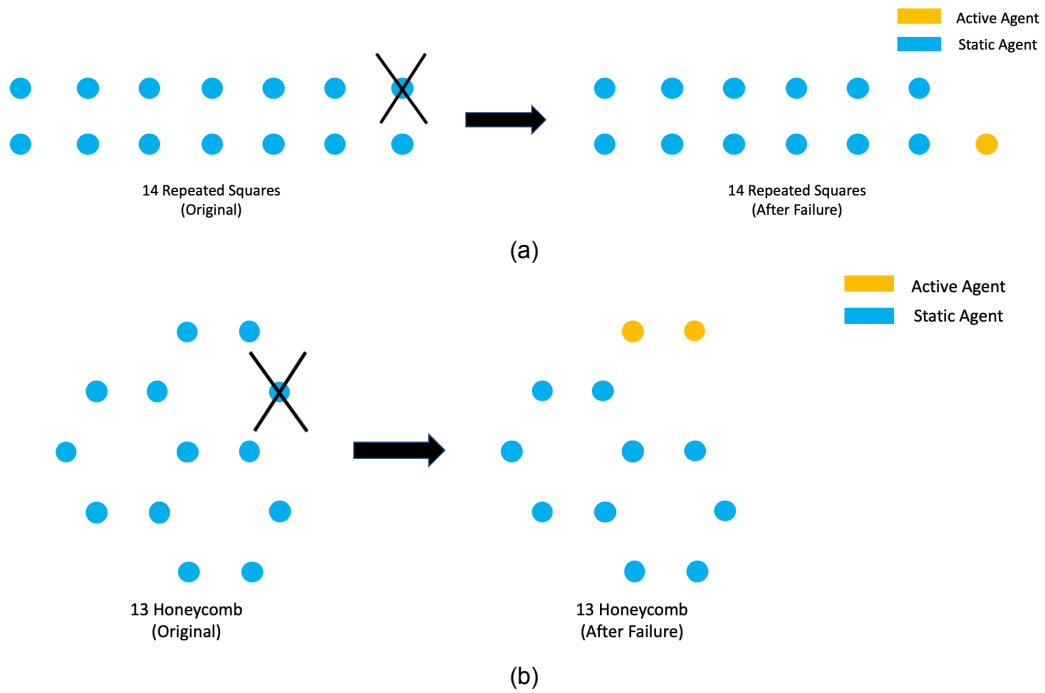


Figure 10.2: Different scenarios before and after the failure of an agent. In 10.2a the failure of an agent only leaves one other agent wandering around the swarm. In 10.2b the failure of an agent only leaves two agents wondering.

Overall DESHA only provides, at least in the applications presented in this report, robustness for a limited number of patterns. Therefore if a robust swarm is required, for example due to long times to send a replacement agent, cell-based patterns seem to be the best choice. Even though some agents will be wondering around, the most part of the swarm will not change its shape. It will be necessary nevertheless to have some kind of policy to send the agent away once it is close to run out of fuel. This will avoid collisions with the remaining agents that run out of fuel. In conclusion, the robustness of DESHA, due to its local approach, is dependent on the selected pattern.

Scalability

In [31] it is mentioned that when the algorithm is scaled to patterns with higher number of agents and complex patterns the number of actions required scales faster. This has been corroborated in Chapters 7 and 8.

Nevertheless, in all cases but in the one with lowest acceleration level all patterns have been formed in acceptable times. It is true that in the case of the 9 agent triangle some strange cases are expected with even higher number of movements according to [31] and the analysis done in Chapter 5. Still, the algorithm allows to form patterns with a wide variety of agents. Furthermore, the optimized techniques or the large-scale techniques allow for the mitigation of this problem, allowing to form swarms of extremely large amounts of agents maintaining certain degree of complexity and the swarm intelligence of DESHA.

Therefore it is concluded that DESHA is scalable in the proposed application to space, as it offers a good performance no matter the number of agents. It is worth noting that the scalability has been analyzed only as the ability of the algorithm to cope with both low and high number of agents. It is true that some patterns require a fixed number of agents such as the 9 agent triangle. This will mean that adding agents will only result in an infinite shuffling of the swarm. Technically to scale the pattern up or down it will be required to upload a new set of desired states. Nevertheless, as proven with the 4 agent triangle, this is possible. Therefore concluding that DESHA allows for scalability is considered correct.

10.1.2. Sub-Question 2: Technical Aspects

The second sub-question aimed to address the implementation of DESHA to a space scenario taking into account the state of the art of the considered spacecraft's technology. The question was formulated as:

> *Does DESHA perform within reasonable limits in use of propellant and time with respect to the reference mission and the current state of the art of small spacecraft technology?*

In Chapter 4 the needs and assumptions taken on the hardware were presented. All of them seem reasonable and attainable, at least by the literature study done previous to this work [78] and the further review done in Chapter 4. In the results part (Chapters 78 and 9) three tentative systems were analyzed, according to the acceleration level that they offered. The results obtained seem to confirm that the use of DESHA is possible for all attempted patterns at least with the system presented for medium acceleration level.

The two other systems proposed failed. The low acceleration level system required too much time to form the most complex patterns. The high acceleration level system proposed ran out of propellant for the most complex patterns. It is therefore concluded that a good trade-off between thrust and propellant use (and availability) is necessary in order to use DESHA in space applications.

Therefore it can be concluded that already with the current state of the art technical solutions it seems that DESHA can be used in small spacecraft based missions for pattern formation with swarm intelligence. For the patterns presented, the formation acquisition time in the most complex patterns reached the level of months. This also will be increased with increasingly larger inter-satellite distances. Nevertheless, optimization techniques and large scale techniques are expected to reduce these times. Therefore, if one or two months are acceptable as formation times, the current state of the art technology seems capable of swarming spacecraft with DESHA.

10.1.3. Sub-Question 3: Failures and Pitfalls

The third sub-question tried to answer if any unexpected situations from the implementation of DESHA were to be identified.

> Does DESHA present any kind of pitfall or unexpected blockage?

After the experiments presented in the third part of this work, it can be said that no unexpected blockages or failures have been identified in DESHA. If the simulation is properly tuned, i.e. the run-time frequency of the high and low level controller is properly tuned, the time step is according to the dynamics, etc., the algorithm is expected to converge in all cases.

It is true that with the time constraints presented in this report, some pitfalls, such as the need for a fairly powerful actuation system with efficient propulsion have been identified. Also in the tuning of the simulator, it has been proven vital to properly select the knowledge radius of the agents. This is due to phenomenons such as the overshoot and drift on the control window of the agents. If the knowledge radius is strictly the inter-satellite distance, the agents that drift apart, within their control window, might not be detected by their neighbours. This might lead to disconnection events. Also, the importance of taking into account possible collision avoidance events, where one of the agents has to turn back to its initial position has been identified . If DESHA is run while this is happening, it might disconnect the returning agent. Still, if a proper tuning is performed, no problem should arise from the use of DESHA.

10.2. Research Question

The main research question of this whole project was:

Is it possible to control and guide a swarm of spacecraft using DESHA?

The simulation tool presented in this work, as well as the experiments presented in the third part of it, show that this question can be answered positively. Throughout this report it has been proven that, if the assumptions presented in this work are met, it is possible to guide and control a swarm of spacecraft using the DESHA algorithm. Furthermore, depending on the scenario, it has been proven that multiple patterns can achieve the formation desired. Also it has been shown in Section 10.1 that if the right pattern is selected, the swarm will be flexible, robust and scalable. This will allow, as compared with conventional formation flying, to cope with unexpected events, such as the loss of agents. Also, the expansion of the mission by adding more agents, or the change of the purpose by modifying the formation will be possible. This will give the formation an adaptability, thanks to the swarm intelligence, which traditional approaches do not posses.

Compared to other approaches in the literature, such as behavioral algorithms or machine learning approaches (see Chapter 1), this algorithm presents a set of advantages. Unlike behavioral algorithms the time to tune DESHA is reduced to following a set really simple practical considerations. Furthermore, no stuck situations have been detected, nor any is expected as, by design the swarm always converges to the pattern. Furthermore, all kinds of patterns that match the rules presented in [31] are possible. This includes non-symmetrical patterns, not achievable with many behavioral approaches [52]. Also, the algorithm and its working is fully understandable by humans, and does not depend on any kind of simulation processes, unlike machine learning approaches [16].

On the other hand, some disadvantages have been detected. To start with, convergence times will be larger than with other algorithms. This is due to the probabilistic nature of the artificial intelligence approach taken. The use of an stochastic policy allows the swarm to actually cope with unexpected situations and have a certain degree of intelligence, understood as the capacity to cope and solve unexpected situations [79]. On the other hand, this and the local knowledge of the algorithm lead to the fact that many movements are required to achieve the pattern. Furthermore, most of the times the movements taken are not useful or even counter-productive. This translates in a large number of thrusting actions, with their corresponding use of propellant.

Some approaches can be taken to reduce the number of actions required for convergence as optimized PFSMs [30] or large scale approaches (see Chapter 6). This was done in Chapter 9 and a significant improvement on the performance of the algorithm was observed. Nevertheless, again due to the probabilistic and local nature of the algorithm, still many movements will be taken with no sense. This is inherent to the algorithm.

Overall it can be concluded that DESHA is capable of forming patterns and giving a certain degree of adaptability through swarm intelligence to a swarm of spacecraft. Nevertheless, the fact that many of the movements that it generates do not lead to productive actions lead to think that this might not be the best option for a swarm to form a pattern. In space, due to the cost associated to put in orbit any object, the use of propellant tends to be maximized. Therefore, using an algorithm that does many wasteful actions does not seem to be the best option. Maybe centralized or global approaches, such as the ones presented in [38, 68] would provide a better solution for pattern formation in spacecraft swarms. Nevertheless, DESHA might be used in combination with those, or as a safety algorithm, to mitigate the risk of losing the swarm when the communications between agents or the key elements of the swarm fail, acting therefore as a safety algorithm.

10.3. Future Work

In this work DESHA has been proven to be useful to guide and control a swarm of spacecraft providing the system with a certain degree of intelligence. DESHA has shown to offer some unique advantages over other state of the art algorithms. Nevertheless, due to its local nature and probabilistic approach

some disadvantages have been detected in its use.

In the future, further research on mitigating this inherited disadvantages of DESHA will be in order. To do so, it is recommended to further explore to research the use of DESHA in conjunction with other approaches. For example DESHA could be used as a secondary algorithm next to a behavioral approach. The behavioral algorithm could be then used as primary high-level controller, but when it would get stuck, DESHA might come into place to move the agents and avoid the stuck situation. Also research on the optimization of the fuel would be in order to reduce the use of it. To do so, it is recommended to study the use of MPC (Model Predictive Control) as low-level controller. MPC is a kind of controller which, by design, allows to minimize both the control output and the input. It is already proposed in literature for swarm control [69] and for general spacecraft control [57] and presents the advantage of also taking into account the control input limits (that is the maximum and minimum levels of thrust). This will also ease the design of the controller, opposite to the algorithm to design a PID controller needed in Chapter 5. Some preliminary studies on its implementation in SwarmSimulator were already done. Originally the use of in-house routines from the TU Delft was attempted as the whole code was available. Nevertheless the routines turned out not to be fitted to the problem at hand and its tuning required further knowledge on control theory and optimization. Due to time constraints this is left as future work. Also the use of MatLab® already implemented code was tested, and although the results were positive, the form of the output was not suitable with the design of SwarmSimulator. Further work on how SwarmSimulator can save and process the whole thrusting schedule created by MatLab®'s can be incorporated will be necessary. Nevertheless, it is recommended that in future versions of SwarmSimulator a MPC controller is developed from scratch to tailor both the optimization algorithms used and its output to SwarmSimulator.

Another interesting development proposed for future work to reduce the number of wasteful movements is the increase of knowledge of the agents. As it has been noted in Section 10 global approaches seem to avoid this problem, or at least mitigate it, as the agents can actually know if they should move or if it is their neighbours who needs to move. For example in Figure 10.3 if the most right agent moves down the pattern is completed. With the canonical implementation of DESHA all three active agents just know that they need to move. Nevertheless, if they were to know a larger portion of the swarm they might be able to actually detect that the one that should move should be the one of the right. It would be interesting to study if there is an optimum on the portion of the swarm that the agents would need to know to actually avoid most wasteful movements without overly increasing the communication needs.

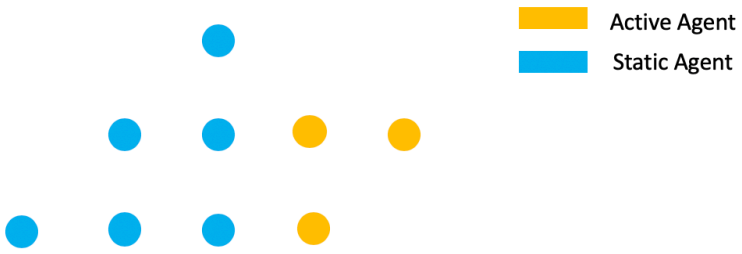


Figure 10.3: Triangle of 9 agents mission one movement for creating the pattern.

Another point of improvement of the system to reduce the use of propellant, mostly for non-electric propulsion systems, could be the use of two impulse approaches. As it has been noted before, in this work a continuous thrust approach has been taken, as the initial needs on the thrusters indicated that electric propulsion will be preferred. Nevertheless, the high level acceleration thruster was a chemical propelled system. This system reduced greatly the time, but ran out of fuel most of the times when the thruster is not extremely efficient (see [86]). That is why it is worth studying the use of two impulse approaches, as the ones used in orbit maneuvers [93, 98] for thrusters with higher thrust levels, but lower efficiency. If the thrust is high enough, it can actually set with a single firing the agent in a course that will intersect the desired position. Then the thrust could be shut off until the desired position is about to be reached, moment when the thruster would be fired again to stabilize the agent in the position. This

will reduce greatly the use of the thruster, more in long inter-satellite distances. On the other hand, the time performance will be probably worst, as the agent might go slower than with continuous high thrust. Therefore a trade-off between these to variables will be in order.

Finally, the last proposed future work is an improvement on the dynamic modelling. The HCW equations are widely used in literature to model formation flying [4]. Nevertheless, they are only a linearization and require a really specific reference orbit, a circular orbit. It is already proven in [51] that even the slightest inclination might actually cause a drift between this dynamics model and reality quite big over time. Furthermore, the linearization might not always be valid for the orbit used depending on the full aperture of the swarm and the reference orbit. In the case that SwarmSimulator were to be used to study and predict the orbit in a real mission, the HCW will not be sufficient. That is why in SwarmSimulator another relative dynamic model has been implemented. This dynamic model is presented in [18] and is based on the Gauss Variational Equations(GVE). This approach is already not a linearization, so it is valid for a wider range of orbits than the HCW. Furthermore, it allows to include the effect of perturbations (important on certain orbits or for long mission times) as shown in [18]. The model implemented is based on the modified orbital elements, as the traditional ones do not allow for the use of equatorial circular reference orbits. The set of elements used is the one presented in [5] as they allow to use all kind of reference orbits. The equations were derived using the procedure showed in [18] (without the J_2 term) and implemented as a state space model (see the HCW in Chapter 5). Also the conversion from these orbital elements to Cartesian coordinates and standard orbital elements are included in SwarmSimulator [5]. In order to use these model it should be validated first. This could be done using specialized already tested software, such as GMAT [50] or real data from a space mission (if available). This was not done during this project due to the limited amount of time, which did not allow to master the use of the software required for validation. Nevertheless, with a small amount of work it will be possible to include this model in SwarmSimulator.

A

Appendix A: Neglected Effects in Dynamics

A.1. Massive Bodies Effects

Following the formula presented in [98](Equation 3.2), allows calculating an estimate of the order of magnitude of the forces affecting the swarm's agents. For the sake of simplicity, and given that the goal is to obtain just an estimate, all agents were considered to have 1kg of mass and be in the reference orbit of 200km of altitude above the Moon. The reader can see that in the event these numbers were changed to the actual masses and positions of the swarm agents, the order of magnitude and even the forces calculated would barely change. Finally, as a reminder, the reference frame is an inertial reference frame whose axes are coincident with the ECI but whose origin is located on the center of the Moon.

$$\frac{d^2\bar{r}_i}{dt^2} = -G\frac{m_i + m_k}{r_i^3}\bar{r}_i + G\sum_{j\neq i,k} m_j\left(\frac{\bar{r}_j - \bar{r}_i}{r_{ij}^3} - \frac{\bar{r}_j}{r_j^3}\right) \tag{A.1}$$

First, the force due to the main body, the Moon, is calculated, which corresponds to the first term on the right side of the equation. Since the mass of the Moon is much larger than that of the agent, the first term is modified to:

$$F_{Moon-Swarm} = -G\frac{m_{Moon}m_{agent}}{r_{Moon-Swarm}^3}\bar{r}_i \tag{A.2}$$

Now the magnitude of the standard gravitational parameter of the Moon (GM) can be obtained from [103] as well as its volumetric mean radius. Taking into account that the distance from the center of the Moon to the swarm is:

$$r_{Moon-Swarm} = r_{Moon} + r_{SwarmAltitude} = 1937.4km$$

Therefore, by applying Equation A.2 is possible to obtain the gravitational acceleration of the Moon on the swarm.

To calculate the effect of the perturbing bodies the second term of Equation 3.2 is used. In this case, the considered perturbing bodies are the Earth and the Sun given that due to their mass/distance they are the ones that most probably affect the most the swarm orbit. The effect of the perturbing bodies is calculated with Equation A.3

$$F_{Perturbing} = Gm_{Body}m_{agent}\left(\frac{\bar{r}_{Moon-Body} - \bar{r}_{Moon-Swarm}}{r_{Swarm-Body}^3} - \frac{\bar{r}_{Body}}{r_{Body}^3}\right) \tag{A.3}$$

The point here is to have an estimate for the Moon-Body distance and the Body swarm distances. Unlike the swarm, the Earth rotates in an elliptical orbit around the Moon. In a first approximation, the same can be said about the Sun. This means that the distances needed will vary with time. The best

approach for estimating the force will be calculating it when it is maximum.

The distance between the body and the swarm can be decomposed in the distance from the Moon to the body and the distance from the Moon to the swarm. Analyzing Equation A.3, the equation will be maximized if the distance from the swarm to the body is minimum and the distance from the Moon to the body is maximum. Nevertheless this leads to contradiction. The distance from the body to the Moon is maximum at the apogee of the orbit of the body around the Moon, whereas since the swarm orbits the Moon, the minimum distance between the swarm and the body will be in the perigee of the orbit of the Moon. Since calculating the optimum to estimate the force is deemed unnecessary another simplification is taken.

A good approximation to obtain the searched order of magnitude will be just to consider both orbits circular and use the semi-major axis of the orbit as the used distance. Unlike the previously presented approach, it is slightly less precise, but it allows for a faster calculating without loosing too much precision, as the eccentricities of the orbits considered are quite low.

Therefore the distances to between the bodies and the swarm will be calculated as:

$$r_{Swarm-Body} = a_{Body} - r_{Moon-Swarm}$$

All needed data for the semi-major axes of the orbit of the Earth around the Moon and the Sun around the Moon, as well as the standard gravitational parameters of both bodies will be obtained from [101, 102]. Since the data of the orbit of the Sun around the Moon is not redily available, the semi-major axis of the orbit of the Sun around the Moon will be estimated as:

$$a_{Sun} = a_{Earth-Sun} - a_{Moon-Earth}$$

Considering therefore that the Moon orbits the Sun in a circular orbit slightly closer to it than the Earth from the Sun's perspective.

Then, to obtain the force ratios, in percentage, of each body's effect body on the swarm, the following formula is used:

$$Ratio_{body} = \frac{F_{body}}{\sum F_{body}}$$

The results and used data are presented in Table A.1:

	Moon	**Earth**	**Sun**	**Spacecraft**
Distance to Origin (10^6km)	0	0.36329644	146.694647528	1.9374 km
Mass (10^{24}kg)	0.07346 $\cdot 10^{22}$	5.9724	1988500	1kg
Source of Data	[103]	[101]	[102]	n/a
Force (N)	1.305	2.7398 $\cdot 10^{-5}$	1.5478 $\cdot 10^{-7}$	n/a
% Force Ratio wrt. Total	99.9978	0.0021	1.1856$\cdot 10^{-5}$	n/a

Table A.1: Gravitational influence of the effect of the Moon, Sun and Earth on the swarm calculated with Equation 3.2

A.2. Perturbations: Spherical Harmonic Gravity

As explained in Section 3.4.2 the gravity potential of a solid body similar to a sphere but with a slightly different shape and non-uniform mass distribution is represented by Equation 3.3 and its produced acceleration is represented by Equation 3.4. In this section the order of magnitude of the J_2 term geopotential acceleration of the Moon on the spacecraft is estimated. By substituting only taking until the J_2 term in Equation 3.3 and substituting it in Equation 3.4 the formula for the acceleration produced by the adding to an sphere a band of mass in its equator. This is represented in Equation A.4.

$$\bar{a_{J_2}} = -\nabla\left[\frac{1}{2}\mu J_2 \frac{R^2}{r^3}\left(3\frac{z2}{r^2} - 1\right)\right] \tag{A.4}$$

Taking the derivatives the result is:

$$\begin{aligned} a_{J_2 r} &= \frac{3}{2}\mu J_2 \frac{R^2}{r^4}(3sin^2\phi - 1) \\ a_{J_2\phi} &= -\frac{3}{2}\mu J_2 \frac{R^2}{r^4} sin(2\phi) \\ a_{J_2\Delta} &= 0 \end{aligned} \tag{A.5}$$

The Equations A.5 have a maximums at $\pm\frac{pi}{2}$ for the radial acceleration and $\pm\frac{pi}{4}$ for the longitudinal acceleration. Said values for the maximums are:

$$|a_{J_2 r}|_{max} = |a_{J_2\phi}|_{max} = 3\mu J_2 \frac{R^2}{r^4} \tag{A.6}$$

Taking the data for the mean radius and of J_2 of the Moon, the value of the swarm distance from the Section A.1 the estimate for $|a_{J_2}|_{max} = 6.384 \cdot 10^{-4} m/s$. Again taking the a 1kg swarm agent this results into a perturbing force of $6.384 \cdot 10^{-4} N$.

A.3. Solar Radiation Pressure

The effects of the solar radiation pressure are estimated, according to Waker [98], as:

$$a_{RP} = -C_R \frac{WA}{Mc} \tag{A.7}$$

The speed of light and the mass of the spacecraft are already known constants. The energy flux can be estimated by considering that the energy output from the Sun is constant. Knowing that the Sun emits around $62.94 \cdot 10^6 W/s$ [102]. Considering that this energy is never dissipated:

$$Q_{SunSurface} = Q_{SwarmOrbit}$$

$$W_{SunSurface} \cdot 4\pi R_{Sun}^2 = W_{SwarmOrbit} \cdot 4\pi R_{SwarmOrbit}^2$$

By considering the swarm orbit as a circular orbit around the Sun and taking the same approximations as in Section A.1 the energy flux in the swarm orbit will be:

$$W_{SwarmOrbit} = 1368.21 W/m^2$$

To obtain an order of magnitude of the area of the spacecraft, the CubeSat standard is used. A CubeSat unit is a $10cmx10cmx10cm$ cube [26]. Therefore a good approximation to obtain the order of magnitude of the affected area is the area of one of its faces, that is, $10^{-2} m^2$. Note that this is just an estimation, in reality not only one face, but probably more, will be affected by the radiation. Finally, in [98] an example of a the effects of solar radiation pressure are presented for the Echol, a giant aluminum coated sphere. It is expected that the agents of the swarm have less reflectivety than a body mainly made to reflect radiation. Since Echol's $C_R = 1.9$ and only an order of magnitude is wanted, it is fair to estimate the $C_R \approx 1$. With all these data:

$$a_{RP} \approx 4.564 \cdot 10^{-8} m/s^2$$

Which, considering that the estimated mass is 1kg, the estimated perturbation force is $f_{RP} = 4.564 \cdot 10^{-8} N$

B

Appendix B: Description and Functions of Swarm Simulator Relevant Classes

This Appendix describes the most relevant elements of some of the classes implemented in Swarm Simulator. This does not intend to be a full software manual, but rather a guide for the reader of this work to understand how some of the results and models have been implemented.

Here an index of the classed and functions within them are reviewed

- *Agent*
- *Thruster*
  - *updatepropellant*
- *Orbit*
  - *propagate*
- *State*

B.1. *Agent*

The agent class intends to incorporate all the information related to each of the elements of the swarm. This class will include also a set of subclasses such as the controller, state or thruster classes which will model specific elements of the design of the agents. To model the agents, the properties included in Table B.1 will be defined:

The class will incorporate the functions compiled in Table B.2:

All functions either have been explained in several chapters of the thesis or are self explanatory. Only the function *agent.collisionavoidance* will be reviewed in this section.

The main cause of collision is the movement by two agents out of sight of each other towards the same target. The more agents, the more local neighbourhoods and the more probability of collision events.

The strategy implemented is pretty simple. Each agent, during a movement, remembers to points, the initial state and the final state. The former will act as a safety target, while the latter will act as the target for the low level controller. In the even that two agents range themselves within less than the safety radius, a collision event is created. The first agent to range the other is set to be in priority and it exchanges its current target for its safety target. As it is the only one in its previous neighbourhood to have moved, it is guaranteed that the agent will not disconnect by going back to the previous target. Once the agent has been returned to its former position, the priority flag is set off.

Name	Property Name	Variable Type
Thruster	Thruster	thruster object
Controller	oController	controller object
Reaction Wheel	Weel	reactionwheel object
Antenna Position	AntPos	real array
Antenna	Antenna	antenna object
Dry Mass of the Agent	Mass	real
Orbit Dynamics Model	OrbitDynamics	character string
Relative Position (in LHLV)	Position	real array
Relative Velocity (in LHLV)	Velocity	real array
Attitude Quaternion	Attitude	real array
State	oState	state object
Knowledge Ratius	rKnowledgeRadius	real
Movement Radius	rMovementRadius	real
Target of the high level controller	arTarget	real array
Active Flag	lActive	logical
DeltaV Used	rTotalDeltaV	real
Priority Flag	lPriority	logical
Safety Target	arSafetyTarget	real array

Table B.1: Properties of the *Agent* class

Function	Description
agent	Class Initializer
CheckLocalState	Checks if the agent is in a desired state
propagateorbit	Orbit propagator
actuateorbit	Control routine. Decides if the error is large enough, and if so calls the controller.
collisionavoidance	Collision avoidance routine
doIknowyou	Function to check in another agent is within the radius of knowledge.
gridpoint	Function to translate a position to the nearest point in the discretization of the space done
CreateReatcionWheel	Creates the reaction wheel objects within the agent.
CreateThrster	Creates thruster objects within the agent.
CreateAntennas	Creates antenna objects within the agent.

Table B.2: Functions of the *Thruster* class

B.2. *Thruster*

The thruster class intends to incorporate the modelling of the thruster. As such, it incorporates the data in Table B.3as properties of the class:

The class will incorporate the functions compiled in Table B.4

The functions *thruster* and *thruster.newthruster* are pretty self explainatory and only have operational relevance. Therefore the only models reviewed will be those of *thruster.updatepropellant*.

For this function the following assumptions are taken:

- **AS-HW-012**: It is supposed that the thrust is always varied through the variation of the mass flow of the engine.

Name	Property Name	Variable Type
Thruster Name	csName	Character String
Maximum Thrust (N)	rMaxThrust	Real
Minimum Thrust (Thrust Step) (N)	rMinThrust	Real
Specific Impulse (s)	rIsp	Real
Propellant Mass (kg)	rPropMass	Real
Max Mass Flow (kg/s)	rMassFlow	Real
ΔV (m/s)	rDeltaV	Real
Dry Mass (kg)	rDryMass	Real

Table B.3: Properties of the *Thruster* class

Function	Description
thruster	Class Initializer
newthruster	Creates a user defined thruster object. It is necessary to define its properties
updatepropellant	Updates the ΔV used of the thruster and the propellant mass

Table B.4: Functions of the *Thruster* class

- **AS-HW-013**: It is supposed that the mass flow is constant (no need for a variation of the current to compensate for variations in the plasma flow)

Assumption AS-HW-012 is used to omit the variations in thrust due to variations in the neutralizer, electric field, etc. This also generates the assumption that the exit velocity of the propellant is constant.

Assumption AS-HW-013 the mass flow might slightly vary due to small variations in the electric field or in the valves of propellant. This assumption implies an ideal working of the elements of the engine.

The function calculates the propellant used and the ΔV done for each thrusting action in the following way. With Equation B.1 the mass flow is calculated (being $\dot{m}$ the mass flow, T the thrust, I_{sp} the specific impulse and g_0 the reference gravity). Then with Equation B.2 the mass of the thrusting event is calculated. Finally, the rocket Equation (B.3 allows to obtain the ΔV and the propellant left in the tank is just updated using the result of Equation B.2. All these equations have been extracted from the classical rocket engine theory. For more information on the derivations see [106].

$$\dot{m} = \frac{T}{I_{sp} g_0} \tag{B.1}$$

$$M_{used} = \dot{m} * t_{thurst} \tag{B.2}$$

$$\Delta V = I_{sp} g_0 log\left(\frac{1 + M_{used}}{M_{propellant} + M_{dry}}\right) \tag{B.3}$$

$$M_{propellant} = M_{propellantinit} - M_{used} \tag{B.4}$$

Note that the I_{sp} is a charactersitic of the engine given by the manufacturer. As it is the ratio between the thrust force and the weight of the propellant expelled, and said ratio was calculated at Earth with the sea level gravity, g_0 will be sea level gravity on Earth.

B.3. *Orbit*

The *orbit* class aims to incorporate all the information about the reference orbit of the swarm. This will include orbit parameters, but also relevant variables such as the period or mean motion. All formulas used to obtain the derived parameters are extracted from [98].

Name	Property Name	Variable Type
Semi-Major Axis (km)	a	Real
Inclination (rad)	i	Real
Eccentricity	e	Real
Argument of Perigee (rad)	omega	Real
RAAN (rad	Omega	Real
Time constant (s)	tau	Real
Mean Motion (rad/s)	rN	Real
Orbital Period (s)	rT	Real
Gravitational Parameter (m^3/s^2)	rMu	Real

Table B.5: Properties of the *Orbit* class

The orbit class also incorporates the following methods:

Function	Description
orbit	Class Initializer
Polar2Cart	Transforms polar coordinates into Cartesian coordinates
GetApogee	Obtains the apogee of the orbit
propagate	Generates, given a time step, a vector with the orbit positions in Cartesian coordinates.

Table B.6: Functions of the *orbit* class

All functions are sufficiently self explanatory or its derivation is presented in [98]. The only function here reviewed will be the propagation.

The propagation function generates, given a time step and the final time of the propagation the positions of the orbit. To do so first, through the mean motion of the orbit a vector with the true anomaly of the orbit at each time is obtained with:

$$\theta = n \cdot t$$

Where θ is the true anomaly, n is the mean motion of the orbit and t is the time. Then, the radius of each position is obtained with the equation for keplerian orbits:

$$r = \frac{a \cdot (1 - e^2)}{1 + ecos\theta}$$

Where a is the semi-major axis and e is the eccentricity. Finally with the function Polar2Cart this result is transformed to Cartesian coordinates.

B.4. *State*

The state class aims to compile all the state shape related information. To do so it incorporates the following properties:

Name	Property Name	Variable Type
State Vector	arState	Real Array
Shape of the State	csType	Character String
Radius Correction	rRadiusFactor	Real
Translation Vector	arMoveMat	Array Real

Table B.7: Properties of the *State* class

The radius correction will refer to the correction on the movement radius to reach the diagonal elements in cube and square state shapes. The translation vector translates the movements encoded in the PFSMs from [30, 31] to SwarmSimulator.

The *state* class will incorporate the following public functions:

Function	Description
state	Class Initializer
create	Generates an array with the corresponding size full of 0s
Translate State	Translates each non-zero element of the state to cartesian coordinate positions (with respect to the agent)
reset	Sets the state back to all 0s
CheckConnectivity	Ensures that the state represented is fully connected
GenerateMovemat	Fills the *arMoveMat* property according to the state type.

Table B.8: Functions of the *state* class

All functions are self explainatory. For more details on the algorithms, the code is step-by-step commented.

C

Number of Total Actions Analysis for Several Patterns

This appendix presents the analysis on the number of total actions required for several patterns used during this work. The procedure on how to obtain this results is presented in Chapter 6.

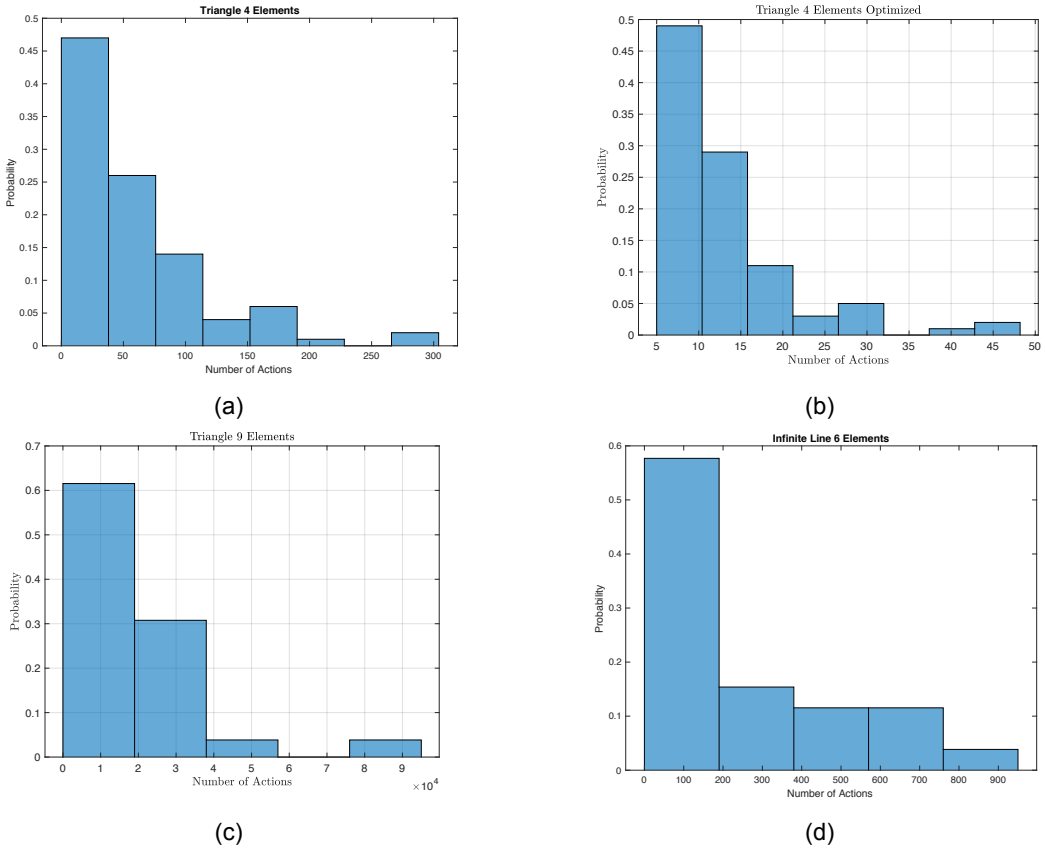


Figure C.1: Convergence test for 4 Element Triangle (C.1a, 4 Element Triangle Optimized (C.1b, 9 Element Triangle (C.1c and line of 6 elements (C.1d

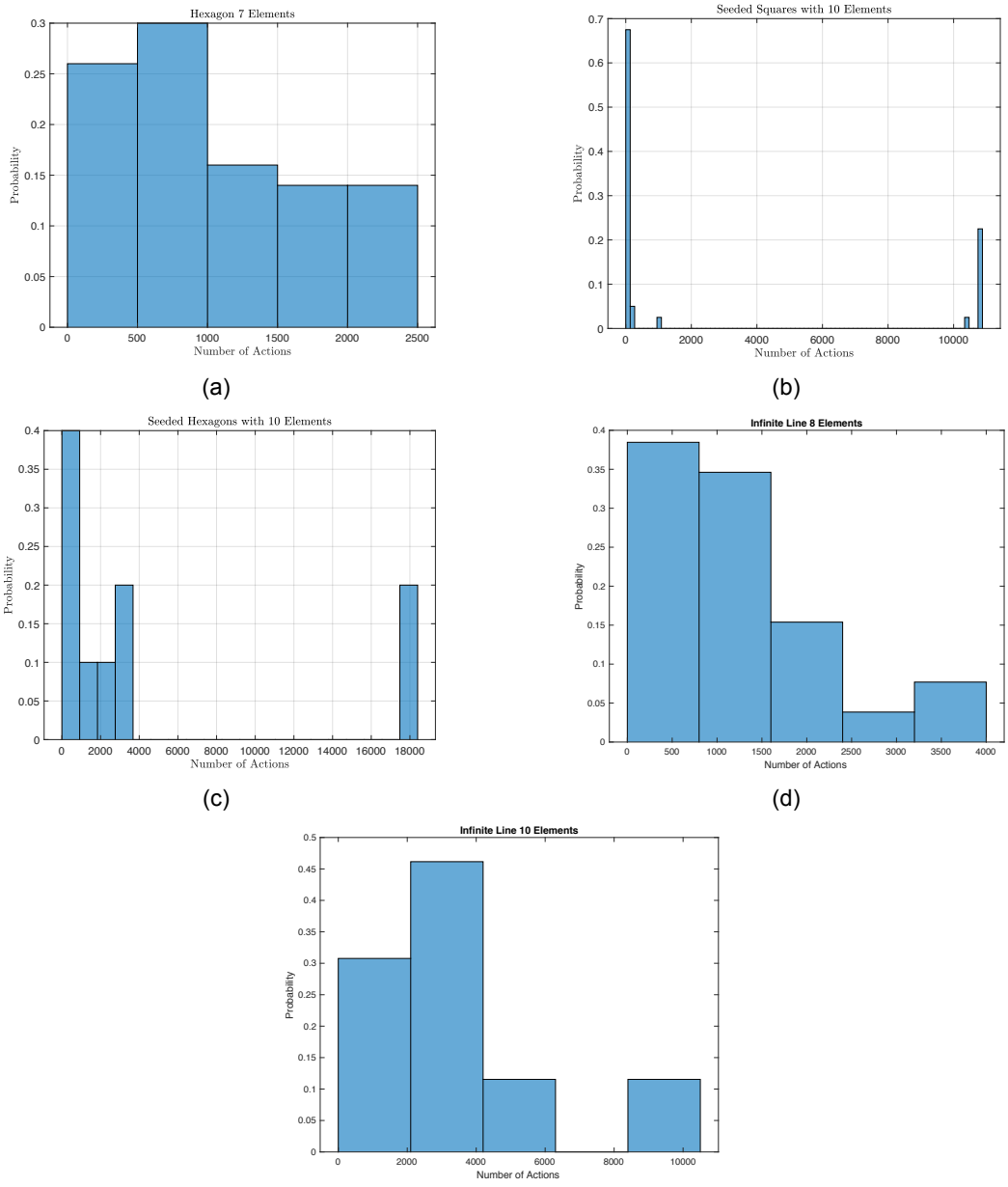


Figure C.2: Convergence test for 7 Element Hexagon (C.2a, 10 Element Square Pattern (C.2b, 10 Element HoneyComb (C.2c, line of 8 elements (C.2d and line of 10 elements (C.2e)

D

Controller Settings

In this appendix the gains of the PID controllers used are presented. Each column will refer to the gain set for the radial direction (first column), along track direction (second column) and cross-track direction (third column).

D.1. BIT-3 Engine

$$\begin{bmatrix} K_P \\ K_I \\ K_D \end{bmatrix} = \begin{bmatrix} -0.000252124450108908 & -0.000252124450108908 & -0.000242124450108908 \\ -1.47708305899997 \cdot 10^{-7} & -1.47708305899997 \cdot 10^{-7} & -1.47708305899997 \cdot 10^{-7} \\ -0.6042252304574286 & -0.6042252304574286 & -0.1642252304574286 \end{bmatrix} \tag{D.1}$$

D.2. BIT-7 Engine

$$\begin{bmatrix} K_P \\ K_I \\ K_D \end{bmatrix} = \begin{bmatrix} -0.000275371534485476 & -0.000275371534485476 & -0.000275371534485476 \\ -1.829586613931737 \cdot 10^{-6} & -3.629586613931737 \cdot 10^{-6} & -3.629586613931737 \cdot 10^{-6} \\ -0.0856598299898347 & -0.0856598299898347 & -0.0856598299898347 \end{bmatrix} \tag{D.2}$$

D.3. BGT-X5 Engine

$$\begin{bmatrix} K_P \\ K_I \\ K_D \end{bmatrix} = \begin{bmatrix} -0.000564130197112355 & -0.000564130197112355 & -0.000564130197112355 \\ -1.52115288208233 \cdot 10^{-6} & -1.52115288208233 \cdot 10^{-6} & -1.52115288208233 \cdot 10^{-6} \\ -0.051117310716962 & -0.051117310716962 & -0.051117310716962 \end{bmatrix} \tag{D.3}$$

Bibliography

[1] *TU Delft Astrodynamics Toolbox*, 2018. Accessed on-line March 6th 2019.

[2] Behçet Açıkmeşe and David S Bayard. Markov chain approach to probabilistic guidance for swarms of autonomous agents. *Asian Journal of Control*, 17(4):1105–1124, 2015.

[3] Behçet Açıkmeşe, Nazlı Demir, and Matthew W Harris. Convex necessary and sufficient conditions for density safety constraints in markov chain synthesis. *IEEE Transactions on Automatic Control*, 60(10):2813–2818, 2015.

[4] Kyle Alfriend, Srinivas Rao Vadali, Pini Gurfil, Jonathan How, and Louis Breger. *Spacecraft formation flying: Dynamics, control and navigation*, volume 2. Elsevier, 2009.

[5] Paul Anderson and Hanspeter Schaub. Impulsive feedback control of nonsingular elements in the geostationary regime. In *AIAA/AAS Astrodynamics Specialist Conference*, page 4585, 2012.

[6] Gianluca Antonelli, Filippo Arrichiello, and Stefano Chiaverini. The null-space-based behavioral control for autonomous robotic systems. *Intelligent Service Robotics*, 1(1):27–39, 2008.

[7] Ronald C Arkin. Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351–364, 1992.

[8] Ronald C Arkin and Tucker Balch. Cooperative multiagent robotic systems. 1997.

[9] Karl Johan Åström, Tore Hägglund, and Karl J Astrom. *Advanced PID control*, volume 461. ISA-The Instrumentation, Systems, and Automation Society Research Triangle …, 2006.

[10] Frank H Bauer, Joel JK Parker, Bryan Welch, and Werner Enderle. Developing a robust, interoperable gnss space service volume (ssv) for the global space user community. 2017.

[11] Levent Bayindir and Erol Şahin. A review of studies in swarm robotics. *Turkish Journal of Electrical Engineering & Computer Sciences*, 15(2):115–147, 2007.

[12] Derek J Bennet and Colin R McInnes. Pattern transition in spacecraft formation flying using bifurcating potential fields. *Aerospace Science and Technology*, 23(1):250–262, 2012.

[13] Marinus Jan Bentum, CJM Verhoeven, AJ Boonstra, AJ Van Der Veen, and EKA Gill. A novel astronomical application for formation flying small satellites. *NERG-Nederlands Elektronica en Radiogenootschap*, 76(1):8, 2011.

[14] Grant Bonin, Niels Roth, Scott Armitage, Josh Newman, Ben Risi, and Robert E Zee. Canx–4 and canx–5 precision formation flight: Mission accomplished! 2015.

[15] S. Boyd and J. Duchi. Ee364b: Convex optimization ii lecture notes. sequential convex programing. Online, 2018. URL `https://web.stanford.edu/class/ee364b/lectures/seq_notes.pdf`.

[16] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[17] Louis Breger and Jonathan How. J2-modified gve-based mpc for formation flying spacecraft. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 5833, 2005.

[18] Louis Breger and Jonathan P How. Gauss's variational equation-based dynamics and control for formation flying spacecraft. *Journal of Guidance, Control, and Dynamics*, 30(2):437–448, 2007.

[19] A Budianu, Arjan Meijerink, and Marinus Jan Bentum. Swarm-to-earth communication in olfar. *Acta astronautica*, 107:14–19, 2015.

[20] Alexandru Budianu, Teodoro J Willink Castro, Arjan Meijerink, and Mark J Bentum. Inter-satellite links for cubesats. In *2013 IEEE Aerospace Conference*, pages 1–10. IEEE, 2013.

[21] ISIS Innovative Solutions In Space B.V. On board computer (iobc). Online, 2015. `https://www.isispace.nl/wp-content/uploads/2016/02/iOBC-Brochure-v1.pdf`.

[22] Jet Propulsion Laboratory Caltech. Iris v2.1 cubesat deep space transponder. Online, 2016. URL `https://www.jpl.nasa.gov/cubesat/pdf/Brochure_IrisV2.1_201611-URS_Approved_CL16-5469.pdf`.

[23] Vincenzo Capuano, Cyril Botteron, and Pierre-André Farine. Gnss performances for meo, geo and heo. In *Space Communications and Navigation Symposium-Space-Based Navigation Systems and Services*, number CONF, 2013.

[24] Angelo Cervone, Barry Zandbergen, Jian Guo, Eberhard Gill, WPW Wieling, F Tata Nardini, and CAH Schuurbiers. Application of an advanced micro-propulsion system to the delffi formation-flying demonstration within the qb50 mission. 2012.

[25] Qifeng Chen, Sandor M Veres, Yaonan Wang, and Yunhe Meng. Virtual spring-damper mesh-based formation control for spacecraft swarms in potential fields. *Journal of Guidance, Control, and Dynamics*, 38(3):539–546, 2015.

[26] Alexander Chin, Roland Coelho, Ryan Nugent, Riki Munakata, and Jordi Puig-Suari. Cubesat: the pico-satellite standard for research and education. In *AIAA Space 2008 Conference & Exposition*, page 7734, 2008.

[27] Soon-Jo Chung and FY Hadaegh. Swarms of femtosats for synthetic aperture applications. 2011.

[28] Pamela E Clark, Ben Malphrus, Kevin Brown, Terry Hurford, Cliff Brambora, Robert MacDowall, David Folta, Michael Tsay, Carl Brandon, and Lunar Ice Cube Team. Lunar ice cube: Searching for lunar volatiles with a lunar cubesat orbiter. In *AAS/Division for Planetary Sciences Meeting Abstracts# 48*, volume 48, 2016.

[29] John Conklin, Nathan Barnwell, Leopoldo Caro, Maria Carrascilla, Olivia Formoso, Seth Nydam, Paul Serra, and Norman Fitz-Coy. Optical time transfer for future disaggregated small satellite navigation systems. 2014.

[30] Mario Coppola and Guido CHE de Croon. Optimization of swarm behavior assisted by an automatic local proof for a pattern formation task. In *International Conference on Swarm Intelligence*, pages 123–134. Springer, 2018.

[31] Mario Coppola, Jian Guo, Eberhard Gill, and Guido CHE de Croon. Provable self-organizing pattern formation by a swarm of robots with limited knowledge. *Swarm Intelligence*, pages 1–36, 2019.

[32] Steven A Curtis, J Mica, J Nuth, G Marr, M Rilee, and M Bhat. Ants(autonomous nano technology swarm)- an artificial intelligence approach to asteroid belt resource exploration. In *IAF, International Astronautical Congress, 51 st, Rio de Janeiro, Brazil*, 2000.

[33] Simone D'Amico. *Autonomous formation flying in low earth orbit*. PhD thesis, TU Delft, Delft University of Technology, 2010.

[34] E Dekens, S Engelen, and R Noomen. A satellite swarm for radio astronomy. *Acta Astronautica*, 102:321–331, 2014.

[35] Nazlı Demir, Utku Eren, and Behçet Açıkmeşe. Decentralized probabilistic density control of autonomous swarms with safety constraints. *Autonomous Robots*, 39(4):537–554, 2015.

[36] G Di Mauro, M Lawn, and R Bevilacqua. Survey on guidance navigation and control requirements for spacecraft formation-flying missions. *Journal of Guidance, Control, and Dynamics*, 41(3): 581–602, 2017.

[37] C Donlon, B Berruti, A Buongiorno, M-H Ferreira, P Féménias, J Frerick, P Goryl, U Klein, H Laur, C Mavrocordatos, et al. The global monitoring for environment and security (gmes) sentinel-3 mission. *Remote Sensing of Environment*, 120:37–57, 2012.

[38] P D'Arrigo and S Santandrea. The apies mission to explore the asteroid belt. *Advances in Space Research*, 38(9):2060–2067, 2006.

[39] Mahmoud El Chamie and Behçet Açıkmeşe. Safe metropolis–hastings algorithm and its application to swarm control. *Systems & Control Letters*, 111:40–48, 2018.

[40] Samsung Electronics. Samsung galaxy s10. on-line, 2019.

[41] Steven Engelen, Chris JM Verhoeven, and Mark J Bentum. Olfar, a radio telescope based on nano-satellites in moon orbit. 2010.

[42] Steven Engelen, Eberhard KA Gill, and Chris JM Verhoeven. Systems engineering challenges for satellite swarms. In *Aerospace Conference, 2011 IEEE*, pages 1–8. IEEE, 2011.

[43] Francisco Amarillo Fernández. Inter-satellite ranging and inter-satellite communication links for enhancing gnss satellite broadcast navigation data. *Advances in Space Research*, 47(5):786–801, 2011.

[44] Eduardo Castelló Ferrer. The blockchain: a new framework for robotic swarm systems. In *Proceedings of the Future Technologies Conference*, pages 1037–1058. Springer, 2018.

[45] Rebecca Foust, Soon-Jo Chung, and Fred Hadaegh. Autonomous in-orbit satellite assembly from a modular heterogeneous swarm using sequential convex programming. In *AIAA/AAS Astrodynamics Specialist Conference*, page 5271, 2016.

[46] L Garrigues and P Coche. Electric propulsion: comparisons between different concepts. *Plasma Physics and Controlled Fusion*, 53(12):124011, 2011.

[47] Eberhard Gill, Prem Sundaramoorthy, Jasper Bouwmeester, B Zandbergen, and R Reinhard. Formation flying within a constellation of nano-satellites: The qb50 mission. *Acta Astronautica*, 82(1):110–117, 2013.

[48] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[49] Fred Y Hadaegh, Soon-Jo Chung, and Harish M Manohara. On development of 100-gram-class spacecraft for swarm applications. *IEEE Systems Journal*, 10(2):673–684, 2016.

[50] Steven P Hughes. General mission analysis tool (gmat). 2016.

[51] Gkhan Inalhan, Michael Tillerson, and Jonathan P How. Relative dynamics and control of spacecraft formations in eccentric orbits. *Journal of guidance, control, and dynamics*, 25(1):48–59, 2002.

[52] Dario Izzo and Lorenzo Pettazzi. Equilibrium shaping: distributed motion planning for satellite swarm. In *Proc. 8th Intern. Symp. on Artificial Intelligence, Robotics and Automation in space*. Citeseer, 2005.

[53] Dario Izzo and Lorenzo Pettazzi. Autonomous and distributed motion planning for satellite swarm. *Journal of Guidance, Control, and Dynamics*, 30(2):449–459, 2007.

[54] Dario Izzo, Luís F Simões, and Guido CHE de Croon. An evolutionary robotics approach for the distributed control of satellite formations. *Evolutionary Intelligence*, 7(2):107–118, 2014.

[55] Adam W Koenig and Simone D'Amico. Robust and safe n-spacecraft swarming in perturbed near-circular orbits. *Journal of Guidance, Control, and Dynamics*, pages 1–20, 2018.

[56] Kristina Lemmer. Propulsion for cubesats. *Acta Astronautica*, 134:231–243, 2017.

[57] Mirko Leomanni, Eric Rogers, and Stephen B Gabriel. Explicit model predictive control approach for low-thrust spacecraft proximity operations. *Journal of Guidance, Control, and Dynamics*, 37 (6):1780–1790, 2014.

[58] Willem JM Levelt. *An introduction to the theory of formal languages and automata*. John Benjamins Publishing, 2008.

[59] Wanyu Ma, Yanchao Sun, Guangfu Ma, and Chuanjiang Li. Multi-spacecraft system distributed cooperative tracking control with dead-zone input. In *Control Conference (CCC), 2017 36th Chinese*, pages 8695–8700. IEEE, 2017.

[60] Malcolm Macdonald and Christopher John Lowe. It's hip to be square: The cubesat revolution. *Aerospace*, pages 22–25, 2014.

[61] María Manzano-Jurado, Julia Alegre-Rubio, Andrea Pellacani, Gonzalo Seco-Granados, Jose A López-Salcedo, Enrique Guerrero, and Alberto García-Rodríguez. Use of weak gnss signals in a mission to the moon. In *2014 7th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC)*, pages 1–8. IEEE, 2014.

[62] David W Matula and Robert R Sokal. Properties of gabriel graphs relevant to geographic variation research and the clustering of points in the plane. *Geographical analysis*, 12(3):205–222, 1980.

[63] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[64] B. Menser, D. Tilbury, R. Hill, and J.D. Taylor. Introduction: Pid controller design. Online, 2017. URL `http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID`.

[65] T. M. Mitchell. *Machine Learning*. McGraw-Hill Education, 1 edition, March 1997.

[66] Andreas F Molisch. *Wireless communications*, volume 34. John Wiley & Sons, 2012.

[67] Daniel Morgan, Soon-Jo Chung, Lars Blackmore, Behcet Acikmese, David Bayard, and Fred Y Hadaegh. Swarm-keeping strategies for spacecraft under j2 and atmospheric drag perturbations. *Journal of Guidance, Control, and Dynamics*, 35(5):1492–1506, 2012.

[68] Daniel Morgan, Soon-Jo Chung, and Fred Y Hadaegh. Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and model predictive control. In *AIAA guidance, navigation, and control conference*, page 0599, 2015.

[69] Daniel Morgan, Giri P Subramanian, Soon-Jo Chung, and Fred Y Hadaegh. Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming. *The International Journal of Robotics Research*, 35(10):1261–1285, 2016.

[70] Mainak Mukhopadhyay, Binay Kumar Sarkar, and Ajay Chakraborty. Augmentation of anti-jam gps system using smart antenna with a simple doa estimation algorithm. *Progress In Electromagnetics Research*, 67:231–249, 2007.

[71] Sreeja Nag and Leopold Summerer. Behaviour based, autonomous and distributed scatter manoeuvres for satellite swarms. *Acta Astronautica*, 82(1):95–109, 2013.

[72] Ranjith Ravindranathan Nair, Laxmidhar Behera, Vinod Kumar, and Mo Jamshidi. Multisatellite formation control for remote sensing applications using artificial potential field and adaptive fuzzy sliding mode control. *IEEE Systems Journal*, 9(2):508–518, 2015.

[73] Shayegan Omidshafiei, Ali-Akbar Agha-Mohammadi, Christopher Amato, Shih-Yuan Liu, Jonathan P How, and John Vian. Decentralized control of multi-robot partially observable markov decision processes using belief space macro-actions. *The International Journal of Robotics Research*, 36(2):231–258, 2017.

[74] Carlo Pinciroli, Mauro Birattari, Elio Tuci, Marco Dorigo, Marco del Rey Zapatero, Tamas Vinko, and Dario Izzo. Self-organizing and scalable shape formation for a swarm of pico satellites. In *Adaptive Hardware and Systems, 2008. AHS'08. NASA/ESA Conference on*, pages 57–61. IEEE, 2008.

[75] Li Qiao, Eamonn Glennon, Andrew G Dempster, and Sebastian Chaoui. Using cubesats as platforms for remote sensing with satellite navigation signals. In *2013 IEEE International Geoscience and Remote Sensing Symposium-IGARSS*, pages 852–855. IEEE, 2013.

[76] Shizhen Qu, Chunlin Chen, and Daoyi Dong. Behavior-based control of swarm robots with improved potential field. In *Control Conference (CCC), 2014 33rd Chinese*, pages 8462–8467. IEEE, 2014.

[77] Raj Thilak Rajan and Alle-Jan van der Veen. Joint motion estimation and clock synchronization for a wireless network of mobile nodes. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2845–2848. IEEE, 2012.

[78] A. Ripoll. Literature study: An analysis on space swarming. June 2018.

[79] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited„ 2016.

[80] Julian Scharnagl, Panayiotis Kremmydas, and Klaus Schilling. Model predictive control for continuous low thrust satellite formation flying. *IFAC-PapersOnLine*, 51(12):12–17, 2018.

[81] Marsil AC Silva, Daduí C Guerrieri, Angelo Cervone, and Eberhard Gill. A review of mems micropropulsion technologies for cubesats and pocketqubes. *Acta Astronautica*, 143:234–243, 2018.

[82] S. Slavney. Gravity recovery and interior laboratory(grail) data sets. on-line, 2018.

[83] Roy Sterritt, Christopher Rouff, James Rash, Walter Truszkowski, and Michael Hinchey. Self-* properties in nasa missions. In *In 4th International Workshop on System/Software Architectures (IWSSA'05) in Proc. 2005 International Conference on Software Engineering Research and Practice (SERP'05*. Citeseer, 2005.

[84] J Sanz Subirana, JM Juan Zornoza, and M Hernández-Pajares. Gnss data processing. volume 1: Fundamentals and algorithms. *ESA Communications ESTEC, PO Box*, 299:2200, 2013.

[85] R Sun, D Maessen, J Guo, and E Gill. Enabling inter-satellite communication and ranging for small satellites. In *9th Symposium on Small Satellites Systems and Services, Funchal, Portugal*, volume 31, 2010.

[86] George P Sutton and Oscar Biblarz. *Rocket propulsion elements*. John Wiley & Sons, 2016.

[87] Michael Tillerson, Gokhan Inalhan, and Jonathan P How. Co-ordination and control of distributed spacecraft systems using convex optimization techniques. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 12(2-3):207–242, 2002.

[88] Cho WS To. *Introduction to Dynamics and Control in Mechanical Engineering Systems*. John Wiley & Sons, 2016.

[89] Walt Truszkowski, Harold Hallock, Christopher Rouff, Jay Karlin, James Rash, Michael Hinchey, and Roy Sterritt. *Autonomous and autonomic systems: with applications to NASA intelligent spacecraft operations and exploration systems*. Springer Science & Business Media, 2009.

[90] Michael Tsay, John Frongillo, Jurg Zwahlen, and Lenny Paritsky. Maturation of iodine fueled bit-3 rf ion thruster and rf neutralizer. In *52nd AIAA/SAE/ASEE Joint Propulsion Conference*, page 4544, 2016.

[91] Michael Tsay, Charlie Feng, and Jurg Zwahlen. System-level demonstration of busek's 1u cubesat green propulsion module "amac". In *53rd AIAA/SAE/ASEE Joint Propulsion Conference*, page 4946, 2017.

[92] Akshay Tummala and Atri Dutta. An overview of cube-satellite propulsion technologies and trends. *Aerospace*, 4(4):58, 2017.

[93] David A Vallado. *Fundamentals of astrodynamics and applications*, volume 12. Springer Science & Business Media, 2001.

[94] Pieter van Vugt, Arjan Meijerink, and Mark Bentum. Calibration of the olfar space-based radio telescope using a weighted alternating least squares approach. In *Aerospace Conference, 2017 IEEE*, pages 1–11. IEEE, 2017.

[95] Tiago Varum, João N Matos, and Pedro Pinho. Direction of arrival estimaation analysis using a 2d antenna array. *Procedia Technology*, 17:617–624, 2014.

[96] Chris JM Verhoeven, Marinus Jan Bentum, GLE Monna, Jeroen Rotteveel, and Jian Guo. On the origin of satellite swarms. *Acta Astronautica*, 68(7-8):1392–1395, 2011.

[97] Enrique Vidal, Franck Thollard, Colin De La Higuera, Francisco Casacuberta, and Rafael C Carrasco. Probabilistic finite-state machines-part i. *IEEE transactions on pattern analysis and machine intelligence*, 27(7):1013–1025, 2005.

[98] Karel Wakker. *Fundamentals of Astrodynamics*. 01 2015. ISBN 978-94-6186-419-2.

[99] P Wang and DP Kwok. Optimal design of pid process controllers based on genetic algorithms. *Control Engineering Practice*, 2(4):641–648, 1994.

[100] Bong Wie. *Space vehicle dynamics and control*. American Institute of Aeronautics and Astronautics, 2008.

[101] David R Williams. Earth fact sheet. *Structural geology of the Earth's interior: Proc. Natl. Acad. Sci. NASA (17 Nov 2010)*, 76(9), 2004.

[102] David R Williams. Sun fact sheet. *Retrieved February*, 1:2011, 2004.

[103] David R Williams. Moon fact sheet. *NASA Fact sheets*, 2006.

[104] Matthew Willis and Simone D'Amico. Analytical approach to spacecraft formation-flying with low-thrust relative spiral trajectories. *Acta Astronautica*, 153:175–190, 2018.

[105] P.H. Winston. *Artificial Intelligence*. Addison-Wesley, 1992.

[106] Barry Zandbergen. Thermal rocket propulsion. *Delft University of Technology*, 2018.

[107] Lin Zhao and Yingmin Jia. Neural network-based distributed adaptive attitude synchronization control of spacecraft formation under modified fast terminal sliding mode. *Neurocomputing*, 171: 230–241, 2016.

[108] An-Min Zou and Krishna Dev Kumar. Neural network-based distributed attitude coordination control for spacecraft formation flying with input saturation. *IEEE transactions on neural networks and learning systems*, 23(7):1155–1162, 2012.