Politecnico di Milano
Department of Aerospace Science and Technology
Doctoral Programme in Aerospace Engineering

AI-augmented Guidance, Navigation and Control for Proximity Operations of Distributed Systems

Doctoral Dissertation of:
Stefano Silvestrini

Supervisor:
Prof. Michèle Lavagna

Tutor:
Prof. Francesco Topputo

Coordinator:
Prof. Pierangelo Masarati

Year 2020 – Cycle XXXIII

Ad Alberto.

Abstract

Future science and exploration missions will implement innovative mission concepts to embark on daring endeavors exploiting cooperating intelligent systems. The concept of Distributed Space Systems boosts the achievable mission objectives due to the flexibility and adaptivity that the system inherently possesses. For instance, large synthetic instrumentation can be built by dedicated configurations of the system or very close-approach fly-by with small bodies can be performed; moreover, the single-point failure, typical of large satellites, is generally avoided in Distributed Systems and Formation Flying architecture. All these features are achieved at the cost of a significant increase in the on-board autonomy. Due to the relative distances involved, each agent composing the system needs to be able to rapidly and autonomously react to unforeseen events, such as collisions. To this end, it is critical that each spacecraft is capable of planning, navigating and controlling itself in unknown or partially-known environment, without ground-intervention.

The research work presented here focuses on the development and testing of a full Guidance, Navigation & Control system aided by Artificial Intelligence techniques. The enhancement provided by Artificial Intelligence techniques allow the system to fly in uncertain environments by incrementally learning its mathematical modeling. The Thesis develops a number of methods to recover the underlying dynamics by simply measuring relative state and processing it with an Artificial Neural Network. This dynamics is then used to plan control action and enhance the navigation and control synthesis. In particular, three methods for dynamics reconstruction are developed, together with their mathematical

foundation. The three approaches integrate the Artificial Neural Network at different levels: from fully integrated, where the dynamics is completely encapsulated into an Artificial Neural Network, to partially integrated in which the network learns either the unknown dynamical accelerations or reconstructs the unknown parameters of the analytical expression. Such reconstruction scheme is used in two different planning and control algorithms: Neural-Artificial Potential Field method and Model-Based Reinforcement Learning. The former is a fast and light algorithm that easily handles collision avoidance but lacks of planning; the latter is able to generate plans and control the spacecraft based on the learnt dynamics. Given the distributed planning architecture, each spacecraft does not know how the rest of the system is evolving. For this reason, it has been necessary to develop an AI-based routine coupling Long-Short Term Memory and Inverse Reinforcement Learning to predict the behavior of external agents, being either in free-motion or controlled-motion.

The algorithms, when compared to classical methods, showed superior performance and constant increase in relevant Guidance, Navigation & control metrics (navigation accuracy, maneuvers Δv, etc.). Finally, in order to increase the Technology Readiness Level of the algorithms, the work presents the Processor-In-the-Loop testing campaign executed with relevant hardware: a micro-controller unit and a single-board computer with similar computational power with respect to flight-hardware. An end-to-end autocoding procedure has been developed to transition from Model-In-the-Loop simulations to Processor-In-the-Loop validation. The tests were deemed successful by evaluating the execution times, resource utilization and achieved accuracy.

The outcome of the Thesis is a complete framework to integrate different AI-based techniques to enhance existing, well-established, algorithms. The methodology described here can easily be extended to other mission scenarios, where the flexibility and adaptivity of the system is critical.

Acknowledgements

I have always deemed the acknowledgments section as the toughest and most delicate part of a thesis. It is the hardest because you do not want to miss anyone. It is the deepest because you would like to express your feelings, to translate them in words. It goes without saying that I will miserably fail in fulfilling both requirements, but I will certainly try my best.

The first person I would like to thank is my supervisor Prof. Michèle Lavagna. Besides being a source of inspiration to become a researcher, I have always been astonished by her passion and dedication. These are still common acknowledgements that you may find in many other thesis works: the most impressive achievement is the success in creating a group, welcoming me from nowhere and to lay trust in each one of us. This is what makes you an incredible Professor! Grazie, ad ASTRA!

Undoubtedly, I would have struggled a lot more in my life if I did not have any special friends supporting me, always.

Pietro: probably the most suitable way to thank you is to tap on your shoulder yelling a cold: "Grazie". You would appreciate this. Nevertheless, I must at least mention how much I felt your support throughout the years. It would be hard to condense our story in one paragraph so I will just point out significant episodes: pista blandas, duna, ADP for Japan, Scotland, chocolate roll, sofa, open-wine, socialism, Kaki,...even the keywords are too many. Thank you for being such a friend!

A special thought to Gianluca and Michela. An everlasting friend who met his counterpart: a wonderful person. What's more? Well, you are simply the most natural and spontaneous support for whichever difficulty one may need to face. The special bond is shared with your peculiar brother, Begi, who has been my "Michela" in our unforgettable *Poracci in Viaggio* endeavors. Vi voglio bene!

A huge virtual hug (hard times!) to my everlasting friends, from Venice and Milan: *I fioi*, Lorenzo, Jessica, Marta, Massimo, Stefania, Anna, Alessia and all the others.

Now, a group of people deserves a dedicated paragraph: ASTRA. It has been such a great time together and hopefully it will be the same for the next years. Andrea, the first person I met at Politecnico, the first coffee and the first friend. You have been such a great wall to hang on in difficult times and mentor in most of the activities but, above all, a true and unique friend. Thank you! Luca, what can I say? Being idiot together became our bonding: now I can tell you, I was almost in tears when you left. It has been simply amazing having you as a friend in these years and even though you live very far away (Torino) I know I can always count on you and your van. Bear in mind that we are still together for the PDA. Indeed, it is time to mention Vincenzo. What a master and intimate friend: we shared so much in such a short time, one would think that we should not have stopped...indeed, Bibbia is coming!

Capa, my classmate, you may have considered the idea of killing me throughout the years but I truly admit that I shared with you deepest fears and best joys. You are unique, for sure.

JP, my wonderful sherpa, you are the person who changed me the most, definitely. I would have never spent so much money if had never met you. Beside jokes and beside work, you have been a true motivation for life. I started climbing thanks to you, I started skialp thanks to you. Most importantly, I just had a lot of fun thanks to you. Grazie, so che apprezzi l'italiano. Another person fell in your trap: Teo. What a special guy! I will never forget you yelling at rocks and, of course, being the first student I supervised.

Vanni, the one and only Gramsci of DAER. I hardly laughed as much as I did with you. Honestly, working with you has been great lately. You are a truly kind-hearted person and I admire you. I admire you because you can solve the Rubik's cube in less than two minutes. Certainly, I admire you for thousands of other aspects, but I prefer to mention only this, since I know I can fully express myself and be sincere with you. Grazie!

Marghe, you are the one who balanced the group. I can tell you that most of the projects we did together were successful, on time and exciting...maybe I should finally give you all the credits! Thank you.

X

To all the others: Bucci, Paolo, Fabio, Pasqua, Brando, ASTRA. Thank you but I am running out of space.

The doctoral dissertation is the last step of a student academic career. Although very far in time, I do not want to miss the opportunity to thank my high school teacher Prof. Del Maschio: you are the one who triggered the learning process, stimulated my curiosity and motivated me. You embody all the fundamental characteristics a teacher should have and, in case I will ever have the chance to teach something, I aspire to do it as you did. Thank you!

The last paragraph of this section is dedicated to the most solid and eternal bond of my life: my family. Fede, brother, we had the chance to live back together after years apart and I think this was a critical brick paving the way to the success of this path. You have been the most genuine support for every single day of this life, the protective wings under which I could fully develop myself. It would be hard to mention even a percentage of the joyful moments together: in engineering terms, it cannot be discretized such continuous flow. Thank you for helping me grow, think, behave. Together with you, a special and unique dedication to my parents: the two pillars of my life. Nothing, literally nothing, would have been possible without your unconditional support. You will always be my life model. Indeed, you were my first teachers who simply enlightened me on how to live.

Venezia-Milano,
20.01.2021

Table of Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence.
ANN	Artificial Neural Network.
APF	Artificial Potential Field.
CW	Clohessy-Wiltshire Model.
DSS	Distributed Space System.
EKF	Extended Kalman Filter.
EO	Earth Observation.
FF	Formation Flying.
FK	Formation Keeping.
FMA	Feature Matching Approach.
FR	Formation Reconfiguration.
GNC	Guidance, Navigation & Control.
HNN	Hopfield Neural Network.
IRL	Inverse Reinforcement Learning.
LRNN	Layer-Recurrent Neural Network.
LSTM	Long-Short Term Memory.
LVLH	Local-Vertical-Local-Horizontal.

MBRL	Model-Based Reinforcement Learning.
MCR3BP	Modified Circular Restricted Three-Body Problem.
MPC	Model Predictive Control.
NARX	Nonlinear Autoregressive Network with Exogenous Inputs.
P2BP	Perturbed Two-Body Problem.
Ph.D.	Philosophiae Doctor.
PIL	Processor-In-the-Loop.
QP	Quadratic Programming.
RBFNN	Radial-Basis Function Neural Network.
RMSE	Root Mean Squared Error.
RNN	Recurrent Neural Network.
ROE	Relative Orbital Elements.
SR	Safe Reconfiguration.
SRP	Solar Radiation Pressure.
TOF	Time Of Flight.
TRL	Technology Readiness Level.

CHAPTER 1

Introduction

> Anyone whose goal is 'something higher' must expect someday to suffer vertigo.
> What is vertigo?
> Fear of falling?
> No, Vertigo is something other than fear of falling. It is the voice of the emptiness below us which tempts and lures us, it is the desire to fall, against which, terrified, we defend ourselves.
>
> — Milan Kundera

During the last decades, new promising space mission concepts are being developed to enable humanity to undertake daring explorations. Among those, Formation Flying (FF) and Distributed Space System (DSS) are attracting the interest of the whole space community. The idea of having multiple spacecraft involved in proximity operations, in known or unknown environments, yields several advantages in terms of achievable mission objectives. First of all, it is claimed that unprecedented mission performance and objectives can be

attained using fractionated instruments, such as telescopes. The capability of distributing the instrument and operating with interferometry measurements yields the possibility of achieving very large synthetic apertures, improving image quality. Secondly, the robustness of the entire mission is significantly enhanced due to the distribution of mission tasks among the system. The single-point failure, inherently present in large monolithic spacecraft mission, is solved thanks to the presence of several autonomous agents. In particular, in Distributed Space Systems each agent autonomously makes decisions by measuring the relative state of the formation. The distributed concept solves the typical shortcomings of centralized and decentralized approaches. The former rely on a processing agent, which gathers information on the state of the formation and delivers command to the fleet. This approach does not improve robustness of the mission, being the decision process concentrated in the *centralized mothercraft*. The latter comprise autonomous agents that process their own information without interacting with the rest of the formation at all. Such approach limits the capability and the achievable mission objectives of the formation due to the blindness of each agent with respect to the others. Moreover, Distributed Space Systems increases the flexibility of the mission thanks to the capability of reconfiguring the formation in orbit. For instance, this means that the synthetic aperture can be varied while flying or a certain formation global attitude can be adjusted in orbit. In this way, several macroscopic objectives can be achieved with one single mission. The platforms used for Distributed Space Systems are typically identical in a sort of formation standardization. This implies a significant cost reduction as well as limited *time-to-flight* for the entire formation. Such cost limitation allows mission concepts in which low-cost agents are embarked into bigger satellites and deployed into very risky operations, such as planetary or asteroid proximity operations [1].

1.1 Context & Motivation

The context of the research covers the implementation of Distributed Space Systems flying in formation to execute risky proximity maneuvers (e.g. close asteroid fly-by, tight formation), see Fig. 1.1. As already mentioned, the usage of such mission concept brings several advantages, such as increased scientific performance, more flexibility and cost reduction. Among all the spacecraft subsystems, the Guidance, Navigation and Control system is the mostly affected. Indeed, it represents the *brain* of the satellite, being in charge of planning, navigating and controlling the spacecraft. The challenges that the GNC needs to face are caused by the short relative distances involved and the uncertain environment that the spacecraft has to witness. To tackle this, a high-level of autonomy is required. It consists in being able to react to unforeseen events and to re-plan control action depending on the surrounding

environments. The reason for excluding human intervention as a feasible option for mission implementation is that the close proximity requires rapid execution and decisions, which are not guaranteed by the inherent delays present in the *ground-in-the-loop* control.

Distributed Space Systems typically rely on multiple satellites of restricted size and cost. This is mandatory in order to initialize and perform the mission using a limited number of launches. Nevertheless, as the platforms become smaller, the computational power lowers. Hence, on one hand, the higher level of autonomy requires more sophisticated algorithms to be run; on the other hand, the spacecraft is constrained in computational power by the size and cost with respect to resources present on-board large monolithic satellites.

The realm of *ground-robotics* has already implemented techniques that try to solve the same shortcomings, without considering the typical constraints of space missions. Among such techniques falls the *umbrella term* of Artificial Intelligence. Artificial Intelligence, here in this Thesis limited to the concept of Artificial Neural Networks and the framework of Reinforcement Learning, rely on extensive training before embarking the models into embedded applications. Traditionally, to create a light algorithm, classical methods rely on linearization or simplification of the mathematical models. The vast majority of the embedded systems have been designed based on linear algebra and linearization. This is true also for the space technology. Although, the linear design (and thinking) has served our purposes very well in the past, it imposes constraints and limitations on the potential of current technology for more demanding space missions. Nature and generally universe has non-linear behavior. Putting linear systems into a non-linear environment requires a lot of effort and resources from the engineers in order to make it right. In addition, this extra effort and resources are reflected in the design which becomes complex. One way to break the barrier of the current technological limitation could be to move into non-linear systems. Non-linear systems increase the capabilities of the technology and new attributes emerge from its implementation. Of course, attention is required in order to enhance the attributes that serve the objectives of the mission and to depreciate the ones that create complications. One space application that could greatly benefit from application of non-linear systems is the autonomous Guidance, Navigation and Control of spacecraft, as remarked before. Indeed, the demand of highly accurate relative GNC in a non-linear and unpredictable harsh environment including time constraints compels the designers to consider alternative methodologies and concepts. One of these ideas is to use adaptive systems with non-linear elements. Here we go: Artificial Intelligence is a product of a non-linear adaptive system. The way it can be manifested depends on the technology level and the current knowledge. For instance, Artificial Intelligence techniques could allow the spacecraft to achieve highly accurate relative navigation when arriving to an unknown target or to plan and control

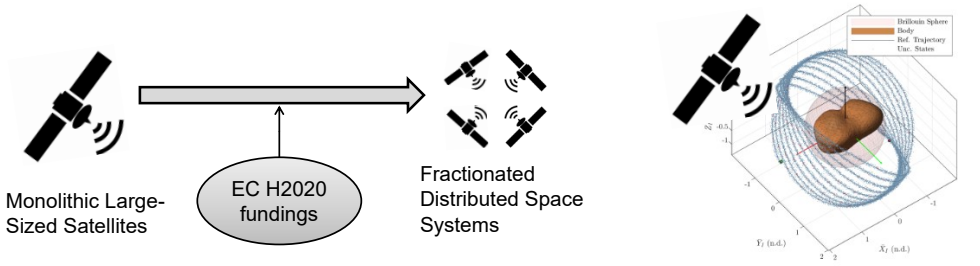


Figure 1.1: Proximity Operations of Distributed Space Systems.

the satellite in uncertain environments. However, there are several challenges to be solved for application of AI techniques in autonomous GNC systems, for instance inadequate data sets for training, demonstrate generalization to different environment, capability to pre-train or transfer learning to speed up the training during the real operations, or modeling mathematically the performance behavior under different environmental conditions. Before moving to the formalization of the research problem, four requirements have been identified as guidelines in the development of the thesis. The autonomous GNC shall rely on algorithms that feature:

- **Accuracy**: The Guidance, Navigation and Control system needs to rely on accurate dynamical models in order to exploit the underlying physics to estimate the state, to generate reference trajectories and to control the orbital state.
- **Flexibility**: The Guidance, Navigation and Control System system should be applicable to different mission scenario without major modifications.
- **Adaptivity**: The Guidance, Navigation and Control system should be able to react to unforeseen events and refine its response to the environment autonomously.
- **Low Computational Resource**: The Guidance, Navigation and Control system needs to be deployed and executed into on-board computers, with limited computational resources. Thus, the algorithms shall be suitable for embedded applications.

1.2 The Research Problem

The research problem arises from the necessity of having autonomous fast, adaptive, flexible and accurate Guidance, Navigation & Control system in microsatellites mission flying in uncertain environments.

1.2.1 Objectives

The development of autonomy in space missions is the core engine of the Thesis, given the context explained in Section 1.1. In particular, given the listed requirements (accuracy, flexibility, adaptivity and low computational power) and heritage of *ground-robotics* in Artificial Intelligence, the research objective can be summarized in this way:

> *To analyze and implement AI-based learning algorithms for the refinement of on-board dynamical model, reconfiguration planning and formation control. To develop on-board AI-augmented autonomous guidance and control algorithms for distributed space systems during proximity operations.*

The research objective has been generated by the translation of research requirements into AI-augmented GNC techniques. Indeed, by analyzing each requirements, generated from the identified shortcomings of current technologies, one can build a workflow depicted in Fig. 1.2.

The Guidance, Navigation & Control system relies on mathematical model of the environment to calculate the estimated states and planned control. The accuracy and adaptivity of the algorithms is pursued by implementing online learning based on AI-techniques. This allows the agent to learn the surrounding environment as it flies, refining its mathematical representation on-board. The refinement increases the accuracy of the model, on which GNC is synthesized on-board, but also it captures variation of the environment, enhancing the adaptivity of the algorithm. As stated, such refined model is integrated in the control and planning routines. In this way, the learning procedures serves the purpose of enhancing the whole GNC. Regardless of the environment, whatever dynamics the agent learns, the planning and control are based on the AI-model reconstructed on-board. For this reason, there is not a specificity of the mission and the algorithms are flexible enough to be used either for asteroid proximity or Earth-bounded close formation. Moreover, given the focus on Distributed Space Systems, the objective is to develop AI-enhanced techniques that are capable of predicting the whole system evolution without communication requirements. Each agent should in principle be able to understand what the rest of the fleet is doing by only observing them. This is required to prevent collision while performing relative maneuvers simultaneously.

1.2.2 Research Questions

To fulfill the research objectives, a set of research questions, which will guide the development of the Thesis is presented:

1. What are the traditional techniques? What are the identified shortcomings of such approaches? What are the innovative aspects that need to be added in order to fulfill the research objective?
2. How can we possibly construct on-board a model of the dynamics? Is it possible to learn the dynamics in orbit, with limited or absent initialization of the method offline?
3. How can we complement existing algorithm with the learnt dynamics? What are the key aspects of the GNC loop that would benefit from the AI-augmentation?
4. How can the whole Distributed Space System simultaneously operate and control without sharing information on each others' future plans? How can the agent predict the behavior of the external environment?
5. Is it possible to embed the developed algorithms in representative hardware to increase their TRL?

1.3 Dissertation Overview

The Thesis dissertation follows the research questions listed above. In particular, the logical flow is depicted in Fig. 1.2. The research work firstly focused on developing methods to learn the dynamics on-board. In particular, three approaches have been developed. The approaches propose incremental integration of the Artificial Neural Networks in the on-board dynamics. In particular, one technique encapsulates the full dynamics within a trained ANN. Recurrent Neural Networks have shown superior performance in learning the dynamics due to their inherent temporal behavior. The second technique utilizes a Radial-Basis Function Neural Network to estimate online all the unmodeled terms of the implemented analytical model. Finally, the third method implements a Recurrent Neural Network for estimating uncertain parameters of a given analytical model. The refinement of the dynamics serves the purpose of enhancing the GNC algorithms. Indeed, an Artificial Potential Field algorithm for Guidance and Control and an Extended Kalman Filter for navigation have been coupled with the ANN-reconstructed dynamics, showing significant improvement with respect to standard approaches. The Artificial Potential Field Guidance and Control generates a guidance dynamics based on the contribution of fictitious attractive potential, toward target state, and repulsive one, generated to avoid collision. The method is very fast and light but it may cause instability. Also, it does not optimize any control action, which is not recommended for space missions. In order to solve several shortcomings of the Artificial Potential Field algorithm, an optimization-based technique has been developed for Guidance and Control. The refined dynamics is used to plan and control in closed-loop, in a similar fashion as the Model

Predictive Control scheme. The developed technique is called Model-Based Reinforcement Learning, or alternatively Neural-Predictive Control. In order to develop an algorithm capable of predicting external agents behavior and trajectories for collision avoidance, Inverse Reinforcement Learning is used aided by a Long-Short Term Memory network. Basically, the latter method is a recurrent neural network that predicts neighboring satellites' trajectories; the former tries to guess the cost function that delivers the observed trajectories through a series of simple nested optimization. Finally, all the algorithms have been tested in relevant hardware and their execution times evaluated against available resource utilization. The test campaign was performed using Processor-In-the-Loop simulations.

The thesis is structured as follows: Chapter 2 presents the background and state-of-the-art of the autonomous GNC for distributed systems. In particular, for each aspect of the GNC, it highlights the major shortcomings of traditional methods as well as the assumptions used so far in literature.

Chapter 3 presents the whole development of AI-based dynamics learning. In particular, all the aforementioned methodologies are presented and thoroughly described. The advantages and disadvantages of each method is highlighted and test cases are reported to show the reconstruction performance.

Chapter 4 focuses on the usage of the learnt dynamics to enhance the Guidance and Control scheme. In detail, the Neural-aided Artificial Potential Field is first described and tested. The remarkable results of enhancing the algorithm with ANN-based dynamics are shown and highlighted. Subsequently, the Model-Based Reinforcement Learning, fully relying on the ANN-based dynamics, is derived and tested.

Chapter 5 discusses the methods to predict the behavior of external agents. This feature is critical for the GNC algorithms in order to prevent collision between concurrent agents that maneuvers simultaneously in a distributed architecture. In particular, Inverse Reinforcement Learning is derived mathematically and tested together with the developed Long-Short Term Memory recurrent network.

Chapter 6 describes the experimental activities focused on the Processor-In-the-Loop validation of the developed algorithms. The hardware suite and the porting procedure are described, on top of the actual test results.

Finally, Chapter 7 reports the conclusions of the research work. In particular, the major results and findings are highlighted and remarked before giving a list of recommendations for future work.

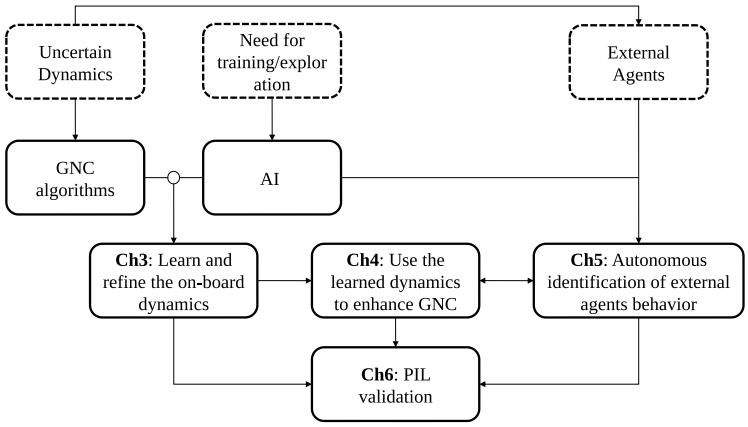


Figure 1.2: Overview of the Thesis logical flow.

1.4 Bibliographic Disclaimer

During the years of my Philosophiae Doctor (Ph.D.), I presented updates of my work in many conferences and I also had the possibility to publish part of them in peer reviewed journals. Therefore, most of the work in this dissertation has already been presented in different articles. The most significant are listed below.

- S. Silvestrini, M. Lavagna, "Neural-based Predictive Control and Relative Trajectory Identification Algorithms for Relative Spacecraft Maneuvers", *Journal of Guidance, Control and Dynamics*, 2020 (under review)

- A. Pasquale, S. Silvestrini, A. Capannolo, P. Lunghi, M. Lavagna, Small Bodies Non-Uniform Gravity Field Online/On-Board Learning through Hopfield Neural Networks, *Acta Astronautica*, 2020 (under review)

- S. Silvestrini, M. Lavagna, "Neural-aided GNC Reconfiguration Algorithm for Distributed Space System: Development and PIL test", *Advances in Space Research*, 2021, doi:10.1016/j.asr.2020.12.014

- V. Pesce, S. Silvestrini, M. Lavagna, "Radial Basis Function Neural Network aided Adaptive Extended Kalman Filter for Spacecraft Relative Navigation", *Aerospace Science and Technology*, vol. 96, 2020, doi:10.1016/j.ast.2019.105527

- S. Silvestrini, M. Lavagna, "Processor-in-the-Loop Testing of AI-aided Algorithms for Spacecraft GNC", *71st International Astronautical Congress*, The Cyberspace Edition, virtual, 12-14 October 2020
- S. Silvestrini, M. Lavagna, "Spacecraft Formation Relative Trajectories Identification for Collision-Free Maneuvers using Neural-Reconstructed Dynamics", *AAS/AIAA SciTech Forum*, Orlando, 6-10 January 2020, doi:10.2514/6.2020-1918
- S. Silvestrini, M. Lavagna, "Inverse Reinforcement Learning for Collision Avoidance and Trajectory Prediction in Distributed Reconfigurations", *70th International Astronautical Congress*, Washington, USA, 21-25 October 2019
- A. Pasquale, S. Silvestrini, M. Lavagna, "Non-uniform gravity field model on board learning during small bodies proximity operations", *70th International Astronautical Congress*, Washington, USA, 21-25 October 2019
- S. Silvestrini, M. Lavagna, "Model-based Reinforcement Learning for Distributed Path Planning", *15th Symposium on Advanced Space Technologies in Robotics and Automation - ASTRA*, ESA-ESTEC, 27-28 May 2019.
- S. Silvestrini, V. Pesce, M. Lavagna, "Distributed Autonomous Guidance, Navigation and Control loop for Formation Flying Spacecraft Reconfiguration", *5th CEAS Conference on Guidance, Navigation & Control*, 3-5 April 2019

CHAPTER 2

Background & State of Art

La casa é come un punto di memoria
Le tue radici danno la saggezza
E proprio questa é forse la risposta
E provi un grande senso di dolcezza
E provi un grande senso di dolcezza.

— Francesco Guccini

The challenges of autonomy in robotic systems, and in particular in space elements, arise from the limited on-board knowledge of the environment in which the agent operates. Such shortcoming descends from several motivations, both practical and theoretical. On one hand, the representation we have of the reality, i. e. what we call *analytical models*, are inherently approximation of the true, unknown dynamics, which is influenced by unpredictable or unforeseen events. On the other hand, even if we had a complete knowledge of the dynamical behavior, it would be hardly manageable to be treated on-board, due to the limited computational power typically available on the spacecraft. Putting linear systems into a nonlinear environment requires a lot of effort and resources from the engineers in order to make it right. In addition, this extra

effort and resources are reflected in the design which becomes complex. One way to break the barrier of the current technological limitation could be to move into adaptive non-linear systems. Non-linear systems increases the capabilities of the technology and new attributes emerge from its implementation. As already mentioned, and hereby stressed, the focus is to enhance the attributes that serve the objectives of the mission and to depreciate the ones that create complications. The most powerful approach is to develop non-linear adaptive systems, which take advantage of a-priori knowledge of the environment, but are able of reacting to external inputs while flying. The Guidance, Navigation & Control system is thus equipped with the capability of adapting its response to the actual inputs it receives from the environment. The way it adapts depend on the level of *intelligence* that the spacecraft is designed to possess. Artificial Intelligence is a product of a non-linear adaptive system. The way it can be manifested depends on the technology level and the current knowledge. For instance, Artificial Intelligence techniques could allow the spacecraft to achieve highly accurate relative navigation when arriving to an unknown target or to generate control and planning action in uncertain environments. As explained in Chapter 1, the objective of this Thesis is to integrate traditional algorithm with AI techniques that refine the modeling and planning tasks according to the actual feedback received from the environment. This, as stated, is extremely promising and transverse in its applicability every time the mission faces partially known, or even completely unknown, scenarios.

The Chapter is organized as follows: Section 2.1 presents the analytical models for the two scenarios of proximity dynamics, namely Earth-bounded relative dynamics for distributed systems and proximity dynamics with small bodies orbiting the Sun. Section 2.2 presents the Guidance, Navigation & Control system, which is the basis on which the AI techniques have been developed in this work. Finally, Section 2.3 reports the main elements belonging to the general term of Artificial Intelligence that are employed, with proper modifications, in this Thesis.

2.1 Uncertain Relative Dynamics

The analytical models that express the dynamical behavior of the systems in different environments are hereby presented. In particular, as the derivation develops, the nonlinearities and neglected perturbations are highlighted. The reason for that is to stress the shortcomings of relying only on analytical models, which are inherently approximations of the actual dynamics due to unknown or unforeseen perturbations. In this Thesis, the relative and proximity dynamics is taken as reference. This yield to potential mission scenarios: the first comprises the relative dynamics of two spacecraft orbiting the same planet (e. g. Earth);

the second represents the environment around a small body, whose shape is typically only partially known.

2.1.1 Distributed Systems Relative Dynamics

The Thesis uses a high-fidelity propagator as *ground-truth* for algorithm validation. The accurate orbital simulator is used to test the algorithms in a realistic environment. In fact, the relative motion between target and chaser is obtained by integrating separately the chaser and the target orbital dynamics considering the perturbations acting on each spacecraft. In particular, the model considers irregularities in the gravitational potential due to non-spherical distribution of Earth's mass, the presence of the Moon and the Sun as third-body, the effect of the Solar Radiation Pressure (SRP) and the atmospheric drag. The adopted Earth gravitational model is the EGM96 with harmonics up to the third degree and order. On the other hand, the atmospheric drag force is computed by using the Jacchia Reference Atmosphere model. The relative dynamics equations here reported refers to those models that are used in the GNC-AI algorithms, for potential on-board applications. In this Thesis, two approaches have been alternatively adopted to model the relative dynamics. The first is referred to *Cartesian* derivation that parametrizes the system state into position and velocity. The other method refers to the *Relative Orbital Elements* parametrization, which uses a combination of the orbital parameters of the spacecraft orbits to describe the dynamical behaviors. They are simplified models suitable for low computational power spacecraft. In addition, linearized models allows the analysis and synthesis of the algorithms, e. g. control margins and effectiveness. The derivation of the *Cartesian* analytical model develops from the orbital equations of motion of each spacecraft expressed in the inertial frame, as schematized in Fig. 2.1.

2.1.1.1 Clohessy-Wilthsire Equations

The most common set of equations to describe the relative dynamics between two spacecrafts are the well-known Clohessy-Wiltshire equations, which are presented hereby. For a full derivation, the authors suggest to refer to [2]. With reference to Figure 2.2, the target spacecraft is in a nominal orbit at a distance $\mathbf{r_0}$ from the attractor. If the chaser is in close proximity of the target, the orbital radius can be expressed as $\mathbf{r} = \mathbf{r_0} + \delta\mathbf{r}$. The equation of motion of the chaser spacecraft is:

$$\ddot{\mathbf{r}} = -\mu \frac{\mathbf{r}}{r^3} \tag{2.1}$$

where μ is the gravitational constant of the central body. Let us attach a co-moving frame of reference centered on the target spacecraft center of mass, as shown in Figure 2.2. The x axis is aligned with $\mathbf{r_0}$, the z is orthogonal to the orbital plane along the positive angular momentum vector and the y axis

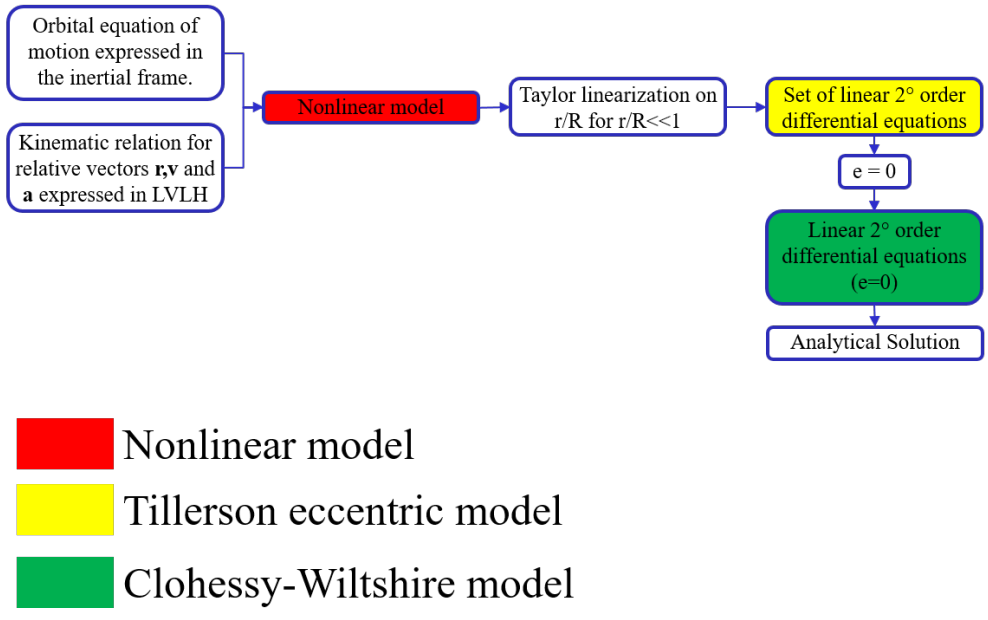


Figure 2.1: Derivation of *cartesian* models of relative dynamics.

completing the right-hand triad. Such reference frame is commonly known as Local-Vertical-Local-Horizontal (LVLH). Substituting the definition of **r** to Eq. 2.1 and expressing everything in the LVLH, the Clohessy-Wiltshire equations are obtained for a circular reference target orbit:

$$\begin{aligned} \delta\ddot{x} - 3n^2\delta x - 2n\delta\dot{y} &= 0 \\ \delta\ddot{y} + 2n\delta\dot{x} &= 0 \\ \delta\ddot{z} + n^2\delta z &= 0 \end{aligned} \tag{2.2}$$

where $n = \frac{2\pi}{T}$ is the orbital mean motion and T is the reference orbital period.

2.1.1.2 Nonlinear Dynamical Model of J_2-Perturbed Relative Motion

The nonlinear dynamical model for J_2-perturbed relative orbit is described in [3]. Hereby, the fundamental equations are solely reported; for a thorough derivation, refer to [3]. With reference to Figure 2.2, the dynamics of the spacecraft can be written as:

$$\begin{aligned} \delta\ddot{x} =& 2\omega_z\delta\dot{y} - (n_j^2 - \omega_z^2)\delta x + \alpha_z\delta y - \omega_x\omega_z\delta z - (\zeta_j - \zeta)s_i s_\theta + \\ & - r(n_j^2 - n^2) + a_x \\ \delta\ddot{y} =& -2\omega_z\delta\dot{x} + 2\omega_x\delta z - \alpha_z\delta x - (n_j^2 - \omega_z^2 - \omega_x^2)\delta y + \alpha_x\delta z + \\ & - (\zeta_j - \zeta)s_i c_\theta + a_y \\ \delta\ddot{z} =& -2\omega_x\delta\dot{y} - \omega_x\omega_z\delta x - \alpha_x\delta y - (n_j^2 - \omega_x^2)\delta z - (\zeta_j - \zeta)c_i + \alpha_z \end{aligned} \tag{2.3}$$

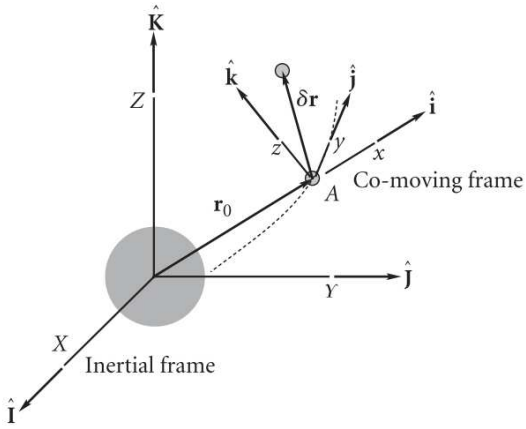


Figure 2.2: Co-moving Local-Vertical-Local-Horizontal (LVLH) frame [2].

where the contributing terms are:

$$
\begin{aligned}
n^2 = \frac{\mu}{r_0^3} + \frac{k_{J2}}{r_0^5} - \frac{5k_{J2}s_i^2 s_\theta^2}{r^5}, \qquad n_j^2 = \frac{\mu}{r^3} + \frac{k_{J2}}{r^5} - \frac{5k_{J2}r_{JZ}^2}{r^7} \\
r_{JZ} = (r_0 + \delta x)s_i s_\theta + \delta y s_i c_\theta + \delta z c_i, \qquad k_{J2} = \frac{3J_2\mu R_e^2}{2} \\
\omega_x = -\frac{k_{J2}s_{2i}s_\theta}{hr_0^3}, \qquad \omega_z = \frac{h}{r_0^2} \\
\alpha_x = \dot{\omega}_x = \frac{k_{J2}s_{2i}c_\theta}{r_0^5} + \frac{3\dot{r_0}k_{J2}s_{2i}s\theta}{r_0^4 h} - \frac{8k_{J2}^2 s_i^3 c_i s_\theta^2 c_\theta}{r^6 h^2} \\
\alpha_z = \dot{\omega}_z = -\frac{2h\dot{r_0}}{r_0^3} - \frac{k_{J2}s_i^2 s_{2\theta}}{r_0^5}, \qquad \zeta = \frac{2k_{J2}s_i s_\theta}{r^4}, \qquad \zeta_j = \frac{2k_{J2}r_{JZ}}{r^5}
\end{aligned}
\tag{2.4}
$$

in which h is the orbital angular momentum, i orbital inclination, J_2 is the zonal harmonic coefficient $1.0826 \cdot 10^{-3}$ for Earth, R_e is the Earth radius, θ is the orbital true anomaly and $\mathbf{a} = [a_x, a_y, a_z]^T$ is the forced acceleration vector. Last, s_x and c_x stand for $\sin(x)$ and $\cos(x)$, where x is a generic angle. The spacecraft relative motion is actually described by 11 first-order differential equations, namely $(\delta x, \delta y, \delta z, \delta\dot{x}, \delta\dot{y}, \delta\dot{z})$ and $(r_0, \dot{r_0}, h, i, \theta)$. Nevertheless in this work, the latter quantities are computed using the high-fidelity propagator previously described. This can be representative of an on-board absolute state estimator. Alternatively, these quantities can be included in the integration step of the dynamical model.

2.1.2 Relative Orbital Elements Parametrization

The spacecraft formation dynamics is described using the relative orbital elements, following the work done by D'Amico [4] The following quasi-singular

relative orbital elements are adopted:

$$\delta\chi = \begin{pmatrix} \delta a \\ \delta\lambda \\ \delta e_x \\ \delta e_y \\ \delta i_x \\ \delta i_y \end{pmatrix} = \begin{pmatrix} \frac{a_f - a_r}{a_r} \\ (M_f + \omega_f) - (M_r + \omega_r) + (\Omega_f - \Omega_r)\cos(i_r) \\ e_f\cos(\omega_f) - e_r\cos(\omega_r) \\ e_f\sin(\omega_f) - e_r\sin(\omega_r) \\ i_f - i_r \\ (\Omega_f - \Omega_r)\sin(i_r) \end{pmatrix} \tag{2.5}$$

where the subscript f stands for any follower spacecraft orbit, whereas the subscript indicates the reference orbital elements. M is the mean anomaly, a the semimajor axis, e the eccentricity, i the orbit inclination, ω the argument of perigee and Ω the right ascension of the ascending node. It is important to remark that in this work the reference orbit is the same for the n spacecraft building up the formation. The benefit of using such model is that, if the perturbations are neglected, the geometry of the relative motion with respect to a reference orbit is uniquely determined by a set of invariant relative orbital elements (ROE), except for the relative true anomaly, which follows the Keplerian propagation. Indeed, the natural evolution of the dynamic system can be described as:

$$\dot{\delta\chi} = \mathbf{A_k} \cdot \delta\chi \tag{2.6}$$

where

$$\mathbf{A_k} = \begin{bmatrix} 0 & \vdots & \\ -1.5n & \vdots & \mathbf{0}_{6\times5} \\ \mathbf{0}_{4\times1} & \vdots & \end{bmatrix} \tag{2.7}$$

Guffanti and Koenig [5][6] later expanded the model to a J_2 perturbed dynamics. The complete dynamical model can be expressed as:

$$\dot{\delta\chi} = (\mathbf{A_k} + \mathbf{A_{J2}}) \cdot \delta\chi + B\mathbf{u} \tag{2.8}$$

$$\mathbf{A_{J_2}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{7}{2}(1+\eta)(3\cos^2 i_r - 1) & 0 & e_xGFP & e_yGFP & -FS & 0 \\ \frac{7}{2}e_yQ & 0 & -4e_xe_yGQ & -(1+4Ge_y^2)Q & 5e_yS & 0 \\ -\frac{7}{2}e_xQ & 0 & (1+4Ge_x^2)Q & 4e_xe_yGQ & -5e_xS & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{7}{2}S & 0 & -4e_xGS & -4e_yGS & 2T & 0 \end{bmatrix} \tag{2.9}$$

where the terms in Eq. 2.9 are:

$$k = \gamma a_r^{-\frac{7}{2}} \eta^{-4},\ \eta = \sqrt{1-e_r^2},\ \gamma = \frac{3}{4} J_2 R_e^2 \sqrt{\mu},\ e_x = e_r \cos\omega_r,$$
$$e_y = e_r \sin\omega_r\ E = 1+\eta,\ G = \frac{1}{\eta^2},\ F = 4+3\eta,\ P = 3\cos^2 i_r - 1, \tag{2.10}$$
$$Q = 5\cos^2 i_r - 1,\ S = \sin 2i_r,\ T = \sin^2 i_r$$

where $J2$ is the zonal harmonic coefficient $1.0826 \cdot 10^{-3}$ for Earth, R_e is the Earth radius, $\mu = 3.986 \cdot 10^{14} m^3 s^{-2}$ is the Earth gravitational constant. The control matrix is derived from Gauss Variational Equation (GVE) as in [7]:

$$\mathbf{B} = \frac{1}{a_r n_r} \begin{bmatrix} \frac{2}{\eta} e_r \sin\nu_r & \frac{2}{\eta}(1+e_r\cos\nu_r) & 0 \\ -\frac{2\eta^2}{1+e_r\cos\nu_r} & 0 & 0 \\ \eta_r \sin\omega_r + \nu_r & \eta\frac{(2+e_r\cos\nu_r)\cos(\omega_r+\nu_r)+e_x}{1+e_r\cos\nu_r} & \frac{\eta e_y}{\tan i_r}\frac{\sin(\omega_r+\nu_r)}{1+e_r\cos\nu_r} \\ -\eta_r \cos\omega_r + \nu_r & \eta\frac{(2+e_r\cos\nu_r)\sin(\omega_r+\nu_r)+e_y}{1+e_r\cos\nu_r} & \frac{-\eta e_x}{\tan i_r}\frac{\sin(\omega_r+\nu_r)}{1+e_r\cos\nu_r} \\ 0 & 0 & \eta\frac{\cos(\omega_r+\nu_r)}{1+e_r\cos\nu_r} \\ 0 & 0 & \eta\frac{\sin(\omega_r+\nu_r)}{1+e_r\cos\nu_r} \end{bmatrix} \tag{2.11}$$

where ν_r is the true anomaly.

2.1.2.1 Coordinates Transformation

The active collision avoidance maneuvers depend on the relative metric distance between two agents. The relative distance is naturally expressed in the Cartesian Local-Vertical-Local-Horizontal (LVLH) reference frame. The mapping between the Hill $\mathbf{X} = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}]$ state to the ROE $\delta\chi$ is required to process the measurements and compute the guidance and control output. The transformation matrices are derived by using the classical orbital elements difference $\Delta OE = [\Delta a\ \Delta M\ \Delta\omega\ \Delta e\ \Delta i\ \Delta\Omega]$ as follows:

$$\mathbf{J}_{\delta\chi}^{\mathbf{X}} = \frac{\partial \mathbf{X}}{\partial \Delta OE} \cdot \frac{\partial \Delta OE}{\partial \delta\chi}, \qquad \mathbf{J}_{\mathbf{X}}^{\delta\chi} = \frac{\partial \delta\chi}{\partial \Delta OE} \cdot \frac{\partial \Delta OE}{\partial \mathbf{X}} \tag{2.12}$$

where a is the semimajor axis, M is the mean anomaly, ω the argument of perigee, e the eccentricity, i the inclination and Ω the right ascension of the ascending node. The first-order approximation of the mapping between the Hill state and classical osculating orbital elements yields [4][8]:

$$\begin{aligned} x &= \frac{r}{a}\Delta a - a\cdot\cos\nu\Delta e + \frac{ae\sin\nu}{\sqrt{1-e^2}}\Delta M \\ y &= \left(a + \frac{r}{1-e^2}\right)\sin\nu\Delta e + \frac{a^2}{r}\eta\Delta M + r\Delta\omega + r\cos i\Delta\Omega \\ z &= r\sin(\nu+\omega)\Delta i - r\sin i\cos(\nu+\omega)\Delta\Omega \end{aligned} \tag{2.13}$$

Differentiating Eq. 2.13 the full transformation is obtained:

$$\begin{aligned}
\dot{x} =& -\frac{ne\sin\nu}{2\sqrt{1-e^2}}\Delta a + n\sin\nu\sqrt{1-e^2}\left(\frac{a^3}{r^2}\right)\Delta e + en\cos\nu\frac{a^3}{r^2}\Delta M \\
\dot{y} =& \left[n\sqrt{1-e^2}\left(1+\frac{r}{a(1-e^2)}\right)\left(\frac{a^3}{r^2}\right)\cos\nu + \frac{aen\sin^2\nu}{(1-e^2)^{\frac{3}{2}}}\right]\Delta e + \\
& - en\sin\nu\frac{a^3}{r^2}\Delta M + \frac{aen\sin\nu}{\sqrt{1-e^2}}\Delta\omega \\
\dot{z} =& \frac{an}{\sqrt{1-e^2}}\left(\sin i\left[\sin(\nu+\omega)+e\sin\omega\right]\Delta\Omega + \right. \\
& \left. + \left[\cos(\nu+\omega)+e\cos\omega\right]\Delta i\right)
\end{aligned} \tag{2.14}$$

Combining Eq. 2.13 and 2.14 the transformation matrix between Hill state $\mathbf{X}$ and classical orbital elements ΔOE, namely $\frac{\partial \mathbf{X}}{\partial \Delta OE}$ and its inverse in Eq. 2.12. To formulate the complete transformation the Jacobian of the transformation between classical orbital elements and relative orbital elements $\delta\chi$ is required. Such transformation is obtained from the definition of $\delta\chi$ for $\Delta OE \to 0$:

$$\begin{aligned}
\frac{\partial \Delta OE}{\partial \delta\chi} &= \begin{bmatrix}
a & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & \frac{\sin\omega}{e} & -\frac{\cos\omega}{e} & 0 & \frac{\cos i}{\sin i} \\
0 & 0 & -\frac{\sin\omega}{e} & \frac{\cos\omega}{e} & 0 & 0 \\
0 & 0 & \cos\omega & \sin\omega & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & \sin i
\end{bmatrix} \\
\frac{\partial \delta\chi}{\partial \Delta OE} &= \begin{bmatrix}
\frac{1}{a} & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & \cos i \\
0 & 0 & -e\sin\omega & \cos\omega & 0 & 0 \\
0 & 0 & e\cos\omega & \sin\omega & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & \sin i
\end{bmatrix}
\end{aligned} \tag{2.15}$$

The just described model is exploited to perform the design of the relative trajectories. However some concerns may arise with respect to the accuracy of such model, reason for which a set of propagation tests have been performed, in order to assess the level of confidence to pose into the ROE formulation. The dynamics has indeed been compared to different Cartesian formulations of the relative dynamics and to a high-fidelity orbital propagator[9, 10]. The procedure for the validation of the ROE formulation with the high-fidelity propagator is presented in Fig. 2.3. The selection of the initial conditions

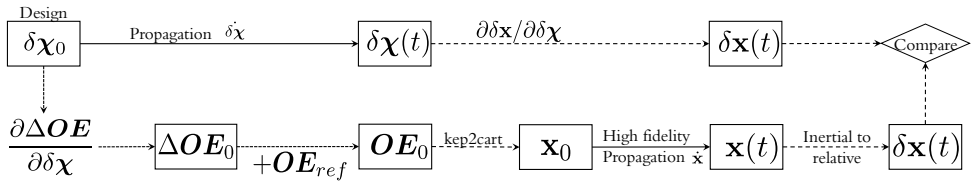


Figure 2.3: Schematics of the ROE formulation dynamical comparison with respect to an high-fidelity propagation of the trajectory.

for the propagation of the Cartesian state is fundamental. An incoherence among the different methods can arise for what concerns the different level of linearization required by the translation of the ROE into the Cartesian state. Indeed, different state propagations arise if such conversion is performed through the nonlinear chain of transformations described in the bottom branch Fig. 2.3 or through the direct linear transformation, as in the last conversion of the upper branch of the graph. The former starts from the ROE and, passing through the difference in orbital elements $\mathbf{\Delta OE}$ and the reference orbital elements $\mathbf{OE}_{ref}$, recovers the inertial state which can be remapped to the relative Cartesian state in the LVLH frame. The latter uses instead a direct transformation from the initial ROE to the relative Cartesian frame, implying a higher degree of linearization. Details on the two mapping can be found in previous works[11]. Figure 2.4 presents the position error δ_r accuracy obtained by a nonlinear Cartesian relative dynamics model, when initialized with the linear transformation (left plot) and the nonlinear one (right plot). The propagation of the ROE with the given formulation is also presented for comparison. The position errors δ_r are computed with respect to the trajectory propagated with the high-fidelity orbital simulator, over a time span of a complete day. It is easy to appreciate the much higher drift rate of the nonlinear Cartesian model exploiting the linearized initial condition, with respect to the evolution using the nonlinear transformation. On the contrary, as highlighted also in other works[12], when dealing with a linearized dynamics, such as the Clohessy-Wiltshire equations, in order to keep the consistency in terms of nonlinear effects, the linearized transformation ensures a better representation of the system. Nevertheless this analysis gave also the hint on the feasibility of exploiting the ROE with J_2 perturbations for the design of the relative trajectories, without introducing huge inaccuracies, as seen by the plots in Fig.2.4, where a maximum of $\sim$2 km error is introduced over a 1 day propagation arc.

2.1.3 Bounded Relative Orbits

This Thesis is focused on the presentation of a GNC architecture for spacecraft formation flying. For this reason, the natural motion is not deeply analyzed.

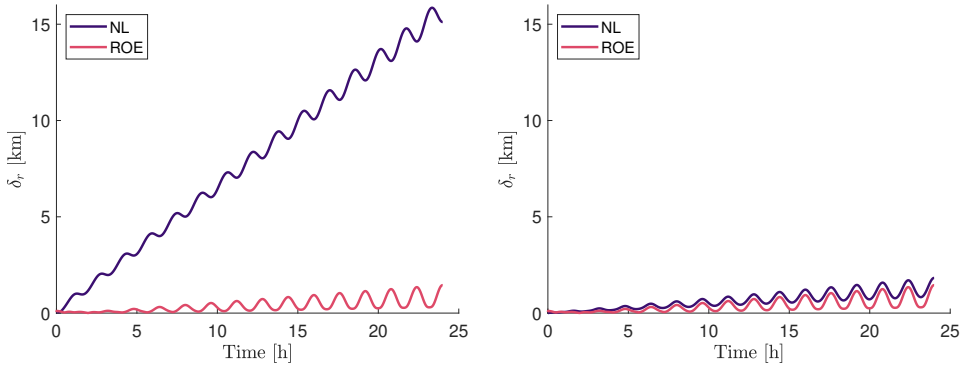


Figure 2.4: Comparison of the position error of the nonlinear Cartesian relative dynamics and ROE formulation with respect to the high-fidelity propagated trajectory. The initial conditions for the nonlinear model are generated using the linear transformation (left) or the nonlinear one (right).

Nevertheless, realistic formation are taken into account to generate initial and final configurations. In order to reduce the fuel consumption for formation-keeping actions, the configurations are chosen among the bounded relative orbits group. To avoid the relative drift, it is critical that the relative motion of the spacecrafts remains bounded. A fundamental concept in spacecraft Formation Flying is the orbital energy-matching method to generate bounded formations. The orbital energy of the satellites is a function of the semi-major axis only:

$$\mathcal{E} = -\frac{\mu}{2a} \tag{2.16}$$

Hence, it is sufficient to match the orbital energy of the reference orbit to generate bounded formations. In order to work out relevant initial conditions, being either in the Cartesian space or $\delta\chi$, we refer to the $\delta\chi$ relative space, cfr. Eq. 2.5. It is sufficient that $\delta a = 0$ for the relative orbital elements defining the formation. Fig. 2.5 shows trajectories propagation based on energy-matching initial conditions in perturbed models.

2.1.4 Small Bodies Relative Dynamics

Some assumptions are made both on the dynamical environment as well as the output of the reconstruction in order to make addressable the problem of the reconstruction of the gravitational field of an unknown, arbitrary shaped body directly on-board of a spacecraft flying around it.

In this work two different dynamical environments are modeled: the one associated with a single body, based on the so called Perturbed Two-Body Problem (P2BP), as well as the one associated to a binary system of bodies,

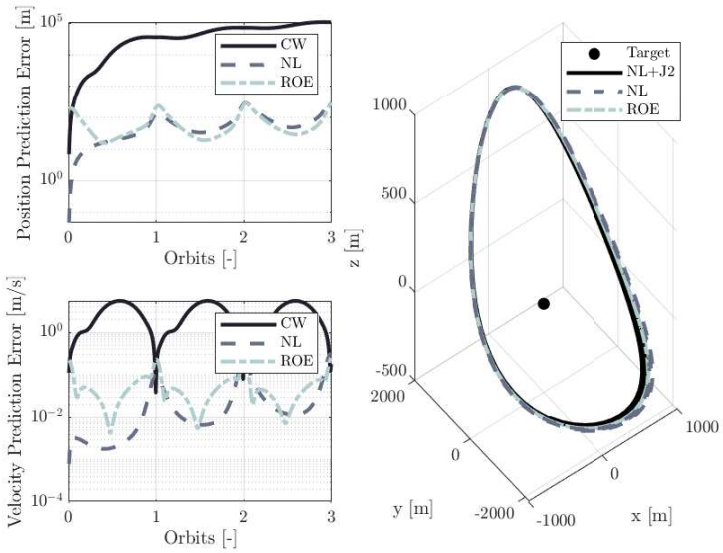


Figure 2.5: Relative bounded orbits in perturbed models

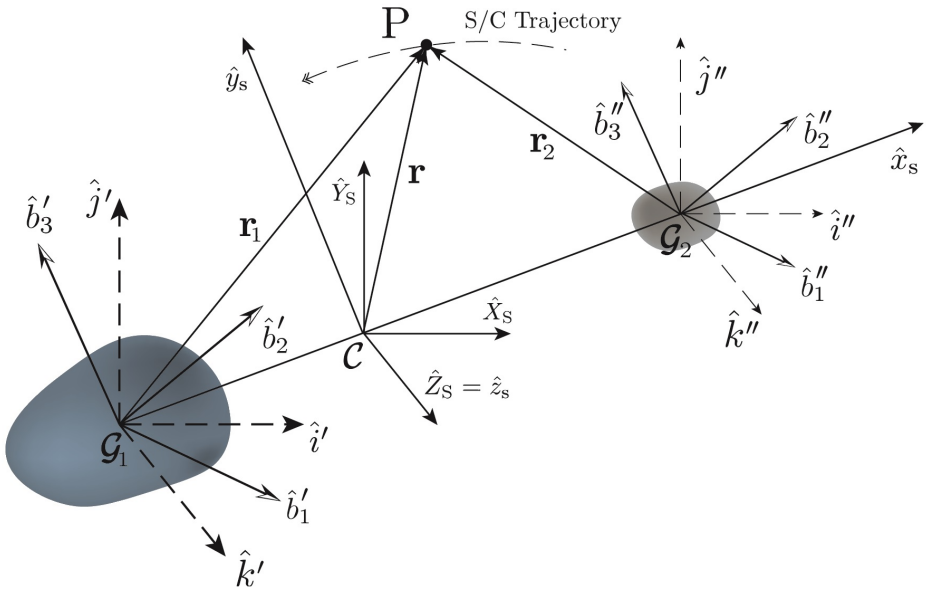


Figure 2.6: Geometry of the MCR3BP

based on the Modified Circular Restricted Three-Body Problem (MCR3BP). This, in order to test the scalability of the network to different dynamical

environments, both from the formulation as well as from the identification performance point of view.

The Perturbed Two-Body Problem The detailed derivation of the dynamical environment model associated to a single body relays on the P2BP model, that is extensively discussed in [13]. Here the equation of motion for a reduced order model are briefly recalled. Under the assumption that the body rotates about its principal inertia axis with uniform angular velocity Ω, the equation of motion written in the body-fixed frame results:

$$\begin{cases} \ddot{x} - 2\Omega\dot{y} = \Omega^2 x + a_{T,x} \\ \ddot{y} + 2\Omega\dot{x} = \Omega^2 y + a_{T,y} \\ \ddot{z} = a_{T,z} \end{cases} \tag{2.17}$$

wherein the acceleration model adopted can be expressed as:

$$\mathbf{a}_{\mathrm{T}}(\mathbf{r}, \mathbf{s}, \mathbf{d}_{\mathrm{k-a}}) = \mathbf{a}_{\mathrm{G}}(\mathbf{r}) \; + \; \mathbf{a}_{\mathrm{SRP}}(\mathbf{r}, \mathbf{s}) \; + \; \sum_{k=1}^{N} \mathbf{a}_{3\mathrm{rd_k}}(\mathbf{r}, \mathbf{d}_{\mathrm{k-a}}) \tag{2.18}$$

being $\mathbf{a}_{\mathrm{G}}$ the gravitational acceleration due to the gravity field of the body, $\mathbf{a}_{\mathrm{SRP}}$ the acceleration contribution due to the Solar Radiation Pressure (SRP) and $\mathbf{a}_{3\mathrm{rd_k}}$ the acceleration contribution due to the k-th third-body. In this work, since the aim is to focus the attention on the gravity field reconstruction, the other perturbations are neglected and the gravitational model used as the reference one is the constant density polyhedron [14]. This model is a subclass of the P2BP, called in this work Shape-Based Two-Body Problem (S2BP).

The Modified Circular Restricted Three-Body Problem The geometry and the formulation of the MCR3BP starts from the CR3BP. The only difference is that the two bodies are assumed to have a certain shape and not to be point masses. Having defined the angular velocity associated to the two-body motion of the primaries as:

$$\Omega_S = \sqrt{\frac{G(m_1 + m_2)}{d_{12}^2}} \tag{2.19}$$

where m_1 and m_2 are the mass of the primary and the secondary, d_{12} the distance between them and G the gravitational constant, and defining $\mathcal{T}_S$ as an inertial frame, fixed at the center of mass of the two primaries, and $\mathcal{T}_s$ a synodic frame, the transformation matrix of a generic vector $\mathbf{R}$ expressed in the $\mathcal{T}_S$ frame to a vector $\mathbf{r}$ in the synodic frame is given by:

$$\mathrm{T}_{\Omega_S}(t) = \begin{bmatrix} \cos(\Omega_S t) & \sin(\Omega_S t) & 0 \\ -\sin(\Omega_S t) & \cos(\Omega_S t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.20}$$

In a similar way, the transformation matrix from the k-th body fixed frame (with $k = 1, 2$) to the k-th inertial frame $\mathcal{T}_n^k$, where $\mathcal{T}_n^k$ is centered in the k-th body and parallel to the $\mathcal{T}_S$ frame, is given by:

$$\mathrm{T}_n^k(t) = \begin{bmatrix} \cos(\Omega_k t) & \sin(\Omega_k t) & 0 \\ -\sin(\Omega_k t) & \cos(\Omega_k t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.21}$$

Then the position vector $\mathbf{r}$, in the $\mathcal{T}_s$ frame can be expressed in the k-th body fixed frame according to:

$$\mathbf{r}^{(k)} = \mathrm{T}_n^k \cdot \mathrm{T}_{\Omega_S}^T(\mathbf{r} - \mathbf{l}_k) \tag{2.22}$$

where here $\mathbf{l}_k$ is the distance of the primary to the center of mass of the system in the $\mathcal{T}_s$ reference. Note that the product $\mathrm{T}_n^k \cdot \mathrm{T}_{\Omega_S}^T$ can be re-arranged, having defined the differential rotation as $\Delta\Omega_k = \Omega_S - \Omega_k$, since:

$$\mathrm{T}_{\Omega_S} \cdot \mathrm{T}_n^{kT} = \begin{bmatrix} \cos(\Delta\Omega_k t) & \sin(\Delta\Omega_k t) & 0 \\ -\sin(\Delta\Omega_k t) & \cos(\Delta\Omega_k t) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathrm{T}_\Delta^k(t) \tag{2.23}$$

The equation of motion in the $\mathcal{T}_s$ frame results, according to [13]:

$$\begin{aligned} \ddot{\mathbf{r}} + \boldsymbol{\Omega}_S \times (\boldsymbol{\Omega}_S \times \mathbf{r}) + 2\boldsymbol{\Omega}_S \times \mathbf{r} = \\ \mathrm{T}_\Delta^1(t)\nabla\mathcal{U}_1\left(\mathbf{r}^{(1)}\right) + \mathrm{T}_\Delta^2(t)\nabla\mathcal{U}_2\left(\mathbf{r}^{(2)}\right) \end{aligned} \tag{2.24}$$

This work implement a simpler version of Eq.2.24. In particular the bodies are assumed to be locked with the respect to the synodic frame resulting in a Shape-Based CR3BP (SCR3BP):

$$\ddot{\mathbf{r}} + \boldsymbol{\Omega}_S \times (\boldsymbol{\Omega}_S \times \mathbf{r}) + 2\boldsymbol{\Omega}_S \times \mathbf{r} = \nabla\mathcal{U}_1(\mathbf{r}_1) + \nabla\mathcal{U}_2(\mathbf{r}_2) \tag{2.25}$$

where here $\mathcal{U}_i(\cdot)$ is the gravitational potential associated to the i-th body.

2.2 Guidance, Navigation & Control Subsystem

The Guidance, Navigation & Control subsystem is in charge of determining the state, planning the steps and executing the actions in order to complete the mission task. The GNC subsystem make use of a complex set of algorithms, which translates the perception (sensors measurements) to actions (actuation). Here, a brief summary of the main characteristics and state of art of the GNC systems for relative and proximity operations is reported. The work developed in the Thesis takes into account the opportunities and limitations described here. The elementary interfaces in a traditional GNC scheme are shown in Fig. 2.7.

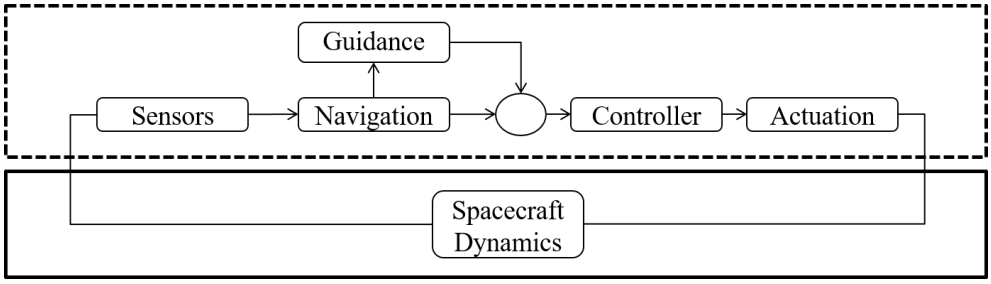


Figure 2.7: Basic interfaces in a traditional GNC subsystem scheme.

2.2.1 Relative Navigation

The relative navigation task is to reconstruct the state of the spacecraft relative to the target, being another spacecraft or the soil of a small-body. The navigation algorithms processes the measurements coming from the navigation sensors, reported in Table 2.1. Typically, the measurements acquisition is coupled with a navigation filter. The navigation filter is in charge of refining the estimate coming from the measurements, either by processing batches of data or including knowledge of the dynamics. The most common technique for measurements filtering is the well-known Kalman Filter. It is a powerful tool for combining information in presence of uncertainty. It produces guesses about where the system is going to be next and couples this with the information coming from measurements. The Kalman filter entails a statistical technique that describes the random structure of experimental measurements. In addition, it provides information about the quality of the estimation (i.e. variance of the estimation error). Finally, a feature of great interest for this Thesis, the Kalman filter has a recursive structure that allows real-time execution (no storing of observations or past estimates). A rigorous derivation of the Kalman filter is outside of the scope, but it will be thoroughly reviewed and adapted throughout the Thesis to favor the integration with AI-based techniques. For a complete overview of the Kalman filter, the author suggests to refer to [15, 16].

Spacecraft absolute and relative navigation are key tasks in the Guidance Navigation & Control (GNC) chain for current and future missions. Current navigation algorithms rely on the accurate knowledge of the system dynamics. This is possible whenever spacecrafts orbiting the Earth are considered, where the environment can be accurately modeled to a great extent of accuracy. Nevertheless, when dealing with relative approach with unknown bodies or interplanetary missions, the modeling of the system dynamics yields inevitable unmodeled uncertainties. This is mainly due to partial knowledge of the operative scenario, e.g. orbital disturbances acting on the target spacecraft. Furthermore, the growing interest towards micro-platforms, both for Earth and interplanetary missions, has significantly reduced the spacecraft available

computational power; hence, very sophisticated models cannot be anymore handled on-board. Such limitation leads to a degradation of performance of the GNC subsystem [17]. In this framework, on one hand, the dynamical model employed in the on-board algorithms needs to be simplified, on the other hand, the accuracy of such model significantly deteriorates the GNC performance [17], due to the absence of nonlinear terms as well as disturbances. The Artificial Neural Networks (ANNs) are a powerful tool to bridge this gap. ANNs are becoming increasingly important when dealing with uncertain processes. In particular, their capability of approximating unknown functions can be employed to reconstruct system nonlinearities, as well as unmodeled environmental disturbances. The advantage of estimating such uncertainties benefits the whole GNC process chain. In this framework, Gurfil et al. [18] presented a nonlinear adaptive neural control method applicable to deep space formation flying. Bae and Kim [19] developed a neural network aided sliding mode control scheme for spacecraft formation flying. Recently, Zhou [20] proposed a neural-network based reconfiguration control for spacecraft formation in obstacle environments. Traditionally, the ANNs are solely employed for disturbance estimation, yet the aim of the Navigation filter is to estimate the system state. In past years, there have been attempts to couple ANNs with Extended Kalman Filters (EKF). In particular, the most common approach is to employ EKFs to train the ANNs [21]. In this configuration, the state of the Kalman filter is augmented with the ANN weights. For a large network this process increases the computational burden. Furthermore, the resulting coupled structure cannot provide an estimate of the uncertainties, unless the disturbance vector is added to the state vector and estimated as a constant parameter. An alternative solution is to use the estimated disturbance term, output of the ANN, directly in the dynamical propagation of the filter [22]. In this way, instead of the state vector, the dynamics of the EKF is augmented by an ANN that captures the unmodeled dynamics. The ANN learns online the function describing the disturbance, i.e. the mismatch between the measurement and the a-priori guess given by the model selected for the EKF. However, in this case, the *augmented* dynamical model accuracy changes in time and therefore, its covariance matrix has to be adapted at each step to capture this variation. In the past years, few solutions have been proposed to derive an efficient formulation for neural network aided filters. Gao et al. [23] derived a Radial Basis Function Neural Network (RBFNN) - Kalman Filter to improve the estimation accuracy for seam tracking during high-power fiber laser welding. They proposed a coupled formulation where the RBFNN is used to compensate for the model and noise uncertainties. However, they do not consider any online adaptation of the filter covariance matrices. Similarly, Stubberud et al. [22] developed a neuro-observer based on an EKF and a multilayer feed-forward neural network. Their formulation involves two coupled Kalman Filters, one to estimate the state and the other to tune the neural

network. Other authors also proposed neural network for system identification based on offline training [24, 25]. Jwo and Huang [26] presented a neural network aided EKF for DGPS positioning. The neural network is used for noise identification to adaptively tune the EKF. However, their neural network relies on an offline training using the steepest descent technique. Recently Harl et al. [27] developed a reduced-order modified state observer for uncertainties estimation in nonlinear systems. They also applied the proposed technique to estimate the uncertain disturbances caused by J_2 perturbation around the Earth. Also in this case, the gain of the observer is user-selected and there is not any kind of adaptation depending on the experienced scenario.

The problem of reconstructing the unknown acceleration terms of the dynamics, as will be thoroughly detailed in the Thesis, is advantageous from a pure guidance and navigation perspective. However, it lacks insight for what concerns the direct knowledge of the perturbation sources (target shape or gravitational properties), being them blended to the other perturbative effects in the overall disturbance acceleration. In such sense, it is desirable to exploit a technique dedicated to the reconstruction of the small body shape and gravity field, to aid the navigation of the spacecraft, while enriching the science output of the mission. There have been a number of studies that proposes different applications of machine learning to this problem: in [28] a single layer forward network, designed and trained by means of extreme learning machines, is shown to be capable to learn the relationship between the spacecraft position and the gravitational acceleration. In [29] instead neural reinforcement learning is used to control a spacecraft around a celestial body whose gravity field is unknown. However, in those cases those methods have to be trained before use. This is possible if the target body shape is already available and so need detailed a-priori knowledge of the target body.

2.2.2 Relative Guidance & Control

The relative guidance task is to generate and plan trajectories and actions to fulfill mission objectives. The relative control task is to execute the actions, most often in feedback loop, in order to follow the guidance plan. It outputs the actuator required actions and instructions. Sometimes, especially in fully on-board implementation the two tasks merge in a single algorithm. For this reason they are here reported together. A lot of algorithms have been developed for relative control in uncertain environment, or more interestingly, in mission scenarios where multiple spacecrafts are involved. It is possible to summarize the adopted strategies for distributed systems found in literature as reported in Table 2.2.

The high-level of autonomy, and consequently increased complexity, is required in distributed mission concept, in which the satellites are expected to react

Table 2.1: Relative navigation sensors

Sensor	Measurements	Accuracy	Remarks
GPS-based	Relative position Absolute position	mm - cm	Earth mission Cooperative target required
RF-based	Range Range rate LOS	cm - m	Cooperative target (transceiver, antennas on target)
Radar	Range Range rate LOS	m	Uncooperative or cooperative targets
Vision-based	Range LOS Relative attitude	μm - mm	Cooperative (patterns, LEDs) or uncooperative targets (shape known and complex image processing) Illumination constraints Feature extracting algorithms required
Laser-based	Range LOS Relative attitude	mm	Retroflectors on target improves performance High mass and power Short operative range

Table 2.2: Guidance & Control strategies used for relative and proximity operations.

Guidance/Control Approach	**Pros**	**Cons**
Optimal Control	It generates optimal trajectories	No feedback control Hard to implement onboard
Artificial Potential Field	Simple to handle CA	It may require excessive ΔV It may lead to instability
STM inversion	Simple and analytic	Simplified model Hard to deal with CA
Impulsive control	Strong heritage	Modification for CA Correction for drift
Continuous Linear Control (e.g. LQR)	Linear control theory applicable	Continuous firing
Nonlinear Control (e.g. CLF)	Eliminate potential instability due to nonlinearities in the dynamics	Continuous firing
Model Predictive Control	Easy to generate optimal path with constraints	Computationally heavier

autonomously to unforeseen events. In particular, the collision avoidance task is critical in formation reconfiguration especially when the number of satellites increases. The GNC algorithms can be implemented following a centralized [30], decentralized or distributed architecture. The centralized architecture assumes there is a master spacecraft that processes the information coming from all the satellites, computes the guidance and control outcome and sends back commands to each spacecraft [31]. Decentralized GNC implements identical algorithms on-board each satellite, which is capable of computing its own action based solely on on-board information [32]. Finally, distributed systems relies on inter-satellite links: indeed, each satellite processes its own information and at least one coming from another agent of the system [33]. On one hand, the centralized architecture presents two different issues: first, it presents a single failure point due to the presence of a master spacecraft; furthermore, the GNC commands are sent to all the agents of the system, inserting complexity on the communication link between the master and the other spacecrafts. On the other hand, the decentralized approach solves the failure point aspect but lacks of a system perception, as each satellite is limited to its own data. The distributed architecture is selected to cope with the aforementioned shortcomings of the centralized and decentralized approach. Even though formation flying missions are not so numerous, in literature [31][33] the path-planning is seldom tackled as an optimum problem solved for the trajectory of each satellite taking into account the collision hazard constraint of the constructed trajectories. Both centralized and distributed architecture have been studied [33]. Nevertheless, the computational burden is high and difficult to handle, especially when on board computational resources are very limited. Mixed Linear Programming and Particle Swarm Optimization were proposed by Di Mauro et al. [31] to solve an optimal continuous control law for satellite reconfiguration. Such approach is hardly fitting the constraints imposed by the micro-platforms and does not take into account any collision avoidance strategy. A strategy for convexifying the collision constraint in optimal control has been reported by Chu [33]. Chernick et al. [34] presented an optimal control based on impulsive maneuver leveraging Keplerian dynamics to determine optimal, predictable maneuvering schemes, without taking into account the collision avoidance constraint. An impulsive strategy based on the state transition matrix of the system is also presented by Vadali and Alfriend [35]. Several authors have partially solved the task of collision-free path-planning using behavior-based algorithms. Izzo [36] presented a behavior-based algorithm, where the guidance desired velocity is determined as a result of the summation of identified behavior (target approach, collision avoidance, etc.). Similar to the behavior-based approach is the calculation of the Artificial Potential Field [37]. The major difference is that APF outputs desired acceleration, contrary to the behavior-based that works with guided velocities. Steindorf et al. [38] proposed a guidance and control approach based on the artificial potential field using relative orbital

elements. The authors include a passive collision avoidance strategy applicable to satellite formations composed of two satellites. The relative motion between spacecrafts flying in formation is typically reconstructed by state measurement expressed in Cartesian coordinates in the Hill frame. Hence, a linear mapping is developed to transform the Cartesian state to the relative orbital elements $\delta\chi$, and vice versa. Schaub [39] used a similar approach, where the mean orbital elements difference were used. The navigation, control and guidance algorithms are highly influenced by the dynamical models that are implemented on-board. For the sake of limiting computational burden, seldom these dynamical models are linearized or intentionally neglect disturbance terms, which might play a role in the motion evolution. Several researchers have utilized Artificial Neural Networks universal approximation theorem to work out the task of estimating disturbances and unmodeled terms [40], nevertheless most of the algorithms work off-line using large amount of data to be trained on. Being confident that the neural reconstruction of disturbances/unmodeled terms have proven the possibility to enhance the guidance and control for space mission [18][19][20], an online strategy to reconstruct the dynamics is still unexplored in literature. The literature is still poor with respect to algorithms that can be implemented in a distributed architecture with low computational power, actively managing the collision avoidance constraint between more than two satellites. Distributed space systems composed of several micro-satellites flying in formation are becoming increasingly attractive for the space community. This concept can significantly impact the capabilities of microsatellites mission (e.g. distributed Earth Observation (EO), distributed antenna, etc.), enhancing flexibility and reducing mission failure points. As the number of satellites increases an unprecedented level of autonomy is required to perform Guidance, Navigation and Control (GNC) tasks for formation maintenance and reconfiguration. Nevertheless, the size of the platforms tend to decrease when conceiving distributed systems due to different constraints, e.g. single launch of the formation. Such limitation implies the reduction of the on-board computational power. Hence, autonomous GNC algorithms need to be light and flexible to meet the challenges of distributed formation flying missions. Model-predictive control is a powerful control strategy that combines optimality and closed-loop control for trajectory generation and control actuation. Model predictive control has already been studied for distributed reconfiguration [41] [42], nevertheless such planning algorithms are dependent on the accuracy of the dynamical model. Moreover, they usually rely on linearized dynamics in order to carry out the optimization successfully.

2.3 AI-aided GNC

2.3.1 Machine Learning & Deep Learning

The research field on Machine Learning (ML) and Deep Learning (DL) is complex and extremely vast. In order to acquire a proper knowledge on the topic, the author suggests to refer to [43]. Hereby, only the most relevant concepts are reported in order to contextualize the work developed in the Thesis. The first important distinction is to be made between the terms Machine Learning and Deep Learning. The highlight of the two approaches are reported in colored box and in Fig. 2.8.

Machine Learning learns to map input to output given a certain world representation (features) hand-crafted for each task.

Deep learning is a particular kind of machine learning that aims at representing the world as a nested hierarchy of concepts, which are self-detected by the DL architecture itself.

The paradigm of ML and DL is to develop algorithms that are *data-driven*. The information to carry out the task are gathered and derived either from structured or unstructured data. In general, one would have a given experience $\mathcal{E}$, which can be easily thought as a set of data $\mathcal{D} = (x_1, x_2, \ldots, x_n)$. It is possible to divide the algorithms into three different approaches:

- Supervised learning: given the known outputs $\mathcal{T} = (t_1, t_2, \ldots, t_n)$, we learn to yield the correct output when new datasets are fed.
- Unsupervised Learning: the algorithms exploits regularities in the data to generate an alternative representation used for reasoning, predicting or clustering.
- Reinforcement Learning: producing actions $\mathcal{A} = (a_1, a_2, \ldots, a_n)$ that affect the environment and receiving rewards $\mathcal{R} = (r_1, r_2, \ldots, r_n)$. Reinforcement learning is all about learning what to do (i.e. mapping situations to actions) so as to maximize a numerical reward.

Even though the boundaries between the approaches are seldom blurred lines, the focus of this Thesis is to develop algorithms that take advantage and inspiration from supervised and reinforcement learning. For this reason, few additional details are provided for such approaches.

Supervised Learning Supervised learning consists in learning to associate some output to a given input, coherently with the set of examples of inputs $\mathbf{x}$

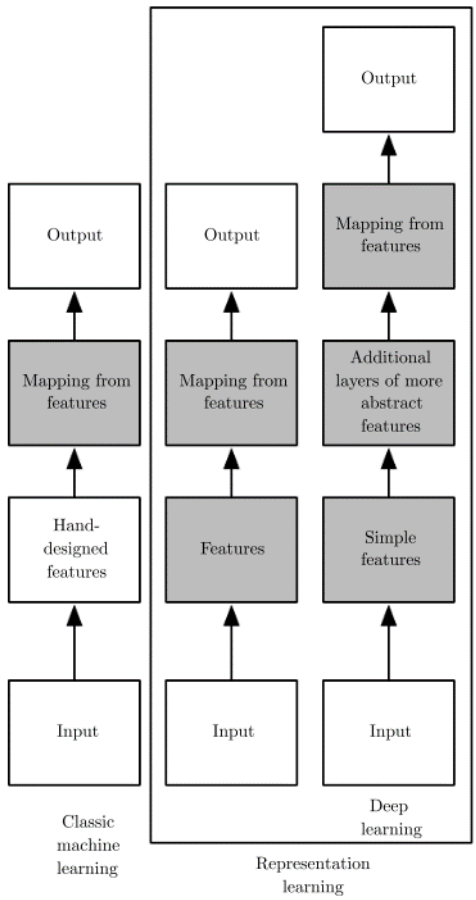


Figure 2.8: Differences between Machine Learning and Deep Learning [43].

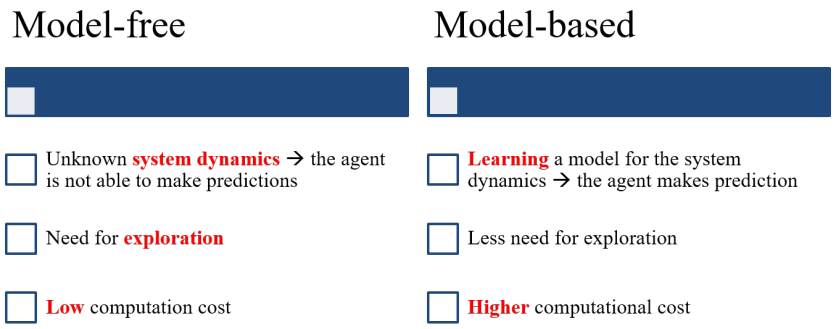


Figure 2.9: Differences between model-based and model-free reinforcement learning. In space we have deterministic representation of dynamical model: it is smart to exploit them. Nevertheless, some scenarios are unknown (e.g. Small bodies) or partially known (perturbations)

and targets $\mathbf{t}$. Quite often, the targets $\mathbf{t}$ are provided by a human supervisor. Nevertheless, supervised learning refers also to approaches in which target states are automatically retrieved by the machine learning model, as widely used in this Thesis. The typical applications of supervised learning are *classification* and *regression*. In few words, classification is the task of assigning a label to a set of input data among a finite class of labels. The output is a probability distribution of the likelihood of a certain input of belonging to a certain class. On the other hand, regression aims at modeling the relationship between a certain number of features and a continuous target variable. The regression task is largely employed in supervised learning for this Thesis.

Reinforcement Learning Reinforcement learning is learning what to do, how to map situations to actions, so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them. One of the challenges that arise in reinforcement learning, and not in other kinds of learning, is the trade-off between *exploration* and *exploitation*. The agent typically needs to explore the environment in order to learn a proper optimal policy, which determines the required action in a given perceived state. At the same time, the agent needs to exploit such information to actually carry out the task. In a space domain, and especially online applications, the balance must be shifted towards exploitation, for practical reasons. Another distinction that ought to be done is between *model-free* and *model-based* RL techniques, as shown in Fig. 2.9. Model-based methods rely on planning as their primary component, while model-free methods primarily rely on learning. Although there are real differences between these two kinds

of methods, there are also great similarities. We call environmental model whatever information the agent can use to make predictions on what will be the reaction of the environment to a certain action. For the reasons above, the model-based approach seems to be beneficial in the context of this Thesis, as it merges the advantage of analytical base models, learning and planning. It is important to report some of the key concepts of reinforcement learning:

- **Policy**: defines the learning agent way of behaving at a given time. Mapping from perceived states of the environment to actions to be taken when in those states.
- **Reward**: at each time step, the environment sends to the reinforcement learning agent a single number called the reward.
- **Value Function**: the total amount of reward an agent can expect to accumulate over the future, starting from that state.

2.3.2 Artificial Neural Networks

Artificial Neural Networks represent the nonlinear extension to the linear machine learning (or deep learning) models presented in Section 2.3.1. A thorough description of artificial neural networks is far beyond the scope of this work. Hereby, the set of concepts necessary to understand the work is reported. In particular, the universal approximation theory is described, which makes the foundation to all the algorithms developed in this Thesis. The most significant categorization of deep neural networks is between *feedforward* and *recurrent* networks. Deep feedforward networks, also often called simply multi-layer perceptrons (MLPs), are the most common deep learning models. The feedforward network is designed to approximate a given function f. According to the required task to execute, the input is mapped to an output value. For instance, for a classifier, the network $\mathcal{N}$ maps an input x to a category y. A feedforward network defines a mapping $y = \mathcal{N}(x, w)$ and learns the value of the parameters w (weights) that result in the best function approximation. These models are called feedforward because information flows from the input layer, through the intermediate ones, up to the output y. Feedback connections are not present in which outputs of the model are fed back as input to the networks itself. When feedforward neural networks are extended to include feedback connections, they are called recurrent neural networks.

The essence of Deep Learning, and Machine Learning extensively, is learning world structures from data. All the algorithms falling into the aforementioned term are *data-driven*. This means that, despite the possibility of exploiting analytical representation of the environment, the algorithms need to be fed with structures of data to perform the training. The learning process can be defined as the algorithm by which the free parameters of a neural network

are adapted through a process of stimulation by the environment in which it works. The type of learning is the set of instructions on how the parameters are changed, as explained in Section 2.3.1. Typically, the following sequence is followed:

1. the environment stimulates the neural network
2. the neural network makes changes of the free parameters
3. the neural network responds in a new way according to the new structure

As one would easily expect, there are several learning algorithms that can consequently be split into different types. To make a relevant categorization for this Thesis, it is possible to divide the supervised learning philosophy into *batch* and *incremental* learning [44]. The *batch* learning is suitable for spatial distribution of data in a stationary environment, meaning that there is no significant time correlation of data and the environment reproduces itself identically in time. Thus, for such applications, it is possible to gather the data into a whole batch that is presented to the learner simultaneously. Once the training has been successfully completed, the neural networks should be able to capture the underlying statistical behavior of the stationary environment. Such kind of statistical memory is used to make predictions exploiting the batch dataset that was presented. On the other hand, in several applications the environment is non-stationary, meaning that information signals coming from the environment may vary with time in its statistics. A *batch* learning would be inadequate as there are no means to track and adapt to the varying environment stimulus. Hence, it is favorable to employ what is called *incremental learning* (or *online* or *continuous*) in which the neural network constantly adapt its free parameters to the incoming information in a real-time fashion.

2.3.2.1 Universal Approximation Theorem

The universal approximation theorem takes the following classical form [44]. Let $\varphi : \mathbb{R} \to \mathbb{R}$ be a non-constant, bounded, and continuous function (called the activation function). Let I_m denote the m-dimensional unit hypercube $[0, 1]^m$. The space of real-valued continuous functions on I_m is denoted by $C(I_m)$. Then, given any $\varepsilon > 0$ and any function $f \in C(I_m)$, there exist an integer N, real constants $v_i, b_i \in \mathbb{R}$ and real vectors $w_i \in \mathbb{R}^m$ for $i = 1, \ldots, N$, such that we may define:

$$F(x) = \sum_{i=1}^{N} v_i \varphi \left(w_i^T x + b_i \right) F(x) = \sum_{i=1}^{N} v_i \varphi \left(w_i^T x + b_i \right) \tag{2.26}$$

as an approximate realization of the function f; that is,

$$|F(x) - f(x)| < \varepsilon \tag{2.27}$$

for all $x \in I_m$.

2.3.2.2 Back-propagation Algorithm

The basis for most of the supervised learning algorithms is represented by *back-propagation*. In general, finding the weights of an artificial neural network means determining the optimal set of variables that minimizes a given loss function. Given $\mathcal{N}$ structured data, comprising input x and target t, one can define the loss function at the output of neuron j for the p^{th} datum presented:

$$\epsilon_j(p) = t_j(p) - y_j(p) \tag{2.28}$$

where $y_j(p)$ is the output value of the j_{th} output neuron. It is possible to extend such definition to derive a mean indication of the loss function for the complete output layer. We can define a total energy error of the network for the p^{th} presented input-target pair:

$$\mathcal{E} = \frac{1}{2} \sum_{j \in \mathcal{C}_j} \epsilon_j^2(p) \tag{2.29}$$

where C_j is the set of output neurons of the network. As stated, the total energy error of the network represents the loss function to be minimized during training. Indeed, such function is dependent on all the free parameters of the network, synaptic weights and biases. In order to minimize the energy error function we need to find those weights that vanish the derivative of the function itself and minimizes the argument:

$$(\mathbf{w}, \mathbf{b})^T = argmin\ \mathcal{E}(\mathbf{w}, \mathbf{b}) \tag{2.30}$$

Closed-form solutions are practically never available, thus it is common practice to use iterative algorithms that make use of the derivative of the error function to converge to the optimal value. The back-propagation algorithm is basically a smart way to compute those derivatives, which can then be employed using traditional minimization algorithms such as *gradient descent*, *Newton and quasi-Newton methods*, *Levenberg-Marquardt*, *conjugate gradient*. Here we consider a simple method that can be applied specifically to sequential learning but easily extended to batch learning. With reference to Fig. 2.10 the induced local field of neuron j, which is the input of the activation function $\phi_j(\cdot)$ at neuron j can be expressed as:

$$v_j(p) = \sum_{i \in \mathcal{C}_i} w_{ji} y_i(p) + b_j \tag{2.31}$$

where $\mathcal{C}_i$ is the set of neurons that share a connection with layer j, b_j is the bias term of neuron j. The output of a neuron is the result of the application of the activation function to the local field v_j:

$$y_j(p) = \phi_j(v_j(p)) \tag{2.32}$$

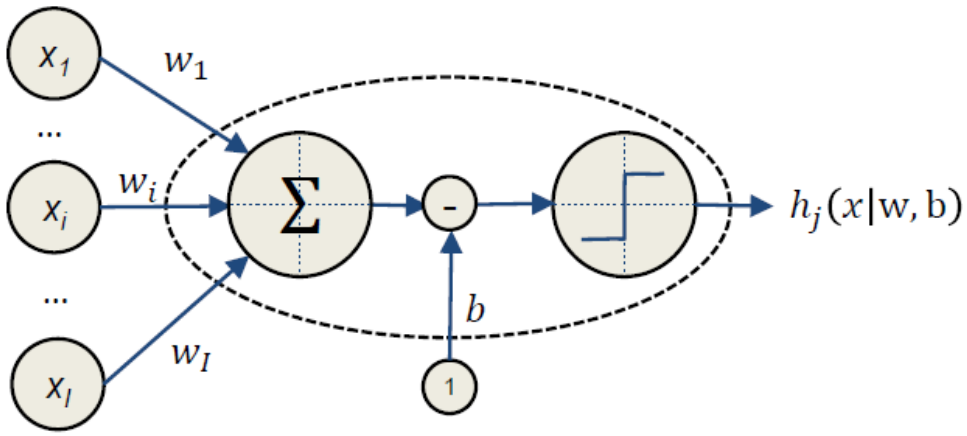


Figure 2.10: Elementary Artificial Neuron architecture.

In gradient based approaches, the correction to the synaptic weights w_{ij} is performed according to the direction identified by the partial derivatives (i.e. gradient), which can be calculated according to the chain rule as:

$$\frac{\partial \mathcal{E}(p)}{\partial w_{ij}(p)} = \frac{\partial \mathcal{E}(p)}{\partial \epsilon_j(p)} \frac{\partial \epsilon_j(p)}{\partial y_j(p)} \frac{\partial y_j(p)}{\partial v_j(p)} \frac{\partial v_j(p)}{\partial w_{ij}(p)} \tag{2.33}$$

Hence, the update to the synaptic weights Δw_{ij} is calculated as a *gradient descent* step in the weight space using the derivative of Eq. 2.33:

$$\Delta w_{ij} = -\eta \frac{\partial \mathcal{E}(p)}{\partial w_{ij}(p)} \tag{2.34}$$

where η is the tunable learning-rate parameter. The back-propagation algorithm entails two passages through the network: the *forward pass* and the *backward pass*. The former evaluates the output of the network as well as the function signal of each neuron. The weights are unaltered during the *forward pass*. The *backward pass* starts from the output layer by passing the loss function back to the input layer, calculating the local gradient for each neuron.

Incremental Learning Algorithms Incremental learning stands for the process of updating the weights each time a pair of input-target $(x, t)_p$ is presented. The two mentioned passes are executed at each step. This is the mode utilized for online application where the training process can potentially never stop as the data keep on being presented to the network.

Batch Learning Algorithms Batch learning algorithms execute the weights update only after all the input-target data are presented to the network. One complete presentation of the training dataset is typically called *epoch*. Hence,

after each epoch it is possible to define an average energy error function, which replaces Eq. 2.29 in the back-propagation algorithm:

$$\tilde{\mathcal{E}} = \frac{1}{2\mathcal{N}} \sum_{p=1}^{\mathcal{N}} \sum_{j \in \mathcal{C}_j} \epsilon_j^2(p) \tag{2.35}$$

The forward and backward pass is performed after each epoch.

CHAPTER 3

Neural-Dynamics Learning & Navigation

E nel mare cambió quella mia vita
E il mare trascurato mi travolse
Seppi che il mio futuro era sul mare
Con un dubbio peró che non si sciolse
Senza futuro era il mio navigare.

— Francesco Guccini

The neural reconstruction of dynamical models, including disturbances and unmodeled terms, is the focus of the first step towards the enhancement of the Guidance, Navigation & Control using Artificial Intelligence techniques. The idea is to develop flexible algorithms that are capable of learning the dynamics while flying in uncertain environments. Such dynamical models are used as basis to synthesize the guidance and control algorithms, together with the delicate task of navigating the spacecraft. The *better* the spacecraft knows the environment, the *more performant* the GNC system is.

The following Chapter is organized as follows: Section 3.1 presents the three developed approaches to employ Artificial Neural Networks for dynamics

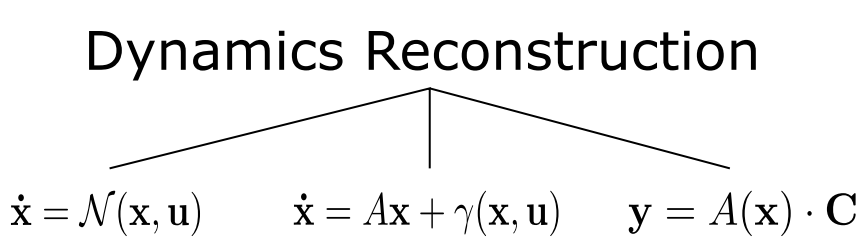


Figure 3.1: System dynamics identification: different approaches to reconstruct system dynamical behavior.

learning; Section 3.2, 3.3 and 3.4 thoroughly describe the methods. The comparisons between the different architectures, including advantages and disadvantages, are highlighted in the discussion.

3.1 Artificial Neural Network Models for Dynamics Reconstruction

The capability of using ANN to approximate the underlying dynamics of a spacecraft is used to enhance the on-board model accuracy and flexibility to provide the spacecraft with a higher degree of autonomy. There are different approaches that could be adopted to tackle the system identification and dynamics reconstruction task, as show in Fig. 3.1. In the following section the three analyzed methods are described but, first, it is important to remark the *universal approximation theorem* (cfr. Chapter 2), which represents the foundation of all the methods. The three methods employs the Artificial Neural Network model at different integration levels. In this Thesis, the nomenclature for the methods (see Fig. 3.1) is the following:

- Fully-Neural Dynamics Learning (Section 3.2)
- Dynamics Acceleration Reconstruction (Section 3.3)
- Parametric Dynamics Reconstruction (Section 3.4)

Fully-Neural Dynamics Learning The dynamical model of a system delivers the derivative of the system state, given the actual system state and external input. Such input-output structure can be fully approximated by an artificial neural network model. The dynamics is entirely encapsulated in the weights and biases of the network $\mathcal{N}$. The neural network is stimulated by the actual state and the external output. In turns, the time derivative of the state, or simply the system state at the next discretization step, is yielded as output, as shown in Fig. 3.2:

$$\dot{\mathbf{x}} = \mathcal{N}(\mathbf{x}, \mathbf{u}) \rightarrow \mathbf{x}_{k+1} = \tilde{\mathcal{N}}(\mathbf{x}_k, \mathbf{u}_k) \tag{3.1}$$

The method rely on the *universal approximation theorem* since it is based on the

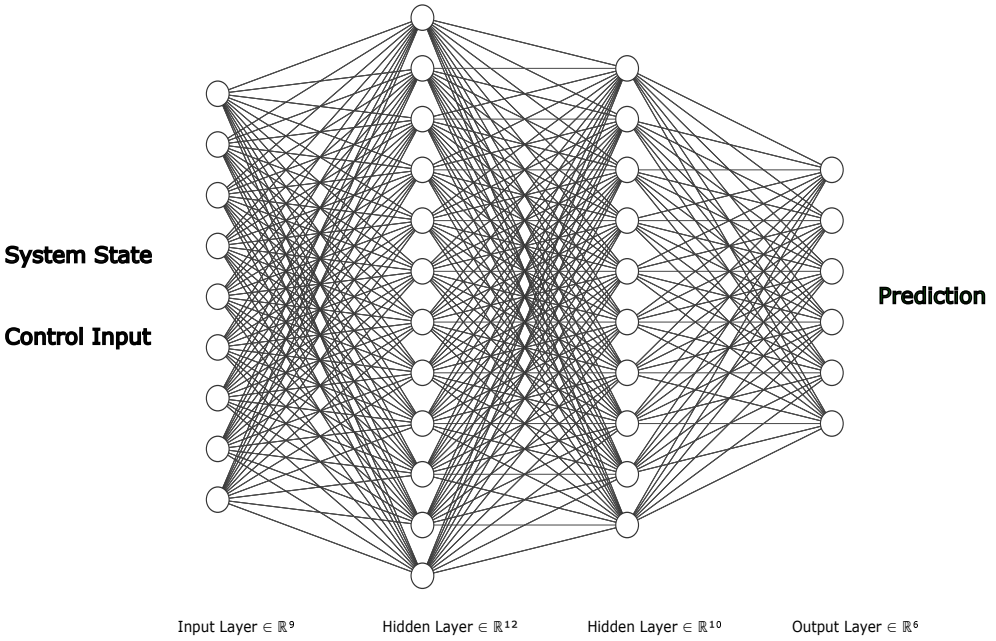


Figure 3.2: Dynamical reconstruction as a neural network model.

assumption that there exists an ANN that approximates the dynamical function with a predefined approximation error. The training set is simply composed of input-output pairs, where the input is a stacked vector of system state and control vector. The dynamics reconstruction based solely on artificial neural networks largely benefits by the employment of recurrent neural networks, rather than simpler feedforward networks.

Dynamics Acceleration Reconstruction The second method uses the capability of the Artificial Neural Networks to approximate an unknown function. In particular, it is wise to exploit every analytical knowledge we may have of the environment. Nevertheless, most of the time the analytical models encompass linearization and do not model perturbations, either because they are analytically complex or simply unknown. For this reason, a Radial Basis Function Neural Network aided Adaptive Extended Kalman Filter (RBFNN-AEKF) for state and disturbance estimation is developed. RBFNN are selected for their simple structure and suitability for fast online training [45]. The neural network estimates the unmodeled terms which are fed to the EKF as an additional term to the state and covariance prediction step. Finally, a recursive form of the adaptive EKF is employed to limit the overall computational cost. The main innovation is represented by the combination of mismodeling estimation by the RBFNN together with the adaptive formulation of the EKF, providing a robust, accurate and computationally efficient navigation filter that can be

initialized and run on-board. In brief, the method can be summarized into three key-points:

- The AI-filter uses the RBFNN to learn and output an estimate of the disturbance/mismodeled terms that is actually used in the EKF to deliver a better predicted estimate;
- The robustness of the filter is guaranteed through the adaptation step. The combination of a RBFNN and a filter can lead to a very wrong state estimation if the RBFNN estimates diverge or converge to a wrong value, hence the adaptivity guarantees the robustness of the navigation algorithm;
- The RBFNN learning is fully performed online. This means that no prior knowledge or learning has to be performed beforehand. This dramatically increases the flexibility of the algorithm;

The method has shown promising results for relative navigation in perturbed orbits with a very simple dynamical model. To strengthen such point, a comparison with other filter architectures with various degrees of dynamics uncertainty. is reported

Parametric Dynamics Reconstruction The third method is developed under the framework of parameter reconstruction. Basically, the Artificial Neural Network is employed to refine the uncertain parameters of a given dynamical model. This method is particularly suitable when the uncertain environment influences primal system constants (e. g. inertia parameters, spherical harmonics, drag coefficients). In this Thesis, the method is developed using a Recurrent Neural Network (cfr. 3.4) to estimate the spherical harmonic expansion coefficients of irregular bodies that populate the Solar System. Such method is very fast and computationally light with respect to traditional algorithms for parameters estimation. Moreover, given the physical knowledge of the parameters to be reconstructed, the method has a very promising *scientific* outcome. For instance, the gravity expansion of Asteroids and Planets can be approximated online while flying, delivering a rough shape reconstruction of the Body.

3.2 Fully-Neural Dynamics Learning

The Fully-Neural Dynamics Learning employs an Artificial Neural Network for encapsulating the whole dynamics reconstruction. Let us assume having a system that evolves according to the following equation:

$$\dot{\mathbf{x}} = \mathcal{F}(\mathbf{x}, \mathbf{u}) \tag{3.2}$$

Furthermore, it is assumed that the observation is equal to the state for the sake of simplicity. The algorithm can easily be extended to different measurement models by introducing the measurement function $h(\mathbf{x})$ or its linear matrix version $\mathbf{H}$:

$$\mathbf{y} = \mathbf{x} \tag{3.3}$$

The system dynamics can be learnt using an Artificial Recurrent Neural Network (RNN) trained by standard back-propagation algorithm [46]. One effective strategy is to leverage the physics of the problem in order to obtain a representation, which needs solely some parameters to be fit. Another approach is to couple the neural network with a estimation algorithm to reconstruct only the perturbation terms, taking as basis the linearized natural dynamics [40]. For this work, a Recurrent Neural Network has been chosen as architecture for the Neural Network. Its simple architecture and and brief evaluation time make it a suitable architecture for on-board applications. Recurrent Networks have the capability of handling time-series data efficiently. The connections between neurons form a directed graph, which allows an internal state memory. This enables the network to exhibit temporal dynamic behaviors. When dealing with dynamics identification, it is crucial to exploit the temporal evolution of the states, hence RNN shows superior performances with respect to MLP [47]. In detail, two recurrent networks are proposed to tackle the system identification problem. Namely:

- Layer-Recurrent Neural Network (LRNN)
- Nonlinear Autoregressive Network with Exogenous Inputs (NARX)

The details of the difference between the two implementation are left to the reader [44]. The highlight that is worth to be remarked is that the NARX model uses control action as inputs and state as output, given a certain n-delay of the training data. The NARX network is particularly suited for the task, being able to make prediction when used in *closed-loop* architecture. Although less performant than RNN, the system dynamics can also be learnt using a Feed-Forward Neural Network trained by standard back-propagation algorithm [46]. For this work, a Multi-Layer-Perceptron has been tested and compared with the Recurrent architecture. Its simple architecture and and brief evaluation time make it a suitable architecture for on-board applications. In particular a 2-layer network is employed, namely one hidden layers as sketched in Fig. 3.3. An hyperbolic tangent is used as the activation function for the hidden layer, whereas a Rectifier Linear Unit (ReLU) is used as neural activation function for the output layer. The dynamics of the spacecraft distributed system can be learnt as a discrete model. The downside of such strategy, especially in terms of implementation is that the ANN is associated with a fundamental time step of discretization. In an operative scenario, this could be the sampling frequency of navigation system. Nevertheless, it could be the case in which the

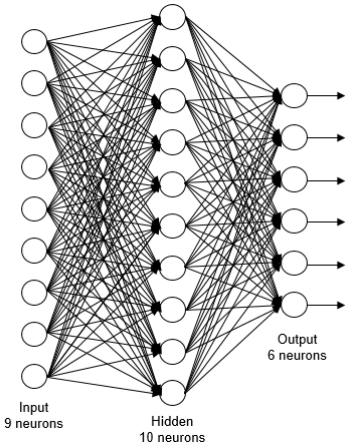


Figure 3.3: Two-layer MLP for dynamics identification.

navigation measurements frequency is much higher than the required planning one. This generates a discrepancy between the model to be learned and the model employed in trajectory generation. However, as stated above, a too fine sampling in a restricted region destroys the generalization capability of the network. The dynamical model can be represented as:

$$\mathbf{x}_{k+1} = \tilde{\mathcal{F}}_{T_s}(\mathbf{x}_k, \mathbf{u}_k) \tag{3.4}$$

where the transition matrix is associated to the sampling time T_s, as stated. The model is learned using a ANN $\tilde{\mathcal{N}}_{T_s}$ trained by back-propagation. In this work, the well established Levenberg-Marquardt [46] algorithm is employed for minimizing:

$$\underset{\mathbf{w}}{min} \sum_i ||\tilde{\mathcal{N}}_{T_s}(\mathbf{x}, \mathbf{u}, \mathbf{w}) - \mathbf{y}_{k+1}||^2 \tag{3.5}$$

where $\tilde{\mathcal{N}}$ is the ANN at the current learning step, hence the dependency on the weights $\mathbf{w}$. The vector $\mathbf{y}$ is the observation vector.

3.2.1 Prediction Performance & Comparison: RNN vs MLP

Fig. 3.4 shows the different propagation of the feed-forward (MLP) and the presented recurrent neural networks (LRNN and NARX), initialized equally and trained in the same scenario [47]. The prediction position and velocity accuracy, compared with the analytical nonlinear J2-perturbed model, demonstrates the superior performance in dynamics reconstruction. The implemented LRNN is a two-layer network with $10 - 6$ neurons, respectively in the hidden and output layer. The recurrent hidden layer has a 1-time delay feedback loop. The RNN architecture is shown in Fig. 3.7. The NARX network is a 2-step delay with the same number of neurons and layers of the LRNN. The

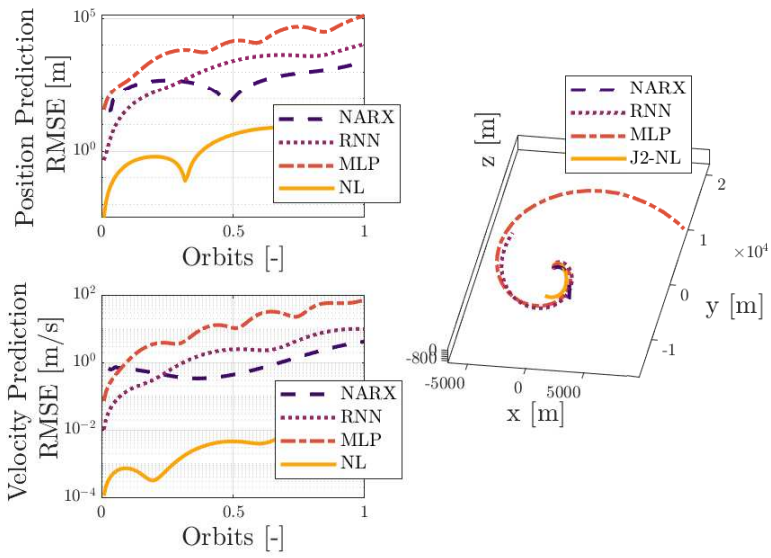


Figure 3.4: Comparison between NARX, LRNN and MLP dynamical propagation for $\mathcal{N}_s = 100$ planning steps.

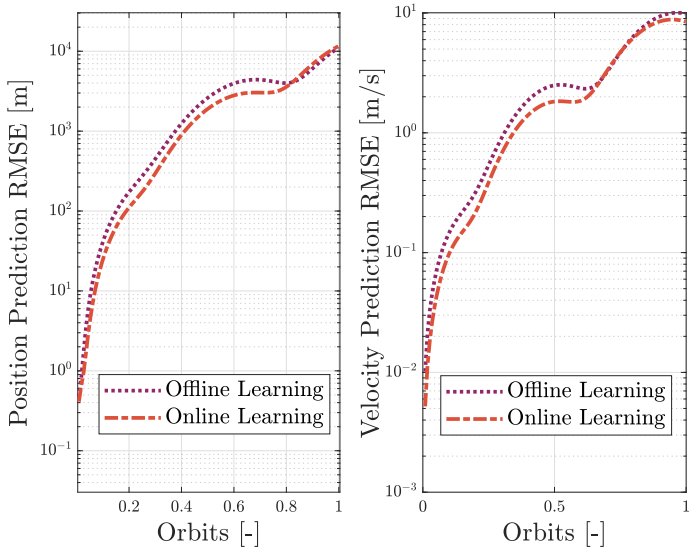


Figure 3.5: Comparison between initialized (offline) and refined (online) network prediction for $\mathcal{N}_s = 100$ planning steps.

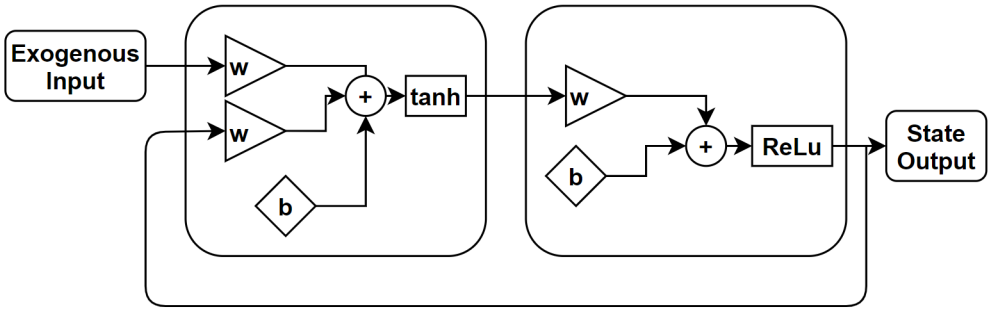


Figure 3.6: Nonlinear Autoregressive Exogenous Model.

process of learning can be initiated off-line based on one of the well-known dynamical model, such as Clohessy-Wiltshire or Yamanaka-Ankersen state transition matrix. The initialized RNN is constantly updated on-board while performing operations. Fig. 3.5 shows the improvement for the same open-loop prediction of $\mathcal{N}_s = 100$ planning steps between the initialized network (offline) and the network generated after performing one reconfiguration simulation. It is important to remark that the prediction reported in Fig. 3.5 is open-loop, namely a forward propagation, which is not representative of the closed-loop utilization of the network in the guidance and control algorithm. As the agents keep performing relative orbital maneuvers, i.e. formation keeping or reconfiguration, the knowledge of the actual dynamics is refined online, which can be used for a incremental performance planning. One issue with online learning is overfitting. In addition, a high sampling frequency of the state and action is not beneficial for the RNN training. Suppose the first batch of data for learning are very much localized in a given portion of $\mathcal{R}^9$ space, several hyper-surfaces approximate the given transition between $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$. The learning data are enclosed in a restricted region, hence several curves yield a low loss function in the back-propagation algorithm but the model but are definitely not suitable for generalization [47]. The limitation of the dataset to a bounded and restricted region is not beneficial for dynamics identification. Especially in a preliminary learning process this would drive the neural network to a wrong convergence.

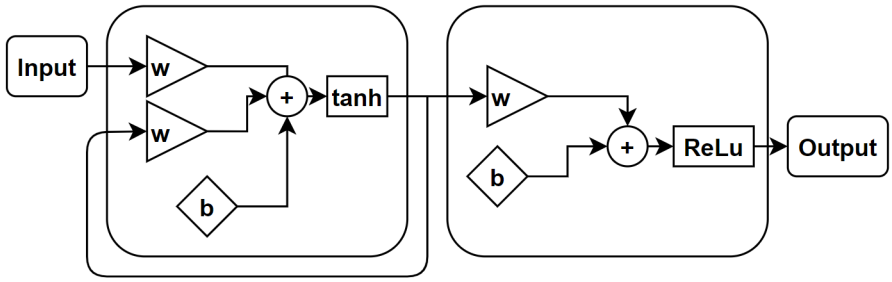


Figure 3.7: Layer Recurrent Neural Network architecture.

3.3 Dynamics Acceleration Reconstruction

3.3.1 Algorithm Architecture

The neural network estimates the disturbances acting on the system, which are then adjunct in the prediction step of the filter. The filter architecture is sketched in Fig. 3.8. The innovation term is used to carry out the adaptivity task. Whereas, the residual term, taking into account the estimation state at step k, is fed into the online learning algorithm of the network's weights. Each block of the RBFNN-AEKF is detailed in the following subsections. The

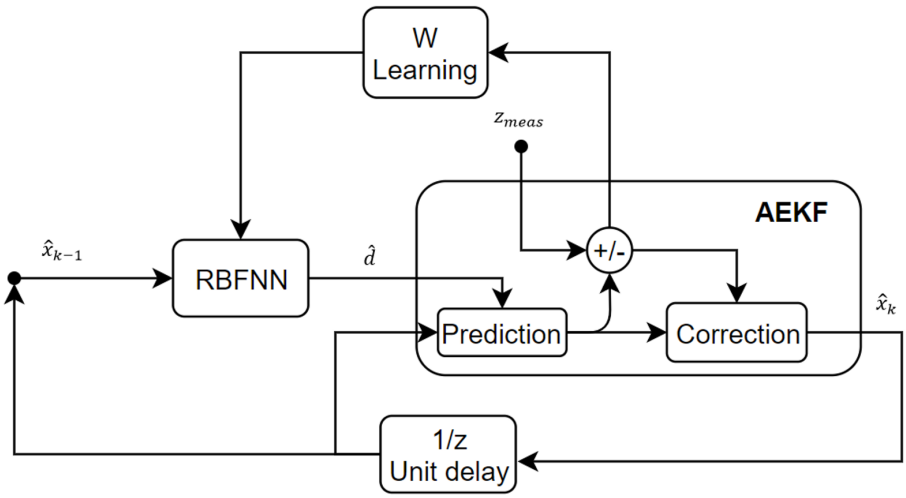


Figure 3.8: Proposed architecture for the RBFNN-AEKF.

system dynamics, taking into account the process noise, is assumed to be described as:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) + \mathbf{w} \tag{3.6}$$

Alongside, the measurements are assumed to be perturbed by white noise as:

$$\mathbf{z_{meas}} = \mathbf{Ix} + \mathbf{v} \tag{3.7}$$

Note that in the derivation presented in the following sections, we assume that the observation model is $\mathbf{z_{meas}} = \mathbf{Ix} + \mathbf{v}$ as in [27]. Normally, an observation function, often nonlinear, is introduced as explained in Section 3.3.3. Such assumption implies that the observation matrix $\mathbf{H} = \mathbf{I}$ or, more in general, $h(\mathbf{x}) = \mathbf{x}$. In other words, the state is assumed to be completely observable for sake of derivation but the approach is applicable to partially observable state and to nonlinear measurement models.

3.3.2 Radial Basis Functions Neural Network

The Radial-Basis Function Neural Network and its associated online learning algorithm is hereby explained. The remarkable feature of such algorithm consists on the online learning method, which guarantees strong flexibility and adaptivity of the system.

3.3.2.1 Neural Network Structure

The Radial Basis Function Neural Network (RBFNN) is a popular network topology, which has the capability of universal approximation [45] [48]. Due to its simple structure and much quicker learning process, it stands out compared to the classic Multi-Layer Perceptron (MLP), especially for function approximation applications [45]. The RBF neural network is a three-layer feedforward network, as seen in Figure 3.9. The RBFNN owns a single hidden-layer because it does not need multiple layers to obtain nonlinear behavior classification, as in Multi-Layer Perceptron. The neurons of RBF are nonlinear Gaussian function hence a shallow network can be used with the same results of Multi-Layer Perceptron. Hence, referring to Fig. 3.8, the lightest network has been chosen, consisting of one input layer, one hidden and one output layer. The input layer processes the state vector $\hat{\mathbf{x}}_{\mathbf{k-1}} = [x_1\ x_2\ ...\ x_n]^T$, where $k-1$ is the time instant. The hidden layer performs a nonlinear mapping of the input, whereas, the output layer is a linear combination of the nonlinear hidden neurons transformed into the resultant output space. The output space is the disturbance vector $\hat{\mathbf{d}}$=$[d_1\ d_2\ ...\ d_n]^T$. A RBFNN is used to estimate the unmodeled disturbances, as well as the nonlinearities present in the system dynamics. The generic layout of the network is sketched in Figure 3.9. The network has a 3-layers structure, comprising an input, output and hidden layer. For the sake of derivation we call $\mathbf{x} \in \mathbb{R}^n$ the input vector. It is hereby remarked that the vector $\mathbf{x}$ is employed to derive the network structure: in the following sections the distinction between state vector and estimated state will be described and treated accordingly. Similarly to the input vector, $\mathbf{\Phi} \in \mathbb{R}^m$ is

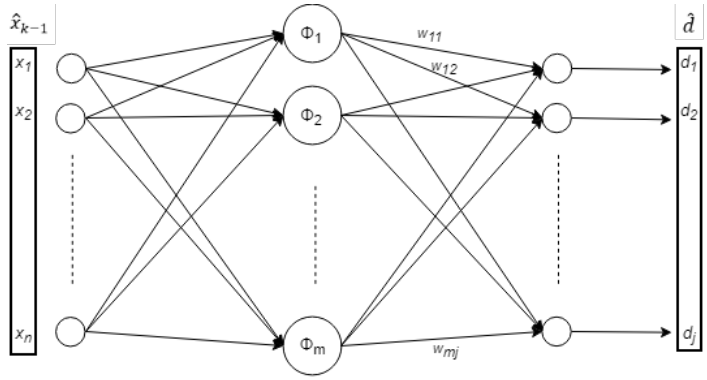


Figure 3.9: Architecture of the RBF neural network. The network processes the estimated states yield an estimate of the disturbance term. The input, hidden, and output layers have n, m, and j neurons, respectively. $\Phi_i(\mathbf{x})$ denotes the radial Gaussian function at the hidden node i.

the hidden layer vector and i the associated index, $\mathbf{d} \in \mathbb{R}^j$ is the output vector and l the associated index. In this derivation we assume that $n \equiv j$. Essentially, the hidden layer evaluates a set of m radial basis functions $\Phi : \mathbb{R}^n \to \mathbb{R}$, where n is the number of states, which are chosen as centered-Gaussian expression:

$$\Phi_i(\mathbf{x}) = e^{-\eta(||\mathbf{x}-\mathbf{c_i}||)^2} \tag{3.8}$$

for $i = 1 : m$, where m is the number of neurons and $\mathbf{c}_i$ is the randomly selected center for neuron i. The number of neurons m is a user-defined parameter: its value is application-dependent and it shall be selected by trading-off the reconstruction accuracy and the computational time. The same consideration holds for the parameter η, which impacts the shape of the Gaussian functions. A high value for η sharpens the Gaussian bell-shape, whereas a low value spreads it on the real space. On one hand, a narrow Gaussian function increases the responsiveness of the RBF network, on the other hand, in case of limited overlapping of the neuronal functions due to too narrow Gaussian bells, the output of the network vanishes. Hence, ideally, the parameter η is selected based on the order of magnitude of the exponential argument in Eq. 3.8. The output of the neural network hidden layer, namely the radial functions evaluation, is normalized:

$$\mathbf{\Phi}_{norm}(\mathbf{x}) = \frac{\mathbf{\Phi}(\mathbf{x})}{\sum_{i=1}^{m} \Phi_i(\mathbf{x})} \tag{3.9}$$

The classic RBF network presents an inherent localized characteristic; whereas, the normalized RBF network exhibits good generalization properties, which decreases the curse of dimensionality that occurs with classic RBFNN [45]. In the following derivation, the output vector of the hidden layer is simply called

$\mathbf{\Phi}(\mathbf{x})$ without the subscript *norm* for the sake of simplicity. For a generic input $\mathbf{x} \in \mathbb{R}^n$, the components of the output vector $\mathbf{d} \in \mathbb{R}^j$ of the network is:

$$d_l(\mathbf{x}) = \sum_{i=1}^{m} w_{il} \Phi_i(\mathbf{x}) \tag{3.10}$$

In a compact form, the output of the network can be expressed as:

$$\mathbf{d}(\mathbf{x}) = \mathbf{W}^T \mathbf{\Phi}(\mathbf{x}) \tag{3.11}$$

where $\mathbf{W} = [w_{il}]$ for $i = 1, ..., m$ and $l = 1, ..., j$ is the trained weight matrix and $\mathbf{\Phi}(\mathbf{x}) = [\Phi_1(\mathbf{x}) \ \Phi_2(\mathbf{x}) \ \cdots \ \Phi_m(\mathbf{x})]^T$ is the vector containing the output of the radial basis functions, evaluated at the current system state.

3.3.2.2 Online Learning Algorithm

The dynamical model can be described by a set of nonlinear differential equations:

$$\dot{\mathbf{x}} = f(\mathbf{x}) + \mathbf{d}_{ext} \tag{3.12}$$

where the term $\mathbf{d_{ext}}$ is representative of the unknown disturbance acceleration that is added to the known dynamics function $f(\mathbf{x})$. In particular the disturbance term gathers the contribution of all the environmental perturbations, and unmodeled terms. These uncertainties need to be estimated online. Hence, an online learning algorithm, which drives the update of the weights, is required. The weights update law is derived to guarantee the stability of the estimation algorithm and neural dynamics, hereby defined as the evolution of the weight matrix in time. In the following mathematical derivation we make use of the universal approximation theorem for neural networks that guarantees the existence of a set of ideal weights $\mathbf{W}$ that approximates a function with a bounded arbitrary approximation error [45]. Such weights are unknown, hence the algorithm is designed to obtain an estimate $\hat{\mathbf{W}}$ of the ideal weights by performing online learning. The neural network learning algorithm relies on the estimation error dynamics, targeting convergence and stability of the estimated weights matrix $\hat{\mathbf{W}}$ evolution towards the ideal weights and the error $\mathbf{e}$, calculated in the EKF. The symbol $\hat{(\cdot)}$ is used to refer to *estimated* quantities.

To derive the error dynamics, let us assume the actual system dynamics is described by Eq. 3.12, where $\mathbf{d_{ext}}$ is the unknown external disturbance term. The actual system dynamics can be rewritten as the following equation, assuming to include all the nonlinear terms into $d(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^j, j \equiv n$, which is the vector-valued function equivalent to the RBFNN output vector:

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + d(\mathbf{x}) \tag{3.13}$$

where the term $d(\mathbf{x})$ captures all the nonlinearities together with the unknown disturbances external to the system, namely $d(\mathbf{x}) = f(\mathbf{x}) - \mathbf{A}\mathbf{x} + \mathbf{d_{ext}}$. The matrix A is a stable, potentially time-varying, matrix representing the linear term, if any, of the original dynamics expression in Eq. 3.12.

The expression of the continuous single-step Kalman filter can be written as:

$$\dot{\hat{\mathbf{x}}} = \mathbf{A} \cdot \hat{\mathbf{x}} + \hat{d}(\hat{\mathbf{x}}) + \mathbf{K_k}\mathbf{H}(\mathbf{x} - \hat{\mathbf{x}}) \tag{3.14}$$

where $\hat{d}$ is estimated using the radial-basis function neural network, $\mathbf{K_k}$ is the time-varying gain matrix of the Kalman filter (subscript k stands for the referred time step) and $\mathbf{H}$ is the observation matrix. In this work, the observation matrix of the measurement model is assumed to be the identity matrix, nevertheless the derivation is not significantly altered if one would consider a different expression for the measurement model. The measurements are assumed to be affected by white noise as in Eq. 3.7. Consider that the continuous form is employed for the sake of derivation, indeed the learning rule is then discretized for the actual implementation. The error dynamics can be derived as:

$$\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}} \tag{3.15}$$

$$\dot{\mathbf{e}} = \dot{\mathbf{x}} - \dot{\hat{\mathbf{x}}} = d(\mathbf{x}) - \hat{d}(\hat{\mathbf{x}}) + (\mathbf{A} - \mathbf{K_k}\mathbf{H})\mathbf{e} \tag{3.16}$$

Invoking the universal approximation theorem for neural networks [45], we can assume there exists an ideal approximation of the disturbance term $d(\mathbf{x})$:

$$d(\mathbf{x}) = \mathbf{W}^T \mathbf{\Phi}(\mathbf{x}) + \epsilon \tag{3.17}$$

where $\mathbf{W}$ is the neural weights matrix, $\mathbf{\Phi}(\mathbf{x})$ is the vector-valued function resulting from the evaluation of the Gaussian functions contained in each neuron of the RBFNN, ϵ is a bounded arbitrary approximation error. Consequently, the error in estimation can be written as:

$$d(\mathbf{x}) - \hat{d}(\hat{\mathbf{x}}) = \mathbf{W}^T \mathbf{\Phi}(\mathbf{x}) + \epsilon - \hat{\mathbf{W}}^T \mathbf{\Phi}(\hat{\mathbf{x}}) \tag{3.18}$$

by adding and subtracting the term $\mathbf{W} \cdot \mathbf{\Phi}(\hat{\mathbf{x}})$ and performing few mathematical manipulations, Equation 3.18 can be expressed as:

$$\tilde{d} = \tilde{\mathbf{W}}^T \Phi(\hat{\mathbf{x}}) + \epsilon' \tag{3.19}$$

where $\tilde{d} = d - \hat{d}$, $\tilde{\mathbf{W}} = \mathbf{W} - \hat{\mathbf{W}}$ and the bounded term $\epsilon' = \epsilon + \mathbf{W} \cdot [\mathbf{\Phi}(\mathbf{x}) - \mathbf{\Phi}(\hat{\mathbf{x}})]$. The aim of the learning rule is to drive the dynamics error to zero, as well as forcing the weights to converge to the ideal ones. Namely:

$$\mathbf{e} \to \mathbf{0}, \ \tilde{\mathbf{W}} \to [\mathbf{0}]$$

Similarly to [27], introducing the following scalar Lyapunov function for the feedback system, including the network weights and the estimation error, the

weights update rule $\dot{\hat{\mathbf{W}}}$ is derived to guarantee the stability and convergence of the estimation algorithm:

$$V = \frac{1}{2} tr(\xi \tilde{\mathbf{W}}^T \tilde{\mathbf{W}}) + \frac{\eta}{2} \mathbf{e}^T \mathbf{e} \tag{3.20}$$

where $tr(\cdot)$ is the trace operator, $\xi,\ \eta > 0$ are user-defined coefficients.

Recalling Eq. 3.16 and 3.19, the derivative of the Lyapunov function can be written as:

$$\begin{aligned}
\dot{V} &= tr(\xi \tilde{\mathbf{W}}^T \dot{\tilde{\mathbf{W}}}) + \eta \mathbf{e}^T \dot{\mathbf{e}} \\
&= tr(\xi \tilde{\mathbf{W}}^T \dot{\tilde{\mathbf{W}}}) + \eta \mathbf{e}^T (\tilde{\mathbf{W}}^T \mathbf{\Phi}(\hat{\mathbf{x}}) + \epsilon' + (\mathbf{A} - \mathbf{K_k H})\mathbf{e}) \\
&= tr(\xi \tilde{\mathbf{W}}^T \dot{\tilde{\mathbf{W}}}) + \eta \mathbf{e}^T \tilde{\mathbf{W}}^T \mathbf{\Phi}(\hat{\mathbf{x}}) + \eta \mathbf{e}^T \epsilon' + \eta \mathbf{e}^T (\mathbf{A} - \mathbf{K_k H})\mathbf{e} \\
&= tr(\xi \tilde{\mathbf{W}}^T \dot{\tilde{\mathbf{W}}} + \eta \tilde{\mathbf{W}}^T \mathbf{\Phi}(\hat{\mathbf{x}}) \mathbf{e}^T) + \eta \mathbf{e}^T \epsilon' + \eta \mathbf{e}^T (\mathbf{A} - \mathbf{K_k H})\mathbf{e} \\
&= tr(\tilde{\mathbf{W}}^T (\xi \dot{\tilde{\mathbf{W}}} + \eta \mathbf{\Phi}(\hat{\mathbf{x}}) \mathbf{e}^T)) + \eta \mathbf{e}^T \epsilon' + \eta \mathbf{e}^T (\mathbf{A} - \mathbf{K_k H})\mathbf{e} < 0
\end{aligned} \tag{3.21}$$

Recalling that $\dot{\tilde{\mathbf{W}}} = -\dot{\hat{\mathbf{W}}}$, the expression for the weights update rule that guarantees stability and convergence of the estimation algorithm and feedback system in Fig. 3.8 is:

$$\dot{\hat{\mathbf{W}}} = \frac{\eta}{\xi} \mathbf{\Phi}(\hat{\mathbf{x}}) \mathbf{e}^T \tag{3.22}$$

Indeed, by inserting Eq. 3.36 into Eq. 3.21, the expression for the derivative of the Lyapunov function reduces to the stability of the error estimation of the Extended Kalman Filter. The error term in Eq. 3.36 is defined as Eq. 3.15. Such term represent the residual between estimated and actual output of the observed system: in practical terms, the expression is the *innovation* of the estimation filter, which takes this form based on the assumption of Eq. 3.7. The ϵ' term is a bounded term that derives from the universal approximation theorem of artificial neural networks [45] that states that the term ϵ can be arbitrarily small. In practice, it represents an upper boundary for the derivative of the Lyapunov function, as in [27]. In the case of linear systems, the term $(\mathbf{A} - \mathbf{K_k H})$ grants asymptotic stability of the Kalman Filter if $\mathbf{A}$ is reachable and $\mathbf{H}$ is observable. In the case of nonlinear systems, this is not always true. However, it has been proved [49] that the estimation error of an EKF is exponentially bounded if:

- $\mathbf{A}$ is non-singular for every $t \geq 0$;
- there exist real constants $p_1, p_2 > 0$ such that $p_1 \cdot \mathbf{I} \leq \mathbf{P_k} \leq p_2 \cdot \mathbf{I}$, where $\mathbf{P_k}$ is the estimated state covariance matrix;
- the initial estimation error satisfies $||\hat{\mathbf{x}}_\mathbf{0} - \mathbf{x_0}|| \leq \epsilon$ and the process and measurements covariance matrices are bounded

where $\hat{\mathbf{x}}_\mathbf{0}$ and $\mathbf{x_0}$ are the estimated and true state vector at the initial step. Given the EKF asymptotic stability with exponential decaying error under the aforementioned conditions, i.e. the derivative of the Lyapunov function of the estimation error is negative, Eq. 3.21 is verified and hence the stability of the estimator is guaranteed.

The weights update rule can be discretized using a first-order Euler method, assuming the measurements interval is small enough:

$$\hat{\mathbf{W}}_{k+1} = \psi\hat{\mathbf{W}}_k + h\dot{\hat{\mathbf{W}}}_k = \psi\hat{\mathbf{W}}_k + h\frac{\eta}{\xi}\mathbf{\Phi}(\hat{\mathbf{x}}_\mathbf{k})\mathbf{e}_k^T \tag{3.23}$$

where k is the time step index, ψ is a user-defined relaxation factor, $h = t_{k+1} - t_k$ is the time interval between two consecutive measurements.

3.3.3 Adaptive Extended Kalman Filter

The EKF is one of the most common techniques for nonlinear state estimation. In fact, it represents the extension of the linear Kalman Filter when dealing with nonlinear dynamical systems [50]. The EKF is the standard approach for relative navigation filters [51, 52, 53, 54]. However, sometimes, accurate dynamical models are not available on-board. This is mostly due to computational limitations (e.g. for Cubesats) or unavailability of precise formulations (e.g. non-Keplerian dynamics). In these cases, a simpler model has to be used and the unmodeled effects have to be estimated. In this section the proposed approach combining the introduced RBFNN and a Kalman Filter is presented. The novelties lie in the formulation of a stand-alone estimator block containing a disturbance estimator and a filter. It is important to underline as the RBFNN and the filter are tightly coupled and they do not constitute two separate pieces. This is also a consequence of the implementation of an adaptive form of the process covariance matrix update.

Let's consider the system and measurement models:

$$\mathbf{x_k} = f(\mathbf{x_{k-1}}, \mathbf{u_{k-1}}, \mathbf{w_{k-1}}) = \mathbf{A}\mathbf{x_{k-1}} + d(\mathbf{x_{k-1}}, \mathbf{u_{k-1}}) + \mathbf{w_{k-1}}, \tag{3.24}$$

$$\mathbf{z_k} = h(\mathbf{x_k}, \mathbf{v_k}) \tag{3.25}$$

with $\mathbf{x}$ being the state vector, $\mathbf{u}$ the control input, $\mathbf{w}$ and $\mathbf{v}$ process and measurement noises, described by zero-mean white noise uncorrelated distributions with covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ respectively. Please notice that the system dynamics is described by a linear part plus a non-linear disturbance function d, by assumption (see 3.13). In this case, the formulation of the RBFNN-AEKF is given by:

$$\hat{\mathbf{x}}_\mathbf{k}^- = \mathbf{A}\hat{\mathbf{x}}_\mathbf{k-1}^- + d(\hat{\mathbf{x}}_\mathbf{k-1}^-, \mathbf{u_{k-1}}) \tag{3.26}$$

$$\mathbf{P}_\mathbf{k}^- = \tilde{\mathbf{F}}_\mathbf{k-1}\mathbf{P}_\mathbf{k-1}^+\tilde{\mathbf{F}}_\mathbf{k-1}^\mathbf{T} + \mathbf{Q_{k-1}} \tag{3.27}$$

$$\mathbf{K_k} = \mathbf{P_k^-}\mathbf{H^T}(\mathbf{H}\mathbf{P_k^-}\mathbf{H^T} + \mathbf{R_k})^{-1} \tag{3.28}$$

$$\mathbf{P_k^+} = (\mathbf{I} - \mathbf{K_k}\mathbf{H})\mathbf{P_k^-}(\mathbf{I} - \mathbf{K_k}\mathbf{H})^{\mathbf{T}} + \mathbf{K_k}\mathbf{R_k}\mathbf{K_k^T} \tag{3.29}$$

$$\hat{\mathbf{x}}_\mathbf{k}^+ = \hat{\mathbf{x}}_\mathbf{k}^- + \mathbf{K_k}(\mathbf{z_k} - \mathbf{H}\hat{\mathbf{x}}_\mathbf{k}^-) \tag{3.30}$$

with

$$\tilde{\mathbf{F}} = \mathbf{A} + \left.\frac{\partial d}{\partial \mathbf{x}}\right|_{\hat{x}_k^-}; \qquad \mathbf{H} = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{x}_k^-} = \mathbf{I} \tag{3.31}$$

and

$$\mathbf{Q_k} = \alpha\mathbf{Q_{k-1}} + (1-\alpha)(\mathbf{K_k}\delta_\mathbf{k}\delta_\mathbf{k}^\mathbf{T}\mathbf{K_k^T}) \tag{3.32}$$

being $\mathbf{P}_k$ the estimation error covariance and $\mathbf{K}_k$ the Kalman gain, α is a forgetting factor and $\delta_\mathbf{k} = \mathbf{z_k} - \mathbf{H}\hat{\mathbf{x}}_\mathbf{k}^-$ is the filter innovation. The Jacobian in Eq. 3.42 of the vector-valued function reconstructed by the RBFNN is derived from Eq. 3.35:

$$\frac{\partial d}{\partial \mathbf{x}} = \frac{\partial \mathbf{W}^T\mathbf{\Phi}(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{W}^T\frac{\partial \mathbf{\Phi}(\mathbf{x})}{\partial \mathbf{x}} \in \mathbb{R}^{n\times n} \tag{3.33}$$

In this formulation, the state is assumed to completely observable. This assumption can be relaxed but a different formulation has to be developed for the neural network (see [27]). The adaptation of $\mathbf{Q}$ is performed according to Eq. 3.43 as in [55] to limit the computational effort. The adaptation step is a fundamental aspect in the implementation of this estimation technique. In fact, the nonlinear term estimated by the RBFNN $d(\mathbf{x})$ directly affects the state estimation and, moreover, it influences the accuracy of the adopted dynamical model at each time step. The evolution in time of the model accuracy is very difficult to be established a-priori being dependent on the effectiveness of the disturbance term estimation, performed by the RBFNN time-varying. Although this aspect is often neglected [23], an online tuning of the process covariance matrix $\mathbf{Q}$ is fundamental to ensure filter accuracy and robustness. In fact, the adaptive formulation guarantees that, even when the neural network produces a completely wrong estimates of the disturbances, yielding a significantly biased dynamical model, the filter, at least, follows the available measurements. In fact, if the network yields a disturbance term that is significantly off, the dynamical model is no longer reliable. This implies a large value of the filter innovation δ, which delivers a high-valued process covariance matrix according to Eq. 3.43. As result, the filter does not diverge but simply follows the measurements (Eq. 3.27).

The formulation reported so far considers a Cartesian dynamical model. If the Relative Orbital Elements $\delta\chi$ representation is used, the equations are recasted according to the following specifications. An artificial neural network is employed to approximate the dynamical terms encompassing all the unmodeled nonlinearities and disturbances. The method could potentially be exploited in several different environments. The disturbances and nonlinear terms may be

caused by different sources, such as gravity harmonics, solar radiation pressure, drag, etc. The ROE dynamics for unperturbed motion is equal to the Keplerian matrix in Eq. 2.8. If the difference in the semimajor axis is null, the Keplerian dynamics in ROE space vanishes. This makes the perturbations relevant even if the magnitude is very low. For instance, the J2 perturbation term in the ROE dynamics is $\sim 10^{-10}s^{-}1$.

The universal approximation theorem of artificial neural networks guarantees the existence of a set of ideal weights $\mathbf{W}$ that approximates a function with a bounded arbitrary approximation error [45]. As mentioned, such network architecture possesses a quick learning process, which makes it suitable for online dynamics identification and reconstruction. The highlights of the mathematical expression of the RBFNN are reported here for clarity. For a generic state input $\delta\chi \in \mathbb{R}^n$, the components of the output vector $\gamma \in \mathbb{R}^j$ of the network is:

$$\gamma_l(\delta\chi) = \sum_{i=1}^{m} w_{il}\Phi_i(\delta\chi) \tag{3.34}$$

In a compact form, the output of the network can be expressed as:

$$\gamma(\delta\chi) = \mathbf{W}^T\mathbf{\Phi}(\delta\chi) \tag{3.35}$$

where $\mathbf{W} = [w_{il}]$ for $i = 1, ..., m$ and $l = 1, ..., j$ is the trained weight matrix and $\mathbf{\Phi}(\mathbf{x}) = [\Phi_1(\mathbf{x})\ \Phi_2(\mathbf{x})\ \cdots\ \Phi_m(\mathbf{x})]^T$ is the vector containing the output of the radial basis functions, evaluated at the current system state.

The online learning algorithm is derived using Lyapunov stability theorem, as before. The RBFNN weights matrix is updated as follows:

$$\dot{\hat{\mathbf{W}}} = \frac{\eta}{\xi}\mathbf{\Phi}(\hat{\delta\chi})\mathbf{e}^T \tag{3.36}$$

where $\mathbf{e} = \delta\chi - \hat{\delta\chi}$. The latter term is the estimated state output of the navigation filter.

The prediction is based on the augmented dynamics taking into account the contribution of the RBFNN. The measurements are the Cartesian state in the LVLH frame of the reference satellite. In this case the formulation of the RBFNN-AEKF is given by:

$$\hat{\delta\chi}_{\mathbf{k}}^{-} = \mathbf{A_k}\hat{\delta\chi}_{\mathbf{k-1}}^{-} + \gamma(\hat{\delta\chi}_{\mathbf{k-1}}^{-}, \mathbf{u_{k-1}}) \tag{3.37}$$

$$\mathbf{P_k^-} = \tilde{\mathbf{F}}_{\mathbf{k-1}}\mathbf{P_{k-1}^+}\tilde{\mathbf{F}}_{\mathbf{k-1}}^{\mathbf{T}} + \mathbf{Q_{k-1}} \tag{3.38}$$

$$\mathbf{K_k} = \mathbf{P_k^-}\mathbf{H^T}(\mathbf{H}\mathbf{P_k^-}\mathbf{H^T} + \mathbf{R_k})^{-1} \tag{3.39}$$

$$\mathbf{P_k^+} = (\mathbf{I} - \mathbf{K_k}\mathbf{H})\mathbf{P_k^-}(\mathbf{I} - \mathbf{K_k}\mathbf{H})^{\mathbf{T}} + \mathbf{K_k}\mathbf{R_k}\mathbf{K_k^T} \tag{3.40}$$

$$\hat{\delta\chi}_{\mathbf{k}}^{+} = \hat{\delta\chi}_{\mathbf{k}}^{-} + \mathbf{K_k}(\mathbf{z_k} - \mathbf{H}\hat{\delta\chi}_{\mathbf{k}}^{-}) \tag{3.41}$$

with

$$\tilde{\mathbf{F}} = \mathbf{A_k} + \left.\frac{\partial d}{\partial \delta\chi}\right|_{\hat{\delta\chi}_k^-}; \mathbf{H} = \mathbf{J}_{\delta\chi}^{\mathbf{X}} \tag{3.42}$$

and

$$\mathbf{Q_k} = \alpha\mathbf{Q_{k-1}} + (1-\alpha)(\mathbf{K_k}\delta_\mathbf{k}\delta_\mathbf{k}^\mathbf{T}\mathbf{K_k^T}) \tag{3.43}$$

being $\mathbf{A}$ the dynamics matrix, γ the output of the network representing the approximation of all the unmodeled terms (relative drag and nonlinearities), $\mathbf{P}_k$ the estimation error covariance, $\mathbf{Q}$ the process covariance matrix, $\mathbf{H}$ the measurement matrix, $\mathbf{R}$ the measurements covariance matrix, $\mathbf{K}_k$ the Kalman gain, α is a forgetting factor, $\mathbf{z_k}$ the measurements and $\delta_\mathbf{k} = \mathbf{z_k} - \mathbf{H}\hat{\delta\chi}_\mathbf{k}^-$ is the filter innovation. The Jacobian in Eq. 3.42 of the vector-valued function reconstructed by the RBFNN is derived from Eq. 3.35:

$$\frac{\partial\gamma}{\partial\delta\chi} = \frac{\partial\mathbf{W}^T\mathbf{\Phi}(\delta\chi)}{\partial\delta\chi} = \mathbf{W}^T\frac{\partial\mathbf{\Phi}(\delta\chi)}{\partial\delta\chi} \in \mathbb{R}^{n\times n} \tag{3.44}$$

In this formulation, the state is assumed to be completely observable. In the simulations, the Gaussian noise is used to corrupt the measurements to be representative of actual sensors output. The noise affects only the EKF update. The overall algorithm implementation is reported in Algorithm 1:

Algorithm 1 Radial-Basis Function Neural Network Adaptive EKF

1: Random initialization of the RBFNN
2: RBFNN estimation step of d using the estimated state at the previous step $d(\mathbf{x}) = \mathbf{W}^T\mathbf{\Phi}(\mathbf{x})$
3: RBFNN weight update $\dot{\hat{\mathbf{W}}} = \frac{\eta}{\xi}\mathbf{\Phi}(\hat{\mathbf{x}})\mathbf{e}^T$
4: EKF Prediction step, considering the computed d term $\hat{\mathbf{x}}_\mathbf{k}^- = \mathbf{A}\hat{\mathbf{x}}_\mathbf{k-1}^- + d(\hat{\mathbf{x}}_\mathbf{k-1}^-, \mathbf{u_{k-1}})$
5: EKF Update step $\hat{\mathbf{x}}_\mathbf{k}^+ = \hat{\mathbf{x}}_\mathbf{k}^- + \mathbf{K_k}(\mathbf{z_k} - \mathbf{H}\hat{\mathbf{x}}_\mathbf{k}^-)$
6: EKF $\mathbf{Q}$ adaptation step $\mathbf{Q_k} = \alpha\mathbf{Q_{k-1}} + (1-\alpha)(\mathbf{K_k}\delta_\mathbf{k}\delta_\mathbf{k}^\mathbf{T}\mathbf{K_k^T})$
7: Estimated state $\hat{x}_k^+$

3.3.4 Application to Spacecraft Relative Navigation

In this section, we introduce one of the possible applications of the proposed RBFNN-AEKF. The relative navigation between two spacecrafts orbiting the Earth is considered for a dual motivation: it is a well-known scenario, hence sophisticated model can be employed to simulate the reality to evaluate the filter performances; also, there are many available dynamical models with increasing levels of accuracy that can be used for comparison. It is worth remarking that this is not the only application nor the most appealing one, since more uncommon scenarios are expected to emphasize the benefit of the proposed filter, such as interplanetary mission or non-keplerian orbits. Hereby, the different dynamical models for filter propagation are presented as well as the filter alternatives used for comparison.

In this subsection, we present the filters used for the comparison. Besides the RBFNN-AEKF, the new filter proposed in this work, other filters are tested under the same simulation scenario:

- a state observer based on the formulation in [27];
- a standard, non-adaptive EKF aided with a RBFNN;
- an EKF exploiting a more accurate, nonlinear dynamical model.

3.3.4.1 Observer

The dynamics of the relative motion between the spacecrafts is reconstructed using a modified full-state observer [27]. In the same fashion as Section 3.3.2, the state observer can be constructed as follows [56]:

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}_{cw} \cdot \hat{\mathbf{x}} + \hat{d}(\hat{\mathbf{x}}) + \mathbf{K_h}(\mathbf{z} - \mathbf{H}\hat{\mathbf{x}}) \tag{3.45}$$

where $\mathbf{A}_{cw}$ is the linear time invariant matrix of the Clohessy-Wiltshire dynamics presented in Section 2.1.1.1, $\hat{d}$ is estimated using the radial-basis function neural network in Section 3.3.2, $\mathbf{K_h}$ is the user-defined observer gain matrix and $\mathbf{H} = \mathbf{I}$ is the identity observation matrix, as already stated in Section 3.3.1.

3.3.4.2 RBFNN-EKF

This filter is a standard EKF aided with RBFNN as presented in Section 3.3.1. The only difference with respect to the proposed RBFNN-AEKF is that the value of the process covariance $\mathbf{Q}$ is fixed in time. It is worth underlying that this can be a very weak point because it is hard to a-priori establish the accuracy of the RBFNN-based disturbance estimation, especially for very uncertain dynamics. In fact, the matrix $\mathbf{Q}$ provides an indication of the accuracy of the dynamical model. Using a neural network to update the dynamical model and to increase its accuracy, it is therefore necessary to adapt the value of $\mathbf{Q}$ at each iteration step. The RBFNN-EKF formulation is based on Equations 3.26-3.30.

3.3.4.3 EKF - Nonlinear Propagation

The last tested filter is an EKF with a different dynamical model. There is not any coupling with the neural network but the used dynamical model is nonlinear and accounting for J_2 perturbations. In particular, a standard EKF is employed where the evolution of the state vector is described by the nonlinear model introduced in Section 2.1.1.2.

Table 3.1: Chaser-Target Orbital Parameters

	Chaser	**Target**
a [km]	8143.1	8143.1
e [-]	$1.4{\cdot}10^{-1}$	$1.4{\cdot}10^{-1}$
i [°]	98.2	98.2
ω [°]	85.9	85.9
Ω [°]	79.2	79.2
θ [°]	0	$1{\cdot}10^{-4}$
A_{sp} $[m^2]$	1.2	0.2

3.3.5 Reconstruction and Navigation Performance

In this section, the numerical simulation environment to evaluate the filter performance is described. First, the selected scenario is presented. Subsequently, the capability of disturbance reconstruction of the RBFNN-AEKF is tested. Then, the estimation algorithms are compared using a realistic orbital environment. At this point, the definition of measurement noise levels and filters tuning are introduced. Finally, the same simulation is performed using non-nominal filter tuning conditions to test the robustness of the navigation filters.

3.3.5.1 Orbital Scenario

The reference relative orbital motion is generated considering two spacecraft with the same initial orbital parameters except for the true anomaly. Table 3.1 reports the chaser and target initial orbital parameters along with the cross sectional area, important for disturbances evaluation. with A_{sp} being the cross sectional area. These orbital parameters result in the following relative initial conditions, expressed in the target LVLH reference frame:

$$\rho_0 = [-0.0017 \;\; -12.2042 \;\; 4.7 \cdot 10^{-4}] \; m \tag{3.46}$$

$$\dot{\rho}_0 = [-0.0017 \;\; 3.9 \cdot 10^{-6} \;\; -5.6 \cdot 10^{-10}] \; m/s \tag{3.47}$$

Please note that the reference orbits are eccentric and that the cross sectional areas are different, resulting in a different perturbation effect and potential dynamical mismodeling. In fact, the different cross sectional areas lead to a different differential drag and solar radiation pressure perturbations. It is worth underlying that at the selected altitude, the effect of the drag is not relevant but still present. A similar reasoning can be done for the solar radiation pressure. The most relevant effect is given by the eccentricity of the orbits: the design models neglecting such contribution (e.g. Clohessy-Wiltshire) deliver a modelling errors of tens of meters after one relative orbits. The presented

scenario has been selected as representative of a leader-follower formation, separated along the orbit by a difference in the true anomaly. The truth model, as discussed before, is propagated through the high fidelity propagator considering all the perturbation effects. The dynamical models used by the filters depend on the selected architecture and are detailed in Chapter 2.

3.3.5.2 Disturbance Reconstruction

The RBFNN disturbance approximation capability is assessed through the simulation of the scenario presented in Section 3.3.5.1. In order to have a quantitative disturbance term, which can be compared to the ANN estimation, the actual relative motion is propagated using the J_2-perturbed relative motion in Section 2.1.1.2. Instead, the filter exploits a simple Clohessy-Wiltshire linearized model described in 2.1.1.1. The normalized ANN consists of 60 hidden neurons with Gaussian-basis radial functions; the function centers are generated randomly. The number of neurons has been selected by trading-off the reconstruction accuracy and the computational time. In such framework, the disturbances that need to be estimated are caused by the following elements:

- J_2 zonal gravity perturbation
- $e \neq 0$, elliptical orbits;

together with the nonlinearities neglected in the derivation in Section 2.1.1.1. Using the J_2-perturbed nonlinear model in Section 2.1.1.2, the disturbance term is explicit in the form of $\mathbf{d} = [d_x \; d_y \; d_z]^T$ acceleration term. For the coherence of vectors dimensionality the estimation is actually performed for the vector $\mathbf{d}_{6x1} = \mathbf{G}\mathbf{d}$, where $\mathbf{G} = [\mathbf{0_{3x3}}; \mathbf{I_{3x3}}]$. The reference orbit is a LEO, which is incidentally assumed to be the orbit of the target spacecraft. Table 3.1 reports the orbital parameters of the two spacecrafts.

It is assumed to have relative position and velocity measurements at 1Hz, with a noise level described by a Gaussian distribution with standard deviation $\sigma_{pos} = 10^{-2}m$ and $\sigma_{vel} = 10^{-4}m/s$. These values are representative of a relative RF metrology system (see [17]). The estimation of the disturbance acceleration term converges after a transient time of nearly 350 s: this represents the *main learning process* of the randomly initialized network. The network is said to be converged when the estimation error is less than 10% of the nominal value. Figure 3.10 shows the learning curve of the network during the early phase of the orbital motion. The RBFNN is randomly initialized: indeed, the estimation is significantly off by almost ~ 6 orders of magnitudes during the initial phase. The disturbance acceleration components, after the *main learning process*, are shown in Figure 3.11. Despite the measurement noise, the estimation yields a Root Mean Squared Error (RMSE) reported in Table 3.2.

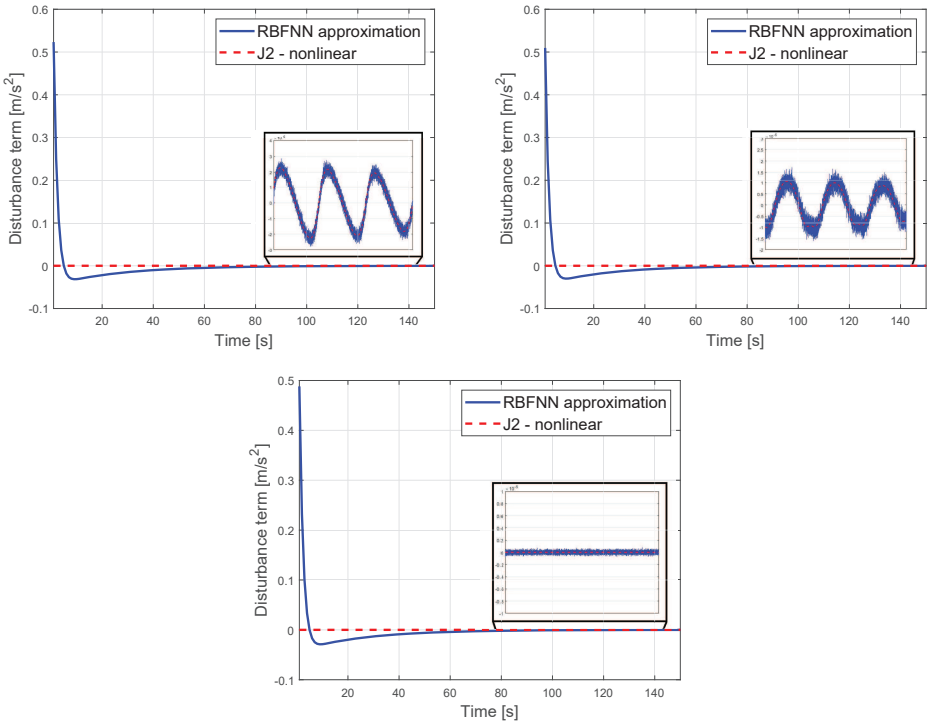


Figure 3.10: Estimation of the disturbance acceleration term for LEO reference orbit. The perturbations are in the order of $10^{-5}\ \frac{m}{s^2}$. The plots show the initial phase of the neural network learning, regarded as the *main learning process*. From left to right: d_x, d_y and d_z.

The RBFNN is used to reconstruct the dynamical terms, which are not directly included in the on-board dynamics representation. As mentioned, the method is quite general and can be applied to different unknown environments regardless of the sources of the dynamical disturbances. In this work, the *ground-truth* dynamics is a fully nonlinear Cartesian model including J2 perturbation, whereas the dynamical model used by the GNC algorithms is based on ROE. For this reason, it is difficult to show a comparison between the *true* disturbances and the *reconstructed* ones. A dedicated simulation is here reported to show the reconstruction capabilities of the RBFNN.

In the reported simulation, the *ground-truth* natural dynamics follows Eq. 2.8, whereas the *on-board* dynamics is simply the Keplerian term A_k of Eq. 2.8 coupled with the RBFNN reconstructed term. Fig. 3.12 shows the network approximation of the disturbance term due to J2 in ROE dynamics. The root mean squared error of the approximation is $\sim 10^{-13} s^{-1}$ once the network has converged to a steady-state.

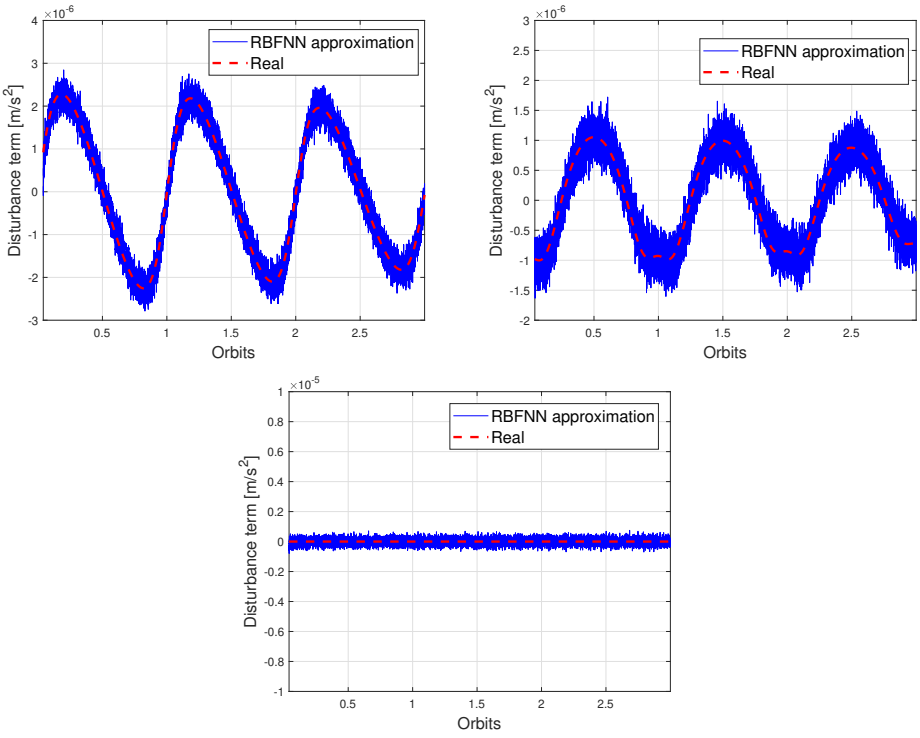


Figure 3.11: Estimation of the disturbance acceleration term for LEO reference orbit after the *main learning process*. The plots show the estimation of the disturbance term by the neural network after the network has converged. From left to right: d_x, d_y and d_z.

3.3.5.3 Relative Navigation - Nominal Case

An accurate orbital simulator is used to test the filters in a realistic environment, as described in Section 2.1.1. The normalized neural network consists of 60 hidden neurons with Gaussian-basis radial functions; the function centers are generated randomly. This reference orbits are also used to generate relative measurements by adding a fictitious noise, representative of realistic sensors uncertainty. In particular, the noise level associated to relative position and velocity measurement respectively, is described by a Gaussian distribution with standard deviation $\sigma_{pos} = 10^{-2}m$ and $\sigma_{vel} = 10^{-4}m/s$, similarly to the previous case. It is important to remark that the orbits are eccentric and the cross sectional areas of the two spacecrafts are significantly different, yielding a strong differential perturbation effect due to the solar radiation pressure and drag. The estimation errors, used for performance assessment, are introduced. The relative position error is defined as:

$$e_\rho = \sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (z_i - \hat{z}_i)^2} \tag{3.48}$$

Table 3.2: Root-mean-squared error of the disturbance estimation term for the LEO reference orbit

	Value
$\sigma_x \left[\frac{m}{s^2}\right]$	$7.2 \cdot 10^{-7}$
$\sigma_y \left[\frac{m}{s^2}\right]$	$7.6 \cdot 10^{-7}$
$\sigma_z \left[\frac{m}{s^2}\right]$	$5.9 \cdot 10^{-7}$

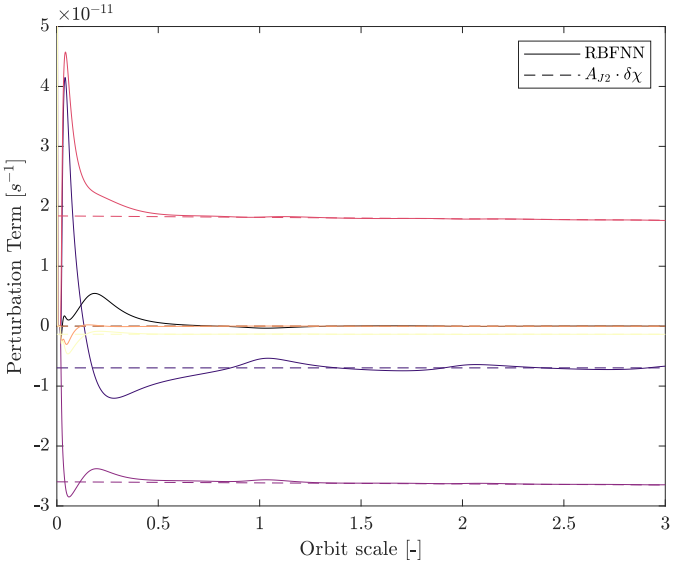


Figure 3.12: RBFNN network approximation of the disturbance terms due to J2 in ROE dynamics. The disturbance term is a vector $\gamma \in \mathbb{R}^6$.

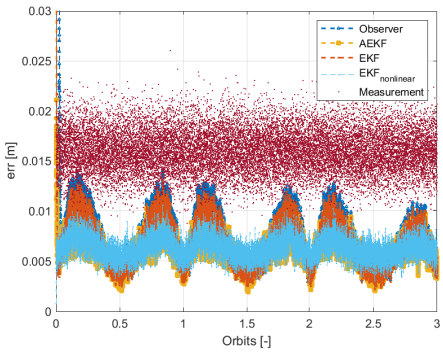


Figure 3.13: Relative Position Error

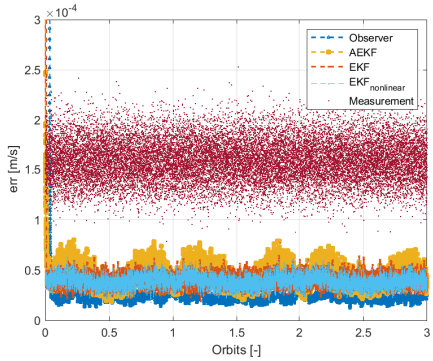


Figure 3.14: Relative Velocity Error

where $\hat{x}, \hat{y}, \hat{z}$ are the position components estimates. Similarly, the relative velocity error is:

$$e_{\dot{\rho}} = \sqrt{(\dot{x}_i - \hat{\dot{x}}_i)^2 + (\dot{y}_i - \hat{\dot{y}}_i)^2 + (\dot{z}_i - \hat{\dot{z}}_i)^2} \tag{3.49}$$

with $\hat{\dot{x}}, \hat{\dot{y}}, \hat{\dot{z}}$ are the velocity components estimates.

The measurement covariance matrix $\mathbf{R}$ for all the filters is tuned according to the imposed measurement noise level. The same process covariance matrix $\mathbf{Q}$ is used for the RBFNN-AEKF and RBFNN-EKF and, for the nonlinear EKF, it is properly selected to guarantee the best steady state error performance. Similarly, the observer gain $\mathbf{K_h}$ is tuned to guarantee the minimum steady state error. A statistical analysis of the filters has been performed over 100 runs for the described scenario. The filters run with a frequency of 1Hz and the simulation duration is set to three chaser orbits to appreciate the disturbances effect. Figures 3.13 and 3.14 show the relative position and velocity error averaged over 100 runs. For a more quantitative analysis of the results, the Root Mean Square Error (RMSE) starting from time step 300 (at steady-state) are computed and reported in Table 3.3 to evaluate the steady state performance of the filters.

Figure 3.13 and 3.14 show the beneficial effect of the filters compared to the measurements error. The RBFNN-AEKF and the EKF-nonlinear show a similar behaviour for the relative position error (Figure 3.13) and, as in Table 3.3, they outperform the other alternatives. On the other hand, for what concerns the velocity estimation, the Observer, with this tuning, has better performance than the other filters. Despite these small differences, the compared filters show similar performance, and the order of magnitude of the RMSE, reported in Table 3.3, is the same.

Table 3.3: Filters RMSE Results

	RMSE - Position [m]	**RMSE** - Velocity [m/s]
Observer	0.0079	$2.39{\cdot}10^{-5}$
RBFNN - AEKF	0.0063	$4.49{\cdot}10^{-5}$
RBFNN - EKF	0.0074	$4.03{\cdot}10^{-5}$
EKF - nonlinear	0.0064	$3.92{\cdot}10^{-5}$

Table 3.4: Filters RMSE Results - Non-Nominal

	RMSE - Position [m]	**RMSE** - Velocity [m/s]
Observer	0.0149	$5.98{\cdot}10^{-5}$
RBFNN - AEKF	0.0064	$4.79{\cdot}10^{-5}$
RBFNN - EKF	0.0090	$9.34{\cdot}10^{-5}$
EKF - nonlinear	0.0110	$9.55{\cdot}10^{-5}$

3.3.5.4 Relative Navigation - Non-nominal Case

A proper tuning of the filter, however, is difficult to achieve when the process dynamics is not well known and time-varying. Moreover, it is very hard to a-priori determine the accuracy in the estimation that the RBFNN can achieve for that particular case. For this reason, we tested all the filters with off-nominal conditions. In particular, for each simulation, the value of $\mathbf{Q}$ and $\mathbf{K_h}$ were randomly selected according to a uniform distribution centered in the nominal value and spanning two order of magnitudes. This can be a very high uncertainty value for some applications, but we wanted to show how the tuning strongly affects the filter performance. Table 3.4 shows the relative position and velocity RMSE computed over 100 runs.

It is possible to appreciate how the estimation error of the RBFNN-AEKF is very similar to the nominal case. This is an evidence of high robustness of the proposed solution. On the contrary, all the other filters are badly affected from the inappropriate selection of $\mathbf{Q}$ or $\mathbf{K_h}$ respectively.

3.4 Parametric Dynamics Reconstruction

3.4.1 The Parametric Identification Problem

As a global approximation technique of the true gravitational field, the SHE has been largely studied and applied for mission analysis purposes in the past years [57],[58],[59]. Being an analytical model, it results to be computationally efficient and light to be evaluated, which makes it suitable for various applica-

tions. In a SHE, the gravity field of the body is assumed to be represented through a potential of the form:

$$\begin{aligned}\mathcal{U} = \frac{GM}{r} + \frac{GM}{r}\sum_{n=2}^{N}\left(\frac{R_0}{r}\right)^n \Bigg[C_n\mathcal{P}_n^0(\cos\theta) + \dots \\ + \sum_{m=1}^{n}\left(C_{nm}\cos(m\lambda) + S_{nm}\sin(m\lambda)\right)\mathcal{P}_n^m(\cos\theta)\Bigg]\end{aligned} \tag{3.50}$$

Here θ is the colatitude, λ the longitude, r the radial distance to the center of mass of the body, R_0 a reference radius and $\mathcal{P}_n^m(x)$ Associated Legendre Polynomials (ALP). For the aforementioned peculiar properties of the model, the SHE is assumed to be reconstructed in this work. Hence, the objective becomes to estimate the coefficients C_n,C_{nm} and S_{nm} of the expansion. In particular, the model to be reconstructed, in the case of the S2BP, is the following:

$$\ddot{\mathbf{r}} + \boldsymbol{\Omega}\times(\boldsymbol{\Omega}\times\mathbf{r}) + 2\boldsymbol{\Omega}\times\mathbf{r} = \nabla\mathcal{U}(\mathbf{r}) \tag{3.51}$$

Writing the SHE in a matrix form, then the model can be written as:

$$\ddot{\mathbf{r}} + \boldsymbol{\Omega}\times(\boldsymbol{\Omega}\times\mathbf{r}) + 2\boldsymbol{\Omega}\times\mathbf{r} = -\frac{GM}{r^3}\mathbf{r} + \mathrm{A}(\mathbf{r})\cdot\mathbf{C} \tag{3.52}$$

Where here, the vector $\mathbf{C}$ contains all the coefficients of the expansion to be estimated. Now, recalling:

$$\mathbf{y} = \ddot{\mathbf{r}} + 2\boldsymbol{\Omega}\times\dot{\mathbf{r}} + \boldsymbol{\Omega}\times(\boldsymbol{\Omega}\times\mathbf{r}) + \frac{GM}{r^3}\mathbf{r} \tag{3.53}$$

The model can be written in the so called Linear-in-parameters (LIP) form:

$$\mathbf{y} = \mathrm{A}(\mathbf{r})\cdot\mathbf{C} \tag{3.54}$$

According to Alonso et al. [60] and Atencia et al. [61], being the model linear in the parameters, the identification problem can be reformulated as an optimization problem. In particular, defining the *prediction error* $\mathbf{e} = \mathbf{y} - \mathrm{A}\cdot\mathbf{C}^*$ the resulting combinatorial optimization problem is [62]:

$$\min_{\mathbf{C}}\left\{\sup_t\left(\frac{1}{2}\mathbf{e}^T\cdot\mathbf{e}\right)\right\} \tag{3.55}$$

With a similar procedure, the formulation can be extended to the MCR3BP.

3.4.2 Hopfield Neural Networks

Several different algorithms are available to solve combinatorial optimization problems. Being a fast, accurate and computationally light technique the

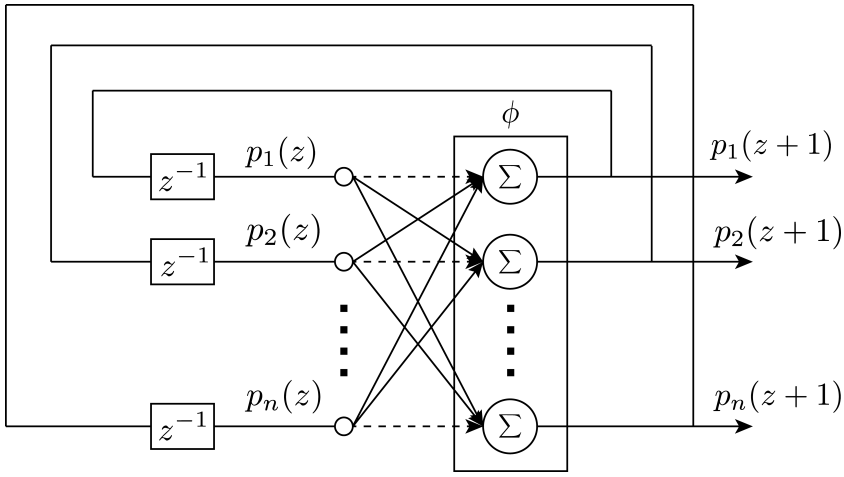


Figure 3.15: The Hopfield Neural Network structure.

Hopfield Neural Network (HNN) is considered in this work, fostering the on-board use of the method. The formulation of the network is due to Hopfield [63]. In this work, Abe [64] modified formulation is used, being the most suited for combinatorial optimization problems. It is a recurrent network whose neuron dynamics can be reduced to:

$$\frac{d\mathbf{s}}{d\tau} = \frac{1}{\beta}\mathrm{D}\left(\mathrm{W}\mathbf{s} + \mathbf{b}\right) \tag{3.56}$$

where $\mathbf{s}(\tau)$ is neuron state, β an hyper-parameter of the network $\mathrm{D} = \mathrm{diag}(1 - s_i^2)$, $\mathrm{W} = -\mathrm{A}^T\mathrm{A}$ is called *weight matrix* and $\mathbf{b} = \mathrm{W}\mathbf{s}_0 + \mathrm{A}^T\mathbf{y}$ is called *bias vector*. Both the weight matrix and the bias vector are associated to the SHE model and can be recovered matching the Lyapunov function of the network with the cost function of the optimization problem [65],[61]. Here $\mathbf{s}_0 = \mathbf{s}(0)$.

3.4.3 Discrete-time Hopfield Neural Network

Usual discrete versions of HNN include Backward Euler methods. However, according to [66] a much better discrete version of the network results to be:

$$(s_i)_{k+1} = \frac{(s_i)_k + \tanh\left(\frac{h}{\beta}(net_i)_k\right)}{1 + (s_i)_k \tanh\left(\frac{h}{\beta}(net_i)_k\right)} \tag{3.57}$$

where h is the time-step, $(s_i)_k$ is the state of the i-th neuron at the k-th step and

$$(net_i)_k = \sum_j (w_{ij})_k (s_j)_k - (b_i)_k \tag{3.58}$$

Note that this version is bounded but is not continuous whenever the denominator is zero. In principle, this condition cannot be achieved since $|s_i| < 1$, but, due to numerical round-off errors it has to be taken into account in a computer implementation of the discrete method. In this study, the choice is to set $(s_i)_{k+1} = (s_i)_k$ whenever the singularity is encountered. This discrete version of the network, however, still suffers the time step choice.

3.4.4 Gravity Field Identification of Small Solar System Objects

The gravitational field reconstruction of some objects through the use of a HNN is deeply analysed in this section with the aim to highlight dependencies with respect to both the orbital conditions as well as the network tuning parameters. In general, for a generic irregular object of mass M, the i-th coefficient C_i^* is coincident with the i-th neuron state s_i. Its behaviour can be written as a function of the orbital state $\mathbf{x}$, and so of $(\mathbf{r}, \mathbf{v})$, as well as the network hyperparameter β, which is associated to the activation function of the neural network. In particular, in this case $s_i(t_k) = \psi(\mathbf{s}(t_{k-1}), \beta)$ where here $\psi(\cdot)$ is the activation function. The HNN activation function is the hyperbolic tangent with a β value that regulate its steepness. In particular, small values of β are associated with a steeper function and so to an activation that is more sensitive to the inputs. On the other hand, values of $\beta \to 1$ make the activation less sensitive. Then, in general:

$$C_i^*(t) = f(M, \mathbf{x}, \beta, \mathbf{s}(t_{k-1}), t_k) \tag{3.59}$$

Note that the dependence on the state $\mathbf{x}$ can be written also in terms of the current osculating elements associated to the trajectory $(a, e, i, \Omega, \omega, \nu)$. Moreover, since $\mathbf{x}(t_k)$ depends on $\mathbf{x}(t_0)$, and t_k is considered as an independent variable, the convergence of the i-th coefficient is assumed to depend on the initial conditions on the orbit, the mass, the number of coefficients that are reconstructed, N_C (instead of directly their neuron states) and the hyperparameter β, so that:

$$C_i^*(t) = f(M, a_0, e_0, i_0, \Omega_0, \omega_0, \beta, N_C, t_k) \tag{3.60}$$

A normalization of the equation of motion is now introduced, to try to decouple the strict dependence of the neural network to the body and the orbit, as highlighted in [13]. To do so, we introduce a reference two-body acceleration:

$$a_{\text{ref}} = \frac{\mu}{R_{\text{ref}}^2} \tag{3.61}$$

In this manner, defining with $\tilde{q}$ the normalized version of a quantity q, the equations of motion in Eq.2.17, becomes:

$$\begin{cases} \tilde{x}'' - 2\tilde{y}' &= \tilde{x} - \left(\frac{R_{\text{ref}}}{r}\right)^2 \frac{\tilde{x}}{\tilde{r}} + \frac{R_{\text{ref}}^2}{\mu}\frac{\partial \mathcal{U}_p}{\partial x} \\ \tilde{y}'' + 2\tilde{x}' &= \tilde{y} - \left(\frac{R_{\text{ref}}}{r}\right)^2 \frac{\tilde{y}}{\tilde{r}} + \frac{R_{\text{ref}}^2}{\mu}\frac{\partial \mathcal{U}_p}{\partial y} \\ \tilde{z}'' &= -\left(\frac{R_{\text{ref}}}{r}\right)^2 \frac{\tilde{z}}{\tilde{r}} + \frac{R_{\text{ref}}^2}{\mu}\frac{\partial \mathcal{U}_p}{\partial x} \end{cases} \tag{3.62}$$

This version of the equation of motion is particularly useful to be used in the neural network since the weight and bias matrix results to be normalized. This process is here presented for the S2BP but can be extended to the SCR3BP. Finally, the choice of R_{ref} is important: to decouple the problem at the most with respect to both the body and the orbit, the choice is $r(t)$, in such a way that always hold:

$$\left| \frac{r^2(t)}{\mu} \nabla \mathcal{U}_{n,m}(r) \right| \leq 1$$

So that the dependencies of Eq.3.60 can be translated in:

$$C_i^*(t_k) = f_i(\beta, a_0, e_0, i_0) + \mathcal{O}(N_c) + \mathcal{O}^2(\Omega_0, \omega_0) \tag{3.63}$$

In order to understand potential critical issues, the normalized neural network is extensively tested on some sample cases. An integral measure of the error of the i-th reconstructed coefficient, $C_i^*(t)$, with respect to its the real value, C_i, is then introduced:

$$\text{iMSE}_i = \sum_k \frac{1}{2}(C_i^*(t_k) - C_i)^2 \tag{3.64}$$

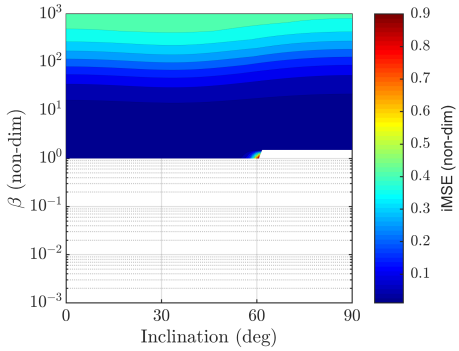
Note that this parameter of merit is an integral measure that weights both the accuracy and the velocity of the network.

Test cases An oblate spheroid with flatness $f = 0.5$ is considered in that case. $R_0 = 1$ km and $\rho = 2200$ kg/m^3 are assumed. This simple test is used to highlight eventual hidden dependencies. The orbital path is discretized with $\Delta t = 60$ seconds.

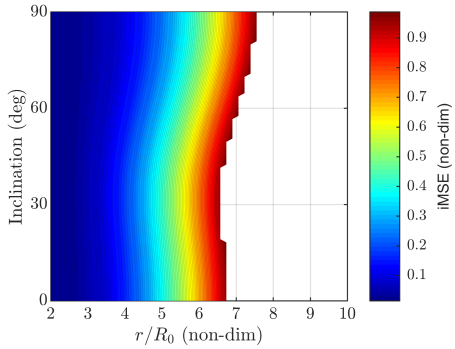
The results for the iMSE associated to the estimation of the parameter C_{20} are presented in Fig.3.16 where are presented a series of cut of the function in Eq.3.60. The areas in white are excluded, having iMSE $\geq$1.

1. From Fig. 3.16a it is evident that the dependence on the inclination of the orbit is *not* a dominant parameter, in this simple example.

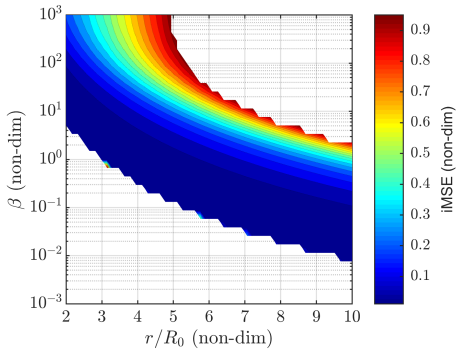
2. Fig.3.16b shows instead that there is a stronger dependence on the distance to the body $r(t)$.



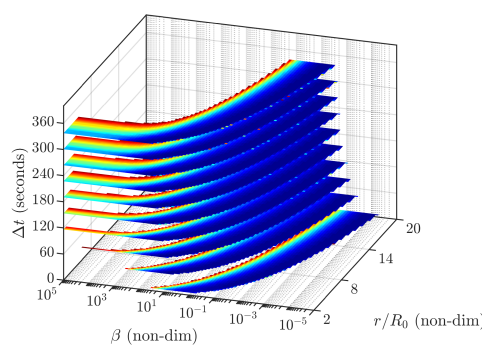
(a) Cut of the surface along $a_0 = 3R_{\max}$.



(b) Cut of the surface along $\beta = 15$.

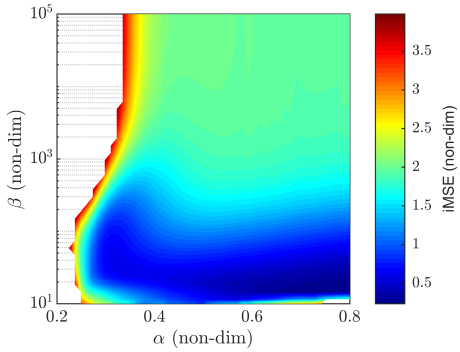


(c) Cut of the surface along $i = 45$ °.

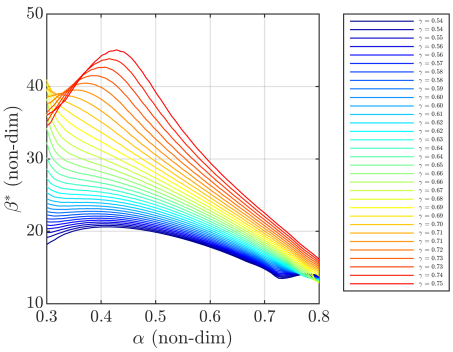


(d) Cut on $i = 45$ °.

Figure 3.16: Results for the test case.



(a)



(b)

Figure 3.17: Cut on $i = 45$ °, $R_0 = 1$ km and $r/R_0 = 3$ as a function of α, β and γ.

3. A cross-dependence between β and $r(t)$ is instead highlighted in Fig. 3.16c where it is evident that β have a decreasing monotonic behaviour with

respect to $r(t)$. Note that on the top right of the surface the iMSE is larger than 1. This means that the network convergence velocity gets smaller, since in all cases presented in this analysis the network do converge in the given time window.

4. Fig. 3.16d highlight a slight dependence on the time discretization $h = \Delta t$, coupled to the choice of β.

5. The normalization introduced *cancel* the dependence on the body mass, meaning that the same body re-scaled have the same optimal β, β^*. This however can be misleading since bodies with the same mass but different shape can have different β^*. It means that considering a triaxial ellipsoidal body, for a fixed ellipsoid's axis tuple (α, β, γ), the network β^* choice do not depend on R_0 (and so on the body mass) but *do* depend on the tuple, as it can be seen in Fig. 3.17.

The results in Fig. 3.17 do formalize the previous discussion. Fig. 3.17a in fact represent the dependence of the network β^* to the ellipsoid parameter α: in order to consider bodies with the same mass, the ellipsoid tuple is build as $(\alpha, 1, 1-\alpha)$. A slight dependence on the α of the β^* is evident. This dependence is more evident if two of the body parameters (namely α, γ) are varied. In Fig. 3.17b it can be seen that the more the body is regular the more the β^* value converge to a single value, while the more the irregular the body is the more the standard deviation on the mean optimal β is larger. Then, the convergence of the coefficient $C_i^*(t)$ results to be function of:

$$C_i^*(t) = f(d(t)) + \mathcal{O}(\text{irregularity}, N_C) \tag{3.65}$$

Then since a given β^* do depend on the degree of irregularity of the body and the time-step as well as on the distance ratio $d(t) = r(t)/R_0$, the convergence finally results a function of:

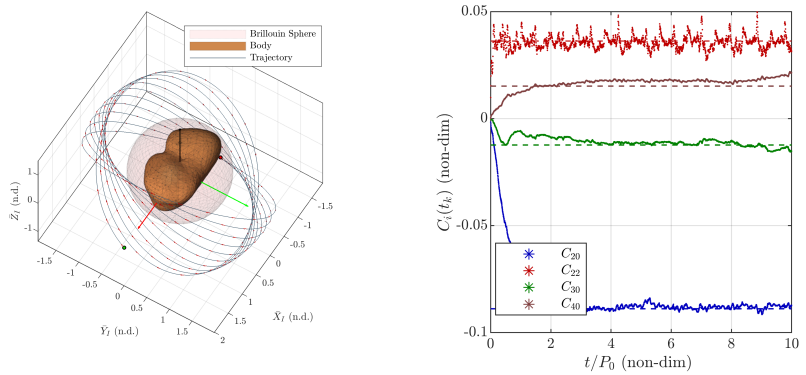
$$C_i^*(t) = f(d(t), \beta^*) + \mathcal{O}(\text{irregularity}, N_C) \tag{3.66}$$

3.4.5 Applications to real dynamical environments

In this section the dynamical environments represented through the P2BP and the MCR3BP are used to generate trajectories, in order to address the performances of the neural network to reconstruct the gravitational field of real objects in a real gravitational environment. In order to have a reliable measure of the network convergence velocity and accuracy, some assumptions are made:

1. The environment does not include SRP nor Sun third-body perturbation.

2. The orbital states are assumed to be reconstructed through an EKF. The state measurements are assumed to be perturbed with zero-mean white



(a) Inertial body-centred frame orbit.

(b) nHNN results.

Figure 3.18: Asteroid Castalia results: $a_0 = 2R_{max}$, $i_0 = 135°$ circular orbit.

Gaussian noise. In particular, position and velocity are perturbed using $\sigma_r = 10^2\ m$ and $\sigma_v = 10^{-2}\frac{m}{s}$.

3. The filter model is assumed to be the classical Two-Body Problem model.
4. Orbits with a retrograde acceleration component ($90° < i < 270°$) are preferred, for their inherent stability properties.

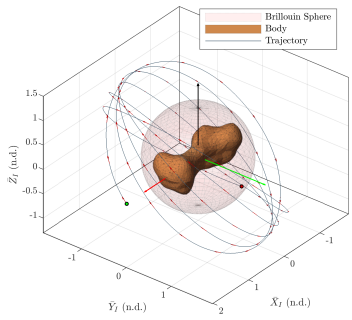
3.4.5.1 Case Studies: Castalia, Kleopatra and Phobos

In Figs. 3.18, 3.19, 3.20 are presented and discussed some examples. In those cases, the state vector is normalized using the maximum body radius R_0 and the orbital time using the initial *Keplerian* orbital period P_0, in such a way the figures represent non dimensional quantities. Note that in all cases the network exhibit really good convergence performances in a relatively small time.

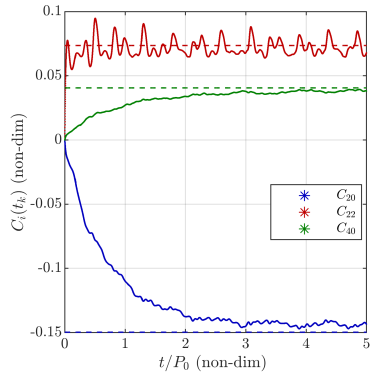
In Fig. 3.18, a time step of 60 seconds is considered. Note that the 2nd degree coefficient converge practically in an exact way, in terms of mean. Higher order coefficients, instead, have less oscillations in time (this is mainly due to the choice of β).

In Fig. 3.19, a time step of 60 seconds is considered. Note that also in that case a good convergence is achieved for the set of estimated coefficients. Note that the convergence is "smoother" with respect to Castalia case, since in that case a $\beta \approx 1.5\beta^*$ is considered.

In Fig. 3.20, again a time step of 60 seconds is considered. Being a fast rotating body, note that the trajectory is perturbed a lot (a) by centrifugal forces. As a consequence, the convergence of the network is not smooth at all, and can present estimation offsets, even for the major harmonics.

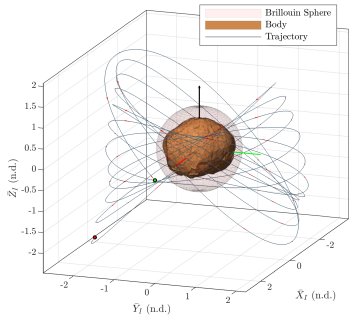


(a) Inertial body-centred frame orbit.

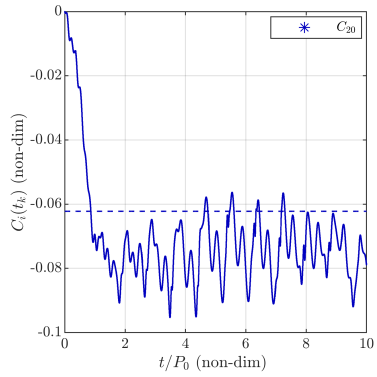


(b) nHNN results.

Figure 3.19: Asteroid Kleopatra results: $a_0 = 2R_{max}$, $i_0 = 135°$ circular orbit.



(a) Inertial body-centred frame orbit.



(b) nHNN results.

Figure 3.20: Phobos case results: $a_0 = 3R_{max}$, $i_0 = 135°$ circular orbit.

3.4.5.2 Binary System Didymos

The case of Didymos binary system is analyzed in order to assess the scalability of the network to a different dynamical environment. According to [67], the dynamical environment can be represented through a polyhedron model for the main body and a ellipsoid model for the moon. The network is implemented here in the un-normalized form associated to the SCR3BP. In this case, Ω_S as well as the mass of the two bodies is assumed to be known so that Eq. 2.25:

$$\mathbf{y} = \ddot{\mathbf{r}} + \boldsymbol{\Omega}_S \times (\boldsymbol{\Omega}_S \times \mathbf{r}) + 2\boldsymbol{\Omega}_S \times \mathbf{r} + \frac{\mu_1}{r_1^3}\mathbf{r}_1 + \frac{\mu_2}{r_2^3}\mathbf{r}_2 \tag{3.67}$$

$$\mathrm{A} = [\mathrm{A}_{\mathrm{main}}(\mathbf{r}_1), \mathrm{A}_{\mathrm{moon}}(\mathbf{r}_2)] \tag{3.68}$$

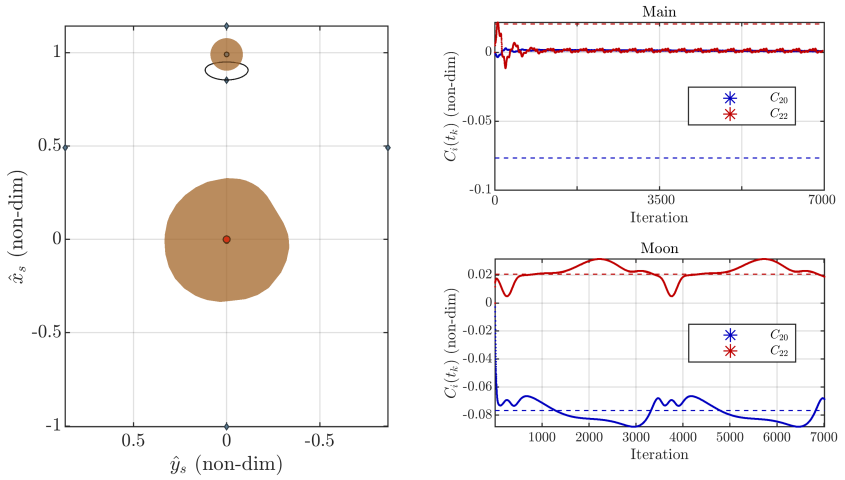


Figure 3.21: Didymos system, Southern Halo.

Giving a LIP form $\mathbf{y} = \mathrm{A} \cdot \mathbf{C}_{\mathrm{ag}}$ where $\mathbf{C}_{\mathrm{ag}} = [\mathbf{C}_{\mathrm{main}}; \mathbf{C}_{\mathrm{moon}}]$. Some orbital families are analysed in order to assess the capability of the network to work in such a perturbed environment. In this work the results for the case of a L1 Halo orbit are presented in Fig. 3.21. Note that the coefficients of Didymoon are reconstructed quite well in mean, however, being the orbit too far from Didymain, the network is not able to estimate any coefficient.

Last, in Fig. 3.21, a time step of 30 seconds is adopted and $\beta = 10^{-7}$. Note that being close to the moon, the convergence of the associated coefficients is fast, in mean. The fact that the orbit is now out of plane helps for the convergence of the moon harmonics only.

3.4.6 Comparison with EKF-based Parameter Identification

The parameter identification problem has been studied in different technical disciplines [68]. One common technique to estimate internal parameters of nonlinear systems is to use an augmentation of the traditional Extended Kalman Filter, under certain observability conditions [68]. The comparison presented hereby focuses on evaluating two approaches both relying on Extended Kaman Filter techniques, in particular:

- the EKF is coupled with the presented HNN. The filter is dedicated to reconstruct the full state of the system, whereas the HNN approximates the unknown spherical harmonics coefficients
- the EKF is used both for estimating the state and the unknown coefficients. Thus, the augmented state of the filter comprises the set of coefficients to be identified.

3.4.6.1 Filter formulation

In order to compare the performance of the HNN approach for estimating spherical harmonics coefficients, an EKF-based estimation algorithm has been developed. The augmented state vector of the EKF is augmented as follows:

$$\mathbf{x} = \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \\ C_{nm} \end{bmatrix} \tag{3.69}$$

where $\mathbf{r}$ and $\mathbf{v}$ are the position and velocity vectors respectively; $\{C_{nm}\}$ is a stacked vector containing the SHE coefficient, whose length depends on the application scenario being p the number of coefficient. The dynamics of the augmented state space resembles the one presented for the HNN-based algorithm, namely:

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{v} \\ \nabla\mathcal{U}\left(\mathbf{r}, C_{nm}\right) + 2\boldsymbol{\Omega} \times \dot{\boldsymbol{r}} + \boldsymbol{\Omega} \times (\boldsymbol{\Omega} \times \mathbf{r}) \\ 0_{nm} \end{bmatrix} \tag{3.70}$$

where it is important to note that the gradient of the potential is dependent on the estimated SHE coefficients. This guarantees system observability for estimating the aforementioned internal parameters. The state transition matrix is approximated using the first order Taylor expansion [1], so that the Jacobian of the dynamics can be constructed as follows:

$$\mathrm{J} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times p} \\ \nabla^2\mathcal{U} & \mathcal{M}_\Omega & \mathrm{A}(\mathbf{r}) \\ \mathbf{0}_{p\times3} & \mathbf{0}_{p\times3} & \mathbf{0}_{p\times p} \end{bmatrix} \tag{3.71}$$

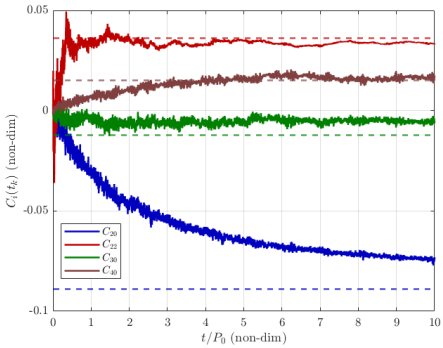
where $\mathcal{M}_\Omega = \frac{\partial\nabla\mathcal{U}}{\partial\mathbf{v}} = 2[\boldsymbol{\Omega}\times]$. For the sake of simplicity, in this work, the EKF measurement equation is assumed to be linear, with the measurement matrix reading:

$$\mathbf{y} = [\mathbf{I}_{6\times6}\ \mathbf{0}_{nm}] \tag{3.72}$$

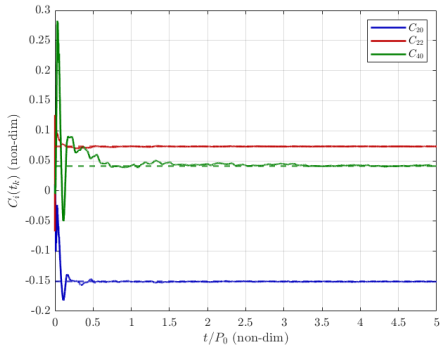
Real application scenarios of an asteroid mission may require more sophisticated measurement function and behavioral model relying on low-observability measurements, as described in [1].

3.4.6.2 Numerical results and comparison

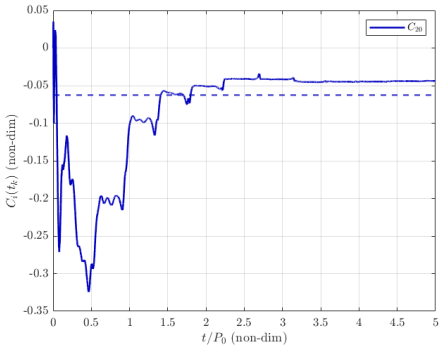
The dynamical environment described in Section 2 is used for numerical simulations. The measurements are generated through propagation of the aforementioned dynamical models. Furthermore, the state measurements are assumed to be perturbed with zero-mean white Gaussian noise. In particular,



(a) Castalia



(b) Case Kleopatra



(c) Phobos

Figure 3.22: SHE coefficients estimation using EKF.

position and velocity are perturbed using $\sigma_r = 10^2\ m$ and $\sigma_v = 10^{-2}\frac{m}{s}$. For the sake of comparison, three test cases have been assessed, namely asteroids Castalia, Kleopatra and the Moon Phobos.

The estimation results are shown in Fig. 3.22. For the parametric identification of gravitational field coefficients, however, being the number of parameters always >1 and usually $\gg 1$, the use of a method out of the EKF can be beneficial from a computational point of view. In fact the computational cost of a filter step do increase at least linearly with the number of states of the augmented vector $\mathbf{x}$. In Fig. 3.23, in fact, a comparison between the two methods is presented considering the case of asteroid Castalia (4 parameters):

- In green, the computational time for a single step of the HNN is reported. The mean $\mu_{\text{hnn}} \sim 15\ \mu$s while the standard deviation $\sigma_{\text{hnn}} \sim 12\ \mu$s.

- In grey, the computational time for a single step of a EKF used for the only state estimation is reported. In this case, the mean $\mu_{\text{hnn}} \sim 600\ \mu$s

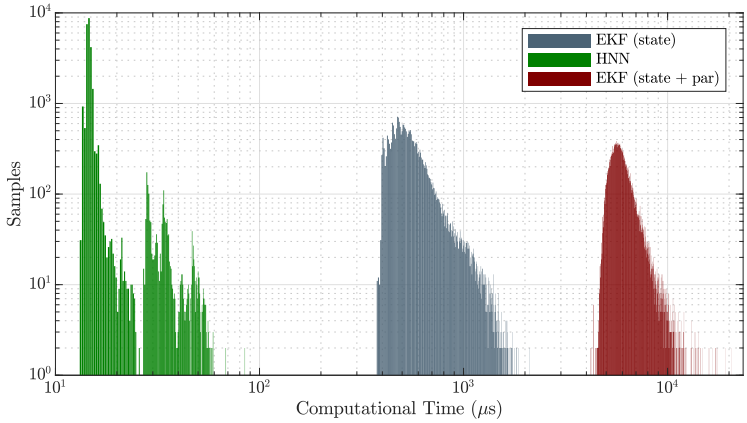


Figure 3.23: Computational time comparison. Note that the HNN step-time is negligible with respect to the EKF for state estimation.

while the standard deviation $\sigma_{\text{hnn}} \sim 386$ μs. The gravitational model used in the EKF in this case is the pure Two-Body Problem.

- In red, the computational time for a single step of a EKF used for both the state and the parameters estimation. In this case, the mean $\mu_{\text{hnn}} \sim 6.25$ ms while the standard deviation $\sigma_{\text{hnn}} \sim 2.15$ ms.

The previous results are computed on a machine with a quad-core, i7-7700, 3 GHz CPU and highlight that the computational time associated to a EKF+HNN in the state & parameters estimation is one order of magnitude smaller than the one associated to an augmented EKF, being beneficial also from a volatile memory point of view.

From the parameters estimation point of view, instead, both the methods are capable to reconstruct the selected Stokes coefficients, as reported in Tab. 3.5. In particular, for asteroid Castalia, the HNN estimation results are presented in Fig. 3.18 while the one associated to the EKF in Fig. 3.22a: in this case the HNN exhibit better convergence properties with respect to the filter that converges slower. It is the opposite for the case of asteroid Kleopatra, Fig. 3.19, Fig. 3.22b. Finally, in the case of Phobos, that is critical for the highly perturbed environment associated to the large centrifugal forces, both methods have troubles in the estimation, giving an offset on the final estimate.

Table 3.5: HNN/EKF results compared for asteroid Castalia. The mean and the standard deviation are computed on the last 5 periods while the integral measure, *int* with the whole set of orbits. *int* is computed as the sum of absolute errors over the number of sample points e.g. $int = \sum_k |e_k|/N$, with $e_k = C_k - \hat{C}_k(t_k)$.

C_{20}						
True value	HNN			EKF		
	mean	std	int	mean	std	int
-0,0890	-0,08723	0,000646	0,0063	-0,0703	0,0035	0,0317
C_{22}						
True value	HNN			EKF		
	mean	std	int	mean	std	int
0,0362	0,0355	0,002782	0,0022	0,0317	0,0012	0,0054
C_{30}						
True value	HNN			EKF		
	mean	std	int	mean	std	int
-0,0124	-0,01266	0,000735	0,0010	-0,0054	0,0013	0,0086
C_{40}						
True value	HNN			EKF		
	mean	std	int	mean	std	int
0,0152	0,01684	0,000546	0,0011	0,0115	0,0012	0,0061

CHAPTER 4

Neural-Aided Guidance & Control

Per chi viaggia in direzione
ostinata e contraria
col suo marchio speciale
di speciale disperazione.

— Fabrizio De André

The methods explained in Chapter 3 served as basis for the synthesis of Guidance, Navigation & Control (GNC) algorithms. In particular, most of the sophisticated planning and control algorithms make use of the system dynamics to tune the response towards the increase of efficiency, short time response or minimization of given quantities (e. g. time, fuel consumption). In particular, in this work, two algorithms are presented. The Neural-Artificial Potential Field is a light and powerful algorithm that encompass a dynamics reconstruction routine to generate the guidance and control law. The collision avoidance constraint is treated efficiently and effectively, as well as the generation of guidance trajectories. Despite delivering promising results in terms of real-time execution and simple reconfiguration, it has two main drawbacks: it shows delicate robustness and it does not entail optimization and predicted

planning, which is critical for space exploration. To solve such shortcomings, at the cost of increasing complexity, an innovative algorithm based on neural dynamics reconstruction and planning has been developed. It is, hereby, called Model-Based Reinforcement Learning but it could be best described by the name Neural-Predictive Control. The optimization of future control action and guidance trajectory is generated based on the reconstructed dynamics.

The Chapter is organized as follows: Section 4.1 describes the Neural-Artificial Potential Field method; Section 4.2 thoroughly discusses the performance of the Neural-Artificial Potential Field method together with its numerical validation; finally, Section 4.3 describes the Model-Based Reinforcement Learning algorithms and Section 4.4 discusses the performance and reports the numerical results.

4.1 Neural-Artificial Potential Field Guidance

The first developed algorithm consists in an autonomous formation reconfiguration GNC algorithm based on Artificial Potential Field. It includes a distributed active collision avoidance based on repulsive potential contribution derived from Cartesian state measurements, suitable for microsatellite applications. A tracking feedback controller, based on Lyapunov theorem, guarantees the artificial potential dynamics to be followed. Additionally, the proposed algorithm exploits the online Radial-Basis Function Neural Network (RBFNN) (cfr. Chapter 3) to reconstruct the disturbances or unmodeled terms in the dynamics to enhance the whole Guidance Navigation & Control (GNC) algorithm. The main features of the algorithm are:

- a fully online algorithm based on Radial-Basis-Function Neural Network (RBFNN) for dynamics reconstruction that can benefit the whole GNC architecture. The refinement of the on-board dynamics improves the navigation performance and guarantees better control accuracy to reach the target state;
- a GNC algorithm that exploits the RBFNN refinement of the on-board dynamics while flying. This work presents a methodology that could potentially be exploited in several different environments. The disturbances and nonlinear terms may be caused by different sources. In this work, a fully nonlinear Cartesian J2-perturbed with relative drag is used for the *ground-truth*, whereas the on-board dynamics used in the GNC is based on ROE dynamics.
- a full GNC algorithm for spacecrafts formation reconfiguration, suitable for micro platforms implementation;
- an extension of the Artificial Potential Field to dynamically control multiple agents (≥ 2) assuring the collision avoidance constraint is respected;

- a collision avoidance procedure that takes as input Cartesian relative measurements and process them in the Relative Orbital Element space.

The distributed guidance algorithm processes locally all the state estimations of the satellite formation members. Each satellite knows the relative position with respect to the rest of the formation. The guidance strategy relies on artificial potential functions designed in the relative orbital elements space in $\mathbb{R}^6$ [38]. The idea is to build a point-wise global potential based on the contribution of attractive and repulsive potential sources, namely the target relative orbits and any other satellite located in close neighboring areas. The attractive potential is directly expressed in terms of relative orbital elements, being a convenient way to express relative orbits geometry. Indeed, a set of relative orbital elements uniquely define one particular formation configuration. On the other hand, the natural way to express the vicinity between two satellites is using the Cartesian distance, expressed in the LVLH reference frame in this particular application. To obtain a uniform expression of the global potential, the Jacobian of the transformation is derived, based on the results presented in Section 2.1.2.1. The output of the guidance, for each satellite, is what we call *guidance state* and indicate as $\delta\chi_g$. The guidance algorithm forces the following dynamics for each satellite i:

$$\dot{\delta\chi_{\mathbf{g}}} = -\nabla\Phi_{glb} \tag{4.1}$$

where Φ_{glb} is the global potential:

$$\nabla\Phi_{glb} = \nabla\Phi_a + \nabla\Phi_r \tag{4.2}$$

where Φ_a is the attractive potential, whereas Φ_r is the repulsive one.

4.1.1 Attractive Potential: Configuration Target

The reconfiguration objective is to drive the satellites to a predefined relative configuration, expressed in terms of relative orbital elements. The set of ROE to be achieved is called *reference state* and indicated as $\delta\chi_r$. The attractive contribution to the global potential is determined as:

$$\Phi_a(\delta\chi) = \frac{1}{2}\xi_a \left\|\delta\chi_g - \delta\chi_r\right\|^2 \tag{4.3}$$

where the parameter ξ_a is a user-defined variable, which can be used to tune the guidance according to the scenario. The gradient in the guidance ROE space is defined as:

$$\nabla_{\delta\chi_g}(\cdot) = \left(\frac{\partial}{\partial\delta a}, \frac{\partial}{\partial\delta\lambda}, \frac{\partial}{\partial\delta e_x}, \frac{\partial}{\partial\delta e_y}, \frac{\partial}{\partial\delta i_x}, \frac{\partial}{\partial\delta i_y}\right)_g \tag{4.4}$$

Consequently, the dynamic contribution to Eq. 4.2 given by the attractive potential is:

$$\nabla_{\delta\chi_g} = \xi_a(\delta\chi_g - \delta\chi_r) \tag{4.5}$$

4.1.2 Repulsive Potential: Active Collision Avoidance

The repulsive potential is useful to calculate the trajectory in presence of other satellites, avoiding collision between agents. As previously stated, to achieve an efficient active collision avoidance maneuver, the potential is best representative in terms of the Cartesian state X in the Hill frame, where the metric distance is defined. Given two satellites, i and j respectively, the repulsive potential to be computed for Eq. 4.2 for satellite i is defined as:

$$\Phi_{r_{ij}} = \begin{cases} \frac{1}{2}\xi_r e^{-\frac{d_{ij}^2}{\eta}} = \frac{1}{2}\xi_r e^{-\frac{\|\mathbf{x_i}-\mathbf{x_j}\|^2}{\eta}} & \text{if } d_{ij} < d_{lim}, \\ 0 & \text{if } d_{ij} > d_{lim} \end{cases} \tag{4.6}$$

where d_{lim} is the threshold distance beyond which the collision maneuver is not required. The state of the relative position of the spacecrafts is known, thus it is possible to calculate the distance vector as the difference between $\mathbf{X_i} - \mathbf{X_j}$. The gradient of the potential is calculated using the chain-rule, which involves the coordinate transformation from Cartesian state $\mathbf{X}$ to ROE $\delta\chi$:

$$\nabla_{\delta\chi_g}\Phi_{r_{ij}} = \nabla_X \Phi_{r_{ij}} \cdot J_{\delta\chi}^X \tag{4.7}$$

where $\mathbf{J}_{\delta\chi}^{\mathbf{X}}$ is the Jacobian of the coordinate transformation, derived in section 2.1.2.1. The gradient in the Cartesian space is defined as:

$$\nabla_X(\cdot) = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right) \tag{4.8}$$

Hence, the gradient of the repulsive potential between agents i and j, below the threshold, can be expressed as:

$$\nabla_{\delta\chi_g}\Phi_{r_{ij}} = -\frac{\xi_r}{\eta} e^{-\frac{d_{ij}^2}{\eta}} \cdot (\mathbf{X_i} - \mathbf{X_j}) \cdot J_{\delta\chi}^X \tag{4.9}$$

The repulsive potential takes into account all the mutual distances between the formation agents; coherently, the repulsive contribution to the global potential for satellite i is the summation of the mutual repulsive potential between satellite i and all the other satellites:

$$\Phi_r = \sum_{j \neq i}^{n} \Phi_{r_{ij}} \tag{4.10}$$

where n is the number of spacecrafts in the formation.

4.1.3 Natural Dynamics: Action Smoothing

The described derivation of the global artificial potential does not take into account the natural dynamics of the system, even though the artificial potential is derived in the ROE space. By including the natural dynamics of Eq. 2.8 into Eq. 4.1, the guidance law is smoothed with respect to the global potential. Hence, the terms expressing the natural dynamics can be integrated to smooth the effect of the guidance law resulting in:

$$\dot{\delta\chi_{\mathbf{g}}} = -\nabla\Phi_{glb} + (\mathbf{A_k} + \mathbf{A_{J2}}) \cdot \delta\chi \tag{4.11}$$

Thus, the guidance dynamics evolves according to Eq. 4.11. Such dynamics is used to generate the desired ROE state, which is subsequently used to determine the control action to steer the actual trajectory of the spacecraft. The natural dynamics becomes dominant when the spacecraft is very close to the target configuration. This eliminates the oscillatory chattering caused by the artificial potential when very close to the target state.

4.1.4 Neural Control

The output of the guidance algorithm is a set of ROE, which may differ from the target reference ones. To guarantee that the forced guidance dynamics in Eq. 4.11 is followed, a feedback control law is employed. The control law is derived using the Lyapunov stability theorem. In the distributed architecture, each spacecraft processes the guidance law, including state of the other agents. The reference signal to track is calculated by the guidance algorithm and follows the dynamics in Eq. 4.11. The current error between the desired guidance state and true state for each satellite is:

$$\mathbf{e}_{\delta\chi} = \delta\chi_g - \delta\chi \tag{4.12}$$

its temporal evolution can be described as:

$$\dot{\mathbf{e}}_{\delta\chi} = \dot{\delta\chi_g} - \dot{\delta\chi} = -\Big(\nabla\Phi_a + \nabla\Phi_r + A(\nu)\delta\chi\Big) - \Big(A(\nu)\delta\chi + \gamma(\delta\chi) + B\mathbf{u}\Big) \tag{4.13}$$

If we introduce the following positive semi-definite Lyapunov function:

$$V = \frac{1}{2}\mathbf{e}_{\delta\chi}^T\mathbf{e}_{\delta\chi} \rightarrow \dot{V} = \mathbf{e}_{\delta\chi}^T\dot{\mathbf{e}}_{\delta\chi} \tag{4.14}$$

$$\dot{V} = \Big(\delta\chi_g - \delta\chi\Big)^T \cdot \Big[-\Big(\nabla\Phi_a + \nabla\Phi_r + \gamma(\delta\chi) + B\mathbf{u}\Big)\Big] \tag{4.15}$$

The control term can be solved to make the derivative of the Lyapunov function negative. The above strategy yields the following control law:

$$\mathbf{u} = \mathbf{B}^{-1}\Big[\Big(\delta\chi_g - \delta\chi\Big) - \Big(\nabla\Phi_a + \nabla\Phi_r\Big) - \gamma(\delta\chi)\Big] \tag{4.16}$$

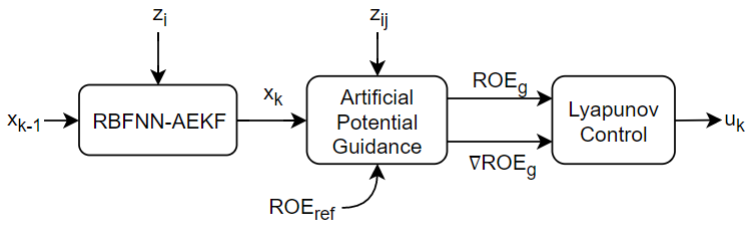


Figure 4.1: GNC architecture overview.

In this way the derivative of the Lyapunov function is negative semidefinite, vanishing only when $\delta\chi = \delta\chi_r$, which is within the validity of the Lyapunov theorem. This approach is similar to the one adopted by [38], with the exception of including the gradient of artificial potential and the RBFNN output in the control law. By including the gradient of the potential, which forces the dynamics, the control law calculates the action taking into account the derivative of the $\delta\chi_g$ determined by the guidance algorithm. The GNC architecture is reported schematically in Fig. 4.1.

Algorithm 2 Neural Distributed GNC for satellite i

1: Initialize RBFNN network $\mathcal{N}_{RBF}$
2: **while** $\delta\chi \neq \delta\chi_r$ **do**
3: Observe state $\delta\chi_i$ through relative measurements
4: Feed $\delta\chi$ to $\mathcal{N}$ (Eq. 3.35)
5: Estimate system state through AEKF (Eq. 3.41)
6: Generate guidance path using APF approach (Eq. 4.1)
7: Execute RBFNN-aided Lyapunov control (Eq. 4.16)
8: **end while**

4.2 Neural-Artificial Potential Field Performance

Three reconfiguration scenarios are treated in the following section involving a formation of four satellites orbiting the Earth. The scenarios are selected to be representative of different reconfiguration maneuvers that may occur during a formation flying mission. The simulations are provided to testify the efficiency of the proposed Neural-Artificial Potential Field reconfiguration algorithm. The system model parameters are chosen and described in a previous work by [11]. The RBFNN neurons are Gaussian functions, whose centers are initialized randomly. The number of neurons is 60. Such number is the results of a trade-off between RBFNN accuracy and complexity. Tab. 4.17 reports the orbital parameter of the reference orbits. The reference orbits are also used to generate relative measurements by adding a fictitious noise, representative of realistic sensors uncertainty (e.g. ranging, optical or Doppler). In particular, the noise level associated is described by a Gaussian distribution

Table 4.1: Reference orbits for numerical simulations

	$\boldsymbol{a}$ [km]	$\boldsymbol{e}$ [-]	$\boldsymbol{i}$ [^{o}]	$\boldsymbol{\omega}$ [^{o}]	$\boldsymbol{\Omega}$ [^{o}]
R1	7975	0.1	10	0	0
R2	10254	0.3	10	0	0

with standard deviation $\sigma_{pos} = 10^{-1}m$ and $\sigma_{vel} = 10^{-3}m/s$ for position and velocity respectively. The Gaussian noise is used to corrupt the measurements to be representative of actual sensors output. The noise affects only the EKF update. The neural network here is used to complement the dynamics with an additional acceleration term, which contains all the nonlinearities or unmodeled terms that are missing in the dynamical expression used in the EKF propagation. Spacecraft *S/C 1* evolution is not reported for analysis as it is assumed to be controlled and to follow the reference absolute orbit. In other words, it is assumed that the absolute navigation and control guarantees the nominal reference orbit to be tracked by the *S/C 1*. This is not a major limitation since the algorithm deals with relative motion with respect to the *S/C 1*. *S/C 1* could also be thought as the virtual center of the defined LVLH frame. The natural dynamics is described by a nonlinear J2-perturbed model [40] expanded with relative drag acceleration (considering identical satellites ballistic coefficient $B = 1.2$). The neural-aided algorithm is beneficial when the dynamics is highly uncertain. Nevertheless, using a *ground-truth* dynamics comprising J2-perturbation, relative drag and nonlinearities is a simple preliminary scenario to demonstrate the effectiveness of the method. Given that the *ground-truth* model is Cartesian and the on-board dynamics is based on ROE, it is difficult to compare the disturbance terms. For Cartesian models, the validation was performed in a previous work by the author [40]. Nevertheless, a dedicated simulation has been performed to show and report the reconstruction capabilities of the neural network in the next section.

4.2.1 Planar to Along-track

The Planar to Along-track (ALO) reconfiguration changes the relative position of the spacecraft such that they no longer belong to a symmetric configuration around the reference orbit but they are placed with a certain along-track offset. Fig. 4.2 shows the controlled neural reconfiguration. The initial and final state, expressed in relative orbital elements, are reported in Tab. 4.2. The reconfiguration is successful and the neural enhanced GNC architecture converges to stable estimation of the unmodeled terms (relative drag and nonlinearities), given that no unstable behavior is experienced, see Section 3.3.5.2. Fig. 4.3 shows the control effort and the navigation results in terms of estimation accuracy. The repulsive potential aided by the RBFNN network guarantees that

Table 4.2: Relative orbital elements of each spacecraft in the ALO reconfiguration

$a\delta\chi^a[m]$	S/C 1	S/C 2	S/C 3	S/C 4
$a\delta a$	$0 \to 0$	$0 \to 0$	$0 \to 0$	$0 \to 0$
$a\delta\lambda$	$0 \to 0$	$0 \to 300$	$0 \to 400$	$0 \to 500$
$a\delta e_x$	$0 \to 0$	$2000 \to 0$	$4000 \to 0$	$6000 \to 0$
$a\delta e_y$	$0 \to 0$	$300 \to 0$	$600 \to 0$	$900 \to 0$
$a\delta i_x$	$0 \to 0$	$500 \to 0$	$500 \to 0$	$500 \to 0$
$a\delta i_y$	$0 \to 0$	$0 \to 0$	$0 \to 0$	$0 \to 0$

[a] dimensional using the semimajor axis of the reference orbit

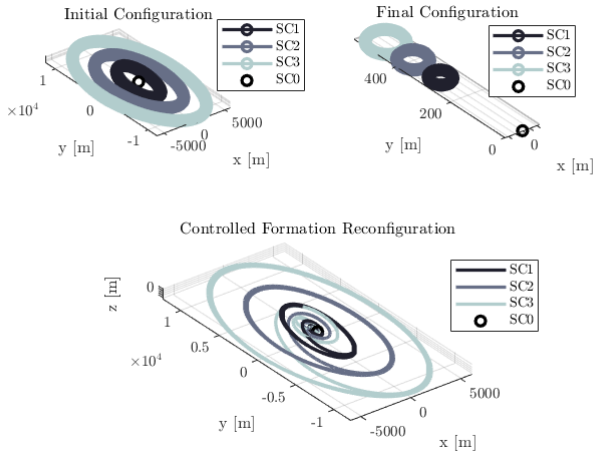


Figure 4.2: Planar to Along-track (ALO) neural reconfiguration.

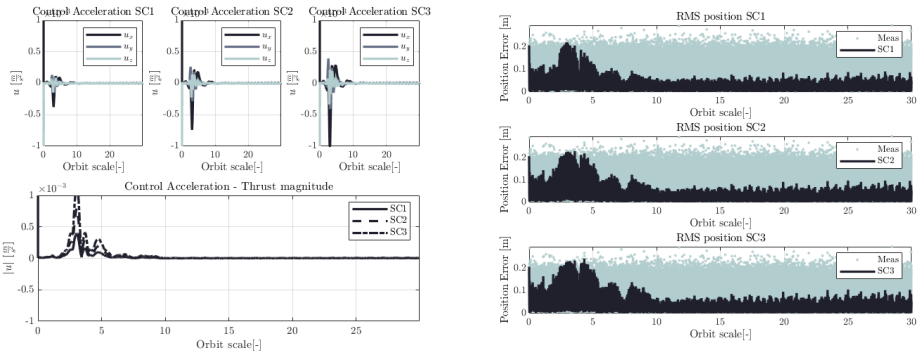


Figure 4.3: Control effort and estimation accuracy for the ALO neural reconfiguration.

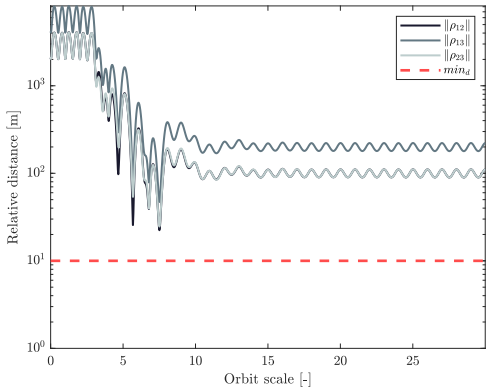


Figure 4.4: Relative distances between the formation spacecraft for the ALO scenario. The dotted line represents the minimum safe distance set for the simulations.

the minimum distance between the agents is respected, as shown in Fig. 4.4. The relative maneuver initiates after one orbital period. The along-track reconfiguration is the most demanding in terms of collision avoidance, as the orbital state are very similar in the final configuration, roughly 100 m apart.

4.2.2 Planar Synthetic Aperture Variation

Formation flying missions are foreseen to be employed in configuration that allow synthetic aperture of the distributed system. A Planar Synthetic Aperture Variation (SAV) is a reconfiguration maneuver in which the equivalent instrument diameter is varied. Fig. 4.5 shows the controlled neural reconfiguration. The initial and final state, expressed in relative orbital elements, are reported in Tab. 4.3.

Table 4.3: Relative orbital elements of each spacecraft in the SAV reconfiguration

$a\delta\chi^a[m]$	S/C 1	S/C 2	S/C 3	S/C 4
$a\delta a$	$0 \to 0$	$0 \to 0$	$0 \to 0$	$0 \to 0$
$a\delta\lambda$	$0 \to 0$	$0 \to 0$	$0 \to 0$	$0 \to 0$
$a\delta e_x$	$0 \to 0$	$100 \to 1000$	$300 \to 3000$	$600 \to 6000$
$a\delta e_y$	$0 \to 0$	$100 \to 1000$	$300 \to 3000$	$600 \to 6000$
$a\delta i_x$	$0 \to 0$	$0 \to 0$	$0 \to 0$	$0 \to 0$
$a\delta i_y$	$0 \to 0$	$0 \to 0$	$0 \to 0$	$0 \to 0$

[a] dimensional using the semimajor axis of the reference orbit

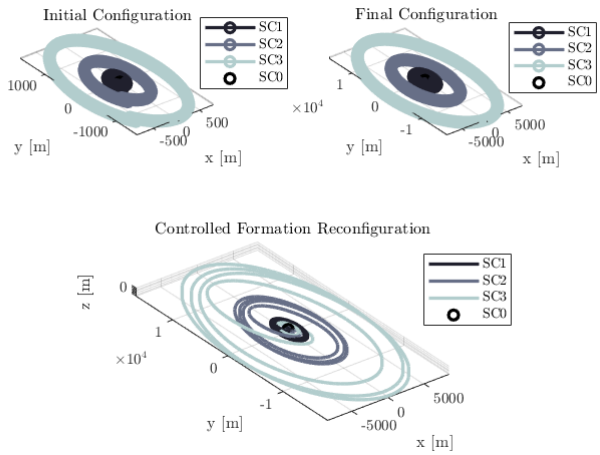


Figure 4.5: Planar Synthetic Aperture Variation (SAV) neural reconfiguration.

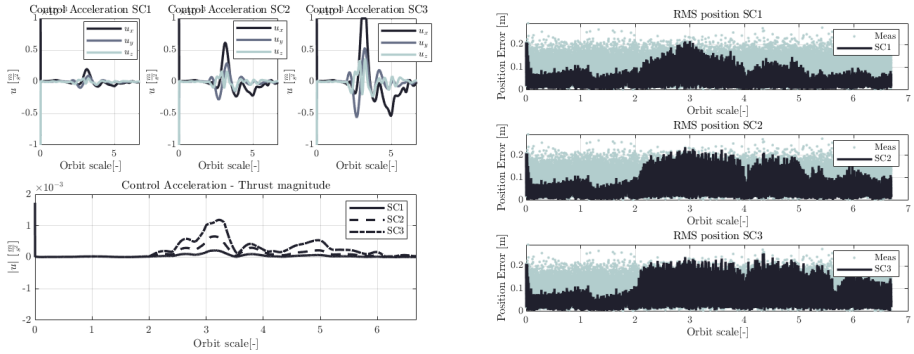


Figure 4.6: Control effort and estimation accuracy for the SAV neural reconfiguration.

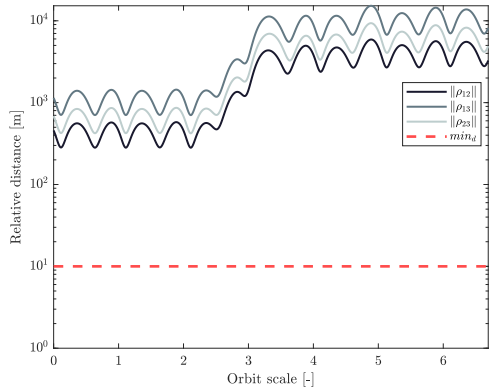


Figure 4.7: Relative distances between the formation spacecraft for the SAV scenario. The dotted line represents the minimum safe distance set for the simulations.

The reconfiguration is successful and the neural enhanced GNC architecture converges to stable estimation of the unmodeled terms, given that no unstable behavior is experienced, see Section 3.3.5.2. Fig. 4.6 shows the control effort and the navigation results in terms of estimation accuracy. The repulsive potential aided by the RBFNN network guarantees that the minimum distance between the agents is respected, as shown in Fig. 4.7. The relative maneuver initiates after one orbital period.

4.2.3 Relative Plane Change

The Relative Plane Change (RPC) reconfiguration is essentially an inversion of the relative inclination vector. The component δi_x, see Eq. 2.5, is actually the algebraic difference of the spacecraft orbital inclination. Hence, the

Table 4.4: Relative orbital elements of each spacecraft in the RPC reconfiguration

$a\delta\chi^a[m]$	S/C 1	S/C 2	S/C 3	S/C 4
$a\delta a$	$0 \to 0$	$0 \to 0$	$0 \to 0$	$0 \to 0$
$a\delta\lambda$	$0 \to 0$	$0 \to 0$	$0 \to 0$	$0 \to 0$
$a\delta e_x$	$0 \to 0$	$200 \to 200$	$400 \to 400$	$600 \to 600$
$a\delta e_y$	$0 \to 0$	$300 \to 300$	$600 \to 600$	$900 \to 900$
$a\delta i_x$	$0 \to 0$	$500 \to -500$	$300 \to -500$	$500 \to -500$
$a\delta i_y$	$0 \to 0$	$0 \to 0$	$0 \to 0$	$0 \to 0$

[a] dimensional using the semimajor axis of the reference orbit

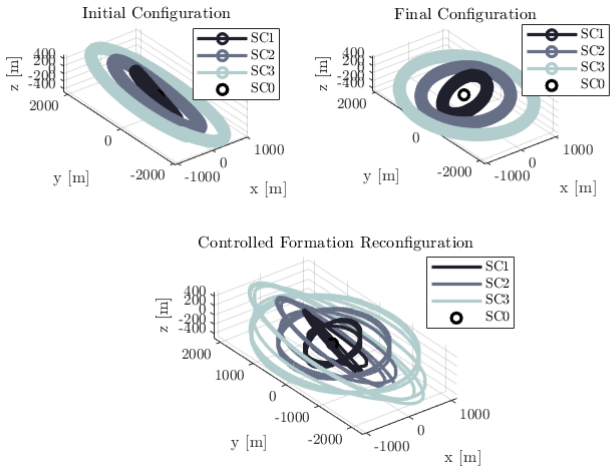


Figure 4.8: Relative Plane Change (RPC) neural reconfiguration.

reconfiguration shown in this section is equivalent to an inclination change maneuver. Such reconfiguration has been chosen because of its complexity on control, involving along-radial-cross track control. Fig. 4.8 shows the controlled neural reconfiguration. The initial and final state, expressed in relative orbital elements, are reported in Tab. 4.4.

The reconfiguration is successful and the neural enhanced GNC architecture converges to stable estimation of the unmodeled terms, given that no unstable behavior is experienced, see Section 3.3.5.2. Fig. 4.9 shows the control effort and the navigation results in terms of estimation accuracy. The repulsive potential aided by the RBFNN network guarantees that the minimum distance between the agents is respected, as shown in Fig. 4.10. The relative maneuver initiates after one orbital period.

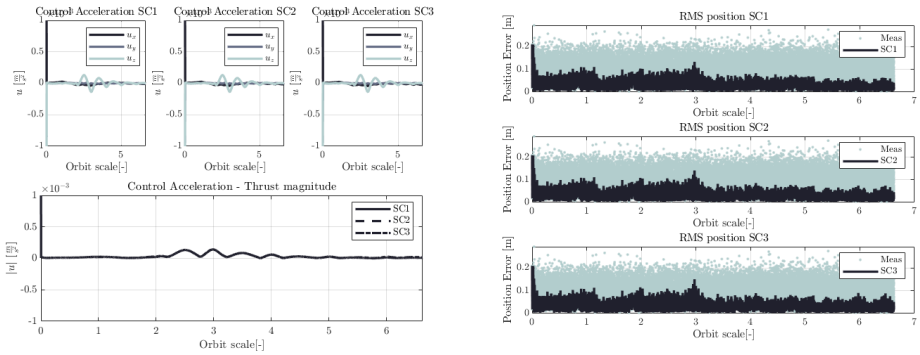


Figure 4.9: Control effort and estimation accuracy for the RPC neural reconfiguration.

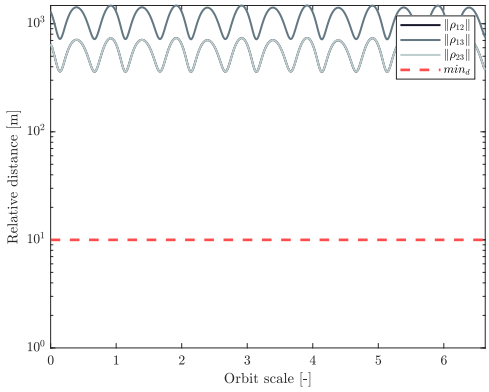


Figure 4.10: Relative distances between the formation spacecraft for the RPC scenario. The dotted line represents the minimum safe distance set for the simulations.

4.2.4 Formation Position Swap

The reconfiguration is specifically designed to include a collision avoidance maneuvers for the sake of demonstration. Basically, the agents 2, 3, 4 are asked to swap relative orbits with respect to Spacecraft 1, once again fixed along the reference orbit. The threshold d_{lim} in Eq. 4.6 is set to 50 m and an exit condition for the simulation due to collisions is set to 10 m. In particular, the maneuver is:

$$\delta\chi_{ref,2} = \delta\chi_{0,3},\ \delta\chi_{ref,3} = \delta\chi_{0,4},\ \delta\chi_{ref,4} = \delta\chi_{0,2}, \tag{4.17}$$

Fig. 4.11 shows the controlled neural reconfiguration. The initial and final state, expressed in relative orbital elements, are reported in Tab. 4.5.

Table 4.5: Relative orbital elements of each spacecraft in the PS reconfiguration

$a\delta\chi^a[m]$	S/C 1	S/C 2	S/C 3	S/C 4
$a\delta a$	$0 \to 0$	$0 \to 0$	$0 \to 0$	$0 \to 0$
$a\delta\lambda$	$0 \to 0$	$0 \to 0$	$0 \to 0$	$0 \to 0$
$a\delta e_x$	$0 \to 0$	$200 \to 400$	$400 \to 600$	$600 \to 200$
$a\delta e_y$	$0 \to 0$	$300 \to 600$	$600 \to 900$	$900 \to 300$
$a\delta i_x$	$0 \to 0$	$500 \to 300$	$300 \to 100$	$100 \to 500$
$a\delta i_y$	$0 \to 0$	$0 \to 0$	$0 \to 0$	$0 \to 0$

[a] dimensional using the semimajor axis of the reference orbit

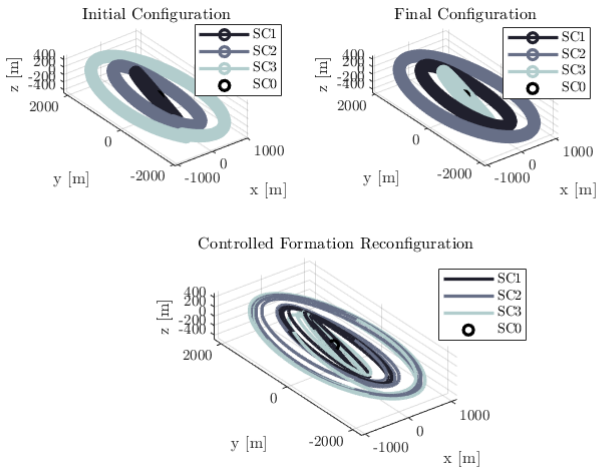


Figure 4.11: Position Swap (PS) neural reconfiguration.

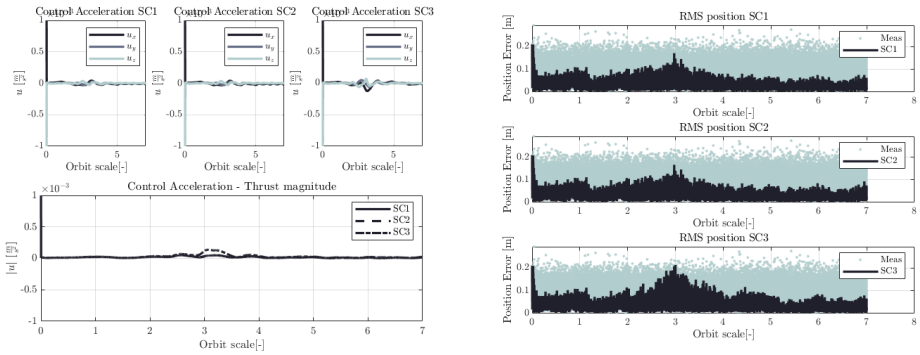


Figure 4.12: Control effort and estimation accuracy for the PS neural reconfiguration.

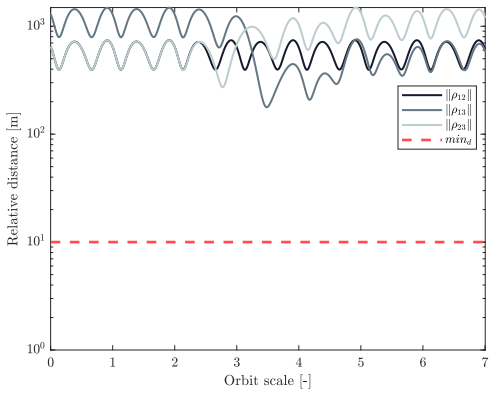


Figure 4.13: Relative distances between the formation spacecraft for the PS scenario. The dotted line represents the minimum safe distance set for the simulations.

The reconfiguration is successful and the neural enhanced GNC architecture converges to stable estimation of the unmodeled terms, given that no unstable behavior is experienced, see Section 3.3.5.2. Fig. 4.12 shows the control effort and the navigation results in terms of estimation accuracy. The repulsive potential aided by the RBFNN network guarantees that the minimum distance between the agents is respected, as shown in Fig. 4.13. The relative maneuver initiates after one orbital period.

4.2.5 Comparison

The Neural-aided algorithm for reconfiguration has been compared with the standard APF algorithm presented in [38] and extended in [11]. Tab. 4.6, 4.7 and 4.8 present the results for the relevant reconfigurations in low-eccentricity reference orbits. In particular, Tab. 4.6 reports the ΔV effort of the Neural-Lyapunov control and the standard Lyapunov control presented in Section 4.1.4. The control effort is generally higher in the case of neural control. This may be due to the fact that during the initial phase of the reconfiguration, the main learning process of the network occurs [40]. In this phase the controller is affected by the non-converged term $\gamma(\delta\chi)$. Nevertheless, as reported in Tab. 4.8 and 4.9, the achieved accuracy of the final configuration using the neural controller is superior, which is linked to the higher required Δv. The dynamics reconstruction based on RBFNN benefits the navigation also. The position estimation mean error is always lower when RBFNN-AEKF is used. The dynamical model refinement allows the filter to generate a more precise *a-priori* estimation, which is then corrected with the same measurements in both methods. The difference in estimation error increases as the nonlinearities become more relevant in the dynamics. Indeed, for high-eccentricity reference orbits, the enhancement in state estimation using RBFNN-AEKF is remarked. Tab. 4.10, 4.11, 4.12 and 4.13 present the results for the analyzed reconfigurations in high-eccentricity reference orbits. The NNAPF algorithm outperforms the standard APF, in terms of accuracy and navigation estimate. The ALO reconfiguration is challenging due to the involved relative distances and the limited maximum thrust available. However, also in this case, the NNAPF strategy delivers better final configuration in terms of accuracy. The high-eccentricity scenario is more demanding since nonlinearities are prominent. The accuracy is poorer because both the controller struggle to handle such disturbance. Nevertheless, the RBFNN achieves a better accuracy (although not satisfactory for ALO) with respect to the traditional APF algorithm. In addition, it is important to remark that the RBFNN can be tuned (number of parameters, weights, learning function) to improve the performance, whereas the APF is less flexible.

4.2.5.1 Highly Perturbed Environment

The comparison presented in Section 4.2.5 showed how the NNAPF algorithm guarantees a more accurate navigation and control, at the cost of slightly higher Δv. The results show that also the legacy algorithm was able to bring the relative maneuvers to completion. This may be due to the fact the the disturbance terms included in the *ground-truth* relative dynamics are still handled by the APF control. In order to showcase the quality of the NNAPF algorithm, a set of numerical simulations have been performed adding fictitious perturbations to the *ground-truth* dynamics to challenge both algorithms. In

Table 4.6: Comparison of control effort between standard APF reconfiguration algorithm and the proposed NNAPF. Low-eccentricity scenario.

Low-e	**NNAPF Control**			**APF Control**		
Scenario	$\Delta V_1[\frac{m}{s}]$	$\Delta V_2[\frac{m}{s}]$	$\Delta V_3[\frac{m}{s}]$	$\Delta V_1[\frac{m}{s}]$	$\Delta V_2[\frac{m}{s}]$	$\Delta V_3[\frac{m}{s}]$
ALO	3.30	6.09	9.04	3.10	5.89	8.83
SAV	1.67	5.02	10.25	1.67	5.04	10.30
RPC	1.13	1.14	1.17	1.10	1.11	1.13
PS	0.57	0.65	1.03	0.54	0.61	0.93

Table 4.7: Comparison of navigation accuracy between standard APF reconfiguration algorithm and the proposed NNAPF. Low-eccentricity scenario.

Low-e	**NNAPF Navigation**			**APF Navigation**		
Scenario	σ_1 [m]	σ_2 [m]	σ_3 [m]	σ_1 [m]	σ_2 [m]	σ_3 [m]
ALO	0.02	0.03	0.03	0.03	0.04	0.04
SAV	0.04	0.05	0.06	0.05	0.06	0.07
RPC	0.03	0.03	0.03	0.03	0.04	0.04
PS	0.03	0.03	0.03	0.04	0.04	0.05

Table 4.8: Comparison of target configuration accuracy between standard APF reconfiguration algorithm and the proposed NNAPF. Low-eccentricity scenario.

Low-e	**Neural Accuracy**[%]			**Accuracy**[%]		
Scenario	$\frac{\lvert\Delta\delta\chi\rvert}{\lvert\delta\chi_{ref}\rvert}_1$	$\frac{\lvert\Delta\delta\chi\rvert}{\lvert\delta\chi_{ref}\rvert}_2$	$\frac{\lvert\Delta\delta\chi\rvert}{\lvert\delta\chi_{ref}\rvert}_3$	$\frac{\lvert\Delta\delta\chi\rvert}{\lvert\delta\chi_{ref}\rvert}_1$	$\frac{\lvert\Delta\delta\chi\rvert}{\lvert\delta\chi_{ref}\rvert}_2$	$\frac{\lvert\Delta\delta\chi\rvert}{\lvert\delta\chi_{ref}\rvert}_3$
ALO	1.80	1.80	1.81	1.93	1.89	1.86
SAV	5.11	5.08	4.76	5.11	5.02	4.76
RPC	5.09	4.85	4.77	5.12	4.86	4.78
PS	3.90	3.50	5.30	4.20	3.80	5.30

Table 4.9: Norm of the Relative Orbital Elements error (dimensionless) with respect to the target ROE state. Low-eccentricity scenario.

Low-e	**Neural Accuracy**			**Accuracy**		
Scenario	$\lvert\Delta\delta\chi\rvert_1$	$\lvert\Delta\delta\chi\rvert_2$	$\lvert\Delta\delta\chi\rvert_3$	$\lvert\Delta\delta\chi\rvert_1$	$\lvert\Delta\delta\chi\rvert_2$	$\lvert\Delta\delta\chi\rvert_3$
ALO	$6.8 \cdot 10^{-5}$	$9.0 \cdot 10^{-5}$	$1.1 \cdot 10^{-4}$	$7.1 \cdot 10^{-5}$	$9.5 \cdot 10^{-5}$	$1.2 \cdot 10^{-4}$
SAV	$9.1 \cdot 10^{-4}$	$2.7 \cdot 10^{-3}$	$5.2 \cdot 10^{-3}$	$9.1 \cdot 10^{-4}$	$2.7 \cdot 10^{-3}$	$5.2 \cdot 10^{-3}$
RPC	$3.9 \cdot 10^{-4}$	$5.3 \cdot 10^{-4}$	$7.1 \cdot 10^{-4}$	$3.9 \cdot 10^{-4}$	$5.4 \cdot 10^{-4}$	$7.1 \cdot 10^{-4}$
PS	$3.8 \cdot 10^{-4}$	$4.8 \cdot 10^{-4}$	$4.1 \cdot 10^{-4}$	$4.1 \cdot 10^{-4}$	$5.2 \cdot 10^{-4}$	$4.1 \cdot 10^{-4}$

Table 4.10: Comparison of control effort between standard APF reconfiguration algorithm and the proposed NNAPF. High-eccentricity scenario.

High-e	**NNAPF Control**			**APF Control**		
Scenario	$\Delta V_1[\frac{m}{s}]$	$\Delta V_2[\frac{m}{s}]$	$\Delta V_3[\frac{m}{s}]$	$\Delta V_1[\frac{m}{s}]$	$\Delta V_2[\frac{m}{s}]$	$\Delta V_3[\frac{m}{s}]$
ALO	9.78	23.77	54.50	13.58	28.48	54.43
RPC	0.72	0.76	0.84	0.93	0.96	1.03
PS	10.50	15.80	7.75	10.23	15.51	9.39

Table 4.11: Comparison of navigation accuracy between standard APF reconfiguration algorithm and the proposed NNAPF. High-eccentricity scenario.

High-e	**NNAPF Navigation**			**APF Navigation**		
Scenario	σ_1 [m]	σ_2 [m]	σ_3 [m]	σ_1 [m]	σ_2 [m]	σ_3 [m]
ALO	0.05	0.06	0.07	0.06	0.08	0.08
RPC	0.04	0.04	0.04	0.05	0.05	0.05
PS	0.05	0.05	0.05	0.07	0.07	0.07

Table 4.12: Comparison of target configuration accuracy between standard APF reconfiguration algorithm and the proposed NNAPF. High-eccentricity scenario.

High-e	**Neural Accuracy**			**Accuracy** [%]		
Scenario	$\frac{\lvert\Delta\delta\chi\rvert}{\lvert\delta\chi_{ref}\rvert}_1$	$\frac{\lvert\Delta\delta\chi\rvert}{\lvert\delta\chi_{ref}\rvert}_2$	$\frac{\lvert\Delta\delta\chi\rvert}{\lvert\delta\chi_{ref}\rvert}_3$	$\frac{\lvert\Delta\delta\chi\rvert}{\lvert\delta\chi_{ref}\rvert}_1$	$\frac{\lvert\Delta\delta\chi\rvert}{\lvert\delta\chi_{ref}\rvert}_2$	$\frac{\lvert\Delta\delta\chi\rvert}{\lvert\delta\chi_{ref}\rvert}_3$
ALO	45.18	75.56	140.50	186.06	180.56	197.07
RPC	3.20	4.60	5.50	5.45	3.00	2.50
PS	26.30	47.80	25.28	26.80	47.56	38.88

Table 4.13: Norm of the Relative Orbital Elements error (dimensionless) with respect to the target ROE state. High-eccentricity scenario.

High-e	**Neural Accuracy**			**Accuracy**		
Scenario	$\lvert\Delta\delta\chi\rvert_1$	$\lvert\Delta\delta\chi\rvert_2$	$\lvert\Delta\delta\chi\rvert_3$	$\lvert\Delta\delta\chi\rvert_1$	$\lvert\Delta\delta\chi\rvert_2$	$\lvert\Delta\delta\chi\rvert_3$
ALO	$1.7 \cdot 10^{-3}$	$3.8 \cdot 10^{-3}$	$8.8 \cdot 10^{-3}$	$6.8 \cdot 10^{-3}$	$9.1 \cdot 10^{-3}$	$1.2 \cdot 10^{-2}$
RPC	$2.5 \cdot 10^{-4}$	$5.1 \cdot 10^{-4}$	$8.2 \cdot 10^{-4}$	$4.2 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$	$3.7 \cdot 10^{-4}$
PS	$2.6 \cdot 10^{-3}$	$6.5 \cdot 10^{-3}$	$1.9 \cdot 10^{-3}$	$2.6 \cdot 10^{-3}$	$6.5 \cdot 10^{-3}$	$3.0 \cdot 10^{-3}$

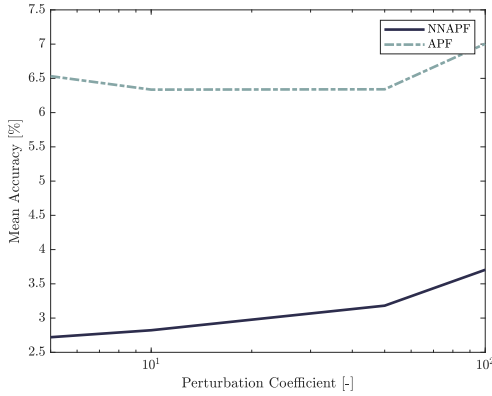


Figure 4.14: Mean accuracy of NNAPF and APF in highly perturbed environment. The perturbation coefficient is the multiplicative term of J_2 perturbation to generate the fictitious disturbance.

particular, the *perturbation coefficient* k_{pert} is introduced. The *perturbation coefficient* k_{pert} is a multiplicative term used to increase the J_2 perturbation to generate the fictitious disturbance:

$$\mathbf{F} = k_{pert} \cdot \mathbf{F}_{J_2} \tag{4.18}$$

Figure 4.14 reports the mean accuracy of the final formation configuration for different values of k_pert. The results are averaged over the scenarios and over the satellites. The results show that a consistent increase in performance is achieved by the NNAPF when perturbations become more dominant. As already noted in Section 4.2.5, the better NNAPF accuracy is achieved at the cost of slightly higher Δv. The increase in Δv is within the range $[0.5 - 1.0]\frac{m}{s}$ in all the simulations.

4.3 Model-based Reinforcement Learning for Trajectory Planning

The Neural Artificial Potential Field algorithm provided promising results, significantly enhanced by the integration of a Artificial Neural Network. Nevertheless, two main drawbacks were identified during its validation:

- the NNAPF may lead to instability due to the active-reactive nature of the algorithm. Nevertheless, user-defined parameters needs to be tuned, tailored to the scenarios, in order to deliver acceptable maneuvers and control actions.
- the NNAPF does not exploit in a predictive manner the knowledge of the dynamics. It is a real-time algorithm with no planning capabilities. This may lead to excessive Δv, being intrinsically sub-optimal.

For these reasons, the focus has been shifted towards the development of a new innovative technique, aimed at resolving the mentioned drawbacks while keeping the strong adaptivity and flexibility of the previous algorithm. Reinforcement learning is a machine learning technique employed to determine how an agent takes actions in the surrounding environment in order to maximize a given reward. One major reason preventing reinforcement learning from being employed in space applications has been its implementation as a model-free algorithm. Model-free approaches rely on the possibility of *exploring* the surrounding space to learn optimal policies. On the other hand, the deterministic orbital environment favors the integration of dynamical models in the path-planning algorithms. Nevertheless they could be unknown (e.g. small bodies) or significantly reshaped to be used on-board, neglecting nonlinearities or perturbation effects. For these reasons, the proposed Model-Based Reinforcement Learning (MBRL) is developed for controlling a formation configuration and generating trajectories for distributed reconfigurations. This approach enables autonomous quasi-optimal reconfiguration in unknown or unmodeled environments as well as fuel-efficient control strategies for formation maintenance, leveraging the incremental knowledge of the environment. In general, the online neural-reconstruction yields an outstanding flexibility to the spacecraft, which updates its representation of the environment to enhance its guidance according to the variable dynamical environment it may encounter. This feature avoids the failure of classical Model Predictive Control-like algorithm relying on a fixed analytical model, but at the same time it allows the algorithm to generate guidance and control strategy leveraging the learned natural dynamics. Moreover, in a distributed architecture no information is globally available on the planning of each agent with respect to the formation, therefore each spacecraft need to predict future maneuvers of potentially colliding agents. Usually, the collision avoidance constraint is enforced by assuming the formation to evolve naturally or along predefined trajectories, which are known by the system [69][70]. Such assumption is quite restricting

when dealing with systems that implement a distributed GNC architecture. The consequence is that the maneuvers may be executed by one spacecraft at a time, preventing the formation to evolve simultaneously. The latter feature may be beneficial when coordinated reconfigurations are required, e.g. to keep a constant synthetic aperture of the formation. Two novel approaches are here presented to solve the shortcoming and enable the formation to maneuver safely and simultaneously, namely Inverse Reinforcement Learning and Long Short-Term Memory network trajectory forecasting. Inverse Reinforcement Learning (IRL) is employed for trajectory prediction of neighboring satellites. Such method was originally developed for imitation learning [71], and then extended to learn demonstrated behaviors and the cost function underlying them [72]. Concerning space applications, Linares [73] proposed an approach based on IRL to determine the behaviour of space objects. The second proposed method is based on LSTM recurrent neural networks, which is trained sequentially to predict the temporal behavior of the observed states. The justification for this work and the goal of the method is to develop an algorithm that addresses the following features:

- To develop a neural-based reconstruction algorithm for system dynamics identification, which allow an autonomous spacecraft to refine the on-board dynamical model as it flies, coping with unmodeled perturbations and nonlinearities
- To develop a planning algorithm that can adapt to the perturbed environments using the neural reconstructed dynamics. This prevents the failure of traditional algorithms in presence of unmodeled terms in the dynamical model enhancing the autonomy and flexibility of the spacecraft, which can operate autonomously also in partially known environment
- To develop a relative trajectories prediction algorithm to ensure collision-free simultaneous reconfigurations. This is required when coordinated maneuvers are needed, where the hypothesis of the formation evolution according to natural dynamics does not hold anymore.

4.3.1 Neural Planning and Control

Model Predictive Control (MPC) is an optimization-based guidance and control strategy, which merges the advantage of optimization and closed-loop control [41]. Model Predictive Control is essentially a receding horizon optimization approach to plan control actions following optimal control theory. The optimization covers N_s time steps, which results in N_s control inputs. Only the first computed control is executed and, at the next time step, the planner solves the optimization problem again based on the current state. Typically, the objective function to minimize includes the quadratic difference between the target state and the spacecraft state, $\mathbf{x}_k$ at each time step, and a quadratic

term representing the control effort:

$$\mathcal{J}(\mathbf{x}_k, \mathbf{u}_k) = \Bigg[(\mathbf{x_{k+N}} - \mathbf{x_k^*})^{\mathbf{T}} \hat{S} (\mathbf{x_{k+N}} - \mathbf{x_k^*}) + \sum_{i=1}^{N-1} (\mathbf{x_{k+i}} - \mathbf{x_k^*})^{\mathbf{T}} S (\mathbf{x_{k+i}} - \mathbf{x_k^*}) + \sum_{i=0}^{N-1} \mathbf{u_{k+i}^T} R \mathbf{u_{k+i}} \Bigg] \tag{4.19}$$

where S, R are positive semi-definite matrices, $\hat{S}$ is the solution of the Discrete Algebraic Riccati Equation (DARE), N_s is the number of prediction steps. At each time step, the optimization problem can be expressed as:

$$\begin{aligned} &\underset{\mathbf{u}}{min}\ \mathcal{J}(\mathbf{x}_k, \mathbf{u}_k) \\ \text{subject to } &\mathbf{x_{k+i+1}} = \tilde{\mathcal{F}}_{T_s}(\mathbf{x_{k+i}}, \mathbf{u_{k+i}}),\ \textit{i=1,..,N} \\ &\mathbf{u_{min}} < \mathbf{u_i} < \mathbf{u_{max}},\ \textit{i=1,..,N} \end{aligned} \tag{4.20}$$

The objective function is minimized respecting the dynamics constraint and the maximum thrust limit. Note that the dynamics is supposed to be known and not learnt in a general MPC problem. This concept will be extended in the next section where the planning phase is coupled with the learning phase. The Model-Based Reinforcement Learning (or Neural-Model Predictive Control) exploits the flexibility of neural dynamics reconstruction presented in Section 3.2 to enhance the robustness, effectiveness and flexibility of the traditional MPC. The dynamics in Eq. 4.20 is replaced by the neural expression $\mathcal{N}_s$. Such strategy creates a highly-coupled learning and planning algorithm that resembles the approach of reinforcement learning but does not prevent the exploitation of the environment model. The complete algorithm reads:

Algorithm 3 Model-Based Reinforcement Learning

1: Initialize system description through dynamical model $\mathbf{x}_{k+1} = \tilde{\mathcal{N}}(\mathbf{x}_k, \mathbf{u}_k)$
2: Acquire target position $\mathbf{x}^*$
3: Acquire formation geometry: $\mathbf{x}_i - \mathbf{x}_k\ \forall i = 1, .., m$
4: **while** $\mathbf{x}_k \neq \mathbf{x}^*$ **do**
5: Observe system state at time k: $\mathbf{x}_k$
6: Optimize $\mathcal{J}(\mathbf{x}_k, \mathbf{u}_k)$ through $\tilde{\mathcal{N}}$ for N_s time steps
7: Execute $\mathbf{u}_k$
8: **end while**

When multiple spacecraft perform the formation reconfiguration maneuvers simultaneously, a collision avoidance strategy is mandatory. Simultaneous reconfiguration is critical for autonomous operations, nevertheless, if a distributed GNC architecture is implemented, it is impossible for each agent to have a complete knowledge of the relative target orbits of the others. Hence, it is mandatory to develop effective algorithms that relies only on information available to one agent.

4.3.2 Collision Avoidance Constraint

An important feature of distributed reconfiguration is to ensure safe trajectory generation to avoid collision between agents. In a distributed configuration, a generalized optimization describing the guidance path of each spacecraft is not available. Hence, at this stage, the reconfiguration is assumed to take place sequentially for the formation. This results in one spacecraft changing its relative orbit while the rest persist in their natural motion. The collision-free constraint is enforced by guaranteeing a minimum distance between the agents. The *keep-out-zone* is an ellipsoid centered on the obstacle and expressed by its quadratic form with the positive semi-definite matrix $\mathcal{P}$. If the *keep-out-zone* is a sphere, the quadratic form is simply a symmetric matrix $\mathcal{P} = r\mathcal{I}_{3\times3}$ where r is the spherical safe region radius [42]. The mathematical formulation in the MPC framework is reported in Eq. 4.21.

$$\begin{aligned} &\forall i = 1, ..., \mathcal{N} \;\; | \;\; 1 - \tilde{\mathbf{x}}_{o,k+i}^T \mathcal{P} \tilde{\mathbf{x}}_{o,k+i} < 0 \\ &\tilde{\mathbf{x}}_{o,k+i} = [C(\mathbf{x}_{k+i} - \mathbf{x}_{o,k+i})] \end{aligned} \tag{4.21}$$

where the subscript o stands for *obstacle*, which in general is another agent of the formation. The constraint in Eq. 5.20 is quadratic and non-convex. The former characteristic prevents the problem to be recast into quadratic programming, which would significantly reduce the computational time. The latter turns the formulation into a non-convex optimization. Convex programming is faster and guarantees the identification of global minima. For this reason the following section presents the derivation to transform the collision avoidance constraint into a convex constraint. The derivation is similar to the approach adopted by Morgan [42]. The idea is to approximate the concave set of feasible positions into a convex set, which contains the original set. In other words, the spherical *keep-out-zone* is approximated by a plane tangent to the sphere and perpendicular to the line of sight of the two agents. The plane perpendicular to the vector $\tilde{\mathbf{x}}_{o,k} = [\tilde{x}, \tilde{y}, \tilde{z}]$ (defined in Eq. 4.21) and tangent to the sphere can be written as:

$$\tilde{x}(x - \frac{r}{||\tilde{\mathbf{x}}_{o,k}||}\tilde{x}) + \tilde{y}(y - \frac{r}{||\tilde{\mathbf{x}}_{o,k}||}\tilde{y}) + \tilde{z}(z - \frac{r}{||\tilde{\mathbf{x}}_{o,k}||}\tilde{z}) = 0 \tag{4.22}$$

The feasible position belongs to the semi-space bounded by the identified plane. By reworking Eq. 4.22 and introducing the inequality indication, the compact form of the constraint at time step k is:

$$\tilde{\mathbf{x}}_{o,k}^T \tilde{\mathbf{x}}_{o,k} \geq r||\tilde{\mathbf{x}}_{o,k}|| \tag{4.23}$$

The natural dynamics is described by the ANN. Each mutual distance is calculated on board based on the reconstructed dynamics. A representation of the convex approximation of the original collision avoidance constraint is sketched in Fig. 4.15.

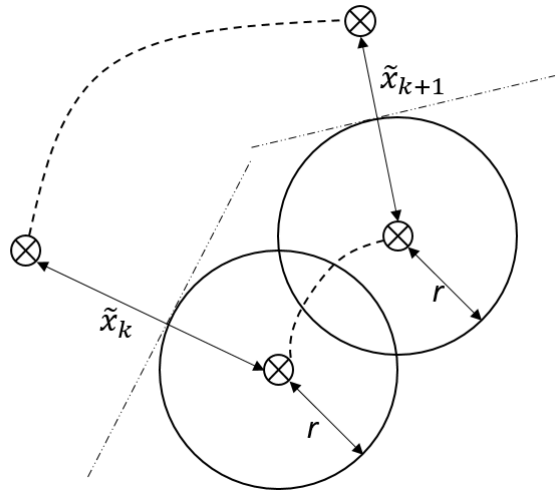


Figure 4.15: Convexification of the collision avoidance constraint. The sketch shows two instant in time.

4.4 Model-based Reinforcement Learning Performance

The presented algorithm is tested in two application scenarios. The first (FFLAS - Formation Flying L-band Aperture Synthesis) is to study a formation consisting of 3 hexagonal arrays, of about 7 m in diameter, flying with their centers at the vertices of an equilateral triangle of about 13m side. Such formation would be equivalent to an aperture of 21 m diameter achieving 9 km nadir resolution with an effective sensitivity better than SMOS [74]. The second scenario is based on CHRISTMAS/MUSICA, a feasibility study performed by an Italian consortium led by Universitá di Milano Bicocca in collaboration with Politecnico di Milano, Istituto Nazionale di Astrofisica and OHB-I, and financed by ASI. The objective of the mission is the Cryosphere Monitoring, operating with an innovative payload in the optical and thermal domain. The mission conceives a constellation of four satellites, flying in across-track formation by couple.

The selected scenarios are tight formations, requiring constant relative position and velocity control to satisfy the strict requirements on relative states. In particular, FFLAS requires rigid formation in triangular shape. The fixed relative position of the satellites is not stable in the natural dynamics. All the scenarios have been tested using the coupled MBRL-IRL and MBRL-LSTM with maximum thrust level equal to 1 N. Since no colliding trajectories are foreseen, the last section explores the effectiveness of the collision avoidance method for estimating neighboring satellites trajectories.

FK	**MBRL**			**MPC**		
	SC1	SC2	SC3	SC1	SC2	SC3
Δv_T $[\frac{m}{s}]$	0.04	0.0003	0.0003	0.05	0.0003	0.0003
ϵ_ρ $[m]$	0.020	0.004	0.004	0.015	0.004	0.004

Table 4.14: Δv for one orbit and accuracy for MBRL and MPC for FK.

4.4.1 In-Plane Maneuvers

The three satellites fly in rigid formation, i. e. a formation in which the relative distances and orientation among the satellites remain fixed. The orbit of the aperture center, from now on *reference orbit*, is the nominal orbit of SMOS, i. e. a 6 am - 6 pm Sun-synchronous orbit of 775 km altitude [74]. The aperture plane is perpendicular to the orbital plane. The perpendicular to the aperture plane is pointed in the nadir-zenith direction. Yaw rotations of the aperture plane are possible as long as the relative formation between the arrays is kept. The position of the arrays within the aperture plane is chosen to obtain an as large as possible coverage in the spatial frequency domain (the space defined by the relative vectors between the centers of all pairs of antenna elements, normalized to the wavelength), without leaving gaps and having a polygonal envelope as convex as possible. The formation flying algorithm requires a collision-free safe maneuver to bring the satellites far apart in case an unforeseen event occurs. Following the above-mentioned guidelines, three sub-scenarios have been simulated:

1. Formation Keeping (FK): Formation maintenance to keep the relative position fixed.
2. Formation Reconfiguration (FR): Reconfiguration in which a yaw rotation of the formation is achieved.
3. Safe Reconfiguration (SR): Reconfiguration of the formation achieving safe separation among the satellites.

The formation maintenance is performed using the presented algorithm. The formation flies with the three satellites at the vertices of an equilateral triangle of 13 m side. The distances between the satellites are limited and the reference orbit is circular. Despite this strong points for using a simple linearized dynamics, the neural algorithm demonstrates comparable results in terms of fuel consumption and accuracy with respect to a standard MPC. Tab. 4.14 shows the similar performance of the algorithms. Fig. 4.16 shows the fixed geometry of the formation. In order to test the capability of reconstructing nonlinearities and unmodeled terms of the dynamics, Fig. 4.17. shows an example of the same formation keeping in an eccentric orbit $e = 0.8$. The main

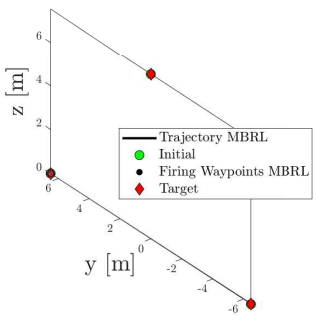


Figure 4.16: Formation geometry.

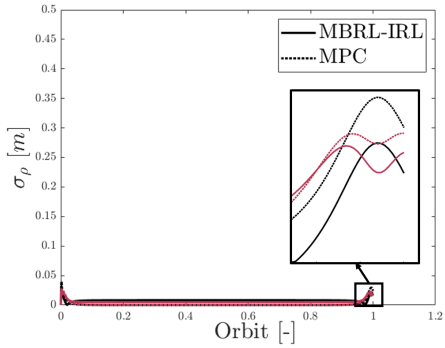


Figure 4.17: Accuracy in position control for the formation keeping in eccentric orbit.

FR	**MBRL**			**MPC**		
	SC1	SC2	SC3	SC1	SC2	SC3
Δv_T $[\frac{m}{s}]$	1.2	1.0	1.0	1.3	1.1	1.1
TOF $[s]$	600	600	600	660	660	660

Table 4.15: Δv and time of flight for MBRL and MPC for FR.

perturbation force occurs at the perigee. Indeed, as shown by the results, the neural reconstruction limits the loss in position control accuracy, demonstrating robustness towards the peculiar dynamics in eccentric orbits. The aperture plane is perpendicular to the orbital plane. The perpendicular to the aperture plane is pointed in the nadir-zenith direction. As stated, yaw rotations of the aperture plane are possible as long as the relative formation between the arrays is kept. The numerical results for a yaw rotation of 30°, shown in Fig. 4.18, are reported in Tab. 4.15. For close reconfiguration, the linearized dynamics used by the MPC delivers similar results with respect to the proposed MBRL as reported in Tab. 4.15. It is interesting to test the algorithm in a hypothetical scenario where the relative distances are two order of magnitude larger. The difference in accuracy and required Δv between the compared algorithms becomes more significant, as shown in Fig. 4.19. This is due to the fact that nonlinearities become more and more relevant as the formation grows in size. The MBRL utilizes the neural reconstructed dynamics that captures these unmodeled terms, delivering an more effective reconfiguration. The required Δv is $\sim 5\%$ lower when using MBRL. In case that relative positioning evolution showed any risk of collision, the system shall be designed to bring the satellites safely apart into a safe-mode flight formation. One strategy is to bring the satellites in along-track formation with a separation of 500 m, being

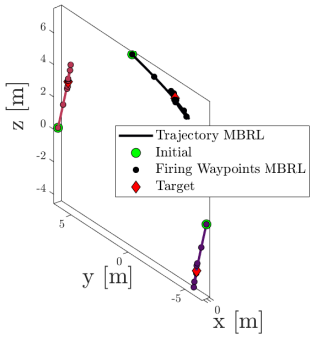


Figure 4.18: Formation reconfiguration following 30° yaw rotation of the synthetic aperture.

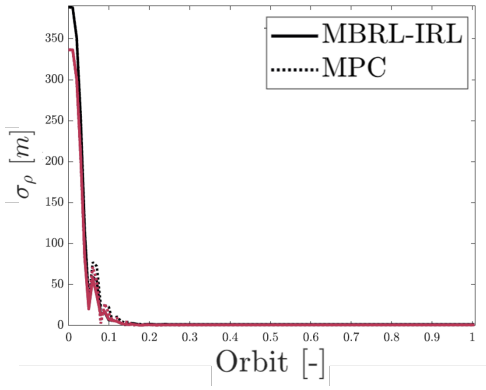


Figure 4.19: Large reconfiguration following 30° yaw rotation of the synthetic aperture. Dynamics reconstruction aids the MBRL algorithm as the formation grows in size.

this configuration stable in the natural dynamics. The uncertainty in predicting the along-track separation is higher than radial and cross-track component. Hence, another approach is proposed. In particular, the formation is enlarged, keeping the same geometry. Tab. 4.16 reports the results in terms of Δv and time of flight. Since the safe formation increases the relative distances, there is a benefit in employing neural reconstructed dynamics, for the reason discussed above. Fig. 4.20 shows the two reconfiguration in case of collision risk. As expected, the reconfiguration to safe-mode that enlarges the formation is more fuel-demanding with respect to the along-track formation. With reference to

SR1	**MBRL**			**MPC**		
	SC1	SC2	SC3	SC1	SC2	SC3
Δv_T $[\frac{m}{s}]$	0.4	10.2	10.2	0.5	11.1	11.1
TOF $[s]$	723	723	723	1085	1085	1085
SR2						
	SC1	SC2	SC3	SC1	SC2	SC3
Δv_T $[\frac{m}{s}]$	15.5	11.1	11.1	16.7	12.0	12.0
TOF $[s]$	1809	1809	1809	2110	2110	2110

Table 4.16: Δv and time of flight for MBRL and MPC for SR.

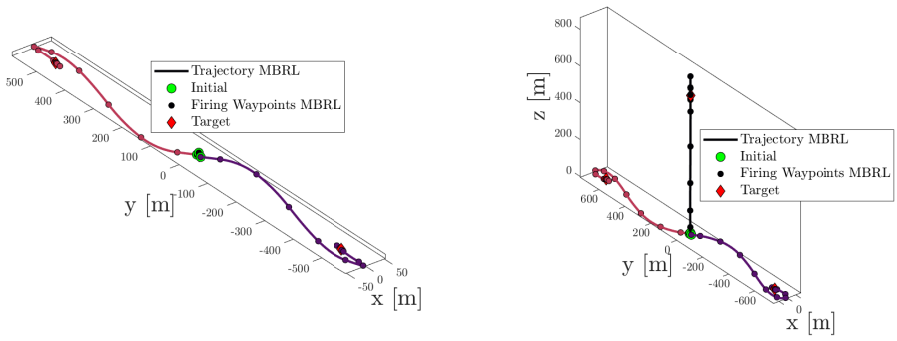


Figure 4.20: Safe-mode flight formation. Two strategies.

the results, the Time Of Flight (TOF) is constantly lower for MBRL algorithm, while requiring less fuel to accomplish the maneuver.

4.4.2 Out-of-Plane Maneuvers

A constellation of four satellites, flying in formation by couple, is the proposed strategy to achieve mission objectives. The proposed architecture allows for further enlargement, being the satellite couple, flying in cross track formation, the atomic unit of the constellation, which can replicated at convenience, according to the desired temporal resolution. The cross-track formation can be achieved by fixing the relative position or exploiting natural relative orbits, which owns a cross-track harmonic motion. In this last type of configuration, it is important to set an elliptical motion, i. e. with radial and along-track components, to avoid collision between the leader and the follower spacecraft. The relative orbital elements, in particular δe and δi completely defines the central bounded relative orbit [11][75]. The simulated reconfiguration is an hypothetical transfer between two cross-track relative orbits with parallel and perpendicular relative inclination/eccentricity vector, namely from $\boldsymbol{\delta e} \perp \boldsymbol{\delta i}$ to $\boldsymbol{\delta e} \parallel \boldsymbol{\delta i}$ with $a\delta i = a\delta e = 2\ km$. In such reconfiguration, quite far from the target, the MPC fails in trajectory control and the controller diverges, whereas MBRL completes the reconfiguration in s and $\Delta v \sim 30\ \frac{m}{s}$, as shown in Fig. 4.21. The reconfiguration is demanding due to the relative distance between the spacecraft. This may be diminished in operation by tuning the gain matrices of the planning algorithm.

4.4.3 Collision-Free Maneuvers

The algorithm has been tested on two different scenarios, representative of absolute orbits with different eccentricity (LEO and HEO). The orbital parameters are summarized in Table 4.17. The dynamics is reconstructed using

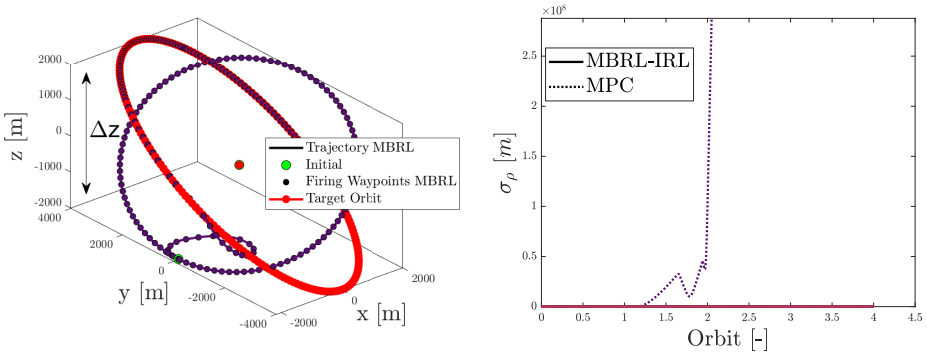


Figure 4.21: Cross-track reconfiguration from $\boldsymbol{\delta e} \perp \boldsymbol{\delta i}$ to $\boldsymbol{\delta e} \parallel \boldsymbol{\delta i}$ with $a\delta i = a\delta e = 2\ km$.

a two-layer MLP, with a 9-10-6 neurons architecture, as sketched in Fig. 3.3. An hyperbolic tangent is used as the activation function for the hidden layer, whereas a Rectifier Linear Unit (ReLU) is used as neural activation function for the output layer. The neural network is initially trained to approximate the CW model. A single input-output batch of 1000 data are used for initialization. The planner executes the optimization using $N_s = 20$ prediction steps and a sampling time of $T_s = 120\ s$. The thrust constraint is enforced by imposing $\mathbf{u}_{max} = [1\ 1\ 1]\ N$ for each axis. To demonstrate the activation of the collision avoidance constraint, it assumed that a reconfiguration is needed to drive one satellite in tight formation around the target satellite, which lies in the center of the LVLH reference frame. The safety region has radius $r_{KOZ} = 100\ m$. The constraint is then convexified to enable faster convergence of the solver. The optimization without collision avoidance constraint are solved as Quadratic Programming (QP) problem with *convex-interior-point* algorithm. When the collision avoidance constraint is activated, for instance when the relative distance is below a certain threshold or simply constantly solved, the convex programming is solved using *sequential-quadratic-programming* algorithm. The results report also the comparison with a standard MPC control scheme, based on CW-dynamics. The propagation is performed with a J2-perturbed nonlinear relative dynamics model, based on the formulation in [3].

The reconfiguration simulates a relative plane change of the formation with distances in the order of magnitude of $10^3\ m$. The collision avoidance constraint is enforced using a 100 m radius safety sphere.

Low Eccentricity The Clohessy-Wiltshire model is derived for small relative distances and nearly circular reference orbit. For this reason the standard MPC and the proposed MBRL algorithm does not differ significantly in terms of fuel-consumption and reconfiguration time. This result was expected, being

Table 4.17: Reference orbits for numerical simulations

		$\boldsymbol{a}$ [km]	$\boldsymbol{e}$ [-]	$\boldsymbol{i}$ [^{o}]	$\boldsymbol{\omega}$ [^{o}]	$\boldsymbol{\Omega}$ [^{o}]
LEO	7179	0	50	86	80	
HEO	35890	0.8	50	86	80	

$e = 0$	**MBRL**	**MPC**
Δv [m/s]	1.91	1.95
T_{man} [min]	31.5	43.6

Table 4.18: Δv and maneuvering time for MBRL and MPC for low-eccentricity reference orbit.

the modeled dynamics (cfr. Section 2.1.1.1) close to the operational scenario. The summary of simulation result is reported in Table 4.18. Figure 4.22 shows the reconfiguration trajectory designed by the planner algorithm in the reference LVLH frame. The comparison demonstrates that the MBRL performs similarly to the MPC implemented using CW model. In particular, the path generated by the algorithm is almost equal, demonstrating the equivalence of the method for the the modeled dynamics. A slight improvement is observed in the reconfiguration duration. Figure 4.23 shows the enforcement of the collision avoidance constraint. The scenario was chosen to be challenging in which the two spacecrafts are forced to orbit very close with respect to each other, relatively to the target spacecraft. The safety distance is always guaranteed throughout the orbital evolution. Figure 4.24 presents the learning process in terms of mean squared error of state prediction during the reconfiguration. At the maneuver time, the spacecraft initiates its planning and *explores* new regions of space, which were barely known beforehand (known to the extent of the model dynamics initialization). For this reason the network degrades rapidly right after the reconfiguration is started. As the satellite follows the planned trajectory the ANN is continuously trained as shown in Fig. 4.24.

High Eccentricity The most interesting testing cases are those in an environment that is significantly different from the modeled dynamics in Section 2.1.1.1. In particular, an highly eccentric orbit is chosen as described in Table 4.17. The initial conditions are the same as in the previous test case. The results in Table 4.19 show a Δv saving of nearly 13 %. This is thought to be linked with the progressive learning of the neural network used for the dynamics reconstruction. The reconfiguration time is in the same order of magnitude. Figure 4.25 presents the learning process in terms of mean squared error of state prediction during the reconfiguration. At the maneuver time, the spacecraft

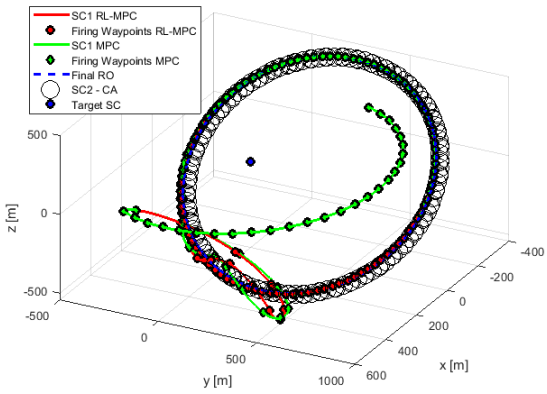


Figure 4.22: Reconfiguration examples in low-eccentricity orbit.

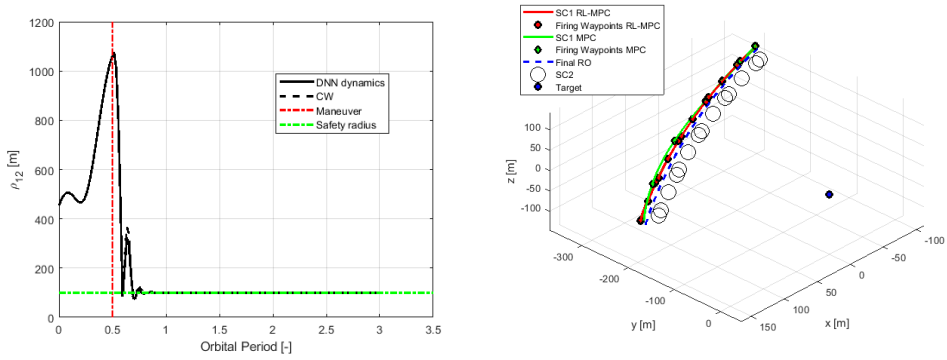


Figure 4.23: Collision avoidance constraint in low-eccentricity reconfiguration.

initiates its planning and *explores* new regions of space, which were barely known beforehand (known to the extent of the model dynamics initialization). For this reason the network degrades rapidly right after the reconfiguration is started. As the satellite follows the planned trajectory the ANN is continuously trained as shown in Fig. 4.25. Such characteristic is the true power of the method. All the numerical simulations are run right after initialization, hence the ANN is almost *new* to the perturbed dynamic environment when starting the reconfiguration. During a mission lifetime, especially if the configuration is changed quite often, the spacecraft will exploit better and better the natural dynamics for planning, leading to more efficient maneuvers as the mission time increases. The ANN learning presents spikes when the spacecraft approaches the perigee. This might be due to the fact the orbital velocity and the relative dynamics is significantly influenced by the current true anomaly on the reference orbit.
An interesting case is hereby presented. The initial condition is one order of

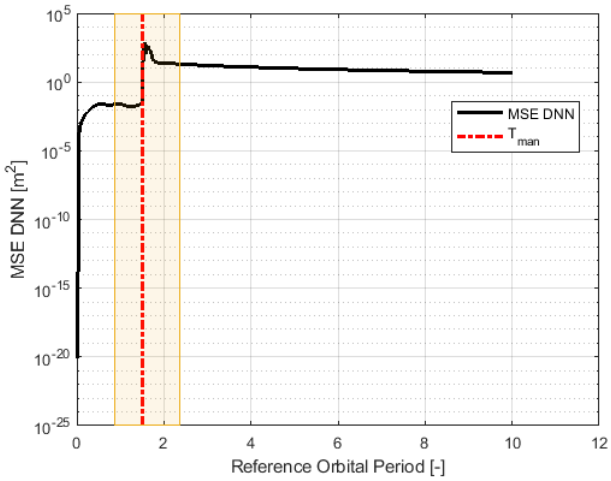


Figure 4.24: Mean Squared Error of ANN state prediction during on-board learning.

$e = 0.8$	**MBRL**	**MPC**
Δv [m/s]	7.06	7.91
T_{man} [min]	30.1	36.1

Table 4.19: Δv and maneuvering time for MBRL and MPC for high-eccentricity reference orbit.

magnitude larger than the previous case, namely $\delta\chi = [0\ 0\ 2000\ 3000\ 2000\ 0]$. The final state is targeted to an along-track separation with the target of 1000 km. Such initial configuration is fairly large, hence nonlinearities may become important in the dynamics. Indeed, as reported in Table 4.20 and Fig. 4.26, the MPC based on CW model fails in the reconfiguration, whereas the MBRL successfully accomplishes the maneuver. The standard MPC relies on a linearized dynamical model valid for nearly circular reference orbits. In addition, the reconfiguration algorithm has a strict thrust constraint that limits the capability of the spacecraft. The MPC fails in solving the receding-horizon optimization because the dynamical model is not adequate to model the dynamics for large distances in highly elliptical orbits. Fig. 4.26 shows the generated trajectory diverging from the formation configuration geometry, i.e. going to unfeasible relative distances.

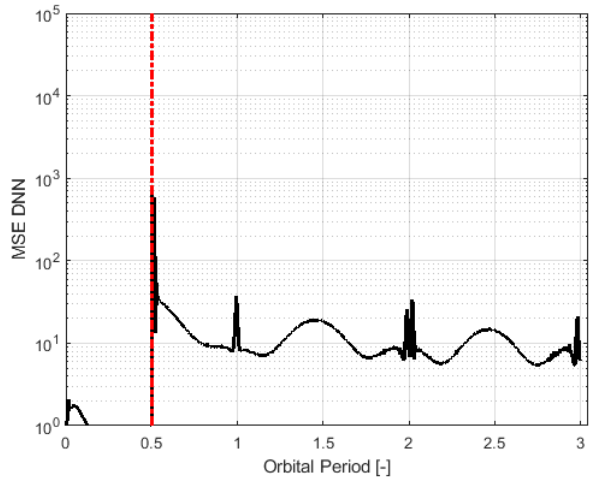


Figure 4.25: Mean Squared Error of ANN state prediction during on-board learning.

$e = 0.8$	**MBRL**	**MPC**
Δv [m/s]	2.65	-
T_{man} [s]	90.0	-

Table 4.20: Δv and maneuvering time for MBRL and MPC for high-eccentricity reference orbit for large-sized reconfiguration.

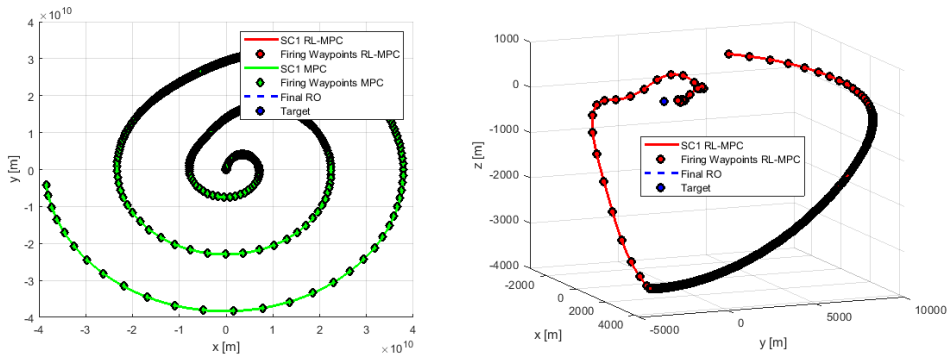


Figure 4.26: Collision avoidance constraint in low-eccentricity reconfiguration.

CHAPTER 5

Environment and External Uncertainty Prediction

> Non piú ottico ma spacciatore di lenti
> per improvvisare occhi contenti,
> perché le pupille abituate a copiare
> inventino i mondi sui quali guardare.
>
> — Fabrizio De André

The distributed architecture examined in this work does not rely on any data exchange between the agents. Only relative state is retrieved by each spacecraft with respect to the neighboring agents or bodies. This poses a severe criticality in terms of planning and control to avoid collision and undesired behavior. For instance, if two agents are maneuvering at the same time, the planned trajectories may intersect if none of the two knows the future evolution of the other. The Neural-Artificial Potential Field solved such shortcoming by recomputing the instantaneous guidance based on current measurements. As stated, this is due to the fact that no planning and optimization is executed in such algorithm. In the Model-Based Reinforcement Learning framework, the

optimization takes place within a receding horizon and collision constraints need to be taken into account beforehand. To solve this shortcomings, two methods have been developed capable of forecasting future states of the neighboring agents, regardless of being controlled trajectories or natural evolution, only relying on relative measurements. In particular, Inverse Reinforcement Learning is a powerful approach that tries to guess the objective function of the reconfiguring spacecraft to generate potential future trajectories. It works very well for long-time predictions based on a restricted number of observations. In addition, the method is coupled with a Long-Short Term Memory network, which shows superior performance in capturing short-horizon motion.

The following Chapter is organized as follows: Section 5.1 presents the mathematical foundation of the methods, in particular Section 5.1.1 discusses the Inverse Reinforcement Learning method, whereas Section 5.1.2 presents the Long-Short Term Memory employed for trajectory prediction. Finally, Section 5.2 reports the numerical validation results.

5.1 Forced Dynamics Prediction for Collision Avoidance

One critical task for distributed operation is to safely maneuver avoiding collision between agents. In a distributed architecture no information is globally available on the planning of each agent with respect to the formation, therefore each spacecraft need to predict future maneuvers of potentially colliding agents. Usually, the collision avoidance constraint is enforced by assuming the formation to evolve naturally or along predefined trajectories [69] [70], which are known by the system. Such assumption is quite restricting when dealing with distributed systems that follow a distributed GNC architecture. It would be helpful if, based on few relative observation between agents, the reference reconfiguration trajectory could be predicted. In this way, collision avoidance constraints can be taken into account by the each spacecraft's MBRL algorithm in the shape of a keep-out-ellipsoid. The path constraint for satellite p is added as:

$$1 - (\mathbf{x_{p,k+i}} - \mathbf{x^o_{j,k+i}})^{\mathbf{T}} P (\mathbf{x_{p,k+i}} - \mathbf{x^o_{j,k+i}}) < 0, \;\; i{=}1,..,N, j{=}1,..,m, j{\neq}\; p \quad (5.1)$$

where m is the number of agents. The $x^o_{j,k+i}$ vector is the prediction trajectory derived by the collision avoidance algorithm. The non-convex constraint in Eq. 5.1 is convexified by unfolding the ellipsoid into a tangent plane, as done in Section 4.3. Convex optimization guarantees the convergence of the algorithm to a global minimum.

5.1.1 Inverse Reinforcement Learning

In general, an optimal control problem for trajectory planning can be described as:

$$\begin{aligned} &\min_{\mathbf{u},\mathbf{x}} \mathcal{J}(\mathbf{x_t}, \mathbf{u_t}) = \sum_{t}^{T-1} c_t(\mathbf{x_t}, \mathbf{u_t}) + \psi(\mathbf{x_T}) \\ &\text{subject to } \mathbf{x_{t+1}} = \mathcal{F}(\mathbf{x_t}, \mathbf{u_t}),\ \textit{t=1,..,T} \\ &\qquad\qquad \mathbf{u_{min}} < \mathbf{u_t} < \mathbf{u_{max}},\ \textit{t=1,..,T-1} \end{aligned} \tag{5.2}$$

where c_t is the running intermediate cost and ψ the terminal state penalty. Similarly to what The cost function can be recast into a feature-based expression where the cost is a linear combination of f nonlinear features. In principle, each state-control pair can represent a feature. In this paper the optimizer is described as an optimization over a linear combination of features ϕ, which represent cumulative cost along feasible trajectories and terminal state penalty. The feasible trajectories are those respecting optimization constraint and dynamics: the set of state-control pairs represents a *policy* π, borrowing a term from the Reinforcement Learning world. Using such approach, the cost function can be rewritten [71]:

$$\begin{aligned} &\mathbf{w} = [w_1, w_2, ..., w_f, w_T]^T,\ \mu(\pi) = \begin{bmatrix} \sum_t^{T-1} \phi_1(\mathbf{x_t}, \mathbf{u_t}) \\ \sum_t^{T-1} \phi_2(\mathbf{x_t}, \mathbf{u_t}) \\ ... \\ \sum_t^{T-1} \phi_f(\mathbf{x_t}, \mathbf{u_t}) \\ \phi_T(\mathbf{x_T}) \end{bmatrix} \\ &\text{cost: } \mathcal{J} = \mathbf{w}^T \cdot \mu(\pi) \end{aligned} \tag{5.3}$$

The above formulation is necessary to introduce and discuss the Feature Matching Approach for Inverse Reinforcement Learning.

5.1.1.1 Feature-Matching Approach

The concept of Inverse Reinforcement Learning is to estimate a cost function that delivers an optimal trajectory compatible with an *expert* demonstrated trajectory, called $\tilde{\gamma} = \{(\mathbf{x_t}, \mathbf{u_t})\}_t^T$ for simplicity, as shown in Fig. 5.1. It is important to note that the expert cost function, parametric in the set of features, is not known, but we assume the demonstration to be optimal for the estimated one. Each trajectory is generated by a policy π, which characterizes the state-control pairs at each instant in time, ideally. In detail, given a set of N_o observations of the demonstrated trajectory $\mathcal{D}_{\tilde{\gamma}} = \{(\mathbf{x_i}, \mathbf{u_i})\}_{i=1}^{N_o}$ we need to find a cost function $\mathcal{J}$, under which the demonstrated trajectory looks optimal. Feature Matching Approach (FMA), first proposed by Abbeel [71] for imitation learning, solves for the weights in Eq. 5.3 by attempting to match the

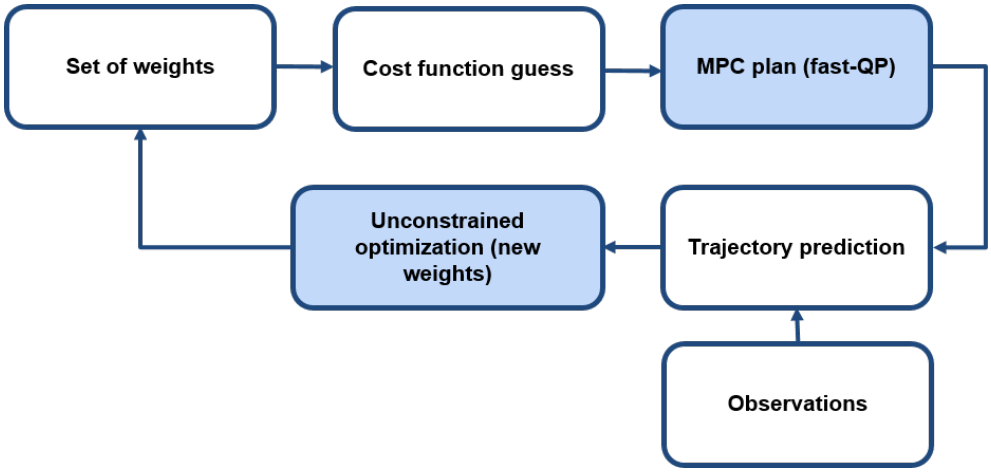


Figure 5.1: Scheme of neighboring agents identification and collision avoidance using Inverse Reinforcement Learning.

cumulative feature cost demonstrated by the *expert* under the optimal policy $\tilde{\pi}$ and the policy π^* based on the estimated cost function. The FMA can be expressed in compact form:

$$\mathcal{J}(\tilde{\gamma}|\tilde{\pi}) < \mathcal{J}(\gamma|\pi) \rightarrow \mathbf{w}^T \cdot \mu(\tilde{\pi}) < \mathbf{w}^T \cdot \mu(\pi), \ \forall \gamma, \pi \tag{5.4}$$

where it is stated that the demonstrated trajectory is optimal with respect to the estimated cost function. We would like the algorithm to look for the cost function optimized by the expert by trying to generate an estimation matching demonstrated feature. The demonstrated trajectory owns significant information about the structure of the cost function, hence the algorithm significantly benefits if the discrepancy between estimated optimal trajectory and expert one is integrated in the optimization. In other words, we would like to encode the effectiveness of a generated policy by comparing the resultant trajectory with the demonstrated one [76]. Loss augmentation represents a cost gap structured margin:

$$\mathcal{L}_{\tilde{\gamma}}(\pi) = \sum_{i=1}^{N_o} ||\tilde{\mathbf{x}}_i - \mathbf{x_i}||^2 \tag{5.5}$$

by inserting Eq. 5.5 in Eq. 5.4 we obtain:

$$\mathcal{J}(\tilde{\gamma}|\tilde{\pi}) < \mathcal{J}(\gamma|\pi) \rightarrow \mathbf{w}^T \cdot \mu(\tilde{\pi}) < \mathbf{w}^T \cdot \mu(\pi) - \mathcal{L}_{\tilde{\gamma}}(\pi), \ \forall \gamma, \pi \tag{5.6}$$

The Feature Matching Approach (FMA) for Inverse Reinforcement Learning can ultimately be written as:

$$\begin{aligned} &\min_{\mathbf{w}} \ ||\mathbf{w}||^2 \\ &\text{subject to} \quad \mathbf{w}^T \cdot \mu(\tilde{\pi}) < \mathbf{w}^T \cdot \mu(\pi) - \mathcal{L}_{\tilde{\gamma}}(\pi) \end{aligned} \tag{5.7}$$

Eq. 5.7 represent a convex optimization. Nevertheless, the set of policies the algorithm sweeps is theoretically not finite. This makes the optimization intractable, in particular for computational constraints of several interesting applications in spacecraft GNC. We may suppose that there exists a policy π^* that minimizes the right-hand-side of the constraints in Eq. 5.7. Then, the constraints in Eq. 5.7 can be written as $\mathbf{w}^T\mu(\tilde{\pi}) < \min_{\pi}\{\mathbf{w}^T\mu(\pi) - \mathcal{L}_{\tilde{\gamma}}(\pi)\}$ without any loss of generality. Similarly to what is proposed by Ratliff [77] we can place the constraint into the cost function. This yields an unconstrained optimization, which can be solved using gradient-based algorithms:

$$\min_{\mathbf{w}} \frac{\eta}{2}||\mathbf{w}||^2 + \left(\mathbf{w}^T\mu(\tilde{\pi}) - \min_{\pi}\{\mathbf{w}^T\mu(\pi) - \mathcal{L}_{\tilde{\gamma}}(\pi)\}\right) \tag{5.8}$$

where η is a user-defined coefficient. The FMA-IRL is actually a nested optimization problem. The outer loop is unconstrained, whereas the inner loop, which is a path-planning optimization, may be constrained, both linearly and non-linearly. The cumulative feature cost for the formation spacecraft trajectories can be expressed as:

$$\mu(\pi) = \begin{bmatrix} \mathbf{X}_k^T\mathcal{S}_s\mathbf{X}_k \\ \mathbf{U}_k^T\mathcal{R}_r\mathbf{U}_k \\ (F\mathbf{X}_k)^T\hat{\mathcal{S}}(F\mathbf{X}_k) \end{bmatrix} \tag{5.9}$$

where $\mathbf{X}_k$, $\mathbf{U}_k$, $\mathcal{S}_s$ and $\mathcal{R}_r$ are the stacked vectors and matrices to shorten the summation in Eq. 5.3. Note that no target state guess is inserted, making the formulation insensitive to such parameter. From Eq. 5.9, the weights vector is $\mathbf{w} = [w_s, w_r, w_{\hat{s}}]^T$.

5.1.1.2 Inner Loop: Fast Quadratic Programming

The nested optimization resulting from the FMA for trajectory prediction relies on an inner loop, evaluating the trajectory generated by the estimated cost function. In this work, the inner *policy* evaluation is performed using a fast Model Predictive Control recast into Quadratic Programming formulation. Such strategy allows a rapid convergence of the inner loop, limiting the computational burden of the whole algorithm. In particular, the MPC resembles the architecture presented in section 4.3 with the additional term entailing the gap structured margin $\mathcal{L}_{\tilde{\gamma}}(\pi)$. The proposed implementation uses the linearized natural dynamics for state prediction. In general, this approach can be extended to a neural reconstructed dynamics. The Quadratic Programming (QP) formulation is:

$$\begin{aligned} &\min_{\mathbf{U}_k} \frac{1}{2}\mathbf{U}_k^T\mathcal{Q}\mathbf{U}_k + \mathcal{H}\mathbf{U}_k \\ &\text{subject to } \mathcal{V}\mathbf{U}_k < \mathcal{W} \end{aligned} \tag{5.10}$$

where $\mathbf{U}_k = [\mathbf{u}_k, ..., \mathbf{u}_{k+N-1}]$ is a stacked vector containing the decision variables for the optimization problem, namely the control action for each discretization time step. The matrices of the QP formulation can be written as:

$$\mathcal{Q} = 2w_r\mathcal{R}_r + 2w_{\hat{s}}\Omega^T\mathcal{S}_{\hat{s}}\Omega - 2\Omega\mathcal{P}_p\Omega \tag{5.11}$$

$$\mathcal{H} = 2\hat{\mathbf{x}}_k^T\Psi\mathcal{S}_{\hat{s}}\Omega + 2\mathbf{X}_\gamma\mathcal{P}_p\Omega \tag{5.12}$$

where $\mathbf{X}_\gamma$ is the stacked vector of the demonstrated trajectory, $\mathcal{P}_p$ is a stacked identity matrix. The matrices Ω and Ψ represent the stacked state transition matrix and control matrix for the linearized relative dynamics. The relative dynamics can be neural-reconstructed or an analytical linearized model. Indeed, one can write:

$$\mathbf{X}_k = \Psi\mathbf{x}_k + \Omega\mathbf{U}_k \tag{5.13}$$

where

$$\begin{gathered}\Psi = [\Phi(t_{k+1}, t_k), \Phi(t_{k+2}, t_k), ..., \Phi(t_{k+\mathcal{N}}, t_k)]^T \in \mathcal{R}^{6\mathcal{N}\times 6} \\ \Omega = \begin{bmatrix} \Phi(t_{k+1}, t_k)B & 0 & \dots & 0 \\ \Phi(t_{k+2}, t_k)B & \Phi(t_{k+2}, t_{k+1})B & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \Phi(t_{k+\mathcal{N}}, t_k)B & \Phi(t_{k+\mathcal{N}}, t_{k+1})B & \dots & \Phi(t_{k+\mathcal{N}}, t_{k+\mathcal{N}-1})B \end{bmatrix}\end{gathered} \tag{5.14}$$

in which the fundamentals matrices are:

$$\begin{gathered}\mathcal{R}_r = \begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & & \ddots & \\ 0 & 0 & \dots & R \end{bmatrix} \in \mathcal{R}^{3\mathcal{N}\times 3\mathcal{N}} \qquad \mathcal{S}_{\hat{S}} = \begin{bmatrix} S & 0 & \dots & 0 \\ 0 & S & \dots & 0 \\ \vdots & & \ddots & \\ 0 & 0 & \dots & \hat{S} \end{bmatrix} \in \mathcal{R}^{6\mathcal{N}\times 6\mathcal{N}} \\ \mathcal{W} = \begin{bmatrix} \mathbf{U}_{max} \\ \mathbf{U}_{min} \end{bmatrix} \in \mathcal{R}^{6\mathcal{N}\times 1} \qquad \mathcal{V} = \begin{bmatrix} \mathbf{I}_{3\mathcal{N}\times 3\mathcal{N}} \\ -\mathbf{I}_{3\mathcal{N}\times 3\mathcal{N}} \end{bmatrix} \in \mathcal{R}^{6\mathcal{N}\times 3\mathcal{N}}\end{gathered} \tag{5.15}$$

5.1.1.3 Outer Loop: Unconstrained Optimization

The outer loop is a convex unconstrained minimization problem that can be solved by *quasi-newton* methods with gradient descent. In the proposed implementation, the gradient reads:

$$\nabla_w = \eta\mathbf{w} + (\mu(\tilde{\pi}) - \mu(\pi)) \tag{5.16}$$

that completes the FMA-IRL algorithm.

5.1.2 Neural-Sequential Trajectory Forecasting

The trajectory prediction for collision avoidance based on IRL is a promising algorithm, which delivers good results during the numerical validation test campaign. Although being simple and effective, IRL presents two drawbacks for working out the delicate task:

- the cost features have carefully been designed by analyzing the dynamics of the spacecrafts. In particular, it is assumed that all the spacecrafts executes the formation reconfiguration using an MBRL algorithm with the same cost function. The major contribution of the IRL algorithm is to estimate the unknown target position (or orbit) of each agent. Such necessary design process limits the flexibility of the method.
- the unknown cost function assumes to be quadratic in the control effort, hence it is required at least an estimate of the observed control thrust impulse. This is not trivial when dealing with typical relative navigation measurements. A possible way to solve this issue is approximating the control term by a discrete velocity differential between two subsequent observations in time, provided that they are close in time.

We may want to increase the flexibility of the proposed method by using a nonlinear model approximation, such as ANN, to represent the more complex cost function. Nevertheless, it is very hard to implement it in a particular scenario where the *expert* trajectory consists only of a few observation. Hence, probabilistic approaches, such as Maximum Entropy, seems inadequate for the spacecraft simultaneous reconfiguration task.

As stated above, Recurrent Networks have the capability of handling time-series data efficiently. The connections between neurons form a directed graph, which allows an internal state memory. This enables the network to exhibit temporal dynamic behaviors. From this premises, the use of RNN can be extended also for trajectory identification of neighboring satellites. Differently from the dynamics reconstruction used in MBRL, the network for neighboring satellites, does not need the control input, since it tries to estimate the future trajectory by only observing the relative state.

5.1.2.1 Long Short-Term Memory Network

The type of recurrent neural network used for neighboring satellites trajectory prediction is a Long Short-Term Memory network. It is a type of recurrent neural network widely used for making prediction based on times series data LSTM are a powerful extension to the standard RNN architecture because they solve the issue of *vanishing gradients*, which seldom occurs in network training. For a general overview of the LSTM network it is suggested to refer to [78]. The architecture of the network used in this paper is reported in Fig. 5.2.

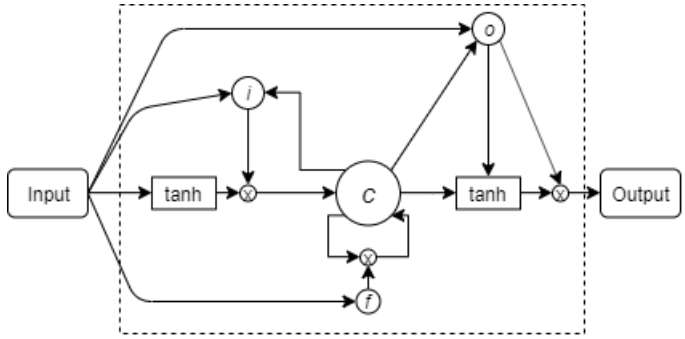


Figure 5.2: LSTM network for neighbouring satellites thrusted-trajectory identification and prediction. The core of the LSTM are the *cell* (C), the *input* gate (i), the *output* gate (o) and the *forget* gate (f) [78].

5.1.2.2 Online Supervised Training

The LSTM can be trained to make predictions based on time sequence data. Given the demonstrated trajectory $\tilde{\gamma} = \{(\mathbf{x_t}, \mathbf{u_t})\}_t^T$, where we neglect the control $\mathbf{u_t}$, the proposed sequential supervised learning is performed by feeding the network with the gathered input states and compared with the same states shifted of one time step.

$$\mathbf{x}_{in,LSTM} = \{\tilde{\gamma}\}_{(t,t+(N_s-1)\cdot T_s)} \tag{5.17}$$

$$\mathbf{y}_{out,LSTM} = \{\tilde{\gamma}\}_{(t+T_S,t+N_s\cdot T_s)} \tag{5.18}$$

In order to obtain a more robust fit and to prevent the training from diverging, it is required to standardize the training data to have zero mean and unit variance. Hence, the training data are transformed as follows:

$$\tilde{\mathbf{x}}_{in,LSTM} = \frac{\mathbf{x}_{in,LSTM} - \mu_{x,in}}{\sigma_{x,in}} \tag{5.19}$$

where μ is the coordinate-wise mean, i.e. for each observed coordinate of the relative position, and σ is the relevant standard deviation.

5.2 Numerical Test: Results & Discussion

The following section presents a numerical validation of Inverse Reinforcement Learning and Long-Short Term Memory for trajectory prediction in distributed system. The simulation is simply a validation of the algorithm for the trajectory prediction. The scenario is based on two spacecraft, which are assumed to be reconfiguring the formation. Each spacecraft observes the relative state of the other one for a set of $\mathcal{N}_o$ observations. Based on such observations, the IRL algorithm yields a predicted trajectory based on the reconstructed

cost-function. The predicted trajectory is compared to the natural motion the spacecraft would have if no maneuvers were employed, which is often used for planning [30] [41]. The assumed natural motion is described by the well-known Clohessy-Wiltshire model, suitable for close-range motion in low eccentricity orbit. The model is reported in Section 2. Tab. 5.1 reports the numerical settings for the simulation.

Table 5.1: Numerical settings for simulation scenario

Parameter	**Value**
Reference Orbit	
a [km]	7975
e [km]	0.1
i [deg]	10
Reconfiguration	
x_0 $[m]$	$[200\ 800\ 0]^T$
x_T $[m]$	$[0\ 0\ 0]^T$
IRL	
$\mathcal{N}_o$	4/6
$\mathcal{N}_p$	4
MPC	
$\mathcal{N}$	10
u_{max} $[N]$	1
$\mathcal{T}_s$ $[s]$	60

The algorithm has been tested with a low eccentricity reference orbit, namely $e = 0.1$. The dynamics in Eq. 2.2 is valid for close-range formation flying in nearly-circular orbits. The aim of the presented numerical validation is to exploit the IRL algorithm to reconstruct the planned trajectory, regardless of the dynamics of the system. The proposed reconfiguration is from a stable periodic orbit to the stable origin reference point. The second agent is supposed to observe the state of the reconfiguring spacecraft and processing the measurement to predict the forced trajectory. Fig. 5.3 shows the reconfiguration trajectory based on the MPC algorithm [79]. In addition, the predicted trajectories are depicted: the red dots represent the IRL predicted positions at the next $\mathcal{N}_p$ sample times (the algorithm sample time is $\mathcal{T}_s = 60\ s$); the blue dots represent the hypothetical trajectory if the natural dynamics was employed for prediction. Fig. 5.3 demonstrates the tremendous offset that the natural motion predicts, which jeopardizes its use for a safe reconfiguration planning or at least for a simultaneous reconfiguration. Fig. 5.4 shows the norm of the error in position prediction for subsequent time instant. At a given time, after $\mathcal{N}_o$ observations, the trajectories is predicted in the future for $\mathcal{N}_p$ time steps. The accuracy of the trajectory prediction degrades in time, but the algorithm is supposed to be

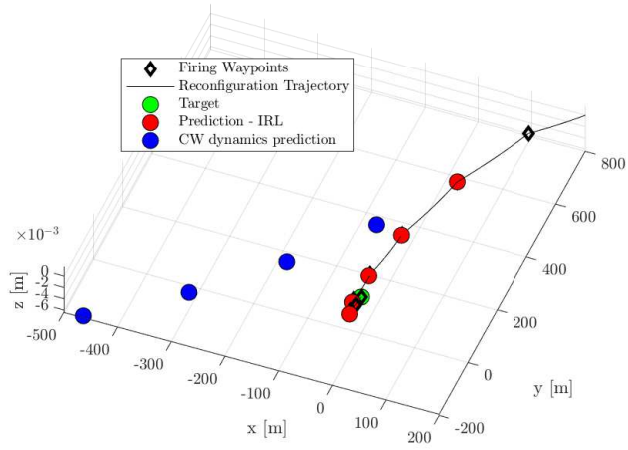


Figure 5.3: Comparison between the IRL predicted trajectory and the hypothetical trajectory the spacecraft would have if it was in natural motion.

working with the same MPC frequency, with $\mathcal{T}_s$ as sample time. This means that the trajectory prediction is refined at each time step. For this reason, the most relevant error is that of the *first prediction* step. The position error for the IRL reconstructed trajectory is lower than $\sim 10\ m$, which is within the typical *keep-out* ellipsoid used for safe reconfiguration [79]. On the other hand, the prediction based on natural dynamics rapidly diverges to $>\sim 10^2\ m$, making it unsuitable to be used in safe trajectory planning.

5.2.1 Collision Avoidance Algorithms Comparison

Two algorithms have been presented to cope with the lack of knowledge on the neighbouring satellites trajectories while performing reconfiguration. If the formation is reconfiguring using impulsive maneuvers the zero-impulse natural dynamics is no longer valid, indeed the keep-out-zone limit may often be intersect, violating the constraint due to wrong trajectory prediction. Fig. 5.5 shows a challenging reconfiguration where the spacecraft swap the along track positions separated by 200 m. As shown, the relative distance between the satellites falls below the Keep-Out-Zone limit of 100 m when the prediction of neighbouring agents is carried out using natural dynamics. On the other hand, coupling the MBRL planner with a impulsive trajectory identification algorithm, such as IRL, allows a safe reconfiguration.

In the distributed architecture it is assumed that each satellite can gather information on the relative state by autonomous navigation but cannot communicate its guidance to the neighboring satellite. A numerical simulation is reported

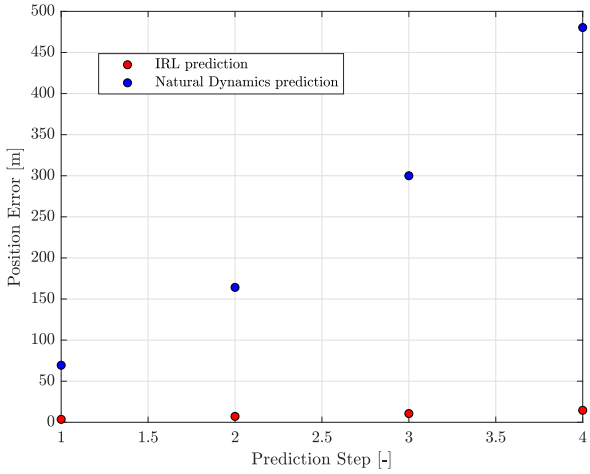


Figure 5.4: Prediction error for subsequent time steps $\mathcal{T}_s$.

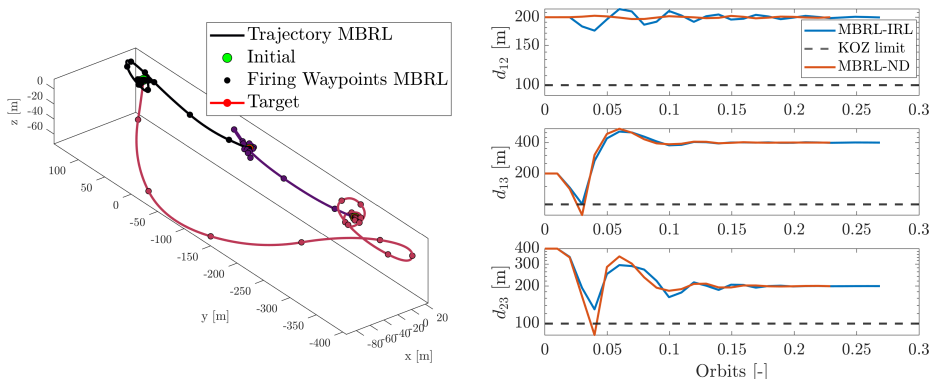


Figure 5.5: Close intersecting reconfiguration. The relative distances between the agents are shown. MBRL-ND coupled with natural dynamics prediction for collision avoidance violates the constraints during close approach.

here, in which colliding trajectories where deliberately chosen. The simulation consist of a sampling of $\mathcal{N}_o$ observation of a neighboring satellite's trajectory. These observation are processed by the IRL algorithm to approximate a cost function, whose optimization delivers a predicted trajectory shown in Fig. 5.6. Simultaneously, the observations are used to train the LSTM network. The number of prediction is set to $\mathcal{N}_p = 5$, whereas $\mathcal{N}_o = 10$. According to Tab.5.2,

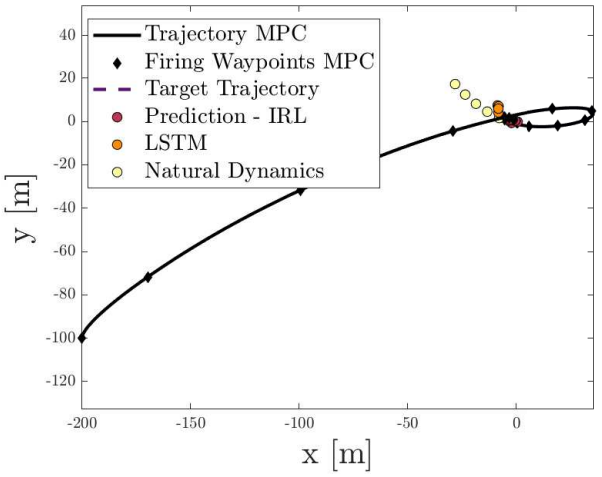


Figure 5.6: Neighboring satellite's trajectory. Predicted trajectories based on IRL, LSTM and natural dynamics are shown in colored dots.

Prediction Step	**IRL**	**LSTM**	**ND**
1^{st} $[m]$	4.7	5.0	1.6
5^{th} $[m]$	0.52	9.9	81.8

Table 5.2: Prediction error for IRL, LSTM and natural dynamics model (ND).

the two algorithms are beneficial with respect to natural dynamics prediction for controlled reconfiguration, as expected. The LSTM shows superior performance in predicting the very first state after the last observation, but the error grows as the number of time step to be predicted increases. An opposite behavior is shown by IRL, which delivers a finer prediction in the last time step. A combination of the two may be beneficial for the inclusion of such trajectory prediction into the MBRL scheme. A more comprehensive analysis has been conducted to map the behavior of the algorithm for different combination of $\mathcal{N}_p$ and $\mathcal{N}_o$ in the same scenario described before. Fig. 5.7 shows the difference in the presented algorithms. On one hand, short-horizon predictions based on few observations are better resembled using LSTM. On the other hand, the IRL accuracy is higher when a larger observation set is used. The common

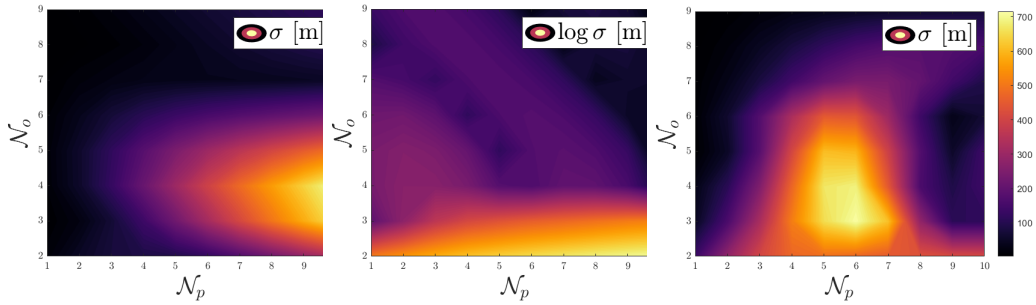


Figure 5.7: Map of RMS error of the neighboring agent trajectory prediction based on CW-model, IRL algorithm and LSTM, left to right, as a function of the number of observations and predictions.

high-RMSE region is due to the particular simulated case, where at the 6^{th} observations, a large control action steers the trajectory. These results are presented for a particular relative trajectory, but the insights reported on the behavior of the two methods can be extended to the range of relative motion pertaining the Formation Flying domain.

5.2.2 Sensitivity on Controller Weights

The planned reconfiguration trajectory depends on the weights that are set in the cost function of the MBRL controller. In particular, the controller can be steered towards a time-efficient or energy-efficient trajectory generation by changing the relative weights of the terms in the cost function. If the objective is to minimize the energy of the control effort, intuitively, the planned trajectory exploits at the most the natural motion of the orbiting spacecraft. On the other hand, if a *time-efficient* strategy is pursued, the controller forces a thrust trajectory to minimize the maneuver execution time, at the cost of larger Δv. As a result, the IRL algorithm is critical when using the latter strategy. Fig. 5.8 shows the cumulative error after 10 steps prediction using IRL and natural dynamics. It demonstrates that for energy-efficient reconfiguration the prediction based on natural dynamics is comparable to the one derived by IRL. Fig. 5.9 shows an example in which the 10 steps prediction is particularly off when dealing with MBRL controllers tuned to achieve *fast* reconfiguration. Based on only $\mathcal{N}_o = 4$ observations, the predicted trajectory is well reconstructed using the IRL algorithm. The predicted positions are fed into the MBRL controller to generate *safe* trajectories. The MBRL treats the obstacle collision avoidance constraint as in Eq. 5.20.

$$\begin{aligned} &\forall i = 1, ..., \mathcal{N} \;\; | \;\; 1 - \tilde{\mathbf{x}}_{o,k+i}^T \mathcal{P} \tilde{\mathbf{x}}_{o,k+i} < 0 \\ &\tilde{\mathbf{x}}_{o,k+i} = [C(\mathbf{x}_{k+i} - \mathbf{x}_{o,k+i})] \end{aligned} \tag{5.20}$$

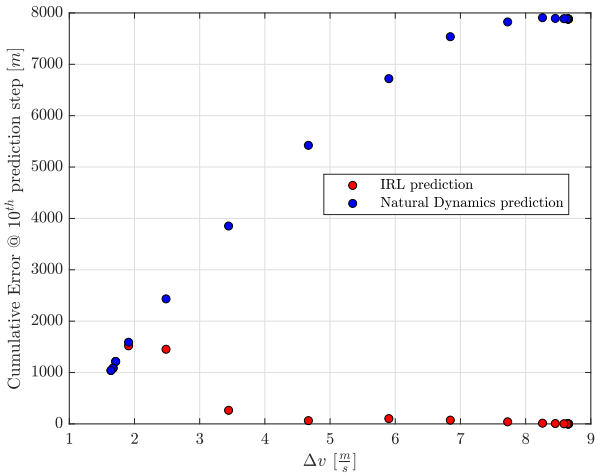


Figure 5.8: Cumulative error for increasing Δv reconfiguration. As the MBRL controller is steered towards *time-optimal* reconfiguration, the IRL algorithm is necessary for prediction.

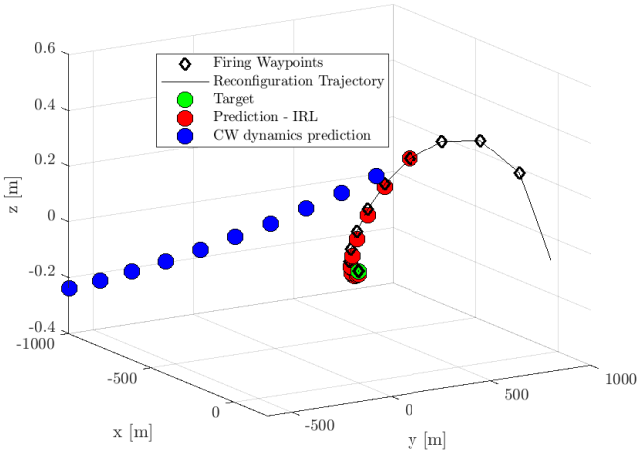


Figure 5.9: An example of a trajectory generated by the MBRL controller tuned towards *fast* control.

where the subscript o stands for *obstacle*, which in general is another agent of the formation. The constraint in Eq. 5.20 is quadratic and non-convex, thus it requires to be convexified as in Section 4.3.

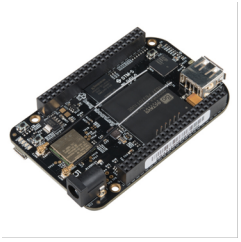
CHAPTER 6

Processor-In-the-Loop Implementation

La noia data da uno non pratico
Che non ha il polso di un matematico
Che coi motori non ci sa fare
E che non sa neanche guidare
Un tipo perso dietro le nuvole e la poesia.

— Francesco Guccini

The engineering work is definitely not finalized if the algorithms are not tested in *relevant* hardware. *Relevant* hardware means some kind of piece of actual technology that could in principle be implemented in satellite missions, according to the size, the mass and the computational power. The whole Thesis focused on the development of algorithms suitable for microsatellite mission, which are protagonist in the futuristic mission concepts involving formation flying and daring proximity operations. Thus, this Chapter presents the activities performed in the context of Processor-In-the-Loop validation of the algorithms presented in Chapter 3, 4 and 5. In particular, this Chapter is organized as follows: Section 6.1 briefly highlights the main procedure to set-up



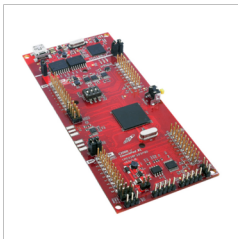


Figure 6.1: Hardware equipment for PIL validation.

the PIL simulation (note that it is not reported a step-by-step procedure); Section 6.2 reports the results of the PIL test campaign.

6.1 Processor-In-the-Loop Simulation Setup

The GNC validation requires the deployment of the algorithms into relevant hardware [80], which is able to emulate the computational power available on-board. The focus is to validate and verify the feasibility of the developed algorithms implementation into relevant processors and compare their performance with CPU execution, as sketched in Fig. 6.2. The PIL simulations features the execution of the GNC algorithms in both Desktop computers and stand-alone boards. As shown in Fig. 6.2, the control action is calculated and fed to an orbital dynamics simulator to integrate the motion equations. The communication is performed using USB-serial link. Such process increases the Technology Readiness Level (TRL) level of the algorithms, raising it to 3/4. During experimental activities, it has been critical to develop a fast and manageable deployment routine to standardize the process. This required significant work during the set-up, as reported in 6.1.3. Once the set-up and porting procedure was fully functional, the rapid prototyping and verification stage could be performed. The boards are shown in Fig. 6.1. The boards have been selected according to performance standard that are available both for nanosatellite missions as well as larger spacecraft.

6.1.1 Microcontroller Unit

The Hardware equipment used in this Thesis is a microcontroller unit MCU TI LAUNCHXL-F28379D Dual-core architecture. It features:

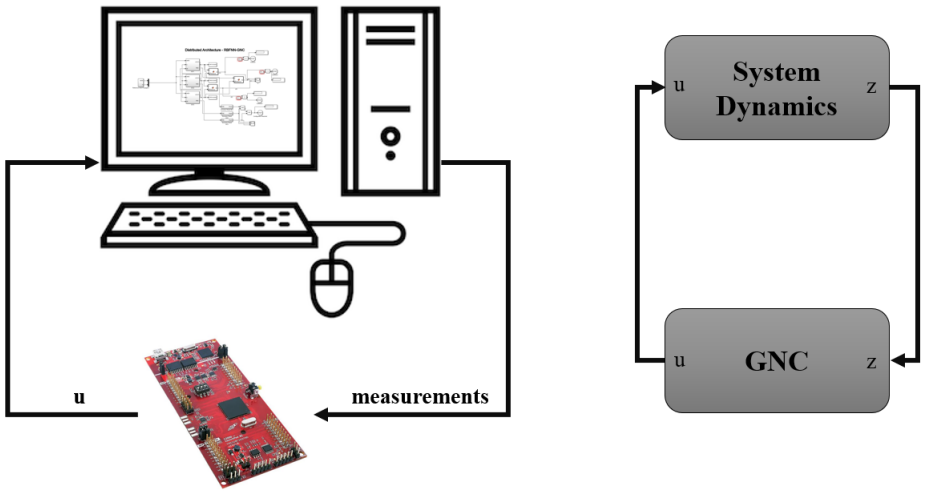


Figure 6.2: Schematics of Processor-In-the-Loop verification and validation process.

- Two TMS320C28x 32-bit CPUs
- 200 MHz processor
- IEEE 754 single-precision Floating-Point Unit (FPU)

The CPU frequency can be regarded as representative of the lowest performing on-board computers, which can be easily integrated in nanosatellites. Thus, the preliminary PIL verification and validation of the algorithms was performed using such resource. It is important to remark that only single-precision is supported by the MCU.

The microcontroller has been selected considering as the main driver both the availability of number of output peripherals and compuational power with respect to similar devices of the same class present on the market. The hardware is developed by Texas Instruments (TI), in particular the model is the F28379D LaunchPad Development Kit from C2000 Real-Time Control MCUs family; electronic specifications and wiring can be found directly in the technical datasheets. The board can be accessed and programmed directly from a personal computer by using the serial connection of a USB cable and the dedicated interfacing software Code Composer Studio, but another great advantage of using this TI board is that a complete support package in present also in the Matlab & Simulink suite, asset that makes possible to program it using a Simulink model with the dedicated Embedded Coder, which generates the necessary code directly from the model. This approach is indeed used in the present work as explained in 6.1.3.

The TI LaunchPad can also be programmed directly from Simulink, in fact this approach has been used in the present thesis work. The rationale is to generate the C, C^{++} code from a Simulink model and to deploy it to the target hardware. The first prerequisite is to work both with the TI CCS IDE and controlSUITE. The following Matlab libraries are then needed and utilized:

1. *MATLAB Coder*: compiler that allows generating C and C++ code from MATLAB code.
2. *Simulink Coder*: compiler that allows generating C and C++ code from Simulink code.
3. *Embedded Coder*: adds support for custom targets to the Simulink Coder.
4. *Embedded Coder Support Package for Texas Instruments C2000 Processors*: adds support for the specific family of microcontrollers, including the F28379D Launch-Pad.

6.1.2 Single-Board Computer Unit

The Microcontroller Unit (MCU) works with single-precision floating-point unit. This is a limitation for certain applications where complex and accurate calculations require the double precision. To solve such shortcoming the Hardware suite is equipped also with an open-source single board computer, namely the BeagleBone Black. The BeagleBone Black boots Debian operative system. It is important to remark that at the current time the Support Package of Matlab/Simulink® supports only Debian 7.9, which is actually a deprecated version of Debian. Nevertheless, BeagleBone Black can boot from MicroSD, where Debian 7.9 was flashed to perform PIL tests, as will be discussed in Section 6.1.3. The board features:

- AM335x 1GHz ARM® Cortex-A8
- 512MB DDR3 RAM
- 4GB 8-bit eMMC on-board flash storage
- 3D graphics accelerator
- NEON floating-point accelerator
- 2x PRU 32-bit microcontrollers

Beside the already mentioned Support Packages for autocoding, the necessary Matlab add-ons are necessary to run PIL simulations with BeagleBone Black Hardware:

1. *Embedded Coder Support Package for BeagleBone Black Hardware*: enables to create and run Simulink models on BeagleBone Black hardware.

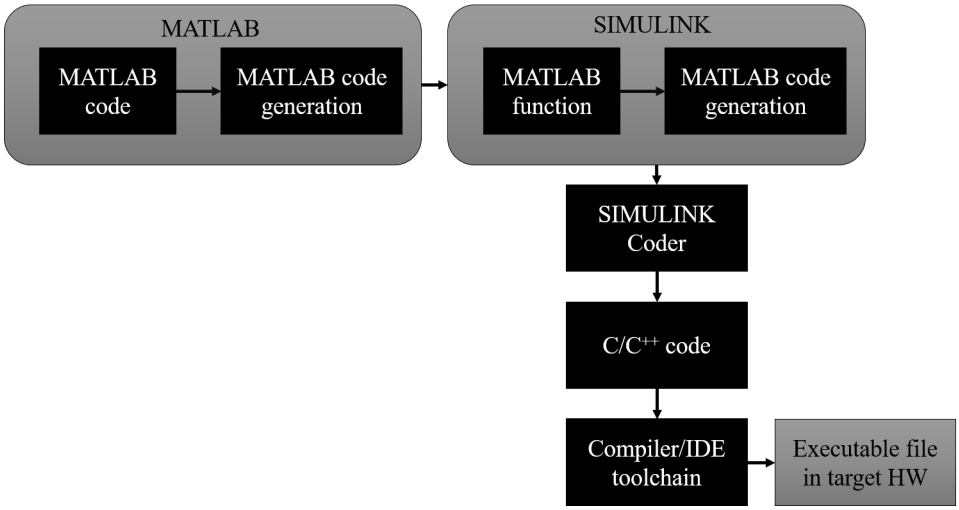


Figure 6.3: Embedded Coder workflow.

The support package includes a library of Simulink blocks for configuring and accessing BeagleBone Black peripherals and communication interfaces.

2. *ARM Cortex-A Support from Embedded Coder*: support for ARM Cortex-A processors by generating processor-optimized code using Ne10 project libraries (Ne10) with the GNU Compiler Collection (GCC) compiler.

6.1.3 Porting Procedure

The porting procedure is executed using the Matlab/Simulink Support Package for both the boards. In particular the Embedded Coder allows the user to autocompile the Simulink code into the target hardware, without requiring deep coding knowledge. This helps the Space community in the rapid prototyping activities. The workflow of the Embedded Coder is schematized in Fig. 6.3. Running a model in external mode or PIL validation allows to rapidly deploy the code into the hardware and to perform parameter tuning while running. To prepare the deployment of the code, the following settings must be changed:

- Simulation Mode: External or Normal (PIL block generated)
- Model Configuration Parameters → Hardware Implementation → Hardware board
- Target Hardware Resources → Build Options → Device Name
- Target Hardware Resources → Clocking → Tick Use Internal Oscillator

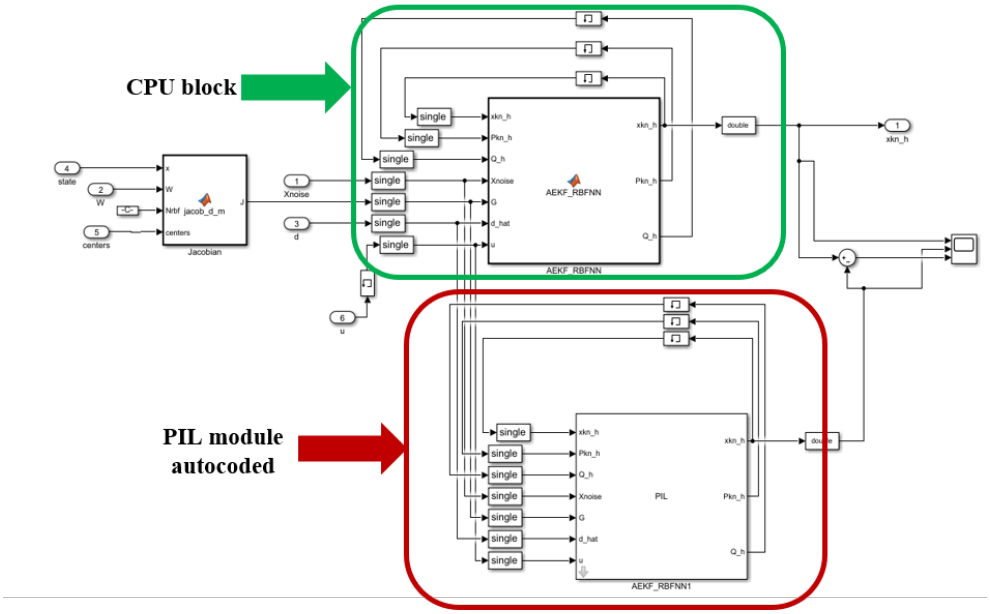


Figure 6.4: Processor-In-the-Loop (PIL) validation framework in MATLAB/Simulink.

- Target Hardware Resources → SCIA → Pin Ass. (Tx) = GPIO42, Pin Ass. (Rx) = GPIO43 required only for the Microcontroller Unit.
- Target Hardware Resources → External Mode → Set the correct COM port
- For PIL simulations: Code generation → Verification → Advanced parameters → Create block: PIL

The same procedure can be executed to set up the BeagleBone Black target interface. Note that the Support Package from Simulink requires to boot from Debian 7.9 and not later versions. The BeagleBone Black runs Debian Linux distribution, hence it is slightly more cumbersome to set-up all the necessary libraries. First of all, it is important to establish SSH communication with the board. Also, it is critical to connect the board to Internet and run the package update routine (*apt-get update*). Given that Debian 7.9 is obsolete, it is recommended to force installation of the packages required by the Support Package. After setting up the BeagleBone Black, the Matlab/Simulink target hardware configuration can be run following the aforementioned procedure. The result is a Simulink project where PIL autocoded module is compared against the CPU-executed block, as shown in Fig. 6.4.

6.2 Processor-In-the-Loop Validation

The experimental tests with the Processor-In-the-Loop focused on evaluating the feasibility of running the developed algorithms into representative hardware. In particular, the metrics used to evaluate the simulations are defined as:

- Execution times of each sub-routine that build up the algorithms ($t_e[s]$)
- Duty cycle and resource utilization with respect to task execution time. It is calculated as the execution times divided by the task allocated time ($DC_e[\%]$)
- Numerical discrepancy with respect to CPU output (ϵ)

The required threshold to be satisfied depend on the mission application and design. Nevertheless, the three metrics are evaluated comprehensively. For instance, the execution times are related with the duty cycle: a threshold of 10% for task duty cycle can be regarded as acceptable. Regarding the numerical discrepancy, it is important to bear in mind that the control action is calculated and fed to actuators in real mission. The sensitivity of such actuators establish the acceptable discrepancy threshold. For this reason, a threshold of $1 \cdot 10^6 \frac{m}{s^2}$ is set, assuming a conservative value for control sensitivity of current actuators. All the algorithms presented in the Thesis have been part of the PIL validation campaign. Nevertheless, Recurrent Neural Network still lack support for hardware implementation. This problem is quite extended for representative flight hardware. Indeed, for instance, Intel Openvino framework, used for optimizing code deployment for Intel board (e.g. Myriad 2), does not offer a full support for Recurrent Neural Networks.

Neural-Network Artificial Potential Field validation The presented simulations based on the NNAPF algorithm, comprising the whole RBFNN-GNC architecture, were simulated using a Desktop computer using Intel® Core^{TM} i5-3470 CPU @3.20 GHz. The computational time for a single step execution of the NNAPF is < 50 ms.
The NNAPF has been developed for an on-board applications, hence it is critical to evaluate its computational burden on relevant processor and hardware. The building blocks of such algorithm encompass the online learning, the artificial potential field guidance, the adaptive EKF for navigation and the Lyapunov controller. The average execution times of the autonomous GNC routines are limited and fully compatible with on-board implementation, as reported in Tab. 6.5. The NNAPF runs at 10 Hz: almost continuous control. For such sample time, the resource utilization is roughly $\sim 2\%$. Fig. 6.5 shows the discrepancy in the calculated control by the MCU and the Desktop simulator. The values are in the order of $\sim 10^{-12} \frac{m}{s^2}$, which can be assumed to be numerical error, thus validating the PIL test.

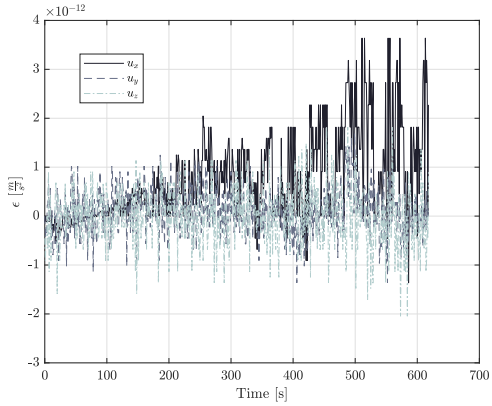


Figure 6.5: Discrepancy in calculation between PIL and MIL simulations for NNAPF run in MCUs F28379D.

Table 6.2: Average and maximum execution time of GNC routines using a single core TMS320C28x 32-Bit CPUs @200 MHz of TI C2000-Delfino MCUs F28379D

Routine	Average $[ms]$	Max $[ms]$
rbfnn(.)	0.89	0.90
aekf(.)	0.81	0.82
apf(.)	0.32	0.32
ctrl(.)	0.28	0.33

Table 6.3: Average and maximum execution time of MBRL routines using a BeagleBone Black single-board computer.

Routine	Average $[ms]$	Max $[ms]$
ann(.)	0.89	0.90
mbrl.quadprog(.)	191.10	272.20
mbrl.fmincon(.)	290.10	337.10

Model-Based Reinforcement Learning validation The Model-Based Reinforcement Learning algorithm relies on optimization techniques, in the same fashion as Model Predictive Control. Analogously, the Inverse reinforcement Learning features nested optimization to generate the collision avoidance constraint. On-board optimization is a delicate task to be executed on-board and certainly requires more computational power with respect to NNAPF. To be able to deploy the code to the target hardware with acceptable outcome, *double-precision* floating-point unit was required. The MCU does not support *double-precision*, hence it was necessary to complement the hardware suite with the single-board computer BeagleBone Black. The MBRL entails the artificial neural network learning and the optimization routine for planning and control. In particular, two solvers were used from Matlab suite, namely *quadprog* and *fmincon*. The reason is that, if no collision avoidance is implemented, the problem can be recast into quadratic programming formulation that guarantees rapid convergence and existence of the solution. For *quadprog* the *active-set* has been chosen as solver algorithm in MATLAB. The collision avoidance constraint is nonlinear, even if convexified as shown in Chapter 4. The constraint is nonlinear, hence the problem transforms into a quadratic optimization with nonlinear constraints. The problem is solved using *fmincon* in MATLAB library implementing *sequential quadratic programming*. The *sqp* has been selected for the medium-scale problem in the receding horizon optimization as well as for its efficiency and robustness. A comparison between the execution of the two methods is reported in Fig. 6.8. As expected, the nonlinear optimization requires more computational time to execute. The maximum number of iterations was set to 1000 in order not to enter infinite loops in the target hardware. The resource utilization grows up to $\sim 0.5\%$. The comparison of the execution times normalized frequency is shown in Fig. 6.9. An important remark is that both the distributions are relatively narrow, meaning that the confidence on the mean is high (see Fig. 6.8) and the outliers still deliver acceptable results. However, one can clearly see from Fig. 6.9 how the collision avoidance constraint may increase the computational time when it is constraining severely the solution (e.g. very close agents). Indeed, the distribution for *fmincon-sqp* is more spread out.

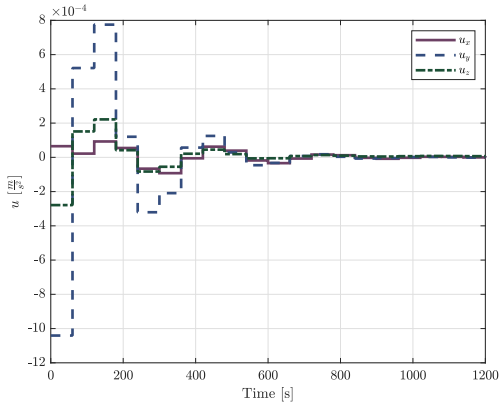


Figure 6.6: Control output delivered by the embedded execution.

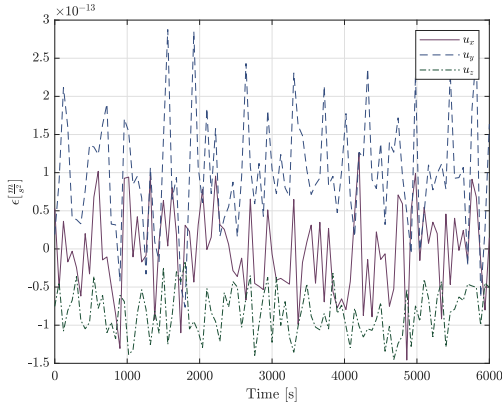


Figure 6.7: Discrepancy in calculation between PIL and MIL simulations for MBRL run in BeagleBone Black single-board computer.

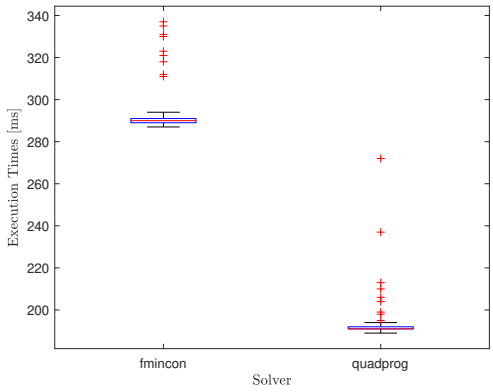


Figure 6.8: Statistics of execution times for the optimization solver used in the embedded application.

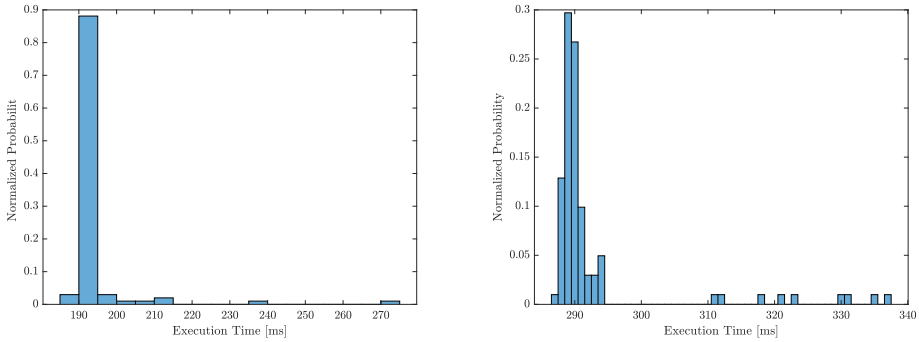


Figure 6.9: Normalized probability for the execution times profiled in the PIL simulation.From left to right, *quadprog* and *fmincon* are shown

The MBRL and IRL have been developed for an on-board applications, hence it is critical to evaluate its computational burden on relevant processor and hardware. Fig. 6.6 shows the control action calculated by the Single Board Computer (SBC) and delivered to the orbital simulator running in the Desktop computer. The average execution times of the autonomous GNC routines are limited and fully compatible with on-board implementation, as reported in Tab. 6.3. The MBRL runs at $\frac{1}{60}$ Hz, which means that the control is output every minute. For such sample time, the resource utilization is roughly $\sim 0.25\%$. Beside evaluating the computational times, it is critical to check whether the performed calculation are compatible with the results from the MIL simulations. Fig. 6.7 shows the discrepancy in the calculated control by the SBC and the Desktop simulator. The values are in the order of $\sim 10^{-13} \frac{m}{s^2}$, which can be assumed to be an acceptable numerical error, thus validating the PIL test.

Considering execution times, task duty cycle and numerical discrepancy, the algorithms have been successfully validated in PIL tests.

The hardware presented in this chapter is representative of the hardware integrated in the friction-less 5DoF facility installed at Politecnico di Milano [81] [82]. Naturally, the algorithm development follows the same workflow adopted here: the numerical algorithms are coded in Matlab/Simulink®, tested in Desktop computers and finally automatically ported into the target hardware using the dedicated coder. This enables fast prototyping and hardware validation of the selected algorithms.

CHAPTER 7

Conclusions

Passano gli anni, i mesi,
e se li conti anche i minuti,
é triste trovarsi adulti
senza essere cresciuti.

— Fabrizio De André

The research process does not have inherently a conclusion. However, what we can try to explore is part of the question: *how far have we gone?* This Chapter draws the tentative conclusions of the work with respect to the research objectives presented in Chapter 1. Section 7.1 reports the major developments and results of the research work and discusses them to draw insights on the research problem. Finally, Section 7.2 tries to pave the way for future steps by delivering some recommendations on the work to be done. Because there is always work to be done.

7.1 Major Results & Findings

The research work followed the logical scheme presented in Chapter 1. The general and overwhelming problem of increasing the autonomy of distributed systems while performing proximity operations using Artificial Intelligence (AI) has been split into several subproblems. First, the critical block on which the whole Guidance, Navigation & Control (GNC) system is built has been analyzed: the on-board dynamical model. The representation of the dynamics that is available on-board is used for all the algorithms regarding Guidance, Navigation & Control (GNC) system: for instance, the navigation EKF a-priori estimate rely on the dynamical model. Three methods have been developed in order to construct an autonomous algorithm capable of refining and learning the dynamical models of the environment in which they are flying. Thanks to the outstanding flexibility and adaptivity of the Artificial Neural Network (ANN), the dynamics reconstruction is accurately performed on-board.

The three methods all rely on ANN, with a different level of integration. The first method, named *neural-dynamics* encapsulates the whole dynamics into a ANN. Basically, the network is stimulated by the external environment, it makes a prediction of future state and trains by supervised learning using the actual measured state at the next time instant. Feed-Forward Networks are able to approximate the input-output relation after a brief initialization on linear models, nevertheless Recurrent Neural Networks have shown superior performances. This is due to the fact that RNN inherently possess a temporal behavior by recursively using information from past data. However, it has been found out that the embedded implementation of Recurrent Neural Network is still somewhat cumbersome and not yet supported by most of the representative hardware. Hence, they are penalized in terms of implementation feasibility, but this is certainly a matter of time.

The second method, here referred as *dynamics acceleration reconstruction* aims at enhancing a given analytical models (such as Clohessy-Wiltshire Model (CW) for relative dynamics, by approximating all the unmodeled terms or disturbances (e. g. drag, nonlinearities, gravity harmonics, etc.). For this, an original approach for state estimation and uncertainties estimation has been developed. The proposed algorithm relies on a Radial-Basis Function Neural Network (RBFNN) coupled with an Extended Kalman Filter (EKF). The proposed neural-network performs an online estimation of the disturbances acting on the spacecraft, which are included in the prediction step of the filter. The online learning algorithm exploits the state estimation worked out by the filter itself to update the neural network weights. Moreover, an innovation-based recursive filter architecture is employed. Simulation results show the capability of the proposed solution to reconstruct the dynamics of a spacecraft in elliptic orbits with the J_2 perturbation in an Earth orbit environment. More,

a fictitious perturbed case has been tested to showcase the quality of network reconstruction. Furthermore, the AI-augmented filter performance is compared to more classical approaches and tested on realistic scenarios, through statistical simulations. It has been shown how the robustness over very poor tuning of the state covariance matrix is consolidated in the algorithm. Satisfactory results have been obtained for the proposed solution in all the presented cases and the sensitivity analysis demonstrated the algorithm robustness in non-ideal situations.

The third method, here named *parametric dynamics reconstruction*, uses a RNN to estimate the parameters of a given dynamical expression. The most promising application has been reported in the Thesis, in which the coefficient of the spherical harmonics expansion of several asteroids have been estimated while flying. Even though the employed Hopfield Neural Network (HNN) has a recurrent structure, it is easily manageable analytically, hence the implementation does not pose major risks. Also, the comparison between EKF and a HNN for the parameter estimation of the gravitational field of small bodies was analyzed. The criticalities of the HNN for this task have been highlighted and consist in the tuning of the activation function though a parameter β. This parameter β results to be dependent on the distance to the body mainly and to have a slight dependence on the degree of irregularity of the flown body. In particular, for high irregular cases, a conservative choice of β should be made. These results have been then validated in the real gravitational environment of some selected bodies, namely Castalia, Kleopatra and Phobos. The case of a binary system (Didymos) is presented too: the re-formulation of the network' associated dynamics appears to be simple as well as all the consideration valid for a single body can be used for the tuning of the network. From the other hand, the same tests are performed with an augmented-EKF. The performance of the EKF, as expected, are good too doing this task but to scale the method to a new dynamical environment an ad-hoc tuning must be performed, over the model re-formulation. Moreover, from the computational point of view, the augmented-EKF result to be heavier than the couple EKF+HNN. Given the results, one could conclude that the use of a HNN online gravity field estimation is a good alternative to an EKF as well as can be use to validate the results of the filter itself.

The online dynamics reconstruction was necessary to enhance the traditional algorithms that work out the navigation, planning and control aspects. The first developed algorithm was based on Artificial Potential Field (APF) on Relative Orbital Elements (ROE) space. The algorithm is very powerful for on-board application, due to the capability of handling collision constraints efficiently. Moreover, it is very light computationally. The base algorithm was developed using a ROE dynamics, which was augmented with AI-reconstructed dynamics. The algorithm takes as input the Cartesian measurements of relative

states between satellites. The developed navigation filter and controller is coupled with a neural reconstructed term that encompass all the disturbances or nonlinearities not modeled in the on-board dynamical model. Such refinement of on-board dynamics enhance the whole GNC performance. Radial-Basis-Function Neural Networks have been employed for the inherent nonlinear neurons structure, which yields a faster learning process. The dynamics reconstruction works online with no mandatory previous training campaign. As presented in Chapter 6, the Neural Network Artificial Potential Field (NNAPF) algorithm has been tested numerically and in Processor-In-the-Loop (PIL) simulations using a single core TMS320C28x 32-Bit CPUs @200 MHz of TI C2000-Delfino MCUs F28379D unit. The numerical and PIL tests demonstrated:

- the neural reconstructed dynamics enables more accurate trajectory reconstruction and final target configuration with respect to Artificial Potential Field algorithm.

- the algorithm can be executed using limited computational power, thus making it suitable for on-board applications.

Despite all the promising features of the neural-aided Artificial Potential Field algorithm, two major drawbacks have been identified: on one hand, the APF may lead to instability depending on the tuning and type of required maneuvers; on the other hand, it does not optimize the maneuver control, potentially leading to high Δv cost, without a-priori estimate of the control effort. For this reason, an innovative strategy for the guidance and control of distributed formation has been developed. In particular, the method solves the shortcomings that arise from the partially known dynamical environment as well as the lack of knowledge of future trajectories of neighboring satellites during coordinated maneuvers. The trajectory and control is generated using a Model-Based Reinforcement Learning (MBRL) approach. A Model-Based Reinforcement Learning approach has been adopted for on-board planning and control of reconfiguration trajectories. The learning process is based on the loss function between prediction and measurements. The neural-reconstructed dynamics is used for the optimization at each sampling instant, taking into account a convex collision avoidance constraint. The first control action is executed until the new planning step, in the same fashion as Model-Predictive Control. This algorithm demonstrated promising results in the validation tests, in particular in those scenarios where simple dynamical models, such as Clohessy-Wiltshire, are not accurate enough for synthesizing the guidance and control. Reconfiguration trajectories generated by the MBRL algorithm require lower Δv for the same formation geometry variation. In addition, MBRL succeeded in generating trajectories when the standard MPC algorithm failed. The algorithm was tested in Earth-bounded orbital environment but

can be applied to scenarios where it is necessary to acquire knowledge of the dynamics, which is unknown or partially unknown until spacecraft arrival.

The distributed system architecture does not imply information sharing among the agents, thus it was necessary to implement an algorithm to prevent collision during proximity maneuvers. Two algorithms are explored to guarantee collision-free operations. Inverse Reinforcement Learning (IRL) reconstructs the cost function of each agent and predicts trajectories of concurring agents during the reconfiguration. The second method exploits a Long-Short Term Memory (LSTM) recurrent network to capture the dynamics and predict the trajectory. In this way, both natural and thrusted dynamics is managed when enforcing the collision-avoidance constraint. The results show that the proposed algorithms perform correctly and solve reconfiguration scenario that are challenging, or even fatal, for traditional algorithms. In particular, long-term horizon predictions are better captured by IRL, whereas short-term predictions are effectively represented by LSTM output.

> Employing Artificial Neural Networks for spacecraft operations can help in bridging the gap between space exploration and distributed autonomous flight. Being computationally light, online Artificial Neural Network aided algorithms can be deployed in micro-satellites, where the computational power is limited.

Also, all the presented algorithms adapt to the environment capturing the unmodeled terms delivering successful control where traditional and not adaptive algorithms may fail. This approach is presented for distributed systems but it is applicable in all the missions where the lack of prior knowledge (or partial) of the environment may jeopardize the on-board autonomy. The optimization-based algorithms have been tested and validated Processor-In-the-Loop (PIL) using a single-board computer BeagleBone Black. In particular, the execution times and resource utilization has been positively assessed to evaluate the feasibility of the embedded implementation.

At the end of the story, this is where we are. We can call it *the end* but *the next start* seems more appropriate.

7.2 Recommendations

It is certainly not sufficient one Ph.D. Thesis to cover all the aspects of autonomy in space for distributed system. This research work aims at building the foundation of a strategic approach for enhancing existent GNC algorithms with AI. The philosophy is to integrate rather than replace AI-techniques into well established GNC routine in order to solve their known shortcomings. Few recommendations for the next steps are here reported:

- the algorithms require the next validation step consisting in Hardware-In-the-Loop (HIL) tests. This is already planned to be performed in a frictionless facility at Politecnico di Milano, which is thoroughly described in [81, 82, 83]. The HIL tests allow to integrate real proximity measurements, such as RF ranging, to be processed by the AI-augmented GNC and to verify the correct execution of the algorithms.
- the Recurrent Neural Network (RNN) still partially lack support for implementation. Given their superior performance in dynamics learning, it is crucial to investigate methods and procedures to embed such models into relevant applications.
- last but not least, it would be interesting to advance in the autonomy level and implement AI-based methods for high-level tasks, such as *formation allocation* and *scheduling*.

The reported points are only few implications directly linked to the present Thesis. The list of applications that can benefit from introducing AI into traditional algorithms can potentially be never-ending and it is out of the scope of this work to present it comprehensively.

In the end, the aim of researching is to stimulate research, no matter what, no matter why.

Bibliography

[1] S. Silvestrini, A. Capannolo, M. Piccinin, M. Lavagna, and J. Gil-Fernandez, "Centralized Autonomous Relative Navigation of Multiple Spacecraft Around Small Bodies", in *AIAA Scitech 2020 Forum*, 2020, pp. 1–20. DOI: 10.2514/6.2020-1204.

[2] H. Curtis, *Orbital Mechanics for Engineering Students*. Elsevier, 2005.

[3] D. Wang, B. Wu, and E. K. Poh, *Satellite Formation Flying*. 2017, vol. 87, Intelligent Systems, Control and Automation: Science and Engineering.

[4] S. D'Amico, "Autonomous formation flying in low earth orbit", Ph.D. Dissertation, 2010.

[5] T. Guffanti, S. D'Amico, and M. Lavagna, "Long-term analytical propagation of satellite relative motion in perturbed orbits", *Advances in the Astronautical Sciences*, vol. 160, pp. 2387–2417, 2017.

[6] A. W. Koenig, T. Guffanti, and S. D'Amico, "New State Transition Matrices for Spacecraft Relative Motion in Perturbed Orbits", *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 7, pp. 1749–1768, 2017.

[7] H. Schaub, S. R. Vadali, J. L. Junkins, and K. T. Alfriend, "Spacecraft formation flying control using mean orbit elements", *Journal of the Astronautical Sciences*, vol. 48, no. 1, pp. 69–87, 2000.

[8] C. M. Lane and P. Axelrad, "Formation Design in Eccentric Orbits Using Linearized Equations of Relative Motion", *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 1, pp. 146–160, 2006.

[9] A. Colagrossi, J. Prinetto, S. Silvestrini, and M. Lavagna, "Sky visibility analysis for astrophysical data return maximization in HERMES constellation", *Journal of Astronomical Telescopes, Instruments, and Systems*, vol. 6, no. 4, pp. 1–25, 2020. DOI: 10.1117/1.JATIS.6.4.048001.

[10] A. Colagrossi, J. Prinetto, S. Silvestrini, M. Orfano, *et al.*, "Semi-Analytical Approach to Fasten Complex and Flexible Pointing Strategies Definition for Nanosatellite Clusters: The HERMES Mission Case from Design to Flight", in *70th International Astronautical Congress, 21-25 October 2019, Washington D.C., USA*, 2019.

[11] S. Silvestrini, V. Pesce, and M. Lavagna, "Distributed Autonomous Guidance , Navigation and Control loop for Formation Flying Spacecraft Reconfiguration", in *5th CEAS Conference on Guidance, Navigation and Control*, 2019, pp. 1–19.

[12] G. Gaias, J.-S. Ardaens, and O. Montenbruck, "Model of j2 perturbed satellite relative motion with time-varying differential drag", *Celestial Mechanics and Dynamical Astronomy*, vol. 123, no. 4, pp. 411–433, 2015.

[13] A. Pasquale, "Small bodies gravity field on-board learning and navigation", M.S. thesis, Politecnico di Milano, Milan, 2019.

[14] R. A. Werner, "Spherical harmonic coefficients for the potential of a constant-density polyhedron", *Computers & Geosciences*, vol. 23, no. 10, pp. 1071 –1077, 1997. DOI: https://doi.org/10.1016/S0098-3004(97)00110-6.

[15] J. L. Crassidis and J. L. Junkins, *Optimal estimation of dynamic systems*. 2004, pp. 1–591. DOI: 10.1201/b11154.

[16] V. Pesce, "Autonomous Navigation for Close Proximity Operations around Uncooperative Space Objects", PhD Thesis, Politecnico di Milano, 2018.

[17] 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*, no. December 2017, pp. 1–22, 2017.

[18] P Gurfil, M Idan, and N. J. Kasdin, "Adaptive neural control of deep-space formation flying", *Journal of Guidance Control and Dynamics*, vol. 26, no. 3, pp. 491–501, 2003.

[19] J. Bae and Y. Kim, "Adaptive controller design for spacecraft formation flying using sliding mode controller and neural networks", *Journal of the Franklin Institute*, vol. 349, no. 2, pp. 578–603, 2012.

[20] N. Zhou, R. Chen, Y. Xia, J. Huang, and G. Wen, "Neural network based reconfiguration control for spacecraft formation in obstacle environments", *International Journal of Robust and Nonlinear Control*, vol. 28, no. 6, pp. 2442–2456, 2018.

[21] D. Simon, "Training radial basis neural networks with the extended kalman filter", *Neurocomputing*, vol. 48, no. 1-4, pp. 455–475, 2002.

[22] S. C. Stubberud, R. N. Lobbia, and M. Owen, "An adaptive extended kalman filter using artificial neural networks", in *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, IEEE, vol. 2, 1995, pp. 1852–1856.

[23] X. Gao, X. Zhong, D. You, and S. Katayama, "Kalman filtering compensated by radial basis function neural network for seam tracking of laser welding", *IEEE Transactions on Control Systems Technology*, vol. 21, no. 5, pp. 1916–1923, 2013.

[24] A Stubberud, H Wabgaonkar, and S Stubberud, "A neural-network-based system identification technique", in *Decision and Control, 1991., Proceedings of the 30th IEEE Conference on*, IEEE, 1991, pp. 869–870.

[25] J. Dah-Jing and J.-J. Chen, "Neural network aided adaptive kalman filter for gps/ins navigation system design", in *Proc. 9th IFAC Workshop*, 2011, pp. 1–7.

[26] D.-J. Jwo and H.-C. Huang, "Neural network aided adaptive extended kalman filtering approach for DGPS positioning", *Journal of Navigation*, vol. 57, no. 3, pp. 449–463, 2004.

[27] N. Harl, K. Rajagopal, and S. N. Balakrishnan, "Neural Network Based Modified State Observer for Orbit Uncertainty Estimation", *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 4, pp. 1194–1209, 2013.

[28] R. Furfaro, R. Linares, V. Reddy, J. Simo, and L. Le Corre, "Modelling irregular small bodies gravity field via extreme learning machines", in *27th AAS/AIAA Spaceflight Mechanics Meeting*, 2017.

[29] S. Willis, D. Izzo, and D. Hennes, "Reinforcement learning for spacecraft maneuvering near small bodies", in *AAS/AIAA Space Flight Mechanics Meeting*, 2016, pp. 14–18.

[30] S. Sarno, J. Guo, M. D'Errico, and E. Gill, "A Guidance Approach to Satellite Formation Reconfiguration Based On Convex Optimization and Genetic Algorithms", *Advances in Space Research*, 2020. DOI: 10.1016/j.asr.2020.01.033.

[31] G. Di Mauro, D. Spiller, R. Bevilacqua, and F. Curti, "Optimal Continuous Maneuvers for Satellite Formation Reconfiguration in J2-perturbed Orbits", *2018 Space Flight Mechanics Meeting*, vol. 0216, no. January, pp. 1–20, 2018.

[32] W. Ren and R. Beard, "Decentralized Scheme for Spacecraft Formation Flying via the Virtual Structure Approach", *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 1, pp. 73–82, 2004.

[33] J. Chu, *Dynamics, Distributed Control And Autonomous Cluster Operations Of Fractionated Spacecraft*. 2015.

[34] M. Chernick and S. D'Amico, "New Closed-Form Solutions for Optimal Impulsive Control of Spacecraft Relative Motion", *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 2, pp. 301–319, 2016.

[35] S. R. Vadali and K. T. Alfriend, "Formation Establishment, Maintenance and Control", in *Distributed Space Missions for Earth System Monitoring*, 2013, pp. 1–675.

[36] D. Izzo and L. Pettazzi, "Autonomous and Distributed Motion Planning for Satellite Swarm", *European Space Agency, (Special Publication) ESA SP*, vol. 30, no. 603, pp. 727–736, 2005.

[37] Q. Li, B. Zhang, J. Yuan, and H. Wang, "Potential function based robust safety control for spacecraft rendezvous and proximity operations under path constraint", *Advances in Space Research*, vol. 62, no. 9, pp. 2586–2598, 2018. DOI: 10.1016/j.asr.2018.08.003.

[38] L. M. Steindorf, S. D'Amico, J. Scharnagl, F. Kempf, and K. Schilling, "Constrained low-thrust satellite formation-flying using relative orbit elements", *Advances in the Astronautical Sciences*, vol. 160, pp. 3563–3583, 2017.

[39] H. Schaub and K. T. Alfriend, "Hybrid Cartesian and Orbit Element Feedback Law for Formation Flying Spacecraft", *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 2, pp. 387–393, 2002.

[40] V. Pesce, S. Silvestrini, and M. Lavagna, "Radial basis function neural network aided adaptive extended Kalman filter for spacecraft relative navigation", *Aerospace Science and Technology*, vol. 1, p. 105 527, 2020. DOI: 10.1016/j.ast.2019.105527.

[41] T. Wahl and K. Howell, "Autonomous guidance algorithms for formation reconfiguration maneuvers", in *AAS/AIAA Astrodynamics Specialist Conference*, Columbia River Gorge, Washington, August 21 - 24, 2017.

[42] D. Morgan, S. J. Chung, and F. Y. Hadaegh, "Model predictive control of swarms of spacecraft using sequential convex programming", *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 6, pp. 1725–1740, 2014. DOI: 10.2514/1.G000218.

[43] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 5. 2012, vol. 91, pp. 1689–1699. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

[44] R. Lippmann, *Neural Networks, A Comprehensive Foundation*, 04. 2005, vol. 05, pp. 363–364. DOI: 10.1142/s0129065794000372.

[45] Y. Wu, H. Wang, B. Zhang, and K.-L. Du, "Using Radial Basis Function Networks for Function Approximation and Classification", *ISRN Applied Mathematics*, vol. 2012, no. March, pp. 1–34, 2012.

[46] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters", *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963. DOI: 10.1137/0111030.

[47] S. Silvestrini and M Lavagna, "Model-based reinforcement learning for distributed path planning", in *15th Symposium on Advanced Space Technologies in Robotics and Automation*, ESA, ESTEC, Noordwijk, 2019.

[48] J. Park and I. W. Sandberg, "Universal Approximation Using Radial-Basis-Function Networks", *Neural Computation*, vol. 3, no. 2, pp. 246–257, 1991.

[49] K. Reif and R. Unbehauen, "The extended kalman filter as an exponential observer for nonlinear systems", *IEEE Transactions on Signal processing*, vol. 47, no. 8, pp. 2324–2328, 1999.

[50] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[51] V. Pesce, M. Lavagna, and R. Bevilacqua, "Stereovision-based pose and inertia estimation of unknown and uncooperative space objects", *Advances in Space Research*, vol. 59, no. 1, pp. 236–251, 2017.

[52] V. Pesce, R. Opromolla, S. Sarno, M. Lavagna, and M. Grassi, "Autonomous relative navigation around uncooperative spacecraft based on a single camera", *Aerospace Science and Technology*, 2018.

[53] V. Pesce, M. F. Haydar, M. Lavagna, and M. Lovera, "Comparison of filtering techniques for relative attitude estimation of uncooperative space objects", *Aerospace Science and Technology*, vol. 84, pp. 318–328, 2019.

[54] C. L. Pasqualetto, R. Fonod, and E. Gill, "Review of the robustness and applicability of monocular pose estimation systems for relative navigation with an uncooperative spacecraft", *Progress in Aerospace Sciences*, 2019.

[55] S. Akhlaghi, N. Zhou, and Z. Huang, "Adaptive adjustment of noise covariance in kalman filter for dynamic state estimation", *arXiv preprint arXiv:1702.00884*, 2017.

[56] R. Burns, *Advanced control engineering*. Elsevier, 2001.

[57] W. MacMillan, *The Theory of Potential*. Dover Publications, 1958.

[58] D. Scheeres, *Orbital motion in strongly perturbed environments: application to asteroid comet and planeraty satellite orbiters*. Springer, 2012.

[59] Y. Takahashi, "Gravity field characterization around small bodies", Ph.D. Dissertation, University of Colorado Boulder, 2013.

[60] H. Alonso, T. MendonÃ§a, and P. Rocha, "Hopfield neural networks for on-line parameter estimation", *Neural networks : the official journal of the International Neural Network Society*, vol. 22, pp. 450–62, Apr. 2009.

[61] M. Atencia, G. Joya, and F. Sandoval, "Parametric identification of robotic systems with stable time-varying hopfield networks", *Neural Computing and Applications*, vol. 13, pp. 270–280, Dec. 2004.

[62] A. Pasquale, S. Silvestrini, A. Capannolo, and M. Lavagna, "Non-uniform gravity field model on board learning during small bodies proximity operations", in *70th International Astronautical Congress*, 2019.

[63] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons", *Proceedings of the National Academy of Sciences*, vol. 81, no. 10, pp. 3088–3092, 1984.

[64] Abe, "Theories on the hopfield neural networks", in *International 1989 Joint Conference on Neural Networks*, 1989, 557–564 vol.1.

[65] M. Atencia, G. Joya, and F. Sandoval, "Hopfield neural networks for parametric identification of dynamical systems", *Neural Processing Letters*, vol. 21, pp. 143–152, Apr. 2005.

[66] Y. Hernández-Solano, M. Atencia, G. Joya, and F. Sandoval, "A discrete gradient method to enhance the numerical behaviour of hopfield networks", *Neurocomput.*, vol. 164, no. C, pp. 45–55, Sep. 2015.

[67] L. Dell'Elce, N. Baresi, S. Naidu, L. Benner, and D. Scheeres, "Numerical investigation of the dynamical environment of 65803 didymos", *Advances in Space Research*, 2017.

[68] R. Munguía, S. Urzua, and A. Grau, "EKF-based parameter identification of multi-rotor unmanned aerial vehiclesmodels", *Sensors*, vol. 19, no. 19, pp. 1–17, 2019.

[69] J. Chu, "Dynamics, distributed control and autonomous cluster operations of fractionated spacecraft", 9789461865113, Ph.D. Dissertation, TU Delft, 2015.

[70] G. Di Mauro, D. Spiller, R. Bevilacqua, and F. Curti, "Optimal Continuous Maneuvers for Satellite Formation Reconfiguration in J2-perturbed Orbits", *2018 Space Flight Mechanics Meeting*, vol. 0216, no. January, pp. 1–20, 2018. DOI: 10.2514/6.2018-0216.

[71] P. Abbeel and A. Y. Ng, "Apprenticeship Learning via Inverse Reinforcement Learning", in *Proceedings of the 21 st International Conference on Machine Learning*, Banff, Canada, 2004.

[72] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization", in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, New York, NY, USA: JMLR.org, 2016, pp. 4–58.

[73] R. Linares and R. Furfaro, "Space Objects Maneuvering Detection and Prediction via Inverse Reinforcement Learning", in *Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, Jan. 2017, 46, p. 46.

[74] A. Zurita, I. Corbella, M. Martin-Neira, M. A. Plaza, F. Torres, and F. J. Benito, "Towards a smos operational mission: Smosops-hexagonal", *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6(3), pp. 1769–1780, 2013. DOI: doi:10.1109/JSTARS.2013.2265600..

[75] S. D'Amico and O. Montenbruck, "Proximity Operations of Formation-Flying Spacecraft Using an Eccentricity/Inclination Vector Separation", *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 3, pp. 554–563, 2006. DOI: 10.2514/1.15114.

[76] B. Taskar, C. Guestrin, and D. Koller, "Max-margin markov networks", in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. K. Saul, and B. Schölkopf, Eds., MIT Press, 2004, pp. 25–32.

[77] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning", in *23rd International Conference on Machine Learning*, 2006, pp. 729–736. DOI: 10.1145/1143844.1143936.

[78] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory", *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[79] S Silvestrini and M. Lavagna, "Model-Based Reinforcement Learning for Distributed Path-Planning", in *ASTRA*, 2019, pp. 0–7.

[80] A Pellacani, M Graziano, and M Suatoni, "Design , Development , Validation and Verification of GNC technologies", in *EUCASS2019*, 2019. DOI: 10.13009/EUCASS2019-38.

[81] P. Visconti, S. Silvestrini, and M. Lavagna, "Dance: a Frictionless 5 DOF Facility For GNC Proximity Maneuvering Experimental Testing And Validation", in *69th International Astronautical Congress*, 2018, pp. 1–5.

[82] D. Ottolina, S. Silvestrini, and M. Lavagna, "DANCE: Design and Characterization of a 5 DOF Facility for Relative GNC", in *ASTRA*, 2019.

[83] S. Silvestrini, D. Ottolina, and M. De Gasperin Riccardo Lavagna, "DANCE: Integration and Avionics Testing of 5 DOF Experimental Facility for Relative GNC", in *71st International Astronautical Congress*, 2020, pp. 1–5.

Colophon

This thesis was typeset with LaTeX and BibTeX, using a typographical look-and-feel created by Stefano Silvestrini. The style was inspired by A. Colagrossi, D.A. Dei Tos `PhD_Dis` and by J. Stevens, L. Fossati `phdthesis` styles.