Enhancing Collision Risk Assessment with Deep Learning Models

AI-Driven Satellite Collision Avoidance with Physics-Informed Models

Master Thesis

Andrea Sanchez Mediavilla

Delft University of Technology

TU Delft

Enhancing Collision Risk Assessment with Deep Learning Models

AI-Driven Satellite Collision Avoidance with Physics-Informed Models

by

Andrea Sanchez Mediavilla

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on Friday December 6, 2024 at 2:00 PM.

Student number:	5697506	
Project duration:	January 2024 – November 2024	
Thesis committee:	Dr. Ir. Erwin Mooij	TU Delft, supervisor
	Dr. Ir. Jian Guo	TU Delft, chair
	Dr. Ir. Joao de Teixeira da Encarnacao	TU Delft, external member

Cover: Space junk orbiting around the Earth - Conceptual of pollution around our planet by NASA

TU Delft

Preface

This document presents the culmination of my journey at TU Delft to obtain the title of MSc Aerospace Engineering. This journey has been both challenging and rewarding, it has provided me with some of my fondest memories as well as my most frustrating experiences. This institution has helped me grow as an engineer, as a researcher, and, most importantly, as a person. These two years of hard work have only increased my interest and passion for aerospace engineering, with a renewed focus on sustainable operations in space. This has driven the topic of my research, alongside a desire to learn more about machine learning which has shown incredible potential for different engineering fields, and I am thrilled that this thesis has allowed me to learn about the latter while being part of the large community of engineers working on making space more sustainable.

I would like to present my gratitude to everyone who has been by my side throughout this project. First of all, I would like to thank my supervisor, Erwin Mooij. Your advice and help have saved my life (no hyperboles are present here) on many occasions. You have always shown me that I could count on you if I encountered any issues and you have motivated me to work hard and produce the best results. From those early morning (but not too early) meetings, to our complaining sessions and all those hours you have sat with me to debug complicated codes even when you were extremely busy, I will be forever grateful to you. I would also like to thank Jian Guo and Joao de Teixeira da Encarnacao for agreeing to be a part of my committee, and Dominic Dirkx and Steve Gehly for answering all my questions and helping me when I was stuck on my research. I would like to also extend my gratitude to Mireia Leon Dasi for meeting with me and solving my doubts when working with the algorithm she developed, even when she was busy.

Next, I want to move forward to all of my friends who have had to listen to me talk endlessly about my thesis. Firstly, I could not have done this without my flatmates in Kitchen 5a and Flat 121: I will never forget all those nights in the kitchen where we would spend hours talking, laughing, and complaining about life. I will also forever remember my fellow countrymen (and women) with whom I spent countless hours working on assignments together. You have made this journey more enjoyable, I never thought I would have so much fun working on projects! I am also thankful to everyone on floor 10 and all those movie nights which helped me take my mind off the thesis and all the stress. To everyone in Glasgow and Palma: thanks for being by my side even when we were so far away from each other, and thank you for never doubting me even when I had lost faith in myself.

Finally, but not less importantly, I would like to thank my parents, Santiago and Begoña, who have been an unwavering support in my life. You have sacrificed so much for me to open the doors to endless possibilities to work for my future. *Os quiero muchísimo y os estaré eternamente agradecida.*

Andrea Sanchez Mediavilla
Palma de Mallorca, November 2024

Abstract

Recent years have seen the exponential growth of the amount of artificial objects orbiting the Earth. From satellites to small pieces of space debris, this poses a threat to current and future missions due to an increase in the possibility of collisions between these objects. Since space debris has the potential to cause substantial damage, different measures are investigated to make space operations more sustainable. Amongst these measures, there is an interest in the development of methods to detect possible collisions between objects in space to deploy avoidance manoeuvres and hence mitigate the creation of more space debris. These methods, however, require highly accurate propagations, with thousands of screenings needed to model the uncertainties for each tracked object, leading to large computational times. Consequently, there is a lot of effort put into researching new methods that are highly accurate and, at the same time, require a lower time to compute. The goal of this research is therefore to deploy a model that can enhance the computation of the probability of collision between satellites for faster and more accurate collision risk assessment.

Traditional methods require complex numerical models with a large computational load to simulate the space environment. To avoid these long propagation times, machine learning algorithms could be used on simpler simulation environments and achieve a similar accuracy to these traditional methods. The use of Artificial Intelligence (AI) in the field of orbit prediction and collision detection has shown potential, mainly because these complex simulation environments do not need to be explicitly modelled with an AI-based approach. As such, the methodology developed in this work relies on the deployment of neural networks to predict the orbits of objects in space with the ultimate goal of calculating the probability of collision between different objects. The literature on this topic has increased in recent years, yet there is a research gap in the calculation of collision probabilities using neural networks and proper testing with real-life scenarios. There is a newly developed hybrid Differential Algebra and Gaussian Mixture Model method which can provide accurate state and uncertainty propagations for satellites at any geometry. This model only requires one iteration to work, thus allowing for faster collision assessments than traditional methods. This has been selected as the algorithm against which to validate the model built for this thesis. However, since it was only recently developed, some extensions to provide more accurate and robust answers are required. This model lacks the modelling of the Solar Radiation Pressure acceleration, which affects mostly satellites at high altitudes, thus it is included as part of a model extension. This extension is tested and verified using a sensitivity analysis, which shows an improvement in the accuracy of the results for a range of altitudes with respect to the original model. A real-life scenario reveals that this extension shows very little improvement for satellites in Low Earth Orbit (LEO) but a significant one at higher altitudes.

With the extension to the DA-GMM algorithm verified, the AI-based approach is developed. The methodology includes the selection of Physics-Informed Neural Networks (PINNs) as the type of neural network with which to work. These algorithms introduce a physics loss, a measure of how much the predicted outcomes of the model deviate from the actual physical laws that govern the studied system, and thus ensure that these laws are conserved. The main architecture of PINNs can change based on the task considered. As such, Long Short-Term Memory (LSTM) algorithms have been selected as the main architecture of the algorithm due to their suitability for time-series forecasting. With the satellite orbit predicted by the neural network, the uncertainties need to be studied to calculate the collision probability. A pseudo-Monte Carlo analysis can be performed with the neural network to estimate the state uncertainty, with the residuals from the algorithm being modelled as uncertainties for more accurate predictions. The probability of collision between two objects can then be calculated using the state prediction and uncertainty estimation for each of them. The method selected for this computation is the time integration of the collision probability rate.

Initially, the machine learning algorithm is trained with data from a single satellite to assess the capacities and limitations of the model. Once the best architecture is selected, the orbit prediction of this algorithm is tested and it is found that the model is able to predict the state of the satellite with a

maximum error of 1% over a 16-hour propagation, which is lower than the DA-GMM. The actual application of the model is tested on a real-life scenario, for which the Cosmos-2251/Iridium-33 collision is chosen. This algorithm can predict the collision probability within the same order of magnitude as the DA-GMM at a fraction of the computational time. This algorithm is shown to be around 500 times faster than the DA-GMM for orbit propagation and over 10,000 times faster for the risk probability calculation. On top of that, it is shown that simplified perturbations models can be used in this approach with an accuracy loss of around 2%. Next, a synthetic crash scenario reveals that the orbit predictions remain stable over a period of at least 72 hours, indicating that this method is suitable for longer lead times. Finally, it is demonstrated that techniques such as transfer learning can be used to build a tool which can generalise to satellites at different orbital regions and with different geometries.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
18SDS	18th Space Defence Squadron
19SDS	19th Space Defence Squadron
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
CAM	Collision Avoidance Manoeuvre
CDM	Conjunction Data Message
CNN	Convolutional Neural Network
CSSI	Center for Space Standards and Innovation
DA-GMM	Differential Algebra-Gaussian Mixture Model
DACE	Differential Algebra Computational Toolbox
ECEF	Earth-Centred Earth-Fixed reference frame
ECI	Earth-Centred Inertial reference frame
ESA	European Space Agency
FFNN	Feed-Forward Neural Networks
GEO	Geostationary Earth Orbit
GME	Gaussian Mixture Element
GP	Gaussian Processes
GPU	Graphics Processing Unit
HAC	Highly Accurate Catalog
HEO	Highly Elliptical Orbit
ILRS	International Laser Ranging Service
ISS	International Space Station
LEO	Low-Earth Orbit
LSTM	Long Short Term Memory
MEE	Modified Equinoctial Elements
ML	Machine Learning
MSE	Mean Square Error
NASA	National Aeronautics and Space Administration
NRLMSISE-00	US Naval Research Laboratory Mass Spectrometer and Incoherent Scatter Radar to Exosphere released in the year 2000
ODE	Ordinary Differential Equation
O/O	Owner/Operator
PDE	Partial Differential Equation
PDF	Probability Density Function
PINN	Physics-Informed Neural Networks
POD	Precise Orbit Determination
RNN	Recurrent Neural Network
RSO	Resident Space Object
RSW	Radial, Along-track, Cross-Track reference frame
SDP4	Simplified Deep space Perturbations 4
SGD	Stochastic Gradient Descent
SGP4	Simplified General Perturbations 4

Abbreviation	Definition
SOCRATES	Satellite Orbital Conjunction Reports Assessing Threatening Encounters in Space
SPICE	Spacecraft Planet Instrument C-matrix Events
SRP	Solar Radiation Pressure
SSA	Space Situational Awareness
SSN	Space Surveillance Network
SST	Space Surveillance and Tracking
SVM	Support Vector Machine
TCA	Time of Closest Approach
TLE	Two-Line Element
Tudat	TU Delft Astrodynamics Toolbox
UN	United Nations
USM	Unified State Model
USSPACECOM	U.S. Space Defence Command
UTC	Coordinated Universal Time

Symbols

Symbol	Definition	Unit
a	Semi-major axis	[m]
$\boldsymbol{a}$	Acceleration vector	[m/s^2]
$\boldsymbol{a}_{aero}$	Aerodynamic acceleration	[m/s^2]
$\boldsymbol{a}_{kep}$	Acceleration of a keplerian orbit	[m/s^2]
$\boldsymbol{a}_{SRP}$	Solar radiation pressure acceleration	[m/s^2]
C_D	Drag coefficient	[-]
$\bar{C}_{lm}$	Normalised cosine spherical harmonics coefficient	[-]
C_R	Reflectivity coefficient	[-]
$CSRP$	Solar Radiation Pressure coefficient	[m^2/kg]
c	Speed of Light	[m]
E	Eccentric anomaly	[rad]
e	Eccentricity	[-]
$\boldsymbol{F}$	Force vector	[N]
f_g	Geometry function	[-]
$\boldsymbol{f}_{pert}$	Perturbing acceleration	[m/s^2]
$\boldsymbol{h}$	Angular momentum vector	[kg·m^2/s]
i	Inclination	[rad]
K	Ballistic coefficient	[m^2/kg]
l	Degree of the spherical harmonics	[-]
M	Mean anomaly	[rad]
m	Mass	[kg]
m	Order of the spherical harmonics	[-]
$\boldsymbol{N}$	Vector normal to the orbital plane	[-]
n	Mean motion	[rad/s]
$\boldsymbol{P}$	Covariance matrix	various
P_c	Collision probability	[-]
$\bar{P}_{lm}$	Normalised Legendre polynomial	[-]
p	Semi-latus rectum	[m]
p_c	Collision probability rate	[-]
$p_g(x;\mu,\boldsymbol{P})$	Gaussian distribution	various
R	Radius	[m]
$R_\oplus$	Radius of the Earth	[m]
$R_\odot$	Radius of the Sun	[m]

Symbol	Definition	Unit
R_p	Apparent radius of the Sun	[m]
$\boldsymbol{r}$	Position vector	[m]
u	Argument of latitude	[rad]
$\bar{S}_{lm}$	Normalised sine spherical harmonics coefficient	[-]
S_{ref}	Reference area	[m^2]
T	Period	[s]
t	Time	[s]
U	Gravitational potential	[J/kg]
$\boldsymbol{v}$	Velocity vector	[m/s]
W	Solar flux	[W/m^2]
α	Angle of attack	[rad]
α	Learning rate	[-]
β	Sideslip angle	[rad]
ε	Reflectivity coefficient	[-]
η	Shadow function	[-]
θ	Longitude	[rad]
θ	Pitch angle	[rad]
θ	True anomaly	[rad]
μ	Standard gravitational parameter	[m^3/s^2]
ρ	Density	[kg/m^3]
σ	Bank angle	[rad]
σ	Standard deviation	
ϕ	Latitude	[rad]
ϕ	Roll angle	[rad]
ψ	Yaw angle	[rad]
Ω	Right ascension of the ascending node	[rad]
ω	Argument of periapsis	[rad]
ω_e	Earth's rotational rate	[rad/s]

Notation

Notation	Definition
$\boldsymbol{C}$	Transformation Matrix
$\hat{f}$	Approximation of function f
$\boldsymbol{r}$	Vector
$\boldsymbol{r}_x$	Projection of vector $\boldsymbol{r}$ into x-axis
$\dot{\boldsymbol{r}}$	First derivative of vector $\boldsymbol{r}$
$\ddot{\boldsymbol{r}}$	Second derivative of vector $\boldsymbol{r}$
$\hat{\boldsymbol{r}}$	Unit Vector of $\boldsymbol{r}$
$\bar{\boldsymbol{r}}$	Normalised vector $\boldsymbol{r}$
x_0	Variable x at time $t = 0$
x_f	Variable x at final time $t = t_f$
$x^{(i)}$	Variable x at an arbitrary time i

1

Introduction

From the earliest civilizations, who looked at the sky and wondered, humankind has been interested in the understanding of the cosmos and our planet. Scientists throughout the centuries have tried to gain a deeper knowledge of how the universe works, and advances made by great minds have pushed humans forward in their quest for knowledge. The launch of the first artificial satellite, Sputnik 1, back in 1957, marked the beginning of the era of space exploration, and a new perspective brought through the observation of the cosmos from outside the Earth. Since then, artificial satellites have been used for a myriad of purposes, with so many different functionalities that nowadays they are ingrained in almost every aspect of human activity. The rapid growth of man-made objects orbiting the Earth has led to a pressing issue in modern society: space debris. This term, by definition, refers to "all non-functional, artificial objects, including fragments and elements thereof, in Earth orbiting or re-entering into Earth's atmosphere" (UN, 2010).

The exponential increase of small pieces of debris is a growing concern for scientists and engineers, as it poses a hazard for other objects in space and current missions. Even small pieces of debris can cause severe damage to the infrastructure of a satellite or a rocket body, hindering the success of current or future missions. In 1978, National Aeronautics Space Administration (NASA) scientist Donald J. Kessler proposed a scenario known as the Kessler syndrome or Kessler effect (Kessler and Cour-Palais, 1978). In this scenario, the density of space debris in the Low Earth Orbit (LEO) region is large enough that collisions between objects lead to a cascade effect, generating multiple fragments of debris in each event that increase the possibility of further collisions. Such a situation could easily and quickly escalate to a point where the number of debris fragments is large enough that the viability of long-term satellite missions could be severely reduced.

The Kessler syndrome poses a catastrophic situation that is to be avoided and raises concerns in the scientific community regarding the sustainability of activities carried out in the Earth's orbit. With a lack of international regulations, it is in the hands of companies and space agencies to take drastic actions to reduce the threat that space debris poses to the future of space operations. There are several ways this can be done, such as the development of active space debris removal missions, the application of methods of safe disposal of satellites at the end of their operational life, or the creation of debris mitigation guidelines. Furthermore, Space Surveillance and Tracking (SST) plays a critical role in space debris mitigation. Through the tracking and monitoring of orbiting objects, agencies and operators can calculate collision risks and plan to prevent manoeuvres to avoid potential collisions and ensure safe in-orbit operations. This project aims to provide a fast and accurate way of calculating the probability of collision between two objects orbiting the Earth, as a means to help mitigate the creation of space debris.

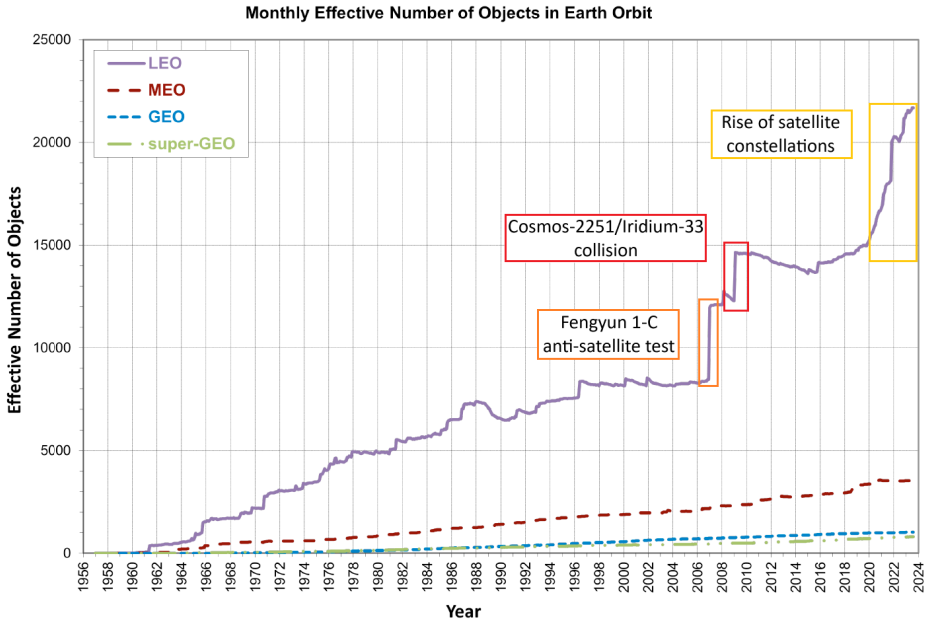


Figure 1.1: Number of objects in Earth orbit catalogued by the US Space Surveillance Network (NASA, 2023c)

1.1. Motivation

The field of space safety has been an increasing concern in the last few decades. The decrease in manufacturing and operating costs of satellites and the commercialisation of the space sector have sparked an interest in large constellations of satellites. SpaceX alone, one of the most well-known companies dedicated to satellite communications, has deployed as many as 5,289 satellites as of January 2024, and is planning to deploy at least 12,000 more (McDowell, 2024). This growth in the interest for mega-constellations (i.e., constellations composed of hundreds or thousands of individual satellites) raises alarms in the scientific community, since they would greatly increase the risk of the Kessler Syndrome.

The space environment nowadays already poses a threat to current missions as it is. The European Space Agency (ESA) estimates that, as of December 2023, there are around 36,500 objects in orbit around the Earth that are larger than 10cm in diameter (ESA, 2023). These objects include both active and inactive satellites, alongside fragments of space debris and others. On top of that, estimations indicate that for diameters between 1 and 10 cm, that number rises to 1 million and even further for objects with diameters comprising from 1 mm to 1 cm, with an estimate of 130 million. Any fragment of space debris is a hazard to current missions in orbit around the Earth. If an object of 10 cm were to crash against a typical satellite, this collision would lead to the fragmentation of said spacecraft. Collisions with 1 cm objects could disable operational satellites, and even 1 mm fragments could be hazardous since they could destroy subsystems onboard active missions (Rossi and Valsecchi, 2006).

The number of catalogued Resident Space Objects (RSOs) orbiting the Earth is observed in Figure 1.1. The objects in space that are catalogued and tracked have doubled in the last 10 years, and only a minority of those are still operational. This rise is due to the increasing interest in satellite constellations. On top of that, in the last 20 years collisions such as the Cosmos-2251/Iridium-33 and fragmentations such as the one resulting from the anti-satellite test on Fengyun-1C have generated a large amount of debris that also needs to be tracked. From the start of the Space Age, approximately 10,000 satellites have been successfully placed into orbit through over 6000 launch missions. However, despite the substantial number of satellites deployed, only about 6000 have not decayed or re-entered the Earth's atmosphere, and thus remain in orbit. Additionally, only a fraction of these maintains operations to this day, with estimations indicating around 3900 active satellites as of September 2023 (ESA, 2023).

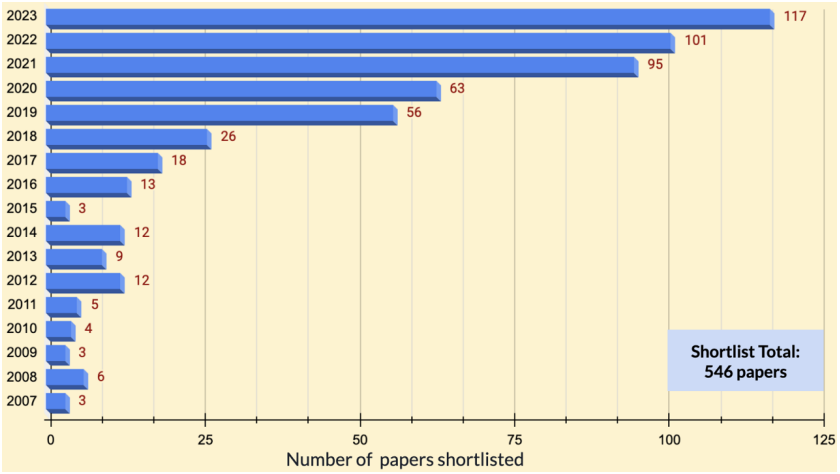


Figure 1.2: Papers shortlisted on Space Surveillance and Tracking (SST) per year (Choumos et al., 2024)

To avoid the partial or complete destruction of active satellites, the tracking of objects in space is crucial, and the estimation of collision risks and application of Collision Avoidance Manoeuvres (CAMs) is, consequently, needed. Currently, it is estimated that each satellite in LEO needs to perform two CAMs per year (NASA, 2023b). This changes depending on the altitude at which the spacecraft is located, since some regions are more densely populated than others. For some of these regimes, manoeuvres can be as frequent as once every three months on average per satellite and might increase due to the deployment of mega-constellations, together with the predicted increase of debris in the future.

Recent work of Choumos et al. (2024) reviews the current state of the art on the topic of orbit prediction and collision avoidance. When reviewing the literature on the field of SST, it was found that there has been an increase in the number of papers published on the topic over the years, with a sharp increase in recent years (Figure 1.2), showcasing the increasing concern of space safety and sustainability. As much as all the subfields of SST are important for a sustainable and safe space environment, there is a particular focus on accurate orbit propagation and prediction. It establishes the basis for collision assessment and manoeuvre planning, both of which are needed for collision avoidance and prevention of space debris generation. Accurate orbit prediction is key to ensuring safe space operations via the mitigation of space debris. More traditional methods require a large level of complexity to obtain a desirable accuracy, which is difficult to achieve. The inherent non-linearities and uncertainties in the space environment make it extremely complicated to capture them via numerical, traditional methods. The use of artificial intelligence (AI) can help overcome these challenges, as predictions can be made without explicit modelling of the simulation environment or space objects.

1.2. Current Conjunction Assessment Practices

Agencies are defining guidelines and best practices for the creation of a safer space environment and operations. Since the active satellites, capable of manoeuvring, are owned by private companies, their maintenance and operations are in the hands of the owners/operators (O/Os). To establish international regulations, agencies like NASA give recommendations on the actions that these O/Os need to take for collision avoidance and manoeuvre planning in their Spacecraft Conjunction Assessment and Collision Avoidance Best Practices Handbook (NASA, 2023b). In said handbook, the current conjunction assessment process as followed by the U.S. Space Defence Command (USSPACECOM, 2024) is detailed. This is a body from the military of the United States that focuses on the defence of national interests via the integration of military spacepower. The 18th Space Defence Squadron (18SDS) and 19th Space Defence Squadron (19SDS) are tactical squadrons under USSPACECOM and are key in the conjunction assessment process performed by this body. It is to be mentioned that the USSPACECOM space catalogue is the one that is used by nearly all conjunction assessment practitioners (18SDS and 19SDS, 2023) and therefore sets the basis of current standard practices.

1 **Conjunction assessment screenings**
The conjunction assessment screening is the first step in the process of conjunction assessment. At USSPACECOM, the 18SDS implements a highly accurate propagator on tracked objects by the Space Surveillance Network (SSN) to predict the orbits of the objects in the Highly Accurate Catalog (HAC). Then, the 19SDS uses this catalogue to screen the orbital trajectories of all tracked objects and identify close approaches. The "close approach" flag is triggered when two objects are predicted to be closer than a set of distances, often much larger than the distances that would pose an actual threat of collision. The time of closest approach (TCA) is also estimated from this data. The results from the screenings are posted on www.space-track.org (Space-Track, 2009), the USSPACECOM space catalogue, in the form of Conjunction Data Messages (CDMs), which are publicly available and can therefore be accessed by the O/Os.

2 **Conjunction risk assessment**
Once the close approaches are identified, each conjunction is analysed for the risk of collision. This is typically done via the calculation of the probability of collision (P_c), but other methods exist in the literature. The value obtained for P_c is the driving factor for the decision-making process of risk mitigation. Any conjunction with $P_c > 10^{-4}$ is considered a high-risk event (18SDS and 19SDS, 2023) and the application of a Collision Avoidance Manoeuvre is recommended. Other risk assessment methods have been developed (Alfano, 2005; Balch et al., 2019; Carpenter and Markey, 2014), but the one described here is common and the threshold defined is deemed safe enough for conjunction assessment (NASA, 2023b). There are, however, several factors affecting the value obtained for P_c, such as the uncertainties in the propagation model, which increase the further in the future the object is propagated. Commonly, the risk of collision reduces as the time of closest approach gets closer because the state of the object is subject to less uncertainty. Many O/Os might choose to not take any immediate actions and wait until P_c is known more accurately. On the other hand, the closer the manoeuvre is postponed to the time of closest approach, the larger the manoeuvre will need to be if the event keeps being considered of high risk.

3 **Conjunction Risk Mitigation**
If one of the objects involved in a high-risk event is manoeuvrable, the O/Os are responsible for planning a mitigation action for collision avoidance. Typical actions include a change in the satellite's trajectory or a change in its attitude, amongst others. It is recommended that the manoeuvre reduces the probability of collision by at least 1.5 orders of magnitude (Hall, 2019), so some post-manoeuvre analysis should be performed to assess the consequences of taking such an action, even before it is taken.

Currently, the practices performed on collision risk assessment, as explained above, rely on physics-based models (Braun et al., 2016). To achieve high accuracy, the computational load required is very large and screenings take a significant amount of time both for the propagation and the analysis of results (NASA, 2023b; Sgobba and Allahdadi, 2013). For the operators to maintain safe space operations, these screenings and collision risk assessments need to be as accurate as possible. This task can prove complicated since these agencies do not have access to relevant data, owned by private companies. Typically this data is not released to the public, hindering the performance of accurate simulations even with the most complex tools. The method developed in this project aims to provide alternative solutions based on the application of artificial intelligence on algorithms that deal with orbit prediction and collision risk assessment.

1.3. Differential Algebra and Gaussian Mixture Model Approach

Another challenge in highly accurate orbit prediction for collision assessment is the incorporation of uncertainties, inherent in the initial state and the environment variables. These need to be taken into account when using traditional, numerical methods to calculate the collision probability between objects. Typically, this is addressed by performing Monte Carlo analyses with an exceedingly large number of iterations on the nominal orbit, which is very time-consuming and computationally expensive. With a complex problem that requires very reliable models, one can easily see how it might sometimes be infeasible to take this approach. Alternative methods have been studied, but there always seems to be a trade-off between accuracy and computational load.

There is, however, a newly developed method that combines Differential Algebra with a Gaussian Mixture Model (DA-GMM) that allows for the accurate propagation of the nominal orbit and uncertainties of the model (Leon Dasi, 2021). It only needs one iteration to achieve a level of accuracy equivalent to 10^9 Monte Carlo runs, showing an incredible improvement with respect to other methods in terms of both accuracy and run time.

This method is very valuable since it can easily be used for validation of the AI-based model that is to be developed in this project with a cut in the run time that is needed. This model is, however, incomplete. It is lacking two elements that can be argued are relevant for the environment simulation: the Solar Radiation Pressure (SRP) and the rotational motion. In her work, Leon Dasi explained that it was not essential to add these elements in the simulation and proof of that is in the 0.04% error that she achieves in her collision probability calculations when compared to a 10^9-sample Monte Carlo analysis. Further details about the work developed by Leon Dasi are detailed in Section 2.2.

The inclusion of the solar radiation pressure and rotational motion, however, would benefit the DA-GMM and make it valid for a larger case of scenarios. It has been shown that solar radiation pressure is an important force acting on satellites (Horstmann and Stoll, 2017; Wang et al., 2018), and therefore there is an interest in analysing how its addition to the model changes the accuracy. This is, however, challenging due to further uncertainties being introduced in the model and the fact that this acceleration is usually modelled as a discrete function, which is not compatible with Differential Algebra. The rotational motion is also considered a key addition to the model, as it affects the interaction of satellites with external forces (Akulenko et al., 2008). The modelling of the rotational motion is, however, more challenging than that for the SRP acceleration. It requires further information about the satellite's orientation and the mass distribution to build the moments of inertia, critical for the rotational equations of motion. Since this information is not publicly accessible, and the addition of the rotational motion induces significant challenges to the DA algorithm (i.e., representing the spacecraft's attitude or integrating with translational orbital dynamics, amongst others), it will not be included due to constraints in the time and workload required for this project. It will nonetheless be added as a recommendation for future work.

1.4. Research Question

Simulating the space environment presents significant challenges due to its complexity. The accurate prediction of the movement of RSOs implies the use of complex numerical models that have a large computational load. In situations where there is an urgency to model the orbit of certain objects to calculate collision risk probabilities, a large computational load is undesired. On the other hand, high accuracy is key for this calculation and so with numerical methods, a trade-off between the complexity of the model (and therefore computational load and time) and the accuracy of the prediction needs to be made. The main goal of this project lies in a faster and more accurate computation of the probability of collision between objects orbiting the Earth to aid in space debris mitigation.

There is a clear potential for the use of AI for conjunction assessment. The hypothesis made here is that machine learning (ML) could be implemented on simpler, less accurate simulation environments that can achieve a similar level of accuracy as more complex numerical methods. It is theorised that the implementation of machine learning will allow the algorithm to learn the underlying patterns of the perturbation forces without the need to explicitly model them in the simulation. This, in turn, reduces the need to develop computationally expensive methods, such as Monte Carlo analyses, for the predictions and uncertainty distribution.

To validate the algorithm implemented, a highly accurate solution for the orbit propagation is needed. The best method to do this is the aforementioned DA-GMM since it provides highly accurate results with a fraction of the computational time needed for similarly accurate methods. Being extensively verified, it is chosen as the best tool to test the results of the approach taken in this project. For a more complete simulation, there is an element that still needs to be included in the DA-GMM algorithm, as explained in Section 1.3, which is the Solar Radiation Pressure acceleration model.

The research gap that is found in the literature is for the development of faster and more accurate models for collision risk assessment, which establishes the main research question that will guide this

work. Possible approaches for this are investigated and discussed. Firstly, the DA-GMM has already achieved to significantly reduce the computational load while keeping a good accuracy in the calculation of P_c. However, there is, as mentioned, a research gap in this method resulting from the incomplete modelling of the forces of the environment. More concretely, the absence of implementation of Solar Radiation Pressure in the DA-GMM framework and research on how that affects the accuracy of the model is the cause of this research gap that will be tackled in this project. Secondly, several studies have already been done on the use of AI-based algorithms for conjunction assessment with positive results. There is, however, a research gap in more thorough studies on how these algorithms compare to traditional methods through more extensive validation tests and the use of real-life data. Following these premises, as well as the main focus of the project on improved collision risk assessment, a series of research questions and sub-questions have been posed to direct the project:

Q1 How can the computation of the probability of collision be enhanced for faster and more accurate collision risk assessments?
As expressed earlier, the main focus of this project is on the increase of accuracy and reduction in the computational load required for conjunction risk assessments with respect to more traditional methods.

Q1.1 How can the DA-GMM be extended for more accurate predictions?
There is a desire in this project to obtain the most accurate answers with a traditional method for comparison purposes with respect to the results of the ML algorithm. The way to do so is by extending the DA-GMM with some models that were not initially included but could be relevant.

Q1.1.1 How does adding the effect of the Solar Radiation Pressure affect the results obtained from the DA-GMM as developed by Leon Dasi?
To obtain more accurate answers, the Solar Radiation Pressure should be added to the model due to its relevance in Earth-orbiting objects' dynamics. It is important to verify that the original code still performs accurately even after some modifications and assess the new level of accuracy that is obtained through the addition of this model.

Q1.2 To what extent can simplified models be used to accurately predict the orbit of Resident Space Objects (RSOs) and compute collision probability with the use of artificial intelligence?
This question is posed concerning one of the core sections of this study, namely the implementation of machine learning models and how they can be used in combination with simplified propagation models to achieve accurate results.

Q1.2.1 How does the accuracy of the ML-based algorithm compare to that of current conjunction assessment algorithms?
Firstly, it is essential to check whether the accuracy that is obtained with the simplified, more inaccurate, models is comparable to that of traditional methods.

Q1.2.2 How does the computational load required for accurate predictions using the ML-based model translate to automatic real-time systems?
Machine learning algorithms can be quite computationally heavy. The work developed here will be aided by supercomputers for analysis and therefore computational load is not an issue, but it is relevant to understand whether it is feasible to use these models for the on-board computers to automatically compute collision probabilities.

Q1.2.3 Which environment and perturbation models need to be used in the simulations to meet the accuracy requirements?
Understanding the models that are needed to simulate the environment, both in the accurate method and in the simplified one, is essential for the development of the project.

Q1.2.4 How far in the future can this method predict the orbit of an RSO accurately?
Provided that the use of AI-based models can provide accurate answers, it would be interesting to study how much further in the future the model can propagate the orbit of an object while maintaining a reasonable level of accuracy.

Q1.2.5 To what extent can the algorithm predict the collision and time of closest approach of the Cosmos-2251/Iridium-33 crash in 2009?
A real-life scenario should be used to assess the performance and application of the method developed in this thesis. The Cosmos-2251/Iridium-33 collision is the most famous, and one that has been extensively studied in the field of collision detection.

The questions posed here will be used as guidelines for the project. It is expected that the work performed in this thesis shall find the answers to these research questions, which will be revisited at the end.

1.5. Outline of the Report

The thesis work to be developed for this research is structured into several parts and chapters, each providing a different focus on the research of AI-based algorithms for collision detection. The first part of the report aims to provide the background information that is required as a baseline to understand the rest of the project. Chapter 2 reports the main findings from the literature study phase of the thesis, with an explanation of the existing research done in the same field. On top of that, it provides information on the missions that are used as part of this thesis to validate the model, as well as the system and mission requirements that are placed for the model to be developed. The research could be separated into two main fields: astrodynamics and machine learning. The background information for the former is introduced in Chapter 3, whereas the core concepts and algorithms of the latter are described in Chapter 4. These two chapters include only the information that is thought needed to understand the models built to provide answers to the research questions.

The next part of the thesis focuses on software development. The first area of study within this part is that of external software integration, discussed in Chapter 5. This chapter provides a view of the software architecture of the models to be developed to discern the external software required for the models. Moreover, their setup and integration are verified once the external software and libraries are identified. Chapter 6 presents the work done on the extension of the Differential Algebra-Gaussian Mixture Model, including the mathematical models used for the formulation of the SRP acceleration and its uncertainty modelling, as well as the verification and validation tests done to analyse the level of improvement of the additions. The core of this thesis work is the development of a Physics-Informed Neural Network (PINN) for collision detection. The software development for this task is reviewed in Chapter 7.

The last part of the report introduces the results obtained from the PINN in Chapter 8, with a thorough analysis focused on the identification of the strengths and limitations of the approach selected for the research. With these identified, Chapter 9 reiterates the conclusions drawn from the research done and provides recommendations for future researchers who might want to work in this same field.

2

Mission Heritage

This chapter introduces the mission heritage for the topic of collision probability calculation and machine learning for orbital dynamics. Firstly, an overview on collision detection and avoidance, with current practices and methods, is given in Section 2.1. Then, the DA-GMM method as developed by Leon Dasi (2021) is reviewed and discussed more in detail in Section 2.2. Since the first part of the work developed in the project is going to be an extension of this method, a detailed explanation of it is thought to be appropriate. Concerning this, the methods for modelling the uncertainties in the Solar Radiation Pressure acceleration are reviewed in Section 2.3. The main block of this project, however, is the application of machine learning algorithms, so the main trends used in the literature are summarised in Section 2.4. The possible data sources that can be used to train the model are discussed in Section 2.5 and a choice is made. For the validation tests, a set of reference missions and cases will be needed, which are explained in Section 2.6. Finally, with all this information, a set of requirements is developed in Section 2.7 and an overview of the methodology is given in Section 2.8.

2.1. Collision Detection and Avoidance

The safety of space operations relies heavily on collision detection and avoidance, processes which aim to prevent collisions between objects in space and thus prevent the creation of debris. The rapidly increasing crowdedness of certain regions in space due to the deployment of mega-constellations creates an increasing need to predict close encounters and perform corresponding manoeuvres.

The initial step to conjunction assessment is the tracking and cataloguing of objects in space. RSO detection (or tracking) refers to the process of using ground-based or space-based sensors to identify diverse objects in space (Tsaprailis et al., 2024). This is the process by which the positions and velocities of different space objects are obtained, which are then uploaded to catalogues maintained by different Space Surveillance Networks such as the USSSN.

With the tracked data obtained, the conjunction assessment screening process begins. This involves the use of highly accurate models to predict the state of different orbital objects and identify potential conjunctions or close approaches. To obtain the collision probability between two objects, their states and covariances need to be propagated using complex numerical models. This is done for the entire catalogue, where the conjunction of all objects with respect to every other object tracked is considered and assessed. This takes an infeasible computational effort, thus filtering techniques are typically used to reduce the possible conjunctions assessed. Leloux (2012) presents and reviews different approaches, from analytical and efficient pre-screenings (Hoots et al., 1984; Khutorovsky et al., 1993) to more accurate analyses (Alarcon-Rodriguez et al., 2002; Klinkrad, 1993). The method selected will depend on the aims and requirements of the mission. The focus of this thesis is on the orbit propagation and collision probability calculation, hence this step is assumed to be completed. Proper screenings between two satellites can be done upon filtering to reduce the number of potential conjunctions.

2.1.1. Orbit and Uncertainty Propagation

Before the calculation of the collision probability can be done, the state and uncertainties of the satellite need to be propagated. This is the most fundamental part of any collision detection algorithm since the adequate estimation of the collision risk depends heavily on the accuracy of the propagations. After a thorough literature study, several approaches have been identified and will be reviewed in the upcoming paragraphs.

Classic Approaches

Classical approaches to orbit prediction refer to those techniques that propagate the state of objects in space following the laws of orbital motion. They can be divided into numerical, analytical, and semi-analytical approaches.

- *Numerical approaches* use integrators to propagate the orbit of an object through a complex, perturbed environmental model. Different numerical approaches to increase the accuracy of the integrators and the long-term stability of their solutions have been extensively researched in the past (Hofsteenge, 2013; Vittaldev, 2010).
- *Analytical approaches* use a simplification of the perturbations affecting the object to use analytical models to solve the equations of motion. These cannot capture the complexity of the actual environment but require a significantly lower computational effort.
- *Semi-analytical approaches* provide a larger accuracy than purely analytical methods while reducing the computational load of numerical models. They involve the simplification of the perturbations affecting a spacecraft and the resolution of the consequent equations of motion using mathematical approximations. One example of this is the Simplified General Perturbations (SGP4), explained more in detail in Appendix D.

These are all methods used for orbit propagation, in which an initial state (obtained from the diverse tracking methods) is used to calculate the state of an object for a period of time in the future. This, however, does not provide any information on the uncertainties of the model.

The most common approach used in current collision assessment practices is the Monte Carlo analysis (Braun et al., 2016). In this, a set of initial states is created following a certain distribution, each of which is then propagated using the desired approach. The set of future states is used to assess the uncertainties of the model at the desired time steps. This method is extremely time-consuming and so alternative approaches for orbit and uncertainty propagation have been investigated.

Non-AI Approaches

Methods that diverge from the classical approaches for uncertainty propagation and rely on innovative mathematical formulations also need to be reviewed. They are typically built on top of classical orbit propagation methods but are aimed to reduce the computational load required for Monte Carlo simulations.

The first approach investigated is the *Interval Method*, developed by Moore (1968) and reviewed by Römgens (2011) for application in the field of collision detection. In this method, the relative motion of two objects is computed at a series of discrete time intervals using a numerical propagator. Then, *bounding volumes* are used to simulate all the possible positions a certain satellite might take within an interval. When the bounding volumes of two objects intersect, a more thorough analysis is performed to assess the collision probability using the relative motion of the objects and their uncertainties through covariance propagation.

The following approach is the Differential Algebra-Gaussian Mixture Model, developed by Leon Dasi (2021). This method combines the DA framework for uncertainty propagation and the GMM to approximate the uncertainty PDFs. The uncertainty propagation is therefore embedded in the state propagation, hence the model requires a single run to propagate both. This will be discussed in more detail in Section 2.2 since it will be used and extended in this thesis.

Both of these methods have been used for collision detection and have demonstrated that they can effectively reduce the computational load of the classic approaches while obtaining accurate predictions.

AI-based Approaches

The use of AI for collision detection and avoidance has largely increased in the last decade (Choumos et al., 2024), demonstrating a potential for faster and more accurate collision assessments. Machine Learning algorithms allow for the propagation of the state of an object in space without modelling the complex dynamics of its environment, effectively reducing the computational load of classical approaches.

Different algorithms have been used for orbit propagation, including *Support Vector Machines* (Peng and Bai, 2018b), *Artificial Neural Networks* (Peng and Bai, 2018a), *Convolutional Neural Networks* (Dave et al., 2020), and *Recurrent Neural Networks* (Curzi et al., 2022; Jadala et al., 2022; Shin et al., 2022). All of these algorithms will be reviewed in detail in Section 2.4, with brief descriptions on the characteristics of each and how they can be useful for orbit prediction. None of the methods reviewed in Choumos et al. (2024) provides an overview of how the non-linear uncertainties of the satellite can be propagated, thus estimation techniques would be required to obtain them using these methods.

Other ML-based models can be used for uncertainty estimation. Peng and Bai (2019) use *Gaussian Processes* (GP) for uncertainty prediction. A GP can be defined as an ML algorithm which can be used to model the uncertainties around a core value assuming a Gaussian distribution. These, however, present a large performance sensibility to the kernel choice and they perform poorly for large datasets, which are expected and required for machine learning in the field of orbit prediction. Other Bayesian inference processes have been used for non-linear dynamics (Lan et al., 2022), demonstrating their use for uncertainty quantification.

2.1.2. Collision Probability Calculation

With the orbital state of the satellite, as well as its uncertainties, the next step of the conjunction risk assessment can be performed: the computation of the probability of collision between different objects in space. There are many methods used in the literature for this, including some suited for short-term encounters and others for long-term.

The calculation of P_c for short-term encounters involves a series of assumptions (Foster and Estes, 1992):

1. The orbit of all objects involved can be represented as a linear trajectory.
2. The uncertainty in the velocity vector of either satellite can be neglected.
3. The uncertainty in the position vector of either satellite can be assumed constant throughout the encounter.
4. The uncertainties in the position and velocity vectors can be assumed to follow a Gaussian distribution and are uncorrelated.
5. Both objects can be modelled as hard-body spheres.

For these types of situations, the encounter geometry can be assumed planar and the P_c calculation can be reduced to a 2-dimensional problem in the conjunction plane. Different methods include the original collision probability formulation by Foster and Estes (1992). Patera (2001) and Alfano (2005) proposed different formulations based on the reduction of the P_c calculation to a 1-dimensional problem, reducing the computational load required for it. Finally, Chan (1997) proposed an analytical 2-dimensional method for this calculation, faster than all the other methods discussed.

These assumptions no longer hold for long-term encounters, where the conjunction cannot be assumed to be 2-dimensional anymore. The encounter geometry is changed to a set of volumes, which simulate the uncertainties surrounding the objects. Chan (2004) extended his method for long-term encounters by adding an integration along the volume of the encounter geometry. Patera (2003) also extended his algorithm, this time using contour integration. Alfano (2006) (method of adjoining cylinders), Alfano (2007) (method of bundled parallelepipeds) and McKinley (2006) (method of voxels) all use models to discretise the geometrical encounter and reduce the computational time required to calculate P_c. Finally, Coppola (2012) uses an instantaneous P_c calculation based on the collision probability rate at each instant in time of the encounter.

Overall, the formulations for long-term encounters are preferred since they are more robust to any type of geometry. All of them present different assumptions on the encounter geometry and shape of the bodies, out of which the time integration method by Coppola (2012) can be used for a wider range of geometries and scenarios. This agrees with the assessment made by Leon Dasi (2021), and so its formulation will be presented in Subsection 2.2.3.

After the collision probability of an encounter has been computed, the event can be classified. If an event is considered of high risk, the O/Os of the satellites are informed and advised to make Collision Avoidance Manoeuvres (CAMs) to mitigate the generation of space debris.

2.2. Differential Algebra - Gaussian Mixture Model (DA-GMM)

An introduction to this method was already given in Section 1.3. This section aims to expand on it by giving an overview of the model in Subsection 2.2.1, how the uncertainties are modelled in Subsection 2.2.2 and the collision probability method chosen in Subsection 2.2.3.

2.2.1. Overview of the model

The DA-GMM is a hybrid method which combines Differential Algebra with Gaussian Mixture Model for uncertainty propagation. Each of these on their own has shown to have suitability to orbital dynamics, but using a hybrid of the two allows to counter their drawbacks and create a more accurate and faster model to predict P_c.

Many uncertainties affect the accuracy of the calculation of the collision probability. It is therefore desired to model the satellite via a Probability Density Function (PDF) and propagate the uncertainties. This can be extremely challenging since there is no analytical approach to solve the time evolution of the PDF of the state vector of the spacecraft when perturbations are also considered. Since the dynamics of the environment becomes highly non-linear, methods to propagate the uncertainties under such conditions are derived.

One of the main issues of these methods is that they tend to be limited to the propagation of Gaussian PDFs. When propagating through non-linear dynamics, Gaussian distributions become non-Gaussian and therefore these methods become unsuitable for this task. To tackle this, the Gaussian Mixture Model is introduced. This method approximates the non-Gaussian distribution via a weighted sum of several individual Gaussian PDFs. GMM, when used in combination with traditional integration methods, still requires a large computational load to achieve a similar accuracy to Monte Carlo analyses. However, it can be combined with other uncertainty propagation techniques that can reduce the computational load by approximating the dynamics of the environment.

The method chosen to address this problem is Differential Algebra (DA). This method relies on the approximation of the non-linear dynamics via a series of Taylor-series expansions, where the first-order terms of the expansion correspond to the linearised dynamics, while high-order terms capture the effects of non-linearities. These can capture the evolution of the mean and standard deviation values of the variables of the system, and therefore propagate the uncertainties in a simplified environment.

The hybrid method presented here allows for the approximation of the initial state via the GMM and propagates the state to the final time via DA. A DA algorithm is first used to propagate the state of the satellite via simplified dynamics. The mean and covariance of each Gaussian Mixture Element (GME), therefore, do not need to be propagated, but instead, they can be obtained from the Taylor-series expansion of the final state, and hence the PDF of the final state can be obtained. This method allows for a reduction in the computational load while obtaining highly accurate results.

2.2.2. Uncertainty Propagation

There are two main sources of uncertainty in the field of orbit prediction: the initial state and the environment models. These need to be treated in the algorithm and as such need to be modelled by means of different probability distributions.

The uncertainty in the initial state is considered to follow a Gaussian distribution. In this type

Table 2.1: Standard deviation of initial state in RSW components (Leon Dasi, 2021)

Mission	$\sigma_{R,R}$, $\sigma_{R,W}$ [m]	$\sigma_{R,S}$ [m]	$\sigma_{V,R}$ [m/s]	$\sigma_{V,S}$, $\sigma_{V,W}$ [m/s]
General LEO satellite	0.2	0.8	0.004	0.001
TLE	100	400	1	0.25
Space debris	250	1000	1	0.25

Table 2.2: Summary of the probability distribution chosen for the modelled uncertain variables (Leon Dasi, 2021)

Variable		Probability Distribution	Range of Values
Initial State		Gaussian	Table 2.1
Observed to model density ratio		Log-normal	[-0.08, 0.25]
Ballistic Coefficient	Generic spacecraft	Uniform	[K-0.015, K+0.015]
	Space debris	Log-uniform	[10^{-4}, 5]

of distribution, the mean corresponds to the expected value for a certain variable (e.g., the expected position of the satellite), while the standard deviation or variance are measure of the uncertainty of this value. When it comes to the environment models chosen, there are uncertainties introduced in the problem due to imperfect knowledge of variables or due to simplifications. To reduce the computational load, the number of variables that are included as uncertainties needs to be limited. Therefore, only those that have a larger effect on the problem are considered by Leon Dasi: the atmospheric density and the ballistic coefficient.

The uncertainties that need to be propagated are modelled via PDFs. The mathematical formulation of these is provided in Appendix A. Typically, uncertainties in the field of orbit determination are assumed to follow a Gaussian PDF, but for the sake of accuracy, other models are considered for the environment variables. The atmospheric density uncertainty is modelled to have a log-normal distribution (Picone et al., 2002), while the ballistic coefficient depends on the characteristics of the spacecraft and is modelled differently for different cases. For a generic spacecraft, a uniform distribution is considered, while the uncertainty for an unknown piece of debris or other spacecraft with unknown mass or size is assumed to follow a log-uniform distribution (Bhusal and Subbarao, 2020). The uncertainties for the initial state are defined in the Radial, Along-track, Cross-track (RSW) frame, introduced in Section 3.1, and are summarised in Table 2.1 following the research by Rongzhi and Kaizhong (2020) and Flohrer et al. (2008). The uncertainty models chosen for the environment variables and the corresponding range of values are provided in Table 2.2.

2.2.3. Collision Probability Calculation

The approach chosen by Leon Dasi in her work to obtain the collision probability follows that developed by Coppola (2012). This is a three-dimensional calculation of P_c and includes the uncertainty of the state of the satellite (i.e., both for the position and velocity) and can be used for any geometry. This method is based on the idea that the total collision probability P_c can be obtained via the integration over time of the collision probability rate P_c over the length of the encounter. This is shown in Equation (2.1), where $P_c(t)$ is the probability that a collision occurs by time t, and is based on the assumption that the probability of collision is 0 at $t = 0$.

$$P_c(t) = \int_0^t p_c(\tau) d\tau \tag{2.1}$$

The collision probability rate is given by Equation (2.2), under the assumption of hard-body spheres with radius R. The derivation shown here follows the work by DeMars et al. (2014).

$$p_c(t) = R^2 \sum_{i=1}^{N_a} \sum_{j=1}^{N_b} w_{a,t}^{(i)} w_{b,t}^{(j)} \int_0^{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} p_g \left(R\hat{\boldsymbol{n}}_k; \boldsymbol{\mu}_{r,t}^{(ij)}, \Sigma_{r,t}^{(ij)} \right) \nu(\hat{\boldsymbol{n}}) \cos\theta d\theta d\phi \tag{2.2}$$

N_a and N_b represent the number of GMEs of Objects A and B respectively. The variable $w_{a,t}^{(i)}$ is the weight of the i^{th} GME of Satellite A at time t, whereas $w_{b,t}^{(j)}$ is the weight of the j^{th} GME of Satellite B at the same time. The angles θ and ϕ are the elevation and azimuth angles, respectively. The Gaussian PDF is described by:

$$p_g\left(\boldsymbol{x};\boldsymbol{m},\boldsymbol{P}\right)=|2\pi\boldsymbol{P}|^{-1/2}\exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m})^T\boldsymbol{P}^{-1}(\boldsymbol{x}-\boldsymbol{m})\right\}$$

and can be used to describe the combined PDF of two independent objects via $p_g\left(R\hat{\boldsymbol{n}}_k;\boldsymbol{\mu}_{r,t}^{(ij)},\boldsymbol{\Sigma}_{r,t}^{(ij)}\right)$, where:

$$\boldsymbol{\mu}_t^{(ij)}=\boldsymbol{m}_{a,t}^{(i)}-\boldsymbol{m}_{b,t}^{(j)},\quad \boldsymbol{\Sigma}_t^{(ij)}=\boldsymbol{P}_{a,t}^{(i)}+\boldsymbol{P}_{b,t}^{(j)}$$

which can be decomposed into position and velocity components:

$$\boldsymbol{\mu}_t^{(ij)}=\begin{bmatrix}\boldsymbol{\mu}_{r,t}^{(ij)}\\ \boldsymbol{\mu}_{v,t}^{(ij)}\end{bmatrix}\quad \boldsymbol{\Sigma}_t^{(ij)}=\begin{bmatrix}\boldsymbol{\Sigma}_{r,t}^{(ij)} & \boldsymbol{\Sigma}_{rv,t}^{(ij)}\\ \boldsymbol{\Sigma}_{vr,t}^{(ij)} & \boldsymbol{\Sigma}_{v,t}^{(ij)}\end{bmatrix}$$

$\boldsymbol{m}_{a,t}^{(i)}$, $\boldsymbol{m}_{b,t}^{(i)}$, $\boldsymbol{P}_{a,t}^{(i)}$ and $\boldsymbol{P}_{b,t}^{(i)}$ are the mean and covariance matrices of the i^{th} GME of Objects A and B at time t. The last element of Equation (2.2) that needs to be explained is $\nu(\hat{\boldsymbol{n}})$, which has an expression as follows:

$$\nu(\hat{\boldsymbol{n}})=\frac{\sigma(\hat{\boldsymbol{n}})}{\sqrt{2\pi}}\exp\left\{-\frac{\nu_0^2((\hat{\boldsymbol{n}}))}{2\sigma^2(\hat{\boldsymbol{n}})}\right\}-\frac{\nu_0(\hat{\boldsymbol{n}})}{2}\left[1-erf\left\{\frac{\nu_0(\hat{\boldsymbol{n}})}{\sigma(\hat{\boldsymbol{n}})\sqrt{2}}\right\}\right] \tag{2.3}$$

where:

$$\begin{aligned}\nu_0(\hat{\boldsymbol{n}})&=\hat{\boldsymbol{n}}^T\left[\boldsymbol{\mu}_{v,t}^{(ij)}+\boldsymbol{\Sigma}_{vr,t}^{(ij)}\left(\boldsymbol{\Sigma}_{r,t}^{(ij)}\right)^{-1}\left(R\hat{\boldsymbol{n}}-\boldsymbol{\mu}_{r,t}^{(ij)}\right)\right]\\ \sigma^2(\hat{\boldsymbol{n}})&=\hat{\boldsymbol{n}}^T\left(\boldsymbol{\Sigma}_{v,t}^{(ij)}-\boldsymbol{\Sigma}_{vr,t}^{(ij)}\left(\boldsymbol{\Sigma}_{r,t}^{(ij)}\right)^{-1}\boldsymbol{\Sigma}_{rv,t}^{(ij)}\right)\hat{\boldsymbol{n}}\end{aligned} \tag{2.4}$$

With all the expressions described here, the next step is to perform the integration of Equation (2.2). This cannot be done analytically, and a numerical approximation is needed to solve this. The method chosen is called Lebedev's quadrature (Lebedev, 1976), and the resulting expression to evaluate the collision probability rate is:

$$p_c(t)=R^2\sum_{i=1}^{N_a}\sum_{j=1}^{N_b}w_{a,t}^{(i)}w_{b,t}^{(j)}\sum_{k=1}^{N_k}w_k p_g\left(R\hat{\boldsymbol{n}}_k;\boldsymbol{\mu}_{r,t}^{(ij)},\boldsymbol{\Sigma}_{r,t}^{(ij)}\right)\nu(\hat{\boldsymbol{n}}_k) \tag{2.5}$$

A set of N_k quadrature points on the surface of the sphere is selected to perform the integral. Each of these points has a weight w_k and the integral is approximated via a weighted summation. The unit vectors of the unit sphere are represented by $\hat{\boldsymbol{n}}_k$.

2.3. Uncertainty Modelling for Solar Radiation Pressure

There are several sources of uncertainty resulting from the addition of the solar radiation pressure perturbation. The expression for this acceleration is given by:

$$\boldsymbol{a}_{SRP}=-\frac{W}{c}\left(\frac{C_R S_{ref}}{m}\right)\frac{\boldsymbol{r}_{\odot_s}}{||\boldsymbol{r}_{\odot_s}||} \tag{2.6}$$

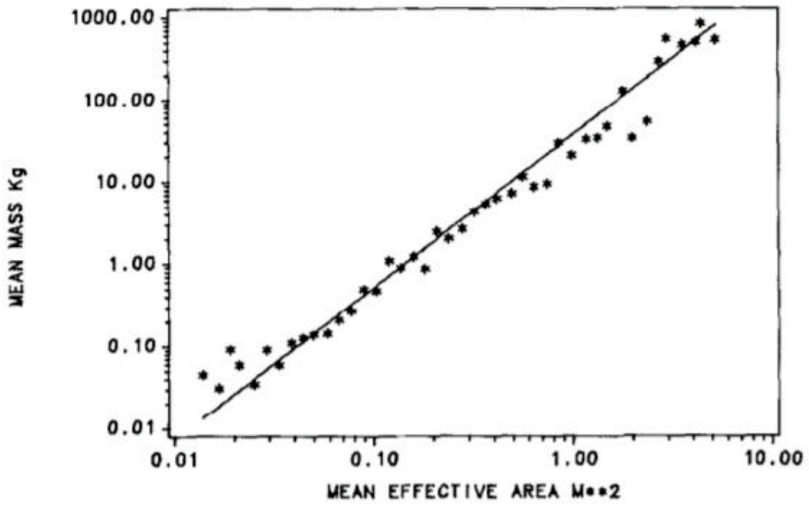


Figure 2.1: Relationship between mean debris mass and mean effective area (Badhwar and Anz-Meador, 1989)

where W is the solar flux, c represents the speed of light, C_R is defined as the radiation pressure coefficient of body A, S_{ref} is the effective or reference area of the body, m is its mass, and $\boldsymbol{r}_{\odot_s}$ is the distance from the body to the Sun.

This will all be discussed more in detail in Section 3.4. The focus of the current section is on understanding the main uncertainties of the SRP and corresponding modelling of them. The parameters and variables that could lead to numerical errors are the solar flux, which has fluctuations in time, and satellite-specific properties such as the radiation pressure coefficient, the reference area, and the mass of the vehicle.

The solar flux has fluctuations in its value due to the translational movement of the Earth around the Sun. Hence, it varies throughout the year, with a value of 1317 W/m^2 at the apohelion and 1408 W/m^2 at the perihelion. There is a desire not to model too many uncertainties for simplicity purposes, and thus this variation due to the Sun-Earth distance can be calculated in the algorithm for the DA-GMM. More uncertainties come from this variable, such as the solar activity cycles (which occur in a 11-year period). Since these variations in the solar flux are only of a 0.1% (Schmutz, 2021; Wilson and Hudson, 1991), these are thought insignificant enough that they are gathered to be negligible.

When it comes to satellite-specific variables, the values are not usually publicly available. As such, some of these need to be modelled as uncertainties: the radiation pressure coefficient C_R, the effective area, S_{ref}, and the mass of the satellite, m. To simplify the modelling, these variables are grouped into one, which will be called solar radiation coefficient: $CSRP = \frac{C_R S_{ref}}{m}$. The case proposed here is similar to that for the ballistic coefficient described previously. The radiation pressure coefficient might change due to surface or material degradation, or with a change in attitude. The former will not be considered under the assumption that the orbit prediction and P_c calculation will happen in a short period. Assuming that the change in attitude of the RSO is constant, the uncertainty of C_R can be modelled as uniform. If the satellite has a known reference area and mass, the PDF of the solar radiation coefficient can be assumed uniform. Typically, the radiation pressure coefficient has values in the range 1.2-1.5 (Doornbos, 2011; Wakker, 2015). Following the logic in the work of Leon Dasi to establish the range of values for the ballistic coefficient of generic spacecraft, the range obtained for C_R is $[CSRP$ - 0.01, $CSRP$ + 0.01$]$. It is defined with respect to the nominal $CRSP$ value since the characteristics of the area and mass are assumed to be known accurately.

However, for pieces of debris and spacecraft with unknown geometrical properties, the PDF distribution can no longer be assumed uniform. In such a case, the size and weight of the object are not known and therefore must be assumed. Badhwar and Anz-Meador (1989) follow a study on the area and mass of different pieces of space debris and the resulting relationship is shown in Figure 2.1. The range of S_{ref}/m is therefore $[10^{-3}, 1]$ m^2/kg following a logarithmic relationship, and, taking into ac-

Table 2.3: Summary of the probability distribution chosen for the solar radiation coefficient

Mission	Probability Distribution	Range of Values
Generic spacecraft	Uniform	$[CSRP$ - 0.01, $CSRP$ + 0.01]
Space debris	Log-uniform	$[10^{-3}, 2]$

count a constant C_R in the range mentioned above, the resulting PDF can be modelled as log-uniform in the range $[10^{-3}, 2]$. Some buffer was taken to account for further uncertainties and errors in the assumptions made. The PDF model chosen and range of values for the $CSRP$ are summarised in Table 2.3.

2.4. Machine Learning for Orbital Dynamics

The use of machine learning algorithms in the literature for topics related to Space Situational Awareness (SSA) and Space Surveillance and Tracking (SST) is becoming more and more popular. As explained in Chapter 1, the focus of this project is going to be on accurate collision risk assessment, and the approach proposed is through the application of artificial intelligence.

A thorough literature study has been conducted on the current state of the art of the use of AI for orbit prediction. The main trends and ideas found are briefly discussed in this section, alongside other algorithms that might be relevant to the context of orbital dynamics. The advantages and disadvantages of each of these, alongside their suitability to the problem considered and use in the literature, will be reviewed as well.

An aspect to take into consideration when choosing an ML algorithm for a certain problem is the depth of the model. In the literature, both shallow and deep machine learning methods have been used. Deep learning has the potential to capture better more complex patterns and scenarios through the use of multiple layers and large amounts of data. While shallow machine learning methods do not have such a large computational load, they lack the structure to capture more accurately the space environment. Therefore, neural networks are preferred for such a problem, which is also the trend that has been observed in the literature. Nonetheless, for the sake of comparison, shallow machine learning methods reviewed in the literature will also be summarised.

Support Vector Machine

The Support Vector Machine (SVM) is an algorithm that can be used for classification problems. These algorithms aim to find the optimal boundaries that separate different classes while maximising the margin between them. The model classifies the different data points, and penalties are applied if the prediction is incorrect. This can be extended to non-linear problems through the incorporation of kernel methods, and to regression problems through the modification of the problem to predict continuous, quantitative values instead of classes.

This method showcases robustness and can handle non-linear problems (Peng and Bai, 2018b). It is suitable for pattern recognition and prediction; however, it is also known to be sensitive to the hyperparameters of the model and for its computational complexity. The latter can become a problem for complex simulations with a large number of samples or features, as it scales in the order of $\mathrm{O}(\mathrm{n}^2)$ or $\mathrm{O}(\mathrm{n}^3)$ for non-linear problems (Bishop, 2006).

The applications reviewed in the literature involved training the model on the prediction errors, which allows for the reduction of the dimensions of the problem. The model is fed with a 'true' solution coming from a highly accurate simulation and an estimation from a simpler model, as developed by Peng and Bai (2018b). Then, the algorithm predicts the error that is induced by using the simpler model and applies that to the estimation to correct for this. An overview of the methodology is found in Figure 2.2. The algorithm is divided into two main blocks: firstly one finds the conventional orbit prediction (depicted in grey in the figure), which provides the data that will be fed into the SVM, and secondly, the machine learning enhancement block (the yellow block in the figure), which produces the orbit prediction from the data collected from the 'true' and assumed dynamics models.

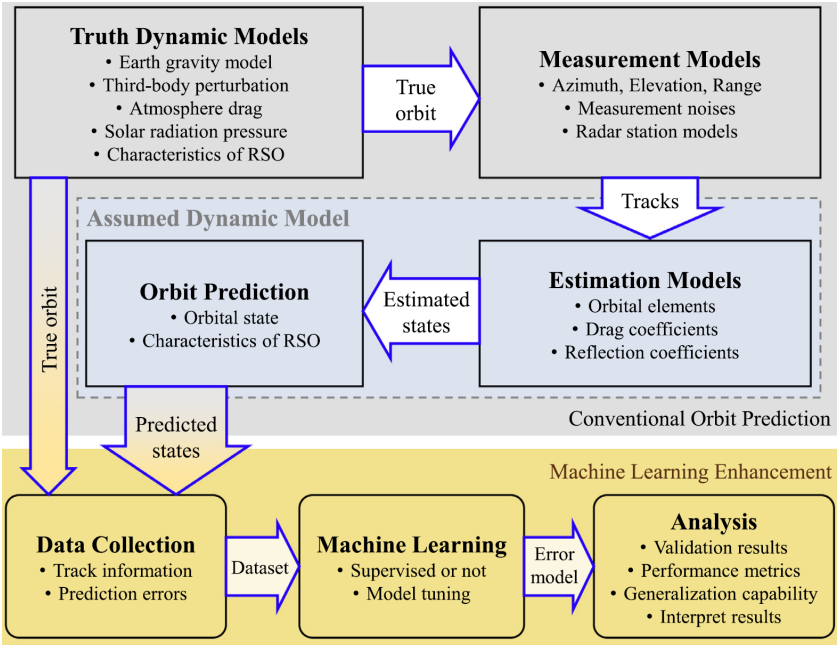


Figure 2.2: Framework of the method developed by Peng and Bai (2018b)

The conclusion from the work developed by Peng and Bai (2018b) is that the SVM developed can improve the prediction of the orbital state of different RSOs with respect to the estimation models that form part of the conventional orbit prediction methods. The training of the model is performed using a single RSO, and so the prediction of the orbit of similar RSOs did more poorly. Overall, the ML model improved the orbit prediction in most of the points of the dataset, but more testing needs to be done to draw further conclusions. The training of the algorithm on a single, simulated RSO resulted in worse predictions for other objects, and thus it is desired to use more satellites for the training phase of the model developed in this project.

Artificial Neural Network

Deep learning is rooted in the use of neural networks, from which Artificial Neural Networks (ANNs) are the most basic and generic form. An ANN is a supervised learning algorithm, meaning that it learns from labelled data to understand the input-output relationship. Its structure is inspired by that of biological networks between neurons in the brain, which allows it to learn from the past. Neural networks, in essence, are composed of layers of artificial neurons, including an input layer (which receives the learning variables, or raw data) and an output layer (which generates the prediction of the target variable), which are interconnected via one or several hidden layers. Each connection between neurons has a different weight, and the algorithm learns to adjust these weights via the relationship between the inputs and outputs of the training data and predict the outcomes as accurately as possible. More information on the architecture of neural networks is provided in Section 4.2.

ANNs can handle more complex problems than those by shallow machine learning methods and are good at working with non-linearities (Zhang et al., 2023). They are highly flexible and are good at generalisation (provided that they are 'well trained'). On the other hand, they are prone to overfitting the data, so prevention measures such as the application of early stopping criteria are typically used. On top of that, their complex structure leads to a larger database (insufficient or biased training data might lead to poor prediction ability) and a large computational load (depending on the number of layers and neurons). Hyperparameter tuning is generally needed to achieve better results.

A typical approach in the literature is to train one ANN for each component of the target variable (Peng and Bai, 2018a; Sanchez et al., 2019). For example, if the error in position and velocity is to be predicted, one will need six ANNs, one for each of the three components of the position and three components of the velocity in the three-dimensional space. Results indicate that the use of ANNs for conjunction assessment can bring higher accuracy and good computational load than current physics-

based methods (Sanchez and Vasile, 2021), but the errors made with this model are still too large to be properly considered for accurate collision risk calculation. Further studies and testing should be done to increase the accuracy of ANNs for orbit prediction.

Convolutional Neural Network

This type of neural network is designed to process grid-like data, such as images. As with ANNs, they consist of a series of layers. In these algorithms, firstly one can find the convolutional layer, which extracts the local patterns and features, followed by the pooling layer, which obtains the most dominant features. The subsequent fully connected layers learn from the output of the sequence of convolutional and pooling layers and can classify and predict the outputs of the network. They are particularly effective for image classification and feature extraction.

As such, they are effective in the extraction of complex relationships and patterns in the data. Convolutional Neural Networks (CNNs) are typically used for image processing and spatial data and are not commonly known for predictions based on time series. However, by treating the latter as spatial data, CNNs can be used for time series forecasting. Nonetheless, Recurrent Neural Networks (RNNs) are typically used instead for sequential data and are more suitable for that type of problem (this will be explained more in detail in the subsequent paragraph). Since the papers reviewed for orbit prediction using deep learning also showed a preference for Recurrent Neural Networks over CNNs (Dave et al., 2020), the latter will not be further considered for this project.

Recurrent Neural Network

Such as CNNs, Recurrent Neural Networks can effectively process structured data. While CNNs are most commonly used for spatial data, RNNs are more effective for the processing of sequential or time series data. In traditional deep learning methods, all inputs and outputs are independent and the information flows in one direction. RNNs are different since they have a 'memory', meaning that the output from the previous node is fed as input to the current node. One further difference of RNNs when compared to ANNs is that, while the latter have different weights for each node, the former share a common weight on each layer. This weight is adjusted as the model learns by calculating the errors of the output layer and using that on the input layer.

When compared to Feed-Forward Neural Networks (FFNN) such as ANNs and CNNs explained above, which are unable to retain previous inputs, RNNs are considered more effective for the analysis and prediction of time series data. As a consequence, they are preferred for this project over feed-forward neural networks.

One of the drawbacks of using RNNs is that they are prone to the vanishing gradients problem, where gradients become smaller over time which causes the weights to become insignificant and, as a consequence, the model stops learning. Long Short Term Memory algorithms were created as a solution to this problem and to address long-term dependencies. In traditional RNNs, the current state might not be predicted accurately if the previous state is not in the recent past. LSTMs, as a result, contain special units called 'memory cells', and the flow of information in and out of these is controlled by gates. Each cell has three gates: an input gate, an output gate and a forget gate. These gates control the information that is needed to predict the output of the network.

As a result, LSTMs are more suitable for capturing long-term dependencies and handling sequential data than traditional RNNs. As such, they are capable of capturing more complex patterns and dependencies. In the literature reviewed by Choumos et al. (2024), the majority of papers used for orbit prediction used LSTM algorithms. Results have shown that the prediction error of the RNN model is lower than traditional mathematical models (Jadala et al., 2022; Yang et al., 2020). When compared to other models reviewed here, results showed improvement over the same datasets (Shin et al., 2022).

Physics-Informed Neural Network

The governing physics laws of a system can typically be described by a set of partial differential equations (PDEs). Physics-Informed Neural Networks (PINNs) form a type of machine learning model that integrates the physical properties of a system into a classic neural network architecture. It is therefore suitable to solve sets of PDEs and model the dynamics of a system, where the availability of data to train the model is limited. By embedding the physical knowledge into the architecture, the information that is contained in the dataset can be increased.

PINNs incorporate physics-informed layers, which contain the knowledge of the physical laws describing the dynamics of the system, into the classical ANN architecture. By integrating this knowledge into the learning process, the PINN can be trained with limited data while ensuring that the solution does not violate the governing physics of the system. It is still a data-driven solution since the model learns from the observations that are input into it, but at the same time, it makes sure that the solution is physically feasible via integrated prior knowledge of the environment.

Although this model is a FFNN, which was regarded earlier as less suitable for the problem than an RNN, the use of a PINN still offers a few advantages over other ML algorithms. Firstly, it is befitting in situations where there is limited available data, which tends to be the case for orbit prediction. This will be explained more in detail in Section 2.5. On top of that, the results obtained from it are more accurate and consistent with the physics that govern the system. Because of this, the predictions provided by this model are easier to interpret. On the other hand, PINNs are more computationally expensive since they also need to solve optimisation problems involving physical constraints and bounds. Since the dynamics considered for orbital dynamics is also quite complex due to non-linearities, the model design and tuning might also prove to be more difficult when compared to other algorithms.

Although a PINN algorithm has not been used in the field of orbit prediction and collision risk assessment, it has been used for other areas in orbital dynamics. A popular one is the modelling of gravity fields over irregular bodies (Cheval, 2023; Martin and Schaub, 2022). Results confirm the ability of this type of neural network to predict the gravity field acceleration around such a body.

Summary

All the algorithms considered have been introduced, and their advantages and disadvantages explained. With this in mind, a decision must be made on the machine learning model to choose for the orbit prediction and collision probability calculation. As mentioned, a deep learning algorithm based on neural networks is preferred, since it can capture the underlying complexities of the dynamics better. From the models reviewed, LSTM and PINN are thought to be the best options for this project due to the ability to deal with sequential data of the former and the integration of the physic laws in the model of the latter. Both of them are considered to be computationally expensive, although the computational load of the model once it is trained is significantly reduced. It is thought that PINNs are more suitable for this problem since the physics of the environment are integrated into the model and, as a consequence, the predictions obtained will adhere to the laws of motion. PINNs can also solve the issue of the lack of public data to train the model since the integration of physics allows for the training data to be more scarce as the model itself will contain some information about the environment already. On the other hand, PINNs are not meant to handle sequential data, thus a hybrid of LSTM and PINN will be used to approach the problem posed regarding using AI for orbit prediction.

2.5. Training Data Sources

One of the main challenges in the use of AI-based approaches in the field of SST is the scarcity of publicly available data. The main trend observed in the literature was the use of synthetic data to train AI models. Access to ground-based or space-based sensors is very limited, and the majority of the assets are owned by private companies, or their data is classified and therefore not publicly published. There are, however, some types of data that can be accessed by the general public, although they come with limitations.

Choumos et al. (2024) provide a list of the types of data that are typically used in the papers reviewed and studied. This is summarised in Figure 2.3. This list, however, is presented for all the papers reviewed, not only the ones in which AI-based methods are used and not only for orbit prediction and conjunction assessment. Appendix B provides an extensive list of the data sources and methods used for orbit prediction, collision avoidance and manoeuvre detection from all the papers reviewed by Choumos et al. (2024), shown in Table B.1. In this table, one can observe that the most common data sources for orbit prediction and collision avoidance are Two-Line Elements (TLEs) and orbital state data (typically synthetic data). These two data sources, alongside laser ranging (which is also used, albeit more scarcely) will be summarised and compared in the following paragraphs.

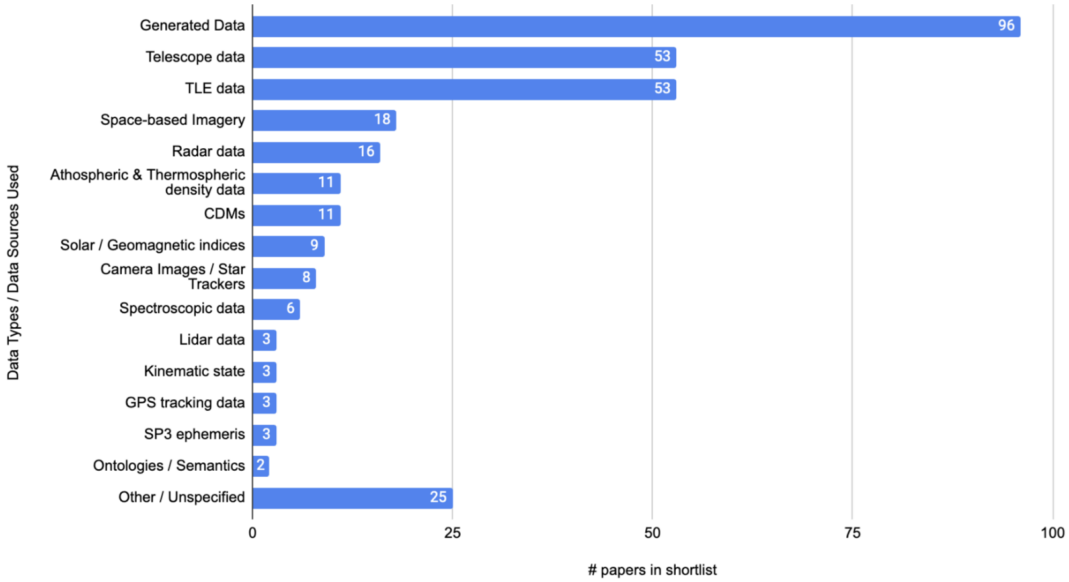


Figure 2.3: Data types and sources used in shortlisted papers (Choumos et al., 2024)

Two-Line Elements

Two-Line Elements (TLEs) are one of the most common data types that are used in the field of orbit prediction and collision avoidance. TLE data for the objects that are tracked by the US Space Surveillance Network (USSSN) is publicly available in Space Track, and includes information on the mean orbital elements at a specific epoch for a particular catalogued object. There are several challenges and limitations to using this type of data, however. Firstly, the TLEs are derived from observations over a short period, meaning that, while accurate at the moment of the observation, they get more inaccurate the further into the future the objects are propagated. Secondly, the models that are used to generate the TLEs are simplified due to a lack of information about the satellite. Since satellite-specific parameters (such as the drag coefficient, the solar radiation pressure coefficient, size and surface area...) are probably not published, the propagation models of different agencies also need to account for this and assume a simplification of the model. TLEs are discussed more in detail in Appendix D, with an explanation of how to interpret them and how they are generated. Since conjunction analyses are based on the calculation of the probability of collision P_c, the covariance information on the different objects is needed, which cannot be obtained via the TLEs. For Conjunction Analysis, the O/Os send their predicted ephemeris (generated with higher accuracy since they have the full information about the satellite) to USSPACECOM, but once again this is not information that is commonly shared.

Laser Ranging

The database of USSPACECOM is the largest public database for the tracking of objects in space. There are, however, other types of data that can be used for orbit prediction. The International Laser Ranging Service (NASA, 2023a), for example, has more accurate ephemeris publicly published, obtained via laser ranging techniques. Measurements are collected from different sources and have been collected for decades now, increasing the robustness of the predictions through an extensive database for specific objects and a reduction of biases. Although the measurements taken by the ILRS are more accurate, it is to be taken into account that the database is more limited and that there is also a limited amount of examples in the literature using this type of data.

Synthetic Data

Other sources can be found with published TLEs or orbital state data for different RSOs, although most of these only have a limited range of objects tracked which mostly include active satellites. This is the reason why many practitioners use the database from USSPACECOM since it is the largest that can be found, even if the orbital information comes in the form of TLEs. Due to the scarcity of publicly available data, another solution is the use of synthetic or generated datasets. Now, this allows for the

creation of a myriad of geometries and orbits that can be fed into the desired ML algorithm and make it more robust. By providing the model with a wider range of scenarios, a more thorough study can be made to test the algorithm. On the other hand, synthetic data (in comparison to real-life, historical data) fails to capture the intrinsic complexities of the space environment and the accuracy of the dataset is limited to the accuracy of the propagator and environment simulation. Another issue with the use of synthetic data in the literature is that the way in which it is generated is not typically mentioned, although it is assumed that the data is generated in such a way that the AI model can make robust predictions and investigate a wide variety of orbits and conjunction scenarios. This, however, remains to be proved.

Summary

The data types considered in the literature were mostly in the form of TLEs, data from the ILRS, and orbital data derived from generated datasets, all of which were discussed above. There are other types of data used in the papers reviewed, none of which are used extensively. It is desired in this work to deal with data from real satellites and pieces of debris for the sake of comparison and validation of the software developed. However, if deep learning techniques are to be used, a large number of data points are going to be needed. Not many studies in the literature specified the size of the database used, but those who did mentioned numbers in the order of 10^4 (Peng and Bai, 2018c) up to 10^5 (Yang et al., 2020). The latter could be achieved if TLEs for several RSOs were to be used to train the model for orbit prediction.

Ideally, one would generate a dataset solely based on real-life measurements to train the model, but that is highly unlikely in the field of orbit prediction. TLEs are one of the few public sources of data for satellite information, and unfortunately these are sparsely spaced in time. At an average of 2 to 3 TLEs per day (18SDS and 19SDS, 2023), this is not considered enough to create a pattern that a neural network can learn. Instead, the approach taken is a hybrid of real and synthetic data: the initial state of the satellite can be taken from a TLE and then will be propagated to create a synthetic dataset. This way, the dataset is ensured to have enough data points to show the fundamental patterns of the state elements over several orbits, while belonging to real satellites which also improves the applicability of the algorithm to be built.

2.6. Reference Missions

This section details the missions and scenarios that are to be selected for this project and aid in the development of the machine learning algorithm for collision avoidance. The desire to use real-life data for both the training and the validation and verification of the algorithm generates a need to look for case missions for these purposes.

A final goal in the field of deep learning for orbit prediction and collision detection would be to have a generic model that can be used to predict the state and uncertainty of all RSOs. However, that is an extremely complicated task, since it would require a vast number of satellites for the training over a myriad of orbit geometries. Due to time constraints in this thesis, and workload limitations, the focus will be put on training a machine learning algorithm with a single satellite and testing it with different objects to test its generalisation capabilities. This will bring some insight on which steps to take to build this more generic tool for orbit prediction.

Since the majority of the satellites and debris tracked are in the LEO region, as was observed in Figure 1.1, only these cases will be considered for now to train the model, even though several RSOs will be used to test its generalisation abilities. When it comes to the altitude chosen for the reference satellite, the lower limit shall be selected so that the satellites considered shall not have drag as their dominant force. The main reasoning behind this is that orbital decay is not to be considered in this work for simplification and that the drag force is mostly dominant in lower altitudes, resulting in higher uncertainties being introduced in the model. To deal with lower uncertainties, the altitude of the objects considered will be higher than 400km. These limits are also selected based on the regions of interest due to higher orbital density shown in Figure 2.4 since they have the highest risk of collision and therefore higher accuracy is needed. In this figure, the first region of high density is around 500 - 600 km altitude, which is where the Starlink satellites, the largest satellite constellation, are located. Another region of

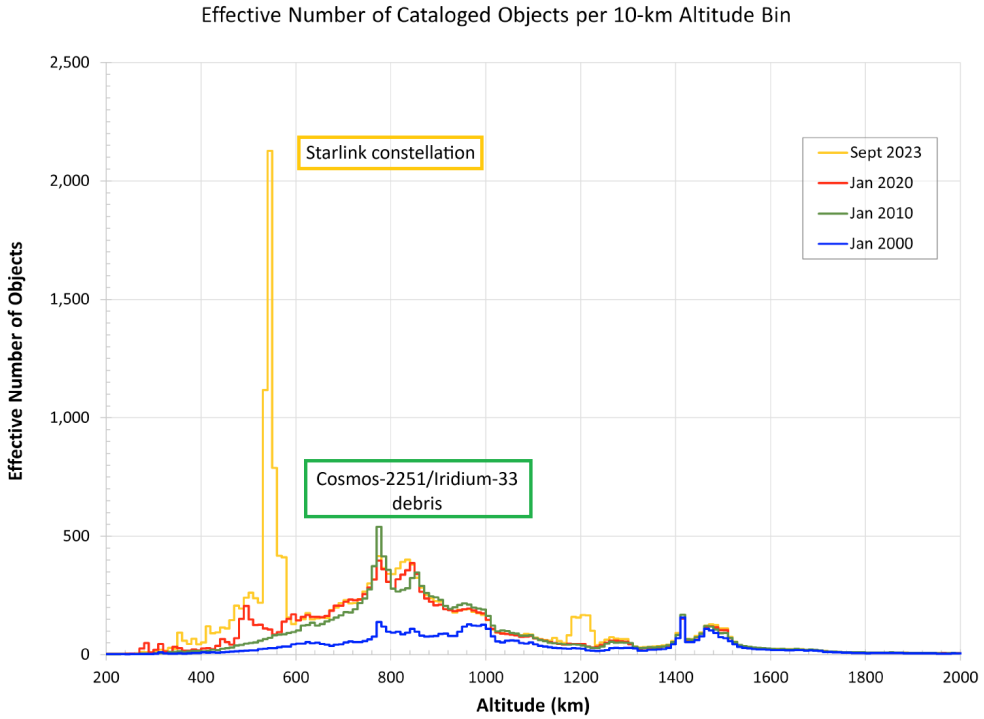


Figure 2.4: Density of RSO tracked by USSPACECOM in the LEO region at four different epochs (NASA, 2023c)

Table 2.4: Summary of parameters for the reference satellites chosen, parameters obtained from Kelso (2009)

Mission	Mass	Reference Area	Altitude	Inclination
Iridium-33	689 kg	2.2 m^2	780 km	86°
Cosmos-2251	900 kg	4 m^2	800 km	74°

high interest is around the 800 km altitude, where there is a large concentration of debris due to the collision of Iridium-33 and Cosmos-2251, explained in more detail later. When possible, satellites with known characteristics will be used to train and test the model.

Aside from these, it is desired to test the model in a real-life scenario. To do so, well-documented collision events from the past will be studied. A famous event is that of the collision between the Cosmos-2251 and Iridium-33 back in 2009, which produced around 1800 pieces of debris larger than 10 cm in diameter (Anz-Meador and Liou, 2010). The Iridium-33 was a communications satellite in the U.S. Iridium constellation, active until the fatal collision on February 10, 2009. This constellation consisted of 66 satellites, located in six different orbital planes. Cosmos-2251 was a Russian satellite launched in 1993 which remained operational for two years. It was used for military communications and had an estimated weight of 900 kg. Details from both satellites are summarised in Table 2.4. While Cosmos-2251 was inactive, Iridium-33 was manoeuvrable until the moment of collision. As such, it had full capacity to perform a collision avoidance manoeuvre but the risk of collision calculated at the moment was not severe enough to grant the manoeuvre. This event highlighted the need to provide more accurate collision assessments, a field that continues to develop day by day.

Considering this desire to test on a real-life scenario, it has been decided that the satellite to train the model with will be Iridium-33. since it was active at the moment of the collision, it is more suitable for this task than Cosmos-2251, since a prevision of the impact would have led to an avoidance manoeuvre.

To test the model's generalisation abilities several RSOs will be used. These will be selected to be at different altitudes, eccentricities and inclinations (amongst others) to assess how well the algorithm fares with different inputs, and what recommendations can be given for a more generic neural network.

2.7. System and Mission Requirements

The work required for this project can be separated into two different blocks: firstly, an extension of the DA-GMM algorithm will be performed, followed by the development of the AI-based algorithm for orbit prediction and collision risk assessment. As such, there will be different requirements for the two parts of the project.

Requirements for the DA-GMM

Since the algorithm developed by Leon Dasi (2021) has already been extensively verified and validated, the requirements specified below will focus on the performance of the model with the addition of the Solar Radiation Pressure.

DAGMMMR-01 The modified model shall calculate the Solar Radiation Pressure acceleration.

DAGMMMR-02 The Solar Radiation Pressure acceleration shall be modelled with a relative error lower than 5% with respect to a verified astrodynamics software tool.

DAGMMMR-03 The error in the P_c calculation with respect to the 10^9-sample Monte Carlo analysis shall remain lower than for the unmodified DA-GMM.

DAGMMMR-04 The inclusion of rotational motion in the model shall not be considered.

Requirements for the Machine Learning Algorithm

The following block of the thesis work focuses on the development of a neural network for orbit prediction. Upon taking a look at the decisions taken in this chapter, the following mission requirements are identified:

MLMR-01 The neural network shall predict the orbit of the reference RSO with an accuracy higher than that obtained through the propagation of simplified numerical models.

MLMR-02 The system, once trained, shall have a lower computational load than traditional collision risk assessment methods.

MLMR-03 The simplifications in the acceleration modelling shall have an error with respect to a highly accurate model of $< 10^{-7}$ m.

MLMR-04 The system shall be able to propagate the orbit of the objects at least two days into the future.

MLMR-05 The system shall accurately predict the collision probability of past documented events.

On top of that, the following system requirements have been identified:

MLSR-01 The algorithm shall be tested with orbital data from different RSOs with different orbital characteristics.

MLSR-02 The system shall be robust to variations in the initial state and uncertainties.

MLSR-03 The neural network shall produce as a result the state of an object at intervals of time.

MLSR-04 The model shall estimate the uncertainty of the state of an object and due to the prediction error of the neural network.

MLSR-05 The model shall calculate the probability of collision between two objects.

MLSR-06 The system shall neglect long-term variations in the environmental forces and variables.

MLSR-07 The algorithm shall be suitable for integration with current conjunction assessment techniques.

2.8. Methodology

Two different methods are taken to answer the main research question: the DA-GMM and the PINN. The former belongs to a model that has already been developed and thus only needs to be extended. The core of this method is fundamentally and mathematically complicated to explain, thus only the overall concepts are discussed. For a more exhaustive understanding of this method, the reader shall be referred to the original thesis detailing it (Leon Dasi, 2021). The architecture of the algorithm is divided into four parts:

1. Gaussian Mixture Model Split
2. Taylor Series Integration in Differential Algebra
3. Uncertainty Propagation
4. Collision Probability Calculation

Details on the architecture are provided in Chapter 5. The only aspect that needs to be reviewed now is that the core of the propagation happens in Step 2, and thus this shall be the algorithm to be modified. The extension is simple, as only the SRP acceleration needs to be added to the propagation block. This is discussed in Chapter 6, with C_R and the uncertainty in the $CSRP$ being added as inputs to the model.

The PINN, on the other hand, needs to be developed from scratch. An overview of the methodology selected for this development is provided in this section as a guide for the reader. Machine learning algorithms, and neural networks especially, can be divided into three blocks: data pre-processing, the core architecture of the algorithm, and the data post-processing:

1 Data Pre-processing
The first step of this method is the data collection. TLEs are selected as the source of initial state information, with a highly accurate propagator used to create a synthetic dataset due to a lack of public tracking data. Most machine learning models require some pre-processing of the input data to ensure their correct functioning. This includes handling missing data points, as they should be equally spaced in time to improve the effectiveness of the training process, and data normalisation to ensure that all input features are in the same range.

2 Physics-Informed Neural Network
Once the data is collected and pre-processed, it is used as the input to the neural network architecture. Other inputs include the model hyperparameters (i.e., the model parameters not optimised in the training process and thus describe its architecture). To start with, this neural network needs to be trained, for which a thorough study of the best possible architecture and its limitations is required. Once a trained model is obtained, the algorithm can be used for orbit propagation.

3 Data Post-processing
The output of the PINN is a set of future normalised satellite states. As such, it is fundamental to do some post-processing to transform this raw data into interpretable material. The first step to achieve this is to de-normalise the data. With the full-scale model, the uncertainties can be estimated and the probability of collision can be calculated, which is the ultimate goal of the algorithm.

The software architecture is presented in Chapter 5. All the steps required to calculate the risk probability of any test case have been briefly discussed. A detailed explanation of each of these steps is provided in Chapter 7.

3

Astrodynamics

The aim of this chapter is to provide a background on all the elements that need to be taken into account when selecting the simulation environment for orbit propagation. The different reference frames that are needed for the model are summarised in Section 3.1, with the required rotation matrices detailed in Subsection 3.1.2. A comparison of the different representations of the state of an object in space is given in Section 3.2. In Section 3.3, the equations of motion describing the movement of RSOs are mathematically formulated. The perturbation forces modelled in the DA-GMM are explained and formulated in Section 3.4 and the environment models used, in Section 3.5. Finally, a summary of the models selected for the simulation environment will be presented in Section 3.6 for easier visualisation.

3.1. Reference Frames

This section includes a review of the different coordinate systems and frames that are used to describe the state of an object which are relevant for conjunction analysis. All frames and frame transformations, unless specified otherwise, are taken from Mooij (2019).

3.1.1. Definitions

The different definitions of the reference frames needed for this project are defined below. These include their origin and axes directions, alongside their use in the context of orbital dynamics and collision risk assessment.

Earth-Centred Inertial Frame (ECI)

Inertial reference systems are required for orbital mechanics, since the equations of motion of a spacecraft are often described with respect to an inertial frame, without taking into account the rotation of the Earth. The ECI is not truly inertial, since the Earth has movement around the Sun. The effect of this is negligible for satellites orbiting this planet, and therefore it is considered a pseudo-inertial coordinate system in which the equations of motion can be described. The origin of this reference frame is the centre of mass of the Earth, with the direction of the axes being defined with respect to a reference direction, called the vernal equinox (Υ). This direction will depend on the epoch that is chosen to represent this coordinate system. A common practice is to use the J2000 frame for this, where the X-axis aligns with the Vernal Equinox on January 1, 2000 at 12:00 (UTC). The Z-axis is aligned with the rotation axis of the Earth, and the Y-axis completes the right-handed system.

Earth-Centred Earth-Fixed Frame (ECEF)

This reference system also has its origin in the centre of mass of the Earth. However, unlike the ECI, it is not inertial as it rotates alongside the planet and is therefore fixed to its surface. The X-axis points towards the Greenwich meridian and is located on the equatorial plane (i.e., points towards the point of 0^{o} latitude and 0^{o} longitude). The Z-axis is, such as with the ECI, aligned with the rotation axis of the Earth and the Y-axis completes the right-handed system. The location of the ground-based sensors

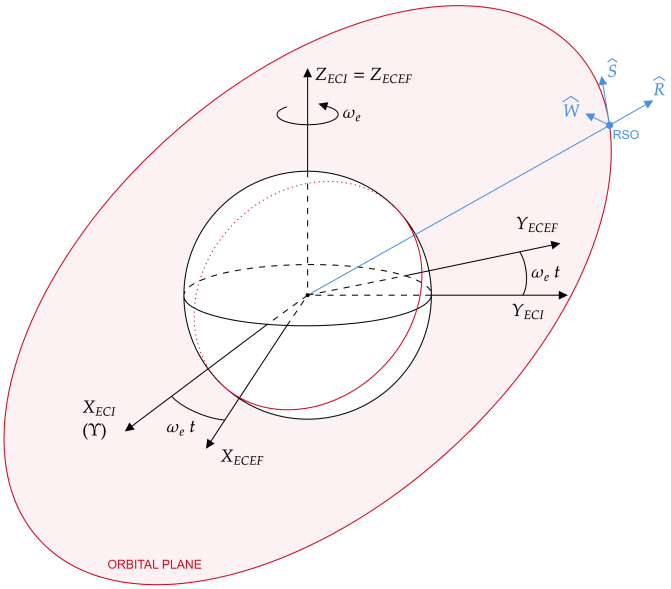


Figure 3.1: Representation of the ECI, ECEF and RSW frames

that track RSOs is described with respect to this reference system and is therefore also needed for conjunction analysis. A representation of this coordinate system is shown in Figure 3.1, alongside the ECI frame for comparison.

Radial Along-Track Cross-Track Frame (RSW)

This reference frame is centred on the centre of mass of the RSO taken into consideration and is considered to be a local frame. The direction of the axes is based on the location of said object with respect to a central object, which in this case will be the Earth. The X-axis (or Radial, R-axis) follows the position vector between the centre of mass of the central object and the RSO, pointing away from the central object. The Y-axis (or Transverse, Along-track axis) follows the direction of the RSO, and thus it forms the orbital plane alongside the X-axis. The Z- axis (or Normal, Cross-track axis) is perpendicular to the orbital plane. This coordinate system is typically used to represent the uncertainties of the state of the object in consideration. Figure 3.1 shows how this frame is related to the ECI and ECEF frames.

3.1.2. Frame Transformations

A number of reference frames have been defined, all of which are relevant in the context of conjunction assessment and collision probability calculation. As measurements and other inputs will be obtained in different frames, it is relevant to introduce the frame transformations required. This can be done via rotation matrices, with unit axis rotations about Euler angles being the most common approach.

Assume a transformation from an arbitrary frame A to a frame B. The rotation matrix between these two frames is denoted by $\boldsymbol{C_{B,A}}$. This matrix can be formed via a combination of unit rotation matrices, depending on the axes and angles of rotation that are needed for the transformation between frames. For reference, the rotation matrices around a single axis for an arbitrary angle α are described in Equation (3.1), Equation (3.2) and Equation (3.3).

$$\boldsymbol{C_x}(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{3.1}$$

$$\boldsymbol{C_y}(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \tag{3.2}$$

$$\boldsymbol{C_z}(\alpha) = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

These represent the unit axis rotations about the X, Y, and Z-axes, respectively. These can also be represented by the subscripts 1, 2, and 3.

A useful property of these matrices is that each of the unit axis rotation matrices is orthonormal (i.e., the inverse of the matrix is equal to its transpose), and therefore their product will also be orthonormal. This is represented as follows:

$$\boldsymbol{C_{B,A}} = \boldsymbol{C_{A,B}}^{-1} = \boldsymbol{C_{A,B}}^{T} \tag{3.4}$$

It is to be noted that the transformation matrix from frame A to frame B can be represented as the transpose of the transformation from B to A.

Using the appropriate angles and axes, transformations between the reference frames mentioned in Section 3.1 can be developed. It is noted that not all possible transformations are provided, only the most relevant ones. Other rotation matrices can be constructed through the ones presented below.

ECEF to ECI

From this point in the report, the ECEF reference frame will be indicated with the subscript R and the ECI, with the subscript I. This is for the sake of simplicity in the representation of the formulae and equations. The transformation from the ECEF to the ECI reference frames will be represented by the transformation matrix $\boldsymbol{C_{I,R}}$.

It is assumed that both frames coincide in origin and axes directions at a desired epoch, which as mentioned earlier is chosen to be on January 1, 2000. The origin and Z-axis for both will still be coincidental for consequent epochs. The rotation of the Earth about the Z-axis now needs to be taken into account for this transformation. Only the rotation matrix about this axis is therefore needed, and the rotated angle between the frames is equal to $\omega_e t$ at an arbitrary time t. The rotational rate of the Earth is represented by ω_e, with a value of approximately 7.2921159 $\cdot 10^{-5}$ rad/s, and t represents the time since the epoch J2000. This angle represents how much both X_R and Y_R are shifted with respect to X_I and Y_I in the orbital plane. The transformation matrix is therefore described in Equation (3.5).

$$\boldsymbol{C_{I,R}} = \boldsymbol{C_3}(-\omega_e t) = \begin{bmatrix} \cos(\omega_e t) & -\sin(\omega_e t) & 0 \\ \sin(\omega_e t) & \cos(\omega_e t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

RSW to ECI

The RSW frame has its origin in the centre of mass of the object into consideration and is defined with respect to the orbital plane. As such, to build the transformation matrix between the RSW and the ECI frames, orbital parameters need to be used. Following Figure 3.2, firstly the rotation of the RSW frame about the Z-axis over the angle $-(\omega + \theta)$ (or rather, $-u$), is performed. Then, a rotation over $-i$ about the line of nodes is applied, and the final rotation is about the Z-axis (i.e., on the equatorial plane) about the angle $-\Omega$. All these angles will be introduced in Subsection 3.2.2. This frame transformation is represented in Equation (3.6), which is obtained from Wakker (2015).

$$\begin{aligned} \boldsymbol{C_{I,RSW}} &= \boldsymbol{C_3}(-\Omega)\boldsymbol{C_1}(-i)\boldsymbol{C_3}(-u) \\ &= \begin{bmatrix} \cos\Omega & -\sin\Omega & 0 \\ \sin\Omega & \cos\Omega & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i & -\sin i \\ 0 & \sin i & \cos i \end{bmatrix} \begin{bmatrix} \cos u & -\sin u & 0 \\ \sin u & \cos u & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.6}$$

3.2. State Vector

There are many different ways of representing the state of an object in the field of astrodynamics. The choice of representation also affects the accuracy achieved in the simulation, as some state models will introduce errors due to singularities and/or a large variation of the elements defining the state. The most common state representation is the use of the Cartesian coordinate system, which indicates the position and velocity of an object in the three-dimensional space. Also common in astrodynamics is the representation of the state of an object via Kepler elements, which define the shape of its orbit. There are alternative methods to describe the state of an RSO to tackle singularities and achieve a higher computational accuracy. These, however, are more difficult to interpret and visualise and add complexity to the modelling of the software. In the literature reviewed, the Cartesian coordinate system (Yang et al., 2020), Kepler elements (Peng and Bai, 2018c) and Modified Equinoctial Elements (MEE) (Sanchez and Vasile, 2021) have been used for collision risk assessment. Therefore, these are the state models that are going to be reviewed in Subsection 3.2.1, Subsection 3.2.2 and Subsection 3.2.3, respectively. The Unified State Model state representation is also considered useful for propagation purposes and for machine learning implementation, and is introduced in Subsection 3.2.4,

3.2.1. Cartesian Coordinates

The state in Cartesian coordinates is represented by the position and velocity vectors in the three-dimensional space with respect to a reference frame:

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{r} & \boldsymbol{v} \end{pmatrix}^T = \begin{pmatrix} x & y & z & v_x & v_y & v_z \end{pmatrix}^T \tag{3.7}$$

This is the most common type of state representation, due to its simplicity and easy interpretation. As these represent the state of an object at a specific moment in time, it is intuitive to understand the trajectory of said object as it is propagated forward in time. Additionally, the equations of motion are easier represented in this system and therefore the numerical integration is quite straightforward. As a downside, the position and velocity components tend to suffer from large variations in their values, resulting in the induction of errors in the propagation of the state due to large state derivatives. These will therefore be avoided for propagation purposes.

3.2.2. Keplerian Elements

Instead of a description of the instantaneous position and velocity of an item, one could find the elements that describe the orbit in which it is found. This is the purpose of the Keplerian elements, for which five components are used to define the shape, size, and orientation of the orbit of an object, whereas a sixth component is used to describe its position inside said orbit:

$$\boldsymbol{x} = \begin{pmatrix} a & e & i & \Omega & \omega & \theta \end{pmatrix}^T \tag{3.8}$$

where the semi-major axis (a) defines the size of the orbit, whereas the eccentricity (e) can be used to define its shape. The inclination (i) defines the angle between the orbital plane and a reference plane, which is usually that of the central body. To describe the actual orientation of the orbital plane with respect to this reference plane, the Right Ascension of the Ascending Node (RAAN, or typically symbolised by Ω) can be used. The argument of periapsis (ω), on the other hand, describes the orientation of the orbit itself regarding the orbital plane. All elements defined in this subsection are shown in Figure 3.2, for an easier and more intuitive understanding. In a Keplerian orbit, which assumes that the only forces acting on a two-body system are due to the point-like gravitational accelerations of these, the five aforementioned elements are constant. The state derivatives would therefore have a null value and would not induce numerical integration errors. Due to the incorporation of perturbing accelerations in the system, the assumption of Keplerian movement is invalidated and thus these elements are not constant anymore. The variation induced is nonetheless still small, and the state derivatives are still noticeably smaller than those resulting from the use of Cartesian coordinates as state representation. The sixth element, which is constantly changing as it specifies the actual position of the object along its orbit, can be represented in different ways. Commonly, one finds it expressed as the true anomaly (θ).

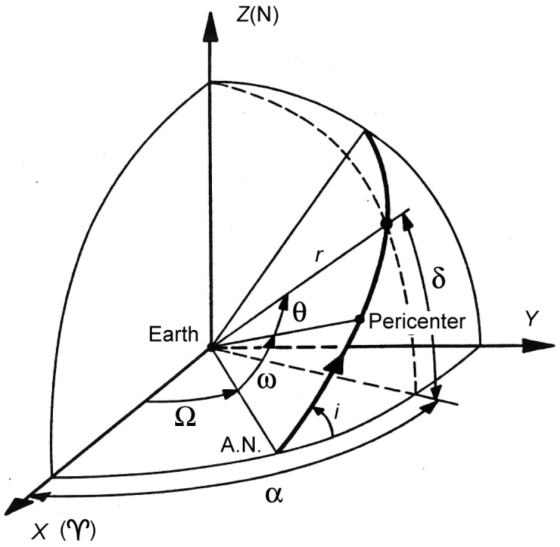


Figure 3.2: Visual representation of the Keplerian elements (Wakker, 2015)

This is defined as the angle between the object and the periapsis of the orbit, with respect to the central body, which sits at one of the focus of the ellipse that defines the orbit. Following from Figure 3.2, the argument of periapsis and true anomaly are both defined in the orbital plane and can be combined to give the argument of latitude: $u = \omega + \theta$.

These elements can be easily linked to Cartesian coordinates. Unless stated otherwise, all equations are adapted from Wakker (2015). Start by defining the local position and velocity:

$$r = ||\boldsymbol{r}|| \quad V = ||\boldsymbol{v}||$$

With these, the semi-major axis and eccentricity can be obtained as follows:

$$a = \frac{r}{2 - r\frac{V^2}{\mu}} \tag{3.9}$$

$$\boldsymbol{e} = \frac{\boldsymbol{v} \times \boldsymbol{h}}{\mu} - \frac{\boldsymbol{r}}{r} \quad \rightarrow \quad e = ||\boldsymbol{e}|| \tag{3.10}$$

To define the rest of the elements, some auxiliary variables need to be defined, namely the angular momentum $\boldsymbol{h}$ and the vector normal to the orbital plane $\boldsymbol{N}$:

$$\boldsymbol{h} = \boldsymbol{r} \times \boldsymbol{v} \quad \rightarrow \quad h = ||\boldsymbol{h}|| \tag{3.11}$$

$$\boldsymbol{N} = \hat{\boldsymbol{z}} \times \hat{\boldsymbol{h}} \quad \rightarrow \quad N = ||\boldsymbol{N}|| \tag{3.12}$$

The inclination (i), RAAN (Ω), argument of perigee (ω) and true anomaly (θ) can be obtained as follows:

$$\cos i = \frac{h_z}{h} \tag{3.13}$$

$$\cos \Omega = \frac{N_x}{N} \tag{3.14}$$

$$\cos \omega = \hat{e} \cdot \hat{N} \tag{3.15}$$

$$\cos \theta = \hat{e} \cdot \hat{r} \tag{3.16}$$

where the values of the inclination are in the range $0° \leq i \leq 180°$; the right ascension of the ascending node has a value $0° \leq \Omega \leq 180°$ if $N_Y \geq 0$ and $180° \leq \Omega < 360°$ otherwise; the argument of periapsis ranges between $0° \leq \omega \leq 180°$ if $e_z \geq 0$ and $180° \leq \omega < 360°$ if that is not the case; and finally the true anomaly has a value in the range $0° \leq \theta \leq 180°$ if the radial component of the velocity, v_r, is greater or equal than 0 and $180° \leq \theta < 360°$ if v_r (i.e., the radial component of the velocity) has a negative value.

It is also common to define in this context the mean anomaly, M, to define the position of a satellite with respect to its orbit. This angle does not represent any physical quantity, since it is based on the assumption of a constant rate of angular change n (typically called mean motion), and therefore, a circular orbit. It is useful to use in orbital mechanics since it is directly related to time. The third and final angular representation of the position of a body in an orbit is introduced: the eccentric anomaly, E. This angle is also used in elliptical orbits, but unlike the true anomaly, it is measured from the centre of the ellipse. The relationship between these is formulated as:

$$n = \frac{2\pi}{T} = \sqrt{\frac{\mu}{a^3}} \quad \rightarrow \quad M = n(t - t_{perigee}) = E - e \sin E \tag{3.17}$$

The six classical orbital elements are those shown in Equation (3.8), but it is relevant to know that the mean and eccentric anomalies can still be used as orbital elements. Depending on the context, the classical orbital elements might be switched with other variables defined in this section to describe the orbit of an object.

The representation shown in this subsection makes for an easier visualization of the geometry of the orbit and thus the motion of the spacecraft inside this orbit is also quite intuitive. The equations of motion defined using this system are still quite simple for the case of a Keplerian orbit, but the addition of perturbations in the model can complicate the differential equations that need to be solved. On top of that, there are a few singularities that are induced as a result of this system representation: as $e \to 0$, since the argument of periapsis becomes indeterminate, and $\sin i \to 0$ (or $i \to 0$, $i \to 180°$), due to the right ascension of the ascending node becoming indeterminate. Keplerian elements will therefore be used in this project for visualization purposes, as it is easier to define the orbit with this set of elements. On top of that, TLEs include the mean Keplerian elements of a satellite at a specific epoch, further supporting their use for orbit definition and analysis.

3.2.3. Modified Equinoctial Elements

To account for the singularities that are induced in the equations of motion for the Keplerian elements, another set of elements was defined to represent the state of an object in space, which were called the Modified Equinoctial Elements (Walker et al., 1985). Similarly to the Keplerian elements, there is one fast-changing element defining the position of the object inside the orbit, named true longitude (L). Through a re-formulation of the other elements, one can still obtain a state representation that applies to all orbits while removing the singularities of the system, leading to a more stable representation of the state. By keeping the state representation closer to that of Keplerian elements, the physical significance can be kept, albeit it is less intuitive than the others reviewed here. Additional geometry knowledge and effort must be put into understanding the geometrical representation of these elements. The elements that are used for this representation are summarised as:

$$\boldsymbol{x} = \begin{pmatrix} p & f & g & h & k & L \end{pmatrix}^T \tag{3.18}$$

The equations of motion become more complex for this representation, although it is more stable than for the two others. The application of these in the simulation environment is more challenging, and there are more limited resources and software packages available. The use of Cartesian coordinates or Keplerian elements is typically more convenient, but simple transformations can be applied to visualise the orbit. MEEs also provide higher numerical stability and are preferred for propagation and integration purposes.

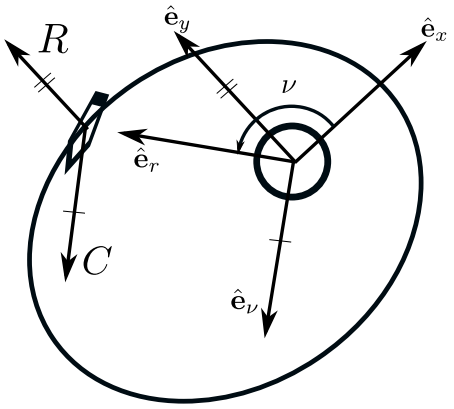


Figure 3.3: Velocity hodograph parameters used in the Unified State Model representation (Vittaldev, 2010)

3.2.4. Unified State Model

Another state representation that is free of singularities and, at the same time, is even more stable numerically than the MEE is the Unified State Model representation (USM7), proposed by Vittaldev (2010). Unlike the state representations introduced up until now, this model is composed of seven elements. Out of these, three elements are related to the velocity hodograph (i.e., a representation of the orbit in the velocity space) and are constant, and the other four are used to represent the orientation of the orbital frame with respect to an inertial frame of reference. This is summarised as follows:

$$\boldsymbol{x} = \begin{bmatrix} C & R_{f1} & R_{f2} & \eta_0 & \epsilon_{01} & \epsilon_{02} & \epsilon_{03} \end{bmatrix}^T \tag{3.19}$$

where the velocity parameters are C and $\boldsymbol{R}$, which can be visualised in Figure 3.3, where the former is the radius of the orbit's velocity hodograph and the latter represents the displacement of the centre of the body with respect to the origin of the reference frame. The orientation of the orbit is described with quaternions, which is a set of four elements (one scalar, η_0, and a vector, $\boldsymbol{\epsilon}$) that provides the rotation of an object in the 3D-space without singularities.

USM7 elements have been proven to be more stable than other state representations (Vittaldev, 2010) and are easier to visualise than MEE. It has also been verified that numerical integration and propagation with USM7 are performed more accurately and faster than with Cartesian coordinates. The only cases in which this set of elements does not outperform the classic representation with Cartesian coordinates are those with highly elliptical and those with highly perturbed orbits. As with MEE, this system representation is free of singularities, but it has a higher numerical stability and is easier to visualise.

3.3. Equations of Motion

The dynamics of different objects in space will be studied in this project in the context of orbit propagation and orbit prediction. To model the movement of these objects, the governing equations of motion are needed and thus will be presented in this section. It is to be noted that, for this project, the rotational motion will not be studied and therefore it will not be included. Only the translational motion of RSOs will be analysed and modelled, and the set of equations needed for that will be introduced.

The equations of motion are derived from Newton's laws of motion. In general, it can be established that:

$$\boldsymbol{a} = \frac{d^2\boldsymbol{r}}{dt^2} = \ddot{\boldsymbol{r}} \quad \rightarrow \quad \ddot{\boldsymbol{r}} = \frac{\sum \boldsymbol{F}}{m} \tag{3.20}$$

This holds under the assumption that the mass of the body does not change throughout the propagation of the orbit. This is a valid assumption for the purposes of the project.

In the case of a Keplerian orbit, where only the gravitational acceleration from the central body is considered, this formulation becomes:

$$\ddot{\boldsymbol{r}}_{kep} = -\frac{\mu}{r^3}\boldsymbol{r} \tag{3.21}$$

In this equation, μ represents the gravitational parameter of the central body. Since the dynamics of the objects orbiting the Earth are noticeably more complex, the perturbation forces need to be added to the model. A generic formulation of this is:

$$\ddot{\boldsymbol{r}} = \ddot{\boldsymbol{r}}_{kep} + \frac{\sum \boldsymbol{F}_{pert}}{m} = \ddot{\boldsymbol{r}}_{kep} + \boldsymbol{f}_{pert} \tag{3.22}$$

The term $\sum \boldsymbol{F}_{pert}$ refers to the sum of all perturbation forces that are acting on the body, and $\boldsymbol{f}_{pert}$ represents the total perturbing acceleration. All the individual perturbing forces considered to be relevant in this project will be described in Section 3.4.

The equations of motion of a certain object, represented in Cartesian coordinates and in an inertial reference frame, are therefore:

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{r} \\ \dot{\boldsymbol{r}} \end{pmatrix}, \quad \dot{\boldsymbol{x}} = \begin{pmatrix} \dot{\boldsymbol{r}} \\ \ddot{\boldsymbol{r}} \end{pmatrix} \quad \rightarrow \quad \frac{d\boldsymbol{x}}{dt} = \begin{pmatrix} v_x \\ v_y \\ v_z \\ -\frac{\mu}{r^3}x + f_{pert,x} \\ -\frac{\mu}{r^3}y + f_{pert,y} \\ -\frac{\mu}{r^3}z + f_{pert,z} \end{pmatrix} \tag{3.23}$$

3.4. Perturbations

In Section 3.3, the equations of motion were introduced. In a scenario with no perturbations, the only elements that are considered are the RSO considered and the central body gravitational acceleration, as reflected in Equation (3.21). This is not enough to model the simulation environment, which is incredibly complex, and perturbations need to be added to the problem to simulate the dynamics of the objects more accurately. A generic form of the problem with perturbations was shown in Equation (3.22), and the individual perturbing accelerations that are considered to affect a spacecraft orbiting the Earth will be introduced in this section.

It has been mentioned repeatedly that the highly accurate model to compare the software developed in this work is that of Leon Dasi (2021). As a consequence, the most accurate models that will be incorporated for study are those established in her work. These are the spherical harmonics gravity, discussed in Subsection 3.4.1, the perturbations from the gravity from third bodies, in Subsection 3.4.2, and the aerodynamic acceleration, shown in Subsection 3.4.3. The only exception to this is the incorporation of the Solar Radiation Pressure acceleration, which was discussed in her work but not implemented. Figure 3.4 shows that this is actually a relevant perturbation to take into account for satellites in all orbital altitudes, and thus it will be presented in Subsection 3.4.4. In general, it is common practice to include all accelerations with a magnitude larger than 10^{-7} $\mathrm{m/s^2}$ (Leon Dasi, 2021; Peña, 2023) for accurate modelling. Figure 3.4 shows a graph with a list of perturbing accelerations and typical magnitudes over a range of altitudes. The different perturbations that will be considered with their respective magnitudes are shown in Table 3.1. Different papers reviewed in the literature also showed similar models used in their methods (Peng and Bai, 2018c; Sanchez and Vasile, 2021) and thus these are considered enough to obtain an accurate prediction using AI-based models.

Decisions will need to be made on which models will be included in the simplified models fed into the machine learning algorithm. These will be discussed in detail in Section 3.6, as this section is only meant to introduce and formulate the perturbation forces that are needed for the project itself.

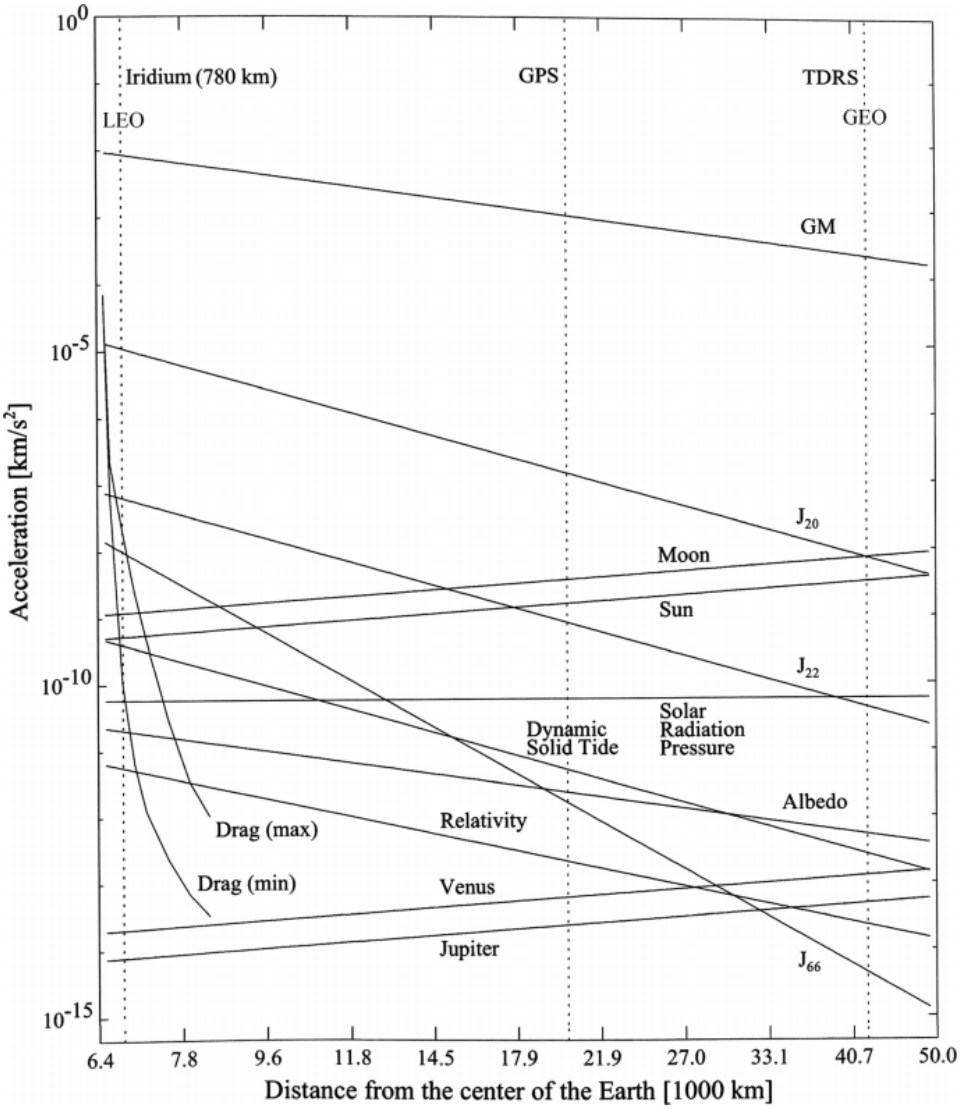


Figure 3.4: Perturbation acceleration magnitude, normalised with respect to the acceleration caused by the point-like central body gravity, plotted against orbital altitude of an arbitrary spacecraft (Montenbruck and Gill, 2000)

Table 3.1: Typical magnitudes of different perturbing accelerations

Perturbing acceleration	**Magnitude** $[\mathrm{m/s^2}]$
Earth's Gravity Field	10^1
Earth's Spherical Harmonics (6, 6)	10^{-6} - 10^{-5}
Third Body Perturbation Sun, Moon	10^{-6} - 10^{-5}
Drag	10^{-2} - 10^{-9}
Solar Radiation Pressure	10^{-7}

3.4.1. Spherical Harmonics

The point-like gravity of celestial bodies was formulated in Section 3.3. This assumes that the central body acts as a point mass, however, this is not an accurate representation of the gravity field of real objects. In reality, irregularities in the surface and density distribution of these bodies will induce anomalies in their gravity field which, in turn, will affect the motion of the satellites.

A more accurate and simplified formulation of the gravitational acceleration of a central body B, acting on a body A, in an inertial frame, can be represented as follows:

$$\boldsymbol{a}_{BA} = \nabla U_B(\boldsymbol{r}_{BA}) \tag{3.24}$$

In this formulation, U_B refers to the gravitational potential. The term $\boldsymbol{a}_{BA}$ represents the acceleration that body B exerts on body A. For a point mass acceleration, this takes the form:

$$U_B(\boldsymbol{r}_{BA}) = \frac{\mu}{||r_{BA}||} \quad \rightarrow \quad \boldsymbol{a}_{BA} = -\frac{\mu}{||r_{BA}||^3}\boldsymbol{r}_{BA}$$

This results in the same formula as shown in Equation (3.21), only with a different notation. When the irregularities in the gravity field of an object are taken into account, the formulation for the gravitational potential changes. The most common representation is via spherical harmonics, as the celestial bodies that will be considered here have a shape close to a spheroid.

For an extended body, the gravitational potential then takes the form:

$$U_B(\boldsymbol{r}_{BA}) = \frac{\mu}{r}\sum_{l=0}^{\infty}\sum_{m=0}^{\infty}\left(\frac{R}{r}\right)^l \bar{P}_{lm}(\sin\phi)[\bar{C}_{lm}\cos(m\theta) + \bar{S}_{lm}\sin(m\theta)] \tag{3.25}$$

where r is the module of $\boldsymbol{r}_{BA}$, R is the radius of the central body, l and m represent the degree and order of the model, ϕ and θ represent the latitude and longitude, respectively, of body A as seen from B, $\bar{P}_{lm}$ are the Legendre polynomials, and $\bar{C}_{lm}$, $\bar{S}_{lm}$ are the spherical harmonics coefficients. The maximum degree and order of the spherical harmonics will reflect the accuracy of the model.

Equation (3.25) includes the point-mass gravity in the formulation (it is the case in which both degree and order are 0). Since the interest of this section is to model the perturbations, this is to be excluded. Therefore, the formulation of the gravity potential only for the spherical harmonics perturbation becomes:

$$U_B(\boldsymbol{r}_{BA}) = \frac{\mu}{r}\sum_{l=2}^{l_{max}}\sum_{m=2}^{m_{max}}\left(\frac{R}{r}\right)^l \bar{P}_{lm}(\sin\phi)[\bar{C}_{lm}\cos(m\theta) + \bar{S}_{lm}\sin(m\theta)] \tag{3.26}$$

This model of perturbation is necessary for the context of collision risk assessment since satellites and other RSOs orbiting the Earth are affected by the anomalies in the gravity field of the Earth. Equation (3.26) will therefore be considered in the simulation model with a maximum degree and order of 6 for the Earth, as that is the highest that was used in the algorithm developed by Leon Dasi (2021). It is to be noted, however, that highly accurate models of the Earth's atmosphere exist, with spherical harmonics of degree and order of 280. For this project, and due to the limitations of the implementation of this model in differential algebra, this will be kept to $l_{max}, m_{max} = 6$ for the DA-GMM algorithm. When building a more accurate model for verification purposes, this will need to be increased. For accurate modelling, the gravity model of the Earth should be considered with at least degree and order 18, following Figure 3.4. Using the same threshold as established here for the acceleration magnitude, (Peña, 2023) selects $l_{max}, m_{max} = 24$ and so this shall be included in more accurate models required. It is to be mentioned that the formulation included here is developed under the assumption that the mass of body A is negligible in comparison to that of body B. This holds for the current case since body A will refer in this context to a satellite or another Earth-orbiting orbit.

3.4.2. Third Body Perturbations

The gravitational acceleration of bodies other than the Earth also needs to be considered for accurate orbit propagation. These 'third bodies' have a gravity field that also affects the movement both of the spacecraft and the Earth, and therefore exert an acceleration that affects both. The general formulation of the gravitational acceleration provided by these is the same as established in Equation (3.24). Earlier, it was assumed that the centre of mass of the Earth could be assumed inertial due to the proximity of the satellites, which allows the expression to be applied in this context. However, this assumption no longer holds for bodies that are much further away, and thus the aforementioned expression needs to be changed to account for that.

Table 3.2: Variables in the Solar Radiation Pressure and typical values

W	c	C_R
$\in [1317 \quad 1408]$ W/m^2	299,796,458 m/s	$\in [1 \quad 2]$

Consider a third body (B) exerting an acceleration on an arbitrary body A, which in turn is orbiting a central body (C). This body B is also affecting the movement of body C, and hence the third body terms can be expressed as:

$$(\boldsymbol{a}_{BA})_C = \boldsymbol{a}_{BA} - \boldsymbol{a}_{BC} = \nabla U_B(\boldsymbol{r}_{BA}) - \nabla U_B(\boldsymbol{r}_{BC}) \tag{3.27}$$

The notation $(\boldsymbol{a}_{BA})_C$ indicates the acceleration that body B exerts on body A, with respect to the central body C (i.e., the origin of the propagation).

For the work developed here, body A is considered to be an arbitrary RSO and body C is the Earth. For a problem of this kind, the only third body perturbations considered are that of the Sun and the Moon (King-Hele, 1987). Considering the perturbing accelerations shown in Figure 3.4, and since only those with magnitudes larger than 10^{-7} m/s^2 are considered, the Sun and the Moon are indeed the only third-body perturbations that should be included in the model.

3.4.3. Aerodynamic Acceleration

The interaction between a body orbiting the Earth and the atmosphere will also generate a perturbing acceleration. This has a higher effect at the lower regions of LEO, as observed in Figure 3.4, due to the higher atmospheric density. There are several aerodynamics accelerations (due to drag, lift...), but in this project only the perturbation caused by atmospheric drag will be considered. The expression for that is:

$$\boldsymbol{a}_{aero} = -\left(\frac{\rho v_{air}^2 C_D S_{ref}}{2m}\right)\hat{\boldsymbol{v}}_{air} \tag{3.28}$$

where ρ represents the atmospheric density at the current body location, v_{air} is the object's velocity with respect to the atmosphere, S_{ref} refers to the reference area, C_D is the drag coefficient, and m corresponds to the mass of the vehicle.

The calculation of the effect of this perturbation depends on the value of the density at the location of the satellite, which requires an atmosphere model. The choice of this model is relevant to the work and, as such, will be discussed in Subsection 3.5.2. Some terms can also be grouped in the ballistic coefficient, K, such that $K = \frac{C_D S_{ref}}{m}$, which then only depends on the physical properties of the spacecraft.

3.4.4. Solar Radiation Pressure

It has been mentioned before that the Solar Radiation Pressure had yet to be added to the DA-GMM algorithm and therefore the formulation of it will be studied here. The photons emitted by the Sun have momentum, and which is transferred upon an encounter with another object, namely a satellite, and causing an acceleration. There are different models for this perturbation, amongst which the cannonball radiation pressure acceleration, which assumes a spherical satellite, is the most common:

$$\boldsymbol{a}_{SRP,full} = -\frac{W}{c}\left(\frac{C_R S_{ref}}{m}\right)\frac{\boldsymbol{r}_{\odot_s}}{||\boldsymbol{r}_{\odot_s}||} \tag{3.29}$$

where the variables have the same meaning as introduced earlier in the report (Section 2.3). It is thought important to mention that C_R is not the same as the reflectivity coefficient of a body (ε), but rather it is defined as 1 + ε. Table 3.2 shows typical values of the variables present in the above equation.

As previously mentioned, this perturbation model has not been implemented in the DA-GMM. This is because it can only deal with continuous functions, which is not the case for the SRP. This acceleration

is also dependent on the position of both the Earth and the Sun, since the value of this force is 0 when the Earth acts as an 'occulting' body and the photons of the Sun do not reach the satellite. This effect is known as 'eclipse' or 'umbra'. The SRP can be added to the DA-GMM if it is modelled as a continuous function, which can be done if the effect of the penumbra is added. A parameter is introduced to aid in the modelling of the influence of eclipses in the acceleration due to SRP: the *shadow function*, represented by ν. This function is designed to be equal to one when there are no objects blocking the light from the Sun, and zero when the satellite is in the umbra region. The new formulation of the SRP acceleration is therefore:

$$\boldsymbol{a}_{SRP} = -\nu \frac{W}{c} \left(\frac{C_R S_{ref}}{m} \right) \frac{\boldsymbol{r}_{\odot_s}}{||\boldsymbol{r}_{\odot_s}||} \tag{3.30}$$

where the shadow function has a value such that:

$$\begin{cases} \text{Full sunlight} & \nu = 1 \\ \text{Partial eclipse} & 0 < \nu < 1 \\ \text{Eclipse} & \nu = 0 \end{cases} \tag{3.31}$$

Part of the work developed in this project will therefore be the modelling of this perturbation force.

The cannonball model presents a spherical representation of an arbitrary satellite, a simplifying assumption, and thus uses a three-degree-of-freedom implementation. More accurate models present a six-degree-of-freedom implementation which takes into account a satellite's attitude and rotational motion. These are not considered since rotational motion is not to be included in the DA-GMM and because this would require a deeper knowledge of the satellite's shape and attitude. Studies have shown that the effect of the spherical simplification for orbit prediction is an error of about 40 m for a non-spherical, near-GEO satellite (where the SRP is more significant) in the across-track direction (where the error is shown to be more prominent) after two days of propagation (Lachut and Bennett, 2016). This would be critical for Precise Orbit Determination (POD) but is considered adequate for the purposes of this thesis.

3.5. Environmental Model

In the previous section, different acceleration models were discussed to be added to the simulation. The next step is therefore to define the models that will be chosen to simulate the environment of the RSOs. Some of the perturbation accelerations will depend on the physical models chosen for the simulation, therefore it is relevant to motivate this choice. With the perturbations considered in Section 3.4, the environment models that will be investigated are the Earth's gravity field in Subsection 3.5.1 and the atmospheric model in Subsection 3.5.2. Once again, this has already been motivated by Leon Dasi (2021), and these are only provided in this report for the motivation of the choice of simplified, less accurate simulation models that will be fed into the ML algorithm. It is to be mentioned that the choice of rotation and shape model was not considered to be relevant and so the simplest model for that was selected. On top of that, the ephemeris of the different bodies is obtained from an information system established by NASA called SPICE (Spacecraft, Planet, Instrument, C-matrix, Events).

3.5.1. Earth's Gravity Field

The most simple model to represent the gravitational effect of the Earth on a satellite orbiting it is the point-mass acceleration, represented in Equation (3.8). More complex models have been developed to represent more accurately the anomalies in the gravity field of the Earth. The one that will be used in this project is that of the spherical harmonics, introduced in Equation (3.26). As explained earlier, a model up to degree and order six will be used for the full dynamics model for the differential algebra algorithm. For more accurate models, it will be studied whether the degree and order need to be increased and up to which value.

From Equation (3.25), the normalised Legendre polynomials and cosine and sine coefficients are obtained through the following expressions:

$$\bar{P}_{l,m} = \left[(2-\delta_{m0})(2l+1)\frac{(l-m)!}{(l+m)!}\right]^{\frac{1}{2}} P_{l,m} \tag{3.32}$$

$$\begin{Bmatrix}\bar{C}_{l,m}\\ \bar{S}_{l,m}\end{Bmatrix} = \left[\frac{1}{(2-\delta_{m0})(2l+1)}\frac{(l+m)!}{(l-m)!}\right]^{\frac{1}{2}} \begin{Bmatrix}C_{l,m}\\ S_{l,m}\end{Bmatrix} \tag{3.33}$$

where δ_{m0} is the Kronecker delta and follows:

$$\delta_{m0} = \begin{cases} 0 & \text{if} \quad m = 0 \\ 1 & \text{if} \quad m \neq 0 \end{cases} \tag{3.34}$$

All these expressions are obtained from NASA (n.d.), and an extensive list of values for the coefficients and Legendre polynomials for different degrees and orders is provided in the same paper.

3.5.2. Atmosphere Model

Atmospheric properties play an important role in the calculation of the different perturbation accelerations, and are particularly relevant for the drag acceleration. Several models have been developed throughout the decades, and the level of accuracy is different for each. However, more complex atmospheric models are also more computationally expensive, and so a trade-off must be made depending on the requirements of each mission or project.

Simpler models include the exponential atmosphere model and the United States Standard Atmosphere. The former is based on the assumption that the gas which comprises the atmosphere behaves as an ideal gas, and further assumptions include constant molecular weight and temperature with respect to altitude. The US Standard Atmosphere model from 1976 differs from the exponential atmosphere in the modelling of the temperature for different layers of the atmosphere, i.e., based on altitude. Since the atmospheric density is an important parameter in the calculation of the aerodynamic acceleration, and this is obtained from these models, the assumptions made introduce errors in the model.

More complex methods have been developed to model the atmosphere more accurately, such as the US Naval Research Laboratory Mass Spectrometer and Incoherent Scatter Radar to Exosphere, released in the year 2000 (NRLMSISE-00). This model takes into account the variations in atmospheric parameters due to the time of day, location and solar activity, amongst others. In adding these variations, the model turns more accurate, yet more complex since it requires more parameters to predict an output. Nonetheless, it is the one chosen for the full dynamics model, since the uncertainty in the atmospheric density and ballistic coefficient are the ones that have a larger effect on the problem.

3.6. Summary of Simulation Model

Throughout this chapter, the different perturbation and environment models have been introduced and the choices for the full dynamics model have also been presented. The question now is what models will be introduced in the simplified models for comparison, and the motivation behind this.

When choosing the simplified model, there are several factors to take into account:

- The knowledge of the object that is required to model the force
- The availability of this knowledge
- The effect of the force model on the object's orbit
- The uncertainties that need to be added due to the force model

Since the availability of the properties of different satellites is one of the main issues in the field of collision assessment as many companies do not release data, one of the most important simplifications

Table 3.3: Summary of the perturbation and environment models chosen for the full and simplified models

Full model	
Perturbations	**Environment**
- Spherical harmonics: Earth (degree and order 6) - Third Body Perturbations: Sun, Moon - Aerodynamic Acceleration - **Solar Radiation Pressure**	- Spherical harmonics: Earth (degree and order 6) - Point mass acceleration: Sun, Moon - Simple rotation and shape model - NRLMSISE-00 atmosphere model
Simplified model	
Perturbations	**Environment**
- Spherical harmonics: Earth (degree and order 5) - Third Body Perturbations: Sun, Moon	- Spherical harmonics: Earth (degree and order 5) - Point mass acceleration: Sun, Moon - Simple rotation and shape model

that could be done in the force model is the reduction of object knowledge needed for the simulation. If one also takes into account the uncertainties that are added to the model due to imperfect knowledge of these properties, the forces that could be simplified or assumed by the algorithm are the aerodynamic acceleration and the solar radiation pressure acceleration. On top of that, Figure 3.4 shows that these two do not have such a large effect on most of the altitude range considered, and so the algorithm will be tested without explicit modelling of these perturbations. Also following the figure mentioned, the spherical harmonics will be modelled with order and degree 5, instead of 6, to assess how that affects the results obtained. The third body perturbations and other spherical harmonics can be modelled accurately enough due to accurate ephemeris and coefficient calculation, and they are the most important perturbations acting on the objects so they will remain in the simplified model. A summary of the models selected is shown in Table 3.3.

4

Machine Learning

Before moving to the implementation of the algorithm for the task at hand, a general view of machine learning will be given in this chapter. The methods reviewed in the literature have been introduced in previous chapters, but this one aims to provide a deeper understanding of relevant topics in the field of machine learning and deep learning that will aid in the implementation later. With this in mind, a generic background on machine learning is given in Section 4.1. Neural networks, alongside their typical architectures and basic concepts, will be introduced in Section 4.2. Following this, a physics-informes approach for the algorithm development is further explained in Section 4.3, and finally Section 4.4 will review some important machine learning concepts that will be useful later in this work.

4.1. Machine Learning Background

The field of machine learning is very extensive, and, throughout the years, many different models and architectures have been developed. In Chapter 2, a neural network model has been selected as the most suitable to tackle the problem at hand. However, there are still different ways of classifying machine learning algorithms, even in the field of deep learning. The most popular type of classification is based on the learning style, by which one can find supervised learning, unsupervised learning, and reinforcement learning. The definitions presented are obtained from Prince (2024) and Zhang et al. (2023).

Supervised learning encompasses the development of algorithms that are trained with labelled data to understand the relationship between the input variables and the output. These methods are best suited to complex problems with tasks involving the prediction of outcomes or pattern recognition. In turn, there are two main trends of supervised learning: classification algorithms and regression algorithms. The former is self-explanatory: it is used to group data into different categories or classes depending on the input data and the trends observed. The latter, on the other hand, is used to predict values (i.e., when the output is quantitative).

Unsupervised learning is a type of machine learning that is provided with unlabelled data and is allowed to discover patterns without any prior training or explicit instructions (i.e., without a 'teacher'). These algorithms learn from raw input data to identify patterns, similarities, and differences. Unsupervised learning algorithms can be classified into clustering algorithms and dimensionality reduction algorithms. Clustering models explore the data and separate it in groups or trends based on similarities between data points. Dimensionality reduction refers to the methods that are used to learn the important features of the dataset, and therefore extract the ones that are identified as non-relevant.

In a different category, reinforcement learning learns from neither labelled nor unlabelled data. Rather, it focuses on trial-and-error learning, with feedback from its actions, to learn the optimal behaviour or path in a specific environment or situation. Unlike supervised learning, reinforcement learning has no training dataset and no 'true' outcome, so instead it has to learn from experience.

While unsupervised learning could be a useful feature for the topic of orbit determination and prediction due to its ability to reduce the features of the model and to identify outliers and anomalies, it will not be considered. It is also trivial to say that reinforcement learning techniques have large computational loads, and in a topic such as the one presented in this project with an increased level of complexity, it is considered impractical. The aim of using a machine learning algorithm for this project is to 'teach' it to recognise an unmodelled relationship between a set of labelled input and output data quantitatively, only regression-supervised learning algorithms will be studied for this. On top of that, literature studies have only considered models that fall into that category, further supporting the claim made.

4.2. Neural Networks

It has been shown in previous sections that the implementation of machine learning in the field of SST has been increasing over the years. More concretely, the use of deep learning techniques and neural networks has spread over a variety of fields due to their heightened performance over shallower algorithms. Appendix B presents an extensive list of methods that have been used in the literature for orbit prediction and collision avoidance as reviewed by Choumos et al. (2024) in Table B.1, where it is clear that deep learning algorithms have become popular in the last years. Within this list, several sources (Jadala et al., 2022; Peng and Bai, 2018a; Sanchez and Vasile, 2021) have demonstrated that neural networks can predict the state of a satellite with a greater accuracy than classic machine learning algorithms. Typically, deep learning algorithms in the context of collision assessment are used to predict the state of a satellite at future epochs and avoid the use of highly accurate propagators, thus reducing the time required to acquire the predicted position of an object in space. Ultimately, what needs to be obtained is the probability of collision between two satellites, but that is extremely difficult to assess directly via ML since the data for that is quite scarce. It is therefore a more common approach to use ML and DL for orbit prediction.

Different deep learning algorithms have been introduced previously, based on their use in the literature. This section aims to expand on the basics of neural networks and basic elements of their architecture for further understanding of the model to be developed in this project.

Neural networks have shown great results in large-scale problems with complex dynamics due to their flexibility and adaptability. The most simple algorithm is the artificial neural network, also called *multilayer perceptron* or feed-forward neural network. In its most basic form, a neural network is used to approximate the behaviour of a non-linear function, $\boldsymbol{f}$, with a general equation:

$$\boldsymbol{y} = \boldsymbol{f}[\boldsymbol{x}, \boldsymbol{\phi}] \tag{4.1}$$

This equation maps a multi-variable input $\boldsymbol{x} \in \mathbb{R}^m$ to a multi-dimensional output matrix $\boldsymbol{y} \in \mathbb{R}^n$ using $\boldsymbol{h} \in \mathbb{R}^d$ hidden units. When approximating the function via neural networks, the parameters of the model, $\boldsymbol{\phi}$, also play an important role in the outcome.

The main goal of neural networks, and what makes them different from numerical algorithms, is to have a model that is dependent on parameters, which need to be adjusted. To do so, firstly the training and test datasets need to be separated. The former is used to 'teach' the model the relationship between the inputs and outputs, and choose the appropriate parameters to fit the data. Once the model has been trained, and the parameters that map the underlying patterns of the problem have been adjusted, the performance of the model needs to be assessed with the test data.

For now, the focus of the explanations will be on training the algorithm and how it works. The process followed for this adjustment comes in two parts: the feed-forward propagation and the network training described in Subsection 4.2.1 and Subsection 4.2.2 respectively. To get a better grasp on how a basic neural network can be built, Subsection 4.2.3 provides examples of how the algorithms work.

4.2.1. Feed-forward Network Functions

Before moving forward to the explanation of the process, some concepts need to be introduced. The elementary units of deep learning algorithms are called *artificial neurons*, which are arranged into dif-

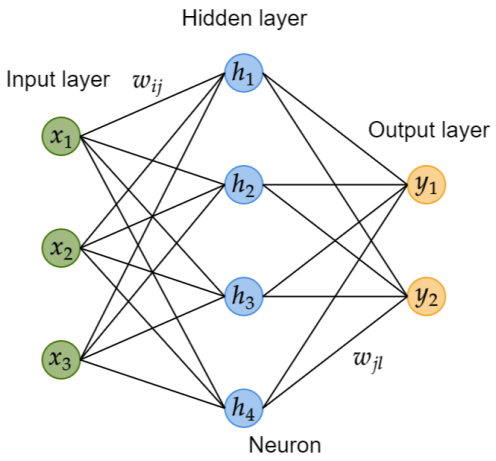


Figure 4.1: Example of a simple neural network, with three inputs and two outputs

ferent layers, as shown in Figure 4.1. The neural network therefore consists of the *input layer*, consisting of m inputs, an *output layer* with n neurons, and at least one *hidden layer*, formed by an arbitrary number of neurons. These layers are interconnected, meaning that all neurons are connected between layers, and each of these connections is assigned a *weight*.

In each of these hidden units, the input vector $\boldsymbol{x}$ is used alongside the parameters of that cell (i.e., the vector containing the weights of the inputs $\boldsymbol{w}_{ij}$ and the bias b) to compute the activation of the neuron j, z_j:

$$z_j = b_j + \sum_{i=1}^{m} w_{ij} x_i \tag{4.2}$$

This number is then used to compute the value of the hidden cell, h_j, via a non-linear *activation function*, represented by $a[\cdot]$. The weights and biases of the system are contained in $\boldsymbol{\phi}$, the parameters of the system. The prediction of the output of neuron l, labelled as $\hat{y}_l$, is the result of the weighted summation of these hidden unit values (Equation (4.4)). The weights used to obtain the output are also parameters of $\boldsymbol{\phi}$. The equations are all taken from Prince (2024).

$$h_j = a[z_j] \tag{4.3}$$

$$\hat{y}_l = b_l + \sum_{j=1}^{d} w_{jl} h_j \tag{4.4}$$

4.2.2. Network Training

The main difference between AI-based algorithms and more traditional approaches is the application of 'intelligence' in the models and the ability of these to 'learn'. The way this is done is via a process called *backpropagation*. In this process, the parameters of the model are adjusted so that the prediction obtained after the feed-forward propagation is as close as possible to the actual output.

In this context, the *loss function* $\mathcal{L}[\hat{\boldsymbol{y}}, \boldsymbol{y}]$ is introduced. Lower values of this function occur when the prediction of the model, $\hat{\boldsymbol{y}}$, is closer to the actual output $\boldsymbol{y}$, and thus it is desired. In the training process, the main aim is to find the parameter values that minimise the loss function and map the inputs to the outputs as accurately as possible. To model this function, the Mean Square Error (MSE) formula can be used:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(\hat{\boldsymbol{y}}^{(i)} - \boldsymbol{y}^{(i)})^2 \tag{4.5}$$

After the loss, or *cost function*, has been defined, the parameter optimisation can be formulated in a generic way such that:

$$\hat{\boldsymbol{\phi}} = \text{argmin}_{\phi}\left(\mathcal{L}[\hat{\boldsymbol{y}}, \boldsymbol{y}]\right) = \text{argmin}_{\phi}\left(\frac{1}{n}\sum_{i=1}^{n}(\hat{\boldsymbol{y}}^{(i)} - \boldsymbol{y}^{(i)})^2\right) \tag{4.6}$$

where $\hat{\boldsymbol{\phi}}$ is the vector containing the parameters of the model that minimise the cost function and are thus considered to be optimal. One could try solving $\frac{\partial \mathcal{L}}{\partial \phi} = 0$ to obtain the minimum of the function, however, this proves tricky since most problems are not convex and thus a solution to this cannot be found. A solution to this is to use an iterative approach, such as *gradient descent*:

1. Generate set of initial parameters $\boldsymbol{\phi}$.
2. Compute the derivative of the loss function with respect to the set of parameters.
3. Update the parameters such that at iteration t:

$$\hat{\boldsymbol{\phi}}_t = \hat{\boldsymbol{\phi}}_{t-1} - \alpha\frac{\partial \mathcal{L}}{\partial \phi} \tag{4.7}$$

   α in the above expression represents the learning rate.
4. Iterate until the function is minimised.

There are some issues regarding the use of gradient descent as an optimisation algorithm. Firstly, it might happen for more complex and non-linear dynamics that the algorithm converges to a local minimum instead of the global one, and hence an optimal solution has not been found. Secondly, the gradient descent calculates the gradient of the loss function for every parameter at every step, which can be computationally expensive. This is also sensitive to the choice of initial parameters chosen. As a result, several optimisation strategies have arisen to overcome these issues.

Optimisation Algorithms

It has been mentioned that the gradient descent can converge to local minima and that it is sensitive to the initial parameters chosen. This can be dealt with through the addition of noise into the gradient at each step, such that the direction in which the function moves is not necessarily the one that reduces the loss function the most. This is known as *Stochastic Gradient Descent* (SGD).

To include randomness in the process, the algorithm selects a random subset, known as *minibatch*, and computes the gradient from this batch only. Equation (4.7) then transforms to:

$$\hat{\boldsymbol{\phi}}_t = \hat{\boldsymbol{\phi}}_{t-1} - \alpha \cdot \sum_{i \in \mathcal{B}_t}\frac{\partial l_i[\phi_t]}{\partial \phi} \tag{4.8}$$

where $\mathcal{B}_t$ contains the indices of the elements that define the random batch, and l_i is the loss of the i^{th} element. The selection of random subsets is done without replacement, meaning that in every iteration a different subset is chosen until all the data has been used. Once this happens, batches are extracted from the full dataset again until the method converges or the maximum number of iterations has been reached.

One could also add the history of past gradients via the addition of the *momentum* term. With this modification, the information on the direction moved in previous steps is added to the parameter update:

$$\begin{aligned} \boldsymbol{m}_t &= \beta \cdot \boldsymbol{m}_{t-1} + (1-\beta) \sum_{i \in \mathcal{B}_t} \frac{\partial l_i[\phi_t]}{\partial \phi} \\ \boldsymbol{\phi}_t &= \boldsymbol{\phi}_{t-1} - \alpha \cdot \boldsymbol{m}_t \end{aligned} \tag{4.9}$$

where $\boldsymbol{m}_t$ is the momentum term and $\beta \in [0, 1)$ is the parameter that controls how much the gradient is smoothed over time. Since the momentum term is calculated at each iteration, the information on the gradient at past iterations is included in this model. This, in turn, results in a smoother trajectory of the gradient descent, and a faster convergence.

There is a downside to all of these modifications explained here: they all use a fixed learning rate. The consequence of this is that the adjustments made for small gradients do not allow the algorithm to properly map this direction, whereas the adjustments for larger gradients might cause larger, undesired oscillations. To tackle this, the gradients can be normalised so that the same distance is travelled in each direction. A simple algorithm that can tackle this is described below:

$$\begin{aligned} \boldsymbol{m}_t &= \frac{\partial \mathcal{L}[\phi_t]}{\partial \phi} \\ \boldsymbol{v}_t &= \boldsymbol{v}_{t-1} + \left(\frac{\partial \mathcal{L}[\phi_t]}{\partial \phi} \right)^2 \end{aligned} \tag{4.10}$$

In this set of equations, $\boldsymbol{v}_t$ is used to normalise the gradient itself, as shown in the expression below:

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t-1} - \alpha \cdot \frac{\boldsymbol{m}_t}{\sqrt{\boldsymbol{v}_t + \epsilon}} \tag{4.11}$$

where $\epsilon > 0$ is a constant that is used to prevent possible singularities caused by the denominator of the fraction being 0 and typically takes a value of $10^{-6} - 10^{-8}$ (Zhang et al., 2023).

The *Adaptive moment estimation*, also known as *Adam* (Kingma and Ba, 2014), optimisation algorithm takes the three ideas presented earlier (the minibatch gradient descent, the addition of the momentum term, and the normalisation of the gradients) and combines them all together. The equations are as follows:

$$\begin{aligned} \boldsymbol{m}_t &= \beta_1 \cdot \boldsymbol{m}_{t-1} + (1-\beta_1) \sum_{i \in \mathcal{B}_t} \frac{\partial l_i[\phi_t]}{\partial \phi} \\ \boldsymbol{v}_t &= \beta_2 \cdot \boldsymbol{v}_{t-1} + (1-\beta_2) \sum_{i \in \mathcal{B}_t} \left(\frac{\partial l_i[\phi_t]}{\partial \phi} \right)^2 \end{aligned} \tag{4.12}$$

The variables β_1 and β_2 are the momentum coefficients, and typically have values of 0.9 and 0.999, respectively (Zhang et al., 2023). The gradient and squared gradient are computed with the addition of momentum in Equation (4.12), and then they are normalised to avoid small estimates:

$$\begin{aligned} \tilde{\boldsymbol{m}}_t &= \frac{\boldsymbol{m}_t}{1-\beta_1^t} \\ \tilde{\boldsymbol{v}}_t &= \frac{\boldsymbol{v}_t}{1-\beta_2^t} \end{aligned} \tag{4.13}$$

The update of the parameters is then performed using the equation below:

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t-1} - \alpha \cdot \frac{\tilde{\boldsymbol{m}}_t}{\sqrt{\tilde{\boldsymbol{v}}_t + \epsilon}} \tag{4.14}$$

The resulting algorithm can hence converge to the global minimum, is more efficient than the classic gradient descent algorithm, is less sensitive to the initialisation of parameters, and accelerates convergence due to the addition of the gradient history via the momentum term and the normalisation of gradients. This optimiser is common in the papers reviewed regarding orbit prediction and is recommended for use with PINNs (Cuomo et al., 2022; Kollmannsberger et al., 2021).

4.2.3. Learning Algorithm

The previous subsections have explained all the concepts that are needed in order to build and train a neural network. Now the question is: *how can one make a simple Feed-Forward Neural Network with this information?*. Let us begin with a summary of the elements required for a neural network for supervised regression tasks:

- A dataset consisting of the inputs to the model $\boldsymbol{X}$ and corresponding outputs $\boldsymbol{y}$, separated into training and test sets with a validation set if required.
- An input, output and hidden layers. The number of layers and weights of their connections are also elements of the algorithm architecture.
- An activation function.
- The loss function.
- The gradients of the loss function calculated with respect to the model parameters. This can be done via backpropagation.
- An optimiser.

These are the basic elements of any deep learning model, although more can be added for more complex neural networks, such as Convolutional or Recurrent Neural Networks. Algorithm 1 describes an Artificial Neural Network, the most basic form of DL algorithm, with a full-batch gradient descent optimiser, such as formulated in Equation (4.7).

Algorithm 1 Algorithm to train a full-batch gradient descent neural network

Require: training data $\boldsymbol{X}$, targets $\boldsymbol{y}$
  Set NN architecture: input, output and hidden layer, activation function, learning rate
  Randomly initialise weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$, which together form the model parameters $\boldsymbol{\phi}$
  for all epochs t **do**
    for element $i \leftarrow 0, k$ **do**
      Apply Feed-Forward Propagation: $\hat{y}_i \leftarrow f_{NN}(\boldsymbol{x_i}; \boldsymbol{\phi})$
      Compute loss: $\mathcal{L}_i \leftarrow (\hat{\boldsymbol{y}}^{(i)} - \boldsymbol{y}^{(i)})^2$
      Apply backpropagation: $\frac{\partial \mathcal{L}_i}{\partial \phi}$
    end for
    Compute full-batch loss: $\mathcal{L} \leftarrow \frac{1}{k} \sum_{i=1}^{k} (\hat{\boldsymbol{y}}^{(i)} - \boldsymbol{y}^{(i)})^2$
    Compute full-batch gradients: $\frac{\partial \mathcal{L}}{\partial \phi} \leftarrow \frac{1}{k} \sum_{i=1}^{k} \frac{\partial \mathcal{L}_i}{\partial \phi}$
    Update the model parameters: $\boldsymbol{\phi}_t \leftarrow \boldsymbol{\phi}_{t-1} - \alpha \cdot \frac{\partial \mathcal{L}}{\partial \phi}$
  end for

In the previous section, it was highlighted that algorithms with this optimiser tend to perform worse, and it was argued that mini-batch algorithms such as Adam are preferred for Orbit Prediction. The overall model for an ANN with the Adam optimiser is then detailed in Algorithm 2. The definitions of β_1, β_2, $\mathcal{B}_t$, and ϵ are as introduced earlier.

This subsection provides a basic guide on how each of the described elements play a part in a NN training algorithm. However, for more complex problems such as Orbit Prediction, there are more elements that can be added to the model. The following section will describe how physics can be added to the algorithm to form Physics-Informed Neural Networks, briefly introduced in Section 2.4. On top of that, performance assessment techniques can also be added to the training process of the algorithm, which will also be discussed in further sections.

Algorithm 2 Algorithm to train a neural network with the Adam optimiser

Require: training data $\boldsymbol{X}$, targets $\boldsymbol{y}$
  Set NN architecture: input, output and hidden layer, activation function, learning rate
  Randomly initialise model parameters $\boldsymbol{\phi}$
  for all epochs t **do**
    Divide $\boldsymbol{X}$ and $\boldsymbol{y}$ into n batches of size k
    for all batches **do**
      for element $i \leftarrow 0, k$ **do**
        Apply Feed-Forward Propagation: $\hat{y}_i \leftarrow f_{NN}(\boldsymbol{x_i}; \boldsymbol{\phi})$
        Compute loss: $\mathcal{L}_i \leftarrow (\hat{\boldsymbol{y}}^{(i)} - \boldsymbol{y}^{(i)})^2$
        Apply backpropagation: $\frac{\partial l_i[\phi_t]}{\partial \phi}$
      end for
      Compute batch loss: $\mathcal{L} \leftarrow \frac{1}{n}\sum_{i=1}^{n}(\hat{\boldsymbol{y}}^{(i)} - \boldsymbol{y}^{(i)})^2$
      Compute batch gradients: $\frac{\partial \mathcal{L}}{\partial \phi}$
      Compute momentum term: $\boldsymbol{m}_t = \beta_1 \cdot \boldsymbol{m}_{t-1} + (1-\beta_1)\sum_{i\in\mathcal{B}_t}\frac{\partial l_i[\phi_t]}{\partial \phi}$
      Compute the gradient normalisation: $\boldsymbol{v}_t = \beta_2 \cdot \boldsymbol{v}_{t-1} + (1-\beta_2)\sum_{i\in\mathcal{B}_t}\left(\frac{\partial l_i[\phi_t]}{\partial \phi}\right)^2$
      Normalise $\boldsymbol{m}_t$, $\boldsymbol{v}_t$: $\tilde{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1-\beta_1^t}$, $\tilde{\boldsymbol{v}}_t = \frac{\boldsymbol{v}_t}{1-\beta_2^t}$
      Update the model parameters: $\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t-1} - \alpha \cdot \frac{\tilde{\boldsymbol{m}}_t}{\sqrt{\tilde{\boldsymbol{v}}_t}+\epsilon}$
    end for
  end for

4.3. Physics-Informed Neural Networks

Physics-Informed Neural Networks (PINN) were developed as a result of the challenges faced when using machine learning in fields where the lower amount of data available makes more traditional methods fail, such as in engineering (Raissi et al., 2017a, 2017b). In such fields, however, there is a large amount of knowledge in the form of physical laws that can be added to the model in the form of Partial Differential Equations (PDEs) to account for the lack of data. This prior knowledge is added to the algorithm as constraints so that the predictions obtained by the model adhere to the principles of physics.

In the previous section, an overview of FFNNs was given, which is essential to understanding PINNs since they work under the same principles. The architecture of these can be described as that of an ANN, but with added terms in the loss function that place the physical constraints of the system. As such, the loss term will have two parts: the standard data loss and the physics loss. The former will be represented as $\mathcal{L}_{data}$ and has the form introduced in Equation (4.5). This is used in the classical way: to help the model predict better the outputs given in the training data. However, with PINNs, one additionally wants to check that the physical variables predicted by the model are as close as possible to the actual governing physics of the system. Hence, the physics loss, $\mathcal{L}_{phys}$, needs to be added to the loss equation. This will be discussed in more detail later.

The general form of the non-linear partial differential equations can be expressed as follows (Kollmannsberger et al., 2021; Raissi et al., 2019):

$$\frac{\partial u(x,t)}{\partial t} + \mathcal{N}[u(x,t);\gamma], \quad x \in \Omega, \quad t \in \mathcal{T} \tag{4.15}$$

In this formulation, $u(x,t)$ is the unknown solution which depends on the time $t \in \mathcal{T}$ and space variable $x \in \Omega$, where Ω represents a space in $\mathbb{R}^D$. The second term of the PDE, $\mathcal{N}[u(x,t);\gamma]$, is a non-linear differential operator where the vector containing the physics-related parameters of the model is represented by γ.

The process of this type of algorithm starts by obtaining a prediction of $u(x,t)$ by a feed-forward neural network, using a set of parameters $\boldsymbol{\phi}$ (using the notation in the previous section). The approximation developed can be described as:

$$\hat{u}(x,t;\boldsymbol{\phi}) \simeq u(x,t) \tag{4.16}$$

The second 'block' of the PINN is the *physics-informed* section, which takes the output of the neural network and uses it to obtain the loss term. The *physics informed neural network* $f(x,t)$ can be defined such that:

$$f := \frac{\partial u}{\partial t} + \mathcal{N}[u] \tag{4.17}$$

which complies with Equation (4.15). It is known that, if the neural network had predicted $u(x,t)$ with no error, the value of f would be 0. The *physics loss* can therefore be defined as:

$$\mathcal{L}_{phys} = \frac{1}{M}\sum_{i=1}^{M} |f(x_i,t_i)|^2 = \frac{1}{M}\sum_{i=1}^{M} \left| \frac{\partial}{\partial t}\hat{u}_\phi(x_i,t_i) + \mathcal{N}[\hat{u}_\phi(x_i,t_i)] \right|^2 \tag{4.18}$$

Hence, the network parameters ϕ that affect both $\hat{u}_\phi(x,t)$ and $f(x,t)$ can be learnt via the minimisation of the loss term:

$$\mathcal{L} = \frac{1}{N_{data}}\sum_{i=1}^{N_{data}} |\hat{u}_\phi(x_i,t_i) - u_i|^2 + \frac{1}{M}\sum_{i=1}^{M} |f(x_i,t_i)|^2 \tag{4.19}$$

There is a difference between the points selected for the data loss and the physics loss. The former represents the points in which ground truth data has been collected and the latter is used to enforce consistency with the physics laws that govern the system. Therefore, the set of points for both is not necessarily the same.

The computation of the derivatives of u with respect to the system's inputs is obtained using automatic differentiation (Baydin et al., 2018). Other methods can be used for this differentiation, but AD is the most common in the literature (Cuomo et al., 2022). This is an advantage since many programming languages have libraries that have this algorithm, which can compute these derivatives fast.

4.4. Machine Learning Performance Assessment

The performance of the ML algorithm depends on a variety of factors. To fully optimise the model to deliver the best results, some things need to be taken into account that affect the operation of the algorithm. First of all, the model has some parameters that are not learnt by the network, these are the hyperparameters. For optimum results of a machine learning algorithm, these parameters need to be tuned before the training process; this is explained in Subsection 4.4.1. The generalisation capabilities of a trained algorithm also need to be assessed, a concept which is discussed in Subsection 4.4.2. Finally, the regularisation techniques used to avoid overfitting of the model are introduced in Subsection 4.4.3.

4.4.1. Hyperparameter Tuning

All machine learning algorithms have a series of parameters that are inherent to the model, parameters that cannot be learnt by these. These are fundamental for the optimum performance of the algorithm, they need to be selected before the algorithm is trained, and proper analysis must be done to choose the best possible set of hyperparameters that provide the best results. The reasonable question to be made now is: *how can one determine the values for these hyperparameters?*

The first step of the choice of hyperparameters is to identify which ones need to be defined and assign some reasonable, initial values to them. After this is done, the bounds of each of these variables need to be defined and an optimisation method to find the best possible set of these needs to be selected. Typical strategies to do this are: grid search, in which a grid is specified and the performance of the model is assessed for all possible combinations, or random search, where the combinations are selected randomly within the established ranges. The latter is preferred for problems with higher-dimensional

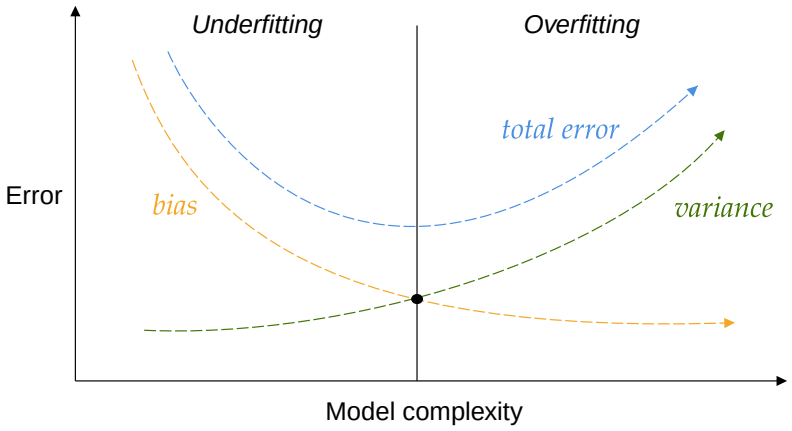


Figure 4.2: Visual representation of the bias-variance trade-off

data due to higher efficiency (Bergstra and Bengio, 2012), as it has a better chance of finding a suitable combination that maximises the performance of the model, especially for the problem considered here which is quite complex and requires a large amount of training data. Other optimisation methods could be considered, such as Bayesian optimisation or Evolutionary algorithms, although these are thought to be too time-consuming for the problem at hand and will thus not be considered. Once the method is set, the training data needs to be split. It can be divided into the data that will be fed into the model for training, and the data that is used for validation purposes. The latter aims to test the performance of the model during the hyperparameter tuning and find the best combination for good results. Once this is done, then the algorithm is tested with the validation set and the optimisation method is selected, to assess its performance for a series of combinations of hyperparameters. Once the optimum set has been chosen, the model can be trained with the training data.

4.4.2. Generalisation

Supervised learning algorithms aim to be able to *generalise* and be able to make accurate predictions on unseen data during training (i.e., test data). The ability of the model to generalise will depend on its complexity. Simpler models might fail to understand the underlying patterns of the data and thus are said to be *underfitting* the data, which is undesirable. However, highly complex models are prone to *overfitting* the data, memorising the data but losing its generalisation ability and poor prediction performance on test data.

Concerning this, the bias-variance trade-off can be introduced. The *bias* of a system results from wrong modelling assumptions and model simplification, and thus the bias is higher the simpler the model is. On the other hand, higher complexity leads to a larger sensitivity of the model to small variations in the data, leading to a high *variance*. A trade-off between bias and variance in the system should be performed to assess the adequate complexity of the model.

The validation set, introduced earlier, is also used for assessing the generalisation of the model in the training process and can be used to select the complexity of the model. By calculating the loss of the training and the validation datasets, the bias and variance can be obtained. A large training loss is typically related to a large bias and low model complexity. On the other hand, a large validation loss indicates that the model is overfitting the data. Finding a good balance between both training and validation losses is desired for good generalisation capacities. Figure 4.2 shows the evolution of bias and variance with model complexity, where a minimisation of the total error can be obtained from a proper trade-off between the accuracy of the model and its complexity.

4.4.3. Regularisation

Some techniques can be added to the model to prevent it from overfitting and forcing it to have a better generalisation. Since neural networks are flexible, they are also quite prone to overfitting the data and

hence regularisation is desired in the algorithm to prevent this from happening. Some methods have been developed for this purpose and will be introduced below.

L1 Regularisation

Also known as *Lasso*, this regularisation approach adds a penalty term on the loss function (Equation (4.5)) that is proportional to the sum of the absolute values of the weights. When performing the backpropagation, the model aims to reduce this regularisation term as well, hence the coefficients of the system are shrunk. Some of the weights in the vector ϕ become 0, and the important features of the model can be extracted. This reduces the complexity of the model while minimising the loss function, avoiding overfitting.

L2 Regularisation

The concept introduced here, also known as *Ridge*, is similar to the Lasso, with the difference that the penalty term introduced on the loss function is proportional to the square of the weights of the model. The coefficients now cannot become exactly 0, but rather they will have a very small value, so the magnitude of the weights can be effectively reduced without being set to 0. The overfitting of the data is prevented for similar reasons as for the Lasso.

Early stopping

This method is characterised by its simplicity. The performance of the model on the validation test is tested and the training process is stopped when the validation loss starts to increase. This typically indicates that the model is beginning to overfit the data, and the training is halted before the model starts to memorise the inherent noise in the training dataset.

Dropout

The dropout technique ignores a random feature in each iteration of the training process. As a consequence, the architecture of the network is different at each step of the process and forces some noise into the system. By adding this artificial noise, the chances of overfitting the data are reduced and the model increases its robustness, as it becomes less sensitive to small variations in the input data. This method has been proven to be efficient and is widely used in neural networks (Srivastava et al., 2014).

Summary

Both early stopping and dropout are desired techniques for regularisation, the former due to its simplicity and the latter due to its efficiency. Due to the increasing complexity of the model chosen, it has been decided that early stopping is an efficient enough method that can be used to prevent overfitting.

5

External Software Integration

There are two main parts in which the project can be divided: the extension of the DA-GMM and the development of a PINN-based model for orbit prediction and P_c calculation. The overview of the architecture and building blocks needed for both models in Section 5.1. The development and verification of these algorithms are aided by adequate external software. Section 5.2 gives an overview of the external tools and libraries that are required for this project. Each of these will need to undergo some acceptance and integration tests. The setup for the different tools and tests will be provided in Section 5.3, Section 5.4, and Section 5.5.

5.1. Software Architecture

Efficient modelling of the software to develop a robust and accurate algorithm can be achieved through the introduction of a preliminary architecture of the model. This helps with the organisation and identification of the key components required for the implementation phase of the project. The modifications in the architecture of the DA-GMM algorithm are discussed in Subsection 5.1.1, whereas Subsection 5.1.2 explains the main ideas behind the architecture of the PINN, expanding on the methodology presented in Section 2.8.

5.1.1. Differential Algebra Gaussian Mixture Model Modified Architecture

Since the DA-GMM is an already developed model that is to be extended, firstly the full architecture will be provided, and then a simplified one with the additions included will be presented. The structure of the code is as follows (Leon Dasi, 2021):

1. **GAUSSIAN MIXTURE MODEL SPLIT**
   This is performed in MATLAB and is the tool that generates the Gaussian Mixture Model for the uncertainty in the initial state of an object.
2. **TAYLOR SERIES INTEGRATION IN DIFFERENTIAL ALGEBRA**
   This is the differential algebra part of the code, which uses DACE to propagate the state and uncertainties. This is the core block of the simulation and where the acceleration due to the solar radiation pressure needs to be added.
3. **UNCERTAINTY PROPAGATION**
   This block, coded in MATLAB, receives the Taylor expansion of the final state for each GME and produces the GMM of the uncertainty of the final state.
4. **COLLISION PROBABILITY CALCULATION**
   This is the final block of the structure of the model, which uses MATLAB to calculate P_c from the GMM of the uncertainty distribution of the final state.

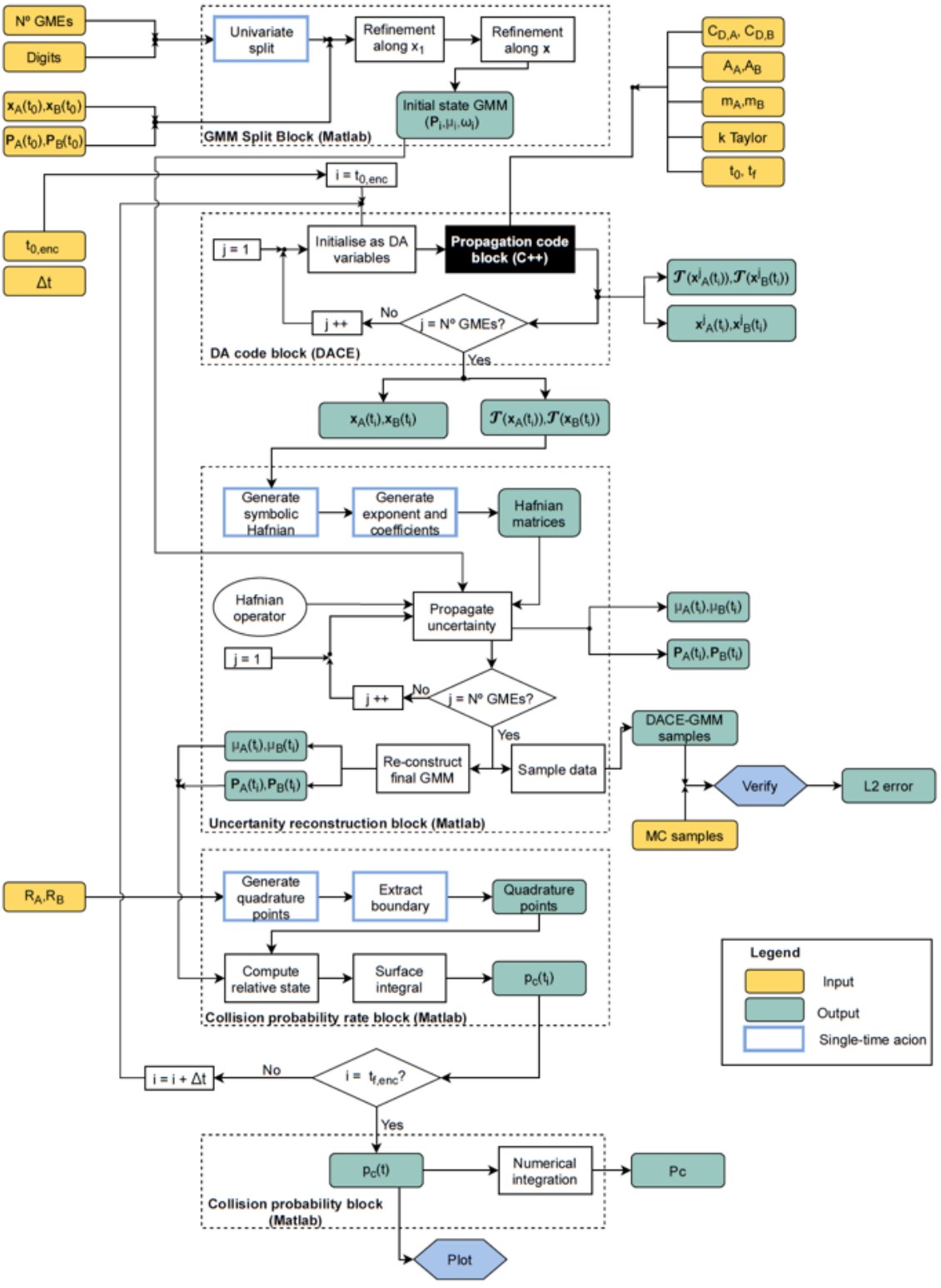


Figure 5.1: Full architecture of the DA-GMM algorithm (Leon Dasi, 2021)

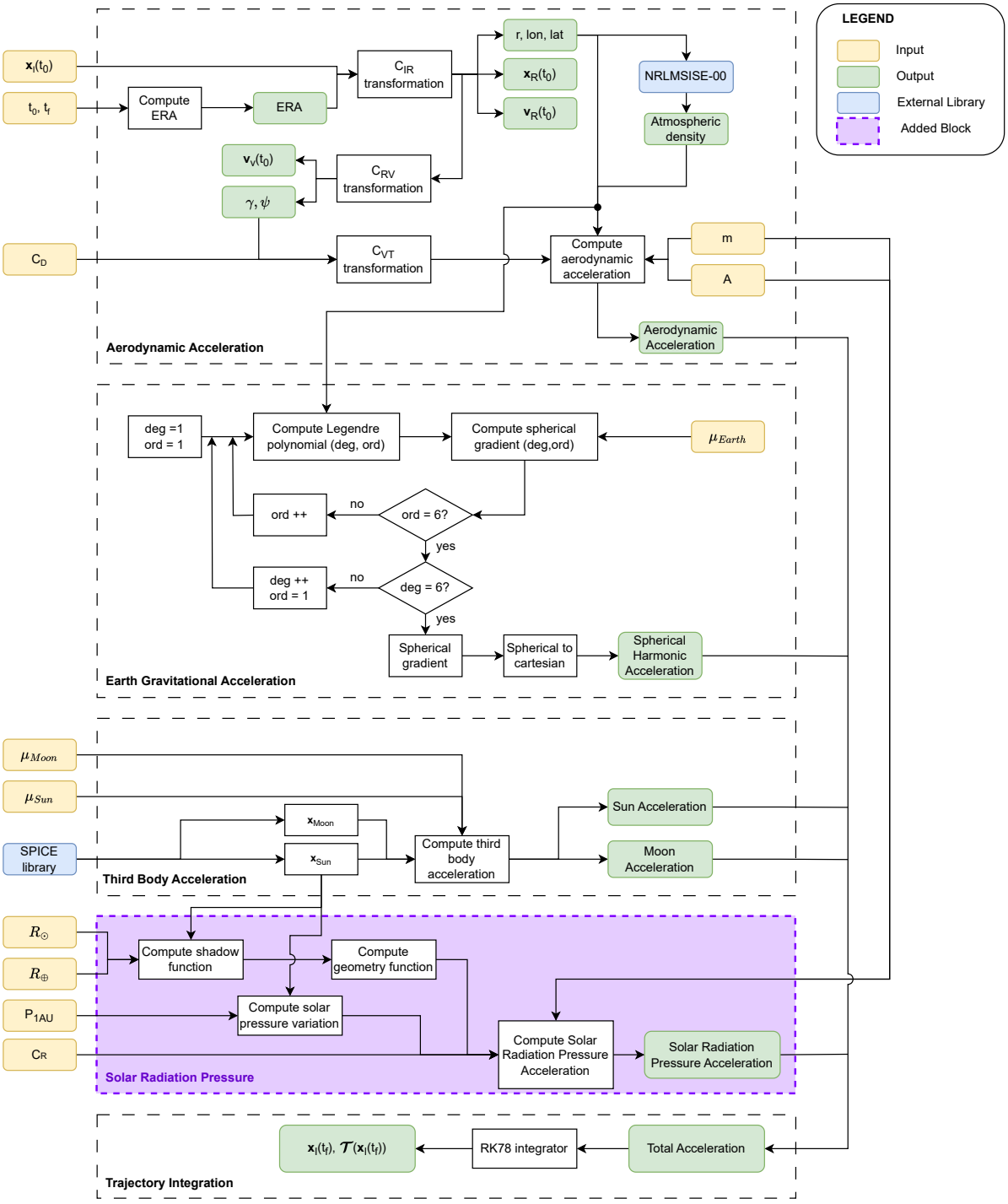


Figure 5.2: Software architecture for the propagator inside the DACE block (Leon Dasi, 2021), with the extension shown in purple

The overall structure of the algorithm, presented in Figure 5.1, does not need to be changed. There are only two main changes that need to be included in this structure: the formulation of the solar radiation pressure and its uncertainty. The former can easily be done through the addition of a function in the propagation code block, shown in black in Figure 5.1. The architecture of this block is obtained from Leon Dasi (2021), and the addition of the required inputs for the formulation of this acceleration and the corresponding output is shown in Figure 5.2. When it comes to the addition of the uncertainty of the $CSRP$, it can be added in the GMM split block consistently with the rest of the uncertainty models added.

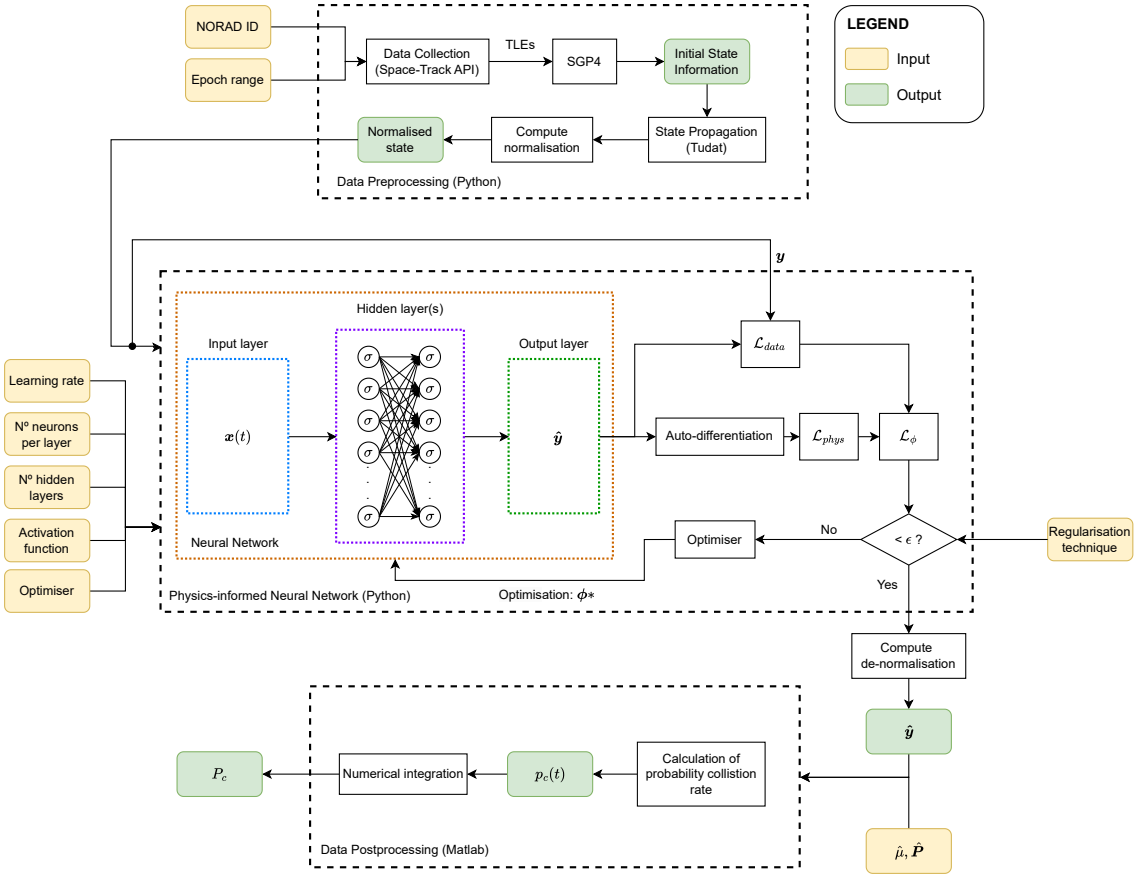


Figure 5.3: Software architecture for the PINN-based model for collision risk assessment

5.1.2. Physics-Informed Neural Network Architecture

This subection expands on the main blocks of the architecture of the PINN-based model, with the inputs and outputs of it. This is useful to get an overview of how the code works and how the different blocks of which it is composed are related to each other.

The PINN-based model, unlike the DA-GMM, needs to be developed from scratch, and the final architecture for it is presented in Figure 5.3. The whole model does not only consist of the PINN, it also requires some other pieces of code for the data pre- and post-processing.

The first block that needs to be included in the model is that of the data pre-processing. The data that will be used for the training of the neural network is the TLE of different satellites. For the sake of simplicity, the Space-Track API [1] for Python will be used so that the only input required is the NORAD ID of the desired satellite and epoch, and the code will automatically retrieve the data. Once the data is obtained, the raw TLE data is read and transformed into state information, which is then normalised and organised into the input-output pairs that will be fed into the PINN block. This block provides the neural network with the classical structure that has been introduced in Section 4.3. By feeding it with the normalised data from the data pre-processing block and the hyperparameters, the PINN will output a predicted state and covariance matrix. Before final use, the hyperparameters need to be tuned for optimal performance and the model trained. Once the model has learnt the underlying relationship between the inputs and outputs, it can be fed some set of initial states and uncertainties for different objects and predict their state and covariance matrix at future epochs. Once this has been obtained, the last part of the software is used to calculate the collision probability rate and, via numerical integration, the probability of collision itself. This is the final block presented, that of the data post-processing.

[1] Space-Track API documentation available at: https://www.space-track.org/documentation#/api. Accessed: 25/02/2024

5.2. External Software Overview

The use of external software can aid in the increase of efficiency of the modelling process, data pre- or post-processing and the verification of newly developed algorithms. On top of that, these can also be an excellent aid for AI-based methods as they allow for the simplification of complex tasks such as optimisation and parallel computing. The external libraries and tools that are needed for this project will be reviewed in this section and will be briefly introduced and explained.

To develop the AI-based model, the DA-GMM is used for comparison purposes and will be introduced as an external software in this section. The reasoning behind this is that it was developed by Leon Dasi and that the work performed here regarding this algorithm will just focus on its extension and its use for verification. In the extension of this model, external software also needs to be used to verify the new uncertainties results via Monte Carlo analyses, and the tool chosen for that is Tudat, useful for astrodynamics simulations. It can also be used for the verification of the PINN algorithm and data processing. Finally, a library is needed for the development of the PINN that can provide a more effective design modelling of the algorithm, and PyTorch is selected as the most intuitive tool to use for that purpose.

5.2.1. Differential Algebra and Gaussian Mixture Model

The Differential Algebra-Gaussian Mixture Model developed by Leon Dasi (2021) has been explained in detail in the previous chapters. The current functions and libraries used have been extensively verified and validated and so it will be considered that no further tests of this nature are needed. This model will be extended to account for the Solar Radiation Pressure acceleration and further verification and validation tests will be presented in Chapter 6. It will also be used for verification purposes of the AI-based model that is to be developed in this project for orbit prediction and collision probability calculations between two objects.

The DA-GMM itself has some dependencies in external libraries, which will also be briefly introduced and explained. The algorithm is developed in Matlab and C++, the former being used for the pre- and post-processing of data and the latter for uncertainty propagation. This propagation is performed using Differential Algebra with a series of libraries and tools discussed below:

DACE The Differential Algebra Computational Toolbox (DACE) [2], developed in C++, is a set of libraries useful for Differential Algebra. The user interface in C++ allows for an intuitive application and works with operations using Taylor series expansions.

NRLMSISE-00 As mentioned in Section 3.4, the atmosphere model that will be used for the computation of the drag acceleration is the NRLMSISE-00. This is included as a C++ library.

SPICE The information system SPICE, developed by NASA, is a toolkit widely used for space environment simulations. It provides accurate ephemeris for different celestial bodies and therefore will be used for the extraction of the position of the Moon and the Sun needed for the third-body perturbation acceleration calculation. This toolkit is also included as a C++ library.

Boost This is a set of C++ libraries [3] that is typically used to complement the standard libraries. It is introduced in the DA-GMM algorithm to perform linear algebra operations, pseudo-random number generator and unit testing.

Eigen This open-source library [4] is typically used for numerical linear algebra. It is used in this algorithm to perform operations with the SPICE and NRLMSISE-00 data.

The uncertainty matrix reconstruction and collision probability are both performed in Matlab.

[2]Dace algorithm available at: https://github.com/dacelib/dace/tree/master. Accessed: 01/03/2024
[3]Boost algorithm available at: https://www.boost.org/. Accessed: 01/03/2024
[4]Eigen algorithm available at: https://gitlab.com/libeigen/eigen/-/releases/3.4.0. Accessed: 01/03/2024

Table 5.1: Initial state for the satellites considered for the verification scenario (Alfano, 2009)

Initial Position	x [km]	y [km]	z [km]
Satellite A	6337.97	1889.31	1889.31
Satellite B	6338.67	1888.22	1888.15
Initial Velocity	**vx [km/s]**	**vy [km/s]**	**vz [km/s]**
Satellite A	-2.95720	4.96018	4.96018
Satellite B	-2.95530	4.96081	4.96062

5.2.2. Tudat

The TU Delft Astrodynamics Toolbox (Tudat) [5] is a set of libraries developed in C++ by members of the Aerospace Engineering faculty at TU Delft that are used in the context of astrodynamics and space research. They form a powerful tool for simulations, as they can be used for a wide variety of scenarios, including Earth Observation, and are very straightforward to use. There is an interface available for Python. Since this is the programming language that will be used for the main block of the PINN-based model, this interface is preferred over the C++ one. As Tudat has been extensively verified, it will be used for verification purposes of the DA-GMM and the PINN model.

5.2.3. PyTorch

Alongside *TensorFlow*, PyTorch [6] is the most popular library to use in the field of machine learning and neural networks, which is open-source and free to the general public. It offers several advantages, such as tensor computing with Graphics Processing Unit (GPU) support (which facilitates the use of GPUs to accelerate the calculations via parallel processing) and automatic differentiation, needed for PINNs, as discussed in Chapter 4. The Python interface of PyTorch is intuitive to use and has been widely verified, which proves its reliability. TensorFlow is also commonly used for the same purposes. Still, since PyTorch offers more flexibility, it is selected as the tool for modelling the neural network structure. Other projects which have focused on the development of neural networks, such as Fanizza (2023), have used PyTorch as a tool for that. The possibility of using this software as a tool to build the model is encouraged, especially since the formulation of PINNs can get complicated due to the modelling of the physics that is required. However, and when possible, the functions will be developed manually and verified with the results from this software, although PyTorch can also be used as a guide to build the desired model with ease.

5.3. DA-GMM Setup

Since the work performed on the DA-GMM is merely an extension of it, the original algorithm must be tested to assess its correct functionality. Due to the extensive verification and validation of the unmodified algorithm, only some acceptance and integration tests must be performed, which will be done in this section.

Due to the use of external libraries such as DACE, Boost, and Eigen, the integration of all the components of the model needs to be assessed. Once DACE is functioning correctly, the integration of this block with those developed in Matlab needs to be studied. To do so, a test study performed by Leon Dasi (2021) will be done again with the algorithm and the results compared to those presented in her work. The case study selected for this is that of an artificial scenario created by Alfano (2009), verified with a 10^9-sample Monte Carlo simulation. The initial states for both satellites are shown in Table 5.1, and their uncertainties in Table 5.2.

The time of closest approach is set to be 172,800 seconds after the epoch, and so the simulation is run from TCA - 1420 s to TCA + 1420 s for consistent results with both Leon Dasi and Alfano. The Taylor series expansion is set to be of order three, the number of GMEs is set to 51, and the time step is 10 s. All of these have been selected by Leon Dasi (2021) and will be kept as such for actual testing.

[5] Tudat documentation available at: https://docs.Tudat.space/en/latest/. Accessed: 27-02-2024
[6] PyTorch documentation available at: https://github.com/pytorch/pytorch. Accessed: 29-02-2024

Table 5.2: Uncertainties for the satellites considered for the verification scenario at epoch (Alfano, 2009)

Standard Deviation	Satellite A	Satellite B	Units
$\sigma_{x,x}$	3.40997e-04	3.41073e-04	$[\mathrm{km}^2]$
$\sigma_{y,y}$	2.34001e-04	2.33957e-04	
$\sigma_{z,z}$	2.34001e-04	2.33957e-04	
$\sigma_{x,y}$	9.89664e-05	9.89162e-05	
$\sigma_{x,z}$	9.89664e-05	9.89125e-05	
$\sigma_{y,z}$	-1.65999e-04	-1.66036e-04	
$\sigma_{vx,vx}$	8.50607e-11	8.50798e-11	$[\mathrm{km}^2/\mathrm{s}^2]$
$\sigma_{vy,vy}$	5.79696e-11	5.79585e-11	
$\sigma_{vz,vz}$	5.79696e-11	5.79166e-11	
$\sigma_{vx,vy}$	2.50580e-11	2.50453e-11	
$\sigma_{vx,vz}$	2.50580e-11	2.50443e-11	
$\sigma_{vy,vz}$	-4.20304e-11	-4.20399e-11	

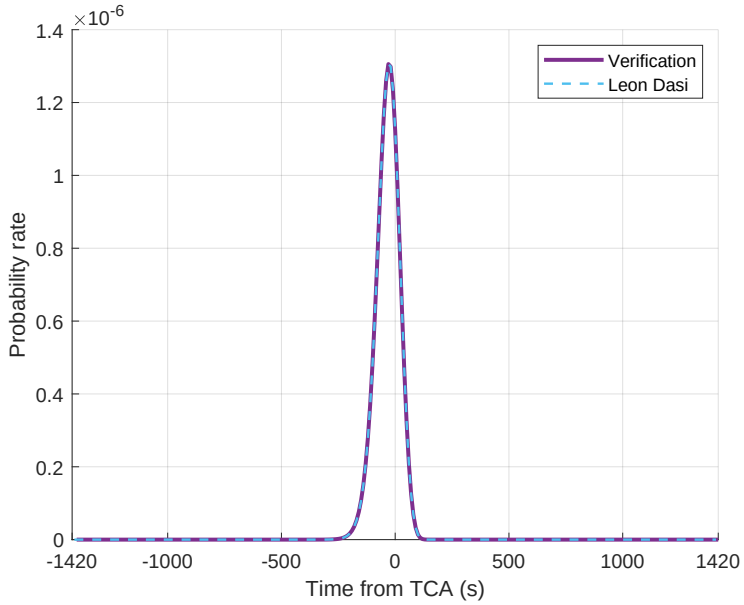


Figure 5.4: Comparison of the results obtained from the simulation performed for the verification of the DA-GMM for Alfano's Case 7 and those from the original work of Leon Dasi

On top of that, both satellites are modelled as hard-body spheres with a combined radius of 10m.

With all the parameters of the model established, the simulation is run to predict the risk of collision and compare it to the results obtained by Leon Dasi to verify the correct compilation of the code. By using the formulation for $p_c(t)$ presented in Equation (2.2), the collision probability rate is calculated. The results from this simulation are then plotted alongside those obtained by Leon Dasi for the same set of parameters, and are consequently shown in Figure 5.4.

From the figure, one can conclude that, using the same parameters as established in the work of Leon Dasi for a certain simulation, the compiled code for the DA-GMM is able to replicate the results correctly and thus its compilation is verified. The value obtained for total collision probability differs only in 0.008%, and the TCA is predicted with the same accuracy. This small discrepancy in results is considered to be due to differences in the hardware and to numerical precision and rounding, and not to any errors in the compilation.

Table 5.3: DA-GMM integration and compilation verification tests

ID	Test	True Value	Pass/Fail
DAGMM-IT-01	Test satellite covariance results	Results from Leon Dasi (2021)	Pass
DAGMM-IT-02	Test satellite position results		Pass
DAGMM-IT-03	Test P_c accuracy		Pass
DAGMM-IT-04	Test time of encounter accuracy		Pass

Table 5.4: Tudat environment setup test

ID	Test	Desired output	Pass/Fail
TU-ET-01	Test environment setup	Equals results from tutorials at Tudat website	Pass

5.4. TUDAT Setup

To properly assess the results obtained from the extended DA-GMM algorithm and those from the PINN, it is highly important to have an orbit propagator. There is a myriad of possibilities to choose from, but it was decided that Tudat was the tool to be used to construct this propagator. The main reasoning behind this is that this set of libraries has been extensively verified and validated and that it offers a highly intuitive interface for space dynamics simulations. However, some acceptance tests need to be performed on Tudat to ensure that its implementation is accurate enough for verification purposes. Since this toolkit has been developed by the Technical University of Delft (TU Delft), the author is well versed in its functioning and can also ask for assistance if it were needed. Table 5.4 includes the test that is performed to make sure that the algorithm is properly integrated and works as expected[7].

Now, several choices need to be made to develop an orbit propagator suitable for the mission considered. Firstly, the choice of environment and perturbation models included to simulate the environment needs to be discussed. Subsection 5.4.1 introduces the results from this analysis. Afterwards, several choices must be made regarding the integrator and propagator settings for the numerical integration and trajectory propagation. These are presented in Subsection 5.4.2. It is desired to include some simplifications in the modelling to avoid rounding errors in the computer and to simplify the dynamics to reduce the computational load required for the Monte Carlo analyses. A trade-off between both shall be made.

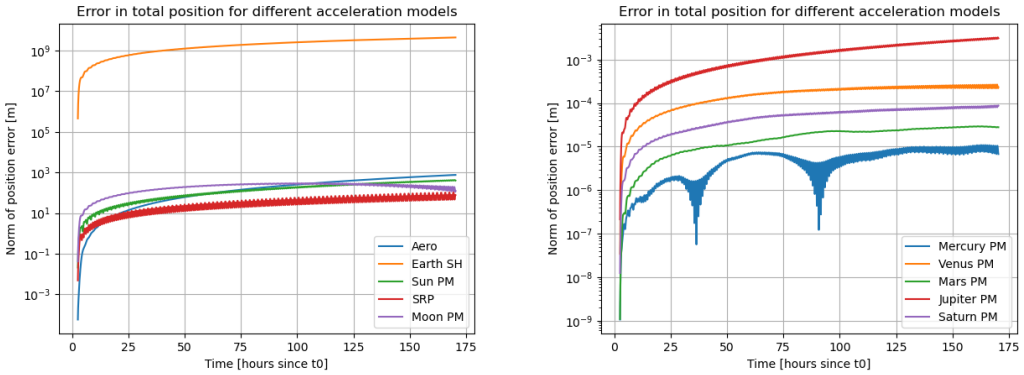
Firstly, a random object is generated. It has been created with a set of semi-random satellite properties: 750kg of weight, a 10 m^2 reference area, a drag coefficient of 2.2, and a reflectivity coefficient of 1.4. The orbit in which it lies is selected to have an altitude in a highly densely populated region in LEO since this is one of the targets considered. This altitude is chosen to be 800 km. The inclination is chosen as the most popular for RSOs, with a value of 60°. The eccentricity, argument of periapsis, RAAN and true anomaly are selected with a random number generator, ensuring that the eccentricity is kept low so that it remains in LEO throughout its orbit.

5.4.1. Perturbation and Environment models

The effect of different perturbation and environment models on a simulated orbit needs to be assessed to ensure the high accuracy of the propagator. To do so, an orbit is propagated for two days, as this is the time that is needed for the PINN verification.

Using a benchmark with a complex dynamics model, the effect of single acceleration models on the position error of the object is evaluated. This complex dynamics model includes the point mass gravity acceleration from all planets in the solar system, the solar radiation pressure from the Sun, the drag acceleration of the Earth and different degrees of spherical harmonics of the central body. It was decided

[7]Tutorials from Tudat available at: https://docs.tudat.space/en/latest/_src_getting_started/_src_examples/tudatpy-examples/propagation/perturbed_satellite_orbit.html. Accessed: 27/03/2024



(a) (b)

Figure 5.5: Norm of the position error resulting from removing certain perturbations from the simulation

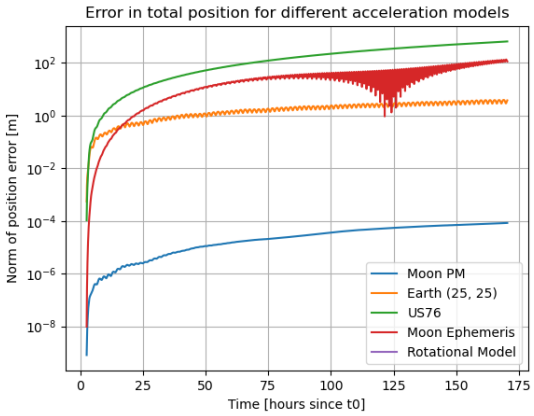


Figure 5.6: Norm of the position error resulting from the simplification of the simulation environment

that the error of the physical model should remain lower than 10^{-1} m for a two-day propagation. To achieve this, and as observed in Figure 5.5a and Figure 5.5b, the only gravity models considered are those from the Sun and the Moon as point mass and that of the Earth as a spherical harmonics model. On top of that, the solar radiation pressure acceleration and aerodynamics perturbation are considered. This agrees well with the analysis performed in Chapter 3.

When it comes to the environment models selected, the ephemeris of the different celestial bodies will be studied, alongside the choice of atmosphere model and gravity model of the Earth and Moon. Such as with the acceleration models, the simulation starts with a highly accurate benchmark, with the most accurate ephemeris, atmosphere and gravity models, and these are simplified in subsequent simulations and compared to the benchmark. The results from this are observed in Figure 5.6. It can be concluded that the ephemeris of the different celestial bodies needs to be obtained from SPICE, the most accurate model, and the atmosphere model selected is NRLMSISE-00 since lower accurate models indeed provide a larger error than desired in the solution. The gravity model of the Moon for the environment setup will be considered to be point mass, since as per Figure 5.6 the error induced in the position at the end of the propagation is low enough, and the spherical harmonics of the Earth are set to degree and order 25 (Peña, 2023).

5.4.2. Integrator and Propagator

With the object and perturbation accelerations defined, the choice of which integrator and propagator to use needs to be made. This decision is crucial for the development of highly accurate propagators as they are responsible for the stability of numerical simulations.

Integrator Settings

Once the perturbing accelerations affecting the spacecraft have been selected, the equations of motion can be defined. Sometimes, analytical solutions might be available to solve these, but generally, a numerical scheme is needed instead. Since the dynamics of the system as explained above is more complex, numerical methods are needed to solve the equations of motion governing the dynamics of the system.

Tudat offers a selection of integrators of different natures: multi-stage (with fixed or variable time-step), multi-step, or extrapolation methods. All of these are described and discussed in Appendix C. Extrapolation methods are extremely useful in the context of orbital dynamics for simulations with a long propagation time, since it allows for taking large time steps without much loss in accuracy. The results of these algorithms, as obtained now by Tudat, are quite sparse due to the large time steps, and integrators are not useful to obtain denser outputs with more continuous results. Since the project at hand requires a lower propagation time with a dense output, these methods will not be considered. Multi-step methods such as Adams-Bashford-Moulton (ABM) will also not be considered due to their incomplete implementation in Tudat (Dirkx, 2022), hence multi-stage integrators (with the Runge-Kutta methods available in Tudat) will be the ones considered for the integrator assessment.

Runge-Kutta integrators of different orders are included in Tudat and the choice of order will depend on the problem at hand. Overall, and out of the list of multi-stage integrators available in Tudat, the RKF7(8) (a Runge-Kutta-Fehlberg integrator of order 8 with an embedded 7th order) or the DOPRI8(7) (known as Dormand-Prince, of order 7 with an embedded 8th order) provide accurate, robust results, while keeping the computational load at a reasonable level with a high output density (Dirkx, 2022). This would make either of them suitable for the simulation desired, with a perturbed (but not overly complex) satellite dynamics, and where a dense output is preferrable. Out of those two options, the DOPRI8(7) is selected due to its use in other collision risk analysis problems, such as Valli et al. (2013) and Jones and Doostan (2013), where this integrator was used for verification purposes. On top of that, this is the default integrator for different ordinary differential equation (ODE) solvers such as in Matlab, Python or Julia. This integrator, as others in the RK family, is available in Tudat with fixed or variable step size. The latter is desirable for some aspects of orbital dynamics, where the dynamics of the system might be faster in some regions of the orbit than others. As for the absolute and relative tolerances for the method, a value of 10^{-12} is set, which agrees with the literature reviewed (Valli et al., 2013).

Propagator Settings

Propagators are used to model how the state of the satellite evolves through time. The choice of propagator is crucial for the accuracy of the simulation, and thus different propagation methods will be discussed and assessed. The numerical stability of a propagator results from the choice of representation of the state of an arbitrary object and the geometry of the orbits to be propagated. Different state vectors were already presented in Section 3.2, with their advantages and disadvantages introduced. The corresponding propagators that are available in Tudat are the following:

- Cowell
- Encke
- Gauss - Keplerian
- Gauss - Modified Equinoctial
- Unified State Model - Quaternions

Each of these propagators has a set of advantages and disadvantages. The Cowell propagator propagates the Cartesian state of the object forward in time and, albeit typically used, it induces large errors in the propagation due to large state derivatives. On the other hand, the Encke propagator, which also uses Cartesian coordinates, only propagates the difference between the state of the body and a reference Keplerian orbit. The latter is useful when perturbations are small. Since this is not the case, it will also not be considered. The Gauss-Keplerian propagator uses the Keplerian elements for the orbit propagation. As discussed earlier, this set of elements provides errors due to the presence of singularities and thus will also be discarded for this assessment. The propagators that are expected to provide

Table 5.5: Summary of the simulation parameters for the highly accurate propagator

Parameter	Settings
Acceleration Models	Spherical Harmonics Earth (25, 25) Point-mass gravity from the Sun and the Moon Drag acceleration from the Earth Solar Radiation Pressure
Environment Models	Spherical Harmonics from the Earth (25, 25) Ephemeris of the different celestial bodies from SPICE NRLMSISE-00 for the Earth's atmosphere
Integration Settings	Variable step size Dormand Prince 8(7) with absolute and relative tolerance equal to 10^{-12}
Propagator Settings	Unified State Model - Quaternions

the lowest error are the Gauss - Modified Equinoctial (which uses MEEs as a state representation) and the Unified State Model - Quaternions (which uses USM7 to describe the state of an object) due to their high numerical stability. Both propagators would be suitable for Tudat propagations, but the latter is chosen due to a higher accuracy over a wider range of scenarios (Römgens, 2011; Vittaldev, 2010).

5.4.3. Final Model

The final model selected for the highly accurate propagator built from Tudat follows a series of tests and analyses on the perturbation and environment models, alongside a suitable choice of integrator and propagator settings. The summary of the final settings of this propagator is shown in Table 5.5.

5.5. PyTorch Setup

PyTorch is the external library that is selected to aid in the development of the PINN-based model in Python. Like other software libraries, it regularly undergoes rigorous verification and validation tests to ensure its correct functionality, as well as identify possible bugs and issues. Nonetheless, it is deemed important, as is common practice, to perform some acceptance tests to assess the reliability, robustness and effectiveness of the library when it comes to machine learning algorithm development. On top of that, these tests will allow for the assessment of the integration of the functions of the library with the rest of the algorithm. The main aim of the acceptance and integration tests is therefore to assess the different functions and classes of the external library.

PyTorch uses a data structure called *tensors*. These are similar to arrays and matrices but can be run on GPUs, allowing for easy parallelisation, essential for the training of neural networks. The first test of the library will therefore be dedicated to assessing tensor operations, following easy operations that should equal the hand calculation of the same and some tutorials to assess the possible operations available with them [8]. The next step is to assess the basic neural network structure, with the possible activation functions, number of artificial neurons and other hyperparameters. Some integration tests will be performed, following a tutorial [9] to analyse whether the model class integrates well with the rest of the code.

There are some elements essential for the development of a neural network: the loss function, its gradient, and the optimiser. These also need to be tested, which will be done by comparing the results of simple functions to hand calculations. The code parallelisation will also be tested, with GPUs and with the help of the supercomputer of TU Delft [10]. Finally, it is to be tested whether the trained model parameters are safely saved and can be loaded from the local computer. All these are summarised in Table 5.6, with the last column of the table indicating whether these tests have been passed or not.

[8] Operations on Tensors with PyTorch Tutorial available at: https://pytorch.org/tutorials/beginner/basics/tensorqs_tutorial.html. Accessed: 21/03/2024

[9] Neural Network Structure PyTorch Tutorial available at: https://pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html. Accessed: 21/03/2024

[10] DelftBlue: the TU Delft supercomputer information available at: https://www.tudelft.nl/dhpc/system. Accessed: 22/03/2024

Table 5.6: PyTorch integration and acceptance tests

ID	Test	Expected Output	Pass/Fail
PT-VT-01	Test Tensor operations	Basic operations with tensors equal hand calculations	Pass
PT-VT-02	Test Neural Network class	The class has the specified structure and can be accessed	Pass
PT-VT-03	Test Input to Neural Network class	The class successfully identifies the inputs	Pass
PT-VT-04	Test Gradient calculation	Gradients equal hand calculations	Pass
PT-VT-05	Test Loss Calculation	Loss equals hand calculations	Pass
PT-VT-06	Test Optimiser	Model parameters are updated correctly	Pass
PT-VT-07	Test GPU Acceleration	The model uses GPUs when specified	Pass
PT-VT-08	Test Model Parameters Saving	The parameters of the trained model are saved	Pass
PT-VT-09	Test Model Performance	The parameters of the trained model are successfully loaded	Pass

The correct functionality of PyTorch has been analysed and confirmed. PyTorch is deemed to be a suitable tool for the development of the PINN since it has the desired functions to make the model work, ensuring that these do not have to be developed for the final model and that the project can be completed within the established timeline.

6

Differential Algebra and Gaussian Mixture Model Setup

Once it has been verified that the DA-GMM algorithm functions correctly, the process that is followed for the extension of the DA-GMM algorithm to include the Solar Radiation Pressure acceleration needs to be detailed, which is done in this chapter. Firstly, the steps followed for the modelling of the Solar Radiation Pressure acceleration will be discussed in Section 6.1, with the corresponding verification tests, whereas Section 6.2 will present the inclusion of its uncertainty in the model. Finally, the verification and validation tests performed on this modified algorithm will be presented in Section 6.3.

6.1. Inclusion of the Solar Radiation Pressure Acceleration

Once the software has been validated and it is ensured that it works correctly, the additions to the model need to be made. The inclusion of the SRP acceleration needs to be done in the DACE block of the model, where the propagation of the state and uncertainties is performed. The modelling of this acceleration is quite straightforward, and uses the equations introduced in Section 3.4.

For an accurate formulation of this perturbation, however, the effect of an occulting body needs to be considered. Since the SRP acceleration arises from the momentum which the solar photons carry and transfer upon collision with an object in space (in this case, a satellite or a piece of debris), it is necessary to consider the situation in which no photons are in contact with the body. This can be done via the shadow function.

6.1.1. Formulation of the Shadow Function

There are two main models that can be applied to obtain the effect of shadowing on a satellite: a cylindrical model (which only considers the effect of umbra) or a conical model (which also considers the penumbra). Since differential algebra does not work well with discrete functions, the effect of penumbra needs to be considered to create a smoother transition between the SRP in full sunlight and the umbra region. The parameter used to formulate the effect of shadowing is called the *shadow function*, represented by the symbol ν.

To obtain a mathematical expression for this, geometry needs to be used to obtain the position vectors needed. Two formulations have been studied and compared, to check which one is more suitable for the project. The choice of models has been made taking into consideration the resources available for the student (such as articles or papers), the level of complexity and the level of predicted accuracy. As such, models that were simple enough to add to DACE (i.e., those based in geometry), yet complex enough that accuracy is not lost due to simplifying assumptions, have been modelled. On top of that, cases that could be modelled in DA have been considered (i.e., models that could easily be adapted to continuous functions).

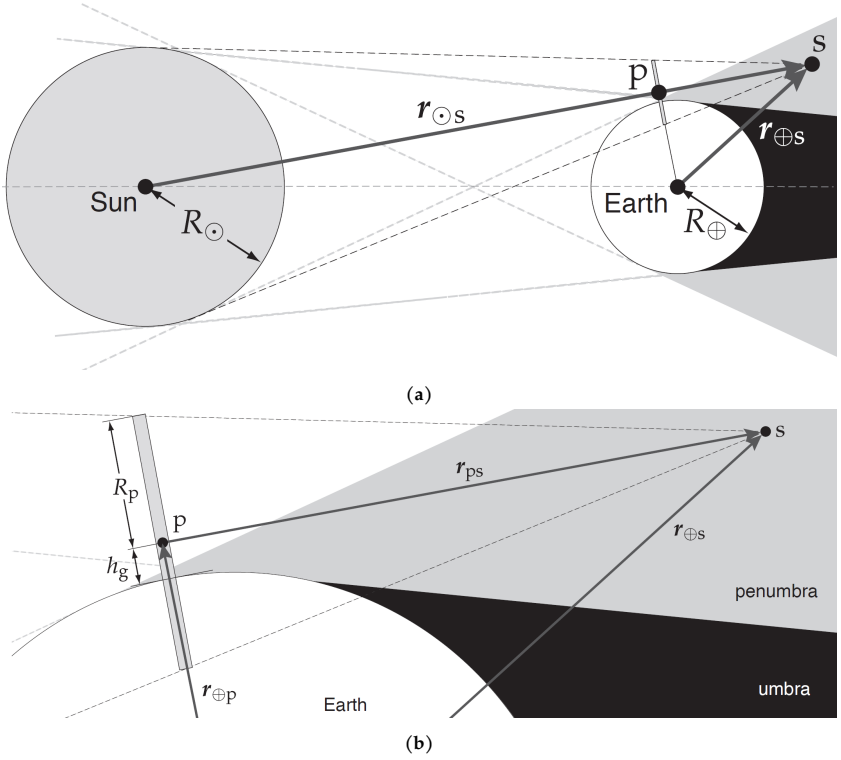


Figure 6.1: Geometric relationship between the satellite, the Earth and the Sun (Doornbos, 2011)

Doornbos formulation of the shadow function

The first formulation analysed is that developed by Doornbos (2011). The geometrical model followed for this formulation is shown in Figure 6.1, and with it the relations below can be found:

$$\boldsymbol{r}_{p_s} = (\hat{\boldsymbol{r}}_{\odot_s} \cdot \boldsymbol{r}_{\oplus_s})\hat{\boldsymbol{r}}_{\odot_s}, \quad \boldsymbol{r}_{\oplus_p} = \boldsymbol{r}_{\oplus_s} - \boldsymbol{r}_{p_s}, \quad R_p = \frac{||\boldsymbol{r}_{p_s}||}{||\boldsymbol{r}_{\odot_s}||} R_\odot \tag{6.1}$$

$$h_c = h_g = ||\boldsymbol{r}_{\oplus_p}|| - R_\oplus, \quad h_b = h_g - R_p \quad \rightarrow \quad \eta = \frac{h_c}{h_c - h_b} \tag{6.2}$$

where $\boldsymbol{r}_{\oplus_p}$ is the position vector from the Earth's centre to point p, which is the closest point to Earth's centre located in the Sun-satellite vector. Similarly, $\boldsymbol{r}_{p_s}$ is the position vector from point p to the satellite, s. R_p is the apparent radius of the Sun. These are all needed to obtain the shadow function, ν. This shadow function uses an auxiliary parameter, η, which represents the case of a full eclipse ($\eta < -1$, partial eclipse ($-1 \leq \eta < 1$), and full sunlight ($1 \leq \eta$):

$$\begin{cases} \eta < -1, & \boldsymbol{a}_{SRP} = 0 \\ -1 \leq \eta < 1, & \boldsymbol{a}_{SRP} = \nu \boldsymbol{a}_{SRP,full} \\ 1 \leq \eta, & \boldsymbol{a}_{SRP} = \boldsymbol{a}_{SRP,full} \end{cases} \tag{6.3}$$

The only thing left is to find the expression for the shadow function:

$$\nu = f_g f_a \tag{6.4}$$

where the parameter f_g is the geometry function and is a measure of the fractional area of the Sun that is blocked by the Earth, and f_a represents the amount of sunlight that is not absorbed by the Earth's atmosphere. The latter will not be considered in this formulation for simplification purposes and therefore will be assumed to have a value equal to one. The formulation of the former is as follows:

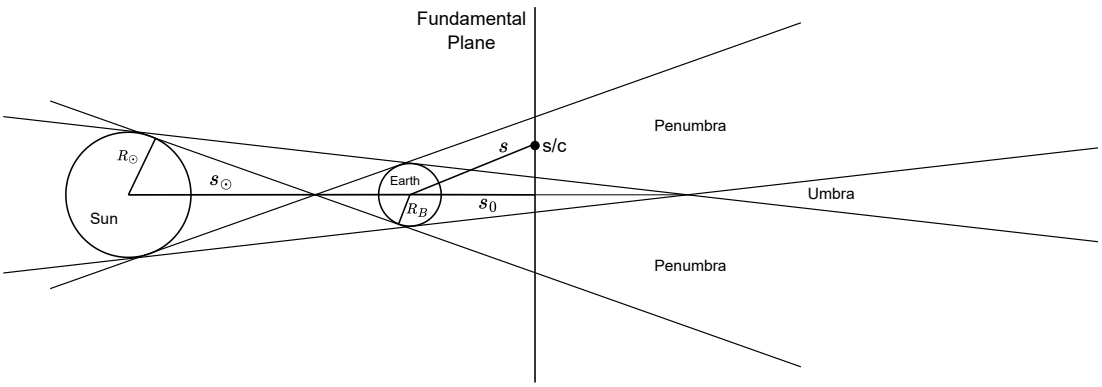


Figure 6.2: Geometry for the conical shadow model (Montenbruck and Gill, 2000)

$$f_g = 1 - \frac{1}{\pi}\arccos(\eta) + \frac{\eta}{\pi}\sqrt{1-\eta^2} \tag{6.5}$$

Montenbruck and Gill formulation of the shadow function

The next formulation is that developed by Montenbruck and Gill (2000). The geometry and parameters used in this work are detailed in Figure 6.2.

In this figure, the variable $\boldsymbol{s}$ simply refers to the position of the satellite with respect to the centre of the Earth and will be represented as $\boldsymbol{r}$ for consistency with the work developed in this project. Similarly, the variable $\boldsymbol{s}_\odot$ represents the position of the Sun with respect to the Earth and will be represented as $\boldsymbol{r}_\odot$. With this in mind, the following auxiliary variables are defined:

$$a = \arcsin\left(\frac{R_\odot}{|\boldsymbol{r}_\odot - \boldsymbol{r}|}\right) \tag{6.6}$$

$$b = \arcsin\left(\frac{R_\oplus}{r}\right) \tag{6.7}$$

$$c = \arccos\left(\frac{-\boldsymbol{r}^T(\boldsymbol{r}_\odot - \boldsymbol{r})}{r|\boldsymbol{r}_\odot - \boldsymbol{r}|}\right) \tag{6.8}$$

where a is the apparent radius of the Sun, b is the apparent radius of the Earth, and c is the apparent distance between the centre of both bodies. Using these, the different scenarios can be obtained: full sunlight ($a + b < c$), full eclipse ($b - a > c$) and partial eclipse (the rest of the cases):

$$\begin{cases} a+b<c, & \boldsymbol{a}_{SRP} = \boldsymbol{a}_{SRP,full} \\ b-a>c, & \boldsymbol{a}_{SRP} = 0 \\ \text{Otherwise}, & \boldsymbol{a}_{SRP} = \nu\boldsymbol{a}_{SRP,full} \end{cases} \tag{6.9}$$

To obtain the formulation of the shadow function as developed by Montenbruck and Gill (2000), the occulted area of the Sun needs to be calculated first:

$$A = a^2 \cdot \arccos\left(\frac{x}{a}\right) + b^2 \cdot \arccos\left(\frac{c-x}{b}\right) - c \cdot y \tag{6.10}$$

where:

$$x = \frac{c^2 + a^2 - b^2}{2c} \tag{6.11}$$

$$y = \sqrt{a^2 - x^2} \tag{6.12}$$

With these expressions, the shadow function at the penumbra region is formulated:

$$\nu = 1 - \frac{A}{\pi a^2} \tag{6.13}$$

The shadow function in the differential algebra environment

The formulation of the shadow function in DACE introduces challenges that are not present for other modelled perturbations. To understand this, a brief review of differential algebra can be provided.

Differential algebra is an area of mathematics that deals with the study and solution of differential equations within an algebraic framework (Kolchin, 1973). In a computational environment, such as DACE, the differential equations are approximated via Taylor series expansions, which are then manipulated algebraically (Evard, 1991). Differentiation is a core aspect of DA and thus the functions that are handled need to be differentiable in the whole of their domain. This is not the case for discrete functions such as ν and might present an issue for its DACE implementation.

The shadow function needs to be represented in a continuous environment, which implies including the effect of penumbra. Both formulations discussed take this into account and thus are considered adequate for the differential algebra environment. The algorithm followed for this implementation is shown in Algorithm 3.

Algorithm 3 Formulation of the shadow function following the Doornbos and the Montenbruck and Gill representations

Require: Satellite position $\boldsymbol{r}$, ephemeris time t, Earth Radius $R_\oplus$, Sun Radius $R_\odot$
  if Doornbos formulation **then**
    Calculate the position of the Sun
    Calculate the position vector from the satellite to point p $\boldsymbol{r}_{p_s}$
    Compute the apparent radius of the Sun R_p
    Compute auxiliary parameter η
    if $\eta \geq 1$ **then**
      $\nu = 1$
    else if $\eta >$ -1 **then**
      $\nu = 0$
    else
      Compute shadow function: $\nu = 1 - \frac{1}{\pi}\arccos(\eta) + \frac{\eta}{\pi}\sqrt{1-\eta^2}$
    end if
  else if Montenbruck and Gill formulation **then**
    Calculate the position of the Sun
    Compute the apparent radius of the Sun a and the Earth b
    Compute the apparent distance between the Sun and the Earth c
    if $a + b < c$ **then**
      $\nu = 1$
    else if $b - a > c$ **then**
      $\nu = 0$
    else
      Compute auxiliary variables x, y
      Compute occulted area of the Sun A
      Compute shadow function: $\nu = 1 - \frac{A}{\pi a}$
    end if
  end if

With the formulas presented above and an overview of the algorithm, it is observed that all operations required are suitable for the DA framework. The *if* statements are usually indicative of discrete

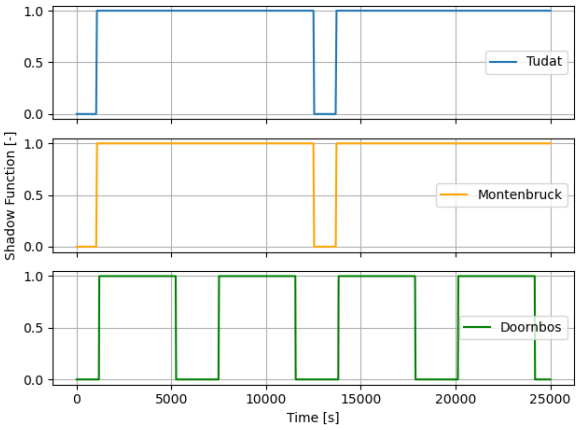


Figure 6.3: Comparison of the shadow function values obtained from Tudat (top), the method proposed by Montenbruck and Gill (2000) (middle) and that proposed by Doornbos (2011) (bottom) for the same satellite and period of time

functions and could be considered an issue in this framework but, as explained, they model a continuous function and thus should not present any modelling error.

6.1.2. Verification of the Solar Radiation Pressure Acceleration

Now that the formulation for the models has been introduced and modelled in the DACE propagator block, their accuracy should be tested. These verification tests are done with Tudat, and the relative tolerance is set to be a 0.1% error with respect to the Tudat value. An arbitrary spacecraft in the LEO region is selected for the verification analysis. The main three aspects to assess is the correct formulation of the solar irradiance, the shadow function, and the SRP acceleration. At an arbitrary point in time, the code calculates these parameters within the set tolerance, as summarised in Table 6.1. Both shadow function models agree in value at the point selected for the test, but this is not deemed enough to measure and assess their accuracy since the satellite was in full sunlight. It is therefore desirable to test the capacity of the model to compute the shadow function during penumbra. On top of that, ensuring that the model accurately predicts the time at which the satellite passes the umbra and penumbra regions is advisable.

Due to the variation of the shadow function, the tests are re-run for different points in the orbit of the same arbitrary satellite. To ensure that the artificial spacecraft selected passes the umbra and penumbra caused by the Earth, its inclination is chosen in such a way that it orbits the Earth on the ecliptic plane. The results obtained from Tudat, the Montenbruck method, and the Doornbos method are plotted and compared in Figure 6.3. In this figure, the shadow function model from Montenbruck and Gill (2000) is shown to be more in accordance with the Tudat values, taken as the 'true' solution, whereas the model proposed by Doornbos (2011) shows a different umbra region than the other two. While both tested methods are heavily based on geometry, the one by Montenbruck and Gill is shown to be more robust and accurate, and hence it is deemed more suitable for this project. On top of that, this method is considered accurate enough that no other methods will be investigated due to time constraints.

Now that the shadow function model has been selected, a more comprehensive analysis of its accuracy shall be made, alongside some verification tests on the formulation of the solar flux and solar

Table 6.1: DA-GMM Solar Radiation Pressure verification tests

ID	Test	Input	Output	True Value	Pass/Fail
SRP-VT-01	Test Solar Irradiance	$\boldsymbol{x}_0, t$	W		Pass
SRP-VT-02	Test Shadow Function	$\boldsymbol{x}_0, t$	ν	Output from	Pass
SRP-VT-03	Test Solar Radiation Pressure Acceleration	$\boldsymbol{x}_0, t$	$\dot{\boldsymbol{x}}_{\boldsymbol{SRP}}$	Tudat	Pass

radiation pressure acceleration models for a period of time. To do so, and with the model of the shadow function proposed by Montenbruck and Gill, the solar flux is tested over a period of 25,000 seconds, as well as the shadow function and hence the total solar radiation pressure acceleration, to ensure that all models are accurate enough for accurate orbit prediction with the DA-GMM method.

While the tests are passed when it comes to a single point in the full sunlight region or in the umbra region, the results on the penumbra region present higher differences. Since the shadow function directly affects both the solar flux and the acceleration calculation, a difference in its value causes discrepancies in all variables. Regardless, these differences cause a small enough error in the SRP acceleration calculation with respect to the results from Tudat and occur over a small enough period (15-20 seconds of penumbra in the LEO region) that it is thought to be acceptable for this thesis.

Tests are then re-run in Medium-Earth Orbit (MEO) since the penumbra region is larger. The discrepancies shown are still deemed acceptable enough due to a low error in the acceleration and the low time spent in penumbra at these altitudes (around 40 seconds in MEO). As the satellite orbit is simulated at higher altitudes (such as in the GEO region) and goes through a larger penumbra region, the errors in the computed shadow function get further reduced. The current formulation of the SRP acceleration is hence considered adequate for the purposes of this thesis. Other formulations could be investigated in differential algebra, but there are limitations on the work done for this project and thus it will remain as it is. The testing of other models is regardless recommended for future students and researchers.

Since this formulation is performed in Differential Algebra, there are a few things that need to be taken into account. First of all, the algorithm cannot handle values equal to zero well, and thus the shadow function value at the umbra region is set to a small enough value (i.e., 1e-10) that it can effectively be counted as 0 while avoiding issues with the algorithm. On top of that, and as mentioned previously, DA does not handle discrete functions well and, since the changes in the shadow function in LEO are significantly abrupt (due to a 10-15 second period of penumbra), it is recommended that the time step chosen for lower altitudes is kept to one second or lower.

6.2. Inclusion of the Solar Radiation Pressure Uncertainty

Now that the formulation of the shadow function is completed, as well as that of the solar radiation pressure perturbing acceleration, the uncertainty in the solar radiation pressure coefficient ($CSRP$) defined previously in Section 2.3 needs to be modelled. As shown and discussed in Figure 5.1, the only change required for this is the addition of this uncertainty in the GMM split block, which generates the Gaussian Mixture Model needed for the propagation in DACE.

There are several types of uncertainty affecting the modelling of the state of an arbitrary spacecraft. In her original work, Leon Dasi includes the uncertainty in the initial state, as well as that in the atmospheric density and ballistic coefficient (due to unknown properties of a satellite). The addition of uncertainty in the initial state is a relevant part of the algorithm developed in Differential Algebra, and has thus been extensively tested. This has not been the case for the uncertainties in the environment, which have not yet been tested in the DACE propagation block. Since this has not yet been done, it has been decided that the uncertainty in the $CSRP$ will not be tested, even though it will be included in DACE consistently with the work developed by Leon Dasi.

6.3. Results of the DA-GMM Extension

Once all the different elements of the inclusion of the SRP perturbing acceleration have been added to the model and verified individually, the performance of the extended algorithm needs to be tested. Since the DA-GMM has already been verified and validated, only the tests that are thought required for the testing of the extension will be performed. Firstly, a sensitivity analysis will be conducted in Subsection 6.3.1, followed by testing on a real-life scenario: the Cosmos-2251/Iridium-33 collision documented in Subsection 6.3.2. To complement this, a synthetic crash scenario is simulated in GEO since this is the region in which the extension could be most beneficial when compared to the original DA-GMM.

6.3.1. Sensitivity Analysis

The performance of a sensitivity analysis is key to the validation of an algorithm, since it allows one to understand the limitations of the model and its accuracy in a variety of scenarios. The tests carried out by Leon Dasi were as follows:

- Effect of orbital parameters:
  - Effect of altitude
  - Effect of eccentricity
  - Effect of inclination
- Effect of DA-GMM model parameters:
  - Effect of lead time
  - Effect of number of GMEs
  - Effect of Taylor series expansion order
  - Effect of integration tolerance
- Effect of uncertainty:
  - Effect of initial state uncertainty
  - Effect of environmental variables uncertainty

Understandably, there is no need to perform all of these analyses again since the changes in the overall model have just been the addition of a perturbing acceleration. It has been shown that the magnitude of the solar radiation pressure acceleration is small enough (typically in the order of 10^{-7} for a satellite in LEO) compared to other accelerations in the model that its effect on the DA-GMM model parameters is considered to be negligible. The sensitivity analyses of the model parameters were used to assess the feasibility of this model in terms of computational load and accuracy, and appropriate trade-offs were made already. Due to the lower magnitude of the SRP acceleration, its addition to the code is thought to maintain the conclusions made regarding the effect of lead time, GME number, Taylor series expansion order, and integration tolerance. Following the same argument, the addition of the perturbing acceleration will not change the conclusions drawn on the effect of the initial state uncertainty, as the original model was accurate enough that this acceleration will not cause large differences in the propagated state of an arbitrary satellite. On top of that, the effect of the uncertainty in different variables from the environment will also not be studied since they have also not been tested for the original model. The sensitivity analyses performed will therefore be of a more physics-related nature, rather investigating the effect of orbital parameters on the model. Since the SRP acceleration affects spacecraft at higher altitudes, it is of high interest to assess the performance of the extended model at different altitudes, and considering that it was found that the eccentricity and inclination did not have a significant effect on the accuracy of the algorithm, they will not be tested.

Effect of a change of altitude in the model

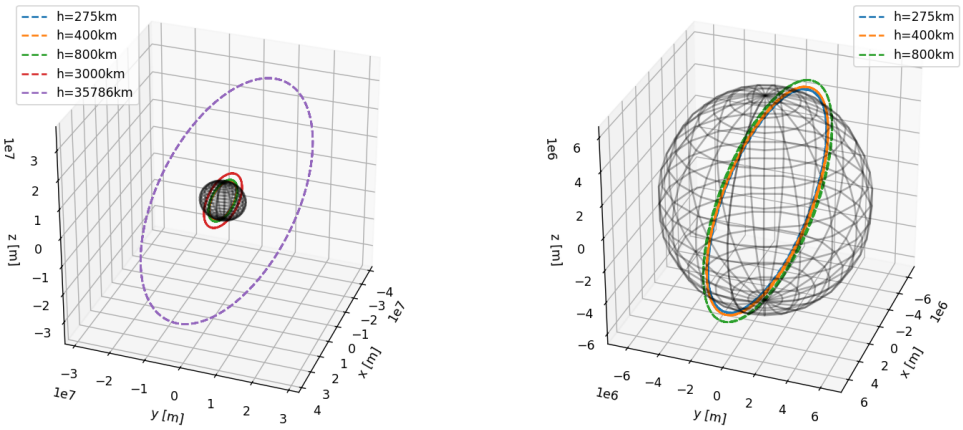
The extension of the DA-GMM will be tested following two criteria: the results from Monte Carlo analyses and the results obtained from the original model. It is thought necessary to re-run Monte Carlo analyses for the range of altitudes selected since the perturbing forces acting on the satellite are different from those considered in the original model. These will provide the geometry of the final uncertainty with which to compare the uncertainty propagated via the algorithm. The nominal scenario is the same as used in the original work of Leon Dasi for the sake of consistency in the comparison of the extended model with respect to its original version. The nominal values of the case considered are summarised in Table 6.2, whereas the parameters used for the propagation of this artificial satellite are shown in Table 6.3. The number of GMEs, Monte Carlo samples and Taylor expansion order are limited by the computational load required to run these analyses. It has regardless been shown (Leon Dasi, 2021) that the error obtained by 201 GMEs with respect to 2001 GMEs is almost negligible ($\sim 5{\cdot}10^{-7}$) and thus is considered accurate enough for this comparison. For higher accuracy, a larger number of MC samples could be selected and the Taylor series expansion order could be set to 4, but for the purposes of this project, the former will be limited to 10,000 and the latter to 3.

Table 6.2: Initial state of Galatea for the nominal scenario considered

Element	h [km]	e [-]	i [deg]	ω [deg]	Ω [deg]	θ [deg]
Value	800	0	60	75	32	40

Table 6.3: Parameters used for the propagation in DACE of Galatea

Parameter	Value	Unit
Number of GMEs	201	-
Number of Monte Carlo samples	10000	-
Taylor Expansion Order	3	-
Reference Area	1.5	m^2
Mass	3.3	kg
C_D	1.2	-
C_R	1.2	-
Epoch	0	seconds from J2000
Propagation Time	5	orbits
$[\sigma_R, \sigma_S, \sigma_W]$	[1, 4, 1]	m
$[\sigma_{v_R}, \sigma_{v_S}, \sigma_{v_W}]$	[0.01, 0.0025, 0.0025]	m/s



(a) (b)

Figure 6.4: Trajectory of an artificial satellite for one orbit at different altitudes

The range of altitudes used by Leon Dasi considered the following values: 200 km, 400 km, 800 km, 3,000 km, and 35,786 km. When performing the simulations with these parameters, however, it is found that the values chosen for a height of 200 km give invalid results in the extended model as the satellite would crash into the Earth's surface. Since Leon Dasi used a different version of Tudat and the DA-GMM, it is highly possible that this went unnoticed. However, this test cannot be run and the lower boundary selected for these tests needs to be changed. After some analysis, the lower boundary of the LEO region by which the satellite does not decay corresponds to an altitude of 275 km. The corresponding trajectories are plotted and shown in Figure 6.4. This shows how the orbits in GEO and the upper limit of LEO considered here will be further away from the effect of the Earth's gravity and aerodynamic drag.

The tests are run, and the results are compared to two sets of Monte Carlo simulations with different dynamical models:

- **Nominal dynamical model**: this corresponds to the full simulation acceleration model, described in Section 3.4 and used in the extended DA-GMM algorithm.

- **Highly accurate dynamical model**: this, on the other hand, corresponds to a more complete dynamical model, which builds up on the nominal model by adding higher spherical harmonics coefficients (up to degree and order 25), and the third body perturbations of Jupiter, Venus and Mars. This is consistent with the results from the perturbation analysis for Tudat performed in Subsection 5.4.1.

To analyse the difference between the final state uncertainty with DACE with respect to that obtained via the Monte Carlo simulations, the L_2 metric is used. The mathematical expression and derivation of this parameter are explained in Section C.2, and it is established that the value of L_2 should be kept lower than 0.6. Due to differences in the Tudat version used for Monte Carlo and the difference in the Matlab version used for the calculation of L_2 to those used by Leon Dasi, a more comprehensive series of tests needs to be run. For the aforementioned range of altitudes, four different test cases are studied:

1. Comparison of the DACE results (with the original model, i.e., without taking the SRP acceleration into account) with respect to the Monte Carlo analyses.
2. Comparison of the DACE results (with the extended model accounting for the SRP acceleration) with respect to the Monte Carlo analyses.
3. Comparison of the DACE results (with the original model, i.e, without taking into account the SRP acceleration) with respect to the Monte Carlo analyses as run by Leon Dasi.
4. Comparison of the DACE results (with the extended model accounting for the SRP acceleration) with respect to the Monte Carlo analyses as run by Leon Dasi.

With this selection of tests, a proper comparison between the two models can be performed. Tests 1 and 2 are used to understand how the DA-GMM algorithm improves when the effect of the SRP is included. From this point onwards, these two tests are grouped in "Case 1" to ease the interpretation and analysis. These will also be compared to those reported in Leon Dasi (2021). However, since the Monte Carlo analyses run by Leon Dasi are different than the ones run here, tests 3 and 4 are necessary to understand the possible differences between the two Tudat versions and the consequences that has on the results of Tests 1 and 2. Tests 3 and 4 will now be grouped in "Case 2".

The L_2 parameter is now calculated for the aforementioned range of altitudes and series of tests, and the results obtained are included in Table 6.4. In this table, the L_2 obtained in these simulations is compared to that obtained with the original DA-GMM algorithm as reported by Leon Dasi. From this table, it is noticeable that the results reported by Leon Dasi for an altitude of 35,786 km are largely different from the ones obtained with the new calculations. This is because an issue arose when trying to quantify the L_2 parameter for this altitude, as the code could not deal with this calculation. This will be explained more in detail later. Furthermore, no results are shown for the altitudes of 200 and 275 km for Tests 3 and 4 since the simulations performed in DACE have not been performed with the former, and the Tudat results from Leon Dasi did not include the latter.

It is observed in Table 6.4 that the results from the analysis of the effect of altitude on the algorithm's performance show an improvement with respect to the original model at all altitudes and cases. The result obtained by Leon Dasi for a height of 200 km was above the error threshold of 0.6 (for both the reduced and full dynamical models), and this was argued to be due to the drag perturbing force. Now, it is known that this was most probably because of errors induced in the model coming from the position of the satellite being within the Earth's surface. The tests for the new lower boundary have shown that the distance between the distribution obtained by the DA-GMM and that obtained with the Monte-Carlo is lower than the threshold established, although the L_2 value is significantly higher than that obtained at other altitudes. This can be attributed to the drag perturbing force, which is the main force at such low altitudes. Even a small uncertainty in the object's state can lead to significant changes in the density in this region. Even so, with this discovery, it can be argued that the model can provide accurate enough results even at low altitudes in LEO, and it has now been modified to assess whether the results of a simulation result in the crash of the object against the Earth's surface, making it more robust. The L_2 value is then observed to decrease when the altitude increases, up to the upper limit of the LEO region considered here. This agrees with the expectations for this analysis since the drag becomes smaller and the density change does not have such a significant effect. On top of that,

Table 6.4: Comparison of L_2 at different altitudes for the different tests, The green-shaded cells demonstrate the simulations with the extended algorithm that perform better than the original one without SRP

Height [km]		200	275	400	800	3000	35786
Reduced dynamical model							
Case 1	Test (1)	-	0.2402	0.0377	0.0085	0.0068	0.2350
	Test (2)	-	0.2371	0.0370	0.0081	0.0062	0.0460
Case 2	Test (3)	-	-	0.0556	0.0207	0.0138	0.2806
	Test (4)	-	-	0.0555	0.0201	0.0131	0.0600
Results by Leon Dasi		1.3775	-	0.0522	0.0199	0.0093	0.0251
Full dynamical model							
Case 1	Test (1)	-	0.2426	0.0453	0.0192	0.0102	0.2388
	Test (2)	-	0.2373	0.0416	0.0186	0.0092	0.0484
Case 2	Test (3)	-	-	0.0595	0.0340	0.0173	0.2842
	Test (4)	-	-	0.0594	0.0331	0.0163	0.0616
Results by Leon Dasi		1.4190	-	0.0593	0.0214	0.0094	0.0273

the SRP acceleration, which the extended algorithm accounts for, increases with altitude, hence further reducing the differences between the extended algorithm and the dynamical models. This is further backed up since the tests in cases 1 and 2 share that same pattern.

For the orbit in GEO, the algorithm is not able to compute L_2 due to the presence of covariance matrices that are not symmetric and positive definite, a necessary condition for the calculation of this parameter. It is to be highlighted that the same code worked for all other tests and provided results consistent with those reported by Leon Dasi. Proper transformations are made to make the covariance matrix positive definite. With these changes, the result produced is different than that obtained by Leon Dasi by about an order of magnitude. Since consistency is kept in the code with respect to all the other altitudes, the values obtained with the modified code will be used for the analysis. In both cases, the L_2 obtained with the extended model is around five times lower than that obtained with the original algorithm. This could be expected as the effect of the SRP is larger at those altitudes, and therefore the extended model which accounts for this acceleration will always be significantly more accurate. Earlier, it was mentioned that, throughout all altitudes in LEO, the difference between distributions got reduced as the altitude increased due to the lower effect of the drag force. However, this is not the case for the GEO satellite, as the L_2 calculated is about 4-7 times larger than that at the upper LEO limit. The reason for this could be rooted in other perturbation uncertainties. In geostationary orbits, the relative relevance of non-Earth forces (i.e., gravitational attraction from the Moon and the Sun and the SRP) increases as the gravitational pull from the Earth decreases. As such, uncertainties in any of these forces will become more significant, especially with the larger propagation times required for this orbit. It has been established that the uncertainties in the third body perturbations from the Moon and Sun are almost negligible, so let's focus on those from the SRP acceleration. It has been argued that differences in the solar flux over the solar period are around 1%. If the models to calculate this flux are different for Tudat and the DA-GMM, this 1% difference can lead to a larger divergence of the results from both propagations. This leads to a larger difference in the nominal state propagated and a larger divergence of the final uncertainties.

It has been shown that the extended model is more accurate than the original DA-GMM, although the difference is generally very small. A larger improvement is observed for an arbitrary satellite in GEO, which was to be expected as the SRP acceleration is more significant at those altitudes. On top of that, the algorithm is valid at all altitudes, since the L_2 parameter is lower than the threshold established at 0.6, although the effect of the drag force at lower altitudes is significant enough that the addition of rotational motion in the model is recommended. To visualise what the L_2 error represents, the state and uncertainty as propagated by DACE (for the case of an altitude of 800 km) are plotted alongside those provided by the Monte Carlo analyses in Figure 6.5. In this figure, the distance between the two distributions seems to be almost negligible as they are mostly aligned. On top of the increase in accuracy, the extended algorithm is able to detect when the satellite orbit is not feasible.

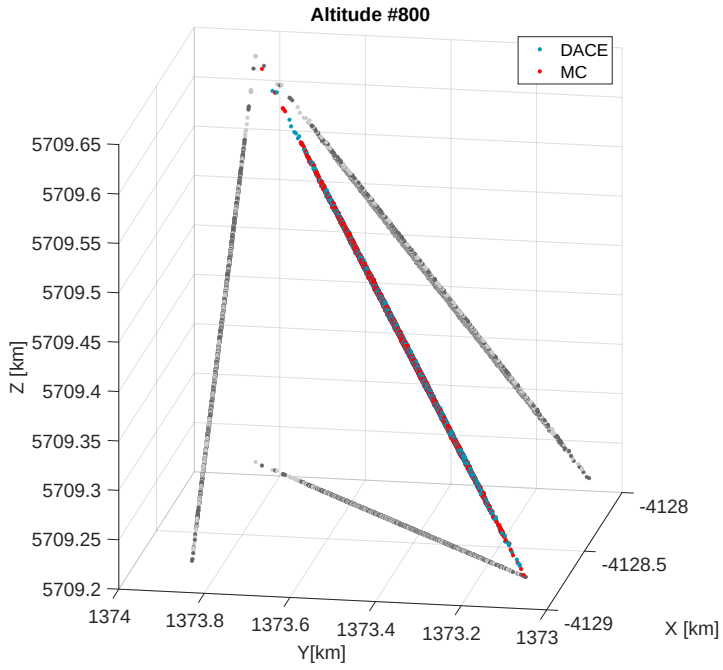


Figure 6.5: Example of L_2 error for an altitude of 800 km, corresponding to a L_2=0.0081

Table 6.5: Cartesian state of Cosmos-2251 and Iridium-33 on February 10, 2009 at 00:00 UTC (Leon Dasi, 2021)

Satellite	x [km]	y [km]	z [km]	v_x [km/s]	v_y [km/s]	v_z [km/s]
Cosmos-2251	2689.06	2726.88	6049.935	-6.53545	-1.16479	3.40109
Iridium-33	-3240.99	4759.60	4246.546	2.01107	-3.9914	5.98213

Although the calculation of L_2 has provided good results, it is key to reiterate that these results were highly sensitive to the random generator used in Matlab. With this in mind, the results shown can be taken as an approximation of the real values, but overall they have been observed to be at least in the same order of magnitude, and thus valid for this analysis. This sensitivity analysis has shown that the extended algorithm is more accurate, particularly at high altitudes, and more robust than the original model. It has also been validated for a series of altitudes, demonstrating its usefulness in a wide range of applications.

6.3.2. Cosmos-2251/Iridium-33 collision

Once the algorithm has been validated throughout a series of altitudes, it is desirable to test it with a real-case scenario. The one chosen for this is the Cosmos-Iridium collision, introduced in Section 2.6. This is the most severe collision between satellites up to date, generating more than 1,000 pieces of debris larger than 10 cm (Kelso, 2009) which currently need to be tracked. The initial state for both satellites before the event is available in the form of a TLE on Space-Track, which is then transformed to Cartesian coordinates in the ECI frame of reference. The resulting coordinates for each satellite are summarised in Table 6.5. The uncertainty in the RSW frame for the state obtained from TLEs was already presented in Table 2.2.

Before presenting the results, the parameters of the simulation shall be introduced. The collision took place on February 10, 2009, at 16:55:59 UTC (Kelso, 2009), set as the TCA, and so the simulation is run from TCA - 150 s to TCA + 150 s. The Taylor series expansion is set to be of order three and the number of GME is set to 37. The integrator selected is a Runge-Kutta 7(8) with a time step of one second. All of these were selected by Leon Dasi (2021) and will be kept as such for actual testing. On top of that, both satellites are modelled as hard-body spheres with a combined radius of 5 m, and the

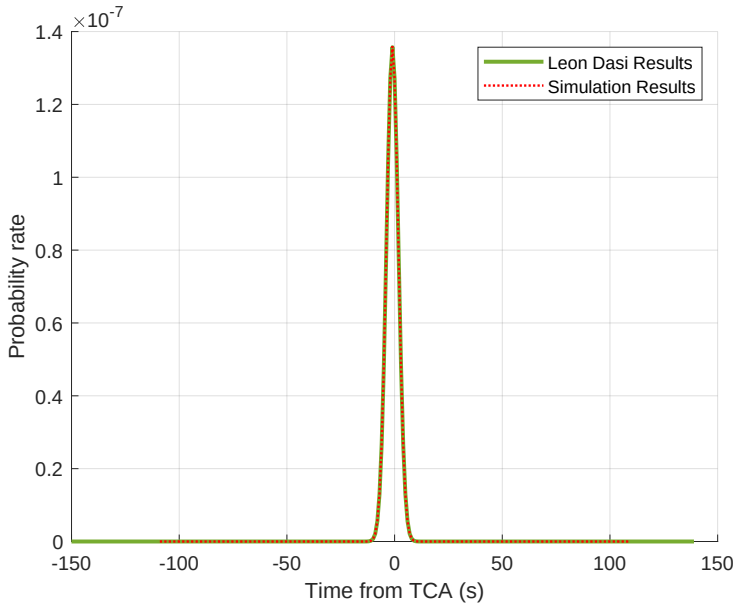


Figure 6.6: Probability of collision rate of the Cosmos-Iridium scenario with the original and extended versions of the DA-GMM

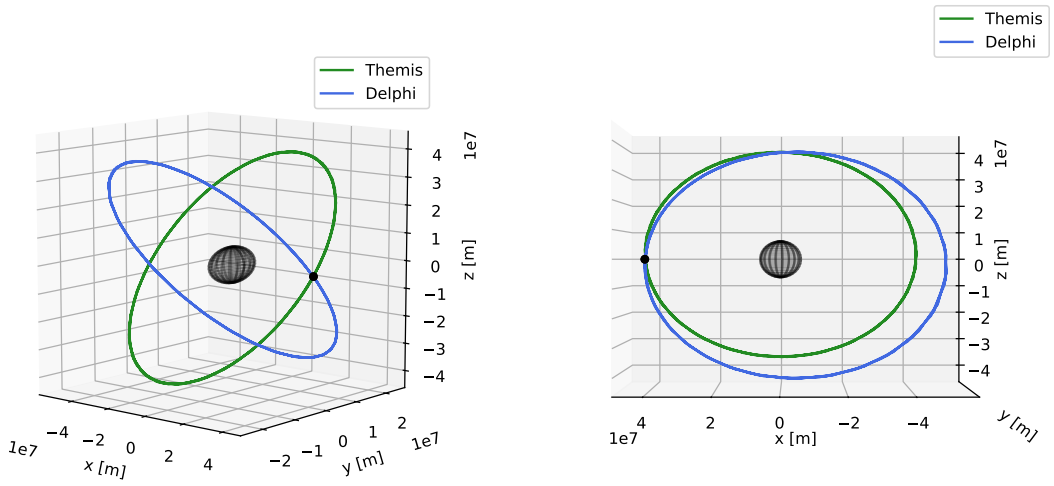
Table 6.6: Collision probability predicted with the original DA-GMM and the extension of the algorithm for the Cosmos-Iridium collision

	$\mathbf{P_c}$	$\mathbf{\Delta P_c}$
Original	$9.4320 \cdot 10^{-7}$	-
Extended	$9.4334 \cdot 10^{-7}$	+0.015%

drag coefficient for both is set to be 2.2. Finally, the radiation pressure coefficient (C_R) of both satellites is selected to be 1.2.

The results obtained for the value of the probability of collision rate for the duration of the encounter are plotted in Figure 6.6 and compared to those obtained with the original algorithm. The most noticeable feature in this figure is the near-identicalness of both curves. This could be expected since the magnitude of the SRP acceleration is typically small (in the order of 10^{-7}) compared to the other perturbing accelerations for LEO. Moreover, the sensitivity analysis performed on the effect of altitude demonstrated that at altitudes of 800 km the difference between both algorithms is quite small. This claim gets reinforced when looking at the values of the probability of collision, summarised in Table 6.6. The difference between the P_c obtained with the original algorithm and the extended one is minimal. In fact, the inclusion of the Solar Radiation Pressure perturbing acceleration results in only a 0.015% increase with respect to the algorithm that does not have this perturbation in the dynamical model.

In the days prior to the collision, no alarms were raised on the event between these two satellites as the predicted collision probability was extremely low. While the results obtained with this algorithm showcase a value larger than what was predicted back then, it is still below the 10^{-4} or 10^{-5} threshold that many companies and organisations use. It is important to recall that the initial state uncertainty in the position and velocity is large, and that the characteristics of the satellites are not known and can only be estimated. These factors make the P_c calculated inaccurate and lower than it should be. Through the attainment of higher quality data for the simulation (i.e., a more accurate initial state and object parameters), this issue could be solved. Regardless, it can be concluded that the algorithm (as was the case with the original version as developed by Leon Dase) is capable of predicting the TCA of the collision event. With highly accurate data, the uncertainty matrix would be reduced, and the calculated P_c would be higher.



(a) (b)

Figure 6.7: Trajectories of Themis and Delphi with their intersection point

6.3.3. Geostationary Orbit Crash Scenario

The Cosmos-2251/Iridium-33 collision has demonstrated that the model does not have a significant effect on the result of the model at lower altitudes, which agrees with the results obtained from the sensitivity analysis. Since the solar radiation pressure acceleration has a larger effect at higher altitudes, it is desired to test the performance of the extended model for a collision event in a geostationary orbit. However, since this region is not as crowded as LEO and the distances between satellites are much larger, no significant collisions have occurred in GEO. Instead, a synthetic crash scenario is simulated.

To simulate this, two artificial satellites named Themis and Delphi in intersecting orbits are created. The orbit of Themis is geostationary, whereas the orbit of Delphi is set to be an elliptical orbit whose perigee is located in the orbit of Themis. Both satellites are positioned at this intersection point at an instant in time and their positions are then backpropagated 24 hours, first with the original DA algorithm and then with the one accounting for the SRP acceleration. It is because of this that the probability of collision expected for such an encounter is P_c=1. However, the introduction of uncertainties in the initial state will lead to a lower probability since the state of the satellite is assumed to not be accurately known.

Since this is a simulated crash scenario, it will be assumed that the initial state is known with an accuracy an order of magnitude higher than can be achieved with the use of TLEs. The uncertainty in position and velocity are defined as such: σ_r=10 m for each position component, and σ_v=10^{-2} m/s for each component of the velocity vector. The uncertainty is not set higher to prevent the *probability of collision dilusion* (Sanchez and Vasile, 2021), which states that large uncertainties lead to a reduction in the probability of collision and therefore to underestimations of the P_c calculated, as happened with the Cosmos-Iridium collision. With a lower uncertainty ellipse, the accuracy of the orbit and uncertainty propagation can be better assessed.

Once the parameters of the satellites have been determined, the Differential Algebra algorithm is run. In the scenario that takes into account the Solar Radiation Pressure acceleration, the model estimates a probability of collision $P_c \simeq 2.89 \cdot 10^{-4}$, as summarised in Table 6.7. As explained in Section 1.2, the threshold established by most conjunction assessment organisations for events of high risk is 10^{-4} and hence this event would be correctly classified as such. As explained before, even though the actual P_c should equal 1, the uncertainties in the state of the satellites, as well as errors present in the propagation of these states cause this value to be reduced. Now, the same scenario is run with the original algorithm, which estimates $P_c \simeq 1.1251 \cdot 10^{-5}$, a value one order of magnitude lower than that obtained with the extended model. On top of that, the TCA (set at t=86,400s since the start of the propagation)

Table 6.7: Collision probability predicted for the simulated scenario Themis-Delphi

	$\mathbf{P_c}$	$\mathbf{\Delta TCA}$ [s]
Original	1.1251e-05	1
Extended	2.8926e-04	0

is accurately predicted with the extended algorithm, whereas the model without SRP acceleration predicts this to be one second later. Both the values of the P_c and the TCA calculated would indicate that the propagation performed with the model that does not take into account the SRP acceleration is less accurate at higher altitudes, as was expected. The addition of this acceleration is then seen to make the model more robust, even if the effect at lower altitudes is not as noticeable.

Although the potential collisions for satellites in GEO are lower than those in LEO, there is still a desire to create an algorithm that can predict these accurately. Collisions in GEO can also be quite catastrophic since the debris generated will likely remain in orbit for long as the effect of gravity is not strong enough to make these pieces decay, as happens in LEO. On top of that, the operations of these satellites at high altitudes are essential since many of them are dedicated to communications, research or defence purposes. The constellation of the Global Positioning System (GPS), key for the operation of other many satellites and operators on Earth, is located at high altitudes (at around 20,000 km), and having models that can accurately predict potential collisions at such altitudes is beneficial for their safety.

6.4. Conclusions

In this chapter, several analyses have been performed to test the state and uncertainty propagation of the extended DA-GMM algorithm and its accuracy with respect to the algorithm as developed by Leon Dasi. Throughout the sensitivity analysis on the altitude and the testing on a real and simulated collision case, several conclusions are drawn, summarised below:

- The extended algorithm provides more accurate propagation results than the original code at all altitudes.
- The effect of the SRP is not as relevant at lower altitudes, where the effect of drag is more noticeable.
- The extended algorithm particularly outperforms the original model at higher altitudes (GEO), where the effect of SRP is more prominent.
- The DA-GMM still performs worse at GEO when compared to other altitudes in LEO, probably due to uncertainties in the solar radiation pressure, which become more prominent at such heights.
- The extension of the code allows for a higher robustness than the original one.
- It is recommended that at low altitudes with a small penumbra region, the time steps for the simulation are kept to 1 second or lower to avoid issues with the differential algebra algorithm for the calculation of a semi-discrete variable as is the shadow function.
- When tested on a real scenario, the extension does not show much improvement with respect to the original code. However, it does show a significant improvement over a synthetic scenario in GEO, reinforcing the previous conclusions drawn.

7

Physics-Informed Neural Network Setup

The approach selected in this project for the mitigation of space debris is the development of a Physics-Informed Neural Network for collision risk assessment and collision avoidance. The aim of using an ML-based approach with integrated physics is to increase the accuracy of the predictions while keeping a low computational load. This chapter aims to provide a detailed view of the setup of the algorithm developed for this project. The data pre-processing, including the methods required to prepare the input data to train the PINN, is discussed in Section 7.1. The actual training of the model is presented in Section 7.2, alongside a thorough analysis of its optimal configuration. The data post-processing, which includes the uncertainty estimation and calculation of the probability of collision P_c, is discussed in Section 7.3. Finally, the tests for the verification and validation of the model are presented in Section 7.4.

7.1. Data Collection and Pre-processing

Neural networks are a revolutionary concept since they are able to learn exclusively from data and predict patterns. It is a series of data that guides the training process of the algorithm, allowing the neural network to adjust the parameters of the model to capture these patterns. Taking this into consideration, the pre-processing of the data is a key step when working with deep learning algorithms since its accuracy will depend on the quality of these datasets.

The first step that needs to be taken is to collect the required data to train the model. TLEs are selected as the data source for the model, following the discussion provided in Section 2.5. These are retrieved from Space-Track[1], the most complete database for the TLE of different objects. Generating the datasets first requires the RSO data to be downloaded, which can be done through their API.

With this API, the data of any tracked satellite or piece of debris can be downloaded. The raw data from Space-Track then needs to be pre-processed to be transformed into data suitable to be fed into the algorithm. TLEs are composed of the mean orbital elements of the satellite, which are not an actual representation of the instantaneous state of an object, but rather a mean over a certain number of observations. Typically, an SGP4 propagator is used to transform TLEs into actual state information. In *Python*, which is the programming language used for the API and the development of the PINN, there is an SGP4 propagator available that transforms TLEs into Cartesian coordinates.

The question now is which state representation is desired to be predicted by the neural network. The representation via Cartesian coordinates might be complicated for the network to learn due to the rapidly changing elements of which it is composed. Kepler elements could be an alternative, with a more constant nature of five of the six elements that represent the orbit. However, for nearly circular orbits (not uncommon in the field of communication constellations such as Starlink), two elements in

[1]Space-Track website available at: https://www.space-track.org/. Accessed: 2024-04-01

the set are underdefined: the argument of periapsis and the true anomaly. It would be desirable to use a state representation free of singularities, such as MEE or USM7. The main issue with MEE is that, for a perturbed orbit, some elements follow a sawtooth pattern. This happens for Kepler elements as well and would pose a large challenge for the neural network since it is a discrete function. The architecture selected for the network makes it unsuitable for these types of functions. The model parameter optimisation, fundamental for the training process, relies on the backpropagation of the gradients of the function. Discrete functions are, by definition, functions that are not differentiable in the whole of their domain. As such, there might be issues with gradient exploding, reducing the capacity of these models to predict these types of functions (Sanchez and Vasile, 2021; Van den Oord et al., 2017). A possible alternative to this would be to split the function into the sections in which it is continuous, thus removing the discrete jumps in the function, and train the model with those sections separately. Due to constraints in the thesis, this will not be considered but will remain an observation and recommendation for future work. All elements in the USM7 and Cartesian coordinates have a smooth, continuous nature and, as such, will be the two state representations considered for the training of the network. Which one is more suitable for this task remains to be tested.

With any of the representations selected, the propagator is used to generate a large dataset, with its points equally distanced in time and close together to enhance the learning process and prediction capabilities. Once this is done, the elements contained in the datasets might have a difference in scale, and thus normalisation is needed to bring all of them to the same scale and avoid some features from dominating the learning process. Moreover, it might help with the large changes in Cartesian coordinates, making them more suitable for training. The method chosen is the min-max normalisation. For a random variable x, this would look like:

$$\bar{x} = \frac{x - x_{min}}{x_{max} - x_{min}}, \qquad \bar{x} \in [0, 1] \tag{7.1}$$

The final step that needs to be done in the pre-processing block is the separation of the data into test, validation and training sets. Typically, a separation of 80/20 for the training and test datasets is chosen, with 10% of the training set used for validation purposes if required (Peng and Bai, 2018a).

7.2. Physics-Informed Neural Network Training

Once everything has been set up, the next step is to train the model. Using a single satellite for this creates a simplified scenario that allows for the assessment and analysis of the PINN's performance, as well as finding which configuration works best for this type of problem. By testing the algorithm for a single satellite, the complexities ingrained in the data are greatly reduced, allowing for a deeper understanding of the results obtained and a more solid basis to make decisions on how to build a more generic tool. As such, several things need to be tested. It is important to recall that the analyses done for this section are focused on the numerical errors that can be induced in the model due to the decisions taken, rather than the underlying physics of the environment.

First of all, a satellite needs to be selected. Since Iridium-33 has already been tested on multiple occasions throughout this thesis, it has been chosen for these analyses. The evolution of the satellite's state for a period of time is calculated with Tudat. Starting at the position indicated in Table 6.5, the position is propagated for one day. At one data point per second, the final dataset to train this model consists of 86,400 points, between the 10^4 - 10^5 recommendation in Yang et al. (2020). This is also considered appropriate because it contains more data points with which to train the model, and capturing the feature's periodicity is easier as more orbits are added to the training data.

The typical training process for a neural network was described in Chapter 4, with the detailed architecture of the PINN introduced in Subsection 5.1.2. To summarise, the inputs to the model (i.e., the normalised satellite state data) are used to compute the values of the cells in the first hidden layer. This is calculated using the randomly initialised weights and biases of the connections between layers. The values of these hidden cells are then used to calculate those for the next hidden layer following the same process. This is repeated for every hidden layer in the neural network architecture, and finally for the output layer. The output of this layer is the predicted normalised state of the satellite in the future, which is compared to the actual output given to the network. With this, the loss of the training

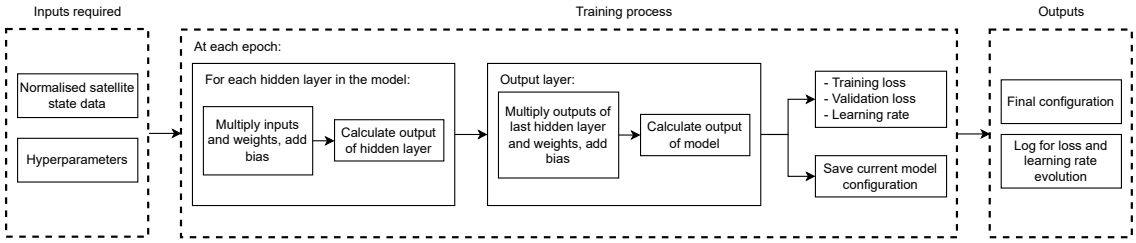


Figure 7.1: Training process flowchart

Table 7.1: Initial configuration for the neural network testing

Hyperparameter	Value
Number of hidden layers	1
Neuron in hidden layer	500
Activation function	Hyperbolic tangent
Learning rate	10^{-4}
Batch size	512
Maximum number of epochs	10,000
Sequence length	1
Optimiser	Adam
Loss function	Mean-square Error
Regularisation method	Early Stopping

and validation datasets is calculated and used to update all the weights of the network. These weights are adjusted at each epoch until the predicted output of the model matches the actual output given. The final configuration (i.e., the weights and biases of the model after convergence) is saved and can be used to predict the state of the satellite with no knowledge of the actual future state. The training process can be visualised in the pipeline in Figure 7.1.

Many tests need to be done regarding the architecture of a deep learning model before finding the set of choices that lead to the most optimal configuration. Since this also depends heavily on the specific task the algorithm is used for, the initial step would be to recur to the literature and assess models currently used for orbit prediction. This can be adapted to suit the dataset used in this specific work.

7.2.1. Hyperparameter Tuning

Based on the work from Cheval (2023) and Sanchez and Vasile (2021), a preliminary set of hyperparameters are chosen and summarised in Table 7.1. A sequence length of 1 is chosen to simplify the architecture for the initial tests but is subject to change in the hyperparameter tuning.

With this initial architecture, the tests can now be run. Each of these tests will require a small training process on the neural network with the initial configuration to assess what works best. The convergence speed and accuracy of the loss of the test dataset, chosen as the performance metric for this set of trials, will be checked and informed choices will be made. The configuration assessments are those described in Section 7.2.

Test 1: Data Pre-processing Techniques

Firstly, the influence of pre-processing techniques on the neural network learning process was analysed. To do this, the raw and normalised data from the USM7 dataset were fed to the model and the test loss was analysed. It is to be noted that the results for this specific test are not entirely comparable, since the loss of the raw data will be by default much larger than that for the normalised data due to the difference in scales. Therefore, the normalised predicted states will be returned to their original scale before performing the MSE by hand.

Test 1 shows that data normalisation is fundamental for machine learning for orbit prediction. The test loss obtained with the normalised data is $1.91184 \cdot 10^{-3}$, much smaller than the (normalised) test

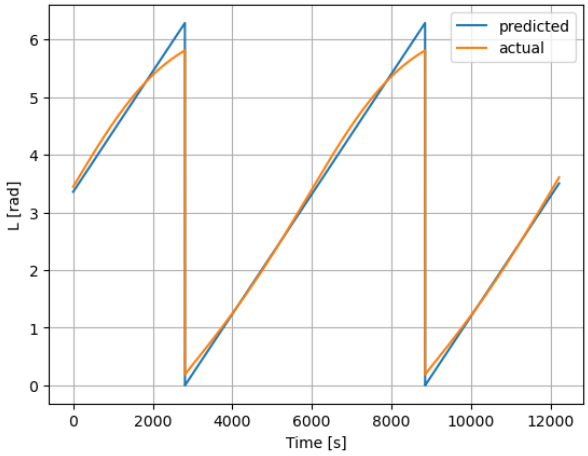


Figure 7.2: Comparison of the predicted L-parameter from the MEE state with its actual value

Table 7.2: Summary of the train and test losses obtained by training the neural network with different state representations, ordered from lowest to highest test loss

State representation	Training loss (at last epoch)	Test loss
Cartesian Coordinates	$2.43061 \cdot 10^{-4}$	$1.91184 \cdot 10^{-3}$
Unified State Model	$1.28471 \cdot 10^{-3}$	$2.88750 \cdot 10^{-3}$
Modified Equinoctial Elements	$2.97648 \cdot 10^{-3}$	$1.33832 \cdot 10^{-2}$
Kepler Elements	$2.98849 \cdot 10^{-3}$	$2.06088 \cdot 10^{-2}$

MSE from the raw data, with a value of 0.56022. On top of that, the neural network trained with raw data could not converge, proving that having large training feature values creates an unstable system where the optimiser struggles to move in the right direction. This test demonstrates that data normalisation is key for the model's performance.

Test 2: State Representation

Once the dataset normalisation is established as necessary for increasing the efficiency of the training process, the best set of elements used for this needs to be selected. Even if it was discussed that only Cartesian coordinates and USM7 would be studied for this, the tests will also be run for Kepler elements and MEE to show whether the algorithm has issues with their prediction.

The performance parameter selected is, once again, the test loss as it indicates how the algorithm fares with the prediction of points in the dataset with which it was not trained. On top of that, the training loss will be used to assess the convergence of the model. As such, lower training losses over the same number of epochs indicate a faster convergence and thus a more efficient configuration. The results from these tests are summarised in Table 7.2. As expected, the model obtains lower errors when trained with a time series of Cartesian coordinates or USM7 elements than trained with Kepler elements or MEE. The prediction of the L parameter of the MEE is shown in Figure 7.2. The algorithm struggles to capture the discrete sections of the element, where the sharp peaks of the actual L-parameter take a curved appearance in the prediction, increasing the training and test losses. This confirms that the type of neural network used here is not suitable for such functions, thus some other approach should be used to predict these discrete elements. On the other hand, both Cartesian and USM7 provide good results and both could potentially be used for the final training of the model. However, since Cartesian coordinates provide a lower error than the USM7, they are preferred for the final training of the algorithm.

Regardless, it is good to know that both elements can provide better results since then the Interval Method can be used to reduce and estimate the uncertainties of the model.

Test 3: Dataset Size

The size of the dataset with which to train any machine learning algorithm is key for training. Theoretically speaking, the larger the dataset, the better the capacity of the algorithm to learn and replicate the underlying patterns of the data. However, there is a trade-off to take into account, as is usually the case. The larger the dataset, the longer it takes to train the model and, although argued earlier that training time is not a big issue since *DelftBlue* will be used, there are limitations to every system.

The current size of the model is in the order of 10^4, but sources like Shin et al. (2022) recommend the use of larger datasets in the order of 10^5. As such, it will be investigated whether using a dataset that doubles the one used in data points yields better results. The dataset is augmented using Tudat to propagate the orbit of Iridium-33 for two days instead of one. This new dataset is tested in the machine learning algorithm and provides a test loss reduction of 2.5 times with respect to the smaller dataset. The dataset is now augmented again, up to four times the size of the original dataset. In this case, the reduction in the test loss that is obtained is not as significant in relation to the dataset containing 172,800 points, while taking a significantly longer time. As such, the dataset will be kept for the future for two days of data.

Test 4: Training Split

Once the dataset size is selected, the optimal training-test split should be assessed. This split evaluation is necessary to maximise the model's generalisation ability and check when overfitting occurs. On the other hand, the split needs to be chosen so that the model still has enough data points to learn the patterns as required. As such, several training-testing splits were tested: 70/30, 80/20 and 90/10.

The 80/20 split was used for the previous tests, which yielded a test loss of $1.92470 \cdot 10^{-4}$. Since the generalisation of the model is being assessed, the test loss is still considered a good performance metric for this case. The split is first changed to 70/30, which yields a higher test loss at $2.24365 \cdot 10^{-4}$. Next, a 90/10 split is assessed, which results in a test loss of $1.10245 \cdot 10^{-4}$, significantly lower than the one obtained with the original 80/20 split. One step further is taken, and the test size is reduced to 5% of the total dataset, leaving 95% for the training. This split sees a test error of $2.091352 \cdot 10^{-4}$, indicating that the model already sees some overfitting. As such, a 90/10 split provides the best prediction results and thus will be considered for the final configuration. The use of a validation dataset (typically 10% of the total dataset size) for regularisation methods is still used to prevent overfitting.

Test 5: Hyperparameter Tuning

Once these architectural choices have been made, it is time to tune the hyperparameters of the model to ensure that the best result is produced upon training. Section 4.4 already detailed different methods to do this tuning, and it was decided that a random search was the most efficient method (Bergstra and Bengio, 2012) while not being too time-consuming. In total, the neural network was run 10,000 times, all with different sets of hyperparameters. Since the optimiser is set to be Adam, the loss function to MSE, and the regularisation method to early stopping, the parameters to be changed are: the number of hidden layers, the number of neurons per hidden layer, the activation function (where the different types considered are discussed and explained in Section C.3), the learning rate and the batch size. All of these are pseudo-randomly generated inside of a range of values established for these. The results of this random search are summarised in Table 7.3.

One final test was done with this set of values to assess carefully the final performance of the algorithm and check if anything needed to be manually tuned. As it turns out, the model was able to quickly reduce the loss of the training and validation datasets, but they were difficult to get lower past a certain value. This would indicate that the learning rate is too high for convergence. However, lowering the learning rate would mean a slower convergence since this was only the issue past a certain optimisation iteration. As such, it was decided to use an adaptive learning rate so that it gets reduced as the NN learns to aid the convergence to lower loss values. The method selected for this is the exponential decay, which has been used for PINNs before (Cheval, 2023):

$$\alpha_i = \begin{cases} \alpha_0 & i < i_0 \\ \alpha_0 \cdot \beta^{\left(-\frac{i-i_0}{s}\right)} & i \geq i_0 \end{cases} \tag{7.2}$$

Table 7.3: Summary of final configuration for the neural network training

Hyperparameter	Value
Number of hidden layers	7
Neuron in hidden layer	100
Activation function	Rectified Linear Unit (ReLU)
Learning rate	10^{-4} (adaptive)
Batch size	512
Maximum number of epochs	10,000
Sequence length	20
Optimiser	Adam
Loss function	Mean-square Error
Regularisation method	Early Stopping

where α_i is the learning rate at the i^{th} epoch, α_0 is the initial learning rate (set at 10^{-4} as per the random search results), i_0 is the epoch at which the learning rate is set to start decaying with β decay rate, and s is simply a scaling factor to prevent the exponent from becoming too large. The decay rate and scaling factor are chosen on a trial-and-error basis, with values of 0.05 and 10000 respectively.

Another aspect to be decided is the sequence length of the training datasets. Since the central part of the neural network is an LSTM layer, it can learn from past data points in the dataset. A sequence length of one is not considered enough for the model predictions to be accurate for the aim of the research in this thesis. As such, a series of values was tried, and a sequence length of 20 was selected. It was seen that increasing the sequence length further led to much slower convergence, infeasible even for *DelftBlue* without much gain in accuracy. All of these architectural choices are summarised in Table 7.3, considered the final configuration of the neural network with which the training will be done.

Once the optimal set of characteristics for the model has been found, the neural network needs to be trained. This implies, in generic terms, optimising the weights and biases of the network to minimise the loss term and therefore predict the data patterns more accurately. Once the algorithm is not able to reduce the loss further, it can be said that it has successfully 'learnt' the behaviour of the data as much as possible and therefore should be able to reproduce these patterns.

7.2.2. Limitations

The proposed approach for the neural network, however, does present some issues and limitations that need to be taken into account. The "main" architecture of the model developed belongs to a LSTM, introduced in Section 2.4, and this type of neural network is designed for fixed time-step predictions because its architecture requires a consistent spacing between data points to understand the underlying patterns of the data. Therefore, the model described previously can only take 20 time steps ($t-20, t-19,$ ... $t-1, t$) and predict the state of the satellite at $t+1$. This could ideally be done recurrently to obtain the state of the satellite at the subsequent time steps. In reality, this would lead to poor predictions after a short amount of time due to the fast propagation of errors, and thus the "accurate" prediction window is too reduced to be considered for collision probability calculation. Other methods shall be considered for a larger stability of the neural network.

It is said that LSTMs are, by default, fixed time-step, but a question that arises is whether the architecture can be changed for a certain variability. If a model is trained for different time steps ($\Delta\mathrm{t} = 1\mathrm{s}$, $\Delta\mathrm{t} = 5\mathrm{s}$, $\Delta\mathrm{t} = 10\mathrm{s}$, etc.), then the state of the satellite could be obtained at any desired point in time. However, this is highly infeasible since it would require to train $N-1$ networks for N time steps. If a prediction of one day needs to be made, this would require training 86,399 neural networks to do so! Even with access to a supercomputer, this is too computationally expensive to be considered.

There is another possibility, much simpler and effective, to induce "variability" in the predictions and improve the stability of the network over a larger period of time: a certain prediction window can be selected so that the algorithm uses a sequence length of 20 time steps to learn the state of the satellite after a certain prediction horizon. For easier visualisation, this is shown in Figure 7.3, where a vector of states of a certain length (i.e., the sequence length) is used to predict the state at $\mathrm{t} + 3600\mathrm{s}$.

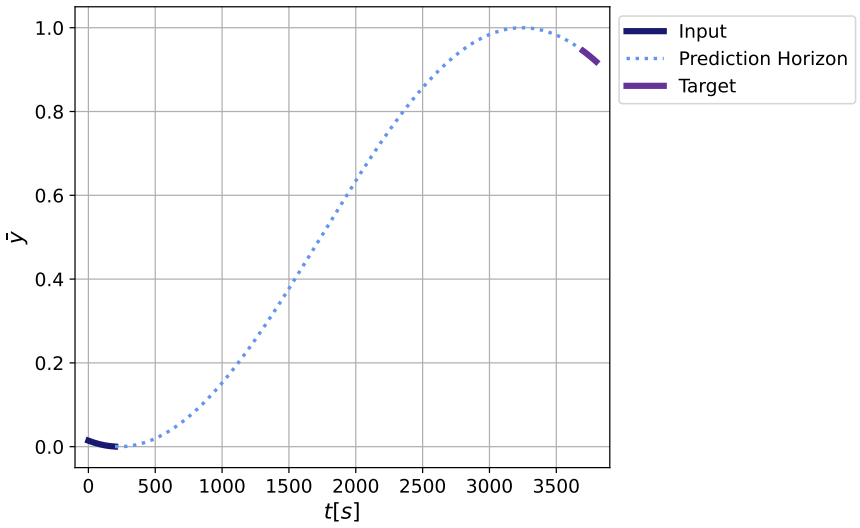


Figure 7.3: Visual representation of the input and target sequences for the neural network method presented

It must be taken into account that, for this method to work, the user will need the state of the satellite from $t - 20$ until $t+\Delta t$, for which a propagator is required. This propagation shall need to be short enough so that the propagation does not take too much time (as this would defeat the use of neural networks) yet long enough so that it is stable for a longer amount of time. As such, an hour is selected as the prediction horizon. The accuracy and error propagation of this new methodology will be assessed against that previously introduced.

7.3. Data Post-processing

The raw output of the neural network is in the form of a time series of predicted normalised states. Data post-processing is, therefore, needed to transform these results into interpretable data for further analysis. The first step required is to revert the normalisation process to obtain the full-scale outputs of the algorithm. Once this is done, the uncertainty of the solution needs to be assessed. This is discussed in Subsection 7.3.1. Once the state uncertainty information is obtained, the probability of collision can be calculated. The different algorithms considered for this step are discussed in Subsection 7.3.2.

7.3.1. Uncertainty Estimation

As mentioned previously, the neural network is used for satellite orbit prediction. One disadvantage of this method is that the uncertainties of the model are not propagated. In such a case, the uncertainty should be estimated some other way.

A classic approach to this would be to perform a pseudo-Monte Carlo analysis. By feeding the neural network with a series of initial states (selected according to a certain distribution which depends on the accuracy of the tracking method), an uncertainty ellipse at each time step could be obtained. However, there is an additional source of uncertainty with neural networks, which is the one resulting from model simplifications.

Since the real state of the satellite is known, as it is the dataset that is used to train the model, the residuals of the NN predictions can be calculated:

$$\boldsymbol{r}_i = \boldsymbol{y}_{true,i} - \boldsymbol{y}_{predicted,i} \tag{7.3}$$

With this, the standard deviation of the residuals for each element in the state of the satellite can be calculated, which is indicative of the uncertainty in each variable:

$$\boldsymbol{\sigma} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{r}_i - \bar{\boldsymbol{r}}_i)^2} \tag{7.4}$$

In this formulation, $\bar{r}$ represents the mean of the residuals and N is the number of data points in the dataset. Equation (7.4) results in a vector with the standard deviation in each component of the residuals. Without accounting for correlations between elements, the covariance matrix in ECI Cartesian coordinates resulting from this rough estimation would follow the form:

$$\boldsymbol{P} = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_z^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{vx}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{vy}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{vz}^2 \end{bmatrix} \tag{7.5}$$

As such, the uncertainty estimation has two components: the uncertainty resulting from the imperfect knowledge of the initial state and the uncertainty resulting from the model's errors.

Interval Analysis

A method to reduce the uncertainties of the model is that of the **Intersection Method** (Römgens, 2011). This method relies on the fact that the combination of multiple independent estimates of the same thing (e.g., different predictions of the satellite's position with different state representations), can reduce the uncertainties in the solution.

Imagine that, due to uncertainties, each point in the trajectory of the satellite can be represented as a box where the nominal trajectory is a point in the centre of it. Each side of this box runs parallel to the axes in the three-dimensional space, and the length of each is directly related to the uncertainty in its corresponding axis. Now imagine that the trajectory of a satellite is predicted with two different sets of state representations, such as Cartesian coordinates and USM7. This creates two independent sets of estimates for the state of the satellite. Because of this, at each point in this trajectory, there are two "boxes": one created by the predicted Cartesian state and corresponding uncertainties and another by the predicted USM7 state, as shown in Figure 7.4. In this figure, both boxes intersect in the red area, and the "true" solution is assumed to be inside of this intersection, thus effectively reducing the uncertainties in each axis.

7.3.2. Collision Probability Calculation

The ultimate objective of the work performed in this thesis is that of collision detection. The orbit prediction is but a step in this whole process, and thus it is necessary to ultimately calculate the collision probability using the satellite state and covariance information. Several algorithms can be used for this calculation, and choosing one that is suitable for the problem at hand is key for this project.

A simple solution would be to use the method derived by DeMars et al. (2014), suitable for Gaussian Mixture Elements. This was used by Leon Dasi (2021) for the DA-GMM algorithm and its formulation was shown in Equation (2.1) and Equation (2.2). This is a three-dimensional method based on the time integration of the collision probability rate to obtain the total P_c and has proved to be very reliable. However, the algorithm for P_c would need to be changed since the work performed for the PINN does not include GMEs. A way to do this is to trace the expression by DeMars et al. (2014) back to the original formulation of Coppola (2012):

$$P_c = R^2 \int_{t_0}^{t_0+T} \int_0^{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} p_g\left(R\hat{\boldsymbol{n}}_k; \boldsymbol{\mu}_{r,t}^{(ij)}, \Sigma_{r,t}^{(ij)}\right) \nu(\hat{\boldsymbol{n}}) cos\theta d\theta d\phi dt \tag{7.6}$$

where all symbols hold the same meaning as established in Subsection 2.2.3. With this formulation, all that is needed is the state of both RSOs for a series of time steps, their corresponding covariances, and their combined radius.

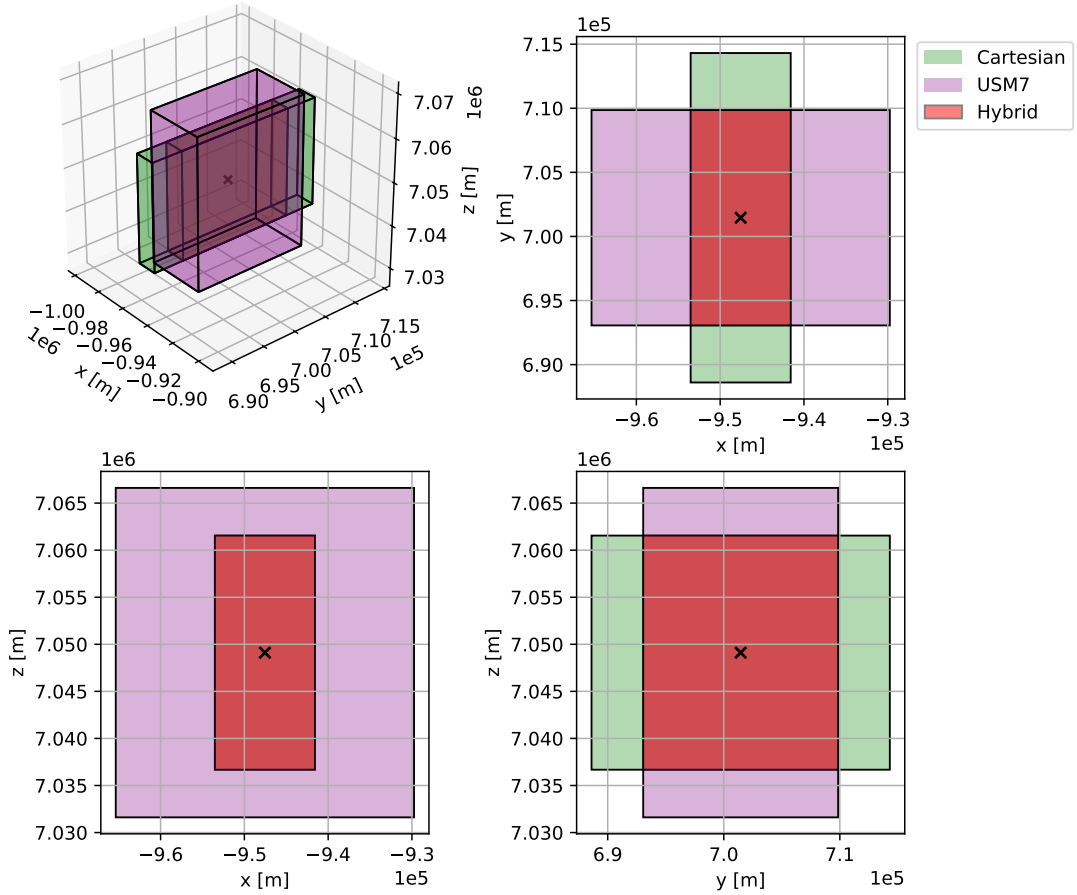


Figure 7.4: Orthogonal representation of the Cartesian and USM7 state of a satellite, with their corresponding uncertainties, and the intersection between both

7.4. Verification and Validation

In the context of research, the verification and validation of software models are key to assessing their performance. They are needed to ensure that the model works correctly and as intended, as well as to find the limits of what the model can achieve. The tests performed for the verification of the PINN are included in Subsection 7.4.1, whereas Subsection 7.4.2 introduces the tests needed for the validation of the same model.

7.4.1. Verification

The model verification step is intended to assess that everything works correctly. It is mostly composed of unit and integration tests, where every small part of the code is tested and it is verified that the inputs and outputs of each part are as required. Tests for the integration of all of these components are also needed to check the model as a whole. All the tests performed for the three main blocks of the PINN are introduced below: firstly for the data pre-processing, then for the PINN architecture, and finally for the data post-processing.

Data Pre-processing Verification

The first step in the data pre-processing is the generation of the datasets. The approach taken for this work is to obtain real satellite data from the Space-Track API. A series of integration tests need to be performed, which include the correct authentication of the user (as an account is required to access their collection of data), the verification that the data downloaded corresponds to the request, and that the API can be integrated with the rest of the code. These are summarised in Table 7.4.

Table 7.4: Space Track API integration tests

ID	Test	Expected Output	Pass/Fail
ST-VT-01	Test Authentication	User can access Space Track data	Pass
ST-VT-02	Test File Structure	Retrieved file has the standard TLE format	Pass
ST-VT-03	Test File Content	Retrieved file contains the correct information corresponding to the established input	Pass
ST-VT-04	Test Integration	API can be integrated with the data processing block	Pass

Table 7.5: Element Transformation unit tests

ID	Test	Expected Output	Pass/Fail
ET-VT-01	Kepler elements to Cartesian coordinates	Equals output from Tudat	Pass
ET-VT-02	Cartesian coordinates to Kepler elements		Pass
ET-VT-03	MEE to Kepler elements		Pass
ET-VT-04	Kepler elements to MEE		Pass
ET-VT-05	USM7 to Kepler elements		Pass
ET-VT-06	Kepler elements to USM7		Pass
ET-VT-07	USM7 to Cartesian coordinates		Pass
ET-VT-08	Cartesian coordinates to USM7		Pass

Table 7.6: Data Pre-processing dataset generation unit tests

ID	Test	Expected Output	Pass/Fail
DS-VT-01	Test Dataset loading	The data is loaded efficiently	Pass
DS-VT-02	Test Data Normalisation	$\bar{\boldsymbol{x}} \in [0 \quad 1]$	Pass
DS-VT-03	Test Dataset Split	The dataset is rightfully split	Pass

Since the data used to train the model is a hybrid of real and synthetic data, once an initial state is obtained from Space-Track, a propagator is used to generate the datasets. The one used for this project is Tudat, which has already been verified in Section 5.4. Since the generation of these datasets can be done with different state representations, some element transformations might be needed for this. Even if Tudat does have some element transformation functions, some were developed as part of the code since these will be needed for the data post-processing. It was preferred if the PINN and post-processing sections of the model were independent of Tudat to make it more accessible to people. All the transformations were then coded and verified with the corresponding outputs from Tudat. The tests are summarised in Table 7.5, where all transformations coded are included.

Different pre-processing techniques were discussed in Section 7.1, such as the dataset loading into the PINN architecture, the normalisation of the data and the data split. All of these are tested for their correct functioning and integration with the rest of the model. These tests are included in Table 7.6. The dataset split and normalisation are visualised for proof of their correct modelling in Figure 7.5, where it is indeed observed that the normalised data falls within the desired range and that the split is performed successfully (i.e., there are no discontinuities between the different sub-datasets).

Physics-Informed Neural Network Verification

When it comes to the block of the model containing the neural network, there are few tests to be done. Since the dataset generation is verified using the tests in Table 7.6, and the external software, Py-torch, is also verified in Table 5.6, the overall prediction accuracy of the PINN is the only aspect to be tested. The neural network will be trained using the data for two days of the Iridium-33 satellite. With this, a part of the dataset will be selected to be the test set with which to assess this accuracy. Some fine-tuning needs to be done to understand how the neural network works and what can be done to reduce the loss of the model and thus increase the accuracy of the prediction. For the state representation selected to

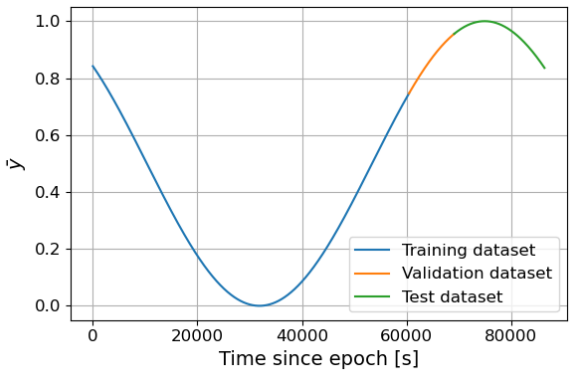


Figure 7.5: Sine wave normalised and separated into training, validation, and test datasets

Table 7.7: Data post-processing unit tests

ID	Test	Expected Output	Pass/Fail
PP-VT-01	Test denormalisation	Function provides the original dataset	Pass
PP-VT-02	Test uncertainty estimation	Function constructs the covariance matrix as desired	Pass
PP-VT-03	Test interval analysis	Function returns the intersection between two estimates	Pass
PP-VT-04	Test new P_c algorithm	Results by DA-GMM algorithm for 1 GME	Pass

train the PINN, the error shall be kept under 1% for all the elements that compose it. This has been studied and shown in Subsection 7.2.1 and the results will be discussed in Chapter 8.

Data Post-processing Verification

The process of transforming the raw data from the neural network into a dataset that is usable and can finally be used to obtain the collision probability between two objects is the final step of the developed model. This process initially requires some denormalisation of the data to match the scale of the original dataset, which needs to be tested. The next step is to verify that the covariance matrix is constructed as expected using following the uncertainty estimation and interval analysis of the residuals of the PINN. Finally, the code used for the collision probability must be verified since it was adapted from the one used by Leon Dasi. To ensure that the new P_c algorithm works correctly, Case 7 from Alfano (2009) is run with 1 GME in the DA-GMM algorithm. The state and covariance at each time step of the encounter are extracted and run firstly with the original P_c algorithm and then with the one not accounting for the GMEs. These situations are completely identical and so both algorithms should provide the same result.

Table 7.7 shares an overview of the verification tests performed for the post-processing block of the PINN. As shown, all the unit tests are passed.

7.4.2. Validation

After careful verification of all the components of the model, it remains to be analysed whether it can be used for its intended application and what its limitations are. This all falls into the validation of the model.

Ideally, the model should be validated by comparing its results to real-life data, to check whether it can be used in the scientific community or if it fails to reach the standards set by other models in the same research field. To do this, the Cosmos-2251/Iridium-33 case is selected. This collision is ideal for testing since it was also used to validate the DA-GMM. Since the latter model has been used in this thesis, all the data regarding the collision (state propagation, uncertainty propagation, and collision probability) is available for comparison and thus shall be used. On top of that, since the initial state

for both satellites was obtained from TLEs, this case fits the overall architecture of the PINN. Due to time limitations, this is the only collision scenario that will be used for validation purposes. Training the model requires a significant amount of time, especially for the level of accuracy that such an application demands.

Another test that can be done with the trained model is to test the lead time with which the neural network still provides an accurate enough answer for collision prediction. For simplicity reasons, a synthetic crash scenario can be generated where the initial state of two intersecting satellites is known accurately.

Neural networks are known for their generalisation capabilities. Because of that, several different satellites will be tested with the Iridium-trained network. The process shall be as above: the initial state will be obtained from a TLE, a dataset generated with a highly accurate propagator, and the state predicted with the trained neural network. This predicted state will be tested against the original data to assess its accuracy. Using different satellites from a range of altitudes, eccentricities and inclinations, the limitations of the model can be analysed. With this, recommendations will be given for future work on a generic tool for orbit prediction, mostly whether this is feasible or if different networks shall be used for different satellites.

Firstly, it would be interesting to test the generalisation capabilities with a satellite with similar characteristics to Iridium-33. For convenience, Cosmos-2251 will be used for this initial test. Afterwards, satellites at different heights will be chosen to assess the influence of the period on the prediction error. It was seen in Figure 2.4 that the most crowded region in LEO is the one that comprises heights around 400-500 km since that is the height of the Starlink constellation. Thus, it should be assessed whether the neural network is able to predict the state of any satellite in this constellation. The satellite is selected with a random number generator. On top of that, and for application purposes, a satellite from the GPS constellation, usually at a height of around 20,200 km and with circular orbits, will also be tested. Finally, the generalisation capacities of the model will be assessed with a satellite in GEO to understand the full scope of the prediction error over different altitudes. The satellites in the MEO and GEO regions have been selected based on the knowledge of their specifications since that will facilitate the orbit propagation process.

The list of selected satellites to test for the altitude of their orbit is therefore the following:

- Cosmos-2215
- Starlink-4437
- NAVSTAR 71
- SES 17

Once there is a higher knowledge of how altitude affects the prediction capabilities of the neural network, the effect of the eccentricity will be tested. For this, a Molniya orbit will be used. These orbits are characterised by their large eccentricities (some can be around 0.7) and will provide an interesting asset to test. The specifications of all satellites are included in Appendix E, including the initial state which was propagated and the mean orbital parameters extracted from their TLEs.

8

Results

After the model presented has been fully developed, tests need to be made to verify and validate it. This chapter presents the results of the application of deep learning algorithms for collision detection, using a series of tests to assess the model's accuracy, reliability across different orbital geometries, and sensitivity to initial state uncertainties. Section 8.1 presents the results for the orbit prediction of a single-satellite neural network. This section therefore focuses on the state prediction accuracy assessment, the first point that needs to be analysed before moving forward to the applicability of the model using the Cosmos-2251/Iridium-33 collision in Section 8.2. A synthetic scenario is created and studied in Section 8.3 to further investigate the uncertainty estimation and the lead time of the algorithm. Finally, Section 8.4 presents the first steps towards the generalisation of the method presented using a diverse set of testing scenarios. As such, this section assesses the approach that can be used to improve the algorithm's robustness. The combination of these tests aims to provide a profound insight into the model's strengths and limitations, offering recommendations and hypotheses on how to improve it.

8.1. Neural Network Orbit Prediction

A deep learning model has been developed in this thesis to predict satellite trajectories based on historical state data. This section provides the results obtained from the model and analyses its predictions by comparing them to actual state information. With this, conclusions can be drawn regarding the accuracy and performance of the developed model. This step is key for the applicability of this model for the goal of this work: collision detection.

After a thorough analysis, the final architecture for the neural network has been designed to ensure the best possible results. Table 7.3 represents the summary of the architectural choices that have proven the most optimal for the performance of the neural network. The final model is then run with this final configuration for training on the time series of data from the Iridium-33 satellite. This is done on the *DelftBlue* supercomputer for a set of Cartesian coordinates, as this was established as the state representation with the smallest error.

Two different approaches were discussed in Section 7.2: an original model aimed to help understand the behaviour of the neural network and an improved model to deal with the limitations of the original model. The results obtained from both will be explained to assess how the second algorithm deals with these limitations and how much the results improve. The original algorithm developed is assessed in Subsection 8.1.1, followed by the improved model in Subsection 8.1.2.

8.1.1. Results of the Original Model

Once the training of the original algorithm is finished, it is time to assess how well the model can generalise to future epochs. This is now done with the test part of the dataset, which is around 10% of the total data points.

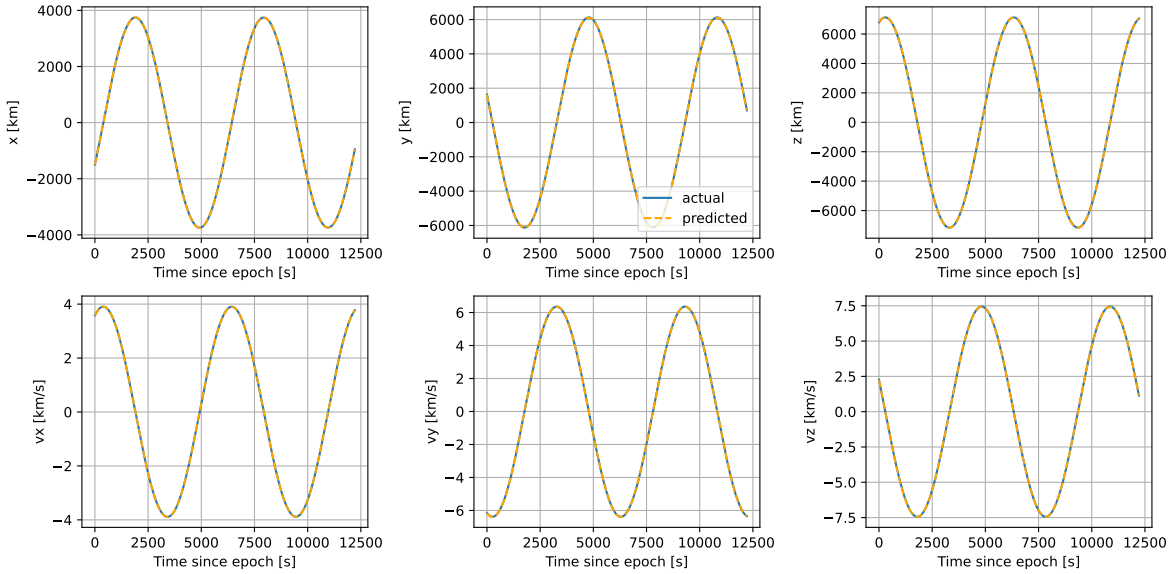


Figure 8.1: Comparison of the PINN-predicted and the actual values of the six elements of the Cartesian state in the test dataset

Until now, the test loss was used as a metric for the model's performance as it indicates how well it can predict the *normalised* state of the satellite. However, this is not enough for the analysis that needs to be carried out since the main interest of this test is how well the *full-scale* state is predicted. Figure 8.1 shows this prediction compared to the "true" state of the satellite. For all elements, both lines overlap, indicating that the difference between both states is significantly smaller than its scale and thus cannot be visualised in these plots.

However, and as discussed in Section 7.2, these results correspond to a model which takes 20 data points to predict the state of the satellite at the next epoch and does so repeatedly with the "true" values of the satellite's state. In reality, only the initial state should be used to test the actual predictive nature of the neural network. As such, the true state of the satellite is only needed for 20 seconds, after which the network should be used to predict the following 20 data points, which in turn would be used for the next set of predictions and so on. With this approach, the error in the neural network's output increases exponentially and the relative error in the satellite's states reaches over 1% in just 80 seconds for some of the Cartesian coordinates. The magnitude of this error showcases the poor predictive nature of the neural network and calls for a change in strategy.

A different approach must be taken to deal with the limitations observed in the built model. Since the neural network seems to work well when "true" states are used for the predictions, the architecture of the improved model will be changed so that 20 data points $(y_{t-20} \ldots y_t)$ are used to predict the state of the satellite after one hour: $\hat{y}_{t+3600}$. With this approach, the initial state of the satellite needs only to be propagated for one hour and the network is expected to be more stable for a much longer period of time. The results obtained from this improved model are discussed in the next section.

8.1.2. Results of the Improved Model

With the architecture changed, the neural network is re-trained. Once this process is completed, the prediction error on the test dataset will be analysed, firstly for the difference in the position induced by each element and then for the combined error of all the elements. Considering that the Interval Analysis requires two sets of estimates, the model will also be trained for the same dataset and split but with the USM7 elements. The collision probability is calculated using position and velocity, so the differences between the predicted and actual test datasets induced by the prediction error in the USM7 elements will be calculated. This way, it can also be understood how a difference in each USM7 element affects the inaccuracy of the Cartesian state prediction. In previous sections, it was found that

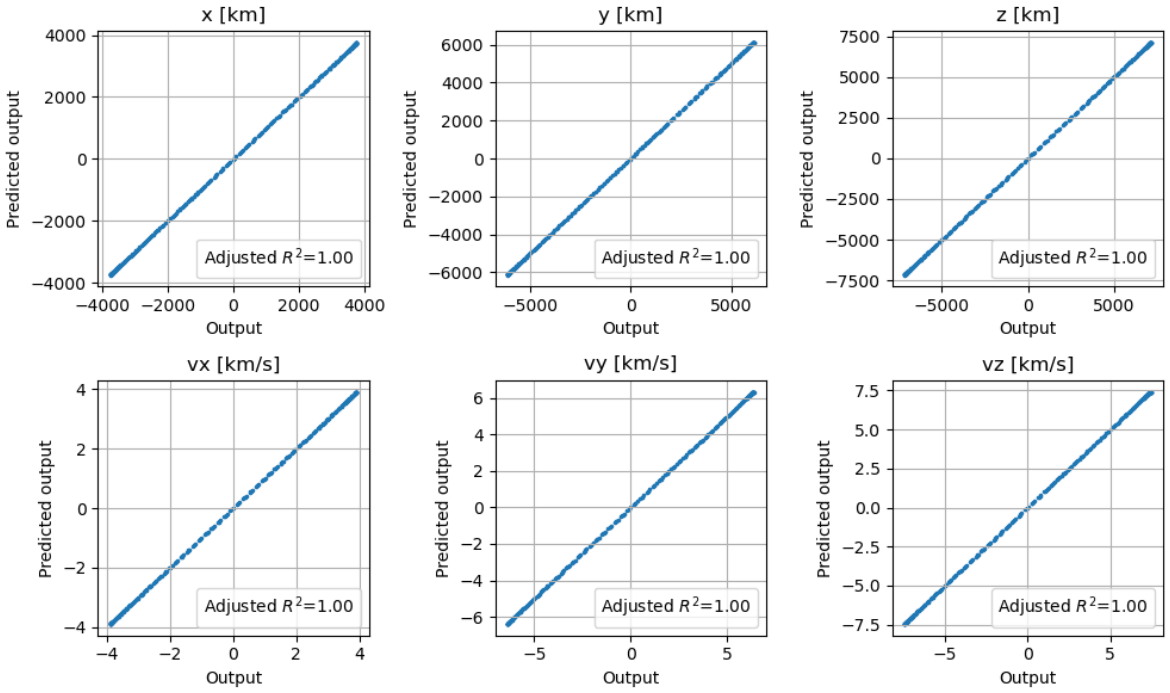


Figure 8.2: Predicted satellite state plotted against the actual state for each Cartesian element for a 16-hour propagation using a neural network

the Cartesian state was predicted with lower training and test losses, but it is unknown how an error in the USM7 will affect the Cartesian state resulting from the element transformation.

Cartesian Coordinates

Since the satellite used to train this neural network is Iridium-33, its state is initially propagated for 3600 seconds using a highly accurate propagator and the neural network is then used to predict the state of the satellite for the next 16 hours (since this is the time window that is required for the Cosmos-2251/Iridium-33 collision). The predicted output is plotted against the actual output in Figure 8.2 for each coordinate in the Cartesian state. The relationship seems to be linear, which indicates that the prediction errors are small when compared to the actual state values. A metric that is typically used in statistics to quantify how much a model's predictions match the actual data is the **adjusted $\mathbf{R^2}$ score**, explained more in detail in Section C.5. The closer this coefficient is to a value of 1, the better the model's predictions are said to be and the more suitable the features selected for the regression task are. From Figure 8.2, it is noticeable that the R^2 score in every coordinate differs extremely slightly from a perfect score, further reinforcing the accuracy of the model.

It was observed that, for the original model developed, the relative error in the prediction reached 1% after only 80 seconds of propagation using the trained neural network. How does the improved model perform when compared to the original one? The absolute error in each state component is now calculated and plotted in Figure 8.3, where the error shows a periodic nature with an increase in its amplitude with respect to the propagation time. On top of that, the periodic wave of the error is accompanied by some noise, which also seems to increase with the propagation time. When discussing the magnitude of these variables, it can be seen that the propagations still yield an error in the order of 10^1 km for some coordinates, significantly large. The fast variations in the state and the huge magnitude of the normalisation factors make it more challenging for the neural network to make accurate predictions, although this could always be solved by increasing the complexity of the architecture. Regardless, the error in each component remains lower than that of the original model after 16 hours of propagation, a much more stable and accurate result, suitable for longer predictions.

To understand the nature of the errors resulting from the neural network, the residuals of the predictions over one hour are plotted in a histogram in Figure 8.4. To improve the accuracy of the results

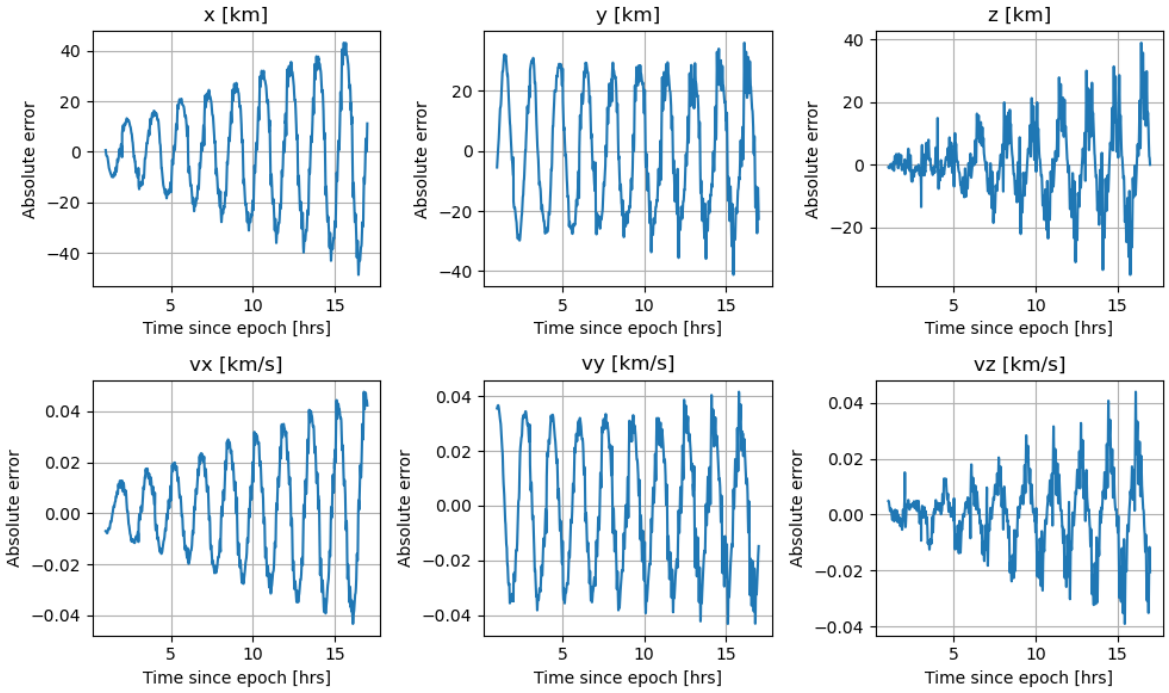


Figure 8.3: Absolute error of the different predicted state elements with respect to their actual values for a 16-hour prediction using the improved architecture

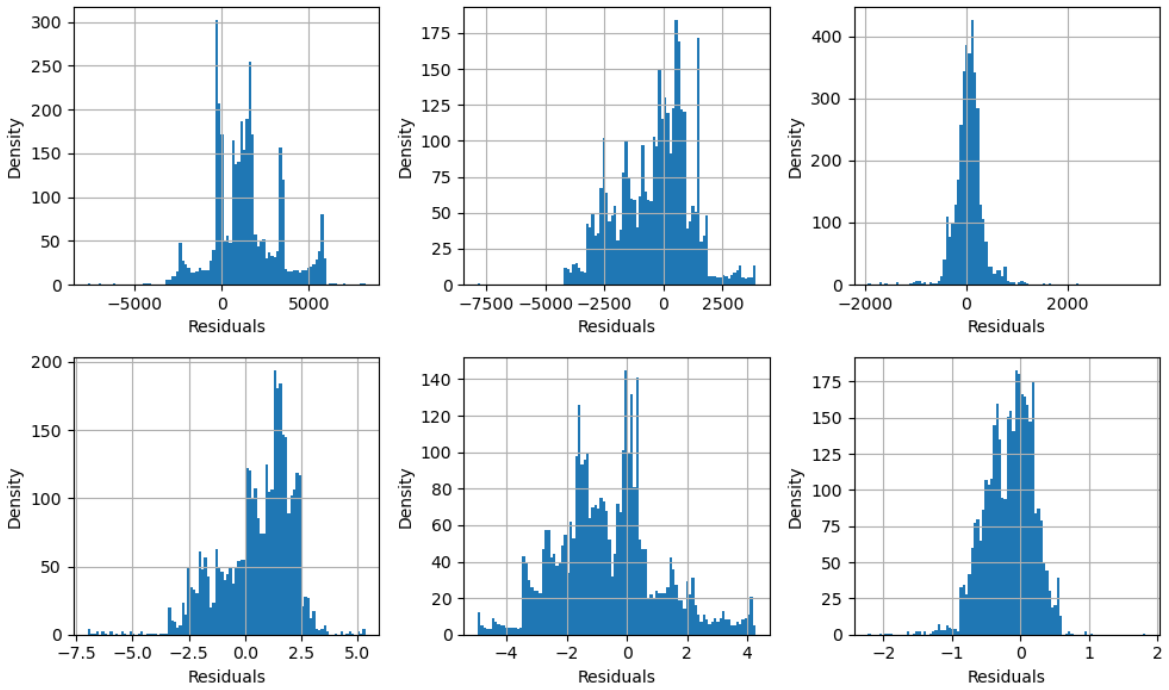


Figure 8.4: Density of the residuals for the different elements of the Cartesian state for a propagation of one hour using the neural network approach

from the neural network, two aspects are considered for the uncertainty estimation: the uncertainty resulting from imperfect knowledge of the initial state and that resulting from errors in the model. In some cases, especially for low lead times, the model uncertainty is expected to be larger than the one stemming from the limited knowledge of the initial state of the satellite. As such, it is important to

model both, as discussed in Subsection 7.3.1. For simplicity, the uncertainties and errors are assumed to follow a Gaussian distribution, centred around 0 and following a bell-shaped curve. So far it has been checked that there is some bias present by which the errors are not centred around 0 (although they are not far from it). If this shift was taken into account, however, the assumption of a normal distribution of the residuals can be initially used for estimations due to its simplicity and resemblance to the actual scenario observed in Figure 8.4.

The relative errors are now summarised in Table 8.1. This table shows that the test loss obtained for the position and velocity in each coordinate is about three orders of magnitude smaller than those obtained on previous tests. As a result, the error on each component can be kept to less than 1% when reverting the normalisation to bring the predictions to their full scale. On top of that, all elements seem to contribute similarly to the error in the final state. However, for a satellite with a semi-major axis of approximately 7,161 km, an error of 1% on the x, y or z components results in a final position error of 71 km, so the question now becomes: is this error acceptable for the application considered?

This can be assessed by taking a look at the errors that are propagated with the DA-GMM. This algorithm has been proven reliable for collision detection, hence the state propagation is analysed and compared to the results obtained from the PINN. The mean and maximum errors of the DA-GMM regarding the Tudat-propagated orbit are calculated and added to Table 8.1. It can be seen that, while the mean error is roughly similar for both methods, the PINN outperforms the DA-GMM regarding the maximum state error achieved. However, one of the strongest points of the DA-based method is that of uncertainty estimation and propagation, thus the applicability of the ML algorithm remains to be tested.

Unified State Model

The training process is repeated, this time for the elements of the Unified State Model. Even though it was shown earlier that the test loss is higher for the USM7 than for Cartesian, this is only related to the error in the elements in their respective state model. How the uncertainties in the USM7 affect the final Cartesian state upon transformation remains to be checked. This is one of the reasons for repeating the training process for the USM7, with the other being that these results can be critical for uncertainty reduction using the interval method.

Several tests are now run to properly assess the prediction errors for the model trained for the USM7 elements. The same performance metrics as the analysis on the Cartesian prediction error are used: the maximum error per element and the maximum and mean error of the final, Cartesian state for all the elements together. On top of that, the mean and maximum error of each element on the Cartesian state is assessed to check for the sensitivity of this state representation to each element in the USM7. All of these are reported in Table 8.1. Although the error in each feature is lower (for some, even a couple of orders of magnitude lower) than that obtained in the position and velocity, the final mean and maximum errors upon transformation are larger. Concretely, it can be seen that the driving features in the final state error are each of the quaternions, for which even a small uncertainty of 0.03% can lead to a position mistake of 1.4%. This is probably due to the interdependency of the quaternions, meaning that a miscalculation in one element causes an error in all of them. On the other hand, it seems like the hodograph parameters do not have a large influence on the mean and maximum error on the final state. This indicates that, due to the difference in scales and difference in the impact of each element, when training a neural network with USM7 the networks for each element can differ in architecture. The NNs used for the quaternions should make sure to provide relative errors at least a couple of orders of magnitude smaller than the ones studied here if the final combined error is to be kept smaller than 1%. The NNs used to train the hodograph parameters can use a more relaxed architecture, which also would take less training time.

With a maximum error of 2.12%, over double that obtained with the Cartesian coordinates, and the large sensitivity of the transformation to uncertainties in the quaternions, the USM7 is less suitable for the prediction of orbits for collision assessment. Regardless, this final error is on the same order of magnitude as the one obtained with the DA-GMM and proves that this is still a good result. On top of that, it means that it is suitable for uncertainty estimation and reduction using the interval method.

Table 8.1: Summary of the errors obtained from the analysis of the neural network for both the Cartesian and the USM7 states

Cartesian Coordinates				
Element	**Test Loss**	**Max. error on element [%]**	**Mean error on final state [%]**	**Max. error on final state [%]**
x	$1.16{\cdot}10^{-7}$	0.501	0.154	-
y	$3.13{\cdot}10^{-7}$	0.901	0.237	-
z	$1.90{\cdot}10^{-7}$	0.874	0.192	-
vx	$1.49{\cdot}10^{-7}$	0.505	0.138	-
vy	$2.16{\cdot}10^{-7}$	0.747	0.191	-
vz	$1.56{\cdot}10^{-7}$	0.932	0.211	-
Combined error on final state			0.399	0.994
DA-GMM error on final state			0.403	1.81
Unified State Model - Quaternions				
Element	**Test Loss**	**Max. error on element [%]**	**Mean error on final state [%]**	**Max. error on final state [%]**
C	$7.68{\cdot}10^{-7}$	$3.92{\cdot}10^{-4}$	$3.22{\cdot}10^{-4}$	$8.07{\cdot}10^{-4}$
Rf1	$2.96{\cdot}10^{-7}$	0.484	$1.67{\cdot}10^{-4}$	$4.57{\cdot}10^{-4}$
Rf2	$1.37\cdot10^{-6}$	0.262	$1.99{\cdot}10^{-4}$	$8.01{\cdot}10^{-4}$
q0	$2.90{\cdot}10^{-7}$	0.395	0.234	1.78
q1	$1.20{\cdot}10^{-7}$	$9.28{\cdot}10^{-2}$	0.181	1.79
q2	$1.01{\cdot}10^{-7}$	0.192	0.165	1.68
q3	$3.24{\cdot}10^{-8}$	$3.32{\cdot}10^{-2}$	0.178	1.41
Combined error on final state			0.568	2.12

8.2. Cosmos-2551/Iridium-33 Collision

The work developed in this thesis aims to use the prediction in the state of a satellite for collision probability estimation in a faster and more accurate manner than other methods currently used. Since the DA-GMM has already proved to be a reliable method for collision assessment (Leon Dasi, 2021), all answers obtained will be compared to this model. The previous section has already proved that the state of a satellite such as Iridium-33 can be predicted with a trained neural network with a smaller error than the DA-GMM. However, whether this model can perform well in a real-life scenario remains to be seen.

The Cosmos-2251/Iridium-33 collision was chosen to validate the model. Subsection 6.3.2 already introduced this event and all the required details to understand the work developed next. Since there was already a neural network trained for the elements of Iridium-33 for the time leading to this crash, only the state of Cosmos-2251 remains to be predicted. The current work has only considered the use of one neural network for a single satellite, this procedure will be repeated and a set of algorithms will be trained on Cosmos-2251 data. The generalisation of the neural networks to other satellites will be studied in Section 8.4 and will not be considered for now. Using the same configuration as Iridium-33, with hyperparameters summarised in Table 7.3, a neural network is trained to predict the state of Cosmos-2251.

With the state of Cosmos-2251 predicted for the time leading up to the collision, this section studies the applicability of the neural network for collision assessment. To do so, the accuracy of the collision probability is discussed in Subsection 8.2.1, where a comparison to the calculations obtained with other algorithms is also included. Since an increased accuracy is not the only aspect that is studied, the computational load required for the orbit prediction using a PINN and for P_c calculation is also studied and compared to these algorithms in Subsection 8.2.2. Finally, the complexity of the simulation environment required for accurate predictions using the neural network is analysed in Subsection 8.2.3.

8.2.1. Collision Probability Calculation

The prediction of the state is performed on the 12,000s prior to the collision. The interval method is then used on the residuals for the PINN for both the Cartesian state and the USM7 and the uncertainties in

Table 8.2: Collision probability predicted with the PINN and with the DA-GMM algorithm for the Cosmos-Iridium collision

Method	$\mathbf{P_c}$
DA-GMM	$9.433 \cdot 10^{-7}$
PINN	$5.226 \cdot 10^{-7}$

the predicted state are calculated. These uncertainties, however, do not reflect the actual uncertainties in the initial state or those resulting from the uncertainties in the environment. As such, different initial states (as done in a Monte Carlo analysis) are tested with the neural network and it is observed that the state uncertainty resulting from the error in the prediction dominates over the initial state uncertainty and uncertainty propagation. The satellites are modelled as spheres with a combined radius of 5 m (the same value as selected by Leon Dasi, chosen for comparison purposes) and with the information of the state of both satellites at each instant of time, as well as their uncertainties, the collision probability can be calculated: the value obtained with the PINN is $P_c \simeq 5.22553 \cdot 10^{-7}$. While being almost half of the value estimated by the DA-GMM, it is on the same order of magnitude meaning that the collision can be predicted with a similar accuracy. On top of that, the maximum probability rate is obtained at the same instant in time as the DA-GMM, less than a second off the actual TCA.

The P_c estimated with any of the two methods is still not large enough for this event to be considered high risk (typically for events with $P_c > 10^{-4}$), even though several factors lead to the low value calculated here. The initial state of both satellites was obtained directly from the last TLE posted prior to the collision, about 17 hours before. It is known that any state being estimated from TLEs and the SGP4 propagator has a large uncertainty that needs to be considered. This uncertainty becomes larger when propagated, and due to the *collision probability dilution*, the probability of collision becomes smaller as the uncertainty grows, leading to the incorrect labelling of events (Balch et al., 2019; Sanchez and Vasile, 2020). It would be expected in this situation that, with a more accurate knowledge of the initial state and a better uncertainty estimation and modelling, this prediction would become large enough for this event to be classified as high-risk. It is restated that SOCRATES (Satellite Orbital Conjunction Reports Assessing Threatening Encounters in Space), the software developed by the Center for Space Standards & Innovation (CSSI) for orbit collision detection, also failed to label this event as high-risk.

8.2.2. Computational Load

It has been seen that the prediction of the collision probability is on the same order of magnitude as that calculated by the DA-GMM. However, the accuracy of the prediction is not the only aspect that matters in this research, also the computational load of the method is something to be assessed.

The main motivation in this work for using a machine learning approach to collision assessment is a reduced computational load compared to other methods. Monte Carlo analyses, still extensively used in this field, take a large amount of time (in the order of several hours) to produce results due to the large number of simulations required for accurate results. The use of complex numerical models to propagate the orbits as accurately as possible asks for new methods that can reduce the computational load required for collision detection. Neural networks have been used extensively in the last years to overcome this problem since it is known that they, once trained, can predict the state of a satellite at a fraction of the time required for other methods such as Monte Carlo analyses. The DA-GMM already solved this problem since it has a faster propagation of the uncertainties, however each GME takes a while to be propagated. Now it is time to assess how much faster (if any) the model developed here is with respect to the DA-GMM.

There are two main aspects to be assessed here: the computation of the orbit prediction and the collision probability calculation. The latter is added here since the use of several GMEs and the formulation by DeMars et al. (2014) implies that, at each time step, the integral needs to be calculated N^2 times, where N is the number of GMEs used (assuming it is equal for both satellites). With the formulation used here, this integration only needs to be performed once, and it would be interesting to assess the computation time of the entire process, instead of just a part of it. The results are shown in Table 8.3, where the DA-GMM was run for both 37 and 51 GMEs since they are the two models most used by Leon Dasi for real collision events. As expected, the PINN is much faster than the DA-GMM.

Table 8.3: Comparison of the computational load of the PINN compared to the DA-GMM with 37 and 51 GMEs

ORBIT PROPAGATION			
Nº GMEs	**Time DA-GMM [s]**	**Time PINN [s]**	**$\frac{\text{Time DA-GMM}}{\text{Time PINN}}$**
37	3,063	6.70	457
51	3,861		576
COLLISION PROBABILITY CALCULATION			
Nº GMEs	**Time DA-GMM [s]**	**Time PINN [s]**	**$\frac{\text{Time DA-GMM}}{\text{Time PINN}}$**
37	13,742	1.34	10,240
51	18,700		13,935

Where the latter model takes over an hour for the orbit propagation using DA, the trained PINN takes about 6.7 seconds, about 500 times less, for a similar state prediction accuracy as shown in previous sections. This proves that the PINN provides a good trade-off between computation time and accuracy for orbit prediction when compared to current methods. It still cannot compare to the accuracy level of highly accurate propagators, but with a more thorough analysis and training, it could easily get to their level. Regarding the collision of probability, the computational time required to calculate this parameter for roughly 200 time steps for two satellites modelled with 51 GMEs is 18,700 seconds (over 5 hours!). With the estimations from the PINN-based model, this time taken for the same amount of steps is only 1.342 seconds, nearly 14,000 times faster. Even if a simpler model was used, with 37 GMEs for example, the PINN still provides a much faster process than the DA-GMM (almost 460 times faster for the orbit propagation and almost 14,000 times faster for the P_c calculation). For the reader's information, this is all calculated with Matlab and could be optimised if another faster programming language was used, especially for the code, which includes GMEs.

8.2.3. Effect of Simulation Environment

The improved model used for orbit prediction using neural networks involves using propagators to obtain satellite data for one hour. As such, the next step is to analyse the effect of the simulation environment on the resulting collision probability calculated with the model. The neural network itself is trained using data from a highly accurate propagator, with the perturbing models discussed in Subsection 5.4.2 and summarised in Table 5.5. The algorithm should therefore be able to provide reliable results using less reliable simulation environments.

The algorithm has already been used with the most complex perturbation models, which resulted in a collision probability estimation of $5.226{\cdot}10^{-7}$. Next, a simplified simulation environment is used. This model follows the criteria discussed in Section 3.6 (i.e., the elimination of the perturbations that introduce larger environment uncertainties) and as such the forces included in this model are: spherical harmonics from the Earth up to degree and order 5, as well as third body perturbations from the Sun and the Moon. Including the accelerations induced in the body resulting from aerodynamic forces or the solar radiation pressure would require further knowledge from RSOs which is rarely publicly available. As such, this simplified model excludes the perturbations with larger uncertainties. The algorithm is run for Cosmos-2251 and Iridium-33 and the resulting collision probability is $P_c = 5.095 \cdot 10^{-7}$. The results from these tests are summarised in Table 8.4, with the information on how much time a 1-hour propagation takes to run each environment model also included. While smaller than the one obtained using the highly accurate model (which is expected since this is just a simplification and some error will be inevitably induced in the 1-hour initial propagation), the difference is only $\simeq$2%. On top of that, the propagation with this simplified model takes seven times less to run and does not require further knowledge of the object's physical properties. Finally, the algorithm is run with a simple Kepler orbit with no perturbations included. This case represents that with the simplest orbital dynamics and, as such, the one with the most expected errors. The P_c calculated is about one order of magnitude lower than that obtained with the most accurate model: $3.540{\cdot}10^{-8}$. Considering that the same test with the DA-GMM algorithm provides a $P_c \simeq 10^{-51}$ for the Kepler orbit case, the deep learning approach is significantly more robust to the perturbations used outside the training process.

Table 8.4: Collision probability predicted with the PINN for the Cosmos-Iridium collision for different simulation environment models

Simulation Environment	$\mathbf{P_c}$	Time spent on 1-hour propagation [s]
Highly Accurate	$5.226 \cdot 10^{-7}$	0.366
Simplified Perturbations	$5.095 \cdot 10^{-7}$	0.0529
Kepler	$3.540 \cdot 10^{-8}$	0.0121

8.2.4. Results Summary

The Cosmos-2251/Iridium-33 collision test case has been useful to draw several conclusions on the performance of the current neural network for real-life scenarios:

- The neural network can calculate the collision probability of the Cosmos/Iridium test case with the same order of magnitude as the DA-GMM.
- The approach taken in this thesis allows this process to be significantly faster than any other method used for collision avoidance (around 500 times faster for the orbit propagation than the DA-GMM and over 10,000 times faster for the P_c calculation).
- Considering that the errors of the neural network drive the uncertainty in the final state rather than the propagation of uncertainties in the initial state or environment variables, the probability of collision will be affected by the dilution of probability, meaning that it will always be lower than it could be predicted due to larger uncertainties. Regardless, it seems to work well enough for a case with an initial state obtained from TLEs, where the uncertainties are larger.
- The use of a simplified perturbations model only has a slight effect on the P_c calculated with respect to the highly accurate one, while taking significantly less time to propagate.
- Using the simplest dynamical model (i.e., a Kepler orbit) for the propagation yields a significant reduction in the probability calculated (albeit having a similar accuracy in the propagated position and velocity of the satellites). This is, however, not as significant as the reduction seen in other methods such as the DA-GMM.
- The differences observed in the P_c calculated seem to stem from the uncertainty estimation. Since this is just an estimation and not the main focus of this thesis, it is recommended to use higher fidelity models for the uncertainty propagation and calculation.

The test case discussed in this section represents the most widely known satellite collision and one of the main motivators for pursuing more accurate collision detection and avoidance tools. In using a real-life scenario, which has been extensively researched, the usefulness of the model is properly demonstrated. However, this is only a single specific case, and further research shall be done on more test cases to show that the model can be used in a wide variety of scenarios.

8.3. Collision Scenario

Previous sections have demonstrated that the neural network built can accurately predict the state of the satellite and, ultimately, provide a good estimation of the probability of collision for real-life scenarios. In this section, a synthetic collision scenario is used to gain further knowledge on how the P_c calculated using the approach followed in this thesis changes with lead time. The use of a synthetic scenario allows testing for the robustness of the neural network and how it is able to generalise to different collision cases and different dynamics. While the Cosmos-2251/Iridium-33 collision provides a real-life scenario with which to validate the model, a synthetic test case makes it simpler to create and test other geometries.

Lead time is crucial in the field of collision avoidance as it represents how much in advance satellite operators can be notified of high-risk events to plan and execute collision avoidance manoeuvres. The larger the lead time, the more can be done to prevent collisions. The PINN model has proved to provide accurate predictions over a time period of at least 16 hours for the Cosmos-Iridium collision. Now the focus shall remain on testing both shorter and longer propagation times, to assess if this method is still applicable for those time windows. The test case will be introduced in Subsection 8.3.1, and the

Table 8.5: Orbital elements of the two satellites used for the synthetic test case at the moment of the collision

Satellite	h [km]	e [-]	i [deg]	ω [deg]	Ω [deg]	θ [deg]
Ourania	773	0	93.6	45.1	301.3	62.1
Clio	773	$9{\cdot}10^{-4}$	86.4	134.9	121.3	297.9

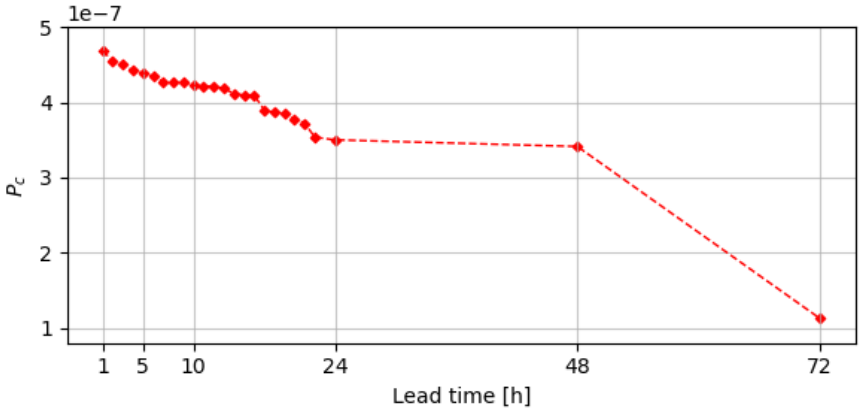


Figure 8.5: Evolution of the collision probability for a synthetic crash scenario as a function of lead time

results regarding the effect of lead time and uncertainty estimation are discussed in Subsection 8.3.2. This section will be wrapped up in Subsection 8.3.3 with the summary of the findings made from this case.

8.3.1. Case Introduction

Two satellites are created for this synthetic crash scenario: Ourania and Clio. Their positions at a random epoch are made to be identical to simulate a crash, and their states are propagated backwards for a certain amount of time to obtain different sets of initial states: 24, 48 and 72 hours. It was decided not to go any further than 72 hours since there was a risk of the errors in the propagation being too large to assess the model accurately. On top of that, the state of the satellite is retrieved every hour (up to 24 hours) to analyse how lower lead times affect the results. This is interesting to assess because the uncertainties of the model for lower lead times are ruled by the accuracy error in the neural network, while the ones at larger lead times are dominated by the propagation of the initial state uncertainty and the error propagation. The orbital elements of both satellites at the collision epoch are summarised in Table 8.5.

With the sets of initial states, the neural network is used to predict the state of Ourania and Clio at TCA and estimate their uncertainties. With this data, the P_c for each lead time is calculated. The initial state uncertainty is selected to be that of TLEs to get a simulation closer to typical real-life scenarios where the general public does not have access to highly accurate tracking data. The altitude of the satellites is selected to be 773 km, with a circular orbit, since it is closer to the geometry of Iridium and Cosmos so that the trained neural networks for that scenario could be reused.

8.3.2. Effect of Lead Time

With the simulations run, the collision probability as a function of lead time is plotted in Figure 8.5. In this figure, the collision probability gets reduced as the lead time increases, which is to be expected since the error in the orbit prediction is propagated through time and the state uncertainties increase. The predictions also seem quite constant and stable through time, where all results obtained up to 72 hours of lead time produce a P_c in the same order of magnitude.

It does seem surprising, however, that the collision probability estimated for this test case is consistently in the order of 10^{-7}, even for small lead times, when this case is simulated such that P_c=1. The

reasoning behind this is that, for lower lead times, the model uncertainty is larger than the propagated state uncertainties. As such, the uncertainties remain large enough to produce the effect known as *uncertainty dilution*, discussed in earlier chapters, by which large uncertainties inherently lead to smaller collision probabilities. This could be corrected if the neural network was made more accurate (i.e., if the model complexity was increased by increasing the number of hidden layers, the number of cells in the hidden layer, or the number of epochs over which to optimise the model). This has not been done in this thesis due to time constraints, since a model with higher complexity would require significantly longer to train and could lead to overfitting.

One surprising finding of this test case is that the use of Monte Carlo analyses for the uncertainty estimation leads to unrealistically small uncertainties, particularly for higher lead times. This is compensated for by the error modelling of the residuals, which count as the "uncertainties" of the model and also lead to more accurate predictions, as with the Cosmos-Iridium collision. Ideally, both should be compensated for, especially for larger lead times where the state uncertainties due to the initial state uncertainty are much larger than the ones estimated using a pseudo-MC analysis with the algorithm and should dominate over the error of the neural network. This will not be investigated for the duration of this thesis due to time and workload constraints, but some recommendations will be given on how to improve this.

8.3.3. Results Summary

The synthetic scenario drawn and discussed in this section has allowed for a series of conclusions that are finally summarised in this subsection:

- The model can provide stable orbit predictions for up to 72 hours of propagation, enough to detect a collision within the same order of magnitude as for a lead time of 1 hour.
- The results from the analysis of collisions using the neural network are highly affected by the uncertainty dilution phenomenon for lower lead times. As such, other algorithms (such as the DA-GMM) are probably more suitable for smaller lead times as the only uncertainty modelled is that resulting from uncertainties in the initial state.
- On the other hand, the neural network is more robust for a longer range of lead times thanks to the neural network error modelling and the stability of the predictions by the neural network. Other methods might outperform the one tested in this thesis but at the cost of a significantly larger computational load.
- The state uncertainty at each time step is modelled as the sum of the propagation of the initial state uncertainties and the uncertainty resulting from the inaccuracies of the model's predictions. At larger lead times, the propagated uncertainties are estimated to be more significant than the error induced by the model. However, these uncertainties are highly underestimated, leading to a loss of accuracy. New methods should be considered to estimate these uncertainties and make this approach more suitable for collision detection.
- Due to the error modelling and the uncertainty dilution, it is improbable that the results improve for a smaller set of initial state uncertainties. This makes this approach useful for real-life scenarios with large initial state uncertainties (as proven in the Cosmos-2251/Iridium-33 case) but might highly underestimate the collision risk for any case with a higher knowledge of the initial state due to the uncertainty dilution.

With this section, the effect of the lead time has been tested and the robustness of this method for extended periods of time has been proven. This section has also allowed for a more thorough study of the uncertainty estimation of the model and thus its flaws have been identified. When compared to a method such as the DA-GMM, which specialises in uncertainty modelling and propagation using Differential Algebra and Taylor Series Expansions, this method is largely outperformed. As such, new ways of estimating uncertainties would need to be studied. The histogram of the residuals of the network for Iridium-33 (in Figure 8.4) shows that the assumption of a Gaussian distribution of the model uncertainties might not be an accurate representation. On top of that, Leon Dasi (2021) argues that the use of GMEs to represent the uncertainties proves to be more accurate since the propagation of

Table 8.6: Orbital elements of Iridium-33 and Cosmos-2251 on February 10, 2009 at 00:00:00h

Satellite	h_p [km]	h_a [km]	e [-]	i [deg]
Iridium-33	767	799	0.00228	86.39
Cosmos-2251	754	801	0.00331	74.04

the initial state uncertainties leads to non-Gaussian distributions. Perhaps the use of a new method to estimate non-Gaussian uncertainties could be considered to increase the accuracy of this method.

The aim of using neural networks for collision detection is to find a way of improving the computational load that is required to propagate the state of a satellite and its uncertainties. It has been shown that using PINNs does indeed vastly improve the time that is required to do this when compared to the DA-GMM, but with its accuracy as it is right now, the results can only deemed to be estimates of P_c. Since the state accuracy does not have a large error (although this can always be improved at a fractional cost of the computational load, which is another benefit of using neural networks), the uncertainty estimation and propagation need to be improved.

8.4. Generalisation of the Neural Network

Up until now, the applicability of neural networks has been tested on different collision scenarios. All of the satellites chosen for this have significantly similar orbits and dynamics. On top of that, a different neural network has been trained for every satellite as the accuracy of the results was pursued instead of its ability to generalise. To make this approach more robust, it is therefore important to test how it can generalise to other satellites and whether a single neural network can be used for many different satellites.

Several tests need to be made to test which are the limit cases and to discuss the first steps to creating a more generic approach. Firstly, it is required to assess how the model can predict the state of orbits with similar dynamics. This is done in Subsection 8.4.1 with a test on the Cosmos-2251 satellite using the trained model with Iridium-33 data. In Subsection 8.4.2, satellites at different altitudes are used to assess which approach works best to make a single neural network that can generalise across different orbital regions. Finally, the effect of eccentricity in this generalisation study is analysed in Subsection 8.4.3. This is done to check how satellites in Highly Eccentric Orbits (HEO) can be studied with neural networks and how this generic tool might need to be changed to do so.

8.4.1. Generalisation to Satellites with Similar Dynamics

The first step that should be taken when testing the generalisation capabilities of a neural network trained for orbit prediction is to assess how it generalises to satellites with similar dynamics. This implies testing with satellites with similar altitudes, eccentricities, or velocities, for example, in such a way that their orbits present only small variations in their behaviour with respect to one another. This is a perfect initial test to verify that the neural network trained is not overfitted to specific satellite behaviours and it can be used for satellites with similar orbital patterns. On top of that, satellites at high risk of collision tend to share orbital regions and might present similar dynamics, making this test also critical for assessing the accuracy of a neural network in these scenarios.

All the tests for hyperparameter tuning and orbit prediction error characterisation have been done on a neural network trained with the Iridium-33 data in the hours leading up to its collision. For the analysis performed in this subsection, a satellite with a similar altitude and eccentricity is desired. The orbital data of Iridium on February 10, 2009 at 00:00:00h (UTC) is provided on Table 8.6. Since Cosmos-2251 is also in the same orbital region and has a near-circular orbit, it is the satellite selected for this test. Its orbital parameters are also summarised on Table 8.6, where they can be compared to those from Iridium-33.

Using the previously trained model, the state of Cosmos-2251 is predicted for two hours. This is then plotted in Figure 8.6, alongside the actual Cosmos state data for visualisation purposes. It is observed from this figure that the patterns followed by the different elements of the satellite state are

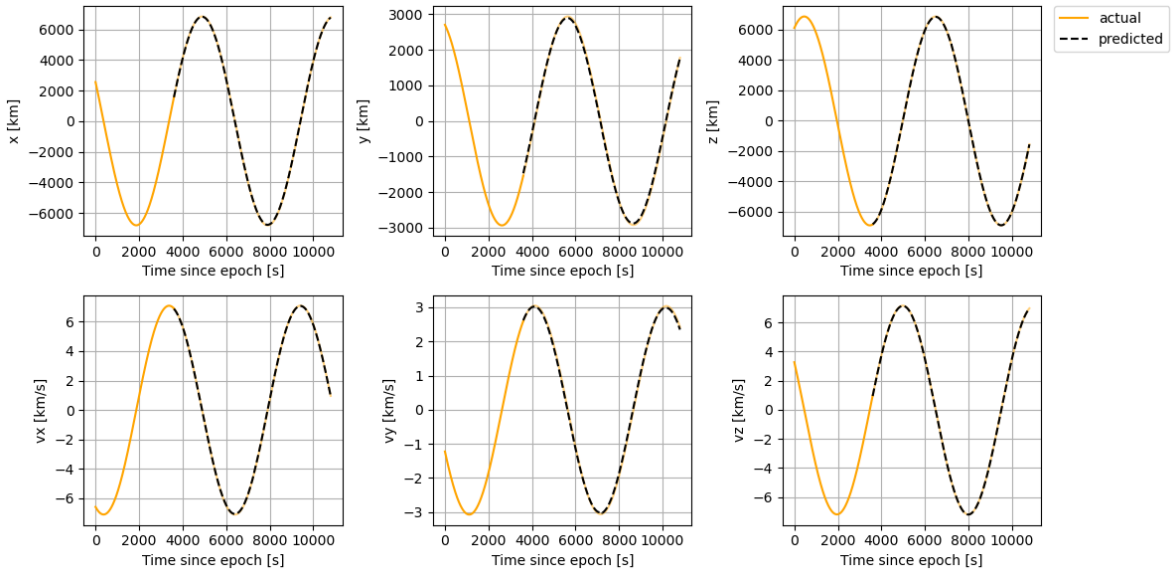


Figure 8.6: State prediction for Cosmos-2251 with a neural network trained on satellites with similar dynamics compared to the actual satellite data

predicted accurately (that is, in such a way that the errors are not visible with respect to the scale of the actual value of each component). The absolute error in each Cartesian coordinate is then quantified and plotted in Figure 8.7. This error can reach values up to 75 km for some coordinates, which is larger than was estimated for a 16-hour prediction using a single satellite and is more erratic (i.e., it has a less predictable behaviour). This increase in the prediction error is to be expected since the neural network was trained using the data from another satellite, thus small deviations from the original dataset lead to larger prediction errors. Since the errors are modelled and taken into account for the calculation of the collision probability, the model would still be able to detect a collision with this accuracy, although it would be much lower due to the uncertainty dilution. This could be fixed by training a neural network with satellite data from different satellites with similar dynamics to make it more robust and accurate for small state deviations. Regardless, it is important to highlight the capacity of the neural network to capture the overall pattern of the state evolution and therefore its capacity to generalise to satellites on the same orbital region.

8.4.2. Generalisation to Satellites at Different Altitudes

Once it has been shown that a neural network can be used for different satellites with similar dynamics using the approach discussed in this thesis, it is important to test how this model works for satellites with different dynamics. The next step towards achieving the goal of a more generic neural network would then be to assess how the model generalises to different altitudes. From a numerical standpoint, if the eccentricity is kept to near-circular the patterns behind the different element states would remain similar in that they still resemble sine waves, only with different periods. As such, it will be easier to assess their predictability using neural networks since those behaviours resemble the ones studied already.

Expanding the neural network to different altitudes is essential for real-life applications. Collisions might happen at any orbital regime and, as such, it is important to develop a single neural network that can work over a wide range of altitudes. Trajectories at different altitudes will experience different dynamics: satellites in Low Earth Orbit (LEO) are affected by a larger atmospheric drag whereas satellites at higher altitudes are more affected by solar radiation pressure. Having a model that can generalise over such a range of orbits is critical for use in the field of collision detection.

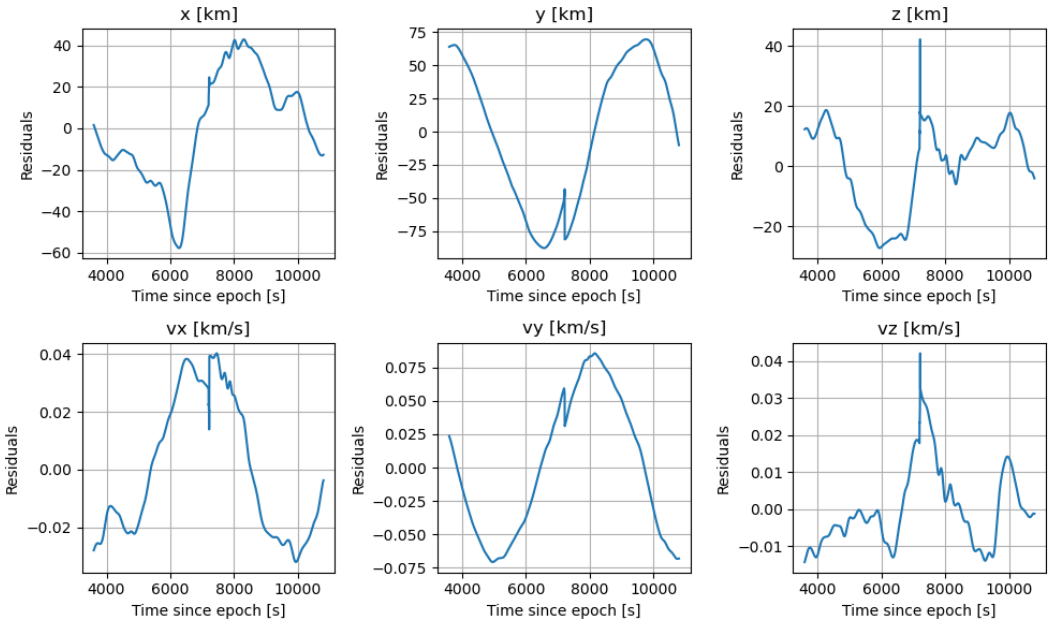


Figure 8.7: Residuals of the prediction of the Cosmos-2251 state using a neural network trained on satellites with similar dynamics

Satellites in Low Earth Orbit

A model trained on data from satellites located at similar altitudes might not be able to capture these different dynamics in a reliable manner. Let us take the trained neural network on Iridium-33 and use it on a satellite at a different altitude. For this, the satellite Starlink-4437 is selected, with TLE and physical properties introduced and discussed in Appendix E. The reasoning behind this is that it is located at the most crowded altitude (about 550 km) regime, as was observed in Figure 2.4, and thus is of high interest to get a neural network that can work for satellites at this altitude. On top of that, this satellite is part of the largest constellation of satellites (with all of them being located at similar heights and with near-circular orbits, showing very similar dynamics), and getting the neural network to work for this satellite means getting it to work on most satellites in LEO at the moment. When the trained neural network is used on this satellite, the results on Figure 8.8 are produced. As expected, the change in altitude does affect the predictions and makes the model unsuitable for satellites at different altitudes. This is also because the algorithm was trained in such a way that the state of the satellite at step $t + 1h$ is predicted. A change in the semi-major axis of the orbit (in the case of circular orbits, the altitude changes with the semi-major axis) will inevitably lead to a change in the orbital period (they are proportional due to Kepler's Third Law), and thus the predicted state at $t + 1h$ will not be a match. This is the phenomenon that is observed in Figure 8.8, where the state predicted does not match the trajectory followed by the Starlink satellite.

A new approach needs to be taken to solve this issue and provide the first step towards working on a neural network that is capable of being used as a generic tool for collision detection and avoidance. The first step to achieve this is to use a technique called *transfer learning*. This is an approach by which a previously developed model is reused for a different, yet similar, task. In this case, instead of training a new model from scratch, the model trained on Iridium-33 can be used as a benchmark for the more generic tool. This way, the training process is much faster as the new model only needs to be taught to generalise to other satellites. On top of that, a way to convey the altitude or orbital period information to the neural networks needs to be studied. The neural network is only taught the relationship between the inputs and outputs of the model, and it can be assumed that it does not know anything else. As such, including information on the orbital period might be helpful for the model to understand the differences between satellites at different altitudes. The orbital period is selected for this since this will make the model more robust for other cases such as highly elliptical orbits, since the altitude of a satellite only stays constant if its orbit is circular. The easiest way to do this is to include this as an

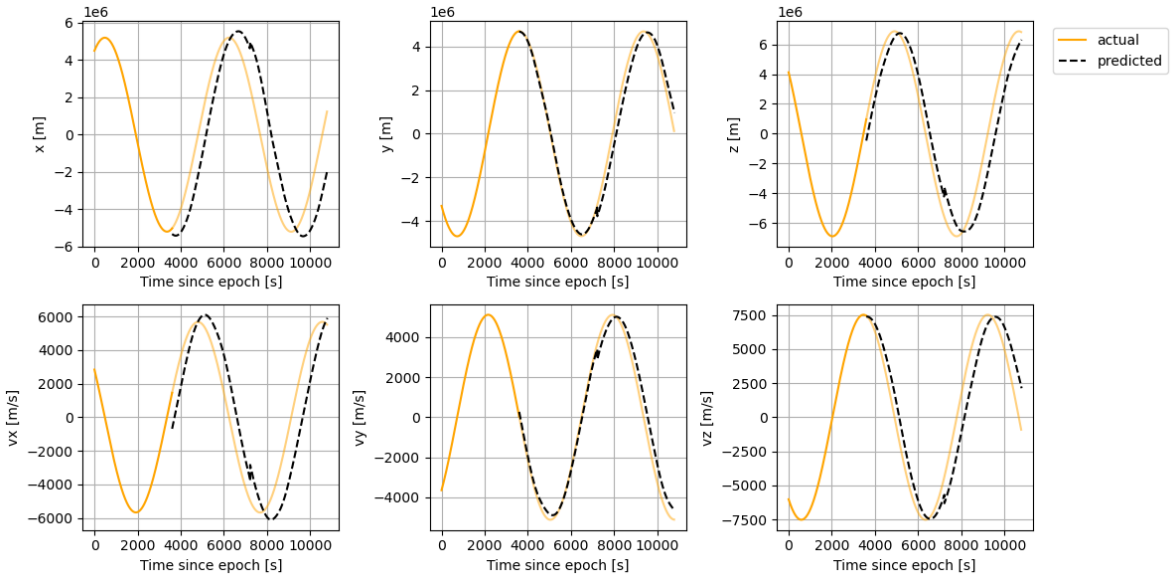


Figure 8.8: State prediction for Starlink-4437 with a neural network trained on Iridium-33 compared to the actual satellite data

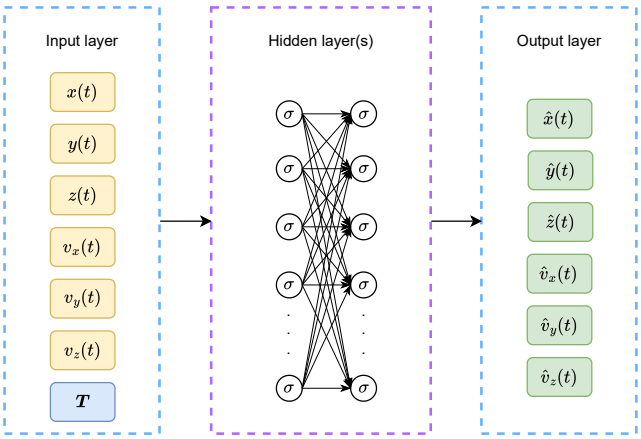


Figure 8.9: Changes on the core architecture of the neural network to include information on the orbital period of a satellite

input feature to the core architecture of the neural network. This part of the algorithm can be taken from Figure 5.3, which depicts the overall architecture of the model. Until now, the input layer of the network has consisted of six or seven elements: the components of the state representation selected. If Cartesian coordinates are selected, this input layer has a total of six input features. A representation then of how the architecture shall be re-modelled is shown in Figure 8.9, where the satellite's period ($\boldsymbol{T}$) is included in the input layer of the model. As observed, this implies an extremely simple change in the core of the neural network architecture but allows for the model to recognise which orbital region is being considered for each satellite.

With this change, and using transfer learning, the neural model trained on Iridium-33 is re-trained to include data from the Starlink satellite. Now, when this model is used to predict the Starlink data, the results are much more accurate, as can be appreciated in Figure 8.10 and Figure 8.11, where the predictions are compared against the actual state values. Figure 8.11 shows that the model trained with transfer learning is more accurate than shown for Cosmos-2251, providing a maximum position error lower than 1 % of the actual state value. Transfer learning uses the Starlink data to extend the pre-trained neural network to satellites at other altitudes, and thus works better than the network used for Cosmos, which did not contain its data but rather was trained with that of Iridium. Regardless, for

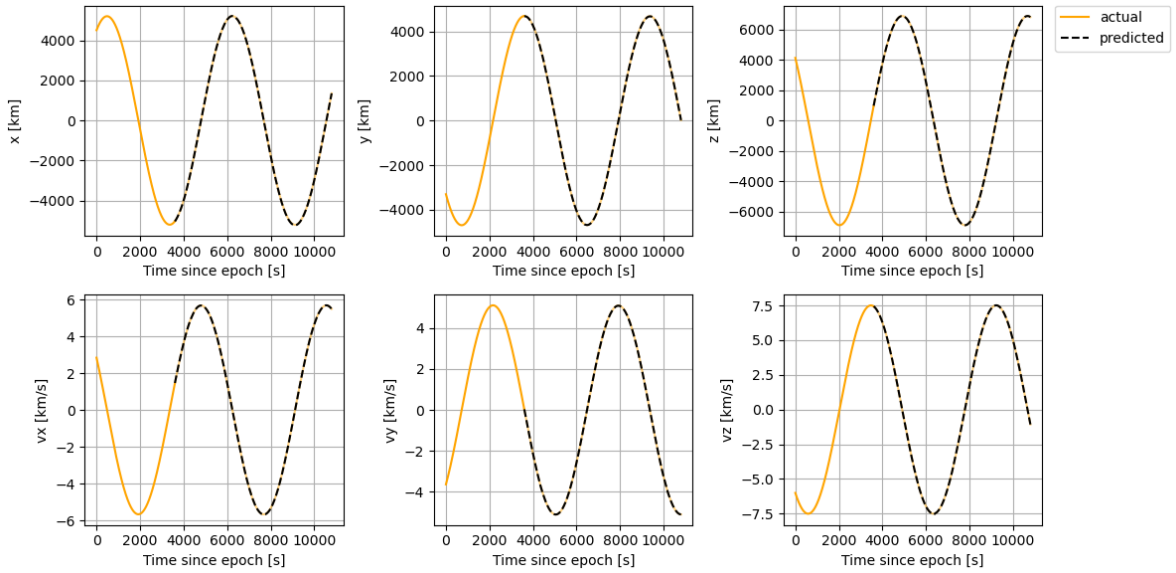


Figure 8.10: State prediction for Starlink-4437 with a neural network trained on different satellites in LEO compared to the actual satellite data

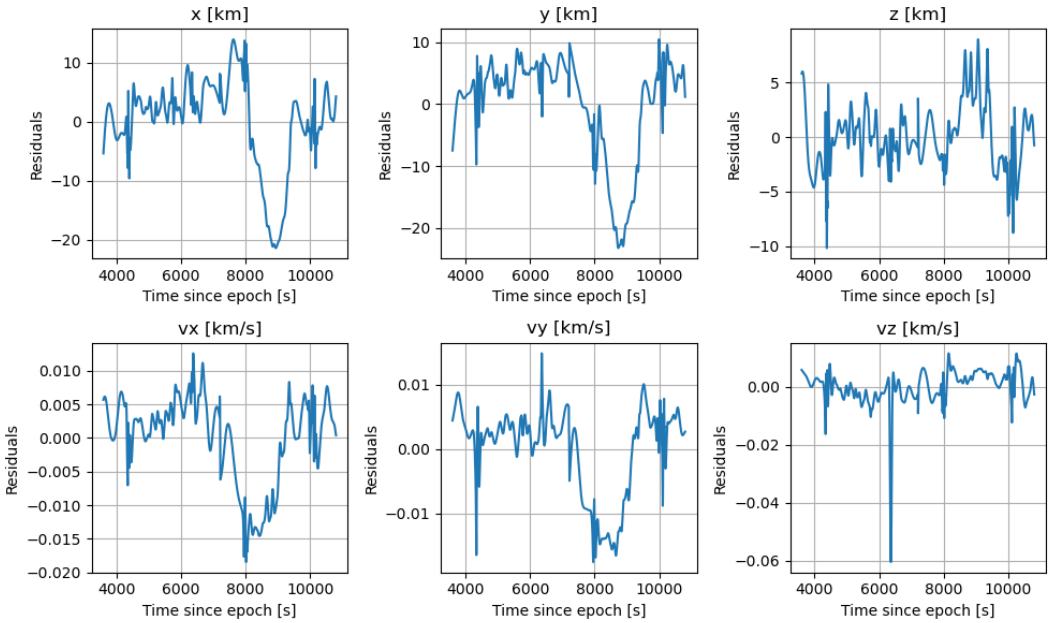


Figure 8.11: Residuals of the prediction of the Starlink-4437 state using a neural network trained on satellites with similar dynamics

all cases in which transfer learning is used in this section, the propagation time is kept to 2 hours since it will be roughly the size of the test dataset. It is important to use the test dataset for the analysis of these results since it is data that has not been used to train the neural network and thus it can be used to show how the model generalises to future epochs. When the same network is used on Iridium-33, the state is predicted with the same level of accuracy as achieved with the improved neural network in Subsection 8.1.2. As such, it can be said that the same neural network is able to predict the state of a Starlink satellite and that of Iridium-33, both belonging to the two most crowded regions in space.

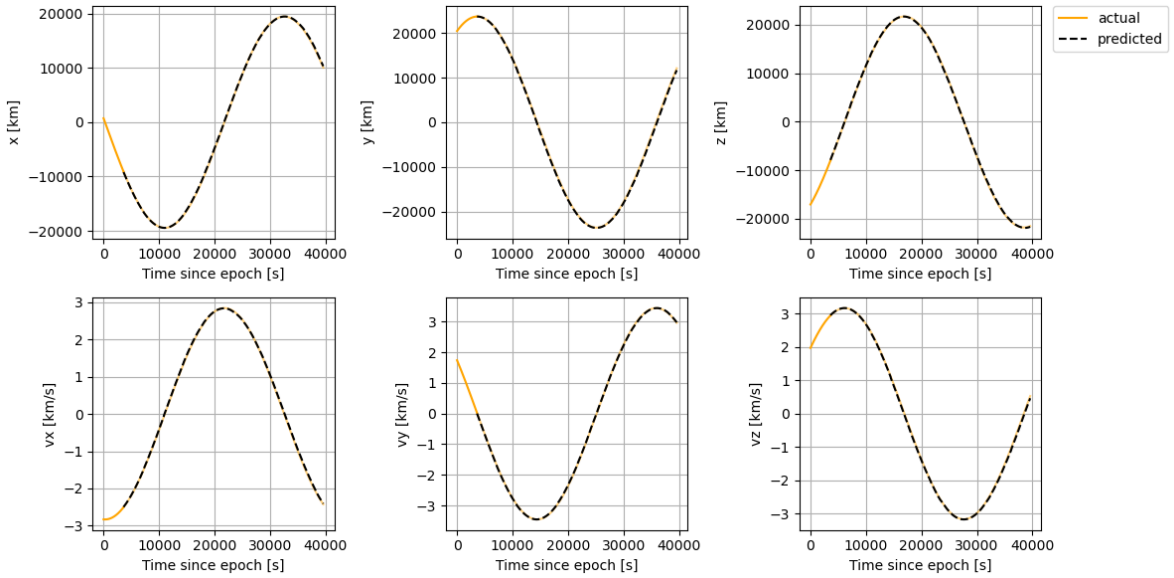


Figure 8.12: State prediction for NAVSTAR 71 with a neural network trained on different satellites in MEO compared to the actual satellite data

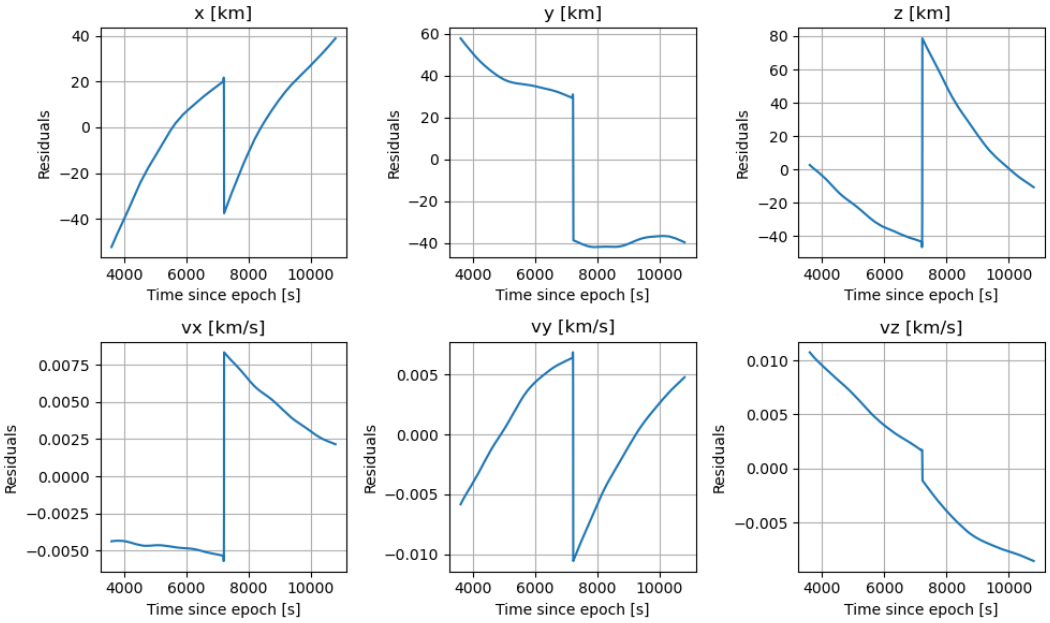


Figure 8.13: Residuals of the prediction of the NAVSTAR 71 state using a neural network trained on different satellites in MEO

Satellites in Medium Earth Orbit

Once the method is confirmed to work for satellites at different altitudes in LEO, it is time to extend the testing on objects in Medium Earth Orbit (MEO). In this orbital regime, the effect of drag on the satellite is largely reduced, which removes one of the largest uncertainties when it comes to environmental variables. On the other hand, the effect of the gravitation from third bodies such as the Sun and the Moon becomes more noticeable. It is therefore essential to assess the ability of the model to generalise to objects in this orbital regime for more robust predictions. By definition, MEO is a region in space comprising any altitude from 2000 km, typically considered the upper boundary of the LEO region, to 35,876 km, the characteristic altitude for geostationary orbits. The satellite selected for this test is NAVSTAR 71, which forms part of the Global Positioning System (GPS). As such, it is located at an

altitude of about 20,200 km and has a low eccentricity, which fits well with the requirements for this analysis.

LEO is the most crowded region in space, much more than MEO or GEO, and therefore there is a need to have large amounts of data from multiple altitudes in this orbital regime. However, to make the addition of an extra feature in the model consistent with the neural network, the orbital period information needs to be normalised. This might become an issue for a neural network working with satellites at both regions since the difference between normalised orbital data is much smaller between satellites at LEO than at MEO, especially since the latter has a smaller satellite density. Indeed, when transfer learning is used for NAVSTAR 71, it is observed that, while the model works well for this satellite, it has significant issues predicting the state of satellites at LEO. As such, it is recommended that these two orbital regions be separated: one neural network can be used for satellites in LEO and a different one for satellites in MEO and GEO.

With this in mind, a new neural network is set for the satellites at altitudes higher than 2000 km. However, transfer learning can still be used to make the training process of the new neural network more efficient, while tuning will help improve its performance on the new satellites. When this is done, the state for NAVSTAR is predicted using this model and compared to the actual satellite data. Figure 8.12 shows this comparison and, as previously, the underlying patterns in the data and the information from the orbital period are captured without observable errors for a prolonged period. However, the scale of the state is one order of magnitude larger for this case than for LEO. As such, a neural network with the same architecture, achieving a similar test loss, will still yield a much larger absolute error. This can be observed in Figure 8.13, where the absolute errors in the position and velocity are plotted. The most striking feature in this figure is the large jump in the error for each coordinate. Since this happens exactly at the one-hour mark since the start of the prediction, it could be a simple matter of a difference in the normalisation parameters with respect to those used in the training process. This highlights the sensitivity of the predictions to these parameters at high altitudes, where the state scale is higher. Nevertheless, the magnitude of the errors suggests that a more complex architecture should be used for satellites in MEO to lower the error.

Satellites in Geostationary Earth Orbit

The Geostationary Earth Orbit (GEO) is a unique type of orbit in which an object circles the Earth above its Equator and has an orbital period that matches the Earth's rotational period. These satellites are located at 35,786 km of altitude and have an eccentricity close to 0. As such, they have significantly different characteristics when compared to satellites located in MEO and LEO. It was discussed earlier that MEO provides a more prominent effect of third-body perturbations and solar radiation pressure. These perturbations have an even larger impact on GEO, where the atmospheric drag can be considered negligible.

Since satellites in GEO have the same rotational speed as the Earth, they appear stationary relative to a fixed position on its surface. This makes this type of orbit special and suitable for communication and weather satellites, where coverage of a specific area is desired. Testing the model on satellites in GEO is the final step in assessing the neural network's generalisation capabilities across all major orbital regions, with significantly different environments. If the model can generalise well to these orbits, it can be further proof of the robustness of the neural networks and can reflect the usefulness and reliability of this approach for collision detection and assessment for diverse missions.

The RSO selected for this assessment is SES 17, a communications satellite which forms part of the SES constellation. The same process described in other sections is repeated, and the predicted state of the satellite is plotted alongside its actual state in Figure 8.14. As seen previously, the model can predict the overall pattern of the state of this satellite for an extended time period. To quantify this, the residuals of the prediction of the test dataset are calculated and plotted in Figure 8.15. Surprisingly, the errors are lower in this test case than they were for the satellite in MEO. This could be explained by the use of a closer match for the normalisation parameters for this satellite (as evidenced as well by the smaller jump in the residual plot at the one-hour mark). On top of that, orbits at such heights have more stable dynamics than in MEO and LEO which in turn makes it easier for the neural network to catch the patterns in the model and can bring the test loss down even for a similar architecture. It is also noticeable that the errors in the z- axis components of the position and velocity are much smaller than in other axes due to their scale difference. Since geostationary orbits are located around the Equator,

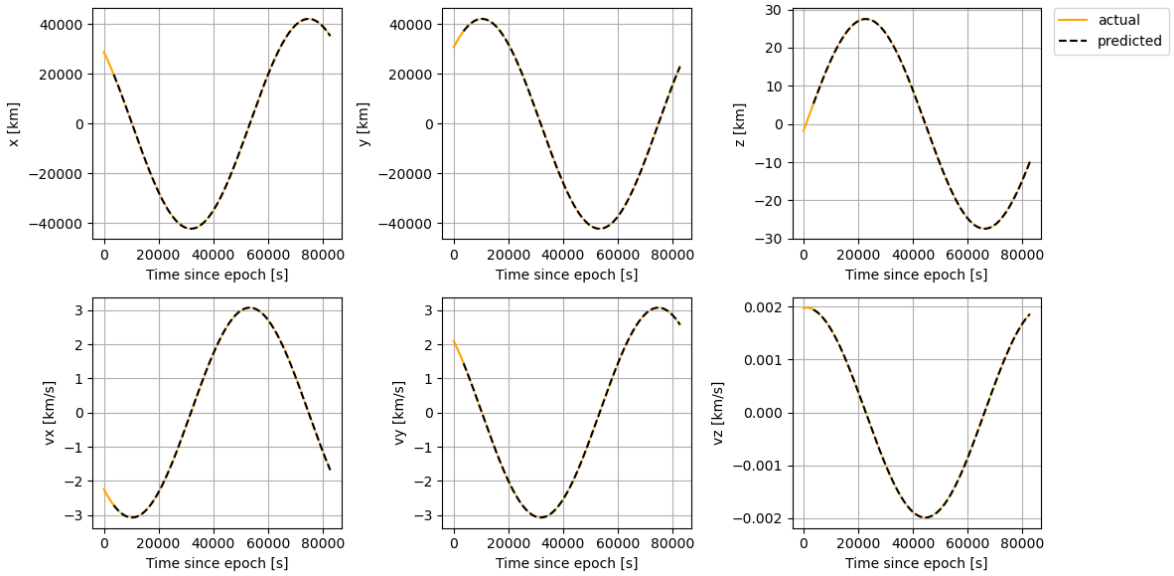


Figure 8.14: State prediction for SES 17 with a neural network trained on satellites in GEO compared to the actual satellite data

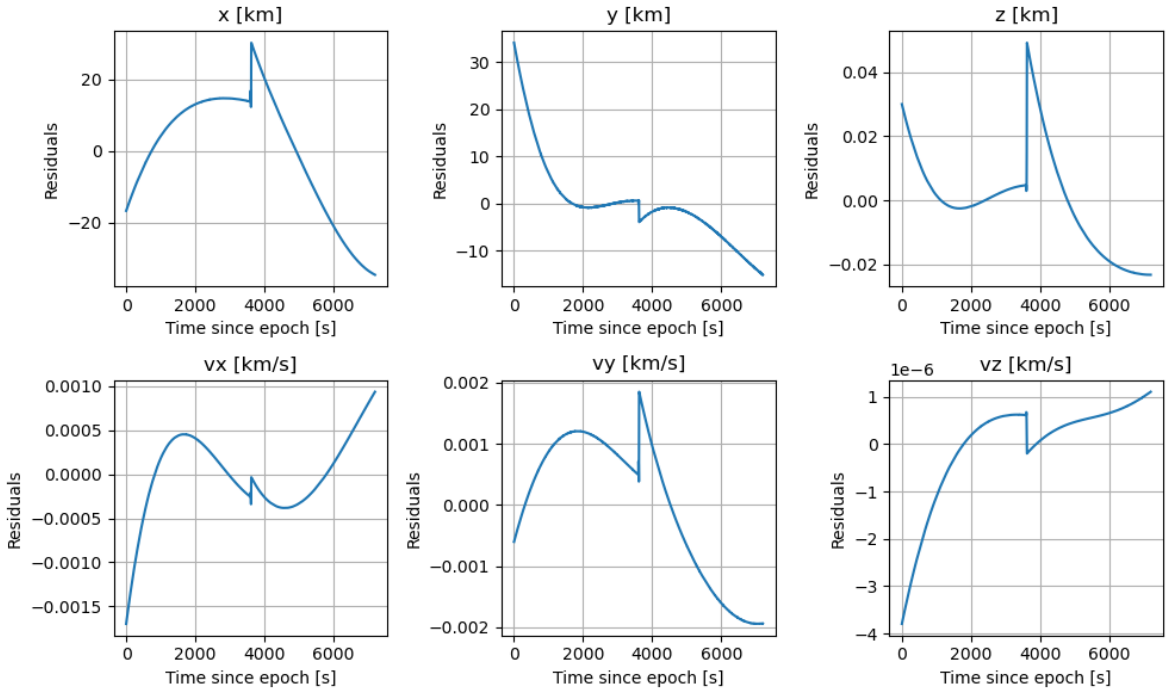


Figure 8.15: Residuals of the prediction of the SES 17 state using a neural network trained on satellites in GEO

their z-axis components will be much closer to 0. This shift in scale brings the absolute errors down as well.

There are several key aspects to take into account when using a neural network for satellites at high altitudes. Firstly, due to the larger scale of the Cartesian coordinates when compared to LEO satellites, the sensitivity of the model to the normalisation is increased: small changes in the parameters used for the normalisation have a larger effect on the model's accuracy for GEO. As a consequence, the prediction errors might become too large if the denormalisation of the data is not handled correctly, which only get further propagated with time. As such, keeping the normalisation parameters as close as possible to the ones used for the training is critical for accurate predictions. Secondly, collision scenarios

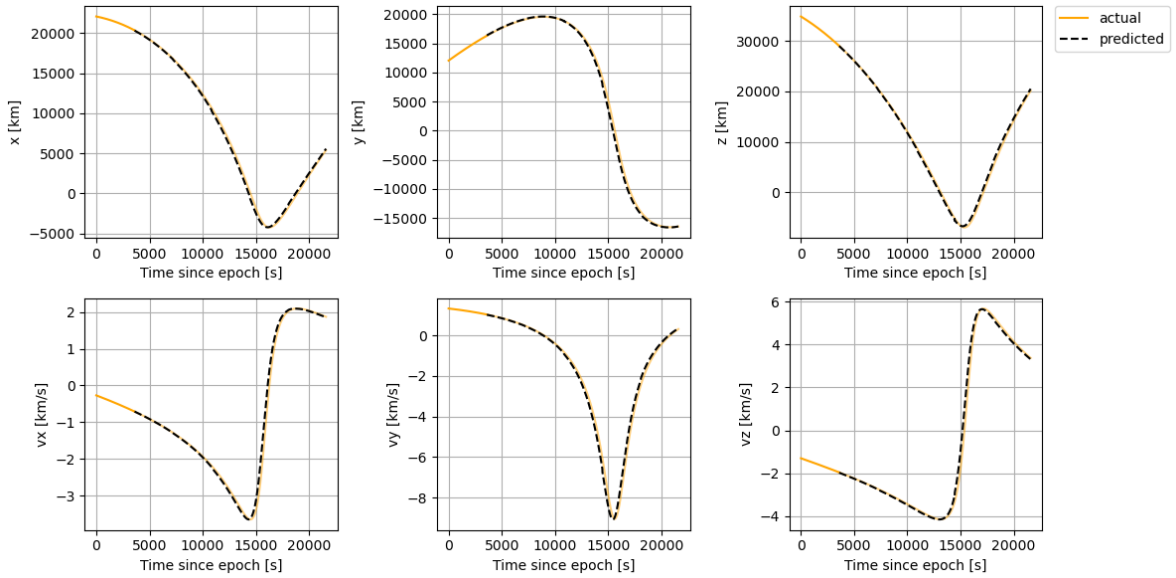


Figure 8.16: State prediction for a Molniya orbit with a neural network trained on satellites in MEO compared to the actual satellite data

located at higher altitudes have a longer encounter time and, as such, the time step in the model can and should be increased. A time step of one second was selected especially for satellites in LEO, where the encounter time is short and a high resolution of the probability of collision distribution is desirable. Moreover, the dynamics at those altitudes are significantly faster than in GEO, with larger relative speeds and more influenced by the atmospheric drag. As such, lowering the model's resolution for satellites at higher altitudes could be considered for faster assessments and a lower risk of overfitting.

8.4.3. Generalisation to Satellites in Highly Eccentric Orbits

The previous subsection has allowed for the identification of limitations in the generalisation of the model depending on the orbit's altitude. It has been shown how it works well in the prediction of underlying patterns and behaviours for different satellites in different orbital regions. However, all the satellites tested had near-circular orbits, leading to sine wave patterns in all scenarios which had already been proven to work in Section 8.1. The next area of interest is the study of the model's generalisation to satellites in Highly Elliptical Orbits (HEO), where there are variations in the speed, altitude, and perturbations acting on the object of interest in the same orbital period.

Due to the large eccentricity of these orbits, the object will cross several orbital regions in a single period and the Cartesian elements' behaviour will no longer resemble that of a sine wave. This might present a challenge for a neural network since it is to be studied whether the approach presented for generalisation purposes can be applied to cases where the orbit is no longer circular.

As discussed in Subsection 7.4.2, the satellite chosen for this test case is the Molniya 3-50, with an eccentricity of 0.7. The semi-major axis of this orbit is approximately 26,551 km, with the altitude corresponding to this being 20,173 km. This is in the same range as the NAVSTAR 71 tested earlier. However, due to its eccentricity, the perigee is located at an approximate height of 1,322 km (located inside of LEO), whereas the apogee presents a height close to 39,024 km (an altitude higher than GEO). The question here is which model should be used since this satellite crosses all the major regions analysed. In the end, it was decided that transfer learning should be used on the neural network trained with MEO satellite data since the orbital period of the Molniya orbit falls within the ranges considered for that network. The model is then trained, and the resulting predictions are plotted and shown in Figure 8.16. It can be concluded from this figure that, interestingly enough, the model trained with MEO data is able to generalise its predictions to highly eccentric orbits, even if the dynamics are largely different, for a prolonged propagation period. The reason behind this is that the model is trained on

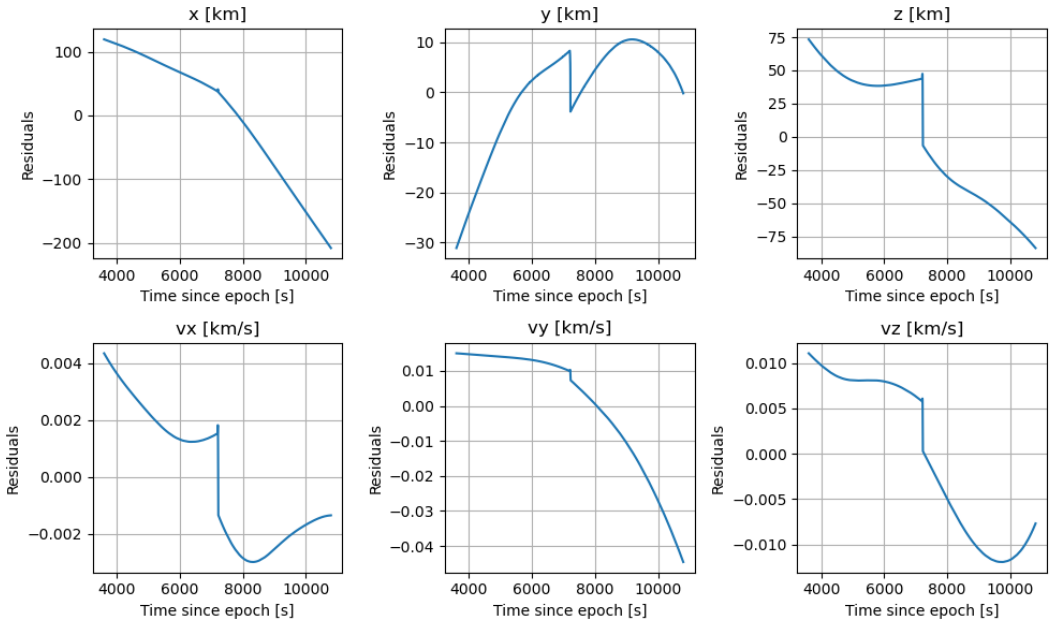


Figure 8.17: Residuals of the prediction of the Molniya orbit using a neural network trained on satellites in MEO

sequences, meaning that 20 consecutive data points are used to predict the next step in a sequence. This makes the model more robust as different input patterns will translate into different outputs. On top of that, this test confirms that using the orbital period instead of the altitude for the state prediction of highly elliptical orbits increases the robustness of the algorithm.

As much as the algorithm can predict the behaviour of the state as a function of time, the errors still need to be quantified. Figure 8.17 shows the absolute errors in the predictions of this orbit using the neural network trained with satellites in MEO. The projections for this orbit present a much larger error than other satellites tested. As such, it seems like the neural network might have issues trying to predict a type of orbit which is significantly different from the others tested, especially when the neural network had been trained with circular orbits. An increase in the orbital geometries with which to train a neural network, together with an increase in the model complexity, would improve the robustness of the model and increase the accuracy for a wider range of orbits.

8.4.4. Results Summary

The constraints in time and workload for this thesis have made it difficult to create a generic neural network to use for collision detection and avoidance. However, some tests have been presented in this section to show the first steps to create this tool and show the generic approach of using neural networks and their robustness to different scenarios. The conclusions drawn from this section are summarised below:

- Neural networks trained on a specific satellite have the ability to generalise to other objects with similar dynamics. However, their accuracy will be increased if data from different satellites is used using transfer learning.
- Transfer learning is a useful tool for increasing the generalisation capabilities of the algorithm since the time that is required to extend a pre-trained neural network to predict data from different orbiting objects is greatly reduced.
- When it comes to using a single deep learning algorithm to predict the state of satellites at different altitudes, the inclusion of their orbital periods as part of the training process is a simple, yet very effective way of increasing the accuracy of the network for different circular orbits. Since the improved approach (which allows the model to have stable predictions over a larger time) implies training the model to predict the Cartesian elements one hour after the input sequence, which

changes significantly depending on the period of the orbit, this is considered a useful feature to include as an input to the model.

- The separation of model parameters for the major orbital regions is recommended since it has been shown to increase the prediction accuracy for the different tested cases. This is due to the different dynamics and characteristics of each region, which make it difficult for a single neural network to learn such different data. The difference in scale for MEO and GEO objects when compared to LEO also complicates the task of learning the differences in dynamics for objects at lower altitudes, especially when the latter are of particular interest due to their crowdedness and increased risk of collision.
- Objects located at higher altitudes, particularly those in GEO, are more sensitive to normalisation parameters. As such, it is key to keep those parameters as accurate as possible to avoid unwanted and avoidable errors in the prediction process.
- The step resolution of the state prediction can be lowered for satellites in MEO and GEO since the encounter times are longer. In LEO it is critical to keep the step between data points at one second due to the errors in the model and the lower encounter time. On the other hand, test cases at higher altitudes do not require such a small time step since the encounter time might even last for a couple of minutes. As such, taking a longer time step is recommended to ease the computation and analysis of test cases at higher altitudes.
- The approach taken for the generalisation of the model studied in this thesis can also be extended to satellites in highly elliptical orbits. By including the orbital period information in the input features, transfer learning allows the model to predict relatively well the Cartesian elements of orbits with highly different dynamics. The accuracy for these orbits can be increased by training the generic tool with a wider range of orbit geometries.
- However, the more a model is taught to generalise to different objects, the lower the prediction accuracy will be. It would therefore be key to increase the complexity of the model when studying different objects.

For each of these satellites, the residuals of the orbit prediction have been calculated and shown, even though no details have been given on the generalisation in the context of the collision probability calculation. The residuals for each satellite have been kept lower than 1% of the total position, which has been proven to be enough for collision detection in Section 8.2 and Section 8.3. These new estimates, however, will probably be much lower than those calculated with more precise models, hence it is restated that ways to increase the accuracy of the predictions (i.e., more complex neural networks or higher quality training data) should be studied. On top of that, using the interval method with the Cartesian and USM7 data has been shown to reduce the uncertainty ellipses of the state at each time step and thus it is encouraged to use it for collision detection.

The test cases studied have shown that it is possible to increase the robustness of the method researched for orbit prediction and collision detection. Previous sections have already showcased the ability of this approach to predict patterns for long periods at a fraction of the computational cost of current methods used for the same task. While its accuracy can always be increased, this preliminary model has been able to detect the Cosmos-2251/Iridium-33 collision within the same order of magnitude as the DA-GMM. If this model can be extended to satellites in different orbital regions to make it more generic and robust, it can become a useful tool for collision detection. As of now, the first tests have proven that this generalisation is possible, but requires significant work and large sets of data to use for training. Since the time and workload for this research are limited, the test cases shown can be taken as the first steps to building a generic tool for collision detection using deep learning models.

9

Conclusions and Recommendations

With the model fully developed and tested, it is time to move on to the final chapter of this thesis. This chapter aims to provide the main findings of the research in Section 9.1. On top of that, the research questions are revisited and answered. Section 9.2 presents a summary of the recommendations for future work, including possible ways of improving the current model as well as the next logical steps for the development of the model.

9.1. Conclusions

This research has mostly been divided into two parts: the extension of the DA-GMM developed by Leon Dasi (2021) and the development of a model that uses a deep learning-based approach, both aimed at collision detection. This section addresses the results obtained for each of these parts, although the main focus of the investigation in this thesis has been put on the creation of the machine learning model.

Differential Algebra-Gaussian Mixture Model

The DA-GMM was already shown to be an accurate and robust model that could be used for any encounter geometry. However, it still showed some limitations for higher altitudes as the Solar Radiation Pressure acceleration was not included in the model. As this is an influential perturbation for satellites at higher altitudes, particularly those in GEO, it was included as an extension of the algorithm.

A sensitivity study was performed using a range of altitudes. This made use of the L_2 parameter, with established error threshold was set at 0.6, and it was observed that the extended algorithm had more accurate propagations and uncertainty distributions than the original model for all the altitudes tested. However, this improvement was only marginal for LEO since this perturbation does not have a strong effect at low altitudes. For satellites in GEO, on the other hand, the extension significantly improved the results obtained by reducing the L_2 distance between distributions by around five times.

Regarding the probability collision calculation, two scenarios were selected. Firstly, the Cosmos-Iridium collision was chosen as the real-life scenario with which to test the algorithm. This case occurred at an altitude of 800 km approximately, and since the model showed small improvements at such altitudes, the P_c calculation yielded only a 0.015% increase in the accuracy of the result. To further assess the effect of the addition of the SRP perturbation, a synthetic crash scenario in GEO was developed. This scenario showed that the extension outperforms the original model, as expected, by about an order of magnitude.

Physics-Informed Neural Network

Regarding the NN-based approach, a model was developed that could be used for orbit prediction and uncertainty estimation with the ultimate goal of collision detection. Due to the complexity of the topic, it was decided that an initial model would be developed using data from a single satellite as an initial step to create a more generic tool. The aspects that are studied related to the neural network are

the orbit prediction accuracy, the uncertainty estimation, the collision probability calculation, and the generalisation of the model.

When it comes to the orbit prediction, the model demonstrates an increased accuracy over a 16-hour propagation when compared to the DA-GMM. The architecture selected for the algorithm allows the maximum error achieved to be lower than 1%, with a mean error of ≈0.4%, for a set of Cartesian coordinates. The results for the Unified State Model, also tested, present a maximum error in the Cartesian state (after conversion) twice as large as that obtained with the model trained with the Cartesian coordinates. Since the error on each element is shown to be much lower than that, it can be assumed that the USM elements are sensitive to this transformation due to the interdependency of the quaternions. The USM could still be used for the interval method to reduce the uncertainties of the model.

Although these were promising results, they were not enough to assess the algorithm's application to real-life scenarios. As such, the Cosmos-2251/Iridium-33 collision case was used to test this application. This test case proved that the collision probability can be predicted within the same order of magnitude as the DA-GMM algorithm, hence demonstrating the potential of this method for collision detection. At the same time, it showed that there were several limitations and flaws with the approach taken. Since the model error was included as a source of uncertainty, this inevitably led to a reduction in the collision probability due to the uncertainty dilution phenomenon, providing a lower estimate than that obtained with the DA-GMM. This same test case showed that this model can be used for much faster P_c calculations. One of the main advantages of using this approach is that it can provide results for the orbit prediction (with a resolution of $\Delta t = 1$ s) much faster than the DA-GMM and even a highly accurate propagator. On top of that, since no GMEs were used, the calculation of the risk probability was over 10,000 times faster than that for the DA-GMM. One further conclusion that was drawn was that simplified perturbation models could be used without much loss in accuracy, significantly reducing the computational load and time required for this method.

More relevant findings were obtained from testing on a synthetic crash scenario, which was built to study the model's lead time and the effect of the uncertainty estimation on the P_c calculation. It was found that the model can provide stable answers for up to 72 hours of propagation. This test clearly showed one of the main challenges of this method: uncertainty estimation. The noise in the state predictions leads to a large model uncertainty for small lead times, which leads to smaller collision probabilities. On the other hand, the uncertainties resulting from the imperfect knowledge of the initial state are underestimated. This underestimation is balanced with the model's error, leading to stable predictions over longer propagation periods. All of this showed that the model works well for real-life scenarios where the satellite presents large state initial uncertainties (such as the Cosmos-Iridium collision), but that the uncertainties should be estimated more accurately to increase the accuracy and applicability of this method.

Regarding the generalisation capabilities of the neural network, it was found that the model generalises well to satellites with similar dynamics, but more work needs to be put into making this approach more generic. The addition of the orbital period of the different satellites as an input feature allowed the model to generalise to objects at different altitudes and with different dynamics. Limitations were found mostly in satellites in MEO and GEO, where the sensitivity to the normalisation parameters was much larger than at lower altitudes. Highly elliptical orbits also showed that the complexity of the model should be increased if the error of a generic tool was to be reduced.

Research Questions

Once the main findings of this thesis have been summarised, the research questions need to be revisited. It is critical for any research project that the obtained results provide the answers to these questions. It can be recalled from Section 1.4 that the main research question related to how the probability of collision can be calculated in a faster and more accurate manner. As previously explained, two approaches are used in this thesis: one based on the extension of the DA-GMM and the other on the development of machine learning algorithms. The first sub-question is related to the former, and how the inclusion of the SRP acceleration has affected the accuracy and robustness of the model. The second sub-question is focused on an AI-based approach to enhance collision risk assessment practices. Since the latter has been the focus of this thesis, several questions can be linked to it. All research questions from Section 1.4 have been answered as follows:

Q1 How can the computation of the probability of collision be enhanced for faster and more accurate collision risk assessments?

Q1.1 How can the DA-GMM be extended for more accurate predictions?

Q1.1.1 How does adding the effect of the Solar Radiation Pressure affect the results obtained from the DA-GMM as developed by Leon Dasi?
The extended algorithm for the DA-GMM has been proven to be more accurate than the original model for a wide range of altitudes and shows significant improvement for satellites in GEO. When tested for the Cosmos-2251/Iridium-33 case, the resulting collision probability was only marginally improved with respect to the original algorithm. However, when tested on a synthetic scenario in GEO, where the effect of the SRP is more noticeable, the improvement was significantly larger (about two orders of magnitude better).

Q1.2 To what extent can simplified models be used to accurately predict the orbit of Resident Space Objects (RSOs) and compute collision probability with the use of Artificial Intelligence?

Q1.2.1 How does the accuracy of the PINN-based algorithm compare to that of current conjunction assessment algorithms?
The results from the PINN have been compared to those from the DA-GMM. The error obtained from the nominal state prediction is lower for the AI-based approach taken in this thesis than for the DA-GMM for a 16-hour propagation. On top of that, the collision probability can be estimated within the same order of magnitude as the more accurate method.

Q1.2.2 How does the computational load required for accurate predictions using the PINN-based model translate to automatic real-time systems?
The main advantage of the ML-based approach is the cut in computational time. This method has been shown to be ≈500 times faster than the DA-GMM for orbit propagation and over 10,000 times faster for the calculation of the collision probability.

Q1.2.3 Which environment and perturbation models need to be used in the simulations to meet the accuracy requirements?
Since the models have been trained with highly accurate data, the simulation environment can be simplified without much loss in accuracy. A model which only considers the Earth's gravitational effect with spherical harmonics up to degree and order 5, as well as the third body perturbations of the Moon and the Sun, shows a difference of only ≈2% with respect to the highly accurate model.

Q1.2.4 How far in the future can this method predict the orbit of an RSO accurately?
The results are stable over a time period of three days, the maximum tested.

Q1.2.5 To what extent can the algorithm predict the collision and time of closest approach of the Cosmos-2251/Iridium-33 crash in 2009?
The algorithm estimates the collision probability at about $5{\cdot}10^{-7}$, 16 hours before the encounter with initial state information obtained from TLEs, which is in the same order of magnitude as that predicted by the DA-GMM.

With the research questions successfully answered, the ML-based method developed in this thesis has been verified and validated for a range of scenarios. Its potential application for collision detection has been demonstrated, although it could benefit from improved uncertainty modelling and orbit prediction.

9.2. Recommendations for Future Work

This section continues from the conclusions drawn and establishes some recommendations that should be considered to continue working on this method.

The first, and probably the most important, recommendation is the improvement of the uncertainty estimation and modelling. So far, the uncertainty is modelled in such a way that it includes the model error and the initial state uncertainty propagation. This has proved to be enough to estimate the risk probability, but not to provide accurate enough answers for a variety of scenarios. The model could

therefore benefit from more accurate modelling. One possibility is to switch the focus of the model and use a Bayesian Recurrent Neural Network as the base model, which is more suitable for uncertainty estimation. Another simpler, yet perhaps less accurate, approach could be to use similar models to those built in this thesis to learn how the uncertainties are propagated with time. A final possibility would be to use a hybrid approach between a neural network and the DA-GMM by which the different GMEs are predicted with a neural network, further reducing the time it takes to propagate all the different GMEs while keeping the accuracy and robustness of the model. This last approach would require a more profound knowledge of the mathematical background of the DA-GMM and neural networks and thus might be more complicated to replicate. This would also allow the removal of the simplifying assumption of normally distributed uncertainties when it is known that propagated uncertainties in a non-linear environment lead to non-Gaussian distributions. On the other hand, the model's uncertainties can be reduced with a more complex architectural design of the neural networks, always with care of avoiding overfitting.

As per the state representation, the Cartesian coordinates have been used due to their sinusoidal nature, easier for the neural network to learn from. The same model was able to learn the USM7, albeit with a large error in the transformation to Cartesian state coordinates. This could be eliminated if no transformation was needed, that is, if the P_c calculation did not require Cartesian coordinates. This could be helpful since USM7 elements do not present such large derivatives and variation in their values. Another state representations could also be considered if another training method were to be applied. A piece-wise modelling of the discrete elements of, for example, the Unified State Model - Exponential Map state representation could also present a solution to the sensitivity of the transformation (since the exponential map elements are not interdependent).

Another line of future work is the generalisation of the model proposed in this thesis. The first steps have been provided for that, by looking at possible strategies and limitations for this generalisation, but there is significant room for improvement. The current model as it stands is highly satellite-specific as it has been trained with data from a limited number of satellites. If the model were to be made more generic, many different orbital geometries at different altitudes and regions should be fed into a rather complex neural network architecture to teach the model to generalise to different satellites and use it to predict the position and velocity for any object using only tracking data. This would bring the model one step closer to real-life application and deployment.

References

18SDS & 19SDS. (2023). *Spaceflight safety handbook for satellite operators* [Available at https://www.space-track.org/documents/SFS_Handbook_For_Operators_V1.7.pdf].

Akulenko, L. D., Leshchenko, D. D., & Rachinskaya, A. L. (2008). Evolution of the satellite fast rotation due to the gravitational torque in a dragging medium. *Mechanics of Solids*, *43*(2), 173–184. https://doi.org/10.3103/S0025654408020027

Alarcon-Rodriguez, J., Martinez-Fadrique, F., & Klinkrad, H. (2002). Collision risk assessment with a 'smart sieve' method. *Proceedings of the Joint ESA-NASA Space-Flight Safety Conference*, 159–164.

Alfano, S. (2005). Relating Position Uncertainty to Maximum Conjunction Probability. *Journal of the Astronautical Sciences*, *53*, 193–205.

Alfano, S. (2006). Addressing nonlinear relative motion for spacecraft collision probability. *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*.

Alfano, S. (2007). Beta conjunction analysis tool. *AAS/AIAA Astrodynamics Specialist Conference*.

Alfano, S. (2009). Satellite conjunction Monte Carlo analysis. *Advances in the Astronautical Sciences*, *134*, 2007–2024.

Anz-Meador, P. D., & Liou, J. (2010). Analysis and consequences of the iridium 33-cosmos 2251 collision. *38th COSPAR Scientific Assembly*.

Badhwar, G., & Anz-Meador, P. D. (1989). Determination of the area and mass distribution of orbital debris fragments. *Earth, Moon, and Planets*, *45*(1), 29–51. https://doi.org/10.1007/BF00054659

Balch, M. S., Marting, R., & Ferson, S. (2019). Satellite Conjunction Assessment and the False Confidence Theorem. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *475*. https://doi.org/10.1098/rspa.2018.0565

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, *18*(153), 1–43.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(10), 281–305. http://jmlr.org/papers/v13/bergstra12a.html

Bhusal, R., & Subbarao, K. (2020). Generalized polynomial chaos expansion approach for uncertainty quantification in small satellite orbital debris problems. *The Journal of the Astronautical Sciences*, *67*(1), 225–253. https://doi.org/10.1007/s40295-019-00176-1

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Braun, V., Flohrer, T., Krag, H., Merz, K., Lemmens, S., Virgili, B. B., & Funke, Q. (2016). Operational support to collision avoidance activities by ESA's space debris office. *CEAS Space Journal*, *8*(3), 333–345. https://doi.org/10.1007/s12567-016-0145-2

Carpenter, J. R., & Markey, F. L. (2014). Wald sequential probability ratio test for space object conjunction assessment. *Journal of Guidance, Control, and Dynamics*, *37*, 1385–1396.

Chan, K. (1997). Collision probability analyses for earth orbiting satellites. *7th International Space Conference of Pacific-Basin Societies*, *96*, 1033–1048.

Chan, K. (2004). Spacecraft collision probability for long-term encounters. *Journal of Guidance, Control, and Dynamics*, *116*(1), 767–784.

Cheval, A. (2023). *Tube neural orbit predictive control in the vicinity of asteroids with uncertain dynamics* [Master's thesis, KTH Royal Institute of Technology].

Choumos, G., Tsaprailis, K., Lappas, V., & Kontoes, C. (2024). Artificial intelligence for a safe space: Data and model development trends in orbit prediction and collision avoidance. *AIAA SCITECH 2024 Forum*. https://doi.org/10.2514/6.2024-2066

Coppola, V. A. (2012). Including velocity uncertainty in the probability of collision between space objects.

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., & Piccialli, F. (2022). Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, *92*(3). https://doi.org/10.1007/s10915-022-01939-z

Curzi, G., Modenini, D., & Tortora, P. (2022). Two-line-element propagation improvement and uncertainty estimation using recurrent neural networks. *CEAS Space Journal*, *14*(1), 197–204. https://doi.org/10.1007/s12567-021-00375-3

Dave, S., Clark, R., Chianelli, G., & Lee, R. (2020). Machine learning implementation for in-orbit rso orbit estimation using star tracker cameras. *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference*, 15–18.

DeMars, K., Cheng, Y., & Jah, M. (2014). Collision probability with gaussian mixture orbit uncertainty. *Journal of Guidance, Control, and Dynamics*, *37*. https://doi.org/10.2514/1.62308

Dirkx, D. (2022). Tudat Mathematical Model Definition [Available at https://github.com/tudat-team/tudat-space/raw/master/Tudat_mathematical_model_definition.pdf].

Doornbos, E. (2011). *Thermospheric density and wind determination from satellite dynamics* [Doctoral dissertation, Delft University of Technology].

ESA. (2023). Esa's annual space environment report. https://www.sdo.esoc.esa.int/environment_report/Space_Environment_Report_latest.pdf

Evard, J.-C. (1991). *Computational differential algebra*. Springer.

Fanizza, V. (2023). *Design of a Spacecraft Pose Estimation System Using Convolutional Neural Networks* [Master's thesis, Delft University of Technology].

Flohrer, T., Krag, H., & Klinkrad, H. (2008). Assessment and categorization of tle orbit errors for the us ssn catalogue. *Advanced Maui Optical and Space Surveillance Technologies Conference*.

Foster, J., & Estes, H. (1992). *A parametric analysis of orbital debris collision probability and maneuver rate for space vehicles* (tech. rep.). NASA, National Aeronautics and Space Administration, Lyndon B. Johnson Space Center.

Hall, D. (2019). Implementation recommendations and usage boundaries for the two-dimensional probability of collision calculation. *2019 AAS Astrodynamics Specialist Conference*.

Hofsteenge, R. (2013). *Computational Methods for the Long-Term Propagation of Space Debris Orbits* [Master's thesis, Delft University of Technology].

Hong, W.-K. (2023). 2 - Understanding artificial neural networks: analogy to the biological neuron model. In W. Hong (Ed.), *Artificial intelligence-based design of reinforced concrete structures* (pp. 7–13). Woodhead Publishing. https://doi.org/https://doi.org/10.1016/B978-0-443-15252-8.00003-0

Hoots, F. R., Crawford, L. L., & Roehrich, R. L. (1984). An analytic method to determine future close approaches between satellites. *Celestial mechanics*, *33*(2), 143–158. https://doi.org/10.1007/BF01234152

Horstmann, A., & Stoll, E. (2017). Investigation of Propagation Accuracy Effects within the Modeling of Space Debris. *7th European Conference on Space Debris*.

Horwood, J., Aragon, N., & Poore, A. (2011). Gaussian Sum Filters for Space Surveillance: Theory and Simulations. *Journal of Guidance, Control, and Dynamics*, *34*(6), 1839–1851. https://doi.org/10.2514/1.53793

Jadala, G., Meedinti, G. N., & Delhibabu, R. (2022). Satellite Orbit Prediction using a Machine Learning Approach.

Jones, B., & Doostan, A. (2013). Satellite collision probability estimation using polynomial chaos expansions. *Advances in Space Research*, *52*(11), 1860–1875. https://doi.org/https://doi.org/10.1016/j.asr.2013.08.027

Kelso, T. (2009). Analysis of the Iridium 33 Cosmos 2251 Collision. *19th AIAA/AAS Astrodynamics Specialist Conference*.

Kessler, D. J., & Cour-Palais, B. G. (1978). Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research*, *83*(A6).

Khutorovsky, Z., Boikov, V., & Kamensky, S. (1993). Direct method for the analysis of collision probability of artificial space objects in leo: Techniques, results, and applications. *Proceedings of the First European Conference on Space Debris*, 491–508.

King-Hele, D. (1987). *Satellite Orbits In an Atmosphere*. Springer Dordrecht.

Kingma, D. P., & Ba, J. L. (2014). Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations*.

Klinkrad, H. (1993). Analysis of near-misses between operational spacecraft and catalogued objects. *Proceedings of the First European Conference on Space Debris*, 551–557.

Kolchin, E. (1973). *Differential algebra and algebraic groups*. Academic Press.

Kollmannsberger, S., D'Angella, D., Jokeit, M., & Herrmann, L. (2021). Physics-informed neural networks. In *Deep learning in computational mechanics: An introductory course* (pp. 55–84). Springer International Publishing. https://doi.org/10.1007/978-3-030-76587-3_5

Lachut, M., & Bennett, J. (2016). Towards relaxing the spherical solar radiation pressure model for accurate orbit predictions. *Advanced Maui Optical and Space Surveillance Technologies (AMOS) Conference*.

Lan, S., Li, S., & Shahbaba, B. (2022). Scaling up bayesian uncertainty quantification for inverse problems using deep neural networks. *SIAM/ASA Journal on Uncertainty Quantification*, *10*(4), 1684–1713. https://doi.org/10.1137/21M1439456

Lebedev, V. (1976). Quadratures on a sphere. *USSR Computational Mathematics and Mathematical Physics*, *16*(2), 10–24.

Leloux, J. (2012). *Filtering Techniques for Orbital Debris Conjunction Analysis* [Master's thesis, Delft University of Technology].

Leon Dasi, M. (2021). *Collision probability through orbital uncertainty propagation* [Master's thesis, Delft University of Technology].

Martin, J., & Schaub, H. (2022). Physics-informed neural networks for gravity field modeling of small bodies. *Celestial Mechanics and Dynamical Astronomy*, *134*(5). https://doi.org/10.1007/s10569-022-10101-8

McDowell, J. (2024). *Jonathan's space report*. Retrieved February 22, 2024, from https://planet4589.org/space/con/star/stats.html

McKinley, D. (2006). Development of a nonlinear probability of collision tool for the earth observing system. *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*.

Montenbruck, O., & Gill, E. (2000). *Satellite orbits* (Vol. 1). https://doi.org/10.1007/978-3-642-58351-3

Mooij, E. (2019). *Ae4870b - re-entry systems lecture notes*. Faculty of Aerospace Engineering of Delft University of Technology.

Moore, R. E. (1968). Practical Aspects of Interval Computation. *Applied Mathematics*, *13*(1), 52–92.

NASA. (2023a). *International laser ranging service*. Retrieved January 20, 2024, from https://ilrs.gsfc.nasa.gov/data_and_products/formats/cpf.html

NASA. (2023b). *Nasa spacecraft conjunction assessment and collision avoidance best practices handbook* [Available at https://nodis3.gsfc.nasa.gov/OCE_docs/OCE_51.pdf].

NASA. (2023c). *Orbital debris quarterly news*. Retrieved January 30, 2024, from https://orbitaldebris.jsc.nasa.gov/quarterly-news/pdfs/ODQNv27i4.pdf

NASA. (n.d.). Spherical Harmonic Representation of the Gravity Field Potential [https://spsweb.fltops.jpl.nasa.gov/portaldataops/mpg/MPG_Docs/Source$%$20Docs/gravity-SphericalHarmonics.pdf].

Patera, R. P. (2001). General method for calculating satellite collision probability. *Journal of Guidance, Control, and Dynamics*, *24*(4), 716–722.

Patera, R. (2003). Satellite collision probability for nonlinear relative motion. *Journal of Guidance, Control, and Dynamics*, *26*(5), 728–733.

Peña, L. M. (2023). *Orbital decay of objects in a low Earth orbit* [Master's thesis, Delft University of Technology].

Peng, H., & Bai, X. (2018a). Artificial neural network–based machine learning approach to improve orbit prediction accuracy. *Journal of Spacecraft and Rockets*, *55*(5), 1248–1260. https://doi.org/https://doi.org/10.2514/1.A34171

Peng, H., & Bai, X. (2018b). Exploring capability of support vector machine for improving satellite orbit prediction accuracy. *Journal of Aerospace Information Systems*, *15*(6), 366–381. https://doi.org/https://doi.org/10.2514/1.I010616

Peng, H., & Bai, X. (2018c). Improving orbit prediction accuracy through supervised machine learning. *Advances in Space Research*, *61*(10), 2628–2646. https://doi.org/https://doi.org/10.1016/j.asr.2018.03.001

Peng, H., & Bai, X. (2019). Gaussian processes for improving orbit prediction accuracy. *Acta Astronautica*, *161*, 44–56. https://doi.org/https://doi.org/10.1016/j.actaastro.2019.05.014

Picone, J., Hedin, A., Drob, D., & Aikin, A. (2002). NRLMSISE-00 empirical model of the atmosphere: Statistical comparison and scientific issues. *Journal of Geophysical Research*, *107*(A12), 1468–1483. https://doi.org/10.1029/2002JA009430

Prince, S. (2024). *Understanding deep learning*. The MIT Press.

Raissi, M., Perdikaris, P., & Em Karniadakis, G. (2017a). Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv preprint arXiv:1711.10561*.

Raissi, M., Perdikaris, P., & Em Karniadakis, G. (2017b). Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. *arXiv preprint arXiv:1711.10566*.

Raissi, M., Perdikaris, P., & Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *378*, 686–707. https://doi.org/https://doi.org/10.1016/j.jcp.2018.10.045

Römgens, B. (2011). *Verified Interval Orbit Propagation with an Application to Satellite Collision Detection* [Master's thesis, Delft University of Technology].

Rongzhi, Z., & Kaizhong, Y. (2020). Outline of spacecraft collision warning. In *Spacecraft collision avoidance technology* (pp. 1–9). Academic Press.

Rossi, A., & Valsecchi, G. B. (2006). Collision risk against space debris in earth orbits. *Celestial Mechanics and Dynamical Astronomy*, *95*(1), 345–356. https://doi.org/10.1007/s10569-006-9028-7

Sanchez, L., & Vasile, M. (2020). AI for Autonomous CAM Execution. *71st International Astronautical Congress*.

Sanchez, L., & Vasile, M. (2021). On the use of machine learning and evidence theory to improve collision risk management. *Acta Astronautica*, *181*, 694–706. https://doi.org/https://doi.org/10.1016/j.actaastro.2020.08.004

Sanchez, L., Vasile, M., & Minisci, E. (2019). AI to Support Decision Making in Collision Risk Assessment. *70th International Astronautical Congress*.

Schmutz, W. K. (2021). Changes in the Total Solar Irradiance and climatic effects. *Journal of Space Weather Space Climate*, *11*. https://doi.org/10.1051/swsc/2021016

Sgobba, T., & Allahdadi, F. (2013). Chapter 8 - Orbital Operations Safety. In *Safety design for space operations* (pp. 411–602). Butterworth-Heinemann. https://doi.org/https://doi.org/10.1016/B978-0-08-096921-3.00008-8

Shin, Y., Park, E., Woo, S. S., Jung, O., & Chung, D. (2022). Selective tensorized multi-layer lstm for orbit prediction. *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. https://api.semanticscholar.org/CorpusID:252904766

Space-Track. (2009). *Home*. Retrieved January 20, 2023, from https://www.space-track.org/

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

Tsaprailis, K., Choumos, G., Lappas, V., & Charalabos, K. (2024). Survey mode: A review of machine learning in resident space object detection and characterization. *AIAA SCITECH 2024 Forum*. https://doi.org/10.2514/6.2024-2065

UN. (2010). Space debris mitigation guidelines of the committee on the peaceful uses of outer space [Available at https://www.unoosa.org/pdf/publications/st_space_49E.pdf].

USSPACECOM. (2024). *U.s. space command*. Retrieved February 2, 2024, from https://www.spacecom.mil/

Vallado, D., & Crawford, P. (2012). Sgp4 orbit determination. *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. https://doi.org/https://doi.org/10.2514/6.2008-6770

Valli, M., Armellin, R., Di Lizia, P., & Lavagna, M. (2013). Nonlinear Mapping of Uncertainties in Celestial Mechanics. *Journal of Guidance, Control, and Dynamics*, *36*(1), 48–63. https://doi.org/10.2514/1.58068

Van den Oord, A., Vinyals, O., & Kavukcuoglu, K. (2017). Neural Discrete Representation Learning. *CoRR*, *abs/1711.00937*. http://arxiv.org/abs/1711.00937

Vittaldev, V. (2010). *The Unified State Model: Derivation and Applications in Astrodynamics and Navigation* [Master's thesis, Delft University of Technology].

Wakker, K. F. (2015). *Fundamentals of astrodynamics* (6th ed.). Institutional Repository, Delft University of Technology.

Walker, M. J. H., Ireland, B., & Owens, J. (1985). A set modified equinoctial orbit elements. *Celestial Mechanics*, *36*(4), 409–419. https://doi.org/https://doi.org/10.1007/BF01227493

Wang, C., Guo, J., Zhao, Q., & Liu, J. (2018). Solar Radiation Pressure Models for BeiDou-3 I2-S Satellite: Comparison and Augmentation. *Remote Sensing*, *10*(1). https://doi.org/10.3390/rs10010118

Wilson, R. C., & Hudson, H. S. (1991). The Sun's luminosity over a complete solar cycle. *Nature*, *351*. https://doi.org/10.1038/351042a0

Yang, H., Jiang, M., Li, Y., & Chen, H. (2020). Research on micro-satellite orbit forecasting technology based on artificial intelligence. *4th International Conference on Computer Engineering, Information Science & Application Technology*, 240–246.

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. (2023). *Dive into deep learning*. Cambridge University Press.

A

Probability Distributions

This chapter provides an overview on PDFs in Section A.1, and the different distributions considered are introduced and briefly explained in Section A.2.

A.1. Probability Density Functions

In the context of uncertainty modelling and propagation, it is relevant to introduce the concept of probability distribution and probability density function (PDF). The probability distribution of a random variable indicates the likelihood that the value of said variable falls within a certain range $S \in [a, b]$, and is mathematically formulated as follows:

$$P[a \leq X \leq b] = \int_a^b p(x)dx \tag{A.1}$$

The term $p(x)$ represents the probability density function, and it describes the likelihood that the value of the random variable is equal to that of the PDF at a specific point in the sample space, S.

A.2. Uncertainty Modelling

There are many different formulations to represent the probability density function, which are presented in the following subsections. Firstly, Subsection A.2.1 introduces the Gaussian distribution, most commonly used in the field of orbit determination. The uniform distribution is presented in Subsection A.2.2 and the log-normal, in Subsection A.2.3.

A.2.1. Gaussian Distribution

The gaussian distribution, also known as normal distribution, can be constructed from the mean (μ) and standard deviation (σ) of a sample. The mathematical formulation is presented as follows:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{A.2}$$

The probability of a random variable having a specific value decreases significantly at larger distances from the mean of the sample. Due to the formulation of this distribution, it is known that roughly 68% of the values drawn from the sample will fall within the $[-\sigma, \sigma]$ range. This increases to about 95% for the 2σ interval, and 99.7% for the 3σ. This is represented in Figure A.1a.

A.2.2. Uniform Distribution

As the name indicates, the uncertainty is modelled in such a way that the probability of the random variable having a certain value is constant throughout the interval considered. This is expressed in Equation (A.3).

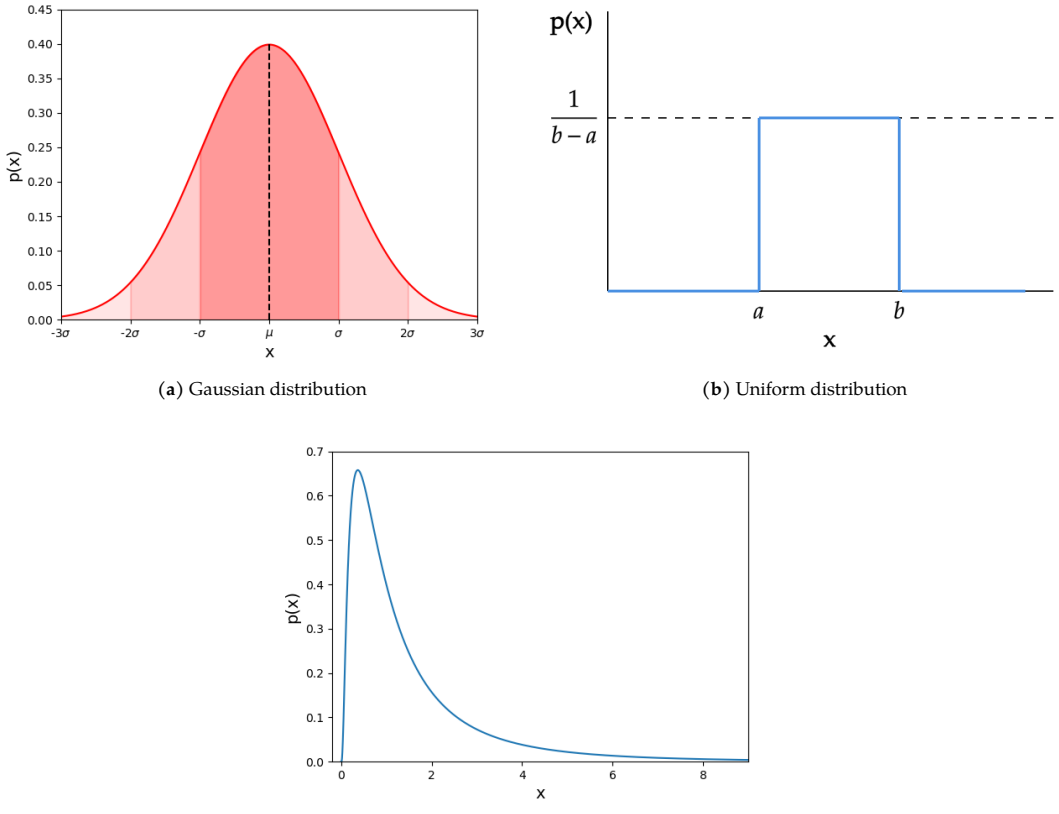
$$p(x) = \begin{cases} \frac{1}{b-a} & \text{for} \quad a \leq x \leq b, \\ 0 & \text{for} \quad x < a \text{ or } x > b \end{cases} \tag{A.3}$$

The values of the mean and standard deviation of the sample are obtained following the equations below:

$$\mu = \frac{a+b}{2} \tag{A.4}$$

$$\sigma = \frac{b-a}{2\sqrt{3}} \tag{A.5}$$

And the visual representation of this distribution is provided in Figure A.1b.



(**a**) Gaussian distribution

(**b**) Uniform distribution

(**c**) Log-normal distribution

Figure A.1: Representations of the different probability density functions considered

A.2.3. Log-normal Distribution

The last PDF that will be presented is the log-normal distribution. This is typically used to represent the uncertainty of variables whose natural logarithm is normally distributed. Following this, if a random variable X is said to follow a log-normal distribution, then $ln(X)$ follows a Gaussian distribution. The expression used to represent this type of PDF is:

$$p(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{(\ln(x-\mu))^2}{\sigma^2}\right)} \tag{A.6}$$

This expression is similar to that presented in Equation (A.2), but unlike the Gaussian PDF, the log-normal distribution is not symmetrical about the mean of the sample. The result of applying a normal logarithm in the expression representing the PDF is that the distribution is right-skewed. The shape of it and the skewness will be defined by the mean and standard deviation of the sample.

Similarly to this, the log-uniform distribution presented in Section 2.2 represents that the uncertainty of the logarithm of the random variable X follows a uniform PDF.

B

Trends in Data Utilisation Reviewed in the Literature

This chapter provides an extensive list of the trend of data sources and methods that are used in the literature reviewed by Choumos et al. (2024).

Table B.1: Data sources and methods used in the papers considered in the work of Choumos et al. (2024)

Year	Task	Data	Models/Methods
2008	Orbit Prediction	TLE	SGP4
2010	Orbit Prediction	Radar	Gauss-Hermite Quadrature
2011	Orbit Prediction	TLE, Telescope	Linkage Algorithms
2012	Collision Avoidance	Orbital State Data	Genetic Algorithm
2012	Orbit Prediction	Orbital State Data	Gaussian Sum Filters
2012	Orbit Prediction	TLE, Telescope	SDP4/SDP8
2012	Orbit Prediction	Orbital State Data	Orbit Dynamics models
2012	Orbit Prediction	Orbital State Data	Particle Filtering, PCA, ICA
2013	Orbit Prediction	Orbital State Data	Genetic Algorithm
2013	Orbit Prediction	Orbital State Data	Genetic Algorithm (SGA, MPGA)
2014	Orbit Prediction	Observational Data, Orbital State Data	Gaussian Mixture
2014	Orbit Prediction	Laser Ranging	Particle Filtering, Genetic Resampling
2014	Collision Avoidance	TLE	Adaptive Splitting, Monte Carlo
2015	Orbit Prediction	Atmospheric / Thermospheric Density Data	ANNs, FTDNN, RTDNN, empirical atmospheric models
2017	Orbit Prediction	Orbital State Data	Separated Representations
2017	Collision Avoidance	TLE	SGP4
2018	Orbit Prediction	TLE	ANNs
2018	Orbit Prediction	Radar	SVM
2018	Orbit Prediction	Orbital State Data	SVM
2018	Orbit Prediction	Orbital State Data	Multiple ML models (AUTOSKLEARN)
2018	Collision Avoidance	TLE	Multiple Regression Analysis, Elastic Net
2019	Collision Avoidance	CDM and other	ML/DL (competition)

Table B.1 continued from previous page

Year	Task	Data	Models/Methods
2019	Collision Avoidance	Orbital State Data	NN (Feed Forward)
2019	Orbit Prediction	TLE	GP
2020	Orbit Prediction	TLE	SVM
2020	Orbit Prediction, Space-based SST	Space-based imagery	RNN-CNN
2020	Collision Avoidance	Synthetic Data of Conjunction Scenarios	Various ML/DL methods (ANN, RF, k-NN, SVM)
2020	Orbit Prediction, Space Weather	Solar/Geomagnetic Indices	Feed Forward NN (N-BEATS)
2020	Orbit Prediction	TLE	Autoregressive
2020	Collision Avoidance, Manoeuvre Detection	Synthetic Data of Conjunction Scenarios	Random Forest
2020	Orbit Prediction	TLE	RNN (LSTM)
2021	Collision Avoidance	TLE	NN
2021	Atmospheric / Thermospheric Density Modelling	Atmospheric / Thermospheric Density Data, GPS Tracking Data	NN (autoencoders)
2021	Orbit Prediction	TLE	GP, EKF
2021	Orbit Prediction	TLE	RNN (LSTM)
2021	Orbit Prediction	Radar	NN
2021	Atmospheric / Thermospheric Density Modelling	Atmospheric / Thermospheric Density Data	NN (autoencoders)
2021	Orbit Prediction	TLE	NN, SGP4
2021	Collision Avoidance	TLE	Random Forest
2022	Orbit Prediction	TLE	RNN (LSTM)
2022	Orbit Prediction	TLE	ANN (TDNN, NARX), EKF, SGP4
2022	Orbit Prediction	Orbital State Data	PCA, XGBoost
2022	Orbit Prediction	Orbital State Data	Multivariate GPR
2022	Collision Avoidance	CDM	NN (FF-NN), RF, XGBoost
2022	Collision Avoidance, Manoeuvre Detection	SP3 ephemeris, TLE	Graph Neural Networks
2022	Orbit Prediction	TLE	RNN (LSTM), Selective Tensorized LSTM (ST-LSTM)
2023	Orbit Prediction	TLE	NN
2023	Orbit Prediction	Orbital State Data	NN
2023	Orbit Prediction	SP3 ephemeris	CNN, RNN (LSTM)
2023	Atmospheric / Thermospheric Density Modelling	TLE	Gradient Descent, SGP4
2023	Orbit Prediction	TLE	Batch Least Squares Differential Correction and high-precision numerical propagators
2023	Atmospheric / Thermospheric Density Modelling	Atmospheric / Thermospheric Density Data, TLE, Earth Orientation Parameters, Space Weather Data	CNN
2023	Atmospheric / Thermospheric Density Modelling	Solar/Geomagnetic Indices	RNN (LSTM)

Table B.1 continued from previous page

Year	Task	Data	Models/Methods
2023	Atmospheric / Thermospheric Density Modelling	Atmospheric / Thermospheric Density Data	PCA (KPCA), Autoencoders
2023	Orbit Prediction	Orbital State Data	Radial Basis Function
2023	Collision Avoidance	TLE	Collision Risk Filtering (Smart Sieve, C-Sieve, and CAOS-D)
2023	Orbit Prediction	TLE	RF
2023	Collision Avoidance	CDM	Various (PCA, k-means, genetic algorithms, ensemble learning)
2023	Atmospheric / Thermospheric Density Modelling	Atmospheric / Thermospheric Density Data, Solar/Geomagnetic Indices	ML models (HASDM-ML-DP, CHAMP-ML-DP, and MSIS-UQ-DP)
2023	Atmospheric / Thermospheric Density Modelling	Atmospheric / Thermospheric Density Data, Solar/Geomagnetic Indices	NN (FF-NN)
2023	Atmospheric / Thermospheric Density Modelling	Atmospheric / Thermospheric Density Data, Solar/Geomagnetic Indices	NN

This list showcases the increasing interest in machine learning and, more concretely, neural networks, in the last years in the field of SST. On top of that, the main subfield that has been researched is that of orbit prediction, since it is the one that is used as a basis for collision avoidance and manoeuvre planning.

The relationship between different types of data used and the area of study is also showcased, where one can also identify the main trends of data sources per task. For the field of orbit prediction, for example, the main data sources come from TLEs or orbital state data, a trend that is also observed for collision avoidance.

C

Numerical Methods

This appendix presents the different numerical methods that have been used throughout the thesis. Firstly, the numerical integration methods used in Tudat are summarised in Section C.1. Next, Section C.2 discusses the metrics used to approximate the distance between two distributions. The different activation functions are introduced in Section C.3, whereas the quaternion normalisation is briefly explained in Section C.4. Finally, Section C.5 presents an overview and mathematical formulation of the R-squared score.

C.1. Numerical Integration Methods

The approach taken in this work for satellite orbit propagation is via neural networks, which have the main purpose of avoiding the use of integrators to propagate the state of an object in space. Regardless, since Monte Carlo analyses need to be performed to verify the results from different algorithms, traditional integration methods need to be used, and hence they will be discussed and studied in this section.

C.1.1. Integrator Overview

Assume that the system can be described via a series of Ordinary Differential Equations (ODEs) in the form:

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y}, t) \tag{C.1}$$

with initial condition:

$$\boldsymbol{y}(t_0) = \boldsymbol{y}_0 \tag{C.2}$$

In such cases, $\boldsymbol{y}$ represents the state of the satellite that needs to be propagated. Since no analytical approach can be taken for this, a numerical scheme is required to approximate the solution of the ODE via discretisation of the equation. This can be represented as:

$$\bar{\boldsymbol{y}}(t_i) \simeq \boldsymbol{y}(t_i), \quad i = 0...N \tag{C.3}$$

Integrators are used for the numerical approximation of the evolution of the equations of motion affecting the satellite and are thus referred to as the solvers of the differential equations of the system. There is a wide range of integrators that could be selected which vary in terms of accuracy, computational load and numerical stability. The selection of the integrator depends on the problem that is considered, and so a wide range of integrators and time steps will be tested.

Before getting to the set of integrators to test, it is relevant to understand their behaviour and the possible sources of error in numerical integration. The error of the numerical scheme can be described as:

$$\boldsymbol{\epsilon}(t_i) = \bar{\boldsymbol{y}}(t_i) - \boldsymbol{y}(t_i) \tag{C.4}$$

There are two main sources of error:

Truncation error: As previously explained, the integrator acts as a numerical solver that approximates the solution to a set of equations that describe the dynamics of the system. The truncation error is a result of limitations in computational precision and the use of finite step sizes. This error rises with larger time steps.
One can define the local truncation error (LTE) as the truncation error made by an integrator during a single time step, whereas the global truncation error (GTE) is the result of the sum of the LTE at all steps in time.

Rounding error: This type of error arises from software implementation since computers have a finite floating-point number representation (i.e., a series of digits with which a number can be represented). The necessary rounding off to represent these numbers in the software leads to errors in the numerical solution due to discretisation at each time step. Smaller time steps are dominated by this type of error.

The rounding error is not typically dominant and has a rather random nature. Truncation error, on the other side, is more regular and has a more predictable behaviour. As mentioned, it is crucial to select an adequate time step to minimise the truncation error in the numerical scheme while not being dominated by rounding errors.

The selection of an appropriate integrator is also critical to reduce the truncation error of the model. Each integrator is said to have an *order*, which is a representation of the GTE. For example, the Euler method is said to be of first order, meaning that the GTE is proportional to the step size (represented as $O(\Delta t)$). Generally, if the GTE is of order p, then the order of the LTE is $p + 1$. Higher-order integrators can achieve a lower error at the same step size than lower-order integrators and tend to be more desirable. However, they also need a larger number of evaluation functions for the same time step and require a large computational load, therefore a trade-off between accuracy and load needs to be made.

C.1.2. Types of Integration Methods

The choice of integrator for the Monte Carlo analyses is limited by what is offered in Tudat. Amongst the integrators available, there are three main types: multi-stage, multi-step and extrapolation methods. Each of these will be described and explained below.

Multi-stage Methods

Multi-stage integrators perform multiple function evaluations of the function $\dot{\boldsymbol{y}}$ in a single time step, i.e., several 'stages' are used to propagate the state from t_i to t_{i+1}. These methods can be represented by the Runge-Kutta integrators, which have a structure as follows:

$$\bar{\boldsymbol{y}}(t_{i+1}) = \bar{\boldsymbol{y}}(t_i) + \sum_{j=1}^{N} b_j \boldsymbol{k}_j \tag{C.5}$$

In this equation, N represents the number of stages performed, b_j is the coefficient for stage j, and $\boldsymbol{k}_j$ is the stage computed at stage j (dependent on the time step and function evaluation). RK integrators are available as fixed step size methods or as variable step size. The latter is achieved via automatic step-size control of the integrator. At each time step, the integrator combines two methods of orders p and $p-1$ and compares the results obtained by each, which gives an estimate of the error in the integration for the lower-order integrator. With this estimate, the integrator can adapt the step-size to keep this error lower than a certain threshold, and once this is done the propagation is performed with either

the method of lower or higher order. Tudat offers both these methods, the former being represented by the Runge-Kutta-Fehlberg methods and the latter by the Dormand-Prince method.

Multi-step Methods

The multi-stage methods, as stated above, only use the information from a single time step to propagate the state of an object to the next time step. Multi-step methods, on the other hand, use the information of previous steps ($\bar{\boldsymbol{y}}(t_i)$, $\bar{\boldsymbol{y}}(t_{i-1})$, $\bar{\boldsymbol{y}}(t_{i-2})$, etc.) to predict the state of the object at time t_{i+1}. These methods are represented by the Adams-Bashforth (AB) (explicit) or Adams-Moulton (AM) (implicit). The explicit multi-step method follows the formulation:

$$\bar{\boldsymbol{y}}(t_{i+1}) = \bar{\boldsymbol{y}}(t_i) + \Delta t \sum_{j=0}^{N} b_j f(t_{i-j}, \bar{\boldsymbol{y}}_{i-j}) \tag{C.6}$$

$$t_{i-j} = t_i - j\Delta t \tag{C.7}$$

whereas the implicit formulation is such that:

$$\begin{aligned} \bar{\boldsymbol{y}}(t_{i+1}) &= \bar{\boldsymbol{y}}(t_i) + \Delta t \sum_{j=-1}^{N} b_j f(t_{i-j}, \bar{\boldsymbol{y}}_{i-j}) \\ &= \bar{\boldsymbol{y}}(t_i) + \Delta t \left(b_{-1} f(t_{i+1}, \bar{\boldsymbol{y}}_{i+1} + \sum_{j=0}^{N} b_j f(t_{i-j}, \bar{\boldsymbol{y}}_{i-j})) \right) \end{aligned} \tag{C.8}$$

The AM method has one main issue, and that is the requirement for the knowledge of $\bar{\boldsymbol{y}}(t_{i+1})$ on the right-hand side of the equation. Tudat offers the Adams-Bashford-Moulton (ABM) approach to deal with this by predicting $\bar{\boldsymbol{y}}(t_{i+1})$ with the AB method and applying it into the right-hand side of Equation (C.8).

Extrapolation Methods

There are some integration methods that include extrapolation within a multi-stage integrator, which are typically called extrapolation methods. The Burlisch-Stoer (BS) integrator in Tudat characterises these methods. The algorithm performs the propagation of state from t_i to t_{i+1} in a series of sub-steps, which are then extrapolated to infinite sub-steps.

C.2. Distance between two distributions

The parameter used to assess the accuracy of the DACE propagated uncertainty with respect to the Monte Carlo samples is the L_k metric. This variable evaluates the difference between two distributions P and Q such that:

$$L_k(P, Q) = \int_{\Omega} |p(\boldsymbol{x}) - q(\boldsymbol{x})|^k d\boldsymbol{x} \tag{C.9}$$

where $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ are the probability densities at point $\boldsymbol{x}$ of their respective distributions. A value of $k = 2$ is selected to evaluate the accuracy of the DACE uncertainty propagation as it can be solved analytically. To calculate the L_2 metric, the Monte Carlo samples are fitted into a Gaussian Mixture Model, for proper comparison with the one that is propagated with the DA-GMM software. The L_2 distance is calculated as follows (Horwood et al., 2011):

$$
\begin{aligned}
L_2(P,Q) &= \int_{\Omega} \left(p(\boldsymbol{x}) - q(\boldsymbol{x})\right)^2 d\boldsymbol{x} \\
&= \int_{\Omega} \left(\sum_{i=1}^{n} \alpha_i p_i(\boldsymbol{x}) - \sum_{j=1}^{m} \beta_j q_j(\boldsymbol{x}) \right)^2 d\boldsymbol{x} \\
&= \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} \int_{\Omega} p_i(\boldsymbol{x}) p_{i'}(\boldsymbol{x}) d\boldsymbol{x} + \sum_{j=1}^{m} \sum_{j'=1}^{m} \alpha_j \alpha_{j'} \int_{\Omega} q_j(\boldsymbol{x}) q_{j'}(\boldsymbol{x}) d\boldsymbol{x} \\
&\quad - 2 \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j \int_{\Omega} p_i(\boldsymbol{x}) q_j(\boldsymbol{x}) d\boldsymbol{x}
\end{aligned}
\tag{C.10}
$$

If probability density is expressed such that:

$$
p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{C.11}
$$

then the integral of the product of two Gaussian PDFs can be performed following the property:

$$
\int_{\Omega} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu'}, \boldsymbol{\Sigma'}) d\boldsymbol{x} = \mathcal{N}(\boldsymbol{\mu}; \boldsymbol{\mu'}, \boldsymbol{\Sigma} + \boldsymbol{\Sigma'}) \tag{C.12}
$$

The lower the value of L_2 is for a set of distributions, the more similar these are. On top of that, Leon Dasi (2021) establishes an error threshold of L_2=0.6, meaning that ideally the value of L_2 should be kept lower than this.

C.3. Activation Functions

Neural networks rely on the activation function of each neuron in the different hidden layers to introduce non-linearities in the model and thus be able to capture more complex patterns. In this section, some of the most common activation functions are introduced: the rectified linear unit (*ReLU*) function, the sigmoid function, and the hyperbolic tangent (*tanh*) function (Hong, 2023).

C.3.1. Rectified Linear Unit Function

The Rectified Linear Unit (ReLU) function, represented mathematically as Equation (C.13) and shown in Figure C.1a, is an activation function commonly used in neural networks due to its simplicity.

$$
a(x) = \max(0, x) \tag{C.13}
$$

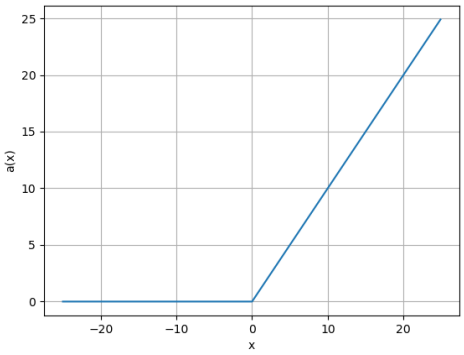
It allows positive inputs to remain unchanged while negative inputs are set to 0, which introduces nonlinearities in the model. Its simplicity makes it quite computationally efficient but has some drawbacks. The function is not differentiable at x=0, which can cause issues during the backpropagation and optimisation step.

C.3.2. Sigmoid Function

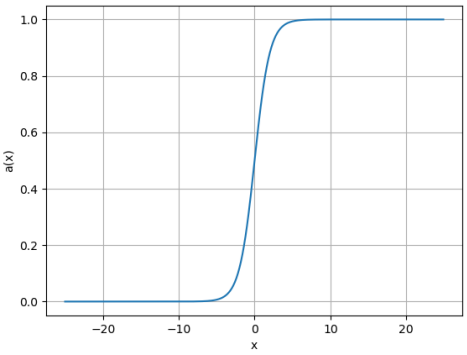
The sigmoid activation function is a function that maps any real value between 0 and 1, as can be observed in Figure C.1b. The S-shaped function follows the mathematical expression:

$$
a(x) = \frac{1}{1 + e^{-x}} \tag{C.14}
$$

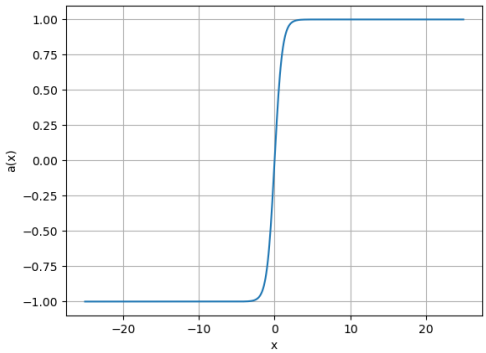
Unlike the ReLU function, the sigmoid presents smooth gradients throughout its curve, an advantage when using gradient descent optimisation methods. It is typically used for probabilistic (i.e., classification) algorithms, although it can also be used for regression. The main drawback with this function



(**a**) Rectified Linear Unit function



(**b**) Sigmoid function



(**c**) Hyperbolic Tangent function

Figure C.1: Common activation functions for neural networks

is that of vanishing gradients, by which the derivatives at very large or very low numbers become small enough to introduce issues in the learning process of the machine learning algorithm.

C.3.3. Hyperbolic Tangent Function

The hyperbolic tangent (tanh) function, mathematically defined as Equation (C.15) and shown in Figure C.1c, is similar to the sigmoid, with one main difference: it maps the inputs of the model to a range from -1 to 1. This is important as this mapping is centred around 0, which leads to a faster convergence in training.

$$a(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{C.15}$$

The main disadvantage of the tanh function is, such as the sigmoid function, that of vanishing gradients, although to a lower extent (Hong, 2023).

C.4. Quaternion Normalisation

Quaternions are typically used in astrodynamics to represent the orientation of a satellite with respect to a reference frame. In the context of this work, they are used as a state representation, together with the hodograph parameters, for the Unified State Model. Typically, the magnitude of the attitude quaternions should be equal to one:

$$\eta_0^2 + \epsilon_{01}^2 + \epsilon_{02}^2 + \epsilon_{03}^2 = 1 \tag{C.16}$$

where η_0 is the scalar part of the quaternion and $\boldsymbol{\epsilon}_0$ is the quaternion vector. It is essential for element changes that this property is maintained, however the errors induced in the model due to inaccuracies in the predicted USM7 state cause this magnitude to be different to one. To prevent this, the quaternion should be normalised:

$$\boldsymbol{\epsilon}_{normalised} = \frac{\boldsymbol{\epsilon}_0}{\sqrt{\eta_0^2 + \epsilon_{01}^2 + \epsilon_{02}^2 + \epsilon_{03}^2}} \tag{C.17}$$

$$\eta_{normalised} = \frac{\eta_0}{\sqrt{\eta_0^2 + \epsilon_{01}^2 + \epsilon_{02}^2 + \epsilon_{03}^2}} \tag{C.18}$$

This normalisation of the parameters ensures that the property of the unit magnitude of the vectors is maintained. However, it also means that an error in one of these four elements will lead to an error in all elements.

C.5. R-squared score

The predictive performance of neural networks has been assessed in this work mostly with relative or absolute errors. While this is helpful, another performance metric is introduced: the **R-squared** ($\mathbf{R^2}$) **score**, also called **coefficient of determination**, typically used in the field of statistics and machine learning.

Deep learning algorithms tend to have extremely complex architectures and results that are typically difficult to interpret. The R^2 score allows for a simple, straightforward assessment of how much of the data variability is captured by the algorithm. Its calculation depends on two different metrics: the Sum of Squared Residuals (SSR) and the Total Sum of Squares (SST).

The Sum of Squared Residuals (or Sum of Squared Errors, SSE) is defined as the sum of the squared difference between the dependent variables considered (y_i) and the values predicted by the neural network ($\hat{y}_i$)):

$$SSR = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{C.19}$$

This parameter is used as a metric to measure how much of the variability of the outcomes of the model cannot be explained by the neural network.

The Total Sum of Squares, on the other hand, is the squared sum of the differences between the actual dependent variable and the mean value across all data points:

$$SST = \sum_{i=1}^{N} (y_i - \bar{y})^2 \tag{C.20}$$

With this definition, it can be seen that this parameter is proportional to the standard deviation of a dependent variable and can be considered a measure of the total variation in that variable.

With these two elements introduced and briefly explained, the formula of the R^2 score is as follows:

$$R^2 = 1 - \frac{SSR}{SST} \tag{C.21}$$

SSR/SST represents the fraction of the dependent variable that cannot be predicted using the neural network, thus its value will be smaller the more of the variable's distribution is captured by the model. By subtracting this fraction from 1, the fraction of the total variability that is explained by the mode is obtained. The R^2 score typically takes a value between 0 and 1. The closer this parameter is to 1, the more of the dependent variable is accurately predicted by the neural network.

The R^2 score takes the combined effect of all the predictors considered. To effectively assess how each of these variables actually affects the prediction of the model, the **adjusted R-squared score** is used. This parameter considers all the predictors, and assesses how much an independent variable influences the prediction capabilities of the model. The formula for this is:

$$R^2_{adj} = 1 - \left[\frac{(1 - R^2)(n - 1)}{n - k - 1}\right] \tag{C.22}$$

where n indicates the number of observations (i.e., datapoints used for the calculation of the R^2 score) and k, the number of predictors.

One might wonder why this parameter is typically used for machine learning. Machine learning algorithms like the one developed in this work have several input and output features. The adjusted R^2 score offers a measure of how much of the outcome features of the model can be predicted with the input features. A low score might reveal that extra features are needed for accurate predictions, whereas a high score would indicate that the input features are the right ones to use for this prediction.

This parameter does have its limitations as it can only be used for linear relationships. Since the dependent variables of the model are the different elements of the state of an object in space, their relationship with the predicted elements is indeed linear. The larger the error in the prediction at a specific point, the larger the deviation from this ”best fit” line relating both variables and the lower the adjusted R^2 score.

D

Two-Line Elements

The use of TLEs for orbital dynamics was discussed in The majority of the catalogues that are available to the general public have the orbital information of the tracked objects in this format (Space-Track, 2009) and thus it is considered relevant to discuss the information they convey and their generation. This is done in Section D.1 and Section D.2, respectively.

D.1. Two-Line Element Format

All TLEs, independently on the spacecraft and epoch considered, follow the same format. These have two lines, each consisting of 69 columns or characters. The information that these contain can be used to estimate the position of that particular object at the specified epoch. An example of what a TLE looks like is given below, and the chosen catalogued RSO is the International Space Station (ISS).

```
1    ISS (ZARYA)
2    1 25544U 98067A   04236.56031392  .00020137  00000-0  16538-3 0  9993
3    2 25544  51.6335 344.7760 0007976 126.2523 325.9359 15.70406856328906
```

At first glance, it is complicated to understand what all of these represent. A table is provided below (Table D.1) to explain what each number, or block of numbers, describes concerning the orbit. In addition, to provide an example for easier understanding, the description is matched with the example TLE above for the ISS. Of particular interest for orbital dynamics are the epoch information and the Kepler or orbital elements that are included. Only this information is really needed for the state generation of the spacecraft at a specific epoch.

The values provided in the TLE elements are just a representation of the mean value of these, obtained via the removal of periodic variations. To obtain the perturbed orbital elements, the model used for the generation of the TLEs needs to be known and understood.

D.2. Generation of Two-Line Elements

The orbital information conveyed in the TLEs is generated via a standard orbit model for consistency. The most common model used for this is the Simplified General Perturbations 4 (SGP4) model (Vallado and Crawford, 2012) and is the one used by the largest TLE catalogues and thus will be the one reviewed here. The perturbations that are accounted for in this model are the gravitation of the Earth and the effects of the oblateness of the Earth, the atmospheric drag, and the solar radiation pressure. This is most accurate for satellites in near-Earth orbits with periods lower than 225 minutes. At higher altitudes, the third-body perturbation of the Sun and Moon becomes larger than the drag force, and thus the model needs to be extended. This model is the Simplified Deep Space Perturbations 4 (SDP4) and is widely used for satellites with periods larger than 225 minutes. Nevertheless, the SGP4 is a simplified perturbation model that can be used to estimate the initial state of satellites in LEO. Due to the simplifications made in this model, however, errors are introduced in the initial state estimation

Table D.1: TLE elements and description (Space-Track, 2009)

Line 0		
Columns	**Example**	**Description**
01-24	ISS (ZARYA)	Object Name
Line 1		
Columns	**Example**	**Description**
01	1	Line number
03-07	25544	Catalogue number
08	U	Classification (U: unclassified, C: classified, S: secret)
10-11	98	International Designator: last two digits of launch year
12-14	067	International Designator: launch number of the year
15-17	A	International Designator: piece of launch
19-20	04	Epoch Year: last two digits of year
21-32	236.56031392	Epoch: day of the year and fractional portion of the day
34-43	.00020137	First derivative of the mean motion
45-52	00000-0	Second derivative of the mean motion (decimal point is assumed)
54-61	16538-3	$B*$ (decimal point is assumed)
63	0	Ephemeris type
65-68	999	Element set number
69	3	Checksum
Line 2		
Columns	**Example**	**Description**
01	2	Line number
03-07	25544	Catalogue number
09-16	51.6335	Inclination [deg]
18-25	344.7760	Right Ascension of Ascending Node [deg]
27-33	0007976	Eccentricity (decimal point assumed)
35-42	126.2523	Argument of Perigee [deg]
44-51	325.9359	Mean Anomaly [deg]
53-63	15.70406856	Mean Motion [revolutions/day]
64-68	32890	Revolution Number at Epoch
69	6	Checksum

and the uncertainties in the initial state are typically in the order of 10^2 m for the position and up to 10^0 for the velocity, as shown in

D.3. Verification of the SGP4

The Space-Track API developed is used to obtain a dataset with a collection of TLEs for a period of one year. The SGP4 implementation in Python takes the mean orbital elements from the TLE and propagates them to a certain epoch, giving the estimated position and velocity at that epoch.

A part of the developed work relies on TLEs and the propagation using the SGP4 and thus there is an interest in verifying how well this works. As such, for this period in time, each TLE is propagated to the next epoch in which a TLE was published. Since the orbital elements that are propagated with the SGP4 cannot be compared directly to those from the TLE (since the former are osculating elements and the latter are mean elements), the mean elements from each TLE are transformed into osculating elements. That way, for each epoch where a TLE is reported there are two sets of data: first, the information of the osculating elements from that TLE and secondly the osculating elements propagated from the previous TLE.

The satellite chosen for this is the Molniya 3-50. The reasoning behind this is that, since the Kepler elements are being compared, using a satellite with a small eccentricity is advised against. Any orbit with an eccentricity close to 0 will have an undefined argument of periapsis and mean anomaly and

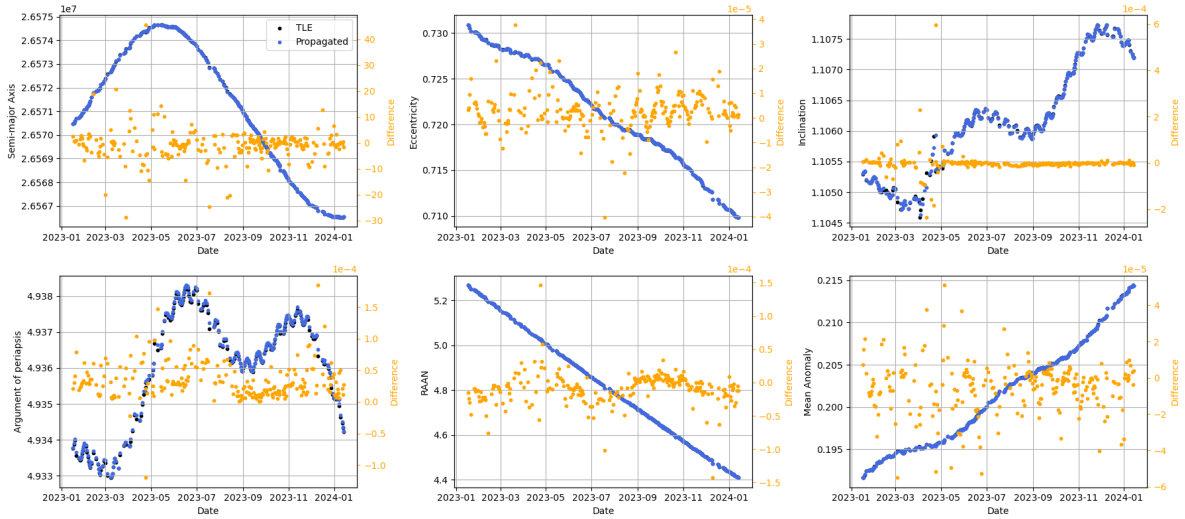


Figure D.1: Comparison of the propagated and actual TLE information at every epoch for a satellite for a year

so the results for those two elements would be inconclusive. Choosing a satellite in a highly eccentric orbit allows for a more thorough analysis of all components.

The propagated TLE information is compared to the actual TLE at each epoch, and the difference for each element is calculated. These are plotted in Figure D.1, where it can be seen that the differences are significantly small for all components of the Kepler state. This indicates that the SGP4 works well for TLE handling and propagation for the purposes of this thesis (for which TLEs are not required to be propagated for long periods of time).

E

Satellite Specifications

Several satellites have been used to assess the generalisation of the ML-based approach of this thesis and validate it across different scenarios. This chapter aims to present the basic information that has been used to obtain the initial state of each satellite.

E.1. Starlink-4437

The Starlink-4437 forms part of the Starlink constellation, the largest ever built and sent to space. Since roughly half of the currently operational satellites in space are part of this mega-constellation, it is crucial to test the algorithm on this satellite. The TLE selected is:

```
1    0 STARLINK-4437
2    1 53506U 22099AT  24015.30700671  .00001493  00000-0  12075-3 0  9992
3    2 53506  97.6540 138.3324 0003376 262.4589  97.6252 15.01270859 78308
```

The satellite is located at an altitude of approximately 550 km and has a near-circular orbit.

E.2. NAVSTAR 71 (USA 256)

The next area of interest is located in Medium-Earth Orbit (MEO). The NAVSTAR 71 is a satellite which forms part of the GPS constellation, thus there is a desire to be able to generalise the model to this object. The TLE is as follows:

```
1    0 NAVSTAR 71 (USA 256)
2    1 40105U 14045A   24015.07178368  .00000021  00000-0  00000-0 0  9999
3    2 40105  54.8654 123.7680 0026678 117.3063 242.9646  2.00555332 68340
```

With an altitude of 20,200 km, this satellite has a much larger period than Starlink, sitting at roughly 12 hours per orbit. Once again, the eccentricity is near zero.

E.3. SES 17

For the satellite in the Geostationary Earth Orbit (GEO), the SES 17 is chosen. This is a communications satellite, part of a constellation of satellites in GEO which aim to provide worldwide internet and broadcasting. Its TLE is:

```
1    0 SES 17
2    1 49332U 21095A   24015.03955240 -.00000268  00000-0  00000+0 0  9994
3    2 49332   0.0149  49.4556 0000945 250.6458 121.0165  1.00266644  8977
```

Two characteristics of the satellites in GEO are that their orbits are circular and that they sit above the Equator. This can be indeed seen in the TLE, where the eccentricity shown is small enough to be considered circular, and the inclination is also shown to be close to 0.

Table E.1: Initial state for the different satellites used to test the generalisation capabilities of the PINN

Satellite	x [km]	y [km]	z [km]
Starlink-4437	4443.6703	-3229.9398	4240.2854
Navstar 71 (USA 256)	784.6750	20425.566	-17076.9067
SES 17	28827.575	30773.551	-1.81271
Molniya 3-50	-2909.216	4732.119	-4540.287
Satellite	**vx [km/s]**	**vy [km/s]**	**vz [km/s]**
Starlink-4437	2.9478017	-3.7188840	-5.9096903
Navstar 71 (USA 256)	-2.8302688	1.7477176	1.9642153
SES 17	-2.243652	2.10215657	0.0019756786
Molniya 3-50	-2.9732517	3.67944633	5.75086185

E.4. Molniya 3-50

The last satellite that has been selected for testing the generalisation of the neural network is one in a Molniya orbit. These orbits are characterised by their large eccentricity, by which satellites go very slowly at the apogee and fast at the perigee. If the orbit was made in such a way that the apogee was located above a certain location, the satellite would spend a long time above that region. With a period of half a day to complete a full orbit, this type of satellite was better suited for communications and broadcasting. The Molniya 3-50 exhibits an orbit of such characteristics, and is selected due to the significant difference in its orbital geometry with respect to the others used for the generalisation assessment. Its TLE is presented:

```
0 MOLNIYA 3-50
1 25847U 99036A   24015.69721638  .00000056  00000-0 -83831-2 0  9994
2 25847  63.4445 252.4518 7107917 282.7109  12.3485  2.00653638179690
```

As such, its eccentricity is observed to be ≈ 0.7, meaning that, with an orbital period of 12 hours, the perigee of the orbit is located inside of LEO, whereas its apogee is higher than GEO.

For every satellite presented in this chapter, the TLE is propagated using the SGP4 built in Python to the date of January 15, 2024. These initial states are all summarised in Table E.1.

F

System Specifications

The coding and script running in this thesis has been done using three main systems: *DelftBlue*, *Eudoxos Server* (TU Delft server) and a personal laptop. Since computational load is a discussed part of the results, it is thought relevant for the reader to mention which codes have been run in each system.

SYSTEM A

DelftBlue

DelftBlue is the supercomputer from TU Delft, which has the following components:

- 228 Intel Xeon compute nodes with 48 CPU cores and 185 GB RAM each
- 10 GPU nodes with four NVIDIA Tesla V100S GPUs with 32 GB video RAM each
- 10 high memory nodes (compute nodes with more RAM)
- GNU/Linux 4.18.0, x86_64

GPUs allow for easy parallelisation of the scripts to be run, making it perfect for higher efficiency in training neural networks. As such, all the training was done using this high-performance computer.

SYSTEM B

Eudoxos Server (TU Delft)

Eudoxos is a server from the Aerospace faculty at TU Delft. This was used for most of the coding regarding the unit testing of functions and analysis of the results from the trained neural networks. It was also used for orbit propagation using Tudat. For these tasks, the server was preferred over the personal laptop as it has a higher capacity: it has a higher number of CPUs and nodes, higher RAM and more storage. These are specified below:

- 2 nodes with 28 Intel(R) Xeon(R) CPU E5-2683 v3 2.00GHz cores each
- 251GB RAM
- GNU/Linux 5.14.21, x86_64

This higher capacity tends to result in a higher efficiency of the code, and thus it was used for the aforementioned tasks.

SYSTEM C

HP Pavilion Plus Laptop 14

Lastly, a personal laptop was used to run the DA-GMM scripts. The core of the script (the Differential Algebra section) is written in C/C++ and uses Cmake to compile the code before it can be run. For this compilation and for building the code, an interface tool is required: Qt. Since Eudoxos is a remote server, and Qt was difficult to install using only the command line, DACE was run on a personal laptop with the following characteristics:

- 1 node with 10 12th Gen Intel(R) Core(TM) i7-1255U 1.70 GHz cores
- 16GB RAM
- Windows x64_64