UNIVERSIDADE
BEIRA INTERIOR

Feedback Neural Network Based Orbital Trajectory Prediction

(Versão final após defesa)

Filipe Miguel Santa Maria Senra

Dissertação para obtenção do Grau de Mestre em
Engenharia Aeronáutica
(Mestrado Integrado)

Orientador: Professor Doutor Kouamana Bousson

junho de 2023

Declaração de Integridade

Eu, Filipe Miguel Santa Maria Senra, que abaixo assino, estudante com o número de inscrição 38051 de/o Engenharia Aeronáutica da Faculdade de Engenharia, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o **Código de Integridades da Universidade da Beira Interior**.

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã, 20 de janeiro de 2023

Assinado por: **Filipe Miguel Santa Maria Senra**
Num. de Identificação: 15360199
Data: 2023.05.31 15:36:11-04'00'

(assinatura conforme Cartão de Cidadão ou preferencialmente
assinatura digital no documento original se naquele mesmo formato)

Dedicated to my family and friends

Acknowledgments

This dissertation concludes my Integrated Master's degree. I have enjoyed this time, mainly due to the people I have met along the way that helped and supported me in this journey.

First and foremost, I would like to thank my supervisor, Dr. *Kouamana Bousson*, for all the support and guidance throughout all phases of this dissertation and for introducing me to this subject. A topic capable of combining both Orbital Mechanics and Artificial Intelligence was something that I was highly interested in, allowing me to develop my mathematical and programming skills in these areas. I deeply appreciate the availability towards explaining any aspect related to the Machine Learning approach or the mathematical procedures behind the Kalman Filter. In addition, thanks to you, I was able to present my work at exciting conferences that allowed me to get more involved in the space industry. Also, I would like to extend my gratitude to Dr. *Milca Coelho*, for all the support in the implementation of the Kalman Filter and the great advise on my dissertation and career.

My sincere gratitude and appreciation towards the Department of Aerospace Sciences of Universidade da Beira Interior, in particular to the unit of Investigation LAETA UBI/AeroG, for providing me with a Research Initiation Scholarship, crucial for the development and conclusion of this dissertation.

To all the friends that made this time go by faster, thank you. I can not express enough how important you all were in making myself grow and keep up with you. Every moment shared, from study sessions, rushing group projects, to nights out, roadtrips and vacations, I will hold dearly and remember it with a smile on my face.

Finally, I would like to thank my parents *Joaquim Senra* and *Melanie Senra* , who supported me throughout all my years at the University of Beira Interior, making it possible for me to study and achieve the goal of becoming an Aeronautical Engineer, one of many. I am lucky to have you at my side, together with my brothers, *Paulo* and *Luís*, helping me succeed in every step of the way.

Resumo

Nos últimos anos, o número de satélites e lixo espacial tem aumentado perigosamente. Com isto, é indispensável que a localização e previsão de órbita destes objetos seja feita com o maior nível de precisão. Atualmente, a previsão de órbitas depende de modelos matemáticos que descrevem a física por detrás do movimento de certo objecto no espaço. Contudo, por vezes, estes modelos podem limitar a precisão da previsão de órbita por serem caracterizados por um alto grau de complexidade e não linearidade. Por outro lado, a aplicação de Machine Learning no setor espacial tem vindo a aumentar rapidamente, sendo de interesse investigar a sua aplicabilidade na área de previsão de órbitas. Na presente dissertação, uma rede neuronal Long Short-Term Memory (LSTM) é projetada e investigada. Os resultados obtidos são posteriormente comparados com os resultados obtidos por um filtro de Kalman Extendido (EKF). Com recurso a dados provenientes de um ficheiro two-line element (TLE) referente ao satélite STARLINK-1028, a órbita deste foi propagada durante 48h, produzindo 17281 vetores de estado que são utilizados para treinar a rede neuronal. Um segundo data set foi gerado, onde ruído gaussiano com uma distribuição $N(0, 100)$ foi adicionado. O propósito deste data set ruidoso é de retratar a presença de erros causados pelas medições e avaliar a robustez dos modelos. A rede neuronal foi desenvolvida com recurso à linguagem Python e às bibliotecas Tensorflow e Keras, tendo sido tomada uma abordagem Multiple Inputs Single Output (MISO). De forma a testar se a performance da rede neuronal aumenta consoante o aumento de dados disponíveis para treino, 3 casos de estudo foram criados, onde os casos de estudo A, B e C usam 41.7%, 83.3% e 100% do data set, respetivamente. Os modelos foram validados utilizando uma validação pragmática e a validação mais comum, onde se demonstra que não há sinais de overfitting ou underfitting. Resultados demonstram que os modelos são robustos face a dados ruidosos e a performance destes aumenta com o tamanho do training set. Contudo, apesar da rede neuronal ter sido validada e possuir baixos erros de previsão, o filtro de Kalman atingiu uma melhor performance.

Palavras-chave

Mecânica Orbital, Previsão de órbita, Propagação de órbita, Machine Learning, Deep Learning, Rede Neuronal, Long Short-Term Memory, Filtro de Kalman, Filtro de Kalman Extendido, Satélite LEO

Abstract

In recent years, the number of satellites and debris in space has dangerously increased. For this reason, it is indispensable that tracking and orbit prediction of these objects is performed with the highest level of accuracy. Currently, orbit prediction depends on mathematical models that describe the physics behind the movement of a certain object in space. However, at times, these models can limit the accuracy of the orbit prediction for being characterized by a high degree of complexity and non-linearity. On another note, the application of Machine Learning to the space sector has been increasing rapidly, being of interest to investigate its applicability in the field of orbit prediction. In the present dissertation, a Long Short-Term Memory (LSTM) neural network is designed and investigated. The obtained results are subsequently compared with the results obtained from an Extended Kalman Filter (EKF). With data from a two-line element (TLE) file belonging to the satellite STARLINK-1028, its orbit was propagated for 48h, producing 17281 state vectors that are utilized for training the neural network. A second data set was generated, where Gaussian noise with a distribution $N(0, 100)$ was added. The purpose of this noisy data set is to represent the presence of errors caused by measurements and assess the robustness of the models. A neural network was developed using the Python language and the Tensorflow and Keras libraries, following a Multiple Inputs Single Output (MISO) approach. To test if the performance of the neural network increases the more data is available for training, three case studies were developed, where case studies A, B and C use 41.7%, 83.3% and 100% of the data set, respectively. The models were validated using a pragmatic validation and the more common validation, where it is shown that there are no signs of overfitting or underfitting. Results demonstrate that the models are robust when faced with noisy data and their performance increases with the size of the training set. However, despite the the neural network having been validated and exhibits low prediction errors, the Kalman filter achieved a better performance.

Keywords

Orbital Mechanics, Orbit Prediction, Orbit Propagation, Machine Learning, Deep Learning, Neural Network, Long Short-Term Memory, Kalman Filter, Extended Kalman Filter, LEO Satellite

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
AU	Astronomical Unit
CNN	Convolutional Neural Network
COE	Classical Orbital Elements
ECEF	Earth Centered, Earth Fixed
EDA	Exploratory Data Analysis
EKF	Extended Kalman Filter
FNN	Feed-Forward Neural Network
GBDT	Gradient Boosting Decision Tree
GEO	Geo-Synchronous Orbit
GLONASS	Global Navigation Satellite System
GP	Gaussian Process
GPS	Global Positioning System
GRU	Gated Recurrent Unit
ILRS	International Laser Ranging System
LEO	Low-Earth Orbit
LFM	Latent Force Model
LSTM	Long Short-Term Model
MAE	Mean Absolute Error
MEO	Medium-Earth Orbit
MIMO	Multiple Input, Multiple Output
MISO	Multiple Input, Single Output
ML	Machine Learning
MSE	Mean Squared Error
NASA	National Aeronautics and Space Administration
NORAD	North American Aerospace Defense Command
OD	Orbit Determination
OP	Orbit Prediction
RMSprop	Root Mean Squared propagation
RNN	Recurrent Neural Network
RSO	Resident Space Object
SGD	Stochastic Gradient Descent
SGP	Simplified General Perturbations Model
SRP	Solar Radiation Pressure
SVM	Support Vector Machine
TEME	True Equator, Mean Equinox
TLE	Two-Line Element

Nomenclature

Symbol	Description	Units
α, RA	Right Ascension	$[s]$
δ, Dec	Declination	$[^\circ]$
θ	True Anomaly	$[^\circ]$
λ	Eigenvalue	$[-]$
μ	Gravitational Parameter	$[m^3/s^2]$
$\tilde{\sigma}$	Relative Error	$[m]$
σ	Standard Deviation	$[m]$
χ	Universal Anomaly	$[-]$
Ω	Right Ascension of the Ascending Node	$[^\circ]$
ω	Argument of Perigee	$[^\circ]$
$\hat{\mathbf{u}}_r$	Unit Vector	$[-]$
$\hat{x}_k$	*a posteriori* State Estimate	$[m]$
$\hat{x}_k^-$	*a priori* State Estimate	$[m]$
A	State Transition Matrix	$[-]$
$\mathbf{a}$	Acceleration Vector	$[m/s^2]$
a	Semimajor Axis	$[m]$
B	Control Matrix	$[-]$
$C(z), S(z)$	Stumpff functions	$[-]$
C_t	Cell State	$[-]$
e	Eccentricity	$[-]$
e_{pred}	Prediction Error	$[m]$
f, g	Lagrange coefficients	$[-]$
H	Observation Matrix	$[-]$
h	Specific Angular Momentum	$[m^2/s]$
h_t	Hidden State	$[-]$
I	Identity Matrix	$[-]$
i	Inclination	$[^\circ]$
K	Kalman Gain	$[-]$
M	Mean Anomaly	$[^\circ]$
P	Error Covariance Matrix	$[-]$
P_k^-	*a priori* Error Covariance Matrix	$[-]$
$\mathbf{p}$	Net Perturbative Acceleration Vector	$[m/s^2]$
Q	Process Noise Covariance Matrix	$[-]$
$\mathbf{r}$	Position Vector	$[m]$
r	Magnitude of the Position Vector	$[m]$
R	Measurement Noise Covariance Matrix	$[-]$
t	Time	$[s]$
V	Measurement Noise Jacobian Matrix	$[-]$
$\mathbf{v}$	Velocity Vector	$[m/s]$

Symbol	Description	Units
v_x, v_y, v_z	Relative Velocity Components	$[m/s]$
W	Process Noise Jacobian Matrix	$[-]$
x_t	State Vector	$[-]$
X, Y, Z	Relative Position Components	$[m]$
y_t	Output Vector	$[m]$
z_t	Measurement Vector	$[m]$

Chapter 1

Introduction

1.1 Motivation

In recent years, the number of satellites and debris in space has dangerously increased. In just two years, the number of active and defunct satellites in Low-Earth Orbit (LEO) has increased by over 50% . Space debris aside, companies like SpaceX, OneWeb and Amazon are placing satellites in orbit at an unprecedented rate to build 'mega-constellations' of communication satellites, increasing the risk of conjunctions and collisions [4]. For this reason and to avert a chain reaction of collisions, termed Kessler Syndrome, it is indispensable to accurately track and predict space debris and satellites' orbits alike.

Current physics-based methods have errors in the order of kilometres for 7 days predictions. Typically, this failure is due to uncertainty around the state of the space object at the beginning of the trajectory, forecasting errors in environmental conditions such as atmospheric drag, as well as specific unknown characteristics [5]. Leveraging data-driven techniques, namely machine learning, the orbit prediction accuracy can be enhanced by the superior abstraction capacity that Deep Learning models have of modelling highly complex nonlinear systems from large amounts of observed data.

Being an advocate of both fields of orbital mechanics and machine learning, having the opportunity to join both in one topic motivated me to contribute to the research in this field. Also, by having no previous experience with machine learning, I found it challenging to understand how far one could go as a beginner in such an extensive area of expertise.

1.2 Objectives

In every machine learning application, the main part of the process is the algorithm that trains the neural network and performs the predictions. Thus, the main objective of this dissertation is to develop an algorithm, written in Python, that can compute a LSTM Neural Network capable of accurate orbit prediction, followed by its validation. Then, the development of an algorithm capable of computing an EKF is to be performed, as to compare the results obtained from both approaches. To achieve the described objectives, the following tasks shall be fulfilled:

1. Study of previous research on the topic to acquire knowledge and understanding of the current state of the art;

2. Review of the theoretical fundamentals and techniques underling the research;

3. Creation of a Python algorithm capable of computing a LSTM Neural Network that performs accurate orbit prediction;

4. Validation of the LSTM Neural Network;

5. Performance assessment of the LSTM Neural Network;

6. Creation of an algorithm capable of computing an EKF;

7. Comparison of both approaches and drawing of conclusions.

1.3 Dissertation Layout

This dissertation is divided into five chapters.

In the current chapter, the motivations and objectives proposed for the development of this dissertation are outlined, as well as the overall structure of the document. Then, the current state of the art of the technology related to orbit prediction using both classical methods and machine learning is presented, including a bibliographic review.

The second chapter reviews various fundamental theoretical notions and foundations, mainly on orbital mechanics and orbit prediction, artificial intelligence and the Kalman Filter, essential for the comprehension of the work developed in this dissertation.

The third chapter describes the methodology of the work, which encompasses the data and the coordinate system utilized, the framework to be employed in the various case studies and the validation of the neural network, as well as the LSTM configuration and its training parameters.

In the fourth chapter, the results are presented and analysed, focusing firstly in the neural networks' performance and secondly in the comparison of its results with the one's obtained by the EKF.

The fifth and final chapter encompasses the main conclusions of this dissertation, summarizing the work conducted throughout this research period. Limitations encountered are also presented, as well as a proposal for future work.

1.4 State of the Art

Orbit prediction can be explained as the process of propagating a known orbital state to a future one. The current physics-based methodology is to solve the differential equations of

motion either analytically or numerically to predict the mean state at a future time. Numerical methods are time consuming and the sheer amount of space debris objects orbiting Earth makes them usable only for academic or non-real time applications [6]. Since orbital mechanics is particularly hard to solve due to the non-linearity of the perturbed dynamics, orbit prediction's numerical approximations usually entail a set of assumptions to ease the difficulty of the task.

The most common approach for orbit prediction is described as the propagation of a Gaussian distribution, according to a first-order system of Stochastic Differential Equations (SDEs) that was obtained from previous Orbit Determination. The most relevant numerical integrators used for orbit propagation found in the literature include explicit Runge-Kutta methods, and predictor-corrector methods, namely Adams-Bashforth-Moulton (ABM) and Gauss-Jackson (GJ) [5].

Analytical methods are more common than numerical methods, and of those, the widely used method is the SGP4 (Simplified General Perturbations Model 4) propagator [7]. Since it was made available to the public in 2006 [8], it has become the most used propagator for orbit prediction, in spite of the lack of accuracy for long-time predictions. The input data for the SGP4 propagator is a TLE file that includes information regarding the object and its orbit, such as satellite number, mean orbital elements, drag, ballistic coefficient, revolution number and a time stamp.

When performing orbit propagation using current methods, two types of errors arise: satellite force model's errors and measurement errors. The first error type relates to simplifications in the model, such as Solar's Radiation Pressure model, or unknown information about the Resident Space Object (RSO), like shape, attitude or cross-sectional area. Tables 1.1 and 1.2 present a summary of error sources in orbit prediction, in accordance with [9, 10].

Table 1.1: Satellite Force Model's Error Sources in Orbital Prediction.

Gravitation parameters	Non-conservative Forces
Earth's mass	Drag (atmospheric density)
Geopotential coefficients	Solar and Earth radiation pressure
Solid Earth and tides perturbations	Earth's magnetic field
Moon gravitational perturbation	
Light-time correction	

Measurement errors relate to all errors associated with the difference between the real state of the satellite (position and velocity, for example) and the measured state, including numerical truncation errors, as well as other navigation errors caused by coordinate systems precision or clock accuracy.

In recent years, data-driven techniques are being used to tackle both sides of the problem, by either employing machine learning to improve the model's representation of reality or to mitigate the errors caused by numerical solutions and linear assumptions. Levit and Mar-

Table 1.2: Measurement Model's Error Sources in Orbital Prediction.

Coordinate Systems	Measurements
Precession and nutation	Numerical and analytical methods
Polar motion	Instrument modeling
Coordinates of tracking stations	Atmospheric refraction
	Clock accuracy

shall [11] showed that with a sufficient number of TLE files, these could be used as pseudo-observations to fit a high-precision special perturbations numerical propagator. When applied to a set of satellites, the orbit prediction error was reduced from 1.5 km/day to 0.1 km/day, showing that past information can be leveraged to improve orbit prediction. This is a key factor for any machine learning setup. This work was closely followed by [12], where the same method was applied, with the addition of a bias correction function. Furthermore, the use of error correction functions based on prediction error periodicity has led authors such as [13, 14] to improve SGP4's prediction.

One other approach is to remove the physics-based propagator entirely. In [15], a data-driven model able to approximate the SGP4 algorithm using a polynomial fit was proposed. In [16], the use of Latent Force Models (LFMs) was introduced to combine the physical models principles with non-parametric data-driven components. This method allows the determination of future orbit positions and the corresponding uncertainty by assuming Gaussian process (GP) priors on unknown forces. Rautalin et al. [17] extended this work and obtained positive results for a set of satellite constellations in Medium Earth Orbit (MEO) and Geo-Synchronous Orbit (GEO).

Peng and Bai have a substantial number of publications in this field. In their first paper [18] within the framework of error correction, a Support Vector Machine (SVM) is used to learn from historical prediction errors and improve the SGP4-based prediction. Applied to a simulated catalogue, their error correcting model has shown to be capable of being deployed to improve RSO's orbit prediction. In further publications [19, 20], they have also examined three ML models and their capabilities. The three ML models were Artificial Neural Networks (ANNs), Gaussian Processes (GP) and SVMs, concluding that the ANN yielded the best results, despite being more prone to overfitting. In [21] the SVM model developed in previous research was evaluated on a real data set. Recently, the authors proposed a data fusion approach to combine the uncertainty information from the EKF orbit determination process with the one obtained from the GP model [22].

Using a ML algorithm to correct errors from a physics based model is undoubtedly the most common approach, having various neural network architectures been leveraged for this purpose. Four independent works show that LSTMs [23], Recurrent Neural Networks (RNNs) [24], Feed-Forward Neural Networks (FNN) [25], and Convolutional Neural Networks (CNNs) [26] can be utilized to improve SGP4's orbit prediction. Similarly, Li et al. [27, 28] increased the accuracy of the along-track direction prediction by more than 75% for five satellites in

LEO and GEO using a Gradient Boosting Decision Tree (GBDT) and a CNN. All previous work mentioned used publicly available satellite data from the International Laser Ranging System (ILRS) as the ground truth and a set of TLE files for a limited number of satellites as the training data. Although extensive work has been done is this line of research, there are still two objectives that have not been achieved. One is the generalisation of machine learning models to unseen RSOs (referred to as a type III generalisation by Peng and Bai [29]), so that the development of a machine learning model for every RSO is no longer necessary. The second objective is the use of variables that are not provided by a TLE file's information.

In conclusion, two sources of error exist when performing orbit prediction. Therefore, two lines of work can be pursued to enhance a physics-based model. The research that is previously shown, very much like this dissertation, has a focus on correcting numerical and observational errors that occur when using simplified assumptions. A different approach is to use ML to improve our knowledge of the Earth, space and RSOs, thus approximating these models to reality.

Chapter 2

Theoretical Background

2.1 Overview

In order to proceed further, it is necessary to present and go over some fundamental theoretical notions and foundations, on which the remainder of this study is built upon.

This chapter begins with an overview of orbital mechanics with a focus on the dynamics of an Earth-orbiting satellite, as well as reference frames. It is followed by a description of the main methods and perturbations used to perform orbital prediction, alongside a detailed review of the Kalman Filter. Finally, the section on Artificial Intelligence covers everything from a brief introduction to the topic, to the area of deep learning and, more specifically, the LSTM architecture.

2.2 Orbital Mechanics

This section follows the content shared in [1] and [30].

At any given time, the state vector of a satellite comprises its velocity $\mathbf{v}$ and orbital acceleration $\mathbf{a}$. Orbital mechanics is concerned with specifying or predicting state vectors over intervals of time. It is known that the equation governing the state vector of a satellite traveling around the earth is, under the familiar assumptions,

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

$\mathbf{r}$ is the position vector of the satellite relative to the center of the Earth and μ the gravitational parameter. The components of $\mathbf{r}$ and, especially, those of its time derivatives $\dot{\mathbf{r}} = \mathbf{v}$ and $\ddot{\mathbf{r}} = \mathbf{a}$, must be measured in a nonrotating frame attached to the earth. A commonly used non rotating right-handed Cartesian coordinate system is the geocentric equatorial frame shown in Figure 2.1. The X-axis points in the vernal equinox direction. The XY plane is the earth's equatorial plane, and the Z-axis coincides with the earth's axis of rotation and points northward. The unit vectors $\hat{\mathbf{I}}$, $\hat{\mathbf{J}}$, and $\hat{\mathbf{K}}$ form a right-handed triad. The non rotating geocentric equatorial frame serves as an inertial frame for the two-body earth satellite problem, as embodied in Equation (2.1). It is not truly an inertial frame, however, since the center of the

earth is always accelerating toward a third body, the sun (not to mention the moon), it is a fact that is ignored in the two-body formulation.

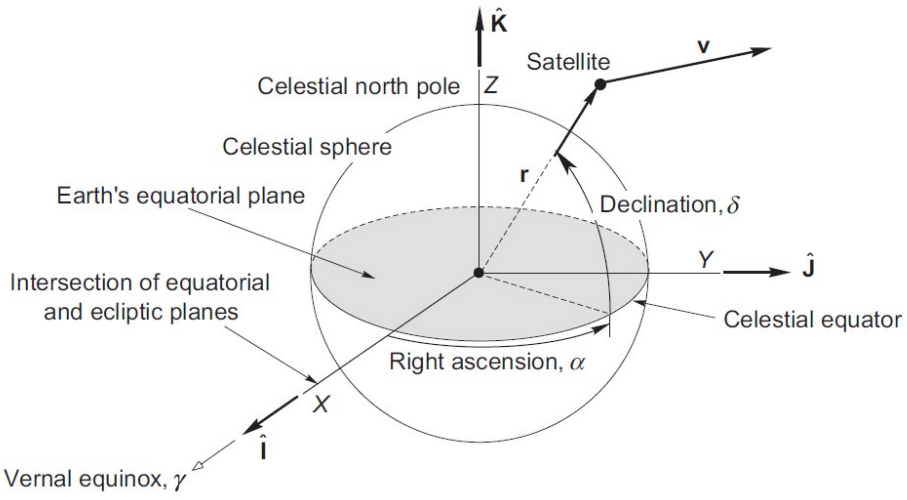


Figure 2.1: The geocentric equatorial frame.

In the geocentric equatorial frame, the state vector is given in component form by

$$\mathbf{r} = X\hat{\mathbf{I}} + Y\hat{\mathbf{J}} + Z\hat{\mathbf{K}} \tag{2.2}$$

$$\mathbf{v} = v_X\hat{\mathbf{I}} + v_Y\hat{\mathbf{J}} + v_Z\hat{\mathbf{K}} \tag{2.3}$$

If r is the magnitude of the position vector, then

$$\mathbf{r} = r\hat{\mathbf{u}}_r \tag{2.4}$$

Figure 2.1 shows that the components of $\hat{\mathbf{u}}_r$ (the direction cosines l, m, and n of $\hat{\mathbf{u}}_r$) are found in terms of the Right Ascension (RA) α and Declination (Dec) δ as follows

$$\hat{\mathbf{u}}_r = l\hat{\mathbf{I}} + m\hat{\mathbf{J}} + n\hat{\mathbf{K}} = \cos\delta\cos\alpha\hat{\mathbf{I}} + \cos\delta\cos\alpha\hat{\mathbf{J}} + \sin\delta\hat{\mathbf{K}} \tag{2.5}$$

From this, it is possible to see that the Dec is obtained as $\delta = \sin^{-1} n$. There is no quadrant ambiguity since, by definition, the Dec lies between $-90°$ and $+90°$, which is precisely the range of the principal values of the $\arcsin$ function. It follows that $\cos\delta$ cannot be negative. Equation (2.5) also reveals that $l = \cos\delta\cos\alpha$. Hence, the RA is found from $\alpha =$

$\cos^{-1}(l/\cos\delta)$, which yields two values of α between 0° and 360°. To determine the correct quadrant for α, the sign of the direction $m = \cos\delta\cos\alpha$ is checked. Since $\cos\delta$ cannot be negative, the sign of m is the same as the sign of $\sin\alpha$. If $\sin\alpha > 0$, then α lies in the range $0^\circ - 180^\circ$, whereas if $\sin\alpha < 0$ it means that α lies between 180° and 360°.

Although the position vector furnishes the RA and Dec, the RA and Dec alone do not furnish **r**. For that, the distance r is needed in order to obtain the position vector from Equation (2.4).

If the state vector (**r**$_0$,**v**$_0$) at a given instant is provided, then the state vector at any other time in terms of the initial vector can be determined by means of the expressions

$$\begin{aligned} \mathbf{r} &= f\mathbf{r}_0 + g\mathbf{v}_0 \\ \mathbf{v} &= \dot{f}\mathbf{r}_0 + \dot{g}\mathbf{v}_0 \end{aligned} \tag{2.6}$$

where the Lagrange coefficients f and g and their time derivatives are given by

$$\begin{aligned} f &= 1 - \frac{\chi^2}{r_0}C(\alpha\chi^2) \\ g &= \Delta t - \frac{1}{\sqrt{\mu}}\chi^3 S(\alpha\chi^2) \\ \dot{f} &= \frac{\sqrt{\mu}}{rr_0}[\alpha\chi^3 S(\alpha\chi^2) - \chi] \\ \dot{g} &= 1 - \frac{\chi^2}{r}C(\alpha\chi^2) \end{aligned} \tag{2.7}$$

where χ is the universal anomaly and $C(z)$ and $S(z)$ are Stumpff functions [31]. These equations specify the total of six components of $\mathbf{r}_0$ and $\mathbf{v}_0$ that completely determine the size, shape, and orientation of an orbit.

2.2.1 Orbital elements and the state vector

To define an orbit in the plane, two parameters are required: eccentricity and angular momentum. Other parameters, such as the semimajor axis, the specific energy, and (for an ellipse) the period, are obtained from these two. To locate a point on the orbit requires a third parameter, the true anomaly, which leads to the time since perigee. Describing the orientation of an orbit in three dimensions requires three additional parameters, called the Euler angles, illustrated in Figure 2.2.

First, the intersection of the orbital plane with the equatorial (XY) plane is located. This line

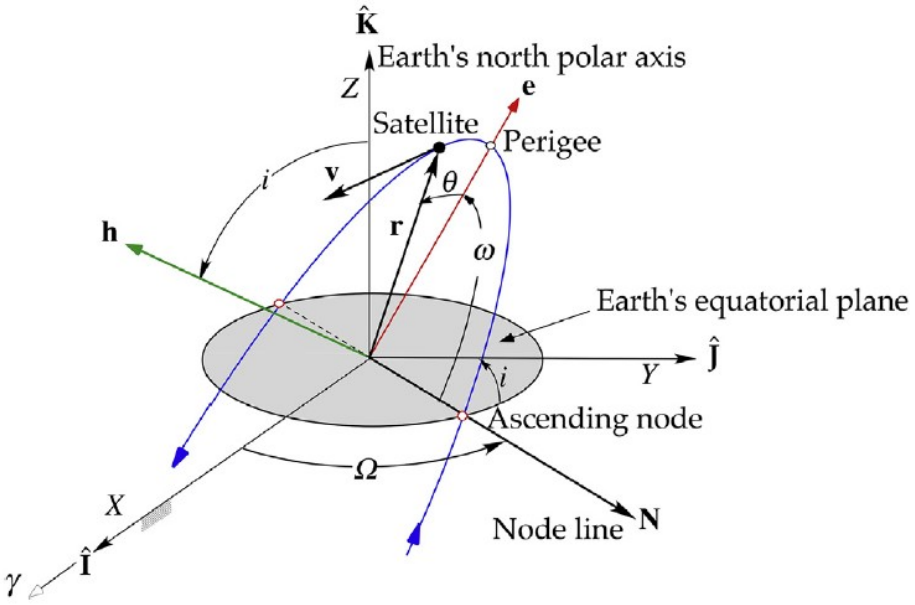


Figure 2.2: Geocentric equatorial frame and the orbital elements.

is called the node line. The point on the node line where the orbit passes above the equatorial plane from below it is called the ascending node. The node line vector **N** extends outward from the origin through the ascending node. At the other end of the node line, where the orbit dives below the equatorial plane, is the descending node. The angle between the positive X-axis and the node line is the first Euler angle Ω, the RA of the ascending node, that lies between $0°$ and $360°$.

The dihedral angle between the orbital plane and the equatorial plane is the inclination i, measured according to the right-hand rule, that is, counterclockwise around the node line vector from the equator to the orbit. The inclination is also the angle between the positive Z-axis and the normal to the plane of the orbit. The two equivalent means of measuring i are indicated in Figure 2.2. The angular momentum vector **h** is normal to the plane of the orbit. Therefore, the inclination i is the angle between the positive Z-axis and **h**. The inclination is a positive number between $0°$ and $180°$.

It remains to locate the perigee of the orbit. The perigee lies at the intersection of the eccentricity vector **e** with the orbital path. The third Euler angle ω, the argument of perigee, is the angle between the node line vector **N** and the eccentricity vector **e**, measured in the plane of the orbit. The argument of perigee is a positive number between $0°$ and $360°$.

In summary, the six orbital elements are

- h: specific angular momentum.
- i: inclination.

- Ω: right ascension of the ascending node.
- e: eccentricity.
- ω: argument of perigee.
- θ: true anomaly.

The angular momentum h and true anomaly θ are frequently replaced by the semimajor axis a and the mean anomaly M, respectively.

2.2.2 Reference frames

Reference frames are required to express the position and velocity of a spacecraft. The focus of this work are objects orbiting Earth, therefore reference frames that are centred at the Earth's center of mass, called geocentric reference frames, are used. Here, very much like the reference frame presented earlier, five types of geocentric reference frames are to be highlighted, namely

Inertial an inertial reference frame is fixed with respect to the stars. The inertial geocentric frame most commonly used is the J2000 frame, defined with respect to the Earth's mean equator and equinox at midday on 1 January 2000. The X-axis is aligned with the mean equinox, the Z-axis normal to the mean equator pointing North and the Y-axis completes the right-handed frame.

Mean of Date this quasi-inertial frame is fixed with respect to the mean equinox and mean equator, also accounting for the procession of Earth's rotation axis.

True of Date this quasi-inertial frame is fixed with respect to the true equinox and true equator, also accounting for both the procession and nutation of Earth's rotation axis.

True equator, Mean equinox (TEME) frame that is used for TLE data. The Z-axis points in the direction of the true rotation axis of the Earth, while the X-axis points into the direction of the mean vernal equinox at the considered time.

Earth-centered, Earth-fixed (ECEF) rotating reference frames that accounts for both the procession and nutation of Earth's rotation axis, as well as Earth's rotation. It is defined with both X and Y axes in the equatorial plane and the Z-axis parallel to the instantaneous Earth's rotation axis, pointing North. The X-axis points in the Greenwich mean meridian direction and the the Y-axis is orthogonal to both of them. The computation of the instantaneous Earth's rotation axis is done using Earth's orientation parameters. This is the reference frame of choice for this dissertation, as it allows for a simplified orbital trajectory computation.

2.2.3 Coordinates

The use of different coordinates, or so-called element sets, can be beneficial when describing the motion of a spacecraft. It can provide better insight into the dynamics of the problem or even reduce its non-linearity. The most commonly used sets of coordinates are

Cartesian coordinates the Cartesian coordinates (x, y, z) are three perpendicular coordinates aligned with the X, Y, and Z axes, respectively. Together with their time derivatives $(\dot{x}, \dot{y}, \dot{z})$ this coordinate set can be used for orbit prediction.

Cylindrical coordinates the cylindrical coordinates (ρ, φ, z) are defined by the distance from the Z-axis ρ, the azimuth angle φ and the height z. Together with their time derivatives $(\dot{\rho}, \dot{\varphi}, \dot{z})$ this coordinate set can be used for orbit propagation.

Classical orbital elements the classical orbital elements (COE), given by [32], were covered previously. The set $(h, e, i, \Omega, \omega, \theta)$ is also called the Keplerian orbital elements.

2.3 Orbit Prediction

This section will continue to follows the content shared in [1] and [30], with the addition of [33].

To safeguard active spacecraft activity, it is necessary to accurately determine where each RSO is, and where it will be, at all times. To create such a complex body of knowledge, the larger problem of orbit estimation is divided into smaller problems, each one largely complex but with a specific goal in mind. Three main sub-problems can be identified (orbit determination, orbit prediction and Thermospheric Mass Density), of which orbit prediction will be covered in this section.

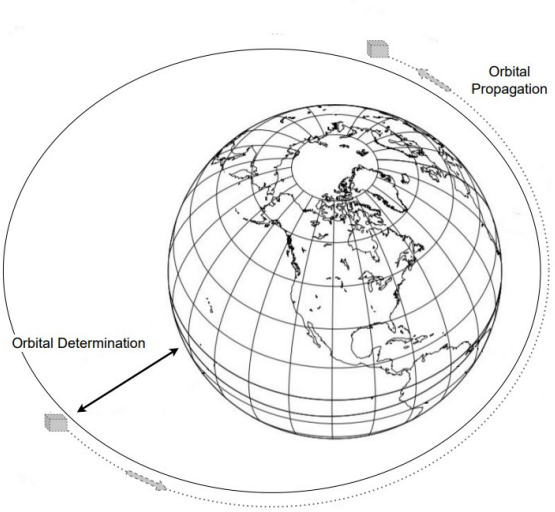


Figure 2.3: The two main stages of orbital estimation necessary for spacecraft tracking. The information obtained from OD is used as input for OP.

Orbit prediction, or orbit propagation, consists of the prediction of the orbital characteristics of a satellite at some future date given the current orbital characteristics. Owing to a host of perturbations acting on a satellite, its orbital elements change with time. The perturbations that act upon a spacecraft can be divided into natural phenomena and those associated with satellite operations. Among the former are the effects of the Earth's gravitational potential, atmospheric effects, the gravitational attraction of the Sun, Moon, and Planets, etc., while the latter consist of the effects of spacecraft manoeuvres.

Orbit propagation, in some form or other, is carried out during most of the mission analysis activities. In many cases the problem can be simplified, and only the effects of the dominant perturbations considered. The literature covering the perturbations that act upon an Earth-centered satellite, and the methods used to evolve the orbit, is extensive. In addition, various orbit integration methods have been developed in the search for greater accuracy, stability, and speed.

At this point, it is useful to define a perturbation. A perturbation is a deviation from some normal or expected motion. Contrary to the usual connotation of the word, it should not be supposed that perturbations are always small, for their effect can be as large as, or larger than, that of the primary attracting force. For example, many interplanetary missions would miss their targets entirely if the perturbing effects of other attracting bodies were not taken into account. Ignoring the effect of the Earth's oblateness on a low altitude satellite would cause a very large error in position prediction over a long period of time.

There are two main categories of propagation techniques, referred to as *special perturbations* and *general perturbations*. Special perturbation techniques deal with the direct numerical integration of the equations of motion, including all the necessary perturbing accelerations. General perturbation techniques involve an analytic integration of series expansions of the perturbing accelerations. This section focuses on the former rather than the latter. Examples of special perturbation techniques are:

- Cowell's Method.
- Encke's Method.
- Variation of Parameters Method.

Getting into the details of these techniques, it was shown previously that Keplerian orbits are the closed-form solutions of the two-body equation of relative motion (Equation (2.1))

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} \tag{2.8}$$

This equation is based on the assumption that there are only two objects in space, and that

their spherically symmetric gravitational fields are the only source of interaction between them. Any effect that causes the motion to deviate from a Keplerian trajectory is a perturbation. To account for perturbations, term **p** is added to the right-hand side of Equation (2.8) to get

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} + \mathbf{p} \tag{2.9}$$

The vector **p** is the net perturbative acceleration from all sources other than the spherically symmetric gravitational attraction between the two bodies. The magnitude of **p** is usually small compared to the primary gravitational acceleration $a_0 = \mu/r^2$. An exception is atmospheric drag which, at an altitude of about 100 km, is large enough to deorbit a satellite. The drag effect decreases rapidly with altitude and becomes negligible ($p_{drag} < 10^{-10}a_0$) above 1000 km.

Starting with a set of initial conditions ($\mathbf{r}_0$,$\mathbf{v}_0$) and the functional form of the perturbation **p**, Equation (2.9) can be integrated to find the position r and velocity v at any time thereafter.

2.3.1 Cowell's Method

Philip H. Cowell (1870–1949) was a British astronomer whose name is attached to the method of direct numerical integration of Equation (2.9). Cowell's work at the turn of the twentieth century relied entirely upon hand calculations to numerically integrate the equations of motion using classical methods dating from Isaac Newton's time. Today, of course, with high-speed digital computers, extremely accurate integration algorithms can be easily implemented.

Cowell's method is, conceptually, the simplest technique, and involves the numerical integration of the satellite's equations of motion, with all the perturbations being considered as accelerations applied to the satellite. The main advantage of the method is the simplicity of formulation and implementation, and the fact that any number of perturbations can be handled simultaneously. But there are disadvantages also. When the motion is near a large attracting body, smaller integration steps must be used, which adversely affects execution time and accumulated error due to round-off.

Classically, and in most current applications, this method has been applied in a Cartesian coordinate system. However, it has been found that some improvement can be realised by formulating the problem in polar or spherical coordinates. In this case **r** will tend to vary slowly, and the angle change will often be monotonic, allowing the use of larger step sizes for the same round-off error.

For the two-body problem with perturbations, the equation of motion can be written as

$$\ddot{\mathbf{r}} + \frac{\mu}{r^3}\mathbf{r} = \mathbf{p} \tag{2.10}$$

For numerical integration this would reduce to two first-order differential equations

$$\dot{\mathbf{r}} = \mathbf{v} \tag{2.11}$$

$$\dot{\mathbf{v}} = \mathbf{p} - \frac{\mu}{r^3}\mathbf{r} \tag{2.12}$$

In cartesian coordinates this is further broken down into the following

$$\begin{aligned} \dot{x} = v_x \quad &; \quad \dot{v_x} = p_x - \frac{\mu}{r^3}x \\ \dot{y} = v_y \quad &; \quad \dot{v_y} = p_y - \frac{\mu}{r^3}y \\ \dot{z} = v_z \quad &; \quad \dot{v_z} = p_z - \frac{\mu}{r^3}z \end{aligned} \tag{2.13}$$

where $r^2 = x^2 + y^2 + z^2$.

The perturbing acceleration, **p**, is the vector sum of all the perturbations acting on the satellite. For example, If the effect of the Moon were to be considered, **p** would be given by

$$\mathbf{p} = \mu_m\left(\frac{\mathbf{r}_{ms}}{r_{ms}^3} - \frac{\mathbf{r}_{me}}{r_{me}^3}\right) \tag{2.14}$$

where μ_m is the gravitational parameter of the Moon, $\mathbf{r}_{ms}$ is the satellite-to-Moon radius vector and $\mathbf{r}_{me}$ is the Earth-to-Moon radius vector. Once the perturbing potential has been defined through an analytic formulation, the state vector can be evolved by applying one of the many available numerical integration schemes. Once the state vector is obtained, the instantaneous orbit elements can be determined in a simple manner.

2.3.2 Encke's Method

In this method, developed originally by the German astronomer Johann Franz Encke (1791–1865), the two-body motion due solely to the primary attractor is treated separately from that due to the perturbation. The two-body osculating orbit $\mathbf{r}_{osc}(t)$ is used as a reference orbit upon which the unknown deviation $\delta\mathbf{r}(t)$ due to the perturbation is superimposed to

obtain the perturbed orbit $\mathbf{r}(t)$.

Let ($\mathbf{r}_0$,$\mathbf{v}_0$) be the state vector of an orbiting object at time t_0. The osculating orbit at that time is governed by Eqn (2.8),

$$\ddot{\mathbf{r}}_{osc} = -\frac{\mu}{r^3_{osc}}\mathbf{r}_{osc} \tag{2.15}$$

with the initial conditions $\mathbf{r}_{osc}(t_0) = \mathbf{r}_0$ and $\mathbf{v}_{osc}(t_0) = \mathbf{v}_0$. For times $t > t_0$, the state vector ($\mathbf{r}_{osc}$,$\mathbf{v}_{osc}$) of the osculating, two-body trajectory may be found analytically using the Lagrange coefficients from Equations (2.6),

$$\begin{aligned} \mathbf{r}_{osc}(t) &= f(t)\mathbf{r}_0 + g(t)\mathbf{v}_0 \\ \mathbf{v}_{osc}(t) &= \dot{f}(t)\mathbf{r}_0 + \dot{g}(t)\mathbf{v}_0 \end{aligned} \tag{2.16}$$

After the initial time t_0, the perturbed trajectory $\mathbf{r}(t)$ will increasingly deviate from the osculating path $\mathbf{r}_{osc}(t)$, so that, as illustrated in Figure 2.4,

$$\mathbf{r}(t) = \mathbf{r}_{osc}(t) + \delta\mathbf{r}(t) \tag{2.17}$$

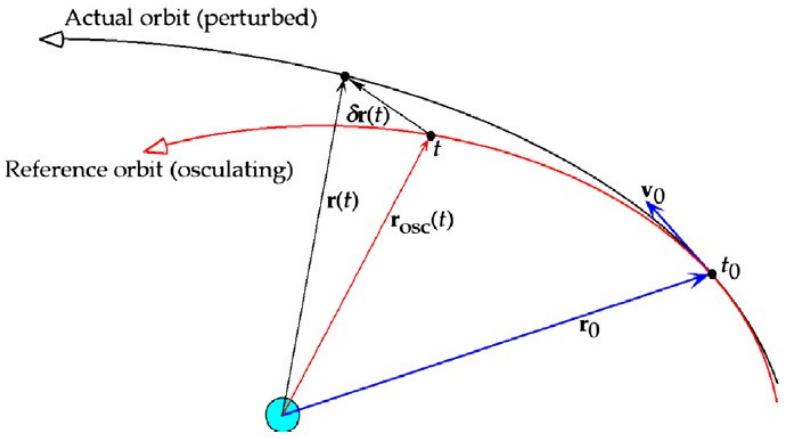


Figure 2.4: Perturbed and osculating orbits.

Substituting $\mathbf{r}_{osc} = \mathbf{r} - \delta\mathbf{r}$ into Equation (2.15), setting $\delta\mathbf{a} = \delta\ddot{\mathbf{r}}$, further substituting Equation (2.9) into the expression and a final arrangement of the terms yields

$$\delta\mathbf{a} = -\frac{\mu}{r_{osc}^3}\left[\delta\mathbf{r} - \left(1 - \frac{r_{osc}^3}{r^3}\right)\mathbf{r}\right] + \mathbf{p} \tag{2.18}$$

In this method, Equation (2.18) is integrated to obtain the deviation $\delta\mathbf{r}(t)$. This is added to the osculating motion $\mathbf{r}_{osc}(t)$ to obtain the perturbed trajectory from Equation (2.17). If at any time the ratio $\delta\mathbf{r}/r$ exceeds a preset tolerance, then the osculating orbit is redefined to be that of the perturbed orbit at time t. This process, called rectification, is illustrated in Figure 2.5.

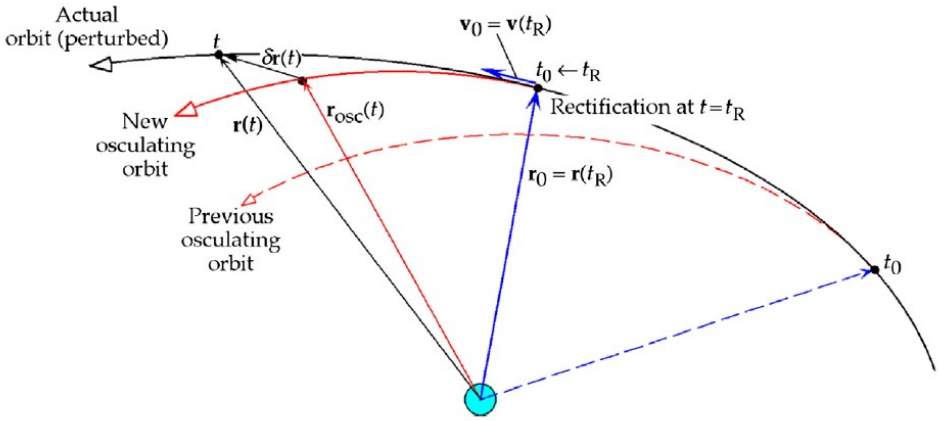


Figure 2.5: Reference orbit resetting at time t_R.

2.3.3 Variation of Parameters Method

This was the earliest successful method of orbit propagation and was first developed by Euler in 1748. Whereas the *Cowell* and *Encke* methods are concerned with calculating the satellite's position and velocity directly, the*Variation of Parameters* method calculates the orbital elements or any other consistent set of parameters which adequately describes the orbit. A two-body orbit can be described by any consistent set of six parameters since it is described by three second-order differential equations. The classical orbital elements are only one of many possible sets. The essence of the method is to find how the selected set of parameters varies with time due to the perturbations. This is done by finding analytic expressions for the rate-of -change of the parameters in terms of the perturbations. These expressions are then integrated numerically to find their values at some later time. (Note that integrating the expressions analytically is the method of *general* perturbations). Since orbital elements vary very much more slowly than position and velocity, larger integration step-sizes are possible than in *Cowell's* or *Encke's* methods.

For details regarding the formulation of this method, refer to [1].

2.4 Perturbations

This section goes over the various perturbations that act upon a satellite in orbit, emphasising those important for Earth-orbiting spacecraft [1, 30]. The basic orbit will be considered to be a Keplerian conic, meaning the orbit that results through the gravitational attraction of a spherical, perfectly homogeneous, central body. The perturbations mentioned are the following

- Earth's Gravitational Potential.
- Third-body perturbations (Sun and Moon).
- Atmospheric drag.
- Solar Radiation Pressure (SRP).

2.4.1 Earth's Gravitational Potential

The perturbing acceleration due to a non-spherical central body can be described by an aspherical potential function [32]. The Earth's gravity potential can be expanded in spherical harmonics and written as

$$U = -\frac{\mu}{r}\sum_{n=0}^{\infty}\sum_{m=0}^{n}\frac{R_e^n}{r^n}P_{nm}(\sin\phi + S_{nm}\sin(m\lambda))(C_{nm}\cos(m\lambda)) \tag{2.19}$$

where r is the radial distance from the center of mass of the Earth, ϕ is the geocentric latitude, λ the geographic longitude, μ is the Earth's gravitational parameter, R_e the Earth's mean equatorial radius, P_{nm} is the associated Legendre function of order m and degree n, and S_{nm} and C_{nm} are the geopotential constant coefficients. This potential includes the spherical central gravity term. Generally, the Earth's gravity field is assumed to be constant and the geopotential coefficients S_{nm} and C_{nm} are provided by a gravity model.

Instead of writing the potential function in spherical coordinates (r, ϕ, λ) one may express it using Keplerian orbital elements, replacing h and θ by a and M, respectively (a, e, i, Ω, ω, M) [34]. This results in

$$U = -\frac{\mu}{r} - \sum_{n=2}^{\infty}\sum_{m=0}^{n}\sum_{p=0}^{n}\sum_{q=-\infty}^{\infty}\frac{\mu}{a}\frac{R_e^n}{a^n}F_{nmp}(i)G_{npq}(e)S_{nmpq}(\omega, M, \Omega, \Theta) \tag{2.20}$$

with

$$S_{nmpq}(\omega, M, \Omega, \Theta) = \begin{cases} C_{nm}\cos(\Psi_{nmpq}) + S_{nm}\sin(\Psi_{nmpq}) & \text{if } (n-m) \text{ even} \\ -S_{nm}\cos(\Psi_{nmpq}) + C_{nm}\sin(\Psi_{nmpq}) & \text{if } (n-m) \text{ odd} \end{cases} \tag{2.21}$$

and

$$\Psi_{nmpq} = (n-2p)\omega + (n-2p+q)M + m(\Omega - \Theta) \tag{2.22}$$

where Θ is the Greenwich mean sidereal time and $F_{nmp}(i)$ and G_{npq} are the so-called inclination and eccentricity functions, respectively [34]. Although the potential function using Keplerian orbital elements (Equation (2.20)) is more complex, it provides more insight into the dynamics compared to using spherical coordinates (Equation (2.19)).

The zonal terms ($m = 0$) cause the strongest perturbing effects. In addition, when the orbital period is commensurable with the Earth's rotation period, then nmpq can become nearly constant and tesseral ($n \neq m \neq 0$) and sectorial terms ($n = m \neq 0$) can cause significant long-periodic effects.

2.4.1.1 Zonal perturbations

The perturbing potential of an axially symmetric gravitational field is of the form [35]

$$R = \frac{\mu}{r}\sum_{n=2}^{\infty} J_n \left(\frac{R_e}{r}\right)^n P_n(\sin\phi) \tag{2.23}$$

where J_n is the n-th zonal harmonic, $P_n(\sin\phi)$ is the Legendre polynomial of degree n in $\sin\phi$. The full potential is then given by

$$U = -\frac{\mu}{r}\left\{1 - \sum_{n=2}^{\infty} J_n \left(\frac{R_e}{r}\right)^n P_n(\sin\phi)\right\} = -\frac{\mu}{r} + R \tag{2.24}$$

The acceleration due to the gravitational potential is obtained by taking the gradient of the potential function, as follows

$$f = -\nabla U \tag{2.25}$$

When the second zonal harmonic is solely considered, i.e. the J_2 term, the Legendre polynomial is $P_2(\sin\phi) = \left(3\sin^2\phi - 1\right)/2$, resulting in the following perturbing potential

$$R_{J_2} = \frac{1}{2}\mu J_2 \frac{R_e^2}{r^3}\left(3\sin^2\phi - 1\right) \tag{2.26}$$

The perturbing accelerations due to J_2 are computed by taking the gradient: $f_{J_2} = -\nabla R_{J_2}$.

The J_2 perturbation is the strongest perturbing force up to geosynchronous altitude and mainly causes secular variation of the longitude of the ascending node, the argument of perigee and mean anomaly. However, because the J_2 perturbation is so strong, coupling effects with other perturbations should also be taken into account.

2.4.1.2 Tesseral resonance

When the orbital period of an object is commensurable with the rotational period of the Earth then the gravitational effect due to tesseral terms can build up over time [36]. This is, for example, the case for geosynchronous and GPS orbits that have 24- hour and 12-hour orbital periods, respectively. Since it does not affect LEO spacecraft directly, not much detail will go into this particular perturbation. For further details into the subject, refer to [32].

One of the major effects of tesseral resonance are long-term oscillations of the semi-major axis, which cause longitudinal drift. If one of the resonance conditions mentioned in [32] applies, tesseral harmonics cannot be neglected.

2.4.2 Third-body perturbations (Sun and Moon)

Satellite orbits are naturally perturbed by the gravity of other bodies in the solar system, of which the two main perturbing bodies are the Sun and Moon. The so-called third-body perturbation is computed as

$$\mathbf{p}_* = \mu_* \left(\frac{\mathbf{r}_* - \mathbf{r}}{\left|\mathbf{r}_* - \mathbf{r}\right|^3} - \frac{\mathbf{r}_*}{\left|\mathbf{r}_*\right|^3} \right) \tag{2.27}$$

where μ_* is the gravitational parameter of the third body and r_* denotes the position vector of the third body with respect to the center of the Earth.

Third-body perturbations become more significant when the satellite's orbital altitude and eccentricity are higher [37]. The effect of the perturbation depends on the orientation of the orbital plane with respect to the orbital plane of perturbing body. If the alignment is near constant for a period of time, such as for Sun-synchronous orbits, then resonance can occur, resulting in secular change of orbital parameters.

2.4.2.1 Lunisolar resonance

The strongest lunisolar resonances occur when commensurability takes place between the arguments of perigee and longitudes of the node of the satellite and the third body [38]

$$\alpha\dot{\omega} + \beta\dot{\Omega} + k\dot{\Omega}_* \approx 0 \tag{2.28}$$

where α, β and k are integer values and Ω_* is the right ascension of the ascending node of the third body defined with respect to the ecliptic plane. Two important resonances occur when $\dot{\omega} = 0$ and **2** $2\dot{\omega} + \dot{\Omega} = 0$. If we assume that ω and Ω change in a secular fashion due to J_2 only, then

- $\dot{\omega} = 0$ when $i = 63.4°$ or $i = 116.6°$.
- $2\dot{\omega} + \dot{\Omega} = 0$ when $i = 56.1°$ or $i = 111°$.

This means that at these inclinations the effect of lunisolar perturbations can be very strong. For example, these resonances can cause large variations in the eccentricity of an orbit [39, 40]. In addition, it must be mentioned that the overlap of lunisolar resonances can result in chaotic behaviour of the orbital evolution [41, 42]. Due to chaos, the trajectory depends strongly on the initial state which makes accurate orbit prediction more difficult.

2.4.3 Atmospheric drag

For Earth, the commonly accepted altitude at which space "begins" is 100 km (the Kármán line). Although over 99.9999% of the Earth's atmosphere lies below 100 km, the air density at that altitude is nevertheless sufficient to exert drag and cause aerodynamic heating on objects moving at orbital speeds. The drag will lower the speed and the height of a spacecraft, and the heating can produce temperatures of 2000 °C or more.

There are a number of models that describe the variation of atmospheric properties with altitude [43] and one of them is the US Standard Atmosphere 1976, USSA76 [44]. Figure 2.6 shows the US Standard Atmosphere density profile from the sea level to an altitude of 1000 km.

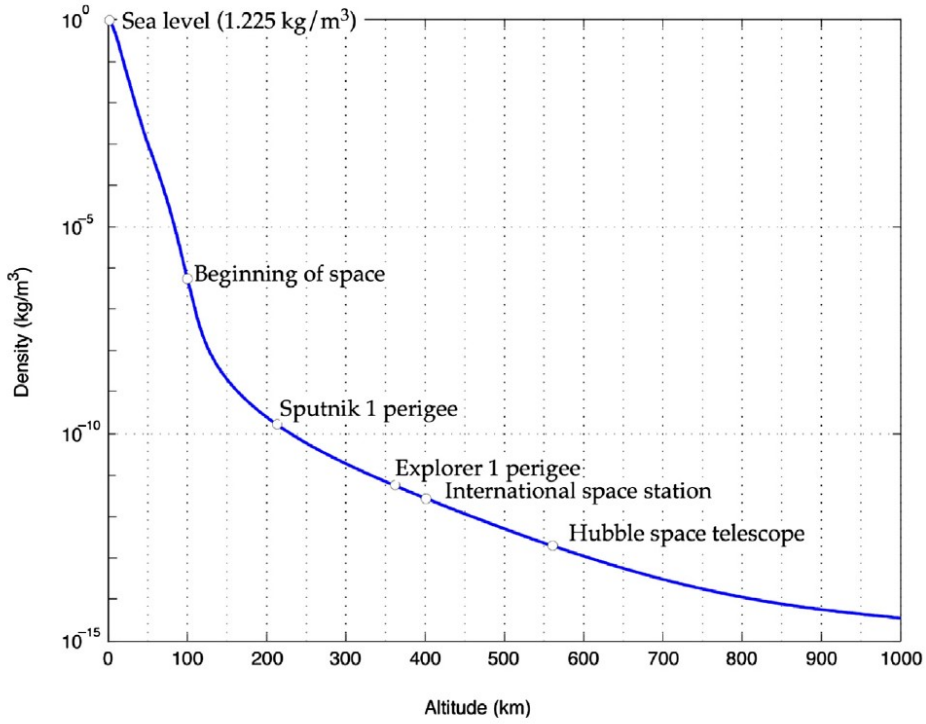


Figure 2.6: US Standard Atmosphere 1976: density versus altitude [1].

According to USSA76, the atmosphere is a spherically symmetric 1000 km thick gaseous shell surrounding the Earth. Its properties throughout are steady state and are consistent with a period of moderate solar activity. The hypothetical variation of the properties with altitude approximately represents the year-round conditions at mid latitudes averaged over many years. The model provides realistic values of atmospheric density which, however, may not match the actual values at a given place or time.

As such, the perturbing acceleration due to atmospheric drag, $\mathbf{p}_{drag}$, depends on the space-craft's drag coefficient C_d, area-to-mass ratio A/m, the relative velocity with respect to the atmosphere $\mathbf{v}_{rel}$, and the atmospheric density ρ. The perturbing acceleration $\mathbf{p}_{drag}$ due to atmospheric drag is then given by

$$\mathbf{p}_{drag} = -\frac{1}{2}\rho v_{rel}\left(\frac{C_{\mathrm{D}}A}{m}\right)\mathbf{v}_{rel} \tag{2.29}$$

Atmospheric drag mainly affects the semi-major axis and eccentricity by decreasing the apogee altitude of the orbit due to energy dissipation.

2.4.4 Solar Radiation Pressure

Photons from the Sun that are reflected or absorbed by an object exchange momentum and cause a perturbing force called Solar Radiation Pressure (SRP). The SRP perturbation depends on the solar pressure $P_S = W/c = 4.56 \times 10^{-6}\text{N/m}^2$ (with W the energy flux density of the Sun at 1 AU and c the speed of light), the reflectivity coefficient C_R which measures the momentum exchange between incoming radiation and the spacecraft, the area-to-mass ratio of the spacecraft A/m and the Sun's position. The perturbing acceleration $\mathbf{p}_{\text{SRP}}$ due to SRP is

$$\mathbf{p}_{\text{SRP}} = -\nu C_R P_S \frac{A}{m} \mathbf{r}_S \tag{2.30}$$

where ν is the *shadow function*, which has the value 0 if the satellite is in the earth's shadow; otherwise, $\nu = 1.$ and $\mathbf{r}_S$ denotes the position vector of the Sun with respect to the satellite (which can be approximated by the position vector of the Sun with respect to Earth). For most satellites, the effect of SRP on the orbit is generally small, but for objects with high area-to-mass ratios, SRP can have large effects [45]. In addition, if the orbital plane is aligned with the direction of the Sun, the SRP effects can build-up and change the orbit significantly.

Besides the perturbations discussed previously, other small perturbations exist. To name a few, solid and ocean tides, relativistic effects, Earth radiation, Earth's albedo, [39] generally have a very small effect and are commonly disregarded.

2.5 The Kalman Filter

This section follows the content shared in [46].

Within the significant toolbox of mathematical tools that can be used for stochastic estimation from noisy sensor measurements, one of the most well-known and often-used tools is what is known as the *Kalman filter*. The Kalman filter is named after Rudolph E. Kalman, who in 1960 published his famous paper describing a recursive solution to the discrete-data linear filtering problem [47].

The Kalman filter is essentially a set of mathematical equations that implement a predictor-corrector type estimator that is *optimal* in the sense that it minimizes the estimated *error* covariance—when some presumed conditions are met. Since the time of its introduction, the *Kalman filter* has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation. This is likely due in large part to advances in digital computing that made the use of the filter practical, but also to the relative simplicity and robust nature of the filter itself. Rarely do the conditions necessary for optimality actually exist, and yet the filter apparently works well for many applications in spite of this situation.

In recent years, various research papers in the field of orbit estimation with the help of Kalman filtering have been submitted. In [48], an orbit determination algorithm was designed and simplified with the purpose of being used for LEO navigation. For comparison, a least square algorithm and an EKF were developed. The study concluded that the EKF algorithm converged faster than least square's and satisfied the criterion of low computation burden which is required for autonomous orbit determination.

In [49], a method to predict satellite orbits in a GPS device without a network connection is presented. Testing of the algorithm showed that in 95% of the cases the error in satellite's prediction position remained under 21 m for one day and under 94 m for three days. Noting that in three days the satellite has travelled approximately 1 million km, this level of accuracy is very high.

In [50], the authors studied the two-week prediction accuracy improvement obtained when adding some smaller forces to their previously developed algorithm. The new model was tested for GPS, GLONASS and Beidou satellites using initial conditions computed from precise ephemerides. It was found that the enhancements gave a small but not negligible improvement. However, the improvements came at the cost of noticeable increase in computational load.

2.5.1 The Discrete Kalman Filter

This section describes the filter in its original formulation [47] where the measurements occur and the state is estimated at discrete points in time.

2.5.1.1 The Process to be Estimated

The Kalman filter addresses the general problem of trying to estimate the state $x \in \Re^n$ of a discrete-time controlled process that is governed by the linear stochastic differential equation

$$\dot{x} = Ax + Bu + w \tag{2.31}$$

that in a discrete form is expressed by

$$x_k = A_d x_{k-1} + B_d u_k + w_{k-1} \tag{2.32}$$

with a measurement $z \in \Re^m$ that is

$$z_k = Hx_k + v_k \tag{2.33}$$

The random variables w_k and v_k represent the process and measurement noise, respectively. They are assumed to be independent (of each other), white, and with normal probability distributions

$$\begin{aligned} p(w) &\sim N(0, Q) \\ p(v) &\sim N(0, R) \end{aligned} \tag{2.34}$$

In practice, the *process noise covariance* Q and *measurement noise covariance* R matrices might change with each time step or measurement, however here they are assumed to be constant.

The $n \times n$ matrix A in Equation (2.32) relates the state at the previous time step $k-1$ to the state at the current step k, in the absence of either a driving function or process noise. Note that in practice A might change with each time step, but it is assumed to be constant here. The $n \times l$ matrix B relates the optional control input $u \in \Re^l$ to the state x. The $m \times n$ matrix H in the measurement equation Equation (2.33) relates the state to the measurement z_k. In practice H might change with each time step or measurement, but here it is also assumed to be constant.

2.5.1.2 The Computational Origins of the Filter

$\hat{x}_k^- \in \Re^n$ (note the "super minus") is defined as the *a priori* state estimate at step k, given knowledge of the process prior to step k, and $\hat{x}_k \in \Re^n$ to be the *a posteriori* state estimate at step k given the measurement z_k. The *a priori* and *a posteriori* estimate errors can then be defined as

$$\begin{aligned} e_k^- &\equiv x_k - \hat{x}_k^- \\ e_k &\equiv x_k - \hat{x}_k \end{aligned} \tag{2.35}$$

To derive the equations for the Kalman filter, the first goal begins with finding an equation that computes an *a posteriori* state estimate $\hat{x}_k$ as a linear combination of an *a priori* estimate $\hat{x}_k^-$ and a weighted difference between an actual measurement z_k and a measurement prediction $H\hat{x}_k^-$ as such

$$\hat{x}_k = \hat{x}_k^- + K\left(z_k - H\hat{x}_k^-\right) \tag{2.36}$$

The difference $z_k - H\hat{x}_k^-$ is called the measurement *innovation*, or the *residual*. The residual reflects the discrepancy between the predicted measurement $H\hat{x}_k^-$ and the actual measurement z_k. A residual of zero means that the two are in complete agreement.

The $n \times m$ matrix K in Equation (2.36) is known as the *gain* or *blending factor* that minimizes the a *posteriori* error P_k [51, 52]. One form of the resulting K is given by

$$K_k = P_k^- H^T \left(HP_k^- H^T + R\right)^{-1} \tag{2.37}$$

$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R} \tag{2.38}$$

where Eq. (2.37) refers to cases where R is a matrix and Eq. (2.38) is applied when R is a scalar. The latter occurs in cases where the output is a scalar instead of a vector.

Looking at Equation (2.37), it is visible that as the measurement error covariance R approaches zero, the gain K weights the residual more heavily. On the other hand, as the *a priori* estimate error covariance P_k^- approaches zero, the gain K weights the residual less heavily.

An alternative way of thinking about the weighting by K is that as the measurement error covariance R approaches zero, the actual measurement z_k is "trusted" more and more, while the predicted measurement $H\hat{x}_k^-$ is trusted less and less. On the other hand, as the *a priori* estimate error covariance P_k^- approaches zero, the actual measurement z_k is trusted less and less, while the predicted measurement $H\hat{x}_k^-$ is trusted more and more.

2.5.1.3 The Discrete Kalman Filter Algorithm

This segment begins with a broad overview, covering the "high-level" operation of one form of the discrete Kalman filter. After presenting this high-level view, the focus will turn to the specific equations and their use in this version of the filter.

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) mea-

surements. As such, the equations for the Kalman filter fall into two groups: *time update* equations and *measurement update* equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the *a priori* estimates for the next time step. The measurement update equations are responsible for the feedback—i.e. for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate.

The time update equations can also be thought of as *predictor* equations, while the measurement update equations can be thought of as *corrector* equations. Indeed the final estimation algorithm resembles that of a *predictor-corrector* algorithm for solving numerical problems. The specific equations for the time update are

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{2.39}$$
$$P_k^- = AP_{k-1}A^T + Q \tag{2.40}$$

while the equations for the measurement update are

$$K_k = P_k^- H^T \left(HP_k^- H^T + R\right)^{-1} \tag{2.41}$$
$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - H\hat{x}_k^-\right) \tag{2.42}$$
$$P_k = \left(I - K_k H\right) P_k^- \tag{2.43}$$

The first task during the measurement update is to compute the Kalman gain, K_k. The next step is to actually measure the process to obtain z_k, and then to generate an *a posteriori* state estimate by incorporating the measurement as in Equation (2.42). Equations (2.41) and (2.42) are simply Equations (2.37) and (2.36), respectively, repeated here for completeness. The final step is to obtain an *a posteriori* error covariance estimate via Equation (2.43).

After each time and measurement update pair, the process is repeated with the previous *a posteriori* estimates used to project or predict the new *a priori* estimates. This recursive nature is one of the very appealing features of the Kalman filter—it makes practical implementations much more feasible than (for example) an implementation of a Wiener filter [52] which is designed to operate on *all* of the data *directly* for each estimate. Instead, the Kalman filter recursively conditions the current estimate on all of the past measurements. Below, Figure 2.7 offers a complete picture of the operation of the filter.

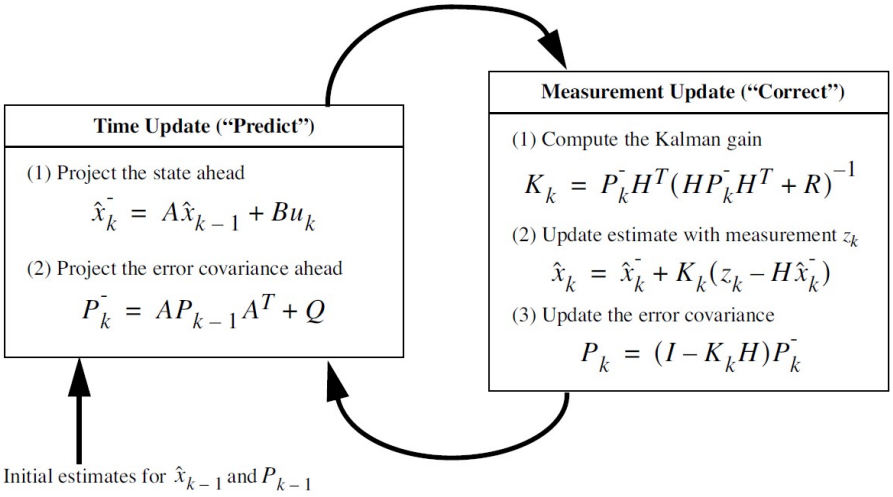


Figure 2.7: A complete picture of the operation of the Kalman filter.

2.5.2 The Extended Kalman Filter (EKF)

2.5.2.1 The Process to be Estimated

As described above, the Kalman filter addresses the general problem of trying to estimate the state $x \in \Re^n$ of a discrete-time controlled process that is governed by a *linear* stochastic differential equation. But what happens if the process to be estimated and (or) the measurement relationship to the process is non-linear? Some of the most interesting and successful applications of Kalman filtering have been such situations. A Kalman filter that linearizes about the current mean and covariance is referred to as an *Extended Kalman Filter* or EKF.

In something akin to a Taylor series, the estimation around the current estimate can be linearized using the partial derivatives of the process and measurement functions to compute estimates even in the face of non-linear relationships. To do so, it is assumed that the process has a state vector $x \in \Re^n$, but that the process is now governed by the *non-linear* stochastic differential equation

$$x_k = f\left(x_{k-1}, u_k, w_{k-1}\right) \tag{2.44}$$

with a measurement $z \in \Re^m$ that is

$$z_k = h\left(x_k, v_k\right) \tag{2.45}$$

where the random variables w_k and v_k again represent the process and measurement noise as

in Equations (2.34). In this case the *non-linear function* in the differential equation Equation (2.44) relates the state at the previous time step $k-1$ to the state at the current time step k. It includes as parameters any driving function u_k and the zero-mean process noise w_k. The *non-linear* function h in the measurement equation Equation (2.45) relates the state x_k to the measurement z_k.

In practice, one does not know the individual values of the noise w_k and v_k at each time step. However, one can approximate the state and measurement vector without them as

$$\tilde{x}_k = f\left(\hat{x}_{k-1}, u_k, 0\right) \tag{2.46}$$

$$\tilde{z}_k = h\left(\tilde{x}_k, 0\right) \tag{2.47}$$

where $\hat{x}_k$ is some *a posteriori* estimate of the state (from a previous time step k).

It is important to note that a fundamental flaw of the EKF is that the distributions (or densities in the continuous case) of the various random variables are no longer normal after undergoing their respective non-linear transformations. The EKF is simply a state estimator that only approximates the optimality of Bayes' rule by linearization.

2.5.2.2 The Computational Origins of the Filter

To estimate a process with non-linear difference and measurement relationships, the process begins by writing new governing equations that linearize an estimate about Equation (2.46) and (2.47)

$$x_k \approx \hat{x}_k + A\left(x_{k-1} - \hat{x}_{k-1}\right) + W w_{k-1} \tag{2.48}$$

$$z_k \approx \tilde{z}_k + H\left(x_k - \tilde{x}_k\right) + V v_k \tag{2.49}$$

where

- x_k and z_k are the actual state and measurement vectors.
- $\tilde{x}_k$ and $\tilde{z}_k$ are the approximate state and measurement vectors from Equations (2.46) and (2.47).
- $\hat{x}_k$ is an *a posteriori* estimate of the state at step k.
- the random variables w_k and v_k represent the process and measurement noise.

- A is the Jacobian matrix of partial derivatives of f with respect to x, that is

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}} (\hat{x}_{k-1}, u_k, 0)$$

- W is the Jacobian matrix of partial derivatives of f with respect to w, that is

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}} (\hat{x}_{k-1}, u_k, 0)$$

- H is the Jacobian matrix of partial derivatives of h with respect to x, that is

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}} (\tilde{x}_k, 0)$$

- V is the Jacobian matrix of partial derivatives of h with respect to v, that is

$$V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}} (\tilde{x}_k, 0)$$

For simplicity, the time step subscript k is not used in the notation with the Jacobians A, W, H, and V, even though they are in fact different at each time step.

A new notation for the prediction error and the measurement residual is now respectively defined,

$$\tilde{e}_{x_k} \equiv x_k - \tilde{x}_k \tag{2.50}$$

$$\tilde{e}_{z_k} \equiv z_k - \tilde{z}_k \tag{2.51}$$

In practice, one does not have access to x_k in Equation (2.50), it is the *actual* state vector, i.e. the quantity one is trying to estimate. On the other hand, one does have access to z_k in Equation (2.51), as it is the actual measurement that one is using to estimate x_k. Using Equations (2.50) and (2.51), the governing equations for an *error process* can be written as

$$\tilde{e}_{x_k} \approx A (x_{k-1} - \hat{x}_{k-1}) + \varepsilon_k \tag{2.52}$$

$$\tilde{e}_{z_k} \approx H \tilde{e}_{x_k} + \eta_k \tag{2.53}$$

where ε_k and η_k represent new independent random variables having zero mean and covariance matrices WQW^T and VRV^T, with Q and R as in (2.34) respectively.

A particularity of Equations (2.52) and (2.53) is that they are linear and closely resemble the

difference and measurement Equations (2.32) and (2.33) from the discrete Kalman filter. This makes it viable to use the actual measurement residual $\tilde{e}_{z_k}$ in Equation (2.51) and a second (hypothetical) Kalman filter to estimate the prediction error $\tilde{e}_{x_k}$ given by Equation (2.52). This estimate, called $\hat{e}_k$, could then be used along with Equation (2.50) to obtain the *a posteriori* state estimates for the original non-linear process as

$$\hat{x}_k = \tilde{x}_k + \hat{e}_k \tag{2.54}$$

Letting the predicted value of $\hat{e}_k$ be zero, the Kalman filter equation used to estimate $\hat{e}_k$ is

$$\hat{e}_k = K_k \tilde{e}_{z_k} \tag{2.55}$$

The second (hypothetical) Kalman filter can be disregarded by substituting Equation (2.55) back into Equation (2.54) and making use of Equation (2.51), as such

$$\begin{aligned} \hat{x}_k &= \tilde{x}_k + K_k \tilde{e}_{z_k} \\ &= \tilde{x}_k + K_k \left(z_k - \tilde{z}_k \right) \end{aligned} \tag{2.56}$$

Equation (2.56) can now be used for the measurement update in the EKF, with $\tilde{x}_k$ and $\tilde{z}_k$ coming from Equations (2.46) and (2.47), and the Kalman gain coming from Equation (2.41) with the appropriate substitution for the measurement error covariance.

The complete set of EKF equations is shown below. To note that $\tilde{x}_k$ has been replaced for $\hat{x}_k^-$ to remain consistent with the earlier "super minus" *a priori* notation, and that the subscript k is now attached to the Jacobians A, W, H, and V, to reinforce the notion that they are different at (and therefore must be recomputed at) each time step. The specific equations for the time update are

$$\hat{x}_k^- = f\left(\hat{x}_{k-1}, u_k, 0\right) \tag{2.57}$$

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \tag{2.58}$$

while the equations for the measurement update are

$$K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + V_k R_k V_k^T \right)^{-1} \tag{2.59}$$

$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - h\left(\hat{x}_k^-, 0 \right) \right) \tag{2.60}$$

$$P_k = \left(I - K_k H_k \right) P_k^- \tag{2.61}$$

The basic operation of the EKF is the same as the linear discrete Kalman filter. Additionally, Figure 2.8 below offers a complete picture of the operation of the EKF.

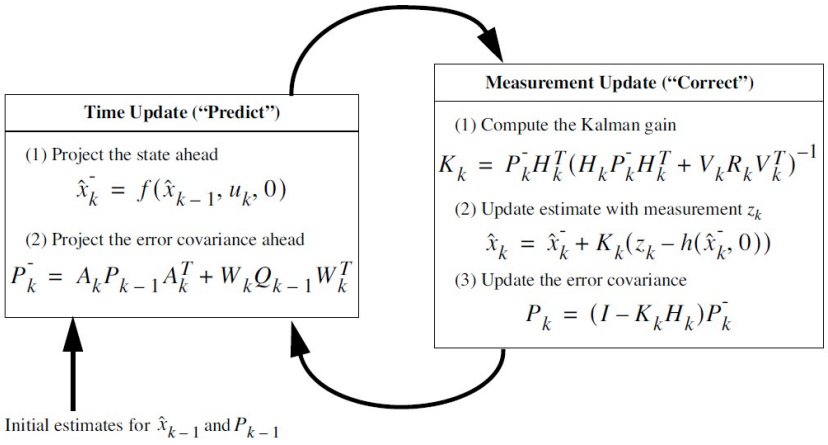


Figure 2.8: A complete picture of the operation of the EKF.

An important feature of the EKF is that the Jacobian H_k in the equation for the Kalman gain K_k serves to correctly propagate or "magnify" only the relevant component of the measurement information. For example, if there is not a one-to-one mapping between the measurement z_k and the state via h, the Jacobian H_k affects the Kalman gain so that it only magnifies the portion of the residual $z_k - h\left(\hat{x}_k^-, 0 \right)$ that does affect the state. Of course if over all measurements there is not a one-to-one mapping between the measurement z_k and the state via h, then as you might expect the filter will quickly diverge. In this case the process is unobservable.

2.6 Contributions from Artificial Intelligence

This section follows the contents shared in [53–56].

Artificial intelligence is a branch of computer science that deals with artificially creating intelligence to a system. The father of artificial intelligence, John McCarthy, says that "Artificial Intelligence is the science and engineering of making intelligent machines especially intelligent computer programs." Artificial intelligence systems are created by studying the process of thinking, learning, observing, and decision-making by the human brain. The main objectives of creating Artificial Intelligence (AI) systems are to create an expert system

and to implement human intelligence in machines. Humanoids and robots are examples of physical devices that are upgraded with artificial intelligence.

Machine learning is the most promising and relevant domain to apply artificial intelligence in systems. It is the most common technique to process big data and a method to learn from data. Machine learning algorithms are designed in such a way that they are self-adaptive and are able to get new patterns to itself through experience. It combines computer science, mathematics, and statistics. Computer science is needed for implementing the algorithms, mathematics for developing machine learning models, and statistics for generating inferences from the data.

Deep learning is a subset of machine learning. It has an artificial neural network to carry out the tasks of machine learning, enabling the system to process data in a non-linear fashion. Deep learning can be defined as a class of machine learning algorithms which are capable of extracting more features from raw input data using multiple (and hidden) layers. To implement deep learning techniques, many computational nodes will be created and each node is trained to analyze the given information and make decisions like human brains. It is exactly similar to how the human brain filters any information into deep layers to understand in its own way.

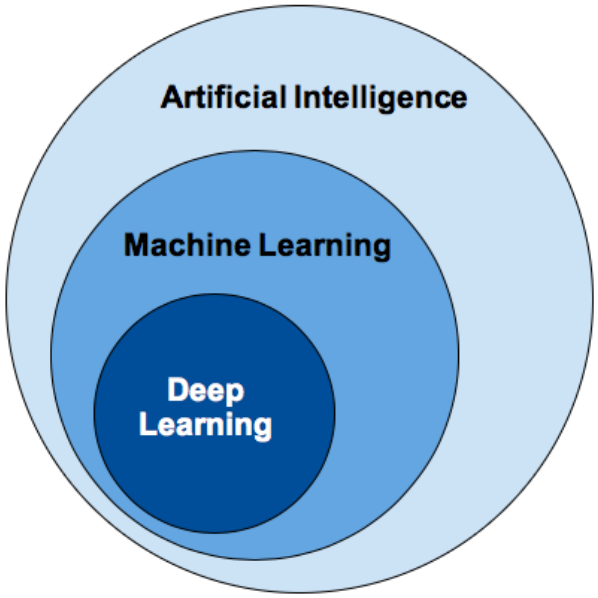


Figure 2.9: Visual representation of the Artificial Intelligence, Machine Learning and Deep learning domains [2].

2.6.1 Machine learning Pipeline

For a machine learning project to be successful, it must follow a pipeline or workflow to make it as simple as possible. This dissertation follows a workflow that can be divided in seven major steps, namely

1. Data collection.
2. Data preparation.
3. Choosing a model.
4. Model training.
5. Model evaluation.
6. Hyperparameter tuning.
7. Model Deployment.

2.6.1.1 Data collection

To determine what data shall be used to train a ML model, the following key points are to be taken into consideration

- The kind of problem.
- Data sources already existent.
- Privacy concerns.
- Whether the data is public.

Lastly, the type of data (structured or unstructured) needs to be chosen. Structured data is noticeable for having a tabulated format (rows and columns style) and containing data from categorical to time series. Unstructured data refers to data that has no rigid structure, such as images, video files, audio files or text files.

2.6.1.2 Data preparation

Most data is not ready to be used as training data in a ML application. As such, it requires preparation. This preparation includes steps such as Exploratory Data Analysis (EDA), data splitting and data preprocessing.

Exploratory Data Analysis (EDA) as the name suggests, EDA is a step that allows one to learn more about the data at hand. It helps answer questions such as

- What are the feature variables (input) and the target variables (output)?
- Where are the outliers? How many of them are there? Why are they there?
- Are there missing values?

Data Splitting in this step, data is split into (in most cases) three sets

- Training set (around 70-80% of the data) from which the model learns.
- Validation set (around 10-15% of the data) from which the model's hyperparameters are tuned.
- Test set (around 10-15% of the data) from which the model's final performance is evaluated on.

Data Preprocessing relates to the final preparation of the data before training. It is comprised of Feature imputation, Feature encoding, Feature scaling and standardization, Feature engineering and Feature selection. The methods relevant for this dissertation are

Feature imputation this method allows missing values to be filled. It is achieved by single imputation, multiple imputation or k-NN (k-nearest neighbours).

Feature scaling (normalization) this method, along with the following one, are useful when numerical variables are on different scales, as some ML algorithms do not perform well under these circumstances. Feature scaling shifts every value to be between zero and one by applying the following equation

$$x_{norm} = \frac{x - min(x)}{max(x) - min(x)} \tag{2.62}$$

Feature standardization this method standardizes every value to have zero-mean and unit-variance by applying the following equation

$$x_{stand} = \frac{x - \mu(x)}{\sigma(x)} \tag{2.63}$$

where μ represents the mean of x and σ the standard deviation of x. Also, values' range is uncapped, making this method more robust to outliers than feature scaling.

2.6.1.3 Choosing a model

A ML model is most commonly approached from one of two ways: supervised learning or unsupervised learning. Unsupervised learning is used to draw inferences and find patterns from input data that has no references to labeled outcomes, or unlabeled data. Two common methods used are clustering and dimensionality reduction.

Alternatively, the goal of supervised learning is to find specific relationships or structure in the input data that allow the model to effectively produce correct output data given the training data provided. In other words, it involves learning a function that maps an input to an output based on *input-output* example pairs. Supervised learning is divided into two categories, regression and classification. Regression models map input to a continuous output whereas classification models map input to output labels. Examples of supervised learning algorithms are

Linear Regression regression model in which the idea is to simply find a line that best fits the data. Extensions of linear regression include multiple linear regression and polynomial regression.

Logistic Regression classification model similar to linear regression but used to model the probability of a finite number of outcomes, typically two. In essence, a logistic equation is created in such a way that the output values can only be between zero and one.

Naive Bayes classification model driven by Bayes' Theorem, where the goal is to find the class y with the maximum proportional probability.

Support Vector Machines (SVMs) can be used both for regression and classification. Assuming that there are two classes of data, a support vector machine will find a hyperplane or a boundary between these two classes that maximizes the margin between them. There are many planes that can separate the two classes, but only one plane can maximize the margin or distance between the classes.

Decision Trees and Random Forests can be used both for regression and classification. Decision trees split data based on criteria, creating nodes.The last nodes, where a decision is made, are called the leaves of the tree. Decision trees are intuitive and easy to build but fall short when it comes to accuracy. Random forests are an ensemble learning technique that builds off of decision trees. Multiple decision trees are created from which the output is the mean of all of the predictions (regression) or the class selected by most trees (classification).

Neural Networks A Neural Network is essentially a network of mathematical equations connected by nodes. It takes one or more input variables, and by going through a network of equations, results in one or more output variables. It is composed of an input layer, one or more *hidden* layers and an output layer. Each node in the hidden layers represents both a linear function and an activation function that the nodes in the previous layer go through, ultimately leading to an output in the output layer. Examples include CNNs, (typically used for computer vision and image processing), RNNs, (used for sequence modeling) and Transformer networks (used for vision and text).

One other aspect of ML algorithms are learning techniques. These include

Batch learning also called offline learning, models are trained with accumulated data from time to time in a batch manner. In other words, the system is incapable of learning incrementally from a stream of data.

Online learning the training happens in an incremental manner by continuously feeding data as it arrives or in small group. Each learning step is fast and cheap, making the system able to learn about new data on the fly.

Transfer learning focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. It is helpful when data is scarce or com-

puting resources are limited.

Active learning also known as ”human in the loop” learning in which a learning algorithm can interactively query a user (or some other information source) to label new data points with the desired outputs.

Ensemble learning ensemble methods use multiple learning algorithms to obtain better predictive performance than could otherwise be obtained from any of the constituent learning algorithms alone.

2.6.1.4 Model Training

At the heart of the ML workflow is the model training. This is where the bulk of the “learning” is done. The slice of data allocated for training is used to teach the model the intrinsic relations between the features in the data set. Training requires patience and experimentation, as it can take several attempts to achieve the desired accuracy. During those attempts, the model may face *overfitting* or even *underitting*.

Underfitting refers to an algorithm that can neither model the training data nor generalize to new data. An underfit ML algorithm is not a suitable model, having poor performance on the training data [57].

Overfitting refers to an algorithm that models the training data *too* well. It happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance on new data. This means that noise or random fluctuations in the training data is picked up and learned as concepts by the model. These concepts do not apply to new data and negatively impact the models' ability to generalize. Regularization is often used to reduce overfitting [57].

Regularization techniques include

- L1 (lasso) and L2 (ridge) regularization.
- Dropout.
- Early stopping.
- Data augmentation.
- Batch normalization.

2.6.1.5 Model Evaluation

Once training is complete, evaluation is used to assess the model's performance. Here, the test set allows the model to be tested against data that has not been used for training. Evaluation metrics vary between regression and classification models. For regression models, the most common metrics used are Mean Squared Error (MSE), Mean Absolute Error (MAE) and R^2. For classification models, typical metrics include accuracy, precision, recall, confu-

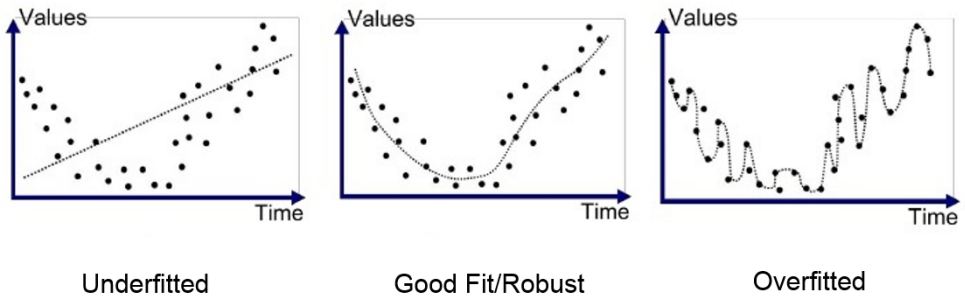


Figure 2.10: Graphical representation of an uderfit, robust and overfit model [3].

sion matrix and the mean average precision. It is also this step that assesses if the model is *overfitting*, *underitting*, or robust.

2.6.1.6 Hyperparameter tuning

Hyperparameter tuning relates to running a series of experiments with different model settings and checking which one performs the best. The adjustment, or tuning, of these hyperparameters, remains somewhat an art, being an experimental process that heavily depends on the specifics of the data set, model, and training process. An often tuned hyperparameter is the learning rate, followed by the number of layers for neural networks, batch size, and the number of iterations.

2.6.1.7 Model Deployment

This final step, also called prediction or inference, is where the model predicts real results. With time, the model's predictions will start to ”age” or ”drift”, mainly due to changes or upgrades related to the data sources. This is when retraining should be considered.

2.6.2 Recurrent Neural Networks (RNNs)

Traditional neural networks are unable to understand long term dependencies, turning out to be a major shortcoming when modeling sequential data. RNNs address this issue by having loops in their network, allowing information to persist.

In Figure 2.11, part of a RNN, A, looks at the input x_t and outputs a value h_t. The loop lets information pass from one step of the network to the next.

Figure 2.11 can make it hard to picture what happens when the loop occurs. A RNN can be thought of multiple copies of the same network, each passing a message to a successor. An easier way to visualize this is by unrolling the loop, represented in Figure 2.12.

The chain-like nature reveals that RNNs are intimately related to sequences and lists, being the natural architecture of choice to be used for such data. There has been incredible success applying RNNs to a variety of problems such as speech recognition, language modeling,

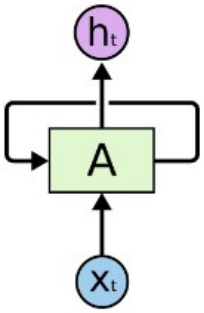


Figure 2.11: A RNN block loop.

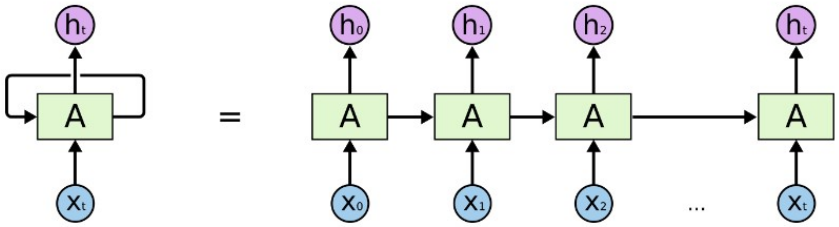


Figure 2.12: An unrolled RNN.

translation and image captioning [58].

A major part of these successes is due to the use of LSTMs, a special kind of RNN that works, for many tasks, far better than the standard version. The LSTM architecture is the one chosen for this dissertation.

2.6.2.1 Long-Term Dependencies

One feature that stands out regarding RNNs is their ability to connect previous information to the present task. In some cases, the need is to look only at recent information to perform the present task, whereas in other cases, there is a need for more context. When the gap between the relevant information and the place that it is needed is small, RNNs can learn to use past information, as shown in Figure 2.13.

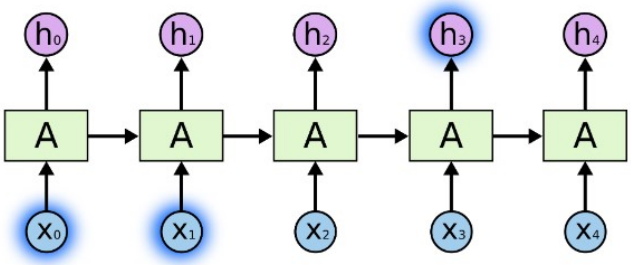


Figure 2.13: RNN connecting the information.

However, when the gap between the relevant information and the point where it is needed becomes very large, RNNs are unable to learn to connect the information.

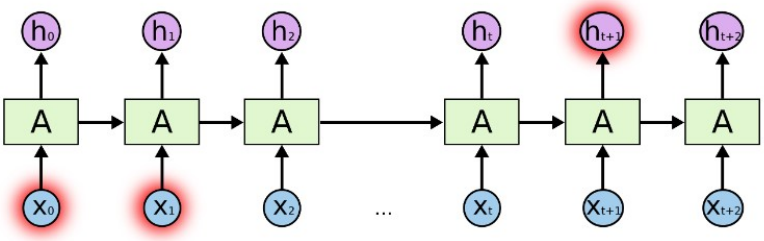


Figure 2.14: RNN unable to connect the information.

In particular, literature states that due to the temporal evolution of the backpropagated error depending exponentially on the size of the weights, the backpropagation dynamics causes the gradients to either (1) *blow up* or (2) *vanish* [59]. Case (1) may lead to oscillating weights, while in case (2), learning to bridge long time lags takes a prohibited amount of time, or does not work at all. It was later found that the exploding gradient concern can be alleviated with a heuristic of clipping the gradients at some maximum value [60]. On the other hand, LSTMs were designed to mitigate the vanishing gradient problem.

2.6.3 Long Short-Term Memory (LSTM) Networks

LSTM networks are designed to learn long-term dependencies. Introduced by Hochreiter and Schmidhuber in 1997 [59], they were later refined and popularized in following work.

All RNNs possess a chain-like set of repeating cell structures, called memory cells. In standard RNNs, these cell structures are composed of a single tanh layer, as shown in Figure 2.15.

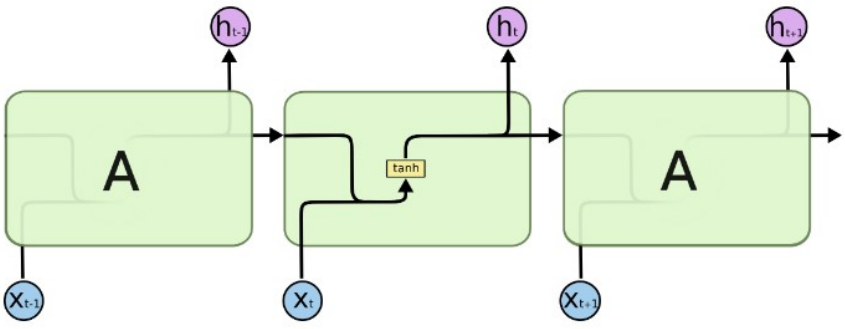


Figure 2.15: Repeating module in a standard RNN with a single tanh layer.

LSTMs are also equipped with this chain-like set of memory cells. However, instead of having a single layer, these memory cells contain four, that interact in a distinctive way. The details of these interactions are discussed further in the section.

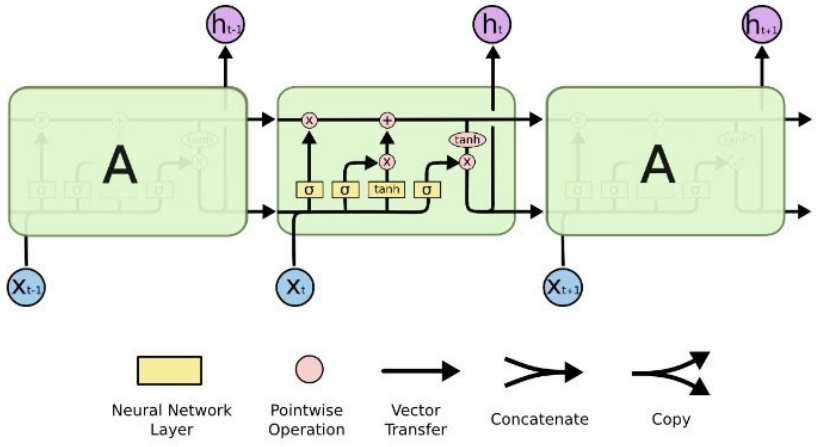


Figure 2.16: Repeating module in a LSTM with four interacting layers.

In Figure 2.16, each line carries an entire vector from the output of one node to the inputs of others. Pink circles represent pointwise operations, such as vector addition or multiplication, while yellow boxes are learned neural network layers. Lines merging represent concatenation, while a line forking denotes its content being copied and the copies going to different locations.

2.6.3.1 LSTM Fundamentals

The prime driver regarding LSTMs is the cell state C, depicted as the horizontal line running through the top of the diagram in Figure 2.17. The cell state can be compared to a conveyor belt that runs across the entire chain, having minor linear interactions along the way. It is very achievable for information to flow along it unchanged.

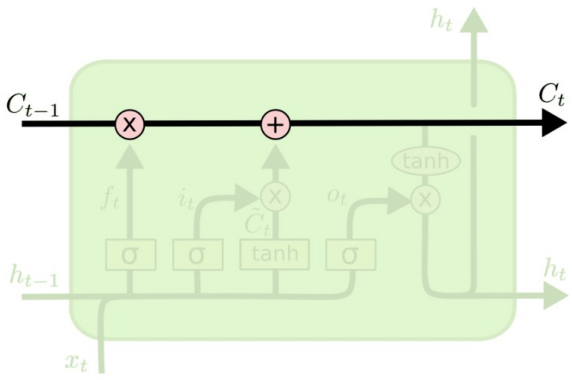


Figure 2.17: The cell state C.

The cell state is carefully regulated by structures called gates. These gates control what information is let through, giving the LSTM the ability to add or remove information to the cell state. Furthermore, they are for the most part composed of a *sigmoid* neural network layer

and a pointwise multiplication operation, as indicated in Figure 2.18.

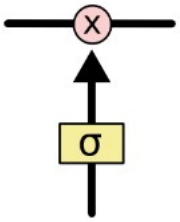


Figure 2.18: The *sigmoid* layer and pointwise multiplication operation.

The *sigmoid* layer outputs numbers between zero and one, describing how much of each component should be let through. If the value is zero, no information is passed. On the other hand, if the value is one, all information is passed. Every LSTM memory cell has three of these gates, to protect and control the cell state C. Figure 2.19 represents the *sigmoid* function in which the *sigmoid* layer is based on.

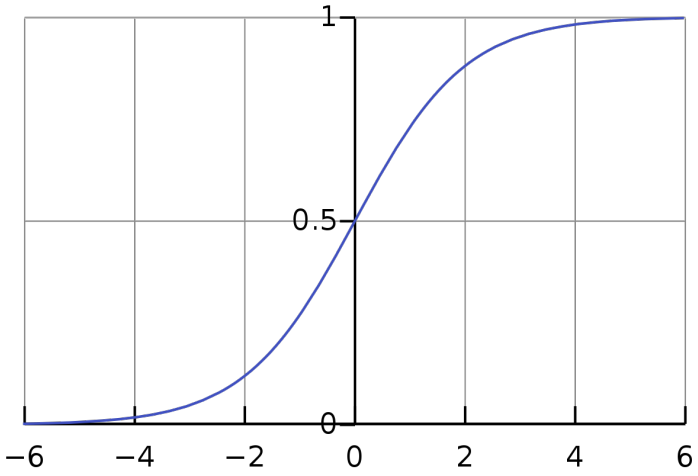


Figure 2.19: The *sigmoid* function.

2.6.3.2 LSTM Workflow

The first step is to decide what information is to be discarded from the cell state C at $t-1$. This decision is made by one of the *sigmoid* layers called *Forget Gate* layer, highlighted in Figure 2.20. This layer takes the hidden state h_{t-1} and x_t as inputs, and outputs a number between zero and one for each sample in the cell state C_{t-1}.

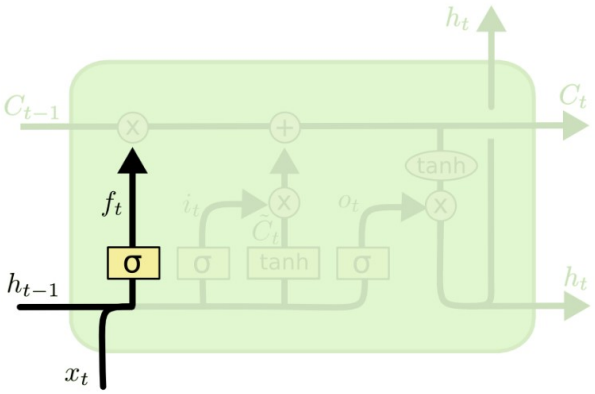


Figure 2.20: The Forget Gate.

The following step is to decide what new information is to be stored in the cell state C at step t. This is divided into two segments where firstly, a second *sigmoid* layer called *Input Gate* layer decides which values shall be updated. Next, a $\tanh$ layer creates a vector of new candidate values, $\tilde{C}_t$, that will be added to the cell state C_t. In the next step, the two segments are combined to create an update to the state.

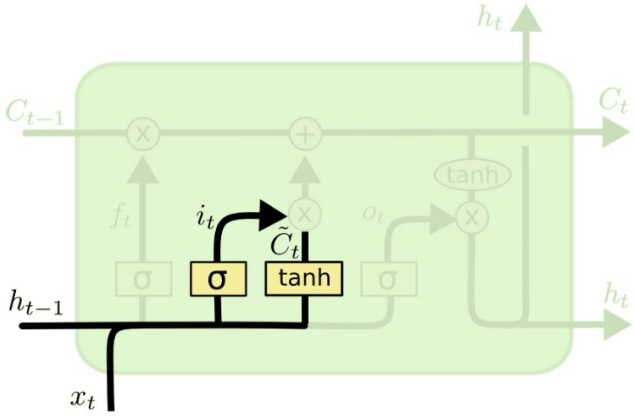


Figure 2.21: The Input Gate.

At this stage, the previous cell state C_{t-1} is updated to the current cell state C_t. Initially, the previous state C_{t-1} is multiplied by f_t, discarding what the *Forget Gate* layer decided. Then, $\tilde{C}_t$ is added, containing the new candidate values scaled by importance by the *Input Gate* layer. This update is depicted in Figure 2.22.

The final step is to decide the output h_t. This output is based on the cell state C_t, yet only after its information is filtered. The filter consists on running a *sigmoid* layer that, alike the one in the first step, decides what shall be discarded from the cell state. In paralel, the cell state is scaled by a $\tanh$ function. Finally, the filtered vector is multiplied by the output of the *sigmoid* layer so that the output h_t only contains what was decided previously. For easier

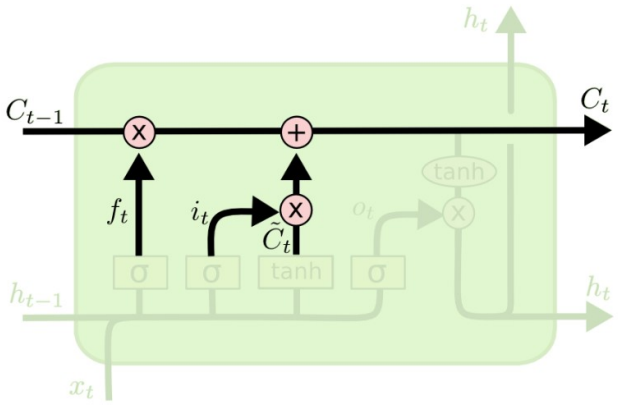


Figure 2.22: The cell state update.

comprehension, refer to Figure 2.23.

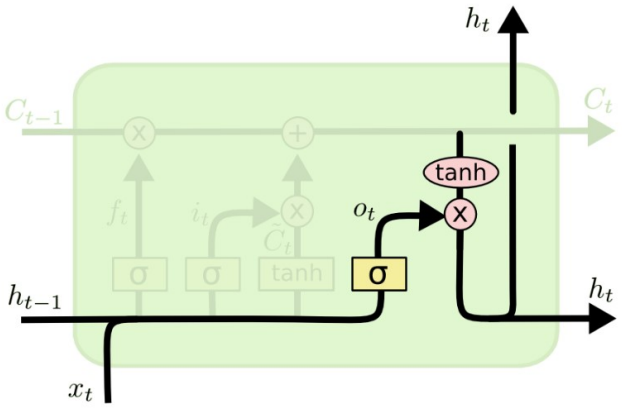


Figure 2.23: The filtered output.

2.6.3.3 LSTM Variants

The workflow described above relates to the standard LSTM architecture. However, literature surrounding LSTMs make use of slight variations to the standard LSTM. Some of these variations include

Peephole connections introduced by Gers and Schmidhuber [61], this variation allows the gate layers to have access to the cell state C.

Coupled *Forget* and *Input* gates this variation couples the decision of what information is to be discarded with the decision of what new information is to be stored.

Gated Recurrent Unit (GRU) introduced by Cho, et al. [62], this dramatic variation combines the *Forget* and *Input* gates into a single *Update Gate*. It also merges the cell state

C with the hidden state h, along with other changes. The result is a simpler architecture than the standard LSTM.

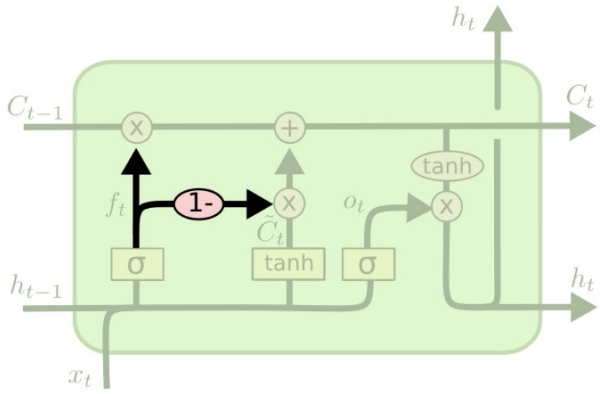


Figure 2.24: Visual representation of the GRU neural network.

Chapter 3

Methodology

3.1 Data Collection

The data set in this dissertation comes from a TLE file. A two-line element set (TLE) is a data format encoding a list of orbital elements of an Earth-orbiting object for a given point in time, the *epoch*. Using a simplified perturbations model (SGP, SGP4, and SGP8), the state (position and velocity) at any point in the past or future can be estimated to some accuracy [63]. The model used in this dissertation is the SGP4, that for a body in a typical low Earth orbit, the accuracy is on the order of 1 km within a few days of the epoch of the element set [64].

The TLE file used describes the state of a *Starlink* satellite, more specifically STARLINK-1028. The contents of this TLE file are detailed in Figure 3.1.

```
STARLINK-1028
1 44733U 19074W   21317.00866860 -.00000348  00000-0 -45010-5 0  9992
2 44733  53.0561  69.1083 0001132  82.4852 277.6266 15.06395718111123
```

Figure 3.1: TLE file of STARLINK-1028.

The meaning of the data presented in Figure 3.1 is as follows

Title line

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	...	26
S	T	A	R	L	I	N	K	-	1	0	2	8													
1																									

Figure 3.2: Title line.

The Title line is a twenty-four character name used to identify the space object in question. The format description is given in Table 3.1.

Table 3.1: Title line description.

Field	Columns	Content	TLE in use
1	$1-24$	Satellite name	STARLINK-1028

Line 1

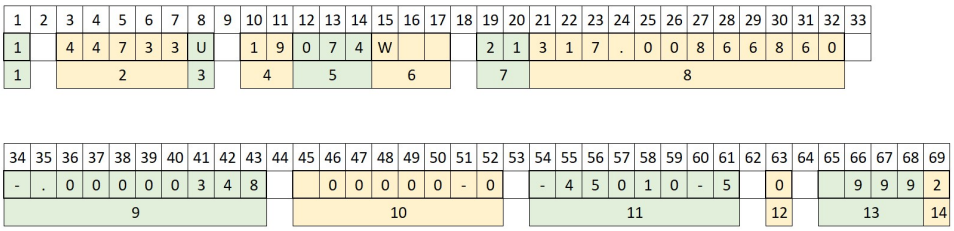


Figure 3.3: Line 1.

Lines 1 and 2 are the standard Two-Line Orbital Element Set Format identical to that used by NORAD and NASA. The format description for Line 1 is given in Table 3.2.

Table 3.2: Line 1 description.

Field	Columns	Content	TLE in use
1	1	Line number of Element Data	1
2	3 – 7	Satellite Catalog number	44733
3	8	Classification (U: unclassified, C: classified, S: secret)	U
4	10 – 11	International Designator (last two digits of launch year)	19
5	12 – 14	International Designator (launch number of the year)	074
6	15 – 17	International Designator (piece of the launch)	W
7	19 – 20	Epoch year (last two digits of year)	21
8	21 – 32	Epoch (day of the year and fractional portion of the day)	317.00866860
9	34 – 43	First derivative of the Mean Motion	−.00000348
10	45 – 52	Second derivative of the Mean Motion (decimal point assumed)	00000 − 0
11	54 – 61	B^* drag term (decimal point assumed)	−45010 − 5
12	63	Ephemeris type	0
13	65 – 68	Element set number	999
14	69	Checksum (Modulo 10)	2

Line 2

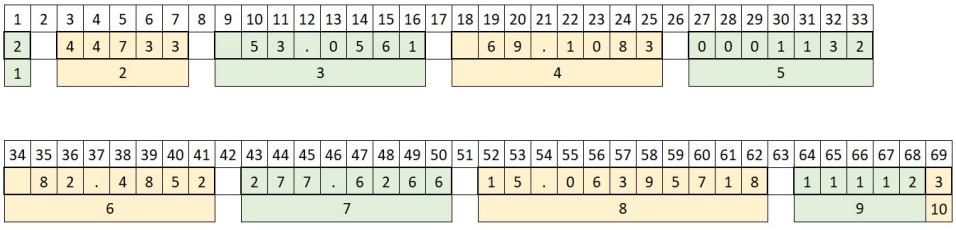


Figure 3.4: Line 2.

To note, the purpose of the colors present in Figure 3.3 and Figure 3.4 is solely to emphasize the change of field. The format description for Line 2 is given in Table 3.3. Also, the main parameters regarding the satellite's orbit at the time of TLE collection are summarized in Table 3.1.

The TLE file was then propagated using the SGP4 propagator with a 10-second time step for 48 hours. The propagation time was not extended due to the decrease in accuracy of the SGP4 after a few days [64].

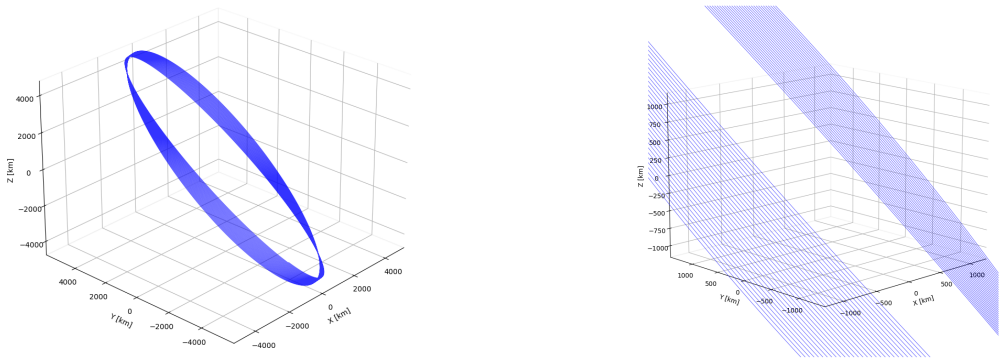
Table 3.3: Line 2 description.

Field	Columns	Content	TLE in use
1	1	Line number of Element Data	2
2	$3-7$	Satellite Catalog number	44733
3	$9-16$	Inclination i [deg]	53.0561
4	$18-25$	Right ascension of the ascending node $RAAN$ [deg]	69.1083
5	$27-33$	Eccentricity e (decimal point assumed)	0001132
6	$35-42$	Argument of perigee ω [deg]	82.4852
7	$44-51$	Mean anomaly M [deg]	277.6266
8	$53-63$	Mean motion [revs per day]	15.06395718
9	$64-68$	Revolution number at epoch [revs]	11112
10	69	Checksum (Modulo 10)	3

Table 3.4: STARLINK-1028 Parameters.

Parameter	Value
UTC epoch	$2021/11/13\ 12:28:97$
Period T [min]	95.59
Inclination i [deg]	53.06
Semi-major axis a [km]	6919
Eccentricity	0.0001504

The result is a data set consisting of a series of state vectors. Each state vector is comprised of the spacecraft's orbital position and velocity, as well as a time stamp. The entire data set has 17281 state vectors provided in an ECEF reference frame and cartesian coordinates ($\{x, y, z\}$ and $\{\dot{x}, \dot{y}, \dot{z}\}$). The data set, as seen in Figure 3.5, contains a record of thirty simulated orbits that the satellite performed around the Earth.



(a) Zoomed out view of the data set.

(b) Zoomed in view of the data set.

Figure 3.5: Overview of the data set.

Although the method of propagating the orbit using the SGP4 is not the most adequate to simulate real-life conditions, the purpose of this dissertation is to assess the applicability of ML to orbit prediction. Additionally, to closer simulate real-life conditions and assess the robustness of the ML model, a parallel data set was generated, where Gaussian noise with a distribution $N(0, 100)$ was added.

3.2 ML Workflow

3.2.1 Data Preparation

The first task performed regarding data preparation was to, although unlikely, check for missing values. Then, to visually inspect whether the data is as anticipated, the state vectors were plotted into a graph.

Additionally, three case studies were designed. Details of each case study are shown in Table 3.5, where, by the number of orbits, one can notice that the number of state vectors increases for each case study. This is to demonstrate that with an increase of data available, comes a higher performance of the model and higher accuracy predictions. For each case study, the data set was split with an average of 80%/10%/10% for the training set, validation set and test set, respectively. The choice for this data split stems from the 80% versus 20% principle of the law of Pareto. Figure 3.6 showcases this split visually.

Finally, before training, the data was scaled between zero and one. The training set was normalized first and then the same transform is blindly applied to the validation set and test set. This is to ensure that there is no risk of information leakage and the validation and test sets remain unbiased and contain ”unseen data”.

Regarding the EKF, the algorithm uses solely the test set of each case study, as it is upon this data that the LSTMs make its predictions.

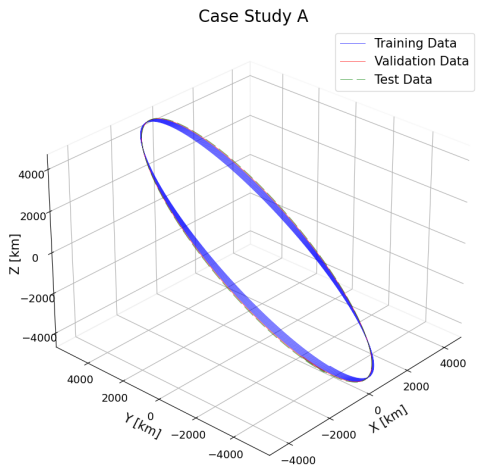
Table 3.5: Case studies details.

Parameter	Case Study A	Case Study B	Case Study C
Training set [%, orbits]	$82, 10$	$80, 20$	$80, 24$
Validation set [%, orbits]	$9, 1$	$10, 2.5$	$10, 3$
Test set [%, orbits]	$9, 1$	$10, 2.5$	$10, 3$

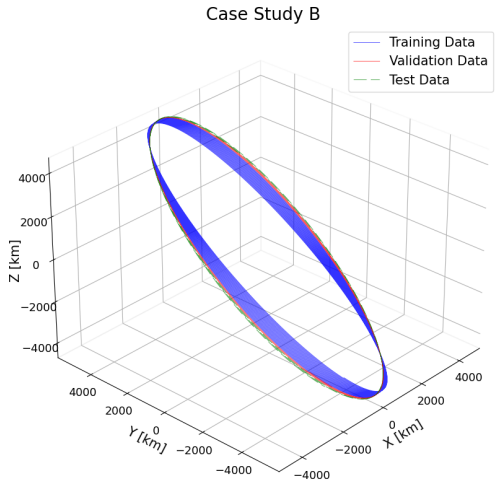
3.2.2 Choosing a model

The model chosen for the purpose of this dissertation is the LSTM, covered in the previous chapter. As for the architecture or configuration, a Multiple Input, Multiple Output (MIMO) approach was taken initially. The reason behind this approach was to feed considerable information to the network in an effort to make it able to learn the underlying patterns of the system. The input comprised of the previous 50 state vectors, resulting in a 300-value input vector. The output comprised of the next time step’s state vector, thus being a 6-value vector in the form of $\{x, y, z, \dot{x}, \dot{y}, \dot{z}\}$. However, this approach turned out to perform poorly after substantial testing.

As an alternative, a Multiple Input, Single Output (MISO) approach was pursued. The goal behind this approach was to simplify the model and make it focus on each single output individually. This means that three LSTM neural networks were necessary, one for each position coordinate. Also, the input for all three LSTMs comprised of the previous 20 state



(a) Case study A data set.



(b) Case study B data set.

(c) Case study C data set.

Figure 3.6: data set of each Case Study.

vectors' position coordinates, as opposed to the the previous 50 state vectors used in the MIMO approach. This resulted in a 60-value input vector and 1 value output for each LSTM.

The LSTMs' parameters were optimized through an iterative process, as there is no standard method to find the perfect parameters for the problem at hand [54]. To note, Pyhton and the Keras and Tensorflow libraries were used to compute all tasks. Also, the Keras Tuner [65] was leveraged to quicken the search, being the Bayesian Optimizer selected as the appropriate tuner to conduct this search.

After an extensive six-month search, testing converged to a simple configuration composed by the input layer, one hidden layer and the output layer. This structure applies to the three LSTMs. However, each model possesses different weights (or neurons) regarding the hidden layer, specified in Table 3.6.

Table 3.6: Models' summary.

Parameter	Input Layer	Hidden Layer	Output Layer
Layer	CuDNNLSTM	CuDNNLSTM	Dense
Neurons (x model)	512	160	1
Neurons (y model)	256	256	1
Neurons (z model)	224	416	1

3.2.3 Model Training

When it comes to training a model, there are various parameters that play a part in its outcome performance. Those parameters include the *loss function*, optimizer, *learning rate*, batch size and the number of epochs the data is trained for. The *loss function* selected was the MSE, for which the computation is described in Equation (3.1).

$$MSE = \frac{1}{n}\sum (y - \hat{y})^2 \tag{3.1}$$

The *Adam* optimizer showed to be the best performer compared to other optimizers like RMSProp, Stochastic Gradient Descent (SDG) and Nadam. The *learning rate* for the optimizer was set at 0.001 with a decay of $1\text{E}-6$. This makes it so that learning is relatively fast initially, though once the model tends towards a local minimum, it decreases [66]. A summary of the training parameters is give in Table 3.7.

Table 3.7: Training parameters.

Parameter	Value
Loss function	MSE
Optimizer	Adam
Learning rate	0.001
Learning decay	$1\text{E}-06$
Training epochs	300
Batch size	32

It is worth mentioning the reason behind the choice of the input and hidden layer, seen in Table 3.6. The CuDNNLSTM layer is an optimized LSTM layer that leverages the use of the GPU and decreases training time considerably. A trade-off when using this layer is the inability to change the activation function, forcing one to work with the $\tanh$ function. This is a minor setback compared to the reduction in training time, being up to five times faster than the standard LSTM layer.

Figure 3.7 explains how training was carried out. Also, training was implemented in a machine equipped with an Intel Core i7-10875H CPU with 32 GB of RAM and a NVIDIA GeForce RTX 2070 GPU with 8 GB of dedicated memory.

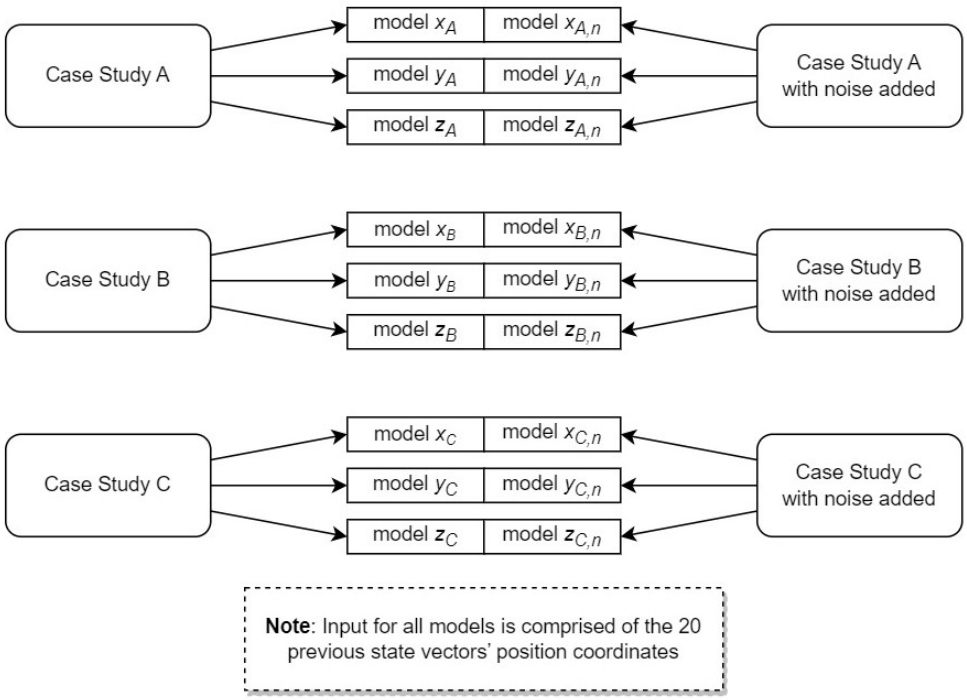


Figure 3.7: Training Scheme.

3.2.4 Model Evaluation

The step that follows training is evaluation. Model evaluation is the process of using different evaluation metrics to understand a machine learning model's performance, as well as its strengths and weaknesses. It is important to assess the efficacy of a model during initial research phases. A common method to perform this evaluation is by using learning curves. Additionally, a pragmatic validation of the model was also performed.

3.2.4.1 Learning Curves

The goal with any machine learning model is *generalization*. Here, the error of the model is plotted as a function of the number of epochs. The error is plotted for both the training set and the validation set.

If the model shows a large bias, a quick convergence to a high error is observed relatively quick for both the training set and the validation set. Since the model is underfitting the data, training on more will not help improve it. A better approach is to consider the addition of features to the data set so that the model can be more equipped to learn the proper relationships.

If the model shows a high variance, a gap between the training error and the validation error is observed. This means the model is performing well for the training data (overfit), and performs poorly for the validation data (not able to generalize). In this case, feeding more data during training can help improve the model's performance.

3.2.4.2 Pragmatic Validation

The pragmatic validation [67] consists of

1. Computing the following three relative errors

   - $\tilde{\sigma}_{train}\left(y_{pred}, y_{true}\right)$, relative error based on 70-80% of the data, or training set.
   - $\tilde{\sigma}_{test}\left(y_{pred}, y_{true}\right)$, relative error based on 20-30% of the data, or test set. The validation set and test set were combined to meet the requirement.
   - $\tilde{\sigma}_{global}\left(y_{pred}, y_{true}\right)$, relative error based on the whole data set.

2. Verifying if the relative errors computed comply with the three constraints shown in Equation (3.2).

The computation of the relative error $\tilde{\sigma}$ is performed by firstly determining the error's ($e_{pred} = y_{pred} - y_{true}$) covariance matrix. y_{pred} represents the predicted position vectors $[\hat{x}, \hat{y}, \hat{z}]$ and y_{true} the true position vectors $[x, y, z]$. Then, from the three *real* and *positive* eigenvalues that can be extracted, the one with the lowest value is selected, λ_1, where $\lambda_1 < \lambda_2 < \lambda_3$. With this, the relative error is taken by the ratio $\frac{\lambda_1}{\lambda_1+\lambda_2+\lambda_3}$. The eigenvalues are *real* and *positive* due to the fact that the error's covariance matrix is a positive definite and symmetric matrix.

$$\begin{aligned} 0 < \tilde{\sigma}_{train}\left(y_{pred}, y_{true}\right) \leq 0.1 \\ 0 < \tilde{\sigma}_{test}\left(y_{pred}, y_{true}\right) \leq 0.2 \\ 0 < \tilde{\sigma}_{global}\left(y_{pred}, y_{true}\right) \leq 0.1 \end{aligned} \tag{3.2}$$

3.2.4.3 Results

Figure 3.8 shows the plotted learning curves, both for the training set and validation set, associated to model x_A (Case Study A, coordinate x). Since the remainder of the models follow the same trends, these are available in Appendix A.

At first, the plot does not resemble typical learning curves (or loss curves) that decrease and remain relatively constant after converging at a local minimum. Several attempts to flatten the curves were made, namely by adjusting the number of epochs, the *learning rate* and its *decay* through the epochs. Although some attempts were successful, predictions were worse than those obtained using the presented models. From [54], promising predictions make for a better trade-off than having flatter learning curves. Also, from Figure 3.8 it is visible that the loss starts off with a low value, in the order of magnitude of $10\text{E}-2$. Over the epochs, the curves tend to, even though slightly, decrease.

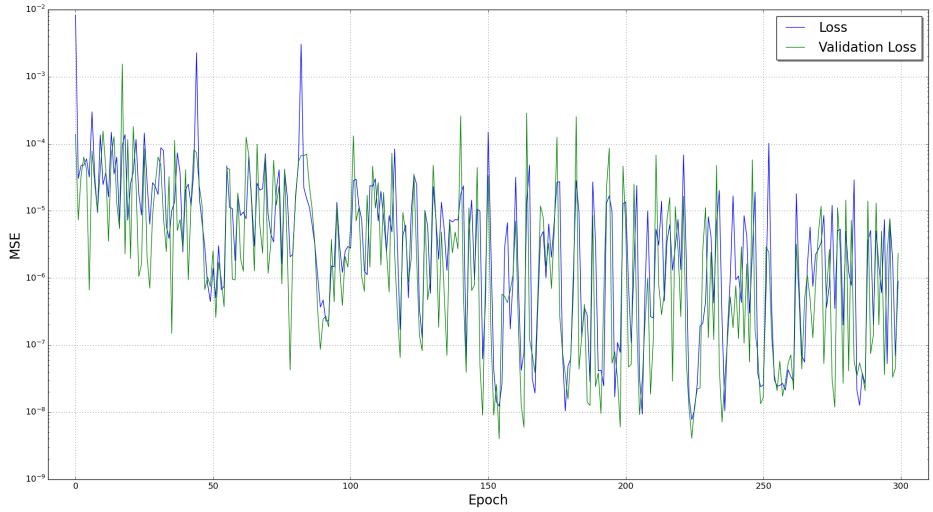


Figure 3.8: Plotted learning curves for model x_A.

For a more in depth analysis, Figure 3.9 details the average loss and validation loss MSE values for each trained model. From the figure, it is noticeable that MSE values tend to decrease as more data is made available for training. It also reinforces Figure 3.8 in showing that both learning curves for each model present values in the same order of magnitude. This is a good indicator that models did not suffer from overfitting or underfitting.

One aspect regarding the use of MSE as a metric is the difficulty to interpret the resulting values. Since the data set is scaled before training, MSE values are dimensionless, having no real meaning if not compared. After carrying out the inverse transformation, values continue to not have a clear interpretation. This is due to the existence of a MSE value for each coordinate position (x, y and z) and the inclusion of the radius of the Earth in the measurements.

As for the Pragmatic Validation, Table 3.8 displays the resulting values, having all models presented satisfactory results that are in accordance with the constraints. A model here is referred to as the complete set of models composed by the three position coordinates. For example, Case Study A refers to the set of models x_A, y_A and z_A. The table also shows that the addition of noise to the data set increased the difficulty of achieving lower relative error values compared to the original data set.

3.2.5 Hyperparameter Tuning

Whenever a model failed its evaluation, tuning was the next step forward. Through the employment of the Keras Tuner [65], time performing this task was greatly reduced. KerasTuner is an easy-to-use, scalable hyperparameter optimization framework that aids the dull stage of hyperparameter search.

The optimizer of choice for this operation was the Bayesian Optimizer. By adaptively select-

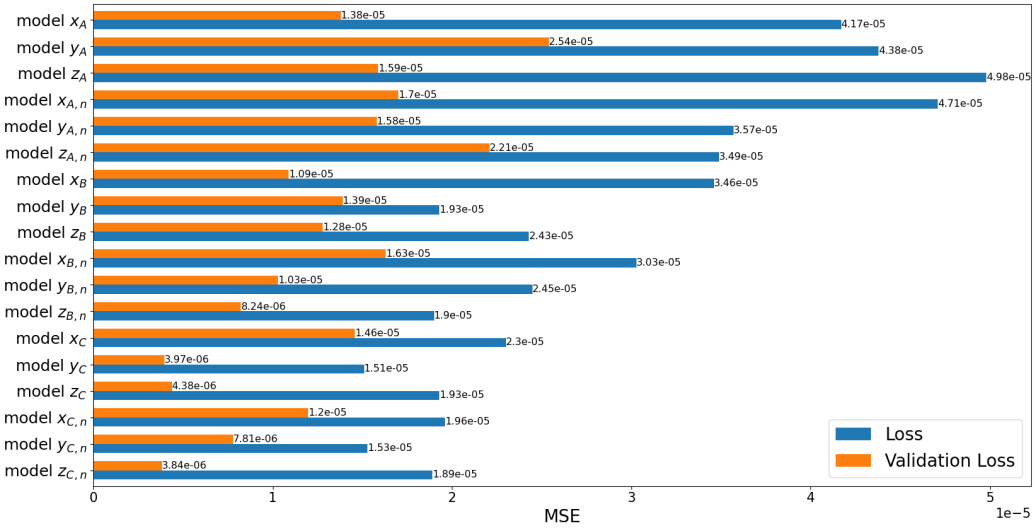


Figure 3.9: Loss and validation loss average MSE values for each trained model.

Table 3.8: Pragmatic Validation results.

	$\tilde{\sigma}_{train}$	$\tilde{\sigma}_{test}$	$\tilde{\sigma}_{global}$
Case Study A	0.031	0.017	0.023
Case Study B	0.070	0.082	0.085
Case Study C	0.062	0.019	0.029
Case Study A with noise added	0.088	0.092	0.084
Case Study B with noise added	0.072	0.065	0.075
Case Study C with noise added	0.085	0.095	0.091

ing configurations, this optimizer takes a mathematical approach to the search. In contrast, other optimizers perform a search that is random in nature. Since the tuner allows for the optimization of a never-ending amount of hyperparameters, the ones this research focused on were

- Number of neurons in each layer.
- Number of *hidden* layers.
- Activation function.
- Addition of regularization.
- Learning rate.
- Training optimizer.
- Loss function.

As expected, the more variables the tuner has to optimize, the longer the search takes. Although not all hyperparameters mentioned were optimized at the same time, it took a cumulative six months of tuning to settle for the configuration used in this dissertation.

3.2.6 Model Deployment

Once hyperparameters are set and the configuration of the models is selected, all that is left is to deploy them. This is done by taking the test set and allowing the models to make predictions on it. To simulate that the test set is composed of new, unseen data, it is scaled according to the parameters that the training set was scaled. The results obtained from the predictions are to be discussed in the following chapter.

3.3 EKF Workflow

Although the focus of this dissertation lies on the development of the LSTM algorithm, it is also worth mentioning the methodology behind the EKF used for comparison of the results.

As previously mentioned, the data given as input to the EKF is composed solely of the test set used for the ML approach. This is due to the simple fact that the EKF does not require training. Also, it utilizes all components of the state vector ($[x, y, z, \dot{x}, \dot{y}, \dot{z}]$). In this approach, EKF processes a single scalar or vector measurement at a time and yields a sequential state estimate at the measurement time. Figure 3.10 represents the flow chart followed to acquire the EKF results. The formulation of the necessary matrices was carried out resorting to [68].

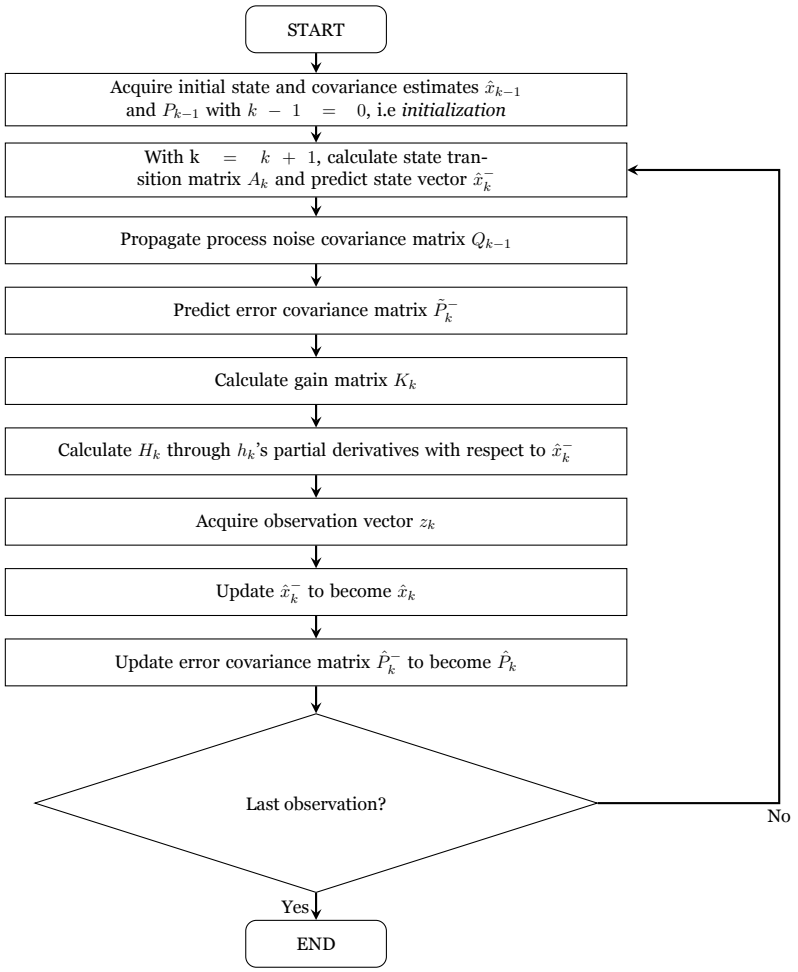


Figure 3.10: Flow chart for EKF based orbit prediction.

Chapter 4

Results

In this chapter, results obtained from the LSTMs are presented and discussed, as well as a comparison between the ML approach and the EKF approach. The LSTMs are to be analyzed regarding their robustness and performance when trained with different training set sizes. Since velocity can be derived from the position vector and changes are greater in position values, results are focused on the prediction error of the position vector $[x, y, z]$. The prediction error e_{pred} is

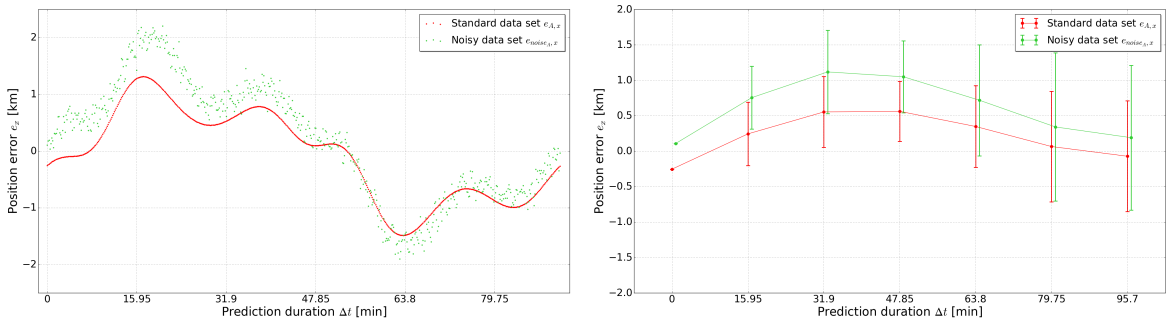
$$e_{pred} = \mathbf{y}_{pred} - \mathbf{y}_{true} \tag{4.1}$$

where $\mathbf{y}_{pred}$ is the position vector predicted by the LSTM or the EKF and $\mathbf{y}_{true}$ is the position vector from the original data set. To note, in the presented plots, the subscript $pred$ in e_{pred} is replaced by the notation of the corresponding case.

4.1 Robustness

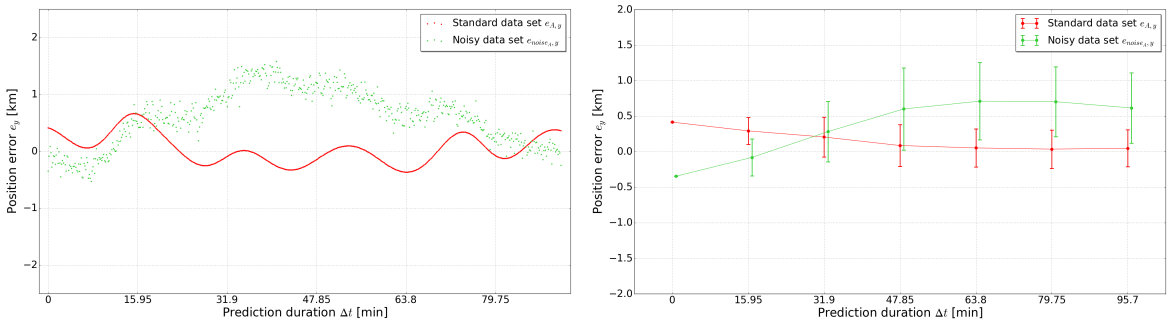
In this section, prediction errors are presented with the purpose of investigating the influence of the addition of noise to the data set. As previously mentioned, a second data set was generated, where Gaussian noise with a distribution $N(0, 100)$ was added.

Figures 4.1, 4.2 and 4.3 demonstrate the results of the LSTM models on both the standard and noisy test data for case study A. Since case studies B and C follow the same trends, plots concerning these case studies are available in Appendix B. The horizontal axis represents the prediction duration Δt. The vertical axis shows the standard and noisy along-track position errors respectively, as annotated by the legend in the figures. The left (a) figures directly show the scatter plots, where the red dot represents the standard data set's error for case study A e_A and the green dot represents the noisy data set's error for the same case study e_{noise_A}. The right (b) figures show the errorbar plots of the left scatter plots, where the center point represents the mean value of each clustered group of the error, and the length from the middle of the bar to the top (or the bottom) represents the standard deviation of the corresponding clustered group. For clarity, the two curves are slightly displaced along the horizontal axis to avoid overlapping.



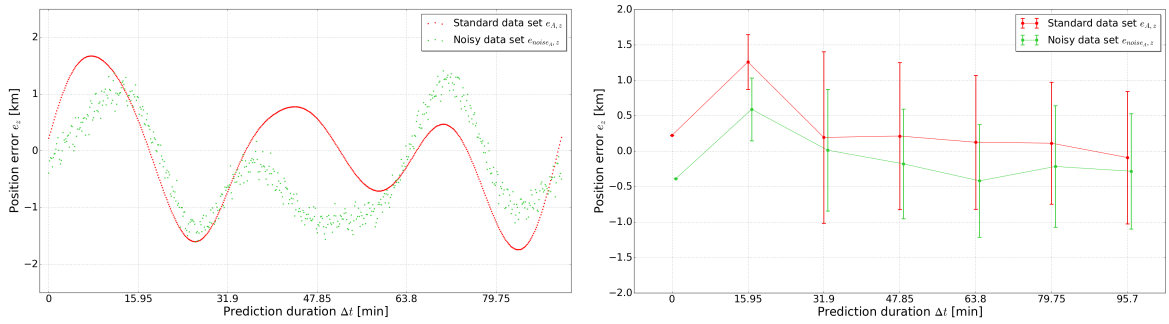
(a) Scatter plot (b) Mean and standard deviation

Figure 4.1: Results of e_x for case study A.



(a) Scatter plot (b) Mean and standard deviation

Figure 4.2: Results of e_y for case study A.



(a) Scatter plot (b) Mean and standard deviation

Figure 4.3: Results of e_z for case study A.

From the left (a) scatter plots one feature stands out: noisy data set errors are more scattered throughout the plot compared to standard data set errors. This is to be expected given the random nature of the Gaussian noise added to the data set. Furthermore, noisy data set errors are in most part greater than standard data set errors. Additionally, in figures 4.1 and 4.3, the noisy data set error follows the trends of the standard data set error, it is assumed that if the standard data set error is known, the behaviour of the noisy data set error is predictable. From the right (b) errorbar plots, it is evident that noisy data set errors have, in general,

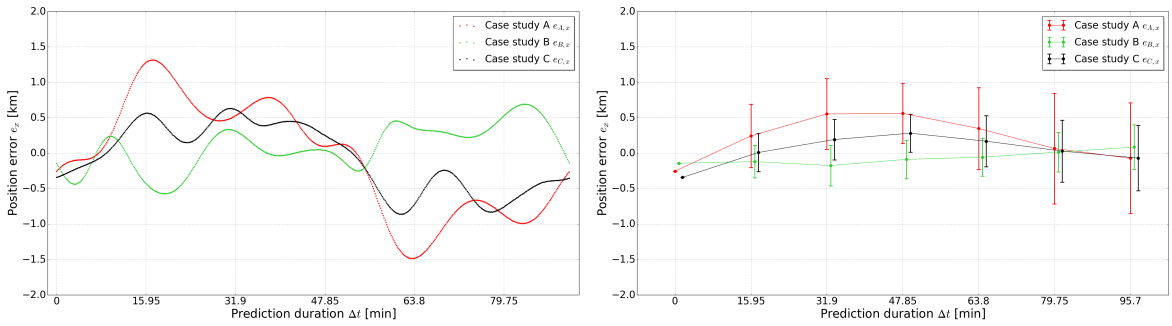
greater mean and standard deviation values than standard data set errors.

It is expected that a noisy data set outputs inferior results compared to a data set with no added Gaussian noise. However, since noisy data set errors do not present a significant difference and are in the same order of magnitude compared to standard data set errors, it is safe to conclude that the models performed well when faced with a data set with added Gaussian noise.

4.2 Training Data Availability

An often accepted concept for the ML method is that the more training data used, the better performance the machine learning model has [29]. In this section, this effect is studied by investigating different partition of the data set on the performance of the trained LSTM models. For context, case studies A and B use 41.7% and 83.3%, respectively, of the training set used in case study C (that makes use of the entire data set). The standard data set was chosen for this purpose as its data is considered as *true* in this dissertation. Also, since the training set for case study A has a time frame of 95.7 minutes, results of case studies B and C were truncated to match this time frame.

Figures 4.4, 4.5 and 4.6 demonstrate the results obtained. The left (a) figures show the scatter plots where the red dot, green dot and the black circle represent the error for case study A e_A, case study B e_B and case study C e_C, respectively. The right (b) figures show the errorbar plots of the left scatter plots. Again, for clarity, the three curves are slightly displaced along the horizontal axis to avoid overlapping.

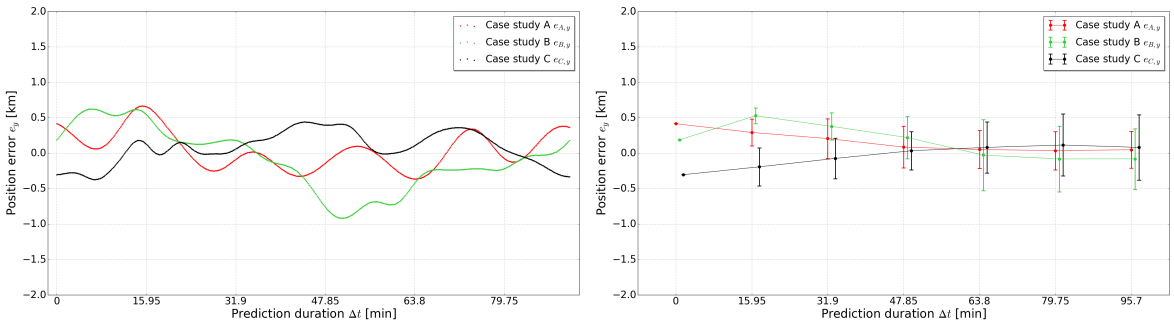


(a) Scatter plot

(b) Mean and standard deviation

Figure 4.4: Results of e_x for all three case studies.

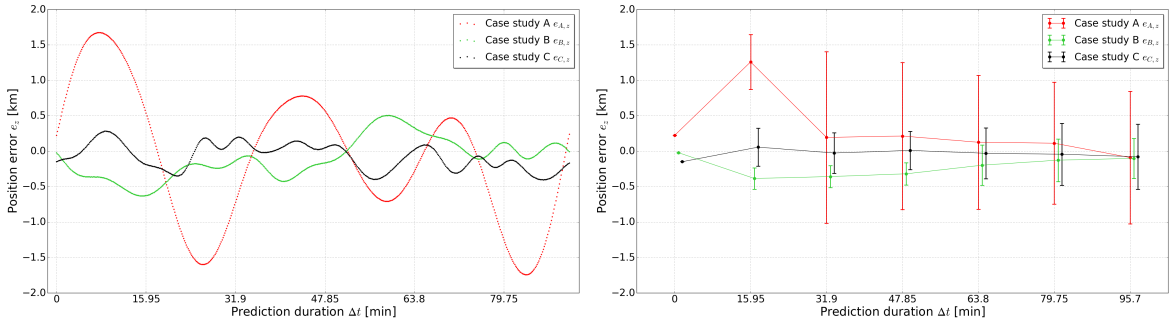
From the left (a) scatter plots, although all figures showcase the effect to some degree, Figure 4.6 demonstrates it in a very noticeable manner. In addition, when comparing e_x, e_y and e_z, it is clear that the position y achieves the best results. Adding to the scatter plots, the right (b) errorbar plots help emphasize the disparity between the errors of case study A and the other two case studies. In contrast, case study B shows comparable performance to case study C, even out performing it at times. In Figure 4.6, while case study C exhibits an error that is distributed around zero almost evenly, case study B denotes an error with a smaller standard



(a) Scatter plot

(b) Mean and standard deviation

Figure 4.5: Results of e_y for all three case studies.



(a) Scatter plot

(b) Mean and standard deviation

Figure 4.6: Results of e_z for all three case studies.

deviation. Given that both case studies' training set size is similar, this is not an abnormal detail.

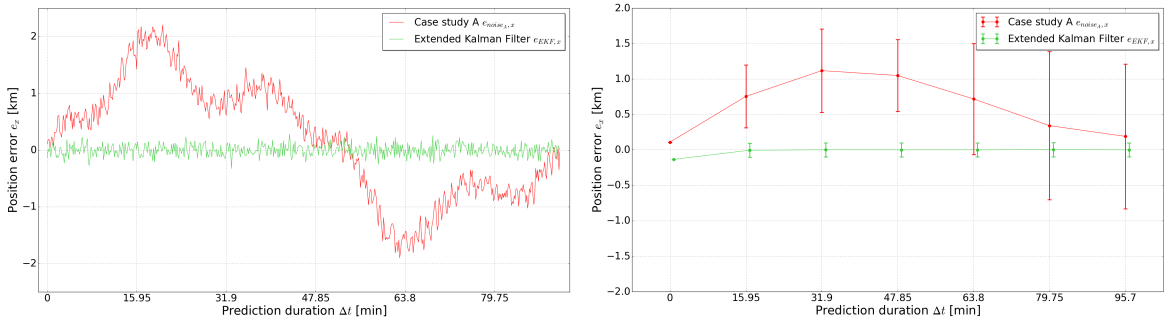
In summary, the effect is obvious when comparing case study A to case study B or C. However, the performance of the latter cases is alike. This is a consequence of the gap between the training set sizes not being significant.

4.3 ML approach versus EKF approach

An EKF was developed with the purpose of comparing its performance to the LSTM's. This section presents the prediction errors for both approaches. The LSTM results displayed pertain to the noisy data set, given that the EKF algorithm utilized this data set. The decision was taken based on the fact that a Kalman filter assumes that measurements (the data set) contain noise and allows one to quantify and take it into account [48].

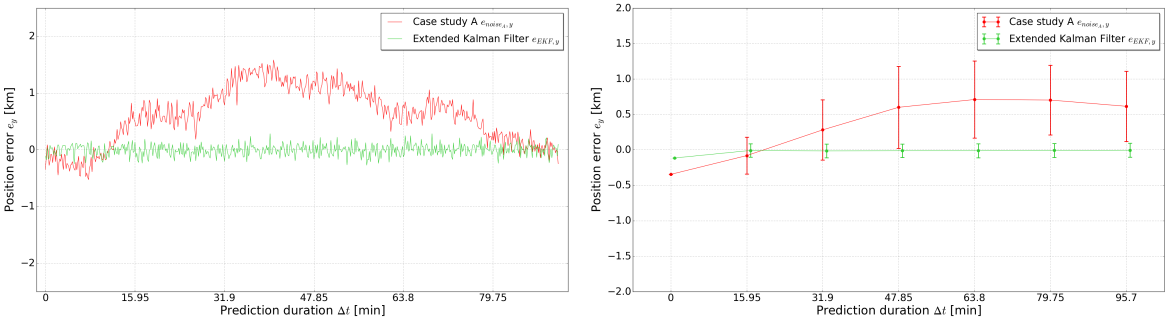
Figures 4.7, 4.8 and 4.9 demonstrate the results of the ML and EKF approaches on the noisy test data for case study A. Since case studies B and C follow the same trends, plots concerning these case studies are available in Appendix C. The left (a) figures showcase the curve plots, where the red curve represents the LSTM's error for case study A e_A and the green curve

represents the EKF's error for the same case study e_{EKF_A}. The right (b) figures show the errorbar plots of the left curve plots.



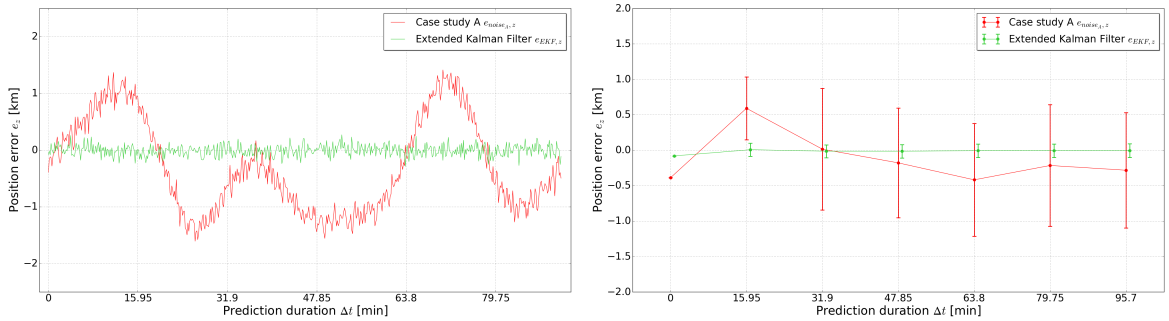
(a) Curve plot (b) Mean and standard deviation

Figure 4.7: Results of e_x for case study A.



(a) Curve plot (b) Mean and standard deviation

Figure 4.8: Results of e_y for case study A.



(a) Curve plot (b) Mean and standard deviation

Figure 4.9: Results of e_z for case study A.

From the left (a) curve plots it is evident that the EKF's error is significantly smaller than the LSTM's. The right (b) errorbar plots further highlight this variation, where EKF's error exhibits a mean that is distributed around zero almost evenly and a minute standard deviation compared to LSTM's error. Also, EKF's error stays consistent throughout the prediction duration, as opposed to LSTM's error, that fluctuates considerably.

Overall, results point out that the EKF has a greater performance than the LSTM. As a side note, since the LSTM models trained on the standard data set and the noisy data set have similar performance, the EKF out performs both perspectives.

Chapter 5

Conclusion and Discussion

A Long Short-Term Memory neural network was developed and investigated.

Firstly, a TLE file pertaining to the satellite STARLINK-1028 was propagated with a 10-second time step for 48 hours using the SGP4 propagator. The resulting data set contained a record of thirty simulated orbits that the satellite performed around the Earth. Additionally, to closer simulate real-life conditions and assess the robustness of the LSTM models, a parallel data set was generated, where Gaussian noise with a distribution N (0, 100) was added.

Then, to demonstrate the effect that the more training data used, the better performance the machine learning model has, three case studies were designed. Case studies A, B and C used 41.7%, 83.3% and 100% of the data set, respectively. Furthermore, for all three case studies, the data set was split with an average of 80%/10%/10% for the training set, validation set and test set, respectively, and scaled between zero and one.

The configuration of the LSTM neural network started with a MIMO approach that turned out to perform poorly after substantial testing. As an alternative, a MISO approach was pursued, meaning a LSTM neural network was necessary for each position coordinate. Each LSTM had an input layer composed of a 60-value vector, a hidden layer and an output layer that returned the prediction. All models followed the same training procedure and were tested for the presence of overfitting and underfitting. Also, a pragmatic validation was performed, where all models presented satisfactory results that were in accordance with the constraints.

The resulting predictions were used to analyze the effect of the size of the training data, the models' robustness and their performance compared to an EKF. Concerning the robustness case, it is expected that a noisy data set outputs inferior results compared to a data set with no added Gaussian noise. However, data revealed that there is not a significant gap between the noisy data set errors and the standard data set errors. Therefore, it is safe to conclude that the models performed well when faced with a data set with added Gaussian noise.

Regarding the effect of the size of the training data, it is evident when comparing case study A to case study B or C, being case study A the under performer. However, between case studies B and C, it is not clear as to which performs best. A reason as to why this might be is that their training set sizes were similar.

Lastly, the EKF demonstrated a greater performance than the LSTM, exhibiting mean val-

ues distributed evenly around zero and almost negligible standard deviation values. Being a beginner in the ML field, this is to show how steep the learning curve is and serves to learn that classical methods are a powerful tool. One other conclusion drawn was that, to design a neural network model that fits the purpose of the research is not straightforward and can take a considerable amount of time.

During the present research dissertation, some assumptions and simplifications were employed, and thus there is still room for improvement.

Regarding data, although a TLE file contains *real* information regarding the satellite at one point in time, this was then propagated, turning it into simulated data. To improve the credibility of the results, working with a data set containing real measurements pertaining to a spacecraft's position and velocity would be the best option. An alternative would be to propagate the TLE file using a higher fidelity propagator than the SGP4.

With respect to the ML workflow, the MSE was selected as the loss function for the training of the models. However, it was later noticed that the use of this metric made it difficult to interpret the resulting values. Investigating other metrics taking into account their interpretation of the data would greatly complement the research. Moreover, the plotted learning curves suggest that the models did not converge at a local minimum. Further tuning the neural networks would result in the flattening of the curves and a higher performance.

Finally, it would be interesting to investigate environments where ML would stand out compared to classical methods, such as orbits where dynamics are harder to model and uncertainty is higher. Also, given that ML is a field in continuous development, applications such as already trained models or physics-based models could yield very promising results.

Bibliography

[1] H. Curtis, *Orbital Mechanics for Engineering Students: Revised Reprint*. Butterworth-Heinemann, 2020. xv, 7, 12, 17, 18, 22

[2] A. Wasicek, "Artificial Intelligence vs. Machine Learning vs. Deep Learning: What's the Difference?" 10 2018. [Online]. Available: https://www.sumologic.com/blog/machine-learning-deep-learning/ xv, 33

[3] A. Bhande, "What is underfitting and overfitting in machine learning and how to deal with it." 6 2018. [Online]. Available: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76 xv, 38

[4] A. C. Boley and M. Byers, "Satellite mega-constellations create risks in low earth orbit, the atmosphere and on earth," *Scientific Reports*, vol. 11, no. 1, pp. 1–8, 2021. 1

[5] F. Caldas and C. Soares, "Machine learning in orbit estimation: a survey," *arXiv preprint arXiv:2207.08993*, 2022. 1, 3

[6] C. Uphoff, "Numerical averaging in orbit prediction," *AIAA Journal*, vol. 11, no. 11, pp. 1512–1516, 1973. 3

[7] N. Z. Miura, "Comparison and design of simplified general perturbation models (sgp4) and code for nasa johnson space center, orbital debris program office," 2009. 3

[8] D. Vallado, P. Crawford, R. Hujsak, and T. Kelso, "Revisiting spacetrack report# 3," in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2006, p. 6753. 3

[9] Y.-z. Luo and Z. Yang, "A review of uncertainty propagation in orbital mechanics," *Progress in Aerospace Sciences*, vol. 89, pp. 23–39, 2017. 3

[10] L. Chen, X.-Z. Bai, Y.-G. Liang, and K.-B. Li, "Orbital prediction error propagation of space objects," in *Orbital Data Applications for Space Objects*. Springer, 2017, pp. 23–75. 3

[11] C. Levit and W. Marshall, "Improved orbit predictions using two-line elements," *Advances in Space Research*, vol. 47, no. 7, pp. 1107–1115, 2011. 4

[12] J. Bennett, J. Sang, C. Smith, and K. Zhang, "Improving low-earth orbit predictions using two-line element data with bias correction," in *Advanced Maui Optical and Space Surveillance Technologies Conference*, vol. 1, 2012, p. 46. 4

[13] J. Sang, B. Li, J. Chen, P. Zhang, and J. Ning, "Analytical representations of precise orbit predictions for earth orbiting space objects," *Advances in Space Research*, vol. 59, no. 2, pp. 698–714, 2017. 4

[14] J. F. San-Juan, I. Pérez, M. San-Martín, and E. P. Vergara, "Hybrid sgp4 orbit propagator," *Acta Astronautica*, vol. 137, pp. 254–260, 2017. 4

[15] A. R. Muldoon, G. H. Elkaim, I. F. Rickard, and B. Weeden, "Improved orbital debris trajectory estimation based on sequential tle processing," *Paper IAC-09 A*, vol. 6, 2009. 4

[16] J. Hartikainen, M. Seppanen, and S. Sarkka, "State-space inference for non-linear latent force models with application to satellite orbit prediction," *arXiv preprint arXiv:1206.4670*, 2012. 4

[17] S. Rautalin, S. Ali-Löytty, and R. Piché, "Latent force models in autonomous gnss satellite orbit prediction," in *2017 International Conference on Localization and GNSS (ICL-GNSS)*. IEEE, 2017, pp. 1–6. 4

[18] H. Peng and X. Bai, "Exploring capability of support vector machine for improving satellite orbit prediction accuracy," *Journal of Aerospace Information Systems*, vol. 15, no. 6, pp. 366–381, 2018. 4

[19] H. Peng and X. Bai, "Comparative evaluation of three machine learning algorithms on improving orbit prediction accuracy," *Astrodynamics*, vol. 3, no. 4, pp. 325–343, 2019. 4

[20] H. Peng and X. Bai, "Gaussian processes for improving orbit prediction accuracy," *Acta astronautica*, vol. 161, pp. 44–56, 2019. 4

[21] H. Peng and X. Bai, "Machine learning approach to improve satellite orbit prediction accuracy using publicly available data," *The Journal of the astronautical sciences*, vol. 67, no. 2, pp. 762–793, 2020. 4

[22] H. Peng and X. Bai, "Fusion of a machine learning approach and classical orbit predictions," *Acta astronautica*, vol. 184, pp. 222–240, 2021. 4

[23] N. F. M. Azmi, S. S. Yuhaniz *et al.*, "An adaptation of deep learning technique in orbit propagation model using long short-term memory," in *2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*. IEEE, 2021, pp. 1–6. 4

[24] G. Curzi, D. Modenini, and P. Tortora, "Two-line-element propagation improvement

and uncertainty estimation using recurrent neural networks," *CEAS Space Journal*, vol. 14, no. 1, pp. 197–204, 2022. 4

[25] J. F. San-Juana, I. Pérezb, E. Vergarac, M. San Martınd, R. Lópeze, A. Wittigf, and D. Izzog, "Hybrid sgp4 propagator based on machine-learning techniques applied to galileo-type orbits," in *69th International Astronautical Congress, Bremen, Germany*, 2018. 4

[26] J. Pihlajasalo, H. Leppäkoski, S. Ali-Löytty, and R. Piché, "Improvement of gps and beidou extended orbit predictions with cnns," in *2018 European Navigation Conference (ENC)*. IEEE, 2018, pp. 54–59. 4

[27] B. Li, Y. Zhang, J. Huang, and J. Sang, "Improved orbit predictions using two-line elements through error pattern mining and transferring," *Acta Astronautica*, vol. 188, pp. 405–415, 2021. 4

[28] B. Li, J. Huang, Y. Feng, F. Wang, and J. Sang, "A machine learning-based approach for improved orbit predictions of leo space debris with sparse tracking data from a single station," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 6, pp. 4253–4268, 2020. 4

[29] H. Peng and X. Bai, "Improving orbit prediction accuracy through supervised machine learning," *Advances in Space Research*, vol. 61, no. 10, pp. 2628–2646, 2018. 5, 61

[30] D. J. Gondelach, *Orbit prediction and analysis for space situational awareness*. University of Surrey (United Kingdom), 2019. 7, 12, 18

[31] V. R. Bond and M. C. Allman, *Modern Astrodynamics: Fundamentals and perturbation methods*. Princeton University Press, 1996, vol. 51. 9

[32] D. A. Vallado, *Fundamentals of astrodynamics and applications*. Springer Science & Business Media, 2001, vol. 12. 12, 18, 20

[33] J. J. Pocha, *An Introduction to Mission Design for Geostationary Satellites*. Springer Science & Business Media, 2012, vol. 1. 12

[34] W. M. Kaula, *Theory of satellite geodesy: applications of satellites to geodesy*. Courier Corporation, 2013. 18, 19

[35] K. F. Wakker, "Fundamentals of astrodynamics," 2015. 19

[36] G. Gedeon, "Tesseral resonance effects on satellite orbits," *Celestial mechanics*, vol. 1, no. 2, pp. 167–189, 1969. 20

[37] A. Rossi, "Resonant dynamics of medium earth orbits: space debris issues," *Celestial Mechanics and Dynamical Astronomy*, vol. 100, no. 4, pp. 267–286, 2008. 21

[38] S. Hughes, "Earth satellite orbits with resonant lunisolar perturbations i. resonances dependent only on inclination," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 372, no. 1749, pp. 243–264, 1980. 21

[39] C.-C. G. Chao, *Applied orbit perturbation and maintenance*. American Institute of Aeronautics and Astronautics, Inc., 2005. 21, 23

[40] A. Rossi, "Resonant dynamics of medium earth orbits: space debris issues," *Celestial Mechanics and Dynamical Astronomy*, vol. 100, no. 4, pp. 267–286, 2008. 21

[41] T. A. Ely and K. C. Howell, "Dynamics of artificial satellite orbits with tesseral resonances including the effects of luni-solar perturbations," *Dynamics and Stability of Systems*, vol. 12, no. 4, pp. 243–269, 1997. 21

[42] A. J. Rosengren, E. M. Alessi, A. Rossi, and G. B. Valsecchi, "Chaos in navigation satellite orbits caused by the perturbed motion of the moon," *Monthly Notices of the Royal Astronomical Society*, vol. 449, no. 4, pp. 3522–3526, 2015. 21

[43] Atmospheric and S. E. CoS, "Guide: Guide to reference and standard atmosphere models (aiaa g-003c-2010 (2016))," 2010. 21

[44] U. S. Atmosphere, *US standard atmosphere*. National Oceanic and Atmospheric Administration, 1976. 21

[45] L. Anselmo and C. Pardini, "Long-term dynamical evolution of high area-to-mass ratio debris released into high earth orbits," *Acta Astronautica*, vol. 67, no. 1-2, pp. 204–216, 2010. 23

[46] G. Welch, G. Bishop *et al.*, "An introduction to the kalman filter," 1995. 23

[47] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960. 23, 24

[48] S. T. Aghav and S. A. Gangal, "Simplified orbit determination algorithm for low earth orbit satellites using spaceborne gps navigation sensor," *Artificial Satellites*, vol. 49, no. 2, pp. 81–99, 2014. 24, 62

[49] M. Seppänen, T. Perälä, and R. Piché, "Autonomous satellite orbit prediction," in *Proceedings of the 2011 International Technical Meeting of The Institute of Navigation*, 2011, pp. 554–564. 24

[50] A. Pukkila, J. Ala-Luhtala, R. Piché, and S. Ali-Löytty, "Gnss orbit prediction with enhanced force model," in *2015 International Conference on Localization and GNSS (ICL-GNSS)*. IEEE, 2015, pp. 1–6. 24

[51] P. S. Maybeck, *Stochastic models, estimation, and control*. Academic press, 1982. 26

[52] R. G. Brown and P. Y. Hwang, "Introduction to random signals and applied kalman filtering: with matlab exercises and solutions," *Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions*, 1997. 26, 27

[53] S. K. Suguna, M. Dhivya, and S. Paiva, "Artificial intelligence (ai): Recent trends and applications," 2021. 32

[54] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022. 32, 51, 54

[55] J. Brownlee, *Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery, 2018. 32

[56] C. Olah, "Understanding LSTM Networks – colah's blog." [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ 32

[57] J. Brownlee, "Overfitting and Underfitting With Machine Learning Algorithms," 2016. [Online]. Available: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/ 37

[58] A. Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks," 5 2015. [Online]. Available: http://karpathy.github.io/2015/05/21/rnn-effectiveness/ 39

[59] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 40

[60] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*. PMLR, 2013, pp. 1310–1318. 40

[61] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 3. IEEE, 2000, pp. 189–194. 44

[62] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical

machine translation," *arXiv preprint arXiv:1406.1078*, 2014. 44

[63] T. Carrico, J. Carrico, L. Policastri, and M. Loucks, "Investigating orbital debris events using numerical methods with full force model orbit propagation," *Adv. Astronaut. Sci*, vol. 130, no. PART 1, pp. 407–426, 2008. 47

[64] T. Kelso *et al.*, "Validation of sgp4 and is-gps-200d against gps precision ephemerides," 2007. 47, 48

[65] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi *et al.*, "Kerastuner," https://github.com/keras-team/keras-tuner, 2019. 51, 55

[66] V. Haswani, "Learning Rate Decay and methods in Deep Learning," 12 2021. [Online]. Available: https://medium.com/analytics-vidhya/learning-rate-decay-and-methods-in-deep-learning-2cee564f910b 52

[67] K. Bousson, "Advanced python and data science," *CFIUTE Lecture Notes, 2021, (in Portuguese)*, 2021. 54

[68] Q. Lam, D. Junker, D. Anhalt, and D. Vallado, "Analysis of an extended kalman filter based orbit determination system," 08 2010. 57

Appendix A

Learning Curves plots

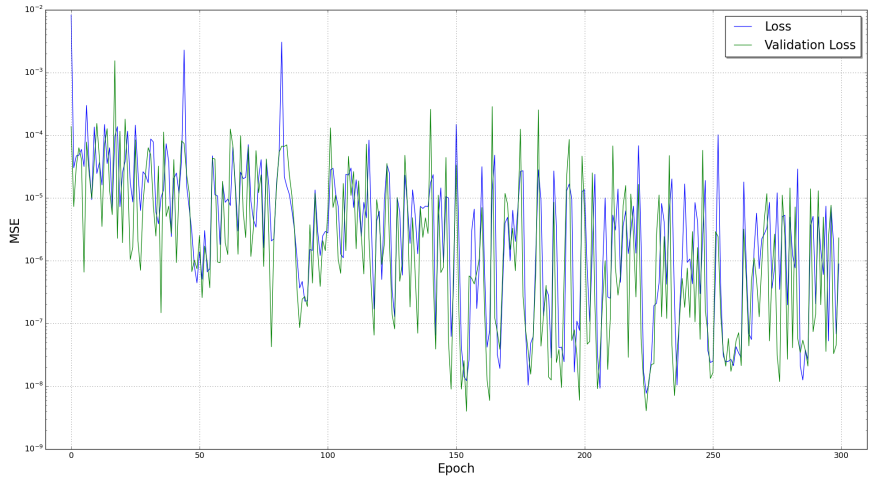


Figure A.1: Plotted learning curves for model x_A.

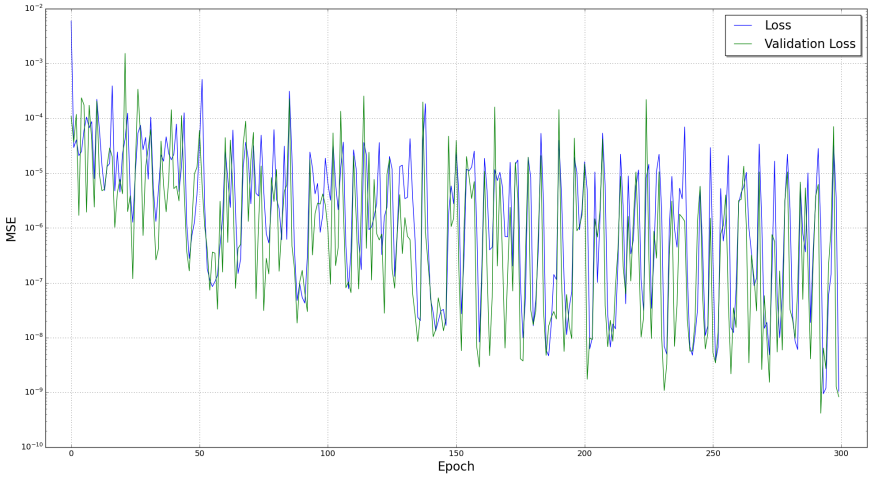


Figure A.2: Plotted learning curves for model y_A.

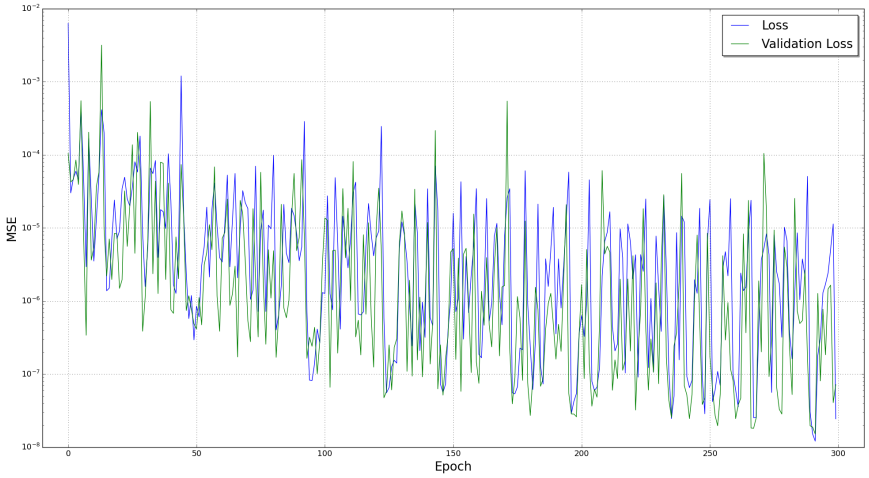


Figure A.3: Plotted learning curves for model z_A.

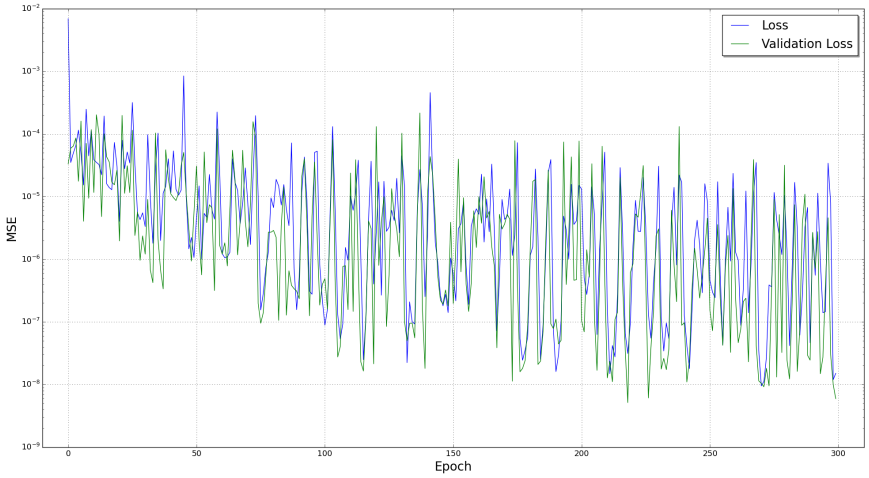


Figure A.4: Plotted learning curves for model $x_{A,n}$.

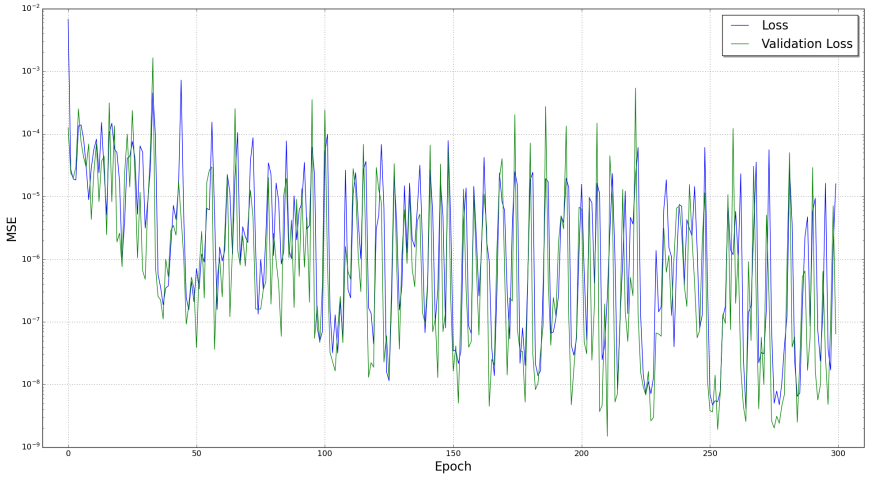


Figure A.5: Plotted learning curves for model $y_{A,n}$.

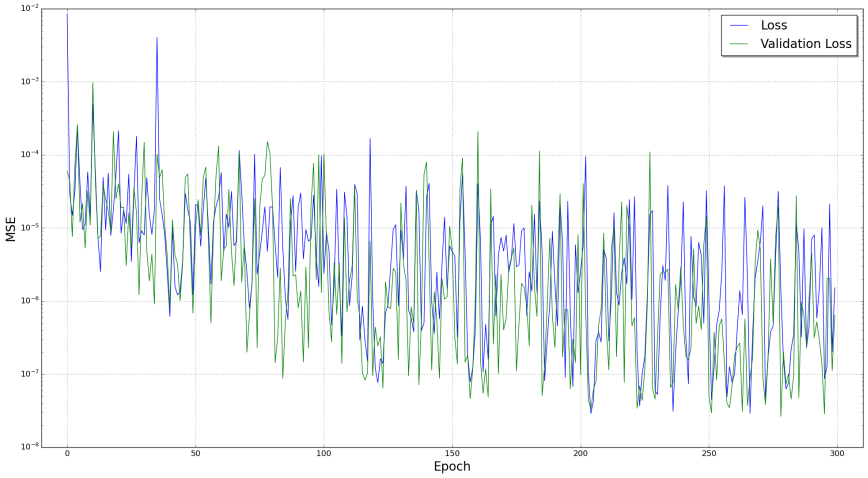


Figure A.6: Plotted learning curves for model $z_{A,n}$.

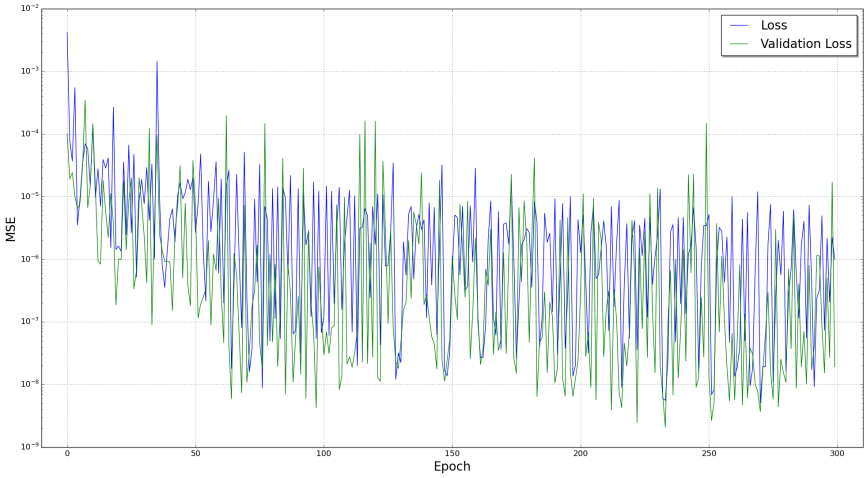


Figure A.7: Plotted learning curves for model x_B.

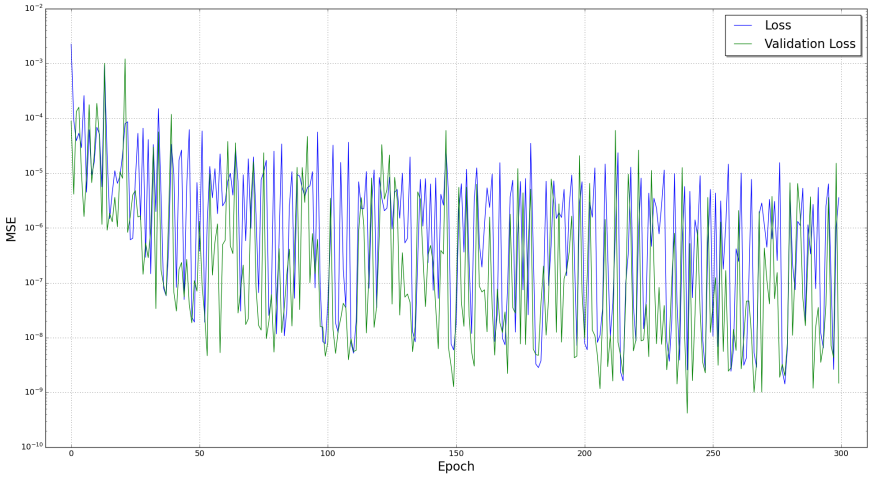


Figure A.8: Plotted learning curves for model y_B.

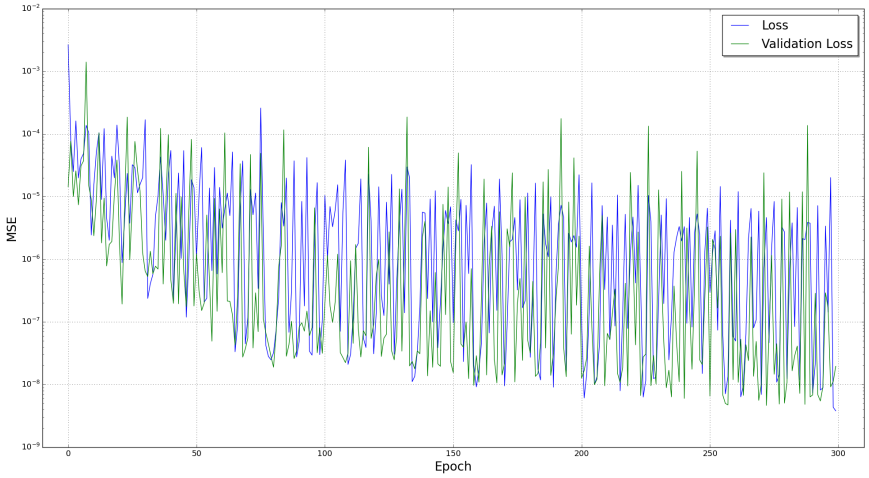


Figure A.9: Plotted learning curves for model z_B.

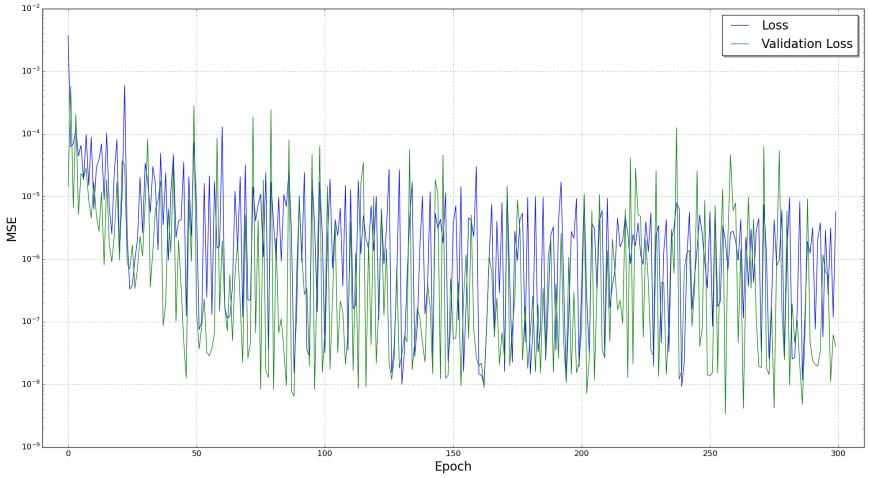


Figure A.10: Plotted learning curves for model $x_{B,n}$.

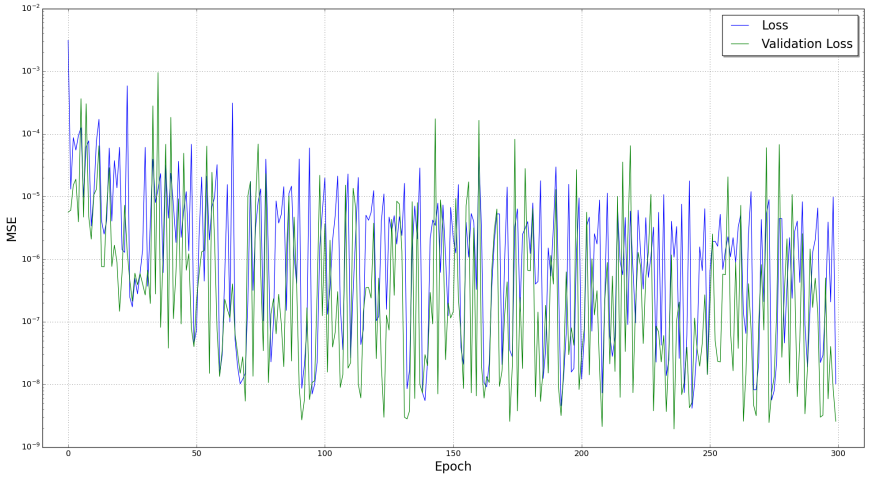


Figure A.11: Plotted learning curves for model $y_{B,n}$.

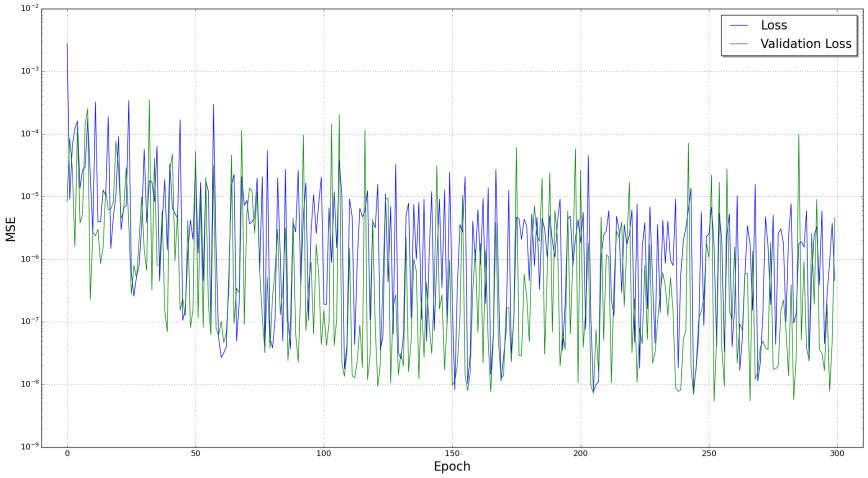


Figure A.12: Plotted learning curves for model $z_{B,n}$.

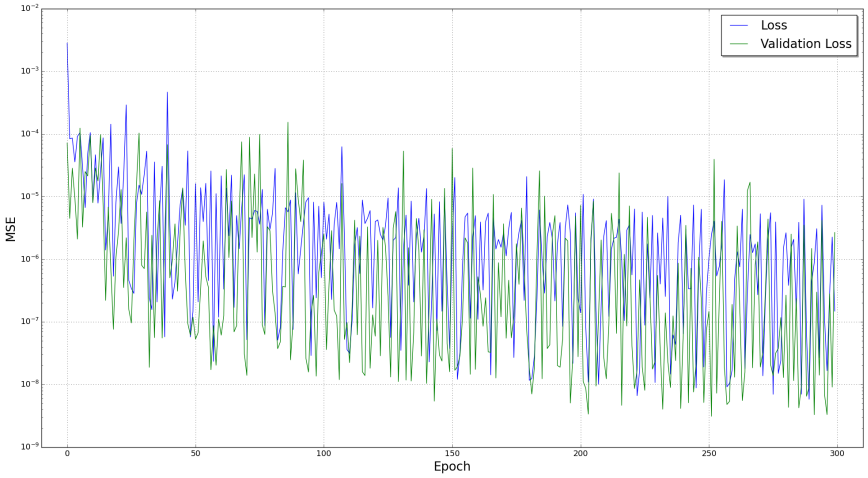


Figure A.13: Plotted learning curves for model x_C.

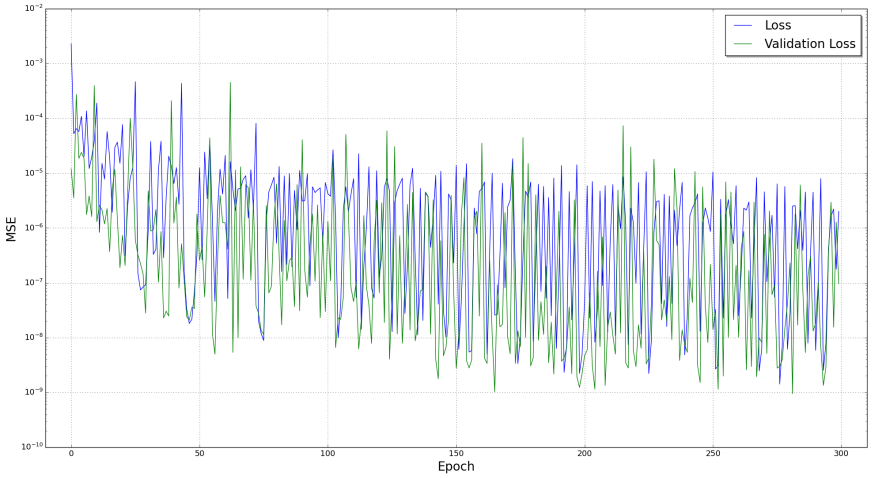


Figure A.14: Plotted learning curves for model y_C.

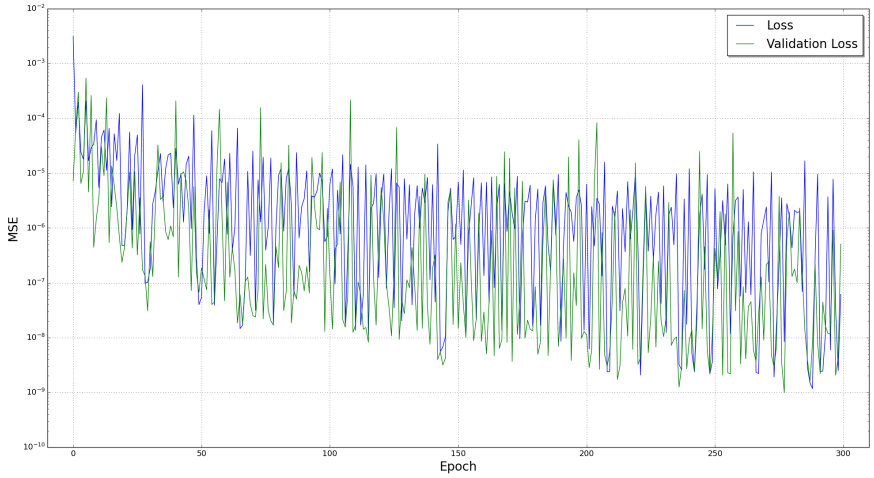


Figure A.15: Plotted learning curves for model z_C.

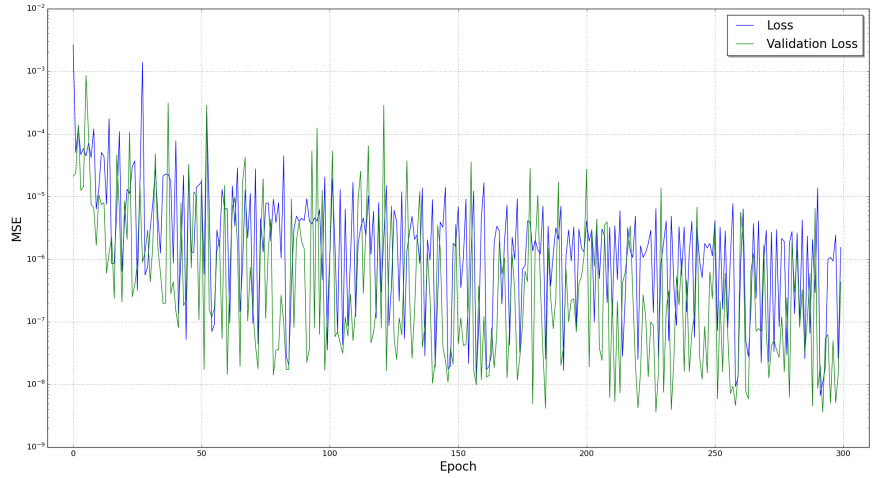


Figure A.16: Plotted learning curves for model $x_{C,n}$.

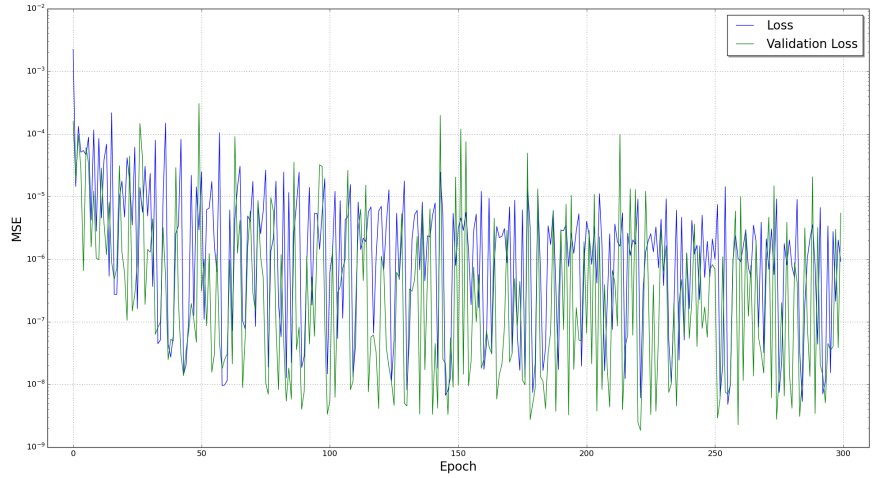


Figure A.17: Plotted learning curves for model $y_{C,n}$.

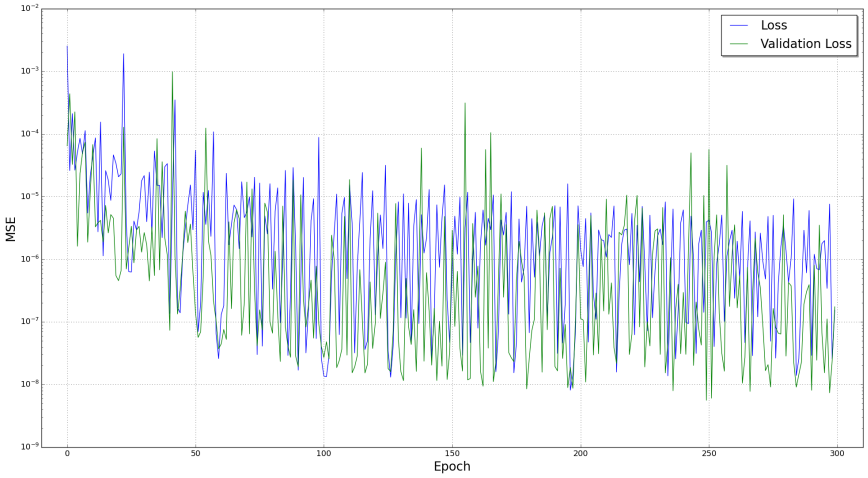
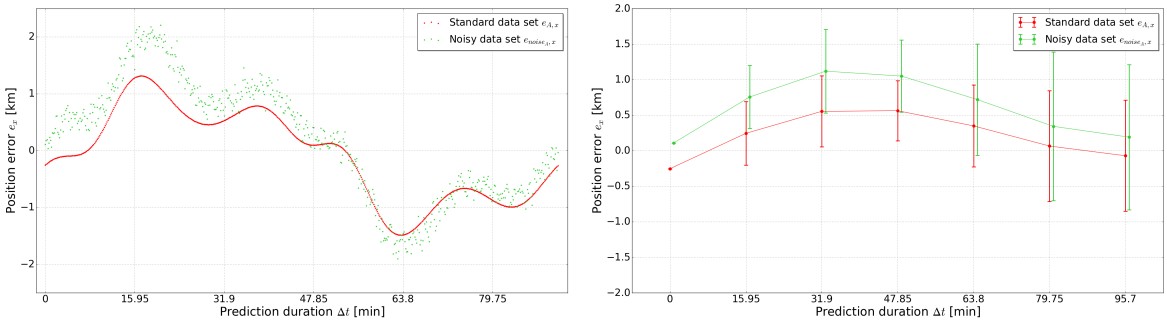


Figure A.18: Plotted learning curves for model $z_{C,n}$.

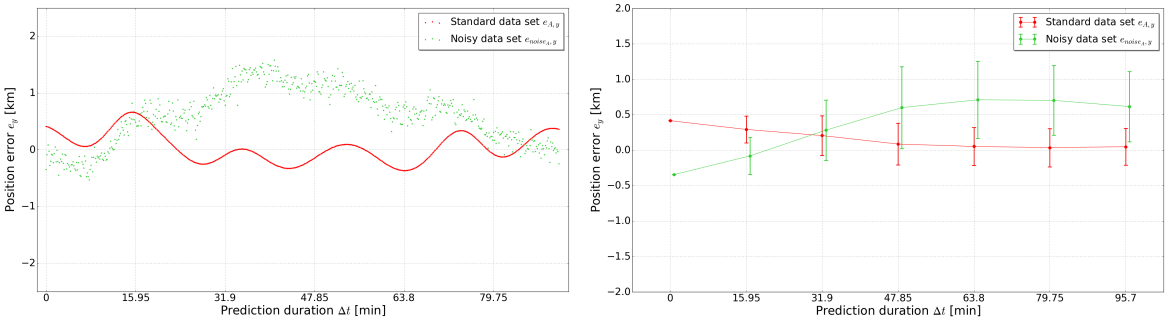
Appendix B

Robustness plots



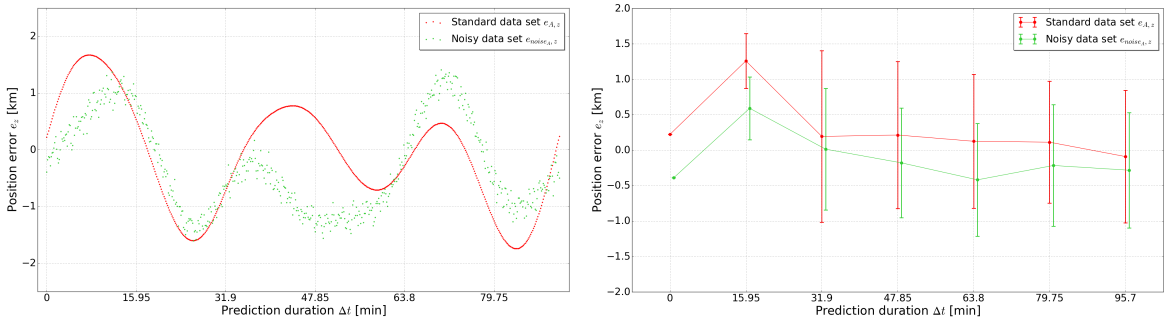
(a) Scatter plot (b) Mean and standard deviation

Figure B.1: Results of e_x for case study A.



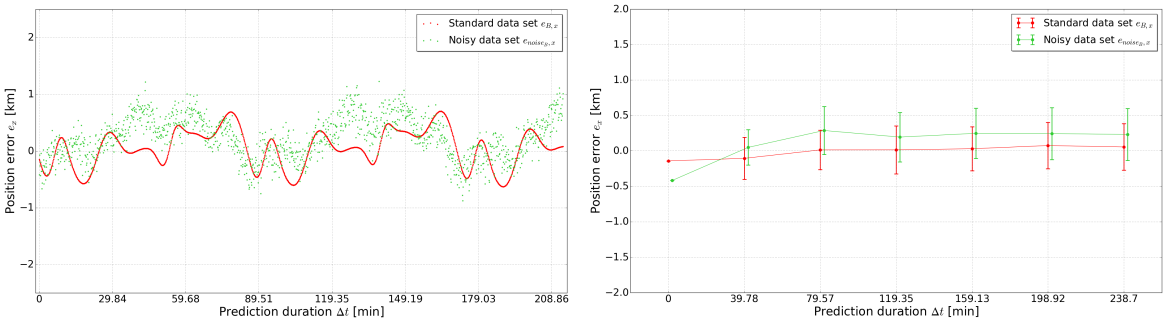
(a) Scatter plot (b) Mean and standard deviation

Figure B.2: Results of e_y for case study A.



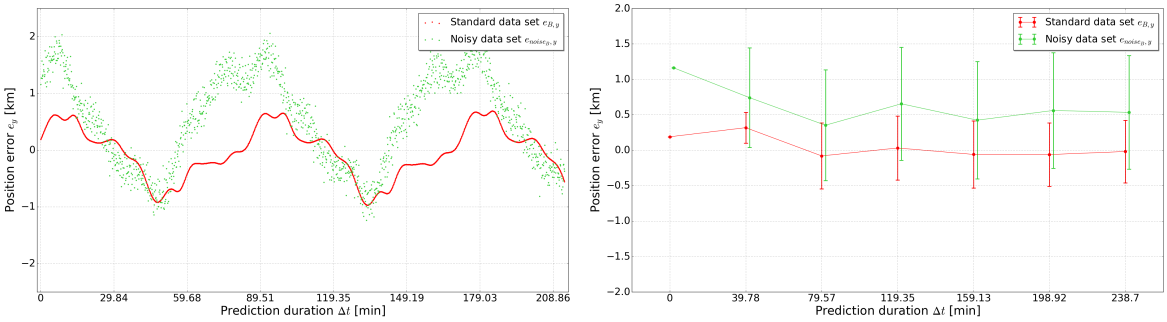
(a) Scatter plot (b) Mean and standard deviation

Figure B.3: Results of e_z for case study A.



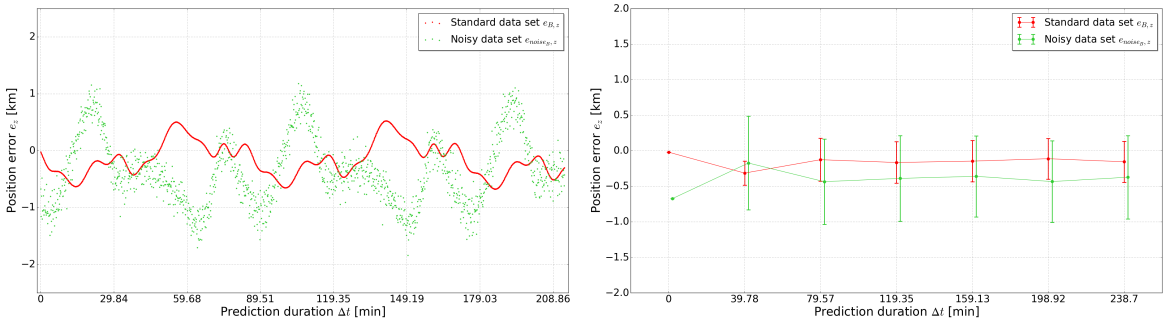
(a) Scatter plot (b) Mean and standard deviation

Figure B.4: Results of e_x for case study B.



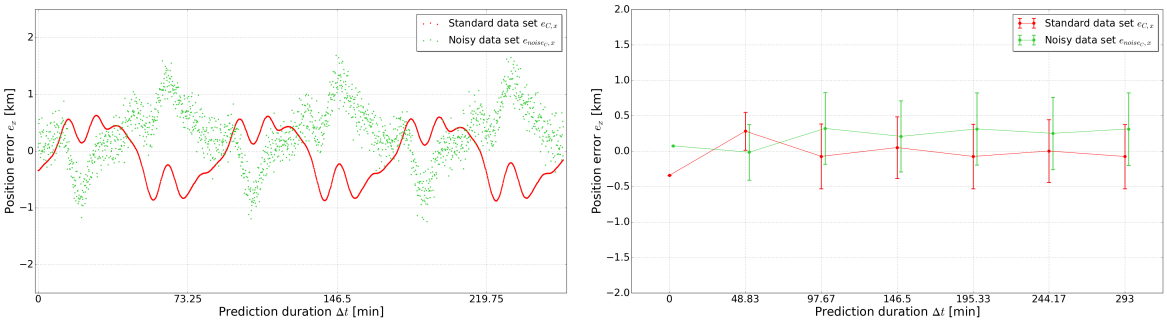
(a) Scatter plot (b) Mean and standard deviation

Figure B.5: Results of e_y for case study B.



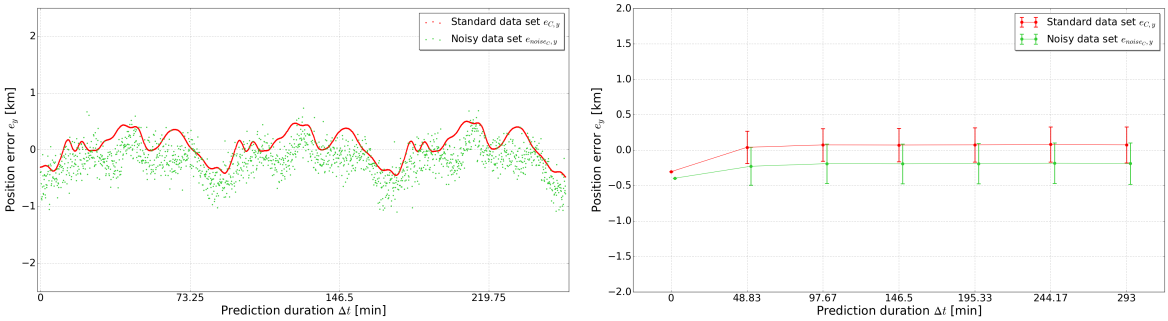
(a) Scatter plot (b) Mean and standard deviation

Figure B.6: Results of e_z for case study B.



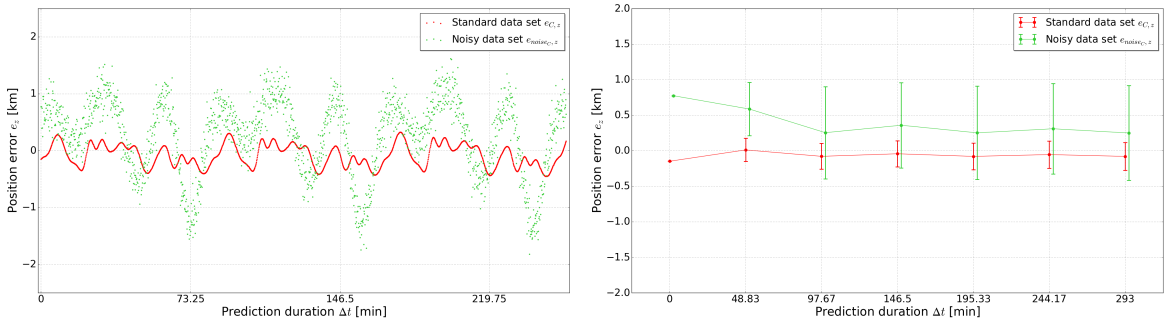
(a) Scatter plot (b) Mean and standard deviation

Figure B.7: Results of e_x for case study C.



(a) Scatter plot (b) Mean and standard deviation

Figure B.8: Results of e_y for case study C.

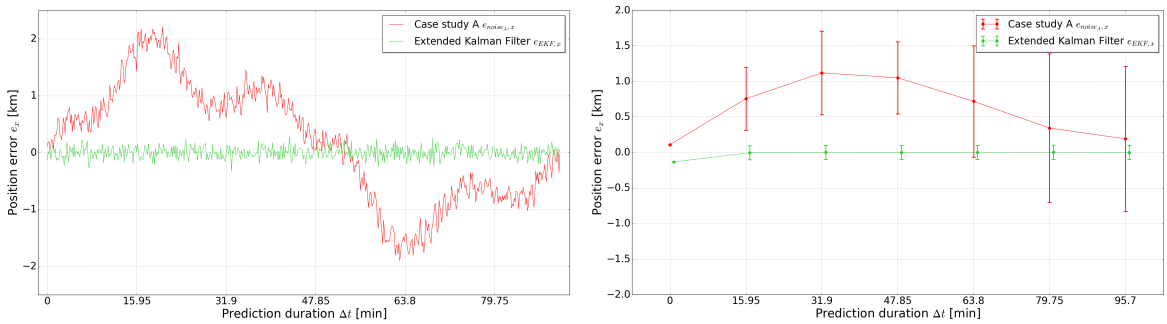


(a) Scatter plot (b) Mean and standard deviation

Figure B.9: Results of e_z for case study C.

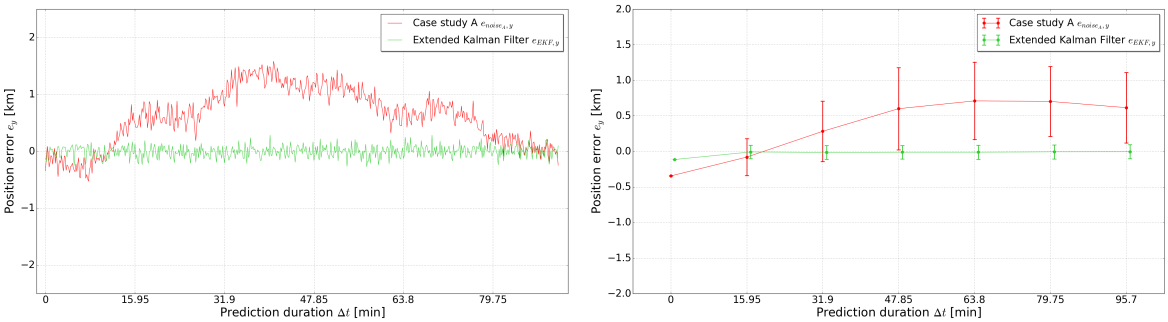
Appendix C

ML approach versus EKF approach plots



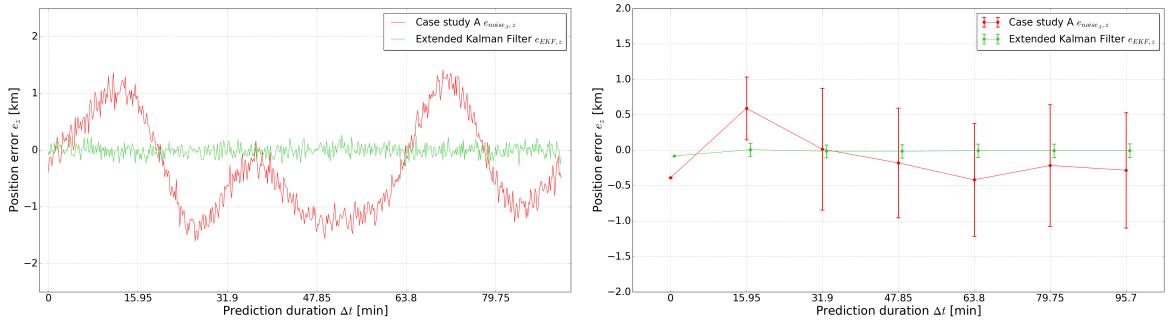
(a) Curve plot (b) Mean and standard deviation

Figure C.1: Results of e_x for case study A.



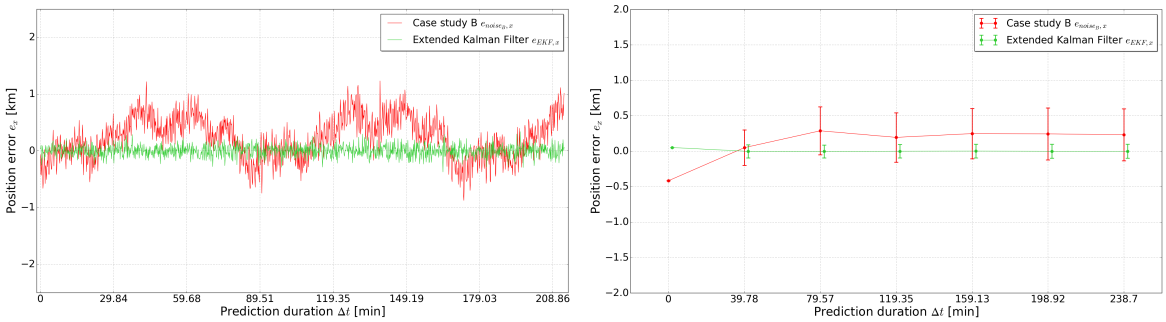
(a) Curve plot (b) Mean and standard deviation

Figure C.2: Results of e_y for case study A.



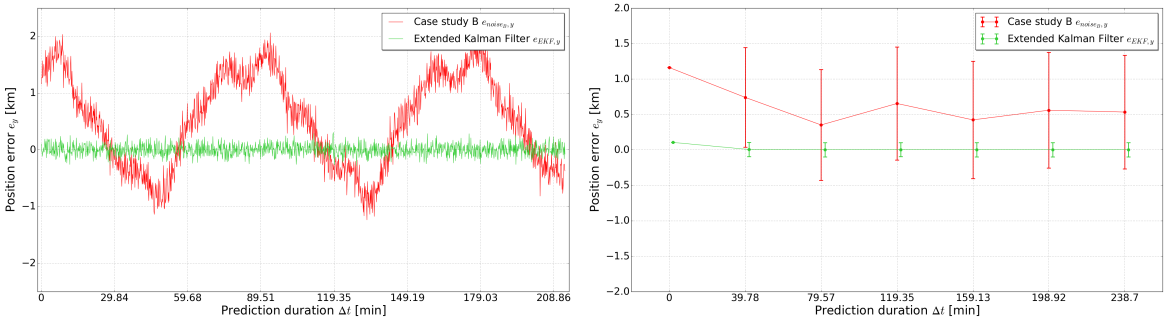
(a) Curve plot (b) Mean and standard deviation

Figure C.3: Results of e_z for case study A.



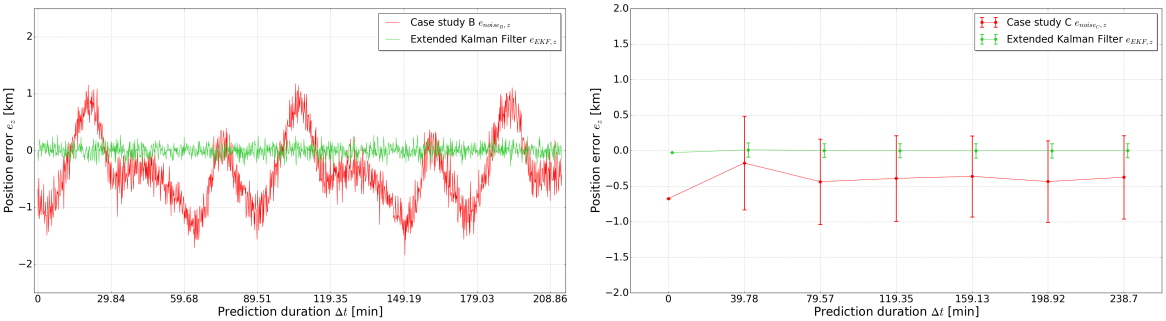
(a) Curve plot (b) Mean and standard deviation

Figure C.4: Results of e_x for case study B.



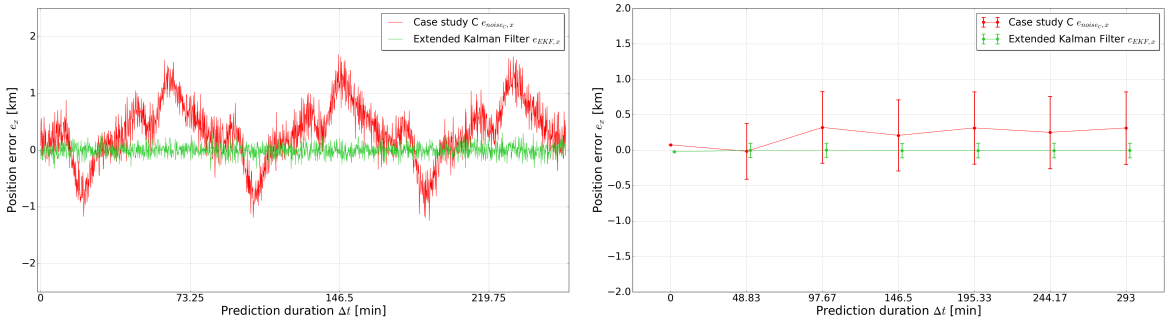
(a) Curve plot (b) Mean and standard deviation

Figure C.5: Results of e_y for case study B.



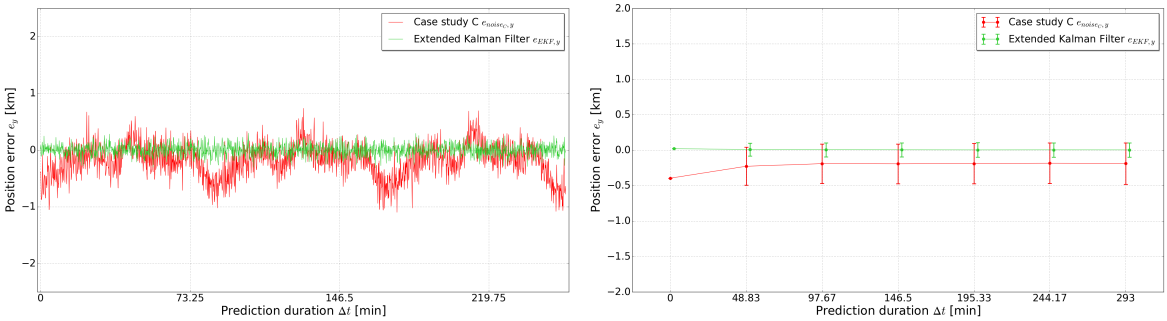
(a) Curve plot (b) Mean and standard deviation

Figure C.6: Results of e_z for case study B.



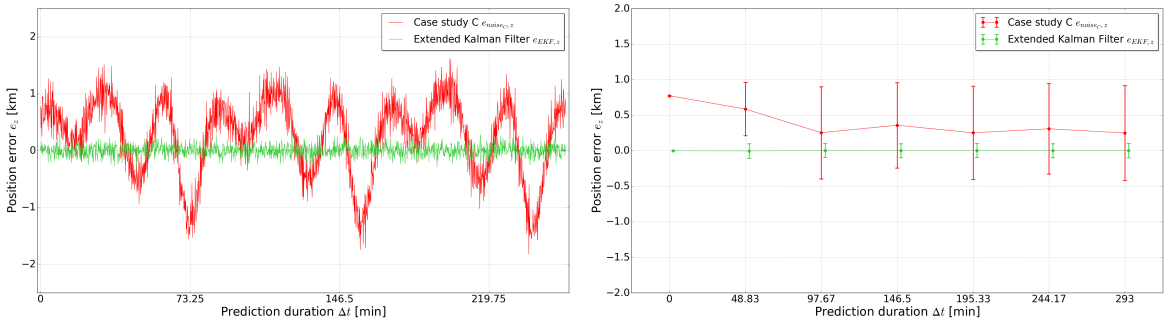
(a) Curve plot (b) Mean and standard deviation

Figure C.7: Results of e_x for case study C.



(a) Curve plot (b) Mean and standard deviation

Figure C.8: Results of e_y for case study C.



(a) Curve plot (b) Mean and standard deviation

Figure C.9: Results of e_z for case study C.

Appendix D

Academic Publications

The development of this dissertations resulted in the following publications.

D.1 5th EJIL-Meeting of Young Researchers of LAETA, Lisbon, 5-6 May 2022

Title Neural Network Based Orbital Trajectory Prediction

Description Physics-based models and estimation methods can often limit orbit prediction accuracy. One other factor is their capability to reduce TTFF (Time To First Fix). With the hypothesis that a Machine Learning (ML) approach can learn the underlying pattern of the orbit prediction errors from large amounts of observed data, in this paper, a LSTM (Long Short Term Memory) Neural Network is explored for improving orbit prediction accuracy and reducing TTFF. The LSTM architecture was chosen since it addresses the common long-term dependency problem (vanishing or exploding gradient) when using BPTT (Back Propagation Through Time). To validate the results, a variation of the conventional Kalman Filter was implemented. The EKF (Extended Kalman Filter) was chosen for being the simplest real-time estimation algorithm with adequate tuning of its parameters. The neural network model used leveraged on its generality, orbit prediction accuracy, and computational cost for real-time orbit prediction and onboard environment. The performance of the algorithm was assessed using actual orbit data of a ballistic trajectory and a LEO satellite orbit.

Keywords Orbit prediction, neural networks, Kalman filtering, ballistic trajectory, LEO satellite.

D.2 73rd International Astronautical Congress (IAC), Paris, 18-22 September 2022

Title Machine Learning based Orbit Prediction

Description Physics-based models and estimation methods can often limit orbit prediction accuracy for being characterized by a high degree of complexity and nonlinearity. With the hypothesis that a Machine Learning (ML) approach can learn the underlying pattern of the orbit prediction errors from large amounts of observed data, a LSTM (Long Short - Term Memory) Neural Network is explored for improving orbit prediction accuracy. The LSTM architecture was chosen since it addresses the common long-term dependency problem (vanishing or exploding gradient) when using BPTT (Back Propagation Through Time). To validate the results, a variation of the conventional Kalman Filter was implemented. The EKF (Extended Kalman Filter) was chosen for being the simplest real-time estimation algorithm with adequate tuning of its parameters. The neural network model that was used leveraged on its generality, orbit prediction accuracy, and computational cost for real-time orbit determination and onboard environment. The performance of the algorithm was assessed using TLE data from a LEO satellite.

Keywords Orbit prediction, neural networks, Kalman filtering, LEO satellite.

D.3 International Symposium on Aircraft Technology (ISAT-ECH) 2022, Belgrade, 14-16 September 2022

This page was purposely left in blank. The academic article begins in the next page.

Neural Control of Space Trajectories with Pseudolinear Models

Pedro M. C. Belizário, K. Bousson, Filipe Senra

Abstract: This paper describes an approach to neural control of a satellite trajectory. A pseudolinear model is created to generate the necessary training data for the neural network. This model uses an H∞ to stabilize the relative motion of a satellite concerning another satellite. The purpose of this paper is to show the feasibility of such an approach and to better understand the benefits of using a previously trained neural network to control a satellite.

Keywords: pseudolinear, relative motion, ANN.

1. Introduction

Countless space missions rely on successful rendezvous which requires precise control with minimal error (Park et al.1999). Linear optimal control theory has worked very well in designing linear controllers that drive the system to its desired output (Çimen et al.2008). Nonetheless, with recent technological advancements also came the recent applications of nonlinear control. Nonlinear controllers have the advantage of being closer to controlling real-life systems. However, nonlinear controllers are often more computationally expensive than linear ones and they are also slower because difficult algorithms must be solved and the solutions must be found online. For that reason, there has been considerable research and interest in the use of neural networks to identify and control nonlinear systems. Neural networks have the promise of being faster and more robust with the downside of having a long training time (Annaswamy and Yu,1998; Calise, 1996). In this paper, the neural network is intended to be trained offline and learn from data created beforehand.

The paper is organized as follows: Section 2 where the problem is introduced, Section 3 where the model is presented, Section 4 where the controller is explained, and Section 5 and 6, where the results and conclusion are shown, respectively.

2. Problem Statement

In the present paper, a rendezvous problem with the target spacecraft in a circular orbit, such as the ISS, is considered. Let us consider that a rendezvous model can be written in the following nonlinear way:

A1. Belizário(✉) A2. Bousson A3. Senra
Universidade da Beira Interior (UBI), Covilhã, Portugal
e-mail: pedro.belizario@ubi.pt; bousson@ubi.pt; filipesenra98@hotmail.com
Supported by LAETA-AEROG in the framework of the Project UIDB/50022/2020.

T.H. Karakoç et al. (eds.), Name of Symposium Proceedings,
DOI 10.1007/…….

$$\dot{x} = A(x)x(t) + B(x)u(t) + D(x)w(t), \;\; x(0) = x_0$$
$$z(t) = Ex(t)$$

(1)

Where $x(t) \; \epsilon \; \mathbb{R}^n$ is the state and vector $u(t) \; \epsilon \; \mathbb{R}^m$ is composed of three independent accelerations used as control inputs and $w(t) \; \epsilon \; \mathbb{R}^p$ is a disturbance vector. Then the purpose of this work is to design a robust neural controller that can find adequate control inputs for any given time.

3. Relative Motion Dynamics

Assuming the target is in a circular orbit. Denoting **r** and **R** as the vector from the target spacecraft to the chaser spacecraft, and vector from the centre of the Earth to the target spacecraft, respectively then the relative motion of the chaser spacecraft in an Earth-centred inertial frame can be written as:

$$\frac{d^2\boldsymbol{r}}{dt^2} = -\mu\left(\frac{\boldsymbol{R}+\boldsymbol{r}}{|\boldsymbol{R}+\boldsymbol{r}|^3} - \frac{\boldsymbol{R}}{|\boldsymbol{R}|^3}\right) + \boldsymbol{a}_f$$

(2)

In which μ is the Earth's gravitational constant and $\boldsymbol{a}_f$ is the acceleration vector due to thrust forces on the chaser spacecraft.

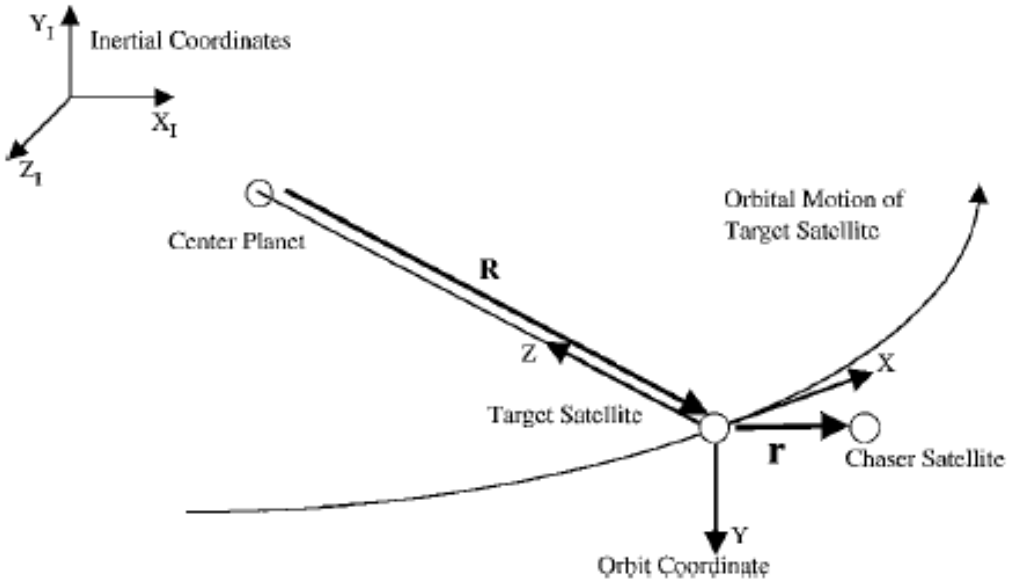


Figure 1 Coordinates and definition (Yamanaka and Ankersen,2002)

Considering the target-orbital coordinate system shown in **Fig.1**, then **r** comes written as $\boldsymbol{r} = [x \; y \; z]^{\boldsymbol{T}}$. Assuming the target is in a circular orbit, then the orbital rate ω is constant. Therefore, equation (2) can be written as the following set of equations:

$$\begin{bmatrix}\ddot{x}\\ \ddot{z}\\ \ddot{y}\end{bmatrix}=\begin{bmatrix}2\omega\dot{z}+\omega x^2-\dfrac{\mu x}{|\boldsymbol{R}+\boldsymbol{r}|^3}\\ \omega^2 z-2\omega\dot{x}-\mu\left(\dfrac{z-R}{|\boldsymbol{R}+\boldsymbol{r}|^3}+\dfrac{1}{R^2}\right)\\ -\dfrac{\mu y}{|\boldsymbol{R}+\boldsymbol{r}|^3}\end{bmatrix}+\boldsymbol{a}_f \quad (3)$$

The model in equation (3) (Zhou et al,2011), now needs to be written as a pseudolinear model, that is, the A matrix is parametrized and written in a state-dependent coefficient way. Therefore, the system of equations in 3 can be written as:

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{z}\\ \ddot{z}\\ \dot{y}\\ \ddot{y}\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0\\ \omega^2-\frac{\mu}{|x,y,(z-R)|^3} & 0 & 0 & 0 & 0 & 2\omega\\ 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & -\frac{\mu}{|x,y,(z-R)|^3} & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\\ \frac{\mu x}{|x,y,(z-R)|^3(1+x^2)} & -2\omega & 0 & 0 & \omega^2-\frac{\mu}{|x,y,(z-R)|^3} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ z\\ \dot{z}\\ y\\ \dot{y}\end{bmatrix}+\begin{bmatrix}0\\ 0\\ 0\\ 0\\ 0\\ -\frac{\mu}{R^2}\end{bmatrix}+\boldsymbol{a}_f \quad (4)$$

Where the constant added at the end can be viewed as a deterministic perturbation.

4. Neural Control Design

This section is divided into two parts, the first part concerns the robust control method chosen and succinctly explains the theory behind it. This controller will generate the training data for the network, as explained in the second part.

4.1 H∞ Controller

As it can be seen by the system of equations (4), the model is not linear, as for every state vector **x**, the matrix A changes. In the literature, this is often called the State-Dependent Riccati Equation method (SDRE) and makes use of state-dependent coefficient matrices (Çimen, 2008). These matrices are used to solve an algebraic Riccati equation to give a suboptimal control law. Since the matrices vary with every point in state-space, the Riccati equation will give a different solution for every point in state-space also. This method captures the nonlinearities of a given system while at the same time permitting great design versatility.

In addition to using a pseudolinear model, the specific controller from which the neural network learns is an H∞ controller. In short, a gain matrix K is found solving the following Riccati equation (Khargonekar et al, 1998):

$$PA+A'P-\frac{1}{\epsilon}PBR^{-1}B'P+\frac{1}{\gamma}PDD'P+\frac{1}{\epsilon}E'E+\varepsilon Q=0 \quad (5)$$

Where γ is a perturbation attenuation constant.

Therefore, the gain matrix K which stabilizes the system can be written as:

$$K = -\frac{R^{-1}B'P}{2\varepsilon} \tag{6}$$

And with $u = K(x)x(\mathrm{t})$

4.2 Neural Network

The artificial neural network (ANN) structure is made up of a hidden layer with 4 neurons, so with 6 inputs, we're left with a 6-4 ANN format. Due to computational constraints, it was decided that the neural controller had to be trained using three separate neural networks, one for each control input. This avoids the need for a deeper network, greatly reducing the computational power for the simulation.

An Adam Optimization algorithm was chosen, and the activation function of every layer was Sigmoid.

5. Numerical Simulation

To first simulate the controller and acquire the necessary data for the ANN, realistic values were given to an initial vector **x**, as to simulate real scenarios in which the relative position isn't equal to 0. As the relative error between x_{k+1} and x_k gets closer to an acceptable value, a new initial vector **x** is given and the process of controlling the system to the desired state starts again. This was repeated until there were enough values to train the ANN. Below are the results of the neural network training.

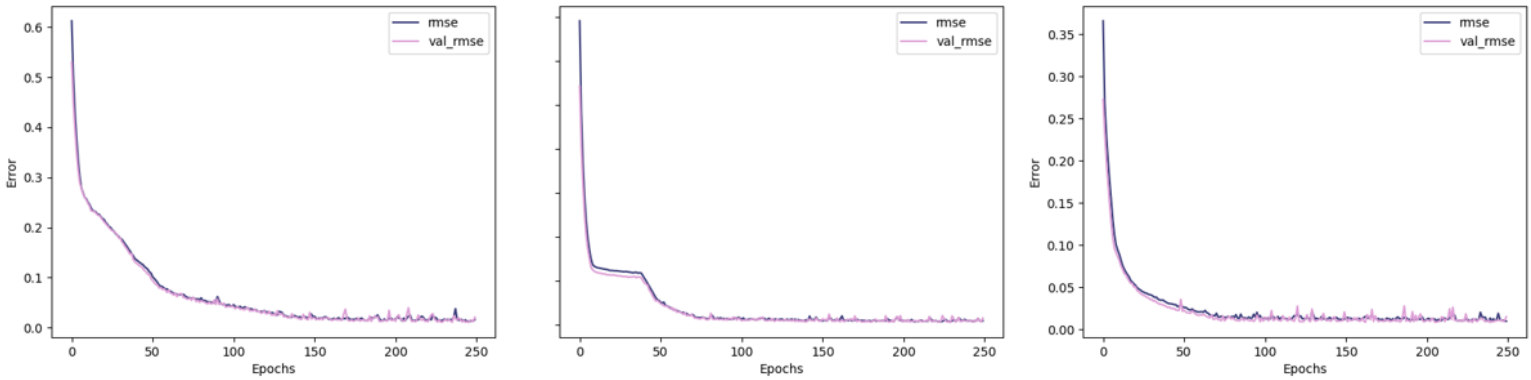


Figure 2 ux,uy,uz neural network, respectively

Overfitting was an issue as at some point the validation error started to increase, however, some parameters such as the learning rate and the number of epochs were tweaked to achieve a good enough model.

Fig.3 shows an example of the neural network model controlling the system for any given, realistic vector of relative motion.

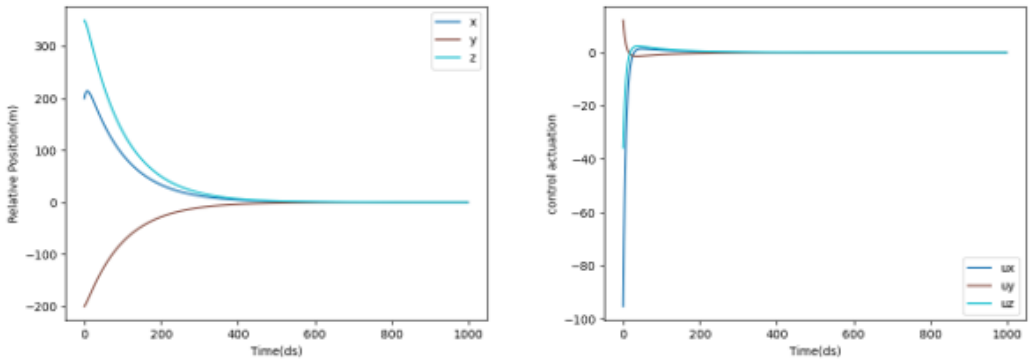


Figure 3 left-Relative position; right-Control inputs

As it can be seen, the relative position vector is stabilized and driven to zero. Since the relative position vector is simply the difference between the position vector of the target and the position vector of the chaser, then this means that the chaser now coincides with the target, and the neural controller is working as intended.

6. Conclusion and Future Work

This paper aimed to design a neural controller based on pseudolinear models. Neural networks have the advantage of being computationally cheaper to use versus computing the solution of a nonlinear controller in real-time, with the disadvantage of the training time, and running into problems related to overfitting. To ensure that a good neural controller was developed, enough training data had to be created. Said data was created using a pseudolinear model of relative motions dynamic, and the system was stabilized for every point using an H_∞. This required solving an algebraic Riccati equation for every single point in state-space since the A matrix was written in a state-dependent coefficient form. Afterward, the neural controller was designed and trained, and the desired result was achieved.

Future work could incorporate the equations of motion for a satellite in the Earth's spherical gravitational field with the J2 perturbation to validate the robustness of the model.

References

1. Annaswamy AM, Yu S (1998) Stable Neural Controllers for Nonlinear Dynamic Sytems, in: Automatica, Vol. 34, No. 5. pp. 641-650 https://doi.org/10.1016/S0005-1098(98)00012-0

2. Calise AJ (1996) Neural Networks in Nonlinear Flight Control, in: IEEE AES Systems Magazine Vol. 11, No. 7, pp. 5-10 10.1109/62.533965

3. Çimen (2008) State-Dependent Riccati Equation (SDRE) Control: A Survey, in: Proceedings of the 17th World Congress The International Federation of Automatic Control Vol. 41, No. 2, pp. 3761-3775 https://doi.org/10.3182/20080706-5-KR-1001.00635

4. Khargonekar PP, Petersen IR, Rotea MA (1988) H_∞ Optimal Control with State-Feedback, in: IEEE Transactions on Automatic Control Vol.33, No 8. pp. 786-788 10.1109/9.1301

5. Park J, Choi K, Lee S (1999) Orbital rendezvous using two-step sliding mode control, in: Aerospace Science and Technology, 1999, no.4, pp. 239-245. https://doi.org/10.1016/S1270-9638(99)80046-7

6. Yamanaka K, Ankersen F (2002) New State Transition Matrix for Relative Motion on an Arbitrary Elliptical Orbit, in: Journal of Guidance, Control and Dynamics Vol.25, No.1. pp. 60-66 https://doi.org/10.2514/2.4875

7. Zhou B, Lin Z, Duan G (2011) Lyapunov Differential Equation Approach to Elliptical Orbital Rendezvous with Constrained Controls, in: Journal of Guidance, Control and Dynamics Vol.34, No.2. pp. 345-358 https://doi.org/10.2514/1.52372