



ORBIT DETERMINATION WITH ANGULAR MEASUREMENTS  
BY USING ARTIFICIAL NEURAL NETWORK

A THESIS SUBMITTED TO  
THE SCHOOL OF GRADUATE STUDIES OF  
UNIVERSITY OF TURKISH AERONAUTICAL ASSOCIATION

BY

KÜRŞAT YENİDOĞAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR  
THE DEGREE OF MASTER OF SCIENCE  
IN  
AEROSPACE ENGINEERING

NOVEMBER 2024



**Approval of the thesis:**

**ORBIT DETERMINATION WITH ANGULAR MEASUREMENTS BY  
USING ARTIFICIAL NEURAL NETWORK.**

submitted by **KÜRŞAT YENİDOĞAN** in partial fulfilment of the requirements for  
the degree of **Master of Science in Aerospace Engineering, University of Turkish  
Aeronautical Association** by,

**Assoc. Prof. Dr. Adnan GÜZEL**  
**Dean, The School of Graduate Studies, UTAA**

**Prof. Dr. Uğur Murat LELOĞLU**  
Head of The Department, Astronautical  
Engineering, UTAA

**Prof. Dr. Tahsin Çağrı ŞİŞMAN**  
Supervisor, Astronautical  
Engineering, UTAA

**Examining Committee Members:**

**Prof. Dr. Bayram TEKİN**  
Physics Department, METU

**Assoc. Prof. Dr. Mecit YAMAN**  
Aerospace Engineering,  
University of Turkish Aeronautical Association

**Asst. Prof. Dr. Aytun ONAY**  
Software Engineering,  
University of Turkish Aeronautical Association

**Asst. Prof. Dr. Esra KARAOĞLU**  
Department Of Mathematics,  
Çankaya University

**Date:** 15 November 2024

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

**Kürşat YENİDOĞAN**



## ABSTRACT

### ORBIT DETERMINATION WITH ANGULAR MEASUREMENTS BY USING ARTIFICIAL NEURAL NETWORK

YENİDOĞAN, Kürşat

M.S., Department of Aerospace Engineering

Supervisor: Prof. Dr. Tahsin Çağrı ŞİŞMAN

NOVEMBER 2024, 148 pages

The main focus of this thesis is to formulate and implement deep learning based (DL) approaches for initial orbit determination (IOD) using angle-only measurements, with the specific aim of overcoming the limitations faced by most of the classical IOD techniques. DL scientists, for the most case smoothing rely on classical algorithms that are significantly dependent on a number of cycles of acquired data. Most of the time these inputs are likely to be noisy or accompanied by limited datasets. To eliminate these challenges, I aim to use interest artificial neural networks (ANN) into the IOD context in order to enhance the required degree accuracy with robustness in varying observational conditions. Such an explanation introduces the research area where work begins with complete problem description and generating synthetic data sets that closely mimic realistic settings starting from geocentric equatorial reference frames to topocentric horizon reference frame. To make the case properly, I then ensure the accuracy of the data undergoing training by conducting pre-preparations such as normalizing or partitioning data. At the phase of model building, I use artificial neural networks adjusting hyper parameters counterparts and tuning different possible activation functions to cut across all possibilities of deep networks.

The best model I have obtained in my thesis known as deep learning based IOD (DLB-IOD) enhances a number of key orbital parameters computed in the traditional manner further, in particular, this model reduces eccentricity estimation errors by more than 20 percent as compared to ordinary IOD techniques, it regularly estimates semi-major axis with errors lower than 1 percent, and the model is still able to make acceptable inclination estimates even when observation conditions are not ideal. In order to verify these results and my approach I have carried out inherent statistical and several error metric analyses in addition to comparing results obtained with classical IOD methods. This comparative assessment of the results proves that the DLB-IOD method outperforms classical methods in terms of accuracy and stability in equal measure but adds enhancement in enhanced sensitivity to noise while increasing the computational efficiency of the method. Machine learning based models clearly demonstrate the objective of approach as they were ascertained to be effective in accomplishing studies on astrodynamics indicating the possibility that in the near future orbit determination processes will by and large be more flexible, able to scale up to large levels and be able to accommodate changes in mission parameters. To summarize, this work offers a thorough rationale for the usefulness of combining deep learning and IOD methods, suggests future research directions, such as the ability to incorporate real-time data acquisition and processing, as well as further considerations on more complicated orbit configurations and the application of these techniques to satellite mission support and shift in space domain awareness.

**Keywords:** Initial Orbit Determination (IOD), Orbit Determination (OD), Artificial Neural Networks

## ÖZ

### YAPAY SİNİR AĞLARI KULLANARAK AÇISAL ÖLÇÜMLERLE YÖRÜNGE BELİRLEME

YENİDOĞAN, Kürşat

Yüksek Lisans, Bölümü

Tez Yöneticisi: Prof. Dr. Tahsin Çağrı ŞİŞMAN

KASIM 2024, 148 sayfa

Bu tezde, sadece açı ölçümlerine dayalı ilk yörünge belirleme (Initial Orbit Determination, IOD) için derin öğrenme (Deep Learning, DL) tabanlı yöntemlerin geliştirilmesi ve uygulanmasına odaklanıyorum; temel amacım, genellikle klasik IOD teknikleri ile ilişkilendirilen kısıtları aşmaktır. Geleneksel yöntemler sıklıkla doğru, yinelemeli gözlem süreçlerine ve çoklu ölçümlere dayanır, ancak bu durum gürültülü veri girişleri veya sınırlı veri mevcudiyeti karşısında önemli ölçüde sekteye uğrayabilir. Bu sorunları gidermek amacıyla, yapay sinir ağlarını (Artificial Neural Networks, ANN) IOD çerçevesine entegre ederek farklı gözlemsel koşullar altında hem doğruluğu hem de dayanıklılığı artırmayı hedefliyorum. Çalışmam, kapsamlı bir problem tanımıyla başlayıp, gerçekçi senaryoları taklit etmek üzere sentetik veri kümeleri oluşturmaya uzanıyor; bu kapsamda, standart jeomerkezli ekvatorial referans çerçevelerden daha karmaşık topomerkezli sistemlere kadar geniş bir yörünge konfigürasyonunu kapsayan durumlar ele alınıyor. Normalizasyon ve bölümlendirme gibi titiz veri ön işleme teknikleri kullanarak, öğrenme modellerine sağlanan verilerinin iyi yapılandırılmış ve sonraki eğitim süreçleri için en uygun hale getirilmesini sağlıyorum. Model geliştirme aşamasında, çeşitli ANN mimarileri üzerinde denemeler yapıyor, hiper parametre ayarlamalarını gerçekleştiriyor ve farklı aktivasyon fonksiyonlarını değerlendirerek en etkili yapılandırmayı

belirliyorum. Ortaya çıkan derin öğrenme tabanlı IOD (DLB-IOD) modeli, bir dizi kritik yörünge parametresinde kayda değer iyileşmeler sergiliyor: geleneksel IOD yöntemleriyle karşılaştırıldığında eksantriklik tahmin hatalarında yüzde 20'yi aşan bir azalma sağlıyor, yarı büyük eksen tahminlerinde ortalama hata yüzde 1'in altına iniyor ve zorlu gözlemsel koşullarda dahi tutarlı eğiklik tahminleri sunabiliyor. Bu sonuçları doğrulamak için kapsamlı istatistiksel analizler, çeşitli hata metrikleri kullanıyor ve bulguları klasik IOD yaklaşımlarıyla karşılaştırıyorum. Bu karşılaştırmalı değerlendirme, DLB-IOD modelinin yalnızca doğruluk ve kararlılık açısından geleneksel teknikleri geride bırakmakla kalmayıp, aynı zamanda gürültüye karşı daha yüksek bir dayanıklılık ve gelişmiş hesaplama verimliliği sergilediğini onaylıyor. Bu bulgular, yapay zekâ temelli stratejilerin astrodinamik alanındaki dönüştürücü potansiyelini vurgulayarak, gelecekte yörünge belirleme süreçlerinin daha uyarlanabilir, ölçeklenebilir ve değişen görev parametrelerine daha duyarlı olabileceğine işaret ediyor. Sonuç olarak, bu tez DL tabanlı yöntemlerin IOD uygulamalarına entegrasyonunun önemli avantajlarını ortaya koyuyor ve ilerleyen araştırmalarda gerçek zamanlı veri işleme, daha karmaşık yörünge konfigürasyonlarına yönelme ve bu yaklaşımların uydu görev planlaması ile uzay durumsal farkındalığını güçlendirmek üzere daha geniş bir yelpazede uygulanması için olası yollar sunuyor.

Anahtar Kelimeler: İlk Yörünge Belirleme, Yörünge Belirleme, Yapay Sinir Ağları

I dedicate this thesis to my grandmother who had died towards the end of this journey which I undertook in a painstaking manner. The memory of her loss overshadowed my last touch of work and made it immensely difficult and emotionally tasking for me. I can only wish that she is in eternal slumber now, devoid of all the worries and stress that life has to offer.

I would also like to thank my brothers and sisters as well as my parents for their help and support. Their support whether emotional or material was crucial during these tough times as it helped me to pull through and finish the work. They stand behind me, providing encouragement and motivation to strive forward.

## ACKNOWLEDGMENTS

Foremost, I would like to thank my supervisor Prof. Dr. Tahsin Çağrı Şişman for sharing with me his best practices in consideration of the development and perfection of this thesis. His professionalism and attentive guidance enabled to determine the boundaries and level of this research.

Of course, I would not be able to thank Assoc. Prof. Dr. Mecit Yaman for assisting me in expanding my understanding of many other aspects of artificial intelligence. In fact, his constructive sessions did indeed boost my ability to look at the ideas surrounding the topics in AI from other angles.

Also, many thanks to Asst. Prof. Dr. Aytun Onay for helping me with technical issues such as how to evaluate and improve the models. These contributions were also important for the work within the method of models building and the obtained models' strength.

Finally, I would like to express my gratitude to Şahin Ulaş Köprülü who is a wizard in initial orbit determination and who greatly 'opened' me the doors to the wider world of orbital mechanics making me able to work more on this topic.

Lastly, I take full responsibility for noting the contribution made by ChatGPT, which was actively used first in the outlining stage and later during the thesis coding. I found it to be very helpful in resolving doubts with regard to text, cutting down on spelling mistakes, as well as in extensive literature reviews. The technological help provided here worked in tandem with the academic inputs of the aforementioned people and contributed to the further improvement of the consistency and quality of this work.

## LIST OF ABBREVIATIONS

|                |                                                         |
|----------------|---------------------------------------------------------|
| AI             | Artificial Intelligence                                 |
| ANN            | Artificial Neural Networks                              |
| AoP            | Argument of Perigee                                     |
| CR3B           | Circular Restricted Three-Body                          |
| Dec            | Geocentric Declination                                  |
| $\text{Dec}_t$ | Topocentric Declination                                 |
| CNN            | Convolutional Neural Network                            |
| D-IOD          | Direct Initial Orbit Determination                      |
| DL             | Deep Learning                                           |
| DLB-IOD        | Deep Learning Based Initial Orbit Determination         |
| ECI            | Earth-Centered Inertial                                 |
| ECEF           | Earth-Centered Earth-Fixed Coordinate System            |
| EKF            | Extended Kalman Filter                                  |
| GPS            | Global Positioning System                               |
| GNSS           | Global Navigation Satellite System                      |
| GEO            | Geostationary Orbit                                     |
| IOD            | Initial Orbit Determination                             |
| IODA           | Initial Orbit Determination via Artificial Intelligence |
| LOS            | Line of Sight                                           |
| LSTM           | Long Short-Term Memory                                  |
| LEO            | Low Earth Orbit                                         |
| MEO            | Medium Earth Orbit                                      |
| MAE            | Mean Absolute Error                                     |
| MSE            | Mean Squared Error                                      |

|        |                                      |
|--------|--------------------------------------|
| MLE    | Maximum Likelihood Estimator         |
| MLP    | Multi-Layer Perceptron               |
| PCA    | Principal Component Analysis         |
| PIOD   | Physics-Informed Orbit Determination |
| RAAN   | Right Ascension of Ascending Node    |
| $Ra_t$ | Topocentric Right Ascension          |
| RBFNN  | Radial Basis Function Neural Network |
| RNN    | Recurrent Neural Networks            |
| ReLU   | Rectified Linear Unit                |
| RMSE   | Root Mean Squared Error              |
| RSSE   | Root Sum Squared Error               |
| SMA    | Semi-Major Axis                      |
| sma    | Semi-Major Axis                      |
| SSE    | Sum Squared Error                    |
| SVM    | Support Vector Machines              |
| SGD    | Stochastic Gradient Descent          |
| VSA    | Very Short Arcs                      |
| Std    | Standard Deviation                   |
| Dec    | Geocentric Declination               |
| Ecc    | Eccentricity                         |
| LST    | Local Sidereal Time                  |
| JD     | Julian Day                           |



## CHAPTER 1

### INTRODUCTION

#### 1.1 Literature review

The study by Ferreira et al, (2024) presents an effective Maximum Likelihood Estimator (MLE)-based extension of the classical trilateration to initial orbit determination (IOD) in Low Earth orbit. Classic trilateration depends on three radars measuring range and range rate. The proposed algorithm utilizes the new MIMO radar techniques which enable measuring several range, angle and Doppler-shift from each radar position in a particular time stamp. The amount of information contained in these expanding datasets richer than traditional datasets to quantify potential contains more relevant information to support decision making allows the MLE-based approach to achieve trilateration-based accuracy in terms of precision with the same numbers of measurements taken. More importantly, it narrows apart the boundaries of the extent of accuracy that can be achieved as more measurements are made. This improvement holds for lower as well as for higher noise levels and under various noise models, Gaussian being the most common. Further, the MLE based techniques are not dependent on integrating ordinary differential equations or propagating a reference path as is the case with the other modes of initial orientation determination. Therefore, redundancy and the amount of computing time involved in conventional satellite systems is minimized. In this respect, the work demonstrates that with appropriate formulation of the IOD problem and patience of the techniques, better estimates of the state vectors in terms of position and velocity items than what traditional methods provide can be achieved through improvements in radar technology.

(Agostinelli, 2024). The research underscores that the rising number of collisions

and the resulting risk posed by the increasing volume of space debris has turned into one of the essential issues urging the need for situational awareness in outer space in managing and mitigating such risks. In this case, it elucidates the purpose of Initial Orbit Determination (IOD) and proceeds to discuss the potential of an AI aided IOD method for short observation periods. The research notes that in many cases the observations that are available can be quite short, rendering precise orbit determination rather difficult. To remedy this problem, the paper presents a new methods which they called Initial Orbit Determination via Artificial Intelligence (IODA). IODA is effective and accurate orbit determination method based on artificial intelligence and metaheuristic characteristics and is intended for Very Short Arcs (VSAs) orbit determination. As Agostinelli indicates, The developed model if compared against the target state eventually approximates a residual state square error RSSE of position under 1 km and velocity under 0.01 km/s in 89% of the cases.

(FERREIRA, 2024). A technique is developed in this work utilizing three monostatic radars along with MIMO technology to acquire more measurements such as a range, angles, and Doppler shifts. FERREIRA clarifies that by structuring this method in the orbit determination form, it allows the orbit to be determined via MLE in a robust and computationally efficient manner, assuming that the satellite has been observed enough times. In the conclusions, FERREIRA argues through numerical experiments that the exact method provides the same degree of accuracy as classical trilateration but comes out to be more accurate as the number of measurements increases.

(Gong, 2024). The vision-enhancing challenges arising in orbit determination are the focus of this paper. According to Gong, it is essential to capture the nonlinear characteristics of orbit dynamics in an effective manner, and investigates scenarios when only one space-based passive sensor is on the scene without any translational maneuvers. To solve angles only problem for very short arcs, a new concept in the orbit determination area has been devised on the basis of RBFNN, which stands for radial basis function neural network. In this paper, a rigid body motion estimation model is developed that appears to involve the concept of RBFNN. In particular, RBFNN is supposed to learn to associate measurement LOS to relative orbit. By the way, Gong's method has been validated by numerical simulations in Low-Earth Orbit LEO scenarios. The findings were seminal as this new method achieved a record of

only 0.08% average error rate while measuring conclusively that 24 seconds was spent with three measured LOS angle. It was further determined that the RBFNN-based approach possessed better noise robustness and lower generalization error. Also it was more productive than BPNN-based technique. Finally, Gong offers an RBFNN, which allows for quick and precise determination of orbital positions relative to one another, in a situation when measurement means only angles basis in fragments and suggests more angles and longer observing time results in better performance.

The approach put forth by Badura et al. (2024) for IOD in cislunar regions with a (ML) assistance is worth examining because three body dynamics brings classical earthbound techniques to their knees. For the authors, a combination of the DIRSIG<sup>TM</sup>s engine and the CR3BP dynamics is used to simulate the high-fidelity light curves and model satellite brightness signatures as a function of orbit family, satellite shape, optical properties and geometry of viewing position whether from the L1 Earth-Moon point or the lunar surface. The resultant datasets display a completely different behaviour in the relationship between visual magnitude (vMag) and time's progression of the angle between the sun and the satellite which in turn allowed an MLP classifier to be able to distinguish between cislunar families of orbits. The values of the accuracy rates recorded by the models for the lunar based and L1 based observations were approximately 93.4 percent and 87.5 percent respectively and combining the two data sets encourages further enhancement of the observations. Thus the results indicate that it is possible to use ML to focus on the highly differentiated electro-optical signatures to restrict the IOD volumetric shape in anticipation of other tasks in the area of space regulation control.

Badura et al. (2024) demonstrate an approach aimed at increasing the stability and the accuracy of training in the case of Physics Informed Neural Networks (PINNs), for the case of cislunar orbit determination. They insert the law involving the nonlinear dynamics of the so-called Circular Restricted Three-Body (CR3B) problem into the loss function of neural networks aiming to determine a satellite's trajectory using only the line-of-sight observations. One issue is, however, that when randomly initialized, such networks have a tendency to converge to either dynamically unstable or outright wrong solutions. The main point is to use transfer learning: one learns the PINN to a known 'initialization' trajectory before pulling the learning of the PINN

more broadly to observed measurements. This transfer learning step allows the guidance of the network's parameter initialization, thus aiding to avoid convergence into gravitationally unstable PINNs and thus improving training yields. The experiments demonstrate that because of transfer learning the rate of PINNs providing useful orbit solutions increased by about four times and after also achieved two orders higher precision in position error than when trained from zero. This evidence suggests that the integration of transfer learning and other ML-base classifiers could be potential substitution for manual assistance in orbit determination systems for cislunar space.

Prol et al. (2024) propose a simulation that is appropriate for LEO-PNT system for the purpose of improving the efficiency of the PPP algorithm. In the research study, it is demonstrated that a correctly designed LEO constellation (approximately four hundred satellites in three orbit planes at an altitude of 800 kilometers) can give a considerable global coverage with constant access. Compared to classical GNSS, the performance of the proposed system is significantly improved as much as 97% considering signal errors and great improvement to the PPP accuracy of as much as 53% when used with the GPS system. The work provides reference parameters, data distributions and modeling criteria for the future improvement of the LEO-PNT system. Areas believed to deserve more work include improving the parameters for beamforming, optimizing the concepts for the scaling of the beamforming for smaller satellites, Improving the signal to noise ratio relations.

Zhang et al. (2024) propose the distributed approach to solve problems related to Angle Only Orbit Determination (on a multiple CubeSats) for non-cooperative spacecrafts which has been achieved using the factor graphs. Using the factor graph concept, the authors describe the new algorithms developed – the DFBP and the DIBP algorithms which are based on Bayesian filtering principles, and adapted to sub-parent and fully distributed spacecraft cluster architectures. Their methodology enhances estimation accuracy and robustness, and computational easiness of the algorithms while solving under non-linear conditions. For the style of sub-parent architecture, the DFBP method has been able to collect measurements from deputy spacecrafts and nullifies some of the discrepancies in ASR with that from a centralized EKF and even provides a coverage in node failures. For the fully distributed encompasses, the DIBP algorithm permits the scanning lateral's to circulate the Gaussian messages to

the immediate encircling one, this would generate rapid circulation towards accuracy synonymous to that of central EKF. It can be noted that in comparison to the previously existing iterative, distributed consensus filtering methods, the DFBP and DIBP have increased the reliability, enhanced the accuracy and have reduced some iteration cycles. Thus under ideal circumstances, this filtering algorithm based on factor graphs could potentially provide favourable conditions for centralised and optimal resolution in the dynamics of cluster spacecraft orbit resolving inclusively.

Gong et al. (2023) focus on the DNN based approach to the angles-only relative orbit determination problem in angles-only tracking range measurements are feasible. This has been the case for space target non/cooperative case studies which have only made use of the nonlinear methods that have been very advanced and have called for long duration of the observation arcing. Here, however, the authors assist in solving the problem for non-linearities bounding a more efficient and costless method that is faster and promises better results. Their task is to build and train a neural network in DNN format for the job of taking orbital state vectors about a target as an input and fine tuning the absolute measurements of bearing angles about screens as an output. To achieve their aim, they first generate a training data set incorporating the non-linear characteristics of the relative motion, as well as the line of sight angles. Then the designing of the DNN framework follows such as, the network structure, encapsulation and preprocessing of data and algorithms for training. The system allows simulating the application so as to test the accuracy and robustness of the application within the scope geostationary orbit. The analysis and outcomes of the functional simulations provide the conclusion that the strategy installed contributed to the improvement of estimation accuracy of motion of the body under observation and enables one to see desired positions – more detailed reconstruction in relative coordinates.

Zeqaj (2023) proposes the advance of a computer onboard for orbit determination aimed at spacecraft navigational autonomy under the DeepNav project. The initiative seeks to respond to the requirements of cutting down the Command and Control Center participation in deep space exploration through low cost micro and nano satellites missions. The advancement of the satellite that integrates artificial intelligence algorithms, optical images processing, and an Extended Kalman Filter (EKF) system

is hoped to achieve the aims of autonomous spacecraft navigation around low gravitation targets like asteroids. The author emphasizes on the dynamic modeling of a spacecraft orbiting a target body with non-dragging motion in the presence of realistic gravitational and non-gravitational perturbing forces, and how the Dynamics-Informed EKF is used to process artificial intelligence measurements with a priori dynamics. The resultant is a design of a spacecraft onboard computer, developed in a microcontroller system and AI systems capable of human interfacing. Basic simulations show that the system will effectively recreate the motion of a spacecraft from very realistic noisy synthetic data. Next step will be intermating the AI segmentation with the EKF orbit segment in a similar chassis and carrying out demonstration tests. Finally, Zeqaj contributions is more making navigational solutions that are cost effective, convenient, and more autonomous when going deep into space.

Scorsoglio et al. (2023) have proposed a new approach of physics-informed orbit determination method (PIOD) which is based on the orbit determination least square methods but integrates machine learning. Specifically, the PIOD approach, which employs the learning concept of an Extreme Learning Machine (ELM) neural network, enforces the governing differential equations of orbital motion by embedding them into the loss function. Thus, it is feasible that the orbital motion generated by the model is computationally valid. The approach is comprehensive, allowing for both the classic KEP orbiting maneuver and orbital movement that does not follow the KEP movement, while also taking into account different types of ranges: range/range-rate or angular measurements. In synthetic datasets, the PIOD procedure achieves performances that are in the order of traditional batch least squares methodologies but without the need of operational guesses and the need for numerical integration and therefore possible errors. By altering the input weights and biases randomly, PIOD makes the estimation of the complete trajectory of the system possible, while eliminating the need for operational guesses. These benefits will certainly enhance the ability of physics informed neural network methods to be used effectively in orbit determination problems, including those set in the increasingly important cislunar space.

Chng et al. (2023) propose a direct initial orbit determination (D-IOD) that eliminates the use of constructing line-of-sight vectors from optical measurements. Rather, D-

IOD adopts a trend reversal approach in which streak images are fitted into orbital parameters by devising an image matching algorithm which employs a non-linear least squares remain to the orbit candidate generated images and observed images. The strategy employs gradients and makes use of computer vision optimization methods so as to perform the task at hand making use of the necessary pixel data and eliminating errors made during the extraction of 'LOS'. When evaluated on simulated, and real datasets that are known to be difficult, D-IOD has proven to be robust and accurate due to the fact that it was able to improve orbital estimates with poor predictions. The suggested approach improves the two-step IOD method in use and provides a viable opportunity for the future work and real-life applications in orbit determination.

Tan et al. (2022) present a fully real-time strategy of precise orbit determination (POD) for five GNSS constellations (GPS, GLONASS, Galileo, BDS as well as QZSS) making use of an hourly ultrarapid orbit update prediction. After a detailed discussion of identifying arcs as consisting of three consecutive twenty-four-hour normal equations and the introduction of a refined solar radiation pressure modelling approach (cuboid model combined with ECOM1), the authors are able to claim orbits accuracy to the levels measured in centimeters. For the case of GPS, Galileo and also BDS having IGSOs, the mean one-dimensional RMS orbital differences accumulated against GFZ final multi GNSS products are respectively 3.7 cm, 5.7 cm and 10.2 cm. Tests of discontinuities in hourly orbit boundary have shown similar trends in accuracy and laser ranging satellite validation yields mean RMS values close to the region 4 to 5 cm. The results are however indicated as providing a relative good level of accuracy for those who will be interested in POD in real time but the authors however acknowledge other force model mismatches which requires further elaboration for BDS and QZSS.

(Pastor, 2021). This paper examines various methods that allow for the orbital determination to be performed in order to take care of the space object. In this paper, however, it is sought to be accomplished IOD from a number of optical observations without being required to many considerations such as the measurement noise present. This method is casted as a boundary value problem with the aim of obtaining a pair of ranges that will nearly minimize the residuals of the observations against the all colonies states A primary concern of Pastor is concerning the is focusing on

the improved toroidal method of orbital perturbation describing classical dynamics. To describe the solution, Pastor employs the Double r-iteration Lambert technique where the angle-only IOD problem is treated as a boundary value problem and uses some noise modelling techniques that resolve a complete state vector along with its covariance.

Shang and associates in (2018) put forward a new approach for initial orbit determination (IOD) which capitalises on very short arc (VSA) radar information from two observational sites. Using its closed-formed solutions originating from range, radial velocity measurements in the form of range, and radial acceleration, the presented approach details the initial orbit formulation. An important aspect of this presentation is the geometric dilution of precision (GDOP) metric to assess and direct the data collection arc which eventually enhances the IOD. Theoretic reasoning in conjunction with Monte Carlo simulations emphasize that several seconds of VSA data can achieve a satellite orbit with a range of a few kilometers and a velocity of several meters per second for all low-Earth-objects if the data is precise. It is tacitly understood that the aforementioned method would be a suitable option for IOD in conditions where no accentuated changes in the position are expected and but it would be ideal for further research to look into spinning targets and cyclic perturbations a terrestrial equator whenever the phenomena occurs.

In a study by Lovell et al. (2018), angles-only initial orbit determination strategies for space-based observers are considered with an emphasis on techniques based on inertial dynamics as opposed to those using relative orbital dynamics. These methods traditionally operate under the constraint that the observer and the target space vehicle are nearly co-located and have nearly co-synchronous orbits, which the authors show is not necessary. This study builds on the previous studies which made use of second-order closed form relative motion solution by adding a third order solution to include more nonlinearity in the relative dynamics. Numerical simulations indicate that both procedures can resolve the orbit of the tracked object using line-of-sight methods, but the third order accuracy of the determined orbit is higher due to better modeling of the nonlinear motion. The authors also examine dependence on the measurement errors and sampling rate, which modeling of the higher order appears to reduce the weak observability problems. In the next stage of their research, the emphasis will be

placed on calculating the observer orbit errors, more sophisticated dynamic models will be included, as well the issues of computational cost will be addressed.

Lee et al. (2018) present a methodology for unknown low Earth orbiting satellites to determine their initial orbit without much observation. Using only one observation millimeter station, they are able to virtually create Doppler shifts in order to use machine learning. As such, with a essence of utilizing only one station, they are able to carry out the IOD for two additional imaginary stations since it is not possible to carry out tracking for three stations. The work explains ML model trainings by employing a month's worth of Doppler range shifts to track the satellites' positions. The authors argue that with minimal Doppler Range Rate settings and six variances created by the imaginary stations, policies for all three data readings would be mechanized to function. Hence ML has potentialability in aiding the situational impact awareness by utilizing its tracking function without rigorous observational input. Subsequents actions would develop the algorithm, address the issue of Doppler shift and measure the signals strays.

Global positioning techniques in Earth satellites or Cislunar volume ( the space region between the Earth and Moon ) is a hot research idea in spacecraft navigation scholars. Among the popular scholars who are working in this respect are Ojeda Romero et al (2022), for instance, developed a system called MCCLOD, which is a quick topics system in clustering magnitudes and coordinate reference frames. The other system with similar working systems and developed developmental frames include Multi-Objective Optimization, which is used in evolutionary algorithms. These researchers used clustering in developing coordinates that validate the computational techniques which are statistically analyzed to yield better effective results than all other previously used positional systems. This made these systems provide better accuracy and at the same time low errors predictive variables which in turn made return to measurement timing and classification errors to become statistically insignificant.

In this work Ardaens and Gaias (2017) describe a set of experimental efforts to determine the relative orbital elements for satellites placed in low-earth orbits using only the angles to the satellites as the inputs. According to them, during two months of experiment of AVANTI effort, AVANTI endeavor was able to determine relative or-

bit parameters with even 50 meters apart, even when the satellites were faced with challenging situations where the cameras were eclipsed or interfered with by other effects allowing for no clearer view. This is a marked difference from the previous demonstrations such as ARGON which had rather convenient orbital positions and visualization as dawn-dusk shades. Instead, AVANTI was confronted with quite the opposite, stronger perturbations and fairly less observation opportunities. The overcoming of such difficulties were not easy but the authors explain in detail how improving target detection techniques caused the dynamical model perturbations to become less of an issue as the spacecraft got closer to its target which caused orbital parameters to be more precise than before. Initial orbit estimation along with rugged outlier omission were challenges in their approach and External TLE (two line elements) data was employed where necessary but still it was no surprise that precision enhancement was possible provided that improvements were done regarding modeling that would in turn incorporate the attitude of the chaser in the drag model and data processes. Ultimately, the results of the AVANTI mission highlight the potential of angles-only navigation even in deteriorated conditions which provides a useful addition to the earlier works and a means towards viability of autonomous close-proximity engagements in the low Earth orbit.

Presently, Restricting to angle measurements only to estimate satellite orbits has been one such area of concern in astrodynamics and SSA(space situational awareness). In 2012, Schaeferkoetter investigated the possibility of improving orbit determination processes that rely on only angular measurements. He noticed that conventional approaches were not accurate due to data shortage and basic assumptions. His new approaches relied on complex mathematical models and algorithms, and thereby, improved the accuracy considerably. For example, he used push a little further progression attitude correction to make final ascertainment of the parameters. His work identified positive and negative aspects of the existing system and provided insight into subsequent developments. The study underscored the fact that with the help of modern computational techniques, satellite orbit determinations that would otherwise be inaccurate can be performed accurately and reliably(Schaeferkoetter, 2012).

As evidenced by the works of Milani et al., (2003), finding an orbit solution may be challenging in the cases when the observational data allowance is mer to a few

arcs. Their own suggest that all possible orbit solutions are contained in a constrained “admissible region” rather than a unique orbit. Instead, this method allows predicting the future position of all virtual asteroids that are created by sampling the region. What this method achieves is not single definite orbit, instead, it offers a good starting position which is even bettered as more observations are made.

Rim et al. (1996) carry out an in-depth simulation and analysis of covariances to examine position obtaining method with simultaneous connectivity the use of Global Positioning System (GPS) data for low earth orbiting satellites, Taking the case of the TOPEX mission. They use both dynamic and kinematic methods of determining the orbit in their research work. The dynamic method depends on force models which can be affected when there are imperfections in said models . On the other hand, the kinematic method determines orbital positions by smoothing pseudo range measurements with carrier phase data while there is no need to use a force model. Nevertheless, the accuracy of the kinematic modeling method diverges ways in which the instruments used like the GPS side real interference multipath and noise. To overcome the them, the authors recommend a hybrid uniformity model that incorporated process noise so as to obtain appropriate weighting with both kinematic and dynamic models. This they do through repeated Monte Carlo simulations where they optimize their error models such that the error models used will reflect realistic measurement and observation. According to the results, this modeling approach which is refined by covariance and sensitivity analyses can properly describe how various errors including GPS orbit errors affect the radial component of the TOPEX’s orbit. It can be measured how shifting and turning inside this modeling structure for example alters reliability and accuracy of POD under realistic operational conditions.

## **1.2 Motivation for the Research**

In today’s world, the task of determining the orbit of a satellite has emerged as an important sphere of research activity owing to its significance in exploration and operational tasks of spacecraft. First orbit determination (IOD) methods are especially needed when observations are taken visually. The initial orbit of a satellite is the basis for all the subsequent tracking and planning of the mission. Consequently, for the

success of the missions of satellites, it is a matter of great significance to increase the accuracy of the methods of IOD.

It is a well-known fact in the literature that multiple observations may be necessary to enhance the accuracy of the data and the orbit determination processes. In order to achieve this, methods such as least squares and Kalman filters are widely used. These methods help to reduce errors and yield better estimates of the orbital parameters of a satellite. Nevertheless, in terms of their effectiveness, the improvement methods discussed above rely heavily on the quality of the data that the IOD methods provide. The classical improvement techniques discussed above work best when the IOD method works most accurately and precisely.

In light of the critical dependence on IOD methods, there is a strong motivation to enhance their accuracy and precision. Enhancing the quality of the initial data may improve the overall effectiveness of the orbit determination processes. This research intends to address this gap by introducing a different IOD method based on deep learning based. In this case, the deep learning based method should be smarter and more accurate than the standard IOD methods. In this way, the intention of this new approach is to deal with the drawbacks of the classical ones by making use of deep learning.

The primary aim of this dissertation work is to assess the performance of the new deep learning based IOD method in comparison to the performance of the classical IOD methods.

The aim of the comparison is to illustrate the efficiency of the AI based method in terms of effectiveness indeed. This situation leads us to the question of interest, how useful would it be to use AI in the process of orbit determination. The expectations of the results are that the deep learning based system is better than all other systems in terms of estimating initial orbits which would in turn result in augmenting the performance of other kinetic data analysis processes.

The objective of the study is to build and improve a more sophisticated deep learning based IOD model which will yield reliable results. The research pursues a goal of demonstrating the superiority of AI applications in the analysis of satellite orbits by

presenting several examples for that purpose. Improving the IOD methods capabilities and efficacy, this study enhances satellite missions planning and operations and addresses the growing needs in space and satellite technologies.





## CHAPTER 2

### CLASSICAL ORBIT DETERMINATION METHODS

#### 2.1 Laplace

To determine the line-of-sight unit vectors, we assume we have the topocentric right ascension  $\alpha$  and declination  $\delta$  of a satellite at three different times. These values can be obtained from observations. Let's define the unit vectors  $\mathbf{L}_1$ ,  $\mathbf{L}_2$  and  $\mathbf{L}_3$  along the line of sight to the satellite at these three observation times:

$$\mathbf{L}_i = \begin{bmatrix} L_{ix} \\ L_{iy} \\ L_{iz} \end{bmatrix} = \begin{bmatrix} \cos \delta_i \cos \alpha_i \\ \cos \delta_i \sin \alpha_i \\ \sin \delta_i \end{bmatrix} \quad i = 1, 2, 3 \quad (2.1)$$

These vectors  $\mathbf{L}_i$  are unit vectors along the slant range vector  $\rho$  from the observation site to satellite. Thus, the position vector  $\mathbf{r}$  of the satellite can be written as:

$$\mathbf{r} = \rho \mathbf{L} + \mathbf{R}_{site} \quad (2.2)$$

where  $\rho$  is the slant range to satellite, and  $\mathbf{R}_{site}$  is the vector from the center of the Earth to the observation site.

Differentiating the position vector equation equation [2.2](#) with respect to time gives:

$$\dot{\mathbf{r}} = \dot{\rho} \mathbf{L} + \rho \dot{\mathbf{L}} + \dot{\mathbf{R}}_{site} \quad (2.3)$$

Differentiating again with respect to time yields:

$$\ddot{\mathbf{r}} = \ddot{\rho}\mathbf{L} + 2\dot{\rho}\dot{\mathbf{L}} + \rho\ddot{\mathbf{L}} + \ddot{\mathbf{R}}_{site} \quad (2.4)$$

From the equations of motion, the acceleration of the satellite is:

$$\ddot{\mathbf{r}} = -\mu \frac{\mathbf{r}}{r^3} \quad (2.5)$$

Substituting this into the differentiated form, we obtain:

$$\ddot{\rho}\mathbf{L} + 2\dot{\rho}\dot{\mathbf{L}} + \rho\ddot{\mathbf{L}} = -\mu \frac{\mathbf{r}}{r^3} - \ddot{\mathbf{R}}_{site} \quad (2.6)$$

Since  $\mathbf{r} = \rho\mathbf{L} + \mathbf{R}_{site}$ , substituting this in and simplifying gives:

$$\ddot{\rho}\mathbf{L} + 2\dot{\rho}\dot{\mathbf{L}} + \rho\ddot{\mathbf{L}} = -\mu \frac{\rho\mathbf{L} + \mathbf{R}_{site}}{r^3} - \ddot{\mathbf{R}}_{site} \quad (2.7)$$

$$\ddot{\rho}\mathbf{L} + 2\dot{\rho}\dot{\mathbf{L}} + \rho \left( \ddot{\mathbf{L}} + \frac{\mu}{r^3} \mathbf{L} \right) = - \left( \ddot{\mathbf{R}}_{site} + \frac{\mu}{r^3} \mathbf{R}_{site} \right) \quad (2.8)$$

Given the values of  $\mathbf{L}$  at three different times  $t_1$ ,  $t_2$  and  $t_3$ , we can numerically differentiate to obtain  $\dot{\mathbf{L}}$  and  $\ddot{\mathbf{L}}$  at the central time  $t_2$ . Using the Lagrange interpolation formula, the line-of-sight vector as a function of time is:

$$\mathbf{L}(t) = \frac{(t-t_2)(t-t_3)}{(t_1-t_2)(t_1-t_3)} \mathbf{L}_1 + \frac{(t-t_1)(t-t_3)}{(t_2-t_1)(t_2-t_3)} \mathbf{L}_2 + \frac{(t-t_1)(t-t_2)}{(t_3-t_1)(t_3-t_2)} \mathbf{L}_3$$

Differentiating this equation twice with respect to time, we get:

$$\dot{\mathbf{L}}(t) = \frac{2t-t_2-t_3}{(t_1-t_2)(t_1-t_3)} \mathbf{L}_1 + \frac{2t-t_1-t_3}{(t_2-t_1)(t_2-t_3)} \mathbf{L}_2 + \frac{2t-t_1-t_2}{(t_3-t_1)(t_3-t_2)} \mathbf{L}_3 \quad (2.9)$$

$$\ddot{\mathbf{L}}(t) = \frac{2}{(t_1-t_2)(t_1-t_3)} \mathbf{L}_1 + \frac{2}{(t_2-t_1)(t_2-t_3)} \mathbf{L}_2 + \frac{2}{(t_3-t_1)(t_3-t_2)} \mathbf{L}_3 \quad (2.10)$$

By setting  $t = t_2$  in these equations, we can obtain numerical values for  $\dot{\mathbf{L}}$  and  $\ddot{\mathbf{L}}$  at the central date. If more than three observations are available, a more accurate value

of  $\dot{\mathbf{L}}$  and  $\ddot{\mathbf{L}}$  may be obtained by fitting higher order polynomials with the Lagrange interpolation formula.

Using the known values of  $\mathbf{L}$ ,  $\dot{\mathbf{L}}$ ,  $\ddot{\mathbf{L}}$ ,  $\mathbf{R}_{site}$ ,  $\dot{\mathbf{R}}_{site}$  and  $\ddot{\mathbf{R}}_{site}$  at  $t_2$  we can solve for  $\rho$  using Cramer's rule. The determinant of the coefficients is given by:

$$D = \begin{vmatrix} L_{ix} & 2L_{ix} & \ddot{L}_{ix} + \frac{\mu}{r^3}L_{ix} \\ L_{iy} & 2L_{iy} & \ddot{L}_{iy} + \frac{\mu}{r^3}L_{iy} \\ L_{iz} & 2L_{iz} & \ddot{L}_{iz} + \frac{\mu}{r^3}L_{iz} \end{vmatrix} \quad (2.11)$$

Simplifying by subtracting  $\mu/r^3$  times the first column from the third column, we get:

$$D = \begin{vmatrix} L_{ix} & 2L_{ix} & \ddot{L}_{ix} \\ L_{iy} & 2L_{iy} & \ddot{L}_{iy} \\ L_{iz} & 2L_{iz} & \ddot{L}_{iz} \end{vmatrix} \quad (2.12)$$

Applying Cramer's rule to solve for  $\rho$ :

$$D\rho = -2 \begin{vmatrix} L_{ix} & \dot{R}_{site_{ix}} \\ L_{iy} & \dot{R}_{site_{iy}} \\ L_{iz} & \dot{R}_{site_{iz}} \end{vmatrix} - 2\frac{\mu}{r^3} \begin{vmatrix} L_{ix} & R_{site_{ix}} \\ L_{iy} & R_{site_{iy}} \\ L_{iz} & R_{site_{iz}} \end{vmatrix} \quad (2.13)$$

Defining the determinants  $D_1$  and  $D_2$ :

$$\rho = -\frac{2D_1}{D} - \frac{2\mu D_2}{r^3 D} \quad (2.14)$$

Since we need to make an initial guess for  $\rho$ ,  $\mathbf{r}_2^2 = (\rho\mathbf{L} + \mathbf{R}_{site})^2$ , using ,we iteratively refine  $\rho$  by solving:

$$r_2^2 = \left(-\frac{2D_1}{D} - \frac{2\mu D_2}{r^3 D}\right)^2 + 2\left(-\frac{2D_1}{D} - \frac{2\mu D_2}{r^3 D}\right) \mathbf{L}_2 \cdot \mathbf{R}_{site_2} + R_{site_2}^2 \quad (2.15)$$

Expanding this and setting the dot product to zero, we get an eighth-degree polynomial:

$$C = \mathbf{L}_2 \cdot \mathbf{R}_{site_2} \quad (2.16)$$

Solving this polynomial for  $\rho$  and iterating ensures that the correct root is found:

$$r_2^8 + \left( \frac{4CD_1}{D} - \frac{4D_1^2}{D^2} - R_{site_2}^2 \right) r_2^6 + \mu \left( \frac{4CD_2}{D} - \frac{8D_1D_2}{D^2} \right) r_2^3 - \frac{4\mu^2 D_2^2}{D^2} = 0 \quad (2.17)$$

Using Cramer's rule again for velocity starting from:

$$D\dot{\rho} = - \begin{vmatrix} L_{ix} & \dot{L}_{ix} & \ddot{R}_{site_{ix}} + \frac{\mu}{r^3} R_{site_{ix}} \\ L_{iy} & \dot{L}_{iy} & \ddot{R}_{site_{iy}} + \frac{\mu}{r^3} R_{site_{iy}} \\ L_{iz} & \dot{L}_{iz} & \ddot{R}_{site_{iz}} + \frac{\mu}{r^3} R_{site_{iz}} \end{vmatrix} \quad (2.18)$$

This determinant splits as:

$$D\dot{\rho} = - \begin{vmatrix} L_{ix} & 2\dot{L}_{ix} & \ddot{R}_{site_{ix}} \\ L_{iy} & 2\dot{L}_{iy} & \ddot{R}_{site_{iy}} \\ L_{iz} & 2\dot{L}_{iz} & \ddot{R}_{site_{iz}} \end{vmatrix} - \frac{\mu}{r^3} \begin{vmatrix} L_{ix} & R_{site_{ix}} & \ddot{L}_{ix} \\ L_{iy} & R_{site_{iy}} & \ddot{L}_{iy} \\ L_{iz} & R_{site_{iz}} & \ddot{L}_{iz} \end{vmatrix} \quad (2.19)$$

Defining the determinants  $D_3$  and  $D_4$ :

$$\dot{\rho} = -\frac{D_3}{D} - \frac{\mu}{r^3} \frac{D_4}{D} \quad (2.20)$$

So you can use the velocity determinants and solve for the middle velocity vector in equation [2.3](#):

$$\dot{\mathbf{r}}_2 = \dot{\rho} \mathbf{L}_2 + \rho \dot{\mathbf{L}}_2 + \dot{\mathbf{R}}_{site_2} \quad (2.21)$$

The outlined method, involving the iterative refinement of  $\rho$  and subsequent determination of the middle velocity vector, ensures precise orbit determination using line-of-sight vectors and observational data.

## 2.2 Gauss

The Gaussian method receives mixed reviews in the astrodynamics community when using only angle data. Opinions range from no concern about the method, as it has performed excellently in interplanetary studies, to doubts about its accuracy in determining satellite orbits, to admiration for its success in an age of limited data. According to Vallado, this method performs best when the angular separation between observations is less than about  $60^\circ$ . It works exceptionally well, especially when the data is separated by  $10^\circ$  or less. For satellites in low Earth orbit, this separation means that observations are made at most five to ten minutes apart. Naturally, you should always analyze the orbits you plan to process before determining the frequency of data collection. The success of the Gaussian method also depends on the technique used to determine the  $f$  and  $g$  series; this issue will be discussed below. In any case, when formulated correctly, the Gaussian method is a moderately robust way of determining the position of a satellite using only angle data. [Vallado]

It is possible to define set of scalar magnitudes, denoted as  $a, b$  and  $c$  for a vector. Since the vectors of three observations lie on the same plane and are linearly dependent, we have the following relationship;

$$a \vec{r}_1 + b \vec{r}_2 + c \vec{r}_3 = 0 \quad (2.22)$$

At this point, a new equation can be established by centering  $\vec{r}_2$  and introducing  $c_1$  and  $c_3$  as coefficients:

$$\vec{r}_2 = c_1 \vec{r}_1 + c_3 \vec{r}_3 \quad (2.23)$$

by crossing  $\vec{r}_2$  with  $\vec{r}_1$  and  $\vec{r}_3$ , we can write;

$$\vec{r}_1 \times \vec{r}_2 = \vec{r}_2 = \vec{r}_1 \times c_1 \vec{r}_1 + \vec{r}_1 \times c_3 \vec{r}_3 \quad (2.24)$$

$$\vec{r}_3 \times \vec{r}_2 = \vec{r}_2 = \vec{r}_3 \times c_1 \vec{r}_1 + \vec{r}_3 \times c_3 \vec{r}_3 \quad (2.25)$$

knowing that  $\vec{r}_1 \times c_1 \vec{r}_1 = 0$  and  $\vec{r}_3 \times c_3 \vec{r}_3 = 0$ , equations must be rearranged;

$$\vec{r}_1 \times \vec{r}_2 = \vec{r}_2 = c_3 \vec{r}_1 \times \vec{r}_3 \quad (2.26)$$

$$\vec{r}_3 \times \vec{r}_2 = \vec{r}_2 = c_1 \vec{r}_3 \times \vec{r}_1 \quad (2.27)$$

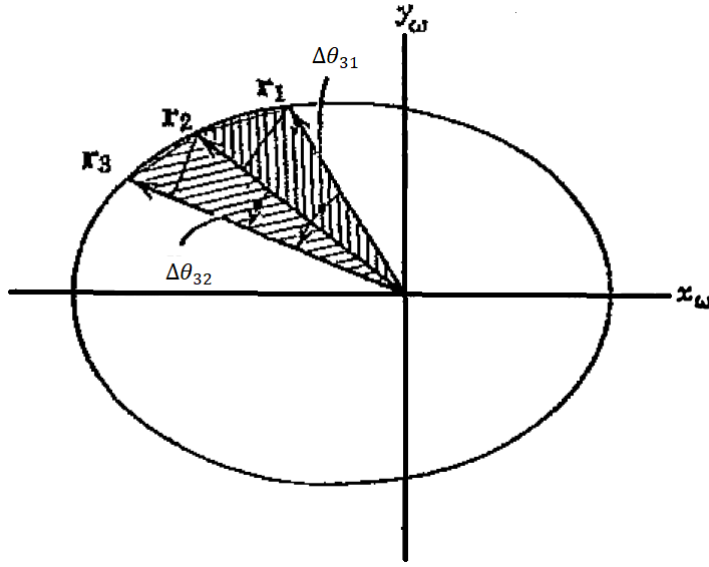


Figure 2.1: Three observation of  $\vec{r}_1, \vec{r}_2$  and  $\vec{r}_3$ [26]

With the aid of [2.1](#), Equations [2.26](#) [2.27](#) can be modified by introducing  $\vec{W}$ , a unit vector normal to the orbit plane, and by evaluating the cross products as follows:

$$\vec{r}_1 \times \vec{r}_2 = r_1 r_2 \sin(\Delta\theta_{12}) \vec{W} = 2A_{12} \vec{W} \quad (2.28)$$

$$\vec{r}_2 \times \vec{r}_3 = r_2 r_3 \sin(\Delta\theta_{23}) \vec{W} = 2A_{23} \vec{W} \quad (2.29)$$

$$\vec{r}_1 \times \vec{r}_3 = r_1 r_3 \sin(\Delta\theta_{13}) \vec{W} = 2A_{13} \vec{W} \quad (2.30)$$

where  $A_{12}$ ,  $A_{23}$  and  $A_{13}$  are the areas of the triangles formed between the respective radius vectors. [2.28](#)

$$2A_{12}\vec{W} = 2c_3A_{13}\vec{W}$$

$$2A_{23}\vec{W} = 2c_1A_{13}\vec{W}$$

$$c_3 = \frac{A_{12}}{A_{13}}, c_1 = \frac{A_{23}}{A_{13}} \quad (2.31)$$

At this point,  $A_{12}$  can be defined as:

$$A_{12} = \frac{1}{2}\vec{W} \cdot (\vec{r}_1 \times \vec{r}_2) \quad (2.32)$$

To advance further in the method, a new formulation is required, involving of the  $f$  and  $g$  series. Therefore, equation [2.32](#) should be reformulated in terms of the  $f$  and  $g$  series. We must first express  $\vec{r}_1$  with respect to  $\vec{r}_2$  and the  $f$  and  $g$  series, and then apply the same to  $\vec{r}_3$ :

$$\vec{r}_1 = f_1\vec{r}_2 + g_1\dot{\vec{r}}_2 \quad (2.33)$$

$$\vec{r}_3 = f_3\vec{r}_2 + g_3\dot{\vec{r}}_2 \quad (2.34)$$

Substituting equation [2.33](#) into equation [2.32](#) yields:

$$\begin{aligned} A_{12} &= \frac{1}{2}\vec{W} \cdot \left[ (f_1\vec{r}_2 + g_1\dot{\vec{r}}_2) \times \vec{r}_2 \right] \\ &= \frac{1}{2}\vec{W} \cdot (f_1\vec{r}_2 \times \vec{r}_2) + \frac{1}{2}\vec{W} \cdot (g_1\dot{\vec{r}}_2 \times \vec{r}_2) \\ &= -\frac{1}{2}hg_1 \end{aligned} \quad (2.35)$$

Since  $\vec{W} \cdot (\vec{r}_1 \times \vec{r}_2) = 0$  and  $f_1\vec{r}_2 \times \vec{r}_2 = 0$ . Similarly,  $A_{23}$  can be expressed as:

$$A_{23} = \frac{1}{2} \vec{W} \cdot \left[ \vec{r}_2 \times \left( f_3 \vec{r}_2 + g_3 \dot{\vec{r}}_2 \right) \right] = \frac{1}{2} h g_3$$

Subsequently,  $A_{13}$  can be defined as follows:

$$\begin{aligned} A_{13} &= \frac{1}{2} \vec{W} \cdot \left[ \left( f_1 \vec{r}_2 + g_1 \dot{\vec{r}}_2 \right) \times \left( f_3 \vec{r}_2 + g_3 \dot{\vec{r}}_2 \right) \right] \\ &= \frac{1}{2} \vec{W} \cdot \left( f_1 \vec{r}_2 \times f_3 \vec{r}_2 + f_1 \vec{r}_2 \times g_3 \dot{\vec{r}}_2 + g_1 \dot{\vec{r}}_2 \times f_3 \vec{r}_2 + g_1 \dot{\vec{r}}_2 \times g_3 \dot{\vec{r}}_2 \right) \\ &= \frac{1}{2} h (f_1 g_3 - f_3 g_1) \end{aligned} \quad (2.36)$$

At this point, referencing equation [2.31](#), we define:

$$c_1 = \frac{A_{23}}{A_{13}} = \frac{\frac{1}{2} h g_3}{\frac{1}{2} h (f_1 g_3 - f_3 g_1)} = \frac{g_3}{f_1 g_3 - f_3 g_1} \quad (2.37)$$

$$c_3 = \frac{A_{12}}{A_{13}} = \frac{-\frac{1}{2} h g_1}{\frac{1}{2} h (f_1 g_3 - f_3 g_1)} = \frac{-g_1}{f_1 g_3 - f_3 g_1} \quad (2.38)$$

Further, the  $f$  and  $g$  series can be explicitly expressed as follows:

$$\tau_3 = t_3 - t_1 \tau_1 = t_1 - t_2 \quad (2.39)$$

$$\tau = \tau_3 - \tau_1 \quad (2.40)$$

then,

$$\begin{aligned} f_1 &\approx 1 - \frac{1}{2} \frac{\mu}{r_2^3} \tau_1^2 & g_1 &\approx \tau_1 - \frac{1}{6} \frac{\mu}{r_2^3} \tau_1^3 \\ f_3 &\approx 1 - \frac{1}{2} \frac{\mu}{r_2^3} \tau_3^2 & g_3 &\approx \tau_3 - \frac{1}{6} \frac{\mu}{r_2^3} \tau_3^3 \end{aligned} \quad (2.41)$$

According to equation 2.41,  $c_1$  and  $c_2$  can be expressed in terms of  $r_2$  which is defined within geocentric frame, and  $\mu$  is the gravitational parameters. The expression  $f_1 g_3 - f_3 g_1$  can be simplified as follows:

$$\begin{aligned} f_1 g_3 - f_3 g_1 &= \left(1 - \frac{1}{2} \frac{\mu}{r_2^3} \tau_1^2\right) \left(\tau_3 - \frac{1}{6} \frac{\mu}{r_2^3} \tau_3^3\right) - \left(1 - \frac{1}{2} \frac{\mu}{r_2^3} \tau_3^2\right) \left(\tau_1 - \frac{1}{6} \frac{\mu}{r_2^3} \tau_1^3\right) \\ &= (\tau_3 - \tau_1) - \frac{1}{6} \frac{\mu}{r_2^3} (\tau_3 - \tau_1)^3 + \frac{1}{12} \frac{\mu^2}{r_2^6} (\tau_1^2 \tau_3^3 - \tau_1^3 \tau_3^2) \\ &= \tau - \frac{1}{6} \frac{\mu}{r_2^3} \tau^3 \end{aligned} \quad (2.42)$$

The coefficients  $c_1$  and  $c_2$  are defined in accordance with equations 2.41 and 2.42. They are given by the following expressions:

$$\begin{aligned} c_1 &= \frac{\tau_3 - \frac{1}{6} \frac{\mu}{r_2^3} \tau_3^3}{\tau - \frac{1}{6} \frac{\mu}{r_2^3} \tau^3} = \frac{\tau_3}{\tau} \left[ 1 + \frac{1}{6} \frac{\mu}{r_2^3} (\tau^2 - \tau_3^2) \right] \\ c_3 &= -\frac{-g_1}{\tau - \frac{1}{6} \frac{\mu}{r_2^3} \tau^3} = -\frac{\tau_1}{\tau} \left[ 1 + \frac{1}{6} \frac{\mu}{r_2^3} (\tau^2 - \tau_1^2) \right] \end{aligned} \quad (2.43)$$

Finally, the coefficients  $c_1$  and  $c_3$  can be determined with a single variable,  $r_2$  which is the component of the of observed distance in the geocentric frame at time  $t_2$  together with the gravitational parameter  $\mu$ . At this point,  $r_2$  emerges as the only variable required to calculate  $c_1$  and  $c_3$ . By adapting and applying equation 2.22, we first resolve  $\rho_2$ , and subsequently, compute  $c_1$  and  $c_3$ . Equation 2.22 as follows;

$$\begin{aligned} c_1 (\vec{\rho}_1 + \vec{R}_{site_1}) + c_2 (\vec{\rho}_2 + \vec{R}_{site_2}) + c_3 (\vec{\rho}_3 + \vec{R}_{site_3}) &= 0 \\ c_1 \vec{\rho}_1 + c_2 \vec{\rho}_2 + c_3 \vec{\rho}_3 &= -c_1 \vec{R}_{site_1} - c_2 \vec{R}_{site_2} - c_3 \vec{R}_{site_3} \end{aligned} \quad (2.44)$$

Now, using the relationship  $\vec{r} = \rho \vec{L} + \vec{R}_{site}$ , where  $\vec{L}$  is the line of sight vector, equation 2.44 can be written in matrix form as follows:

$$\vec{L} \begin{bmatrix} c_1 \rho_1 \\ c_2 \rho_2 \\ c_3 \rho_3 \end{bmatrix} = \begin{bmatrix} \vec{R}_{site_1} & \vec{R}_{site_2} & \vec{R}_{site_3} \end{bmatrix} \begin{bmatrix} -c_1 \\ -c_2 \\ -c_3 \end{bmatrix} \quad (2.45)$$

To calculate  $\rho_2$ , therefore, the left side of equation 2.45 must be isolated. By multiplying  $\vec{L}^{-1}$ :

$$\begin{aligned} \vec{L}^{-1} \vec{L} \begin{bmatrix} c_1 \rho_1 \\ c_2 \rho_2 \\ c_3 \rho_3 \end{bmatrix} &= \vec{L}^{-1} \begin{bmatrix} \vec{R}_{site_1} & \vec{R}_{site_2} & \vec{R}_{site_3} \end{bmatrix} \begin{bmatrix} -c_1 \\ -c_2 \\ -c_3 \end{bmatrix} \\ \begin{bmatrix} c_1 \rho_1 \\ c_2 \rho_2 \\ c_3 \rho_3 \end{bmatrix} &= \vec{L}^{-1} \begin{bmatrix} \vec{R}_{site_1} & \vec{R}_{site_2} & \vec{R}_{site_3} \end{bmatrix} \begin{bmatrix} -c_1 \\ -c_2 \\ -c_3 \end{bmatrix} \end{aligned} \quad (2.46)$$

At this point, we can examine the slant-range  $\rho_2$  equation from 2.46 and redefine  $\vec{L}^{-1} \mathbf{R}_{site}$  as matrix  $\mathbf{M}$ , and knowing  $c_2 = 1$ . The result of equation 2.46 for  $\rho_2$  follows as:

$$\begin{bmatrix} c_1 \rho_1 \\ c_2 \rho_2 \\ c_3 \rho_3 \end{bmatrix} = \mathbf{M} \begin{bmatrix} -c_1 \\ -c_2 \\ -c_3 \end{bmatrix} \quad (2.47)$$

$$\rho_2 = -(M_{21}c_1 + M_{22} + M_{23}c_3) \quad (2.48)$$

where

$$\begin{aligned} \rho_2 &= -(M_{21}c_1 + M_{22} + M_{23}c_3) \\ &= -\left(M_{21} \left( \frac{\tau_3}{\tau} \left[ 1 + \frac{1}{6} \frac{\mu}{r_2^3} (\tau^2 - \tau_3^2) \right] \right) + M_{22} + M_{23} \left( -\frac{\tau_1}{\tau} \left[ 1 + \frac{1}{6} \frac{\mu}{r_2^3} (\tau^2 - \tau_1^2) \right] \right) \right) \end{aligned} \quad (2.49)$$

To express the equation 2.49 more simply, we assign new variables;

$$\begin{aligned} A &= -M_{21} \frac{\tau_3}{\tau} + M_{22} + M_{23} \frac{\tau_1}{\tau} \\ B &= -\frac{1}{6} \left( M_{21} (\tau^2 - \tau_3^2) \frac{\tau_3}{\tau} - M_{23} (\tau^2 - \tau_1^2) \frac{\tau_1}{\tau} \right) \end{aligned} \quad (2.50)$$

By inserting equation 2.50 into equation 2.47, the equation becomes:

$$\rho_2 = A + \frac{\mu B}{r_2^3} \quad (2.51)$$

This allow us to express  $\rho_2$  using the sole unknown variable  $r_2$ . To resolve  $\rho_2$ , we cannot depend solely on equation 2.51. Instead, we should consider the following equations:

$$\vec{r}_2 = \rho_2 \vec{\rho}_2 + \vec{R}_{site_2} \quad (2.52)$$

$$r_2^2 = \rho_2^2 + 2E\rho_2 + R_{site_2}^2 \quad (2.53)$$

where

$$E = \vec{R}_{site_2} \cdot \vec{\rho}_2 \quad (2.54)$$

Applying the value of  $\rho_2$  from equation 2.51 into equation 2.53, it leads to:

$$r_2^2 = \left(A + \frac{\mu B}{r_2^2}\right)^2 + 2E \left(A + \frac{\mu B}{r_2^2}\right) + R_{site_2}^2 \quad (2.55)$$

The rearrangement of these terms forms an eighth-degree polynomial:

$$x^8 + ax^6 + bx^3 + c = 0 \quad (2.56)$$

Here,  $x = r_2$ , with the coefficients defined as follows:

$$a = -(A^2 + 2AE + R_{site_2}^2) \quad b = -2\mu B(A + E) \quad c = -\mu^2 B^2 \quad (2.57)$$

Solving this complex polynomial allows us to determine  $r_2$ . We can then compute the coefficients  $c_1$  and  $c_3$ . From equation 2.47, it is possible to calculate  $\rho_1$  and  $\rho_3$ . With these positions known, velocities are needed next. Using techniques like Gibbs, Herrick-Gibbs, or p-iteration, velocities accurately determined.

### 2.3 Double-R

Determining orbits from angular observations is a well-established problem, with Gauss's and Laplace's methods being historically significant, as discussed in Escobal's work. These methods, initially developed primarily for minor planets, have been adapted for modern applications, such as near-Earth satellite orbit determination.

According to Escobal, Gauss's method is highly effective for closely spaced observations. However, he notes that for observations over longer intervals, the method becomes complex and may not converge. To address this issue, Escobal introduces the double-r iteration method as an alternative.

The double-r iteration method, as outlined by Escobal, estimates orbits solely from angular measurements. It involves iteratively guessing the magnitudes of the radius vectors at two different times to solve the orbital equations. Despite the challenge of guessing these magnitudes, Escobal emphasizes that this method allows flexibility in determining orbits with varying observation intervals.

The slant range vector  $\mathbf{L}_i$  is expressed as:

$$\mathbf{L}_i = \begin{bmatrix} L_{x_i} \\ L_{y_i} \\ L_{z_i} \end{bmatrix} = \begin{bmatrix} \cos \delta_i \cos \alpha_i \\ \cos \delta_i \sin \alpha_i \\ \sin \delta_i \end{bmatrix} \quad (2.58)$$

where  $i$  corresponds to observation data at times  $t_1$ ,  $t_2$  and  $t_3$ , with  $\delta$  being declination and  $\alpha$  being right ascension.

Using the relationship between  $r$ ,  $\rho$  and  $R_{site}$  from given figure [2.2](#):

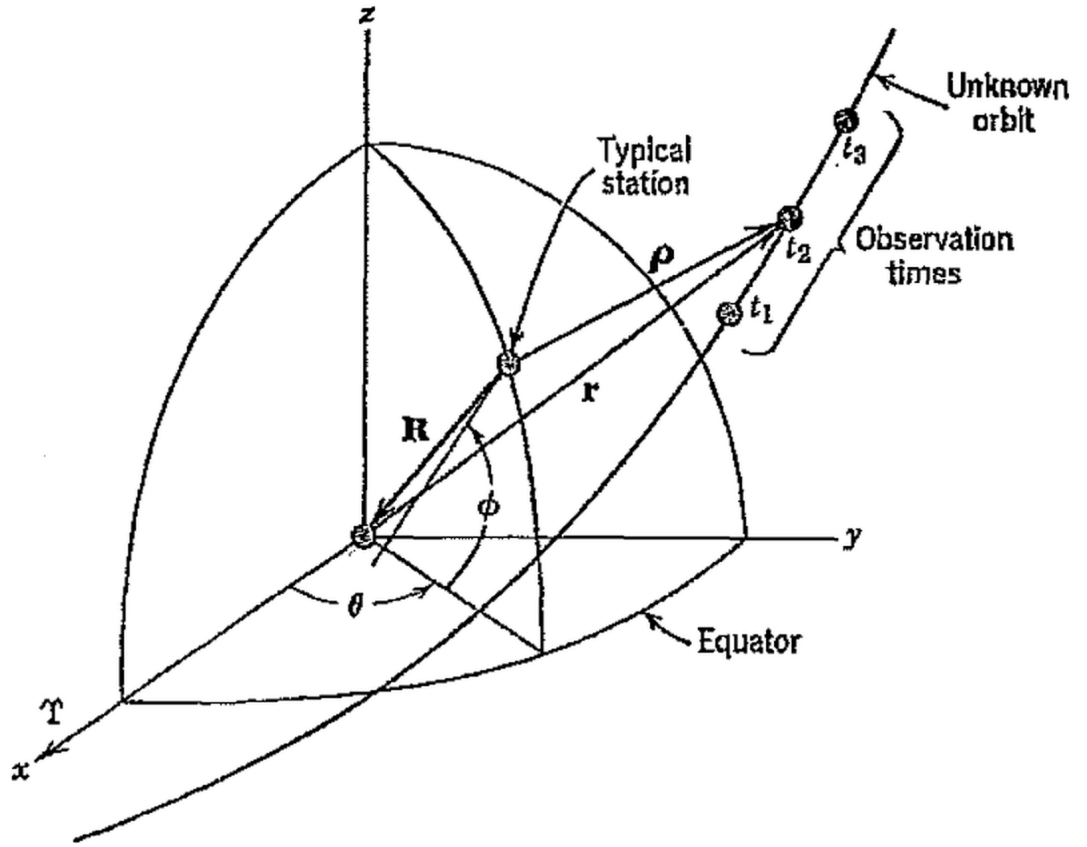


Figure 2.2: Orbit Geometry[26]

$$\rho_i = r_i + R_{site_i} \quad (2.59)$$

By transposing  $R_{site_i}$  and taking the dot product of equation [2.59](#) upon itself, we obtain:

$$\rho_i^2 + \rho_i C_{vt} + (R_{site_i}^2 - r_i^2) = 0 \quad (2.60)$$

where

$$C_{vt} = -2\mathbf{L}_i \cdot \mathbf{R}_{site_i} \quad (2.61)$$

The values of  $C_{vt}$  are derived from the station position and observational data, Solving

for

$$\rho_i = \frac{-C_{vt} \pm \sqrt{C_{vt}^2 - 4(R_{site_i}^2 - r_i^2)}}{2} \quad (2.62)$$

Choosing the correct sign based on  $R_{site_i}^2 - r_i^2 < 0$  and  $C_{vt} > 0$ ,  $\rho_3$  is determined by:

$$\rho_3 = \frac{\mathbf{R}_{site_3} \cdot \mathbf{W}}{\mathbf{L}_3 \cdot \mathbf{W}} \quad (2.63)$$

where  $\mathbf{r}_3 \cdot \mathbf{W} = 0$  and  $\mathbf{W}$  is:

$$\mathbf{W} = \frac{\mathbf{r}_1 \times \mathbf{r}_2}{\mathbf{r}_1 \mathbf{r}_2} \quad (2.64)$$

To calculate the time interval between  $(\mathbf{r}_3, \mathbf{r}_2)$  and  $(\mathbf{r}_1, \mathbf{r}_2)$  using:

$$\cos(\theta_j - \theta_k) = \frac{\mathbf{r}_j \cdot \mathbf{r}_k}{\mathbf{r}_j \mathbf{r}_k} \quad (2.65)$$

$$\sin(\theta_j - \theta_k) = s [1 - \cos^2(\theta_j - \theta_k)]^{1/2} \quad (2.66)$$

where

$$s = \pm \frac{(x_k y_j - x_j y_k)}{x_k y_j - x_j y_k} \quad (2.67)$$

where,  $\theta_j - \theta_k$  for  $j = 2, 3$  and  $k = 1, 2$ .

By iteratively adjusting the radius vectors and recalculating the time intervals, the double-r iteration method provides a robust solution for orbit determination using angular observations over varying intervals.

To refine the assumed values  $r_1$  and  $r_2$ , it is essential to compute the resulting time interval between  $(\mathbf{r}_3, \mathbf{r}_2)$  and  $(\mathbf{r}_1, \mathbf{r}_2)$ . This involves using gaussian equation:

$$p = \frac{c_1 r_1 + c_3 r_3 - r_2}{c_1 + c_3 - 1} \quad (2.68)$$

where  $p$  is the semi-parameter of the orbit,  $c_1$  and  $c_3$  are geometric ratios defined as:

$$c_1 = \frac{r_2 \sin(\Delta\theta_{32})}{r_1 \sin(\Delta\theta_{31})} \quad c_3 = \frac{r_2 \sin(\Delta\theta_{21})}{r_3 \sin(\Delta\theta_{31})} \quad (2.69)$$

To avoid singularities in [2.68](#), especially when  $\Delta\theta_{31} = \pi$ , an alternative form is used:

$$p = \frac{r_1 + c_{r3}r_3 - c_{r1}r_2}{1 + c_{r3} - c_{r1}}$$

where

$$c_{r1} = \frac{r_1 \sin(\Delta\theta_{31})}{r_2 \sin(\Delta\theta_{32})} \quad c_{r3} = \frac{r_1 \sin(\Delta\theta_{21})}{r_3 \sin(\Delta\theta_{32})} \quad (2.70)$$

By eliminating singularities, the semi-parameter  $p$  can be determined, allowing the products of eccentricity and true anomaly  $[e \cos \theta]$  to be calculated:

$$[e \cos \theta]_i = \frac{p}{r_i} - 1 \quad i = 1, 2, 3 \quad (2.71)$$

The next step is determining the product  $[e \sin \theta]$ , using the identities:

$$e \sin \theta_1 = \frac{\cos \Delta\theta_{21} (e \cos \theta_1) - (e \cos \theta_2)}{\sin \Delta\theta_{21}} \quad (2.72)$$

$$e \sin \theta_2 = \frac{-\cos \Delta\theta_{21} (e \cos \theta_2) + (e \cos \theta_1)}{\sin \Delta\theta_{21}} \quad (2.73)$$

For cases where  $\sin \Delta\theta_{31} \neq \pi$ , we use:

$$e \sin \theta_3 = \frac{\cos \Delta\theta_{32} (e \cos \theta_2) - (e \cos \theta_3)}{\sin \Delta\theta_{31}} \quad (2.74)$$

$$e \sin \theta_3 = \frac{-\cos \Delta\theta_{32} (e \cos \theta_2) - (e \cos \theta_3)}{\sin \Delta\theta_{31}} \quad (2.75)$$

The eccentricity  $e$  and semi-major axis  $a$  are then calculated:

$$e^2 = (e \sin \theta_2)^2 + (e \cos \theta_2)^2 \quad (2.76)$$

$$a = \frac{p}{1 - e^2} \quad (2.77)$$

To compute the mean motion  $n$  :

$$n = \sqrt{\frac{\mu}{a^3}} \quad (2.78)$$

where  $\mu$  is the dynamic gravitational constant of the planet.

Using Kepler's equation  $M = E - e \sin E$ , and expanding factors of  $e \sin \theta$  and  $e \cos \theta$  into eccentric anomaly  $E$ , we get:

$$\sin E = \frac{r}{p} \sqrt{1 - e^2} \sin \theta \quad (2.79)$$

$$\cos E = \frac{r}{p} (e + \cos \theta) \quad (2.80)$$

By transforming differences in true anomalies to eccentric anomalies, we derive the time intervals:

$$\sin \Delta E_{32} = \frac{r_3}{\sqrt{ap}} \sin \Delta \theta_{32} - \frac{r_3}{p} [1 - \cos \Delta \theta_{32}] S_e \quad (2.81)$$

$$\cos \Delta E_{32} = 1 - \frac{r_3^2}{ap} [1 - \cos \Delta \theta_{32}] \quad (2.82)$$

where

$$S_e = e \sin E_2 = \frac{r_2}{p} \sqrt{1 - e^2} (e \sin \theta_2) \quad (2.83)$$

$$C_e = e \cos E_2 = \frac{r_2}{p} [e^2 + (e \cos \theta_2)] \quad (2.84)$$

Finally, Kepler's equation and time intervals are given by:

$$M = E - e \sin E \quad (2.85)$$

where  $M$  is the mean anomaly and  $E$  the eccentric anomaly. By referencing these to an epoch  $t_2$ , we get:

$$M_3 - M_2 = \Delta E_{32} + 2S_e \sin^2 \left( \frac{\Delta E_{32}}{2} \right) - C_e \sin \Delta E_{32} \quad (2.86)$$

$$M_1 - M_2 = -\Delta E_{21} + 2S_e \sin^2 \left( \frac{\Delta E_{21}}{2} \right) - C_e \sin \Delta E_{21} \quad (2.87)$$

With the mean motion  $n$  :

$$M_i - M_2 = n (t_i - t_2), \quad i = 1, 3 \quad (2.88)$$

The time intervals as a function of  $r_1$  and  $r_2$  are:

$$\bar{t}_3 - \bar{t}_2 = \frac{M_3 - M_2}{n} \quad \bar{t}_1 - \bar{t}_2 = \frac{M_1 - M_2}{n} \quad (2.89)$$

Here, the overhead bars denote times computed from the assumed  $r_1$  and  $r_2$ . To refine these values, we introduce the following functions:

$$F_1 \equiv r_1 - \left( \frac{M_1 - M_2}{n} \right) \quad (2.90)$$

$$F_2 \equiv r_3 - \left( \frac{M_3 - M_2}{n} \right) \quad (2.91)$$

The functions  $F_1$  and  $F_2$  must be forced to zero match hypothetical modified times based on incorrect assumed values of  $r_1$  and  $r_2$ , with true modified times calculated from:

$$r_1 = t_1 - t_2 \quad r_3 = t_3 - t_2 \quad (2.92)$$

where  $t_1$ ,  $t_2$  and  $t_3$  are the universal observation times with the adopted epoch  $t_2$ . This process will be tackled using a numerical procedure.

If the observations are taken over multiple revolutions, it is necessary to increment the  $F_1$  and  $F_2$  functions as follows:

$$F_1 = \tau_1 - \left( \frac{M_1 - M_2}{n} \right) + \left( \frac{2\pi}{n} \right) \lambda \quad (2.93)$$

$$F_2 = \tau_3 - \left( \frac{M_3 - M_2}{n} \right) - \left( \frac{2\pi}{n} \right) \lambda \quad (2.94)$$

where  $\lambda$  is a multiplier. One assumes  $\lambda = 1, 2, 3, \dots, q$ , and for each integer value of  $\lambda$  attempts to continue the solution. This increment is essentially adding multiples of the orbital period to the respective modified time variable.

To numerically solve for  $r_1$  and  $r_2$ , it is necessary to force the functions  $F_i$  to zero:

$$F_i(r_1, r_2) = 0 \quad i = 1, 2 \quad (2.95)$$

Using a two-dimensional Taylor expansion, the linear approximation of  $F_i$  can be expressed as:

$$-F_i = \left( \frac{\partial F_i}{\partial r_1} \right) \Delta r_1 + \left( \frac{\partial F_i}{\partial r_2} \right) \Delta r_2 \quad (2.96)$$

Replacing the differentials with finite differences and solving the linear equations yields:

$$\Delta r_1 = -\frac{\Delta_1}{\Delta} \quad \Delta r_2 = -\frac{\Delta_2}{\Delta} \quad (2.97)$$

where

$$\Delta = \left( \frac{\partial F_1}{\partial r_1} \right) \left( \frac{\partial F_2}{\partial r_2} \right) - \left( \frac{\partial F_2}{\partial r_1} \right) \left( \frac{\partial F_1}{\partial r_2} \right) \quad (2.98)$$

$$\Delta_1 = \left( \frac{\partial F_2}{\partial r_2} \right) F_1 - \left( \frac{\partial F_1}{\partial r_2} \right) F_2 \quad (2.99)$$

$$\Delta_2 = \left( \frac{\partial F_1}{\partial r_1} \right) F_2 - \left( \frac{\partial F_2}{\partial r_1} \right) F_1 \quad (2.100)$$

These partial derivatives can be approximated numerically:

$$\frac{\partial F_1}{\partial r_1} \approx \frac{F_1(r_1 + \Delta r_1, r_2) - F_1(r_1, r_2)}{\Delta r_1} \quad (2.101)$$

$$\frac{\partial F_1}{\partial r_2} \approx \frac{F_1(r_1, r_2 + \Delta r_2) - F_1(r_1, r_2)}{\Delta r_2} \quad (2.102)$$

By iteratively refining  $r_1$  and  $r_2$  using these updates, we improve the estimates. The process involves computing new values:

$$(r_i)_{j+1} = (r_i)_j + (\Delta r_i)_j \quad j = 1, 2, \dots, q \quad (2.103)$$

Once  $r_1$  and  $r_2$  stabilize, we can determine the closed-form  $f$  and  $g$  coefficients:

$$f = 1 - \frac{a}{r_2} [1 - \cos \Delta E_{32}] \quad (2.104)$$

$$g = \tau_3 - \sqrt{\frac{a^3}{\mu}} [\Delta E_{32} - \sin \Delta E_{32}] \quad (2.105)$$

Finally, solving for  $\dot{r}_2$ :

$$\dot{\mathbf{r}}_2 = \frac{\mathbf{r}_3 - f\mathbf{r}_2}{g} \quad (2.106)$$

This iterative process continues until convergence, at which point the fundamental set  $r_1, r_2$  is found, and the orbit is determined.

## 2.4 F and G Series

In this approach, we determine the initial orbit using three optical observations at times  $t_1, t_2$  and  $t_3$  with the help of the  $f$  and  $g$  series. This method is followed from R.Bate.

From a single observation site, three sets of right ascension  $\alpha$  and declination  $\delta$  data can be obtained at times  $t_1, t_2$  and  $t_3$ . These observations allow the unit line of sight vector from observation point to observation point to be determined. This line of sight vector should be:

$$\mathbf{L}_i = \mathbf{I} \cos \delta_i \cos \alpha_i + \mathbf{J} \cos \delta_i \sin \alpha_i + \mathbf{K} \sin \delta_i \quad (2.107)$$

Following this, the vector  $\mathbf{r}$  from the center of Earth with using the unit line of sight vector  $\mathbf{L}$ , observation site vector  $\mathbf{R}$  and slant range  $\rho$ , is expressed as:

$$\mathbf{r} = \mathbf{R} + \rho \mathbf{L} \quad (2.108)$$

The position vector  $\mathbf{r}$  can be expressed using the  $f$  and  $g$  series as:

$$\mathbf{r} = f(\mathbf{r}_2, \mathbf{v}_2, t - t_2) \mathbf{r}_2 + g(\mathbf{r}_2, \mathbf{v}_2, t - t_2) \mathbf{v}_2 \quad (2.109)$$

For specific instances  $t_i$ , we define  $f_i = f(\mathbf{r}_2, \mathbf{v}_2, t_i - t_2)$  and  $g_i = g(\mathbf{r}_2, \mathbf{v}_2, t_i - t_2)$ , resulting in the following equations:

$$f_1 \mathbf{r}_2 + g_1 \mathbf{v}_2 = \mathbf{R}_1 + \rho_1 \mathbf{L}_1 \quad (2.110)$$

$$\mathbf{r}_2 = \mathbf{R}_2 + \rho_2 \mathbf{L}_2 \quad (2.111)$$

$$f_3 \mathbf{r}_2 + g_3 \mathbf{v}_2 = \mathbf{R}_3 + \rho_3 \mathbf{L}_3 \quad (2.112)$$

These provide nine equations involving unknowns  $r_2, \mathbf{r}_2, \mathbf{v}_2, \rho_1, \rho_2$  and  $\rho_3$ . By cross-multiplying the  $i$ th equation with  $\mathbf{L}_i$ , we eliminate  $\rho_i$ , leading to:

$$f_1 \mathbf{L}_1 \times \mathbf{r}_2 + g_1 \mathbf{L}_1 \times \mathbf{v}_2 = \mathbf{L}_1 \times \mathbf{R}_1 \quad (2.113)$$

$$\mathbf{L}_2 \times \mathbf{r}_2 = \mathbf{L}_2 \times \mathbf{R}_2 \quad (2.114)$$

$$f_3 \mathbf{L}_3 \times \mathbf{r}_2 + g_3 \mathbf{L}_3 \times \mathbf{v}_2 = \mathbf{L}_3 \times \mathbf{R}_3 \quad (2.115)$$

To further elucidate, these equations can be expanded into Cartesian components for  $\mathbf{r}_2(x, y, z)$  and  $\mathbf{v}_2(\dot{x}, \dot{y}, \dot{z})$ ;

$$f_1 L_{1z}x - f_1 L_{1x}z + g_1 L_{1z}\dot{x} - g_1 L_{1x}\dot{z} = R_{1x}L_{1z} - R_{1z}L_{1x} \quad (2.116)$$

$$f_1 L_{1z}y - f_1 L_{1y}z + g_1 L_{1z}\dot{y} - g_1 L_{1y}\dot{z} = R_{1y}L_{1z} - R_{1z}L_{1y} \quad (2.117)$$

$$L_{2z}x - L_{2x}z = R_{2x}L_{2z} - R_{2z}L_{2x} \quad (2.118)$$

$$L_{2z}y - L_{2y}z = R_{2y}L_{2z} - R_{2z}L_{2y} \quad (2.119)$$

$$f_3 L_{3z}x - f_3 L_{3x}z + g_3 L_{3z}\dot{x} - g_3 L_{3x}\dot{z} = R_{3x}L_{3z} - R_{3z}L_{3x} \quad (2.120)$$

$$f_3 L_{3z}y - f_3 L_{3y}z + g_3 L_{3z}\dot{y} - g_3 L_{3y}\dot{z} = R_{3y}L_{3z} - R_{3z}L_{3y} \quad (2.121)$$

## 2.5 Lambert

Lambert's problem involves determining the orbit of a body given two position vectors,  $\mathbf{r}_1$  and  $\mathbf{r}_2$  and the time of flight between them. This classical problem is essential in orbital mechanics, particularly for interplanetary mission planning and astrodynamics.

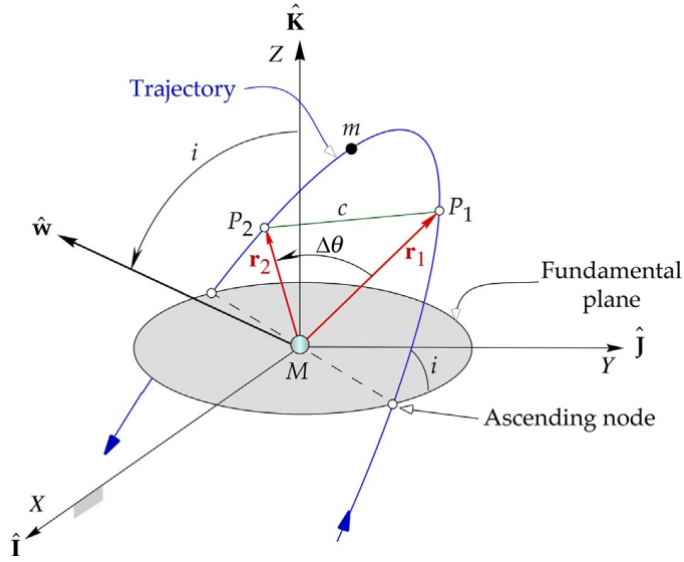


Figure 2.3: Lambert[24]

Consider two points  $P_1$  and  $P_2$  in the orbit of a mass  $m$ , with position  $\mathbf{r}_1$  and  $\mathbf{r}_2$  respectively. The true anomaly change  $\Delta\theta$  between these points can be determined using the relationship:

$$\cos \Delta\theta = \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{r_1 r_2} \quad (2.122)$$

where  $r_1$  and  $r_2$  are the magnitudes of the position vectors  $\mathbf{r}_1$  and  $\mathbf{r}_2$ .

The vector  $\mathbf{w}$  is defined as the unit normal to the orbital plane:

$$\mathbf{w} = \frac{\mathbf{r}_1 \times \mathbf{r}_2}{|\mathbf{r}_1 \times \mathbf{r}_2|} \quad (2.123)$$

Using  $\mathbf{w}$ , the z-component of the cross product  $\mathbf{r}_1 \times \mathbf{r}_2$  can be calculated to resolve

the quadrant ambiguity in  $\Delta\theta$ :

$$\mathbf{r}_1 \times \mathbf{r}_2 = r_1 r_2 \sin \Delta\theta \mathbf{w}_z \quad (2.124)$$

By determining the sign of  $\sin \Delta\theta$ , we can resolve the quadrant of  $\Delta\theta$ . The angle  $\Delta\theta$  can thus be expressed concisely as follows:

$$\Delta\theta = \left\{ \begin{array}{ll} \cos^{-1} \left( \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{r_1 r_2} \right) & \text{if } (\mathbf{r}_1 \times \mathbf{r}_2)_z \geq 0 \\ 360 - \cos^{-1} \left( \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{r_1 r_2} \right) & \text{if } (\mathbf{r}_1 \times \mathbf{r}_2)_z < 0 \\ 180 - \cos^{-1} \left( \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{r_1 r_2} \right) & \text{if } \mathbf{r}_1 \cdot \mathbf{r}_2 < 0 \text{ and } (\mathbf{r}_1 \times \mathbf{r}_2)_z \geq 0 \\ 180 + \cos^{-1} \left( \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{r_1 r_2} \right) & \text{if } \mathbf{r}_1 \cdot \mathbf{r}_2 < 0 \text{ and } (\mathbf{r}_1 \times \mathbf{r}_2)_z < 0 \end{array} \right\} \quad (2.125)$$

J.H. Lambert proposed that the time of flight between two points on an orbit is independent of the orbit's eccentricity and depends only on the sum  $r_1 + r_2$ , the semi-major axis  $a$ , and the length of the chord joining  $P_1$  and  $P_2$ :

$$c = \sqrt{r_1^2 + r_2^2 - 2r_1 r_2 \cos \Delta\theta} \quad (2.126)$$

Lambert's problem is to find the trajectory joining  $P_1$  and  $P_2$  given  $t_f$ . The trajectory is determined once the velocities  $\mathbf{v}_1$  and  $\mathbf{v}_2$  are found.

Using this, find the orbital elements from either  $\mathbf{r}_1$  and  $\mathbf{v}_1$  or  $\mathbf{r}_2$  and  $\mathbf{v}_2$ . The Lagrange  $f$  and  $g$  coefficients and their time derivatives are functions of the change in true anomaly  $\Delta\theta$ .

$$f = 1 - \frac{\mu r_2}{h^2} (1 - \cos \Delta\theta) \quad (2.127)$$

$$g = -\frac{r_1 r_2}{h} (1 - \sin \Delta\theta) \quad (2.128)$$

$$\dot{f} = \frac{\mu}{h} \frac{1 - \cos \Delta\theta}{\sin \Delta\theta} \left[ \frac{\mu}{h^2} (1 - \cos \Delta\theta) - \frac{1}{r_1} - \frac{1}{r_2} \right] \quad (2.129)$$

$$\dot{g} = 1 - \frac{\mu r_1}{h^2} (1 - \cos \Delta\theta) \quad (2.130)$$

## 2.6 Gibbs

The Gibbs method of orbit determination is based on three position vectors  $\mathbf{r}_1, \mathbf{r}_2$  and  $\mathbf{r}_3$  at different times ( $t_1 < t_2 < t_3$ ). The primary focus of this method is to determine velocities  $\mathbf{v}_1, \mathbf{v}_2$  and  $\mathbf{v}_3$  using vector analysis, a technique pioneered by J.W. Gibbs (1839-1903). This explanation follows the framework established by Curtis.

It is understood that angular momentum must be conserved through the positions defined by  $\mathbf{r}_1$  to  $\mathbf{r}_3$ . Additionally, these three position vectors  $\mathbf{r}_1, \mathbf{r}_2$  and  $\mathbf{r}_3$  must lie in the same orbital plane of the body. Based on this principle, the normal vector to the plane of  $\mathbf{r}_2$  and  $\mathbf{r}_3$  ( $\hat{\mathbf{C}}_{23} = \mathbf{r}_2 \times \mathbf{r}_3 / \|\mathbf{r}_2 \times \mathbf{r}_3\|$ ) must be perpendicular to the unit vector of  $\hat{\mathbf{u}}_{r_1} = \mathbf{r}_1 / r_1$ . Thus, by taking the dot product of these two vectors, the result must be zero, confirming their perpendicularity.

$$\hat{\mathbf{u}}_{r_1} \cdot \hat{\mathbf{C}}_{23} = 0 \quad (2.131)$$

In figure 2.1, the position vectors  $\mathbf{r}_1, \mathbf{r}_2$  and  $\mathbf{r}_3$  along with their respective planes, are illustrated. The planes and position vectors lie within the same plane so that scalar factors  $c_1$  and  $c_3$  can be applied to the position vectors  $\mathbf{r}_1$  and  $\mathbf{r}_3$ . Following this, the  $\mathbf{r}_2$  vector is defined as the sum of  $\mathbf{r}_1$  and  $\mathbf{r}_3$  with their scalar factor.

$$\mathbf{r}_2 = c_1 \mathbf{r}_1 + c_3 \mathbf{r}_3 \quad (2.132)$$

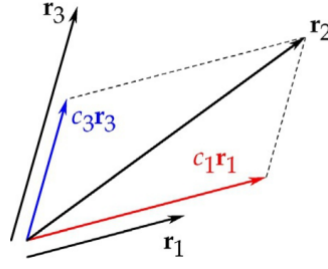


Figure 2.4: Gibbs [24]

To determine the velocity vector  $\mathbf{v}$  using given three position vectors  $\mathbf{r}$ , we shall begin with following equation:

$$\mathbf{v} \times \mathbf{h} = \mu \left( \frac{\mathbf{r}}{r} + \mathbf{e} \right) \quad (2.133)$$

where  $\mathbf{h}$  is angular momentum and  $\mathbf{e}$  is the eccentricity vector. To extract the velocity vector, we take the cross product of the equation with the angular momentum;

$$\mathbf{h} \times (\mathbf{v} \times \mathbf{h}) = \mu \left( \frac{\mathbf{h} \times \mathbf{r}}{r} + \mathbf{h} \times \mathbf{e} \right) \quad (2.134)$$

By using bac-cab rule, left side becomes;

$$\mathbf{h} \times (\mathbf{v} \times \mathbf{h}) = \mathbf{v}(\mathbf{h} \cdot \mathbf{h}) - \mathbf{h}(\mathbf{h} \cdot \mathbf{v}) \quad (2.135)$$

But, we know that;  $\mathbf{h} \cdot \mathbf{h} = h^2$  and  $\mathbf{v} \cdot \mathbf{h} = 0$ ;

$$\mathbf{h} \times (\mathbf{v} \times \mathbf{h}) = h^2 \mathbf{v} \quad (2.136)$$

2.133 becomes;

$$\mathbf{v} = \frac{\mu}{h^2} \left( \frac{\mathbf{h} \times \mathbf{r}}{r} + \mathbf{h} \times \mathbf{e} \right) \quad (2.137)$$

Utilizing the perifocal coordinate system, where the unit vector  $\hat{\mathbf{p}}$  aligns with direction of the eccentricity  $\mathbf{e}$ , and  $\hat{\mathbf{w}}$  is the unit vector perpendicular to the orbital plane in the direction of the angular momentum  $\mathbf{h}$ . The eccentricity  $\mathbf{e}$  and angular momentum  $\mathbf{h}$  can be written as;

$$\mathbf{e} = e\hat{\mathbf{p}} \quad (2.138)$$

$$\mathbf{h} = h\hat{\mathbf{w}} \quad (2.139)$$

2.137 becomes;

$$\mathbf{v} = \frac{\mu}{h^2} \left( \frac{h\hat{\mathbf{w}} \times \mathbf{r}}{r} + h\hat{\mathbf{w}} \times e\hat{\mathbf{p}} \right) = \frac{\mu}{h} \left[ \frac{\hat{\mathbf{w}} \times \mathbf{r}}{r} + e(\hat{\mathbf{w}} \times \hat{\mathbf{p}}) \right] \quad (2.140)$$

Recall that;  $\hat{\mathbf{p}} \times \hat{\mathbf{q}} = \hat{\mathbf{w}}, \hat{\mathbf{w}} \times \hat{\mathbf{p}} = \hat{\mathbf{q}}$  and  $\hat{\mathbf{q}} \times \hat{\mathbf{w}} = \hat{\mathbf{p}}$ . Then;

$$\mathbf{v} = \frac{\mu}{h} \left( \frac{\hat{\mathbf{w}} \times \mathbf{r}}{r} + e\hat{\mathbf{q}} \right) \quad (2.141)$$

At this stage, we can take the dot product of 2.132 with the eccentricity vector  $\mathbf{e}$  to be able to use 2.141 in future equations.

$$\mathbf{r}_2 \cdot \mathbf{e} = c_1 \mathbf{r}_1 \cdot \mathbf{e} + c_1 \mathbf{r}_1 \cdot \mathbf{e} \quad (2.142)$$

and recall that

$$r + \mathbf{r} \cdot \mathbf{e} = \frac{h^2}{\mu} \quad (2.143)$$

where  $\mathbf{r}$  represents the position vector,  $\mathbf{e}$  denotes the eccentricity vector,  $h$  is defined as the angular momentum and  $\mu$  is identified as the gravitational parameter so that;

$$\mathbf{r}_1 \cdot \mathbf{e} = \frac{h^2}{\mu} - r_1 \quad \mathbf{r}_2 \cdot \mathbf{e} = \frac{h^2}{\mu} - r_2 \quad \mathbf{r}_3 \cdot \mathbf{e} = \frac{h^2}{\mu} - r_3 \quad (2.144)$$

and then,

$$\frac{h^2}{\mu} - r_2 = c_1 \left( \frac{h^2}{\mu} - r_1 \right) + c_3 \left( \frac{h^2}{\mu} - r_3 \right) \quad (2.145)$$

Let us take the cross product to [2.132](#) with both  $\mathbf{r}_1$  and  $\mathbf{r}_3$  for to eliminate  $c_1$  and  $c_3$ .  
Using  $\mathbf{a} \times \mathbf{b} = -(\mathbf{b} \times \mathbf{a})$  and  $\mathbf{a} \times \mathbf{a} = 0$ ;

$$\mathbf{r}_2 \times \mathbf{r}_1 = c_1 \mathbf{r}_1 \times \mathbf{r}_1 + c_3 \mathbf{r}_3 \times \mathbf{r}_1$$

$$\mathbf{r}_2 \times \mathbf{r}_1 = 0 + c_3 \mathbf{r}_3 \times \mathbf{r}_1$$

$$\mathbf{r}_2 \times \mathbf{r}_1 = c_3 (\mathbf{r}_3 \times \mathbf{r}_1) \quad (2.146)$$

$$\mathbf{r}_2 \times \mathbf{r}_3 = c_1 \mathbf{r}_1 \times \mathbf{r}_3 + c_3 \mathbf{r}_3 \times \mathbf{r}_3$$

$$\mathbf{r}_2 \times \mathbf{r}_3 = c_1 \mathbf{r}_1 \times \mathbf{r}_3 + 0$$

$$\mathbf{r}_2 \times \mathbf{r}_3 = c_1 (\mathbf{r}_1 \times \mathbf{r}_3)$$

$$\mathbf{r}_2 \times \mathbf{r}_3 = -c_1 (\mathbf{r}_3 \times \mathbf{r}_1) \quad (2.147)$$

Now multiply [2.145](#) by  $\mathbf{r}_3 \times \mathbf{r}_1$ ;

$$\frac{h^2}{\mu} (\mathbf{r}_3 \times \mathbf{r}_1) - r_2 (\mathbf{r}_3 \times \mathbf{r}_1) = c_1 (\mathbf{r}_3 \times \mathbf{r}_1) \left( \frac{h^2}{\mu} - r_1 \right) + c_3 (\mathbf{r}_3 \times \mathbf{r}_1) \left( \frac{h^2}{\mu} - r_3 \right) \quad (2.148)$$

Using [2.146](#) and [2.147](#);

$$\frac{h^2}{\mu} (\mathbf{r}_3 \times \mathbf{r}_1) - r_2 (\mathbf{r}_3 \times \mathbf{r}_1) = -(\mathbf{r}_2 \times \mathbf{r}_3) \left( \frac{h^2}{\mu} - r_1 \right) + (\mathbf{r}_2 \times \mathbf{r}_1) \left( \frac{h^2}{\mu} - r_3 \right)$$

$$\frac{h^2}{\mu} (\mathbf{r}_1 \times \mathbf{r}_2 + \mathbf{r}_2 \times \mathbf{r}_3 + \mathbf{r}_3 \times \mathbf{r}_1) = r_1 (\mathbf{r}_2 \times \mathbf{r}_3) + r_2 (\mathbf{r}_3 \times \mathbf{r}_1) + r_3 (\mathbf{r}_1 \times \mathbf{r}_2) \quad (2.149)$$

Now, we can introduce new coefficients for the practical solution method and [2.149](#) becomes;

$$\mathbf{N} = r_1 (\mathbf{r}_2 \times \mathbf{r}_3) + r_2 (\mathbf{r}_3 \times \mathbf{r}_1) + r_3 (\mathbf{r}_1 \times \mathbf{r}_2) \quad (2.150)$$

$$\mathbf{D} = (\mathbf{r}_1 \times \mathbf{r}_2 + \mathbf{r}_2 \times \mathbf{r}_3 + \mathbf{r}_3 \times \mathbf{r}_1) \quad (2.151)$$

Then, 2.149 can be expressed in terms of  $\mathbf{N}$  and  $\mathbf{D}$ ;

$$\mathbf{N} = \frac{h^2}{\mu} \mathbf{D} \quad (2.152)$$

To find the angular momentum  $h$ , we can use  $N = \|\mathbf{N}\|$  and  $D = \|\mathbf{D}\|$  so that;

$$h = \sqrt{\mu \frac{N}{D}} \quad (2.153)$$

We know that the vectors of  $\mathbf{r}_1 \times \mathbf{r}_2$ ,  $\mathbf{r}_2 \times \mathbf{r}_3$  and  $\mathbf{r}_3 \times \mathbf{r}_1$  must lie in the same plane, which is normal to the orbital plane. Thus,  $\mathbf{D}$  may be considered to normal to the orbital plane. Based on this information, the  $\mathbf{D}$  vector can be expressed in the perifocal frame and can be written in terms of  $\hat{\mathbf{w}}$ . Consequently,  $\mathbf{D}$  becomes;

$$\hat{\mathbf{w}} = \frac{\mathbf{D}}{D} \quad (2.154)$$

To use 2.137, we need to find  $\hat{\mathbf{q}}$  and 2.154 must be written in terms of  $\hat{\mathbf{q}}$ . Then, using 2.138 and 2.139;

$$\hat{\mathbf{q}} = \hat{\mathbf{w}} \times \hat{\mathbf{p}} = \frac{1}{De} (\mathbf{D} \times \mathbf{e}) \quad (2.155)$$

Then,

$$\hat{\mathbf{q}} = \frac{1}{De} ((\mathbf{r}_1 \times \mathbf{r}_2 + \mathbf{r}_2 \times \mathbf{r}_3 + \mathbf{r}_3 \times \mathbf{r}_1) \times \mathbf{e}) \quad (2.156)$$

By using bac-cab rule  $(\mathbf{A} \times \mathbf{B}) \times \mathbf{C} = -\mathbf{C} \times (\mathbf{A} \times \mathbf{B}) = \mathbf{B} (\mathbf{A} \cdot \mathbf{C}) - \mathbf{A} (\mathbf{B} \cdot \mathbf{C})$ ;

$$(\mathbf{r}_2 \times \mathbf{r}_3) \times \mathbf{e} = \mathbf{r}_3 (\mathbf{r}_2 \cdot \mathbf{e}) - \mathbf{r}_2 (\mathbf{r}_3 \cdot \mathbf{e}) \quad (2.157)$$

$$(\mathbf{r}_3 \times \mathbf{r}_1) \times \mathbf{e} = \mathbf{r}_1 (\mathbf{r}_3 \cdot \mathbf{e}) - \mathbf{r}_3 (\mathbf{r}_1 \cdot \mathbf{e}) \quad (2.158)$$

$$(\mathbf{r}_1 \times \mathbf{r}_2) \times \mathbf{e} = \mathbf{r}_1 (\mathbf{r}_2 \cdot \mathbf{e}) - \mathbf{r}_2 (\mathbf{r}_1 \cdot \mathbf{e}) \quad (2.159)$$

By employing [2.144](#):

$$(\mathbf{r}_2 \times \mathbf{r}_3) \times \mathbf{e} = \mathbf{r}_3 \left( \frac{h^2}{\mu} - r_2 \right) - \mathbf{r}_2 \left( \frac{h^2}{\mu} - r_3 \right) = \frac{h^2}{\mu} (\mathbf{r}_3 - \mathbf{r}_2) + r_3 \mathbf{r}_2 - r_2 \mathbf{r}_3 \quad (2.160)$$

$$(\mathbf{r}_3 \times \mathbf{r}_1) \times \mathbf{e} = \mathbf{r}_1 \left( \frac{h^2}{\mu} - r_3 \right) - \mathbf{r}_3 \left( \frac{h^2}{\mu} - r_1 \right) = \frac{h^2}{\mu} (\mathbf{r}_1 - \mathbf{r}_3) + r_1 \mathbf{r}_3 - r_3 \mathbf{r}_1 \quad (2.161)$$

$$(\mathbf{r}_1 \times \mathbf{r}_2) \times \mathbf{e} = \mathbf{r}_2 \left( \frac{h^2}{\mu} - r_1 \right) - \mathbf{r}_1 \left( \frac{h^2}{\mu} - r_2 \right) = \frac{h^2}{\mu} (\mathbf{r}_2 - \mathbf{r}_1) + r_2 \mathbf{r}_1 - r_1 \mathbf{r}_2 \quad (2.162)$$

Summing up [2.160](#), [2.161](#) and [2.162](#):

$$\mathbf{D} \times \mathbf{e} = \frac{h^2}{\mu} (\mathbf{r}_3 - \mathbf{r}_2 + \mathbf{r}_1 - \mathbf{r}_3 + \mathbf{r}_1 - \mathbf{r}_3) + \mathbf{r}_1 (r_2 - r_3) + \mathbf{r}_2 (r_3 - r_1) + \mathbf{r}_3 (r_1 - r_2) \quad (2.163)$$

$$\mathbf{D} \times \mathbf{e} = \mathbf{r}_1 (r_2 - r_3) + \mathbf{r}_2 (r_3 - r_1) + \mathbf{r}_3 (r_1 - r_2) \quad (2.164)$$

[2.155](#) can be expressed as  $\mathbf{D} \times \mathbf{e} = \mathbf{S}$  to simplify the expression, and thus, [2.155](#) and  $\mathbf{S}$  become;

$$\hat{\mathbf{q}} = \frac{1}{De} (\mathbf{D} \times \mathbf{e}) = \frac{1}{De} \mathbf{S} \quad (2.165)$$

$$\mathbf{S} = \mathbf{r}_1 (r_2 - r_3) + \mathbf{r}_2 (r_3 - r_1) + \mathbf{r}_3 (r_1 - r_2) \quad (2.166)$$

Finally, by incorporating 2.153, 2.154 and 2.165 into 2.137, we obtain;

$$\mathbf{v} = \frac{\mu}{h^2} \left( \frac{\mathbf{h} \times \mathbf{r}}{r} + \mathbf{h} \times \mathbf{e} \right) = \frac{\mu}{\sqrt{\mu \frac{N}{D}}} \left[ \frac{\frac{D}{D} \times \mathbf{r}}{r} + e \left( \frac{1}{De} \mathbf{S} \right) \right]$$

$$\mathbf{v} = \sqrt{\mu \frac{N}{D}} \left( \frac{\mathbf{D} \times \mathbf{r}}{r} + \mathbf{S} \right) \quad (2.167)$$

## 2.7 Gooding

The Gooding method, a range-iteration technique, significantly differs from Escobal's approach. Instead of using two time differences as target functions, it calculates the body's position at time  $t_2$  from positions at times  $t_1$  and  $t_3$ . This inversion allows for accurate solutions to Lambert's problem, achieving 13-digit precision after only three iterations. The method is effective in handling multiple revolutions, maintaining accuracy even in highly non-linear situations. It also uses a set of 'universal' elements to propagate state vectors. These elements include  $\alpha$ ,  $r_p$ ,  $p$ ,  $\Omega$  and  $\tau$ . Here,  $\alpha$  is defined as  $\mu a$  (where  $\mu$  is the standard gravitational parameter and  $a$  is the semi-major axis),  $r_p$  as  $a(1 - e)$  (with  $e$  being the eccentricity),  $\tau$  as the time at periapsis,  $p$  as the semi-latus rectum ( $p = a(1 - e^2)$ ) and  $\Omega$  as the longitude of the ascending node in ECEF.

The Gooding method starts with data points  $R_{site_i}$  and  $\Delta_i$  at times  $t_i$  for  $i = 1, 2, 3$ . Here,  $R_{site_i}$  are the position vectors at the respective times, and  $\Delta_i$  are differences in position vectors, specifically  $\Delta_i = R_{i+1} - R_i$ . Additionally, the method assumes a value for  $k$ , representing the number of half-revolutions. The method aims to solve the angles-only problem when  $\Delta_1 \neq \Delta_2$ . To simplify calculations, the method assumes that  $\Delta_i$  are unit vectors. The target functions  $f$  and  $g$  are defined on a plane normal to a vector  $\lambda$ , which is itself defined by the positions and differences in position.

In each iteration, the method computes the position,  $P_c$  within this plane. The plane is redefined at each step to ensure the accurate calculation of the afocal family passing

through  $P_c$ . If  $\Delta_1$  and  $\Delta_2$  align, indicating potential issues, the method interchanges the data to maintain consistency. Convergence is rigorously checked at each step to ensure the accuracy of the results.

The computational framework of the Gooding method is implemented using Fortran-77. It consists of a main program responsible for managing the overall solution process. This program calls a custom-developed subroutine to generate individual solutions for the angles-only problem. The subroutine iteratively updates estimates for variables  $x$  and  $y$ , which correspond to the parameters  $\rho_1$  and  $\rho_3$ . Here,  $\rho_1$  and  $\rho_3$  are initial estimates for the positions along the orbital path. These parameters are refined through each iteration until convergence is achieved.

The subroutine evaluates the partial derivatives of the target functions,  $f_x$  and  $g_y$ , where  $f_x$  is the partial derivative of  $f$  with respect to  $x$ , and  $g_y$  is the partial derivative of  $g$  with respect to  $y$ . These derivatives are used to update  $x$  and  $y$ . The initial estimates for  $\rho_1$  and  $\rho_3$  are iteratively adjusted to meet the convergence criteria. For cases involving multiple revolutions ( $k \geq 2$ ), this method searches multiple solution sets. Each iteration includes refined calculations to ensure alignment of the axis system in the computational framework.

A special subroutine is required to solve the angle-based problem. This subroutine simplifies the solution process by storing the necessary information in a shared data structure. During each iteration, the subroutine checks the convergence of  $x$  and  $y$  and ensures the accuracy of the results. This iterative process includes multiple evaluations and refinements to minimize computational errors and effectively handle multi-cycle scenarios.

The execution of the subroutine is guided by detailed operations and additional documentation. This enables the method to efficiently handle various computational challenges, providing robust and exact solutions to the Lambert problem using comprehensive iteration techniques and universal elements.

The classical Newton-Raphson formula is a basic tool for iteratively refining the root of an equation and can be extended to multiple variables. The formula for two variables is expressed as:

$$\begin{pmatrix} \delta x \\ \delta y \end{pmatrix} = - \begin{pmatrix} f_x & f_y \\ g_x & g_y \end{pmatrix}^{-1} \begin{pmatrix} f \\ g \end{pmatrix} \quad (2.168)$$

Here:

- $\delta x$  and  $\delta y$  are the corrections to the current estimates of the roots of the equations  $f(x, y) = 0$  and  $g(x, y) = 0$ .
- $f_x$  and  $f_y$  are the partial derivatives of  $f$  with respect to  $x$  and  $y$ , respectively.
- $g_x$  and  $g_y$  are the partial derivatives of  $g$  with respect to  $x$  and  $y$ , respectively.

Assuming  $g = 0$ , the equation simplifies to :

$$\delta x = -D^{-1} f g_y \quad (2.169)$$

$$\delta y = D^{-1} g_x \quad (2.170)$$

where  $D$  is the determinant of the Jacobian matrix, defined as:

$$D = f_x g_y - f_y g_x \quad (2.171)$$

The Jacobian matrix captures the rates of change of the functions with respect to the variables and is crucial for determining the corrections  $\delta x$  and  $\delta y$ .

To efficiently derive partial derivatives, the method requires only a minimum number of additional calls to the function at each iteration. Specifically, this includes:

- Incrementing  $x$  slightly to compute  $f_x$  and  $g_x$ .
- Incrementing  $y$  slightly to compute  $f_y$  and  $g_y$ .

For example, the partial derivative  $f_x$  can be computed as:

$$f_x = \frac{f(x + \delta x, y) - f(x, y)}{\delta x} \quad (2.172)$$

Similarly,  $g_x$  is computed as:

$$g_x = \frac{g(x + \delta x, y) - g(x, y)}{\delta x} \quad (2.173)$$

These simple formulas provide accurate approximations of the derivatives and are then used to update the estimates of the roots.

Halley's method is a significant development of the Newton-Raphson method that includes second-order derivatives and provides cubic approximation. This section delves deeper into the derivation, application, and advantages of Halley's method. Halley's method extends the Taylor series expansion to quadratic terms. For a function  $f(x, y)$  and  $g(x, y)$ , the Taylor series expansion up to the second-order term is:

$$f(x + \delta x, y + \delta y) \approx f(x, y) + f_x \delta x + f_y \delta y + \frac{1}{2} (f_{xx} \delta x^2 + 2f_{xy} \delta x \delta y + f_{yy} \delta y^2) \quad (2.174)$$

For Halley's method, we use this expansion to set up the iteration formula:

$$f + f_x \delta x + f_y \delta y + \frac{1}{2} (f_{xx} \delta x^2 + 2f_{xy} \delta x \delta y + f_{yy} \delta y^2) = 0 \quad (2.175)$$

Similarly, for  $g$ :

$$g + g_x \delta x + g_y \delta y + \frac{1}{2} (g_{xx} \delta x^2 + 2g_{xy} \delta x \delta y + g_{yy} \delta y^2) = 0 \quad (2.176)$$

To use Halley's method, we need the second-order partial derivatives of  $f$  and  $g$ . These are:

$$\begin{aligned} f_{xx} &= \frac{\partial^2 f}{\partial x^2} & f_{xy} &= \frac{\partial^2 f}{\partial x \partial y} & g_{yy} &= \frac{\partial^2 g}{\partial y^2} \\ g_{xx} &= \frac{\partial^2 g}{\partial x^2} & g_{xy} &= \frac{\partial^2 g}{\partial x \partial y} & f_{yy} &= \frac{\partial^2 f}{\partial y^2} \end{aligned} \quad (2.177)$$

For instance,  $f_{xx}$  can be approximated using finite differences:

$$f_{xx} = \frac{f(x + \partial x, y) - 2f(x, y) + f(x - \partial x, y)}{\partial x^2} \quad (2.178)$$

Similarly,  $f_{yy}$  is:

$$f_{yy} = \frac{f(x, y + \partial y) - 2f(x, y) + f(x, y - \partial y)}{\partial y^2} \quad (2.179)$$

And the mixed derivative  $f_{xy}$  is:

$$f_{xy} = \frac{f(x + \partial x, y + \partial y) - f(x + \partial x, y) - f(x, y + \partial y) + f(x, y)}{\delta x \delta y} \quad (2.180)$$

The iterative formula for Halley's method becomes:

$$\begin{pmatrix} \delta x \\ \delta y \end{pmatrix} = - \begin{pmatrix} f_x & f_y \\ g_x & g_y \end{pmatrix}^{-1} \left( \begin{pmatrix} f \\ g \end{pmatrix} + \frac{1}{2} \begin{pmatrix} f_{xx}\partial x^2 + 2f_{xy}\delta x\delta y + f_{yy}\partial y^2 \\ g_{yy}\partial x^2 + 2g_{xy}\delta x\delta y + g_{yy}\partial y^2 \end{pmatrix} \right) \quad (2.181)$$

This formula ensures that the corrections  $\delta x$  and  $\delta y$  consider the curvature of the functions  $f$  and  $g$ , leading to faster and more accurate convergence.

Halley's method offers several advantages over the Newton-Raphson method:

- **Cubic Convergence:** The method converges more rapidly, with the error term decreasing as the cube of the previous error term.
- **Improved Accuracy:** By incorporating second-order derivatives, Halley's method provides more accurate corrections at each iteration.
- **Stability:** The method is more stable for functions with significant curvature, as the second-order terms help in navigating the function landscape more effectively.

Gooding say that; “when implementing Halley’s method, careful selection of the increments  $\delta x$  and  $\delta y$  crucial for numerical stability. These increments should be large enough to avoid significant rounding errors but small enough to maintain accuracy. Common choices are  $\delta x = 0.001$  and  $\delta y = 0.001$  but these values may need adjustment based on the specific problem.

Additional strategies for enhancing numerical stability include:

- Reorienting the axis system if corrections lead to non-positive values, ensuring that increments result in positive corrections and avoid overshooting the solution.
- Using adaptive step sizes that adjust dynamically based on the convergence behavior.”

The Gooding method combines the classical Newton-Raphson approach with improvements for higher-order derivatives and numerical stability. By iteratively improving the  $x$  and  $y$  estimates and using efficient techniques to derive partial derivatives, the method provides robust and accurate solutions to the Lambert problem. The inclusion of second-order derivatives and adaptive step adjustments further increases the reliability and convergence speed of the method.

This comprehensive approach allows the Gooding method to effectively address complex astrodynamic problems and provides accurate solutions for orbit calculations and state vector propagation. The careful treatment of partial and second-order derivatives and the combination of strategic numerical stability measures make the method a powerful tool for precise and efficient computation in celestial mechanics.

In any angles-only procedure, there’s a significant degree of arbitrariness in both initiating and terminating the iteration process. This is managed through the starting estimates for  $\rho_1$  and  $\rho_3$  (denoted  $x$  and  $y$ ), and through the specific convergence test used.

It’s essential to set starting values that allow various options to be covered. These values can be based on assumed knowledge of the orbit type or use default values. In my procedure, Gooding sets  $\rho_1$  and  $\rho_3$  to  $2(R_{site_1} + R_{site_2} + R_{site_3})$  to ensure observer-

independence.

The convergence test involves two aspects: the quantity tested and the numerical criterion value. A practical choice is the non-dimensional  $\frac{f}{c}$ , when  $c$  is scaling factor. However, this can be sensitive when  $c$  is very small. Therefore, Gooding uses the calculated version of  $\epsilon_2$  and set a conservative criterion value of  $10^{-12}$ , aligned with IEEE standard double-precision accuracy.



## CHAPTER 3

### DEEP LEARNING

In the past, the majority of software applications could be characterized as complex or precise formations of languages and rules that were supposed to be followed by the machines. Say, we attempt to create an commerce website, we could start with stating some basic principles such as users are able to engage with the application through web or mobile interfaces, the application uses a commercial-grade database for handling user transaction and users' data or the central business logic controls the responses of that application under different situations and events. To automatically implement this logic, we would assume all the possible situations and plan the required actions, for instance, if the user selects an item, moving that item to a shopping cart is a required action. Developers would spend countless hours testing and modifying these rules until they accounted for every conceivable nook and cranny so as to achieve absolute software functionality prior to market release. The ability to manually create automated systems comes off as a great accomplishment and achievement of intelligence, yet if such systems function properly, there is usually no need for machine learning of any kind.

Nonetheless, we are aware that there exist tasks that are not easily solved through a coding language. Picture for instance the processes of writing the computer programs from scratch, predicting weather patterns with the use of geographic information and satellite imagery, free text answering, recognizing people in pictures or suggesting products that best matches the preferences of people. Such tasks would be impossible for the best programmers ever because they are complex and interconnected and require too much computation. For example, it takes a lot of effort in image recognition and it is complex thing, even though it is done overs without thinking on conscious

level.

In contrast, machine learning is quite different and it uses algorithms that have allowance for improvement with practice. They get better at accuracy as they handle more and more observational information or even get to interact with different environments. This scenario is pretty much different from ours where we have a fixed architecture and logic of an commerce platform until the developers come along and change it. The self-modifying machine learning algorithms are able to do this a great deal as they constantly acquire new information without the need for additional re-programming.

In this text, we study the basic concepts underlying machine learning, particularly the specifics associated with deep learning. These advanced technologies have already transformed such areas as computer vision, natural language processing, healthcare and genomics. Once we understand these fundamentals, we see how machine learning systems progress over time, and provide solutions in those areas, where classical programming fails.

### **3.1 Fundamental components of AI**

The development and enhancement of models in machine learning is dependent on many interrelated factors that are essential in the machine learning process. Acquisition of knowledge of those factors is important in addressing different problems related to machine learning. These key components include:

1. Data for Learning
2. Models to Transform Data
3. Objective Functions for Evaluation
4. Optimization Algorithms for Training

### **3.1.1 Data for Learning**

Data is the most important pillar for every machine learning project. Without data, the training and building of models becomes impossible. Data comprises of instances, or examples, each characterized by a vector of attributes or features. In supervised learning one of the objectives is to generate the label value or the target value based on these features. For instance, in the case of recognition of images, the features can be considered to be values of the pixels of an image and the label would be the name ascribed to the object represented in that image. In medical datasets, the patient information such as age, blood pressure, medical history could be some of the features, while the label could represent the absence presence of a relevant disease.

In order for machine learning models to interpret the information, specific features must be encoded. The use of fixed-length vectors is popular whereby every example has the same determinable number of features. However, such is not the case with all cases. Text and images, for instance, have a different pixel count and word count, respectively. But for deep learning models, such a “sequence” of images or texts with varying lengths is not a problem at all.

The characteristics and the amount of data that is available considerably determines the results of the judgment that the machine learning model provides. The algorithm has a lean towards improving training because there will be a large model due to data factors but it has to be valuable data that has been cleaned. Very poor data will, however, result in many faulty results and stereotypes in many areas but especially healthcare and criminal areas. In this regard, clean and high-value data must be possessed for the building of reliable models.

### **3.1.2 Models to Transform Data**

In the context of machine learning, a model is defined as a function that maps input data to output predictions. For every input, a particular feature specified by a user is applied according to the learned patterns in the data. The models may be as simple as linear regression or as complex as a neural network. The type of model used is determined by the problem at hand. Straightforward tasks are achieved with simpler

models; in other instances, advanced models perform more complex tasks like image and speech recognition.

Deep learning models, which have multiple layers of transformation, are better suited to work with deeper data. These are capable of learning deep representation and meanings, hence are applicable in natural language and computer vision tasks. Simpler models, as may be the case, still can be applied to a variety of problems and act as a basis of comprehension of more sophisticated methods.

### **3.1.3 Objective Functions for Evaluation**

Other terms used for objective functions are loss functions and in machine learning models, they serve as an indication of the performance of the model. They determine the degree to which the predictions made by the model and the actual values correspond. The purpose of training a model is thus to reduce the value of the objective function nothing more, nothing less, because by reducing the objective function the correctness of the model is improved.

Parameter optimization is essential in order to determine appropriate losses. Cross entropy loss is often used for regression problems which means that the model outputs probabilities. Whereas for regression tasks, the goal is to predict a continuous value, the mean squared error is a popular option. The set task of cross entropy loss tends to predict categorical outputs but different transcriptions require different objective functions.

### **3.1.4 Optimization Algorithms for Training**

An optimization algorithm modifies a model's parameters with the goal of minimizing the objective function. One of the most widely employed optimization strategy in machine learning is gradient descent. In particular, the gradient descent approach seeks loss minimization by iteratively adjusting the model parameters through first order derivatives of the loss function with respect to the model parameters.

However, variations of gradient descent are widely applicable, for instance, in stochas-

tic gradient descent (SGD), the estimates of the parameters are done for each sample while in mini-batch the updates are-- as the name suggest-- done using mini sets of data (mini-batches). They are useful in ascertaining the best parameters estimates under large data samples.

Optimization methods like Adam and RMS prop incorporate adaptive learning rates and momentum and have shown to improve convergence rate and increase performance. The choice of the optimization algorithm is also paramount in ensuring that models with good performance are trained as its this selection that determines how fast and accurately learning will occur.

Moreover, machine learning engineers can create reasonable models for many tasks and domains by leveraging these fundamental components together— data, models, objectives and optimizations.

## **3.2 Types of Machine Learning Problems**

The field of machine learning consists of multiple types of problems, each of which has different solutions or approaches. In this unit, we will consider different types of machine learning problems, their properties and areas of application. Emphasis \$ will be on supervised learning, which is further divided into its sub types regression and classification as well sequence learning.

### **3.2.1 Supervised Learning**

Supervised learning is an important term in machine intelligence where models are taught using dataset with input features and their labels. The core objective is to create a model that is able to estimate the labels for new & not seen inputs with sufficient accuracy. This approach has emerged as “supervised” because the models are trained with guidance specific to the existing dataset. The reason why supervised learning is often used in business is because it can be applied to many tasks:

- Medical Diagnosis: Determining whether a patient has a certain disease based

on diagnostic tests.

- **Language Translation:** Translating text from one language to another.
- **Stock Price Prediction:** Predicting future stock prices based on historical financial data.

The process of supervised learning has always had a few procedures that were standardized. The first step is gathering enough height of an amplitude meter shaft of a few key words in the database. The model is built by entering the features alongside with their respective labels into the supervised learning algorithm. The algorithm makes prediction based on the relationship between the features and the consonant nuclei - in other words, between the inputs and outputs. The learned features are eventually used to make predictions based on new features with the constructed model.

The above process demonstrates this process; trading binary options entails the buying of both call and put options in a single transaction. Once the model has been built, a put option can be placed into the market to decode other new inputs. That is the complementarity of the supervised learning cycle and such action augurs well for more natural techniques like regression and classification.

### **3.2.2 Regression**

Regression is described as a supervised learning technique that seeks to estimate future events based on past events. Unlike in a classification problem, where the prediction can only fall within a set number of outcomes, the target variable in a regression problem can take any value within a defined range. Some instances of regression problems are:

- **Real Estate Valuation:** Estimating the market value of properties based on features such as the home's square footage, number of rooms, and location.
- **Weather Forecasting:** Predicting future temperatures, precipitation, or wind speeds.

- Health Analytics: Predicting the length of hospital stays for patients based on their medical profiles.

The predictive purposes associated with the regression models tend to revolve around indicating the differences between the projected values and the actual values. To achieve this objective, a number of relevant loss functions are deployed with the mean square error (MSE) being one of them. This approach facilitates the understanding of the interactions between the input features and the continuous target variable, hence making it possible for the model to be able to predict unseen future data accurately.

Regression can also be exemplified by home price prediction. Such data set would potentially include a target variable in the form of the sales price, and a number, or set, of other related variables including the size of the house, the number of rooms and the geographical area where the house is located. By employing the regression model on this dataset, we can make predictions about the selling prices of new houses based on the variables. This is particularly useful in real estate as different pricing strategies can be employed when buying or selling houses.

### **3.2.3 Classification**

Classification is another key deficiency of supervised learning in which the objective is to predict categorical variables rather than real-valued outputs. In classification problems, the model assigns each input to one of the predefined categories. Examples of classification tasks include:

- Image Recognition: Identifying objects or people in photos or videos.
- Sentiment Analysis: Determining whether a text expresses positive, negative, or neutral sentiment.
- Medical Imaging: Diagnosing conditions by classifying medical images, for example identifying tumors on x-rays.

Classification models normally estimate each possible category in terms of the probability distributions over them as an output. The approach also helps the model in

dealing with the vague or noisy input. For instance, a classifier can output a set of likely probabilities assigned to a range of classes and it is then up to the decision-maker to determine in which direction to go.

A typical classification problem is the classification of spam emails. This model was trained on dataset consisting of labeled emails which were classified as either spam or not spam. This learning allows a model to determine the features inherent in each set and as a result classify a new email and enable the defection of unwanted email that is off each set through filtering. This functionality is highly common in email systems in order to enhance the user interactivity while reducing spams.

### **3.2.4 Sequence Learning**

Sequence Learning encompasses models that take in data where elements cannot be treated in isolation but as part of a whole with a specific order of importance. Unlike the cases of the models with fixed input and output, sequence learning has to consider the time or sequential lag between the data. Examples of sequence learning problems are as follows:

- **Speech Recognition:** Processing audio recordings to convert spoken language into written text.
- **Machine Translation:** Preserving grammatical and contextual accuracy while translating sentences from one language to another.
- **Patient Monitoring:** Continuously assessing a patient's health status based on a set of medical measurements over time.

The difficulties presented by the problems are mainly because they have to maintain the context in translations that are of different lengths. To address these issues, Time Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks are commonly applied because these networks archive connections through time steps.

In sequence learning, models often work with input and output sequences of various lengths, such as in speech recognition where the audio input can vary in length and

so does the text output. This intricacy makes it necessary for models that work with changing length sequences, while maintaining the context of the data throughout time.

A specific kind of sequence learning, referred to as sequence-pair sequence learning, deals with situations in which both inputs and outputs are sequences. It includes translating from one language to another or transforming spoken language into writing. Such processes imply non-trivial alignments between input and output sequences and, as such, the problems of their adequate transformations are also quite complex.

### **3.2.5 Unsupervised and Self-Supervised Learning**

In supervised learning, the model interfaces with a well-defined dataset that contains both the features and the corresponding labels. In this case, it is better to envision having a very competent and strict manager who in every situation is ready to provide detailed instructions. This makes the task relatively easy since all that the learner is required to do is to identify and imitate the patterns dictated by the manager.

On the other hand, working for a boss who isn't sure what he's looking for is much more daunting and difficult, and it's the scenario that you would experience with unsupervised learning. In unsupervised learning, the model is trained on a set of data that is vast but lacks specific pointers or labels. The objective of the model becomes to find congested data christened as noise or to filter information congested within the data set. This type of learning can be very broad based so that the only limits to one's questioning are the imagination of the data scientist. For instance, unsupervised learning is often applied for clustering, which is a common learning task that seeks to split the data into appropriate clusters based on the input similarity of different data points. This could include tasks such as sorting of images and theme categorization, as well as organizing users according to their web surfing profiles.

Subspace estimation, which consists of determining the best set of a small number of parameters that encompass the most important characteristics of the data, is another important aspect of unsupervised learning. In more detail, one can think that a ball can be characterized by its speed, the diameter of the ball and the mass of the ball. In the same line, tailors consider several measurements of a human body shape in

order to make properly fitted clothes. A popular technique for subspace estimation is principal component analysis (PCA) which is a linear technique and is concerned with data compression by means of the conservation of variance.

Moreover, the use or the architecture of the neural network can also be customized to learn the relationships between the objects in a Euclidean space, such as cities and countries (e.g., be expected that the word “Paris” will be closer to the word “France” than other cities and countries words, and similarly “Rome” will be located nearby “Italy”). This can be extended to more complex relationships that can provide the reason for such a relationship. Because the relationship exists there is always questions as to the bi modal and causal relationship between the variables.

### **3.2.5.1 Interacting with an Environment**

In contrast to supervised and unsupervised learning, which are primarily conducted offline, interaction with an environment incorporates an added feature to the entire process of learning. Learning offline is done by fetching a huge dataset and building the model without any further contact with the environment. Such a method makes it relatively easier to learn how to recognize patterns because this learning is not coupled with environmental interactions. This automation of the learning process however has its disadvantages since, in the case of intelligent agents, which need to operate in real time with a number of decisions and actions, such an approach may be impossible.

Taking into account the interaction between a model that has gone through machine learning and a specified environment, the model will be taking into account the effects of its actions on future observations. The diagram that has been provided illustrates this interaction model. As an example, an intelligent agent is said not only to make predictions but in some cases communicates with the environment and takes actions towards it. Such actions may turn out to have lasting effects that require the agent to know and predict such effects.

An environment brings about several questions that are considered vital to understanding the nature of the environment. For example, does the environment have a memory of what previous actions were undertaken? Is the environment competitive

or is it of a more cooperative nature? Are there more dynamics that change with time? These questions portray the notion of distributional shift that has been defined as the difference between the training data and the real world phenomena. Such externalities are crucial when considering effective and efficient machine learning systems.

Considering these interrelations is important for the creation of intelligent agents that can function in environments characterized by dynamic changes and uncertainty. The shift from exclusively predictive frameworks to agents who actively take actions necessitates and renders new opportunities and challenges in the ML frontier and emphasizes the requirement of techniques which are able to support decision making and adjustment in real time.

### **3.2.6 Reinforcement Learning**

Reinforcement learning (RL) can be described as a learning framework in which agents are trained through the experience gained from the consequences of their actions within the context. Different from supervised learning, which is based on a given data set, in RL an agent is active; they make decisions and are rewarded or punished according to their decisions. This repeated work enables the agent to find the best ways for fulfilling particular objectives.

In reinforcement learning, an agent takes actions in the environment for a certain number of times. Each time the agent receives an indication from the environment, the agent gets to decide an action to take, and based upon the results of the action approximately that agent is rewarded. This repetition goes on and the agent's objective is now to increase the overall returns over a period. This is done by a policy which states what action is to be taken following a certain observation. The target of RL is to estimate the policy which will produce the highest reward.

One of the main weaknesses of many approaches to learning is the wide range of aspects they cover, this however is one of the main merits reinforcement learning has. For the reinforcement learning technique can be utilized in the development of a broad area of tasks, from the teaching of robots to perform intricate actions to making AI systems for games.

To illustrate, for instance, the development of human-level performance on Atari games using DQNs, as well as AlphaGo's win in the board game Go, indicate the achievements of RL in tasks that require strategic decisions.

Supervised learning is, at times, inadequate, such as in the case of RL, which incorporates its own set of problems. In RL, for every observation, an explicit label is not provided. Instead, the agent struggles to learn which actions to take from the perceived reward(s). Credit assignment problem which refers to the case where an agent needs to identify the actors responsible for achieving the eventual outcomes must also be resolved by RL. Also, RL must consider the situations of partial observations where an agent has to make decisions under conditions of uncertainty and limited information about the environment.

### **3.3 Linear Regression**

Whenever we have to foresee a value in the numerical form, we are faced with regression problems. Consider examples such as the estimating the price of a house, the amount of time a patient will spend caring in a hospital or the demand for retail stores. This is because all of these try to estimate a measurable quantity, and thus are regression problems. In contrast, there are classification problems, that will be considered later, whose aim is to determine the existence of an object in a particular class or category.

Let us consider the example of predicting the price of a house considering the features such as the area measured in square feet and the age of the house in years. To be able to build such a predictive model, we require a data set that captures, for different homes, the selling price, area and age. This is referred to as a training set, and each piece of data is called an example or sample. The predicted variable is called as a label or target, whereas the predicting variables (area and age) are said to be features or covariates.

### 3.3.1 Basics

Linear regression is one of the simplest and most widely used methods for solving regression problems. This method dates back to the early 19th century (Gauss, 1809; Legendre, 1805) and is based on the assumption that the relationship between features and the target is approximately linear. In other words, we assume that the conditional mean can be expressed as the weighted sum of the features  $E = [Y \mid X = x]$ . This model also assumes that deviations from the expected value due to observation noise follow a Gaussian distribution.

Usually,  $n$  indicates the number of samples in our dataset. Superscripts are used to number samples and targets, and subscripts are used to index coordinates. For example,  $x^i$  represents the  $i^{th}$  sample, and  $x_j^i$  denotes its  $j^{th}$  coordinate.

### 3.3.2 Model

A model serves as the foundation for translating features into target predictions. In linear regression, we assume that the expected value of the target (house price) can be expressed as a weighted sum of the features (area and age):

$$price = w_{area} \cdot area + w_{age} \cdot age + b \quad (3.1)$$

Here,  $w_{area}$  and  $w_{age}$  are weights that determine the influence of each feature on the prediction, and  $b$  is the bias term. The bias term allows us to represent linear functions of the features without restricting the model to lines through the origin. Although there are no newly constructed houses with zero area, the bias term is necessary to express all possible linear functions of the features.

When dealing with high-dimensional datasets, it is more convenient to use linear algebra notation. For  $d$  features, the prediction  $\hat{y}$  can be compactly expressed as:

$$\hat{y} = w_1 x_1 + \dots + w_d x_d + b \quad (3.2)$$

We can further simplify this by collecting all features into a vector  $\mathbf{x} \in \mathbb{R}^d$  and all weights into a vector  $\mathbf{w} \in \mathbb{R}^d$ :

$$\hat{y} = \mathbf{w}^T \mathbf{x} + b \quad (3.3)$$

In this equation,  $\mathbf{x}$  represents the features of a single example. To handle multiple examples, we use the design matrix  $\mathbf{X} \in \mathbb{R}^d$ , where  $\mathbf{X}$  contains one row for each example and one column for each feature. For a collection of features  $\mathbf{X}$ , the predictions  $\hat{\mathbf{y}} \in \mathbb{R}^d$  can be written as:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} + b \quad (3.4)$$

with broadcasting applied during summation.

Given the features  $\mathbf{X}$  and known labels  $\mathbf{y}$  in a training dataset, the goal of linear regression is to find the weight vector  $\mathbf{w}$  and bias term  $b$  that minimize the prediction error for new data samples drawn from the same distribution as  $\mathbf{X}$ .

Even if the true relationship between  $\mathbf{y}$  and  $\mathbf{x}$  is linear, real-world datasets often contain noise and measurement errors. Therefore, the observed value  $y^i$  might not exactly equal  $\mathbf{w}^T \mathbf{x}^i + b$  for every  $i$ . To account for these errors, we include a noise term in our model.

Before we can optimize the model parameters  $\mathbf{w}$  and  $b$ , we need two essential components:

- A measure of the model's quality,
- A procedure to update the model and improve its performance.

### 3.3.3 Loss Function

To effectively fit a model to the data, we need a way to measure how well the model's predictions match the actual values. This is achieved using loss functions, which

quantify the difference between the predicted values and the true target values. In regression problems, the most commonly used loss function is the squared error. For a given prediction  $\hat{y}^{(i)}$  and the true label  $y^{(i)}$ , the squared error is defined as:

$$l^i(\mathbf{w}, b) = \frac{1}{2} (\hat{y}^{(i)} - y^{(i)})^2 \quad (3.5)$$

The factor  $\frac{1}{2}$  is included for convenience, as it simplifies the derivative of the loss function. Since the training dataset is fixed, the empirical error depends solely on the model parameters. Figure 3.1 illustrates how a linear regression model fits a one-dimensional dataset.

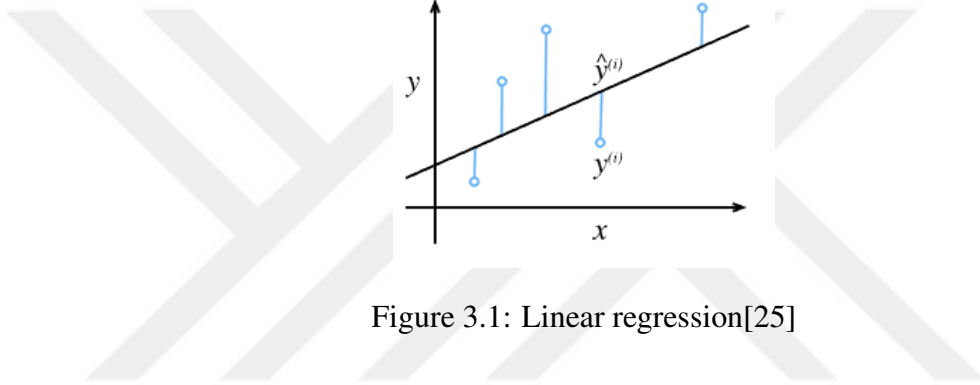


Figure 3.1: Linear regression[25]

Notice that large discrepancies between predictions  $\hat{y}^{(i)}$  and true values  $y^{(i)}$  lead to significantly higher loss values due to the quadratic nature of the loss function. This feature encourages the model to avoid large errors, but it can also make it sensitive to outliers. To evaluate the performance of the model on the entire dataset of  $n$  samples, we need to average (or sum) the losses for all training examples:

$$L(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^n l^{(i)}(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} (\mathbf{w}^T \mathbf{x}^{(i)} + b - y^{(i)})^2 \quad (3.6)$$

During training, our objective is to find the parameters  $\mathbf{w}^*$  and  $b^*$  that minimize the total loss across all training examples:

$$\mathbf{w}^*, b^* = \operatorname{argmin}_{\mathbf{w}, b} L(\mathbf{w}, b) \quad (3.7)$$

### 3.3.4 Analytic Solution

Linear regression offers a unique case where the optimization problem can be solved analytically. This involves determining the optimal parameters using a straightforward formula. To simplify calculations, the bias term  $b$  is incorporated into the weight vector  $\mathbf{w}$  by adding a column of ones to the design matrix  $\mathbf{X}$ . This reformulates the optimization problem as minimizing  $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$ . Provided that the design matrix  $\mathbf{X}$  has full rank, meaning no feature is linearly dependent on another, there will be a unique critical point that minimizes the loss across the entire domain.

We start by expressing the model as:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} + b \quad (3.8)$$

To integrate the bias term  $b$  into the parameter vector, we can rewrite this equation as:

$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{X}\mathbf{w} + \mathbf{I}b \\ &= \mathbf{X}\mathbf{w} + \mathbf{X}\mathbf{X}^{-1}b \\ &= \mathbf{X}(\mathbf{w} + \mathbf{X}^{-1}b) \end{aligned} \quad (3.9)$$

and redefining  $(\mathbf{w} + \mathbf{X}^{-1}b) \rightarrow \mathbf{w}$ , we obtain:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} \quad (3.10)$$

However, this assumes that  $\mathbf{X}$  is known and invertible, which is not always the case. Thus, a more detailed approach is required:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} + b \quad (3.11)$$

In expanded form, this looks like:

$$\begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{pmatrix} = \begin{pmatrix} X_{11} & X_{12} & \dots & X_{1d} \\ X_{21} & X_{22} & \dots & X_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \dots & X_{nd} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{pmatrix} + b \quad (3.12)$$

Since the bias term  $b$  cannot be simply added as a scalar, we express it as a vector:

$$\begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{pmatrix} = \begin{pmatrix} X_{11} & X_{12} & \dots & X_{1d} \\ X_{21} & X_{22} & \dots & X_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \dots & X_{nd} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{pmatrix} + \begin{pmatrix} b \\ b \\ \vdots \\ b \end{pmatrix} \quad (3.13)$$

This can be further simplified as:

$$\begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{pmatrix} = \begin{pmatrix} X_{11} & X_{12} & \dots & X_{1d} & 1 \\ X_{21} & X_{22} & \dots & X_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ X_{n1} & X_{n2} & \dots & X_{nd} & 1 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \\ b \end{pmatrix} \quad (3.14)$$

By redefining the design matrix  $\mathbf{X} \in \mathbb{R}^{n \times (d+1)}$  and the weight vector  $\mathbf{w} \in \mathbb{R}^{(d+1)}$ , we get:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} \quad (3.15)$$

The loss function for this model can be defined as:

$$L = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 = \frac{1}{2} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \quad (3.16)$$

Taking the derivative with respect to  $\mathbf{w}$ :

$$\frac{\partial L}{\partial w_i} = \frac{1}{2} (\mathbf{y} - \mathbf{X}\mathbf{w})^T \frac{\partial}{\partial w_i} (\mathbf{y} - \mathbf{X}\mathbf{w}) + \frac{1}{2} \left[ \frac{\partial}{\partial w_i} (\mathbf{y} - \mathbf{X}\mathbf{w})^T \right] (\mathbf{y} - \mathbf{X}\mathbf{w}) \quad (3.17)$$

This simplifies to:

$$\frac{\partial L}{\partial w_i} = -\frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T \frac{\partial}{\partial w_i}(\mathbf{X}\mathbf{w}) - \frac{1}{2} \frac{\partial}{\partial w_i}(\mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \quad (3.18)$$

Further simplifying:

$$\frac{\partial L}{\partial w_i} = -\frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T \begin{pmatrix} X_{1i} \\ X_{2i} \\ \vdots \\ X_{ni} \end{pmatrix} - \frac{1}{2} \begin{pmatrix} X_{1i} & X_{2i} & \dots & X_{ni} \end{pmatrix}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \quad (3.19)$$

Thus:

$$\frac{\partial L}{\partial w_i} = - \begin{pmatrix} X_{1i} & X_{2i} & \dots & X_{ni} \end{pmatrix}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \quad (3.20)$$

where  $i = 1, \dots, (d+1)$ . To minimize the loss, we set the derivatives to zero:

$$\frac{\partial L}{\partial w_i} = 0 = - \begin{pmatrix} X_{1i} & X_{2i} & \dots & X_{ni} \end{pmatrix}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \quad (3.21)$$

In matrix form, this becomes:

$$\mathbf{0} = \mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \quad (3.22)$$

$$\mathbf{0} = \mathbf{X}^T \mathbf{y} - \mathbf{X}^T \mathbf{X} \mathbf{w} \quad (3.23)$$

$$(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \mathbf{w} \quad (3.24)$$

Solving for  $\mathbf{w}$ :

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (3.25)$$

Solution to the same problem using index notation:

$$L = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 = \frac{1}{2} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \quad (3.26)$$

$$L = \frac{1}{2} (y_i - X_{ik}w_k) (y_i - X_{ij}w_j) \quad (3.27)$$

which can be expanded to:

$$L = \frac{1}{2} \sum_i \left( y_i - \sum_k X_{ik}w_k \right) \left( y_i - \sum_j X_{ij}w_j \right) \quad (3.28)$$

Taking the derivative with respect to  $w_l$ :

$$\frac{\partial L}{\partial w_l} = \frac{1}{2} \frac{\partial}{\partial w_l} [(y_i - X_{ik}w_k) (y_i - X_{ij}w_j)] \quad (3.29)$$

$$\frac{\partial L}{\partial w_l} = \frac{1}{2} (y_i - X_{ij}w_j) \frac{\partial}{\partial w_l} (y_i - X_{ik}w_k) + \frac{1}{2} (y_i - X_{ik}w_k) \frac{\partial}{\partial w_l} (y_i - X_{ij}w_j) \quad (3.30)$$

$$\frac{\partial L}{\partial w_l} = -\frac{1}{2} (y_i - X_{ij}w_j) X_{ik} \frac{\partial}{\partial w_l} w_k - \frac{1}{2} (y_i - X_{ik}w_k) X_{ij} \frac{\partial}{\partial w_l} w_j \quad (3.31)$$

$$\frac{\partial L}{\partial w_l} = -\frac{1}{2} (y_i - X_{ij}w_j) X_{ik} \delta_{lk} - \frac{1}{2} (y_i - X_{ik}w_k) X_{ij} \delta_{lj} \quad (3.32)$$

Since  $\frac{\partial}{\partial w_l} w_k = \delta_{lk}$  (the Kronecker delta), the derivatives simplify to:

$$\frac{\partial L}{\partial w_l} = -\frac{1}{2} (y_i - X_{ij}w_j) X_{il} - \frac{1}{2} (y_i - X_{ik}w_k) X_{il} \quad (3.33)$$

$$= -\frac{1}{2} (X_{il}y_i - X_{il}X_{ij}w_j) - \frac{1}{2} (X_{il}y_i - X_{il}X_{ik}w_k) \quad (3.34)$$

changing the dummy repeating index:

$$\frac{\partial L}{\partial w_l} = -\frac{1}{2} (X_{il}y_i - X_{il}X_{ij}w_j) - \frac{1}{2} (X_{il}y_i - X_{il}X_{ij}w_j) = -(X_{il}y_i - X_{il}X_{ij}w_j) \quad (3.35)$$

and equating to zero:

$$0 = -(X_{il}y_i - X_{il}X_{ij}w_j) \quad (3.36)$$

$$0 = X_{il}y_i - X_{il}X_{ij}w_j \quad (3.37)$$

and solving for  $w_j$  yields:

$$w_j = (X_{il}X_{ij})^{-1} X_{il}y_i \Rightarrow \mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (3.38)$$

### 3.3.5 Minibatch Stochastic Gradient Descent

In many deep learning tasks, finding an exact analytical solution for model training is often impractical. However, iterative optimization techniques, such as gradient descent, can effectively train models.

**Gradient Descent:** The simplest form involves calculating the loss function's derivative, aggregating losses from all examples in the dataset. This method, while accurate, is computationally expensive (Liu and Nocedal, 1989).

**Stochastic Gradient Descent (SGD):** Instead of using the entire dataset, SGD updates the model parameters using a single example at a time, making it computationally more efficient (Bottou, 2010). However, processing one sample at a time can be slow and less efficient in utilizing computational resources.

Minibatch Stochastic Gradient Descent (Minibatch SGD): This technique strikes a balance by using small subsets of data (minibatches) for updates, typically ranging from 32 to 256 samples (Li et al., 2014). Each iteration  $t$  involves:

1. Randomly sampling a minibatch  $\beta_t$  of size  $|\beta|$
2. Computing the gradient of the average loss on the minibatch.
3. Updating parameters using the gradient scaled by a learning rate  $\eta$ :

$$(\mathbf{w}, b) \leftarrow (\mathbf{w}, b) - \frac{\eta}{|\beta|} \sum_{i \in \beta_t} \partial_{(\mathbf{w}, b)} l^{(i)}(\mathbf{w}, b) \quad (3.39)$$

Minibatch size and learning rate are hyperparameters, often tuned using techniques like Bayesian optimization (Frazier, 2018). This method balances computational efficiency and frequent updates, making it effective for deep learning.

### 3.3.6 Predictions

Using the trained model's expression,  $\mathbf{w}^T \mathbf{x} + b$  we can make predictions on new data. For example, we can predict the price of a house based on its area  $x_1$  and age  $x_2$ . Although this process is commonly known as "inference" in deep learning, we prefer to call it "estimation" to avoid confusion with parameter estimation in statistics.

### 3.3.7 The Normal Distribution and Squared Loss

The squared loss objective function can be formally motivated by assuming that noise follows a normal distribution. Linear regression, historically debated between Gauss and Legendre, shares a deep connection with the normal distribution.

A normal distribution with mean  $\mu$  and variance  $\sigma^2$  is given by:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} (x - \mu)^2\right) \quad (3.40)$$

Assuming noisy observations where  $\epsilon \sim \mathcal{N}(0, \sigma^2)$ :

$$y = \hat{y} = \mathbf{w}^T \mathbf{x} + b + \epsilon \quad (3.41)$$

The likelihood of  $y$  for  $x$  is:

$$P(y | \mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} (y - \mathbf{w}^T \mathbf{x} - b)^2\right) \quad (3.42)$$

Maximizing the likelihood for the dataset  $P(y | \mathbf{X})$  by minimizing the negative log-likelihood leads to:

$$-\log P(y | \mathbf{X}) = \sum_{i=1}^n \frac{1}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)} - b)^2 \quad (3.43)$$

Ignoring constants, this simplifies to minimizing the squared error, making it equivalent to maximum likelihood estimation under Gaussian noise.

### 3.3.8 Linear Regression as a Neural Network

Although linear models lack the complexity to represent the sophisticated networks discussed later in this book, they can still be interpreted as a form of neural network. In particular, linear models can be seen as neural networks where each feature is an input neuron, all directly connected to a single output neuron.

Figure 3.2 illustrates this concept by depicting linear regression as a neural network. The diagram shows the pattern of connectivity, with each input neuron linked to the output neuron, without specifying the weights or biases' values.

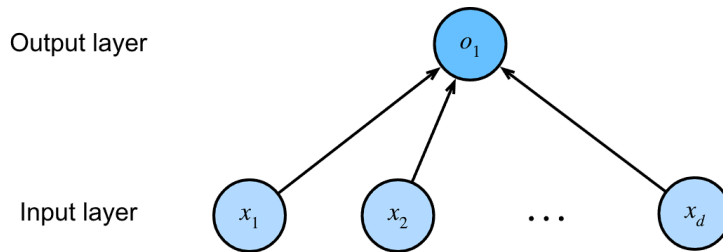


Figure 3.2: Linear regression constitutes a neural network with only one layer.[25]

In this setup, the inputs are  $x_1, \dots, x_d$  where  $d$  denotes the number of inputs or the feature dimensionality in the input layer. The network's output is  $o_1$  the predicted numerical value. Since our objective is to predict a single value, there is only one output neuron. The input values are given directly, and only the output neuron's value is computed.

Thus, linear regression can be viewed as a single-layer fully connected neural network. In the upcoming chapters, we will explore networks with many more layers, increasing their complexity and capability.

### 3.3.9 Biology

Although linear regression predates the emergence of computational neuroscience, understanding its relation to neural networks is enriched by examining biological neurons. Early models of artificial neurons by Warren McCulloch and Walter Pitts were directly inspired by the structure and function of real neurons.

Figure fig:Real-neuron-(source: illustrates a simplified biological neuron. The neuron comprises dendrites (input terminals), the nucleus (central processing unit), the axon (output conduit), and axon terminals (output points), connecting to other neurons through synapses.

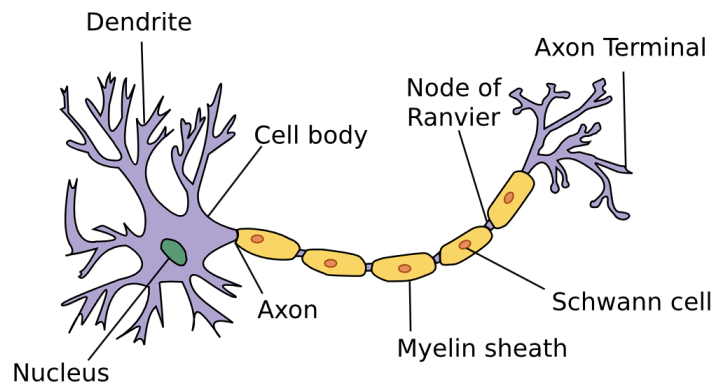


Figure 3.3: Real neuron (source: SEER Program by the US National Cancer Institute).

Inputs  $x_i$  from other neurons or environmental sensors are received by the dendrites.

These inputs are weighted by synaptic weights  $w_i$ , which determine the impact of each input via the product  $x_i w_i$ . The weighted inputs from multiple sources are aggregated in the nucleus as a weighted sum  $y = \sum_i x_i w_i + b$ , potentially undergoing nonlinear transformation through a function  $\sigma(y)$ . This processed information travels along the axon to the axon terminals, where it is either transmitted to another neuron or activates an effector like a muscle.

This high-level understanding shows that combining many such units, with appropriate connectivity and learning mechanisms, can lead to behaviors far more sophisticated than any single neuron could produce. While the study of biological neural systems provided the initial inspiration, contemporary deep learning research is influenced by a broader array of disciplines. As Russell and Norvig (2016) note, just as modern aeronautics, though inspired by birds, now draws on diverse fields for innovation, deep learning integrates insights from mathematics, linguistics, psychology, statistics, and computer science.

### 3.3.10 Linear Model

To determine conditional probabilities for multiple classes, we need a model that has multiple outputs, one for each class. When classifying using linear models, we need an affine function for each output class. Theoretically, we could use the last class as the difference of the sums of the other classes and then use an affine function, but for symmetry and simplicity, we usually prefer to use a few more parameters. So, each output has its own affine function.

Given 4 features and 3 possible output categories, we need 12 weight parameters (denoted as  $w$  with subscripts) and 3 bias parameters (denoted as  $b$  with subscripts). This results in the following equations:

$$o_1 = x_1 w_{11} + x_2 w_{12} + x_3 w_{13} + x_4 w_{14} + b_1 \quad (3.44)$$

$$o_2 = x_1 w_{21} + x_2 w_{22} + x_3 w_{23} + x_4 w_{24} + b_2 \quad (3.45)$$

$$o_3 = x_1w_{31} + x_2w_{32} + x_3w_{33} + x_4w_{34} + b_3 \quad (3.46)$$

Figure 3.4 shows the corresponding neural network diagram. Similar to linear regression, this setup uses a single layer neural network. Each output is calculated  $o_1$ ,  $o_2$  and  $o_3$  depends on every input  $x_1$ ,  $x_2$ ,  $x_3$  and  $x_4$ , the output layer is also described as a fully connected layer.

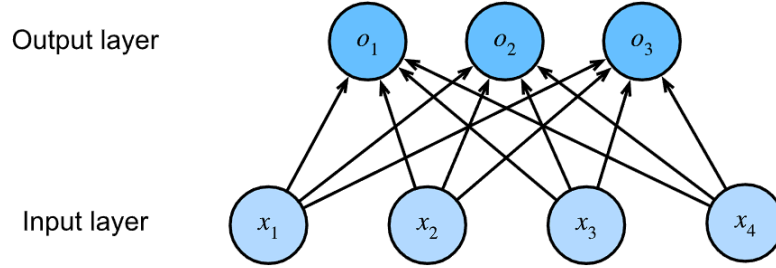


Figure 3.4: Regression is a single-layer neural network.[25]

In a more compact notation using vectors and matrices, we represent the outputs as  $\mathbf{o} = \mathbf{W}\mathbf{x} + \mathbf{b}$ . Here, the weight matrix  $\mathbf{W}$  is a  $3 \times 4$  matrix covering all weights, and the bias vector  $\mathbf{b} \in \mathbb{R}^3$  contains all the biases. This notation is more suitable for both mathematical formulations and code implementation.

### 3.4 Multilayer Perceptrons

Building on the basics of linear models, we now delve into multilayer perceptrons, which offer more complexity and capability. These networks extend beyond simple affine transformations to capture intricate patterns in data.

#### 3.4.1 Hidden Layers

Linear models map inputs directly to outputs through a single transformation, assuming a straightforward relationship. However, real-world data often involves more complex dependencies that linear models cannot capture.

### 3.4.2 Limitations of Linear Models

Linear models assume a monotonic relationship between features and output; that is, an increase in a feature will cause the output to either increase or decrease continuously. For example, estimating the probability of loan repayment based on income assumes that higher income will always lead to a higher probability of repayment. However, this relationship is not strictly linear; for example, an increase in income from \$0 to \$50,000 will have a greater impact than an increase from \$1 million to \$1.05 million.

Nonlinear relationships also present challenges. In health prediction based on body temperature, higher temperatures above 37°C increase risk, while lower temperatures below 37°C increase risk. Simple linear models cannot address these complexities.

Image classification, such as distinguishing cats from dogs, highlights the shortcomings of linear models. The importance of the intensity of a pixel is context-dependent, and linear models cannot account for these dependencies. For example, simply increasing the intensity of a pixel cannot reliably predict the presence of a dog or cat.

Deep neural networks solve these problems by learning complex representations using hidden layers. These layers transform the inputs in multiple stages, capturing complex patterns and dependencies that linear models miss. This hierarchical approach allows deep networks to excel at tasks that involve nonlinear and context-dependent features, such as image and speech recognition.

### 3.4.3 Incorporating Hidden Layers

To overcome the limitations of linear models, we can add hidden layers to our neural network. By stacking multiple fully connected layers, where each layer feeds into the next, we create a more powerful architecture known as a multilayer perceptron (MLP). Figure 3.5 illustrates this concept.

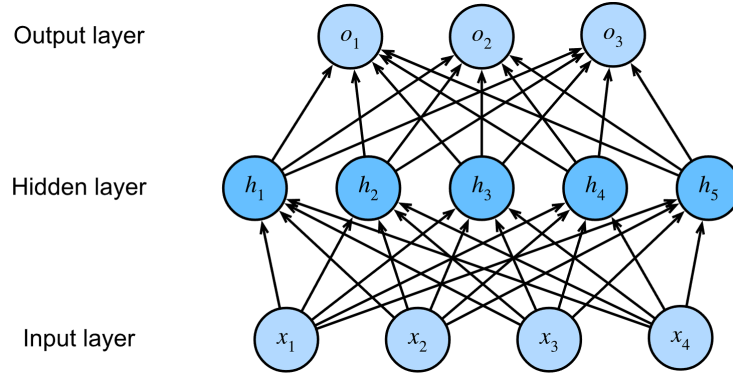


Figure 3.5: Hidden Layer[25]

This MLP has four input neurons, three output neurons, and one hidden layer consisting of five neurons. The input layer passes data to the hidden layer without any computation. The hidden layer processes this data and feeds the outputs to the final output layer. Both the hidden and output layers are fully connected, meaning that every neuron in one layer is connected to every neuron in the next layer.

By adding hidden layers, the network can learn complex patterns and relationships within the data, which a simple linear model cannot capture. This structure allows MLPs to perform effectively on tasks requiring the detection of non-linear and context-dependent patterns, enhancing their ability to handle diverse and complicated datasets.

#### 3.4.4 From Linear to Nonlinear

To model more complex relationships, we move from linear models to models with hidden layers and nonlinear activation functions. Consider a multilayer perceptron (MLP) with one hidden layer and a minibatch of  $n$  samples, each with  $d$  features. Let  $\mathbf{X} \in \mathbb{R}^{n \times d}$  represent the input matrix. The hidden layer, containing  $h$  units, outputs  $\mathbf{H} \in \mathbb{R}^{n \times h}$ .

The weights and biases for the hidden and output layers are as follows:

- Hidden-layer weights:  $\mathbf{W}^{(1)} \in \mathbb{R}^{n \times h}$ .

- Hidden-layer biases:  $\mathbf{b}^{(1)} \in \mathbb{R}^{1 \times h}$ .
- Output-layer weights:  $\mathbf{W}^{(2)} \in \mathbb{R}^{h \times q}$ .
- Output-layer biases:  $\mathbf{b}^{(2)} \in \mathbb{R}^{1 \times q}$ .

The calculations for the outputs  $\mathbf{O} \in \mathbb{R}^{n \times q}$  of the one-hidden-layer MLP are:

$$\mathbf{H} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)} \quad (3.47)$$

$$\mathbf{H} = \mathbf{X}\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \quad (3.48)$$

Adding hidden layers initially does not introduce nonlinearity, as they merely perform affine transformations. This can be seen by collapsing the hidden layer:

$$\mathbf{O} = (\mathbf{X}\mathbf{W} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)} = \mathbf{X}\mathbf{W} + \mathbf{b} \quad (3.49)$$

where  $\mathbf{W} = \mathbf{W}^{(1)}\mathbf{W}^{(2)}$  and  $\mathbf{b} = \mathbf{b}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}$ .

To fully leverage the power of multilayer architectures, we introduce a nonlinear activation function  $\sigma$  applied to each hidden unit after the affine transformation. A common choice is the ReLU (rectified linear unit) activation function:

$$\sigma(x) = \max(0, x) \quad (3.50)$$

With the activation function, the calculations become:

$$\mathbf{H} = \sigma(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}) \quad (3.51)$$

$$\mathbf{O} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \quad (3.52)$$

Here,  $\sigma$  is applied elementwise, meaning each hidden unit's activation is computed independently of others. This prevents the collapse into a linear model and allows the network to learn complex, nonlinear relationships.

By stacking multiple hidden layers, we build deeper and more expressive MLPs. For instance, with two hidden layers, the calculations are:

$$\mathbf{H}^{(1)} = \sigma_1 \left( \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)} \right) \quad (3.53)$$

$$\mathbf{H}^{(2)} = \sigma_2 \left( \mathbf{H}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \right) \quad (3.54)$$

$$\mathbf{O} = \mathbf{H}^{(2)}\mathbf{W}^{(3)} + \mathbf{b}^{(3)} \quad (3.55)$$

This layered approach allows MLPs to model increasingly complex patterns and dependencies in data, enabling them to tackle a wide range of challenging tasks effectively.

### 3.4.5 Relu Activation Function

ReLU (Rectified Linear Unit) activation function is widely used in deep learning models due to its simplicity and effectiveness. Mathematically, ReLU is defined as:

$$f(x) = \max(0, x) \quad (3.56)$$

This function outputs the input value directly if it is positive; otherwise, it outputs zero. The behavior of ReLU can be described in two main cases:

1. When  $x \geq 0$ , the function output is  $x$ , i.e.,  $f(x) = x$
2. When  $x < 0$ , the function output is 0, i.e.,  $f(x) = 0$

The primary advantage of ReLU lies in its simplicity and computational efficiency. By zeroing out negative values and retaining positive ones, ReLU enhances the computational speed and performance of deep learning models. Additionally, unlike Sigmoid and Tanh functions, which can suffer from the vanishing gradient problem due to very small derivatives for large positive or negative inputs, ReLU provides a derivative of 1 for positive inputs, thus mitigating this issue and allowing for faster and more effective learning.

This derivative ensures that positive input values contribute to gradient updates, preventing the gradients from vanishing and facilitating efficient learning in deep networks.

ReLU's straightforward and effective nature makes it one of the most preferred activation functions in modern deep learning applications. Its ability to maintain computational efficiency while addressing the vanishing gradient problem significantly contributes to its popularity in various neural network architectures.

### 3.4.6 Sigmoid Activation Function

The sigmoid activation function is a widely used activation function in neural networks, especially in binary classification problems. Mathematically, the sigmoid function is defined as:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (3.57)$$

This function converts any input value to a value between 0 and 1, which makes it particularly useful for models that require a probabilistic output. To understand the sigmoid function, consider its behavior over different ranges of input values.

- When  $x \rightarrow \infty$ , as  $x$  becomes very large and positive,  $e^{-x}$  approaches 0. Thus  $f(x) \approx \frac{1}{1+0} = 1$ .
- When  $x \rightarrow -\infty$ , as  $x$  becomes very large and negative,  $e^{-x}$  approaches  $+\infty$ . Thus  $f(x) \approx \frac{1}{1+\infty} = 0$ .

- At  $x = 0$ , the value of the sigmoid function is  $f(x) = \frac{1}{1+e^0} = \frac{1}{2} = 0.5$

The sigmoid function's S-shaped curve (or sigmoid curve) smoothly transitions from 0 to 1, with an inflection point at  $x = 0$  where the function value is 0.5.

While the sigmoid function is smooth and differentiable, aiding in gradient-based optimization techniques used in training neural networks, it also has some notable properties. The derivative of the sigmoid function, crucial for backpropagation, is  $f'(x) = f(x) \cdot (1 - f(x))$ . This derivative shows that the gradient of the sigmoid function is highest around  $x = 0$  and decreases for very large positive or negative values of  $x$ .

The sigmoid function outputs values between 0 and 1, which can be interpreted as probabilities, making it suitable for binary classification tasks. However, it can suffer from the vanishing gradient problem, where very large positive or negative input values result in very small gradients, slowing down the learning process in deep networks. Additionally, the outputs are not zero-centered, affecting the dynamics of gradient descent optimization. Despite these limitations, the sigmoid function remains fundamental in neural networks, particularly in the output layer of binary classifiers, due to its probabilistic interpretation and smooth gradient.

### 3.4.7 Tanh Activation Function

The tanh (hyperbolic tangent) activation function is another activation function commonly used in neural networks. Mathematically, the tanh function is defined as:

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3.58)$$

This function maps any input value to a value between  $-1$  and  $1$ , making it useful for models that require outputs centered around zero. To understand the tanh function, consider its behavior with different ranges of input values.

- When  $x \rightarrow \infty$ , as  $x$  becomes very large and positive,  $e^{-x}$  approaches 0. Thus  $f(x) \approx \frac{e^x - 0}{e^x + 0} = 1$ .

- When  $x \rightarrow -\infty$ , as  $x$  becomes very large and negative,  $e^x$  approaches 0. Thus  $f(x) \approx \frac{0 - e^{-x}}{0 + e^x} = -1$ .
- At  $x = 0$ , the value of the tanh function is  $f(x) = \frac{e^0 - e^0}{e^0 + e^0} = \frac{1}{2} = 0.5$

The tanh function's S-shaped curve (or sigmoid-like curve) smoothly transitions from  $-1$  to  $1$ , with an inflection point at  $x = 0$  where the function value is  $0$ .

The tanh function, like the sigmoid function, is smooth and differentiable; these properties aid gradient-based optimization techniques used when training neural networks. The derivative of the tanh function is given by:

$$f'(x) = 1 - \tanh^2(x) \quad (3.59)$$

This derivative indicates that the gradient of the tanh function is highest around  $x = 0$  and decreases for very large positive or negative values of  $x$ . The tanh function has several advantages. It outputs values between  $-1$  and  $1$ , centering the data, which can help with faster convergence in training neural networks. However, like the sigmoid function, the tanh function can suffer from the vanishing gradient problem, where very large positive or negative input values result in very small gradients, slowing down the learning process in deep networks.

Despite these limitations, the tanh function remains a fundamental activation function in neural networks, thanks to its zero-centered outputs and smooth gradient, which continues to be used as a versatile tool in various neural network architectures.

## CHAPTER 4

### APPLICATIONS

#### 4.1 Introduction

Over the past few decades, considerable progress in space technologies coupled with the decrease of space activity costs has enabled more access to space activities, and as a result the number of satellites and other space objects in orbit has grown substantially. There has been an increase in the total amount of traffic in orbits, and therefore an increase in the amount of space traffic collisions due to the growth of launched commercial and scientific satellites, as well as military ones. Therefore, efficient and reliable tracking of space objects, accurate orbit prediction, and reduction of collision probability have become very crucial.

In the processes of determining and monitoring orbits of satellites, data from various sensors like optical measurements or radar data is employed. Initial orbit determination (IOD) methods are among the most important procedures in the process of using this data and solving the orbit determination problem. Classical methods of IOD are based on a number of assumptions and mathematical models but in general can give effective results within a range of operating conditions. Yet, in the modern space capabilities, particularly with the expansion of data, these methods can become difficult in terms of their robustness, accuracy and computational efficiency.

With the development of the artificial intelligence of immense importance were the advancements made in deep learning which brought about new ways of approaching and solving complex and nonlinear problems. Deep learning models are effective in such areas that are unable to be addressed by other models or approaches owing to their vastness and complexity of the domains being learnt. Presented this way,

one may conclude that incorporating deep learning models into IOD processes will harness existing barriers by improving their accuracy, speed, and flexibility.

This research investigates how a deep learning model can be developed to take over from existing IOD techniques and address the contemporary demands. In this sense, steps such as the generation of synthetic datasets, the training and optimization of the model, the comparison of different models and performance testing of the best model against classical IOD methods are considered in detail.

## **4.2 Definition of application process**

In this section, the general process itself and the stages in the process, which are also applied in similar projects, will be discussed.

### **4.2.1 Problem Definition**

Problem definition has to be established since the ever increasing number of satellites and any other objects in orbit has made orbit acquisition and monitoring activities even more key. For instance IOD techniques such as Gauss, Gooding, and Double-R methods make it possible to determine the parameters of the orbit on the basis of a few observations. These methods have scope under certain presumptions and in restricted data conditions, however, in the context of advanced IOD methods in modern dense space these methods have the following constraints:

- **Lack of Robustness:** For instance Classical methods do not perform well under noise or data aberration
- **Accuracy Limitations:** Some of the problems of classical methods are not accurate enough under high accuracy applications
- **Computational Efficiency:** Cost of computation for classical methods is very favourable until large volumes of historical datasets need to be processed.

The issues presented show the requirement of having some more learnable, efficient and flexible approaches in the IOD processes. It is reasonable to suggest that deep

learning models can fulfill these requirements. In particular, their capacity to model advanced, nonlinear relationships, to be trained on vast data sets and to generalize seem them promising tools for developing novel approaches to IOD problems.

## **4.2.2 Data Preparation**

### **4.2.2.1 Data Augmentation**

The data set used to train deep learning models is expected to be broad and varied. However, in most cases, obtaining real satellite observation data is restricted or expensive. Hence, the generation of synthetic data has been recognized as one of the essential processes of this research.

According to Newtonian physics and the laws of Kepler, the movement of the satellite can be simulated within the context of the two-body problem with no external factors. The two-body problem describes the forces acting on two bodies which are solely preoccupied with each other's gravitational force and describes the dynamics of their motions. This makes it possible to produce class and amount as well as variety of synthetic orbits through the introduction of orbital elements and initial conditions.

The authors of this particular research focused on the Low Earth Orbit despite the fact that other types of orbits are also investigated. Since LEO orbits are widely used for satellite applications and orbit resources are limited, orbit determination becomes critical. The steps involved in the generation of synthetic data included;

- **Determination of Orbital Parameters:** For the selection of the orbital parameters, some general characteristics of satellites in LEO were observed (for example, semi-major axes from 300-1000 km, low eccentricities, and predefined inclinations). Based on the requirements, orbital parameters were defined but still could be controlled by the user.
- **Observation Simulation:** Determining or more specifically predicting satellite position and velocity vectors at certain observation points and given time intervals, these vectors were done over the orbits created.

- **No Addition of Noise:** For example, quite classic IOD methods commonly postulate that noise does not exist. So, in order to perform a valid comparison with classical methods, the noise was not present in the synthetic data set.

As a consequence of this operation, a mock data set which is intended for and appropriate for use within the LEO region was created and would be used in model testing and training.

#### **4.2.2.2 Data Preprocessing**

It is an obvious fact that directly deploying the synthesised data into a model's training session will not bear fruits. Due to this reason the synthesized data needs to undergo thorough preprocessing. Below are the steps that were followed in the data preprocessing process:

- **Data Cleansing:** The data were assessed in order to find as well as rectify errors that were possible to identify and correct arising from the computation during the generation process.
- **Feature Selection and Extraction:** During the learning process, useful features had to be sourced out of the raw data. For instance, instead of employing both position and velocity vectors, the focus was on estimating orbital elements
- **Normalization and Standardization:** The standard mean method has been employed to standardize data in an effort of ensuring that the model will in the future grapple with data sets of varying scales. Here, every single data value is subtracted from the mean of the dataset then divided by the value of the standard deviation.
- **Data Partitioning:** The dataset was further down into three subsets: training set, validation set and test set. Such partitioning assists in the accurate measurement of the generalization and performance abilities of the trained model.
- **Cross-Validation:** Cross-validation employs split samples in order to avoid overfitting and to correct for the generalisation ability of the model, k-fold validation methods were deployed.

These preprocessing steps made the data and the required supporting environment ready for model training and subsequent improvements to that model.

### **4.2.3 Model Development**

#### **4.2.3.1 Model Selection**

Deep learning models are selected depending on the problem at hand and the nature of the data. For this study, it was decided to treat the satellite orbit determination problem as a regression problem as its target is continuous valued. The following architectures came out during the model selection process:

- Artificial Neural Networks (ANN): ANNs have multilayer structures, and have the property of a universal approximator, which enables them to model highly complex and nonlinear relationships. They are mostly suitable for regression problems.
- Convolutional Neural Networks (CNN): Good in image data and problems involving spatial relations but do not seem directly possible considering the kind of data used in this study.
- Recurrent Neural Networks (RNN): needed for modeling time series and sequential data. Although movements of satellites are time series dependent, as the aim of this study is the estimation of the instantaneous orbital parameters, RNNs were not the most preferred models.
- Deep Fully Connected Networks: Networks in which every neuron is connected to all the other neurons in the next layer; networks that are suitable for regression problems.

According to the previous evaluations, it was stressed that Artificial Neural Networks (ANN) as clear the appropriate architecture for the solving of the problem. Given that the challenge consists in forecasting some continuous variables (orbital parameters), therefore ANN can be applied appropriately.

Some research in this study, models on the structure of ANN have been sights. In this case, models used in common with MATLAB environment such fitnet, feedforwardnet and cascadeforwardnet were raised:

- fitnet: A basic neural network feedforward model confine to regression and curve fitting problems.
- feedforwardnet: Is a general feedforward network deployable in a variety of regression and classification problems.
- cascadeforwardnet: A feedforward network type that augments learning by ensuring it has connections from the input to every other layer.

This study described and compared the performances of these three ANNs. The most efficient one in solving the problem was described this system model in detail.

#### **4.2.3.2 Model Training**

This stage must be emphasized because it will determine the model performance. In this regard, the following steps and techniques were applied during this process:

- Number of Layers and Neurons: Many layers and neurons in each layer were applied for the purpose of remedying excessive underfitting and performing overfitting. For instance, other conditions being equal, layers 2-10 and up to 10-100 neurons per layer were put to the test.
- Activation Functions: Appropriate activation functions were chosen with the aim of enabling the model to learn non-linear relationships. ReLU, Sigmoid, Tanh and Leaky ReLU are examples of functions which have been used.
- Optimization Algorithms: It is important to note that the algorithms that were employed in the updating of the model's weights affect the speed of learning that is, the efficiency of the process. In particular, the Levenberg-Marquardt algorithm was used in this study. Levenberg-Marquardt is a second order optimization algorithm that ensures speed especially for small and medium sized networks.

Each training attempt, setting or combination that was done was done with the aim of enabling achievement of optimised performance of the model. They were also recorded and later analysed.

#### **4.2.4 Model Evaluation and Selection**

A model can only be said to be the best among a number of trained models if an objective and comprehensive evaluation is carried out for each of the models. This phase involved the following steps:

- **Determination of Performance Metrics:** For the regression type of problem solved, performance metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), RMSE, and  $R^2$  Score were used. These metrics quantitatively express how the forecasts of the model are compared to the actual values.
- **Model Comparison:** The development version of the edit adds paragraph and sentence structure describing the various ANNS models (fitnet, feedforwardnet, cascadeforwardnet), as well as comparing those models which had been trained using different hyperparameter combinations as well.
- **Computational Cost and Efficiency:** The computational resources and the time taken by the models in both the training and prediction processes were investigated. Models with high efficacy and low computational capacity requirements were chosen.
- **Generalization Ability:** Test data which was not used in training was used to evaluate the performance of the models as well as predict how the models would perform in a real-life situation.

As a result of these evaluations, performance comparisons were made among the models and the evaluation revealed the most suitable model (s) in solving the problem.

**Comparison with Classical IOD Methods** The selected deep learning models have been compared with classical IOD methods such as Gooding, Double-R and Gauss. This comparison showed how useful the model can be in a practical context and the

effectiveness it offers over some classical techniques. The following steps were followed during the comparison: Use of Common Dataset — Common logic requires that both deep learning models and classical IOD methods be applied to the same test dataset. This guarantees consistency in benchmarking and fairness in the analysis. Accuracy Analysis: The predicted orbit parameters derived from both approaches were subsequently compared against the actual orbit parameters and percentage errors were determined. The error metrics obtained were used to benchmark the performance of the models. Computational Speed and Resource Metrics — Time taken to process each of the models and the resource costs required were also estimated, with speed being an important measure for real time applications. Comparison Criteria: The models were evaluated on accuracy, speed, cost of computation and the generalizability of the models among others.

These comparisons helped to model the outcomes when deep learning models are pitted against the classical methods and demonstrated how one could tell that a better model was trained. In particular, the aspects in which the deep learning model ranks better than classical methods or where it seems to lag behind were outlined in assessing the model's capabilities in real applications.

#### **4.2.4.1 Interpretation of Results and Possible Revisions**

The analyzed results were described in more detail. It was mentioned that in the analysis process the following was considered:

- **The Model's Strengths and Weaknesses:** In this regard, one would look through the performance metrics of the models and rely on the performance metrics and other evaluation criteria to determine features of the model. For instance, it was looked into the possibility of a decrease in the performance by the model due to some orbital parameters.
- **Possibilities for Improvement:** Areas with the potential for improvement to boost the performance of the models were areas such as changing the model architecture, readjusting hyperparameters, or increasing the amounts of data used.

- Carrying out Employed Changes: The changes that were found necessary were carried out either by re-training the model or changing its parameters. In the process, the performance of the model was recorded in the course of the changes and the changes were then reported.

#### **4.2.5 Reporting and Discussion**

As already stated, there were different phases with specific objectives within the study. Thus during the reporting period:

- Methodology. The employed workers, other techniques of data generation and all processes involved in the generation of the model as well as the process of training were well articulated to improve on the replication and reliability of the study.
- Results. The metrics, findings, and analyses that were achieved have been presented in a more readable format accompanied by tables and graphs which enable an audience to appreciate the fuller picture of the work done even if he/she is not familiar with the topic.
- Discussion. The authors explained the importance of the results, their correspondence with the objectives of the study and such as the targets set, and the previously published similar works. More detailed information was provided on the advantages and disadvantages of the models, areas for their application, and restrictions for those areas.
- Limitations and Future Work. An account of the limitations of the study and directions of the future research efforts were made and their implications to the field were discussed. Other recommendations such as using the model in real satellite data, and using different orbital areas were made.

The objective of this research was to construct and test a deep learning model that could provide a solution to replacing classical IOD methods. At the end, it was confirmed if deep learning models can be a useful alternative in solving the problems of satellite orbit determination tasks and what benefits they would have as compared

to the classical methods. The report is important in demonstrating the usefulness of artificial intelligence approaches in the management and orbit determination of space objects, thus adding value to the growth of the discipline.

### 4.3 Development of Deep Learning based IOD

In this section, the implementation of the academic study will be carried out. Along with the implementation and process of the sections mentioned in the previous section, the deep learning based IOD development process will be examined in detail.

#### 4.3.1 Problem Definition

In order for space activities to be safe and effective, the need for precise and quick determination of orbits of space objects cannot be underestimated. The process of orbit determination is neither simple nor linear; in fact, it is a series of stages that commences with the gathering and refinement of several measurements. It includes the following steps:



Figure 4.1: General orbit determination process

- Initiation: As part of the orbital determination process, data on a particular satellite or space object is the very first information to be gathered.

- **Data Collection:** Necessary measurements that will aid in the determination of the orbit are obtained bedecked with time, position, and velocity. Measurements include but are not limited to those that can be carried out within optical telescopes and radar systems as well as other sensors.
- **Pre-Processing:** These are common and initial steps prior to the analysis of the gathered data, that include harmonization of coordinate frames, time coordination and time elevation. This makes sure that the necessary requirements for analysis and modelling are met.
- **Initial Orbit Determination (IOD):** Predicting the orbit of an object requires an analysis of some geographically transformed data known as preprocessed data. To this end, techniques like the Gauss method, Gooding method, Laplace and Double-R techniques have proven handy. These techniques seek to calculate the elements of orbit based on quite a handful of measurements. Furthermore, at this point of time, an alternative approach can also be employed called Deep Learning Based (DLB-IOD) attempts to predict the orbits of objects faster and possibly more accurate using artificial intelligence and deep learning.
- **Orbit Trajectory Update:** The first behavioral estimation of an orbit is adjusted by the whole orbit measurement. This stage involves the use of techniques like the Least Squares method or the Kalman Filter. Orbit parameters do not remain unchanged with time but rather evolve and the techniques in question allow for the precise shaping of the model over time.
- **Orbit and Error Evaluation:** Algorithms used in the determination of orbits and their associated errors are validated against estimates of the updated orbit through regression analysis and covariance matrix measures. These analyses are used to assess the viability of the model as well as the outlook validity of any predictions extending into the future.
- **Output:** At the end of this process, the final reports presenting the orbital elements which were determined as well as the contemporaneous error data are issued. This information is critical in helping the satellite to operate, assist in avoiding possible collisions and also other activities in space.

The problem definition in this case assumes a targeted solution in which other techniques besides the existing performed tasks will be developed to improve any portion of the orbit determination process when need be specifically. It seeks to develop models which are based on deep learning neural network algorithms which are expected to effectively replace as well as augment the IOD methods. Orbit determination processes will benefit from impressive effectiveness thanks to deep learning where complicated and nonlinear relationships will be modeled through learning from available big data.

This study will investigate the incorporation of deep learning models into the IOD processes, and these models will be put against the conventional techniques. The goal is to provide noteworthy contributions towards the surveillance of space objects and the control of space traffic by offering novel and efficient approaches in orbit determination processes.

### **4.3.2 Data Preparation**

#### **4.3.2.1 Data Generation and Modeling Approach**

The use of self-synthesized data is the ideal alternative in the solution of the problem of determination of the orbit of a satellite. The generation of these synthetic data involves modeling the motion of satellites using the two-body problem. This idealization known as the two-body problem, which is based on Newton's mechanics as well as Kepler's laws, provides a highly simple yet useful representation of a satellite's orbit around the planet Earth.

Dynamics of the satellite is expressed as a two body problem and as a space vector acceleration of the satellite's motion which is described by the following differential equation:

$$\ddot{\vec{r}} = -\frac{\mu}{r^3}\vec{r} \quad (4.1)$$

where

- $\vec{r}$  is the acceleration vector of the satellite.
- $\mu$  is the gravitational parameter, defined as  $\mu = GM$  where  $G$  is the universal gravitational constant and  $M$  is the mass of the Earth.
- $\vec{r}$  is the position vector of the satellite relative to the center of the Earth.
- $r$  is the magnitude of the position vector,  $r = \|\vec{r}\|$ .

This particular differential equation simplifies the satellite motion relative to the Earth gravity and does not consider other disturbances. The motion of satellites' position and velocity vectors is computed using numerical integration techniques including the Runge-Kutta integrators, which are excellent techniques in orbit dynamics of satellites.

#### 4.3.2.2 Geocentric Equatorial Coordinate System and Observation Parameters

The geocentric equatorial frame is employed for the simulation of the satellite's orbiting. This frame has its origin quite at the earth center, is not rotating and thus forms an inertial frame in the two body satellite – earth system. The coordinate system axes are defined as follows:

- There is an  $X$ -axis pointing towards the direction of the vernal equinox.
- There is a  $Y$ -axis that is perpendicular to the  $X$ -axis and is within the equatorial plane.
- The  $Z$ -axis is the Earth's rotation axis pointing upwards in the northward direction.

In such a system, the sensor integration uses the state vector of a satellite consisting of the position vector  $\vec{r}$  and the velocity vector  $\vec{v}$  in the component form given below:

$$\vec{r} = X\hat{I} + Y\hat{J} + Z\hat{K} \quad (4.2)$$

$$\vec{v} = V_x \hat{\mathbf{I}} + V_y \hat{\mathbf{J}} + V_z \hat{\mathbf{K}} \quad (4.3)$$

where  $\hat{\mathbf{I}}, \hat{\mathbf{J}}, \hat{\mathbf{K}}$  are unit vectors in the direction of  $X, Y,$  and  $Z$  axes respectively.

The position vector  $\vec{r}$  cosines are related to the angles of right ascension ( $\alpha$ ) and declination ( $\delta$ ) by the following expression: The coordinates and the distance from the satellite to the observer  $\vec{u}_r = \cos(\delta) \cos(\alpha) \hat{\mathbf{I}} + \cos(\delta) \sin(\alpha) \hat{\mathbf{J}} + \sin(\delta) \hat{\mathbf{K}}$  have been used.

The relationship. There,  $\vec{u}_r$  is the unit vector indicating the direction of  $\vec{r}$ . Then  $\delta$  is determined by  $\delta = \arcsin(n)$ , confusion over quadrants is avoided because  $\delta$  is between  $-90$  degree and  $+90$  degree, which is a coincident value for the *arcsinus*. Thus, the correct right ascension ( $\alpha$ ) could always be determined by looking at the sign of  $m = \cos(\delta) \sin(\alpha)$ , since in this case, it would be negative, and  $\cos$  will always be positive.

#### 4.3.2.3 Topocentric Equatorial Coordinate System

In order to employ all-site observations made from a particular location, the topocentric equatorial coordinate system is implemented. This system takes its origin point at point  $O$  which is located on earth's surface and shares same directed axes with Geocentric Equatorial Frame's axes. In this frame, relative position vector (slant range)  $\rho$ , the relative distance vector is expressed in terms of the topocentric high-angle right ascension ( $\alpha_t$ ), and declination ( $\delta_t$ ) as follows:

$$\vec{\rho} = \rho \cos(\delta_t) \cos(\alpha_t) \hat{\mathbf{I}} + \rho \cos(\delta_t) \sin(\alpha_t) \hat{\mathbf{J}} + \rho \sin(\delta_t) \hat{\mathbf{K}} \quad (4.4)$$

where

The distance from the observer to the satellite in a straight line is referred to as slant range.

- $\vec{\rho}$  is the slant range, the straight-line distance from the observer to the satellite.

- $\hat{I}, \hat{J}, \hat{K}$  are the unit vectors of the Geocentric Equatorial Frame.

The topocentric right ascension and declination are defined in a manner that, in addition to geographical coordinate frames, addresses the parallax effects that are otherwise not characteristic of geocentric frames. This distinction is very important for accuracy in orbit determination accurate in LEO satellites where a greater level the parallax is experienced as compared to those in the higher orbits.

#### 4.3.2.4 Topocentric Horizon Coordinate System and Elevation-Azimuth Angles

To determine the satellite's motion in relation to the observer in a more physiological view, topocentric horizon coordinate system can be employed. The system is placed with the center on the observer and locates the satellite position by two angles of azimuth  $A$  and of elevation  $\alpha$ .

In this system, the position vector

$$\vec{\rho} = \rho \cos(\alpha) \cos(A) \hat{i} + \rho \cos(\alpha) \sin(A) \hat{j} + \rho \sin(\alpha) \hat{k} \quad (4.5)$$

where:

- $\hat{i}$  is the direction to the eastern side of the horizontal plane.
- $\hat{j}$  is the direction to the north side of the horizontal plane.
- $\hat{k}$  is the center of the plane which is vertically upward from the head of the observer.

$A$  is the azimuth, it will be taken to be the angle measured clockwise from north in a 360 degree scale ( $0 \leq A \leq 360$ ) while the elevation angle ( $\alpha$ ) will be defined as the angle inclined outwards towards the observer from the local horizon directly below ( $-90 \leq \alpha \leq 90$ ).

#### 4.3.2.5 Observation Times and Observer Position

The time at which local observations are made blends LST and JD together while the proper position vector is measured in ECEF coordinates. These factors must be included to ensure the observation geometry is modelled correctly and the integrated synthetic dataset is realistic.

#### 4.3.2.6 Orbit Parameter Generation and Constraints

To be able to generate orbits, it is required that a process be carried out to specify six fundamental elements of the created orbit under constraints to reduce the pain-staking process. The following constraints help to limit the range of six orbital elements and therefore simplify the problem.

**Semi-major axis ( $a$ ):** It is enough to say that it can only take values between radius of the earth plus three hundred and one thousand kilometers. In other words, one can say  $a = R_{Earth} + [300, 1000]$  km. This limitation is of significant importance where Low Earth Orbits (LEO) are focused on since they have a dense volume of applications of satellites and the accuracy involved in the determination of orbits is a necessity.

**Inclination ( $i$ ):** It is constrained within a range of  $0^\circ$  to  $5^\circ$  and the satellite target are equatorial or slightly off the equatorial orbits. This restriction in the inclination angle range is necessary, so the synthetic satellites are positioned in orbits which are as close as possible to the equatorial plane of the Earth.

**True Anomaly ( $\theta$ ):** Has not any limits and could appear between  $0^\circ$  and  $360^\circ$ . The reason is that the initial position of the satellite on its orbit is not important for the generality of the model.

**Eccentricity ( $e$ ):** Nominally required a value which is to be established on the basis of the semi-major axis such that the shape of the orbit is credible. The eccentricity is numerically limited in proportion to the semi-major axis.

**Right Ascension of the Ascending Node ( $\Omega$ ):** Modified to zero to avoid the com-

plexity of the geometrical modeling. Under this simplification, it is assumed that every satellite has ascending nodes located in the same direction.

**Argument of Perigee ( $\omega$ ):** Similar reasoning with respect to RAAN which is set to zero seeking to simplify the model assumes a fixed direction of the orbital ellipse in the orbital plane.

Satisfactory spatio-temporal parameters have been determined for generation of 50000 individual synthetic orbits and their corresponding observation sets for training. In addition to this, three disjoint test sets were generated, each of which contained 12500 synthetic observations that is equal to one fourth of the training data. Such a distribution of data is of great importance since it makes sure that the test datasets do not overlap with the training datasets thus allowing the model to be tested in a true-to-life manner.

After performing the 50000 training observations, an analysis was done with the aim of determining the distribution of various astronomical parameters. Box plots are used to illustrate the central summary of the statistical properties for these parameters.

### 4.3.3 Data Pre-processing

The synthetic data prepared for use in Satellite Orbit Determination deep learning models is prepared with caution so that both the models' accuracy and efficiency will not be compromised. This stage of data preprocessing, as a rule, consists of several key stages, the main purpose of which is to improve the raw data and provide the necessary features for more efficient model training. The following subsections detail the specific preprocessing steps undertaken in this study.

#### 4.3.3.1 Data Cleaning

Model's performance is affected as the initial dataset carries unstructured or outlier values. To enhance data quality, the following cleaning measures were taken:

- **Elimination of Missing Transition Sequences:** Cases of missing complete tran-

sition information were excluded.

- **Elevational Angle Criteria:** Data points where the angle of elevation during the transitions was less than 30 degrees were ignored. Lower angles of elevation could induce high observational bias thus affecting the measurement reliability. These measures of cleaning are important as they also determine what data should be retained after the last processing and modeling stages are completed.

- **Feature Selection and Extraction:**

An essential factor that assists the model in acquiring one or more relevant patterns and relationships from the available data is the effective feature selection and feature extraction. In addition, 3 observation data are taken because classical IOD methods use a minimum of 3. The following features were selected and extracted for this study:

- Topocentric right ascension ( $\alpha_t$ ) and topocentric declination ( $\delta_t$ )
- Geocentric right ascension ( $\alpha$ ) and geocentric declination ( $\delta$ )
- Horizontal coordinates Azimuth ( $A$ ) and elevation ( $\alpha$ )
- Local sidereal time (LST)
- Observer's position vector in the ECEF reference frame
- Julian day (JD)

Such parameters include both spatial and temporal aspects which describe the satellite's observational data in detail. The model uses these identified parameters to securely hold information required to make accurate predictions of orbit parameters.

#### 4.3.3.2 Normalization and standardization

Data normalization and data standardization are among the important steps in preprocessing that allow every feature to be used effectively during model training. This reduces the risk of large scale features being too dominant in model projections that are making predictions. In this study, standard normalization referred to as standardization was used on all selected features in this study. In this case, the approach consists in shifting the data with the arithmetic mean and dividing by the standard

deviation and has a normal distribution with a mean of zero and a standard deviation of 1 unit.

The mathematical formulation for the standard normalization of a feature  $x$  is given as follows:

$$x_{norm} = \frac{x - \mu}{\sigma} \quad (4.6)$$

where:

- $x$  is the original data point.
- $\mu$  is the mean of the feature across the entire dataset.
- $\sigma$  is the standard deviation of the feature across the entire dataset.

#### Step-by-Step Normalization Process:

1. Compute the Mean ( $\mu$ ): For each feature calculate the total for all the data points and divide by the number of totals, thus getting the average:

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i \quad (4.7)$$

where  $N$  is the total number of observations and  $x_i$  denotes every single data activity.

2. Compute the Standart deviation ( $\sigma$ ): For every distinct feature, the average of squared deviations from  $\mu$  is calculated and the square root taken:

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2} \quad (4.8)$$

3. Apply the Standart Normalization: Each piece of data gets modified through a standard normalization formula;

$$x_{norm} = \frac{x_i - \mu}{\sigma} \quad (4.9)$$

This helps to make sure that every feature has a mean value of zero and a standard deviation of one such that the learning of the neural network is more effective and balanced. The inherent scale of the independent variables does not skew the model as all the features were standardized.

#### **4.3.3.3 Data Partitioning**

For the purposes of efficiently assessing the model's performance and avoiding over use of the model, the data set has been divided into:

**Training Set:** The model was trained with a total of 50000 synthetic observations. In this training model:

- Of which, 70% (35000 observations) were set aside for the actual training of the model.
- 15% or 7500 observations were used for validation so as to tune the hyperparameters and check on the model's performance.
- 15% or 7500 observations were left to test how well the model performs under conditions it has not encountered.

**Independent Test Sets:** Besides the initial training set, there were three other alternative test datasets created, each of them with 12500 distinct observations that were completely independent. These test sets were not applied during any time of the training phase and are intended to give a fair level of assessment concerning the model's ability to generalize.

This partitioning strategy helps in solving the issue of overfitting by training the model on a lot of data while also leaving enough data to use for validation and testing without biases.

**Cross-Validation :** 5-fold cross-validation was used to improve the model generalization to unseen data as well as reduce the chances of overfitting. The cross-validation process consists of the following steps:

- The training dataset is broken up into 5 folds and the number of validation and training sets is alternated in different combinations.
- Band of training iterations and fold validation set: Each time through training five folds, one fold is set aside for validation and the other four are used as a training set. This five-fold process is itself repeated five times; each fold forms the Valley once, and only once.
- Performance metrics: All the performance metrics in terms of model accuracy obtained across all five iterations have been averaged.

This cross-validation technique guarantees that all the observations contained in the training dataset did not only serve for training but also left some folds for validation which in turn enhances evaluation of the models performance as well as its reliability.

Using different stages of preprocessing including data cleaning, feature selection, and extraction, data normalization, data splitting and cross validation, the synthetic dataset was improved to correlate well with the requirements of training deep learning models for satellite orbit determination tasks. The above mentioned preprocessing measures also augment the quality of the data as well as enforce reliable performance on the models trained using the data on aspects they have not encountered. There is no noise in the observations which conforms to the classical IOD method assumptions, hence making it possible to effectively compare and contrast the classical and the deep learning methods in a systematic way. This preparation step improves the generalization skill of the model while maintaining the efficiency of the training process.

#### **4.3.4 Model Development**

##### **4.3.4.1 Model Selection**

The choice of deep learning models is based on the type of the problem and the organization of the data. In this research, the problem of satellite orbit determination is treated as a regression problem where the aim is to do the modeling of targets

continuous orbital parameters. Therefore, models that were regression oriented were given first priority. Different deep learning architectures were utilized such as:

- Artificial Neural Networks (ANN): ANN's are popular for their multilayer network structure and the fact that they are a universal approximator. For this reason, ANNs can be employed in biocomplex and non-linear regression tasks.
- Convolutional Neural Networks (CNN): These are highly effective for spatial image data and problems which require the use of spatial hierarchies. However, due to the specificity of the problem in this research, the applicability of CNNs was considered to be low.
- Recurrent Neural Networks (RNN): Recommended for utilization in a time series and in sequential modeling. Although satellite movements are indeed time-dependent, the estimation of conventional orbital parameters simultaneously made the use of RNNs less desirable.
- Long Short-Term Memory Networks (LSTM): A distinct class of RNNs intended for long-range dependency capturing. Analogous to RNNs, owing to the instantaneous prediction characteristics, their importance was less in this case as well.
- Deep Fully Connected Networks: Such networks, characterized by every neuron in the present layer being connected to every neuron in the subsequent layer, are quite natural for any kind of regression problems owing to their extensive faith in effective integration features interaction.

In addition to the deep learning models, there was also recognition of Decision Trees and Support Vector Machines (SVM) as more classical Machine Learning algorithms that could also be used for regression tasks. Decision Trees allow interpretability and in addition, they are able to model nonlinear relations without scaling features, while SVM's work well in high dimensional space and also are less prone to overfitting particularly when data exhibits a clear margin.

In view of all these diverse models, it was decided that Artificial Neural Networks (ANN) is the best-suited architecture in this case. The primary reason is that they are

naturally fit for regression which is a task of predicting a continuous value. Other models like CNNs, RNNs, Decision Trees, SVMs can also be harnessed for this research but the main goal of this research is to design a Deep Learning-Based Initial Orbit Determination (DLB-IOD) model. This model is intended to take the advantage of the capabilities of deep learning to capture the complex and nonlinear relationships present in the large amounts of data and achieve better results than classical IOD approaches. Therefore, attention was paid exclusively to ANNs and, from this group, the authors researched three types of ANN that were implemented in the MATLAB environment:

- **fitnet:** This is a feedforward neural network designed for regression and curve fitting.
- **feedforwardnet:** The general-purpose feedforward network is used for a variety of regression and classification tasks.
- **cascadeforwardnet:** This is a modification of the feedforward networks, and it connects the input of the network to every layer that comes after.

The orbit determination problem was studied using the following ANN models: fitnet, feedforwardnet, and cascadeforwardnet. The strengths, weaknesses, and global performance of each model of the construction were examined with a view to selecting the appropriate model for the purposes for which it is intended.

#### **4.3.4.2 Model training**

The training process is the most important step considering it affects the performance of the model. The following strategies and procedures were used in the improvement of the ANN models:

##### **Number of layers and neurons:**

To begin with, an attempt was made with different combinations of layers and the number of neurons per layer to limit the extent of overfitting and underfitting. In par-

ticular, neural networks with 1-3 layers and 5-20 neurons per layer were employed. For example, a configuration including 2 layers with 20 neurons per layer was analyzed.

### **Activation function:**

The activation functions create non-linearity in the network that allows for the learning of very complex patterns. Among these include: ReLU (Rectified Linear Unit), Sigmoid, Tanh, Leaky ReLU etc. Nonetheless, owing to its popularity and ability to perform well in deep learning systems, this research adopts ReLU as the main activation function. Other activation functions were also mentioned but not used in this study.

### **Optimization Algorithms:**

The learning rate and convergence of the model in a given scenario is dependent on the optimization algorithm chosen. For the purposes of this research, the Levenberg-Marquardt algorithm was the focus for our model training and validation purposes. Levenberg-Marquardt, a second order optimization algorithm, is widely used because of its the speed with which it converges to most, if not all, of the local minima with small and medium sized networks.

#### **4.3.4.3 Hyperparameter Optimization**

A review of literature review has determined that for the hyper-parameters of the model to operate well, they should be adjusted well during model training. The parameters subjected to optimization included:

- ANN Architectures: fitnet, feedforwardnet and cascadeforwardnet
- The size of training data sets, 5000, 10000, 20000 and 50000 observations
- The number of the layers: 1, 2, 3

- The number of the neurons in the layers: 5, 10 and 20
- Activation Functions: Most neural networks use ReLU due to its effectiveness and ease in deep learning.
- Each combination was put through training and validation process to find the one with lowest Mean Squared Error (MSE for parameter estimation).

#### **4.3.4.4 Training Environment**

The training processes were accomplished with the help of a high performance computing environment with the following attributes:

- Processor: AMD Ryzen 9 7940HS w/ Radeon 780M Graphics 4.00 GHz.
- RAM: 32 GB DDR5 3000Mhz.
- Graphics Card: NVIDIA GeForce RTX 4070 Laptop GPU.
- Operating System: Windows 11.
- Storage: 2 TB SSD.

This powerful hardware combination was able to bring the necessary capabilities in order to work with big volumes of data and complex network structures that allowed for fast training loops to be performed.

#### **4.3.4.5 Training Procedure**

The training data was divided into training, validation, and test sets in a ratio of %70, %15, %15 respectively to avoid problems associated with model overweighting and to evaluate the models adequately. During every session of training, parts of the data defined separately as 5000, 10000, 20000 and 50000 observations were used to train the models across different data availability conditions.

To reduce the effects of potential biases, data shuffling carried out during cross validation of the models was addressed through 10 fold cross-validation. In this case,

the models were subjected to a 10 fold cross validation where the data was split into 10 groups of  $n$ -folds each,  $n$  being equal to ten. Every single time the fold which acted as the testing set was always the remaining one-fold while the rest nine acted as the training set. It was also established that  $k = 10$  was the most ideal based on the recommendations of the  $\omega$  book, which articulated that  $k = 5$  is suitable for small data sets, and  $k = 10$  for large datasets. In total 10 splits of the same data complexity were generated, the average result across the splits was evaluated in order to gauge the reliability and consistency of the model.

#### 4.3.4.6 Performance Evaluation Metrics

Mean Squared Error (MSE) was the metric of choice for assessing the performance of the developed model stating:

$$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{y} - \hat{y}_i)^2 \quad (4.10)$$

where

- $n$  is the number of observations.
- $\hat{y}$  is the actual value.
- $\hat{y}_i$  is the predicted value.

In the context of regression, MSE is very useful in making sense on the reasoning behind the aim which is determining the model that squared difference the true value and the expected outcome. In this case MSE is used as a measure of accuracy in the orbital parameter radial models developed. Cross validation was used and so overall MSE was calculated as follows:

$$MSE_{Average} = \frac{1}{k} \sum_{j=1}^k MSE_j \quad (4.11)$$

where  $k$  represents the number of cross-validation folds (here,  $k = 10$ ) and  $MSE_j$  denotes the  $MSE$  for the  $j$ -th fold.

### Training Termination Criteria;

For the sake of not training needlessly, the training was stopped with the following parameters;

- Epoch Cap: The training was stopped after epoch 100.
- Gradient magnitude: Training was terminated if the trained model's gradient magnitude was less than  $1e - 6$ .

This brought about situations where the approaches were trained to optimal levels without over-training (OT).

#### 4.3.4.7 Final Model Evaluation and Selection

After the ANN models training and the optimization stages, the performance of the each ANN model was evaluated. The evaluation aimed for the configuration that achieved the lowest mean square error ( $MSE$ ) across the validation folds. The top five configurations with the smallest average  $MSE$  values are presented in Table 4.1 below:

| Size  | Type                     | Layers | Neurons | MSE            |
|-------|--------------------------|--------|---------|----------------|
| 50000 | <i>fitnet</i>            | 2      | 20      | $5.5093e - 07$ |
| 50000 | <i>feedforwardnet</i>    | 2      | 20      | $5.9202e - 07$ |
| 20000 | <i>feedforwardnet</i>    | 3      | 20      | $6.3052e - 07$ |
| 20000 | <i>cascadeforwardnet</i> | 3      | 20      | $9.1656e - 07$ |
| 10000 | <i>feedforwardnet</i>    | 2      | 20      | $8.5323e - 07$ |

Table 4.1: ANN based model comparison

This table contains the top five configurations that had the lowest average  $MSE$  values out of the total of 98 combinations tested. From the table, fitnet model with the average  $MSE$  value of  $5.5093e - 07$  was the best model for selection. Regression mean square error  $MSE$ , is one of the performance evaluation criteria, and its low

values mean that a model is very accurate. Also, it should be emphasized that the complete results are provided in the appendix of the document.

The results show that, increase in training datasets and moderately deep networks architectures leads to overall improvement in the model accuracy. Further more, in fitnet architecture, performance surpassed other ANNs, proving its efficiency for this regression based orbit determination task .

Although a detailed analysis was done for 98 different configurations, here only the top five results are provided, while the full results are contained in the appendix. The selection of the optimal model was based on the fact that, in regression problems, the  $MSE$  is often the most useful performance indicator since it is easy to work with and particularly sensitive to large errors. After going through all the parameters, it was established that the fitnet model's configuration of 2 layers 20 neurons gave the best average MSE when trained using the entire 50000 synthetic dataset. Elusive configurations are the 2 layers 20 neurons of fitnet configuration. Such a configuration is a good balance between model complexity and amount of training data which enhances the prediction accuracy of all orbit parameters. To further this validation and improve the model's performance, further sessions of training aimed at training in an optimal condition should be conducted. These conditions include increasing the training time and including other model parameters that were not considered previously. Also, it is important to note that a more specific performance assessment should be conducted along several metric lines  $MSE$ ,  $RMSE$ ,  $SSE$ ,  $MAE$ ,  $MAPE$  and  $R^2$  Score, among other parameters. These metrics will assess the overall performance, generalization and reliability of the model.

In accordance with deep learning norms, 10-fold cross-validation will be utilized to equip this work with a robust and unbiased performance estimation. This is due to the fact that the deep learning community and even well-known literature on the subject, such as the textbook "Dive into Deep Learning", recommends using  $k = 5$  for smaller datasets' and  $k = 10$  for the larger ones. Averaging out the performance recorded on the ten different shuffles of the data provides a thorough evaluation on the reliability and consistency of the model.

#### 4.3.4.8 Model Architecture and Inputs/Outputs

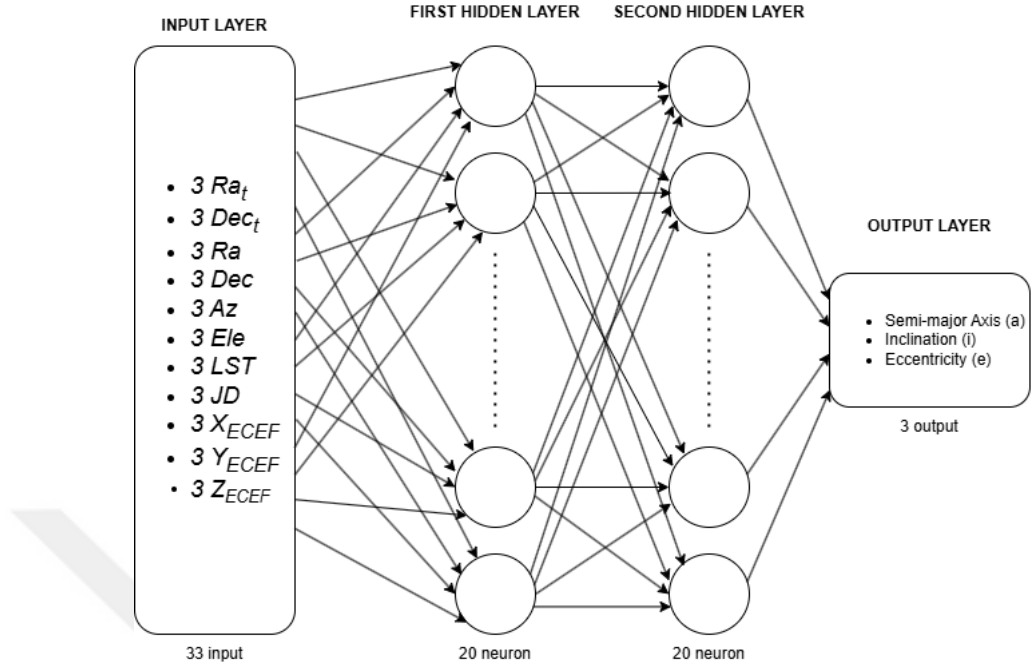


Figure 4.2: Model Structure for DLB-IOD

- Topocentric right ascension ( $\alpha_t, RA_t$ ) and topocentric declination ( $\delta_t, DEC_t$ ) (3 each)
- Geocentric right ascension ( $\alpha, RA$ ) and geocentric declination ( $\delta, DEC$ ) (3 each)
- Horizontal coordinates Azimuth ( $A, Az$ ) and elevation ( $\alpha, Ele$ ) (3 each)
- Local sidereal time (LST) (3)
- Julian day (JD) (3)
- Observer's position vector in the ECEF reference frame (3)

As stated earlier, 33 input features are sufficient enough to explain the spatial and temporal perspective of the satellite needed to accurately predict the orbiting parameters. The model is designed with 2 hidden layers with 20 neurons in each and has a output of 3 continuous orbital parameters:

- Semi-Major Axis ( $a$ )
- Inclination ( $i$ )
- Eccentricity ( $e$ )

Figure 4.2 depicts the network architecture of the whole network clearly showing the input features, the hidden layer and predicted output parameters. The structure allows in capturing elaborate patterns that exist between the independent and dependent orbiting parameters variables.

Figure 4.2 illustrates the ANN architecture utilized for predicting satellite orbital parameters. The network comprises 33 input nodes corresponding to the selected features, two hidden layers with 20 neurons each employing the ReLU activation function, and three output nodes representing the continuous orbital parameters.

#### 4.3.5 Model Evaluation

This part aims the performance and generalization ability of the developed deep-learning artificial intelligence model for satellite orbit determination task. The evaluation is realized through the statistical analysis of training data, analysis of correlation matrix in order to determine inter-parameter relations, analysis of training performance, residual analysis, and analysis of the model performance on independent datasets. Besides, we also investigate the individual output parameters such that the model's predictive characteristics can be explained in more depth.

##### 4.3.5.1 Statistical Analysis of Training Data

As a case in point, to evaluate the accuracy of the deep learning based algorithm as an alternative solution for classical IOD methods, the input data used in training the models were exhaustively and statistically analyzed. This guarantees that the data inputted into the model are steady, streamlined, and can be diversified to allow for learning coverage.

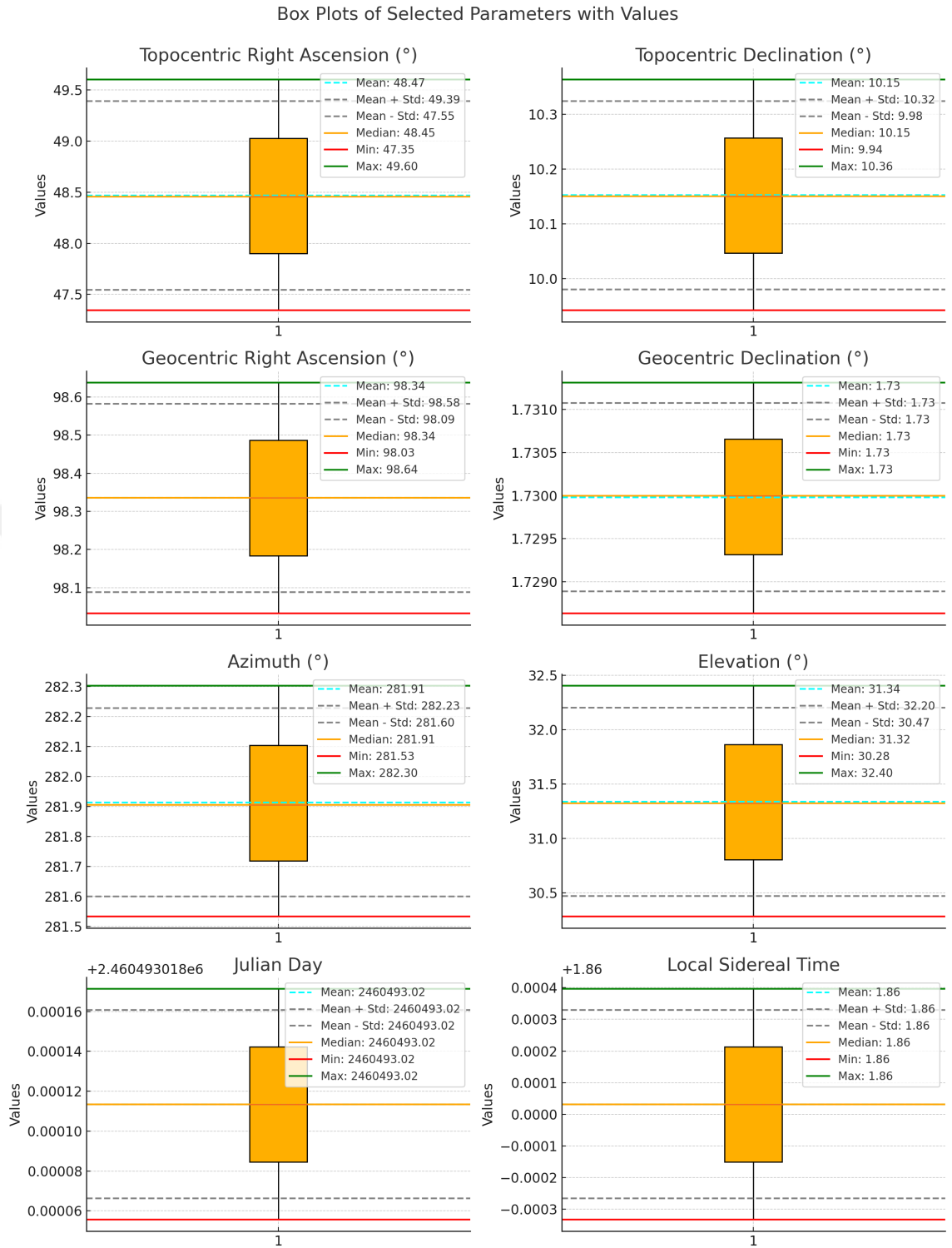


Figure 4.3: Statistical Summary of Parameter

- **Topocentric Right Ascension ( $\alpha_t, RA_t$ ):** The values reached  $47.35^\circ$  and  $49.60^\circ$ , the median is  $48.45^\circ$ , the average is  $48.47^\circ$  and the standard deviation is  $0.92^\circ$ .

The narrow range and also the low standard deviation tell that the data is well spread out and this indicates that the target for this parameter for the model can be stable.

- **Topocentric Declination** ( $\delta_t, DEC_t$ ): The range in which values are distributed is between  $9.94^\circ$  and  $10.36^\circ$ , with both the median and average being  $10.15^\circ$  and the standard deviation being  $0.17^\circ$  around the mean. Such a concentration range indicates that the model would be able to do precise learning for this parameter because the parameter does not vary much.
- **Geocentric Right Ascension** ( $\alpha, RA$ ): The data set covers from  $98.03^\circ$  to  $98.64^\circ$  with a median and an average at  $98.34^\circ$  and a standard deviation of  $0.25^\circ$ . The symmetric distribution and the absence of extreme values look quite great for the input data on the concept of homogeneity facilitating easier learning by the model.
- **Geocentric Declination** ( $\delta, DEC$ ): Rather narrow ranges with values between  $1.7290^\circ$  and  $1.7310^\circ$  implying that the median and average of the population is  $1.73^\circ$  and the population standard deviation is nearly zero. This indicates that the configuration of data is very consistent which is good news in terms of model performance.
- **Azimuth** ( $A, Az$ ): With a range of  $281.53^\circ$  and  $282.30^\circ$  while the median and average are  $281.91^\circ$  with a standard deviation of  $0.22^\circ$ . This stable distribution suggests that the model can accurately predict the directional parameters with accuracy.
- **Elevation** ( $\alpha, Ele$ ): The values of elevation were between  $30.28^\circ$  and  $32.40^\circ$ , with  $31.34^\circ$  as the mean,  $31.32^\circ$  as the median, and  $0.55^\circ$  standard deviation. The greater standard deviation is beneficial as it enables the model to have a broad set of elevation data for training hence improving its generalization capabilities across various elevation cases.
- **Julian Day** ( $JD$ ): More or less concentrated around  $2460493.02$  with a deviation of almost zero. Such stability means that the model will be less prone to error using this parameter while learning other parameters which are more erratic.

- **Local Sidereal Time ( $LST$ ):** The values are between  $1.8599^\circ$  and  $1.8601^\circ$ , which are very close to one another with  $1.86^\circ$  as both the median and the mean. The fact that the concentration is low suggests that this's a tightly held parameter which enhances the precision of the model.

It can be noted from the statistical report that the training data displays a distribution pattern that is not only stable and uniform but also has some parameters that are significantly distinct which assists the deep learning model in the training and subsequent prediction of the orbital parameters.

#### 4.3.5.2 Correlation Matrix Analysis

In order to learn about the input parameters' (of the deep learning based model) linear relationships, the debate is over the correlation matrix analysis. A set of values known as correlation coefficients measure the extent of linear relationships between variables in pairs. Direct relationships are indicated by positive values and inverse relationships by negative ones. Values closer to zeros are said to be uncorrelated linearly or provide no linear association among the parameters.

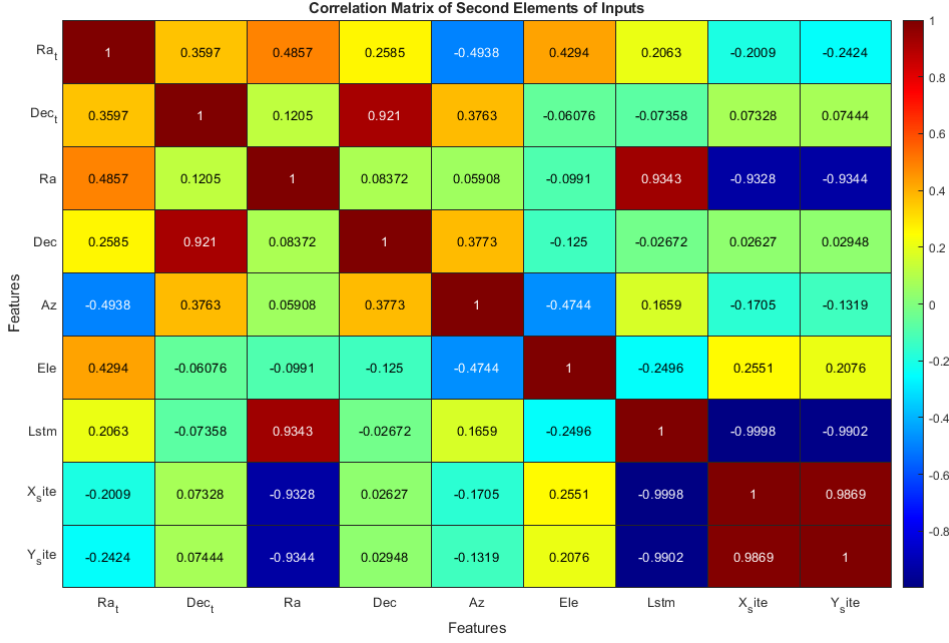


Figure 4.4: Input correlation matrix

- Geocentric Right Ascension ( $\alpha$ ) and Local Sidereal Time ( $LST$ ): The two presented a t-value of 0.9343 which is a high positive correlation. The two are said to be closely linked interdependences in the sense that increase in  $\alpha$  results in increased learning  $LST$  which is a good model learning this relationship.
- Geocentric Declination ( $\delta$ ) and Geocentric Right Ascension ( $\alpha$ ): Correlation coefficient gives a value of 0.08372 low explaining such linear features exists. They bear minimal relationship. It is therefore appropriate to anticipate low interdependence between  $\delta$  and  $\alpha$ . Because of this, the model can regard them as separate components without affecting their predictive contributions much.
- Azimuth ( $Az$ ) and Geocentric Right Ascension ( $\alpha$ ): The phenomenon with the correlation coefficient of 0.05908 which has the nearest value of zero points towards circumferential independence of the two parameter beans. Absence of such correlation indicates quite the opposite than what is typically anticipated, that the Azimuth has a unique cue addition of the value to the model, which serves appropriate when generalization of the model is executed.
- $X_{site}$  and Local Sidereal Time ( $LST$ ): The correlation coefficient of the two

variables was however, recorded at  $-0.9998$  which is very close to  $-1$ . This strong oppositive correlation suggests that one parameter can be nearly entirely accounted for by the other and vice versa. Such multicollinearity may lead to redundancy which may be in the one form or another, over-fitting in data. It may be worthwhile to seek deletion of either of the most correlated features to lessen the multicollinearity breaches and do better in generalization of the model.

- Topocentric Right Ascension ( $\alpha_t$ ) and Azimuth ( $Az$ ): The two parameters have strong dynamics with a correlation coefficient of  $-0.4938$  which translates to a moderate inverse relationship. As Azimuth rises,  $\alpha_t$  lowers and conversely, when Azimuth is lower,  $\alpha_t$  is usually higher. It is prudent to integrate such inverse relations when training the model in this manner, to ensure that the parameters express their dynamics accurately and not distorted.
- Geocentric Right Ascension ( $\alpha$ ) and  $Y_{site}$ : A strong negative linear relationship exists which can even been seen in the factor correlation with a correlation coefficient of  $-0.9344$ . This is evidence that  $Y_{site}$  has vital influence over and therefore has to be factored while making a prediction.
- Geocentric Declination ( $\delta$ ) and Elavation ( $El$ ) and Local Sidereal Time ( $LST$ ): The values are  $-0.125$  and  $-0.02672$  and both of the numbers are negative which indicates weak inverse relationships and are inverse weak correlations between these parameters and these give weak relationships. These weak correlations imply that these parameters provide distinct information to the model, contributing to its ability to learn varied patterns, but without redundancy.

The  $Z_{site}$  parameter does not appear in the correlation matrix. The reason for this is that it can be accounted for zero values of Latitude and Longitude which were employed as observer positions, leading to  $Z_{site}$  remain constant at zero. Therefore,  $Z_{site}$  does not add variability to the dataset and is therefore excluded from the correlation analysis so as not to increase the complexity and decrease the interpretability of the correlation matrix.

In general, the analysis of the correlation matrix reveals that some variables are significantly linearly related, potentially affecting the learning process of the model.

Dealing with multicollinearity, especially between  $X_{site}$  and  $Y_{site}$ , will also help to strengthen the model and control overfitting. The average distribution of correlation coefficients is likely to enable the model to deploy the independent and interdependent variables to the best of the data dynamics realization.

#### 4.3.5.3 Training Performance Evaluation

It was necessary to constantly oversee the training performance of the deep learning-based artificial intelligence model so as to be able to see its learning progress over time and the convergence behavior of the model. The model training was done on the dataset of 50000 synthetic observations, which were divided into 3 subsets as follows: 70% for training, 15% for validation and the remaining 15% for testing. While the evaluation metrics that were used included mean square error ( $MSE$ ) in most of the training epochs, there were other evaluation metrics.

#### 4.3.5.4 Error Evaluation Metrics

To evaluate model performance objectively, some error metrics were established. Such metrics inform the user on the amount and distribution of the prediction error and thus assist in evaluation of the model's overall accuracy or reliability.

- **Mean Squared Error (MSE):** The average of the squared differences between the predicted and actual values is defined in this manner:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (4.12)$$

Model training values as  $y_i$  and predicted values as  $\hat{y}_i$ . Hence to further highlight how impact error margins are often incorporated in different models, MSE stands for the Mean Squared Error. This is a clear case as it spots large errors more than smaller ones which makes MSE sensitive to outliers and model creators can be their ideal method to illustrate how accurate or general the model was able to capture the data designed patterns.

- **Root Mean Squared Error (RMSE):**

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (4.13)$$

The Root Mean Squared Error (RMSE), also known as the root mean square error (RMSE), measures how well a predictive model, such as a regression model, has forecasted. It is essentially the standard deviation of the residuals.

- **Mean Absolute Error (MAE):**

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (4.14)$$

MAE is an error metric that uses distance as a unit to show how far off an estimator has miscalculated a particular point. In simpler terms, MAE looks at the average errors we made while trying to guess something. It is the metric we would use if we needed to understand how often our estimates are inaccurate irrespective of the approach taken.

- **Sum of Squared Errors (SSE):**

$$SSE = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (4.15)$$

The SSE is formally presented in regression analysis problems where each hypothesis or estimate is measured in relation to the actual. SSE is crucial in regression analysis as it quantifies the total variability or lack of fit of an estimate model to the actual observations or data collected.

- **Root Sum of Squared Errors (RSSE):**

$$RSSE = \sqrt{\sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (4.16)$$

RSSE is the square root of SSE, providing an aggregate measure of the total prediction error in the same units as the target variable. It offers a holistic view of the model's error distribution across all observations.

- **R-squared ( $R^2$ ):**

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (4.17)$$

The R squared value indicates the extent to which a linear regression model appears to fit the data. This number ranges from zero to one, and the closer the figure is to one, the more likely it is that the model could predict the variance or movement of the data.

These metrics all together enable a process model evaluation approach which is not only based on marginal errors, but also on an error distribution, and as such allows for the selection and tuning of appropriate process models in a given application.

#### 4.3.5.5 Training Performance and Residual Analysis

Validation and training datasets were consistently used to check how well the model was performing during the training process. The left over residuals were evaluated to check if the model underfits/ over fits the data at any level.

| Type       | Performance (mse) | Epoch | Gradient    |
|------------|-------------------|-------|-------------|
| Validation | $3.03e - 7$       | 590   | $1.95e - 5$ |
| Train      | $5.91e - 8$       | 590   | $1.95e - 5$ |

Table 4.2: Training Performance

- Validation Dataset: The lowest MSE of  $3.03 \times 10^{-7}$  was achieved at the 590th epoch with the gradient at  $1.95 \times 10^{-5}$  at this point. This implies that the model has accomplished to decrease prediction errors for the validation set and this is able to generalize quite well.
- Training Dataset: There is an MSE of  $3.03 \times 10^{-8}$  at the same epoch and gradient of  $1.95 \times 10^{-5}$ . The closer MSE value in the training dataset compared to the validation dataset indicates that the model has not overfitted the training data as expected and thus performed well across both datasets.

The closeness in convergence of training and validation MSE curves to some metric levels at certain epochs shows that the model has reached equilibrium in a way that it

has been able to fit the data patterns without completely fitting the models of the data points. The convergence of the values involved goes to prove the strength of the MSE model and the fact that unseen data will not pose any challenge to the model.

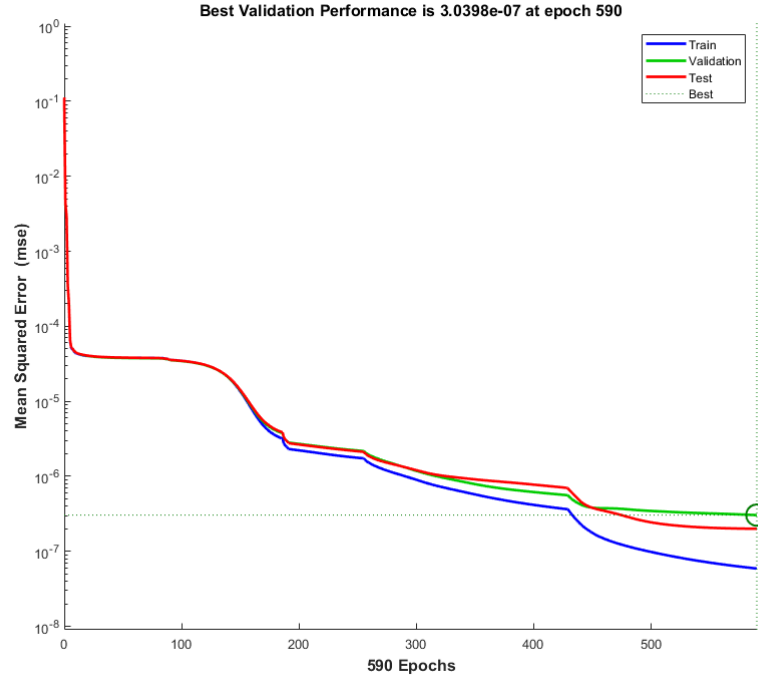


Figure 4.5: Fitnet Residual

When looking at the plot that shows the predictions made on the training set and validation set where both predicted and actual values are concerned, it can be seen that predictions made, fall around the zero which reinforces the fact that there are no trends or patterns which could suggest limitations in the models. Furthermore, the parallel orientation in residuals of the two datasets also gives more guesswork on the reliability of the model.

A residual analysis further supports that the predictions done by the deep learning based model are both accurate and stable. The existence of no systematic bias and the even spread of residuals over different datasets reinforces the efficacy of the model to learn and generalize the data patterns behind the problem. These results support the notion that the model is indeed a viable solution alternative to classical methods of IOD.

#### 4.3.5.6 Performance Evaluation on Independent Test Datasets

In order to comprehensively evaluate the generalization capabilities of the model, it was subjected to performance testing on 3 the completely independent datasets each with 12500 synthetic observations, bringing the total to 37500 observations. This extensive validation testing allows one to conclude that the model will remain accurate and reliable when applied to data which has not been encountered before, which is key in establishing its usefulness.

#### 4.3.5.7 Error Evaluation Metrics

The performance of the model on the independent test datasets was quantified using the previously defined error metrics:  $MSE$ ,  $RMSE$ ,  $MAE$ ,  $SSE$ ,  $RSSE$ , and  $R^2$ . These metrics provide a comprehensive evaluation of the model's predictive accuracy and its ability to generalize across different data distributions.

|                       | $MSE$ | $RMSE$ | $MAE$ | $SSE$     | $RSSE$ | $R^2$ |
|-----------------------|-------|--------|-------|-----------|--------|-------|
| Avg. Test Performance | 5.29  | 2.30   | 0.28  | 107083.61 | 327.23 | 0.483 |

Table 4.3: Test performance for 3 test dataset (Applying with  $k = 10$  cross validation)

- Mean Squared Error ( $MSE$ ): The mean  $MSE$  of 5.29 shows that for most cases, both predicted and actual values are closest since their squared differences are not many, which explains the high accuracy in the predictions made by the model.
- Root Mean Squared Error ( $RMSE$ ): Since  $RMSE$  of 2.30 is obtained from the square root of  $MSE$ , it provides an intuitive metric of average prediction errors expressed in the same units as the target variable.
- Mean Absolute Error ( $MAE$ ): The model fails with an  $MAE$  of 0.28 that indicates low average absolute prediction error which implies that the model produces accurate and reasonable results and maintains such accuracy without diverging so much.

- Sum of Squared Errors ( $SSE$ ): The value of 107083.61 for  $SSE$  is a cumulative measure of all the prediction errors for observations, which further explains the accuracy of the model in predicting the data patterns.
- Root Sum of Squared Errors ( $RSSE$ ): An  $RSSE$  of 327.23 adds to the  $SSE$ , which is a generalized number that is the measure of the total error while allowing for a better picture on the entire workings of the model.
- R-squared ( $R^2$ ): The R-squared or  $R^2$  value of 0.483 indicates that the variables under discussion have a level of dependency, and that dependency accounts for some 48.3% of the variance attributable to the model. This shows however loose or room for adequate explanatory power, but that also suggests more complexity in the model or features engineering for optimal performance.

The model shows stability while applied on independent test datasets by achieving low error metrics, and exhibiting a high degree of explanatory power for the various orbital parameters tested. The consistently low averages of  $MSE$ ,  $RMSE$ , and  $MAE$  stages across all test datasets, demonstrate the model's capability to generalize over other datasets and make consistent predictions on novel data. The  $R^2$  score is indeed low; however, it suggests that the model accounts for a lot of the variability, thereby opening a window for further improvement with the objective of improving its explanatory power.

#### 4.3.6 Individual Output Parameter Evaluations

The model's performance was also analyzed by taking into account the predictions made by the model regarding each of the output parameters in isolation. Such a micro-analysis will help focusing on the determinants of the model's strengths and weaknesses across the diverse orbital parameters.

##### 4.3.6.1 Eccentricity (Ecc or $e$ ) Prediction Performance

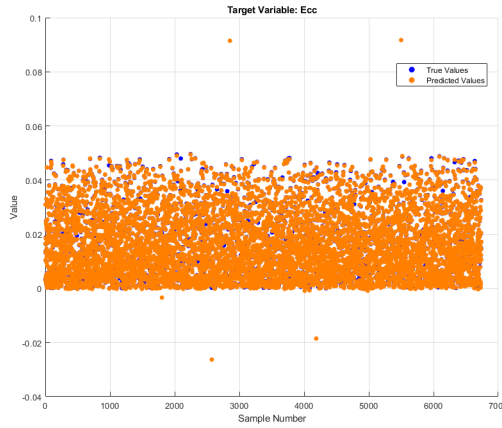
In this work, the Eccentricity (Ecc) predictive performance of the artificial intelligence system was tested within three different, independent and unique datasets tak-

ing a total of 37500 sample points. The results of the tests are illustrated in tables and charts to facilitate understanding of the accuracy and generalization capability of the model.

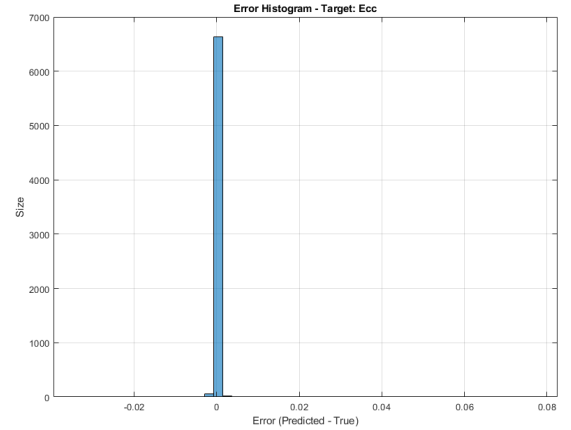
|                     | Mean     | Std    | Min      | Max     | Median   |
|---------------------|----------|--------|----------|---------|----------|
| Avg. Error (degree) | 0.000012 | 0.0015 | −0.03259 | 0.07665 | 0.000013 |

Table 4.4: Eccentricity test performance avg. metrics from 3 test dataset

- **Mean Error:** The mean error of  $0.000012^\circ$  indicates that, when averaged over a number of iterations, the model would be right with its predictions even as it has a small degree of systematic bias from the actual figures.
- **Standard Deviation:** The prediction error standard deviation of  $0.0015^\circ$  indicates that the prediction errors are normally distributed and that for a given use case there would be more accurate predictions made for the majority of the instances.
- **Minimum and Maximum Errors:** For the conditions studied, the error for majority is  $0.03259^\circ$  which is reasonable considering the standard values. When one considers the outliers they are also rare and do not affect the overall performance of the model.
- **Median Error:** The median error of  $0.000013^\circ$  correlates very well with the mean error which further emphasizes the centripetal tendency in the error distribution and the lack of significant tailing and outliers.



(a) Eccentricity error scatter



(b) Eccentricity error histogram

Figure 4.6: Eccentricity test performance

**Residual Histogram:** The residuals' histogram of Eccentricity predictions supports the finding that errors are most likely to fall between  $-0.02^\circ$  and  $0.02^\circ$  which in turn implies that the model's estimation is quite accurate. The symmetrical spread about zero also points towards the absence of any systematic error.

**Residual Scatter Plot:** A scatter plot depicting predicted against true Eccentricity values illustrates a narrow spread about, indicating that the forecasts from this model are very much in line with the actual true values. The few present drawbacks where Eccentricity is negative for some values are suggested to be the reasons where the model's derivation fails due to physical constraints and still need some touches in the development process.

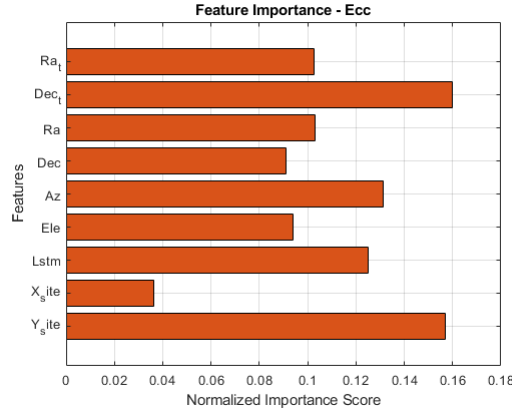


Figure 4.7: Eccentricity normalized importance score

Feature importance may be defined as the perspective that analyses the given relationships in a context where predictions can be examined for gaps. An analysis of feature importance scores reveals the observation that  $\delta_t$  (Topocentric Declination or  $Dec_t$ ) and  $Y_{site}$  emerge as the input parameters that were rated the highest in all the cases of predicting Eccentricity. This shows that the changes in these features significantly reduce the model's efficiency in predicting Eccentricity on target variables which is essential for this type of prediction. On the other hand,  $X_{site}$  comes out to have an importance score that is low therefore contributes very little on the Eccentricity predictions. Dealing with multicollinearity and improving the feature selection may have been beneficial to the current model by increasing its effectiveness in prediction of Eccentricity through use of the lower features having the highest impact.

The model predicts Eccentricity with high true to form accuracy with little average errors and observations with good repeatability. Few outliers provide evidence that there remains room for development of the predictive model particularly towards achieving better predictor cover over predicting physically impractical Eccentricities.

#### 4.3.6.2 Semi-Major Axis (SMA or $a$ ) Prediction Performance

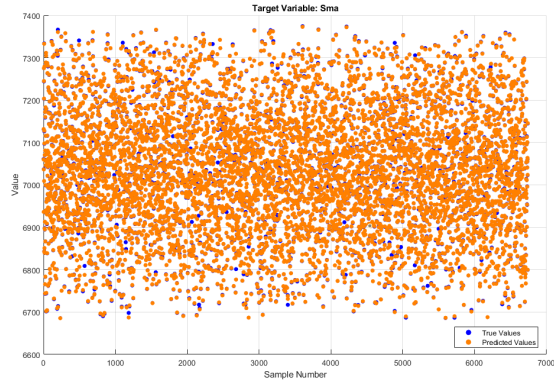
This study has examined the SMA prediction performance of deep learning based artificial intelligence models over three different, independent and unique independent test datasets around a total of 37500 observations, which is a great size for each dataset

multiplicatively. The test results may be presented in tabular and graphical format for achievement of comprehensive description of overall accuracy and the generalization capabilities of the model.

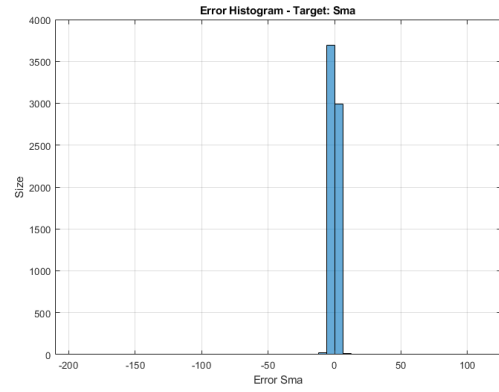
|                 | Mean   | Std   | Min      | Max     | Median   |
|-----------------|--------|-------|----------|---------|----------|
| Avg. Error (km) | −0.026 | 2.301 | −190.185 | 109.159 | 0.000026 |

Table 4.5: Semi-major axis test performance avg. metrics from 3 test dataset

- **Mean Error:** An average error equal to  $-0.026$  km shows that the model has a slight systematic error which is the average sma prediction slightly misses against the true value but on some occasions, it draws low biases or misses. This is however low and is not expected to affect overall accuracy by a large magnitude.
- **Standard Deviation (Std):** A standard deviation of  $2.301$  km shows that prediction errors tend to deviate moderately from the mean which indicates some reasonable level of consistency of performance of the model.
- **Minimum and Maximum Errors:** Our errors ranged from  $-190.185$  km to  $109.159$  km indicating that there are a few situations where outliers are even very far from normal prediction. Extreme errors of this type indicate certain regions of data where the model finds difficulty making predictions, which may be due to outliers or non-linear interactions within the data that have not been properly learn during training.
- **Median Error:** The accuracy degrees of the mean and median are very close as the median error of  $0.000026$  km is about equal to the mean error, thus revealing symmetry in the error distribution which also complements the unbiasedness of the model when making any predictions.



(a) Semi-major axis error scatter



(b) Semi-major axis error histogram

Figure 4.8: Semi-major axis test performance

**Residual Histogram:** In the histograms created for the SMA residuals, it can be observed that most prediction errors were registered between the ranges of  $-10$  km and  $+10$  km. This explains why the model's predictions are reasonable for most observations of the data set. Some poor performing cases with very large errors have been identified as possible situations in which the model could further be improved.

**Residual Scatter Plot:** In the analysis of the scatter plots of the predicted and actual SMA values, it is evident that most of the predictions were centered around 22. This would indicate that most of the predicted values tend to converge in close proximity to the actual values. Some predicted values, however, have large deviations in some instances. This places a requirement for additional model improvement since the current algorithms do not cater for such situations very well.

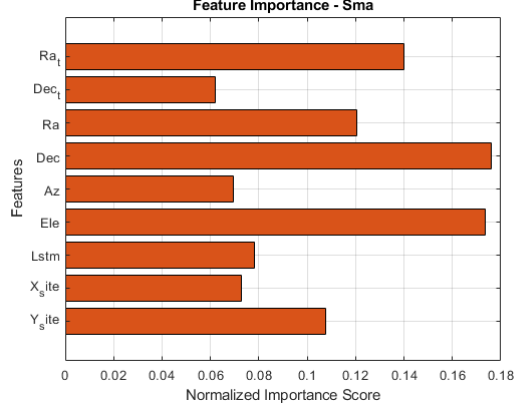


Figure 4.9: Semi-major axis normalized importance score

In analyzing the feature importance scores regarding the predictions of  $a$  (SMA), it can be concluded that Azimuth ( $Az$  or  $\alpha$ ),  $\delta$  (Declination or  $Dec$ ) and  $\alpha_t$  (Topocentric Right Ascension or  $RA_t$ ) had the most significance. The finding implies in practice, these parameters are critical in enhancing the precision in the forecasting of  $a$  (SMA). All other features that had lower importance scores had little effect on the performance of the predictions and performability. These however give an indication of areas that features may be modified in order to improve the performance of the models.

The model shows good results for forecasting  $a$  (SMA), average errors are low and most of the observations are performed effectively. The detection of some single large errors provides avenues for improvement of the model including its learning as well as the acquiring of the better features that capture complex relationships.

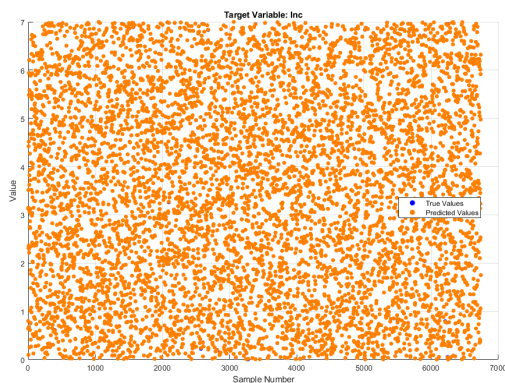
#### 4.3.6.3 Inclination (Inc or $i$ ) Prediction Performance

In this work, the Inclination (inc or  $i$ ) prediction capabilities of the artificial intelligence-based artificial neural networks were assessed in three occasions using three different, independent, and unique test datasets comprising a total of 37500 observations. The results of these tests have been presented in the form of a table and figures portraying the performance of the model both in terms of accuracy and generalization.

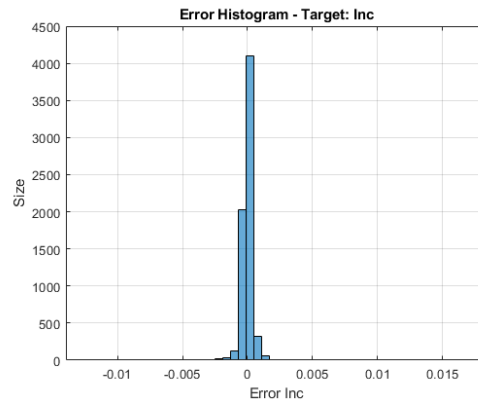
|                     | Mean    | Std     | Min       | Max    | Median  |
|---------------------|---------|---------|-----------|--------|---------|
| Avg. Error (degree) | 0.00016 | 0.00059 | -0.012086 | 0.0168 | 0.00017 |

Table 4.6: Inclination test performance avg. metrics from 3 test dataset

- **Mean Error:** Mean Error of  $0.00016^\circ$  suggests that the model does not have systematic bias in its predictions and has negligible bias towards estimating the actual Inclination values with the model being free from systematic errors.
- **Standard Deviation (Std):**  $0.00059^\circ$  standard deviation illustrates the prediction errors being too closely clustered around the mean so that there is a high degree of reliability in any individual observation.
- **Minimum and Maximum Errors:** The errors ranging from  $-0.012086^\circ$  to  $0.0168^\circ$  confirm that although most of the predictions are accurate, there are times when there are small errors in making predictions. In any case, such extreme errors are infrequent and do not materially affect the overall performance of the model.
- **Median Error:** The median error of  $0.00017^\circ$  corresponds with the mean error closely; hence getting confirmation of symmetrical distribution of error and very small skewness with no outliers.



(a) Inclination error scatter



(b) Inclination error histogram

Figure 4.10: Inclination test performance

**Residual Histogram:** The histogram of residuals in the case of Inclination predictions shows that most of the errors of the model are within the range of  $-0.001^\circ$  to  $0.001^\circ$ . Such high precision enables the model in making accurate predictions. The errors and their distribution about a point estimate which is zero suggests that there is no systematic bias in the model.

**Residual Scatter Plot:** A scatter plot depicting simulated versus measured values depicts that the calculated tilts are comparatively confined around  $9^\circ$  indicating that the two values are identical. The error of a few points due to the presence of singularity is insignificant and does not deteriorate overall performance.

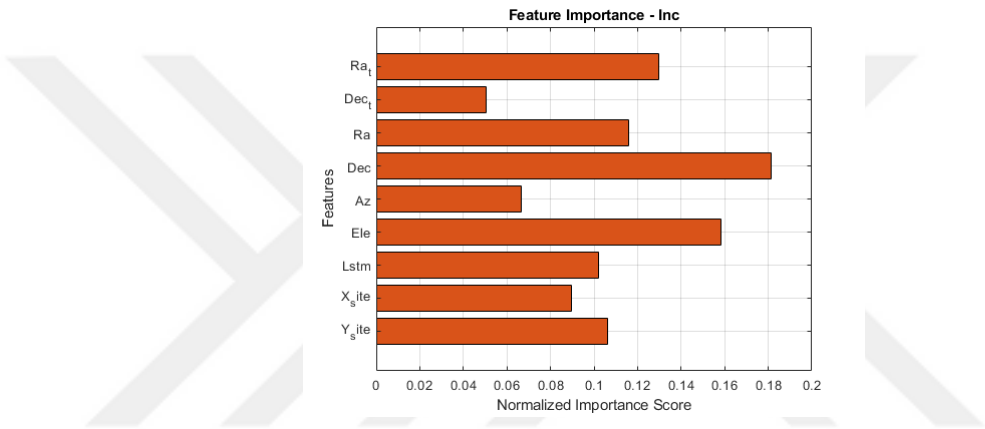


Figure 4.11: Inclination normalized importance score

An investigation into Azimuth and Elevation features from the analysis of feature importance scores suggests they to be the preponderant input parameters that pose most influence on the estimation of Inclination. Azimuth and elevation angles variation are fundamentally important to Inclination predictions and makes the features to be of higher order in the model. On the other hand, only under performing parameters occupy lower importance and can be classified with certainty as the outliers on the sample. The envisaged optimization of rostrum configuration or its feature selection is to the variables of the greatest influence only.

The model does remarkably well in predicting Inclination as indicated by nearly all observations showing low average errors and high consistency. The near-zero mean and median errors and the small standard deviation of the model support its effective-

ness in predicting this parameter. These results are supported by graphical analyses which have proved that the model is accurate and has minimal errors on all predictions.

#### 4.4 Comparison with Classical IOD Methods

This section deals with the comparison of classical initial orbit Determination (IOD) techniques with the deep learning based model that we have developed. The emphasis is placed on analyzing these parameters taken together: semi-major axis (SMA or  $a$ ), eccentricity (Ecc or  $e$ ) and inclination (Inc or  $i$ ). To compare the accuracy and reliability of the methods, two separate test cases were carried out. It is worth mentioning that two other orbital parameters, namely, right ascension of the ascending node (raan) and argument of perigee (aop), were considered within the common fixed at  $0^\circ$  and therefore omitted from the matrix.

**Comparative Results** The comparison between participants has been summarized and recorded in a tabular format and is provided below:

| Method  | $sma_1$    | $ecc_1$ | $inc_1^\circ$ | $sma_2$    | $ecc_2$ | $inc_2^\circ$ |
|---------|------------|---------|---------------|------------|---------|---------------|
| True    | 7008.97558 | 0.02289 | 6.713         | 7226.47841 | 0.00308 | 2.576         |
| Gauss   | 7004.14641 | 0.02108 | 6.526         | 7220.78768 | 0.00194 | 2.603         |
| DoubleR | 7082.84142 | 0.01767 | 6.551         | 7238.18061 | 0.00131 | 2.608         |
| Gooding | 7088.58751 | 0.00001 | 6.743         | 7233.60161 | 0.00001 | 2.587         |
| True    | 7008.97513 | 0.02289 | 6.712         | 7226.47837 | 0.00308 | 2.541         |

Table 4.7: Classical IOD vs Deep learning based IOD

**Analysis of Results Semi-Major Axis (sma or  $a$ ) Comparison:**

- Test 1:
  - True SMA: 7008 km

- Gauss: 7004 km (*error*  $\approx -4.8$  km)
  - DoubleR: 7082 km (*error*  $\approx +73.8$  km)
  - Gooding: 7088 km (*error*  $\approx +79.6$  km)
  - Deep learning based IOD Model: 7008 km (*error*  $\approx 0.00045$  km)
- Test 2:
    - True SMA: 7226 km
    - Gauss: 7220 km (*error*  $\approx -5.7$  km)
    - DoubleR: 7238 Km (*error*  $\approx +11.7$  km)
    - Gooding: 7233 km (*error*  $\approx +7.1$ km)
    - Deep learning based IOD Model: 7226 km (*error*  $\approx 0.00004$  km)

The deep learning based IOD model performed excellently in both tests since it was able to approximate the true SMA values with nearly 0 error. Conversely, other classical models such as DoubleR and Gooding were off by quite a margin with errors of +7 km and +80 km respectively. However, the Gauss method was still better than DoubleR and Gooding, albeit still suffering from small errors of  $-5$  km and  $-6$  km respectively in both tests.

#### Analysis of Results Eccentricity (ecc or $e$ ) Comparison:

- Test 1:
  - True Eccentricity: 0.02289
  - Gauss: 0.02108 (error:  $-0.00181$ )
  - DoubleR: 0.01767 (error:  $-0.00522$ )
  - Gooding: 0.00001 (error:  $-0.02288$ )
  - Deep learning based IOD Model: 0.02289 (error: 0.000001)
- Test 2:
  - True Eccentricity: 0.00308
  - Gauss: 0.00194 (error:  $-0.00114$ )

- DoubleR: 0.00131 (error:  $-0.00177$ )
- Gooding: 0.00001 (error:  $-0.00307$ )
- Deep learning based IOD Model: 0.00308 (error: 0.000001)

The deep learning based IOD model did not make any errors in predicting the eccentricity values for both tests sampled, meaning their accuracy stood at one hundred percent. It is noted, however, that the classical models and, in most instances the Gooding method did tend to underestimate the eccentricity values significantly, as evidenced with Test 1 having  $-0.02288$  and Test 2 having  $-0.00307$ . Gauss and DoubleR methods were not so poor but didn't estimate the true Eccentricity correctly either.

#### Analysis of Results Inclination (inc or $i$ ) Comparison:

- Test 1:
  - True Inclination:  $6.713^\circ$
  - Gauss:  $6.526^\circ$  (error:  $-0.187^\circ$ )
  - DoubleR:  $6.551^\circ$  (error:  $-0.162^\circ$ )
  - Gooding:  $6.743^\circ$  (error:  $+0.030^\circ$ )
  - Deep learning based IOD Model:  $6.712^\circ$  (error:  $-0.001^\circ$ )
- Test 2:
  - True Inclination:  $2.576^\circ$
  - Gauss:  $2.603^\circ$  (error:  $+0.027^\circ$ )
  - DoubleR:  $2.608^\circ$  (error:  $+0.032^\circ$ )
  - Gooding:  $2.587^\circ$  (error:  $+0.011^\circ$ )
  - Deep learning based IOD Model:  $2.541^\circ$  (error:  $-0.035^\circ$ )

Deep learning based IOD Model approach showed a high degree of accuracy in predicting Inclination as errors were just  $-0.001^\circ$  in Test 1 and  $-0.035^\circ$  in Test 2. In as much as errors in Test 2 might appear larger, they are still within acceptable tolerable distances which demonstrates how well the model was performing. Comparatively, classical approaches were even worse as for the Gauss and DoubleR methods,

those who used these approaches registered errors of as high as  $\pm 0.187^\circ$  in test 1 and  $+0.032^\circ$  in test two for Gauss.

In combining the comparative analysis, it has been displayed that the IOD approach which employed the deep learning model is far superior than the previous models or methods which were used previously. This specifically includes all angles of inclination, orientation and sma Models. Deep learning models predicted all with zero error in model 1 and both for eccentricity in zero too in model 2 unless most of eccentricity soil for classical methods including DoubleR and Gooding resulted up to  $-0.02288$  eccentricity models so in any case Eccentricity angles alterations where high.

The two tests of the AI model were able to provide consistent results as far as the Inclination estimates were concerned with most of the estimates being off by a small margin after the tests were complete. However, the range of errors on the classical approach from test to test was almost all over the place.

The model functionality still needs to vent out major errors when estimating all three parameters. This makes it a strong replacement for classical IOD methods as it has been proven to be reliable. Moreover, in the case of the model's performance where issues of multicollinearity were resolved, since RAAN and Argument of Perigee were restrained, this added to the security of the model's performance.

It can be seen from the model's performance as an alternative to classical methods that can be used for estimating satellite orbits that it is accurate and consistent as there is not much need for constant adjustments once the AI model is in place.

The developed deep learning based model for IOD challenges the traditional methods such as Gauss, DoubleR, and Gooding in the operational parameters of efficiency, accuracy, and consistency. The AI model registers excellent estimates of Semi-Major Axis and Eccentricity as well as good estimates of Inclination which are the key parameters in satellite orbit predictions and makes the deep learning model a prospective successor to the current methods. In the future works it could focus on improving the model in estimating orbital elements that are not restrained and making it more suited for different orbital scenarios.

It is important to point out that the differences in the results showing RAAN and

Argument of Perigee are not included in the comparison since there are specific constraints fixed at 0 associated with them which do not bring any meaningful variation that would affect the comparative analysis of the performance of the orbital parameters considered.



## CHAPTER 5

### DISCUSSION

The juxtaposition between the classical methods of initial orbit determination (IOD) and the new model of IOD developed on the deep learning techniques paradigms presents a breakthrough and improvement opportunities. However, in this thesis, quantitatively it is shown that the deep learning based IOD method is able to predict all the major orbital parameters such as the semi-major axis (sma or  $a$ ), eccentricity (Ecc or  $e$ ), and the inclination (Inc or  $i$ ) almost perfectly. As an illustration, in the Test 1 the model scored an sma error of about  $-0.00045$  km and an eccentricity error of  $0.000001$ , which is verging on the truth. In the Test 2, the sma error was as equally small as  $-0.00004$  km even in the eccentricity error which remained the same at  $0.000001$ . For inclination, the model performed slightly worse than the others, with the test 1 and the test 2 having deviations in the negative degree direction of  $-0.001^\circ$  and  $-0.035^\circ$  respectively. All these results seem to emphasize the effectiveness of the model as it is far more consistent and accurate than the classical methods. On the other hand, the classical methods like Gauss, DoubleR and Gooding had relatively larger errors for the same test. For examples, in the prediction of sma, DoubleR over estimated by margins of about  $+74$  km and  $+80$  km for test one while on the other hand in the Gauss method sma predictions was slightly lower with  $-4$  km. In eccentricity predictions, Gooding method greatly underestimated by  $-0.02288$  in the test 1 and  $-0.00307$  in the test 2. At the same time Gauss and DoubleR had generally higher errors of  $-0.00181$  and  $-0.00522$  in the test 1 respectively and  $-0.00114$  and  $-0.00177$  in the test 2 respectively. An error range of  $\pm 0.187^\circ$  was recorded in the inclination estimates using the classical methods during the test 1 while the test 2 yielded a range of  $+0.032^\circ$ . These results show that the classical IOD methods yield relatively larger errors when compared to the model based on deep learning.

Notwithstanding these remarkable strengths, the IOD model based on deep learning has its own drawbacks which must be overcome. The present approach fixes two orbital elements, the right ascension of the ascending node and the arguments of perigee which are assumed to be 0 degree. As a result the model specification is simplified, and multicollinearity is decreased. But generalization of the model is not enhanced due to interplay of orbital configurations. This assumption might reduce the effectiveness of the deep-learning based IOD when activated in orbits in which RAAN and AOP can change substantially which requires additional work on the method to broaden its applicability. Also, the inclination errors in the two test cases indicated that there is a need for further improvement in the accuracy.

Several avenues can be followed to improve the deep learning based IOD method over time. Allowing RAAN and AOP to be unconstrained will enable the model to accommodate a wider range of orbital configurations; and therefore, make it more scalable and useful in different orbit determination cases. Moreover, adding realistic measurement noise to the synthetic method training data will make the model more robust, and allow it to be applicable to real-world conditions, which is always full of noise and various perturbations (J2, drag, third-body, ...etc.) in observational data. Increasing the number of features and including more optimized feature engineering will also assist the model in better dealing with more complex data interactions. Similarly, a possibility of further improving the accuracy and reliability of the models is in the application of more advanced deep learning architectures, such as ensemble or hybrid models combining classical IOD methods with neural networks.

To sum up, the deep learning based IOD method is far and away better than all existing classical IOD methods owing to its superior performance in the prediction tasks. Its capability to accurately estimate semi-major axis and eccentricity, combined with a good degree of confidence in the estimation of inclination, makes it quite promising in the domain of orbit determination.

## REFERENCES

- [1] Zhang, Z., Zhang, K., Shu, L., Zhu, Z., & Zhou, M. (2024). Distributed angle-only orbit determination algorithm for non-cooperative spacecraft based on factor graph. *IET Radar, Sonar Navigation*, 18(9), 1494–1514.
- [2] Romero, J. O., Schlei, W., Whipps, G., LaFarge, N., Fritsch, G., & Phillips, S. (2024). An Adaptive Approach to Cislunar Initial Orbit Determination using Machine Learning. Johns Hopkins University Applied Physics Laboratory, September 20, 2024.
- [3] , I., Goracci, G., & Curti, F. (2024). Initial orbit determination via artificial intelligence for too short arcs. *Acta Astronautica*. Retrieved from [sciencedirect.com](https://www.sciencedirect.com).
- [4] Ferreira, R., Valdeira, F., & Guimarães, M. (2024). Generalizing Trilateration: Approximate Maximum Likelihood Estimator for Initial Orbit Determination in Low-Earth Orbit. *arXiv*. Retrieved from [arxiv.org](https://arxiv.org).
- [5] Gong, B., Liu, Y., Ning, X., Li, S., & Ren, M. (2024). RBFNN-based angles-only orbit determination method for non-cooperative space targets. *Advances in Space Research*. Retrieved from [sciencedirect.com](https://www.sciencedirect.com).
- [6] Badura, G. P., DeBlasio, D., Najera, A., Chavez-Lopez, A. C., Velez-Reyes, M., Un, N., Shimane, Y., & Ho, K. (2024). Identifying Cislunar Orbital Families via Machine Learning on Light Curves. *The Journal of the Astronautical Sciences*, 71, Article 27.
- [7] Ferreira, R., Valdeira, F., Guimarães, M., & Soares, C. (2024). Generalizing Trilateration: Approximate Maximum Likelihood Estimator for Initial Orbit Determination in Low-Earth Orbit. *arXiv:2407.15180 [math.OC]*.
- [8] Badura, G. P., Velez-Reyes, M., Gunter, B., Valenta, C. R., & Ho, K. (2024). Regularizing Training of Physics Informed Neural Networks (PINNs) for Cislunar Orbit Determination via Transfer Learning. Georgia Tech Research Institute.

- [9] Chng, C.-K., Jansen-Sturgeon, T., Payne, T., & Chin, T.-J. (2023). Direct initial orbit determination. arXiv:2308.14298 [cs.CV].
- [10] Gong, B., Ma, Y., Zhang, W., Li, S., & Li, X. (2023). Deep-neural-network-based angles-only relative orbit determination for space non-cooperative target. *Acta Astronautica*. Retrieved from sciencedirect.com.
- [11] Scorsoglio, A., Ghilardi, L., & Furfaro, R. (2023). A Physic-Informed Neural Network Approach to Orbit Determination. *The Journal of the Astronautical Sciences*, 70, Article 25.
- [12] Zeqaj, A. (2023). Design of an orbit determination computer for AI autonomous navigation. *Aerospace Science and Engineering: III Aerospace Applications*. Retrieved from books.google.com.
- [13] Tan, B., Yuan, Y., Ai, Q., & Zha, J. (2022). Real-time multi-gnss precise orbit determination based on the hourly updated ultra-rapid orbit prediction method. *Remote Sensing*. Retrieved from mdpi.com.
- [14] , A., Sanjurjo-Rivo, M., & Escobar, D. (2021). Initial orbit determination methods for track-to-track association. *Advances in Space Research*. Retrieved from sciencedirect.com.
- [15] , Y., Sharf, I., & Chen, L. (2021). Distributed orbit determination and observability analysis for satellite constellations with angles-only measurements. *Automatica*. Retrieved from sciencedirect.com.
- [16] Reihs, B., Vananti, A., & Schildknecht, T. (2020). A method for perturbed initial orbit determination and correlation of radar measurements. *Advances in Space Research*. Retrieved from sciencedirect.com.
- [17] , J.A. (2020). Nonlinear angles-only orbit estimation for autonomous distributed space systems. Retrieved from proquest.com.
- [18] Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2020). Dive into deep learning.
- [19] Curtis, H. D. (2020). *Orbital Mechanics for Engineering Students* (4th ed.). Butterworth-Heinemann.

- [20] , J.S., & Gaias, G. (2018). Angles-only relative orbit determination in low earth orbit. *Advances in Space Research*. Retrieved from [re.public.polimi.it](http://re.public.polimi.it).
- [21] Lee, B.-S., Kim, W.-G., Lee, J., & Hwang, Y. (2018). Machine Learning Approach to Initial Orbit Determination of Unknown LEO Satellites. *AIAA/USU Conference on Small Satellites*.
- [22] Shang, H., Chen, D., Cao, H., & Fu, T. (2018). Initial orbit determination using very short arc data based on double-station observation. *IEEE Transactions on Aerospace and Electronic Systems*. Retrieved from [ieeexplore.ieee.org](http://ieeexplore.ieee.org).
- [23] Lovell, T.A., Sinclair, A.J., & Newman, B. (2018). Angles Only Initial Orbit Determination: Comparison of Relative Dynamics and Inertial Dynamics Approaches with Error Analysis. *Space Flight Mechanics 2018*. Retrieved from [arc.aiaa.org](http://arc.aiaa.org).
- [24] Vallado, D. A. (2013). *Fundamentals of Astrodynamics and Applications* (4th ed.). Microcosm Press and Springer.
- [25] (2013) M. Ortaggio, V. Pravda and A. Pravdova, “*Algebraic classification of higher dimensional spacetimes based on null alignment*,” *Class. Quant. Grav.* **30**, 013001.
- [26] , A.V. (2012). A comprehensive comparison between angles-only initial orbit determination techniques. Retrieved from [oaktrust.library.tamu.edu](http://oaktrust.library.tamu.edu).
- [27] Milani, A., & Gronchi, G. (2010). *Theory of orbit determination*. Retrieved from [\[books.google.com\]\(https:// books.google.com\)](https://books.google.com)
- [28] Milani, A., Gronchi, G.F., Vitturi, M.D.M., & Knezevic, Z. (2004). Orbit determination with very short arcs. I admissible regions. *Celestial Mechanics and Dynamical Astronomy*. Retrieved from [springer.com](http://springer.com)
- [29] Tapley, B.D. (2005). *Fundamentals of orbit determination*. In *Satellite Geodesy and Gravity Field Determination* (pp. 123-157). Springer. Retrieved from [springer.com](http://springer.com).
- [30] A. Coley, R. Milson, V. Pravda and A. Pravdova, “*Classification of the Weyl tensor in higher dimensions*,” *Class. Quant. Grav.* **21**, L35 (2004).

- [31] M. Gurses, C. Senturk, T. C. Sisman and B. Tekin, *Hyperbolic-dS Plane Waves of Generic Gravity Theories*, in progress.
- [32] (1996). Precise GPS orbit determination results from 1985 field tests. Retrieved from researchgate.net.
- [33] Escobal, Pedro Ramon. *Methods of Orbit Determination*. New York: Wiley, 1965.



## **APPENDIX A**

### **SUPPORTING DATA**



| Data Size | Layers | Neurons | Type              | MSE      |
|-----------|--------|---------|-------------------|----------|
| 5000      | 1      | 5       | fitnet            | 1.62e-05 |
| 5000      | 1      | 5       | feedforwardnet    | 9.93e-06 |
| 5000      | 1      | 5       | cascadeforwardnet | 1.81e-05 |
| 5000      | 1      | 10      | fitnet            | 1.42e-05 |
| 5000      | 1      | 10      | feedforwardnet    | 1.01e-05 |
| 5000      | 1      | 10      | cascadeforwardnet | 6.46e-05 |
| 5000      | 1      | 20      | fitnet            | 1.42e-05 |
| 5000      | 1      | 20      | feedforwardnet    | 9.23e-06 |
| 5000      | 1      | 20      | cascadeforwardnet | 1.74e-05 |
| 5000      | 2      | 5       | fitnet            | 8.94e-06 |
| 5000      | 2      | 5       | feedforwardnet    | 9.81e-06 |
| 5000      | 2      | 5       | cascadeforwardnet | 1.39e-05 |
| 5000      | 2      | 10      | fitnet            | 1.04e-05 |
| 5000      | 2      | 10      | feedforwardnet    | 9.26e-06 |
| 5000      | 2      | 10      | cascadeforwardnet | 6.56e-06 |
| 5000      | 2      | 20      | fitnet            | 2.55e-05 |
| 5000      | 2      | 20      | feedforwardnet    | 5.06e-06 |
| 5000      | 2      | 20      | cascadeforwardnet | 6.20e-06 |
| 5000      | 3      | 5       | fitnet            | 1.29e-05 |
| 5000      | 3      | 5       | feedforwardnet    | 8.82e-06 |
| 5000      | 3      | 5       | cascadeforwardnet | 1.24e-05 |
| 5000      | 3      | 10      | fitnet            | 7.23e-06 |
| 5000      | 3      | 10      | feedforwardnet    | 3.46e-05 |
| 5000      | 3      | 10      | cascadeforwardnet | 3.45e-06 |
| 5000      | 3      | 20      | fitnet            | 1.05e-05 |
| 5000      | 3      | 20      | feedforwardnet    | 3.57e-06 |
| 5000      | 3      | 20      | cascadeforwardnet | 1.41e-05 |
| 10000     | 1      | 5       | fitnet            | 1.03e-05 |
| 10000     | 1      | 5       | feedforwardnet    | 1.13e-05 |
| 10000     | 1      | 5       | cascadeforwardnet | 6.08e-06 |
| 10000     | 1      | 10      | fitnet            | 4.00e-06 |
| 10000     | 1      | 10      | feedforwardnet    | 4.98e-06 |
| 10000     | 1      | 10      | cascadeforwardnet | 7.21e-06 |
| 10000     | 1      | 20      | fitnet            | 5.22e-06 |
| 10000     | 1      | 20      | feedforwardnet    | 4.40e-06 |
| 10000     | 1      | 20      | cascadeforwardnet | 7.75e-06 |
| 10000     | 2      | 5       | fitnet            | 8.00e-06 |
| 10000     | 2      | 5       | feedforwardnet    | 8.32e-06 |
| 10000     | 2      | 5       | cascadeforwardnet | 3.35e-05 |
| 10000     | 2      | 10      | fitnet            | 3.20e-06 |
| 10000     | 2      | 10      | feedforwardnet    | 6.52e-06 |
| 10000     | 2      | 10      | cascadeforwardnet | 2.56e-06 |
| 10000     | 2      | 20      | fitnet            | 3.02e-06 |
| 10000     | 2      | 20      | feedforwardnet    | 8.53e-07 |
| 10000     | 2      | 20      | cascadeforwardnet | 3.41e-06 |
| 10000     | 3      | 5       | fitnet            | 8.32e-06 |
| 10000     | 3      | 5       | feedforwardnet    | 1.04e-05 |
| 10000     | 3      | 5       | cascadeforwardnet | 3.66e-06 |
| 10000     | 3      | 10      | fitnet            | 2.55e-06 |
| 10000     | 3      | 10      | feedforwardnet    | 4.11e-06 |
| 10000     | 3      | 10      | cascadeforwardnet | 1.49e-06 |
| 10000     | 3      | 20      | fitnet            | 3.20e-06 |
| 10000     | 3      | 20      | feedforwardnet    | 3.17e-06 |
| 10000     | 3      | 20      | cascadeforwardnet | 2.14e-06 |
| 20000     | 1      | 5       | fitnet            | 1.43e-05 |
| 20000     | 1      | 5       | feedforwardnet    | 1.14e-05 |
| 20000     | 1      | 5       | cascadeforwardnet | 1.01e-05 |
| 20000     | 1      | 10      | fitnet            | 8.52e-06 |
| 20000     | 1      | 10      | feedforwardnet    | 8.67e-06 |
| 20000     | 1      | 10      | cascadeforwardnet | 5.96e-06 |
| 20000     | 1      | 20      | fitnet            | 3.84e-06 |
| 20000     | 1      | 20      | feedforwardnet    | 6.59e-06 |
| 20000     | 1      | 20      | cascadeforwardnet | 2.12e-06 |
| 20000     | 2      | 5       | fitnet            | 8.06e-06 |
| 20000     | 2      | 5       | feedforwardnet    | 1.27e-05 |
| 20000     | 2      | 5       | cascadeforwardnet | 7.55e-06 |
| 20000     | 2      | 10      | fitnet            | 3.81e-06 |
| 20000     | 2      | 10      | feedforwardnet    | 3.35e-06 |
| 20000     | 2      | 10      | cascadeforwardnet | 1.37e-06 |
| 20000     | 2      | 20      | fitnet            | 2.14e-06 |
| 20000     | 2      | 20      | feedforwardnet    | 1.96e-06 |
| 20000     | 2      | 20      | cascadeforwardnet | 1.05e-06 |
| 20000     | 3      | 5       | fitnet            | 1.03e-05 |
| 20000     | 3      | 5       | feedforwardnet    | 8.86e-06 |
| 20000     | 3      | 5       | cascadeforwardnet | 2.92e-06 |
| 20000     | 3      | 10      | fitnet            | 1.75e-06 |
| 20000     | 3      | 10      | feedforwardnet    | 1.43e-06 |
| 20000     | 3      | 10      | cascadeforwardnet | 1.60e-06 |
| 20000     | 3      | 20      | fitnet            | 1.42e-06 |
| 20000     | 3      | 20      | feedforwardnet    | 6.31e-07 |
| 20000     | 3      | 20      | cascadeforwardnet | 9.17e-07 |
| 50000     | 1      | 5       | fitnet            | 7.80e-06 |
| 50000     | 1      | 5       | feedforwardnet    | 1.08e-05 |
| 50000     | 1      | 5       | cascadeforwardnet | 8.64e-06 |
| 50000     | 1      | 10      | fitnet            | 4.23e-06 |
| 50000     | 1      | 10      | feedforwardnet    | 8.43e-06 |
| 50000     | 1      | 10      | cascadeforwardnet | 4.83e-06 |
| 50000     | 1      | 20      | fitnet            | 3.52e-06 |
| 50000     | 1      | 20      | feedforwardnet    | 2.10e-06 |
| 50000     | 1      | 20      | cascadeforwardnet | 5.12e-06 |
| 50000     | 2      | 5       | fitnet            | 1.07e-05 |
| 50000     | 2      | 5       | feedforwardnet    | 8.51e-06 |
| 50000     | 2      | 5       | cascadeforwardnet | 5.10e-06 |
| 50000     | 2      | 10      | fitnet            | 1.80e-06 |
| 50000     | 2      | 10      | feedforwardnet    | 3.51e-06 |
| 50000     | 2      | 10      | cascadeforwardnet | 1.85e-06 |
| 50000     | 2      | 20      | fitnet            | 5.51e-07 |
| 50000     | 2      | 20      | feedforwardnet    | 5.92e-07 |
| 50000     | 2      | 20      | cascadeforwardnet | 1.54e-06 |
| 50000     | 3      | 5       | fitnet            | 1.14e-05 |
| 50000     | 3      | 5       | feedforwardnet    | 1.35e-05 |
| 50000     | 3      | 5       | cascadeforwardnet | 1.80e-06 |
| 50000     | 3      | 10      | fitnet            | 1.84e-06 |
| 50000     | 3      | 10      | feedforwardnet    | 1.54e-06 |
| 50000     | 3      | 10      | cascadeforwardnet | 1.19e-06 |
| 50000     | 3      | 20      | fitnet            | 1.39e-06 |
| 50000     | 3      | 20      | feedforwardnet    | 1.23e-06 |