Aircraft trajectory prediction in crowded terminal areas

António Fallah

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor(s): Prof. Rodrigo Martins de Matos Ventura
              Prof. Cláudia Alexandra Magalhães Soares
External Supervisor: Eng. José Duarte Ribeiro Afonso

Examination Committee

Chairperson: Prof. Fernando José Parracho Lau
Supervisor: Prof. Cláudia Alexandra Magalhães Soares
Member of the Committee: Prof. Cláudia Rita Ribeiro Coelho Nunes Philippart

October 2019
"To most people, the sky is the limit. To those who love aviation, the sky is home."
Acknowledgments

I would like to begin by extending my deepest gratitude to my advisors, Professor Cláudia Soares and Professor Rodrigo Ventura from the Institute for Systems and Robotics in Instituto Superior Técnico who have been ever-present in teaching and supporting me throughout this thesis.

I am additionally grateful to Eng. Duarte Afonso for giving me the chance to develop this thesis with TAP Air Portugal and for his guidance in the air traffic management field. I would also like to thank all the team in the department of Safety and Security for granting me access to very valuable information.

Last but certainly not least, gratitude is extended to my family, who has always motivated me throughout my lifetime. In particular, I would like to thank to my parents, for giving me all the chances in life to succeed. My mother, whose endless worries helped me in taking care of me. My father, with a great passion for aviation, has always been by my side, or by a Facetime call distance, to help me with anything I needed. Of course, my brother for his medical and moral support. Finally, I would like to express my warmest gratitude to my girlfriend for her loving support. Her constant companionship and encouragements helped me, without any doubts, in becoming an Aerospace Engineer.
Resumo

No decorrer da última década, o tráfego aéreo cresceu por todas as partes do mundo e criou atrasos devido ao congestionamento do espaço aéreo. Para sincronizar as operações no ar e no solo, e, consequentemente, serem mais competitivas, as companhias aéreas necessitam de prever a trajetória das suas aeronaves e estimar, a curto-prazo, a hora de chegada das mesmas, com base no congestionamento em tempo-real do espaço aéreo na vizinhança dos aeroportos.

Baseado em trabalho anterior, é apresentado um modelo generativo de previsão de trajetórias de aeronaves em áreas terminais congestionadas. O modelo usa trajetórias reais que requerem uma etapa de pré-processamento para serem utilizadas pelo algoritmo de aprendizagem. Com base nas matrizes de covariância intra-cluster, o processo generativo usa um modelo de mistura gaussiano que irá gerar a trajetória posterior da aeronave e inferir sobre a hora de aterragem.

As trajetórias contidas no modelo foram complementadas por: (1) uma métrica que conta o número de aeronaves no espaço aéreo, e (2) uma variável binária definida pela direção do vento que permite inferir com maior precisão a pista em uso.

Finalmente, as trajetórias geradas foram testadas por especialistas de aviação num teste de Turing, revelando que são indistinguíveis das trajetórias reais.

As contribuições deste trabalho são: (1) a utilização de trajetórias estimadas para calcular a hora de aterragem, (2) o estudo da melhoria da precisão do modelo pela adição de informação sobre a utilização do espaço aéreo e o estado do vento, e (3) a implementação e avaliação do modelo usando dados reais.

Palavras-chave: Gestão de tráfego aéreo, predição de trajetórias, espaço aéreo terminal, inferência da hora de aterragem, modelo generativo
Abstract

During the last decade, the demand for air travel grew in all parts of the world which created delays due to airspace congestion. To trim ground with air operations, and therefore be more competitive, airlines need to predict the trajectory and the arrival time of their aircraft, based on the actual congestion of the airspace, to allow them to adapt their operation if necessary.

Building on prior research, we present a probabilistic generative model of aircraft trajectory prediction in crowded terminal areas completed with landing time inference. The model uses a dataset containing real trajectories that require completion and smoothing as a preprocessing step for the machine learning algorithm. Based on the intra-cluster covariance matrices and on held-out measurements, a generative model using a Gaussian model mixture will generate the posterior trajectory of aircraft and infer on the landing time.

The feature set was extended with: (1) a measure of the number of aircraft potentially sharing the airspace with each trajectory, and (2) a feature giving information on wind direction that allows our model to infer more accurately the active runway.

Finally, the generated trajectories were tested by aviation specialists in a Turing test revealing that they are indistinguishable from real paths.

The three contributions of this work are: (1) using estimated trajectories to compute time of landing, (2) the study of accuracy improvement by addition of two new features indicated by domain knowledge, and (3) to evaluate and implement the delay estimator on real data.

Keywords: Air traffic management, trajectory prediction, terminal airspace, landing time inference, generative model
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgments</td>
<td>v</td>
</tr>
<tr>
<td>Resumo</td>
<td>vii</td>
</tr>
<tr>
<td>Abstract</td>
<td>ix</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xiii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xv</td>
</tr>
<tr>
<td>Nomenclature</td>
<td>xvii</td>
</tr>
<tr>
<td>Glossary</td>
<td>xix</td>
</tr>
</tbody>
</table>

## 1 Introduction

1.1 Motivation | 1
1.2 Thesis contribution and objectives | 3
1.3 Thesis Outline | 3

## 2 Background

2.1 Terminal Airspace Organization | 5
2.2 Trajectory Prediction Algorithms | 8
2.3 Trajectory reconstruction methods | 9
2.3.1 Spline interpolation | 10
2.3.2 Convex optimization | 10
2.4 Clustering Algorithms | 11
2.4.1 Centroid-based clustering | 12
2.4.2 Distribution-based clustering | 12
2.5 Generative model | 13

## 3 Exploratory Data Analysis and data preprocessing

3.1 Trajectory data sources | 15
3.1.1 Demand Data Repository of EUROCONTROL | 15
3.1.2 Dataset from TAP Air Portugal | 18
3.2 Trajectory preprocessing | 22
3.2.1 Convex optimization problem derivation | 22
3.2.2 Regularization parameters estimation | 24
3.3 Feature engineering | 27
# List of Tables

3.1 Parameter estimation calculating the RMSE between the reconstructed trajectories and the measurements of TAP Air Portugal. 

3.2 Comparative table of the value of $R^2$ for the dataset considering all aircraft and the dataset only considering traffic landing in Lisbon airport.

5.1 Mean absolute error of the generative model for $T_{ldg}=35$ minutes compared with the baseline of TAP Air Portugal.

5.2 Evolution of $MAE_{ldg}$ and $RMSD_{path}$ with increasing values of $T_{ldg}$.

5.3 Evolution of $MAE_{ldg}$ and $RMSD_{path}$ for the different density features for the dataset considering only aircraft landing in Lisbon airport.

5.4 Evolution of $MAE_{ldg}$ and $RMSD_{path}$ for the different density features for the dataset considering all aircraft flying in Portuguese airspace.

5.5 Mean estimation error comparison and trajectory root mean square deviation evolution for the inclusion of the wind measures in the generative model.

5.6 Mean estimation error comparison and trajectory root mean square deviation evolution for each feature and the combined model for $T_{ldg}=45$ minutes.

5.7 Turing test results for the test conducted to aviation experts.
List of Figures

1.1 Average number of incoming flights to Lisbon airport over 24 hours ............................... 2
2.1 Free Route Airspace principle as implemented in Lisboa FIR ............................................. 6
2.2 Controlled airspace diagram showing the different type of it ............................................... 7
2.3 Graphical representation of Lisboa FIR and the TMA of Lisbon airport ............................... 7
2.4 Example of clustering of a small dataset ............................................................... 11
3.1 Number of flights per AIRAC cycle during one year ..................................................... 16
3.2 Density of points of the dataset around the airport .......................................................... 17
3.3 Most frequent waypoints inside one AIRAC cycle ......................................................... 18
3.4 Plot of 5 flights with raw data coming from EUROCONTROL dataset ............................ 19
3.5 Sporadic outliers in the dataset of TAP Air Portugal ...................................................... 20
3.6 Major faulty measurements recorded in the dataset of TAP Air Portugal .......................... 21
3.7 Clean trajectory where we can see constant latitude in consecutive timestamps (a) compared with a noisy trajectory where this particularity is not observed (b) ........................................ 21
3.8 Influence of the regularization parameters on the reconstructed trajectories ...................... 25
3.9 Trajectory reconstruction using varying parameters on the dataset of EUROCONTROL ......... 26
3.10 Reconstructed trajectories for the selected values for the parameters ............................... 27
3.11 Out of sample validation for the reconstructed trajectories ........................................... 28
3.12 Plot of the flight delay inside LPPC versus the number of aircraft inside the airspace ...... 29
3.13 Plot of the flight delay inside LPPC versus the number of aircraft inside the TMA of Lisbon airport .......................................................... 30
3.14 Visual representation of the circle centered in Lisbon airport passing by the aircraft of interest .... 31
3.15 Plot of the flight delay inside LPPC versus the number of aircraft within a circle centered in Lisbon airport .......................................................... 32
4.1 Elbow plot to estimate the optimal number of clusters ..................................................... 36
4.2 Cluster centers representation for k = 20 ............................................................................. 37
4.3 Random trajectories generated from a cluster landing in runway 03 ................................. 40
5.1 Plot of a generated trajectory compared with the real flight path flown ............................ 45
5.2 Empirical Cumulative Density Function of the estimation error of the implemented model compared with the baseline of TAP Air Portugal .................................................. 46
5.3 Posterior trajectory generated with noise for $T_{\text{ldg}} = 50$ minutes. ........................................ 48
5.4 Cluster centers for $T_{\text{ldg}} = 45$ minutes including the density feature. ................................. 49
5.5 Empirical CDF plot comparing the inference performance of the density model with the different
metrics of density for the generative model using $T_{\text{ldg}} = 45$ minutes. ............................... 50
5.6 Wrong runway inference done by the generative model. ......................................................... 51
5.7 Empirical CDF plot comparing the inference performance of the wind metrics for the generative
model using $T_{\text{ldg}} = 45$ minutes. ...................................................................................... 52
5.8 Empirical CDF plot comparing the inference performance of for the combination of the density
and wind features for the generative model using $T_{\text{ldg}} = 45$ minutes. ................................. 53
A.1 Code structure of the different .py files developed for the implementation of the model. ....... 61
Nomenclature

Greek symbols

$\mu$ Cluster mean.

$\epsilon$ Landing time inference error.

$\lambda_1, \lambda_2$ Regularization parameters.

$\Sigma$ Intracluster covariance matrix.

Roman symbols

$CO_2$ Carbon Dioxide.

$D_2$ Acceleration operator matrix.

$D_3$ Jerk operator matrix.

$K$ Number of clusters.

$MAE_{ldg}$ Mean absolute error of the predicted landing time.

$N$ Number of points of a trajectory.

$P$ Reconstructed trajectory.

$\hat{P}$ Measured trajectory.

$P_{post}$ Generated posterior trajectory.

$P_{real}$ Real trajectory.

$P^\alpha$ Held-out trajectory.

$p_i$ Point of a trajectory at a given timestamp.

$q$ Sampling rate.

$r$ Number of principal deviations from the cluster mean.

$RMSD_{path}$ Root Mean Square deviation of the generated flight path.

$t_{traj}$ Predicted landing time.
$t^{\text{real}}$  Real landing time.

$T_{ldg}$ Trajectory duration.

**Subscripts**

$F$  Frobenius norm.

**Superscripts**

$T$  Transpose.

-1  Inverse.

*  Adjoint.
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACARS</td>
<td>Aircraft Communications Addressing and Reporting System</td>
</tr>
<tr>
<td>ACAS</td>
<td>Airborne collision avoidance system</td>
</tr>
<tr>
<td>ACMS</td>
<td>Aircraft Condition Monitoring System</td>
</tr>
<tr>
<td>ADS-B</td>
<td>Automatic Dependent Surveillance-Broadcast</td>
</tr>
<tr>
<td>AIRAC</td>
<td>Aeronautical Information Regulation And Control</td>
</tr>
<tr>
<td>ANAC</td>
<td>Autoridade Nacional de Aviação Civil</td>
</tr>
<tr>
<td>ATC</td>
<td>Air Traffic Control</td>
</tr>
<tr>
<td>ATM</td>
<td>Air Traffic Management</td>
</tr>
<tr>
<td>ATS</td>
<td>Air Traffic Services</td>
</tr>
<tr>
<td>CDF</td>
<td>Cumulative Density Function</td>
</tr>
<tr>
<td>CTAS</td>
<td>Center — Trac Automation System</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>Density-based spatial clustering of applications with noise</td>
</tr>
<tr>
<td>DDR2</td>
<td>Demand Data repository</td>
</tr>
<tr>
<td>DTW</td>
<td>Dynamic Time warping</td>
</tr>
<tr>
<td>EASA</td>
<td>European Union Aviation Safety Agency</td>
</tr>
<tr>
<td>EDA</td>
<td>Exploratory Data Analysis</td>
</tr>
<tr>
<td>EM</td>
<td>Expectation Maximization</td>
</tr>
<tr>
<td>FAA</td>
<td>Federal Aviation Authority</td>
</tr>
<tr>
<td>FDIMU</td>
<td>Flight Data Interface Management Unit</td>
</tr>
<tr>
<td>FDR</td>
<td>Flight Data Recorder</td>
</tr>
<tr>
<td>FIR</td>
<td>Flight Information Region</td>
</tr>
<tr>
<td>FRA</td>
<td>Free Route Airspace</td>
</tr>
<tr>
<td>GMM</td>
<td>Gaussian Mixture Model</td>
</tr>
<tr>
<td>ICAO</td>
<td>International Civil Aviation Organization</td>
</tr>
<tr>
<td>IFR</td>
<td>Instruments flight rules</td>
</tr>
<tr>
<td>LCS</td>
<td>Least Common Subsequence</td>
</tr>
<tr>
<td>LPPC</td>
<td>ICAO code for Lisbon airport</td>
</tr>
<tr>
<td>LPPC</td>
<td>Lisboa FIR</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>LVO</td>
<td>Low Visibility operations</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean Absolute Error</td>
</tr>
<tr>
<td>METAR</td>
<td>METeorological Aerodrome Report</td>
</tr>
<tr>
<td>MLE</td>
<td>Maximum Likelihood Estimator</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>NEST</td>
<td>Network Evaluation Strategic Tool</td>
</tr>
<tr>
<td>OCC</td>
<td>Operation Control Center</td>
</tr>
<tr>
<td>RMSD</td>
<td>Root Mean Square Deviation</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Square Error</td>
</tr>
<tr>
<td>RVR</td>
<td>Runway Visual Range</td>
</tr>
<tr>
<td>SAAM</td>
<td>System for traffic Assignment and Analysis at a Macroscopic Level</td>
</tr>
<tr>
<td>STAR</td>
<td>Standard Arrival Route</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>TCAS</td>
<td>Traffic Collision Avoidance System</td>
</tr>
<tr>
<td>TMA</td>
<td>Terminal Manoeuvre Area</td>
</tr>
<tr>
<td>UIR</td>
<td>Upper Information Region</td>
</tr>
<tr>
<td>VFR</td>
<td>Visual Flight Rules</td>
</tr>
<tr>
<td>WQAR</td>
<td>Wireless Quick ACMS Recorder</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation

In 1903 the first flight of the the Wright Brothers captured the public imagination and set a wide range of possible applications of aviation [1]. World War I accelerated the development of aviation technology and, rapidly, the need to manage the airspace arose. Less than 120 years later, more than 25 thousand commercial aircraft carrying more than 4 billion people each year around the globe cross the skies above ground. This number will grow for the next 20 years, especially in emerging economies where the growth is expected to be around 10% per year. In contrast, developed economies, such as the United States or across Europe, are expected to have a growth of around 2% per year. In total, IATA forecasts predict 8.2 billion air travelers by 2037, doubling the number of passengers in less than 20 years time [2]. With the current number of flights, some airspaces are already saturated causing numerous delays across the network. At around 41%, the most significant contributor to aviation delay in Europe is a lack of ATC capacity. In recent years, total investments of 3.3 Billion USD were made in order to increase air traffic management capacities and meet the demand of the industry.

From the point of view of an operator, these delays are the cause of an important economic impact, threatening the profit of the airlines in a time where airfares tend to fall. Taking an average of 74 USD per delayed minute [3], airlines try to prevent and predict delays through statistical analysis, network representation and operational research combined with machine learning techniques.

Nevertheless, nowadays, the amount of historical data and real-time information available is valuable for the different actors playing a role in the industry. It is not a surprise that companies such as FlightRadar 24 or FlightAware 2 sell turnkey solutions to Airlines, e.g., to Emirates Airlines, KLM or Lufthansa, to monitor operations worldwide. These costly solutions may not meet the need of every operator and may be an overkill solution for the need of some airlines. It is the case of TAP Air Portugal that is keen to develop a tool that allows the company to predict if whether or not their aircraft need to loiter around Lisbon airport. There, their operation is heavily impacted by the saturation of the capacity of the airport regarding runway usage, available gates and

---

handling equipment. It is interesting to see the evolution of the number of landings in Lisbon airport over 24 hours. In Figure 1.1, we can see the plot of the average number of incoming flights per two hours during April 2019.

![Bar chart showing the average number of incoming flights to Lisbon airport over 24 hours. The plot indicates a high flow above 30 landings per two hours from 6:00 to midnight.](image)

Figure 1.1: Average number of incoming flights to Lisbon airport over 24 hours showing a high flow above 30 landings per two hours from 6:00 to midnight.

On the histogram, the number of flights landing at the airport seems to be relatively constant between 6:00 and midnight, ranging from 30 to 36 flights per two hours. It means that on average, during the peak hours of the day, an aircraft lands every 3 minutes. Supposing that each arrival is interleaved with a departure, there is one aircraft taking off or landing every 1 minute and 30 seconds, revealing the saturation of single-runway operations in Lisbon airport.

Predicting, at short-term, the arrival time of incoming flights, would allow TAP Air Portugal to adapt its operation in case a delay emerges in the final phase of a flight. Aircraft sequencing for landing can lead to delays over 15 minutes, during which the aircraft hold around the airport waiting for their clearance to land. At this point, TAP only knows that an incoming aircraft is loitering when the pilots communicate that information with the Operation Control Center (OCC) via ACARS (Aircraft Communications Addressing and Reporting System). If the OCC could predict that delay with some advance, the airline would be able to adapt the operation in order to reduce the impact to other flights, preventing a snowball effect on the entire operation. As an example, if TAP could predict 35 minutes in advance that an incoming flight will have to wait around 10 minutes before landing, the airline can request earlier a later slot for the outbound flight and request handling personnel at a different time in order to optimize the allocation of the limited resources on the ground. For example, catering, handling and cleaning crew can be allocated to another incoming flight instead of waiting while the original scheduled aircraft does not arrive, smoothing the overall operations.

Currently, conflict detection and resolution tools use trajectory prediction algorithms but do not make time inference to predict whether an aircraft needs to wait before landing and the consequent arrival time. As a result,
the purpose of this thesis is to use data science and machine learning techniques in order to meet the need of TAP Air Portugal to predict the arrival time of their aircraft during the approach towards Lisbon airport.

This thesis is developed in cooperation with the Operating Performance Team at TAP Air Portugal. This newly founded team inside the Operations Department lead by Engineer Duarte Afonso is responsible for evaluating the efficiency of processes, improving them and identifying opportunities to optimize the operations inside the company. Consequently, the work here presented fits in the scope of the team.

1.2 Thesis contribution and objectives

Based on previous work, we present an implementation and adaptation of previous work on trajectory prediction. Historical data on aircraft trajectories feed the data science pipeline and the model generates a posterior trajectory taking into account the previous $D$ position measurements of a determinate aircraft, the real traffic scheduled to land in an airport and weather conditions at the airport. The posterior trajectory then has an associated remaining flight time used to calculate the landing time of the aircraft at the airport. The landing time inference is based on the generated posterior trajectories and the influence of side information, airspace congestion and wind conditions at the airport, will be studied further on.

Before tackling the learning, generative and inference processes, we need to understand how air traffic is controlled and how do aircraft behave in terminal areas. This understanding can help the algorithm to return better results. In the second phase, the data that drive the model need to be filtered and prepared before being fit to train the algorithm. This work is adapted from [4], where an optimization problem is built to smooth, interpolate and extrapolate the noisy, low-resolution data coming from the different datasets. The optimization problem established is convex and therefore has a unique optimal solution: a reconstructed full trajectory.

The generative model is a probabilistic model based on a Gaussian Mixture Model derived in order to determine the aircraft trajectory from the current position and infer on the remaining flight time of the aircraft. Based on the current situation of the airspace and wind condition at the landing site, the proposed method should be able to determine, with high probability, if an aircraft will follow a shorter or longer route to the runway.

Under this topic, a publication with title “Prediction of aircraft trajectory in crowded terminal areas” is in preparation, to be submitted to the IEEE Transactions on Knowledge and Data Engineering.

1.3 Thesis Outline

This document is structured by six chapters organized as follows:

In Chapter 2, we explain the way that terminal airspace is structured and introduce previous work in trajectory prediction algorithms. In addition, a theoretical review is done to crucial notions that will be used over this thesis.

In Chapter 3, we start by making an Exploratory Data Analysis (EDA) of the data sets available that will allow the shaping of the learning scheme, definition of the data models to use, and necessary preprocessing steps. Also, a feature engineering subsection is added where the different metrics are studied in order to tie a link between the Portuguese airspace occupation and delay inside it. In addition, a metric informing on the wind condition at the airport will be presented.
Then, in Chapter 4 the learning process is presented where a K-Means learning setup is adopted to the data at hands and a subsequent inference mechanism supported on the Gaussian mixture model is carried out. Finally, landing time is computed from nominal speed and length of the trajectories.

In Chapter 5, we present and discuss the results of the simulations of the full model in Lisbon airport and compare the output of the algorithm with two metrics used by TAP Air Portugal to predict the landing time of their aircraft.

As conclusion, Chapter 6 allow us to wrap up the work developed during this thesis and explore some possibilities regarding future work.
Chapter 2

Background

This Chapter provides an overview of the current airspace organization around an airport and explains the basic standard procedures in order to bring an aircraft to land. Then, Section 2.2 provides a presentation on the current algorithms available in order to tackle the work proposed. Finally, Sections 2.3, 2.4 and 2.5 are used to introduce a theoretical statement on trajectory reconstruction methods, clustering algorithms and generative probabilistic models.

2.1 Terminal Airspace Organization

The necessity to control the airspace began when the first regular air routes entered in operations in early XX\textsuperscript{th} century in the United States of America. Nowadays, the bonfires, rotating beacons and flags firstly used to direct aircraft to land [5] evolved to a complex invisible infrastructure to our eyes.

Every airspace around the world is divided into Flight Information Regions (FIR) managed by the authority responsible for assuring that air traffic services (ATS) are provided to the aircraft flying within it. In Portugal, ATS provider is NAV Portugal E.P.E and supervised by ANAC - Autoridade Nacional de Aviação Civil [6, 7].

Portuguese airspace is divided into two FIRs. The larger Santa Maria FIR covering the Azores islands and the Atlantic ocean, and the smaller Lisboa FIR containing the Portuguese mainland and the Madeira archipelago. Depending on size and traffic, FIRs can be divided vertically where the lower section remains referred to as a FIR, but the upper section is referred to as an Upper Information Region (UIR) [7].

Lisboa FIR has the particularity of being a Free Route Airspace (FRA) defined as “a specified airspace within which users can freely plan a route between a defined entry point and a defined exit point, with the possibility of routing via intermediate (published or unpublished) waypoints” [8]. It allows the operators to plan shorter routes than before, where airplanes needed to follow specific air routes in order to cross a FIR. In FRA, flights need to ensure they enter and exit the airspace via defined entry and exit waypoints. Besides, flights cannot overfly danger or restricted areas. Figure 2.1 illustrates the concept of a Free Route Airspace.

Estimates show that FRA helps saving every day 3 000 tons of fuel consumption, translating to 10 000 tons of CO\textsubscript{2} per day, across Europe. The environmental benefits are unquestionable and by 2022, all FIRs in Europe will be Free Route Airspaces[8].
Figure 2.1: Free Route Airspace principle as implemented in Lisboa FIR. No mandatory routes exist in this type of airspace where aircraft are free to take the route that suits the best their need. Figure from [8].

The portion of the airspace we need to focus on is located inside the lower part of the FIR. Similarly to every FIR, Lisboa FIR is divided into various airspace classification. Generally, airspaces can be classified from A, where only aircraft following Instrument Flight Rules (IFR) can fly and all flights are provided with air traffic control service. To G, where flights are permitted and receive flight information service if requested. Lisboa FIR is classified “C”, “D” and “G” [9]. The majority of the airspace is classified as “C” meaning that separation is provided to all traffic, VFR and IFR, and a speed restriction of 250 Knots (Indicated Air Speed) that exist below 10,000 feet. In addition to being given a class, which specifies rules for flying, controlled airspace may be further defined by its ‘type’ depending on where it is and the function it provides.

As seen in figure 2.2, Upper Air Routes sit the highest. Usually, between FL250 and FL460 it is a Class ”C” controlled airspace, and all traffic is subject to full Air Traffic Control service. Just below, Airways are corridors of airspace connecting the Control Areas under them. If this type of controlled airspace is located above one or a group of airports, a part of it is called Terminal Control Areas containing the Terminal Maneuver Area (TMA). They are located above the Aerodrome Control Zone that control the immediate vicinity of aerodromes [6].

Even though Lisboa FIR is a Free Route Airspace, the guidance of aircraft to Lisbon Airport needs to be done in an ordered manner to deconflict potentially conflicting traffic. Every airport in the world has Standard Terminal Arrival Routes (STARs) published inside the Terminal Control Area that aircraft need to follow unless instructed by ATC, giving them radar vectors to navigate. These STARs begin with a fix waypoint and the procedure ends on the final approach to a runway. In summary, the STARs help converging incoming traffic from various directions to the TMA in order to start the sequencing of the incoming aircraft to land. In figure 2.3 we can see the shape of Lisboa FIR and the TMA of Lisbon airport.

Now that we have established the guidelines of how airspaces are arranged, we can conclude that they are
Figure 2.2: Controlled airspace diagram showing the different types of it. Terminal Maneuver Areas are located inside Control Areas above one or more airport. Figure from [6]

Figure 2.3: Lisboa FIR covering the mainland of Portugal up to Madeira Archipelago (a) and the TMA of Lisbon airport in the extended vicinity of the airport (b). Figure from [10].

heavily controlled and regulated. Therefore we need to guarantee that the model reflects the STARs published for Lisbon Airport in order to generate plausible trajectories.
2.2 Trajectory Prediction Algorithms

In the context of this work, as landing time inference is based on the trajectory prediction, it is mandatory to review some of the existing algorithms in the trajectory prediction field. The application of the vast majority of these algorithms is conflict detection and resolution between aircraft such as the widely used Traffic Collision Advisory System (TCAS). Also, trajectory prediction is a core element of future air transportation system which is intended to improve the operational ability and the predictability of air traffic, necessary in order to increase airspace capacity to meet the increase in air traffic expected for the next two decades.

An early approach to predict the trajectory of aircraft is by looking at the current state of an aircraft and propagate that estimate integrating with respect to time the physical equations of motion. The current states of the aircraft are estimated by tracking the aircraft position data received from the primary and secondary surveillance radar systems. The limitation of this kind of algorithm lies on the difficulty on getting the flight controls input needed to accurately estimate of the present states of the aircraft [11].

A way to improve these predictions is by synthesizing trajectories by combining individual segments defined by different modes of operation - fast, nominal and slow - and estimating vertical paths based on the weight of the aircraft. This trajectory synthesis method is used in the Center–Terminal Radar Approach Control Automation System (CTAS) developed at NASA Ames Research Center for air traffic control automation. It is a widely used system in the US where CTAS generates computer advisories regarding flight paths that help controllers to produce safer air traffic management over an extended terminal area [12].

Another line of work is to learn Bayesian network statistical representations of dynamic variables from historical radar data. This algorithm construct aircraft trajectories that are statistically similar to those observed in the radar data. The result is a framework from which trajectories can be generated and used in simulations to provide estimates of collision risk between various aircraft [13]. In conjunction with probability flow, this kind of algorithms is the baseline of more advanced trajectory prediction models used in aviation systems, namely in the TCAS and ACAS X, the next generation of TCAS [14].

In addition, hybrid dynamics models of aircraft are used, taking into account both continuous and discrete dynamic behavior. The aircraft position and velocity change continuously (continuous dynamics), whereas the flight segment changes discretely (discrete dynamics). It is possible to propagate the state of the airplane through a stochastic hybrid system model to perform conflict detection. Combining the dynamical system model with a method for inferring on the navigational intent of the aircraft leads to a commonly used set of modeling tools for aircraft trajectory prediction [15]. A more recent proposed method incorporates the hybrid dynamics model with a Bayesian intent model. The main key in this algorithm is to learn the expected behavior of pilots from observations of previous aircraft trajectories [16].

The majority of the models presented so far focus on learning probabilistic trajectory models for aircraft in unstructured airspace. Unstructured airspace refers to airspace designs that offer operators complete freedom in path planning. In terminal regions, specific paths are used in order to sequence aircraft into landing in an airport. For that reason, modeling aircraft trajectories in terminal regions cannot be done with models for unstructured operation.

The easiest way to solve this issue is to combine the models for unstructured airspace with the published
Tackling the trajectory prediction in terminal airspaces, a well-known method involves clustering turning points, defined as spacial points where a change of heading occurs. These points are clustered and then the trajectories are reconstructed using these turning points. The least common subsequence (LCS) algorithm is used to find similarities between trajectories. The conducted test found that the turning point model does not perform well on real, noisy radar data. A Gaussian hidden semi-Markov model was, for that matter, introduced to represent the transitions between turning points. This approach only works well on smaller airports where the number of turning points is low [17].

Instead of clustering turning points, it is possible to cluster entire trajectories. This is the main key behind air traffic flow modeling of en-route traffic to predict airspace congestion [18]. Unfortunately, trajectories are different in length meaning that the majority of algorithms used in clustering cannot be applied as they require that all trajectories have the same length. This issue can be addressed by using the Density Based Spatial Clustering of Applications with Noise (DBSCAN). By creating a common time to landing, DBSCAN clusters time-aligned trajectories to discover traffic patterns [19]. Another way to cluster the trajectories is by using the Dynamic Time Warping algorithm (DTW) that find similarities between varying length trajectories by warping time and aligning all trajectories to solve the length issue [19].

In [4], the line of work consists of learning a probabilistic trajectory model in terminal airspaces uniquely based on position measurements. No other information apart of latitude, longitude, altitude and time is available on the dataset used. The first issue the authors address is the different length of trajectories, by time aligning the trajectories at landing and select the points available within a fixed time-frame for all flights. Firstly, they interpolate, extrapolate, and smooth the trajectories before being clustered using a K-means algorithm. The method then fits a generative model to the clusters based on the intra-cluster covariance matrices.

Our approach builds on the one presented in [4]. Nevertheless, the paper is limited to an area of a 5NM radius around the airport. To meet the proposed work requirements, we need to increase this radius to be able to predict the landing time of aircraft with a large enough time-frame to provide a useful prediction for the operator. In addition, the prediction of landing time needs to be added to the existing model, as well as the airspace occupation and wind information.

To better understand this approach, a background on trajectory reconstruction, a review on clustering algorithms, and on generative probabilistic models will be done before the derivation of the used model.

### 2.3 Trajectory reconstruction methods

The study of trajectory data is not simple and many issues need to be solved prior to learning from the trajectories in the dataset. As seen in section 2.2, authors use different methods and algorithms to preprocess the data before stepping to the clustering phase.

A flight path is a sequence of space-time data points that combined, form a trajectory. The variable way these...
points are recorded present a real challenge to the clustering algorithms that will analyze whole trajectories and find similarities between each the data points. The major two difficulties authors encounter are the different lengths of trajectories and the different recording rates. These issues are solved by resampling the trajectories to a fixed number of points, and similarly, to prevent misalignment between points, a sampling rate is introduced.

### 2.3.1 Spline interpolation

A well known technique is interpolating the current points constituting a trajectory using splines, and then introduce a sampling rate in order to ensure the time distance between two reconstructed points of the trajectory [20].

The most commonly used splines are cubic, i.e, using 3rd degree polynomials. This way, the interpolant function of data points is twice continuously differentiable, producing smooth trajectories.

Given a set of data points, \(x(\tau_i), \ldots, x(\tau_m)\), the interpolant function \(f\) is constructed to agree with \(x\) for each \(\tau\). On each interval \([\tau_i, \tau_{i+1}]\) we have \(f(\tau) = P_i(\tau)\), where \(P_i(\tau)\) is the 3rd degree polynomial defined as

\[
P_i(\tau) = c_{1,i} + c_{2,i}(\tau - \tau_i) + c_{3,i}(\tau - \tau_i)^2 + c_{4,i}(\tau - \tau_i)^3
\]  

(2.1)

with

\[
c_{1,i} = P_i(\tau_i), \quad c_{2,i} = P_i'(\tau_i) \]
\[
c_{3,i} = P_i''(\tau_i)/2, \quad c_{4,i} = P_i'''(\tau_i)/6
\]  

(2.2)

where \(P_i', P_i'', P_i'''\) denote the first, second and third derivative of \(P_i(\tau)\). The full calculation of these parameters can be found in [20].

Having fit the splines for all coordinates of a trajectory, uniform resampling can be done by evaluating the splines at uniformly spaced \(\tau\). If \(N\) points are desired, splines are evaluated at

\[
\tau_i = \tau_1 + \frac{\tau_N - \tau_1}{N-1} (i - 1), \quad i = 1, \ldots, N
\]  

(2.3)

This way, the trajectories are reconstructed based on the splines calculated from the original data points with a constant sampling rate throughout the trajectories. Unfortunately this kind of approach does not take into account the attitude of the airplanes and the interpolation is done only by looking at the measured points. Without introducing smoothing splines, we are not able to control the interpolation parameters to tune the reconstructed trajectories. Another approach used in data completion is by using convex optimization problems, where an objective function can be derived and regularization parameters introduced, in order to guarantee that a reconstructed trajectory mimics successfully the dynamics of an airplane.

### 2.3.2 Convex optimization

Convex Optimization can also be used to perform data completion. In our case, reconstruct the trajectories and ensure they have the same length and same sampling rate.

As in [21], a general mathematical optimization problem is defined as
\[
\begin{align*}
\text{minimize} \quad & \quad f_0(x) \\ 
\text{subject to} \quad & \quad f_i(x) \leq b_i, \quad i = 1, \ldots, m.
\end{align*}
\] (2.4)

in which \(x = (x_1, \ldots, x_n)\) is the optimization variable, \(f_0 : \mathbb{R}^n \rightarrow \mathbb{R}\) being the objective function and \(f_i : \mathbb{R}^n \rightarrow \mathbb{R}, \quad i = 1, \ldots, m\), the constrain functions limited by the bounds \(b_1, \ldots, b_m\). The solution of the problem described in 3.5 is the vector \(x^*\) which attain the minimal value for \(f_0(x)\) while respecting the constrains of the problem. This type of problems can be defined as **convex** or **non-convex**. To be convex, the objective function and the inequality constraints must be convex.

Using this method to reconstruct the trajectories, we are free to construct the objective function that best fits our needs. By deriving an objective function that takes into account the data points in the trajectories and control the aircraft dynamics with regularization parameters, we are able to reconstruct and fine tune the parameters to guarantee that the trajectories reflect possible flight dynamics. This second approach to data completion is, therefore, more adequate to our low resolution dataset and very variable sampling rate.

### 2.4 Clustering Algorithms

Clustering is the assignment of a set of observations into subsets called clusters, so that observations in the same cluster are more similar to each other than to those in other clusters. It is an unsupervised learning method that finds similarities between observations without pre-existing patterns or information about the data. Figure 2.4 illustrates the clustering of a dataset into three different clusters.

![Figure 2.4: Example of clustering of a small dataset in three different clusters.](image)

There are various types of algorithms that allow clustering data. The main categories are Hierarchical Clustering, Centroid-based clustering, Distribution-based clustering, and Density-based clustering. Each method can return different solutions for the same dataset as they work using different metrics in order to measure similarities between the data. Regarding trajectory clustering and the final objective of this work, two separate clustering approaches will be considered: centroid-based clustering and model-based clustering. This section is based on information found in [22].
2.4.1 Centroid-based clustering

Centroid-based clustering use cluster centers that represent a specific cluster. Depending on the data clustering, the center may be a scalar or a vector which may not necessarily be a member of the data set. The most common algorithm that falls into this category is the K-Means algorithm. It is a simple and intuitive method: given a dataset and the number of desired clusters determined by the user, the algorithm iterates a assignment step followed by a centroid \( \mu_k \) update step, until it finds a locally optimal clustering assignment:

1. In the assignment step, the algorithm sets, for all \( i \),

\[
    c^i := \arg \min_{k} \| x_i - \mu_k \|^2
\]

where \( c^i = k \) if the \( i \)th trajectory belongs to cluster \( k \). For each trajectory, the algorithm assigns each trajectory to the closest centroid by computing the squared Euclidean distance.

2. In the centroid update step, the algorithm computes

\[
    \mu_k := \frac{\sum_{i=1}^{K} 1\{c^i = k\} x_i}{\sum_{i=1}^{K} 1\{c^i = k\}}
\]

for all \( k \), where \( 1\{c^i = k\} \) is an indicator function, equal to 1 if \( c^i = k \) and 0 otherwise. This step updates the location of the centroid of the cluster selected in the assignment step.

These two steps are repeated for all trajectories until a convergence criterion is met.

The issue using this algorithm lies in the time-complexity of the method, classified as being \textit{NP-hard}, Non-deterministic Polynomial-time Hard. Exponential time is required in order to fully achieve the final result of the clustering process. This issue is surpassed by looking for approximate solutions to the problem.

K-means algorithm is very dependent of the cluster centers initialization as the number of iterations is directly linked to these centroids. The idea behind a good initialization is to choose an initial set of clusters as spread-out as possible. K-means++ initialization method chooses centroids randomly from training data, but with weights according to their distance to other centroids. Then, K-means is run until the assignments no longer change between two iterations. Time-complexity is linear with \( K \) and it can be shown that the converged solution is at most \( \log(K) \) worse than the optimal solution.

2.4.2 Distribution-based clustering

This clustering model, closely related to statistics, is based on probabilistic distribution models. Clusters are defined as objects belonging most likely to the same distribution which is different in relation with K-means. In K-means, each sample can only belong to one cluster, corresponding to what is called hard clustering. Distribution-based clustering, such as Gaussian Mixture Model (GMM), perform soft clustering, allowing a trajectory to belong to multiple clusters, assigning a probability to each case. Another difference is that GMMs are generative models, that is, they model the data generation process in order to categorize it, while K-means is a discriminative method categorizing data directly based on a distance metric.
It is a multivariate Gaussian density with unknown parameters $\mu$ and $\Sigma$. These parameters can be estimated using the Maximum Likelihood Estimates (MLEs), or the approximate Expectation Maximization (EM) algorithm as follows:

1. **Expectation step** - Given the current parameters, compute

$$
\tau_{ik} = \frac{\pi_k \phi(x_i; \mu_k, \Sigma_k)}{\sum_{l=1}^{K} \pi_l \phi(x_i; \mu_l, \Sigma_l)}
$$

2. **Maximization step** - Update the parameters by considering

$$
\pi_k = \frac{1}{K} \sum_{i=1}^{K} \tau_{ik}, \quad \mu_k = \frac{\sum_{i=1}^{K} \tau_{ik} x_i}{\sum_{i=1}^{K} \tau_{ik}}
$$

$$
\Sigma_k = \frac{\sum_{i=1}^{K} \tau_{ik} (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{K} \tau_{ik}}
$$

where:

- $x_i$ represents a trajectory
- $\pi_k$ is the frequency of cluster $k$
- $\mu_k$ is the center of cluster $k$
- $\Sigma_k$ is the covariance matrix of cluster $k$

If we ignore the updates to $\mu_k$ and $\Sigma_k$, we get the same basic structure as in K-means: assign data points to classes, and update the means according to the assigned classes.

To gather things up, K-means algorithm makes hard clustering on each iteration, with each trajectory assigned to one class, while the EM algorithm uses soft clustering, probabilistic assignments for each trajectory.

For our case, we are particularly interested in learning the spacial similarities between the different trajectories in our dataset. We can admit that trajectories that are close to each other will belong to the same cluster, so a hard clustering of our trajectories seems to fill our needs. It is possible to achieve hard clustering from both GMM and K-means. The GMM is particularly interesting for trajectory generation, and as K-Means is a special case of EM, the parameters estimated with the K-means algorithm can be used in this generative model.

### 2.5 Generative model

The generative model will allow us to make a prediction of the posterior flight path marginalized on consecutive measurements that will initialize the posterior trajectory. The model used in [4] is based on a Gaussian mixture model. A Gaussian mixture is a probability distribution that is comprised of several Gaussians. In a machine learning context, the number of Gaussian distributions is imposed by the number $K$ of clusters in the model. Each Gaussian $k$ in the mixture is defined by a mean $\mu_k$ and a covariance matrix $\Sigma_k$ where $k$ follows a binary distribution over all clusters.
\[ k \sim \text{Categorical}(K) \]
\[ p \sim N(\mu_k, \Sigma_k) \]  \hspace{1cm} (2.5)

One benefit of modeling the trajectories using GMMs is that conditioning a Gaussian distribution on measurements gives another Gaussian distribution. The marginal distribution of a Gaussian being another Gaussian, the first measurements of the trajectory come from a Gaussian distribution. Then, the posterior trajectory conditioned on the initial measurements follows a Gaussian distribution. Iterating this process multiple times, to build a full trajectory using this model is done by emitting multiple Gaussian distributions dependent on the past.

A trajectory can therefore be modulated by a multivariate normal distribution where the vector \( p = (p_1, p_2, \ldots, p_n) \) is the predicted trajectory that follows a normal distribution on a cluster \( k \).
Chapter 3

Exploratory Data Analysis and data preprocessing

In this chapter, we introduce the different datasets that will be the base of our algorithm in order to achieve the proposed work. We derive the different components constituting our model, beginning with the optimization problem, then the clustering method, followed by the probabilistic generative model and how landing time estimation is made.

3.1 Trajectory data sources

Prior to building a model, we need to know what kind of data is available. As said in section 1.1, independent tracking companies such as FlightRadar24 and FlightAware collect ADS-B signals from aircraft via an extensive network of receivers in the world that allows them to build global flight tracking services providing real-time information about thousands of aircrafts around the world. They have all kind of data about the flights, starting with the aircraft type, registration, origin, destination, as well as spatial position. This information is very valuable and they managed to create real tools and solutions for worldwide operators. Being able to gather data from this kind of extensive data source would be an excellent start for the algorithm. As we know, the more data is fed to a model, the more accurate it can be. Unfortunately, the data from this kind of source is costly to get and therefore is not available without monetary compensation. Open-source databases like OpenSky Network have a limited number of ADS-B stations across Europe, so therefore the data collected by them may not be complete and may not suit our needs.

3.1.1 Demand Data Repository of EUROCONTROL

Demand Data Repository from EUROCONTROL (DDR2) stores historical traffic data in a compressed format. It is a pervasive database that provides data to various tools built by EUROCONTROL, namely SAAM (System for traffic Assignment and Analysis at a Macroscopic level) and NEST (Network Evaluation Strategic Tool). SAAM is used for operational planning purposes to optimize strategic traffic flows, to design the route network and airspace, and to analyze past and future traffic flows. NEST is a simulation software for network capacity planning and
airspace design. With these tools, EUROCONTROL provides air traffic management stakeholders an accurate picture of the pan-European air traffic demand. The data set files in DDR2 are organized by AIRAC (Aeronautical Information Regulation And Control). Since the aviation environment data is constantly changing, from airspace structures and revised routes, changes in navigation aids, amendments to STARs, AIRAC cycles were adopted by ICAO in the 1960s so that the entire air traffic management community could use the same dataset. Each AIRAC cycle is 28 days long, so each year is composed by 14 AIRAC cycles.

On average, between 25 000 and 35 000 flights are operated each month in Europe, where the IATA summer starts on the last Sunday of March and ends on the last Saturday of October - is the busiest period of the year. The amount of data recorded by EUROCONTROL is therefore immense. In order to optimize the download and the size of the dataset, we selected the flights landing in Lisbon airport as our model will be tested for the operations of TAP Air Portugal in that airport. In order to have a picture of the operations of the airport over a whole year, we chose to download the AIRAC cycles 1806 to 1906. They account for all flights that landed in Lisbon from June 2018 to June 2019. Globally, data in excess of 118 000 flights were downloaded. In Figure 3.1, it is possible to see that the cycles 1812 to 1903 are the ones with the least number of flights, corresponding to IATA winter.

![Figure 3.1: Number of flights per AIRAC cycle where we can clearly see the the lower number of operated flights between cycles 1812 to 1903 corresponding to the less busy IATA winter season.](image)

The data downloaded is formatted under .so6 files and can be opened by any raw text editor. Each file corresponds to one day of operation. Each line of the file corresponds to one segment of a flight. Each segment is delimited by an existing route points or by a "special" point created by EUROCONTROL. The information available for each segment is the Latitude, Longitude, Flight level, time of the beginning and the end of the segment. Each line is complemented by the date of operation, origin and destination of the flight, the aircraft type, callsign and segment length. A comprehensive exploratory data analysis (EDA) was made in order to understand the data available collected by EUROCONTROL.

Looking at a .so6 file, information about all flights are stacked in an ordered manner. Flight can be delimited by looking at the counter of the number of segments of each flight, where the counter begins at 1 for each flight.
When running code to understand if there is missing data inside the dataset, we quickly understand that the dataset presented is complete. The information of the end of one segment is the same as the data of the beginning of the following segment. Besides, all entries contain data letting us imagine that if data is missing, the segment is not recorded or the data of the beginning of the segment is the same as the end of the segment. The second guess is supported by the fact that in our dataset we recorded 420 segments with 0 NM, meaning that the endpoint of the segment is the same as the beginning one. In addition we can see that there is a significant discrepancy regarding that metric as some segment lengths are minimal, under 1km, and others are over 300km. To try to understand how points are distributed around Lisbon Airport, we have plotted the segment duration (time elapsed between two measurements) with respect to the distance of Lisbon airport.

![Figure 3.2: Evolution of the duration of the segments in the dataset with distance to Lisbon airport. It is possible to observe that the segment duration increase with the distance to the airport means that the density of points in the dataset is greater around the airport. Horizontal lines are due to the discretization of time (in seconds) and the vertical lines correspond to fix points.](image)

In Figure 3.2 it is clear that the density of points around Lisbon airport is large. We can see a clear break at around 300km from the airport where the majority of the segment duration are much larger than when the initial point of the segment is closer to 300km from the airport. The horizontal lines correspond to the discretization of the time in seconds. The vertical lines correspond to fix points. The majority of these vertical lines can be well seen between 300km and 450km. They suggest that the reported points are entry points to Lisboa FIR as it is a requirement to overfly an entry point to fly in a FRA. From around 400km to 600km, it seems to be a logarithmic trend followed by the segment duration variable as the distance to the airport increases. Some tests were conducted in order to determine and explain the trend but no clear explanation was found. Furthermore, to confirm that the majority of points in the dataset is around Lisbon airport, we have made a histogram of the frequency of the waypoints in the data set.

In Figure 3.3, we can see the points that were overflown more than 100 times around one AIRAC cycle. All of these 49 points are inside Lisboa FIR. The ones with the highest count, such as PESEX, UPKAT, FTM and...
INBOM are navigation points contained in STAR procedures of Lisbon Airport. In addition, some other points such as ABUPI, CCS and BEXAL are entry points to Lisboa FIR. These observations confirm our initial statement that the majority of recorded points are located in the extended vicinity of the airport. In addition, the algorithm did not record any special point created by EUROCONTROL meaning that they are unique and have no important meaning in this work. We looked after outlier measurements inside the dataset, namely in time and position and no glitch were found. It is almost certain that the data available via DDR2 is filtered prior to be available for download.

The next step is to plot some flights to verify the validity and accuracy of the measurements available. To do so, gmap library for Python was used to plot the trajectory of the flights using Google Maps background to have a more meaningful background representation.

The 5 flights plotted in Figure 3.4 are random flights from our dataset. It is clear that despite the conclusion made earlier, when we determined that the majority of the points of the dataset are around the arrival airport, the resolution of the trajectory is not very high. Trajectories are sampled as straight line segments with sharp changes in heading that are not plausible and are not feasible trajectories. In order to have a bigger accuracy on the real aircraft trajectory, we need to resample the raw data, increase the frequency of the points and approximate the data available to the real trajectories followed by the airplanes on their approach.

### 3.1.2 Dataset from TAP Air Portugal

Being this thesis done in cooperation with TAP Air Portugal, the airline was able to provide data coming directly from its aircraft. The data collected comes from the WQAR - Wireless Quick ACMS Recorder - onboard each aircraft. It is a system that allows operators to access flight data by transmitting it wirelessly over a cellular connection while on the ground.
Figure 3.4: Plot of 5 flights with raw data coming from EUROCONTROL dataset. The trajectories have very low resolution which is translated to an inaccurate plot of the flight path.

The data acquired by the WQAR is provided by the **Flight Data Interface Management Unit - FDIMU**-, processing the data coming from the sensors and feeds the processed data to the **Flight Data Recorder - FDR**. In addition, a back-up copy of the data collected by the WDAR is encrypted and stored in **Compact Flash** memory cards onboard the aircraft. This system is customizable by the operators and can record as much data as the FDR if instructed to do so.

The dataset that was presented to us was composed by 2845 .xlsx files each corresponding to data from one flight operated between June 2018 and June 2019. The dates were chosen by us to match the timeframe of the dataset of EUROCONTROL. In compliance with the regulation of the airline, the data that was given to us was anonymized, meaning that the specific date of the operation of a flight is unknown to us. We had access to the day of the week of the operation, flight number, longitude, latitude and altitude of the aircraft at a specific time. In addition, the flight phase was included for each measurement. Similarly to the dataset of EUROCONTROL, an EDA of the dataset presented by TAP was made in order to understand how the data is structured.

First and foremost, we should expect that data coming from a parallel system to the FDR aircraft would be precise and the write rate high. In part, this statement is true. The .xlsx files show a consistent 4-second interval between measurements, from the moment an aircraft turns on the avionics until the moment an aircraft reaches its final parking position and turns off the engines. Looking closely at the data, we see that each timestamp has four measurements. At first impression, we can think that it is a record redundancy to make sure measurements are made for a specific time, but, in fact, each measurement corresponds to one second. In conclusion, the record
rate is 1 Hz. What happens is that the clock of the WDAR refreshes every 4 seconds, so the data transmitted to the ground station translates in having four measurements per timestamp. This issue can be easily addressed by rewriting the timestamps with 1-second interval.

Regarding the position measurements of latitude, longitude and altitude, plotting some flights let us conclude that the measurements of some flights are well-behaved and others have a significant amount of noise and outliers in the measurements.

In Figure 3.5, the outliers present on this flight from Brussels are sporadic and can be overcome by merely eliminating these wrong measurements.

In contrast, other flights have a large number of erroneous measurements that are characterized by a sequence of points having a constant longitude or latitude. The department of Safety & Security at TAP Air Portugal was questioned about these outliers. They justified these glitches by stating that the data recorder is raw, with no filter coming from the sensors. We can therefore assume that these errors are due to sensor inaccuracies and writing errors. In figure 3.6 we can see the erroneous information plotted, where jumps in position occur and points with constant longitude or latitude are displayed.

Another error contained in the dataset is a time inaccuracy that may be present at the beginning of the record in each flight. As not all sensors and systems start at the same moment in an aircraft, the clock inside WDAR may not have been updated since the end of the last flight. When this event happens, the time shown in the first few

Figure 3.5: Sporadic outliers (in red) in position measurements of a flight coming from Brussels. This issue can be overcome by filtering punctual outliers.
Figure 3.6: Major faulty measurements recorded in the dataset of TAP Air Portugal. This kind of outlier measurement is characterized by constant latitude or longitude.

measurements is the last timestamp recorded during the precedent flight.

One similarity among the flights that contain no significant glitch is that in two consecutive timestamps, longitude or latitude is constant. This can be seen in Figure 3.7, where we can see that this particularity does not stand for flights containing major glitches of position having the noisy measurements (b).

Figure 3.7: Clean trajectory where we can see constant latitude in consecutive timestamps (a) compared with a noisy trajectory where this particularity is not observed (b).

It is clear that we have two very distinct data sets. The main difference is their size. The first one, coming from the Data Demand Repository of EUROCONTROL, is much larger, containing all flights landing in Lisbon.
Airport during one year of operation. In this dataset, the number of measurements per flight is uneven, and the record frequency is not standard throughout one flight. We have seen that the majority of the measurements in the dataset are located in the vicinity of the arrival Airport. One positive point in that dataset is that the measurements are very clean, with little to no noise. It is possible to resample these trajectories to add information in the dataset without losing or changing the initial information in it.

In contrast, the dataset of TAP Air Portugal is much smaller and contain only the flights of the company. The operation throughout the year is not complete due to confidentiality policies and the data that recreate the flight trajectory is not very precise due to a significant number of glitches. In the end, it seems that the dataset of EUROCONTROL is the most adequate when we look at the requirements of this work. The dataset of TAP Air Portugal may be used to compare the trajectories of their aircraft with the generation made by our model.

Nevertheless, we can not learn the model with the dataset of EUROCONTROL before processing the trajectories. It is a required step that will increase the number of points in each trajectory so that our model can learn from more information. In addition, resampling the trajectories using convex optimization will us to smooth, extrapolate and interpolate the trajectories to ensure that they all have the same number of points.

3.2 Trajectory preprocessing

3.2.1 Convex optimization problem derivation

We presented in section 2.3 a background on trajectory reconstruction methods. As established, to reconstruct the trajectories, we need to minimize a convex objective function over its domain. The goal of the optimization problem is to get a reconstructed trajectory that fit the requirements of the clustering algorithm, K-means, and reflects plausible flight dynamics.

The raw trajectories in the dataset of EUROCONTROL have a different number of measurements that need to be uniformed by a common length in seconds, $T_{ldg}$. That parameter is the timeframe in which the algorithm will be able to predict the arrival time of an aircraft. Using a larger $T_{ldg}$ will allow the operator to predict arrival time sooner. Unfortunately, as seen in subsection 3.1.1, the density of points in the dataset decrease as the distance to Lisbon airport increases. That means that if $T_{ldg}$ is sufficiently large, the resolution of measurements can drop to a certain extent that the measurements in the dataset will not be able to produce well-reconstructed trajectories. A sampling rate of $q$ (in seconds) is introduced in order to control the number of points in each trajectory. The optimization variable is the reconstructed trajectory

$$P = (p_1, p_2, ..., p_N), \quad N = \frac{T_{ldg}}{q}$$

where N is the number of spatial points $p_i = (lat_i, lon_i, alt_i)$ making $P \in \mathbb{R}^{N \times 3}$.

The key here is to build an objective function that not only takes into account the measured trajectories in the dataset but considers as well the maneuverability of the aircraft to recreate real plausible trajectories. First, we want to ensure data fidelity to the measurements in the dataset. It means that the reconstructed trajectories need to follow the general path of the flight. To do so, we use the least-squares regression method with the data fidelity term $\|AP - \hat{P}\|_F^2$, where $F$ defines the Frobenius norm. The vector $\hat{P} \in \mathbb{R}^{N \times 3}$ is the measured vector of points.
coming from the raw dataset. It is the trajectory $P$ that needs to be reconstructed by solving the optimization problem. The diagonal matrix $A \in \mathbb{R}^{N \times N}$ is introduced in the formulation of this term to control which points of $P$ are completed. If there is a measurement at time $i$, $a_{ii}$ is one. Otherwise, $a_{ii}$ is zero, so the optimization process will add information relative to missing points in the reconstructed trajectory $P$.

Now the issue lies in recreating feasible air maneuvers based on the measured trajectory. Introducing an acceleration term will allow us to control the attitude of the aircraft using previous measurements as guidelines on how the aircraft performs. Mathematically this is implemented by the term $\|D_2 P\|^2_F$, where $D_2$ is the discrete acceleration operator where each line $i$ of the matrix is defined by

\[(D_2)_i = e_i - 2e_{i+1} + e_{i+2}, \quad \text{for} \quad i = 1, \ldots, N - 2\]  

with $e_i$ denoting the $i$th standard unit vector. In addition, it is possible to control the rate of change of the acceleration by introducing the third-order difference matrix, also known as the jerk operator. Similarly to the acceleration matrix, this operator is defined by

\[(D_3)_i = -e_{i-1} + 2e_i - 2e_{i+2} + e_{i+3}, \quad \text{for} \quad i = 1, \ldots, N - 4\]  

and the term to be added in the objective function is $\|D_3 P\|^2_F$. Matrices $D_2 \in \mathbb{R}^{N-2 \times N}$ and $D_3 \in \mathbb{R}^{N-4 \times N}$ are banded matrices and the trajectory reconstruction can be tuned by multiplying the matrices by two regularization parameters $\lambda_1$ and $\lambda_2$.

Summing the terms, the objective function is

\[f(P) = \|AP - \widehat{P}\|^2_F + \lambda_1 \|D_2 P\|^2_F + \lambda_2 \|D_3 P\|^2_F\]  

In conclusion, the first term of the function in (3.4) is the term that refers to fidelity to the measurements and the last two terms control the smoothing of the trajectory. The optimization problem is unconstrained and the final formulation of the optimization problem is

\[
\minimize_{P} \quad \|AP - \widehat{P}\|^2_F + \lambda_1 \|D_2 P\|^2_F + \lambda_2 \|D_3 P\|^2_F.
\]  

Using least-squares, the problem stated in (3.5) has an analytic solution as the objective function can be expressed as the convex quadratic function

\[
\begin{align*}
 f(P) &= P^T A^T A P - 2\widehat{P}^T A P + \widehat{P}^T \widehat{P} + 2\lambda_1 P^T D_2^T D_2 P + 2\lambda_2 P^T D_3^T D_3 P \\
 &= P^T [A^T A + 2\lambda_1 D_2^T D_2 + 2\lambda_2 D_3^T D_3] P - 2\widehat{P}^T A P + \widehat{P}^T \widehat{P}
\end{align*}
\]  

In order to get a unique solution that minimizes the objective function we need to ensure that the problem is strongly convex. For that matter,

\[M = A^T A + 2\lambda_1 D_2^T D_2 + 2\lambda_2 D_3^T D_3\]  

(3.7)
must be positive definite. In appendix B, we study the definiteness of the matrix $M$, where we conclude that the matrix is definite positive when $A$ has, at least, one diagonal entry equal to 1. Mathematically, the optimization problem has a unique solution as long as there is one measurement on the raw trajectory, $\hat{P}$. Besides, looking at the dimensions of $D_2$ and $D_3$ being $N-4 \times N$ and $N-4 \times N$ respectively, we conclude that an additional condition stands on the problem, where $N \geq 5$ in order to be able to build matrix $D_3$.

The reconstructed trajectory, $P$, minimizes $f$ if and only if

$$\nabla f(P) = 2A^TAP - 2A^T\hat{P} + 2\lambda_1 D_2^T D_2 P + 2\lambda_2 D_3^T D_3 P = 0$$

(3.8)

i.e., if and only if $P$ satisfies the so-called normal equation

$$2A^TAP + 2\lambda_1 D_2^T D_2 P + 2\lambda_2 D_3^T D_3 P = 2A^T\hat{P}$$

(3.9)

which always have a solution

$$P = [A^T A + \lambda_1 D_2^T D_2 + \lambda_2 D_3^T D_3]^{-1} A^T \hat{P}.$$

(3.10)

By computing equation (3.10) for each raw trajectory $\hat{P}$, we obtain the reconstructed trajectories $P$ that will be used in the learning process.

### 3.2.2 Regularization parameters estimation

The parameters that control the optimization problem, $\lambda_1$ and $\lambda_2$, need to be chosen from the data assuming they are larger than zero. If we chose $\lambda_1 = \lambda_2 = 0$, the smoothing terms would be null, and therefore, the trajectories will not be smoothed. As said before, we can see the parameters as attitude controllers of the aircraft. If $\lambda_1$ and $\lambda_2$ are low, the maneuvers are very rapid and miss the purpose of the inclusion of these terms. As a contrast, if $\lambda_1$ and $\lambda_2$ have large values, they will limit the attitude of the aircraft and likely the turn angle and descent rate of the aircraft can be very low and once again fail to simulate real flight attitude.

This can be understood by looking at figure 3.8 as the reconstructed trajectory in blue has a lower root mean square error (RMSE) than the trajectory in green. Nevertheless, if we analyze the trajectory from an attitude point of view, the trajectory that mimics better real flight attitude is the green. The red trajectory shows the case where the parameters are set to a value that is too high. The trajectory is over-smoothed and fails to follow with precision the initial trajectory in black.

The trajectories in the dataset from TAP Air Portugal have a very high sampling rate, so the trajectories represent very well the actual flight path of the aircraft and can be used as ground truth. We can compare our reconstructed trajectories coming from the optimization problem with the ones coming from WDAR and calculate the RMSE between these trajectories. In figure 3.9 we plot a raw trajectory from the dataset from NEST (black line) with the corresponding flight available in the dataset of TAP (black scatter) and the reconstructed trajectory with different values of $\lambda_1$ and $\lambda_2$.

Similarly to figure 3.8 we can see in figure 3.9 the influence of the parameters on the reconstruction of our trajectories from NEST. If we compare the red trajectories with the black scatter, we can see that some trajectories
Figure 3.8: Influence of the regularization parameters on the reconstructed trajectories. When $\lambda_1$ and $\lambda_2$ increase, turns are smoother but trajectories present a larger RMSE.

seem to reproduce with accuracy the trajectory recorded by TAP. As suggested earlier, in table 3.1 we have calculated the average RMSE between reconstructed ten trajectories for various values of $\lambda_i$ based on the dataset from NEST and the correspondent trajectory found in the dataset of TAP.

Table 3.1: Root mean square error (RMSE), in meters, between the reconstructed trajectories and the trajectories in the dataset of TAP Air Portugal varying parameters. The lowest value of RMSE over the 10 studied flights is attained for $\lambda_1 = 70$ and $\lambda_2 = 10$.

<table>
<thead>
<tr>
<th>$\lambda_1$</th>
<th>1</th>
<th>10</th>
<th>50</th>
<th>70</th>
<th>100</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>604</td>
<td>559</td>
<td>418</td>
<td>383</td>
<td>354</td>
<td>416</td>
</tr>
<tr>
<td>10</td>
<td>400</td>
<td>390</td>
<td>350</td>
<td>340</td>
<td>352</td>
<td>429</td>
</tr>
<tr>
<td>50</td>
<td>399</td>
<td>406</td>
<td>433</td>
<td>445</td>
<td>458</td>
<td>494</td>
</tr>
<tr>
<td>70</td>
<td>457</td>
<td>461</td>
<td>477</td>
<td>484</td>
<td>494</td>
<td>521</td>
</tr>
<tr>
<td>100</td>
<td>517</td>
<td>519</td>
<td>530</td>
<td>535</td>
<td>542</td>
<td>564</td>
</tr>
<tr>
<td>200</td>
<td>677</td>
<td>678</td>
<td>683</td>
<td>685</td>
<td>689</td>
<td>701</td>
</tr>
</tbody>
</table>

Looking at table 3.1 the lowest average RMSE is found for $\lambda_1 = 70$ and $\lambda_2 = 10$. This means that these values for the parameters are the ones that minimize the deviation between the reconstructed trajectories from NEST and the ones recorded by TAP. The value of 340 meters for the RMSE is satisfactory when taking into account that it only takes less than 5 seconds for an aircraft in the final approach to fly 340 meters. Using $\lambda_1 = 70$ and $\lambda_2 = 10$ on NEST trajectories in figure 3.10 we can see that we have smooth trajectories that replicate the expected flight dynamics shown in the dataset of TAP.

Finally, a cross-validation of the values of $\lambda_1$ and $\lambda_2$ on other trajectories available in the dataset from NEST was made to check that the trajectories are well reconstructed and are faithful to the raw trajectories. In figure 3.11 are successfully plotted two random trajectories that follow the baseline trajectory and seem to mimic very well
Figure 3.9: Reconstructed trajectories (in red) with the different values for the parameters. The black line is the plot coming from the raw trajectory of EUROCONTROL, the black scatter is the trajectory recorded by TAP Air Portugal.

For the initial tests of the model, $T_{ldg}$ was set to 2100 seconds (35 minutes), and $q$ to 10 seconds. The value for $T_{ldg}$ was set as it is the minimum useful time-frame for TAP Air Portugal. Below that time, the estimations are too short-termed and do not help them in everyday operations. Regarding the sampling rate, a rule of thumb is that at final approach speed, an aircraft travels around 1.5 times its length per second. That way, using $q = 10$ seconds we have an average displacement of 15 aircraft lengths or around 650 meters.
Figure 3.10: Reconstructed trajectories, in red, for $\lambda_1 = 70$ and $\lambda_2 = 10$. The black line is the plot coming from the raw trajectory from EUROCONTROL, the black scatter is the trajectory recorded by TAP Air Portugal. For the three flights, we can see that the turns of the reconstructed trajectory are smooth and seem to replicate plausible flight paths.

3.3 Feature engineering

In addition to the trajectories, we aim to include in this work the influence of traffic and weather in air operations. Our goal is to complement the landing time inference with a delay prediction on the final phase of a flight.
3.3.1 Aircraft density models for delay prediction

To reflect the influence of traffic in the trajectories of aircraft, we can include a density metric that will help our generative model to have more accurate previsions learning how aircraft behave in crowded terminal areas.

Historical delay inside Portuguese airspace

From takeoff to landing, a large number of factors can influence and impact on the arrival time of an aircraft. The final landing time of an aircraft can suffer changes until the moment when landing gear touches down on the runway. On take-off, air traffic control can restrict the speed of aircraft and give vectors to ensure sufficient separation between traffic that are not encountered in a flight plan. Additionally, during cruise, wind can have a knock-on effect on landing time as tailwind increases the ground speed of an aircraft, and headwind decreases the speed and therefore increases flight time. Following the same path, unexpected weather, as large cumulonimbus or heavy turbulence, could oblige the pilots to ask ATC for a deviation in order to avoid this kind of adverse meteorology. In contrary, pilots can ask for a more direct routing that will save flight time in relation to the flight time submitted in the flight plan. On a final phase of a flight, ATC can ask a crew to reduce speed on descend, put aircraft on hold or even give radar vectors to take a larger approach to allow enough separation between incoming traffic. This kind of last-minute delays are the delay we want to predict and include in our model.

The original dataset coming from EUROCONTROL presented in Section 3.1 contains the actual flight path that aircraft flew. Nevertheless, as the planned route on flight plans often differ from the actual route that pilots fly, it is essential to eliminate the en-route delays that may have been encountered by a flight up until the entrance in the Portuguese airspace. For that matter, we need to compare, for each flight, the actual flight times inside LPPC

Figure 3.11: Two reconstructed trajectories landing in different runways. The optimization problem is able to capture the holding pattern of the aircraft landing from the north.
with the ones scheduled on the flight plans. Another dataset coming from EUROCONTROL DDR2 was used, containing the information about the original flight plans submitted that generated the operated routes of the flights in the initial dataset.

In figure 3.12, there is plotted the difference, in seconds, between the actual flight time inside LPPC and the scheduled flight time inside LPPC submitted on the flight plan. This variable was plotted against the average number of aircraft inside the Portuguese airspace during the overfly time of the flights. In this plot, only flights landing in Lisbon airport were considered as we supposed that cruising traffic inside the Portuguese airspace does not interfere with the flight that approaches Lisbon airport. Nevertheless, this affirmation will be confirmed further on.

![Figure 3.12: Plot of the flight delay inside LPPC versus the number of aircraft inside the airspace. Despite the value of $R^2 = 2.37\%$, the linear fitting with a positive slope shows that there is a slight link between the delay and the density of aircraft.](image)

The plot reveals a wide scatter of points of over 7000 flights. Delay over the Portuguese airspace ranges from -1000 seconds to over 2000 seconds. A negative value of the delay means that a flight was quicker in LPPC compared to the flight time stated in the flight plan. In contrary, a positive value of the delay means that the flight took a longer time inside Portuguese airspace than initially estimated in the flight plan. To show that the delay is correlated with airspace occupancy, we adjusted a linear fitting to the plot. was adapted in order to show a relation between delay and occupancy of the airspace. This fitting revealed that there is, in fact, a light trend between the variables, but it is not very emphasized. The linear regression estimates that the delay (in seconds) is equal to around 13.4 times the number of aircraft inside LPPC plus an offset of 71 seconds. For example, if ten aircraft are expected to land in Lisbon, we can expect an average delay of 204 seconds, or about 2 minutes and 24 seconds. Unfortunately, this regression does not fit the data correctly, as shown by the coefficient of determination $R^2$ of 2.37%.

**Historical delay inside the TMA of Lisbon airport**

Another line of investigation is to look at the number of aircraft that are only inside Lisbon the TMA of Lisbon
airport. This is done in figure 3.13, where only the occupancy of the much smaller TMA of Lisbon airport was considered. It is particularly interesting to study the congestion of this area as it is in the immediate vicinity of the airport.

![Figure 3.13: Plot of the flight delay inside LPPC versus the number of aircraft inside the TMA of Lisbon airport. A more evident correlation between the delay and the density is visible, as the value of the coefficient of determination is equal to 12.74%.

In this plot, it is more evident that there is a relation between the number of flights in the TMA and the delay inside the airspace. The new linear fitting has now a value for $R^2$ of 12.7% and it estimates that the delay is equal to 86 times the number of aircraft in the TMA minus 71 seconds. Meaning that if 10 aircraft are in the vicinity of the airport, an incoming flight will have almost 10 minutes of delay. We can believe that this method of counting aircraft to predict delays will have better results as the previous method as it has a higher coefficient of determination.

**Delay model refinement**

The last line of work regarding the traffic that could delay aircraft landing in Lisbon consists of considering the number of aircraft that are closer to the airport than the one of interest. It consists in counting the number of aircraft contained in a circle centered in Lisbon airport that passes through the aircraft of interest. By doing so, we suppose that the aircraft inside that circle will land before the aircraft on the edge of the circle. In addition, we include a variable, $\theta$ that measures the difference in heading between the aircraft of interest and other aircraft. Based on a maximum value for $\theta$, we can control whether or not an aircraft is counted inside the circle. In figure 3.14, we can see the circle delimited by an aircraft of interest.

In figure 3.15 we can see that our plot is similar to figure 3.13 but the linear regression shows different values for the slope and intercept. It seems to be a more accurate representation as the value of $R^2$ is the highest, at 16.6%. In addition, we varied the angle $\theta$ from 0 to 360 to see if the direction of the incoming traffic has an impact on whether or not a flight is delayed due to another. Simulations showed that the highest value for $R^2$ is when
Figure 3.14: Visual representation of the circle centered in Lisbon airport passing by the aircraft of interest. This circle is dynamic and permits only to capture the aircraft that are closer to Lisbon airport.

$\theta = 360^\circ$ meaning that the azimuth position of an aircraft inside the circle is not an important aspect to take in mind for the delay model.

To include this model in the learning and generative models, we associate each point of our training trajectories with an additional variable that is the number (density) of aircraft in the circle at that timestamp. Each trajectory is now $P \in \mathbb{R}^{N \times 4}$ and each spatial point is $p_i = (lat_i, lon_i, alt_i, dens_i)$. Therefore, the clustering algorithm can generate new clusters, with different archetypal trajectories, while considering the occupancy of the airspace in addition to the geographical position of the aircraft. The inclusion of this feature of density will allow us to evidence if aircraft take different trajectories as a result of the airspace occupancy.

**Dataset variation**

In subsection 3.3.1, we considered that only traffic landing in Lisbon airport could have an impact on the trajectories of the flights incoming for landing. Nevertheless, to confirm this statement it is easy to introduce a dataset containing all flights overflying Portuguese airspace and filtering them by altitude.

This dataset was tested to evaluate the density of aircraft inside LPPC, the TMA and the circle centered in Lisbon airport. For all tests, the density measure reflects the number of aircraft that are inside a certain airspace and
Figure 3.15: Plot of the flight delay inside LPPC versus the number of aircraft within a circle centered in Lisbon airport. It is the plot that shows the most noticeable relation between the two variables. For this case, $R^2 = 16.6\%$. Nevertheless, a large amplitude of delay is present for every density of aircraft.

below the aircraft of interest. In table 3.2 are compared the values for $R^2$ for the linear fitting of each measurement of density.

<table>
<thead>
<tr>
<th>Density model, aircraft inside:</th>
<th>Value of $R^2$ (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aircraft landing in LPPT</td>
<td>All aircraft</td>
</tr>
<tr>
<td>LPPC</td>
<td>2.37</td>
</tr>
<tr>
<td>TMA</td>
<td>12.74</td>
</tr>
<tr>
<td>Circle</td>
<td>16.6</td>
</tr>
</tbody>
</table>

Table 3.2: Comparison of the values of the coefficient of determination of the different density measures using the two distinct data sets. For all metrics, the value of $R^2$ is lower for the dataset considering all aircraft.

As expected, the values of $R^2$ decrease when switching to the dataset containing all aircraft. This is explained by the fact that for the number of aircraft inside LPPC and the circle, the density measures capture aircraft that are cruising and traffic outgoing and incoming all airports in the mainland of Portugal that do not interfere with the operations of Lisbon airport. This leads to a decrease in the accuracy of a density model that tries to link the number of aircraft to the observed delay. The influence of the different density metrics will be studied in the further on, alongside with the inclusion of key features of meteorology that may affect the trajectory of airplanes.

Note on the linear fittings

It is important to underline that the usage of the linear fitting was introduced as a first approach to show that the delay aircraft have on the last phase of a flight can be related with airspace occupancy. It is, as shown by the low values of $R^2$, by no means the best fitting as the residuals of the regression can be very high. The study of the residuals can help to better evaluate the correlation between the variables. In addition, the large number of observations (7000) can contribute to the not so precise linear fitting.
3.3.2 Weather information

Adding to the airspace capacity issues, weather can have a significant impact on airport operations. In 2017, weather accounted for 32.7% of delays around Europe, and therefore it is the second cause of flight delays in Europe. At an airport level, the wind direction has a direct impact on the runway in use since aircraft should take off and land with headwind in order to reduce takeoff roll and landing distance. Plus, reduced visibility can drastically reduce airport capacity if Low Visibility Operations (LVO) are active, when Runway Visual Range (RVR) is lower than 550m or visibility is under 800 meters. In those cases, aircraft on the ground and in the air need to have increased separation between them and causing the reduction of the capacity of an airport to handle traffic and delays are certain to occur.

Airports and operators have access to aeronautical weather under different formats and are separated into two major categories. Upper air observations provide all the information needed for the pilots to be aware of the weather conditions throughout their flight, while surface weather observations grant information on the weather reported at aerodromes, such as the Meteorological Terminal Air Report (METAR). These standard reports are issued every 30 minutes and provide actual weather conditions recorded at specific aerodromes at specified times. Here is an example of a METAR message and how to read it:

LPPT 221130Z 31005KT 5000 -RA OVC040 21/14 Q1018

- Location: LPPT - ICAO identification for Lisbon airport.
- Day and time: 221130Z - 22nd (of September -the month is not stated in the report) at 11:30 Zulu time zone.
- Wind direction and intensity: 31005KT - wind direction is 310 degrees and its intensity is 5 Knots.
- Visibility: 5000 - Visibility is 5000 meters
- Meteorological phenomena: -RA - slight rain showers.
- Cloud coverage: OVC040 - Overcast clouds at 4000 feet.
- Temperature and dew point: 21/14 - Static air temperature is 21 degrees Celsius and the dew point is 14 degrees Celsius.
- Atmospheric pressure: 1018hPa.

These are the main information contained in a standard METAR report. Additional information can be found if, for example, wind direction is variable or if there is fog in the vicinity of the airport. All weather variables have an impact on a flight and must be analyzed by the crew in order to take important decisions, such as diversions if severe weather is encountered. For everyday operations, the wind observed in METAR imposes the runway in use and gives valuable information for pilots in order to prepare their approach to land in an airport. Therefore, we estimate that including the wind direction observed in Lisbon airport, as well as its intensity will help the generative model to choose a cluster that will direct the aircraft to the appropriate runway based on the wind conditions.

\(^{1}\text{CAPA - Center for Aviation, https://centreforaviation.com/analysis/reports/european-airspace-control-the-promise-delays-the-need-action-424075, accessed on 09-08-2019}\)
To have access to historical METARs, a database of historical weather information from the Iowa State University \footnote{https://mesonet.agron.iastate.edu/request/download.phtml, accessed on 07/04/2019} was downloaded. The user can choose, for a specific period of time and a specific airport, the kind of weather information they want to download. In our case, we downloaded METAR messages, from June 2018 to June 2019, to match the timeframe of the dataset of EUROCONTROL.

We will test the wind feature by introducing it in our learning dataset using two different approaches. The first one is by, similarly to what has been done for the airspace occupancy, adding two columns to the trajectories in the training set. The first column is wind direction and the second is intensity so the trajectories are $P \in \mathbb{R}^{N \times 6}$ and each spacial point $p_i = (lat_i, lon_i, alt_i, dens_i, dir_i, intens_i)$.

The second approach is by adding just one column to the trajectories with a binary variable corresponding to the runway in use. This information can be deducted from the METAR: if the wind direction is between 117 and 296 degrees, it is runway 21 in use. In opposite, if the wind direction is from 297 and 116 degrees, it is runway 03 in use. Using this way of including the wind feature to the dataset, we will have a trajectory $P \in \mathbb{R}^{N \times 5}$ and $p_i = (lat_i, lon_i, alt_i, dens_i, rwy_i)$. 
Chapter 4

Learning and generative models

In Section 2.4 we presented the various families of clustering algorithms. We determined that for our case, the learning process would be done using a Centroid-based algorithm as they deal well with large datasets and are computationally efficient. In addition, the main advantage of this kind of algorithm is, as said before, the creation of a centroid for each cluster. In our case, this centroid is the average of each trajectory inside the cluster.

4.1 Data Normalization

Data normalization is defined as the process of rescaling original data without changing its behavior or nature. It is useful in classification algorithms involving neural network or a distance-based algorithm such as K-means. The goal of the normalization is to change the values of numeric variables in the dataset to a common scale, without distorting differences in the ranges of values. It is required only when features have different ranges [23].

In our case, latitudes range from -90 to 90, and longitudes range from -180 to 180. In addition, we can consider that altitude ranges from 0 to 40 000 feet. As the amplitude in altitude is much larger than the other features, the clustering will be governed by the altitude. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance.

Different types of normalization exist to achieve the same goal. For example, Min-Max normalization performs linear transformation on original data using the boundary values of the dataset to normalize the data as it follows:

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}.$$  (4.1)

Another type of normalization is the Z-score normalization, also known as standardization. It is a widely used technique rescaling the features so they have the properties of a normal Gaussian distribution. The standard score, also known as a z-score, is calculated for each point of a dataset as follows:

$$z_i = \frac{x_i - \bar{x}}{s}.$$  (4.2)

where $\bar{x}$ is the average of the values of a feature, and $s$ is the standard deviation of the sample values of the feature. For the purpose of this thesis, we will use the standardization technique to normalize all the features used
in the clustering process as it is less sensitive to possible outliers in the dataset [23].

4.2 K-means parameter estimation

The trajectories to be clustered are the reconstructed trajectories coming from the optimization problem in 3.2 and are ready to be clustered by the clustering algorithm K-means. The only parameter required for this process is the number of clusters $K$.

Determining the optimal number of clusters in a data set is a fundamental step in order to guarantee optimal results from the algorithm. Unfortunately, there is no definitive answer to this question, but there are some metrics used to indicate an appropriate number of clusters for our dataset. The most popular method is the Elbow method in which is calculated the sum of the squared distance of each trajectory to the respective cluster centers (Within cluster sum of squares - WSS). Then, it is plotted against the number of clusters. The total WSS measures the compactness of the clustering and the K-means algorithm returns the cluster centers that for a given $k$ minimize the WSS. One should choose a number of clusters so that adding another cluster does not improve much the total WSS. In figure 4.1 this procedure is done for the dataset of EUROCONTROL varying the number of clusters from 3 to 70.

![Elbow plot for the dataset of the model. The elbow is located around K=13. From this point, WSS does not vary significantly if a cluster is added.](image)

Figure 4.1: Elbow plot for the dataset of the model. The elbow is located around $K=13$. From this point, WSS does not vary significantly if a cluster is added.

In this plot, we can see the shape of the so-called elbow. For around $K = 13$ clusters, the total WSS does not improve much with the addition of new clusters, meaning that the optimal value for the number of clusters lie around that number. Nevertheless, as for AIRAC cycle 1910, the number of STARs in Lisbon airport is 28, meaning that aircraft can come from roughly 28 directions towards Lisbon airport. For that being, we decided to
choose an intermediate number of clusters of $K = 20$.

Figure 4.2: Cluster centers for $k = 20$ clusters. In (a) we can see that the algorithm is able to separate the flights landing in both runways and capture the directions where aircraft come from. In (b) random trajectories inside the cluster with the highest frequency are plotted. They all follow the same general path.

In subfigure 4.2 (a) we can see the output of the clustering process. We have 20 archetypal trajectories that are the centroids of the 20 clusters. Coming from all directions, the centroids seem to have captured the different trajectories that aircraft can take towards landing in Lisbon airport.

Additionally, K-means generated clusters for both for runways 03 (from the south) and 21 (from the north), showing the landing configurations available for the aircraft. This result is satisfactory as it reflects well the principal routes that lead to the landing in both runways. As said before, a cluster groups trajectories that are similar between them. This can be seen in figure 4.2 (b) where there are plotted 25 of the trajectories that were grouped together in the cluster with the highest frequency. As all the trajectories clustered have a common length of $T_{ldg}$ seconds, the duration of the archetypal trajectories is naturally $T_{ldg}$. Having satisfactory clusters and centroids is a crucial element in our model as the generative model will be based on the archetypal trajectories and on the intra-cluster covariance matrices.

4.3 Intra-cluster covariance matrix approximation

Each cluster has a covariance matrix associated that measures the dispersion of the clustered trajectories around the centroid and is defined as

$$\Sigma = \mathbf{E}[(X - \mathbf{E}[X])(X - \mathbf{E}[X])^T]$$

where $X$ is a vector containing all trajectories in a cluster and $\mathbf{E}[X]$ is the centroid of the cluster. The end result is $K$ symmetric positive definite matrices $\Sigma \in \mathbb{R}^{F \times F \times N}$, where $F$ is the number of features in the trajectories. To eliminate the influence of outlier trajectories and to reduce noise, following [4], a Singular Value Decomposition (SVD) is made on the intra-cluster covariance matrices. The goal of this operation is to approximate matrix $\Sigma$ by
\[ \Sigma = U \Sigma U^T. \] In this case, \( \Sigma \) is a diagonal matrix with the highest \( r \) singular values and \( U \) is a column matrix with the singular vectors associated with the singular values. In this approximation, we only select the trajectories that are the most similar to the mean. By doing that, we ensure that the generative model generates plausible trajectories that are operated with high frequency. Additionally, this procedure reduces the overall size of the trained model and speeds significantly the generative model that is limited to a fewer number of operations. Due to the high number of clusters, 20, we established that \( r = 5 \) would capture the majority of the significant trajectories inside a cluster. As no systematic procedure exists to estimate \( r \), this parameter should be chosen by studying its influence on the results. If the results are not satisfactory for \( r = 5 \), the parameter can be tuned depending on the data set used.

### 4.4 Generative model

As introduced in section 2.5, the generative model present in this work is based on a Gaussian Mixture Model. It is formulated by 2.5 with \( k = 20 \). At this point, after the clustering process was done, we have learned the vector \( \mu_i \) and the covariance matrix \( \Sigma_i \) for each cluster, meaning that we have all variables needed to construct the generative model giving us the posterior trajectory conditioned on the current position of an aircraft.

In the first place, we need to assign the current position of an aircraft to a cluster so the model knows the parameters it needs to use for the normal distribution. To be precise, the current position of an aircraft will be given by a vector \( P^n = (p^n_1, p^n_2, \ldots, p^n_n) \) corresponding to the last \( n \) measurements of a trajectory prior the generation of a posterior trajectory. The joint probability density of the position and a cluster is

\[
p(P^n, z_{kj}) = p(z_{kj})p(P^n|z_{kj}) \tag{4.4}
\]

where \( z_{kj} \) represents the position \( j \) in cluster \( k \). Breaking down the terms \( p(z_{kj}) \) is the marginal distribution of the cluster, \( k \), and position, \( j \), and \( p(x|z_{ij}) \) is the conditional distribution of \( P^n \) given the cluster and position, \( z_{kj} \). The calculation of the first term is straightforward as it can be interpreted as the fraction of trajectories in each cluster:

\[
p(z_k) = \pi_k = \frac{\text{trajectories in cluster } z_k}{\text{trajectories in the model}}. \tag{4.5}
\]

It is important to note that \( \pi_i \) is constant for all position \( j \) in cluster \( i \). Regarding the second term, \( p(x|z_{ij}) \), it follows a multivariate Gaussian distribution over cluster \( z_i \), meaning that

\[
p(P^n|z_{kj}) \sim N(P^n|\mu_{kj}, \Sigma_{kj}) \tag{4.6}
\]

and its joint probability density is

\[
p_{kj} = (2\pi)^{-D/2}|\Sigma_{kj}|^{-1/2}\exp\left[ -\frac{1}{2}(P^n - \mu_{kj})^T\Sigma_{kj}^{-1}(P^n - \mu_{kj}) \right] \tag{4.7}
\]

where \( D = n \) is the number of elements in \( \mu_{kj} \), which is a vector containing \( D \) points of the centroid \( \mu_k \) of cluster \( k \). The matrix \( \Sigma_{kj} \) is the covariance matrix of size \( \mathbb{R}^{D \times D} \) relative to the portion of the centroid represented by \( \mu_{kj} \).
Nevertheless, we need to have in mind that the original covariance matrices have been approximated using the SVD. Using this low-rank approximation of $\Sigma_{kj}$, the approximated covariance matrices are now positive semi-definite and no longer positive definite, which is a requirement to have a valid multivariate density. Fortunately, we can overcome this issue using the Moore-Penrose inverse (pseudoinverse) and the pseudodeterminant of $\Sigma_{kj}$. This operation is fairly simple as SVD was already performed on these matrices.

The calculation of $p(P^n|z_{kj})$ needs to be made for all clusters and every sequence of $n$ consecutive points of their cluster center. The pair cluster/position is chosen by the highest value of the joint probability density. At this point, the cluster in which the posterior trajectory is chosen and can, therefore, be generated.

Let $P^n$ represent the position measurements held to the model and $P^{post}$ be the posterior trajectory that needs to be generated by the model. We suppose that vectors $P^n$ and $P^{post}$ are jointly Gaussian random vectors as follows [24]:

$$
\begin{bmatrix}
P^n \\
P^{post}
\end{bmatrix}
\sim N
\begin{pmatrix}
\mu_n \\
\mu_{post}
\end{pmatrix},
\begin{bmatrix}
A & C \\
C^T & B
\end{bmatrix}
$$

The marginal distribution of $P^n$ is

$$
P^n \sim N(\mu_n, A).
$$

By using a permutation matrix,

$$
P = \begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}
$$

we can determine the marginal distribution of $P^{post}$ and the conditional distribution of $P^{post}$ given $P^n$.

Posing $\bar{x} = Px$, we have

$$
P \begin{bmatrix}
P^n \\
P^{post}
\end{bmatrix} = \begin{bmatrix}
P^{post} \\
P^n
\end{bmatrix}
$$

and

$$
P \begin{bmatrix}
\mu_n \\
\mu_{post}
\end{bmatrix} = \begin{bmatrix}
\mu_n \\
\mu_{post}
\end{bmatrix}
$$

In equation (4.4), substituting $X$ by $\bar{X}$ we get

$$
\bar{\Sigma} = E[(PX - E(PX))(PX - E(PX))^T] = PE[(X - E(X))(X - E(X))^T]P^T = P\Sigma P^T
$$

Applying this transformation to the covariance matrix in (4.8) we get:

$$
P \begin{bmatrix}
A & C \\
C^T & B
\end{bmatrix} P^T = \begin{bmatrix}
B & C^T \\
C & A
\end{bmatrix}
$$
From here, the marginal distribution of $p^{post}$ is

$$p^{post} \sim \mathcal{N}(\mu_{post}, B)$$  \hspace{1cm} (4.15)$$

and the conditional distribution of $y$ given $x$ is

$$p^{post} | p^n \sim \mathcal{N}(\mu_{post} + C^T A^{-1} (P^n - \mu_n), B - C^T A^{-1} C)$$ \hspace{1cm} (4.16)$$

Our posterior trajectory will be the mean vector of equation 4.16 computed by

$$\mu_{post} + C^T A^{-1} (P^n - \mu_n)$$ \hspace{1cm} (4.17)$$

where $\mu_{post}$ is the posterior centroid of the cluster chosen from position $j$ and $C^T A^{-1} (P^n - \mu_n)$ is a weighted deviation of the held out measurements $P^n$ with the most probable mean.

All of the parameters needed were learned in the clustering process. Given the held out measurements, the generative process is automatic and does not require any additional information.

To see if the generative model works alone, we can draw trajectories from each cluster by sampling a vector $z \sim \mathcal{N}(0, I)$ and emit a trajectory $\mu_k + \tilde{U}_k \tilde{\Sigma}_k^{1/2} z$. This trajectory sampling is interpreted by being the mean of cluster $i$ plus a random normal deviation present on the approximated covariance matrix of cluster $i$.

Figure 4.3: Random trajectories generated from a cluster landing in runway 03. The model is able to produce similar flight paths for all trajectories.

We plotted in figure 4.3 25 random trajectories from the cluster containing the highest number of trajectories. It is possible to see that all trajectory follow the same general path and are noise free, meaning that our generative model has enough data to produce coherent flight trajectories.
4.4.1 Initial landing time inference

Apart from the posterior trajectory, we are able to get the corresponding most probable position, \( j \), inside the chosen cluster. As all the trajectories used in the learning process have the same length in seconds, \( T_{ldg} \), the cluster centers, representing archetypal trajectories, are therefore \( T_{ldg} \) seconds long. As the time elapsed between two points of a trajectory is \( q = 10 \) seconds, we can compute, from the position \( j \) an initial prediction of the landing time of the aircraft, based only on the current position and the most probable position in the cluster center. The remaining flight time to landing is computed by

\[
t = (N - j) \cdot q \quad .
\]  

(4.18)

The result coming from this operation is given in seconds. To predict the landing time, we just add the remaining flight time to the current time of the day.

Unfortunately, this prediction is made based on the trajectories that are contained in a determined cluster. It is possible that not all trajectories inside a specific cluster experienced delay due to airspace congestion. For that reason, in order to study the influence of aircraft sharing the same airspace on both the trajectory and landing time inference, we need to introduce this feature in the clustering phase.

4.5 Implementation

The implementation of the models described was performed in Python 3 language using Anaconda distribution and PyCharm Integrated Development Environment. All software and packages used are open source and available for Windows, macOS and Linux. The computer used for the implementation of the model presented is a Windows-based laptop, running on Windows 10 Home with an Intel Core i7 processor and 16GB of RAM. In appendix A, a general flowchart of the different files of the algorithm implemented is presented.
Chapter 5

Generative model results

Up to this point, we have derived the models needed in order to predict aircraft trajectories and infer on their landing time in crowded terminal areas. The model is general enough to be used with any trajectory dataset in any airport around the globe, but the parameters that control the trajectory reconstruction, the clustering process, and the generative model need to be chosen accordingly for every situation.

In this Chapter, the results presented come from the implementation of the model using the trajectories of the flights landing in Lisbon available in the dataset of EUROCONTROL. From the 13 AIRACs that were downloaded, 11 of them were selected randomly and 93535 flights were considered in order to learn the parameters of the model. Once the trajectories are reconstructed and clustered, we can run the generative model and add the different aircraft density models and the meteorology indicator in order to study the influence of these features in the final results.

The results presented are the outputs of our model, posterior trajectory, and remaining time to landing based on initial measurements generated from Monte Carlo tests of flights operated within AIRAC 1901.

5.1 Baseline of TAP Air Portugal

To evaluate the performance of our model, TAP Air Portugal provided us an Excel spreadsheet with information about the historical average flight time estimated by the flight plans and the historical average observed flight time over a period of one year. This information can be used by the airline to predict the landing time of their aircraft in Lisbon airport and therefore can be used to be a baseline comparison of our model.

The average flight times in the spreadsheet are only accessed for the operation of TAP Air Portugal, meaning that flights operated by other airlines are not considered in their data. To make a fair comparison between our model and this baseline, it is necessary to filter the flights of TAP Air Portugal in AIRAC 1901 and only use the flights in the dataset operated by the airline to evaluate our model.

5.2 Test dataset preparation

The trajectories in the AIRAC cycle 1901 need to be reconstructed before being given to the generative model. This has to be done to generate the ground truth trajectories with whom the predicted trajectories will be compared.
The ground truth trajectories are the reconstructed trajectory using the same hyperparameters used to reconstruct the trajectories of the learning dataset. In addition, the trajectories \( P^* \) held out to the generative model come from the reconstructed trajectories of the test dataset.

5.3 Results

We initially tested our model for trajectories with \( T_{ldg} = 35 \) minutes, or 2100 seconds to landing as it is a minimum requirement set by TAP Air Portugal in order to provide useful time inference for them to use. For this first simulation, the trajectories in the model were only composed by the spatial position of the aircraft at each timestamp, meaning that our model does not look at the aircraft density or wind component to cluster the trajectories during the learning and generative phases. Running this base model will allow us to understand if our model is able to generate plausible trajectories before increasing \( T_{ldg} \) and introducing the additional features in the model.

5.3.1 RMSD and MAE calculation for the generated trajectories and landing time inference

The quantitative comparison between the generated trajectories from our model and the real trajectories operated is established using the root mean square deviation (RMSD) and the mean absolute error (MAE). Following the formulation in [25], calculation of the root mean square deviation of the predicted path from the real one is

\[
RMSD_{path} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (p^\text{post}_i - p^\text{proj}_i)^2}
\]  

where \( p^\text{post}_i \) is the predicted position by our model, and \( p^\text{proj}_i \) is the orthogonal projected position of the aircraft on the real trajectory. This measure calculates the lateral deviation of the posterior trajectory from the real trajectory.

Similarly, MAE for the landing time inference for \( M \) Monte Carlo simulations is defined as follows:

\[
MAE_{ldg} = \frac{1}{M} \sum_{i=1}^{M} | t^\text{pred}_i - t^\text{real}_i |
\]  

where \( t^\text{pred}_i \) is the predicted landing time by the model and \( t^\text{real}_i \) is the real landing time.

5.3.2 Base model results

After reconstructing the trajectories with \( T_{ldg} = 2100 \) seconds, \( \lambda_1 = 70 \) and \( \lambda_2 = 10 \), as established in subsection 3.2.2, the clustering algorithm was run with \( K = 20 \) clusters as chosen in section 4.2. The results of the clustering are the cluster centers that were shown in figure 4.2 (a). We have already established that the results from the clustering process were satisfactory as archetypal trajectories were generated for both runways. It is now time to evaluate the generative model using multiple Monte Carlo simulations.
The held out trajectories, $P^n$, that will allow the generative model to predict the posterior trajectory consist of $n = 10$ consecutive measurements randomly chosen in a time window of 5 minutes between 35 and 30 minutes to landing. The generative model has now to predict the posterior trajectory and infer on the landing time of the aircraft.

In figure 5.1 we can see an example of the output of the generative model regarding the posterior trajectory prediction. In black, there are plotted the archetypal trajectories that are compared with $P^n$, plotted in yellow. The generative model calculates the probability of $P^n$ belonging to each position in each cluster, and the archetypal trajectory with the higher probability is shown in red where the most probable position on that cluster is highlighted in white. Then, we can see in dark blue the posterior trajectory generated by our model and in orange the real trajectory taken by the airplane. The results obtained are encouraging as the predicted trajectory is smooth and follows a similar path when compared with the ground truth trajectory. For a total of 1000 Monte Carlo simulations, the average root mean square deviation for each predicted path is equal to 6790 meters.

![Figure 5.1: Illustrative example of the posterior trajectory in blue, compared with the real flown trajectory in brown. In black, we can see the archetypal trajectories that were compared with the held out points to the model in yellow. The probabilistic model compares the held out points with the cluster centers and chooses the most probable equivalent points on the cluster center plotted in white.](image)

Regarding time inference, to have an error distribution of the estimation error of our model defined as

$$\epsilon = |t_{\text{Pred}} - t_{\text{real}}|,$$

we have plotted the empirical cumulative density function (CDF) of $\epsilon$. (5.3)

Using the data on the average estimate flight time and average historical flight time coming from TAP Air Portugal, we can compare the estimations of the landing time inference with the benchmarks of the airline by plotting the empirical CDF of each estimator.

In figure 5.2, we can quickly establish that the time inference used in our model outperforms significantly the benchmarks given by TAP Air Portugal. The generative model is able to predict more accurately the arrival time of the aircraft than the average estimate and average historical estimators as the CDF curve in yellow has a steeper
Figure 5.2: Empirical Cumulative Density Function of the estimation error of the implemented model, the average estimate flight time, and the average historical flight time coming from TAP Air Portugal. At the left side of the plot we can see the CDF function in yellow corresponding to the estimations made by our model. It is therefore more accurate than the baseline estimations from TAP Air Portugal.

slope than the ones in grey and orange. Regarding the maximum absolute error of our model, $MAE_{ldg}$, it is half the maximal error coming from the benchmarks from the airline.

![CDF Plot]

<table>
<thead>
<tr>
<th></th>
<th>Generative model</th>
<th>Average estimate</th>
<th>Average historical</th>
</tr>
</thead>
<tbody>
<tr>
<td>$MAE_{ldg}$ (in seconds)</td>
<td>220</td>
<td>650</td>
<td>539</td>
</tr>
</tbody>
</table>

Table 5.1: Mean absolute error of the generative model for $T_{ldg} = 35$ minutes. Compared to the baseline of TAP Air Portugal, $MAE_{ldg}$ of our inference is up to 66% lower.

The mean absolute error for the three models are in line with the CDF curves as compared in table 5.1. Out of all models, the generative algorithm presents the best results with an average error of 220 seconds, or 2 minutes and 40 seconds when compared with the ground truth. The two estimators from the airline do not seem to be a good indicator of the landing time as their average error is around 10 minutes and 50 seconds regarding the average estimate in the flight plan and 8 minutes and 59 seconds for the average historical landing time. The excessive error of the average estimate and the average historical can be explained by various factors that influence the landing time throughout the flight such as the availability of direct routing, deviations from the planned route, or adverse wind conditions inflight. These factors cannot be planned in advance and are unique to each flight.

Before adding the density and the meteorological features to the generative model, we need to study the influence of $T_{ldg}$ on the quality of the posterior trajectories generated and on the landing time inference. Our goal is to maximize $T_{ldg}$ in order to give TAP Air Portugal the earliest precise estimation of the landing time and the trajectory of their aircraft.
5.3.3 Increasing trajectory length

As the model is data-driven, meaning that our algorithm is learned using real data coming from NEST, we are subject to the limitations of the dataset. We saw in figure 3.2 that segments duration increase when the aircraft is far from Lisbon airport. This means that if we increase $T_{ldg}$ the number of measured points at the beginning of our trajectories will be low, forcing our completion model to extrapolate with lower precision our trajectories. This leads to an inaccurate estimation of the position of the aircraft at a given timestamp and, therefore, a wrong landing time inference is done.

From 35 minutes, we increased $T_{ldg}$ by 5 minutes steps and studied, for 1000 Monte Carlo simulations, the trajectories and mean error from our model. In parallel, we increased the timeframe by 10 minutes for every 5 minutes increase in $T_{ldg}$.

<table>
<thead>
<tr>
<th>$T_{ldg}$ (in minutes)</th>
<th>Generative model</th>
<th>Average estimate</th>
<th>Average historical</th>
<th>$RMSD_{path}$ from the real trajectories (in meters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>220</td>
<td>650</td>
<td>539</td>
<td>6790</td>
</tr>
<tr>
<td>40</td>
<td>233</td>
<td>655</td>
<td>577</td>
<td>7189</td>
</tr>
<tr>
<td>45</td>
<td>241</td>
<td>710</td>
<td>599</td>
<td>7429</td>
</tr>
<tr>
<td>50</td>
<td>280</td>
<td>719</td>
<td>624</td>
<td>10131</td>
</tr>
</tbody>
</table>

Table 5.2: Evolution of $MAE_{ldg}$ and $RMSD_{path}$ with increasing values of $T_{ldg}$. Up to $T_{ldg} = 45$ minutes, a steady expected increase in the error metrics is observed. For $T_{ldg} = 50$ minutes, noise is visible in the generated trajectories which explains the sudden increase in $MAE_{ldg}$ and $RMSD_{path}$.

Table 5.2 give us a good indicator of the role played by $T_{ldg}$ in the quality of the estimation of our model. As expected, both $RMSD_{path}$ of the posterior trajectory and the of the landing time inference increase with $T_{ldg}$.

The error evolution is not proportional to the increase in the duration of the trajectories. From 35 to 40 minutes, $RMSD_{path}$ increases 399 meters, whereas from 45 to 50 minutes, this error increases by 1804 meters. The same analysis can be done for the landing time inference error, where an increase of 13 seconds is reflected from 35 to 40 minutes versus an increase of 39 seconds from 45 to 50 minutes. It is when looking at the posterior trajectories we understand this sudden increase in the estimation errors of our model. In figure 5.3 we can see that for $T_{ldg} = 50$ minutes, the posterior trajectory generated by our model presents noise, explaining the increase in the position estimation error.

On the other hand, the generated trajectories for $T_{ldg} = 45$ minutes are free of noise the trajectories are smooth. The appearance of noise in the trajectories is explained by the lack of points available in the model to predict the required ones. This issue can be overcome by adding more AIRAC cycles to the learning algorithm.

From now on, in order to study the influence of the density and meteorological features, we set $T_{ldg}$ to 45 minutes as it gives a good compromise between the accuracy of our base model and the length of the trajectories.

5.3.4 Aircraft density contribution to the base model

As the title of the thesis suggests, we aim to predict the aircraft trajectory and consequently infer the landing time taking into account airspace congestion. For that, we have studied in section 3.3.1 various metrics to evaluate the density of aircraft around the airport. By adding these metric for every timestamp in the trajectories of the learning set, the clustering algorithm will cluster the trajectories not only by their spatial position but also con-
Figure 5.3: Based on the held-out trajectory in yellow, the posterior trajectory generated with noise (in blue), increasing the root mean square path deviation for $T_{ldg} = 50$ minutes.

considering the aircraft density. This way, flight paths having similar trajectories as well as similar densities will be clustered together and will help us to underline the influence of the general density of aircraft around the airport in the flight path and landing time of the aircraft.

As presented in figure 5.4 the clustering algorithm for $T_{ldg} = 45$ minutes with the density feature returns archetypal trajectories differ slightly from those shown in figure 4.2 where no density was considered in the clustering. From a trajectory point of view, this result is very satisfactory, meaning that our posterior generated trajectories will reflect the variation in trajectory length.

For $T_{ldg} = 45$ minutes, the cluster centers are able to capture the complete trajectories of aircraft inside Portuguese airspace and therefor, the generative model is able to generate the full trajectory of flights over Portugal.

It is important to have in mind that the different measures for the density feature presented section 3.3.1 generate different cluster centers when running the learning algorithm. This means that the potential reconstructed trajectories differ with the different density measures. Following a similar procedure done for $T_{ldg}$, we can compare the results given by the generative model by computing the CDF of the error for each of the density measurements and compare the average inference error of the landing time and $RMSD_{path}$ of each generated trajectory.

In figure 5.5, the CDF of the error $\epsilon$ defined in equation (5.3) for the three density methods are plotted. The model in blue is more accurate than the models considering the number of aircraft in LPPC and the TMA of Lisbon airport. This result is expected because the value of the coefficient of determination $R^2$ of the linear fit is the highest for the model that counts the number of aircraft inside the circle centered in Lisbon airport.

Table 5.3 reflects the same scope of results as in figure 5.5. The density measure counting the number of aircraft inside the circle centered in Lisbon airport is the more accurate of the three. Compared to the model without density, there is an improvement of about 15% both for $RMSD_{path}$ and $MAE_{ldg}$.

To conclude the analysis on the influence of the different density measurements on the generative model, we implement the same density models using now the dataset considering all aircraft flying as presented in subsection
Figure 5.4: Cluster centers for $T_{ldg} = 45$ minutes including the density feature. The model is able to capture all directions from which aircraft can come from.

<table>
<thead>
<tr>
<th>Density model, aircraft inside:</th>
<th>$MAE_{ldg}$ (in seconds)</th>
<th>$RMSD_{path}$ from the real trajectories (in meters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No density</td>
<td>241</td>
<td>7429</td>
</tr>
<tr>
<td>LPPC</td>
<td>236</td>
<td>7191</td>
</tr>
<tr>
<td>TMA</td>
<td>227</td>
<td>7002</td>
</tr>
<tr>
<td>Circle</td>
<td>205</td>
<td>6321</td>
</tr>
</tbody>
</table>

Table 5.3: Mean absolute error comparison and trajectory root mean square deviation evolution for the different density measurements. From the model without features, there is an improvement up to 15% both for $MAE_{ldg}$ and $RMSD_{path}$ with the inclusion of the density feature.

3.3.1 where the alternative dataset was presented. To do so, a table 5.4 similar to table 5.3 was computed in order to confirm whether or not this new dataset improves the results obtained with the dataset considering only the aircraft landing in Lisbon airport.

As expected, the lower value of the coefficient of determination of the linear fittings for this dataset are reflected in the generative model with higher estimation errors. In table 5.4, the error metrics used to evaluate the results are higher than in table 5.3, where only the aircraft landing in Lisbon were considered. Nevertheless, the inclusion of any density feature improves the estimations made by our base model without density. The next step is to include the weather feature, namely the wind direction to study its influence on the generative model.
Figure 5.5: Empirical CDF plot comparing the inference performance of the density model with the different metrics of density for the generative model using $T_{ldg} = 45$ minutes. As expected, the model considering the aircraft inside the circle centered in the airport outperforms the landing time inference made by the model without features and with other models of density.

<table>
<thead>
<tr>
<th>Density model, aircraft inside:</th>
<th>$MAE_{ldg}$ (in seconds)</th>
<th>$RMSD_{path}$ from the real trajectories (in meters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No density</td>
<td>241</td>
<td>7429</td>
</tr>
<tr>
<td>LPPC</td>
<td>238</td>
<td>7193</td>
</tr>
<tr>
<td>TMA</td>
<td>230</td>
<td>7184</td>
</tr>
<tr>
<td>Circle</td>
<td>212</td>
<td>6541</td>
</tr>
</tbody>
</table>

Table 5.4: Mean absolute error comparison and trajectory root mean square deviation evolution for the different density measurements using the dataset considering all aircraft flying in Portuguese airspace. Compared to the dataset where only aircraft landing in Lisbon airport are considered, this dataset increases the landing time inference error and flight path deviation.

5.3.5 Wind conditions at the airport

As mentioned in section 3.3.2, wind conditions at the airport are very important for the air operations as they impact the runway in use and therefore the general landing sequencing. Without information on the wind, our cluster centers reflect the approaches for both runways in Lisbon airport.

When looking at the posterior trajectories based on $P^n$ without wind information, our model fails to predict the runway on which the aircraft of interest will land. We show figure 5.6 as an illustration of a wrong inference on the runway.

Looking at the statistics, for 45 minutes, our model fails to estimate the runway in use in 13% of the trajectories generated with a mean landing time inference error of 480 seconds, more than doubling the 205 seconds of the mean estimation error for the 1000 Monte Carlo tests.

In order to improve our results, the two meteorological models, the first considering wind intensity and direction
and the second only considering the wind direction using a binary variable, were tested in the base model. Running
the clustering of the trajectories with the new features, the learning algorithm clustered the trajectories and the
cluster centers did not suffer any change. From a clustering point of view, no additional information is added by
the wind features regarding the learning process.

To see if any of their addition is helpful regarding landing time prediction, the generative model was rerun
including the wind features. In figure 5.7 we have plotted the CDF of the landing time inference error for the
base model with and without density to study the influence of this feature on the generative model. The feature
considering wind direction and its intensity has similar progress as the model with the binary variable up until an
error of 150 seconds. At that point, the model with the binary variable keeps returning a lower estimation error
than the model considering wind direction and wind. In addition, at around 400 seconds for the error, the base
model outperforms the model with the wind direction and intensity.

The introduction of the wind feature in the base model improves the overall performance of the generation as
seen in table 5.5. When comparing the two metrics tested for the inclusion of the wind, we see that the inclusion of
the binary variable is more effective than the inclusion of the wind intensity and direction. This can be explained by
the fact that in most cases, the intensity of the wind does not influence the landing in which the aircraft land on. It
is true that below 10 knots of intensity, aircraft can land with a tailwind, but in most cases, air traffic control decide
Figure 5.7: Empirical CDF plot comparing the inference performance of the wind metrics for the generative model using $T_{ldg} = 45$ minutes. The use of the binary variable outperforms the base model and the model considering the wind direction and intensity.

<table>
<thead>
<tr>
<th>Wind Model</th>
<th>$MAE_{ldg}$ (in seconds)</th>
<th>$RMSD_{path}$ from the real trajectories (in meters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No wind</td>
<td>241</td>
<td>7429</td>
</tr>
<tr>
<td>Binary variable</td>
<td>211</td>
<td>6538</td>
</tr>
<tr>
<td>Intensity and Direction</td>
<td>233</td>
<td>7189</td>
</tr>
</tbody>
</table>

Table 5.5: Mean estimation error comparison and trajectory root mean square deviation evolution for the inclusion of the wind measures in the generative model. By improving the runway landing inference to 96%, $MAE_{ldg}$ is improved by 30 seconds when compared to the base model.

which runway is in use by only looking at the wind direction. In total, the model is able to predict the runway correctly in 96% of the Monte Carlo tests for the binary variable and 90% of the cases with the wind direction and intensity. In conclusion, the inclusion of the wind variable helps to improve the initial accuracy of 87% when no wind feature is considered.

5.3.6 Complete model results

Now that the density and wind features were included and studied separately in the base model, we will combine the best models and evaluate the final performance of our model.

We have seen that independently, both the density feature and the wind feature are able to improve the performance of the generative model. For instance, the best results were attained for the density feature that counts the number of aircraft inside a circle centered in Lisbon airport containing the aircraft of interest. Regarding the density feature, the binary variable metric that informs on the wind direction showed a lower mean absolute error
for the landing time inference as well as the lower $RMSD_{\text{path}}$. Combining the best estimation for the density and the wind, we are able to plot the CDF curves and compute $MAE_{ldg}$ for the landing time inference and $RMSD_{\text{path}}$ that we used to evaluate the performance of the models so far.

![Empirical CDF plot comparing the inference performance of for the combination of the density and wind features for the generative model using $T_{ldg} = 45$ minutes. The plot in blue shows that the model with both features outperform the results of the model with only one of the features.](image)

The plot of the CDF of the estimation error of the landing time inference shows that the combination of the density and wind features allows the full generative model to perform better compared to the base model and the model with either the wind or density features. This shows that the two features complement each other and grant the model significantly better performance than the base model that only used position data.

![Empirical CDF plot comparing the inference performance of for the combination of the density and wind features for the generative model using $T_{ldg} = 45$ minutes.](image)

### Table 5.6: Mean estimation error comparison and trajectory root mean square deviation evolution for each feature and the combined model for $T_{ldg} = 45$ minutes.

<table>
<thead>
<tr>
<th>Model with side information</th>
<th>$MAE_{ldg}$ (in seconds)</th>
<th>$RMSD_{\text{path}}$ from the real trajectories (in meters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No side information</td>
<td>241</td>
<td>7429</td>
</tr>
<tr>
<td>Only density in circle</td>
<td>205</td>
<td>6321</td>
</tr>
<tr>
<td>Only binary variable</td>
<td>211</td>
<td>6538</td>
</tr>
<tr>
<td>Density and wind</td>
<td>178</td>
<td>5490</td>
</tr>
</tbody>
</table>

From a quantitative point of view, by adding both the density and wind features to the base model, we were able to improve our mean estimation error by 63 seconds, from 4 minutes and 1 second to 2 minutes and 58 seconds. Considering that we are estimating the landing time up to 45 minutes in advance, this error is very reasonable and gives TAP Air Portugal a very good indication of the estimated landing time of an aircraft. Trajectory root mean square deviation improved in excess of 26%, down from 7429 to 5490 meters showing the relevance of including additional features to position measurements.
5.3.7 User study

The trajectories generated by our model are based on real data coming from the EUROCONTROL dataset. Nevertheless, to evaluate how realistic our generated trajectories are, we performed a Turing Test, widely used in artificial intelligence, in order to determine whether or not our trajectories are different from the real ones [26].

To perform the Turing test, as in [4], take 25 random generated trajectories and 25 random real trajectories of our dataset. Then, we give the 25 pairs of trajectories to the experts and ask to distinguish the generated trajectory from the real one. The test is successful if the accuracy is around 50%, which indicates that the experts distinguish successfully the trajectories. If the accuracy of the experts is much higher than 50%, the generative model is unable to generate plausible trajectories.

<table>
<thead>
<tr>
<th></th>
<th>Pilot</th>
<th>Engineer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>54%</td>
<td>56%</td>
</tr>
</tbody>
</table>

Table 5.7: Turing test results for the test conducted with an airline pilot and engineer. The average accuracy is 55%, thus, the results are close to a random guessing exercise. The aviation experts are not able to successfully guess if a trajectory is real or not, meaning that the generated trajectories are indistinguishable from the real ones.

The Turing test was conducted with one engineer and one pilot working at TAP Air Portugal. The resulting accuracies are found in table 5.7. On average, the accuracy is equal to 55%, meaning that the pilot and engineer performed slightly better than random guessing. We can conclude that our model is able to generate trajectories that are, in practice, visually indistinguishable from real trajectories.
Chapter 6

Conclusion and future work

The present work focuses on predicting aircraft trajectories in crowded terminal areas. Using a data-driven approach, we developed a model to predict a posterior trajectory based on a held-out sample and infer on the landing time of the aircraft.

The base of this work is a dataset consisting of 13 AIRACS available in the Demand Data Repository of EUROCONTROL, and access to a range of FDR position measurements was granted by TAP Air Portugal.

An Exploratory Data Analysis was performed on both datasets to understand the data available before proceeding to the pre-processing phase. It allowed us to identify that the dataset coming from EUROCONTROL is complete, composed by every flight landing in Lisbon airport within a certain period of time. Unfortunately, the sampling rate of the measures points is variable and can be very low when the flight is far from the airport. On the other hand, the trajectories in the much smaller dataset from TAP Air Portugal had a very high and constant sampling rate throughout a flight. This data, coming from a system fed by the flight data recorder, presents a wide variety of inaccurate data that could compromise, from the beginning, the performance of our model. Following previous work, we formulated a data completion convex problem on EUROCONTROL data and compare the reconstructed trajectories with the ones available in the dataset of TAP Air Portugal. This ground truth was used to set the two parameters of the completion problem. The comparison between real trajectories from the dataset of TAP Air Portugal and our reconstructed trajectories revealed that after fine-tuning the reconstruction algorithm, the root mean square error of our reconstructed trajectories is 340 meters for a sampling rate of 10 seconds. To put this value into perspective, it corresponds to less than 5 seconds of flight inaccuracy. This error position can be decreased by decreasing the sampling rate, which will increase the overall size of our dataset and impact the computational performance.

Now that the trajectories are reconstructed and meet the requirements of the centroid-based clustering algorithm, the number of clusters was chosen following the elbow technique. With the number of clusters set to 20, the clustering algorithm generated 20 archetypal trajectories, each one representing a cluster. The analysis of the centers revealed that our algorithm seems to capture the majority of the directions from which aircraft come to land in Lisbon airport. Following previous work, the intra-cluster covariance matrices computed were approximated using singular value decomposition to retain the significant deviations inside the cluster. This way, outlier trajectories are eliminated and the shape of the covariance matrix is simplified, which will improve the efficiency of the generative
The generative model generating the predicted trajectory and infer on the landing time of an aircraft is based on a Gaussian Mixture Model that uses the cluster centers and the intra-cluster covariance matrices to evaluate the probability of given measurements belong to a specific cluster and its relative location inside it. The most probable cluster and position are chosen, from which the posterior trajectory is generated using a multivariate gaussian distribution based on the centroids of the chosen cluster and the intra-cluster covariance matrix. Landing time inference falls directly from the most probable position inside the chosen cluster. The performance of our model was evaluated for 1000 Monte Carlo simulations considering the last 35 minutes of the flights. The cumulative density function of the landing time estimation error was plotted and compared with the CDF computed from the average estimate landing time and the average historical landing time made available by TAP. The analysis of these curves showed that our model is, on average, up to 66% more precise than the metrics from TAP Air Portugal. By increasing incrementally $T_{ldg}$ we determined that up to 45 minutes prior to landing, our model is able to compute posterior trajectories without noise.

Different density measures were studied in order to find a direct relationship between the delay inside Portuguese airspace and the number of aircraft flying in it. It seemed that no clear linear relationship between these two variables could be found as the value of the coefficient of determination of the linear fitting was very small. In the end, the best density measure counted the number of aircraft inside a circle centered in Lisbon airport and delimited by the aircraft of interest, with a coefficient of determination close to 17%. It is a relatively low value that reflects the high amplitude of the delay when the density of aircraft is low. This linear model is more accurate for larger values of aircraft inside the airspace. In order to introduce this feature in the dataset, a column was added to the original trajectories containing the number of aircraft inside the so-called circle at every timestamp. This way, the clustering algorithm was able to reflect in its cluster centers the density of aircraft and generates posterior trajectories based as well on the density. In addition, the linear fitting of the delay with respect to aircraft density was added to the landing time inference to improve the initial landing time estimation of the model. The Monte Carlo simulations revealed that the density feature improves the landing time inference by 15%, from 241 to 205 seconds, whereas the trajectory root mean square deviation was improved by 1107 meters.

As we know, meteorology plays an important role during a flight. At the airport, wind direction impacts the runway in use as aircraft tend to take-off and land with a headwind. For this reason, we tested two ways of introducing the influence of the wind in the training dataset. The first experiment consisted in appending to the existing trajectories the wind direction and intensity and we compared this feature with a binary variable that indicated the wind direction for the runway in use. For both cases, the inclusion of the wind feature improved the average indicators. The best results were attained using the binary variable for which $MAE_{ldg}$ went from the original 241 seconds to 211 seconds and $RMSD_{path}$ from 7429 meters to 6538 meters. Globally, the accuracy of our model in predicting the runway in use improved from 87% to 96%. Now that we have established that individually, the inclusion of density and wind improve the performance of the model, both features need to be joined and the final model tested.

Similarly to the tests done to evaluate the base model and with every one of the features individually, the performance of the model with both the density and wind feature was evaluated. The quantitative evaluation of the performance shows that $MAE_{ldg}$ is now equal to 178, meaning that it improved 63 seconds, allowing to
estimate the landing time up to 45 minutes prior to landing with an average error of under 3 minutes. Regarding $RMSD_{path}$, the trajectory root mean square deviation improved 26%, from 7429 meters originally to 5490 meters.

The quality of the generated trajectories are underlined by the Turing test performed where the people tested averaged an accuracy of 55%, meaning that our generated trajectories are indistinguishable from the real trajectories.

The results from our model are satisfactory, bearing in mind the initial limitations of our dataset. The three main contributions of this work are: (1) using estimated trajectories to compute time of landing, (2) the study of accuracy improvement by addition of two new features indicated by domain knowledge, and (3) to evaluate and implement the delay estimator on real data.

6.1 Future work and operational implementation

Even though the proposed goals within the scope of this thesis were achieved, the overall model may be improved and further features contributing for a more precise and versatile model to predict aircraft trajectories in crowded terminal areas could be added.

Starting with the data in the model, the access to a more precise dataset with a higher resolution would benefit the overall precision of the model presented. In addition, the inclusion of model-based simulations, studying aircraft motion for accurate attitude, would guarantee that our trajectories are truthful to the attitude of commercial aircraft. There is a good room for improvement regarding path deviation from the real trajectory.

In parallel, the study of the density feature could be furthermore improved by looking at different ways of evaluating the delay caused by airspace congestion. Information on traffic taking-off would improve the landing time inference as the separation between take-off and landings need to be ensured.

Regarding operational implementation, this model can easily be used by TAP Air Portugal to help the different actors to monitor the arrival time of their flights in Lisbon airport. To predict the landing time, the model needs, in real-time, the position of all traffic scheduled to land in Lisbon airport and the position of the flights of TAP Air Portugal. A user interface should be added to visualize the landing time predicted by the model. In addition, useful information such as the sequence of landing aircraft can therefore be presented to the users.
Bibliography


Appendix A

Code Structure

Figure A.1: Code structure of the different .py files developed for the implementation of the model.
Appendix B

Definiteness of matrix M

In this appendix we find in which conditions M is positive definite.

As stated in section 3.2, to ensure that there is only one solution, the optimization problem needs to be strongly convex [21]. This condition is translated by

\[ M = A^T A + 2\lambda_1 D_2^T D_2 + 2\lambda_2 D_3^T D_3 \]  

(B.1)

being positive definite.

By definition, a Hermitian matrix H is positive definite if

\[ x^* H x > 0 \quad \text{for all nonzero} \quad x \in \mathbb{C}^N. \]  

(B.2)

Also, a positive definite matrix has all of its eigenvalues positive [27].

A special case of Hermitian matrices are real symmetric matrices. Fortunately, our matrix M falls in this case as \( A^T A \), \( D_2^T D_2 \) and \( D_3^T D_3 \) are all symmetric. In our case, the condition is (B.2) can therefore be written as

\[ x^T M x > 0 \quad \text{for all nonzero} \quad x \in \mathbb{R}^N. \]  

(B.3)

Before substituting the expression of M in (B.3), we can simplify its expression. The term \( A^T A \) is the product of the transpose of A by itself. In our case, matrix A is diagonal, meaning that \( A^T A = A^2 \). In addition, the entries in the diagonal of A, \( a_{ii} \), are 0 or 1, meaning that \( A^2 = A \). In addition, we can pose \( B = 2\lambda_1 D_2^T D_2 + 2\lambda_2 D_3^T D_3 \).

Therefore, we have

\[ M = A^T A + 2\lambda_1 D_2^T D_2 + 2\lambda_2 D_3^T D_3 \]
\[ = A + B \]  

(B.4)

By replacing (B.5) in (B.3), we get
\[ x^T M x = x^T (A + B) x = x^T A x + x^T B x > 0 \]  \hspace{1cm} (B.5)

At this point, we can evaluate the definiteness of \( M \) by looking separately to \( A \) and \( B \).

Matrix \( B \) is equal for all trajectories needing to be reconstructed. Looking at this matrix, composed by the addition of \( 2\lambda_1 D^T_1 D_2 \) and \( 2\lambda_2 D^T_3 D_3 \), it is easy to determine that it is a positive semi-definite matrix, as long as \( \lambda_1 \) and \( \lambda_2 \) are strictly positive because:

\[ x^T D^T_i D_i x = (D_i x)^T (D_i x) \geq 0 \]  \hspace{1cm} (B.6)

It means that we get \( x^T B x \geq 0 \), as long as the regularization parameters are strictly positive.

In order for \( M \) to be positive definite, we need to ensure that \( x^T A x > 0 \) for \( x^T A x + x^T B x > 0 \). We know that, for a diagonal matrix,

\[ x^T A x = \sum_{i=1}^{N} a_{ii} x^2_{ii} \]  \hspace{1cm} (B.7)

We said earlier that \( a_{ii} \) is equal to 0 or 1. For the sum in equation B.7 to be strictly positive, matrix \( A \) must have at least one diagonal entry equal to 1.

In conclusion, in order for \( M \) to be positive definite, and the optimization problem strictly convex, at least one element of the diagonal matrix \( A \) needs to be 1.