UNIVERSITY OF SOUTHAMPTON

FACULTY OF ENGINEERING AND THE ENVIRONMENT

Aeronautics, Astronautics, and Computational Engineering

# Applications of uncertainty quantification in the spatial analysis of trajectories

by

**Willem Johannis Eerland**

Thesis for the degree of Doctor of Philosophy

1st November 2017

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND THE ENVIRONMENT
Aeronautics, Astronautics, and Computational Engineering

Doctor of Philosophy

APPLICATIONS OF UNCERTAINTY QUANTIFICATION IN THE SPATIAL ANALYSIS OF TRAJECTORIES

by Willem Johannis Eerland

This dissertation looks at analysing security threats via passively guided rocket based weapons of important infrastructure, e.g., airports, military bases, and power stations. It also examines the vulnerability of flight routes, specifically near the airport where the aircraft fly at a relatively low altitude. A key component of the applications presented in this dissertation is quantifying uncertainty, such that scenarios where deviations from *a plan* occur can be analysed and associated risks can be mitigated. First, a method is proposed to capture motion patterns found in trajectory data. To achieve this, all data are assumed to be generated from a probabilistic model that takes the shape of a Gaussian process. By relying on the Gaussian process framework, the method is able to handle noisy and missing trajectory data. Aircraft trajectory data measured in the vicinity of various airports are analysed via the proposed method. In these examples, flight corridors are visualised and the probability of conflict based on the structure of the corridors is quantified. Furthermore, a strategy aimed at performing the large scale analysis of security risks via a terrain map is introduced. This strategy has shown to improve the detection rate of security threats by at least 20% and detect all risky locations in half the time compared to a brute-force approach. Finally, an open-source stochastic, six-degrees-of-freedom rocket flight simulator is introduced. This simulator assists with the conceptual design of sounding rockets, and produces confidence bounds for a landing location. The uncertainty quantification of the landing location is expanded to the entire flight by capturing the produced (by the simulator) trajectory data in a probabilistic model via the proposed method. These applications lead towards a data-driven approach to the analysis of security risks of important infrastructure.

# Declaration of Authorship

I, Willem Johannis Eerland , declare that the thesis entitled *Applications of uncertainty quantification in the spatial analysis of trajectories* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: Eerland and Box (2016), Eerland, Box and Sóbester (2016), Eerland, Box, Fangohr et al. (2017b), Eerland, Box and Sóbester (2017), Eerland, Box, Fangohr et al. (2017a), and Eerland, Box, Fangohr et al. (2017c).

# Acknowledgements

I would like to thank my first primary supervisor Simon Box for proposing this project and providing me with excellent guidance and advice throughout the start of my research at the University of Southampton. I am also extremely grateful to András Sóbester for his willingness to guide and assist me through the final phases of my thesis. For the assistance to make this work accessible to many, I thank Hans Fangohr for his contributions.

The project itself would not have been possible without the contribution from the industry sponsor Cunning Running Ltd, where I received valuable assistance from many who work there, but in particular I wish to thank Chris Barrington Brown and Mark Cooter.

Finally, I would like to thank my parents, for supporting me in whatever I choose to do.

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **aPC** | arbitrary Polynomial Chaos |
| **ATM** | Air Traffic Management |
| **CDF** | Cumulative Distribution Function |
| **CSV** | Comma Separated Values |
| **DBSCAN** | Density-Based Spatial Clustering of Applications |
| **DFW** | Dallas Fort Worth airport |
| **DTW** | Dynamic Time Warping |
| **ECDF** | Empirical Cumulative Distribution Function |
| **EKF** | Extended Kalman Filter |
| **EM** | Expectation-Maximization |
| **GP** | Gaussian Process |
| **GPR** | Gaussian Process Regression |
| **GUI** | Graphical User Interface |
| **INM** | Integrated Noise Model |
| **IPP** | Impact Point Prediction |
| **K-S** | Kolmogorov-Smirnov |
| **KDE** | Kernel Density Estimate |
| **OPTICS** | Ordering Points To Identify the Clustering Structure |
| **PCA** | Principal Component Analysis |
| **ROSIE** | ROcket Simulation and Impact Estimation tool |

| | |
|---|---|
| **RPGs** | Rocket-Propelled Grenades |
| **TFM** | Traffic Flow Management |
| **XML** | Extensible Markup Language |

# Glossary

**Calligraphic Lettering**

| | |
|---|---|
| $\mathcal{C}$ | complexity measure |
| $\mathcal{GP}$ | Gaussian process |
| $\mathcal{L}$ | likelihood function |
| $\mathcal{N}$ | Gaussian function |

**Greek Symbols**

| | |
|---|---|
| $\beta$ | precision parameter |
| $\hat{\beta}$ | precision parameter estimate |
| $\Delta$ | Mahalanobis distance |
| $\epsilon$ | $\epsilon$-neighborhood parameter of the DBSCAN algorithm |
| $\mu$ | mean, scalar |
| $\hat{\boldsymbol{\mu}}$ | mean vector estimate |
| $\boldsymbol{\mu}$ | mean vector |
| $\phi$ | basis function for a scalar |
| $\boldsymbol{\phi}$ | linear basis |
| $\boldsymbol{\Phi}$ | block diagonal linear basis |
| $\sigma$ | standard deviation |
| $\sigma^2$ | variance |
| $\hat{\boldsymbol{\Sigma}}$ | covariance matrix estimate |
| $\boldsymbol{\Sigma}$ | covariance matrix |

| | |
|---|---|
| $\tau$ | normalised time [-] |
| $\boldsymbol{\tau}$ | normalised time vector [-] |
| $\boldsymbol{\omega}$ | linear basis weights vector |

**Latin Symbols**

| | |
|---|---|
| $D$ | number of dimensions |
| $d$ | dimension identifier |
| $d_w$ | weighted distance |
| $\mathbb{E}$ | expectation |
| $\mathbf{I}$ | identity matrix |
| $J$ | number of basis functions |
| $j$ | basis function identifier |
| $\mathbf{k}$ | covariance kernel |
| $k$ | order of complexity |
| $M$ | number of data-points |
| $\mathbf{m}$ | mean function |
| $m$ | data-point identifier |
| $N$ | number of trajectories |
| $n$ | trajectory identifier |
| $n_{grid}$ | number of datapoints found within a *grid square* |
| $n_{regions}$ | number of regions assigned to the map |
| $p$ | parameter used to select optimal sampling location, prioritises between the centre and the highest point |
| $\mathbf{S}$ | ensemble of sampled trajectories |
| $\mathbf{C}$ | covariance matrix of an individual trajectory |
| $t$ | time [s] |
| $\mathbf{t}$ | time vector [s] |
| $\mathbf{u}$ | vector holding a three-dimensional coordinate [m] |

| | |
|---|---|
| $\mathbf{v}$ | column vector holding three-dimensional coordinates [m] |
| $g$ | single point, scalar |
| $w$ | parameter used to generate regions, prioritises between lateral distance and surface height |
| $\mathbf{g}$ | single dimension vector |
| $(x)^{grid}_{avg}$ | average of eastings of a grid square [m] |
| $(x)^{region}_{avg}$ | average of eastings of a region [m] |
| $\mathbf{Y}$ | ensemble of trajectories |
| $(y)^{grid}_{avg}$ | average of northings of a grid square [m] |
| $(y)^{region}_{avg}$ | average of northings of a region [m] |
| $\mathbf{y}$ | northing vector [m] |
| $(z)^{grid}_{avg}$ | average of surface height of a grid square [m] |
| $(z)^{grid}_{max}$ | maximum of surface height of a grid square [m] |
| $(z)^{grid}_{var}$ | variance of surface height of a grid square [m$^2$] |
| $(z)^{region}_{avg}$ | average of surface height of a region [m] |
| $(z)^{region}_{var}$ | variance of surface height of a region [m$^2$] |

# Chapter 1

# Introduction

This chapter aims at providing a general overview of how the topic, sponsors, and goals link together. First, section 1.1 introduces uncertainty quantification and the specific application in engineering. Following this, the sponsors, a combination of an industry sponsor and a research council, are introduced in section 1.2. Furthermore, section 1.3 states the aims and objectives of the thesis. And finally, section 1.4 presents an overview of the entire document.

## 1.1 Uncertainty quantification in engineering

Uncertainty quantification is the science of quantifying and characterising uncertainty in computational and real world systems. Such a quantification is useful for applications across many industries, for example designing aircraft (Green, Lin and Khalessi 2002), ships (Diez et al. 2014), windmills (Padrón et al. 2014), or computer chips (Greene et al. 2011). For these, and many other products, design and analysis is done digitally. Both the design and analysis can be performed by choosing ideal material properties and optimal dimensions to ensure performance under specific conditions. However, such an approach can be hardly called robust, as there are many different sources of uncertainty. Starting from mining the raw materials to the moment the product fails, various levels of uncertainty are added. Examples are the source of raw materials used, the manufacturing method selected, the environmental conditions the product finds itself in, and the wear and tear to which the product is exposed.

Conditions supplied by the real world are always different from the planned original design. However, by including the uncertainty for a range of conditions, it is possible to understand how the design will actually perform in normal, and more extreme conditions. This assists in obtaining a better understanding the design and the corresponding requirements. The design will end up being more robust, risks are mitigated, and better, more informed decisions can be made.

As these uncertainty quantification methods focus on the input/output behaviour, uncertainty quantification may assist in various other tasks other than product design, e.g., the power grid in the energy sector (Constantinescu et al. 2011). Besides new technologies and changes in regulations, the power grid is constantly plagued by uncertainty in supply and demand. Here, uncertainty quantification can use demand forecasts and system simulations to assist with improving reliability and efficiency.

The next two sections introduce the sponsors and state the aims and objectives. This provides an overview of the tasks for which uncertainty quantification will be applied in this dissertation.

## 1.2 Sponsors

The project is sponsored by both the Engineering and Physical Sciences Research Council (EPSRC) under research grant *EP/L505067/1*, and the corporate sponsor Cunning Running Software Ltd. (referred to as Cunning Running in the remainder of the report). As is stated on their website (Cunning Running Software Ltd. 2017), they are company based in Romsey that develop vulnerability assessment software. Their solutions enable end-users to identify and manage threats to critical infrastructure and high-value assets such as aircraft.

Many of their solutions are driven by the question of *where's the terrorist?*, which is of interest both in military and civil counter-terrorism situations. Their customers, as stated on their website, are aviation security forces at airports in the United Kingdom, the United States of America, Australia and others. Furthermore, their software is also in-service with the Royal Air Force Regiment, the Royal Australian Air Force and the French Government.

There are numerous challenges involving the development of such vulnerability assessment software. Several of these challenges have been tackled using direct and indirect results of the work presented in this thesis. Furthermore, where possible, examples on how the results are applied to these applications are presented.

One of the major benefits of having Cunning Running as a sponsor, is the significant impact the work has on the world. As previously mentioned, their software product is exported to a number of countries, thus the contributions made to their products are channelled directly to the end-user (at time of writing, over 90 airports in the US alone). This contribution to aviation security has been recognised by industry; The Honourable Company of Air Pilots and the Air Pilots Trust have awarded the work with the prize for Aviation Safety Research 2016.

## 1.3 Aims and objectives

The aim of the dissertation originates from the corporate sponsor and envisions a system able to produce a tactical analysis based on measured data and expert's know-how. Such a system may contain sensitive data and therefore must be contained on an isolated (possibly portable) computer system. Furthermore, in case of an immediate threat, initial results are desired within a reasonable (2-3 hours) timeframe. These requirements result in a constraint on the computational costs as there will be no access to a high-performance computer cluster and the available time may be limited.

Therefore, this dissertation covers the following applications related to uncertainty quantification. First, the uncertainty quantification of historical trajectory data is examined. A key element here is that real life rarely goes as planned, and capturing the uncertainty using historical data allows for a better subsequent analysis compared to assuming the planned path. Secondly, uncertainty quantification in a large scale security threat analysis is considered. A strategic analysis of a $30 \times 30\ km^2$ area encompasses 900 million evaluations. Hence, strategies are explored to avoid having to evaluate all points with certainty. Finally, effects of rocket flight trajectories for a given uncertainty in numerous conditions are analysed. However, instead of performing endless empirical tests, these trajectory data used in the analysis are produced using a rocket flight simulator, where lauching and tracking 100 *similar* rocket flights is very costly (both in financial cost and time).

To summarize, the aims and objectives of this work are:

1. Develop a method to extract information from a trajectory dataset, specifically to capture the uncertainty found in measured trajectory data with respect to the expected trajectory
2. Establish a strategy suitable to perform a large scale security threat analysis
3. Construct a tool capable of of producing rocket flight trajectories based on uncertain launch- and atmospheric conditions

## 1.4 Thesis outline

The remainder of the dissertation is organised as follows. Chapter 2 provides a review of trajectory analysis methods, and indicates where such methods are currently being applied.

Chapter 3 proposes a systematic probabilistic modelling approach using historical data. Such a model assists in visualisation tasks and is capable of coping with missing and noisy data. Several applications are included in this chapter, such as visualising corridors capturing a given percentage of trajectories and generating complexity maps for identifying potential conflict areas.

Chapter 4 applies the probabilistic modelling approach to aircraft trajectories as measured by ground-based radar. These case studies demonstrate the applications in aerospace, based on the work described in chapter 3.

Chapter 5 considers optimal sampling strategies for mapping data. It considers the associated computational cost and both *rural* and *urban* environments.

Chapter 6 describes various methods for uncertainty quantification of stochastic systems. Furthermore, this chapter also provides an overview of rocket dynamics applied in research, and available rocket simulation software in general.

Chapter 7 introduces a stochastic rocket simulator, producing uncertainty bounds of impact location and expected performance (e.g., maximum altitude reached). Furthermore, the probabilistic modelling approach is integrated with the simulator via a Python package developed by the author called Teetool. This integration supports synchronization of trajectory meta-data, providing better results in the probabilistic analysis.

Finally, chapter 8 contains a summary of conclusions provided by the research and recommendations for future work.

# Chapter 2

# Trajectory data analysis and applications

This chapter aims at identifying where trajectory analysis is applied. It reviews the current methods used, and identifies the gaps in the literature. Given the setting of the following chapters found in this thesis, a specific focus on the aerospace sector and aircraft flight trajectories is included throughout this chapter.

Section 2.1 provides a definition of a trajectory as referred to in this work, and describes how the analysis may assist in a variety of fields. It also introduces the concept of a motion pattern, i.e., repetitive motions and/or specific paths found in trajectory data. Section 2.2 provides an overview of clustering techniques applied to identify motion patterns. Section 2.3 describes the methods used to model the motion patterns, once the motion patterns have been identified. The term *modelling* here refers to the creation of a statistical model that indicates where objects may be located in space, given the available trajectory data. Section 2.4 covers a mathematical method to capture time-series called Gaussian Process (GP), and shows the applications to trajectory analysis as already seen in the literature. Finally, section 2.5 concludes with some final remarks, highlighting the importance of a new approach to trajectory analysis, so far missing in the literature.

## 2.1 Trajectory data

Given the ambiguity of *trajectory data*, we start with the definition of a trajectory. The Oxford dictionary provides two definitions for **trajectory**:

1. *The path followed by a projectile flying or an object moving under the action of given forces.*
2. *Geometry – a curve or surface cutting a family of curves or surfaces at a constant angle.*

This dissertation concerns itself with the first definition; a type of trajectory that is found in everyday life. The analysis of these trajectories has become more relevant in the last decades as trajectories are being stored digitally. If the availability of data is any indication of the desire for analysis, it should be noted that 90% of the world's data is generated in the last two years (IBM 2016). This number can be attributed by the increase in number of sensors, and the reduced costs for data storage. Indeed, technological developments in the last years have increasingly provided means to gather and store trajectory data. This has enabled access to an abundance of sources, ranging from tracking cars and aircraft to people and wildlife. And in some cases, e.g., Strava (Strava 2017), people choose to upload trajectory data to analyse their activity or simply share it with other people. A consequence of these developments is a large amount of trajectory data, also called *spatio-temporal* data.

In most cases, these trajectories have patterns in them, as the tracked objects perform repetitive motions and/or follow specific paths. These are referred to as motion patterns. Extracting these motion patterns may be important in a variety of fields (Bak et al. 2015). For example: air traffic controllers wishing to monitor the highly regulated airspace around airports (Gariel, Srivastava and Feron 2011), or environmental policy makers wishing to investigate aircraft noise abatement (Girvin 2009); researchers who fly passively controlled atmospheric sounding rockets and wish to predict safe regions of splashdown (Box, Bishop and Hunt 2010); researchers who develop models of pedestrian movement in urban centres (Helbing and Molnár 1995); to name only a few.

Specifically for aerospace, there are challenges in visualising these large amounts of trajectory data (Marzuoli, Hurter and Feron 2012; Hurter, Conversy et al. 2014). Once the data has been processed, it can assist with the decision-making process: Visualise changes in aircraft trajectories as result of a new protocol, re-organise the airspace to reduce traffic density, and track environmental considerations (e.g., fuel consumption, noise pollution).

The next section provides an overview of the first step in handling motion patterns, identifying the patterns within the trajectory data.

## 2.2 Identifying motion patterns

This section focusses on identifying motion patterns within the trajectory data. In more general terms, this is also referred to as *data mining*, the analysis of data, as to discover patterns or relationships using statistical methods. Several methods that will be discussed in this section reduce the trajectories to a number of *features*, in effect projecting a single trajectory to a point in a high(er) dimensional space. This allows the application of point-clustering algorithms, which identifies motion patterns based on their position in the high-dimensional space. In this case, a pattern is described as a collection of points (multiple trajectories) that show a relation to each other in this high dimensional space. However, for this to be successful, care must be taken to parametrise the trajectory as a number of features that keep the relevant information, where the relevant information itself is very dependent on the question being asked.

This brings us to the various methods of parametrizing a trajectory, where a key result is that each trajectory has an identical number of features, as to *fit* into the same high dimensional space as previously mentioned. The most basic version is to resample the trajectory data, keeping only a finite number of data-points. This assumes no specific shape of the trajectories, however smoothness information (the relation between data-points) is lost in the process. Smoothness information can be stored in features by applying polynomial basis functions, an approach that has been successfully applied to weather patterns (Gaffney and Smyth 1999; Camargo et al. 2007). Other basis functions available are Bézier curves (Faraway, Reed and Wang 2007), or even a Fourier decomposition (Annoni and Forster 2012). For the latter, the focus is on cyclic behaviour, as expected when applying a Fourier analysis. A method that resembles resampling in the way that it doesn't assume a specific shape of the trajectories, is parametrizing via radial basis functions (Park and Sandberg 1991), where a parameter related the width of each node to capture the smoothness information is included.

Once a finite set of features have been extracted from the trajectory data, there are numerous clustering algorithms available to detect patterns and group the trajectories accordingly. There is extensive literature available, describing numerous methods and techniques capable of extracting *relevant* information, and ignoring *irrelevant* information. Most, if not all, techniques rely on the spatial information, i.e., points close together form a pattern, and points far away from these patterns are outliers. One such technique is $k$-means clustering (Hartigan and Wong 1979), which finds the optimal allocation based on spatial distance given $k$ clusters (i.e., patterns). This will assign each point to the nearest cluster as defined by their mean (the average location of all points within the cluster). Additionally, there are methods such as Density-Based Spatial Clustering of Applications (DBSCAN) (Ester et al. 1996) and Ordering Points To Identify the Clustering Structure (OPTICS) (Ankerst et al. 1999) that allow filtering, ignoring the outliers previously mentioned. A method similar to k-means is a mixture of Gaussians, however, instead of presenting the answer to which cluster (i.e., pattern) the point (i.e., trajectory) belongs as an absolute truth, it supplies the probability of a point belonging to a cluster (Bishop 2006). Furthermore, under the assumption that the data are Gaussian distributed, a mixture of Gaussians is able to express a preference to the number of clusters by comparing the likelihood between the models.

A survey to learn motion patterns by clustering is available (Morris and Trivedi 2009), and includes methods such as Dynamic Time Warping (DTW) (Berndt and Clifford 1994; Müller 2007), which not only focus on the spatial distance, but also take the temporal component into account. An important conclusion of this survey is that prior knowledge is the deciding factor on which method to use – there is no silver bullet.

There are other methods to identify relations between trajectories. One such method is the partition-and-cluster framework (Lee, Han and Whang 2007; Buchin et al. 2011), where the trajectories are partitioned in sub-trajectories (e.g., straight lines, constant curves), and grouped together based on similarity (e.g., distance). This allows the identification of common segments found within the data, such as an often travelled corridor. However, this framework also leaves the data divided, losing the relation between the short segments, and the behaviour of an entire trajectory. In other words, it captures the behaviour of the segments, but not the behaviour of a single trajectory as found in the data.

Another clustering approach can be found in fitting multiple vector fields (Ferreira et al. 2013) to discover underlying patterns. While this method is successful in matching patterns with general trends (e.g., move towards the East, then move towards the North), it does not discriminate strongly towards shifts in trajectories (e.g., a trajectory can shift 50km towards the South, yet still have a similar trend).

In the case where the information required for clustering cannot be captured in designed features, it is possible to consider using latent variables (Liu et al. 2014). Such an approach is particularly helpful when there is an unstructured scene (e.g., no designated paths), and it is not obvious which features are helpful in the clustering step. While such a method can assist in clustering when little is known about a scene, it is not always clear what the latent variables represent (e.g., position, direction, etc.). As a result, when it is not clear what the latent variables represent, it is unclear what variable is prioritised in the clustering step.

The next section focusses on identifying motion patterns specifically for aircraft flight trajectories.

### 2.2.1 Identifying motion patterns of aircraft flight trajectories

This part of section 2.2 focusses specifically on aircraft flight trajectories for identifying motion patterns. The reason why a specific focus is required, and why the general approach is not sufficient, relates back to the question of which features to use in the selected clustering algorithm, as the relevant information captured in the features is very problem dependent.

Multiple aircraft trajectories often follow an identical flight path, giving rise to the idea of pathways in the sky. The term flight path here is used to indicate the planned route, where a trajectory is the route actually flown. As such, it is possible to cluster aircraft trajectories based on a common flight path. The information used to cluster based on the flight path need not to be limited to coordinates found in a trajectory, but can also include features specific to the scenario, such as the bearing (Eckstein 2009). This feature is important as aircraft often fly based on a specific bearing, hence being a good identifier for having a common flight path. Eckstein (2009) has applied the resampling of the trajectory data and included additional features such as their bearing to the k-means clustering algorithm. Gariel, Srivastava and Feron (2011) extended this approach by replacing the k-means the DBSCAN clustering algorithm, also allowing the filtering trajectories that occur less frequent.

Often, there are additional data to be considered that are not available in the trajectory data. For example, in the case of aircraft flight trajectories near an airport, this could be changes in weather conditions (Grabbe, Sridhar and Mukherjee 2014). This, as the wind direction dictates which runways are used for approach, and which are used for departure. In both cases, the aircraft will be positioned into the wind, such that the wind assists in decreasing the relative velocity to the Earth, resulting in using a shorter part of the runway for both landing and takeoff. Other data may include the local time, as larger airports such as Heathrow apply runway alternation to avoid exceeding the locally allowed noise pollution level (Heatrow 2017).

The next section considers modelling an individual motion pattern after the trajectories have been clustered (e.g., grouped) accordingly.

## 2.3 Modelling a motion pattern

The clustering techniques as discussed in the previous section dissects the data into corresponding, smaller, chunks of data, however, no information on the trend is captured.

This section focusses on techniques that model the *motion pattern*, where modelling (i.e., create a statistical model of where they may be located in space) and visualising motion patterns are important steps in understanding the behaviour of the trajectory data.

The techniques discussed in this section assume that the trajectory data follow a single motion pattern, where the identification of motion patterns can have occurred using one the techniques described in the previous section. There are also approaches that model the behaviour of multiple motion patterns not requiring the clustering step. An example of such an approach is the method that produces a vector field via Gaussian processes (Ellis, Sommerlade and Reid 2009), where a model is trained to store the probability of a point moving in a given direction. This avoids the clustering entirely, as each position in space learns the behaviour (e.g., moving North, North-East, East, etc.) based on the trajectory data. In a similar fashion the underlying pattern of items drifting on the ocean surface can be identified by learning stochastic models (Sykulski et al. 2015). However, in these cases the structure of the individual motion pattern is lost, as these methods learns the individual decisions at each point in time only.

It is possible to model and visualise the patterns based on the density, often referred to as density maps. This can be done either by aggregating the trajectories in a grid (Meratnia and de By 2002), or placing a radial basis function at each data point (Lu et al. 2014) to obtain a continuous version. While these methods are successful in identifying and modelling the often travelled paths, they tend to ignore the (relatively) low density paths.

According to Morris and Trivedi (2013), the approach of first identifying, and then modelling the motion patterns is the most frequently used in literature focussed on vehicle traffic behaviour. This work also refers to other techniques used in capturing motion patterns (e.g., using a decision tree of paths to follow), however this section focusses on techniques that are able to produce a complete spatial picture. An example of this is seen in figure 2.1, where the actual bounds can be identified.

Most existing approaches to model the common path and dispersion do so in discrete space. Makris and Ellis (2005) capture pedestrian movement via visual surveillance,

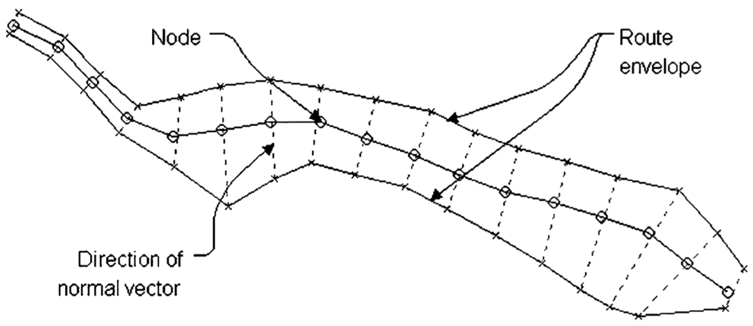


Figure 2.1: An example of a route model, capturing the main axis and borders of a specific route (Makris and Ellis 2005).

and after clustering based on a common path, they model the two-dimensional trajectories in a route model. This route model contains a mean function, the average of all trajectories, described by a discrete number of equidistant nodes. Furthermore, the dispersion of the trajectories at each node is calculated by examining the cross-section perpendicular to the local node direction, and fitting a univariate Gaussian to the points where the trajectories cut this cross-section. The result, seen in figure 2.1, is an envelope with a spline-like representation, capturing a given percentage of trajectories in a two-dimensional space.

In Hu et al. (2006), Saleemi, Shafique and Shah (2009), and Morris and Trivedi (2011) the common path and dispersion is also captured in a model. However, while the mean function is captured in a similar fashion (using a discrete number of equidistant nodes), the dispersion is captured via a two-dimensional multivariate Gaussian. Therefore, the bounds of the envelope based on a constant variance are not visualised as a single line (corresponding to a univariate Gaussian), but as an ellipse (corresponding to a two-dimensional multivariate Gaussian). Here the envelope capturing a given percentage is shown by merging the ellipses (evaluated at each node) into an area. The paths themselves are stored by capturing the transition between nodes.

The next section focusses on modelling the motion pattern of aircraft trajectories.

### 2.3.1 Modelling a motion pattern of aircraft flight trajectories

This section introduces many of the areas that are closely related to, or are expected to benefit from, motion pattern modelling. It includes work on conformance monitoring, visualisation techniques, quantifying the air traffic complexity, and constructing flight corridors. Finally, the current approach for calculating noise abatement around an airport is also discussed.

Conformance monitoring aims to detect any excessive deviations of aircraft from their expected behaviour. Key elements in a monitoring framework are comparing the current state versus the expected states (Reynolds and Hansman 2002). Analysis based on flight test data has shown that monitoring the states for conformance performs best when the intended flight path is straight and level (Reynolds and Hansman 2003). Where particular challenges lie in state transitions, such as near a waypoint where a change of heading occurs, or during an altitude change, mainly due to the larger margins and late detection (Reynolds and Hansman 2005).

A probabilistic approach to determining future states can be taken by applying Monte Carlo simulations based on previously measured data (Prandini, Hu et al. 2000). This methodology is aimed at predicting the future states of the aircraft for short- and mid-range, used for detecting future conflicts between aircraft. Path prediction can be extended with an *intent* model, based on information inferred from the previously observed data (Krozel and Andrisani 2006; Yepes, Hwang and Rotea 2007). This approach to path prediction is implemented as a basis for conflict detection (Hwang and Seah 2008) and conformance monitoring (Seah, Aligawesa and Hwang 2010).

Basic components of the conformance monitoring system can be decomposed into trajectory prediction, calculation of deviation metrics, and decision logics for declaring non-conformances. By taking a probabilistic approach, the deviation metric can defined as a probability with the benefit that it can take into account uncertainties (e.g., surveillance errors) (Zheng and Zhao 2012). The deviations as described in this work are assumed to be deviations from a nominal (or assigned) trajectory.

However, modelling air traffic as if it moves along a limited set of predefined air routes is unrealistic, therefore there is a need to model the air traffic flow from the trajectory data as measured (e.g., by radar). One approach to visualise the airspace traffic flow is graph bundling by density estimation (Hurter, Ersoy and Telea 2012). By applying graph bundling by density estimation, it is possible to gain an understanding of the traffic flow based on the trajectory data (Marzuoli, Hurter and Feron 2012). This approach has been included in an interactive program, capable of visualising traffic information with the focus on air traffic flow analysis (Hurter, Conversy et al. 2014). These and the aforementioned methods focus on visualising the traffic flow in two dimensions.

Visualising large datasets in three dimensions brings its own challenges, including the occlusion of data (Elmqvist and Tsigas 2008). Technically, it is possible to process the data real-time by using a GPU-based visualisation pipeline (Buschmann, Trapp and Döllner 2014), but the occlusion of information remains a problem. Visual analysis tools based on air traffic density in three dimensions are available (Albrecht, Lee and Pang 2012). This approach requires the airspace to be discretized in a three-dimensional grid, and does not automatically identify the motion patterns. Furthermore, the conflict probability is calculated by setting a separation minima and combining the probabilities of any grid-points that fall within this separation bound. This fails to take into account that the aircraft are flying in the same direction and therefore cause no conflict.

Besides visualising the trajectory data, it is possible to present alternative information that identifies likely conflict areas. The Traffic Flow Management (TFM) is a function of the Air Traffic Management (ATM) systems that operates on a long term horizon by defining the flow patterns to ensure an efficient organization of the overall air traffic (Prandini, Piroddi et al. 2011). A term used to express the amount of effort required to manage the airspace is *air traffic complexity*. Here, the relation between the air traffic complexity and presence of conflict areas is such that a reduction in the number of conflict situations (i.e., situations requiring a deviation from the intended flight path), reduces the overall burden on the system and thus reducing the air traffic complexity. Air traffic density, estimated via flow models has been applied to predict the regions where a high density of air traffic will occur (Menon, Sweriduk and Bilimoria 2004;

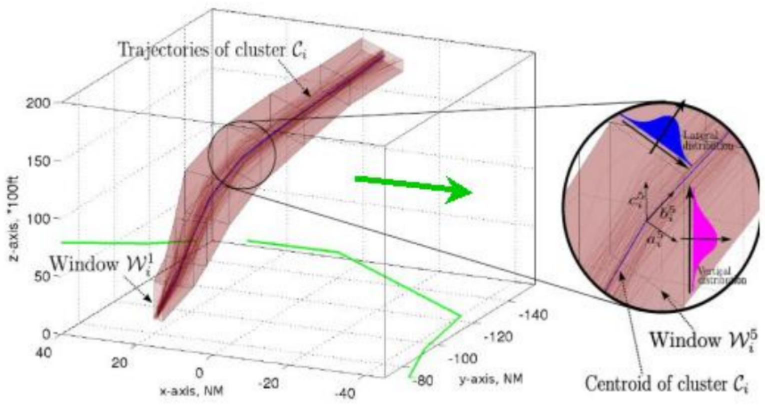


Figure 2.2: An example of three-dimensional flight corridor, capturing the dispersion in vertical and lateral direction (Salaun et al. 2012).

Sridhar, Soni et al. 2006). These flow models work on a global scale, and only identify a particular region such as near an airport, in a holding area, or 'in transit'. On a local scale, analysing potential conflict between aircraft, ellipsoids have been used as a measure for air traffic complexity in three-dimensional airspace, with a focus on the *traffic structure* (Prandini, Putta and Hu 2010). When the focus is *air traffic controller workload*, there is evidence that including factors such as heading, speed, and altitude change provide a better measure than traffic density only (Laudeman et al. 1998; Sridhar, Sheth and Grabbe 1998). In both is the air traffic complexity not directly linked to the traffic density, however it is possible to estimate a density at which the airspace becomes saturated (Jardin 2005).

Salaun et al. (2012) does not visualise the raw trajectory data nor the air traffic complexity, but constructs three-dimensional flight corridors from aircraft trajectories. These corridors are described by a mean function, the average of all trajectories, stored as a discrete number of equidistant nodes in a three-dimensional space. On the surface area perpendicular to the nodes is a window frame, constructed by a vertical and lateral line, through which the trajectories manoeuvre. The variance in the vertical and lateral direction is calculated independently by fitting a univariate Gaussian distribution. The result is seen in figure 2.2.

An application for flight corridors, be it in two dimensions, is found in the laws and regulatory schemes for noise abatement. The current methodology on calculating noise contours around civil airports in Europe uses several sub-tracks to model the dispersion

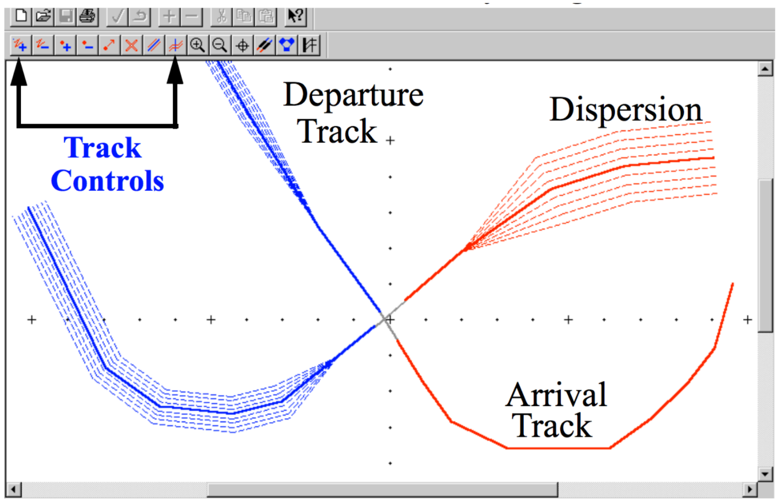


Figure 2.3: An example of the input for INM, including the track dispersion (Trani 2016).

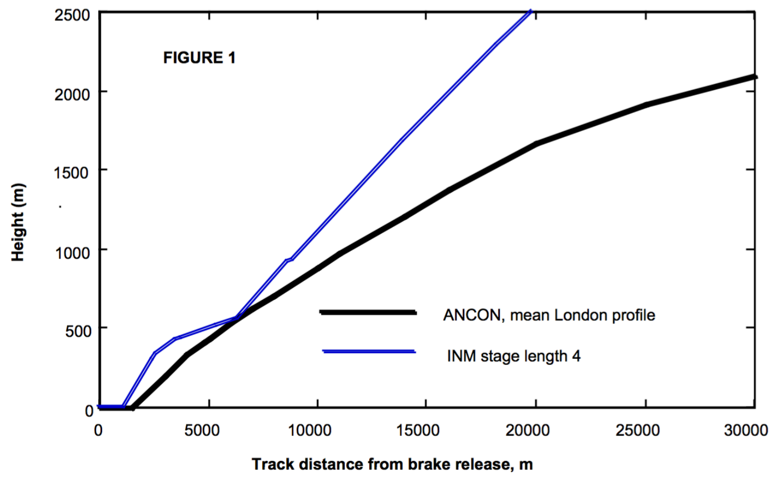


Figure 2.4: Average height profile of B767-300 (Jopson, Rhodes and Havelock 2002).

along a single flight path (European Civil Aviation Conference 2016). An example of the input for the Integrated Noise Model (INM) (Boeker et al. 2008) is seen in figure 2.3. For purpose of estimating the sound produced, the dispersion is modelled using weighted (percentage of total traffic) trajectories that are to represent the entire traffic flow. In the ongoing situation, the vertical dispersion not modelled, and the expected vertical flight path is taken from the aircraft manual. However, evidence indicates that the average vertical paths used are not what occurs in practice, as the airliners adjust the procedure to reduce operating costs (Jopson, Rhodes and Havelock 2002). A visualisation of the difference in the INM trajectory and the mean trajectory as seen in the data is seen in figure 2.4.

The next section introduces Gaussian processes and the applications in various fields regarding trajectory analysis.

## 2.4 Trajectory analysis via Gaussian processes

The trajectory data seen in this thesis can, in a more general context, also be called *functional data*. The reason is, that each dimension (e.g., easting, northing, and altitude) can be described as a function of another parameter (e.g., time). Consequently, it is possible to apply Gaussian Process Regression (GPR), a technique that provides a mathematical framework to deal with these *functional data* in a continuous and probabilistic manner. A GP is defined by its mean function $\mathbf{m}(t)$ and covariance kernel $\mathbf{k}(t, t')$, where $t$ and $t'$ are two (possibly multi-dimensional) values of some functional value (e.g., time). The interested reader is directed to Rasmussen and Williams (2006) for a complete description of GP. However, in relation to the modelling of a motion pattern, creating a statistical analysis of the dispersion of trajectories from a common path, it is important to understand that the common path relates directly to the mean function $\mathbf{m}(t)$ and the dispersion of the trajectories from a common path is captured in the covariance kernel $\mathbf{k}(t, t')$.

Gaussian processes have been used to discover groups in data (i.e., clustering) by applying the Gaussian mixture model (Gaffney and Smyth 1999). Each cluster is modelled independently as a regression model with a polynomial basis, after which the mean function $\mathbf{m}(t)$ and the linear covariance kernel $\mathbf{k}(t, t')$ is used to assign the membership of each individual trajectory found in the data in a probabilistic manner. However, while the trajectory data are (or can be) multi-dimensional, each dimension is modelled with an independent GPR model. Though this assumption is not an issue when clustering, it is important to capture the relation between dimensions when modelling the dispersion or generating new trajectories, as discussed in detail in chapter 3.

In Tay and Laugier (2008), the authors model trajectories as a Gaussian process and clustered based on the Gaussian mixture model. They point out that the framework provided by GP naturally lends itself to the analysis of trajectories. However, the trajectories used for training the model are re-sampled, so the information (data-points and smoothness) is lost; The authors of Gaffney and Smyth (1999) sidestep this by introducing basis functions. Furthermore, the covariance kernel $\mathbf{k}(t, t')$ is expressed as the squared exponential kernel. It is important to note that this kernel is *stationary*, meaning that the value only depends on the difference $t - t'$. This implies that a change is only dependent on the *time difference*. For example, the covariance kernel $\mathbf{k}(t, t')$ can capture the behaviour where trajectories that are on the outside of a turn at the start (time $t$), are more likely to also be at the outside of the turn once the manoeuvre has been completed (time $t'$). The inverse can be true for the next turn – this behaviour can only be captured using a *non-stationary* kernel, e.g., the linear covariance kernel. Additionally, the envelope containing 95% of the trajectories is only represented as a 'bar' (a minimum and maximum value) in both dimensions separately. While for a single dimension the envelope is indeed described by a minimum and maximum value, for two dimensions the envelope should be represented by a merger of ellipses, analogous to Hu et al. (2006) and Morris and Trivedi (2011).

Other work applies GPR to learn the underlying model structure for further analysis (Cox, Kachergis and Shiffrin 2012). While they do mention trajectory analysis, the primary focus is on calculating the model parameters using GPR and relating these values to phenomena seen in cognitive psychology. Moreover, GPR has been applied to model the relative motion against the current position (Ellis, Sommerlade and Reid 2009). While this can identify general trends, and even be used for clustering (Ferreira et al. 2013), information of the spatial distribution is not recoverable.

Finally, aircraft trajectories have been modelled via Gaussian processes with the purpose of conformance monitoring (Yan et al. 2017). Here the Gaussian processes trained by the data are used as an acceptable range of state values. Such an approach allows for automated conformance monitoring within a changing environment, reducing the workload of the air traffic controllers. While the model is capable of capturing three-dimensional trajectories, the visualisations are limited to two dimensions.

## 2.5 Conclusion

A review of existing scientific literature shows that the available work on probabilistic modelling of trajectories is limited to the discrete domain. The application of Gaussian processes to learn a trajectory pattern is incomplete, or has a specific focus on clustering. Furthermore, methods generating the flight corridor depend on ad hoc methods. Applications for modelling motion patterns of aircraft trajectories exist in conformance monitoring, visualisation techniques, quantifying the air traffic complexity, and constructing flight corridors.

The next chapter, chapter 3, focusses on the probabilistic modelling of trajectory data as discussed in this chapter (chapter 2). Following this, the proposed approach from chapter 3 is applied to radar measured aircraft trajectory data in chapter 4.

# Chapter 3

# A proposed probabilistic modelling approach

Chapter 2 provided a literature review indicating that several practical engineering applications would benefit from a systematic probabilistic modelling approach of trajectory data. Specifically in aerospace such an approach can assist in visualising trends, reducing the computational cost of expensive calculations, and offers practical benefits such as planning and optimisation of future infrastructure. The work presented in this chapter is inspired by this need; a probabilistic modelling approach is proposed, and applied to a trajectory dataset of which the spatial distribution is known to be Gaussian distributed as a benchmark. The type of trajectories suitable for analysis follow a specific motion pattern, where available methods to identify motion patterns in the dataset have been discussed in section 2.2.

This chapter starts with specifying the expected structure of the input data in section 3.1. Following this, the proposed approach is introduced in section 3.2. This section starts with defining the task at hand, provides a representation of the probabilistic model, and states how to deal with noisy and missing data. Next, section 3.3 provides a method for selecting the optimal model and a technique for evaluating the model using an evaluation dataset. The proposed approach is then applied to a trajectory dataset of which the spatial distribution is known to be Gaussian distributed as a benchmark in section 3.4.

Following that is section 3.5, where several applications for the probabilistic model are introduced. One method generates corridors capturing a given percentage of trajectories, and another generates complexity maps, identifying potential conflict areas based on the structure of the motion patterns. Furthermore, an approach for extracting weighted trajectories from the probabilistic model is discussed, these can be directly implemented in existing systems without converting to a probabilistic framework. In section 3.6 an open-source Python package dubbed 'Teetool' is introduced, with the main purpose of making the probabilistic modelling approach accessible and improve its reusability. Finally, section 3.7 concludes with some final remarks, summarising the work presented in this chapter.

## 3.1 Structure of the input data

This section specifies the structure of the input data. It starts in section 3.1.1 by specifying the format of the matrices in which the trajectory data is expected to be in. Following that is section 3.1.2, introducing the concept of time-warping and how it is implemented in this work.

### 3.1.1 Input trajectory data

Given the dataset $\mathbf{Y} = \{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$ of $N$ trajectories, each $D$-dimensional trajectory $\mathbf{v}_n$ has $D$ concatenated vectors $\mathbf{g}_d$, resulting in the column vector $\mathbf{v}_d = [\mathbf{g}_1 \ldots \mathbf{g}_D]^\intercal$. Furthermore, all $\mathbf{g}_d$ in $\mathbf{v}_n$ have $M_n$ data points, resulting in $\mathbf{g}_d = [g_d(t_1) \ldots g_d(t_{M_n})]^\intercal$. Where the $M_n \times 1$ vector $\mathbf{t}_n = [t_1 \ldots t_{M_n}]^\intercal$ is identical for all $\mathbf{g}_d$. Therefore, $\mathbf{y}_n$ is an $DM_n \times 1$ vector, where no assumption is made on the number of data points per trajectory. As a result, each trajectory is described by a set of coordinates $g_1 \ldots g_D$ in $D$-dimensional Euclidean space, at each time-step $t_1 \ldots t_{M_n}$. Finally, in order to allow a sensible comparison between the individual dimensions $\mathbf{g}_d$ within the model, the entire dataset $\mathbf{Y}$ is normalised per individual dimension, resulting in $\mathbf{g}_d = [0, 1]$.

### 3.1.2 Time-warping

As the interest lies in finding the *spatial* distribution of trajectories and we are not concerned with their variation in time, it is convenient to normalise each trajectory into the interval $\tau = [0, 1]$, where we use $\tau$ to indicate *normalised time*. This procedure scales the velocity component over the whole trajectory, letting each object spend an equal amount of time in the controlled area, and as a result, all trajectories start at $\tau = 0$ and end at $\tau = 1$. Another approach is applying the Dynamic Time Warping (DTW) technique that is used in the speech recognition field as a pattern detection algorithm (Berndt and Clifford 1994; Müller 2007). In this approach, the points in each trajectory are shifted (time-warped), such that if one object was moving faster than the other, or even if there are changes in velocity during the observation, the trajectories would be identical. The important difference here is that normalising works only when the objects have a constant velocity, while DTW also works when the objects have a variable velocity.

## 3.2 Modelling multi-dimensional trajectories

This section introduces an approach to analyse an ensemble of trajectories and model the mean trajectory and the dispersion from the mean trajectory via the Gaussian process framework. It should be noted that the linear regression model as implemented here is a special case of a Gaussian process, and therefore references to the general Gaussian process framework are included. The emphasis here lies on obtaining a method that includes the dependence between each dimension (e.g., easting, northing, and altitude) of the trajectories by capturing the co-variance. This allows for a generative model that achieves a higher performance when it comes to modelling the dispersion in comparison to an independent model. Section 3.2.1 starts by introducing the linear representation used for the linear regression model. Section 3.2.2 follows by introducing the maximum likelihood formulation, used to estimate the model parameters.

### 3.2.1 Linear representation

We capture each trajectory $\mathbf{g}_d$ as a linear combination of basis functions. Thus for a vector of discrete times $\boldsymbol{\tau}_d$ a single dimension $\mathbf{g}_d$ can be approximated using equation (3.1).

$$\mathbf{g}_d(\boldsymbol{\tau}_n) \approx \sum_{j=1}^{J} g_j \phi_j(\boldsymbol{\tau}_n) = \boldsymbol{\phi}_d(\boldsymbol{\tau}_n)\boldsymbol{\omega}_d \tag{3.1}$$

where $\boldsymbol{\phi}_d(\boldsymbol{\tau}_n) = [\phi_1(\boldsymbol{\tau}_n) \ldots \phi_J(\boldsymbol{\tau}_n)]^\intercal$ and $\boldsymbol{\omega}_d$ is a $J \times 1$ vector taking the values that minimize the square error between a single dimension $\mathbf{g}_d$ and the right hand size of equation (3.1).

However, as stated in section 3.1.1, each trajectory $\mathbf{v}_n$ has $D$ individual dimensions, each described by $\mathbf{g}_d$. Therefore, the trajectory $\mathbf{v}_n$ is approximated using equation (3.2).

$$\mathbf{v}_n \approx \boldsymbol{\Phi}_n(\boldsymbol{\tau}_n)\boldsymbol{\omega}_n \tag{3.2}$$

where $\boldsymbol{\omega}_n$ is a $DJ \times 1$ vector and $\boldsymbol{\Phi}_n$ is a block-diagonal $DM_n \times DJ$ matrix:

$$\boldsymbol{\Phi}_n = \begin{pmatrix} \boldsymbol{\phi}_1(\boldsymbol{\tau}_n) & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{\phi}_D(\boldsymbol{\tau}_n) \end{pmatrix} \tag{3.3}$$

where the dependency of $\boldsymbol{\Phi}_n$ on $\boldsymbol{\tau}_n$ is not explicitly shown to keep the notation uncluttered. In this thesis, identical basis functions are used for all dimensions ($\boldsymbol{\phi}_i = \boldsymbol{\phi}_j, \forall\ i, j \in D$), however, if desired, it is possible to define these individually per dimension $d$.

### 3.2.2 Maximum likelihood formulation

In a dataset of $N$ trajectories the probability of each weights vector $\boldsymbol{\omega}_n$ is assumed to be Gaussian distributed with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

$$p(\boldsymbol{\omega}_n) = \mathcal{N}(\boldsymbol{\omega}_n | \boldsymbol{\mu}, \boldsymbol{\Sigma}), \forall\ n \in N \tag{3.4}$$

where $\boldsymbol{\mu}$ is a $DJ \times 1$ vector and $\boldsymbol{\Sigma}$ is a $DJ \times DJ$ matrix.

Each trajectory $\mathbf{v}_n$ is modeled as zero-mean Gaussian white noise added to the function described by equation (3.2), resulting in equation (3.5).

$$p(\mathbf{v}_n|\boldsymbol{\omega}_n, \beta) = \mathcal{N}(\mathbf{v}_n|\boldsymbol{\Phi}_n\boldsymbol{\omega}_n, \beta^{-1}\mathbf{I}_n), \forall n \in N \tag{3.5}$$

where $\beta^{-1}$ is the variance of the white noise and $\mathbf{I}_n$ is a $DM_n \times DM_n$ identity matrix.

From Bayes' theorem for Gaussian variables (Bishop 2006) the marginal distribution of $\mathbf{v}_n$ is given by

$$p(\mathbf{v}_n) = \mathcal{N}(\mathbf{v}_n|\boldsymbol{\Phi}_n\boldsymbol{\mu}, \boldsymbol{\Phi}_n\boldsymbol{\Sigma}\boldsymbol{\Phi}_n^\intercal + \beta^{-1}\mathbf{I}_n), \forall\ n \in N \tag{3.6}$$

and the conditional distribution of $\boldsymbol{\omega}_n$ given $\mathbf{v}_n$ is equal to

$$p(\boldsymbol{\omega}_n|\mathbf{v}_n) = \mathcal{N}(\boldsymbol{\omega}_n|\mathbf{C}_n(\beta\boldsymbol{\Phi}_n^\intercal\mathbf{v}_n + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}), \mathbf{C}_n), \forall\ n \in N \tag{3.7}$$

where

$$\mathbf{C}_n^{-1} = \boldsymbol{\Sigma}^{-1} + \beta\boldsymbol{\Phi}_n^\intercal\boldsymbol{\Phi}_n \tag{3.8}$$

The likelihood function $\mathcal{L}$, which represents the probability of the data given the parameters and viewed as a function of those parameters is given by

$$\mathcal{L} = \prod_{m=1}^{N} \{p(\mathbf{v}_n|\boldsymbol{\omega}_n, \beta)\ p(\boldsymbol{\omega}_n)\} \tag{3.9}$$

By maximizing $\mathcal{L}$ the values of the parameters for which the probability of the observed data is maximized can be determined. Equivalently we can minimize the negative log of $\mathcal{L}$, as this is more convenient both analytically and numerically. It can be shown by substituting equation (3.5) and equation (3.4) in equation (3.9), that the negative log of $\mathcal{L}$ is given by equation (3.10).

$$\begin{aligned}-\ln\mathcal{L} = \frac{DM^*}{2}\ln(2\pi) - \frac{DM^*}{2}\ln(\beta) + \\ \frac{\beta}{2}\sum_{m=1}^{N}\{\mathbf{v}_n^\intercal\mathbf{v}_n - 2\mathbf{v}_n^\intercal(\boldsymbol{\Phi}_n\boldsymbol{\omega}_n) + \mathrm{Tr}(\boldsymbol{\Phi}_n^\intercal\boldsymbol{\Phi}_n\boldsymbol{\omega}_n\boldsymbol{\omega}_n^\intercal)\} \\ + \frac{NJD}{2}\ln(2\pi) + \frac{N}{2}\ln(|\boldsymbol{\Sigma}|) + \\ \frac{1}{2}\sum_{m=1}^{N}\{\mathrm{Tr}(\boldsymbol{\Sigma}^{-1}(\boldsymbol{\omega}_n\boldsymbol{\omega}_n^\intercal - 2\boldsymbol{\omega}_n^\intercal\boldsymbol{\mu} + \boldsymbol{\mu}\boldsymbol{\mu}^\intercal))\}\end{aligned} \tag{3.10}$$

where

$$M^* = \sum_{m=1}^{N}\{M_n\} \tag{3.11}$$

Minimization with respect to $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and $\beta$ can be done numerically using the Expectation-Maximization (EM) algorithm. The expected log-marginal likelihood is given by equation (3.10) where the terms $\boldsymbol{\omega}_n$ and $\boldsymbol{\omega}_n\boldsymbol{\omega}_n^\intercal$ take their expected values.

From equation (3.7) the expected values are given by

$$\mathbb{E}[\boldsymbol{\omega}_n] = \mathbf{C}_n(\beta\boldsymbol{\Phi}^\intercal\mathbf{v}_n + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}) \tag{3.12}$$

$$\mathbb{E}[\boldsymbol{\omega}_n\boldsymbol{\omega}_n^\intercal] = \mathbf{C}_n + \mathbb{E}[\boldsymbol{\omega}_n]\mathbb{E}[\boldsymbol{\omega}_n^\intercal] \tag{3.13}$$

Minimizing the expected negative log likelihood with respect to $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and $\beta$ respectively give the expressions for $\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{\Sigma}}$ and $\hat{\beta}$ as seen in equations (3.14) to (3.16).

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{m=1}^{N}\{\mathbb{E}[\boldsymbol{\omega}_n]\} \tag{3.14}$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{m=1}^{N}\{\mathbb{E}[\boldsymbol{\omega}_n\boldsymbol{\omega}_n^\intercal] - 2\mathbb{E}[\boldsymbol{\omega}_n^\intercal]\boldsymbol{\mu} + \boldsymbol{\mu}\boldsymbol{\mu}^\intercal\} \tag{3.15}$$

$$\frac{1}{\hat{\beta}} = \frac{1}{DM^*}\sum_{m=1}^{N}\{\mathbf{v}_n^\intercal\mathbf{v}_n - 2\mathbf{v}_n^\intercal(\boldsymbol{\Phi}\mathbb{E}[\boldsymbol{\omega}_n]) + \mathrm{Tr}(\boldsymbol{\Phi}^\intercal\boldsymbol{\Phi}\mathbb{E}[\boldsymbol{\omega}_n\boldsymbol{\omega}_n^\intercal])\} \tag{3.16}$$

Equations (3.12) to (3.16) can be solved iteratively for $\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{\Sigma}}$ and $\hat{\beta}$.

Once the likelihood given by equation (3.10) has converged (in this thesis a convergence limit of $10^{-1}$ is used), it is possible to find the probability distribution of $\mathbf{v}$ at any given $\tau$ using equation (3.6). The solution fits into a GP framework, where a function is defined as a Gaussian process by a mean function $\mathbf{m}(\tau)$ and a covariance kernel $\mathbf{k}(\tau, \tau')$:

$$\mathbf{v}(\tau) \sim \mathcal{GP}(\mathbf{m}(\tau), \mathbf{k}(\tau, \tau')) \tag{3.17}$$

where

$$\mathbf{m}(\tau) = \mathbb{E}[\mathbf{v}] = \boldsymbol{\Phi}(\tau)\boldsymbol{\mu} \tag{3.18}$$

$$\mathbf{k}(\tau, \tau') = \mathbb{E}[(\mathbf{v}(\tau) - \mathbf{m}(\tau))(\mathbf{v}(\tau') - \mathbf{m}(\tau'))] = \boldsymbol{\Phi}(\tau)\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\intercal}(\tau') + \beta^{-1}\mathbf{I} \tag{3.19}$$

A more intuitive representation of the mean function $\mathbf{m}(\tau)$, given by equation (3.18), is a point that moves along a $D$-dimensional trajectory as a function of $\tau$. In turn, the covariance kernel $\mathbf{k}(\tau, \tau')$, given by equation (3.19), can be represented as a $D$-dimensional (hyper-)volume at a constant probability.

When sampling from equation (3.6) it can be convenient to ignore the white noise term ($\beta^{-1}\mathbf{I}_n$). This term is there to capture the uncertainty remaining between the parametrised model and the data. For this reason, when taking samples from a probability distribution, it is done according to equation (3.20).

$$p(\mathbf{v}_n) = \mathcal{N}(\mathbf{v}_n | \boldsymbol{\Phi}_n\boldsymbol{\mu}, \boldsymbol{\Phi}_n\boldsymbol{\Sigma}\boldsymbol{\Phi}_n^{\intercal}), \forall\, n \in N \tag{3.20}$$

## 3.3 Evaluating the model performance

This section evaluates the model performance. First, in section 3.3.1 the most likely model is selected based on the marginal likelihood. Various models exist, which for a linear regression model relates to the number and shape of the basis functions. The trajectory data used here is the training dataset. Finally, a method to evaluate the modelling performance against another dataset is introduced in section 3.3.2. The trajectory data used here is the evaluation dataset and has not been used in the algorithm.

### 3.3.1 Model selection

The likelihood function $\mathcal{L}$ being maximized is also known as the marginal likelihood, as the model parameters $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and $\beta$ have been marginalised out. This marginal likelihood is a useful quantity for comparing models, as Bayes factors are defined as ratios of marginal likelihoods of two competing models. For this reason, the model with the highest likelihood is the best model - if, and only if, the data are Gaussian distributed. This approximation is referred to as type II maximum likelihood (Rasmussen and Williams 2006).

Hence, by evaluating the likelihood at different values of the model complexity $J$, the optimal complexity can be calculated. Note that in this thesis only uniformly distributed Gaussian basis functions (as seen in appendix B) are considered, however it is possible to compare any model structure (e.g., linear, exponential, polynomial, or even Bézier curves (Faraway, Reed and Wang 2007)).

The uniformly distributed Gaussian basis functions the model complexity $J$ has an upper limit for which the likelihood function $\mathcal{L}$ (equation (3.10)) can be evaluated, this is a direct result from the term $\ln(|\boldsymbol{\Sigma}|)$. As one of the eigenvalues of $\boldsymbol{\Sigma}$ turns zero, the log-term goes to infinity, indicating redundancy in the model. In other words, there is a value in $\boldsymbol{\omega}_n$ that has no (or a very small) contribution to the final model. When this happens, it becomes impossible to evaluate the likelihood unless resorting to computational tricks such as assuming a minimum value for the eigenvalues of $\boldsymbol{\Sigma}$. For the model selection in this thesis, the model is trained up to the point where the redundancy is introduced. This allows for evaluation of $\mathcal{L}$ without introducing tricks, and thus permitting a convergence check of the EM algorithm.

### 3.3.2 Evaluation of the model using the Kolmogorov-Smirnov statistic

In a typical experiment, data collected in one situation is compared to the data in another situation with the aim of seeing if the results differ. The Kolmogorov-Smirnov (K-S) statistic does exactly this for probability distributions and quantifies the difference by returning the maximum difference between the cumulative distribution function.

Here the probability distribution $p(y)$ returns the odds of event $y$ occurring, where the cumulative distribution function is the integral of a range of $y$, and returns the expected percentage of samples drawn from $p(y)$ to fall within this range. The range of interest for this test is the area around the mean, bounded by the standard deviation $\sigma$, making it possible for a $D$-dimensional multivariate Gaussian to capture the area ($D = 2$), volume ($D = 3$) or even hyper-volume ($D > 3$) at any given $\sigma$.

For the application to trajectories, $p(\mathbf{v})$ is numerically integrated over the interval $\tau = [0, 1]$, of which can be determined whether the points of the trajectories are inside ($< \sigma$) or outside ($> \sigma$) the bounded range. In this thesis the trajectory data used in the experiments are compartmentalized in a *training* and an *evaluation* dataset. While the *training* trajectory data are used to create a probabilistic model $p(\mathbf{v})$ according to the method described in this section, the *evaluation* trajectory data are kept apart, only to be used for this test. The Empirical Cumulative Distribution Function (ECDF) is therefore created by increasing the bound $\sigma$ and calculating the percentage of points from the *evaluation* trajectory data that are within the range. For this reason the ECDF represents the true distribution as observed without any modelling. In case of the univariate Gaussian probability distribution, there is an analytical solution to evaluate the Cumulative Distribution Function (CDF) belonging to the model; such a formulation is not available for the multivariate Gaussian probability distribution. However, Justel, Peña and Zamar (1997) showed that this function can be approximated using a large number of generated samples. Hence, 2000 sample trajectories are drawn from $p(\mathbf{v}_n)$ (seen in equation (3.17)) over $\tau = [0, 1]$ in 100 uniform steps to represent the CDF. These values for sampling $p(\mathbf{v})$ and integrating over $\tau$ have been selected to ensure a 'good enough' approximation when performing the test, meaning that re-running the test 5 times does not cause a difference larger than 0.5%. After all, when sampling there is always a component of randomness included in the system.

To reduce the computational cost of this test, the method presented in Taylor and McAssey (2013) is implemented. This paper describes an approach to obtain similar results at a lower computational complexity by making use of the Mahalanobis distance to see whether a point falls within a given bound.

In simple terms, we create an area/volume/hyper-volume at a standard deviation $\sigma$, which expands as $\sigma$ increases. Here, the percentage of points of the three-dimensional trajectories (not complete trajectories) inside, is counted for both the *evaluation* trajectory data (representing the ECDF) and the generated samples (approximating the CDF belonging to the model). Of these functions, the maximum difference represents the Kolmogorov-Smirnov (K-S) statistic.

Finally, it is important to note that the K-S statistic remains an approximation, where both the sampling to create the CDF, and the numerical integration play a part. This cannot be avoided as both sampling from a probabilistic distribution and integrating numerically are an approximation by nature.

## 3.4 Benchmark: Gaussian distributed two-dimensional trajectories

This section demonstrates the probabilistic modelling method as applied to Gaussian distributed two-dimensional trajectories, and serves as a benchmark test. This dataset contains two-dimensional 'toy' trajectories that follow a circular path where the radius is sampled from a Gaussian distribution. The aim here is to determine a baseline performance and investigate two important concepts, the modelling of dependence between dimensions and the time-warping method.

The data are generated according to the following equations, where the radius $r$ is selected using:

$$r = \mathcal{N}(r|1, \frac{1}{2}) \tag{3.21}$$

and the $M$ data-points are generated using this array of angle $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = \{0, \frac{\pi}{2M}, \ldots, \frac{\pi}{2}\} \tag{3.22}$$

These values are converted to the Cartesian coordinate system by these relations:

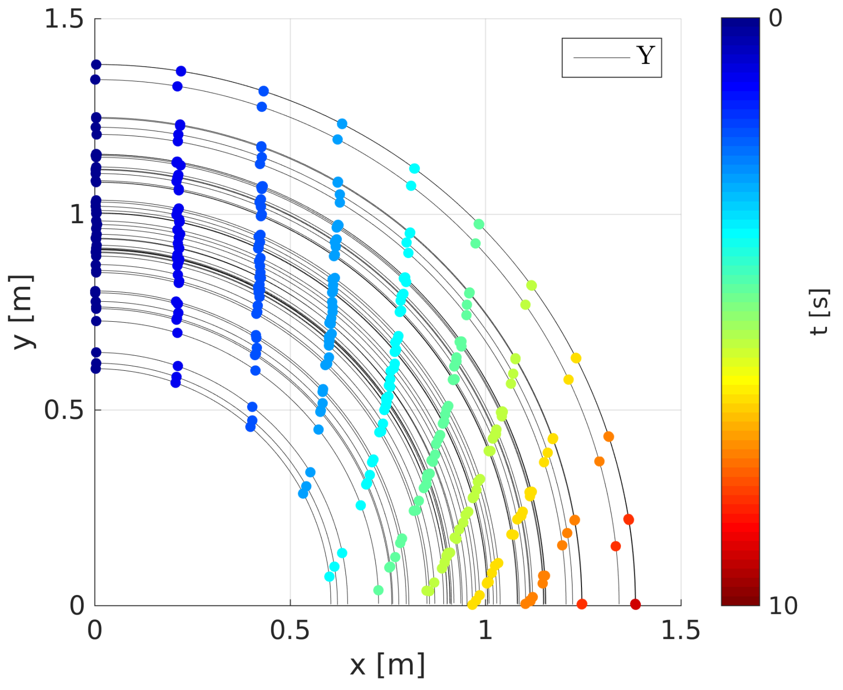
$$\mathbf{x} = r\cos(\boldsymbol{\theta}) \tag{3.23}$$

$$\mathbf{y} = r\sin(\boldsymbol{\theta}) \tag{3.24}$$

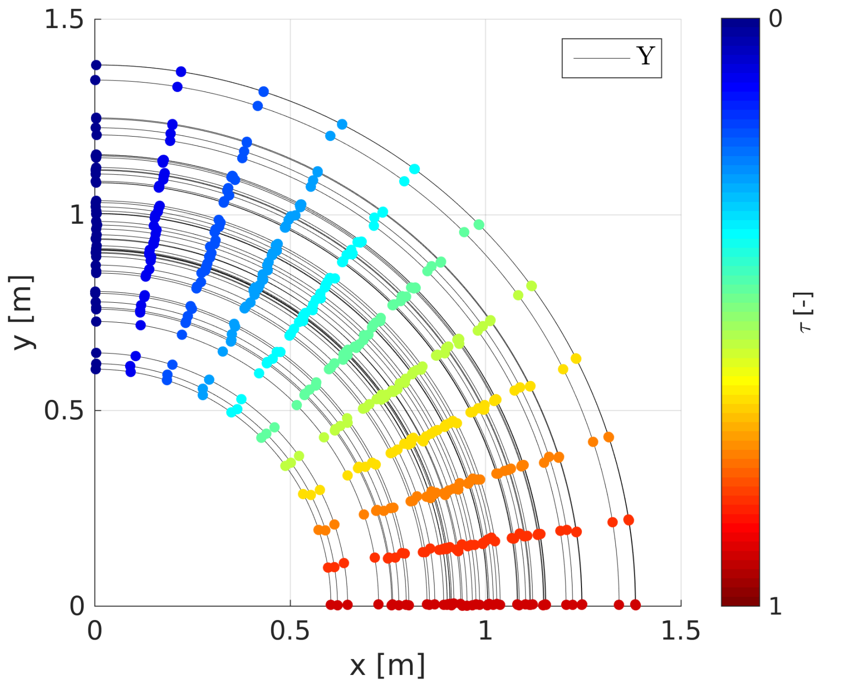
As for the time-warping, these results are visible in figure 3.1. In figure 3.1a, 50 trajectories at a constant velocity are visible, where the dots indicate the points between the start $t = 0s$ and the end time of the longest trajectory $t = 10s$ in 10 steps. In figure 3.1b, all trajectories have been normalised to interval $\tau = [0, 1]$, and the dots represents the start $\tau = 0$ and the end $\tau = 1$ in 10 steps. The trajectory data are expected to have a Gaussian distributed radius; the mean is expected to run though the middle. For the trajectories with a constant velocity this is not the case, as it represents more a wave-like effect, however, for the normalised trajectories, the average of each set of points does run through the middle. This indicates the method of normalising is indeed suitable for the purpose of time-warping the trajectories.

In this experiment, the model selection is done based on the likelihood as suggested in section 3.3.1. The maximum value for the marginal likelihood occurs at the model complexity $J = 21$, and is therefore selected to be the best model. The 50 trajectories used as training data are seen in figure 3.2a. Furthermore, the mean trajectory $\mathbf{m}(\tau)$ is visualised as a dashed line and the region captured by one standard deviation ($1\ \sigma$) is shown by a shaded area. From the probabilistic model $p(\mathbf{v})$, 50 sample trajectories are generated using equation (3.20). These sampled trajectories $\mathbf{S}$ are shown in figure 3.2b.

One of the key points of this approach is that it models the dependence between the dimensions by including the co-variance, and how it allows the probabilistic model to generate trajectory data (sampled data) similar to the input trajectory data (training data). An additional claim is that it also provides an improvement when it comes to modelling the dispersion. To simulate what will happen when the covariance between dimensions is not taken into account, only the off-diagonal blocks $\boldsymbol{\Sigma}_{1,2}$ and $\boldsymbol{\Sigma}_{2,1}$ in the

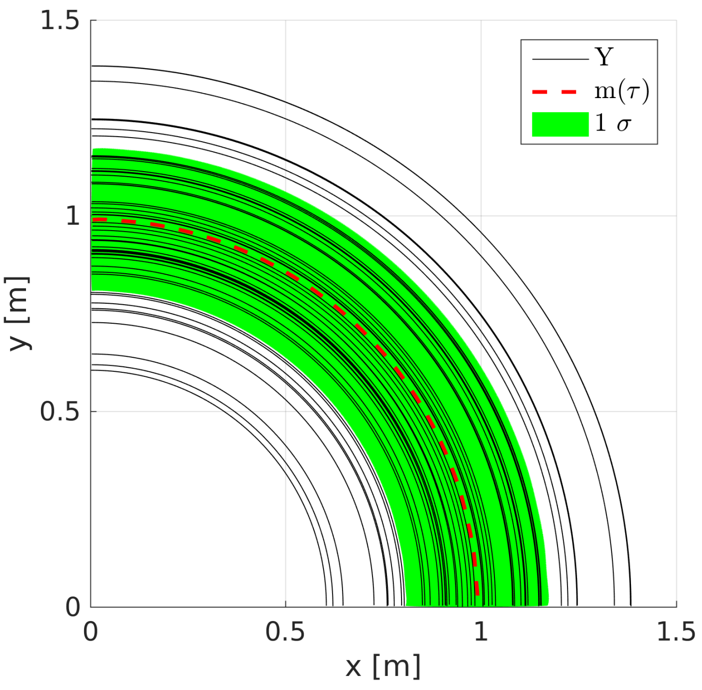


(a) un-normalised time, $t = \{0s, 1s, ...10s\}$

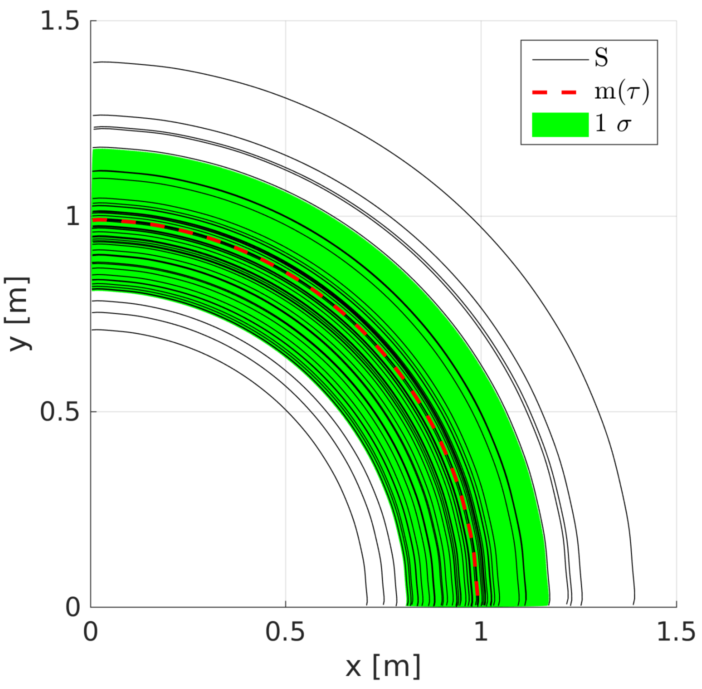


(b) normalised time, $\tau = \{0, 0.1, .., 1\}$

Figure 3.1: Examining the effect of normalising time.



(a) training data **Y**



(b) sampled data **S**

Figure 3.2: Visualisation of Gaussian distributed training and sampled trajectory data.

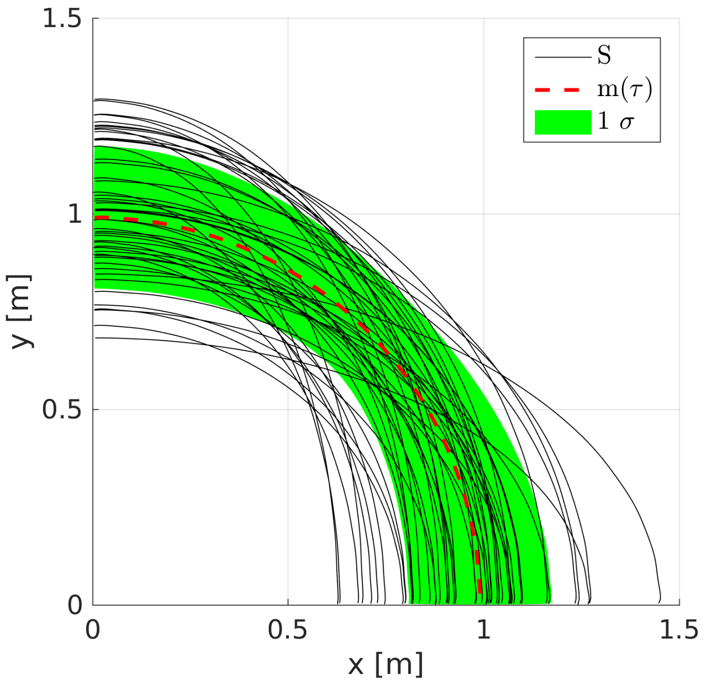


Figure 3.3: Visualisation of Gaussian distributed sampled trajectory data from an independent model.

co-variance matrix as seen in equation (3.25) need to be removed.

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{1,1} & \boldsymbol{\Sigma}_{1,2} \\ \boldsymbol{\Sigma}_{2,1} & \boldsymbol{\Sigma}_{2,2} \end{pmatrix} \tag{3.25}$$

The sampled trajectory data, generated from $p(\mathbf{v})$ using this reduced covariance matrix is visible in figure 3.3. Besides the 50 sampled trajectory data, the mean trajectory ($\mathbf{m}(\tau)$) is visualised as a dashed line and the region captured by one standard deviation ($1\ \sigma$) is shown by a shaded area. It shows that the sampled trajectory data overlap, a trend not seen in the training data. Furthermore, the area of one standard deviation is narrowed halfway through the corner (around $\tau = .5$), an effect supported by the sample trajectories as none move along the outer edges.

To conclude, the performance of the model is evaluated according to the technique discussed in section 3.3.2. For this experiment, the ECDF is created using 2000 trajectories from the *evaluation* dataset. Figure 3.4 displays both the ECDF (evaluation trajectory data) and the CDF (sampled trajectory data), showing that the K-S statistic is 0.049, corresponding to a maximum deviation of 4.9%. Furthermore, while the *training* and

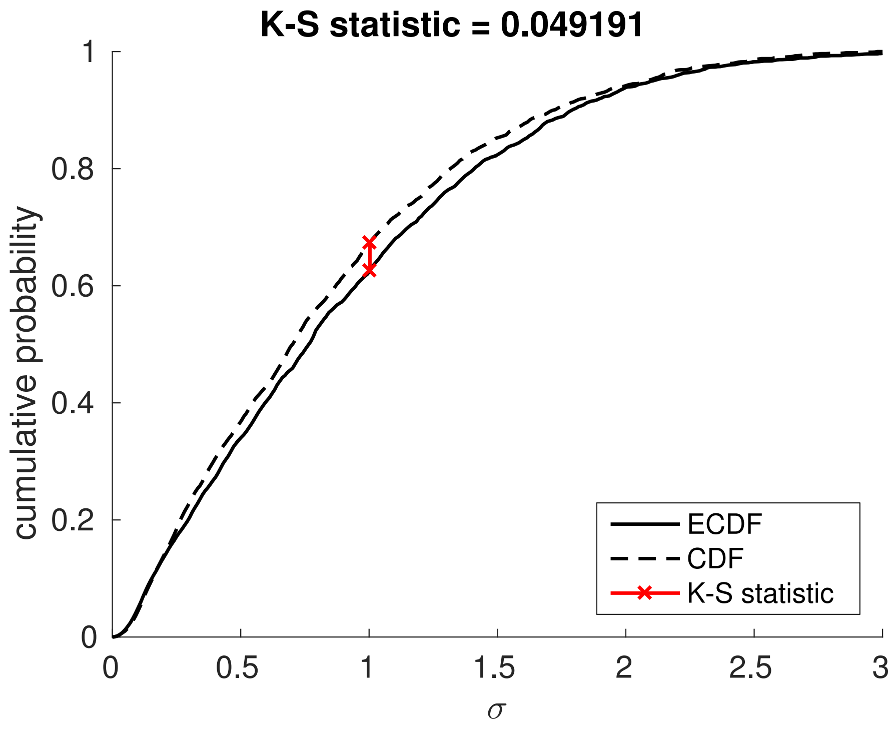


Figure 3.4: Comparing the Gaussian distributed evaluation and sampled trajectory data.

*evaluation* data were actually drawn from a Gaussian distribution, it is not surprising to see a number like this as only 50 trajectories were used as training data. However, our thesis here, is that it allows for the approximation of a probabilistic model using (limited) available data.

## 3.5 Applications of the modelling approach

This section introduces several applications for the probabilistic model. As the applications of the probabilistic modelling method presented in chapters 4 and 7, for aircraft and rocket flight trajectory data respectively, are in three dimensions, the examples provided here focus on three-dimensional trajectory data. Hence the multivariate Gaussian discussed are represented as ellipsoids, where for two-dimensional data this would be an ellipse. Section 3.5.1 starts by providing a method to generate corridors capable of capturing a given percentage of the trajectories. Section 3.5.2 introduces the generation of complexity maps that can be used to identifying potential conflict areas based on the trajectory structure. Finally, section 3.5.3 shows that the probabilistic model can also be used to generate weighted trajectories. Such an approach allows for an implementation in existing applications, without switching to a complete probabilistic framework.

### 3.5.1 Corridors

This section introduces an algorithm to generate corridors capable of capturing a given percentage of the trajectories. It starts from the probabilistic model described in section 3.2.1, introducing the conversion to a set of multivariate Gaussian distributions from a Gaussian process. Furthermore, a method to efficiently evaluate whether a point is inside a defined corridor is proposed.

Starting from the mean function $\mathbf{m}(\tau)$ and covariance kernel $\mathbf{k}(\tau, \tau')$ as shown in equations (3.18) and (3.19) repectively, the probabilistic model is converted from the continuous domain to a discrete number of instances evaluated over the domain $\tau = [0, 1]$. While it is beneficial for the model to exist in the continuous domain, allowing trajectories with any number of datapoints (including those with missing data) to be added, it is less favourable when generating corridors. The reason is that when considering the continuous domain there exist an infinite number of multivariate Gaussian distributions, one at each value of $\tau$. In this thesis, unless otherwise specified, 100 uniform steps over the domain $\tau = [0, 1]$ are taken for the conversion. The number of steps are related to the volatility of the trajectories, where the results for rocket- and aircraft trajectories as analysed in this thesis show that this number of steps is sufficient. In effect, the model is transformed from:

$$\mathcal{GP}(\mathbf{m}(\tau), \mathbf{k}(\tau, \tau')) \tag{3.26}$$

to a set of 100 multivariate normal distribution:

$$\mathcal{N}(\boldsymbol{\mu}(\tau), \boldsymbol{\Sigma}(\tau, \tau)) \tag{3.27}$$

evaluated at $\tau = [0, \frac{1}{100}, \frac{2}{100}, \ldots, 1]$, with $\tau' = \tau$ as the relation between the multivariate Gaussian distributions is only important when sampling trajectories from the model and not for the spatial analysis.

As stated before, at each time-step for a given standard deviation $\sigma$, a three-dimensional multivariate normal distribution can be expressed as an ellipsoid. In order to construct a corridor, for two sequential time-steps, and the corresponding two ellipsoids, the volume of these ellipsoids, and the volume between the two ellipsoids is assumed to be part of the corridor, i.e., the behaviour is assumed to be linear between the two time-steps.

Next, in order to evaluate whether a point is within the ellipsoids or the volume between the ellipsoids, the Quickhull algorithm (Barber, Dobkin and Huhdanpaa 1996) is applied. Essentially, the points belonging to the ellipsoids are taken as a point-cloud, thereafter the Quickhull algorithm efficiently converts this to a hull constructed by the outer points. Using the hull, it is possible to quickly evaluate which points lie inside the flight corridor, and which lie outside. The figures generated by the Teetool package (to be introduced in section 3.6) are produced by evaluating a grid of points, however there is no such restriction. For example, the same approach can evaluate whether a corridor conflicts with any restricted space.

### 3.5.2 Complexity maps

This section describes the method implemented to evaluate traffic complexity. This method works via the probabilistic models as described in section 3.2, hence the traffic complexity as discussed here only relates to the *traffic structure*.

The complexity measure used in the complexity map has been inspired by Prandini, Putta and Hu (2010), and allows for the identification of conflict regions. This approach allows the visualisation of different orders of complexity, where the first order complexity $\mathcal{C}^1$ represents the probability of at least one flight corridor intersecting with a selected point in space. The second order complexity $\mathcal{C}^2$ represents the probability of at least two flight corridors, and so forth. The mathematical description of the $k$-th order complexity $\mathcal{C}^k$ evaluated at point $x$ is calculated via equation (3.28):

$$\mathcal{C}^k(x) = \sum_{T}^{\binom{\mathbf{S}}{k}} \left( \prod_{n}^{T} \left( p(x|n) \right) \right) \tag{3.28}$$

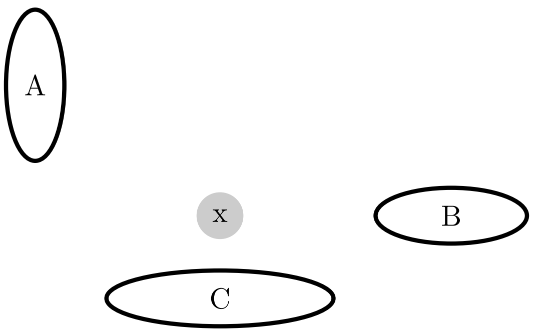


Figure 3.5: An example where point $x$ is surrounded by the corridors $A$, $B$, and $C$. In this example, the contours surrounding the corridors represent an area corresponding to a given standard deviation.

where $\binom{\mathbf{S}}{k}$ is a $k$-th combination of the set of all corridors $\mathbf{S}$, and $p(x|n)$ represents the probability of corridor $n$ intersecting with point $x$. Note that following the method from section 3.5.1, each corridor is described by 100 normal distributions $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Therefore, there are 100 probabilities corresponding to corridor $n$ intersecting with point $x$. However, as we are not concerned with their variation in time (each normal distribution represents a time-step), $p(x|n)$ is taken equal to the highest probability. This corresponds to the largest chance of point $x$ intersecting with corridor $n$ at any given time.

Finally, a demonstration of equation (3.28) is provided here. A schematic overview of the example is shown in figure 3.5, where point $x$ is surrounded by the set of corridors $\mathbf{S} = \{A, B, C\}$. Following equation (3.28), the corresponding complexity measures $\mathcal{C}^1$, $\mathcal{C}^2$, and $\mathcal{C}^3$ are then equal to equations (3.29) to (3.31):

$$\mathcal{C}^1(x) = p(x|A) + p(x|B) + p(x|C) \tag{3.29}$$

$$\mathcal{C}^2(x) = p(x|A) \cdot p(x|B) + p(x|A) \cdot p(x|C) + p(x|B) \cdot p(x|C) \tag{3.30}$$

$$\mathcal{C}^3(x) = p(x|A) \cdot p(x|B) \cdot p(x|C) \tag{3.31}$$

This example confirms that $\mathcal{C}^1$ is indeed the sum of probability of each individual corridor intersecting with point $x$, $\mathcal{C}^2$ the sum of probability for all combinations of two corridors intersecting with point $x$, and $\mathcal{C}^3$ the probability of all three corridors intersecting with point $x$.

### 3.5.3 Weighted trajectories

The previous section described how to estimate the probabilistic model. This section provides a method to convert this model to *weighted* trajectories that can approximate an ensemble of trajectories.

For a 3-dimensional vector $\mathbf{v}(\tau)$ as seen in equation (3.27), the multivariate Gaussian distribution (appendix A.2) takes the form:

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^3|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\Delta^2)\right) \tag{3.32}$$

where $\Delta$ represents the Mahalanobis distance:

$$\Delta^2 = (\mathbf{u} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{u} - \boldsymbol{\mu}) \tag{3.33}$$

At a constant Mahalanobis distance, this equation takes the form of an ellipsoid described by $|\boldsymbol{\Sigma}|^{-1/2}$, centred at $\boldsymbol{\mu}$. In this scenario the axes of the covariance ellipse are given by the eigenvectors $\mathbf{r}$ of the covariance matrix $\boldsymbol{\Sigma}$. The corresponding lengths, for an ellipse with unit Mahalanobis radius, are given by the square roots of the corresponding eigenvalues $\lambda$. Both can be found using the eigenvalue decomposition of the matrix $\boldsymbol{\Sigma}$.

$$\mathbf{R}\boldsymbol{\Lambda}\mathbf{R}^{-1} = \boldsymbol{\Sigma} \tag{3.34}$$

where

$$\mathbf{R} = [\mathbf{r}_1 \ \mathbf{r}_2 \ \mathbf{r}_3] \tag{3.35}$$

and

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \tag{3.36}$$

In short, the shape of the ellipsoid is described by the eigenvalues found in $\boldsymbol{\Lambda}$, where $\boldsymbol{\mu}$ and $\mathbf{R}$ are merely a translation and rotation respectively. The equation for the ellipsoid is given by:

$$\frac{g_1^2}{\lambda_1} + \frac{g_2^2}{\lambda_2} + \frac{g_3^2}{\lambda_3} = 1 \tag{3.37}$$

where $g_1$, $g_2$, and $g_3$ are the basis for the coordinate system $\mathcal{G}$; the coordinate system that is not yet translated and rotated by $\boldsymbol{\mu}$ and $\mathbf{R}$ respectively.

The plane perpendicular to the mean function at time $\tau$, can be described with a unit normal vector $\mathbf{n}$. While $\boldsymbol{\mu}(\tau)$ is evaluated at a discrete number of $\tau$, $\mathbf{m}(\tau)$ is continuous. Therefore the unit normal vector $\mathbf{n}$ can be both derived, or calculated numerically according to:

$$\mathbf{n} = \frac{\mathbf{m}(\tau + d\tau/2) - \mathbf{m}(\tau - d\tau/2)}{d\tau} \tag{3.38}$$

where $d\tau$ is an arbitrarily small number. The mathematics required to calculate the intersection between the plane and ellipsoid is given in Klein (2012). By doing so, the *two-dimensional* ellipse can be evaluated at any angle. However, it is important to note that the plane generated at time $\tau$ intersects multiple ellipsoids, it is therefore necessary to evaluate all those that intersect and store the point corresponding with the largest deviation from the centre point (provided by $\mathbf{m}(\tau)$).

To obtain a representative trajectory, the ellipse at any specific angle, which is a single point in a three-dimensional space, can be evaluated over $\tau = [0, 1]$ in any number of steps. As stated before, in this thesis by default 100 steps are used. Thus the combination of a constant Mahalanobis distance (representing a confidence interval, to be discussed next) and a given angle provides one representative trajectory. However, when it comes to selecting the angles to obtain a selection of representative trajectories, there are an infinite number of options. In this section two options are compared. In the handbook on generating noise contours (European Civil Aviation Conference 2016), only the dispersion in lateral direction is taken into account. Here the cross-section containing the artificial trajectories appears like figure 3.6, where the Gaussian distribution is included as a reference. In this figure, 5 trajectories are shown to capture a given percentage (the area under the curve). This corresponds with the CDF, which is equal to chi-square with 1 degree of freedom. The resulting weight per trajectory is shown in table 3.1. The area under the curve described by the Gaussian distribution corresponds with a confidence

Table 3.1: Weights generated using the chi-square distribution table for 1 degree of freedom.

| **range** | **total percentage captured** [%] | **percentage per trajectory** [%] |
|---|---|---|
| $[0.0^2, 0.5^2]$ | 38.29 | 38.29 *($N$ = 1)* |
| $[0.5^2, 1.5^2]$ | 48.35 | 24.17 *($N$ = 2)* |
| $[1.5^2, 2.5^2]$ | 12.12 | 6.06 *($N$ = 2)* |
| $[0.0^2, 2.5^2]$ | 98.76 | 98.76 *($N$ = 5)* |

Table 3.2: Weights generated using the chi-square distribution table for 2 degrees of freedom.

| **range** | **total percentage captured** [%] | **percentage per trajectory** [%] |
|---|---|---|
| $[0.0^2, 0.5^2]$ | 11.75 | 11.75 *($N$ = 1)* |
| $[0.5^2, 1.5^2]$ | 55.78 | 6.97 *($N$ = 8)* |
| $[1.5^2, 2.5^2]$ | 28.07 | 3.51 *($N$ = 8)* |
| $[0.0^2, 2.5^2]$ | 95.61 | 95.61 *($N$ = 17)* |

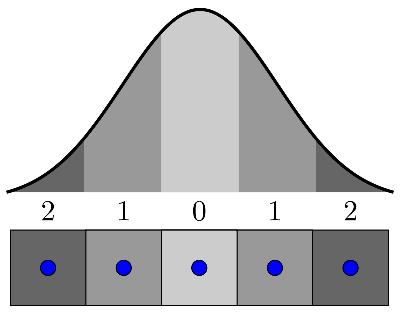


Figure 3.6: Cross-section of the representative trajectories (indicated by blue dots) when only taking the lateral dispersion into account, with a schematic representation of the Gaussian distribution included as a reference.

interval (thus capturing a certain percentage of the complete data) - and this percentage is divided over multiple trajectories due to symmetry. For example in the case of lateral dispersion only, in the range $[0.5^2, 1.5^2]$, the total percentage 48.35% is divided over two trajectories centred at a standard deviation of one ($N$ = 2), resulting in 24.17% per trajectory. When including the vertical dispersion, the model show more similarity to figure 3.7. As it now encompasses two dimensions, the chi-square with 2 degrees of freedom is used. Combined with angles at various ranges, there are 17 representative trajectories. The weight per trajectory is shown in table 3.2. In any approximation these percentages should be multiplied with the total traffic to obtain representative number of trajectories, in other words, the percentages shown here represent the weighted averages.

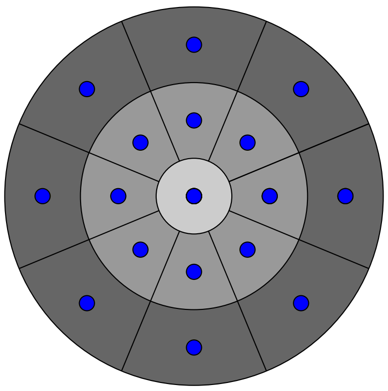


Figure 3.7: Cross-section of the representative trajectories (indicated by blue dots) when taking both the lateral and vertical dispersion into account.

## 3.6 Teetool – an open-source Python package

Teetool is a Python package developed by the author to learn the motion patterns found in trajectory data, and it does so by modelling the motion patterns found in trajectory data via Gaussian processes. This approach enables the recovery of missing data and the handling of noisy measurements as described in section 3.2. The Python package is freely available on GitHub (`https://github.com/WillemEerland/teetool`), where it is licensed under the MIT License. The package takes as input both two- and three-dimensional trajectory data as a function of another value (e.g., time or absolute distance covered). However, as an output the statistical modelling only concerns itself with the spatial characteristics and by default considers the complete range of this value. Furthermore, the package assists in visualising the patterns in two and three dimensions.

The calculations in the Teetool package are performed using NumPy (Walt, Colbert and Varoquaux 2011) and Scipy (Jones, Oliphant, Peterson et al. 2001). The data are visualised in two and three dimensions using Matplotlib (Hunter 2007) and Mayavi (Ramachandran and Varoquaux 2011) respectively. Data are handled via Pandas (McKinney 2010), and examples are included using Jupyter Notebooks (Kluyver et al. 2016).

The remainder of this section covers the implementation and architecture, quality control, and reuse potential of the package.

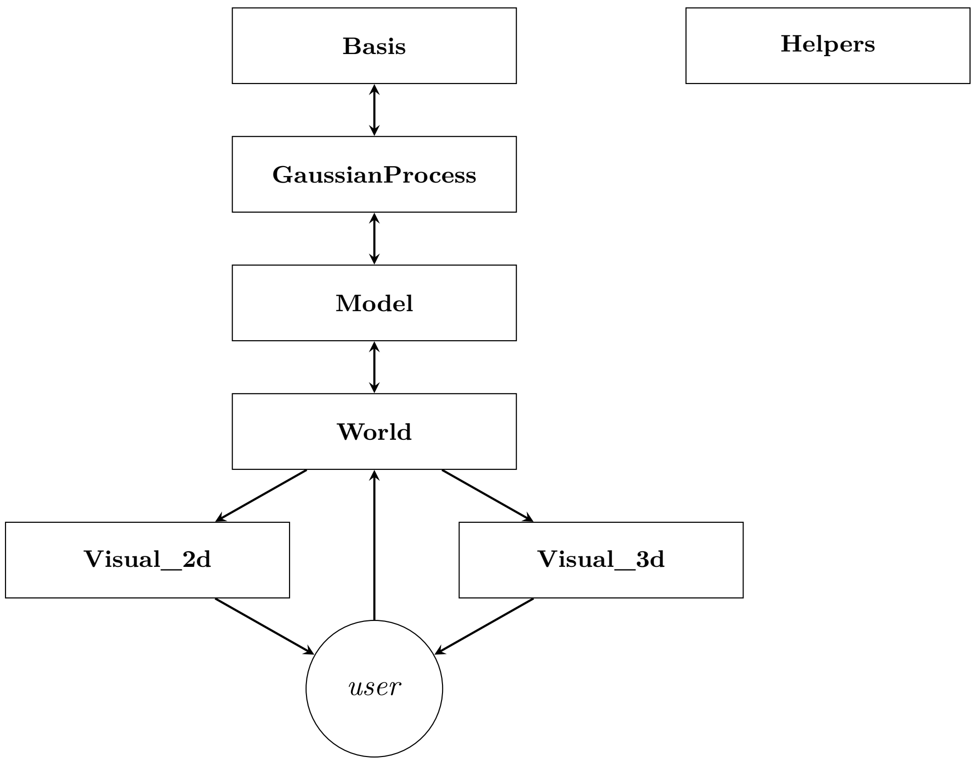


Figure 3.8: A schematic representation of the seven classes found within Teetool, and the interaction with the user.

### 3.6.1 Implementation and architecture

The package is written in Python and consists of seven classes. An overview how these classes communicate with the user is available in figure 3.8. The user starts by initialising a **World** class and adding trajectory data – this class handles multiple sets of trajectory data. From each set, a model is created with parameters as specified by the user. The specifics of these parameters are described in the documentation and include a selection of the modelling method. Based on these parameters the **GaussianProcess** and **Basis** class are initialised and called with the correct settings. Note that from a user's perspective the classes **Model**, **GaussianProcess**, and **Basis** are not visible, as these are called via the **World** class, hence there is no direct line going towards these classes starting from the user. The visualisations are presented via the **Visual_2d** and **Visual_3d** classes, for two- and three-dimensional representations respectively. A sequence diagram of the initialization of a **World** object and the modelling via resampling is displayed in figure 3.9.

An example using two-dimensional trajectory is visible in figure 3.10. It shows the trajectory data (figure 3.10a) and the confidence region corresponding to two standard

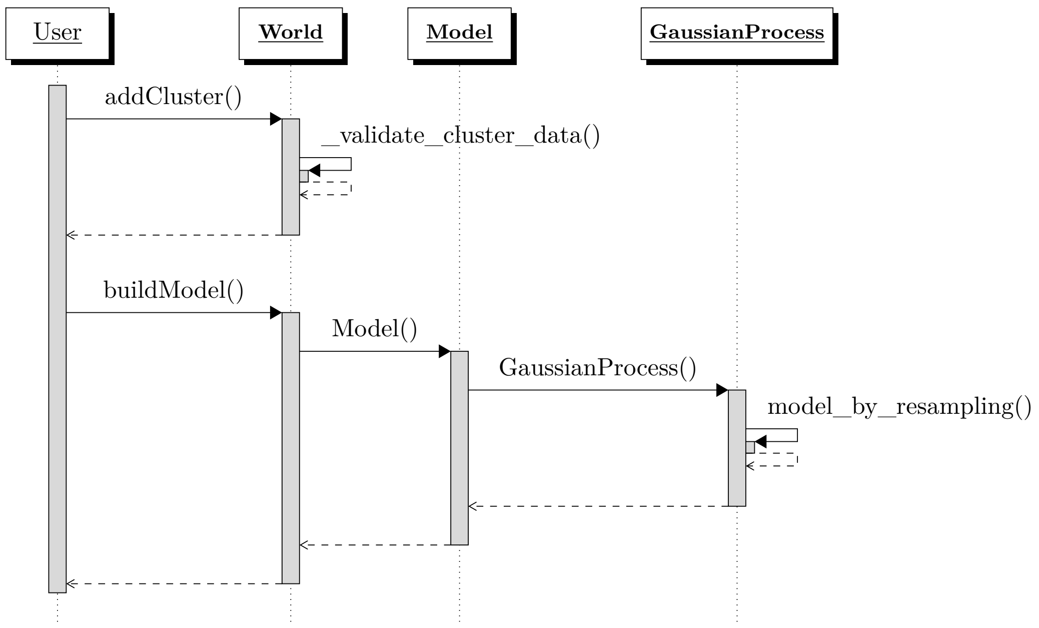
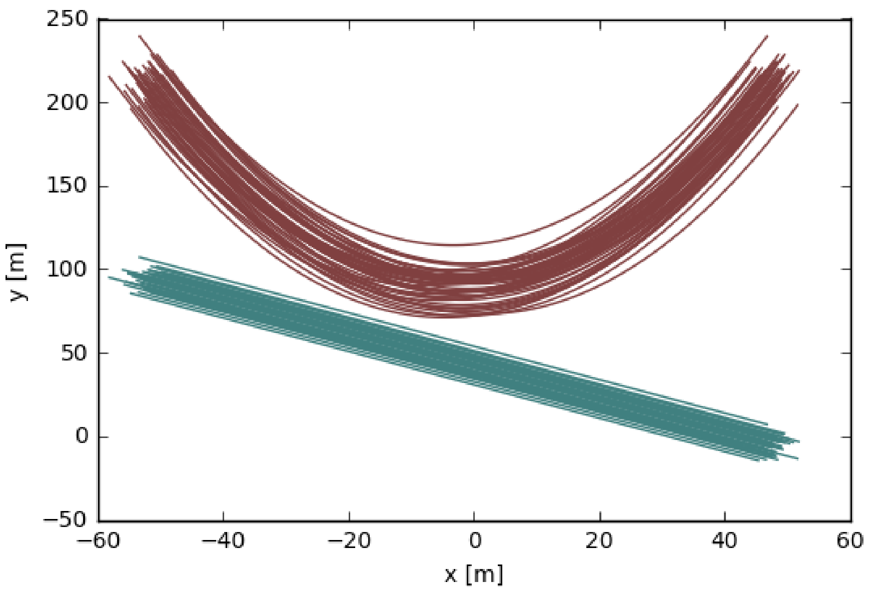


Figure 3.9: A sequence diagram visualising the chain of events following the initialization of a **World** object and the modelling via resampling.

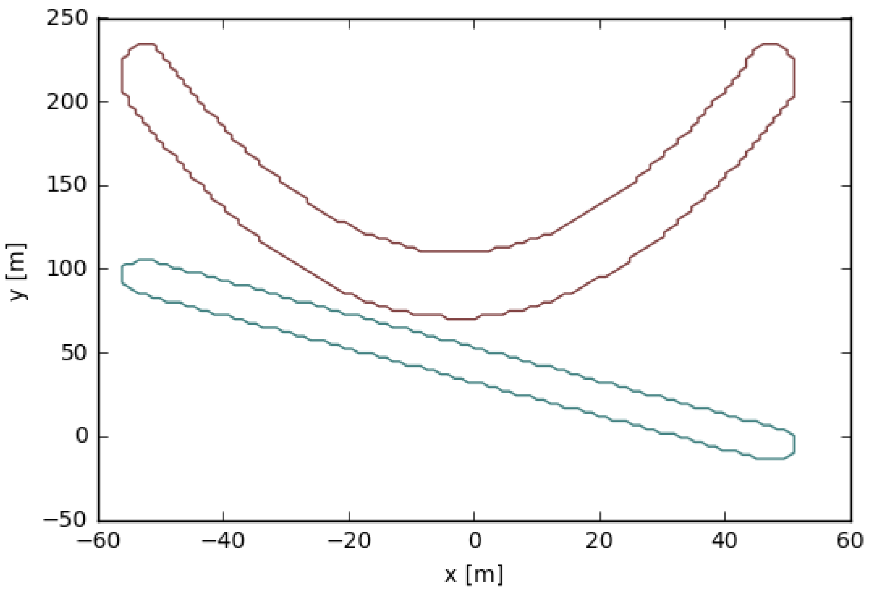
deviations (figure 3.10b). The trajectory data in this example is artificially generated using random sampling from a Gaussian distribution, and aims to demonstrate the confidence region in two dimensions. These figures are generated using the Jupyter notebooks included in the repository.

### 3.6.2 Quality control

Unit tests with an overall coverage of 87% are available and executed using `pytest`. Instructions on how to run these tests are included in the README.MD file. Continuous integration has been implemented using Travis CI, and the latest test results are available online at `https://travis-ci.org/WillemEerland/teetool`. The Jupyter notebook used to produce figure 3.10 can be found in the EXAMPLE folder, and provides a demonstration of basic functionality to the user. A code snippet of EXAMPLE/EXAMPLE_TOY_2D.IPYNB is included in figure 3.11 where **cluster_data** is a list of **(x, Y)**. Here **Y** is a $[N \times D]$ matrix, representing the multi-dimensional coordinates as a function of the $[N \times 1]$ vector **x**. These are NumPy arrays where $N$ is the total number of data-points and $D$ is the dimensionality of the trajectory data.



(a) trajectory data



(b) confidence region corresponding to two standard deviations

Figure 3.10: Gaussian distributed artificially generated trajectory data (a), aimed at demonstrating the functionality of the confidence region (b).

```
# import package
import teetool as tt
# create a world
world = tt.World(ndim=2, resolution=[100, 100])
# add data
world.addCluster(cluster_data=cluster_data_1)
```

Figure 3.11: a code snippet of the Jupyter notebook.

### 3.6.3 Reuse potential

The package is capable of modelling (i.e., create a statistical model of where they may be located in space) and visualising the motion trend of any kind of trajectory data. It has already been applied to model the motion patterns of three-dimensional trajectory data in two applications. The first application is to visualise the flight corridors as found in trajectory data as measured by a ground-based radar. In the second application the package has been used to visualise the behaviour of the output trajectory data from a stochastic rocket simulator. Furthermore, examples have been included to analyse two-dimensional trajectories, which are often seen in the field of road traffic analysis.

The modelling methods are accessible via the **World** class, and when desired it is possible to access this class directly and request the raw output of the model. Documentation on how to address each of the classes is available at `https://willemeerland.github.io/teetool/`, and is automatically generated via `doxygen`. The package is open-source and available on GitHub (`https://github.com/WillemEerland/teetool`), which is the preferred channel for any potential contributors to contact the developers. Support is available and provided via GitHub Issues.

## 3.7 Conclusion

In this chapter a probabilistic modelling approach for the analysis of a multi-dimensional trajectory dataset was proposed. The main contributions of the work within this chapter are summerarised as follows. The approach is able to *learn* a probabilistic model from an ensemble of trajectories as found in a trajectory dataset. Furthermore, due to the implementation of the Gaussian process framework in the approach, noisy and missing data can be handled. Several applications for such an estimated probabilistic model has been introduced. Finally, a description of an open-source Python package 'Teetool' is included, aimed at making the probabilistic modelling approach accessible and improve the reusability.

# Chapter 4

# Probabilistic modelling of aircraft trajectories

This chapter aims to demonstrate the application of the method introduced in chapter 3 to aircraft trajectories. Various applications of probabilistic modelling of aircraft trajectories have been identified in chapter 2 and include gaining situational awareness, conformance monitoring, and producing complexity maps.

Section 4.1 starts with the validation of the method similar to an approach taken with the benchmark seen in section 3.4. Following this is section 4.2, where the performance of weighted trajectories are evaluated when performing calculations on the ground-level. The specific case study included in this section, takes a simplistic approach to calculate from which locations the aircraft (where the trajectories are modelled via the data) are vulnerable to an attack using Rocket-Propelled Grenades (RPGs), without taking into account any mapping nor specific weapon characteristics. Section 4.3 demonstrates the application of open-source Python package Teetool as a decision support tool for air traffic management. Efforts to develop a decision support tool in order to assist ATM have started recently and are currently focussed on data management (Vourous 2017). The focus here is on generating corridors, displaying the motion patterns of the air traffic, and producing complexity maps, quantifying conflict regions via a dataset.

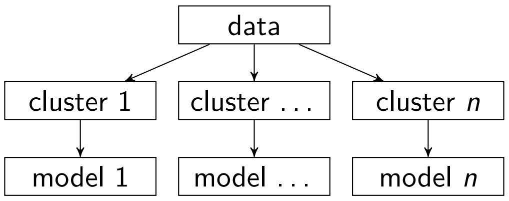


Figure 4.1: Procedure to handle trajectory data with multiple motion patterns.

Finally, section 4.4 concludes with some final remarks, summarising the work presented in this chapter.

## 4.1 On modelling aircraft trajectories

In this section the modelling approach as presented in chapter 3 is applied on three-dimensional flight trajectory data as measured by ground-based radar departing from Dallas Fort Worth airport (DFW) airport. The aim of this section is to evaluate the performance of the method on a aircraft trajectory data as measured by a surveillance radar, where the distribution over the trajectories may not even be truly Gaussian. First, the dataset is clustered according to the flight path following the procedure to handle multiple motion patterns as presented in figure 4.1. Secondly, the largest cluster is modelled according to the probabilistic approach, after which the performance is evaluated via the K-S statistic. Finally, a demonstration is included on how this approach can provide situational awareness around an airport via an overview of flight paths.

### 4.1.1 Clustering the dataset

The clustering step is essential as the method presented in chapter 3 is only valid for ensembles of trajectories that follow a similar motion pattern. The clustering technique implemented uses the approach and departure labelling, runway location and the clustering technique for aircraft trajectories as seen in Gariel, Srivastava and Feron (2011):

1. *approach* or *departure*.
2. assign nearest runway based on start location.

Table 4.1: An overview of the clustered trajectory data.

| **cluster** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | *total* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **size** | 82 | 73 | 60 | 56 | 53 | 51 | 46 | 35 | 33 | 32 | 25 | 20 | 20 | 566 |

3. re-sampling, Principal Component Analysis (PCA) (dimension reduction) and DBSCAN clustering.

While the first two steps reduce the generality of the clustering method, they allow for a clean separation of the trajectories. In the third step, the feature vector (a vector of set size containing numbers that describe each trajectory) contains 30 data-points spread uniformly over the total length of each individual trajectory. The dimension is then reduced to 5 using PCA. The resulting vector is used as input for the DBSCAN algorithm. The $\epsilon$-neighbourhood parameter $\epsilon$ of the DBSCAN algorithm is set to 0.20 and the minimum number of trajectories in each cluster is 20. This can be considered an aggressive clustering approach, resulting in one-fourth of the data being seen as noise (outliers). However, as a result, the remaining trajectories display common trends. An overview specifying the number of trajectories found in each cluster is available in table 4.1.

It should be noted, that when combining the proposed technique with clustering algorithms a trade-off arises between choosing a few large clusters, or many smaller clusters. Having larger clusters provides more data, resulting in a capturing more information with a single model. However, with the agglomeration of 2 or 3 clusters, the distribution will be more similar to a mixture of Gaussian distributions. As a result, the maximum deviation between the probabilistic model and the test data, the K-S statistic, is expected to increase.

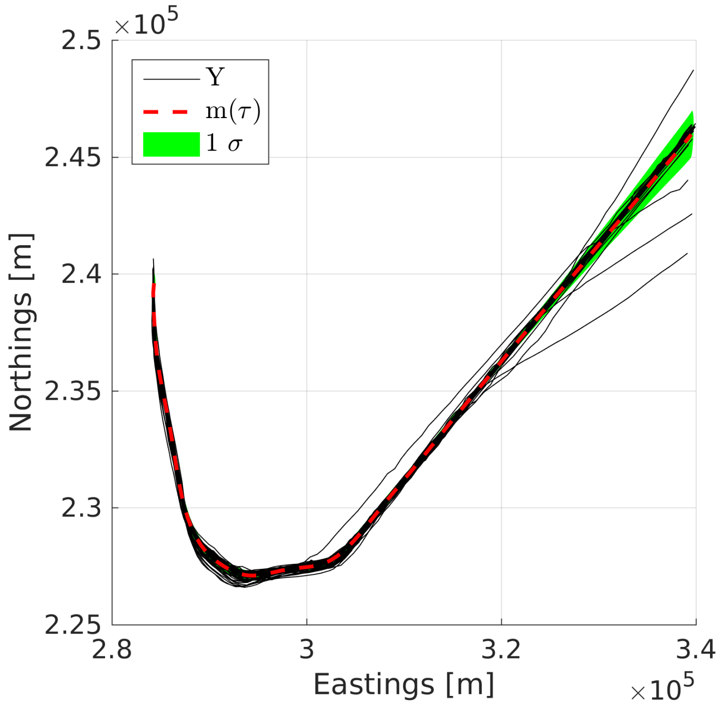
### 4.1.2 Modelling aircraft trajectories via Gaussian processes

From the clustered data, the largest cluster of 82 trajectories is analysed in this section. These trajectory data are collected within a single day and includes no specific information on the runway, aircraft type, intented flight path, and so forth. These 82 aircraft trajectories are compartmentalized, where 50 trajectories are used as training data $\mathbf{Y}$ to build the probabilistic model $p(\mathbf{y})$ according to the approach described in section 3.2. The model selection is done based on the likelihood as indicated in section 3.3.1. The maximum value for the marginal likelihood occurs at the model complexity $J = 22$, and is therefore selected to be the best model. The remaining 32 trajectories are kept apart as evaluation trajectory data, to be used in determining the K-S statistic later on in this section. These numbers are based on the general rule, where 50% is used for training, and 25% each for validation and testing (Hastie, Tibshirani and Friedman 2009). Here the training and validation data are combined into one dataset, as the likelihood method trains and selects the model using the same data.

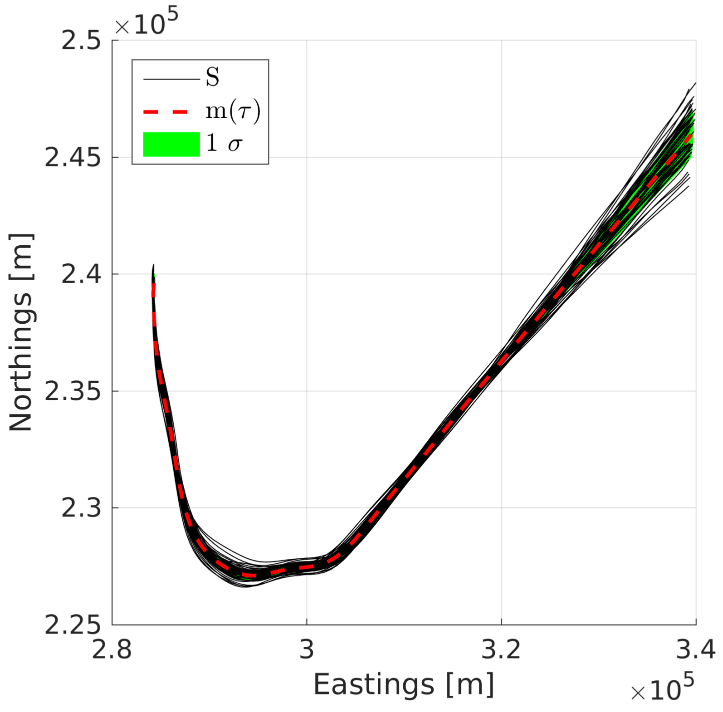
The 50 aircraft trajectories in the training data $\mathbf{Y}$ are visible in figures 4.2a and 4.3a for both the top- and side-view. Furthermore, the mean trajectory ($\mathbf{m}(\tau)$) is visualised as a dashed line and the region captured by one standard deviation ($1\ \sigma$) is shown by a shaded area. From the probabilistic model $p(\mathbf{y})$, 50 sample trajectories (described as $\mathbf{S}$) are generated using equation (3.20). These sampled trajectories $\mathbf{S}$ are shown in figures 4.2b and 4.3b. While the side-view shows similar trajectories, the top-view shows a greater symmetry in the sampled trajectory data $\mathbf{S}$. The training data has a few trajectories on the inside of the curve, which causes the sampled data shows a similar effect on the outside. This is a direct consequence of using a Gaussian probability distribution, which is assuming symmetry around the average. Additionally, a three-dimensional representation of the training data $\mathbf{Y}$, sampled data $\mathbf{S}$ and the volume containing one standard deviation is presented in figures 4.4 to 4.6 respectively.

### 4.1.3 Evaluation via KS statistic

The performance of the model is evaluated according to the technique discussed in section 3.3.2. For this experiment, the ECDF is created using the 32 trajectories from the *evaluation* dataset. Figure 4.7 displays both the ECDF (evaluation trajectory data) and the CDF (sampled trajectory data), showing that the K-S statistic is 0.131, corresponding to a maximum deviation of 13.1%. This low performance is attributed to the differences seen in the training data and sampled data and the limited number of trajectories in the evaluation data. A suggested approach to increase the performance

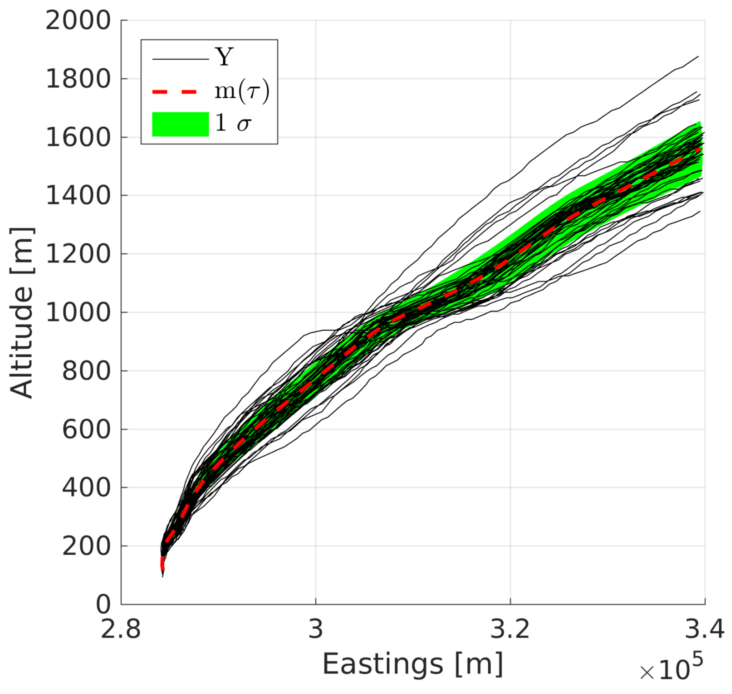


(a) aircraft trajectory training data **Y**

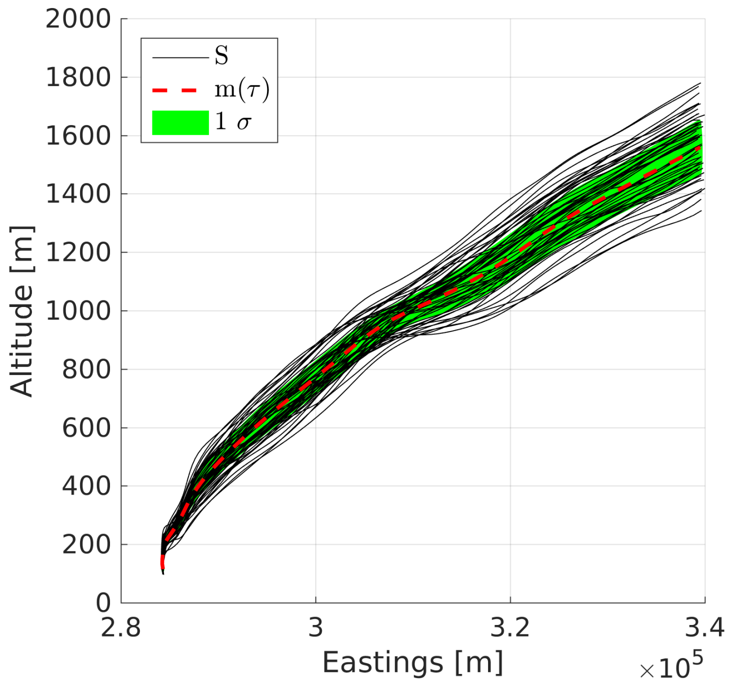


(b) aircraft trajectory sampled data **S**

Figure 4.2: Visualisation (top-view) of aircraft training and sampled trajectory data.



(a) aircraft trajectory training data **Y**



(b) aircraft trajectory sampled data **S**

Figure 4.3: Visualisation (side-view) of aircraft training and sampled trajectory data.

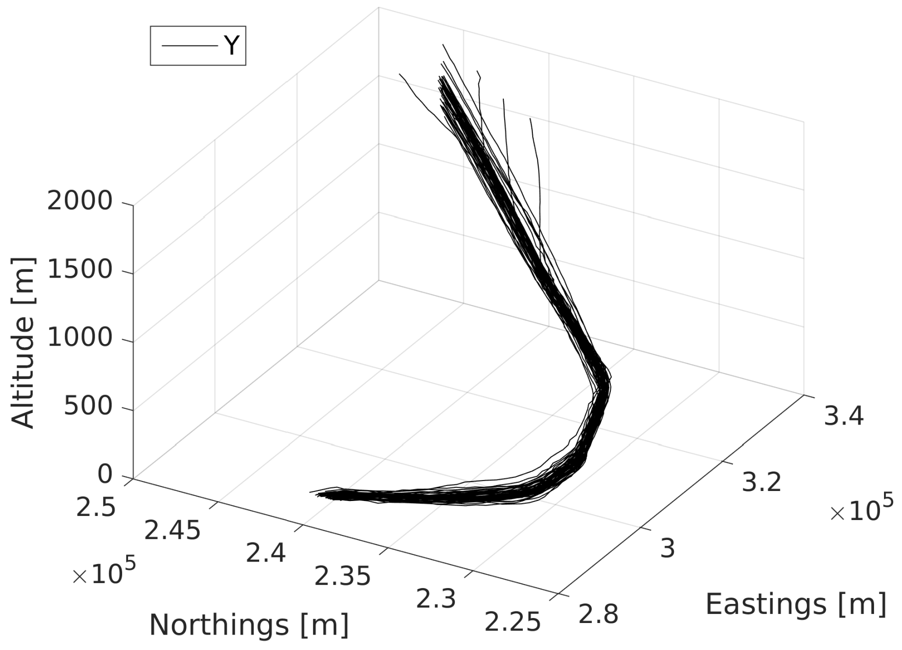


Figure 4.4: A three-dimensional representation of the aircraft training trajectory data $\mathbf{Y}$.

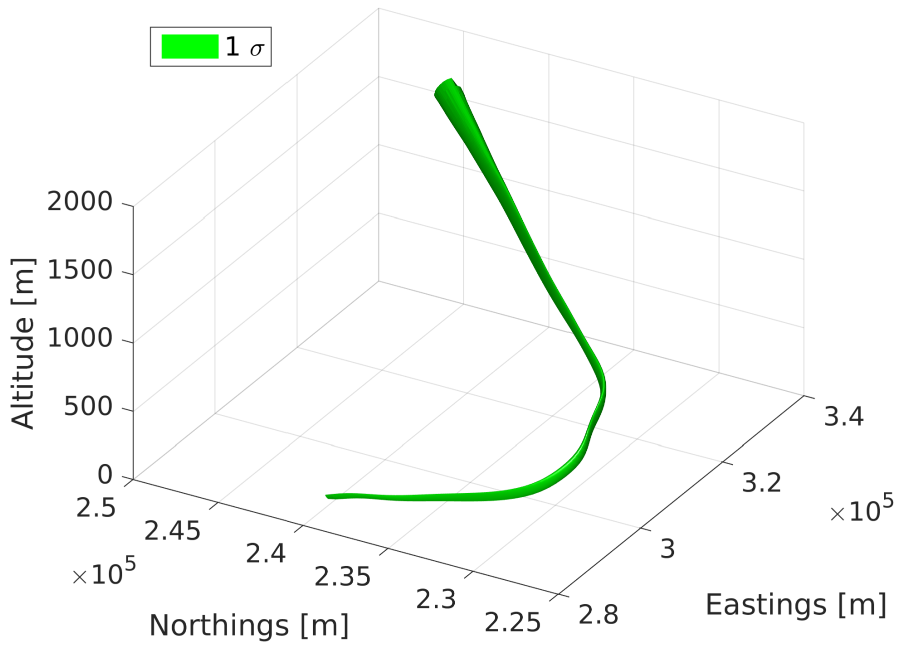


Figure 4.5: A three-dimensional representation of the volume containing one standard deviation for the aircraft trajectories.

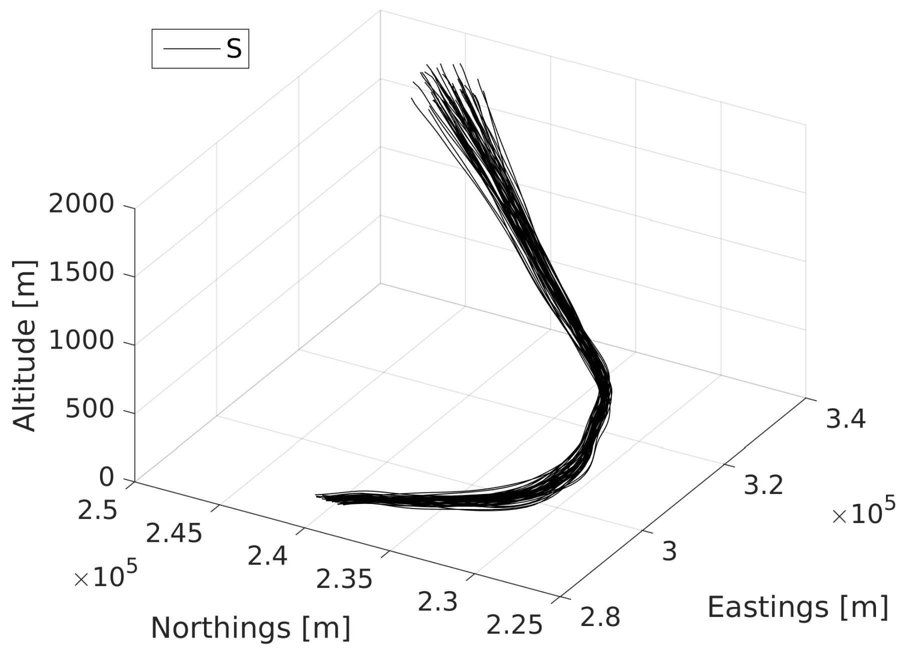


Figure 4.6: A three-dimensional representation of the aircraft sampled trajectory data **S**.

would be to filter the trajectories based on additional meta-data when available, e.g., aircraft type and intended flight path.

### 4.1.4 Situational awareness via dataset analysis

When performing a similar procedure on all clusters seen in figure 4.8a, a spatial analysis of the complete dataset, provides the overview as seen in figure 4.8b. Here, by displaying the standard deviation for each cluster, a quick overview of activities of the dataset is provided. The three-dimensional view of all aircraft trajectory data can be seen in figure 4.9a, with the corresponding overview in figure 4.9b. Using an Intel(R) Core(TM) i7-3770 CPU 3.40 GHz with 8 GB 1600 MHz DDR3, the complete analysis of 13 datasets was done in under 2 minutes. In this, the largest dataset containing 82 trajectories takes up 24 seconds, and the smallest dataset containing 20 trajectories only takes 4 seconds.

The overview as seen in figure 4.8b is a snapshot of the specific conditions captured in the trajectory data. For an airport, the conditions would include the time of day (e.g., alteration of runway use due to noise regulations), date (e.g., holiday season),

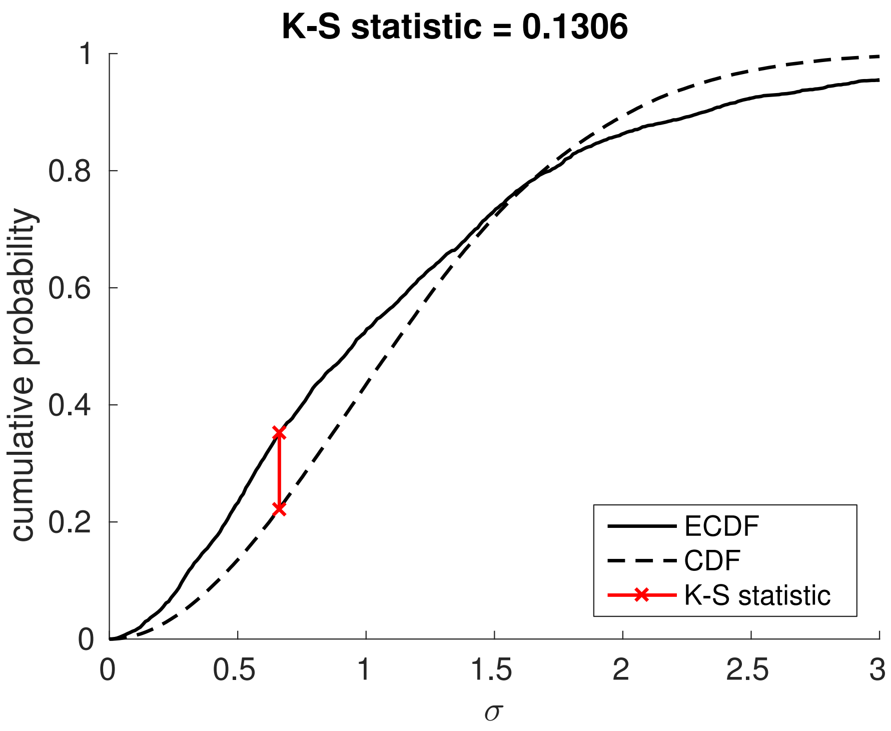
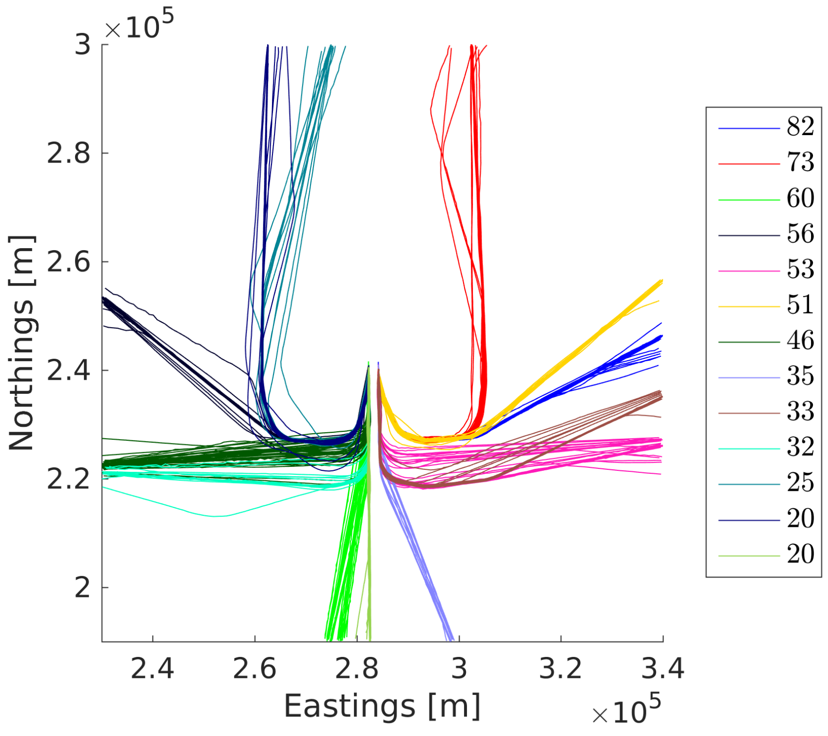


Figure 4.7: Comparing the aircraft evaluation and sampled trajectory data.

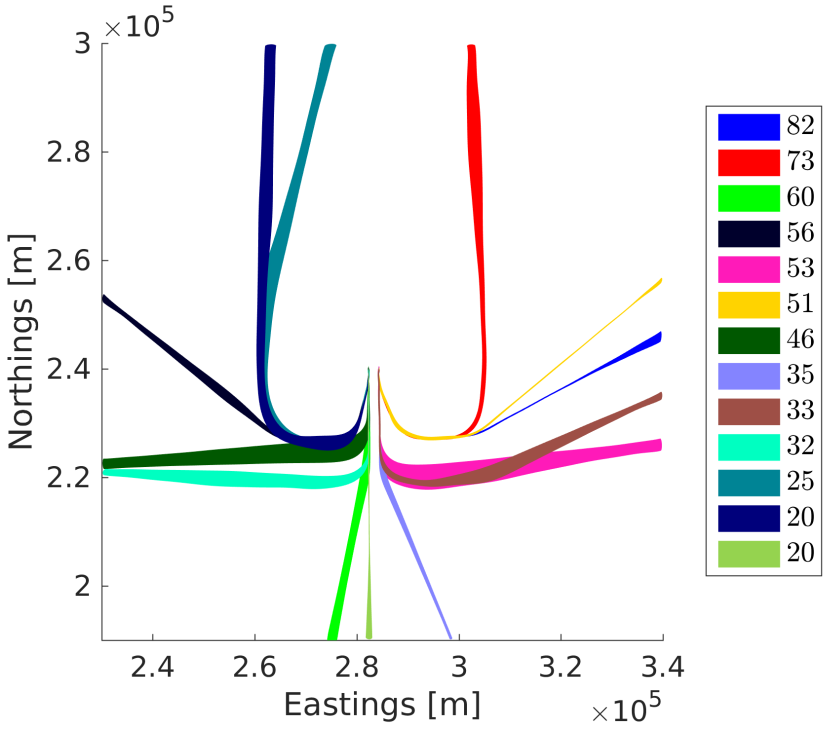
and weather conditions. In this particular dataset, as the trajectory data only shows aircraft departing towards the south, indicating there was a southern wind throughout the entire measurement period. It is up to the user to decide whether an average over all conditions is desired, or the dataset should be separated such that the probabilistic model represent specific conditions.

## 4.2 Application of weighted trajectories with the focus on airspace protection

This section applies the step-by-step guide as described in section 3.5.3 on aircraft trajectories near DFW airport as measured by ground-based radar. The first step is to cluster the trajectories, after which the individual clusters of trajectories (sorted per common flight path) are modelled, and in the final step a sub-set of representative trajectories is generated. For the evaluation a $100 \times 100$ m$^2$ grid is placed near the airport at a ground level. For each of these grid-points the percentage of the total number of trajectories in the data that come within 300 m, as measured from the centre of

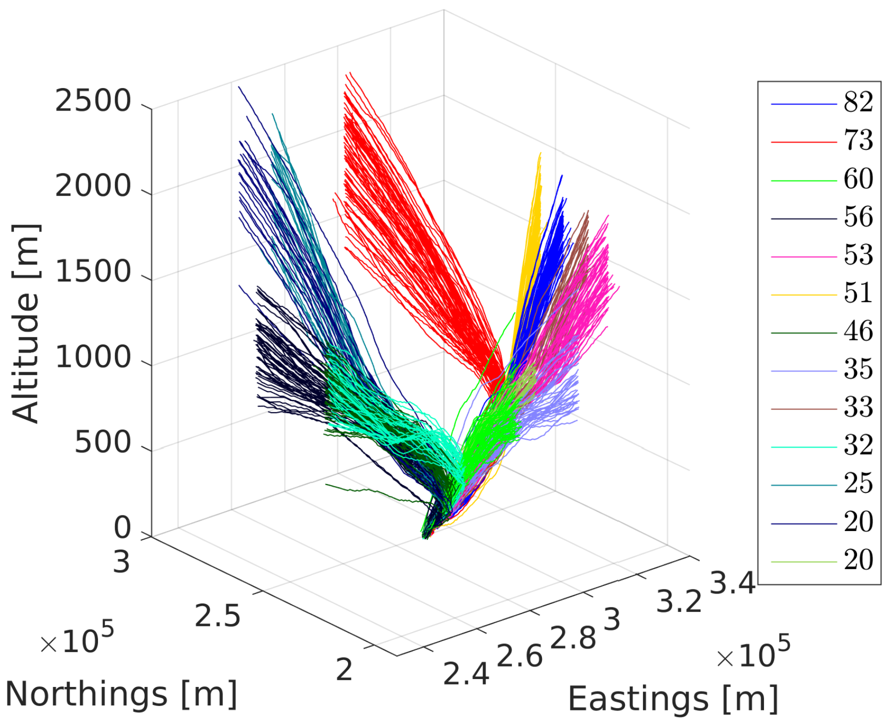


(a) the aircraft trajectory data **Y**

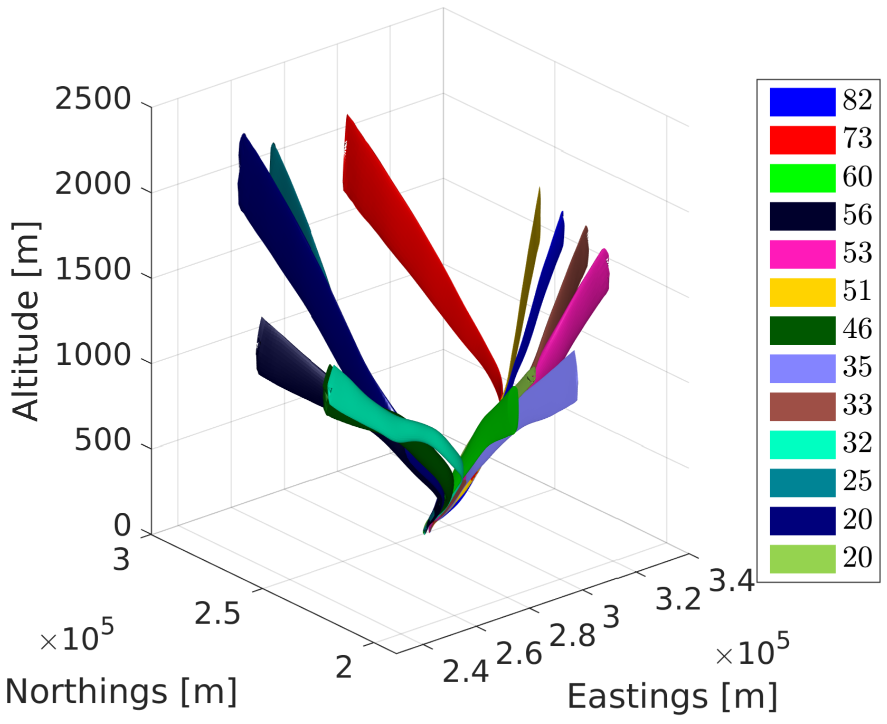


(b) the area covering one standard deviation for the aircraft trajectories

Figure 4.8: A top-view of all departing aircraft trajectories from two runways at DFW.



(a) the aircraft trajectory data **Y**



(b) the volume containing one standard deviation for the aircraft trajectories

Figure 4.9: A three-dimensional view of all departing aircraft trajectories from two runways at DFW.

Table 4.2: An overview of the clustered trajectory data.

| **cluster** | 1 | 2 | 3 | 4 | *total* |
|---|---|---|---|---|---|
| **size** | 267 | 194 | 178 | 176 | 815 |

each grid-point, is calculated. Therefore, under the assumption that each trajectory is equally likely to occur, it can be said that a person standing in the area with the highest percentage is more likely to 'spot' an aircraft within a 300 m radius, and in the areas with 0% there is no chance at all. This case study relates to an article about beating ballistic threats (Colucci 2008). In this article it was pointed out that there is evidence that terrorists have access to RPGs, and that these weapons are effective up to a range of 300 m against a moving target.

Section 4.2.1 starts by clustering and modelling the dataset. Next, section 4.2.2 constructs the weighted trajectories following the procedure from section 3.5.3. Finally, section 4.2.3 evaluates the performance of the weighted trajectories as an approximation method.

### 4.2.1 Clustering and modelling

The trajectory data are clustered according to the same approach as described in section 4.1.1, where the result is visible in figure 4.10. For the analysis in this section, only the *departure* trajectories are included to keep the amount of trajectories manageable and the eventual evaluation of a reasonable scale. The number of trajectories found in each cluster is shown in table 4.2. The $\epsilon$-neighbourhood parameter $\epsilon$ of the DBSCAN algorithm is set to 0.20 and the minimum number of trajectories in each cluster is 25, causing just over 19% (193 trajectories) to be considered outliers. These outliers are not shown. Furthermore, as the ultimate goal is to create a footprint based on the distance of 300 m, only the parts of the trajectories below an altitude of 500 m are used in the analysis.

Via the approach introduced in chapter 3, a model is generated for each cluster. The basis functions $\phi(\tau)$ in this section consist of 17 radial basis functions, uniformly distributed over the interval $\tau = [0, 1]$. This brings the total number of basis function $J$

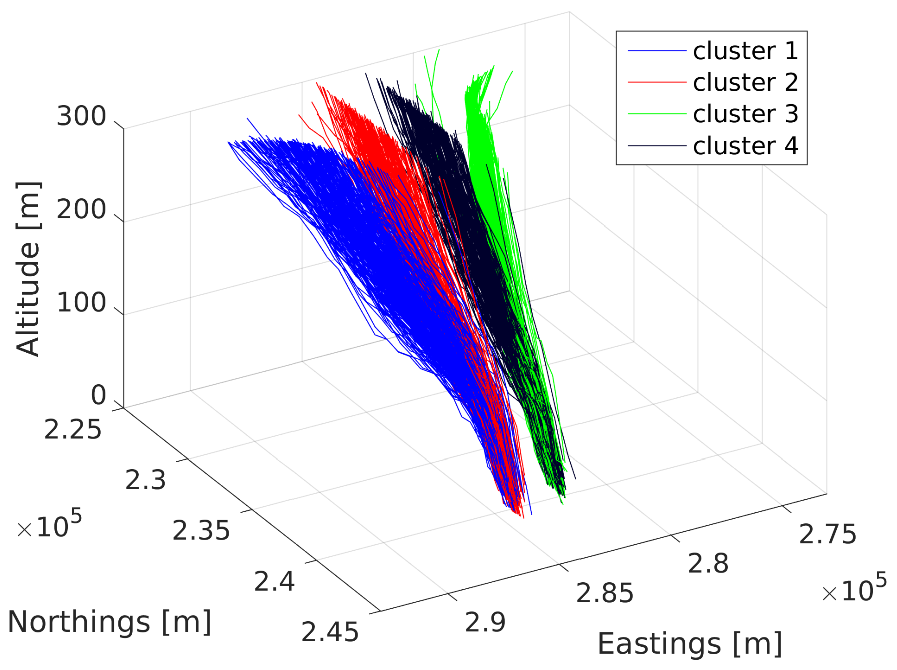


Figure 4.10: Three-dimensional view of all the clustered original trajectory data.

to 18, as there is an additional parameter due to the bias. The convergence criteria is set such that the difference between two sequential iterations, and evaluation of the log-likelihood, is at most $10^{-3}$. The result is shown in figure 4.11, where the volume represents the range corresponding with a standard deviation of two ($\sigma = 2$).

### 4.2.2 Construct weighted trajectories

In section 3.5.3 two approaches to generate representative trajectories were described. One automatically generates the trajectories while only taking into account the lateral dispersion. The cross-section of this *flat* version is seen in figure 3.6, and the resulting top-view is visible in figure 4.13. This modelling approach is currently being used in calculating noise footprints around airports, as described in the official publication (European Civil Aviation Conference 2016). The other approach takes into account both the lateral and vertical dispersion simultaneously. The cross-section of this *round* version is seen in figure 3.7, and the resulting top-view is visible in figure 4.14.

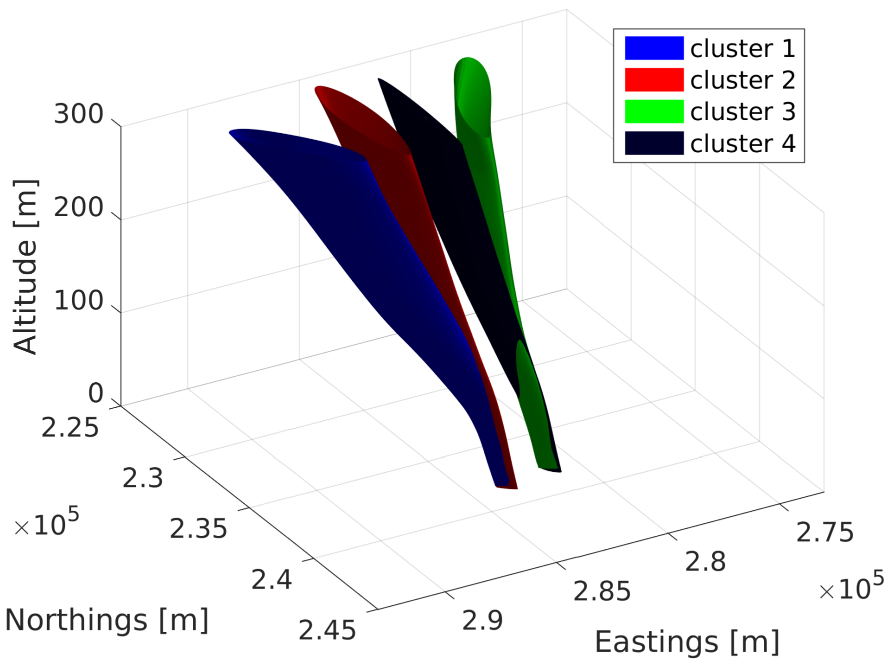


Figure 4.11: Three-dimensional view of all models - where the volume corresponds with a standard deviation of two.

The difference by taking into account the vertical dispersion becomes apparent when examining the side-views seen in figures 4.15 to 4.17, where the original trajectory data and the two version of representative trajectories are displayed. It clearly shows that modelling the vertical dispersion, actually *represents* the vertical dispersion as seen in the original data.

### 4.2.3 Evaluation of footprint

For the evaluation a 500 × 500 grid is placed near the airport at a ground level. For each of these grid-points the percentage of the 815 trajectories in the data that come within 300 m is calculated. This simulates a simplistic approach to calculate from which locations the aircraft are vulnerable to an attack using RPGs, without taking into account any mapping nor specific weapon characteristics. For comparison, the analysis is performed on the original trajectory and the two versions of the representative trajectories.

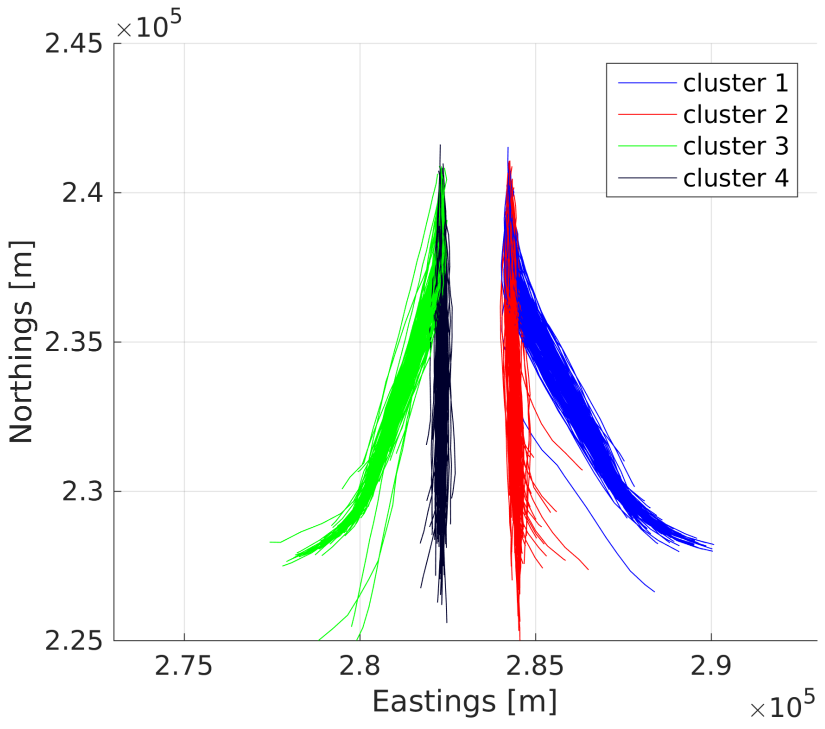


Figure 4.12: Top-view of all the clustered trajectories.

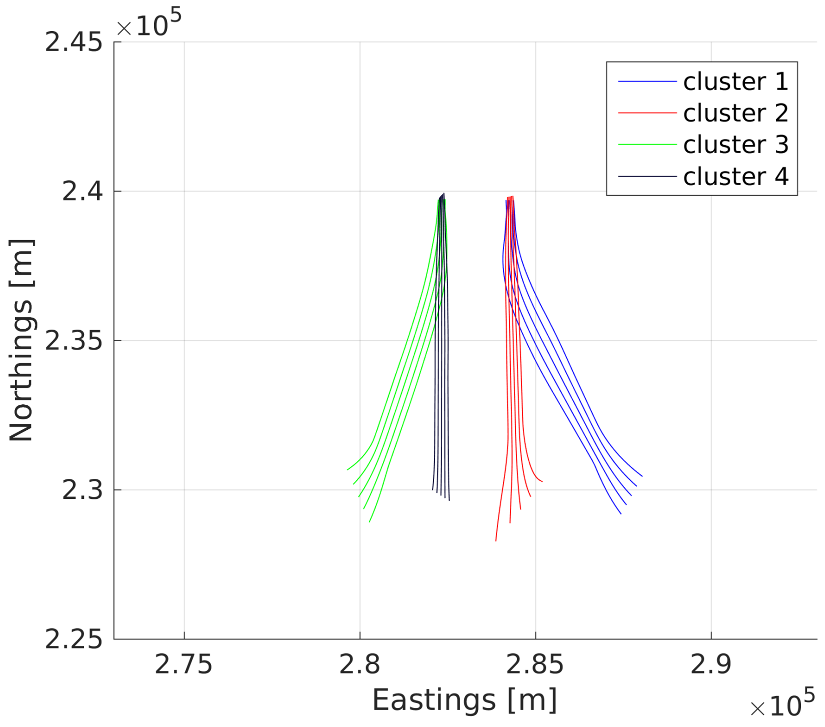


Figure 4.13: Top-view of the representative trajectories, lateral direction only.

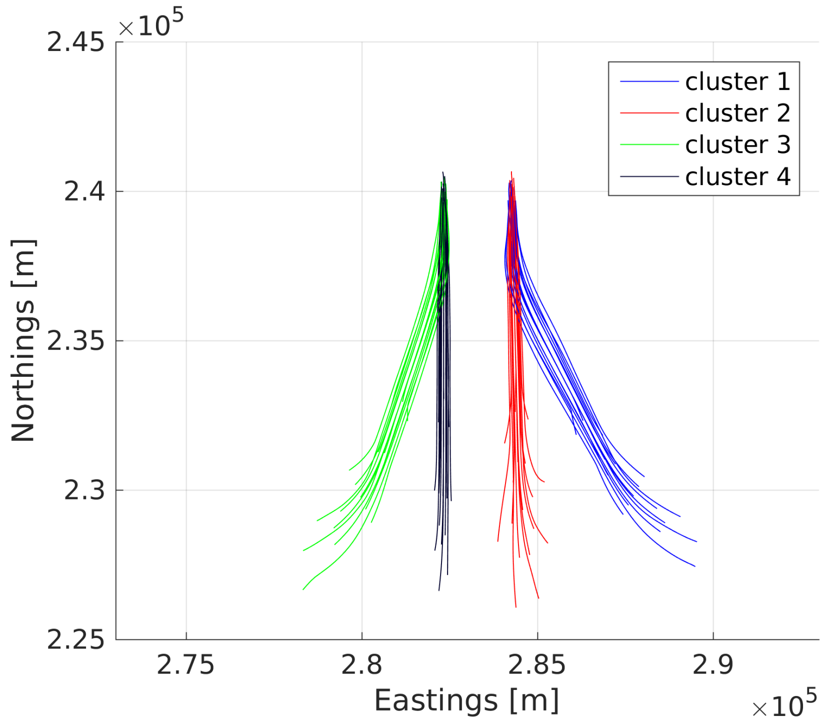


Figure 4.14: Top-view of the representative trajectories, lateral and vertical direction.

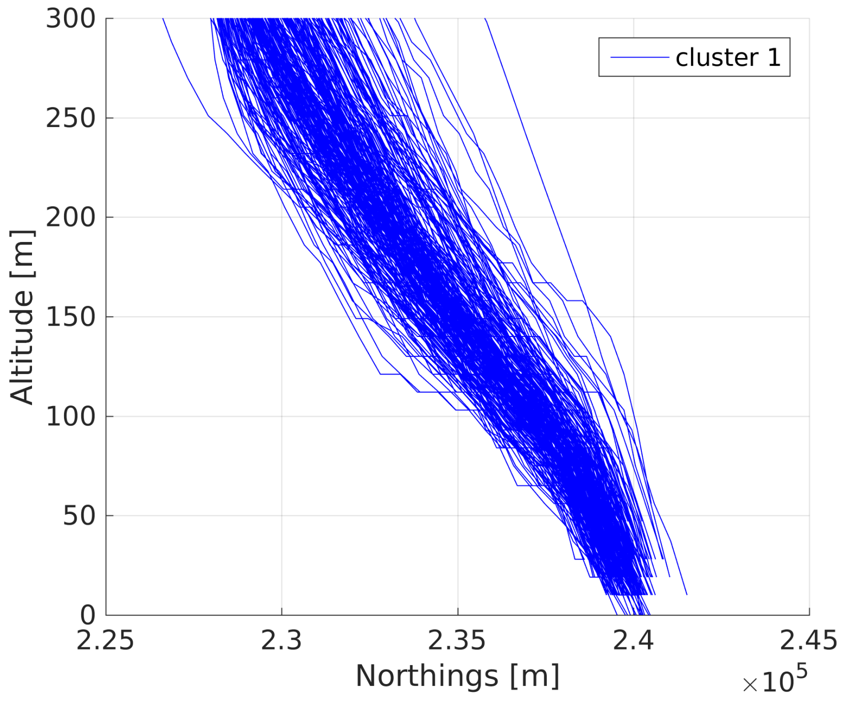


Figure 4.15: Side-view of the largest cluster of the original trajectory data.

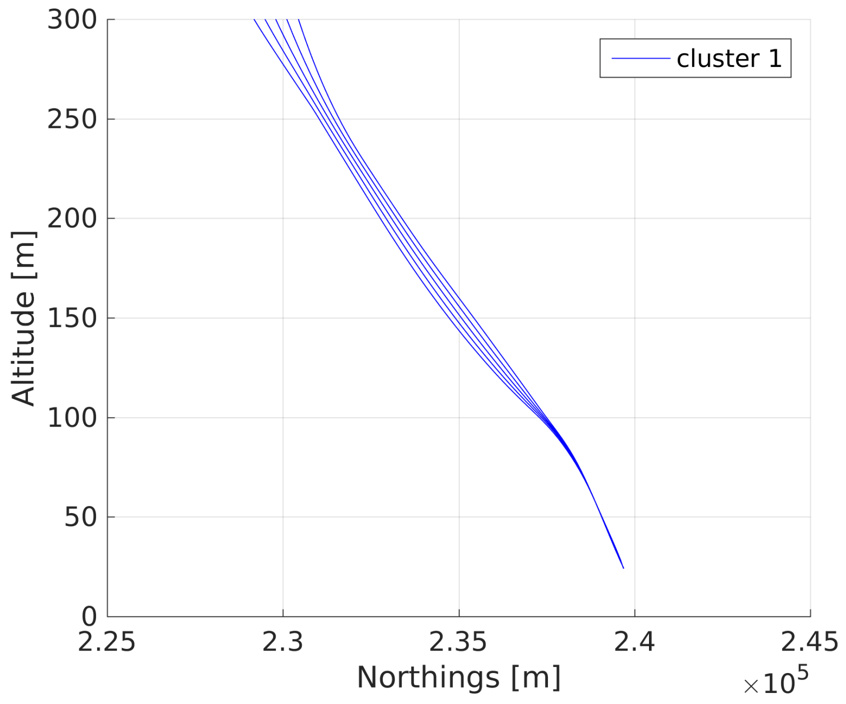


Figure 4.16: Side-view of the representative trajectories, lateral direction only.

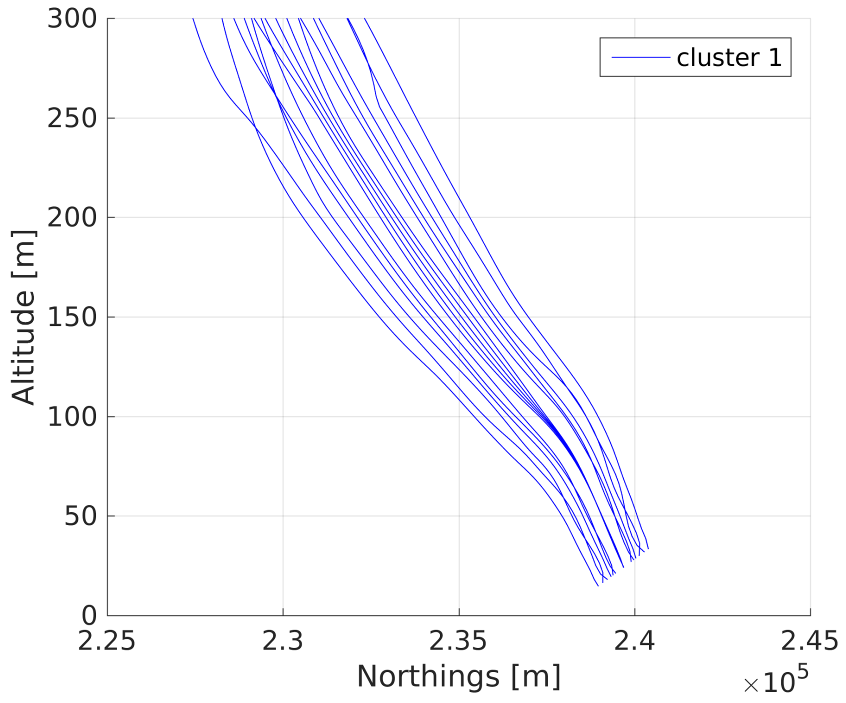


Figure 4.17: Side-view of the representative trajectories, lateral and vertical direction.

The resulting footprint calculated for the original trajectory data is visible in figure 4.18, where the range of percentages has been set to visualise the sweep of low percentages towards the end. This sweep is missing from figure 4.19, where only the lateral dispersion is taken into account. While the visible roughness is a direct effect of only using 5 trajectories to represent each cluster, the missing sweep is a direct result of missing trajectories at a lower altitude. As such, this sweep is visible in figure 4.20. Here there are representative trajectories at a lower altitude, even at a distance further away from the airport. These differences are emphasised in figures 4.21 and 4.22, where the difference in percentages are plotted.

The quantitative results comparing the methods to generate representative trajectories have been gathered in table 4.3, showing the percentages of grid-points that are under- and overestimated. These percentages are taken from the total number of non-zero grid-points, as otherwise including a large area far away from the airport would decrease the values. The results with *no margin* represent the balance between under- and overestimation. When allowing a 5% margin of error, it shows that only 0.2 of the non-zero grid-points are underestimated. From a safety perspective, this value should be as low as possible. Finally, 13.8% is overestimated more than 5% for the lateral and vertical dispersion, while 24.3% is overestimated when only modelling the lateral dispersion. This can be interpreted as the vertical dispersion expanding the area of influence (see expansion in figure 4.22), yet reducing the intensity of the peaks.

## 4.3 A decision support tool for air traffic management

This section will apply the approach of modelling the trajectory data as a Gaussian process to the visualisation of air traffic trajectory data as flight-corridors. This shifts the focus from density-based method to a probabilistic analysis. As a result, less frequently used flight paths are not automatically ignored, but included in the analysis. Furthermore, by focusing on identifying and visualising flight corridors used by dominant flows of air traffic, areas that have a structural presence of high air traffic complexity in terms of traffic structure can be identified and visualised. The approach is data-driven, and

Table 4.3: Comparing the footprints of the representative trajectories with the original trajectory data.

| | **Lateral dispersion only** | **Lateral and vertical dispersion** |
|---|---|---|
| | *no margin* | |
| underestimated [%] | 38.8 | 27.1 |
| overestimated [%] | 61.2 | 72.9 |
| total false [%] | 100.0 | 100.0 |
| | *5% margin* | |
| underestimated [%] | 1.4 | 0.2 |
| overestimated [%] | 24.3 | 13.8 |
| total false [%] | 25.7 | 14.0 |
| | *10% margin* | |
| underestimated [%] | 0.7 | 0 |
| overestimated [%] | 4.6 | 0.7 |
| total false [%] | 5.3 | 0.7 |

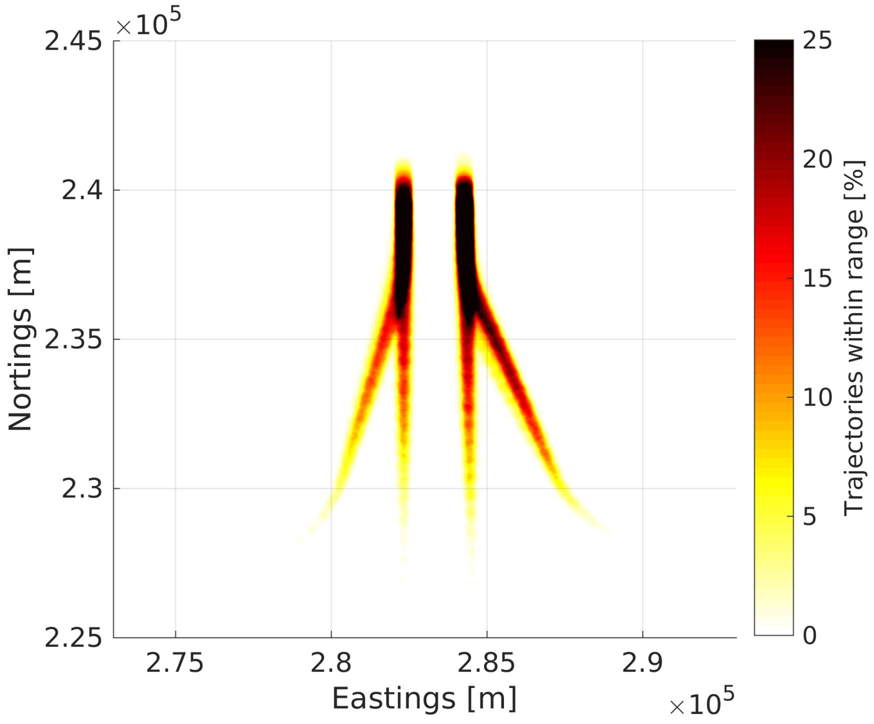


Figure 4.18: Footprint generated using the original trajectory data.

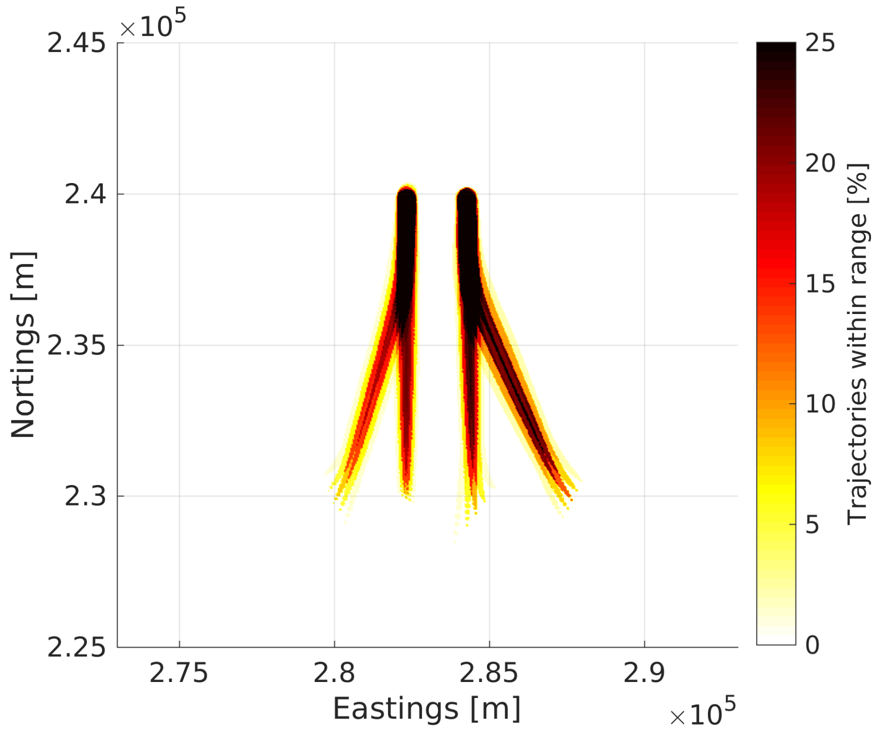


Figure 4.19: Footprint generated using representative trajectories capturing lateral dispersion only.

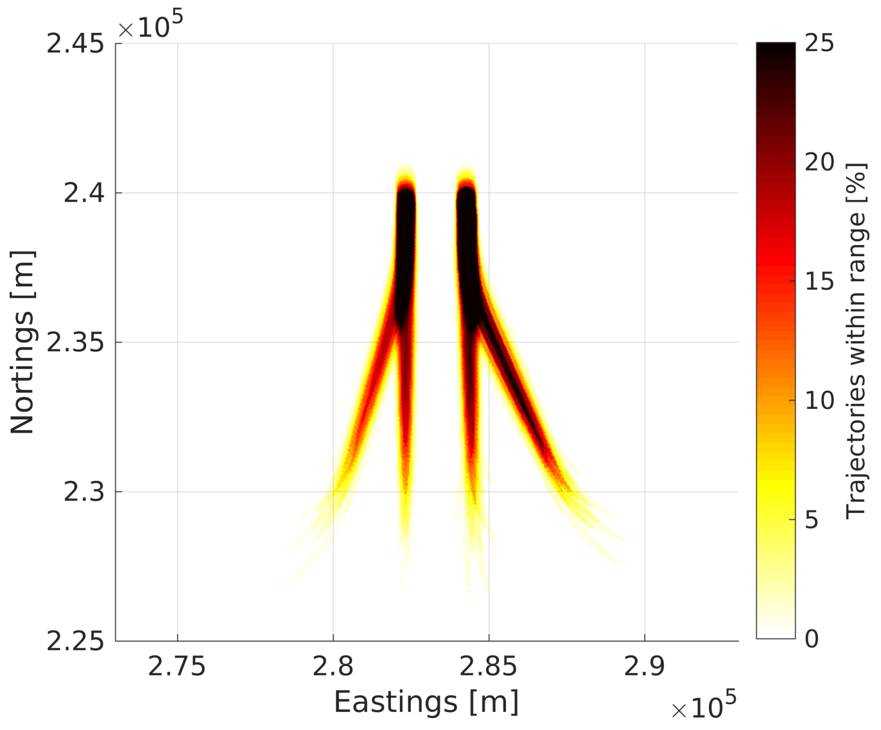


Figure 4.20: Footprint generated using representative trajectories capturing lateral and vertical dispersion.

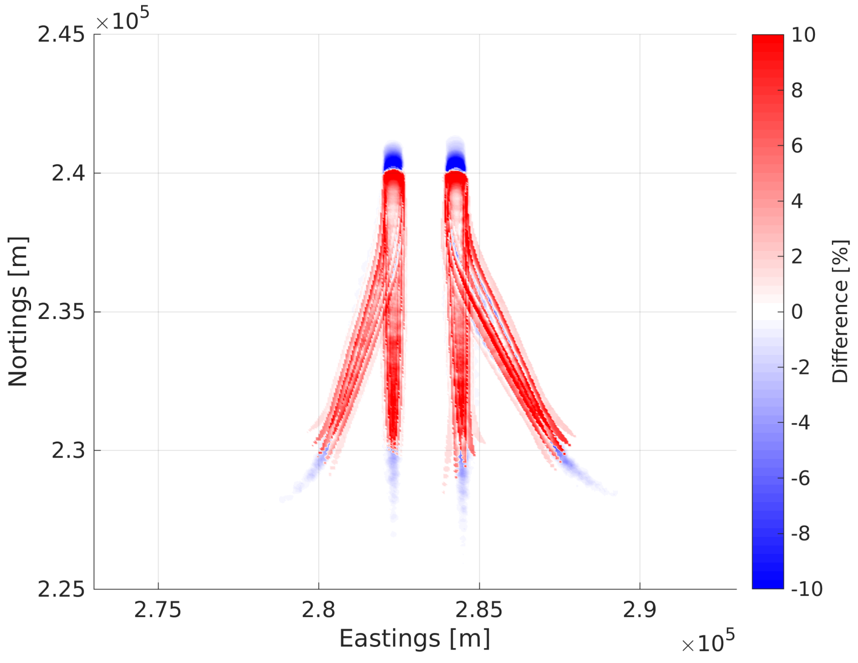


Figure 4.21: Difference in footprint between the original trajectories and representative trajectories capturing lateral dispersion only.

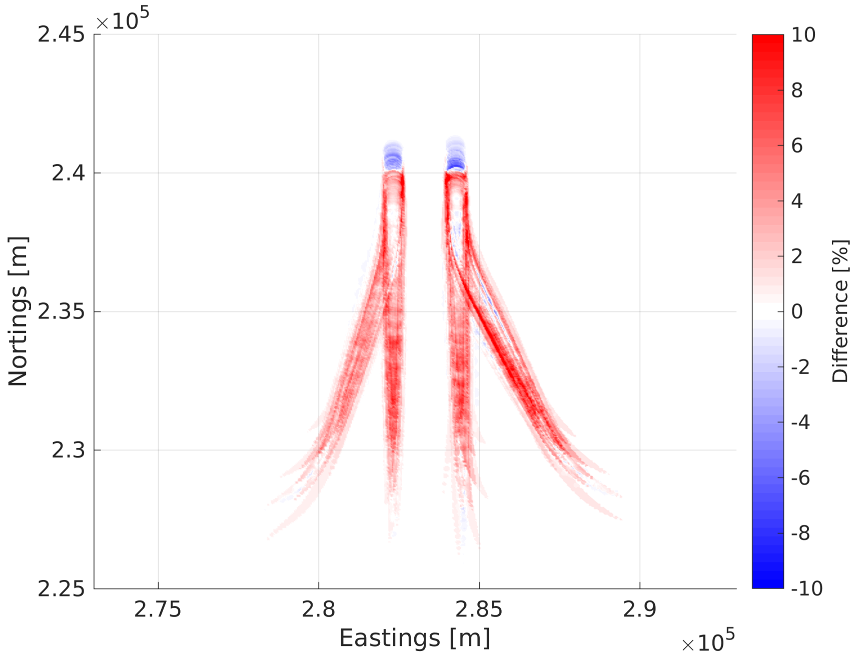


Figure 4.22: Difference in footprint between the original trajectories and representative trajectories capturing lateral and vertical dispersion.

Table 4.4: An overview of the clustered trajectory data.

| **cluster** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | *total* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **size** | 80 | 63 | 58 | 37 | 26 | 25 | 10 | 89 | 104 | 78 | 570 |

requires minimum human interaction to construct the flight corridors and complexity maps from the trajectory data, save selecting the resolution of the evaluation. The flight corridors are generated via the open-source Python toolbox as introduced in section 3.6.

First, section 4.3.1 elaborates on how the motion patterns are extracted from the dataset. Then section 4.3.2 visualises the corridors, providing situational awareness. Finally, section 4.3.3 demonstrates how the complexity map is able to identify conflict regions using complexity measures.

### 4.3.1 Clustering

In this section, the previous described methods are applied to a trajectory dataset measured by ground-radar at Denver airport (DEN). The trajectory data for *approaching* aircraft are clustered per common flight path via the method presented by Gariel, Srivastava and Feron (2011). The epsilon parameter used in the clustering is 0.3, with a minimum number of trajectories per cluster of 10. As a result, 909 trajectories were placed in 10 clusters, where 339 trajectories are considered outliers. A summary of the clustered trajectory data is available in table 4.4. The trajectory data have been cut-off at 800 m prior to the clustering step.

Up to 50 trajectories per clustered data are seen in figures 4.23a and 4.24a, showing identical trajectory data as seen from a different viewpoint. A maximum of 50 trajectories per cluster are shown to prevent visual clutter. Furthermore, the airport runways have been included and are visible as red lines.

### 4.3.2 Flight corridors

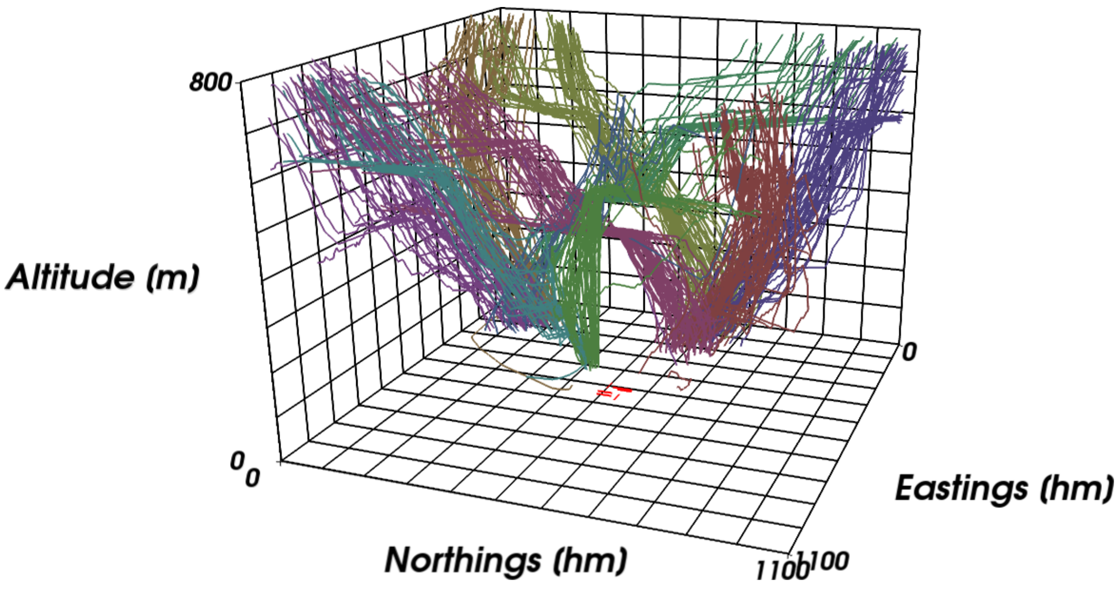
By applying the method as presented in section 3.5.1, the flight corridors are generated for a standard deviation of one. The results are visible in figures 4.23b and 4.24b, with identical viewpoints to the trajectory data. Again, the airport runways are visible as red lines. Each cluster of trajectory data and the corresponding flight corridor are linked via colour coding.

The same models capturing the flight corridors at a standard deviation of one are presented at a constant altitude in figure 4.25. As the aircraft in the trajectory data are on approach, the altitude changes from high (550 m) to low (400 m). The airport runways are visible as red lines, and the average trajectory for each cluster is visible as a dashed line. The colour coding is consistent with figures 4.23 and 4.24. The overlap of the flight corridors at an altitude of 450 m, both to the east and west of the airport, reveal a high complexity in terms of traffic structure.

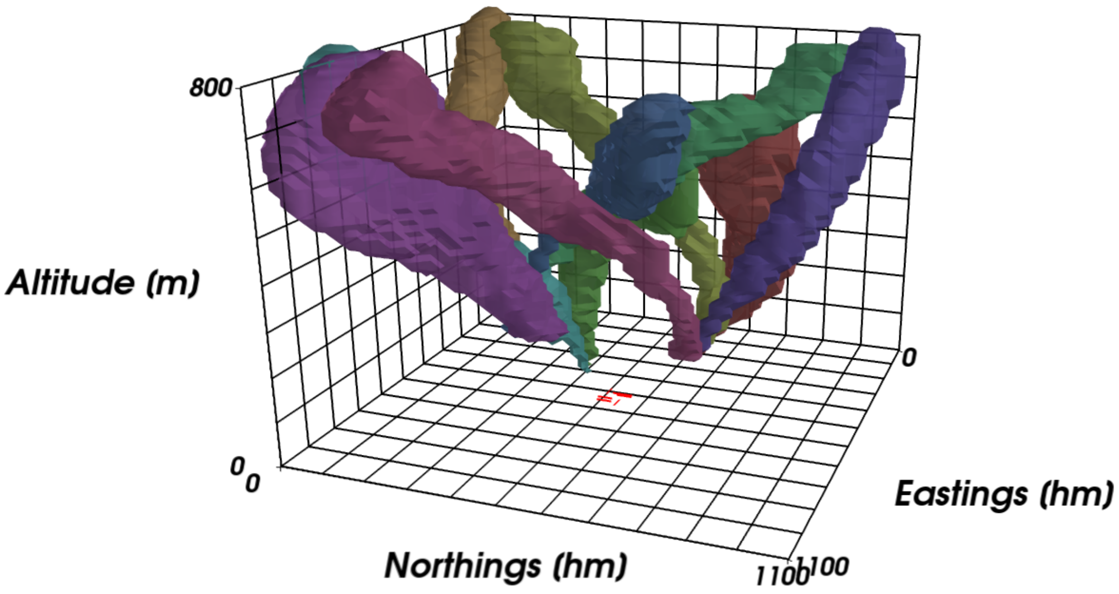
### 4.3.3 Complexity map

These results are supported by the complexity map, visible in figure 4.26 at varying degrees of complexity measures at an altitude of 450 m. Here $\mathcal{C}^1$ represents the log-probability of at least the presence of one flight corridor, $\mathcal{C}^2$ at least two corridors, and so forth. It should be noted that the complexity maps shown are normalised, hence not displaying a large difference between the complexity measures. Furthermore, each complexity measure represent the *background noise* from the lower measure (e.g., $\mathcal{C}^2$ is the lower complexity measure of $\mathcal{C}^3$). This follows from the definition, where $\mathcal{C}^2$ represents the log-probability of *at least* 2 flight corridors colliding – this includes the values of $\mathcal{C}^3$, which represents the log-probability of *at least* 3 flight corridors and more.

Figure 4.26b shows the most important complexity measure for identification of multi-aircraft conflict situations, the log-probability of two or more corridors intersecting ($\mathcal{C}^2$). These results highlight the regions to the east and west of the airport, as previously determined via the corridors seen in figure 4.25.

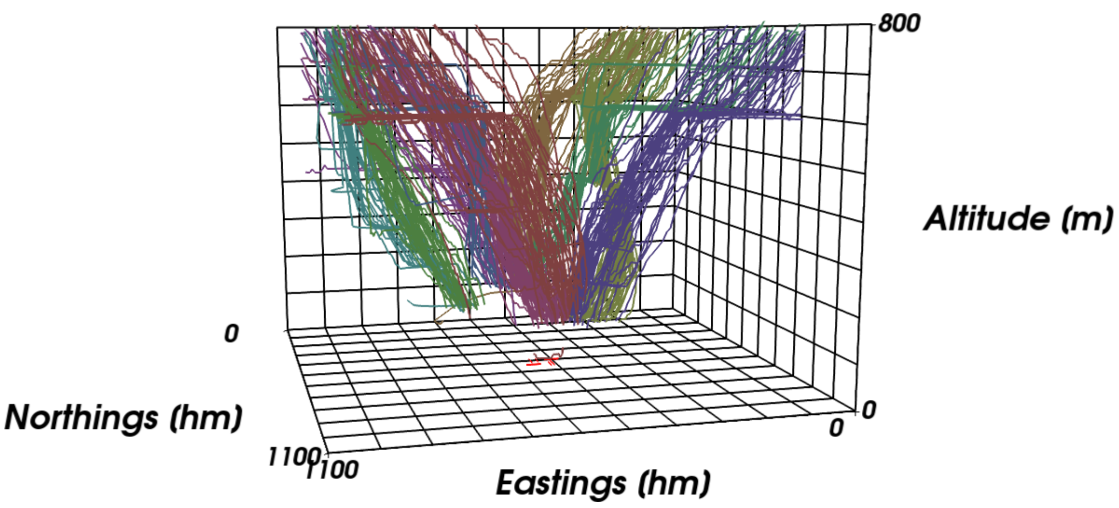


(a) trajectory data

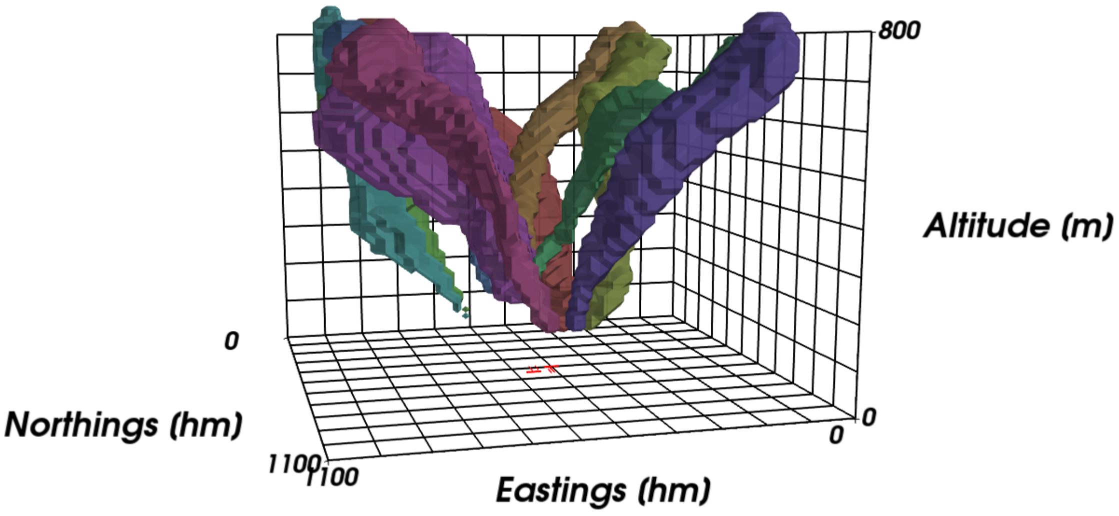


(b) flight corridors

Figure 4.23: Comparing trajectory data and flight corridors, first viewpoint.



(a) trajectory data



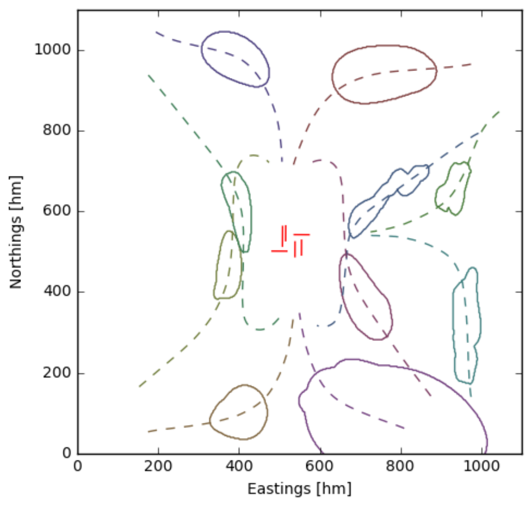
(b) flight corridors

Figure 4.24: Comparing trajectory data and flight corridors, second viewpoint.

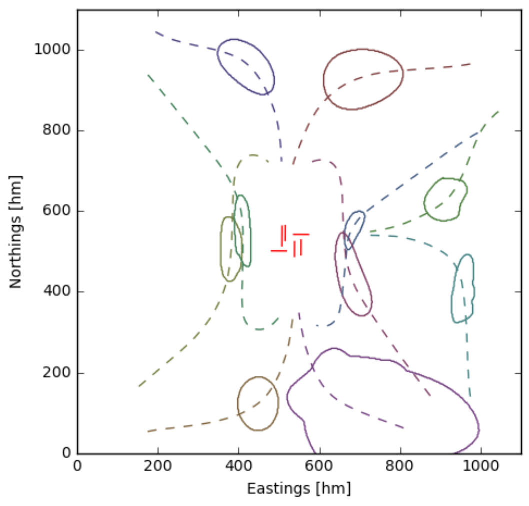
## 4.4 Conclusion

The analysis of an ensemble of commercial aircraft trajectories recorded for departures from DFW airport showed a maximum deviation of 13.1%. Further analysis showed that – by combining the proposed modelling approach with a clustering algorithm – large sets of trajectories covering multiple discrete flight paths may be analysed. A key point here is the ability to visualise the trends of large amounts of data in a single figure (or perhaps in the near future, a hologram).

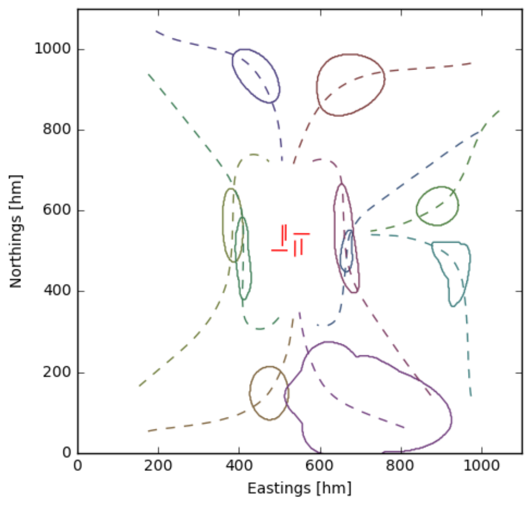
Furthermore, this method captures trajectory data in a probabilistic model, allowing



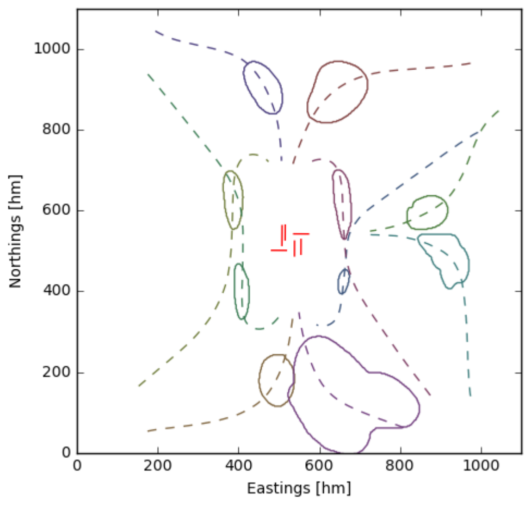
(a) 550 m



(b) 500 m



(c) 450 m

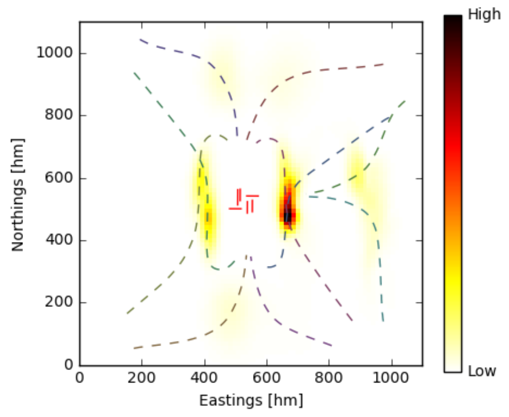


(d) 400 m

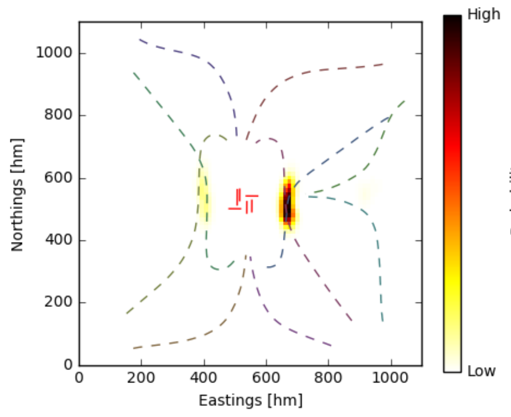
Figure 4.25: Flight corridors at a constant altitude.

subsequent calculations to be done on the model and not the vast amount of measured trajectory data, thus reducing the computational cost. The approximation method via weighted trajectories appears to be conservative when considering a distance measure on ground-level, where less than 0.2% of the points were underestimates more than 5%, and less than 14.08% was overestimated with less than 5%.

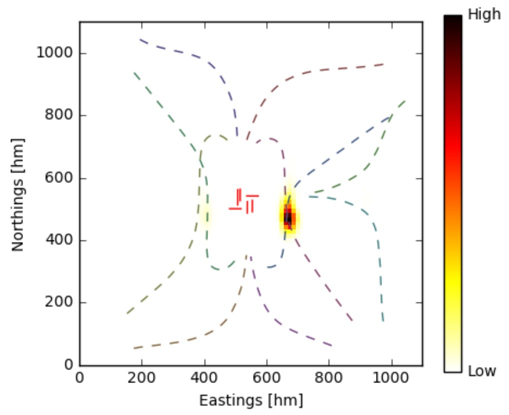
Finally, the modelling method has demonstrated how it can be applied as a decision support tool for air traffic management by visualising flight corridors and identifying conflict regions via the complexity map.



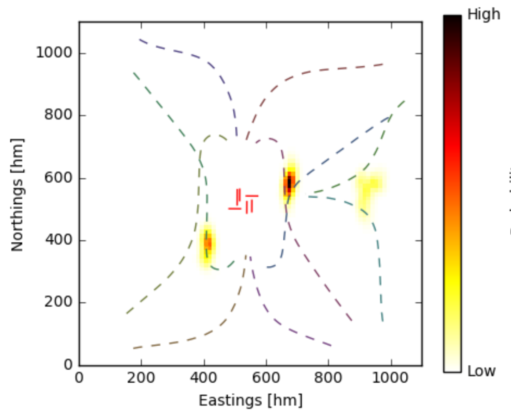
(a) $\mathcal{C}^1$



(b) $\mathcal{C}^2$



(c) $\mathcal{C}^3$



(d) $\mathcal{C}^4$

Figure 4.26: Complexity map at varying degrees of complexity measures at an altitude of 450 m.

# Chapter 5

# Evaluation of security threats in urban and rural environments

This chapter is about applying sampling strategies that lead towards an effective and efficient strategic analysis of a selected area. Such a strategic analysis can involve determining visibility of a target via line-of-sight, identifying launch sites able to reach a target using expert knowledge on specific weapon capabilities, or even more complex analysis making use of a missile flyout model (Antonio 1986). Therefore, the physical system of interest (e.g., line-of-sight or a rocket flight trajectory), in general, is captured in a computational model. Hence, the strategic analysis, as discussed in this chapter, is essentially a computational experiment with coded algorithm as an evaluation function.

Section 5.1 begins with an overview on the existing literature on sampling strategies for computer experiments. Next, section 5.2 provides a description of the problem at hand and introduces the various scenarios considered. These scenarios are a mixture of *mostly urban* and *mostly rural* case studies, corresponding to the surface data of London and the area surrounding Southampton respectively. Furthermore, section 5.3 proposes techniques suitable for solving the problem at hand. Thereafter, Section 5.4 presents the results based on the case studies. And finally, section 5.5 concludes with some final remarks, summarising the work presented in this chapter.

## 5.1 Sampling strategies for computer experiments

This section describes certain methods available in the literature that solve problems concerning sampling strategies. In this chapter, the term *computer experiments* corresponds to a computational model being used for the evaluation of an experiment. Such a model can be seen as a function that given inputs, produces outputs at a computational cost. Such computational models refer to systems that are based upon mathematical equations and then implemented as a coded algorithm as no analytical solution is available. In other words, while the function and the inner workings are known, it is too complicated to solve analytically.

Computational models are used in various engineering disciplines, such as structural design optimisation, computational fluid dynamics, heat transfer, and many more applications. Typical for computational models is that there are many inputs, resulting in a high dimensional input space. Furthermore, as the computational model grows more complex, the computational cost of evaluating a point increases. For these reasons it makes sense to construct a plan to spend the computational budget wisely and learn as much of the relations between the inputs of the computational model as possible.

In the remainder of this section, methods to generate sampling plans are discussed. Following this is an overview of active data selection methods. Essentially, the sampling plan is about selecting points to evaluate just knowing about the surface map, or in more general terms, the input domain. The active data selection is about selecting points after *some* points are evaluated, while there is still computational budget left to evaluate more points, where the advantage lies in selecting those points to allow for either *exploitation* or *exploration*, depending on what is prioritised.

### 5.1.1 Sampling plan

Computer experiments vary from physical experiments, where a helpful property of computer experiments in contrast to physical experiments is that these have few sources of noise. Where physical experiments may involve people, measurements, and other sources of noise, computer experiments only have to deal with numerical errors. That is, considering a specific input (e.g., a position on the map) the same result is produced every time the code is executed. Therefore, at each of the evaluated points, in the input space there is zero prediction error, corresponding with the absence of uncertainty about the provided output value. Although it is possible to introduce randomness in the system to obtain an uncertainty analysis, as done for the rocket simulator in chapter 7, these exact input variables can be stored and will always produce the same answer, no matter how often the experiment is repeated.

Without the need to evaluate identical points twice, and assuming no knowledge of the relation between the input and output, the optimal design is spreading the evaluation points evenly throughout the input domain. There are several ways to spread the points evenly throughout the domain, these methods are usually referred to as *space-filling* or *exploratory* designs (Box and Draper 1987; Santner, Williams and Notz 2003). Via the maximin distance design criterion, a score of how space-filling a design is can be calculated (Johnson, Moore and Ylvisaker 1990). This geometric criterion (based on distance) is shown to correspond to a D-optimal criterion, the overall function uncertainty remaining after following sampling plan. Here, *D-optimality* minimizes the determinant of the covariance matrix with the assumption that no prior knowledge of the estimated function is available.

Initially, when confronted with many inputs and an unknown relation between those inputs, *factor screening* and *sensitivity analysis* are two terms that come up. This is all about discovering the impact specific inputs have on the outputs, where this information can be used to prioritise sampling in a specific input dimension.

When only one or a few input dimensions have a large impact on the output, a completely uniform spread, also called uniform sampling, may not be beneficial. This would spend an equal amount of computational budget on sampling along each dimension, spending the budget to learn about inputs that have little to no impact on the outputs. A sampling plan often used for an initial screening is the Latin hypercube design (Iman and Conover 1980), where the samples are evenly distributed for each one-dimensional projection. Here the values along each dimension in the input domain of interest are generated using a restricted random procedure. As a result, along each dimension there are $n$ unique values selected from $n$ overlapping intervals. These intervals span the input domain of interest, which is to be specified in advance.

In effect, a high score of the maximin distance design criterion indicates good global performance, where the minimum distance between pairs in a sampling plan is maximised. On the other hand, Latin hypercube designs (for a given number of unique samples, there are a multitude of Latin hypercube designs possible) guarantee good projective properties in each dimension. The combination, where several Latin hypercube designs are generated and the most space-filling design is selected via the maximin distance design criterion, yield designs which are effective for predicting both when few or many inputs are important (Morris and Mitchell 1995). However, it should be noted that the real advantage of using a Latin hypercube design over a uniform grid, is in the case where a limited number of inputs correlates with the output.

### 5.1.2 Active data selection

The previous section assumes no prior information about the objective function is available. However, when prior information is available and samples are selected properly, the data requirements for some problems decrease drastically (Angluin 1988). Here, the selection of samples based on prior information is referred to as *active data selection*. In practice, active data selection offers its greatest rewards in situations where data are expensive or difficult to obtain. For this reason, much of the work in this section is found in the field of geostatistics (Cressie 1993). In fact, the original problem for which geostatistical methods were developed is to predict the likely yield of a mining operation over a region, given the samples of ores extracted from a set of locations (Gelfand et al. 2010). The remainder of this section focusses on the question where the next ore sample should be extracted to obtain a better prediction in the region of interest.

One of the earlier examples of improved learning are performed using neural networks (Hwang, Choi et al. 1991; Baum 1991). However, the selection in these examples are based on human designed algorithms, and it is unclear what goal is prioritised in the selection. MacKay (1992a) formalises this question in active data selection by defining objective functions and deriving the corresponding data selection criterion. Here, each objective function with a different goal, leads to a different data selection criterion. Two important conclusions follow from this work. Firstly, in order to actively determine the next sampling point while balancing exploration and exploitation, a probabilistic method is required to calculate the corresponding uncertainty, i.e., error bars should be available. Secondly, when maximizing the total information gain of the entire input domain, the sampling points end up uniformly throughout the domain, similar to the distance based criteria, minimax, as discussed before. This is not unexpected, as it is based upon the assumption that the uncertainty grows proportionally to the distance. MacKay (1992b) also extends these ideas to classification problems.

Due to the probabilistic nature of Gaussian processes, it is a logical choice when determining where the next sample should be (Rasmussen and Williams 2006). Once trained, these function approximators provide both the expected value and variance of any given point in the input domain. These data can be used in an objective function as previously mentioned. For example, the framework has been implemented to determine which sensors provide most significant information, prioritising the activation and recovery of sensors (Rogers et al. 2008).

Much literature related to the application of Gaussian processes in the design and analysis of computer experiments is available, see for example, Santner, Williams and Notz (2003). However, in order for the implementation of such models to be beneficial to the overall modelling effort in terms of computational cost, the evaluation of the objective function should be, in general, significantly more expensive than the evaluation of such a function approximator.

When focussing on the expected value only, ignoring the variance which provides the uncertainty, it is not possible to *explore* using the function approximator, however, it is still possible to *optimise*. Function approximators without uncertainty propagation range from high performance, high complexity, at a significant computational cost (e.g., deep learning neural networks) to basic interpolation methods (e.g., piecewise linear, cubic, and linear regression).

A schematic, where the decision where to sample is made by an information agent based on the available information, is presented in figure 5.1. In this figure, the information

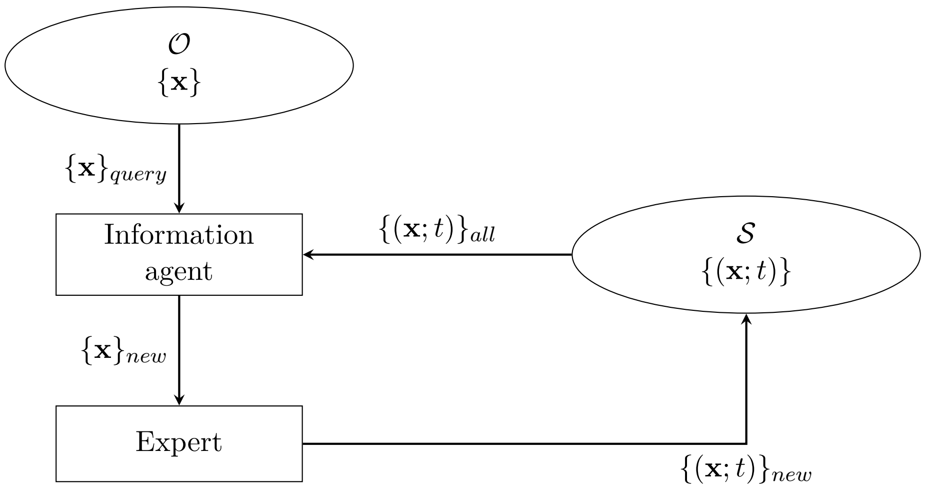


Figure 5.1: A sequential approach to uncertainty sampling via an information agent.

agent selects a point $\mathbf{x}$ from the pool of available points in the input domain $\{\mathbf{x}\}$. Once a point is sampled, it is added to the set of sampled points $\{(\mathbf{x};t)\}$, which is available to the information agent to select a new point. The expert in this system is considered a costly evaluation.

The next section introduces the problem at hand; the evaluation of security threats in urban and rural environments. Thereafter, an initial sampling plan and a scheme for active data selection suitable for this specific problem are proposed.

## 5.2 Introduction to the problem at hand

This section introduces the problem of determining from which locations in a terrain a given target is within the line-of-sight. This is a specific example of the more general case where an unknown function is evaluated in a given surface map. Typically, the properties of such functions are:

- The input space of the function is limited to the surface of the terrain – for example, a rocket or mortar is launched from the surface.
- The function itself has a relation to the landscape – both visibility and rockets are blocked by buildings.

- The function output holds a high correlation with the elevation of the terrain – higher surface height reduces the chances of any blocking occurring.

The results in this chapter are produced using 32 case studies, consisting of 16 surface maps from the London area and 16 surface maps from the area surrounding Southampton, each covering a $2 \times 2$ km$^2$ area. Examples of the surface belonging to a case study in the London and Southampton area are visible in figure 5.2 and figure 5.5 respectively. The height as displayed in these figures, is the height above mean sea level. The corresponding collision margins, visible in figures 5.3 and 5.6, represent the minimal clearance between the line-of-sight to a target positioned at the southwest corner of the map and the surface. Hence, a negative margin corresponds to the target not being visible from that location, and vice versa. The visibility maps, marking the areas with positive (visible) or negative (not visible) collision margins are presented in figures 5.4 and 5.7.

When considering terrain mapping data with a resolution of $1 \times 1$ m$^2$, a $2 \times 2$ km$^2$ map results in 4 million evaluations. When performing a strategic analysis, map sizes of $30 \times 30$ km$^2$ are not uncommon (Cunning Running Software Ltd. 2017). Such large scale analysis results in a total of 900 million evaluations. Therefore, even when assuming each evaluation only takes one second, evaluating the entire map would take over three months.

An important conclusion that can be drawn from this information is that it is not the cost of the evaluation that is the problem, it is the enormous amount of evaluations. For this reason, methods to avoid having to sample everywhere are investigated and evaluated. Furthermore, methods to prioritise *likely dangerous* regions (subsections of the complete map) in the evaluation are considered – such a method is helpful when preliminary results are desired quickly, whist the full computation may continue in the background. The next section proposes several methods to tackle these two objectives.

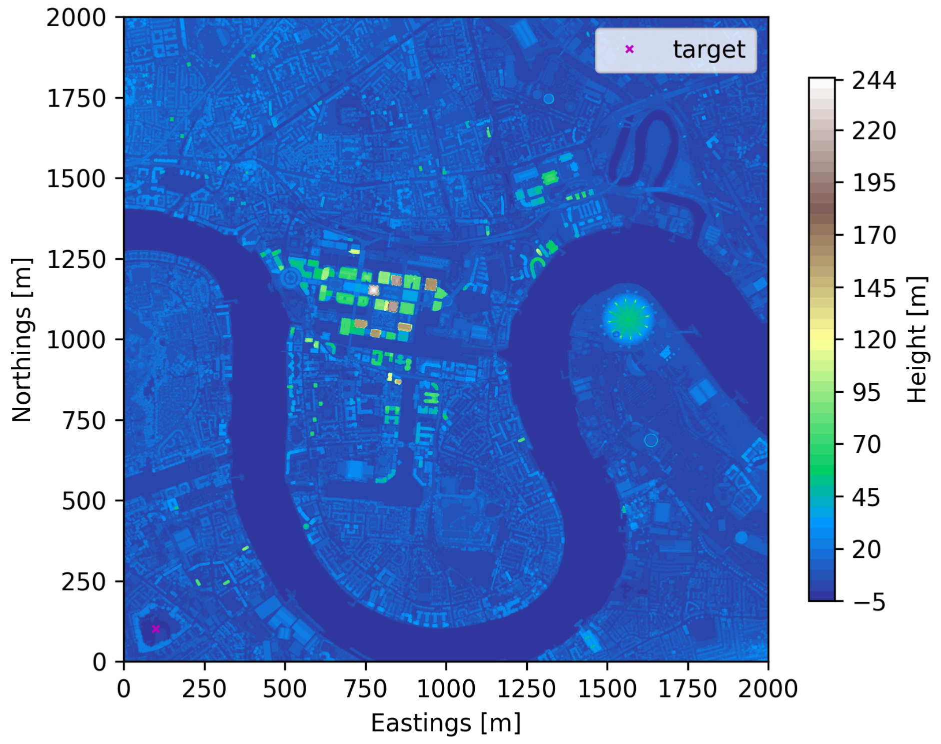


Figure 5.2: The surface map of a case from the London area, with the River Thames clearly visible.

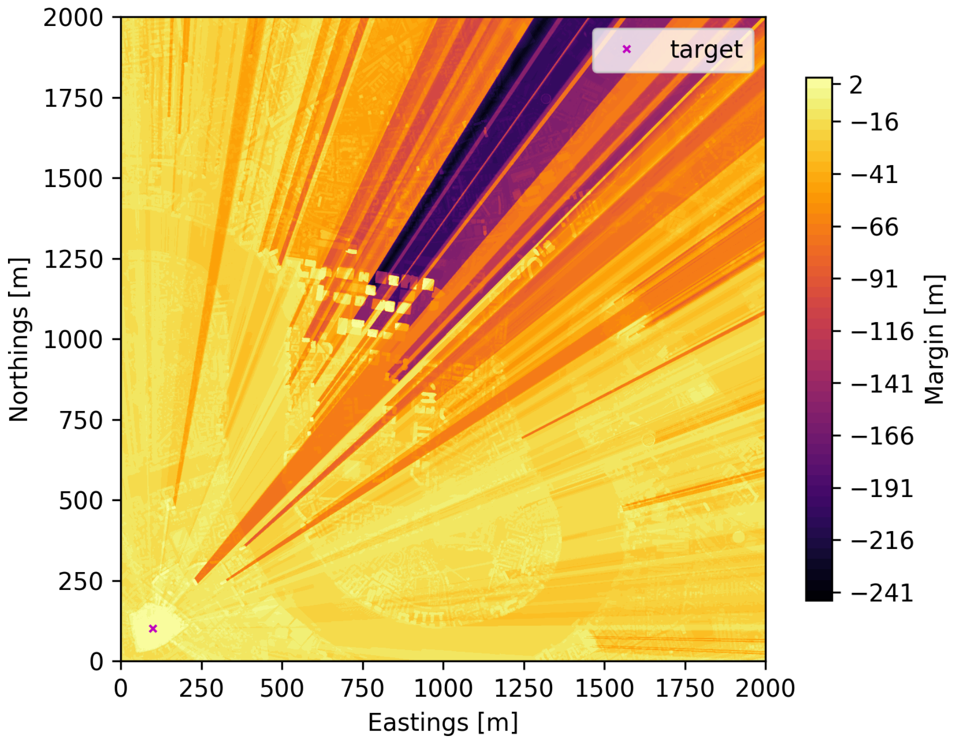


Figure 5.3: The collision margin map of a case from the London area, corresponding to the surface map from figure 5.2.

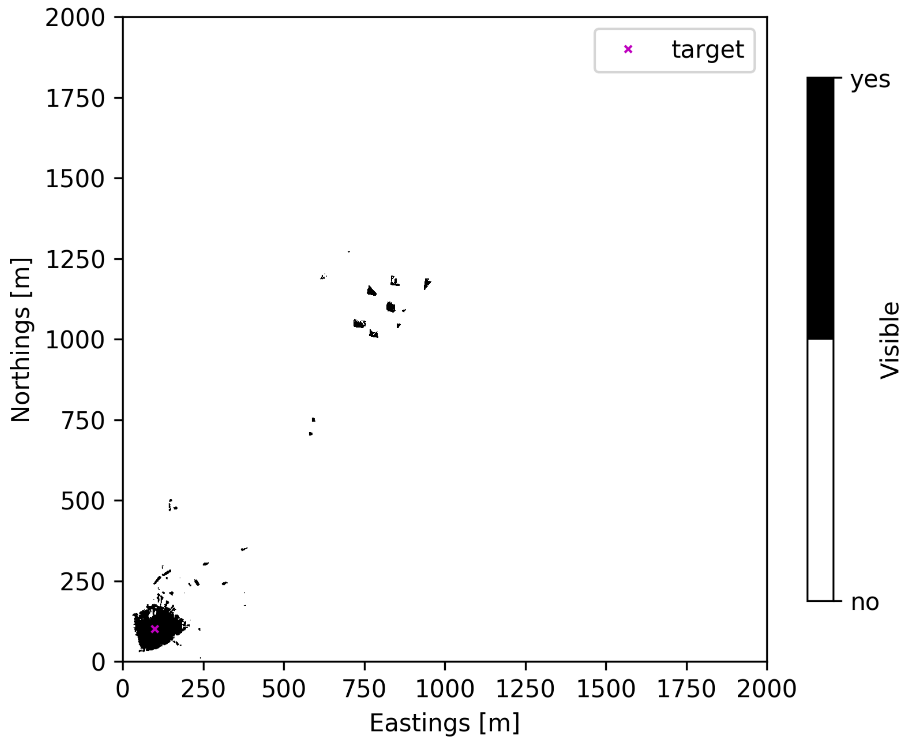


Figure 5.4: The visibility map of a case from the London area, corresponding to the surface map from figure 5.2.

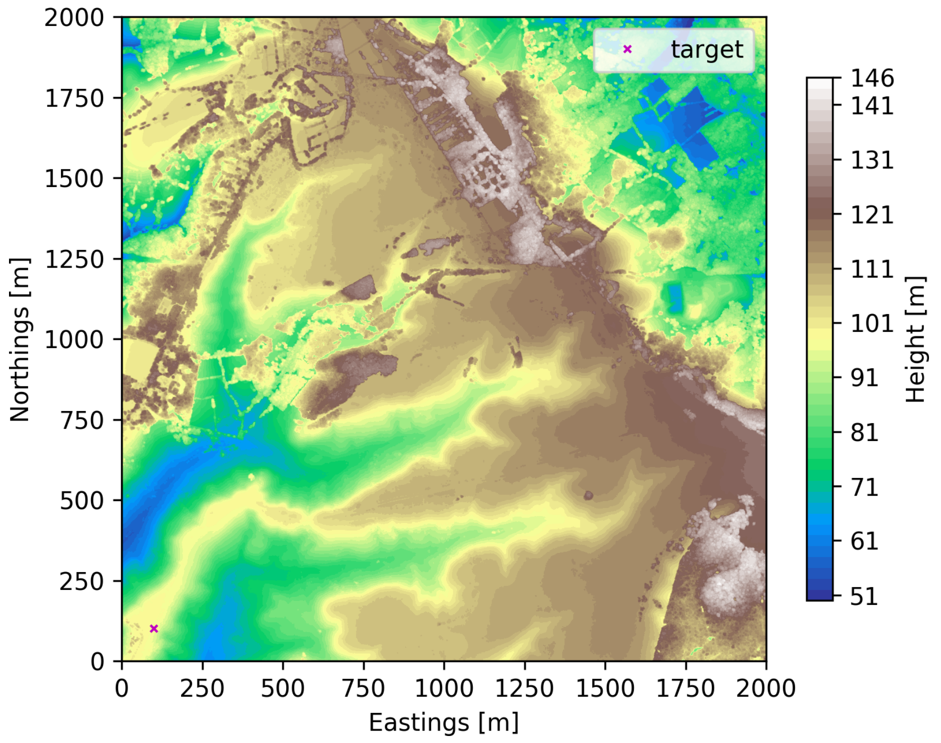


Figure 5.5: The surface map of a case from the Southampton surrounding area, specifically, the northern part of the New Forest.

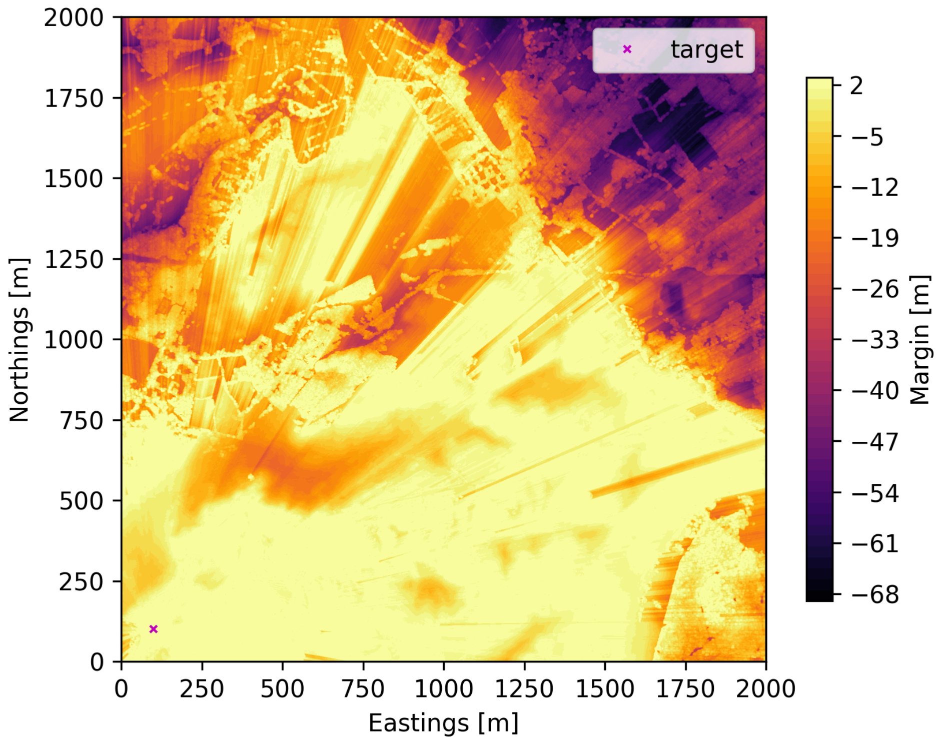


Figure 5.6: The collision margin map of a case from the Southampton area, corresponding to the surface map from figure 5.5.

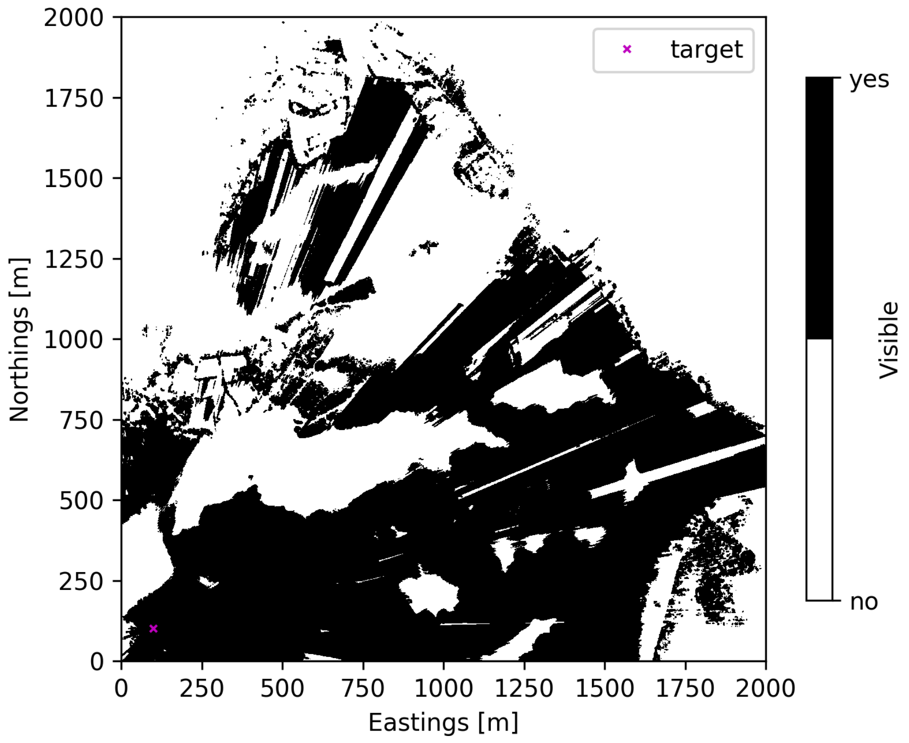


Figure 5.7: The visibility map of a case from the Southampton area, corresponding to the surface map from figure 5.5.

## 5.3 Methodology

Given the background of existing methods provided by section 5.1 and the constraints of the problem described in section 5.2, this section introduces methods aimed at reducing the overall computational cost and prioritising the identification of dangerous regions.

First, section 5.3.1 proposes a method to reduce the total number of evaluations by only sampling once in a $n_{grid} \times n_{grid}$ space, effectively lowering the sampling resolution. Then, section 5.3.2 introduces a method that divides the terrain mapping into regions based on their surface height. Assuming a strong correlation between surface height and the collision margin, the hypothesis is that this reduces the total number of dangerous regions. Finally, due to the scale of the problem (i.e., the large number of points to be evaluated), a linear interpolation is used to prioritise *dangerous* regions, ignoring probabilistic methods due their substantial computational complexity. A detailed description of this, and the initial sampling scheme, is presented in section 5.3.3.

### 5.3.1 Reducing the sampling resolution

By far the most basic method to reduce the associated computational cost with evaluating an entire map, is to not evaluate all the points. Therefore, given a certain grid positioned on the surface, referred to as a grid square, only a single point is evaluated. In this dissertation grid square sizes of $10 \times 10$ m$^2$ and $50 \times 50$ m$^2$ are considered. The goal of this section is to propose a method which selects the most likely position to be dangerous within such a grid square, using only the surface map. An example of four grid squares build from 400 data points using $n_{grid} = 10$, resulting in $10 \times 10$ m$^2$ grid squares, is visible in figure 5.8. In the case of the reduced resolution, each individual grid square will only have one evaluation.

Based on the sampling theory, when assuming no prior knowledge and maximizing the overall information obtained, placing the points in the middle of each grid square provides most information. However, prior knowledge is available; a greater collision margin is expected with an increase in altitude. This information follows from the

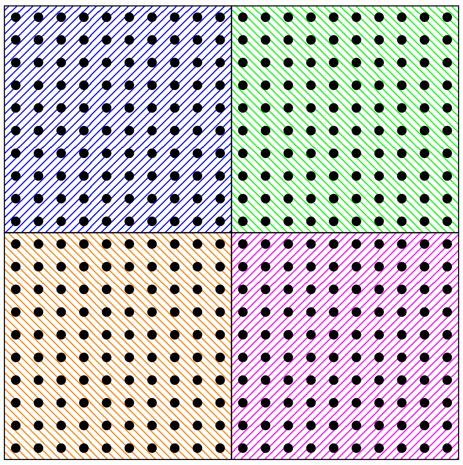


Figure 5.8: An example of building *grid squares* using $n_{grid} = 10$, reducing the original 400 data points (dots) to four grid squares (hatched squares). In each grid square, only one data point will be evaluated.

statements related to the typical properties of the problem introduced in section 5.2. More importantly, the surface map is readily available at no computational cost.

In order to select a location between the centre of the grid square and the highest point, the following approach is taken. First, the difference vector $\mathbf{d}$ for all grid points $\mathbf{x}$ in the set $\{\mathbf{x}\}$ are calculated:

$$\mathbf{d}(\mathbf{x}) = \mathbf{x} - \mathbf{x}^{centre} \tag{5.1}$$

where $\mathbf{x}^{centre}$ represents the grid point in the middle, with

$$\mathbf{x} = [\text{ Eastings Northings Height }]^{\mathsf{T}} \tag{5.2}$$

and

$$\mathbf{d} = [\ d_x\ d_y\ d_z\ ]^{\mathsf{T}} \tag{5.3}$$

Then, each difference vector $\mathbf{d}$ is modified according to the following equation:

$$\mathbf{d}^* = [\ d_x\ d_y\ p \cdot d_z\ ]^{\mathsf{T}} \tag{5.4}$$

where $p$ is a parameter that balances the position between the centre ($p = 0$) and the highest point ($p = \infty$). This balancing occurs via the following equation:

$$\mathbf{x}^{select} = \arg\min\left(|\mathbf{d}^*(\mathbf{x})|\right) \tag{5.5}$$

this states that the location $\mathbf{x}^{select}$ is selected from the set $\{\mathbf{x}\}$, of which $\mathbf{x}^{select}$ provides the smallest $|\mathbf{d}^*|$.

The performance of this approach is evaluated in section 5.4.1. The next section presents an algorithm that combines grid squares into *low surface height variance* regions, an attempt to create regions with a strong correlation between lateral position (eastings and northings) and the surface height.

### 5.3.2 Forming regions

The method presented in this section attempts to combine the grid squares as introduced in the previous section in larger regions. In a similar fashion as the grid squares were constructed, it is possible to form regions by dividing the complete area into equal spaced areas, this will result in square regions. In turn, each region can be described by several parameters. In this dissertation the centroid in lateral position (i.e., the average eastings and northings), and the maximum surface height of a specific region are considered. These parameters can be used to predict where the dangerous regions are located, how this is done is explored in the next section. Furthermore, it is possible to extend this set of parameters with, for example, the mean and variance of the surface height. However, initial testing indicates that extending the parameter space with these parameters does not show an improvement. This corresponds with results seen later in this chapter, where a strong correlation between the maximum surface height and the maximum collision margin is found.

Another approach to form regions is explored here, starting from the *grid squares* introduced in the previous section. First, each grid square is reduced to the average eastings $(x)^{grid}_{avg}$, average northings $(y)^{grid}_{avg}$, the maximum surface height $(z)^{grid}_{max}$, and the variance in $z$ direction $(z)_{var}^{grid}$. Here, the average surface height $(z)_{avg}^{grid}$ is replaced by the maximum surface height $(z)_{max}^{grid}$ such that each grid square is considered to be as conservative as possible (i.e., it is all about the highest elevation of the surface).

Next, regions are generated by combining grid squares, and when assuming independent univariate Gaussian distributions (i.e., the covariance matrix of the multivariate Gaussian is diagonal), the lateral averages $(x)_{avg}^{region}$ and $(y)_{avg}^{region}$, the surface height average $(z)_{avg}^{region}$, and surface height variance $(z)_{var}^{region}$ of the combination of $N$ grid squares are given by:

$$(x)_{avg}^{region} = \frac{(x)_{avg}^{grid,1} + \cdots + (x)_{avg}^{grid,N}}{N} \tag{5.6}$$

$$(y)_{avg}^{region} = \frac{(y)_{avg}^{grid,1} + \cdots + (y)_{avg}^{grid,N}}{N} \tag{5.7}$$

$$(z)_{avg}^{region} = \frac{(z)_{max}^{grid,1} + \cdots + (z)_{max}^{grid,N}}{N} \tag{5.8}$$

$$\begin{aligned}(z)_{var}^{region} = & \frac{(z)_{var}^{grid,1} + \left[(z)_{avg}^{grid,1}\right]^2}{N} + \cdots \\ & + \frac{(z)_{var}^{grid,N} + \left[(z)_{avg}^{grid,N}\right]^2}{N} - \left[(z)_{avg}^{region}\right]^2\end{aligned} \tag{5.9}$$

As the goal is to obtain regions with a low variance, the score associated with each region is taken to be the region's variance $(z)_{var}^{region}$.

The algorithm starts by initialising a set number of regions $n_{regions}$ uniformly distributed with a single grid square. Each region is then expanded in turns, where the region with the lowest score (i.e., the lowest variance in surface height) takes priority. For the expansion, a region can choose from valid neighbours according to two examples given in figure 5.9. From the valid neighbours, the grid square with the lowest *weighted* distance $d_w$ is expanded. The *weighted* distance to grid square $n$ is calculated according to:

$$\begin{aligned}d_w = & ((x)_{avg}^{region} - (x)_{avg}^{grid,n})^2 \\ & + ((y)_{avg}^{region} - (y)_{avg}^{grid,n})^2 \\ & + (w \cdot (z)_{avg}^{region} - (z)_{max}^{grid,n})^2\end{aligned} \tag{5.10}$$

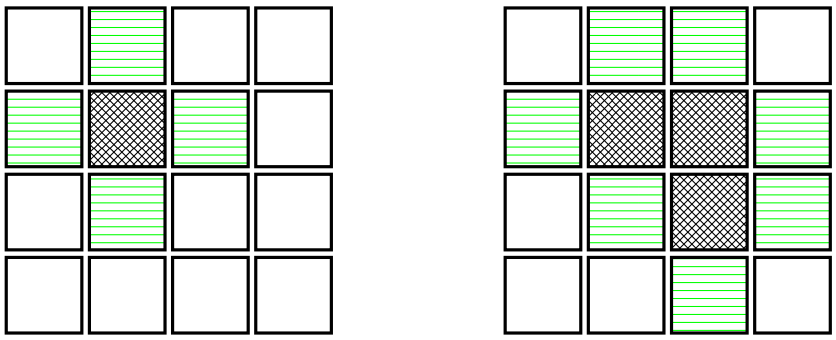


Figure 5.9: Examples of valid neighbours (horizontal lines green), of the region identified with black crosshatch.

where a weight $w$ has been applied to the surface height. This weight $w$ allows the algorithm to prioritise between the lateral coordinates (eastings and northings) using $w = 0$, and surface height using $w = \infty$. The entire procedure for generating these *free-shaped* regions is shown in algorithm 1. The results are discussed later in section 5.4.2, and in the next section an approach to evaluating the dangerous regions first is introduced.

**Algorithm 1** Algorithm to assign regions.

```
1: procedure ASSIGN REGIONS
2:     initialise uniform distribution
3:     while grid squares not assigned to a region do
4:         score regions                                ▷ scored via equation (5.9)
5:         for the regions do                           ▷ lowest score takes priority
6:             expand region by one grid square         ▷ chosen via equation (5.10)
7:         end for
8:     end while
9:     return assigned grid squares
10: end procedure
```

### 5.3.3 Dangerous regions first

The previous section introduced regions, and how each region of the map can be described in terms of parameters. This section utilises those parameters to *predict* which region is most likely to contain the most dangerous launch locations. While the words *predict* and *most likely* indicate a preference to a probabilistic approach, such an approach is not implemented[1]. This choice is related to the trade-off to be made when

[1]Experiments using a user-friendly Python implementation of kriging called pyKriging (Paulson et al. 2017) have been performed. However, the computational cost involved deems this approach impractical.

forming the regions. Dividing a terrain mapping in many small regions provides a better separation between the dangerous areas and the safe areas. This allows the method to perform better in terms of prioritising the dangerous areas. However, in the extreme case where each *grid square* is considered a region, a large portion of the computational budget is spent on prioritising, and not the evaluation itself. For this reason, the relatively low-cost (in terms of the computational budget required) piecewise linear function approximator is considered in this section. This allows the use of many smaller regions, while keeping the allocated computational cost associated with the prioritising low.

The initial sampling plan is generated using an equidistant grid in a lateral direction (eastings and northings). Latin hypercube design is not implemented here as it is already known that both dimensions are equally important and would only add complexity without adding benefits (i.e., there is no reason to sample eastings more than northings, and vice versa). Furthermore, the terrain mapping is relatively flat, meaning the distance covered in eastings and northings is much larger than the variation in surface height. Therefore, optimising using the maximin distance design criterion is expected to show little deviation from this equidistant grid, other than the occasional shift towards a high or low point. Again, keeping in mind a large number of regions to evaluate and the limited computational budget available to prioritise, the sampling plan is not optimised using the surface data.

Instead, the surface height information is transferred to the piecewise linear function approximator. Recall from the previous section that from each region the average lateral position and the maximum surface height are calculated and stored. The use of a linear function approximator is also motivated by the shape of the objective function, as seen in figures 5.3 and 5.6. These results show linear lines originating from the target, with tall buildings disrupting the line of sight. Two versions are considered in this work; a two-dimensional piecewise linear function approximator that uses the average lateral position only, and a three-dimensional piecewise linear function approximator that uses the average lateral position and the maximum surface height. For each evaluation of a region, the maximum collision margin is returned. This is a conservative approach, as when one grid square within a region is considered dangerous, the entire region is considered dangerous.

Lacking any probabilistic information, the approach is straightforward. From the regions not yet evaluated, the expected collision margins are predicted using the function approximator (either the two- or three-dimensional variant). The region with the largest collision margin is then evaluated next. This process is repeated until all regions are evaluated. This last step is a direct consequence of having no probabilistic information, which is required to assert *with certainty* that a specific region is safe and does not need to be evaluated.

The results of applying this approach are presented in section 5.4.3. The next section evaluates the methods as presented in this section.

## 5.4 Results

This section evaluates and tests the methods proposed in the previous section. As stated in section 5.2, in total there are 32 cases, each covering a $2 \times 2$ km$^2$ area. These cases are used to evaluate the methods. First, the reduction of the sampling resolution and the positioning of the sample is investigated. Here the sample position makes use of the available surface height data. Next, the effects of *free-shaped* regions are explored. Instead of using square regions, these regions have been given the freedom to modify themselves to produce regions of similar surface height. Finally, a method that prioritises dangerous regions via a piecewise linear function approximator is evaluated.

### 5.4.1 An evaluation of reducing the sampling resolution

This section investigates the reduction of the sampling resolution, according to the methodology proposed in section 5.3.1. As a result of a $n_{grid}$ reduction, only one point in every $n_{grid} \times n_{grid}$ m$^2$ grid square is evaluated. The position is dictated by the parameter $p$, where for $p = 0$ each sample is located in the centre of each $n_{grid} \times n_{grid}$ m$^2$ grid square, and for $p = \infty$ each sample is located at the highest point of each $n_{grid} \times n_{grid}$ m$^2$ grid

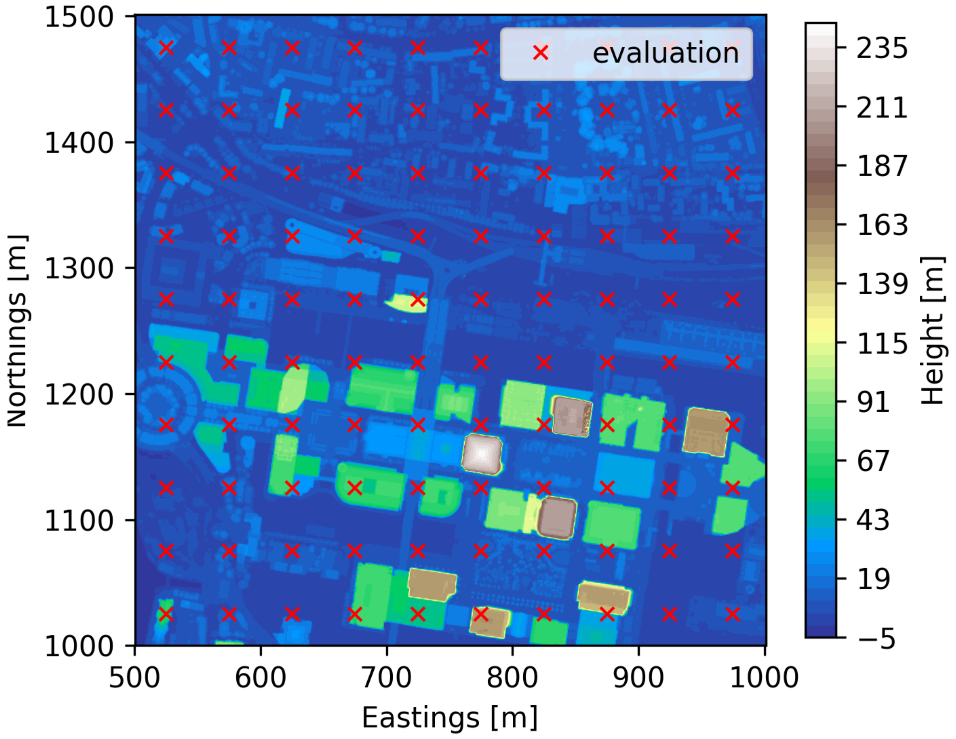


Figure 5.10: The surface map with samples positioned in an equidistant grid ($p = 0$).

square. This is illustrated by figures 5.10 and 5.11, where the two extremes are shown. A larger value of $p$ moves the evaluations towards a higher point, starting from the centre. The goal is to identify the dangerous locations, or in this case, the locations from where the target is visible. Hence, the results from figures 5.12 and 5.13 show that $p = 100$ is successful at identifying these areas.

This process is repeated for all 32 cases, at a range of values for the position $p$. These results, visible in figures 5.14 and 5.15, show that for both the $10 \times 10$ m$^2$ and $50 \times 50$ m$^2$ grid square sizes, the dangerous area is more likely to be detected when selecting the highest point. These results also indicate that reducing the sampling resolution to $10 \times 10$ m$^2$ grid squares allows, on average, 20% of the dangerous grid squares to be undetected. For $50 \times 50$ m$^2$ grid squares, this percentage increases to 25%. While this can hardly be considered a trend using only two datapoints, it stands to reason that the correct positioning of the samples becomes increasingly important for larger grid squares.

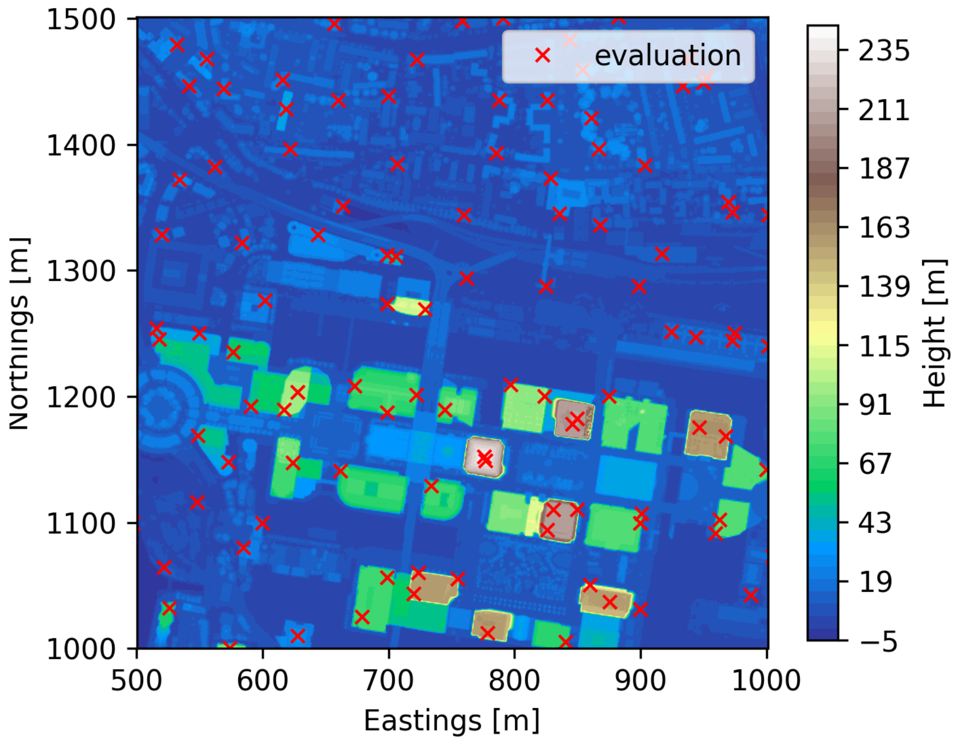


Figure 5.11: The surface map with samples positioned at the highest point ($p = 100$).

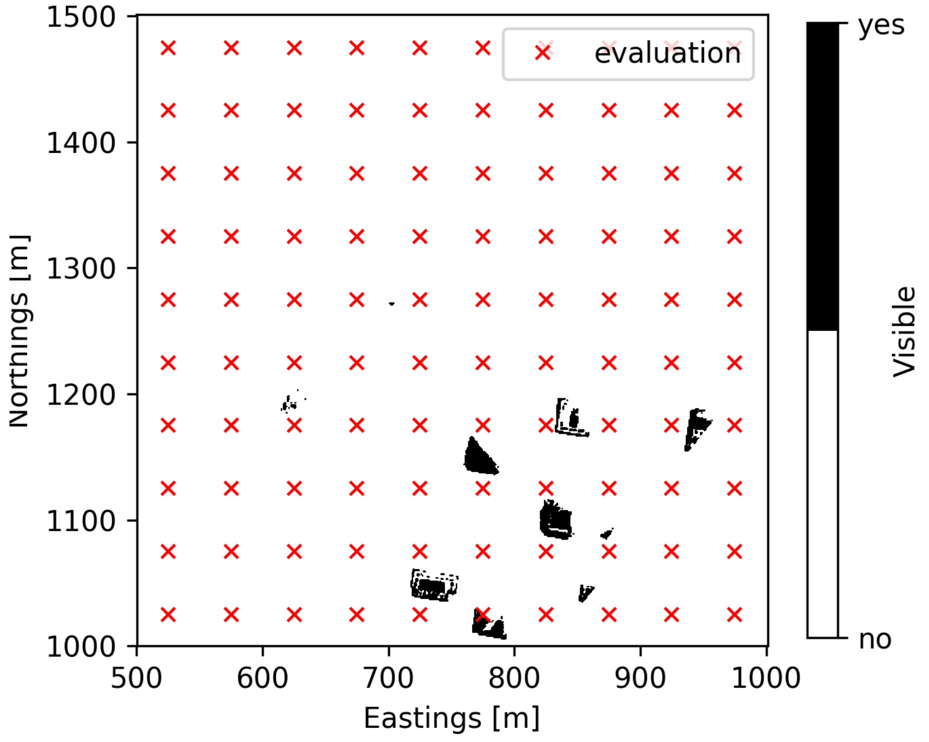


Figure 5.12: The visibility map with samples positioned in an equidistant grid ($p = 0$).

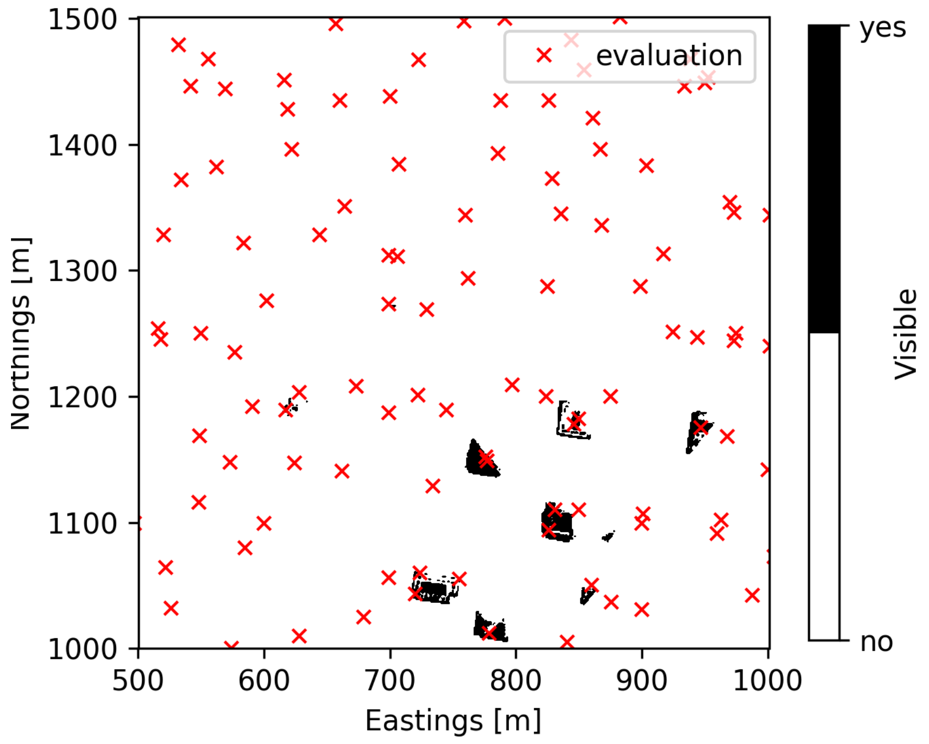


Figure 5.13: The visibility map with samples positioned at the highest point ($p = 100$).

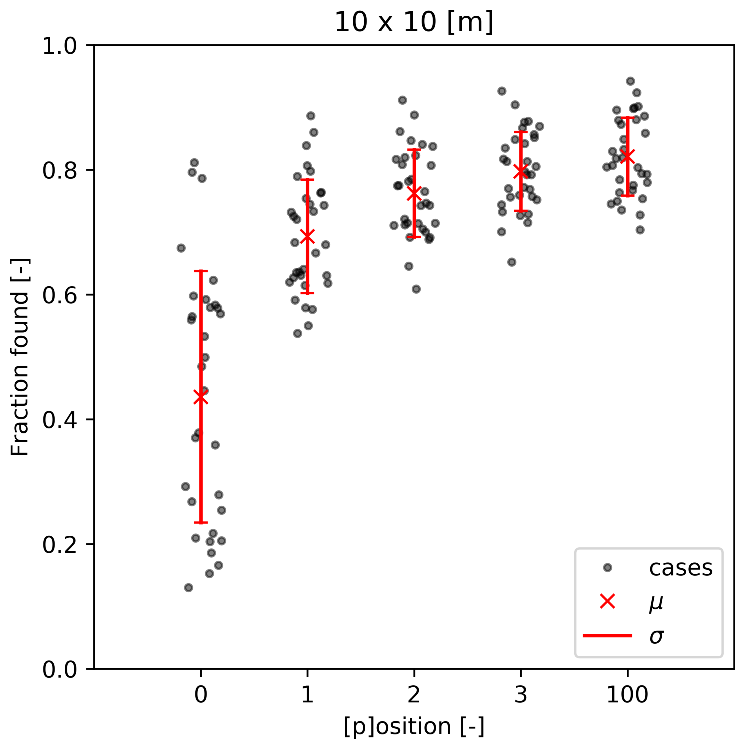


Figure 5.14: An overview of the results when evaluating once in a $10 \times 10$ m$^2$ grid square.

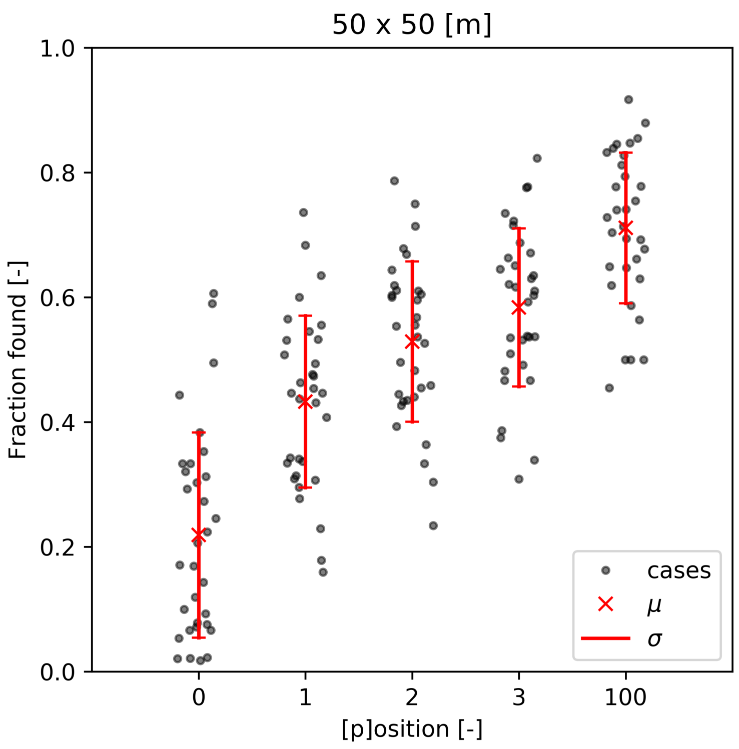


Figure 5.15: An overview of the results when evaluating once in a $50 \times 50$ $m^2$ grid square.

### 5.4.2 Evaluation of free-shaped regions

This section applies the method as introduced in section 5.3.2, and evaluates the performance. It starts with a qualitative check, demonstrating the effects of the region forming algorithm. Then, using 32 different cases, the reduction in total number of dangerous regions is evaluated.

Figures 5.16 to 5.18 show the results of forming 400 regions for various values of the weight $w$ as used in $d_w$ as calculated via equation (5.10). For a weight $w = 0$, the lateral dispersion is weighted more heavily than the height while still prioritising regions with a low height variance. This produces a *mostly square grid* as visible in figure 5.16. For $w = \infty$, the focus is on the height only, losing any consideration to keeping a more organised shape. This is demonstrated by the $w = 100$ as seen in figure 5.18. A balanced approach using $w = 10$ is visible in figure 5.17.

Furthermore, the number of regions plays an important role. The effects of forming more, smaller, and fewer, larger regions are shown in figures 5.19 and 5.20. These show 100 and 900 regions, formed via a weight $w = 10$.

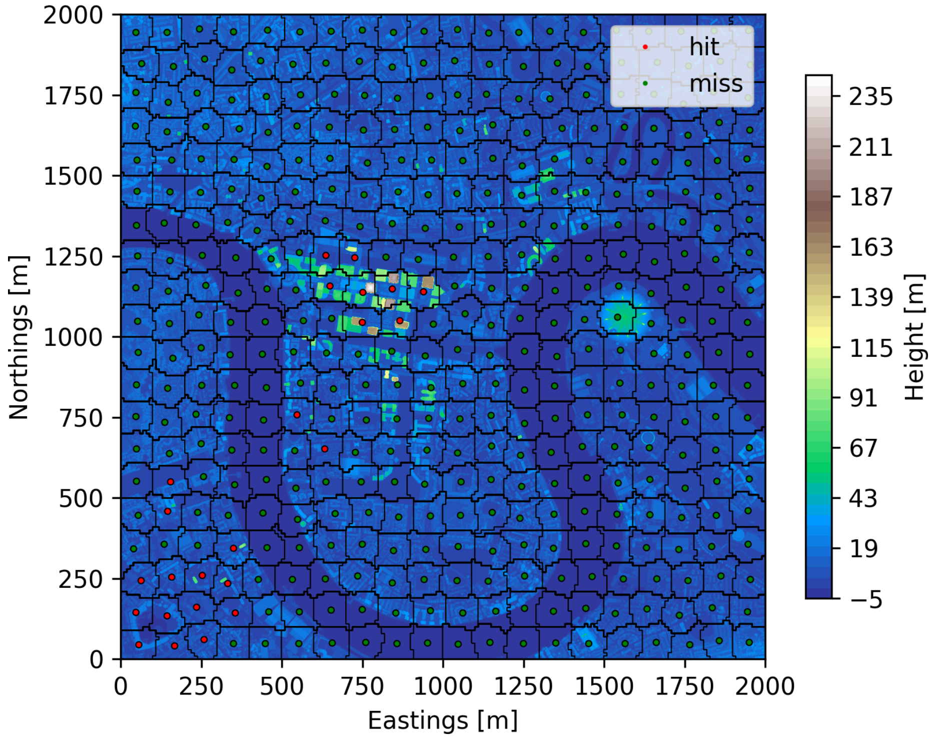


Figure 5.16: The surface map divided into 400 regions via $w = 0$.

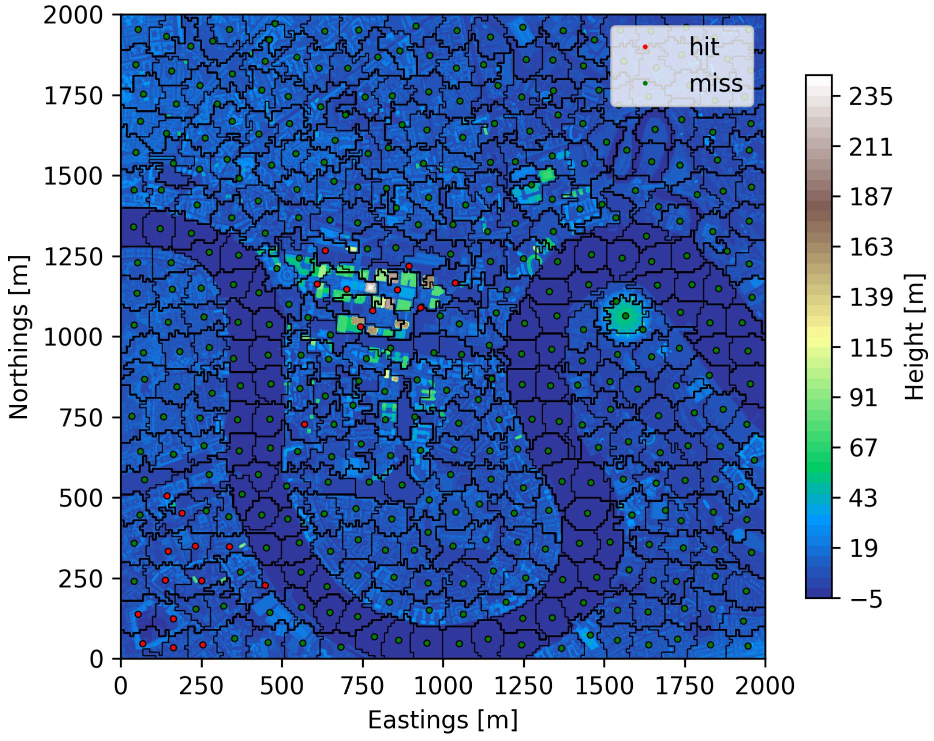


Figure 5.17: The surface map divided into 400 regions via $w = 10$.

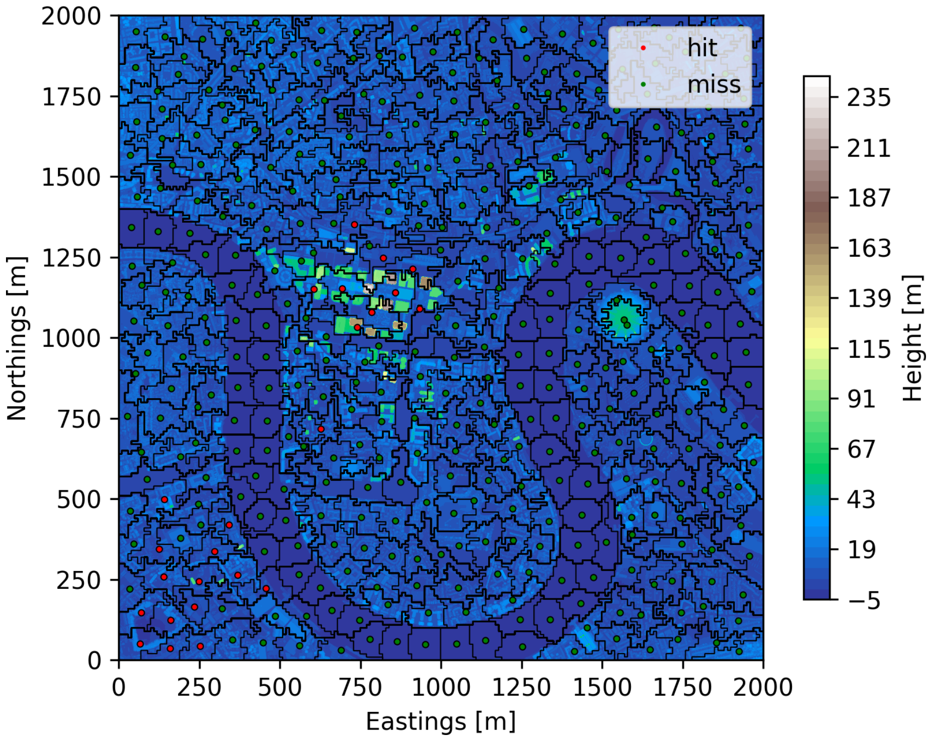


Figure 5.18: The surface map divided into 400 regions via $w = 100$.

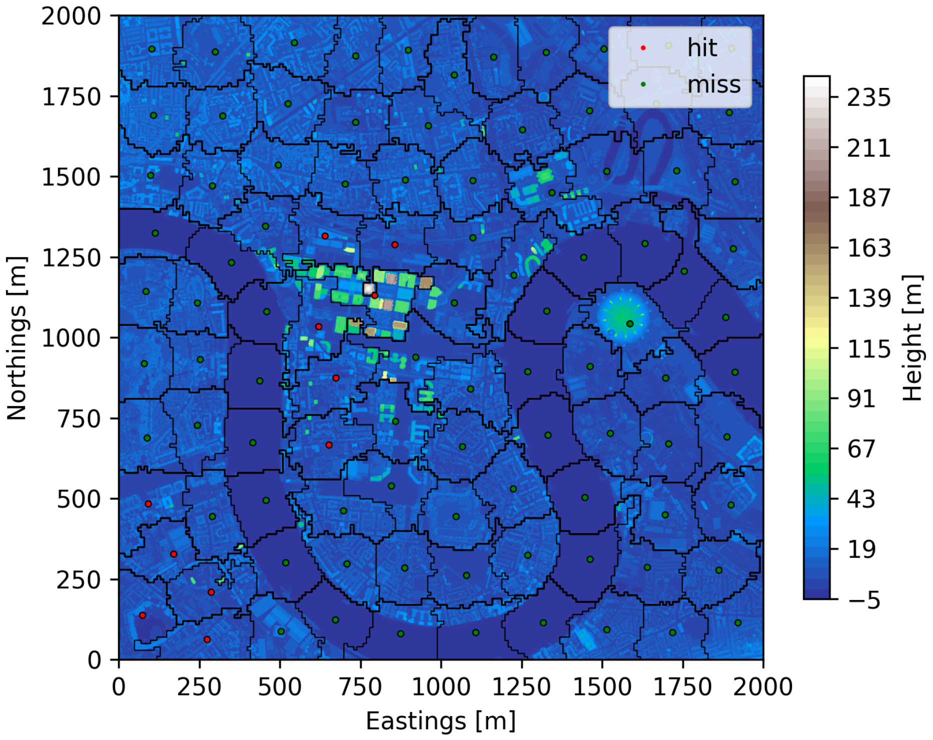


Figure 5.19: The surface map divided into 100 regions via $w = 10$.

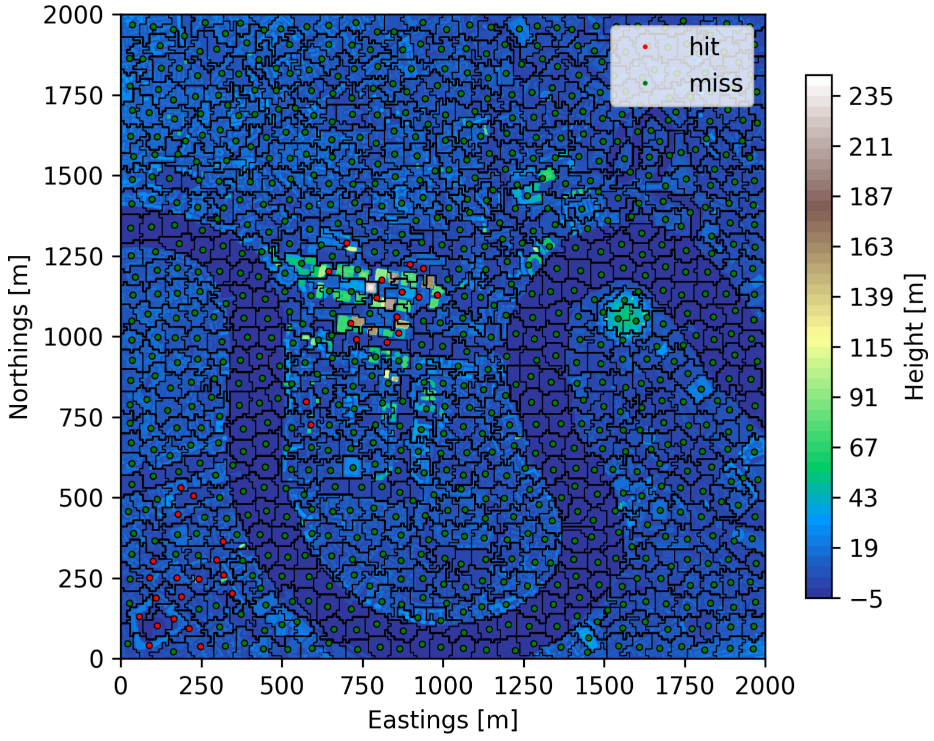


Figure 5.20: The surface map divided into 900 regions via $w = 10$.

From these results it can be concluded that the algorithm is indeed effective at following contours for a positive weight $w > 0$. This statement is supported by figures 5.17 and 5.18, where the border of the regions follow the pathway of low surface height (caused by the River Thames).

The effectiveness of the approach is measured by the decrease of the total number of dangerous regions. The idea here is, that by allowing the shape of the regions to be dictated by the surface height, the map will be better partitioned in areas of similar height. This reduction in crossover between areas is expected to reduce the total number of dangerous regions. For example, a tall building in a flat area is likely to fall within one region, and not be divided over two regions.

Figure 5.21 compares various values of $w$ against $w = 0$. Each of the 32 maps have been divided into 100 regions according to the algorithm. The results indicate that on average little changes. A similar analysis has been performed when dividing the maps in 400 and 900 regions, visible in figures 5.22 and 5.23 respectively. The best results appear to occur at a large number of small regions, however the benefits appear limited with an average reduction of 5%. Due to the limited advantage, and the computational

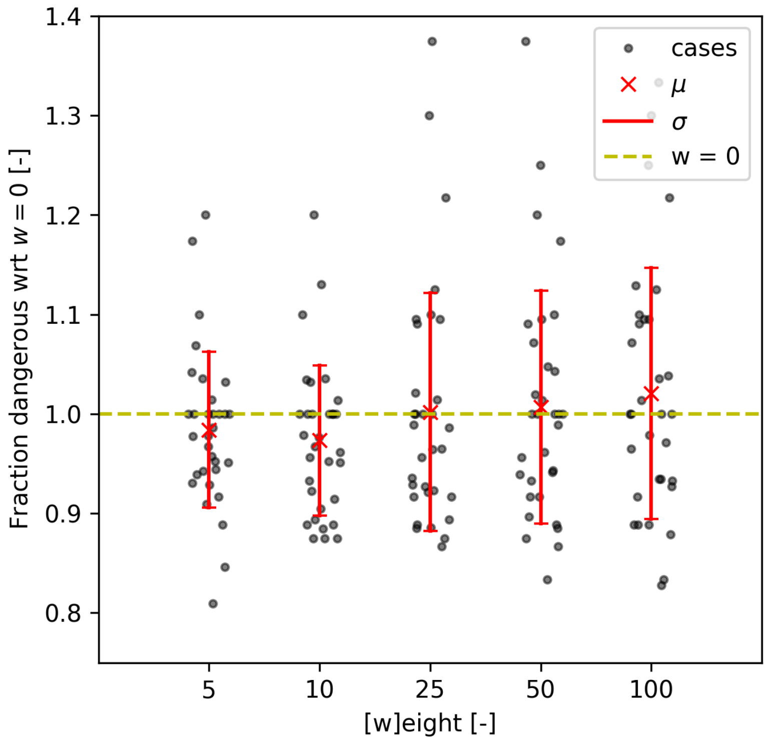


Figure 5.21: Comparing the fraction of *dangerous* regions with respect to $w = 0$ for a map divided into 100 regions.

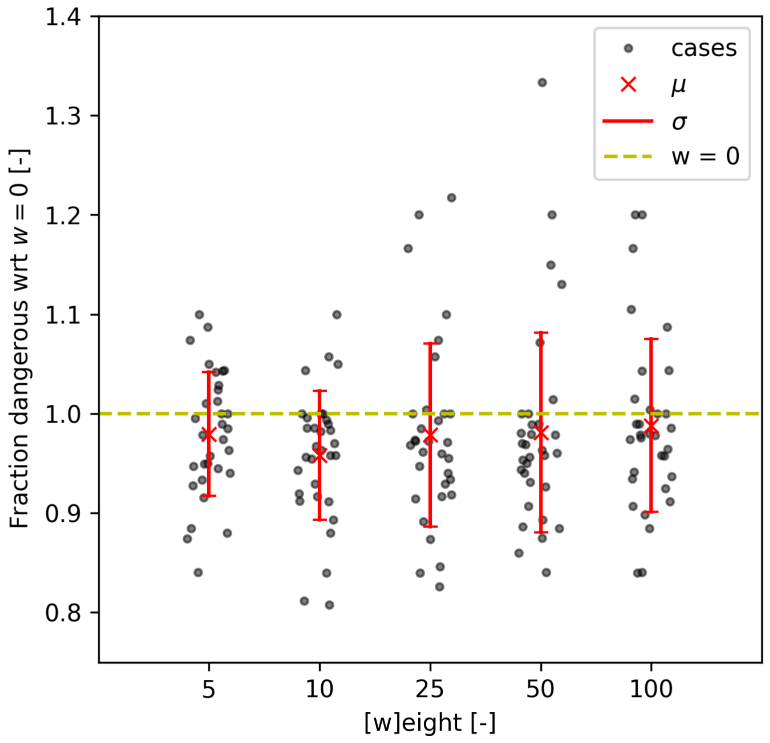


Figure 5.22: Comparing the fraction of *dangerous* regions with respect to $w = 0$ for a map divided into 400 regions.

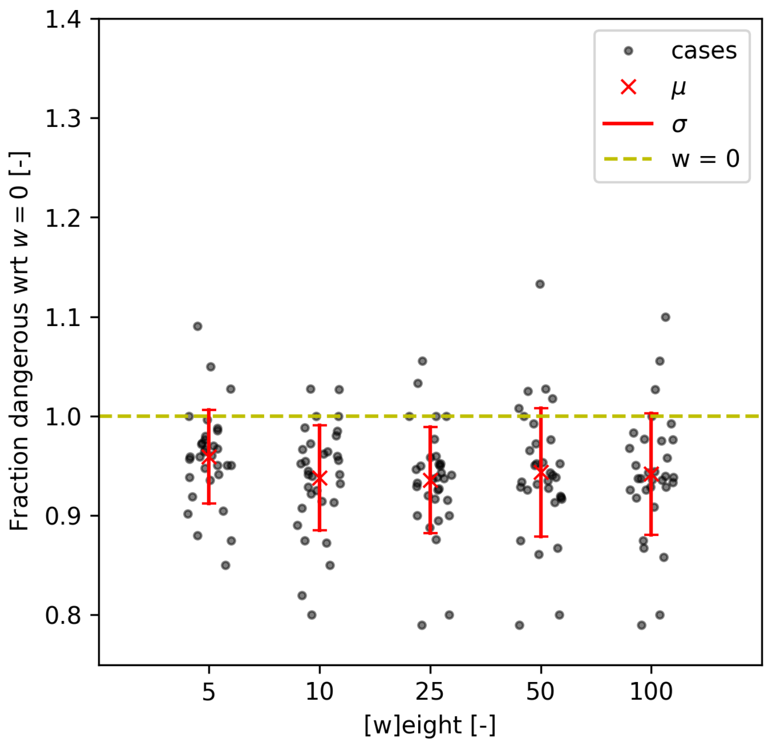


Figure 5.23: Comparing the fraction of *dangerous* regions with respect to $w = 0$ for a map divided into 900 regions.

cost involved with creating these regions, square regions are used instead. Examples of these are seen in figure 5.24 for regions sizes of $40 \times 40$ m$^2$, and in figure 5.25 for regions sizes of $50 \times 50$ m$^2$.

### 5.4.3 Evaluation of dangerous regions first

This section applies a two- and three-dimensional piecewise linear function approximator to prioritise the likely dangerous regions first. This approach is discussed in section 5.3.3 and the evaluation takes place here.

The initial sampling plan consists of an equidistant grid of sampling points. In this work, 10% of the total number of samples is included in the initial sampling plan. An example of this is presented in figure 5.26, where the surface map with the $40 \times 40$ m$^2$ regions, is visible in figure 5.24. The parameters used as the input space are consistent with the surface map, however, the two-dimensional version only uses the eastings and northings, while the three-dimensional version also takes into account the surface height.

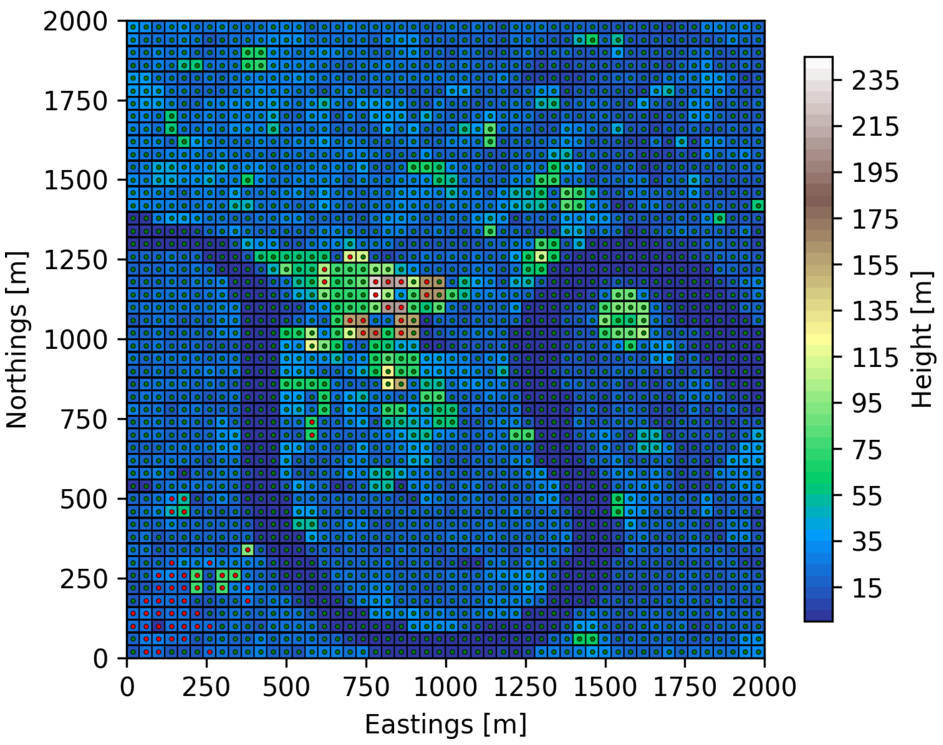


Figure 5.24: The surface map of an area divided into $40 \times 40$ m$^2$ regions.

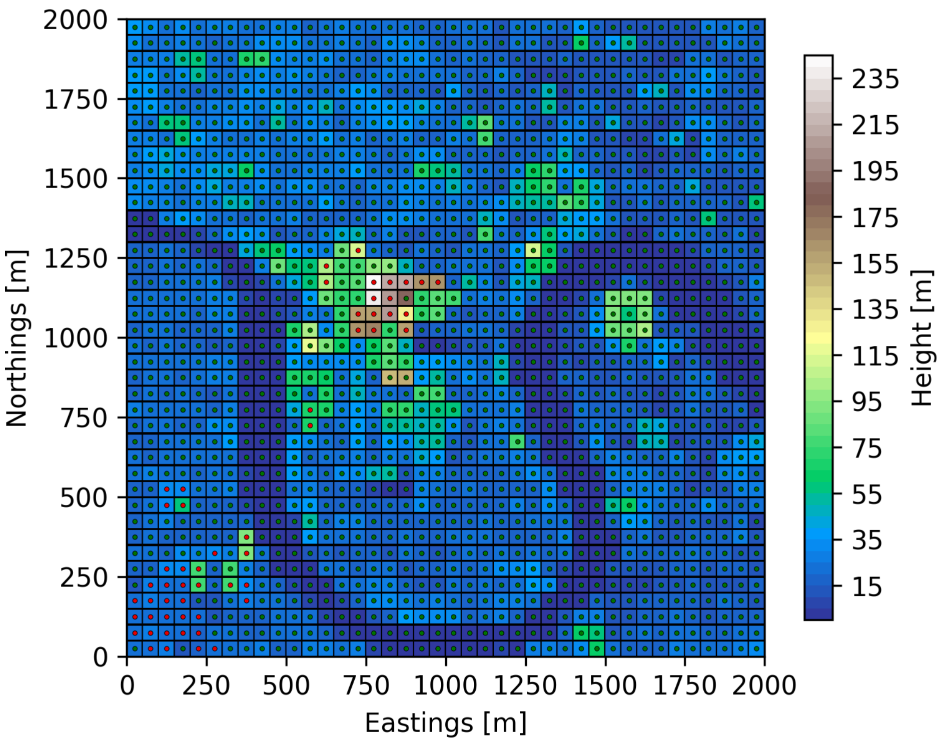


Figure 5.25: The surface map of an area divided into $50 \times 50$ m$^2$ regions.

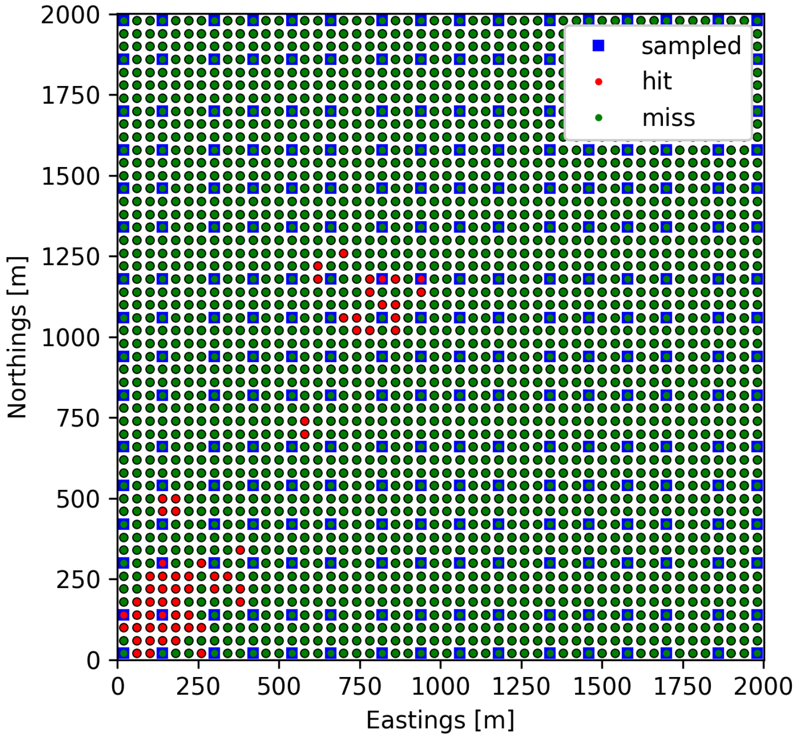


Figure 5.26: The initial sampling plan of an area divided into $40 \times 40$ m$^2$ regions.

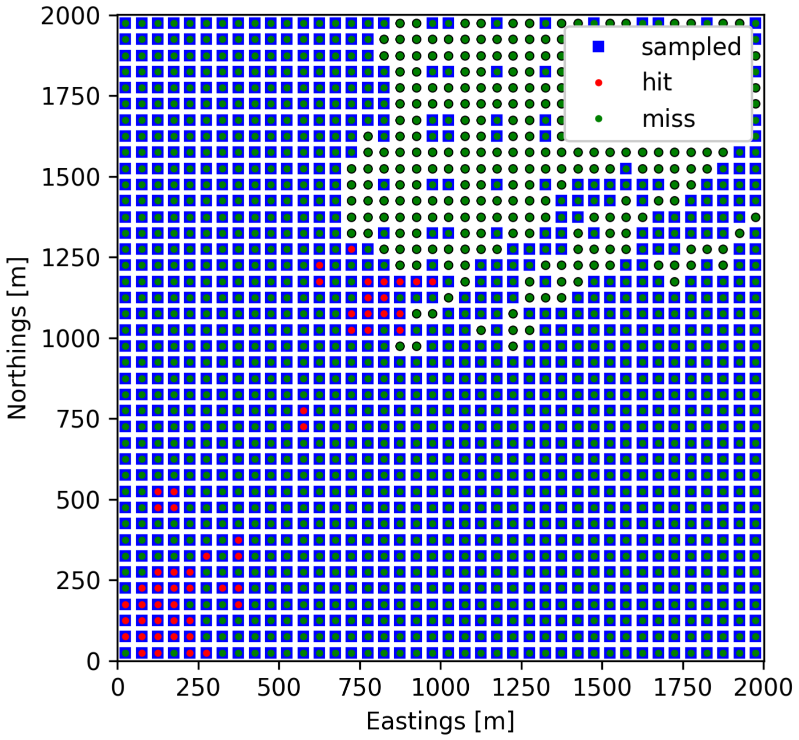


Figure 5.27: An example where the two-dimensional piecewise linear function approximator found all dangerous regions after 1050 evaluations. The map is divided into $50 \times 50$ m$^2$ regions.

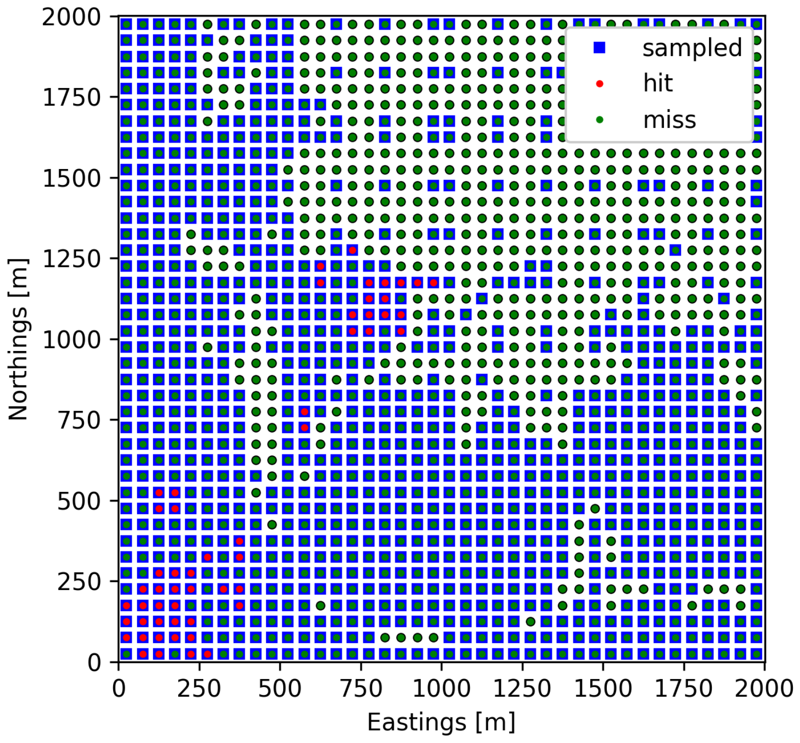


Figure 5.28: An example where the three-dimensional piecewise linear function approximator found all dangerous regions after 773 evaluations. The map is divided into $50 \times 50$ m$^2$ regions.

The advantage of including the surface height, is that the function approximator can capture the relation between the collision margin and the surface height. An example is visible in figures 5.27 and 5.28 where a two- and three-dimensional function approximator analyse the map as seen in figure 5.25. The three-dimensional version has identified all dangerous regions after evaluating 773 regions, while the two-dimensional version takes 1050. It is also possible to see that regions of low surface height (caused by the River Thames) have not been evaluated in the three-dimensional case.

For the overall evaluation, the methodology is applied to the 32 cases previously introduced. The results for dividing up the mapping in regions of $40 \times 40$ m$^2$ are visible in figure 5.29. This compares the fraction of dangerous regions, with the fraction of regions that had to be evaluated until all dangerous regions were identified. Hence, no datapoints are present at the lower left part of the diagonal. Results for the $50 \times 50$ m$^2$ and $100 \times 100$ m$^2$ regions are available in figures 5.30 and 5.31. As expected, as the regions increase in size, the fraction of regions that is considered dangerous increases as well, shifting the cases towards the right. This is a direct result of the assumption that

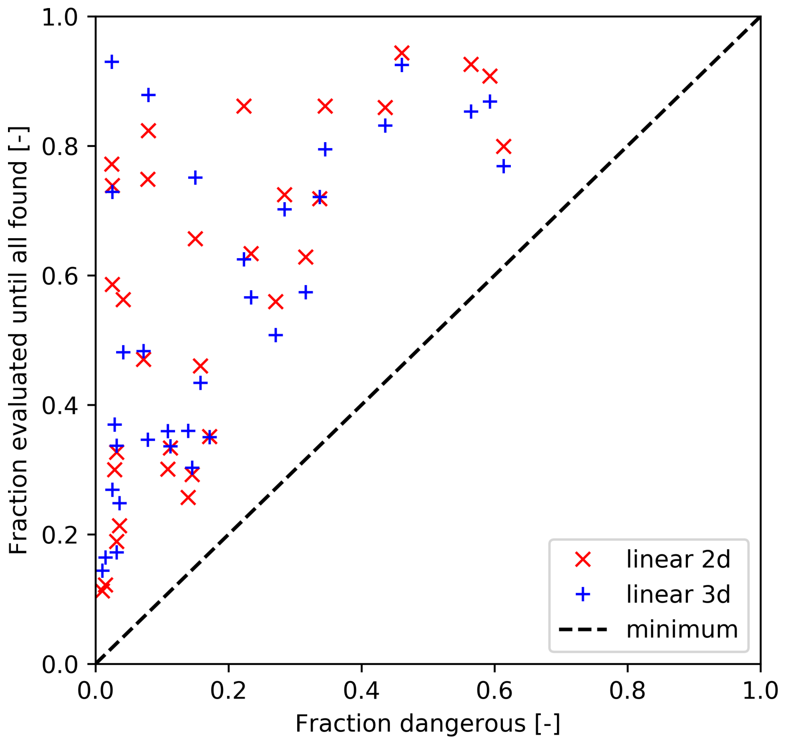


Figure 5.29: Analysing the performance of the two- and three-dimensional piecewise linear function approximator when dividing the map into 40×40 $m^2$ regions.

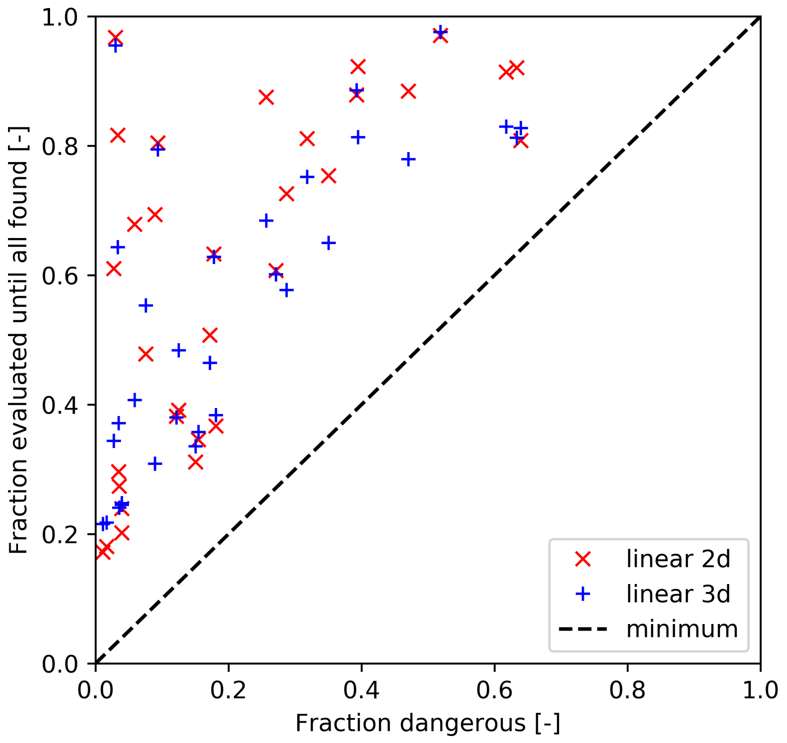


Figure 5.30: Analysing the performance of the two- and three-dimensional piecewise linear function approximator when dividing the map into 50×50 $m^2$ regions.

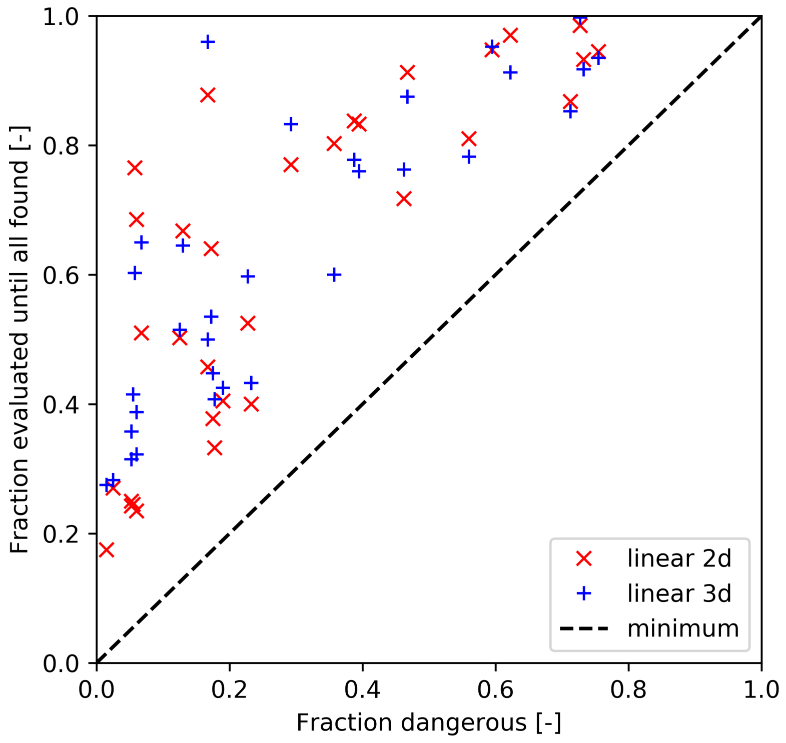


Figure 5.31: Analysing the performance of the two- and three-dimensional piecewise linear function approximator when dividing the map into $100 \times 100$ m$^2$ regions.

if one grid square within a region is dangerous, the entire region is dangerous.

Finally, these three variations on the region size, and the two function approximators are compared in figure 5.32. It shows that the three-dimensional function approximator performs slightly better overall, however, the difference is marginal. An in-depth analysis revealed that the three-dimensional version was indeed able to pick out regions with a specifically high surface height (possibly due to a tall building), but that often it was surrounded by other regions with high surface height. Hence, in the cases where the initial sampling plan detected one of these surrounding dangerous regions, it would also sample the neighbours. In a nutshell this is the primary approach of the two-dimensional function; if one region is dangerous, evaluate the neighbouring regions.

In all cases the regions were prioritised successfully, reducing the number of regions evaluated before all dangerous regions were identified. However, as no probabilistic information is available, this is not known with certainty until all regions *are* evaluated.

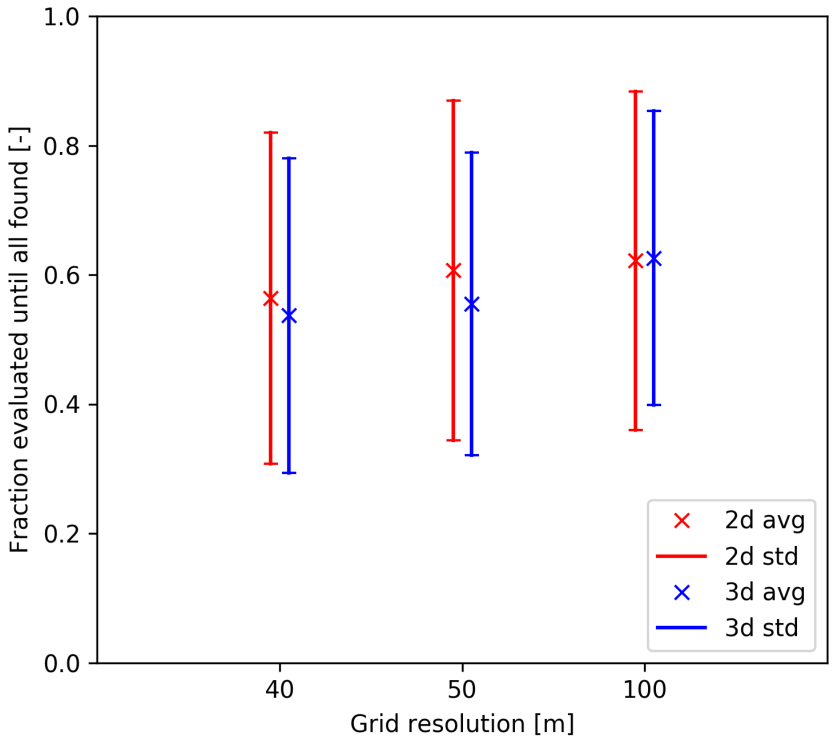


Figure 5.32: Comparing the average performance of the two- and three-dimensional piecewise linear function approximator for various sizes of the regions.

## 5.5 Conclusion

In this chapter sampling strategies that lead towards an effective and efficient strategic analysis of an area have been introduced and evaluated. First, the effects of shifting the sampling location from the centre to another location using the surface height have been investigated. This showed the probability of identifying dangerous areas increases significantly, shifting the chances of detection from as low as 50% to over 70%.

Furthermore, a method to prioritise the dangerous regions of a map, again using surface height data, has been proposed. This method, on average over the 32 cases, successfully identifies all dangerous regions in less than half the time compared to the brute-force approach.

Future work, in particular concerning assurances, may include interval analysis. Probabilistic methods are not viable due to the computational cost, however, by including assumptions on the shape of the objective function, interval analysis may provide better results. For example, if the surface map appears to be concave, and the two highest points are safe, then the area between those two points may also be considered safe. Also, the method as applied in this work identifies dangerous locations using a hard boundary. This assumes that the provided information, i.e., the surface map, is accurate to the millimetre. Extending the analysis to include *possibly dangerous* regions will provide the means to have a more conservative approach, dealing with uncertainty in both the available terrain data and risk analysis.

Chapter 6

# Uncertainty quantification in a rocket simulator

This chapter aims at identifying existing techniques capable of quantifying uncertainty in systems, specifically the prediction of rocket trajectories. First, section 6.1 places uncertainty quantification in the context of computer simulations, introducing stochastic systems and methods to propagate uncertainties from the input to the output. Furthermore, a review of existing rocket flight simulators, and the areas in which these are applied is presented in section 6.2. This provides an overview of existing research in modelling rocket dynamics, and includes a list of available software aimed at predicting rocket performance. Finally, section 6.3 concludes with some final remarks, highlighting the importance of a new approach to rocket trajectory prediction, so far missing in the literature.

## 6.1 Uncertainty quantification

This section focusses on uncertainty quantification in the context of computational systems (i.e., simulations). Uncertainty quantification is the quantitative characterization of uncertainties, providing an estimation of what is likely to happen based on the available, or lack of, information. An advantage of uncertainty quantification methods is that these require no knowledge of the inner workings of the system, but only the input/output relation. Therefore, systems that demonstrate similar behaviour may benefit from the same uncertainty quantification method, irrespectively of the purpose of the system (e.g., predict rocket performance or the prices of stocks).

The first section looks into what makes a system stochastic, focussing on what occurs within the system. Following this is a section that examines approaches that capture the relation between the input/output of stochastic systems.

### 6.1.1 Stochastic systems

The word '*stochastic*' can be expressed in simpler terms as '*randomly determined*', and is described by the Oxford English Dictionary as '*that follows some random probability distribution or pattern, so that its behaviour may be analysed statistically but not predicted precisely*'. A fundamental word in this description is *pattern*, which implies that even though things might appear random when looking at individual samples, there is a pattern when looking at a large number of samples. Such a pattern is captured in what is called a probability distribution.

The opposite of stochastic systems are deterministic systems, which always produce the same output for a given input. However, by introducing stochastic components, a deterministic system can become stochastic. Examples of such components are stochastic inputs, random events, and noisy disturbances.

Stochastic systems are at the centre of numerous disciplines in engineering, such a reliability design (Nikolaidis, Ghiocel and Singhal 2004). They also find application elsewhere, such as social processes (Bartholomew and Bartholomew 1967), biochemical dynamics (Wilkinson 2009), molecular biology (Székely and Burrage 2014), epidemiology (Bartlett 1960), and financial markets (Kennedy 2016).

Stochastic modelling concerns the use of probability distributions to model situations in which uncertainty is present. It is important to note that the use of a stochastic model not always implies that there is a fundamental belief that there is randomness in the system, as often it merely acknowledges that the available data and models are imperfect (MacKay 2003).

For example, consider the atmospheric conditions in an environment at any given time. This uncertainty in the exact value remains even when the atmosphere is monitored by multiple agents at several locations simultaneously (King, Scanlan and Sóbester 2015). A further reduction in uncertainty can be achieved by actively steering the agents towards a flight path optimised for exploration (Crispin and Sóbester 2015; Crispin and Sóbester 2016), and yet, part of the uncertainty remains. In such a case, the atmospheric conditions can be considered stochastic, and quantify the remaining uncertainty to include it in the analysis (Box, Bishop and Hunt 2010).

The next section talks about how to quantify the uncertainty, and capture the relation between the input and output of stochastic systems.

### 6.1.2 Uncertainty propagation

The uncertainty of any variable is captured in a probability distribution. For a single variable it is possible to see such a probability distribution as a normalised, continuous version of a histogram. The requirements being that the complete area under the curve sums up to one and the distribution cannot be negative. Our perception of reality is that the world around us is continuous, however, all measurements made by instruments are a snapshot at a given time, thus discontinuous. These distributions can either be *parametric* (e.g., Dirichlet distribution, Gaussian distribution) or *non-parametric* (e.g., Kernel Density Estimate (KDE)), and can be trained via a frequentist or Bayesian treatment. The frequentist treatment would be to optimise a specific criterion (e.g., the likelihood) – and is used in this thesis. A Bayesian treatment would involve introducing prior distributions over the parameters, and compute the posterior distribution given the observed data.

A widely used parametric probability distribution is the *Gaussian*, also known as the normal distribution. Not only does it describe many natural phenomena, but it also has computationally attractive analytical properties such as being *symmetric* and *unimodal*.

Furthermore, a linear transformation of a Gaussian produces again a Gaussian. This allows the propagation of uncertainty through linear systems, or in the other direction, the identification of the probability distribution of the input based on the output. The Kalman filter is an example that makes use of these properties. It determines the true state of a system by observing noisy measurements – with the assumption that the noise distribution is Gaussian. The Extended Kalman Filter (EKF) (Sorenson 1985) extends this method to work on non-linear systems, by locally linearising the system at every update.

An advantage of the Gaussian distribution is that it has such an analytical solution, while many other distributions require a numerical method of maximizing the likelihood. Also, although there is no limit on the complexity of the estimated probability distribution other than the number of samples, it makes sense from a practical standpoint to reason why a particular distribution is valid. It reduces the computational cost of learning the probabilistic model, and can increase the accuracy in the presence of a low sample size as it deals with the problem of missing data. Furthermore, there are methods available for sampling from a Gaussian probability distribution with a low computational cost.

However, when the system of interest is non-linear, or subject to a non-Gaussian uncertainty (e.g., KDE), there are methods available for non-linear uncertainty propagation. Examples are local linearisation, Monte Carlo methods (Doucet, Freitas and Gordon 2001), and arbitrary Polynomial Chaos (aPC) (Oladyshkin and Nowak 2012). Where Monte-Carlo methods are often a implemented as they are able to handle large state spaces, arbitrary input distributions, and require no additional information of the system such as the state derivatives. Furthermore, there are cases where making assumptions (e.g., a linear model, no error propagation over time) are useful for a quick evaluation of robustness in a system, for example, the uncertainty quantification of an aircraft simulation model (Rosi and Diekmann 2014). Here a polynomial-chaos approach is compared with the Monte Carlo, showing the same level of accuracy in the sensitivity analysis, while being characterized by several orders of magnitude computation time.

The limitations here are that the uncertainty of the input parameters are Gaussian distributed and the uncertainty is not propagated over time.

In order to cope with the variation of uncertainty as a function of time, it is possible to look at Gaussian processes, as introduced in section 2.4. In a more general description, the data are considered to originate from a process where the observation occurs as a function of a continuous variable (e.g., time or space). Therefore, not only the uncertainty of a single state is considered, but also the relation between the different states as a function of another variable. However, while these methods can capture the probabilistic model of a variable as a function of time (or any other variable), the propagation of uncertainty through a non-linear system remains a challenge.

The next section introduces research on rocket dynamics, and how the uncertainties in design variables and the initial state are propagated throughout such a system. The uncertainty quantification is useful for predicting the point of impact of the rocket, both in the planning phase and during the flight.

## 6.2 Simulating rocket dynamics

This section provides an overview of existing research in modelling rocket dynamics, and includes a list of available software aimed at predicting rocket performance. First, this section introduces rocket simulators and work related to rocket dynamics found in research, then it continues with software implementations predicting rocket performance.

### 6.2.1 Rocket simulation in research

Most research performed on predicting the trajectories of projectiles uses numerical integration. The exception is the work of Chudinov (2003), who predicts the trajectories by applying an analytical solution. While this approach is beneficial in terms of computational costs, the prediction is limited to modelling a point mass with a quadratic drag force.

For higher fidelity dynamics, numerical models are used in the analysis. Such models refer to systems that are based upon equations of motion, and then implemented as a coded algorithm as no analytical solution is available. These implementations often include a method for numerical integration, such as an implementation of the Runge-Kutta algorithm (Press et al. 2007). For modelling rocket dynamics, the numerical integration deals with the changes over time, and how the state of the rocket evolves as a function of time.

The literature concerning the equations of motion of a rigid projectile starts in 1964 (Lieske and McCoy 1964). Two years later the equations of motion for a modified point mass trajectory are introduced as a low fidelity model in order to reduce the computational cost (Lieske and Reiter 1964). This balance between fidelity and computational cost is a returning topic. For example, there are advanced simulators where advanced coupled computational fluid dynamics/rigid body dynamics are used to predict the rocket behaviour even as it enters supersonic speeds (Sahu 2008). However, as the additional computational cost is significant due to the calculation of unsteady aerodynamics associated with supersonic flight, it is computationally costly to perform an uncertainty analysis of parameters via a Monte Carlo method. That does not take away that advanced simulations remain invaluable when testing the performance of new control methods, as performed by Gomez and Miikkulainen (2003), who test their control method for a finless rocket via simulations.

The area where rocket flight dynamics and uncertainty propagation intersect is Impact Point Prediction (IPP), where the impact point of a projectile is estimated in advance (both before launch, or in-flight). Such a prediction of trajectories can achieved using a mathematical model of the motion via numerical integration (Kashiwagi 1968). Furthermore, by incorporating the uncertainty due to various sources (e.g., wind), the accuracy of the landing location confidence bounds can be increased (Yuan et al. 2014). In-flight corrections are also possible, for example via Doppler radar measurements (Khalil et al. 2014). Besides applying the uncertainty analysis to predict the impact point, it is also possible to use the same information to *guide* the rocket such that it avoids objects and reaches the target location without collisions (Rogers 2014).

For passively controlled sounding rockets, the rocket design stage is the most important part of ‘controlling’ the rocket. Additionally, there is a limited amount of control over the launch conditions, such as the time, location, and setup. However, once the rocket launches, there is no additional control via a ground station. It is possible to have a performance prediction, taking into account uncertainties in design, via a rocket model and the implementation of a Monte Carlo scheme (Nassiri, Roushanian and Haghighat 2004). Furthermore, by including stochastic weather conditions in this Monte Carlo scheme, the confidence in the predictions can be increased (Box, Bishop and Hunt 2010). This combination of uncertainty in design- and atmospheric conditions leads to splash-down patterns, useful in predicting where the rocket may land. This work has been translated to a rocket simulation software, and will be discussed in the next section.

### 6.2.2 Rocket simulation software

Flying rockets is both a popular hobby for amateur enthusiasts and an academic activity. For hobbyists, scheduled meetings are organised by the U.K. Rocketry Association (`http://www.ukra.org.uk/`) and National Association of Rocketry (`http://www.nar.org/`) for the United Kingdom and United States respectively. Furthermore, many universities are associated with societies that are involved in rocketry, such as the MIT Rocket Team (`http://rocketry.mit.edu/`), Cambridge University Spaceflight (`http://www.cusf.co.uk/`), Delft Aerospace Rocket Engineering (`http://dare.tudelft.nl/`), and the University of Canterbury Rocketry Project (`http://www.ucrocketry.org/`), where the involvement in rocketry can include the following three components: the design, building, and testing of passively controlled rockets (Buchanan et al. 2015).

There are various software packages that facilitate conceptual studies of rocket design, including commercial solutions such as RockSim (Apogee 2008), open-source solutions such as OpenRocket (Niskanen 2015), and even computer games such as the Kerbal Space Program (Squad 2011). All these packages simulate the performance of a rocket under nominal conditions, returning the expected performance to the user.

However, without the prediction of the trajectories, or more specifically, the confidence bounds of the landing location, you may never get permission from the aviation authorities to launch your rocket (Commercial Space Transportation 2007). While RockSim Pro (Apogee 2008) does display confidence bounds around the predicted landing location to the user, it is only available to citizens from the United States. As previously mentioned, the work by Box, Bishop and Hunt (2010), involving a stochastic six-degrees-of-freedom flight simulator, has been translated to a rocket simulation software called the Cambridge Rocketry Simulator (Box and Eerland 2017). This software was the first with a peer-reviewed, open-source simulator core, capable of quantifying the uncertainty in the landing position. However, in terms of design, it is lacking any visual representation of the rocket, making it difficult for end-users to compose and verify their conceptual design. A similar attempt to quantify the uncertainty in the landing position has been made with ROcket Simulation and Impact Estimation tool (ROSIE) (Engelen 2012), however, this is less software and more a collection of scripts to be run, lacking a visual interface entirely.

## 6.3 Conclusion

A review of existing research in modelling rocket dynamics shows that trajectory prediction of rocket trajectories, in particular Impact Point Prediction (IPP), is an active field. These models include dynamics from passively controlled solid-propellant rockets as well as actively controlled liquid-propellant rockets. Furthermore, the implemented models range from high fidelity models, used in the design of new, experimental techniques, to low fidelity models, used for quantifying the uncertainty. This quantification is both used for robust design and predicting the point of impact. Here, the general trade-off between the high and low fidelity models is the accuracy of the simulation versus the computational cost associated with obtaining results. For this reason, uncertainty is quantified via low fidelity models using Monte Carlo schemes.

The overview of existing rocket simulation software indicates that there are tools available for conceptual design, simulating a rocket under nominal conditions, returning the expected performance to the user. Furthermore, the *availability* of software suitable for obtaining confidence bounds on the landing location is limited. The tools available to end-users (e.g., hobbyists and students) require studying scripts and interpreting input variables to match the a rocket design with prediction results. A single piece of software combining these aspects is missing.

The next chapter, chapter 7 introduces a stochastic rocket flight simulator, meeting the requirements as discussed in this chapter (chapter 6). The trajectory data, generated by the rocket flight simulator, is then also analysed via the probabilistic modelling approach as proposed in chapter 3.

# Chapter 7

# Probabilistic modelling of rocket flight trajectories

This chapter introduces a rocket flight simulator capable of the conceptual design, simulating a rocket under nominal conditions, and returning the expected performance to the user. As identified in chapter 6, a single piece of software combining these aspects is missing. Furthermore, this chapter also describes how the probabilistic trajectory analysis approach introduced in the previous chapter can be applied to the trajectory data output given by the simulator.

The rocket simulator is a piece of software that is designed for use with passively controlled rockets including model rockets, High Power Rockets (HPR), and sounding rockets. Typically, these will perform a sub-orbital flight, collect some data, and deploy a parachute for recovery to Earth. The software models the flight dynamics of the rocket in six-degrees-of-freedom and of the parachute descent in three-degrees-of-freedom. It has a range of uses. Firstly, it can guide the design process. For example, it allows the engineer to select the appropriate size motor for a desired apogee; or to design the fins for an appropriate margin of stability; or to optimise the timing of stage separation or parachute deployment. Secondly, it enables the operator to predict the landing location, which helps in determining the required launch safety exclusion zone, as well as facilitating the retrieval of the reusable components of the rocket. In order to map uncertainties in the dynamics and the atmospheric conditions into confidence bounds around the predicted landing location, a Monte Carlo approach is combined with a numerical integration scheme.

Section 7.1 starts with the introduction of the rocket simulator, focussing on the architecture and how the rocket flight trajectories are generated. Following that section 7.2 verifies the performance of the trajectory analysis approach on sounding rocket trajectories. Section 7.3 elaborates on how the trajectory analysis tool Teetool is integrated within the existing architecture, and how pitfalls discovered in section 7.2 are resolved. Finally, section 7.4 concludes with some final remarks, summarising the work presented in this chapter.

## 7.1 A stochastic six-degrees-of-freedom rocket flight simulator

This section provides an overview of the inner workings of the rocket simulator named *The Cambridge Rocketry Simulator*. This rocket simulator holds several advantages: Firstly, it is free and open-source. Secondly, the physics model is verified, peer-reviewed and published (Box, Bishop and Hunt 2010). Thirdly, by using a Monte Carlo wrapper, it incorporates uncertainties in both rocket dynamics and atmospheric conditions, making it possible to produce a splash-down area with confidence bounds. Finally, the atmospheric model supports a three-dimensional wind vector, air density, and air temperature, all as a function of altitude. These atmospheric data may be populated from a recent meteorological forecast to maximize the accuracy of predictions. Furthermore, with the introduction of the third version of the Cambridge Rocketry Simulator parts of the Graphical User Interface (GUI) from OpenRocket (Niskanen 2015) are integrated to assist the user in their rocket design.

The remainder of this section covers the implementation and architecture, stochastic elements, quality control, and reuse potential of the simulator.

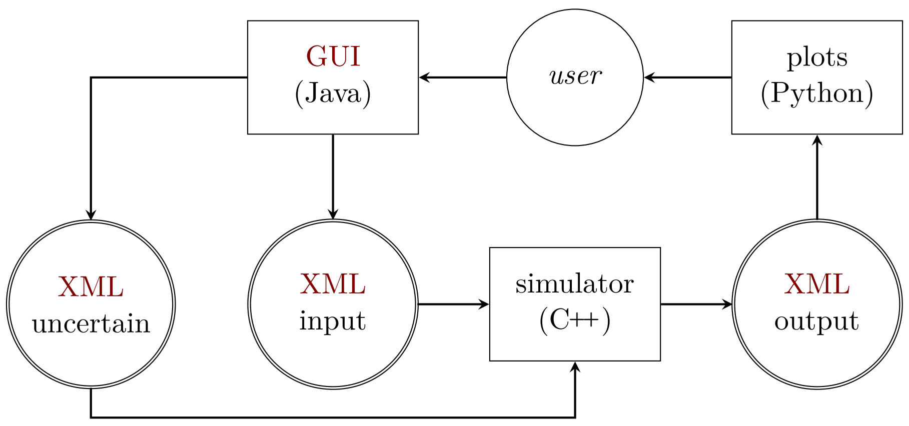


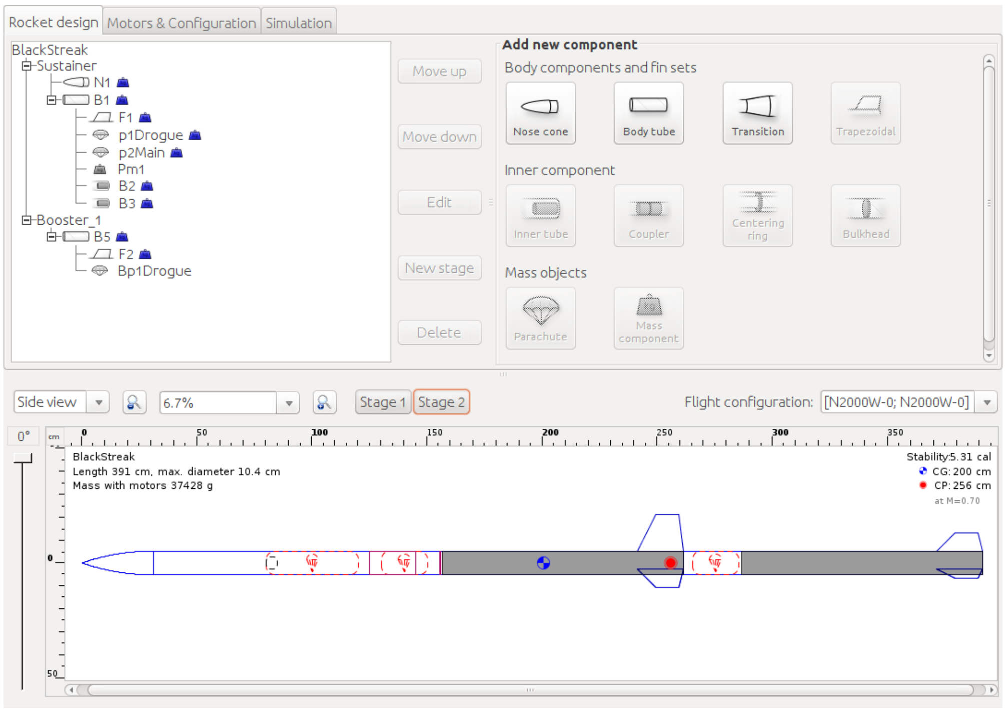
Figure 7.1: Schematic representation of the components and the corresponding programming languages.

### 7.1.1 Implementation and architecture

The software package consists of three components, each written in a different language to suit their individual purpose: A GUI, a simulator, and a visualisation module. A schematic representation of the three components, and how they interact, is presented in figure 7.1.

The GUI used to design the rocket is coded in Java, producing the design page as seen in figure 7.2, where the user can select the different components to build the rocket. More details about the GUI, and the corresponding internal structure can be found in Niskanen (2013). The example here shows a two-stage rocket. Once the rocket is designed, all the parameters are passed to the simulator core (written in C++) via an Extensible Markup Language (XML) data-sheet. These parameters are required in simulation, and consist of the moments of inertia, centre of pressure, drag coefficient and the thrust curve. A full list of parameters and a description of the components are available in the user guide USER_GUIDE.PDF, which is located at DOC/.

The core of the physical simulation, the ROCKETC source code, is located in CPP/ which includes a Makefile that compiles the binary and moves it to the simulator folder, found in SIMULATOR/. The Java GUI calls the simulation binary, after which the simulation



Side view

6.7%

Flight configuration: [N2000W-0; N2000W-0]

BlackStreak
Length 391 cm, max. diameter 10.4 cm
Mass with motors 37428 g

Stability: 5.31 cal
CG: 200 cm
CP: 256 cm
at M=0.70

Figure 7.2: GUI for designing a rocket. Contains all the basic components required to design a rocket.

starts by reading the input XML generated by the Java code. The XML file SIMULATIONINPUT.XML can be found in the Data folder at DATA/. Upon completion of the simulation an output XML file SIMULATIONOUTPUT.XML is generated in the same folder.

This output of the simulation is then input to the Python visualization code which is located in PLOTTER/. This presents the user with an overview of the flight trajectories and the splash-down area. The splash down-area includes confidence bounds, as seen in figures 7.3 and 7.4.

When running the software under Linux, the three folders, SIMULATOR/, DATA/, and PLOTTER/, are expected to be located in the users home directory ~/.CAMROCSIM/. Therefore, to prepare the system for execution from source code, a script called PREPARE_LINUX.SH is included in the repository. This copies the three relevant folders to this location. The information presented in this section, and build instructions for Windows, are available in the README.MD file found in the repository.

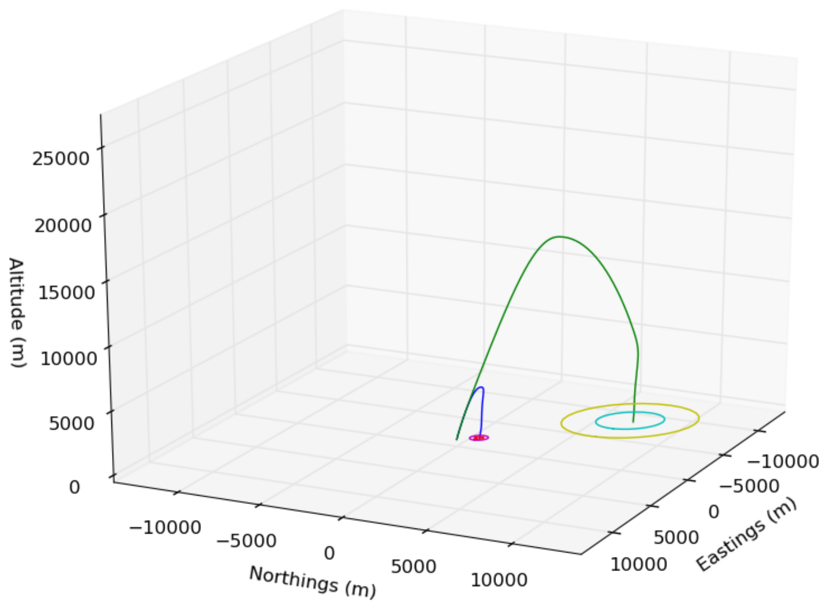


Figure 7.3: Mean trajectories of both stages of a two-stage rocket including a staged parachute descent. The 1 and 2 $\sigma$ bounds of confidence in landing position are also shown.

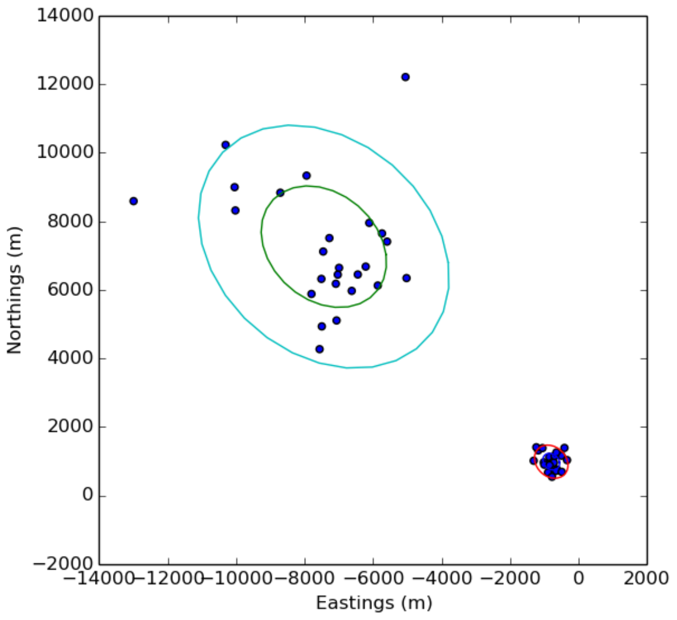


Figure 7.4: Landing position confidence bounds from figure 7.3 along with the individual landing locations of 25 Monte Carlo iterations.

### 7.1.2 Stochastic elements

This section describes the process of capturing the uncertainty of an input variable (e.g., drag coefficient) in the output (e.g., landing location), as this is not straightforward in a system with coupled input variables. In simulating systems that have many coupled input variables such as this one, Monte Carlo methods are useful to obtain a probabilistic interpretation. In a nutshell, Monte Carlo methods work by sampling over the probability distribution of the input space, and reflect that uncertainty in the output of the simulation. While this provides a method of obtaining a probabilistic interpretation of the output, here we focus on the probability distribution of the input space.

An overview of the input variables, their probability distribution, the default values, and how they are applied can be seen in table 7.1. Not all variables are directly available in the GUI (seen in figure 7.6), however, those not visible can be modified in the uncertainty XML file. The probability distribution of all input variables is assumed to be normally distributed, however, the mean value, and the manner in which the uncertainty (represented as the standard deviation $\sigma$) is applied differs. For both angles, declination and azimuth, the random variables is applied as an addition, with a mean equal to 0. The reasoning here is that the positioning uncertainty is typically independent of the variable value. For the other variables the uncertainty is applied as a multiplication, and the mean is equal to 1. The result is an uncertainty as a percentage, causing large values of the variable to have a larger variation than smaller values.

Based on the assumption of a constant burn rate, extending the thrust curve in the time dimension relates to a larger quantity of fuel, where an increase in the thrust dimension relates to a better quality of fuel. Here, the quality corresponds with the specific impulse, which is the impulse per kilogram of propellant. This assumes the total mass of the propellant is constant and therefore the specific impulse is proportional to the total impulse (i.e., change in momentum), which is the integral of the thrust curve. A schematic representation of these effects is presented in figure 7.5. In this case the choice has been made to model the uncertainty of the quality of the fuel. Therefore, as the thrust curve consists of multiple points as a function of time, the uncertainty is modelled

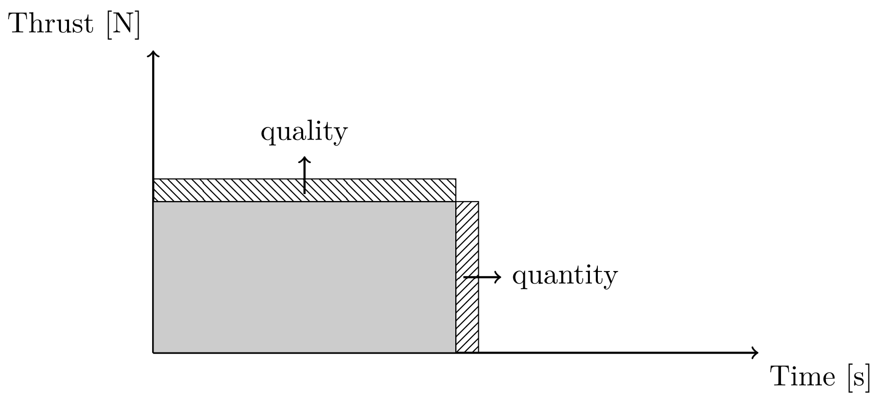
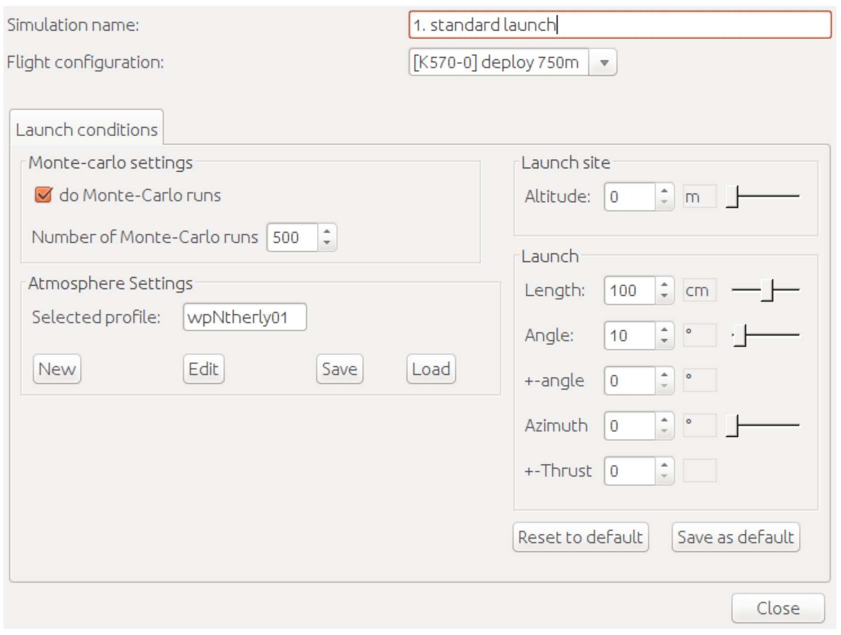


Figure 7.5: A schematic representation visualising the effects of rocket fuel on the thrust curve.


Simulation name: 1. standard launch
Flight configuration: [K570-0] deploy 750m

Launch conditions

Monte-carlo settings
☑ do Monte-Carlo runs
Number of Monte-Carlo runs 500

Atmosphere Settings
Selected profile: wpNtherly01

Launch site
Altitude: 0 m

Launch
Length: 100 cm
Angle: 10 °
+-angle 0 °
Azimuth 0 °
+-Thrust 0

Figure 7.6: Launch- and atmospheric conditions tab of the user interface.

by drawing a sample from the normal distribution, then all thrust values are multiplied with this sample value, causing the entire curve to shift up or down accordingly.

Finally, it should be noted that there is a ceiling of 250 as a maximum thrust-to-weight ratio set within the simulation. This is to prevent a scenario of unreasonable thrust that would produce numerical errors in the simulator.

Table 7.1: Overview of stochastic variables and their implementation. Note that these are only examples, and these probability distributions *may* not be a representative for their true distribution.

| **Variable** | **Probability distribution** | **How applied** |
|---|---|---|
| Drag coefficient | $\mathcal{N}(1, \sigma^2)$, $\sigma = 0.2$ | Multiplication |
| Centre of pressure | $\mathcal{N}(1, \sigma^2)$, $\sigma = 0.1$ | Multiplication |
| Normal coefficient | $\mathcal{N}(1, \sigma^2)$, $\sigma = 0.1$ | Multiplication |
| Parachute drag coefficient | $\mathcal{N}(1, \sigma^2)$, $\sigma = 0.1$ | Multiplication |
| Thrust curve | $\mathcal{N}(1, \sigma^2)$, $\sigma = 0.01$ | Multiplication |
| Declination launch angle | $\mathcal{N}(0, \sigma^2)$, $\sigma = 1$ | Addition |
| Azimuth launch angle | $\mathcal{N}(0, \sigma^2)$, $\sigma = 1$ | Addition |

### 7.1.3 Quality control

The performance of the simulator has been evaluated by comparison to telemetry data recorded in rocket flights as described in Box, Bishop and Hunt (2010).

The performance and stability of the GUI and the visualiser have been tested in two rounds of user-testing, including cross-platform testing, which was done on Ubuntu 14.04 *Trusty Tahr* and Windows 10.0 (Build 10240).

Finally, there are unit tests available for the GUI and simulator, using JUnit for the GUI (Java), and GoogleTest for the simulator (C++). Instructions on how to run these are included in the README.MD file.

### 7.1.4 Reuse potential

The modular design on the Cambridge Rocketry Simulator's source code makes it inherently reusable. The simulator itself – as described above – is a stand-alone program callable from the command-line, accepting a XML simulation input file as its single parameter. Therefore this can be ported into other applications.

Furthermore, individual classes of the simulator have reuse potential. For example the RKF45.CPP class is a general ordinary differential equation solver which implements the fourth/fifth order Runge-Kutta algorithm with Fehlberg step-size control (Press et al. 2007). This may be used in any dynamics simulation application.

Similarly the VMATHS.CPP class may be used for modelling the six-degrees-of-freedom dynamics of rigid bodies. In particular the implementation of quaternions in this class is helpful for modelling unconstrained rotation without the need to deal with the singularities that arise when using Euler angles (Baraff 1997).

## 7.2 On modelling rocket flight trajectories

In this section the modelling approach as presented in chapter 3 is applied to three-dimensional rocket flight trajectory data as generated via a stochastic rocket flight simulator (Box, Bishop and Hunt 2010). The aim here is evaluating the performance of the method applied to the rocket flight trajectory dataset. First, a number of trajectories are generated according to specific settings, then modelled according to the probabilistic approach, and finally the performance of this probabilistic model is evaluated via the K-S statistic.

### 7.2.1 Producing rocket flight trajectories

The rocket flight trajectory data are generated via a Monte-Carlo simulation, where a number of stochastic variables in the underlying deterministic simulator are sampled over a Gaussian distribution with each run. The simulator includes a physical model, making the trajectories representative of trajectories that can occur in reality. The rocket considered here is passively controlled and consists of two stages, which results in a sudden acceleration of the rocket at second stage ignition. The launching rail, which guides the rocket at the launch, is pointed towards the North-East at a declination angle of 45 degrees. Ordinarily this simulator, which is designed for sounding rockets, models a parachute deployment at apogee and descent under parachute. However, for simplicity the trajectories generated for this work are all *ballistic*, that is without parachute deployment. This, combined with atmospheric data simulating a wind from the west, results in the trajectories as seen in section.

```
% Loads the rocket data into a variable called INTAB
load intab_CLV2s1_K660.mat;
% Loads the atmospheric data into a
% variable called INTAB4
load intab4_2006100809.mat;
% Altitude [m] at which the rockets second parachute
% will deploy
Parachute2Alt = 300;
% Length of the launching rail in meters
RailLength = 2;
% Angle of declination of the launching rail
% in degrees
RailDeclination = 45;
% Bearing of the launching rail in degrees from
% true north
RailBearing = 45;
% Number of monte carlo iterations to be performed
noi = 2500;
% Starts the simulator
[AscDat,DesDat,Land,Apo] = rocketflight_monte(INTAB,
  INTAB4, Parachute2Alt, RailLength,RailDeclination,
  RailBearing, noi,"ballisticfailure");
```

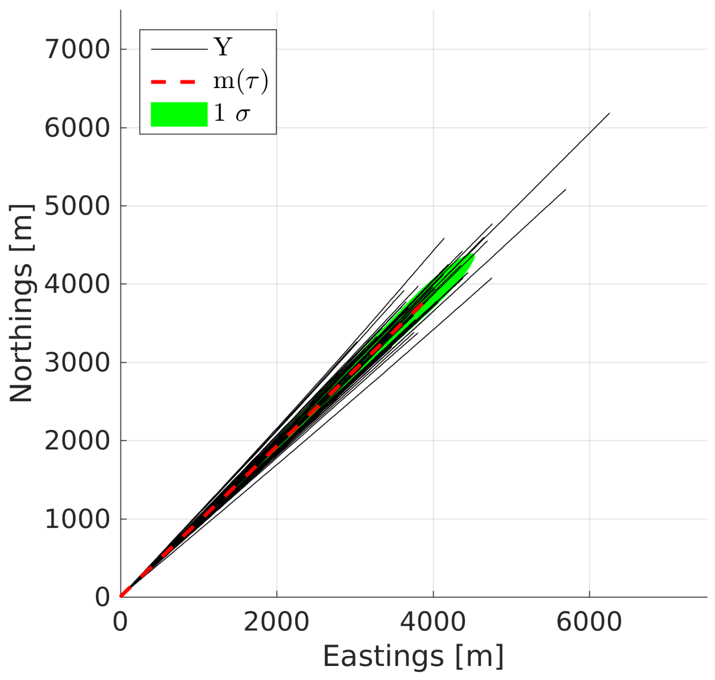
Figure 7.7: Input settings for the stochastic rocket simulator.

The specific settings of the simulator to generate the trajectories seen in this section can be reviewed in figure 7.7. The MATLAB version of the Cambridge Rocketry Simulator is available for download at `http://cambridgerocket.sourceforge.net/`. Here, the MATLAB version has been selected due to the reproducibility of the random generated data by selecting a *static seed* (i.e., the random generated data always looks the same). In total 4000 trajectories are generated; 2000 for the training dataset $\mathbf{Y}$ and 2000 for the evaluation dataset.

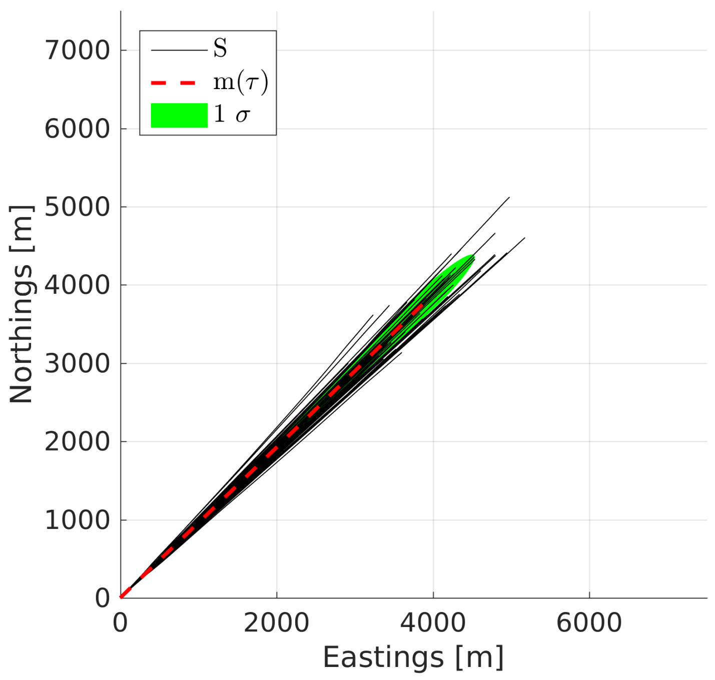
### 7.2.2 Modelling rocket flight trajectories via Gaussian processes

This section analyses sounding rocket flight trajectories generated via a stochastic rocket flight simulator as discussed in section 7.2.1.

The 2000 rocket trajectories in the training data $\mathbf{Y}$ are used to build the probabilistic model $p(\mathbf{v})$ according to the approach described in section 3.2, where the model selection is done based on the likelihood as described in section 3.3.1. The maximum value for

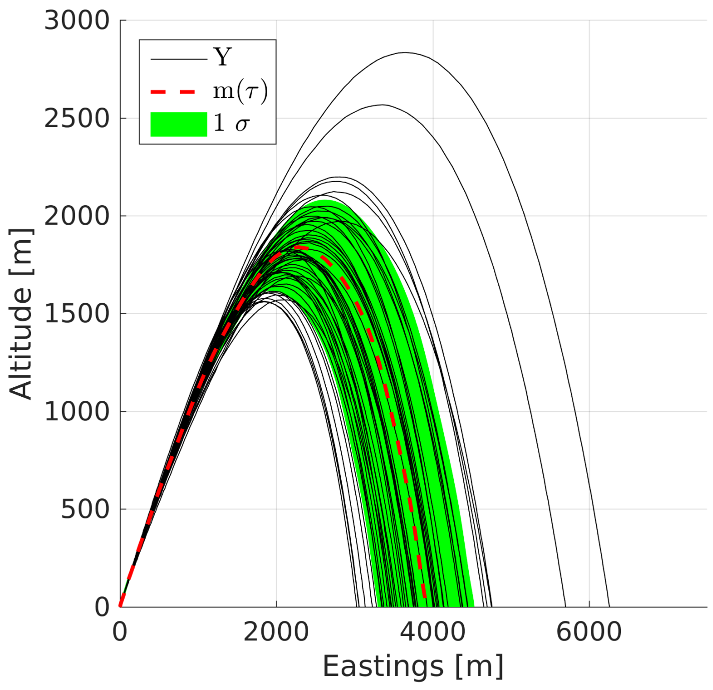


(a) rocket trajectory training data **Y**

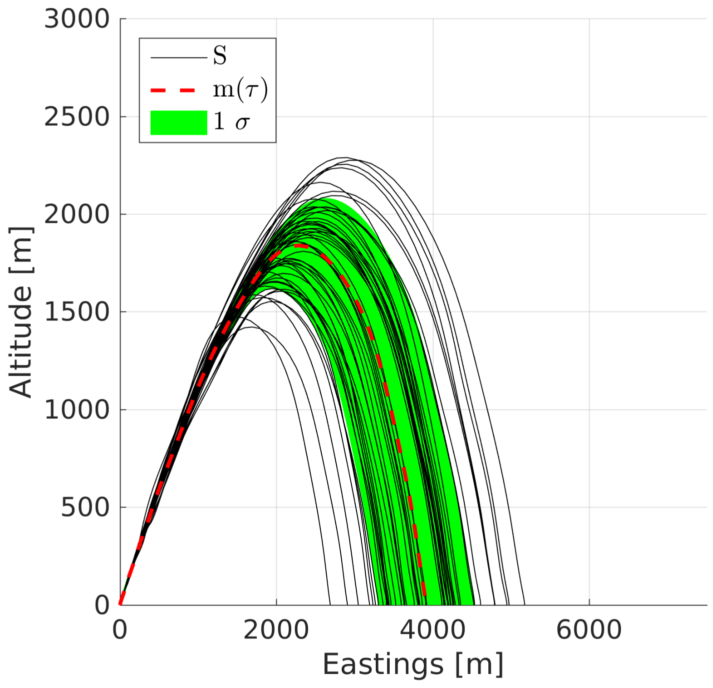


(b) rocket trajectory sampled data **S**

Figure 7.8: Visualisation (top-view) of sounding rocket training and sampled trajectory data.



(a) rocket trajectory training data **Y**



(b) rocket trajectory sampled data **S**

Figure 7.9: Visualisation (side-view) of sounding rocket training and sampled trajectory data.

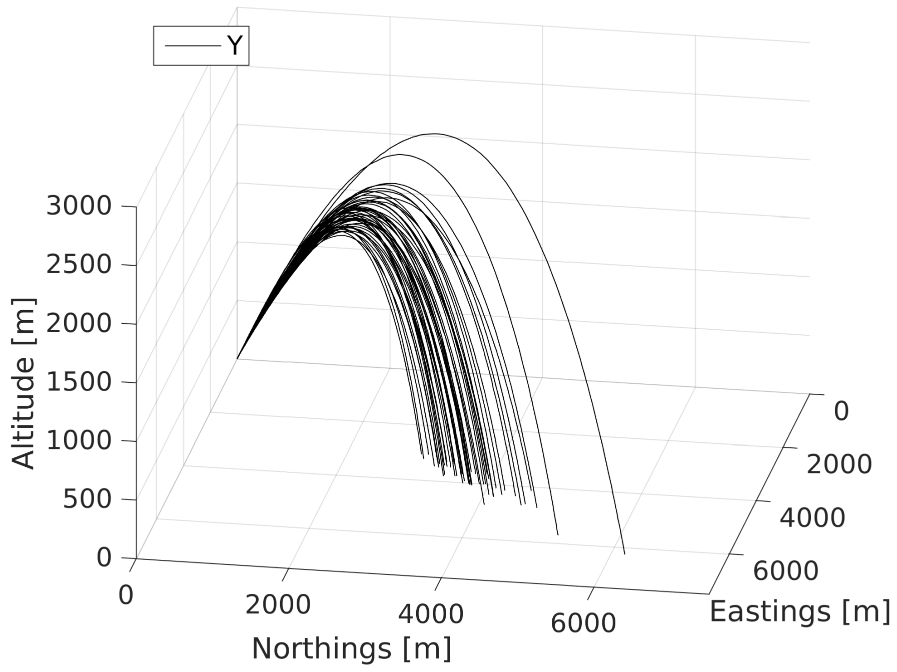


Figure 7.10: A three-dimensional representation of the sounding rocket training trajectory data **Y**.

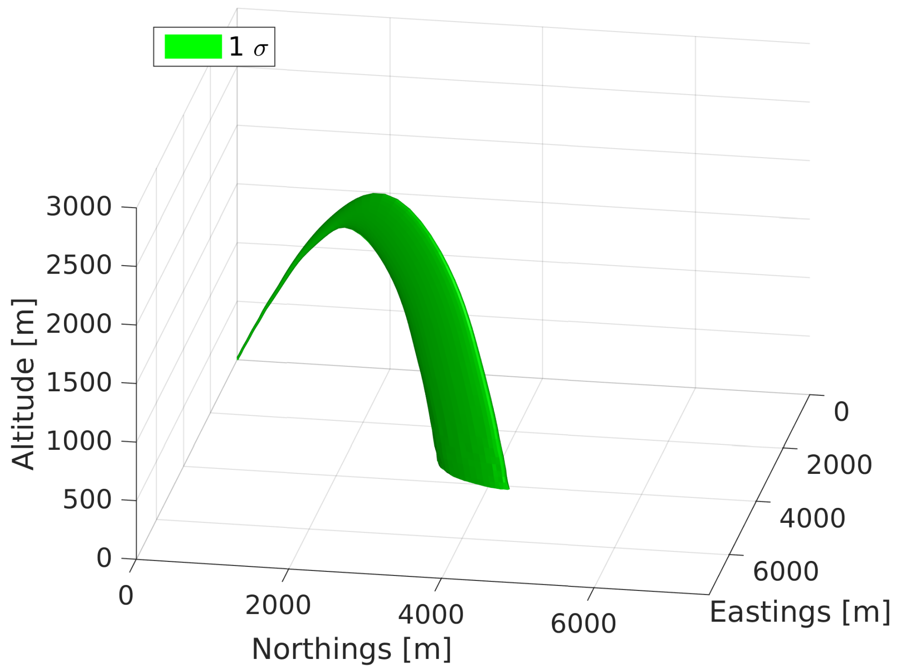


Figure 7.11: A three-dimensional representation of the volume containing one standard deviation for the sounding rocket trajectories.

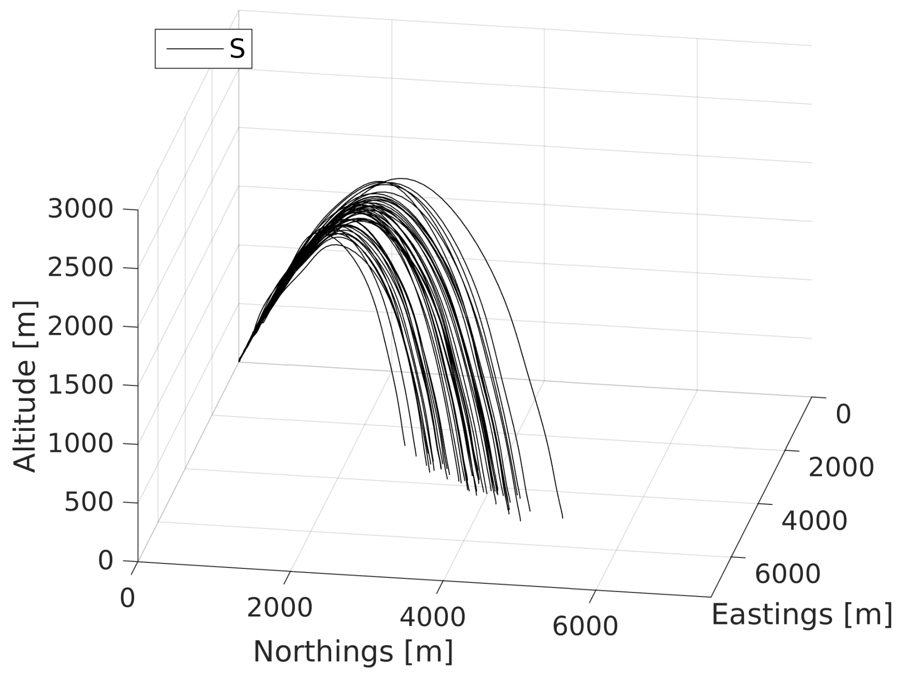
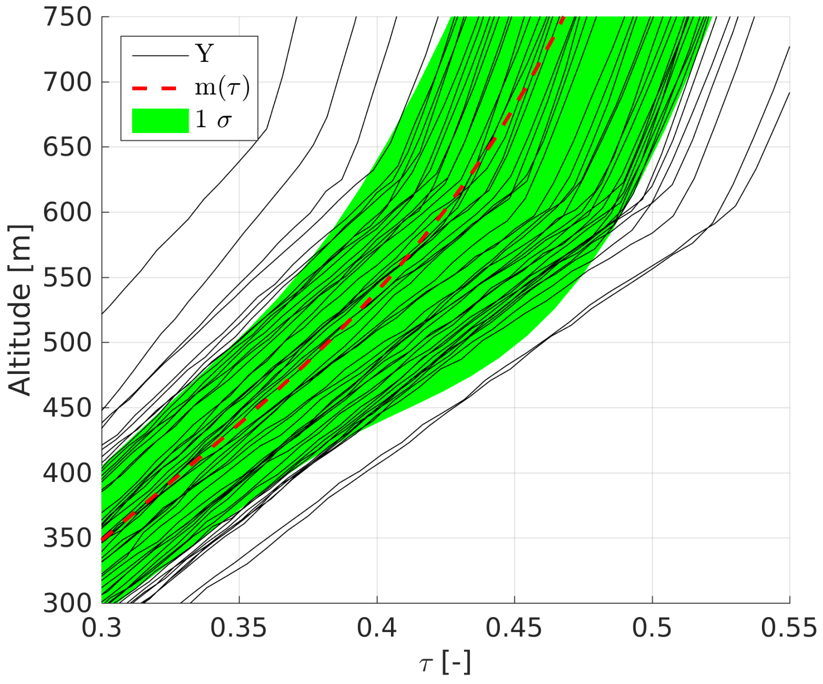


Figure 7.12: A three-dimensional representation of the sounding rocket sampled trajectory data **S**.

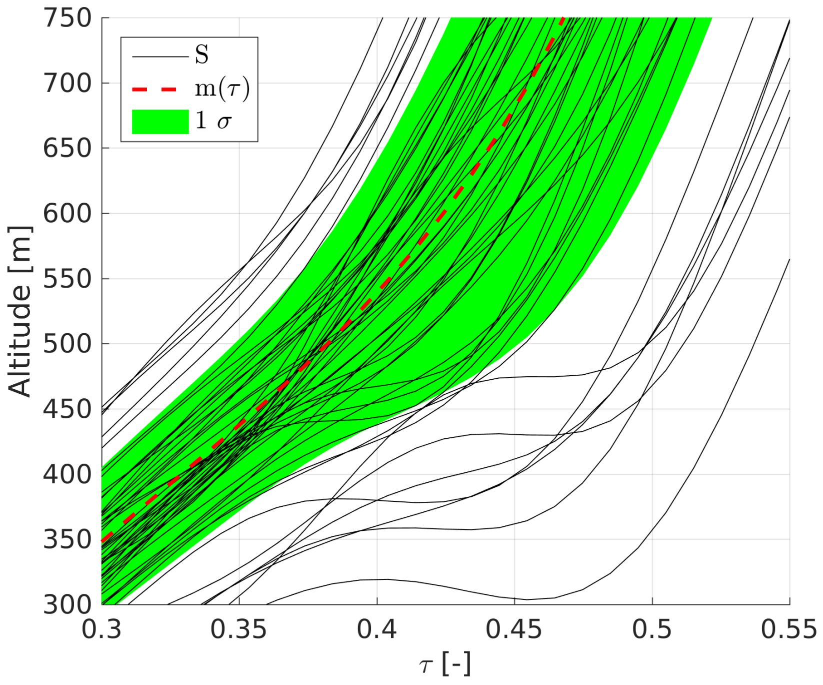
the marginal likelihood occurs at the model complexity $J = 18$, and is therefore selected to be the best model.

Only 50 trajectories from the training data are seen in figures 7.8a and 7.9a to prevent visual cluttering. It shows the trajectories seen from two directions, the top-view and a side-view. Furthermore, the mean trajectory ($\mathbf{m}(\tau)$) is visualised as a dashed line and the region captured by one standard deviation ($1\ \sigma$) is shown by a shaded area. From the probabilistic model $p(\mathbf{v})$, 50 sample trajectories (described as **S**) are generated using equation (3.20). These sampled trajectories **S** are shown in figures 7.8b and 7.9b. Additionally, a three-dimensional representation of the training data **Y**, sampled data **S** and the volume containing one standard deviation is presented in figures 7.10 to 7.12 respectively.

The sampled trajectory data **S** behaves as expected when assuming a Gaussian distribution in so far as they exhibit greater symmetry about the mean than the training data. As the trajectories are based on physics, i.e. follow specific rules, generating samples based purely on a probability distribution can lead to samples that conflict with the



(a) rocket trajectory training data **Y**



(b) rocket trajectory sampled data **S**

Figure 7.13: Altitude time-series of the sounding rocket trajectories at $\tau = [0.3, 0.55]$, illustrating the difference between the physical and probabilistic model.

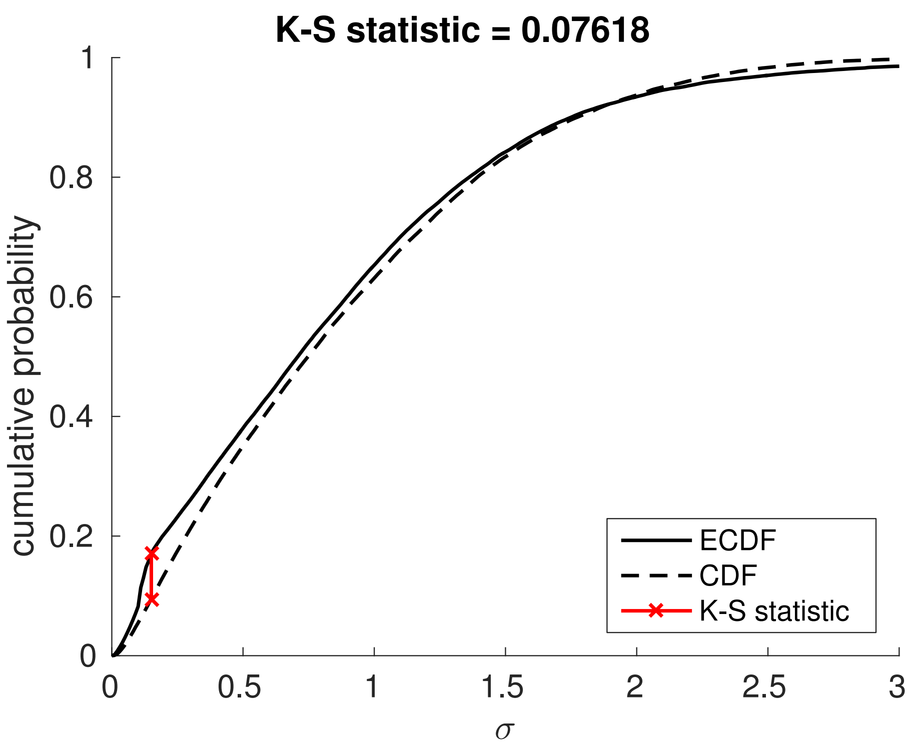


Figure 7.14: Comparing the sounding rocket evaluation and sampled trajectory data.

expected behaviour. An example of this is visible when looking at the time-series in figure 7.13, where during $\tau = [0.3, 0.55]$ the second stage is activated for the training data. As the training data shows many trajectories at a higher altitude than the mean, the probabilistic model assumes there are also trajectories at a lower altitude, causing the samples to dive. However, based on the physical model at the core of the simulator, the sounding rockets would require unusual circumstances to make such a manoeuvre.

### 7.2.3 Evaluation via K-S statistic

Similar to the previous experiment, the performance of the model is evaluated according to the technique discussed in section 3.3.2. For this experiment, the ECDF is created using 2000 trajectories from the *evaluation* dataset. Figure 7.14 displays both the ECDF (evaluation trajectory data) and the CDF (sampled trajectory data), showing that the K-S statistic is 0.076, corresponding to a maximum deviation of 7.6%. This reduction in performance corresponds with the differences seen in the training data and sampled data.

## 7.3 An integrated approach via Teetool

The analysis as seen in section 7.2.2 indicated that while the probabilistic modelling approach is able to model the rocket flight trajectories, changes in flight phases (e.g., start of the second stage, deployment of the parachutes) cause an abrupt change in the trajectory data. This section shows how Teetool, the trajectory analysis tool introduced in section 3.6, fits into the existing architecture and discusses how knowledge available in the rocket simulator is used to implement an effective method of time-warping. Furthermore, various scenarios are compared and their difference is *quantified* using Teetool.

### 7.3.1 Integration and synchronisation

Previously, in section 7.1.2 the process of capturing the uncertainty of an input variable in the output via the Monte Carlo method is described. However, as the input variables are sampled independently from a normal distribution, it is possible to end up with launch- and atmospheric conditions that correspond with a low probability (i.e., multiple input variables are far away from their nominal value). As the Monte Carlo method simulates more and more conditions, eventually one of these instances will result in an unstable rocket flight, but this one outlier will be a bad representation of the whole. To capture these effects, the trajectory analysis method is implemented to model the spatial distribution of the rocket trajectories as produced by the simulator. This allows for an unrestricted sampling space as the input, which is useful because there is an unknown influence of the variables on the output, while obtaining the spatial statistics based on the unfiltered output from the simulator.

The trajectory data is translated into a mean trajectory and the dispersion from the mean trajectory is captured via the modelling approach as presented in chapter 3. However, a modified time-warping approach is described here to handle the abrupt changes seen in the rocket trajectories (e.g., start of the second stage, deployment of the parachutes). The trajectory analysis package integrates with the rocket simulator output as seen in figure 7.15.

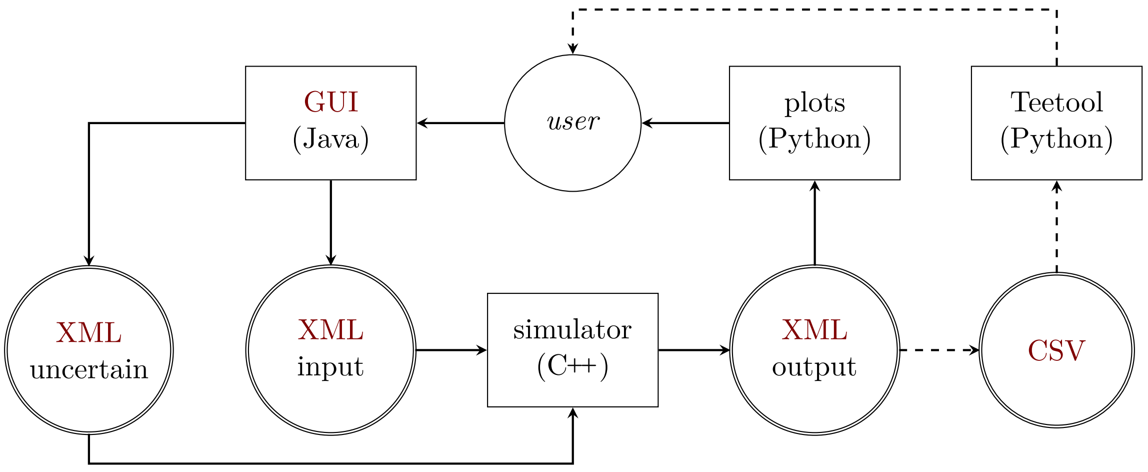


Figure 7.15: Schematic representation of the components and the corresponding programming languages. The dashed lines represent a path specific to the third version.

As the data are produced by the simulator, information about position, time, and the particular stage of flight are available. By definition, all trajectories start at the normalised time $\tau = 0$, and end at $\tau = 1$. There are time-warping methods available to align the behaviour of the trajectories (Müller 2007), however, such a method is not required here as the moments that these events occur are well known. These are the moments where there is a change from one stage of flight to the next, for example when the rocket reaches apogee. By synchronizing these moments in the normalised time vector $\boldsymbol{\tau}$, the time-warping is complete. For this reason the label EVENT_INT, an event identifier (e.g., apogee reached), is included in the Comma Separated Values (CSV) file, as it refers to a particular stage of the flight. Hence, the normalised time vector $\boldsymbol{\tau}$ to accompany the coordinate vector $\mathbf{v}$ is equal to:

$$\boldsymbol{\tau} = \begin{bmatrix} 0 & \frac{1}{M} & \dots & \frac{M}{M} \end{bmatrix}^{\intercal} \tag{7.1}$$

where the points in time when the trajectory changes from one stage of flight to the next are aligned. The values of these points in time are set by the nominal flight. For example, if the nominal flight reaches apogee at $\tau = 0.3$, all trajectories reach apogee at $\tau = 0.3$.

There are several advantages to this approach. Firstly, it is shown how a probabilistic model can capture the general trend and dispersion seen in the trajectory data. The direct application here is the ability to visualise the behaviour of a large dataset without cluttering the screen. Besides helpful in visualising, it also allows *quantifying* the difference between the scenarios. The latter is done by evaluating the number of grid-points found within the one standard deviation ($\sigma = 1$) region for multiple scenarios. Changes in these points reflect changes in the trend modelled from the trajectory data.

### 7.3.2 Visualising trajectories via Teetool

This section demonstrates the effects of integrating Teetool with the rocket simulator, in particular the advantages of the time-warping approach introduced in the previous section.

For the scenario seen in this section, 500 trajectories are generated via the Monte Carlo method as described in section 7.1.2. It shows the results of modelling the output data as a Gaussian process, using the additional information on the events as discussed in section 7.3.1. All 500 trajectories produced by the rocket flight simulator are used to learn the corresponding probabilistic model, however, to prevent visual clutter only 50 of the 500 trajectories are shown in the figures.

The flight path trajectories seen in figure 7.16 are generated using the settings seen in figure 7.6. In this simulation the wind conditions represent a light northerly wind (no more than 2.2 m/s), and the parachute deploys at an altitude of 750 m. Specific launch conditions are a declination angle of 10 degrees, and an azimuth angle of 0 degrees (i.e., launched towards the North). The volume identifying the trend corresponds with the one standard deviation region ($\sigma = 1$).

For comparison, two situations are considered, one with the data synchronized, and another without the data synchronized. This results in two probabilistic models $p(\mathbf{y})$, from which each 50 samples are generated. The sampled data $\mathbf{S}$ is presented in figure 7.17, including the one standard deviation region. Of particular interest are the effects near the parachute deployment, where when the data is not synchronized, *kinks* occur in an otherwise smooth trajectory towards the ground level.

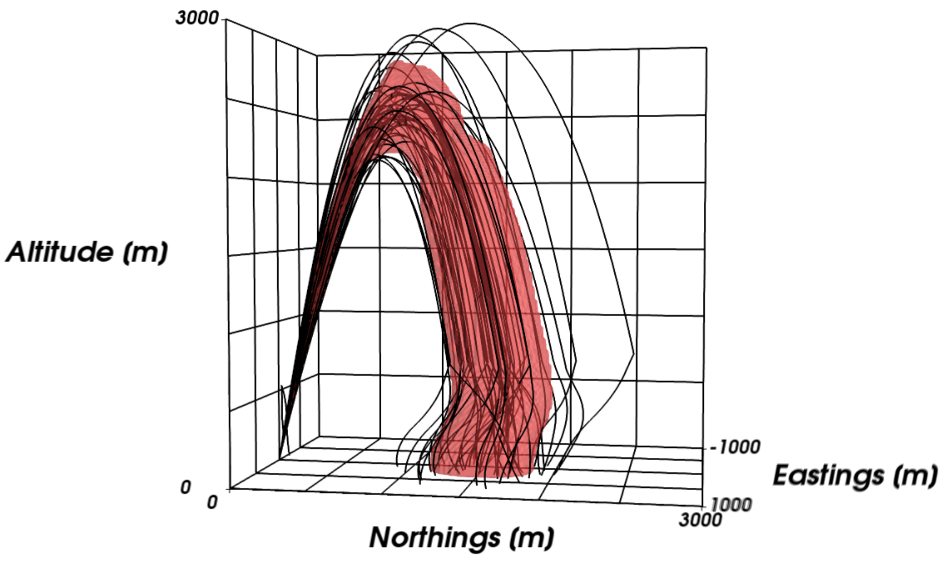
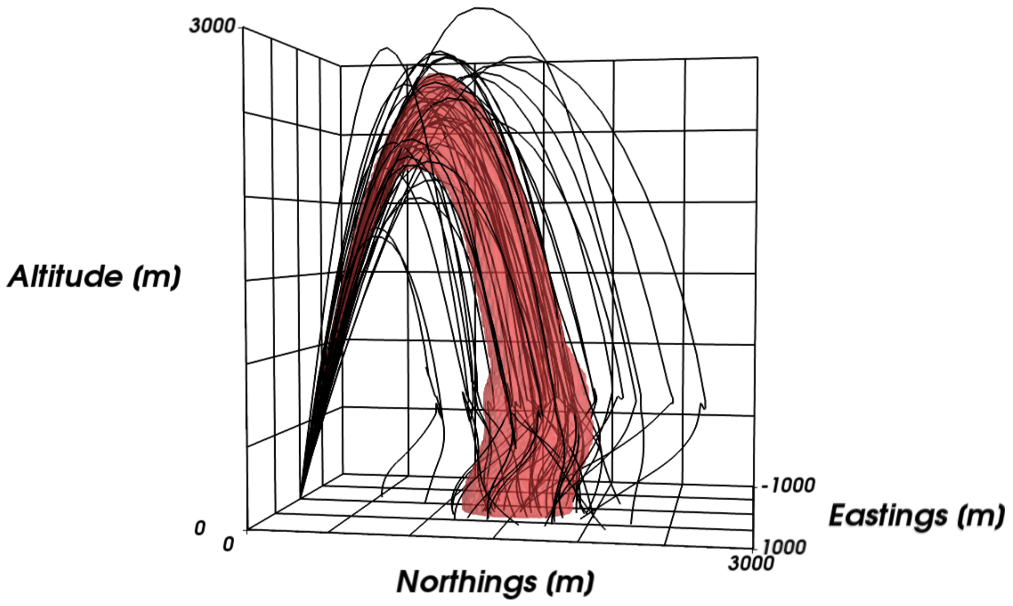


Figure 7.16: Rocket trajectory data $\mathbf{Y}$ produced by standard launch conditions, and the one standard deviation region.

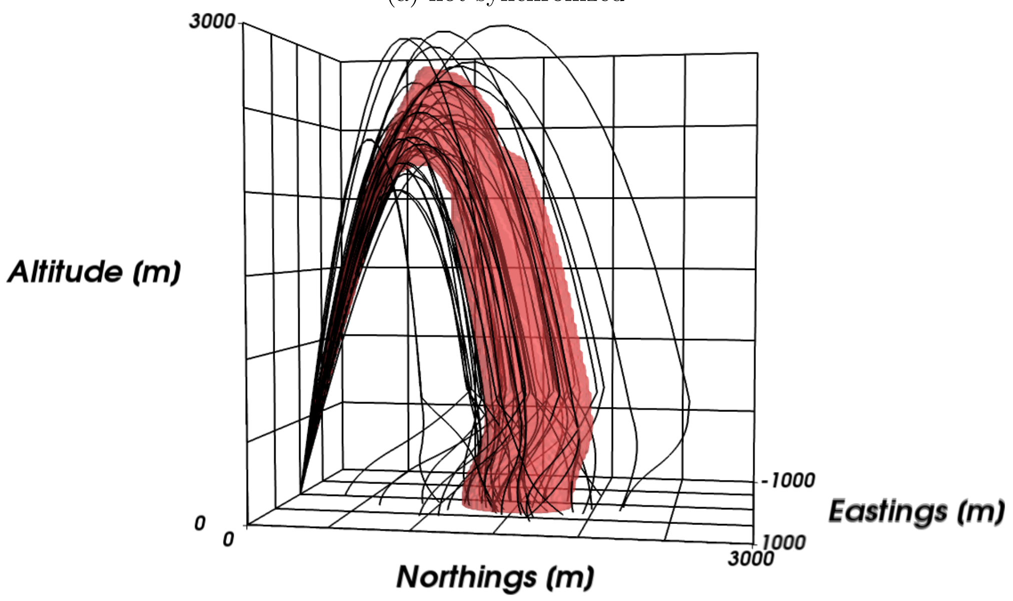
These effects are investigated in further detail by examining the altitude time-series of both cases in figures 7.18 and 7.19. While the trajectory data $\mathbf{Y}$ is identical in both cases, the associated normalized time vectors $\boldsymbol{\tau}$ are altered, hence the three dimensional visualisations showing $\mathbf{Y}$ are identical and these time-series are different. These results show that the sampled trajectory data $\mathbf{S}$ of the synchronized trajectory data $\mathbf{Y}$, visualised in figure 7.19d, show no artefacts due to the abrupt change.

### 7.3.3 Comparing scenarios

This section demonstrates the effects of the stochastic input variables to the output trajectory data and how the trajectory analysis tool assists in visualising and quantifying the differences. Similar to the scenario in the previous section, for each of the three additional scenarios, 500 trajectories are generated via the Monte Carlo method as described in section 7.1.2. It shows the results of modelling the output data as a Gaussian process, using the additional information on the events as discussed in section 7.3.1. All 500 trajectories produced by the rocket flight simulator are used to learn

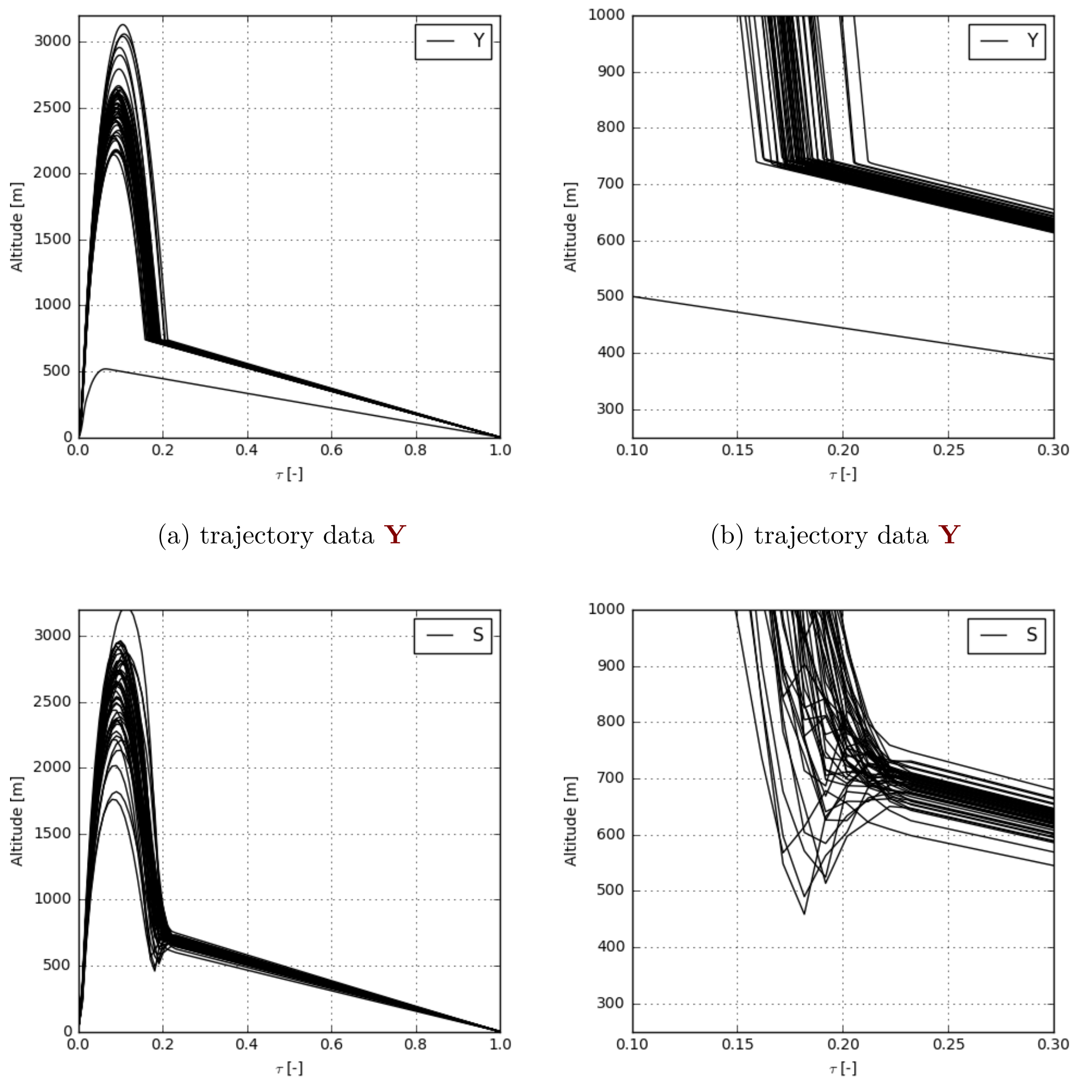


(a) not synchronized



(b) synchronized

Figure 7.17: Rocket trajectory sampled data **S** and the one standard deviation region.



(a) trajectory data **Y**

(b) trajectory data **Y**

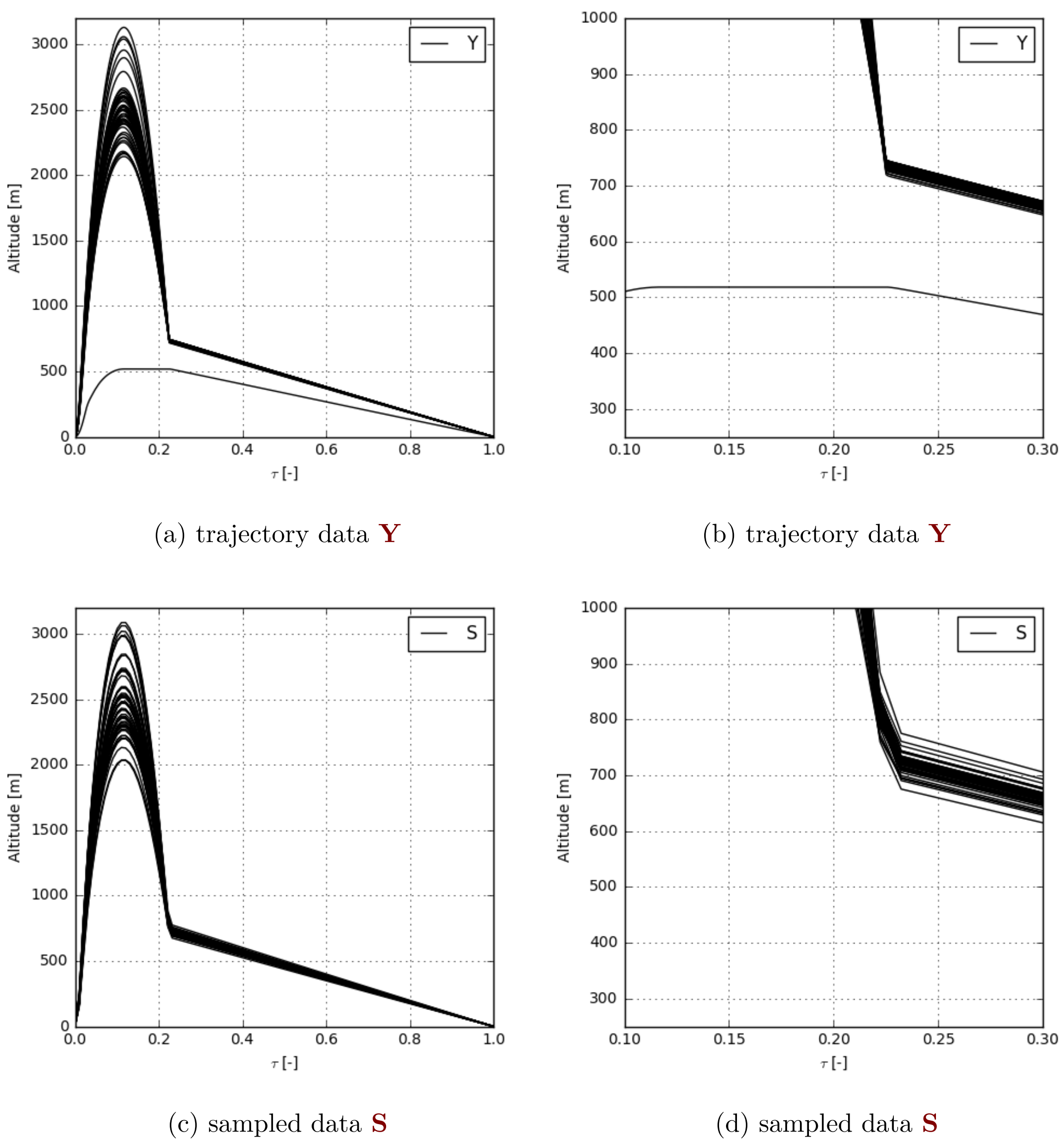
(c) sampled data **S**

(d) sampled data **S**

Figure 7.18: Rocket flight trajectory data shown as time-series, where abrupt changes are not synchronized.

the corresponding probabilistic model, however, to prevent visual clutter only 50 of the 500 trajectories are shown in the figures.

The first alternate scenario changes the wind-conditions to a strong southerly wind (with an average wind speed up to 13.5 m/s). An overview of both the northerly and southerly wind-conditions is available in figure 7.20. Northerly and southerly are merely labels, and these wind-conditions represent real data measured by a weather balloon. The x-wind and y-wind components correspond with the easting and northing directions respectively, where both vary as a function of altitude. Both wind-conditions have a



(a) trajectory data **Y** (b) trajectory data **Y**

(c) sampled data **S** (d) sampled data **S**

Figure 7.19: Rocket flight trajectory data shown as time-series, where abrupt changes are synchronized.

0 m/s vertical wind component. The results are visible in figure 7.21, where again 50 trajectories and the one standard deviation region ($\sigma = 1$) are shown. The comparison plot with the standard launch is seen in figure 7.22. The blocks used to construct this volume have a volume of $20 \times 20 \times 20$ m$^3$, and the blocks relating to the addition, removal, and lack of change in the standard launch versus the southerly wind scenario are coloured green, red, and blue respectively. While the declination angle of the launch tower is 10 degrees in both cases, the southerly wind has a steeper trajectory. This is due to an effect called weathercocking, which was a key part of the validation of the

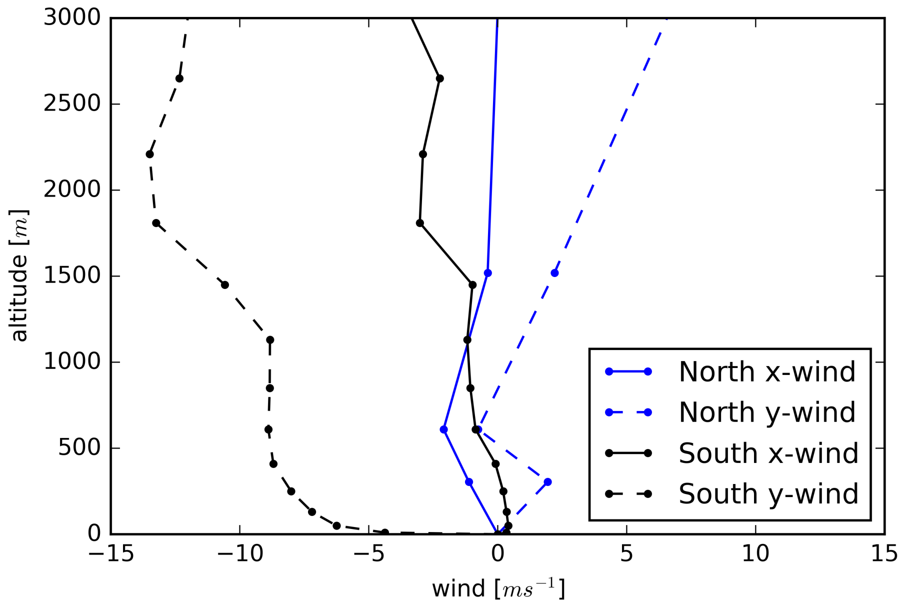


Figure 7.20: Wind conditions north(erly) wind versus south(erly) wind.

rocket simulator (Box, Bishop and Hunt 2010).

The second alternate scenario introduces a failure to deploy the parachute, resulting in a fully ballistic flight trajectory. The resulting flight trajectories are visible in figure 7.23. The comparison with the standard launch is visible in figures 7.24 and 7.25, whereas expected, the final part of the trajectory shows the difference between the parachute- and ballistic landing. The failure to deploy the parachute is only visible in the final 750 m of the descent, as this is where the parachute deploys. Figure 7.26 shows a two-dimensional slice of the one standard deviation region at a constant altitude for both 750 m and 300 m. It also includes a two-dimensional projection of the mean trajectory belonging to both scenarios, represented by a dashed line. The dispersion seen until the 750 m is almost (it *is* an approximation via a Monte-Carlo method) identical. The models diverge after this point, which is visible in figure 7.26b.

Finally, the third alternate scenario introduced a 5% variation in the thrust curve, according to the method discussed in section 7.1.2. This method increases/decreases the thrust curve over the entire duration, where the percentage is sampled from a normal distribution. Similar to the other plots, the 50 trajectories and the one standard deviation region are visible in figure 7.27. The comparison plot is seen in figure 7.28, indicating

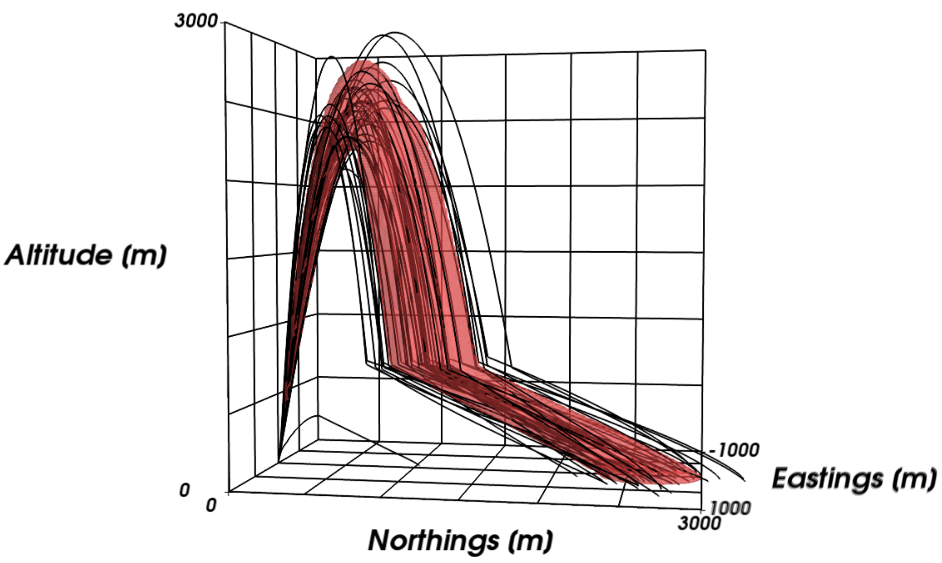


Figure 7.21: Trajectories produced by southerly wind conditions, and the one standard deviation region.

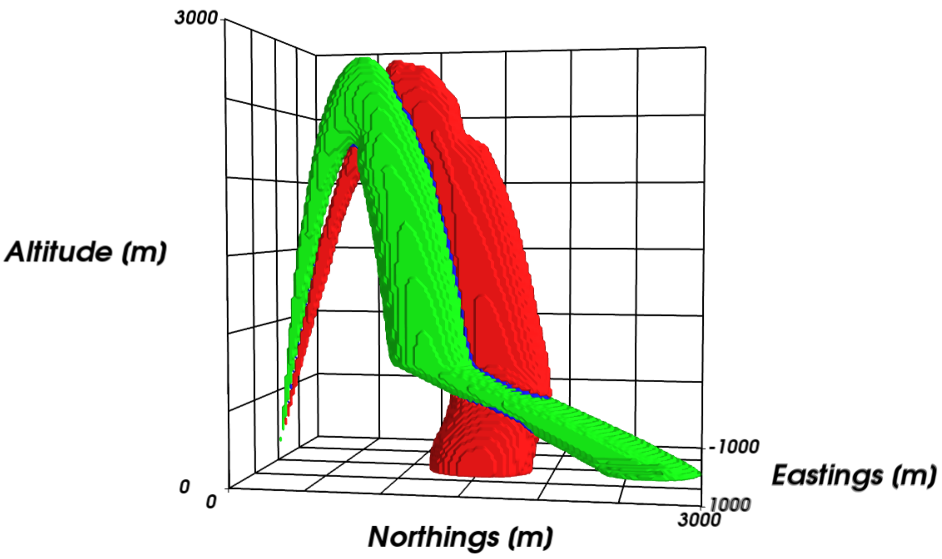


Figure 7.22: Comparison of the one standard deviation region in the standard launch conditions versus the southerly wind conditions.

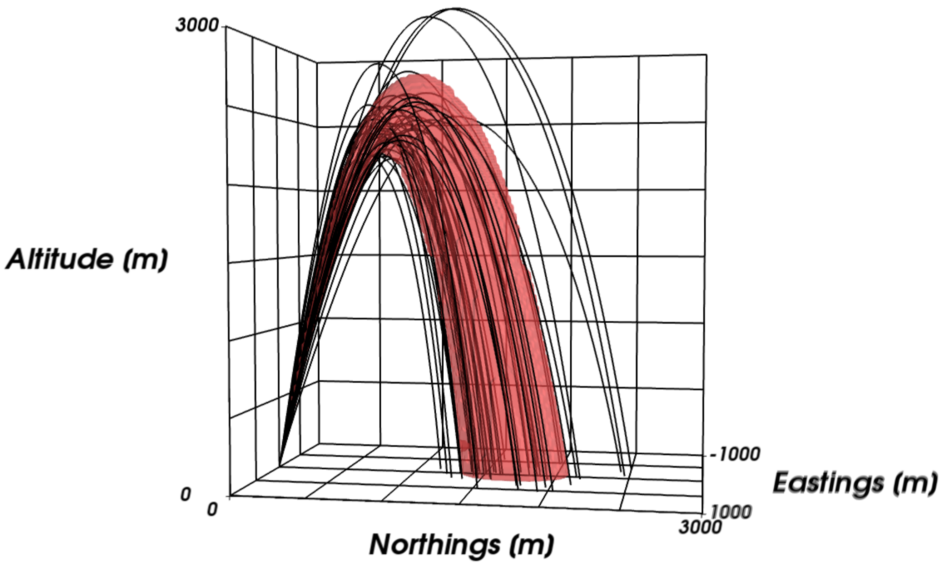


Figure 7.23: Trajectories produced with the parachute failure condition, and the one standard deviation region.

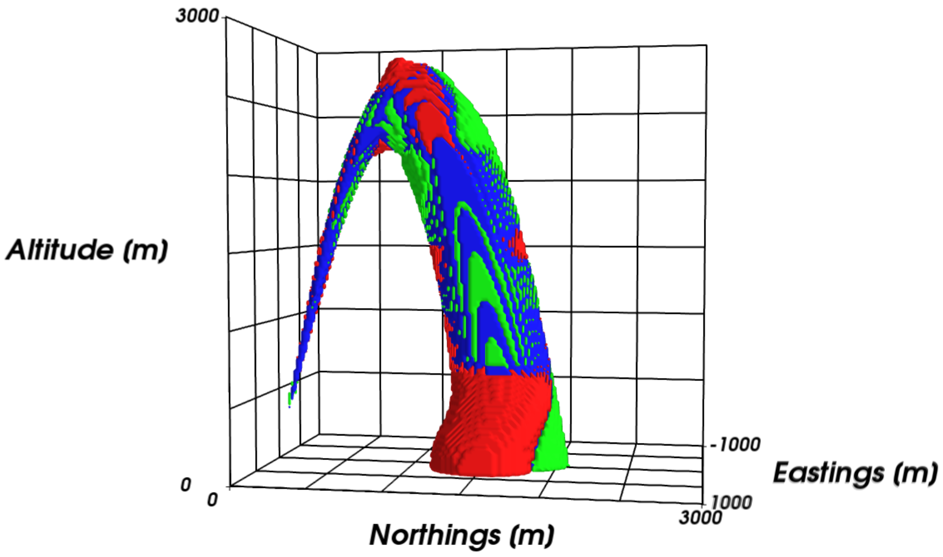


Figure 7.24: Comparison of the one standard deviation region in the standard launch conditions versus the parachute failure condition.

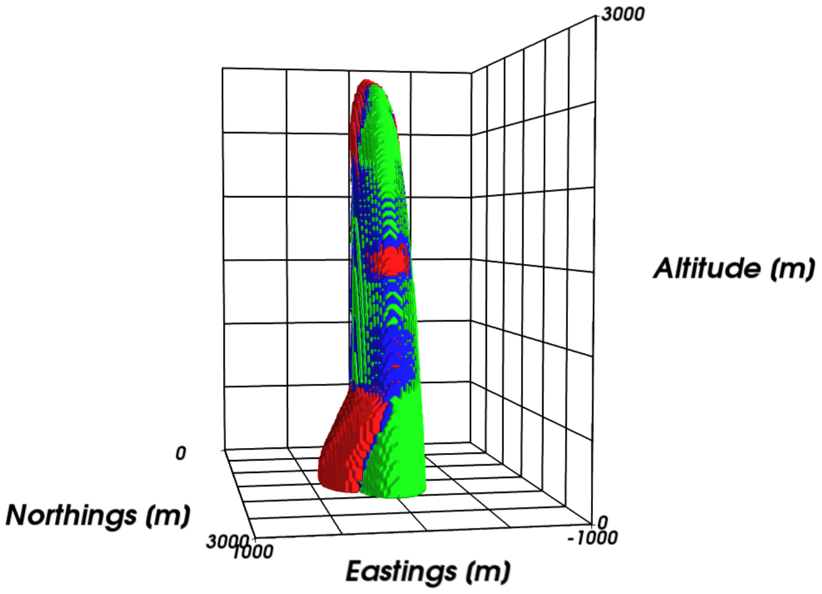
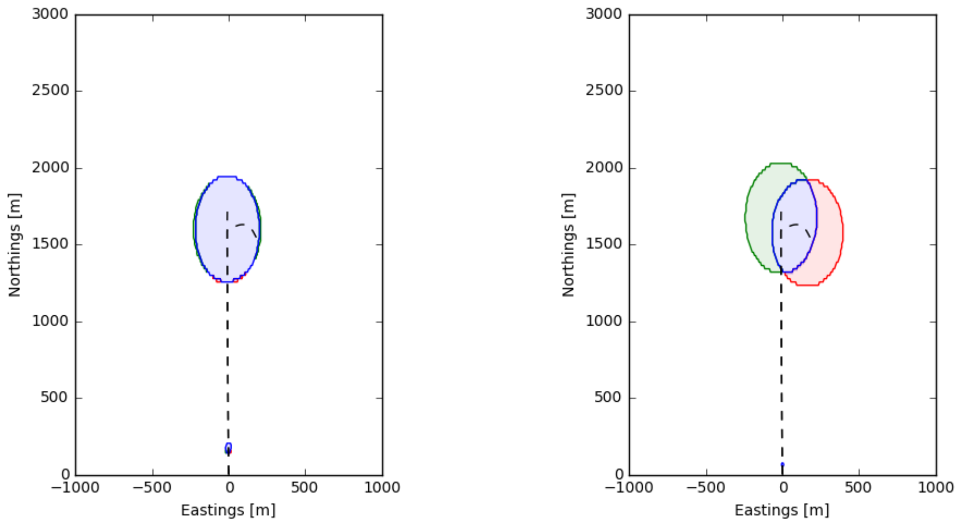


Figure 7.25: Comparison of the one standard deviation region in the standard launch conditions versus the parachute failure condition, alternate view-point.



(a) 750 m (b) 300 m

Figure 7.26: One standard deviation region at a constant altitude, comparing the standard launch conditions versus the parachute failure condition.

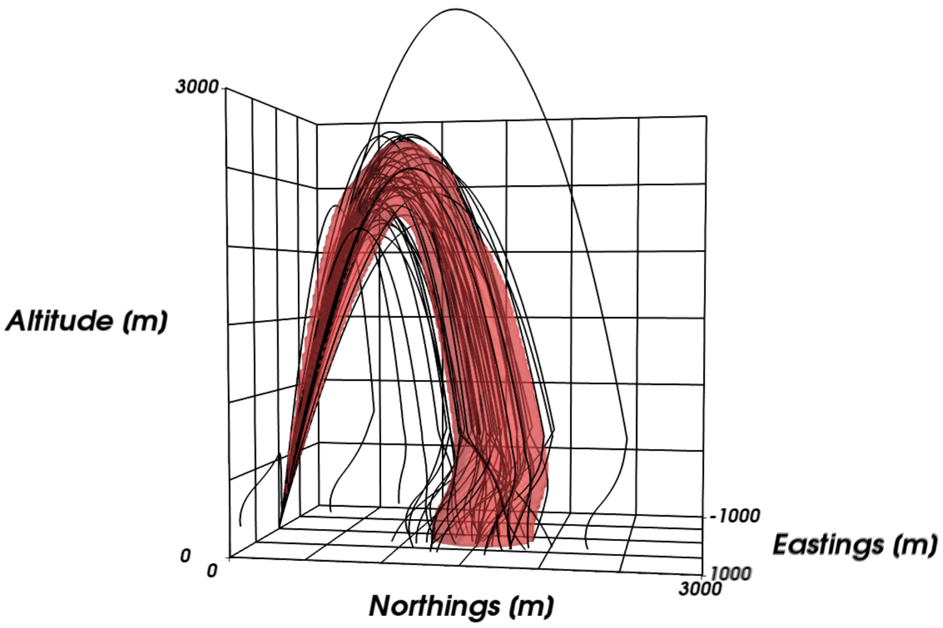


Figure 7.27: Trajectories produced with the variable thrust curve condition, and the one standard deviation region.

that the thrust has a strong correlation with the steepness of the launch trajectory, an effect that can be attributed to weathercocking. The two-dimensional projection of the one standard deviation region is visible in figure 7.29. At a constant altitude of 2100 m (figure 7.29a) the effect of the thrust variation on the dispersion in the launch phase is pronounced, however, closer to the ground, at 500 m (figure 7.29b) the differences in divergence between the two scenarios have reduced.

The numerical results that quantify the changes seen between the scenarios are visible in table 7.2. To demonstrate the sensitivity of the block size, the percentages have been calculated at varying sizes of 100 m, 50 m, and 20 m in all three dimensions (easting, northing, altitude). The percentages are related to the total number of blocks identified to be inside the one standard deviation region for the standard conditions, i.e. the region as seen in figure 7.16, hence the percentage *removed* and *unchanged* adds up to 100%. The difference in the percentage added and removed relates to the uncertainty region expanding or shrinking with respect to the standard conditions. Based on the percentage unchanged, the variable thrust curve shows the least change, followed by the parachute failure conditions, and the changed wind condition shows most change.

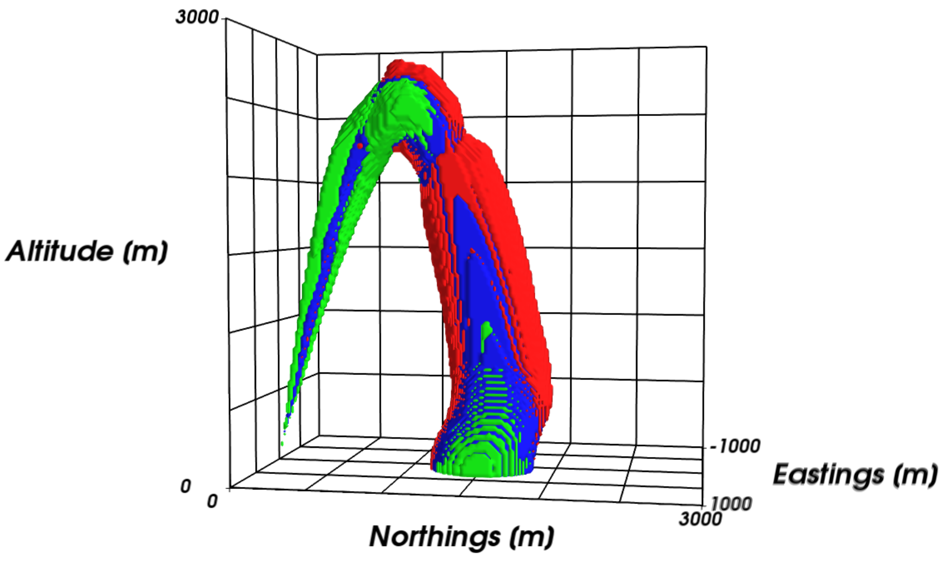
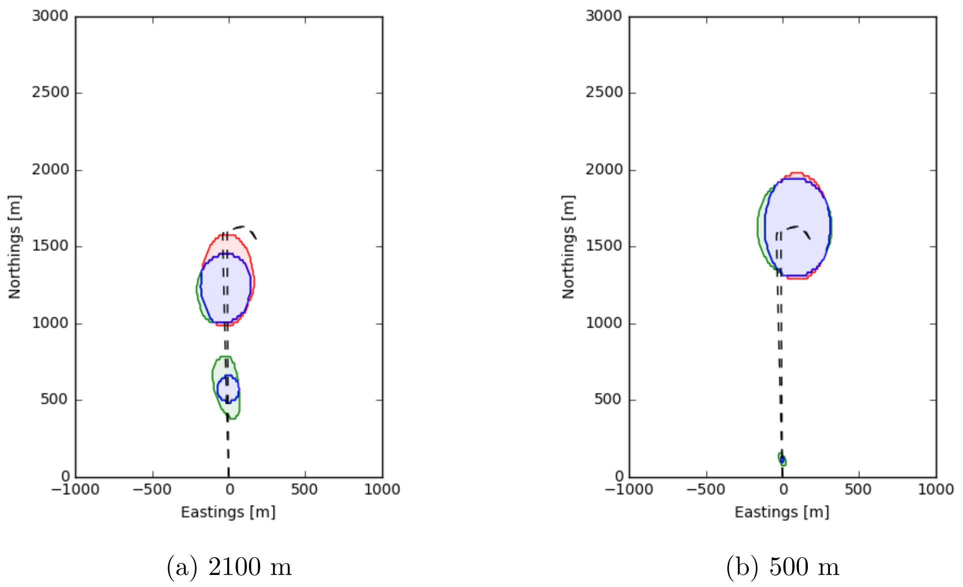


Figure 7.28: Comparison of the one standard deviation region in the standard launch conditions versus the variable thrust curve condition.



(a) 2100 m (b) 500 m

Figure 7.29: One standard deviation region at a constant altitude, comparing the standard launch conditions versus the variable thrust curve condition.

Table 7.2: An overview of results when quantifying changes against the standard conditions at various block volumes.

| **Block volume [$m^3$]** | **Added** | **Removed** | **Unchanged** |
|---|---|---|---|
| *versus southerly wind conditions* | | | |
| $100 \times 100 \times 100$ | 79.7% | 78.1% | 21.9% |
| $50 \times 50 \times 50$ | 76.1% | 78.5% | 21.5% |
| $20 \times 20 \times 20$ | 75.4% | 78.3% | 21.7% |
| *versus the parachute failure condition* | | | |
| $100 \times 100 \times 100$ | 22.4% | 15.7% | 84.3% |
| $50 \times 50 \times 50$ | 20.3% | 17.5% | 82.5% |
| $20 \times 20 \times 20$ | 20.7% | 18.1% | 81.9% |
| *versus the variable thrust curve condition* | | | |
| $100 \times 100 \times 100$ | 21.1% | 8.1% | 91.9% |
| $50 \times 50 \times 50$ | 18.2% | 8.6% | 91.4% |
| $20 \times 20 \times 20$ | 17.6% | 8.6% | 91.4% |

## 7.4 Conclusion

This chapter has introduced a stochastic, six-degrees-of-freedom rocket simulator. An overview of the architecture and insight into how to the uncertainty is propagated throughout the simulator is provided. Results show that special care needs to be taken at those moment where an abrupt change occurs. The exact moment when these changes occur are known when simulating the dynamics, and by integrating the trajectory analysis package Teetool into the simulator these anomalies can be avoided. Teetool has also demonstrated the ability to quantify changes between different scenaries set-up in the simulator.

# Chapter 8

# Conclusions

This chapter summarises the conclusions provided by the research. Also, it offers suggestions for work to be done in the future, combining experiences gained and recent developments.

## 8.1 Concluding remarks

First, chapter 2 identified the absence of a structured approach to perform a probabilistic analysis of trajectory data. Existing methods depend on ad hoc methods, and implementations using the Gaussian process framework are incomplete, or have a specific focus on clustering.

In chapter 3, a probabilistic modelling method for multi-dimensional trajectory data was proposed. This method makes use of the Gaussian process framework, providing the mathematical foundation. Due to this framework it is able to cope with noisy and missing data. Besides being able to learn the probabilistic model, the method can also generate new trajectory data. These data are then used in the performance evaluation as the Kolmogorov-Smirnov (K-S) statistic. Also, an open-source Python toolbox based on the trajectory analysis technique is introduced, increasing the accessibility and reusability of the work.

The method is applied to an ensemble of commercial aircraft trajectories departing from Dallas Fort Worth airport (DFW) airport in chapter 4. The performance evaluation of the K-S statistic between the measured (by radar) and sampled data (generated via the probabilistic model) showed a maximum deviation of 13.1%. Further analysis showed that by combining the method with a clustering algorithm, a large dataset of trajectories covering multiple discrete flight paths may be analysed.

Finally, chapter 5 explored methods to optimise the evaluation of security threats in urban and rural environments. Results indicated that evaluating the locations based on the surface height data increases the chances of detecting dangerous areas by at least 20%. And, by using a piecewise linear interpolation method, specific regions of the map can be prioritised in the evaluation. On average this leads to a complete detection of all dangerous regions in less than half the original time required.

Chapter 6 reviewed rocket flight simulator software, and rocket dynamics in research. It showed that there are developments in quantifying the uncertainty in the landing position of a rocket in the research, and models are available to evaluate conceptual rocket designs. However, the available software lacks the combination of all aspects; assisting conceptual design via a Graphical User Interface (GUI), evaluate rocket performance, and quantify the uncertainty in the landing position.

Chapter 7 introduced a stochastic six-degrees-of-freedom rocket flight simulator. This software assists the user in their rocket design, and is able to predict the landing location with confidence bounds, based on uncertainty in both rocket dynamics and atmospheric conditions. Also, by integrating the Python package Teetool, a probabilistic model can be created from these trajectory data produced by the simulator. This allows for a probabilistic analysis of the rocket trajectory data through all phases of the flight. Such an analysis is shown to be useful in visualising trends and quantifying differences between various scenarios.

The main accomplishments of this thesis are:

- proposing a new approach to trajectory analysis, using Gaussian processes to deal with noisy measurements and missing data.

- creating a Python package containing this new approach, increasing the accessibility and reusability of the work.
- proposing methods to optimise the evaluation of security threats, increasing the detection rate by at least 20% and decreasing the time to completion to half the original time required.
- creating a single piece of software that is able to assist users in their rocket design and predict the landing location with confidence bounds. This software is available for download and tested for Ubuntu 14.04 and Windows 10.

## 8.2 Future work

Three suggestions are offered here to continue the work. The first two tasks relate to the proposed trajectory analysis method as proposed in chapter 3, and how it can be further developed. The third task relates to the future development of the stochastic rocket simulator.

### 8.2.1 Expand modelling method to include non-Gaussian distributions

The proposed method assumes a Gaussian distribution of the parameters found in the probabilistic model. This assumption has advantages in terms of computational cost when learning the parameters from historical data, and generating new data by sampling from the model. Additionally, by assuming a Gaussian relation, the geometric representation of the probabilistic model up to three dimensions are easily interpreted and visualised. However, it is possible to capture other distributions within the model using sampling methods. While this will lead to approximations via sampling methods and thus increasing the computational costs involved, the probabilistic model might be a better fit than the Gaussians. A promising distribution would be the student's t-distribution, due to the ability to handle outliers well. Another promising distribution would be the Rayleigh distribution, a distribution often used to characterize the average wind velocity. The main argument here is that the distribution is asymmetric, this property is also seen in other distributions, e.g., log-normal. In order to obtain a visual representation, it is possible to also use sampling methods, however, it is expected to be very costly to then use these samples to visualise confidence regions as seen before. The brute-force method via sampling methods will be too computationally costly – however, possibly suitable approximation schemes exist.

### 8.2.2 Integrate trajectory analysis method with a aircraft flight trajectory database

The trajectory analysis method as proposed in this dissertation has shown to be effective at modelling trajectories that demonstrate a similar motion pattern. To demonstrate this, trajectory data are clustered based on a common flightpath. However, there are many other parameters that dictate the runway use and the planned flightpath. For example, destination, current wind direction, and aircraft type. By integrating data sources of aircraft flight trajectory data (e.g., `www.flightradar24.com`), the flight data can be clustered based on these other parameters. The results based on such an analysis will provide an excellent overview, suitable for generating situational awareness of the airspace surrounding an airport. Although, in order to obtain access to the historical trajectory data, a paid subscription to the service is required. The work associated with the task includes setting up data handling schemes and implementing new clustering algorithms. These clustering algorithms will be very specific to the available data and the question the analysis should answer.

### 8.2.3 A web-based rocket simulator with an integrated database

The work as presented in this dissertation produced an open-source stochastic six-degrees-of-freedom rocket flight trajectory simulator capable quantifying the uncertainty in the landing location. As the software is open-source, developers are able to make contributions, and researchers are able to use the whole, or parts of the software in their research. Furthermore, in order to improve the accessibility, installers are available for Windows and Linux at `https://sourceforge.net/projects/camrocsim/`. However, in the current form, the project holds three separate languages; C++, Java, and Python. By rewriting the code base in a single language, the project will be more accessible for developers, as few are able (or desire to do so) to work with three different coding languages. The proposed language here would be Python, as this coding language is more commonly taught at universities. The majority of the work here will be in developing a GUI. The proposed method here would be to use a web-based interface. This automatically supports hosting the rocket simulator online, making it even more accessible to the public and rocket enthusiasts. And, if the simulator is hosted online, it is possible to include an integrated database. This database can include common rocket designs, and, if the users desire to do so, they can upload their own design. This database may also include rocket motors with the corresponding thrust curves, as standard, off the shelf rocket motors are often used. An example of a web-based interface can be found at `astra-planner.soton.ac.uk/`, which is a simulator that assists with the flight planning of high altitude balloons. The integration with weather and mapping data would also be desirable.

# Appendix A

# The Gaussian probability distribution

## A.1 The univariate Gaussian distribution

The equation belonging to the univariate Gaussian distribution is:

$$p(x) = \mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} \tag{A.1}$$

where he parameters $\mu$ and $\sigma^2$ can be learned from the data by maximizing the likelihood. For further details of the mathematics involved (maximizing the negative log-likelihood with respect to these two parameters), please consult Bishop (2006), or any other book that includes a basic introduction to statistics. The resulting equations are:

$$\mu = \frac{1}{N} \sum_{n=1}^{N} x_n \tag{A.2}$$

$$\sigma^2 = \frac{1}{N} \sum_{n=1}^{N} (x_n - \mu)^2 \tag{A.3}$$

Where the mean $\mu$ corresponds to the average of all samples $x_n$, and variance $\sigma^2$ corresponds to the squared distance between the individual samples $x_n$ and the mean $\mu$.

## A.2 The multivariate Gaussian distribution

The equation belonging to the multivariate Gaussian distribution is:

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\Delta^2)\right) \tag{A.4}$$

$$\Delta^2 = (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \tag{A.5}$$

with $D$ as the dimensionality, and where $\Delta$ represents the Mahalanobis distance, which at constant values, can represented the Gaussian as an ellipse (two-dimensional), an ellipsoid (three-dimensional), or a hypersphere (higher-dimensional). These are referred to as confidence intervals, as a given percentage of samples are expected to fall within this region. Furthermore, the mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ can be learned via maximum likelihood using the following equations:

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n \tag{A.6}$$

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{n=1}^{N} \{(\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^{\intercal}\} \tag{A.7}$$

where in a two-dimensional example ($D = 2$), the variable takes the form:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{A.8}$$

with the resulting parameters:

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \tag{A.9}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} {\sigma^2}_{1,1} & {\sigma^2}_{1,2} \\ {\sigma^2}_{2,1} & {\sigma^2}_{2,2} \end{bmatrix} \tag{A.10}$$

# Appendix B

# Equal spaced radial basis functions

This section introduces a radial basis function method applied to equal spaced data. Radial basis function methods are important tools for interpolating scattered data (Sarra 2017). In some cases there are a particular set of basis functions best suited to model the output function. However, in the case where this is not known we propose using a set of equal spaced radial basis functions. The particular shape is used in the function, here it is also known as a *Gaussian basis function* (not related to a probability function) and takes the shape of:

$$\phi_k(\tau) = \exp\left(\frac{-(\tau - c_k)^2}{r_k^2}\right) \tag{B.1}$$

with $c_k$ as the centre location and $r_k$ as the optimal width for that particular basis function. The equation for optimal width for *uniformly* distributed data is found in Nakayama, Arakawa and Sasaki (2002) to be:

$$r_1 = r_2 = \ldots = r_k = \frac{d_{max}}{\sqrt[n]{nK}} = \frac{d_{max}}{K} \tag{B.2}$$

where $n = 1$, representing the single dimension of the data, $d_{max}$ denotes the maximal distance among the data and $K$ is the number of Gaussian basis functions.

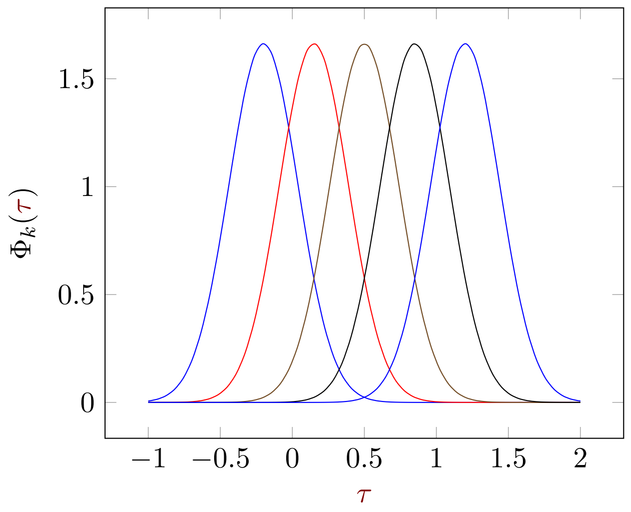


Figure B.1: Example of $K = 5$ over $\tau = [0, 1]$, including the extra width.

In practice, in order not to have less approximation power along the borders $\tau = \{0, 1\}$, as a rule of thumb, an extra $1/K$ width is added on either side when placing the centre locations. An example where $K = 5$ and the data ranges $\tau = [0, 1]$, is visible in figure B.1. The basis functions are spaced uniformly at $c_k = \{-0.2, 0.15, 0.5, 0.85, 1.2\}$ and $r_k = d_{max}/K = 1.4/5 = 0.28$.

Finally, there is also an additional *bias* parameter (not a 'bias' in a statistical sense) added to correct for any fixed offset of the data. Therefore, when considering a model of $J$ basis functions (also referred to as model complexity $J$), there are $K = J - 1$ Gaussian basis functions.

# Bibliography

Albrecht, G., H.-T. Lee and A. Pang (2012). 'Visual Analysis of Air Traffic Data Using Aircraft Density and Conflict Probability'. In: *Infotech @ Aerospace*. DOI: 10.2514/6.2012-2540.

Angluin, D. (1988). 'Queries and concept learning'. In: *Machine learning* 2.4, pp. 319–342. DOI: 10.1007/BF00116828.

Ankerst, M., M. M. Breunig, H.-P. Kriegel and J. Sander (1999). 'Optics: Ordering points to identify the clustering structure'. In: *ACM Sigmod Record*. Vol. 28. ACM, pp. 49–60. ISBN: 1581130848. DOI: 10.1145/304182.304187.

Annoni, R. and C. H. Q. Forster (2012). 'Analysis of Aircraft Trajectories Using Fourier Descriptors and Kernel Density Estimation'. In: *IEEE Conference on Intelligent Transportation Systems (ITSC), 2012 15th International*. IEEE, pp. 1441–1446. ISBN: 9781467330633.

Antonio, D. D. (1986). 'A missile flyout model for ISEAS'. PhD thesis. Monterey, California: Naval postgraduate school.

Apogee (2008). *RockSim v9*. URL: http://www.apogeerockets.com (visited on 31/01/2017).

Bak, P. et al. (2015). 'Visual analytics for movement behavior in traffic and transportation'. In: *IBM Journal of Research and Development* 59.2/3, 10:1–10:12. ISSN: 0018-8646. DOI: 10.1147/JRD.2015.2400252.

Baraff, D. (1997). 'An introduction to physically based modeling: Rigid body simulation I – Unconstrained rigid body dynamics'. In: *SIGGRAPH '97 Tutorial notes*.

Barber, C. B., D. P. Dobkin and H. Huhdanpaa (1996). 'The Quickhull algorithm for convex hulls'. In: *ACM Transactions on Mathematical Software* 22.4, pp. 469–483. DOI: 10.1145/235815.235821.

Bartholomew, D. J. and D. J. Bartholomew (1967). *Stochastic models for social processes*. Wiley London.

Bartlett, M. S. (1960). 'Stochastic population models in ecology and epidemiology'. In:

Baum, E. B. (1991). 'Neural net algorithms that learn in polynomial time from examples and queries'. In: *IEEE Transactions on Neural Networks* 2.1, pp. 5–19. ISSN: 1045-9227. DOI: 10.1109/72.80287.

Berndt, D. and J. Clifford (1994). 'Using Dynamic Time Warping to Find Patterns in Time Series'. In: *Proceedings of KDD Workshop*. Vol. 10. Seattle, WA, pp. 359–370.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer. ISBN: 9780387310732. URL: https://books.google.co.uk/books?id=kTNoQgAACAAJ.

Boeker, E. R. et al. (2008). 'Integrated noise model (INM) version 7.0 technical manual'. In: *Washington, DC, Federal Aviation Administration, Office of Environment and Energy*.

Box, G. E. P. and N. R. Draper (1987). *Empirical model-building and response surface*. Wiley series in probability and mathematical statistics: Applied probability and statistics. New York : Wiley, c1987. ISBN: 0471810339. URL: https://books.google.com/books?id=QO2dDRufJEAC.

Box, S., C. M. Bishop and H. Hunt (2010). 'A stochastic six-degree-of-freedom flight simulator for passively controlled high power rockets'. In: *Journal of Aerospace Engineering* 24.1, pp. 31–45. ISSN: 0893-1321. DOI: 10.1061/(ASCE)AS.1943-5525.0000051. URL: http://eprints.soton.ac.uk/73938/.

Box, S. and W. Eerland (2017). *Cambridge Rocketry Simulator*. URL: https://sourceforge.net/projects/camrocsim/ (visited on 10/06/2017).

Buchanan, G. et al. (2015). 'The Development of Rocketry Capability in New–Zealand World Record Rocket and First of Its Kind Rocketry Course'. In: *Aerospace* 2.1, pp. 91–117. ISSN: 2226-4310. DOI: 10.3390/aerospace2010091.

Buchin, K., M. Buchin, J. Gudmundsson, M. Löffler and J. Luo (2011). 'Detecting commuting patterns by clustering subtrajectories'. In: *International Journal of Computational Geometry & Applications* 21.03, pp. 253–282.

Buschmann, S., M. Trapp and J. Döllner (2014). ‘Real-Time Animated Visualization of Massive Air-Traffic Trajectories’. In: *Cyberworlds (CW), 2014 International Conference on*, pp. 174–181. DOI: 10.1109/CW.2014.32.

Camargo, S. J., A. W. Robertson, S. J. Gaffney, P. Smyth and M. Ghil (2007). ‘Cluster analysis of typhoon tracks. Part I. General properties’. In: *Journal of Climate* 20.14, pp. 3635–3653. ISSN: 08948755. DOI: 10.1175/JCLI4188.1.

Chudinov, P. S. (2003). ‘Analytical investigation of point mass motion in midair’. In: *European Journal of Physics* 25.1, p. 73. DOI: 10.1088/0143-0807/25/1/010.

Colucci, F. (2008). *Beating Ballistic Threats*. URL: http://www.aviationtoday.com/2008/10/01/beating-ballistic-threats/ (visited on 10/04/2017).

Commercial Space Transportation (2007). *Supplemental Application Guidance for Unguided Suborbital Launch Vehicles (USLVs)*. Tech. rep. Federal Aviation Administration (FAA). URL: https://www.faa.gov/.

Constantinescu, E. M., V. M. Zavala, M. Rocklin, S. Lee and M. Anitescu (2011). ‘A Computational Framework for Uncertainty Quantification and Stochastic Optimization in Unit Commitment With Wind Power Generation’. In: *IEEE Transactions on Power Systems* 26.1, pp. 431–441. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2010.2048133.

Cox, G. E., G. Kachergis and R. M. Shiffrin (2012). ‘Gaussian Process Regression for Trajectory Analysis’. In: *Proceedings of the 34th Annual Conference of the Cognitive Science Society*, pp. 1440–1445.

Cressie, N. (1993). *Statistics for Spatial Data: Wiley Series in Probability and Statistics*. Wiley: New York, NY, USA.

Crispin, C. and A. Sóbester (2015). ‘An Intelligent, Heuristic Path Planner for Multiple Agent Unmanned Air Systems’. In: *AIAA Infotech @ Aerospace*. Kissimmee, Florida, USA: American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.2015-0361.

– (2016). ‘Co-operation in an Autonomous, Decentralised, Unmanned Air System for Atmospheric Research’. In: *AIAA Infotech @ Aerospace*. San Diego, California, USA: American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.2016-1410.

Cunning Running Software Ltd. (2017). *Threat Visualisation for the Real World.* URL: www.cunningrunning.co.uk (visited on 17/07/2017).

Diez, M., W. He, E. F. Campana and F. Stern (2014). 'Uncertainty quantification of Delft catamaran resistance, sinkage and trim for variable Froude number and geometry using metamodels, quadrature and Karhunen–Loève expansion'. In: *Journal of Marine Science and Technology* 19.2, pp. 143–169. ISSN: 1437-8213. DOI: 10.1007/s00773-013-0235-0.

Doucet, A., N. de Freitas and N. Gordon (2001). *An Introduction to Sequential Monte Carlo Methods.* Ed. by A. Doucet, N. de Freitas and N. Gordon. New York, NY: Springer New York. ISBN: 978-1-4757-3437-9. DOI: 10.1007/978-1-4757-3437-9_1.

Eckstein, A. (2009). 'Automated flight track taxonomy for measuring benefits from performance based navigation'. In: *Integrated Communications, Navigation and Surveillance Conference, 2009. ICNS'09.* IEEE, pp. 1–12. DOI: 10.1109/ICNSURV.2009.5172835.

Eerland, W. J. and S. Box (2016). 'Trajectory Clustering, Modeling and Selection with the focus on Airspace Protection'. In: *AIAA Infotech @ Aerospace.* San Diego, California, USA: American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.2016-1411. eprint: http://eprints.soton.ac.uk/386955/.

Eerland, W. J., S. Box, H. Fangohr and A. Sóbester (2017a). 'A Gaussian process based decision support tool for air traffic management'. In: *AIAA conference proceedings.* Denver, Colorado, USA: American Institute of Aeronautics and Astronautics, p. 16. DOI: 10.2514/6.2017-4264. eprint: http://eprints.soton.ac.uk/408419/.

– (2017b). 'An open-source, stochastic, six-degrees-of-freedom rocket flight simulator, with a probabilistic trajectory analysis approach'. In: *AIAA Modeling and Simulation Technologies Conference.* Grapevine, Texas, USA: American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.2017-1556. eprint: http://eprints.soton.ac.uk/403364/.

– (2017c). 'Teetool – a probabilistic trajectory analysis tool'. In: *Journal of Open Research Software* 5.1, p. 6. DOI: 10.5334/jors.163. eprint: https://eprints.soton.ac.uk/408279/.

Eerland, W. J., S. Box and A. Sóbester (2016). ‘Modeling the Dispersion of Aircraft Trajectories Using Gaussian Processes’. In: *Journal of Guidance, Control, and Dynamics* 39.12, pp. 2661–2672. DOI: 10.2514/1.G000537. eprint: http://eprints.soton.ac.uk/399818/.

– (2017). ‘Cambridge Rocketry Simulator – A Stochastic Six-Degrees-of-Freedom Rocket Flight Simulator’. In: *Journal of Open Research Software* 5.1, p. 5. DOI: 10.5334/jors.137. eprint: http://eprints.soton.ac.uk/405278/.

Ellis, D., E. Sommerlade and I. Reid (2009). ‘Modelling pedestrian trajectory patterns with Gaussian processes’. In: *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pp. 1229–1234. DOI: 10.1109/ICCVW.2009.5457470.

Elmqvist, N. and P. Tsigas (2008). ‘A Taxonomy of 3D Occlusion Management for Visualization’. In: *IEEE Transactions on Visualization and Computer Graphics* 14.5, pp. 1095–1109. ISSN: 1077-2626. DOI: 10.1109/TVCG.2008.59.

Engelen, F. (2012). ‘Quantitative risk analysis of unguided rocket trajectories’. Technical University Delft. URL: http://resolver.tudelft.nl/uuid:2dfae5a6-809e-484d-8b3e-ebd146d891f3.

Ester, M., H. P. Kriegel, J. Sander and X. Xu (1996). ‘A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise’. In: *Second International Conference on Knowledge Discovery and Data Mining*, pp. 226–231. ISBN: 1577350049.

European Civil Aviation Conference (2016). *Doc 29 4th Edition Report on Standard Method of Computing Noise Contours around Civil Airports*. Tech. rep. European Civil Aviation Conference (ECAC). URL: https://www.ecac-ceac.org/ecac-docs (visited on 27/07/2017).

Faraway, J. J., M. P. Reed and J. Wang (2007). ‘Modelling three-dimensional trajectories by using Bézier curves with application to hand motion’. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 56.5, pp. 571–585. ISSN: 1467-9876. DOI: 10.1111/j.1467-9876.2007.00592.x.

Ferreira, N., J. T. Klosowski, C. E. Scheidegger and C. T. Silva (2013). ‘Vector Field k-Means: Clustering Trajectories by Fitting Multiple Vector Fields’. In: *Computer*

*Graphics Forum*. Vol. 32. 3. Wiley Online Library, pp. 201–210. DOI: 10.1111/cgf.12107.

Gaffney, S. and P. Smyth (1999). ‘Trajectory clustering with mixtures of regression models’. In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining KDD 99*. Vol. 10. ACM, pp. 63–72. ISBN: 1581131437. DOI: 10.1145/312129.312198.

Gariel, M., A. N. Srivastava and E. Feron (2011). ‘Trajectory clustering and an application to airspace monitoring’. In: *IEEE Transactions on Intelligent Transportation Systems* 12.4, pp. 1511–1524. ISSN: 15249050. DOI: 10.1109/TITS.2011.2160628.

Gelfand, A. E., P. Diggle, P. Guttorp and M. Fuentes (2010). *Handbook of spatial statistics*. CRC press.

Girvin, R. (2009). ‘Aircraft noise-abatement and mitigation strategies’. In: *Journal of Air Transport Management* 15.1, pp. 14–22. ISSN: 09696997. DOI: 10.1016/j.jairtraman.2008.09.012.

Gomez, F. J. and R. Miikkulainen (2003). ‘Active Guidance for a Finless Rocket Using Neuroevolution’. In: *Genetic and Evolutionary Computation — GECCO 2003: Genetic and Evolutionary Computation Conference Chicago, IL, USA, July 12–16, 2003 Proceedings, Part II*. Ed. by E. Cantú-Paz et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 2084–2095. ISBN: 978-3-540-45110-5. DOI: 10.1007/3-540-45110-2_105.

Grabbe, S., B. Sridhar and A. Mukherjee (2014). ‘Clustering Days and Hours with Similar Airport Traffic and Weather Conditions’. In: *Journal of Aerospace Information Systems* 11.11, pp. 751–763. DOI: 10.2514/1.I010212.

Green, L. L., H.-Z. Lin and M. R. Khalessi (2002). ‘Probabilistic methods for uncertainty propagation applied to aircraft design’. In: *20th AIAA Applied Aerodynamics Conference*. St. Louis, Missouri, USA: American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.2002-3140.

Greene, M. S., Y. Liu, W. Chen and W. K. Liu (2011). ‘Computational uncertainty analysis in multiresolution materials via stochastic constitutive theory’. In: *Computer Methods in Applied Mechanics and Engineering* 200.1, pp. 309–325. DOI: 10.1016/j.cma.2010.08.013.

Hartigan, J. A. and M. A. Wong (1979). 'Algorithm AS 136: A k-means clustering algorithm'. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1, pp. 100–108. DOI: 10.2307/2346830.

Hastie, T., R. Tibshirani and J. Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer New York. URL: https://books.google.co.uk/books?id=tVIjmNS3Ob8C.

Heatrow (2017). *Runway alternation – Periods of relief from aircraft noise*. URL: http://www.heathrow.com/noise/heathrow-operations/runway-alternation (visited on 17/07/2017).

Helbing, D. and P. Molnár (1995). 'Social force model for pedestrian dynamics'. In: *Physical Review E* 51.5, pp. 4282–4286. ISSN: 1063651X. DOI: 10.1103/PhysRevE.51.4282.

Hu, W. H. W. et al. (2006). 'A system for learning statistical motion patterns'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.9, pp. 1450–1464. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2006.176.

Hunter, J. D. (2007). 'Matplotlib: A 2D graphics environment'. In: *Computing In Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

Hurter, C., O. Ersoy and A. Telea (2012). 'Graph Bundling by Kernel Density Estimation'. In: *Computer Graphics Forum* 31.3pt1, pp. 865–874. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2012.03079.x.

Hurter, C., S. Conversy, D. Gianazza and A. Telea (2014). 'Interactive image-based information visualization for aircraft trajectory analysis'. In: *Transportation Research Part C: Emerging Technologies* 47, pp. 207–227.

Hwang, I. and C. E. Seah (2008). 'Intent-Based Probabilistic Conflict Detection for the Next Generation Air Transportation System'. In: *Proceedings of the IEEE* 96.12, pp. 2040–2059. ISSN: 0018-9219. DOI: 10.1109/JPROC.2008.2006138.

Hwang, J. N., J. J. Choi, S. Oh and R. J. Marks (1991). 'Query-based learning applied to partially trained multilayer perceptrons'. In: *IEEE Transactions on Neural Networks* 2.1, pp. 131–136. ISSN: 1045-9227. DOI: 10.1109/72.80299.

IBM (2016). *What is big data?* URL: https://www.ibm.com/software/data/bigdata/what-is-big-data.html (visited on 10/05/2016).

Iman, R. L. and W. J. Conover (1980). 'Small sample sensitivity analysis techniques for computer models with an application to risk assessment'. In: *Communications in Statistics – Theory and Methods* 9.17, pp. 1749–1842. DOI: 10.1080/03610928008827996.

Jardin, M. R. (2005). 'Analytical Relationships Between Conflict Counts and Air-Traffic Density'. In: *Journal of Guidance, Control, and Dynamics* 28.6, pp. 1150–1156. DOI: 10.2514/1.12758.

Johnson, M., L. Moore and D. Ylvisaker (1990). 'Minimax and maximin distance designs'. In: *Journal of Statistical Planning and Inference* 26.2, pp. 131–148. ISSN: 0378-3758. DOI: 10.1016/0378-3758(90)90122-B.

Jones, E., T. Oliphant, P. Peterson et al. (2001). *SciPy: open source scientific tools for Python*. URL: http://www.scipy.org (visited on 31/01/2017).

Jopson, I., D. Rhodes and P. Havelock (2002). *Aircraft noise model validation – How accurate do we need to be?* Tech. rep. UK Civil Aviation Authority.

Justel, A., D. Peña and R. Zamar (1997). 'A multivariate Kolmogorov-Smirnov test of goodness of fit'. In: *Statistics & Probability Letters* 35.3, pp. 251–259. ISSN: 01677152. DOI: 10.1016/S0167-7152(97)00020-5.

Kashiwagi, Y. (1968). *Prediction of ballistic missile trajectories*. Tech. rep. DTIC Document.

Kennedy, D. (2016). *Stochastic Financial Models*. Chapman and Hall/CRC Financial Mathematics Series. CRC Press. ISBN: 9781439882719. URL: https://books.google.co.uk/books?id=MDrOBQAAQBAJ.

Khalil, M., X. T. Rui, Q. C. Zha and H. Hendy (2014). 'Investigation on Spin-Stabilized Projectile Trajectory Observability Based on Flight Stability'. In: *Advances in Measurements and Information Technologies*. Vol. 530. Applied Mechanics and Materials. Trans Tech Publications, pp. 175–180. DOI: 10.4028/www.scientific.net/AMM.530-531.175.

King, P. H., J. Scanlan and A. Sóbester (2015). 'From Radiosonde to Papersonde: The Use of Conductive Inkjet Printing in the Massive Atmospheric Volume Instrumentation System (MAVIS) Project'. In: *AIAA Infotech @ Aerospace*. Kissimmee, Florida, USA: American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.2015-0985.

Klein, P. P. (2012). 'On the Ellipsoid and Plane Intersection Equation'. In: *Applied Mathematics* 3.11, p. 1634. DOI: 10.4236/am.2012.311226.

Kluyver, T. et al. (2016). 'Jupyter Notebooks - a publishing format for reproducible computational workflows'. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas: Proceedings of the 20th International Conference on Electronic Publishing*. IOS Press, pp. 87–90. DOI: 10.3233/978-1-61499-649-1-87.

Krozel, J. and D. Andrisani (2006). 'Intent Inference with Path Prediction'. In: *Journal of Guidance, Control, and Dynamics* 29.2, pp. 225–236. ISSN: 0731-5090. DOI: 10.2514/1.14348.

Laudeman, I. V., S. Shelden, R. Branstrom and C. Brasil (1998). *Dynamic density: An air traffic management metric*. Monograph. National Aeronautics and Space Administration Moffett Field, CA 94035 USA: Ames Research Center. URL: https://ntrs.nasa.gov/search.jsp?R=19980210764.

Lee, J.-G., J. Han and K.-Y. Whang (2007). 'Trajectory clustering: a partition-and-group framework'. In: *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, pp. 593–604.

Lieske, R. F. and R. L. McCoy (1964). *Equations of motion of a rigid projectile*. Tech. rep. URL: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=AD0441598 (visited on 15/04/2017).

Lieske, R. F. and M. L. Reiter (1964). *Equations of motion for a modified point mass trajectory*. Tech. rep. URL: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=AD0441598 (visited on 15/04/2017).

Liu, W., X. Chong, P. Huang and N. I. Badler (2014). 'Learning motion patterns in unstructured scene based on latent structural information'. In: *Journal of Visual Languages & Computing* 25.1. Special Issue on Multimedia Information Processing, pp. 43–53. ISSN: 1045-926X. DOI: 10.1016/j.jvlc.2013.10.005.

Lu, L. et al. (2014). 'Visual Analysis of Uncertainty in Trajectories'. English. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by V. S. Tseng, T. B. Ho, Z.-H. Zhou, A. L. Chen and H.-Y. Kao. Vol. 8443. Lecture Notes in Computer Science. Springer International Publishing, pp. 509–520. ISBN: 978-3-319-06607-3. DOI: 10.1007/978-3-319-06608-0_42.

MacKay, D. J. C. (1992a). 'Information-Based Objective Functions for Active Data Selection'. In: *Neural Computation* 4.4, pp. 590–604. ISSN: 0899-7667. DOI: 10.1162/neco.1992.4.4.590.

– (1992b). 'The Evidence Framework Applied to Classification Networks'. In: *Neural Computation* 4.5, pp. 720–736. ISSN: 0899-7667. DOI: 10.1162/neco.1992.4.5.720.

– (2003). *Information theory, inference, and learning algorithms.* Vol. 7. Citeseer.

Makris, D. and T. Ellis (2005). 'Learning semantic scene models from observing activity in visual surveillance'. In: *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 35.3, pp. 397–408. DOI: 10.1109/TSMCB.2005.846652.

Marzuoli, A., C. Hurter and E. Feron (2012). 'Data visualization techniques for airspace flow modeling'. In: *Conference on Intelligent Data Understanding (CIDU).* IEEE, pp. 79–86. DOI: 10.1109/CIDU.2012.6382187.

McKinney, W. (2010). 'Data Structures for Statistical Computing in Python'. In: *Proceedings of the 9th Python in Science Conference.* Ed. by S. van der Walt and J. Millman, pp. 51–56. URL: http://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf.

Menon, P. K., G. D. Sweriduk and K. D. Bilimoria (2004). 'New Approach for Modeling, Analysis, and Control of Air Traffic Flow'. In: *Journal of Guidance, Control, and Dynamics* 27.5, pp. 737–744. DOI: 10.2514/1.2556.

Meratnia, N. and R. A. de By (2002). 'Aggregation and comparison of trajectories'. In: *Proceedings of the 10th ACM international.* ACM, pp. 49–54. ISBN: 1581135912. DOI: 10.1145/585147.585158.

Morris, B. T. and M. M. Trivedi (2009). 'Learning trajectory patterns by clustering: Experimental studies and comparative evaluation'. In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009.* IEEE, pp. 312–319. ISBN: 9781424439935. DOI: 10.1109/CVPRW.2009.5206559.

– (2011). 'Trajectory Learning for Activity Understanding: Unsupervised, Multilevel, and Long-Term Adaptive Approach'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.11, pp. 2287–2301. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2011.64.

– (2013). 'Understanding vehicular traffic behavior from video: a survey of unsupervised approaches'. In: *Journal of Electronic Imaging* 22.4, pp. 041113–041113. DOI: 10.1117/1.JEI.22.4.041113.

Morris, M. D. and T. J. Mitchell (1995). 'Exploratory designs for computational experiments'. In: *Journal of Statistical Planning and Inference* 43.3, pp. 381–402. ISSN: 0378-3758. DOI: 10.1016/0378-3758(94)00035-T.

Müller, M. (2007). 'Information Retrieval for Music and Motion'. In: Berlin, Heidelberg: Springer Berlin Heidelberg. Chap. Dynamic Time Warping, pp. 69–84. ISBN: 978-3-540-74048-3. DOI: 10.1007/978-3-540-74048-3_4.

Nakayama, H., M. Arakawa and R. Sasaki (2002). 'Simulation-based optimization using computational intelligence'. In: *Optimization and Engineering* 3.2, pp. 201–214. ISSN: 1389-4420. DOI: 10.1023/A:1020971504868.

Nassiri, N., J. Roushanian and S. Haghighat (2004). 'Stochastic Flight Simulation Applied to a Sounding Rocket'. In: *International Astronautical Congress (IAF)*. American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.IAC-04-A.1.07.

Nikolaidis, E., D. Ghiocel and S. Singhal (2004). *Engineering Design Reliability Handbook*. CRC Press. ISBN: 9780203483930. URL: https://books.google.co.uk/books?id=gdHKBQAAQBAJ.

Niskanen, S. (2013). *OpenRocket technical documentation*. Tech. rep. URL: http://openrocket.sourceforge.net (visited on 31/01/2017).

– (2015). *OpenRocket*. URL: http://openrocket.sourceforge.net (visited on 31/01/2017).

Oladyshkin, S. and W. Nowak (2012). 'Data-driven uncertainty quantification using the arbitrary polynomial chaos expansion'. In: *Reliability Engineering & System Safety* 106, pp. 179–190.

Padrón, A. S., J. J. Alonso, F. Palacios, M. Barone and M. S. Eldred (2014). 'Multi-fidelity uncertainty quantification: application to a vertical axis wind turbine under an extreme gust'. In: *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. Atlanta, Georgia, USA.

Park, J. and I. W. Sandberg (1991). 'Universal approximation using radial-basis-function networks'. In: *Neural computation* 3.2, pp. 246–257. DOI: 10.1162/neco.1991.3.2.246.

Paulson, C., T. Kluyver, N. Pezolano and M. Jacob (2017). *pyKriging.* DOI: 10.5281/zenodo.800702.

Prandini, M., J. Hu, J. Lygeros and S. Sastry (2000). 'A probabilistic approach to aircraft conflict detection'. In: *IEEE Transactions on Intelligent Transportation Systems* 1.4, pp. 199–220. ISSN: 1524-9050. DOI: 10.1109/6979.898224.

Prandini, M., L. Piroddi, S. Puechmorel and S. L. Brazdilova (2011). 'Toward Air Traffic Complexity Assessment in New Generation Air Traffic Management Systems'. In: *IEEE Transactions on Intelligent Transportation Systems* 12.3, pp. 809–818. ISSN: 1524-9050. DOI: 10.1109/TITS.2011.2113175.

Prandini, M., V. Putta and J. Hu (2010). 'A probabilistic measure of air traffic complexity in 3-D airspace'. In: *International Journal of Adaptive Control and Signal Processing* 24.10, pp. 813–829. ISSN: 1099-1115. DOI: 10.1002/acs.1192.

Press, W., S. Teukolsky, W. Vetterling and B. Flannery (2007). *Numerical Recipes.* Cambridge University Press.

Ramachandran, P. and G. Varoquaux (2011). 'Mayavi: 3D Visualization of Scientific Data'. In: *Computing in Science & Engineering* 13.2, pp. 40–51. ISSN: 1521-9615. DOI: 10.1109/MCSE.2011.35.

Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian Processes for Machine Learning.* MIT Press. URL: http://mitpress.mit.edu/catalog/item/default.asp?ttype=2%5C&tid=10930.

Reynolds, T. G. and R. J. Hansman (2002). 'Conformance monitoring approaches in current and future air traffic control environments'. In: *Proceedings. The 21st Digital Avionics Systems Conference.* Vol. 2, pp. 1–12. DOI: 10.1109/DASC.2002.1052922.

– (2003). 'Analyzing Conformance Monitoring in Air Traffic Control Using Fault Detection Approaches & Operational Data'. In: American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.2003-5574.

– (2005). 'Investigating Conformance Monitoring Issues in Air Traffic Control Using Fault Detection Techniques'. In: *Journal of Aircraft* 42.5, pp. 1307–1317. ISSN: 0021-8669. DOI: 10.2514/1.10055.

Rogers, A., M. Osborne, S. D. Ramchurn, S. Roberts and N. R. Jennings (2008). 'Information Agents for Pervasive Sensor Networks'. In: *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pp. 294–299. DOI: 10.1109/PERCOM.2008.22.

Rogers, J. (2014). 'Stochastic Model Predictive Control for Guided Projectiles Under Impact Area Constraints'. In: *Journal of Dynamic Systems, Measurement, and Control* 137.3, pp. 034503–034503-8. DOI: 10.1115/1.4028084.

Rosi, B. V. and J. H. Diekmann (2014). 'Methods for the Uncertainty Quantification of Aircraft Simulation Models'. In: *Journal of Aircraft* 52.4, pp. 1247–1255. DOI: 10.2514/1.C032856.

Sahu, J. (2008). 'Time-accurate numerical prediction of free-flight aerodynamics of a finned projectile'. In: *Journal of Spacecraft and Rockets* 45.5, pp. 946–954. DOI: 10.2514/1.34723.

Salaun, E., M. Gariel, A. E. Vela and E. Feron (2012). 'Aircraft proximity maps based on data-driven flow modeling'. In: *Journal of Guidance, Control, and Dynamics* 35.2, pp. 563–577. DOI: 10.2514/1.53859.

Saleemi, I., K. Shafique and M. Shah (2009). 'Probabilistic Modeling of Scene Dynamics for Applications in Visual Surveillance'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.8, pp. 1472–1485. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2008.175.

Santner, T. J., B. J. Williams and W. I. Notz (2003). *The Design and Analysis of Computer Experiments*. 1st ed. New York, NY: Springer-Verlag New York. DOI: 10.1007/978-1-4757-3799-8.

Sarra, S. (2017). 'The Matlab Radial Basis Function Toolbox'. In: *Journal of Open Research Software* 5.1, p. 8. DOI: 10.5334/jors.131.

Seah, C. E., A. Aligawesa and I. Hwang (2010). 'Algorithm for Conformance Monitoring in Air Traffic Control'. In: *Journal of Guidance, Control, and Dynamics* 33.2, pp. 500–509. ISSN: 0731-5090. DOI: 10.2514/1.44839.

Sorenson, H. (1985). *Kalman Filtering: Theory and Application*. IEEE Press selected reprint series. IEEE Press. ISBN: 9780879421915. URL: https://books.google.co.uk/books?id=2pgeAQAAIAAJ.

Squad (2011). *Kerbal Space Program.* URL: http://www.kerbalspaceprogram.com (visited on 31/01/2017).

Sridhar, B., K. Sheth and S. Grabbe (1998). 'Airspace complexity and its application in air traffic management'. In: *2nd USA/Europe Air Traffic Management R&D Seminar*, pp. 1–6.

Sridhar, B., T. Soni, K. Sheth and G. Chatterji (2006). 'Aggregate Flow Model for Air-Traffic Management'. In: *Journal of Guidance, Control, and Dynamics* 29.4, pp. 992–997. DOI: 10.2514/1.10989.

Strava (2017). *Strava.* URL: https://www.strava.com/ (visited on 27/07/2017).

Sykulski, A. M., S. C. Olhede, J. M. Lilly and E. Danioux (2015). 'Lagrangian time series models for ocean surface drifter trajectories'. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics).* ISSN: 1467-9876. DOI: 10.1111/rssc.12112.

Székely, T. and K. Burrage (2014). 'Stochastic simulation in systems biology'. In: *Computational and Structural Biotechnology Journal* 12.20-21, pp. 14–25. ISSN: 2001-0370. DOI: 10.1016/j.csbj.2014.10.003.

Tay, M. K. C. and C. Laugier (2008). 'Modelling smooth paths using Gaussian processes'. In: *Springer Tracts in Advanced Robotics.* Vol. 42. Springer, pp. 381–390. ISBN: 9783540754039. DOI: 10.1007/978-3-540-75404-6_36.

Taylor, P. and M. P. McAssey (2013). 'An empirical goodness-of-fit test for multivariate distributions'. In: *Journal of Applied Statistics* 40.November, pp. 37–41. DOI: 10.1080/02664763.2013.780160.

Trani, A. A. (2016). *INM - Integrated Noise Model basics.* URL: 128.173.204.63/courses/cee5614/cee5614%5C_pub/inm%5C_basics.pdf (visited on 12/05/2016).

Vourous, G. (2017). *White paper: Data-Driven Aircraft Trajectory Prediction Exploratory Research.* Tech. rep. University of Piraeus, Piraeus, Greece, p. 23. URL: http://dart-research.eu/2017/04/03/dart-white-paper/.

Walt, S. v. d., S. C. Colbert and G. Varoquaux (2011). 'The NumPy Array: A Structure for Efficient Numerical Computation'. In: *Computing in Science & Engineering* 13.2, pp. 22–30. DOI: 10.1109/MCSE.2011.37.

Wilkinson, D. J. (2009). 'Stochastic modelling for quantitative description of heterogeneous biological systems'. In: *Nat Rev Genet* 10.2, pp. 122–133. ISSN: 1471-0056. DOI: 10.1038/nrg2509.

Yan, H., B. Yang, H. Yang and R. Wang (2017). 'Probabilistic Approach to Conformance Monitoring Using Gaussian Processes'. In: *Journal of Guidance, Control, and Dynamics*. DOI: 10.2514/1.G002383.

Yepes, J. L., I. Hwang and M. Rotea (2007). 'New algorithms for aircraft intent inference and trajectory prediction'. In: *Journal of guidance, control, and dynamics* 30.2, pp. 370–382. DOI: 10.2514/1.26750.

Yuan, T., Y. Bar-Shalom, P. Willett and D. Hardiman (2014). 'Impact point prediction for thrusting projectiles in the presence of wind'. In: *Aerospace and Electronic Systems, IEEE Transactions on* 50.1, pp. 102–119. DOI: 10.1117/12.930875.

Zheng, Q. M. and Y. J. Zhao (2012). 'Probabilistic Approach to Trajectory Conformance Monitoring'. In: *Journal of Guidance, Control, and Dynamics* 35.6, pp. 1888–1898. DOI: 10.2514/1.53951.