# Modelling the dispersion of aircraft trajectories using Gaussian processes

Willem J. Eerland*, Simon Box†, András Sóbester‡
*University of Southampton, Southampton, United Kingdom*

DOI: 10.2514/1.G000537

**This work investigates the application of Gaussian processes to capturing the probability distribution of a set of aircraft trajectories from historical measurement data. To achieve this, all data are assumed to be generated from a probabilistic model that takes the shape of a Gaussian process. The approach to Gaussian process modelling used here is based on a linear expansion of trajectory data into set of basis functions that may be parametrized by a multivariate Gaussian distribution. The parameters are learned through maximum likelihood estimation. The resulting probabilistic model can be used for both modelling the dispersion of trajectories along the common flightpath and for generating new samples that are similar to the historical data. The performance of this approach is evaluated using three trajectory datasets; *toy* trajectories generated from a Gaussian distribution, sounding rocket trajectories that are generated by a stochastic rocket flight simulator and aircraft trajectories on a given departure path from Dallas Fort Worth airport, as measured by ground-based radar. The results indicate that the *maximum* deviation between the probabilistic model and test data obtained for the three data sets are respectively** 4.9%**,** 7.6% **and** 13.1%**.**

## Nomenclature

$\beta$ precision parameter

$\hat{\beta}$ precision parameter estimate

$\mathbb{E}$ expectation

$\epsilon$ $\epsilon$-neighborhood parameter of the DBSCAN algorithm

$\phi$ basis function for a scalar

$\mathbf{\Phi}$ block diagonal linear basis

$\boldsymbol{\phi}$ linear basis

$\mathbf{I}$ identity matrix

$\mathbf{k}$ covariance kernel

$\mathcal{L}$ likelihood function

$\mathbf{m}$ mean function

$\boldsymbol{\mu}$ mean vector

$\hat{\boldsymbol{\mu}}$ mean vector estimate

$\mathcal{N}$ Gaussian function

$\mathbf{S}$ dataset of trajectories generated from a probabilistic model

$\mathbf{S}$ covariance matrix of an individual trajectory

$\mathbf{\Sigma}$ covariance matrix

$\hat{\mathbf{\Sigma}}$ covariance matrix estimate

*Postgraduate researcher, University of Southampton, Transportation Research Group, w.j.eerland@soton.ac.uk
†Lecturer, University of Southampton, Transportation Research Group
‡Senior Lecturer, University of Southampton, Astronautics and Computational Engineering Unit

$\sigma$ standard deviation

$\tau$ normalized time

$\boldsymbol{\tau}$ normalized time vector

$t$ time, *seconds*

$\mathbf{t}$ time vector, *seconds*

$w$ single weight

$\mathbf{w}$ weight vector

$x$ single point

$\mathbf{x}$ vector of a single dimension

$\mathbf{Y}$ dataset of $N$ trajectories

$\mathbf{y}$ vector of a single trajectory

# I. Introduction

Technological developments in the last decade have increasingly provided means to gather trajectory data. This has enabled access to an abundance of sources, ranging from tracking cars and aircraft to people and wildlife. A consequence of this 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. Extracting these patterns from aircraft trajectories specifically may be important for air traffic controllers wishing to monitor the highly regulated airspace around airports [1], or environmental policy makers wishing to investigate aircraft noise abatement [2], researchers who predict high-altitude gas balloon trajectories [3], researchers who fly unguided atmospheric sounding rockets and wish to predict safe regions of splashdown [4] and for a range of other applications.

This paper is concerned with the statistical analysis of the dispersion of trajectories from a common path. Given a large set of trajectories that are correlated – for example aircraft following the same flightpath – we would like to *learn* the underlying probabilistic model that effectively characterizes both the common path and the dispersion of the trajectories.

Most existing approaches to modelling the common path and dispersion do so in discrete-space. [5] capture pedestrian movement via visual surveillance, 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 is an envelope with a spline-like representation, capturing a given percentage of trajectories in a two-dimensional space. [6] extended this approach to aircraft trajectories. Here the nodes are again discrete, but in a three-dimensional space, and as a result the cross-section perpendicular to the nodes is no longer a line, but a surface. This is visualised as 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, which is the same approach [5] took, but now seen from two sides (vertical and lateral).

In both [7] and [8] 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.

These papers and corresponding models have already demonstrated the usefulness of assuming that the trajectories are Gaussian distributed. However, one of the limitations is that the data are handled in discrete space. As a result, input data has to be re-sampled, causing the model to only be valid at specific, discrete, points. Now, the trajectory data seen in this paper can, in a more general context, also be called *functional data*. The reason is, that each dimension (e.g. longitude and latitude) can be described as a function of another parameter (e.g. time). Consequently it is possible



This is a pre-print draft of the paper, which was accepted for publication in the Journal of Guidance, Control, and Dynamics on 17/06/2016

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 Gaussian Process (GP) is defined by its mean function $\mathbf{m}(t)$ and covariance kernel $\mathbf{k}(t, t')$, where $t$ and $t'$ are two (possibly multidimensional) values of some functional value (e.g. time). The interested reader is directed to [9] for a complete description of GP. However, in relation to the practical problem at hand, 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 $\mathbf{k}(t, t')$.

In the work by [10], the Gaussian mixture model is used to discover groups in the data (i.e. clustering). 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 section 1.

In [11], 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 [10] 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 [7] and [8]. In conclusion the focus is on clustering, however, when it comes to modelling the individual clusters as a GP, there are other elements to consider, as discussed later in this paper.

Moreover, the authors of [12] use GPR to learn the underlying model structure for further analysis. 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.

In this paper, we propose a method that enables the reconstruction of the probabilistic model that governs a set of aircraft trajectories using the GP technique. The primary goal is to establish a method supported by a theoretical basis, with the purpose of estimating the dispersion seen in a aircraft trajectories that follow a common flightpath. In order to capture this dispersion, a probabilistic model is reconstructed from the trajectory data, hence turning into a data-driven method. The probabilistic model itself is expressed as a Gaussian process, where the mean function $\mathbf{m}(t)$ is described as a continuous function using basis functions. The covariance kernel $\mathbf{k}(t, t')$ is a combination of a linear covariance kernel and a noise term, in which the linear covariance kernel has the property of being non-stationary and the noise term captures the precision of the probabilistic model in emulating the training data. To our knowledge, Gaussian processes have never been applied for capturing the dispersion found in trajectory data to this extent. The application of basis functions in both the mean- and covariance function is particularly suitable when analysing civil aircraft trajectories, as their trajectories tend to be smooth. In other words, when looking at the individual time-series (altitude, latitude and longitude) all the derivatives are continuous, as the flight dynamics of transport aircraft does not allow abrupt changes. Thus, it is possible to capture the characteristics of the trajectories using a set of pre-defined radial basis functions, as done in this paper. Some advantages of this approach are:

- The training data can be treated as continuous functions, avoiding re-sampling and thus also capturing smoothness information.
- The output of the probabilistic model is continuous, allowing the rendering of a smooth volume as a confidence interval.
- The variation of the model with respect to the training data is captured in a precision parameter, preventing over-fitting.
- Co-variances between dimensions of the trajectory are accounted for.



This is a pre-print draft of the paper, which was accepted for publication in the Journal of Guidance, Control, and Dynamics on 17/06/2016

- The model is *generative*, allowing an infinite number of sample trajectories to be produced for a reconstructed model.
- The evaluation of the *marginal* likelihood function allows for quantification of model performance.

A potential limitation, which will be discussed in more detail in the paper is:

- The entirely statistical approach to modelling trajectories can lead to some of the trajectories sampled from the model exhibiting physically implausible behaviours (see results in section 2).

By capturing trajectory data in a probabilistic model, any subsequent calculation can be done on this model, and not the vast amount of measured trajectory data, thus reducing the computational cost. Furthermore, the model can be used as a support tool for Air Traffic Management (ATM): As a method to generate aircraft proximity maps [6], i.e. creating a map that holds the probability an aircraft is present at any given point. As well as a method to construct containment regions to deal with conformance monitoring [13], where it is possible to create a priority listing to observe which aircraft deviate most (based on historical data) from the common flight paths. Finally, the method is capable of assisting with the evaluation of noise abatement [2], where weighted aircraft trajectories can be generated from historical data, as done in [14].

The remainder of the paper is organised as follows. In section II, the mathematics of the method, written for $D$-dimensional trajectories, are explained. Following this, three different datasets of trajectories are comprehensively evaluated using the method in section III. The first two datasets (presented in section A) act as benchmarks, validating the method by applying it on trajectories with known characteristics. The first dataset contains simple two-dimensional 'toy' trajectories that follow a circular path where the radius is sampled from a Gaussian distribution. This dataset sets a baseline performance for the method based on trajectories that truly have a Gaussian distribution. It also helps to illustrate the importance of modelling the co-variances between spatial dimensions. The second dataset contains three-dimensional flight trajectories for sounding rockets that have been generated by a stochastic flight simulator. This dataset is useful for evaluating the method in three dimensions, and to observe the limitations of the statistical approach when it comes to producing physically plausible sample trajectories. Once the method has been validated on the benchmarks, it is applied on the three-dimensional flight trajectories for aircraft departing from Dallas Fort Worth (DFW) airport in section B. These are measured trajectories collected using ground radar. This dataset is useful for evaluating the performance of the method on a 'real' dataset where the distribution over trajectories may not be truly Gaussian. It also allows a demonstration of how the method presented here can be combined with a clustering algorithm to fully model all flight paths around an airport. Finally, in section IV, our conclusions are presented.

## II. Modelling of multi-dimensional trajectories

The task at hand is to analyse a set of trajectories and model the mean trajectory and the dispersion from the mean trajectory. The emphasis here lies on obtaining a method that includes the dependence between each dimension (e.g. latitude, longitude 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.

### A. Data

Given the dataset $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ of $N$ trajectories, each $D$-dimensional trajectory $\mathbf{y}_n$ has $D$ concatenated vectors $\mathbf{x}_d$, resulting in $\mathbf{y}_n = [\mathbf{x}_1 \ldots \mathbf{x}_D]^\intercal$. Furthermore, all $\mathbf{x}_d$ in $\mathbf{y}_n$ have $M_n$ data points, resulting in $\mathbf{x}_d = [x_d(t_1) \ldots x_d(t_{M_n})]^\intercal$. Where the $M_n \times 1$ vector $\mathbf{t}_n = [t_1 \ldots t_{M_n}]^\intercal$ is identical for all $\mathbf{x}_d$. 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 $x_1 \ldots x_D$ 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 $\mathbf{x}_d$, the entire dataset $\mathbf{Y}$ is normalised per individual dimension, resulting in $\mathbf{x}_d = [0, 1]$.

### B. 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



This is a pre-print draft of the paper, which was accepted for publication in the Journal of Guidance, Control, and Dynamics on 17/06/2016

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 presented in [15], where the Dynamic Time Warping (DTW) technique used in the speech recognition field is applied in a pattern detection algorithm. In this approach, the points in each trajectory are shifted (time-warped), such that if one person was walking 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.

### C. Linear representation

We capture each trajectory $\mathbf{x}_d$ as a linear combination of basis functions. Thus for a vector of discrete times $\boldsymbol{\tau}_d$ a single dimension $\mathbf{x}_d$ can be approximated using equation (1).

$$\mathbf{x}_d(\boldsymbol{\tau}_n) \approx \sum_{j=1}^{J} w_j \phi_j(\boldsymbol{\tau}_n) = \boldsymbol{\phi}_d(\boldsymbol{\tau}_n)\mathbf{w}_d \tag{1}$$

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

However, as stated in section A, each trajectory $\mathbf{y}_n$ has $D$ individual dimensions, each described by $\mathbf{x}_d$. Therefore, the trajectory $\mathbf{y}_n$ is approximated using equation (2).

$$\mathbf{y}_n \approx \boldsymbol{\Phi}_n(\boldsymbol{\tau}_n)\mathbf{w}_n \tag{2}$$

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

$$\boldsymbol{\Phi}_n = \begin{pmatrix} \boldsymbol{\phi}_1(\boldsymbol{\tau}_n) & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{\phi}_D(\boldsymbol{\tau}_n) \end{pmatrix} \tag{3}$$

where the dependency of $\boldsymbol{\Phi}_n$ on $\boldsymbol{\tau}_n$ is not explicitly shown to keep the notation uncluttered. In this paper, 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 $d$.

### D. Maximum likelihood formulation

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

$$p(\mathbf{w}_n) = \mathcal{N}(\mathbf{w}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \forall n \in N \tag{4}$$

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

Each trajectory $\mathbf{y}_n$ is modeled as zero-mean Gaussian white noise added to the function described by equation (2), resulting in equation (5).

$$p(\mathbf{y}_n|\mathbf{w}_n, \beta) = \mathcal{N}(\mathbf{y}_n|\boldsymbol{\Phi}_n\mathbf{w}_n, \beta^{-1}\mathbf{I}_n), \forall n \in N \tag{5}$$

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 [16] the marginal distribution of $\mathbf{y}_n$ is given by

$$p(\mathbf{y}_n) = \mathcal{N}(\mathbf{y}_n|\boldsymbol{\Phi}_n\boldsymbol{\mu}, \boldsymbol{\Phi}_n\boldsymbol{\Sigma}\boldsymbol{\Phi}_n^\intercal + \beta^{-1}\mathbf{I}_n), \forall n \in N \tag{6}$$

and the conditional distribution of $\mathbf{w}_n$ given $\mathbf{y}_n$ is equal to

$$p(\mathbf{w}_n|\mathbf{y}_n) = \mathcal{N}(\mathbf{w}_n|\mathbf{S}_n(\beta\boldsymbol{\Phi}_n^\intercal\mathbf{y}_n + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}), \mathbf{S}_n), \forall n \in N \tag{7}$$

where

$$\mathbf{S}_n^{-1} = \boldsymbol{\Sigma}^{-1} + \beta\boldsymbol{\Phi}_n^\intercal\boldsymbol{\Phi}_n \tag{8}$$



This is a pre-print draft of the paper, which was accepted for publication in the Journal of Guidance, Control, and Dynamics on 17/06/2016

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

$$\mathcal{L} = \prod_{n=1}^{N} \{p(\mathbf{y}_n|\mathbf{w}_n, \beta)\ p(\mathbf{w}_n)\} \tag{9}$$

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 (5) and equation (4) in equation (9), that the negative log of $\mathcal{L}$ is given by equation (10).

$$\begin{aligned} -\ln \mathcal{L} = \frac{DM^*}{2}\ln(2\pi) - \frac{DM^*}{2}\ln(\beta) + \\ \frac{\beta}{2}\sum_{n=1}^{N}\{\mathbf{y}_n^\intercal\mathbf{y}_n - 2\mathbf{y}_n^\intercal(\mathbf{\Phi}_n\mathbf{w}_n) + \mathrm{Tr}(\mathbf{\Phi}_n^\intercal\mathbf{\Phi}_n\mathbf{w}_n\mathbf{w}_n^\intercal)\} \\ + \frac{NJD}{2}\ln(2\pi) + \frac{N}{2}\ln(|\mathbf{\Sigma}|) + \\ \frac{1}{2}\sum_{n=1}^{N}\{\mathrm{Tr}(\mathbf{\Sigma}^{-1}(\mathbf{w}_n\mathbf{w}_n^\intercal - 2\mathbf{w}_n^\intercal\boldsymbol{\mu} + \boldsymbol{\mu}\boldsymbol{\mu}^\intercal))\} \end{aligned} \tag{10}$$

where

$$M^* = \sum_{n=1}^{N}\{M_n\} \tag{11}$$

Minimization with respect to $\boldsymbol{\mu}$, $\mathbf{\Sigma}$ and $\beta$ can be done numerically using the Expectation-Maximization (EM) algorithm. The expected log-marginal likelihood is given by equation (10) where the terms $\mathbf{w}_n$ and $\mathbf{w}_n\mathbf{w}_n^\intercal$ take their expected values.

From equation (7) the expected values are given by

$$\mathbb{E}[\mathbf{w}_n] = \mathbf{S}_n(\beta\mathbf{\Phi}^\intercal\mathbf{y}_n + \mathbf{\Sigma}^{-1}\boldsymbol{\mu}) \tag{12}$$

$$\mathbb{E}[\mathbf{w}_n\mathbf{w}_n^\intercal] = \mathbf{S}_n + \mathbb{E}[\mathbf{w}_n]\mathbb{E}[\mathbf{w}_n^\intercal] \tag{13}$$

Minimizing the expected negative log likelihood with respect to $\boldsymbol{\mu}$, $\mathbf{\Sigma}$ and $\beta$ respectively give the expressions for $\hat{\boldsymbol{\mu}}$, $\hat{\mathbf{\Sigma}}$ and $\hat{\beta}$ as seen in equations (14) to (16).

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{n=1}^{N}\{\mathbb{E}[\mathbf{w}_n]\} \tag{14}$$

$$\hat{\mathbf{\Sigma}} = \frac{1}{N}\sum_{n=1}^{N}\{\mathbb{E}[\mathbf{w}_n\mathbf{w}_n^\intercal] - 2\mathbb{E}[\mathbf{w}_n^\intercal]\boldsymbol{\mu} + \boldsymbol{\mu}\boldsymbol{\mu}^\intercal\} \tag{15}$$

$$\frac{1}{\hat{\beta}} = \frac{1}{DM^*}\sum_{n=1}^{N}\{\mathbf{y}_n^\intercal\mathbf{y}_n - 2\mathbf{y}_n^\intercal(\mathbf{\Phi}\mathbb{E}[\mathbf{w}_n]) + \mathrm{Tr}(\mathbf{\Phi}^\intercal\mathbf{\Phi}\mathbb{E}[\mathbf{w}_n\mathbf{w}_n^\intercal])\} \tag{16}$$

Equations (12) to (16) can be solved iteratively for $\hat{\boldsymbol{\mu}}$, $\hat{\mathbf{\Sigma}}$ and $\hat{\beta}$.

Once the likelihood given by equation (10) has converged (in this paper a convergence limit of $10^{-1}$ is used), it is possible to find the probability distribution of $\mathbf{y}$ at any given $\tau$ using equation (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 $\mathbf{k}(\tau, \tau')$:

$$\mathbf{y}(\tau) \sim \mathcal{GP}(\mathbf{m}(\tau), \mathbf{k}(\tau, \tau')) \tag{17}$$

where

$$\mathbf{m}(\tau) = \mathbb{E}[\mathbf{y}] = \mathbf{\Phi}(\tau)\boldsymbol{\mu} \tag{18}$$

$$\mathbf{k}(\tau, \tau') = \mathbb{E}[(\mathbf{y}(\tau) - \mathbf{m}(\tau))(\mathbf{y}(\tau') - \mathbf{m}(\tau'))] = \mathbf{\Phi}(\tau)\mathbf{\Sigma}\mathbf{\Phi}^{\intercal}(\tau') + \beta^{-1}\mathbf{I} \tag{19}$$

A more intuitive representation of the mean function $\mathbf{m}(\tau)$, given by equation (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 (19), can be represented as a $D$-dimensional (hyper-)volume at a constant probability.

When sampling from equation (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 (20).

$$p(\mathbf{y}_n) = \mathcal{N}(\mathbf{y}_n|\mathbf{\Phi}_n\boldsymbol{\mu}, \mathbf{\Phi}_n\mathbf{\Sigma}\mathbf{\Phi}_n^{\intercal}), \forall n \in N \tag{20}$$

### E. Model selection

The likelihood function $\mathcal{L}$ being maximized is also known as the marginal likelihood, as the model parameters $\boldsymbol{\mu}$, $\mathbf{\Sigma}$ 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 [9].

Hence by evaluating the likelihood at different values of the model complexity $J$, the optimal complexity can be calculated. Note that in this paper only the uniform distributed Gaussian basis functions (as seen in appendix A) are considered, however it is possible to compare any model structure (e.g. linear, exponential, polynomial, or even Bézier curves [17]).

Furthermore, for the uniform distributed Gaussian basis functions the model complexity $J$ has an upper limit for which the likelihood function $\mathcal{L}$ (equation (10)) can be evaluated, this is a direct result from the term $\ln(|\mathbf{\Sigma}|)$. As one of the eigenvalues of $\mathbf{\Sigma}$ turns zero, the log-term goes to infinity, indicating redundancy in the model. In other words, there is a value in $\mathbf{w}_n$ 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 $\mathbf{\Sigma}$. For the model selection in section III, 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.

### F. 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, $p(\mathbf{y})$ 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 paper 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 $p(\mathbf{y})$ 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 uni-variate 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, [18] showed that this function can be approximated using a

large number of generated samples. Hence, 2000 sample trajectories are drawn from $p(\mathbf{y}_n)$ (seen in equation (17)) over $\tau = [0, 1]$ in 100 uniform steps to represent the CDF. These values for sampling $p(\mathbf{y})$ and integrating over $\tau$ have been selected to assure 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.

Additionally, to reduce the computational cost of this test, the method presented in [19] 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 (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's 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.

# III. Results

In this section, the previously described method is applied to three different datasets. In section A, two datasets are analysed to benchmark the performance of the method. First, *toy* trajectories are analysed with the aim to determine the baseline performance. This also visualises the importance of modelling dependence between dimensions and the time-warping of the trajectories. Secondly, the analysis is extended to three dimensions using a stochastic rocket simulator. This analysis validates the approach to trajectories in a three-dimensional space, and describes the limitations of using a probability function to describe a physical process. Once the methodology has been validated on the benchmarks, the technique is applied to a dataset containing aircraft trajectories as measured by the ground radar in section B. It also combines the method presented in this paper with the clustering algorithm as described in appendix D to obtain an overview of all trends present in the data.

## A. Benchmark experiments

### 1. *Dataset I – Gaussian distributed trajectories*

This dataset contains two-dimensional 'toy' trajectories that follow a circular path where the radius is sampled from a Gaussian distribution, all according to the equations presented in appendix B. 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.

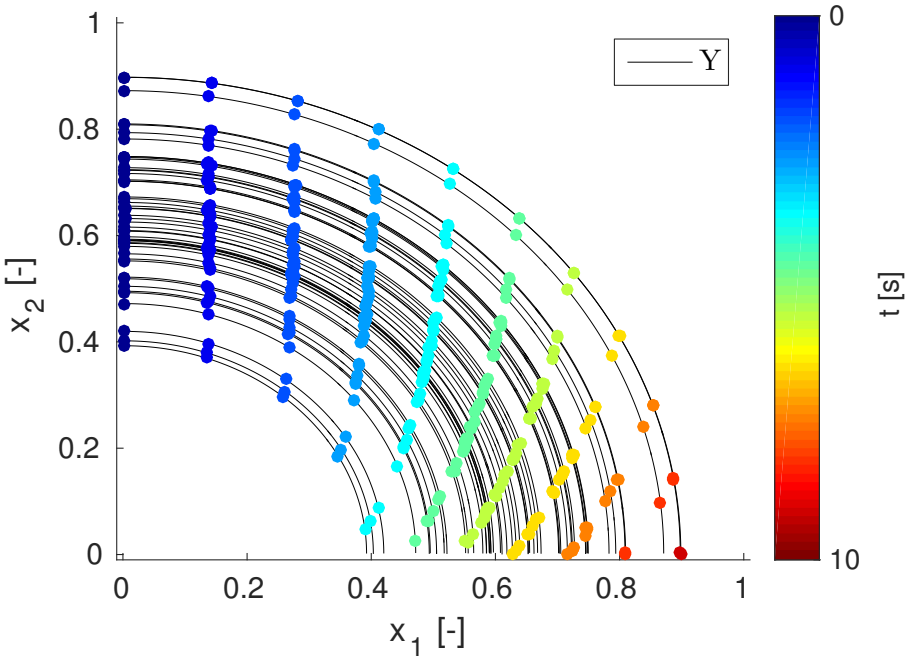
As for the time-warping, these results are visible in figure 1. On the left, 50 trajectories at a constant velocity are visible, where the dots indicate the points between the start $t = 0s$ and the end time of the longest trajectory $t = 10s$ in 10 steps. On the right, all trajectories have been normalised 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 through 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 E. 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 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 $p(\mathbf{y})$, 50 sample trajectories are generated using equation (20). These sampled trajectories $\mathbf{S}$ are shown in figure 2b.

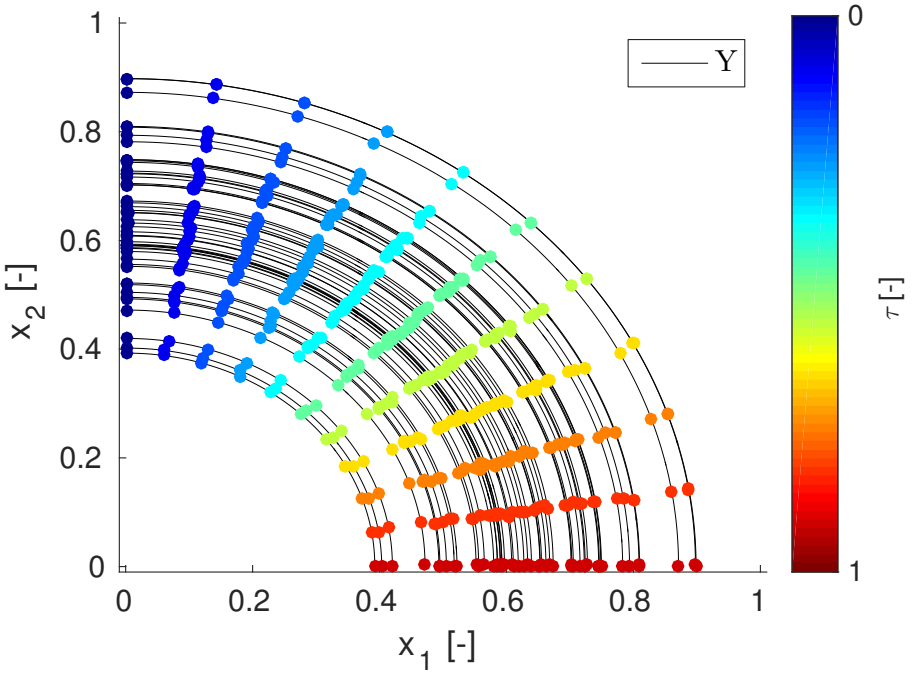
One of the key contributions of this paper is that it models the dependence between each dimensions by including the co-variance, and how it allows the probabilistic model to generate trajectory data (sampled data) similar to the input trajectory data (training data). An additional claim is that it also provides an improvement when it comes to modelling



This is a pre-print draft of the paper, which was accepted for publication in the Journal of Guidance, Control, and Dynamics on 17/06/2016

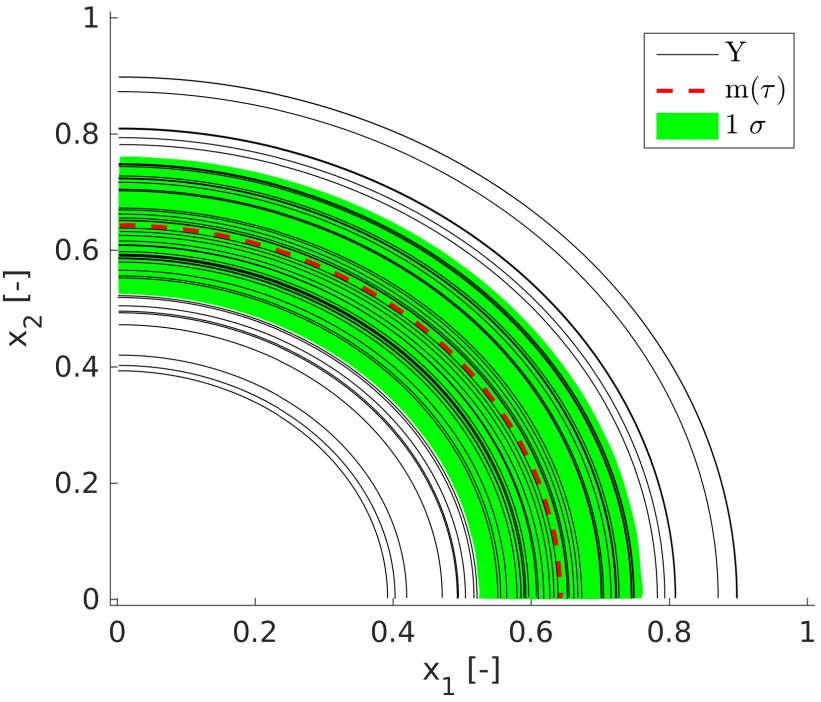



**a) un-normalised time,** $t = \{0s, 1s, ...10s\}$

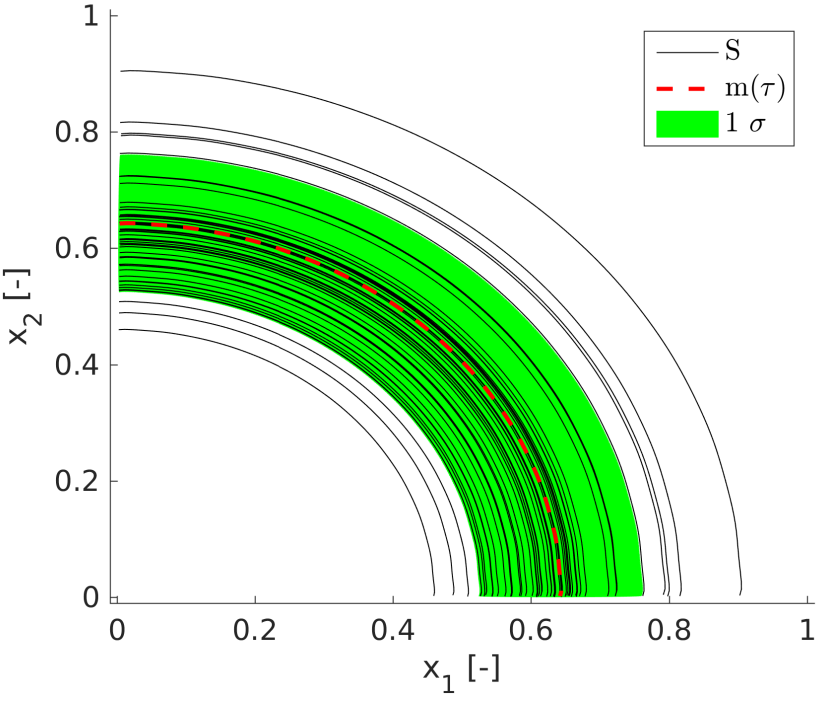



**b) normalised time,** $\tau = \{0, 0.1, .., 1\}$

**Figure 1. Examining the effect of normalising time**




**a) training data Y**




**b) sampled data S**

**Figure 2. Visualisation of Gaussian distributed training and sampled trajectory data.**

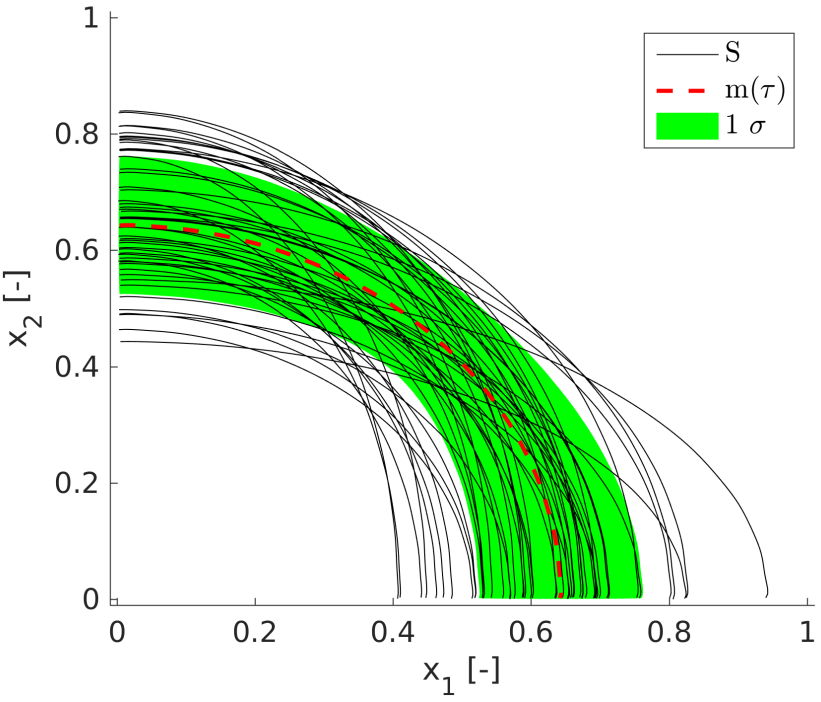



**Figure 3. Visualisation of Gaussian distributed sampled trajectory data from an independent model.**

the dispersion. To simulate what will happen when the covariance between dimensions is not taken into account, only the off-diagonal blocks $\mathbf{\Sigma}_{1,2}$ and $\mathbf{\Sigma}_{2,1}$ in the co-variance matrix as seen in equation (21) need to be removed.

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{1,1} & \mathbf{\Sigma}_{1,2} \\ \mathbf{\Sigma}_{2,1} & \mathbf{\Sigma}_{2,2} \end{pmatrix} \tag{21}$$

The sampled trajectory data, generated from $p(\mathbf{y})$ using this reduced covariance matrix is visible in figure 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 F. For this experiment, the ECDF is created using 2000 trajectories from the *evaluation* dataset. Figure 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%. And while the *training* and *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, the point being made here, is that it allows for the approximation of a probabilistic model using (limited) available data.

### 2. *Dataset II – Sounding rocket trajectories*

The second benchmark experiment analyses sounding rocket flight trajectories. In order to have access to a large number of rocket flight trajectories, the stochastic rocket flight simulator based on [4] was used. The data are generated from 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

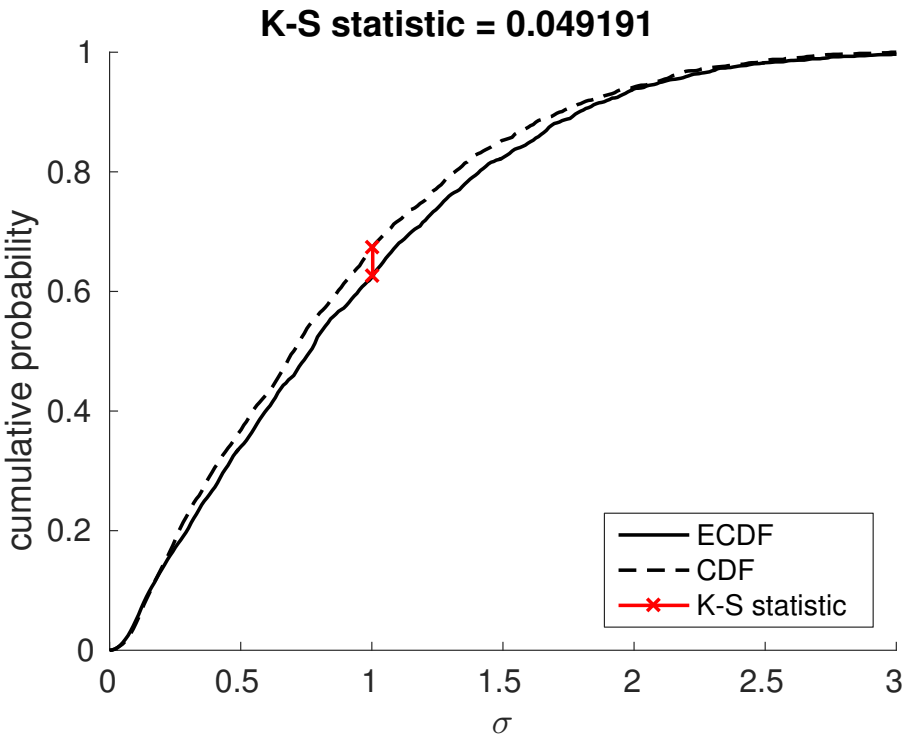



**Figure 4. Comparing the Gaussian distributed evaluation and sampled trajectory data.**

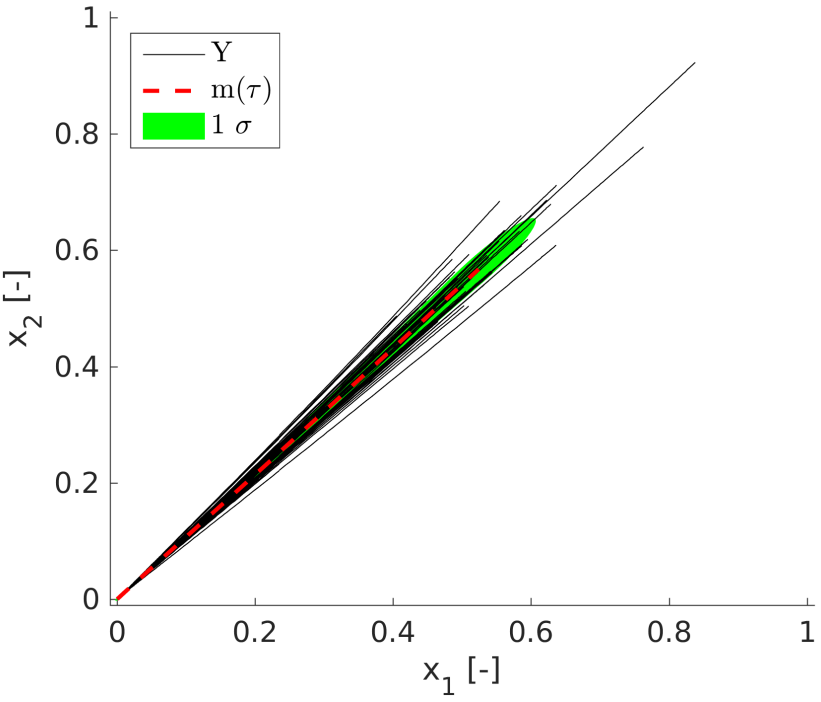
this simulator, which is designed for sounding rockets, models a parachute deployment at apogee and descent under parachute. However, for simplicity the trajectories generated for this work are all 'ballistic', that is without parachute deployment. This, combined with atmospheric data simulating a wind from the west, results in the trajectories as seen in this section. The specific settings of the simulator to generate the trajectories seen in this section, can be reviewed in appendix C.

The model selection is done based on the likelihood as described in section E. The maximum value for the marginal likelihood occurs at the model complexity $J = 18$, and is therefore selected to be the best model.

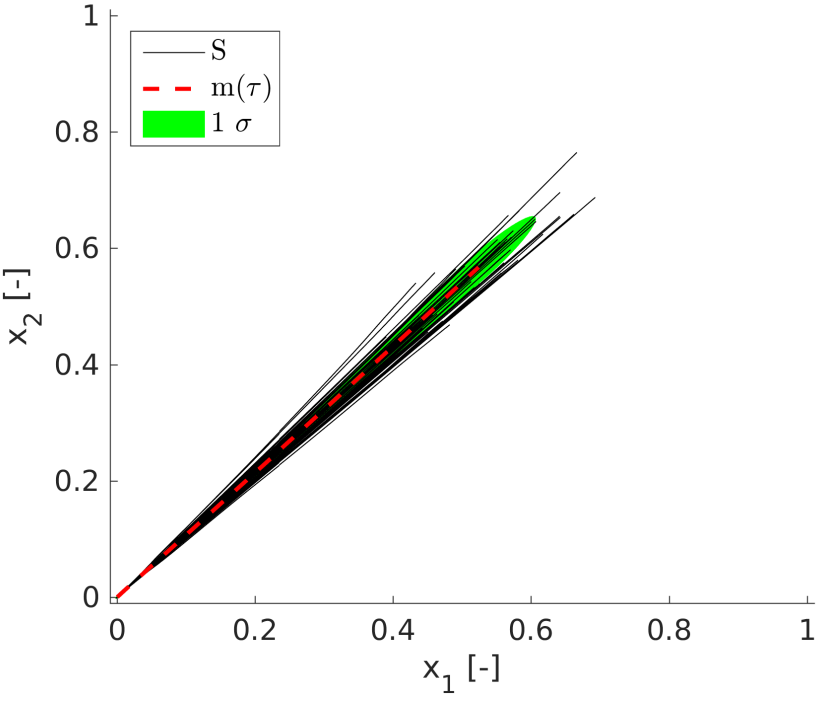
The 50 trajectories used as training data are seen in figures 5a and 6a. It shows the trajectories seen from two directions, the top-view and a side-view. Furthermore, the mean trajectory ($\mathbf{m}(\tau)$) is visualised as a dashed line and the region captured by one standard deviation (1 $\sigma$) is shown by a shaded area. From the probabilistic model $p(\mathbf{y})$, 50 sample trajectories (described as $\mathbf{S}$) are generated using equation (20). These sampled trajectories $\mathbf{S}$ are shown in figures 5b and 6b. 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 7 to 9 respectively.

The sampled trajectory data $\mathbf{S}$ 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 expected behaviour. An example of this is visible when looking at the time-series in figure 10, 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.

Similar to the previous experiment, the performance of the model is evaluated according to the technique discussed in section F. For this experiment, the ECDF is created using 2000 trajectories from the *evaluation* dataset. Figure 11 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.

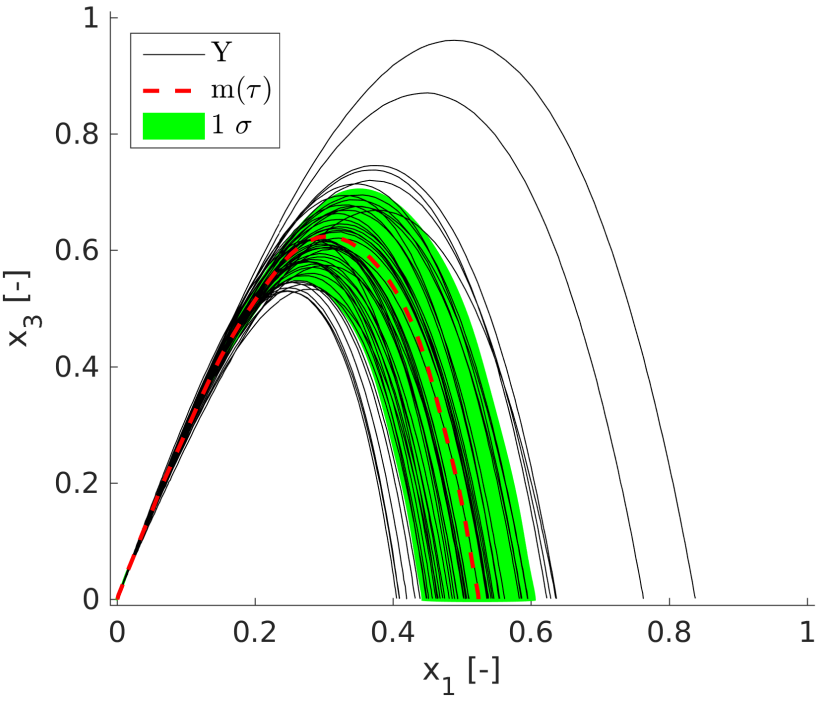



**a) rocket trajectory training data Y**

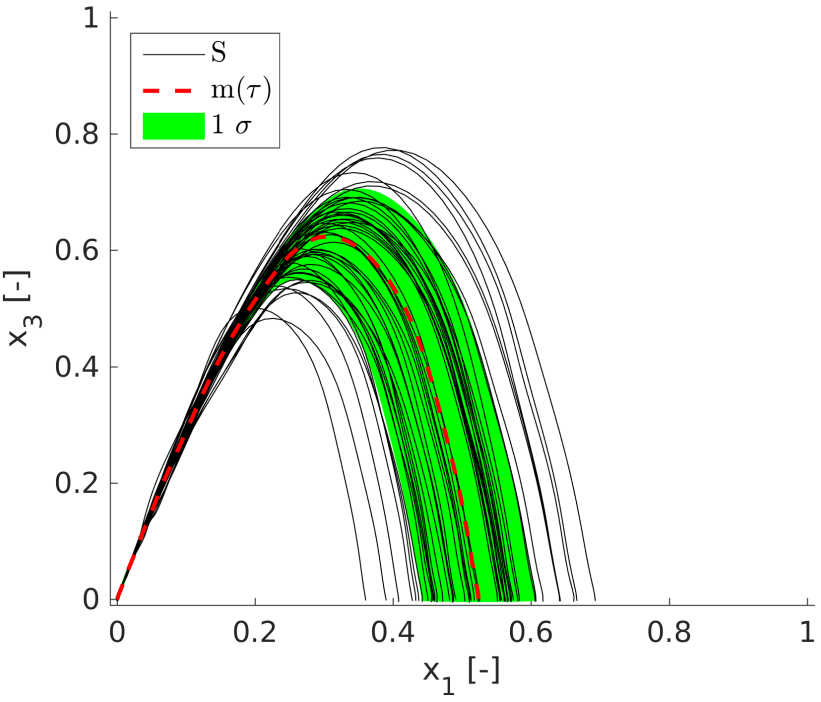



**b) rocket trajectory sampled data S**

**Figure 5. Visualisation (top-view) of sounding rocket training and sampled trajectory data.**




**a) rocket trajectory training data Y**




**b) rocket trajectory sampled data S**

**Figure 6. Visualisation (side-view) of sounding rocket training and sampled trajectory data.**

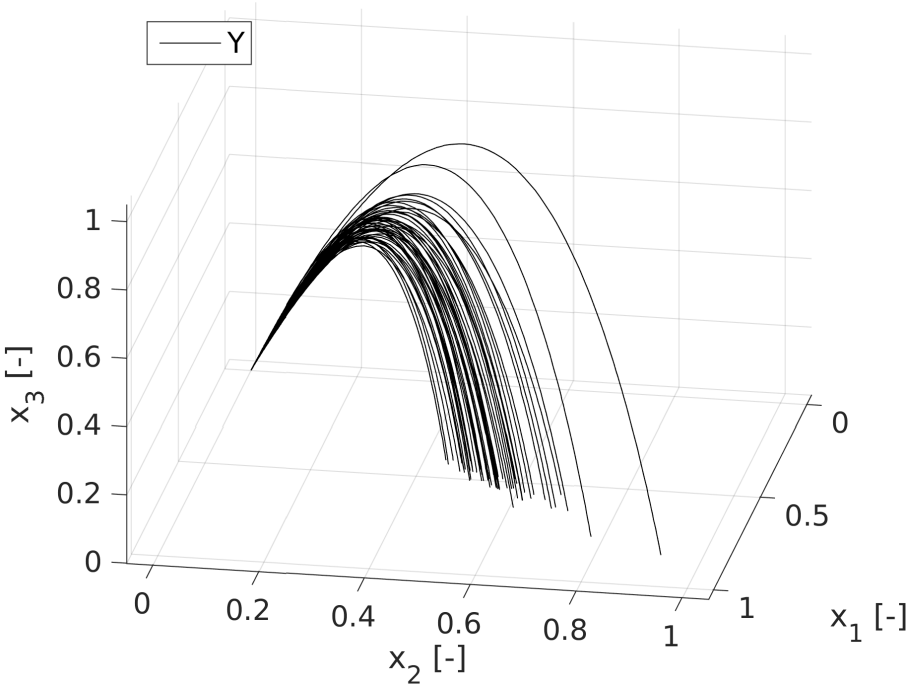



**Figure 7. A three-dimensional representation of the sounding rocket training trajectory data Y.**

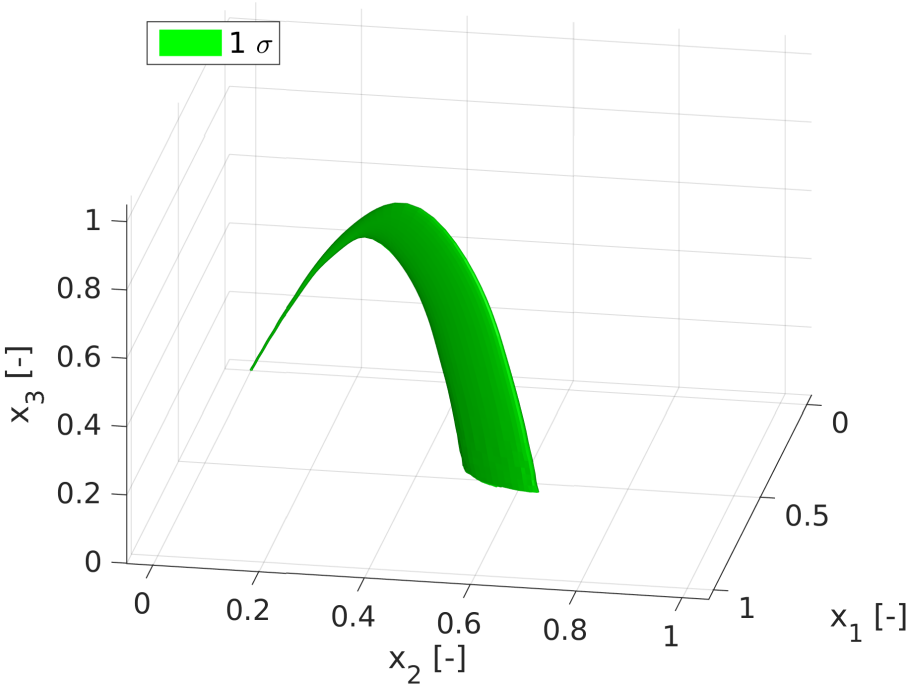



**Figure 8. A three-dimensional representation of the volume containing one standard deviation for the sounding rocket trajectories.**

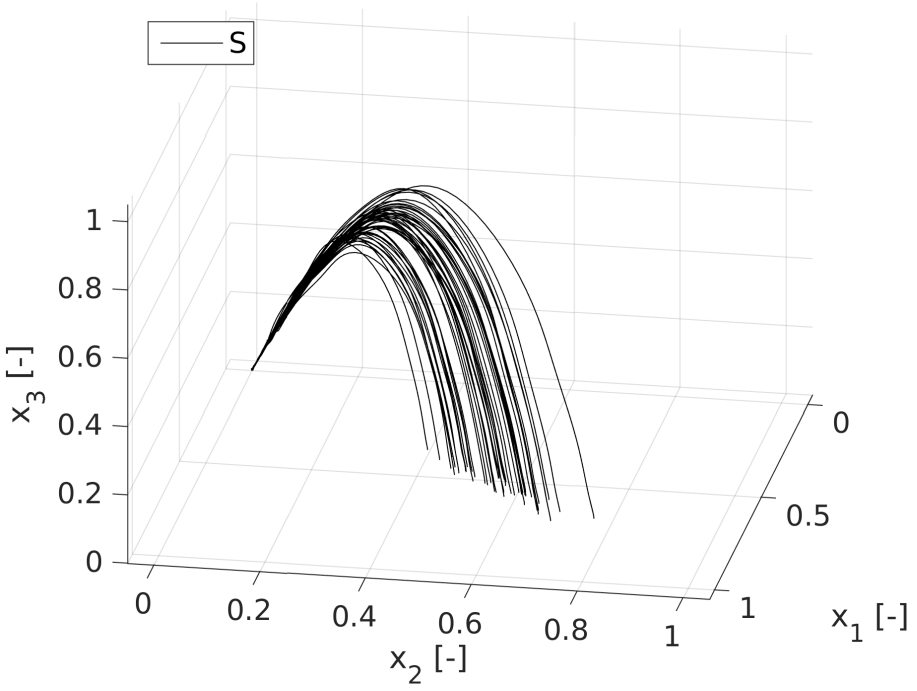


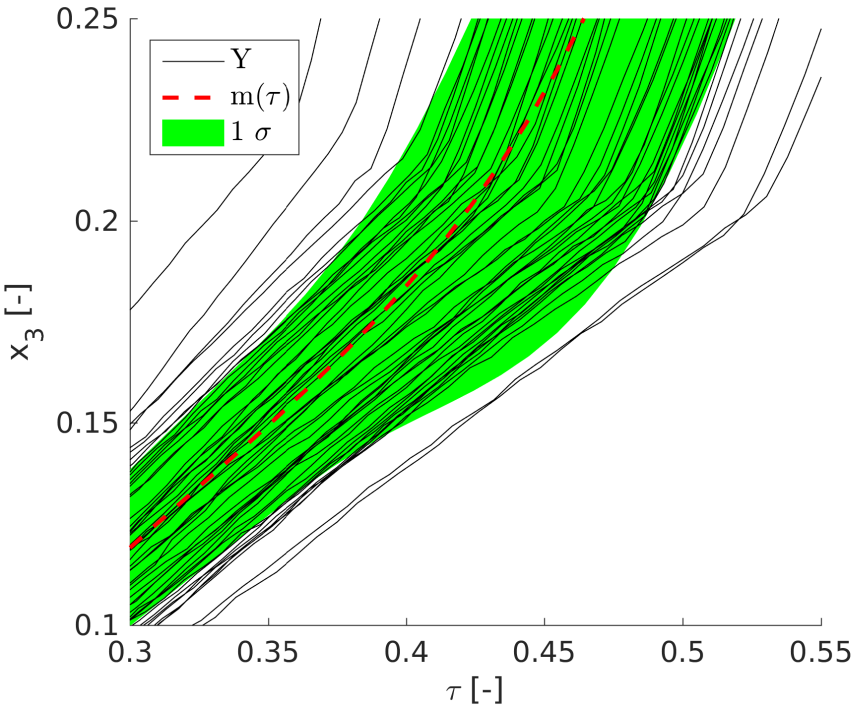

**Figure 9. A three-dimensional representation of the sounding rocket sampled trajectory data S.**

## B. Modelling aircraft trajectories using Gaussian processes

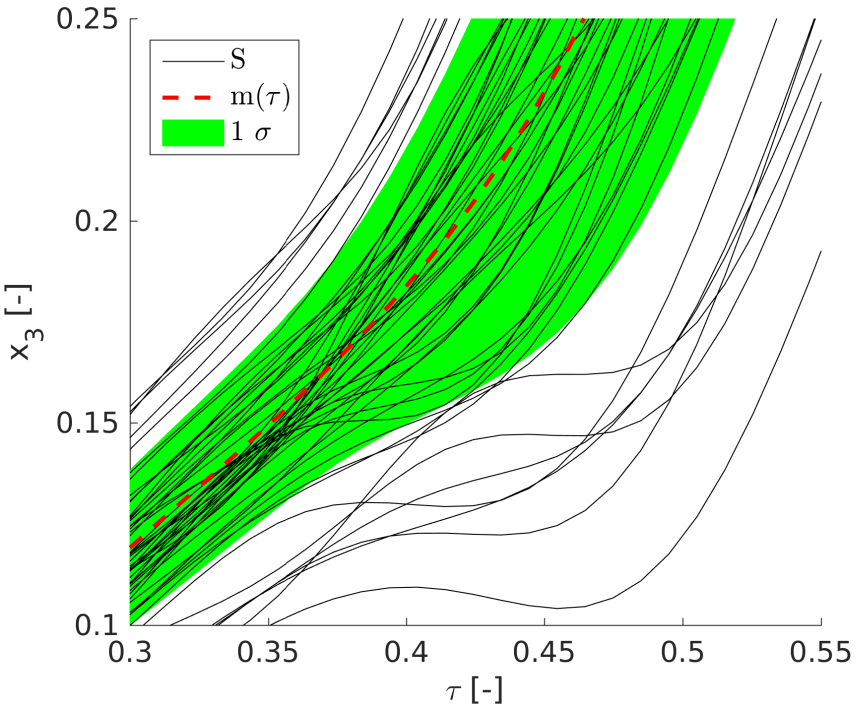
This dataset contains three-dimensional flight trajectories for aircraft taking off at DFW. These are measured trajectories collected using ground-based radar. The aim here is evaluating the performance of the method on a real dataset where the distribution over trajectories may not be truly Gaussian. Furthermore, it includes a demonstration of how the method presented in this paper can be combined with a clustering algorithm for aircraft [1] to provide an overview of all flight paths around an airport. The clustering step is essential as the method presented in section II is only valid for sets of trajectories that follow a common flightpath. The clustering algorithm used here relies on re-sampling the trajectories, reducing the dimension via Principle Component Analysis (PCA), and finally cluster via the Density-Based Spatial Clustering of Applications (DBSCAN) technique. A detailed description of the entire algorithm is included in appendix D. 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 of 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. And as a result, the maximum deviation between the probabilistic model and the test data, the K-S statistic, is expected to increase.

From the clustered data, the largest cluster of 82 trajectories is analysed in this section. 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 II. The model selection is done based on the likelihood as indicated in section E. 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.

The 50 aircraft trajectories in the training data $\mathbf{Y}$ are visible in figures 12a and 13a 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 (20). These sampled trajectories $\mathbf{S}$ are shown in figures 12b and 13b. While the side-view shows similar trajectories, the top-view shows a greater symmetry in the sampled data, an identical effect was seen when modelling the sounding rocket trajectories (section 2). 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. Again, this is a direct consequence




**a) rocket trajectory training data Y**




**b) rocket trajectory sampled data S**

**Figure 10. Altitude time-series of the sounding rocket trajectories at $\tau = [0.3, 0.55]$, illustrating the difference between the physical and probabilistic model.**

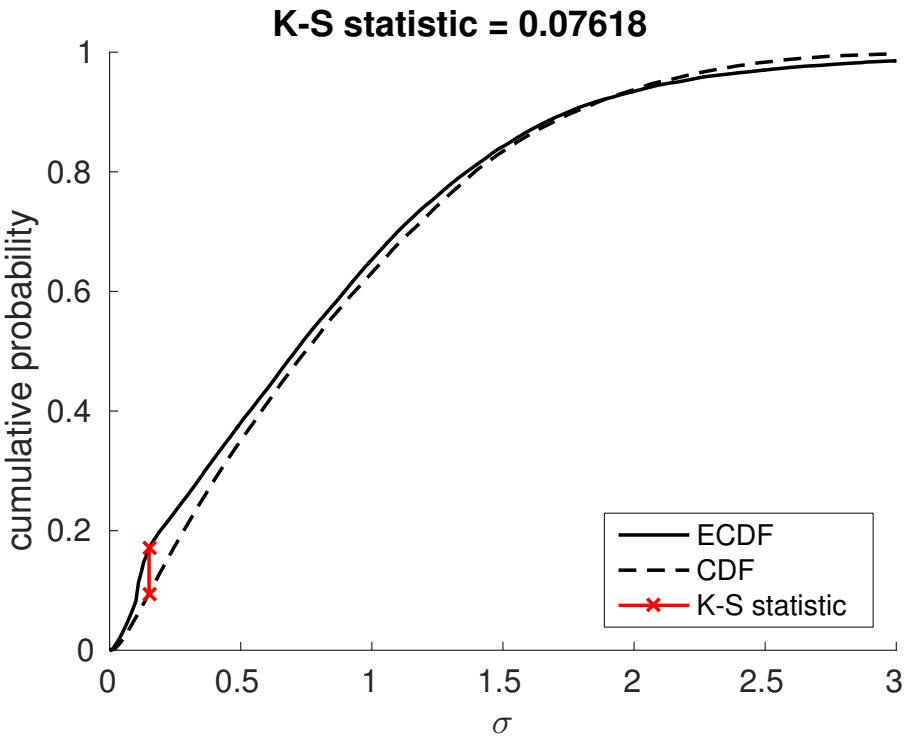



**Figure 11. Comparing the sounding rocket evaluation and sampled trajectory data.**

of using a Gaussian probability distribution, which is assuming symmetry around the average. Additionally, a three-dimensional representation of the training data **Y**, sampled data **S** and the volume containing one standard deviation is presented in figures 14 to 16 respectively.

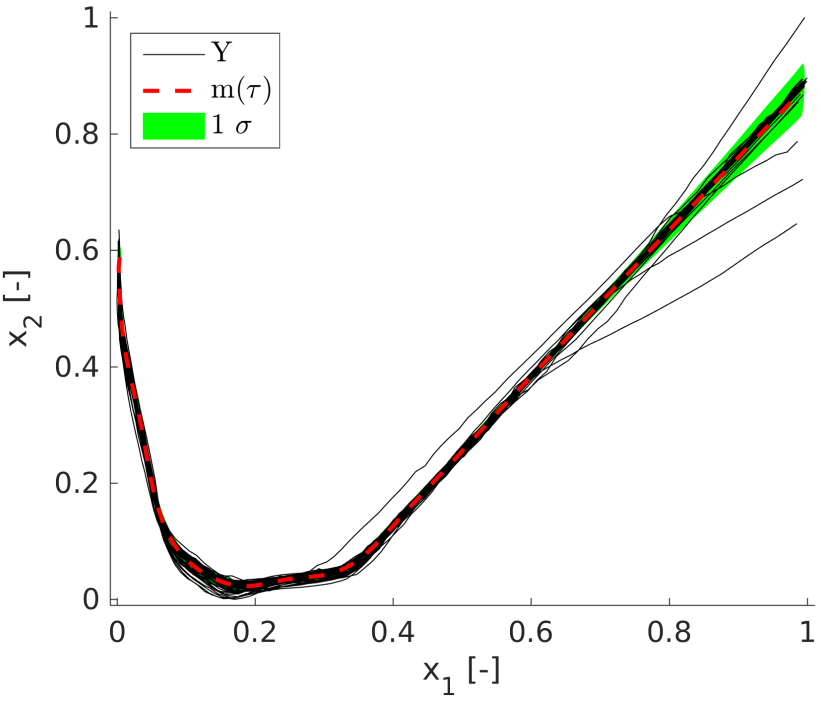
Similar to the previous two experiments, the performance of the model is evaluated according to the technique discussed in section F. For this experiment, the ECDF is created using the 32 trajectories from the *evaluation* dataset. Figure 17 displays both the ECDF (evaluation trajectory data) and the CDF (sampled trajectory data), showing that the K-S statistic is 0.131, corresponding to a maximum deviation of 13.1%. This reduction in performance is attributed to the differences seen in the training data and sampled data and the limited number of trajectories in the evaluation data.

When performing a similar procedure on all clusters seen in figure 18a, a spatial analysis of the complete dataset, provides the overview as seen in figure 18b. 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 19a, with the corresponding overview in figure 19b. 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.

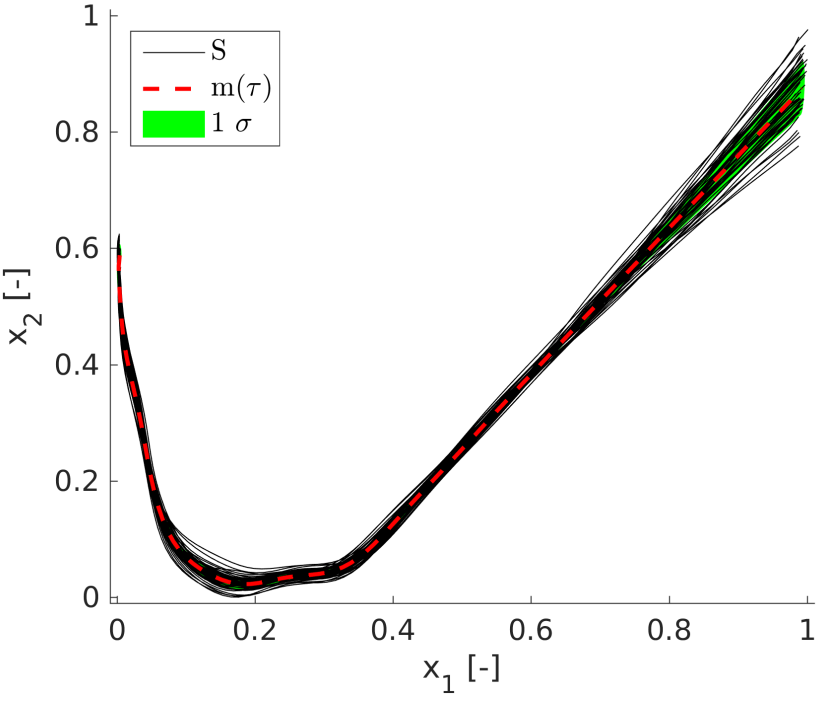
The overview as seen in figure 18b is a snapshot of the specific conditions captured in the trajectory data. For an airport, the conditions would include the time of day (e.g. alteration of runway use due to noise regulations), date (e.g. holiday season), and weather conditions. In this particular dataset, 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.

## IV. Conclusion

This paper explores the application of a Gaussian processes approach to the analysis of aircraft trajectories. The method works in a continuous framework and has been evaluated in a range of tests, giving an insight into the maximum difference between the trajectory data generated by the probabilistic model and the measured trajectory data.

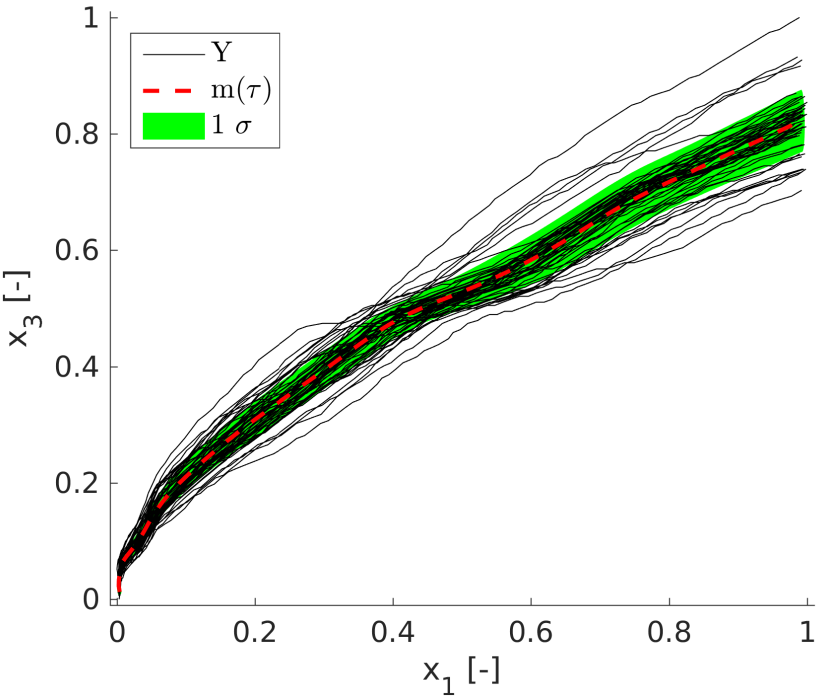



**a) aircraft trajectory training data Y**

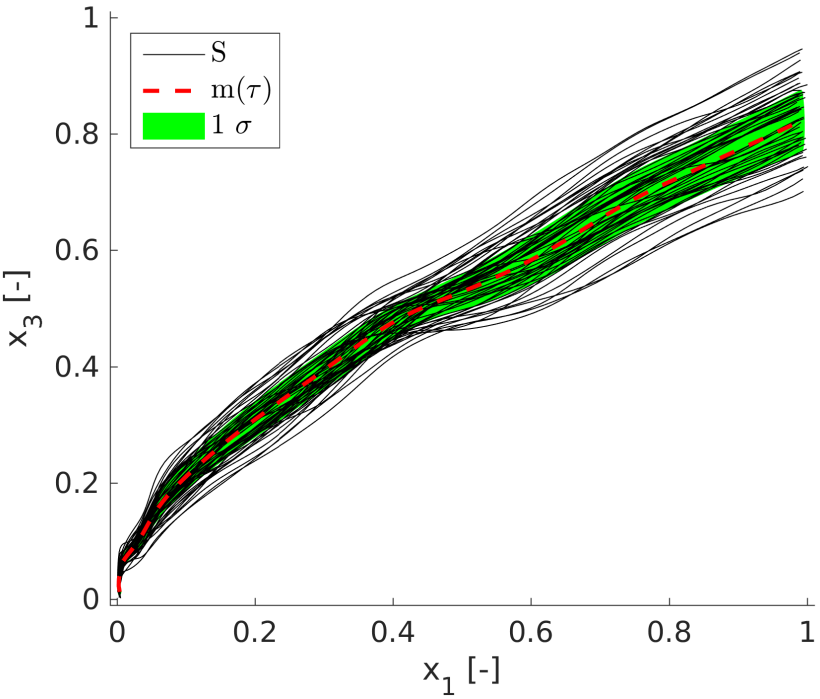



**b) aircraft trajectory sampled data S**

**Figure 12. Visualisation (top-view) of aircraft training and sampled trajectory data.**




**a) aircraft trajectory training data Y**




**b) aircraft trajectory sampled data S**

**Figure 13. Visualisation (side-view) of aircraft training and sampled trajectory data.**

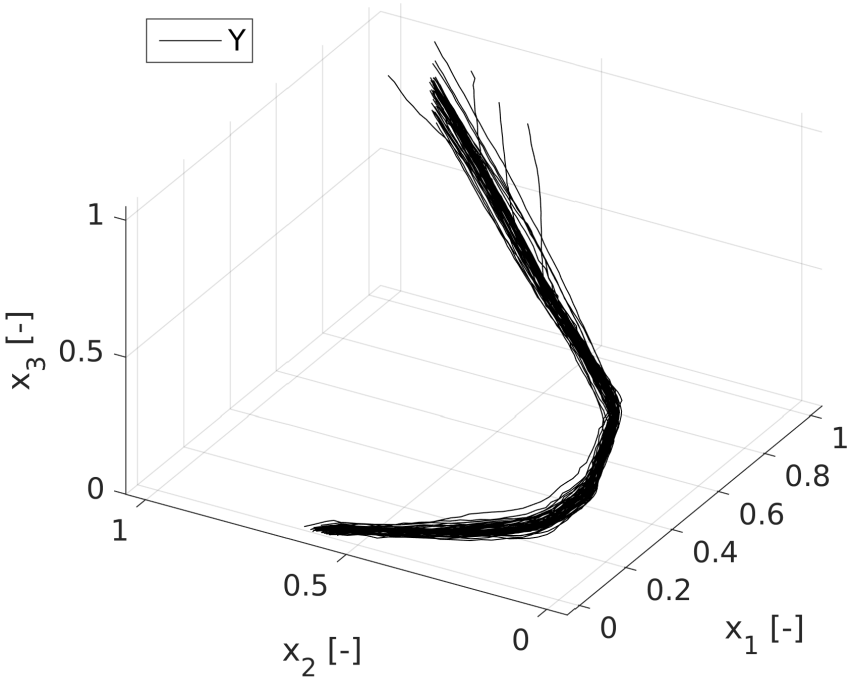



**Figure 14. A three-dimensional representation of the aircraft training trajectory data Y.**

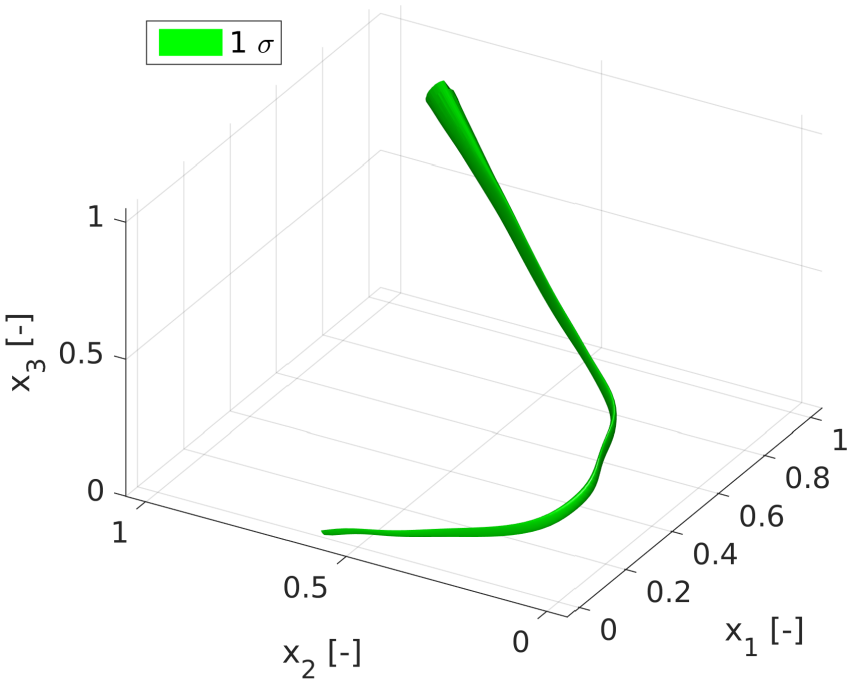



**Figure 15. A three-dimensional representation of the volume containing one standard deviation for the aircraft trajectories.**

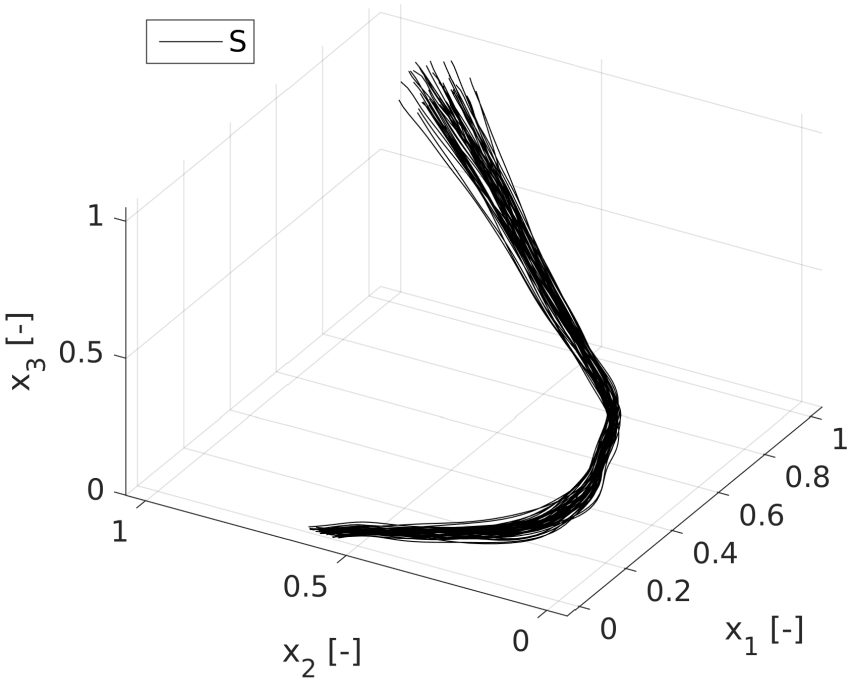



**Figure 16. A three-dimensional representation of the aircraft sampled trajectory data S.**

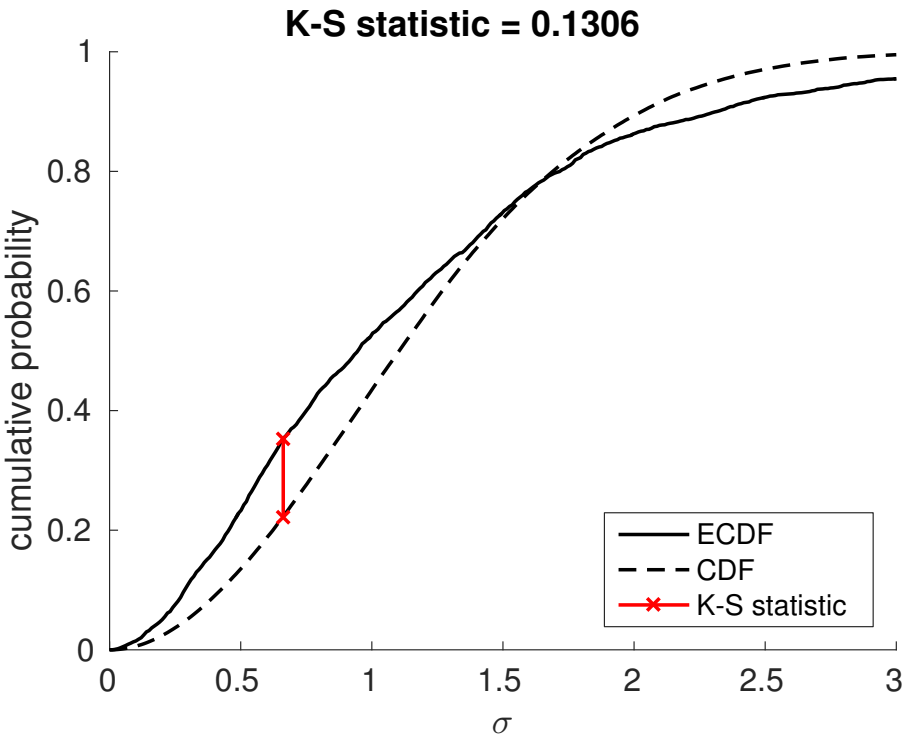
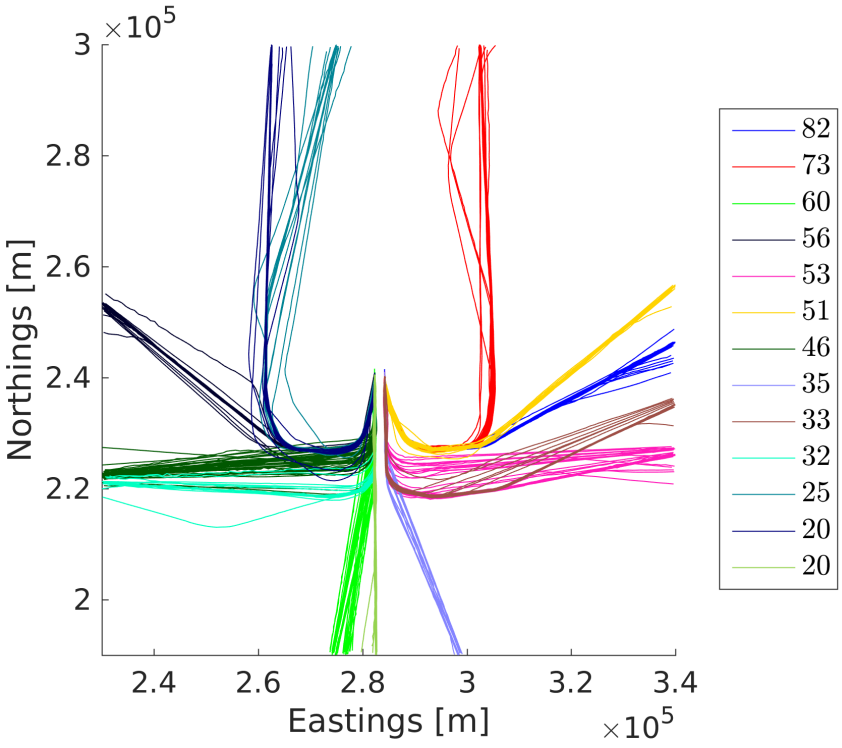
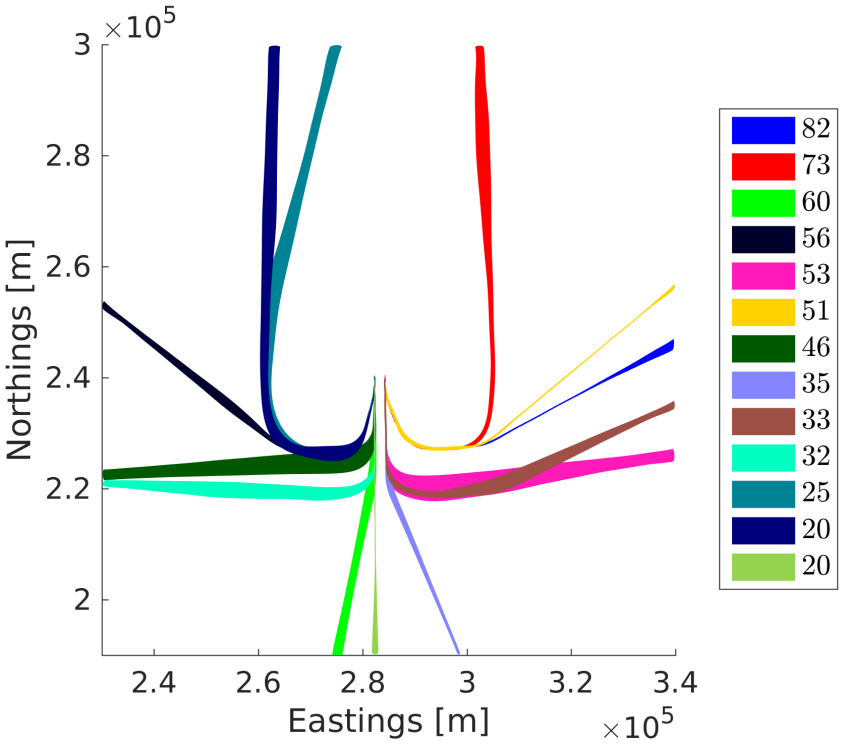



**Figure 17. Comparing the aircraft evaluation and sampled trajectory data.**

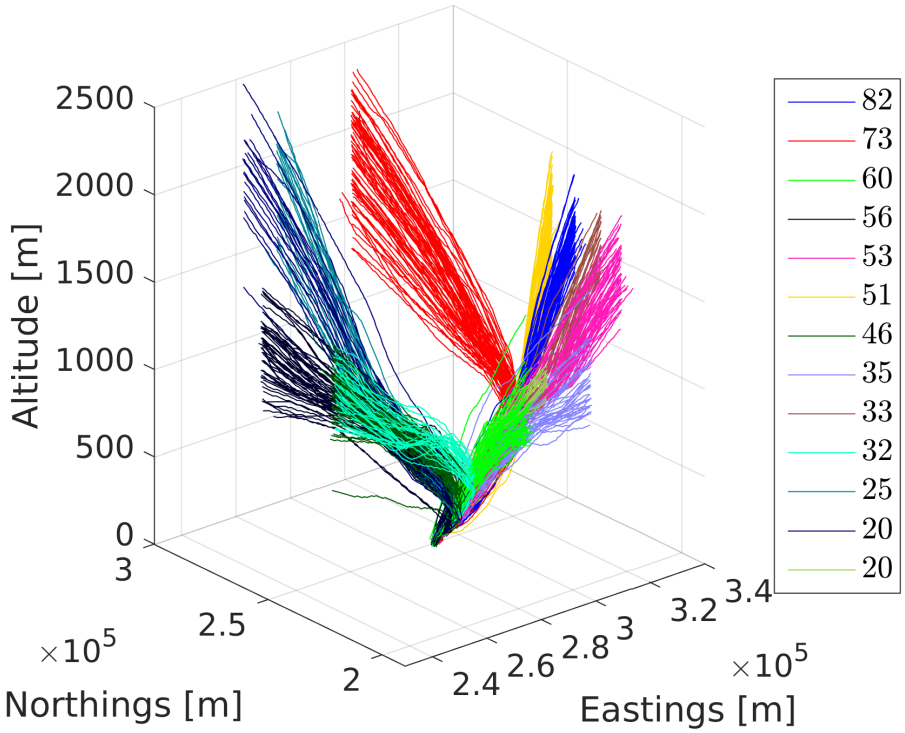



**a) the aircraft trajectory data Y.**

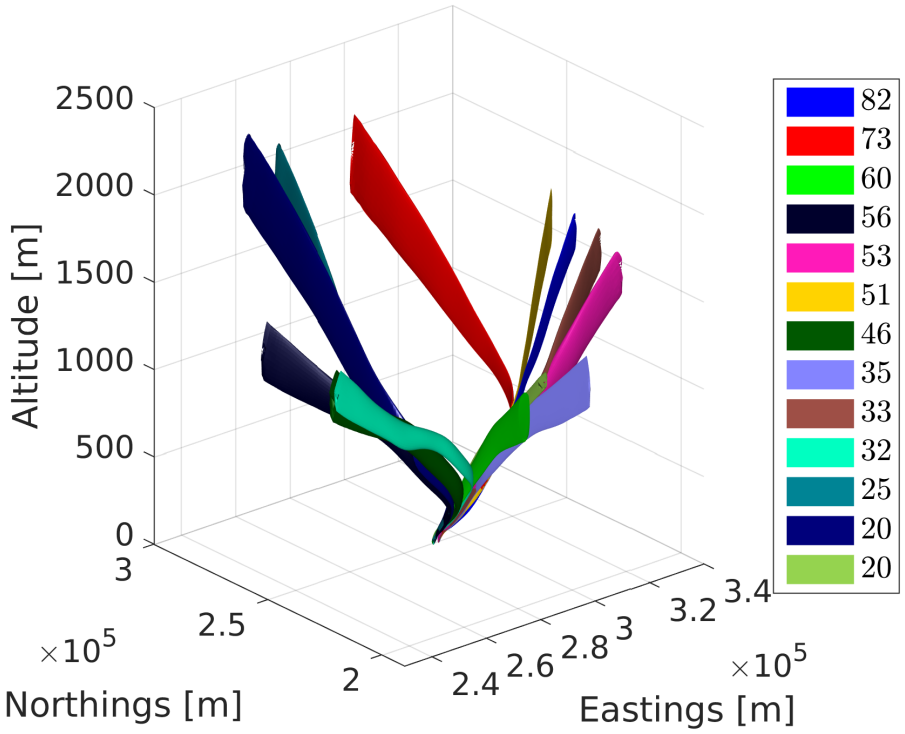



**b) the area covering one standard deviation for the aircraft trajectories.**

**Figure 18. 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 19. A three-dimensional view of all departing aircraft trajectories from two runways at DFW.**

A benchmark test – using a dataset that is known to have a Gaussian distribution – showed a maximum deviation between the model and test data of 4.9%. Further analysis indicated that including the co-variance between dimensions is important when trying to capture the longitudinal correlations between trajectories.

Another benchmark test on sounding rocket trajectories showed a maximum deviation between the model and test data of 7.6%. Further analysis also demonstrated the presence of physically implausible behaviours in sample trajectories generated over the model. It follows that caution should be exercised when using the generative capabilities of this type of probabilistic model in applications where the physical dynamics of trajectories are important.

The analysis of a set of commercial airline trajectories recorded for departures from Dallas Fort Worth (DFW) airport showed a maximum deviation of 13.1%. Further analysis showed that – by combining the proposed modelling approach with a clustering algorithms – large sets of trajectories covering multiple discrete flightpaths 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). Ultimately, this method captures trajectory data in a probabilistic model, allowing subsequent calculations to be done a model and not the vast amount of measured trajectory data, thus reducing the computational cost.


## Acknowledgements

The authors gratefully acknowledge the funding provided under research grant EP/L505067/1 from the Engineering and Physical Sciences Research Council, and the industry sponsor Cunning Running Software Ltd.


## A. Equal spaced Gaussian shaped basis functions

In some cases there is 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 Gaussian shaped basis functions. Note that this Gaussian shape is not related to a probability function, but is only this particular shape is used in the function, here it is also known as a *radial basis function* and takes the shape of:

$$\phi_k(\tau) = \exp\left(\frac{-(\tau - c_k)^2}{r_k^2}\right) \tag{22}$$

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 [20] to be:

$$r_1 = r_2 = \ldots = r_k = \frac{d_{max}}{\sqrt[n]{nK}} = \frac{d_{max}}{K} \tag{23}$$

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.

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 20. 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.

## B. Generating toy trajectories

This section summarises the equations that governs the data as seen in section 1. The radius $r$ is selected using:

$$r = \mathcal{N}(r|1, \frac{1}{2}) \tag{24}$$

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

$$\boldsymbol{\theta} = \{0, \frac{\pi}{2N}, \ldots, \frac{\pi}{2}\} \tag{25}$$

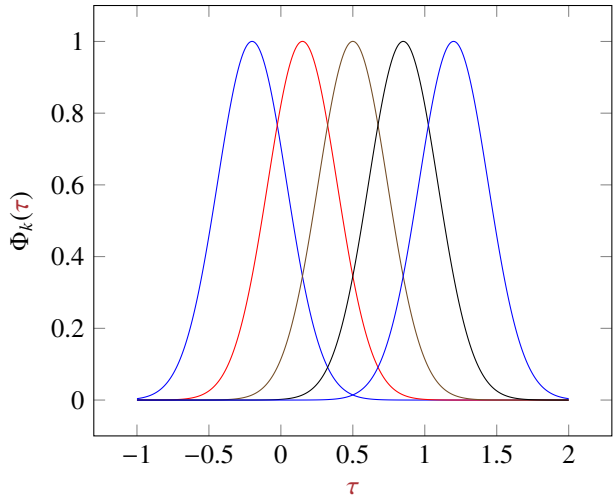



**Figure 20. Example of $K = 5$ over $\tau = [0, 1]$, including the extra width**

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

$$\mathbf{x} = r\cos(\boldsymbol{\theta}) \tag{26}$$

$$\mathbf{y} = r\sin(\boldsymbol{\theta}) \tag{27}$$

## C. Simulator settings

This section summarises the settings used to generate the data as seen in section 2. These settings are the input for the stochastic rocket simulator as seen in [4]. The MATLAB version of the Cambridge Rocketry Simulator is available for download at http://cambridgerocket.sourceforge.net/.

```
% 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");
```

## D. Clustering aircraft trajectories

The clustering technique implemented uses the approach and departure labelling, runway location and the clustering technique for aircraft trajectories as seen in [1].

1. *approach* or *departure*.

2. assign nearest runway based on start location.

3. re-sampling, Principle Component Analysis (PCA) (dimension reduction) and Density-Based Spatial Clustering of Applications (DBSCAN) clustering.

While the first two steps reduce the generality of the clustering method, it allows 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.

## References


[1] Gariel, M., Srivastava, A. N., and Feron, E., "Trajectory clustering and an application to airspace monitoring," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 12, No. 4, 2011, pp. 1511–1524.

[2] Girvin, R., "Aircraft noise-abatement and mitigation strategies," *Journal of Air Transport Management*, Vol. 15, No. 1, 2009, pp. 14–22.

[3] Sóbester, A., Czerski, H., Zapponi, N., and Castro, I., "High-Altitude Gas Balloon Trajectory Prediction: A Monte Carlo Model," *AIAA Journal*, Vol. 52, No. 4, 2014, pp. 832–842.

[4] Box, S., Bishop, C. M., and Hunt, H., "A stochastic six-degree-of-freedom flight simulator for passively controlled high power rockets," *Journal of Aerospace Engineering*, Vol. 24, No. 1, 2010, pp. 31–45.

[5] Makris, D. and Ellis, T., "Learning semantic scene models from observing activity in visual surveillance," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, Vol. 35, No. 3, 2005, pp. 397–408.

[6] Salaun, E., Gariel, M., Vela, A. E., and Feron, E., "Aircraft proximity maps based on data-driven flow modeling," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 2, 2012, pp. 563–577.

[7] Hu, W. H. W., Xiao, X. X. X., Fu, Z. F. Z., Xie, D., Tan, T. T. T., and Maybank, S., "A system for learning statistical motion patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 9, 2006, pp. 1450–1464.

[8] Morris, B. and Trivedi, M., "Trajectory Learning for Activity Understanding: Unsupervised, Multilevel, and Long-Term Adaptive Approach," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 33, No. 11, 11 2011, pp. 2287–2301.

[9] Rasmussen, C. E., Williams, C. K. I., Rasmussen, C. E., and Williams, C. K. I., *Gaussian Processes for Machine Learning*, MIT Press, 2006.

[10] Gaffney, S. and Smyth, P., "Trajectory clustering with mixtures of regression models," *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining KDD 99*, Vol. 10, ACM, 1999, pp. 63–72.

[11] Tay, M. K. C. and Laugier, C., "Modelling smooth paths using Gaussian processes," *Springer Tracts in Advanced Robotics*, Vol. 42, Springer, 2008, pp. 381–390.

[12] Cox, G. E., Kachergis, G., and Shiffrin, R. M., "Gaussian Process Regression for Trajectory Analysis," *Proceedings of the 34th Annual Conference of the Cognitive Science Society*, 2012, pp. 1440–1445.

[13] Zheng, Q. M. and Zhao, Y. J., "Probabilistic Approach to Trajectory Conformance Monitoring," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 6, 2012, pp. 1888–1898.

[14] Eerland, W. J. and Box, S., "Trajectory Clustering, Modelling and Selection with the focus on Airspace Protection," *AIAA Infotech @ Aerospace*, Jan 2016, p. 1411.

[15] Berndt, D. and Clifford, J., "Using Dynamic Time Warping to Find Patterns in Time Series," *Proc. of KDD Workshop*, Vol. 10, Seattle, WA, 1994, pp. 359–370.

[16] Bishop, C., *Pattern Recognition and Machine Learning*, Information Science and Statistics, Springer, 2006.

[17] Faraway, J. J., Reed, M. P., and Wang, J., "Modelling three-dimensional trajectories by using Bézier curves with application to hand motion," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, Vol. 56, No. 5, 2007, pp. 571–585.

[18] Justel, A., Peña, D., and Zamar, R., "A multivariate Kolmogorov-Smirnov test of goodness of fit," *Statistics & Probability Letters*, Vol. 35, No. 3, 1997, pp. 251–259.

[19] McAssey, M. P., "An empirical goodness-of-fit test for multivariate distributions," *Journal of Applied Statistics*, Vol. 40, No. 5, 2013, pp. 1120–1131.

[20] Nakayama, H., Arakawa, M., and Sasaki, R., "Simulation-based optimization using computational intelligence," *Optimization and Engineering*, Vol. 3, No. 2, 2002, pp. 201–214.