Statistical Modelling and Inference for Ocean Trajectory Data

Michael O'Malley, B.Sc.(Hons.), M.Res

Submitted for the degree of Doctor of

Philosophy at Lancaster University.

June 2022

STOR-i

excellence with impact

Abstract

Trajectory data are generated by tracking the position of objects that move through space, for example using GPS technology. This thesis focuses on the statistical modelling of thousands of trajectories to infer various oceanographic statistics of interest. The motivating example used throughout the thesis is the Global Drifter Program which provides over 25,000 trajectories of free-floating buoys known as drifters, the motion of which track near-surface currents.

In the first piece of work we propose a novel multi-output probabilistic prediction model. The motivation for this arises because drifter trajectories are often used by practitioners to calibrate and estimate oceanic models which predict northward and eastward velocities using two independent models. This independence assumption is seldom realistic. As such, we extend an existing framework known as Natural Gradient Boosting (NGBoost) to predict multivariate outputs. The model is applied to predict a conditional distribution of drifter velocities.

We then develop a novel method to compute the most likely path taken by drifters between arbitrary fixed locations in the ocean. We also provide an estimate of the travel time associated with this path. Our method, which utilises Markov transition

matrices, is purely data driven and requires no simulations of drifter trajectories, in contrast to existing approaches.

Finally, we propose an alternative method of estimating travel times in a model-free way. Specifically, we investigate how long it takes a collection of drifters to travel to and from locations of interest. However, this direct approach, when applied naively, results in estimates which can be biased, missing, and noisy. Hence we use transformed multidimensional scaling to lessen these undesirable properties. We discuss the merits and disadvantages of our two proposed methods for estimating travel times, and we provide real-world studies to motivate and justify each methodological approach.

Acknowledgements

I am extremely grateful for my supervisor Adam Sykulski for the extremely valuable guidance, endless patience and feedback over the whole process. He has provided me so many interesting opportunities over the duration of my problem. I will always appreciate how much I developed and learned due to his guidance.

The financial support from EPSRC is very much appreciated both for myself and the funding which goes towards STOR-i. I am also extremely grateful to everyone, both students and staff who have contributed to making STOR-i such an stimulating and enjoyable atmosphere to work in. Namely I would like to thank Jon, Idris, Kevin, Kim, Wendy, Jen, Oli and Nicky for all of their work making STOR-i what it is today. I am especially appreciate for my cohort for all the laughs provided over the last 4 years.

I would also like to take the opportunity to thank everyone who I had the privilege of working with on projects during my PhD. I learned a great amount from all of you and these interactions were some of the highlights of my PhD years. In particular I would like to mention Romuald Laso-Jadart, Alejandro Schuler, and Rick Lumpkin.

I would also like to thank my larger support group outside of PhD program for

all of their help and support over the years. All of the friends I have made from the canoe club have been so great at helping me forget about the day to day problems and stress. I also thank my family for all their motivation, patience, and always being there whenever I need them.

Last but certainly not least, I would like to thank Jess for all of her help and support through the PhD. She has been doing a great job in keeping me sane over the last few years. I honestly do not think I would have managed to get this far without her.

Declaration

I declare that the work in this thesis has been done by myself and has not been submitted elsewhere for the award of any other degree.

A version of Chapter 2 is available as a preprint as *Multivariate Probabilistic Regression with Natural Gradient Boosting* by Michael O'Malley, Adam M. Sykulski, Rick Lumpkin and Alejandro Schuler.

A version of Chapter 3 has been published as *Estimating the travel time and the most likely path from Lagrangian drifters* by Michael O'Malley, Adam M. Sykulski, Romuald Laso-Jadart and Amin Madoui in the Journal of Atmospheric and Oceanic Technology.

Chapter 4 is original work yet to be submitted for publication.

Contents

List of Figures

List of Tables

List of Abbreviations

CTDS	Continuous-Time Discrete-Space
DTDS	Discrete-Time Discrete-Space
GDP	Global Drifter Program
GB	Gradient Boosting
Lon	Longitude
Lat	Latitude
MCST	Monte Carlo Super Trajectory
MDS	Multi Dimensional Scaling
MLP	Most Likely Path
NOAA	National Oceanographic and Atmospheric Administration
NGBoost	Natural Gradient Boosting
NLL	Negative Log-likelihood
NN	Neural Network
RMSE	Root Mean Squared Error

Chapter 1

Introduction

Trajectory data arises from the tracking of objects (e.g. vehicles, animals, persons, floating objects) as they move through space. The modelling and analysis of trajectory data is extremely informative in many applications. The most common tracked quantity is position via a satellite positioning system such as GPS. However, often the device being tracked will also record other information which is of interest for example: the instantaneous speed of a car, temperature at that location, or heart rate of the individual being tracked. This thesis proposes novel statistical methodology to utilise and model the position observations from trajectory data to gain new insights.

Modelling trajectory data is a challenging area in itself. The process which generates the trajectory is often driven by a spatial-temporal process which can be difficult or impossible to model. In contrast, the common alternative data collection method is to measure a quantity at single or multiple fixed spatial locations. These measurements are typically easier to model individually, as only temporal differences are observed, but the joint distribution over multiple locations can be challenging to char-

acterise. The dichotomy between trajectories and fixed-point measurements is referred to in the fluid dynamics community as the *Lagrangian* versus *Eulerian* approach.

Additionally, the nature of trajectory data introduces unique challenges. Some common properties are:

- High errors on location observations. Typically the devices being tracked are made on a restricted budget which sometimes means compromises need to be made on data quality. Additionally, outliers in GPS locations can be caused by inclement weather, interfering objects, and limited satellite coverage.

- Irregular sampling. Often the frequency of location observations are dependent on satellite passes or are intentionally limited to preserve battery.

- Finite lifetime of trajectories. The trackers are usually battery operated, hence are subject to a finite battery life.

These combined challenges mean that trajectory data are in general: highly non-stationary, noisily observed, and irregularly and sparsely sampled—all of which in both space *and* time—which motivates their careful and bespoke modelling, as will be a common theme throughout this thesis.

1.1 The Global Drifter Program (GDP)

The linking component of the work in this thesis is the motivating application dataset: The Global Drifter Program (GDP) [Lumpkin and Centurioni, 2020]. In this section, we provide a brief background outlining the current state of the drifter data processing,

models, and how they are used. Each chapter of this thesis aims to build statistical methods and tools which can be used to utilise the dataset and infer interesting quantities. As a brief summary: Chapter 2 aims to predict a distribution on the drifter velocities given satellite observations such as wind and sea surface height; Chapters 3 and 4 aim to answer the question "How long would it typically take for an object to travel from point A to point B in the ocean?" As we introduce concepts in this section we shall indicate to which chapters they are relevant.

The GDP is a very influential dataset and entirely free to access. There are over 1,100 associated publications using this dataset[1]. The applications of which include: modelling sea surface temperature [Elipot et al., 2022, Kennedy et al., 2019], creating global ocean current models [Bonjean and Lagerloef, 2002], identifying ocean pathways [Drouin et al., 2022], search and rescue for the MH370 airline wreckage [Miron et al., 2019], identifying ocean garbage patches [van Sebille et al., 2012, Maximenko et al., 2012], and relating animal movements to currents [Bentivegna et al., 2007]. This motivates the importance of the work in this thesis. Specifically, by creating and improving general use tools, software, and statistical models for trajectory data, we help oceanographers and environmental scientists more broadly utilise the dataset to have impact in multiple domains.

The GDP is an initiative led by the National Oceanographic and Atmospheric Administration (NOAA). The program tracks the movements and measurements obtained by *drifters*. In Figure 1.1.1 we show a diagram of a drifter. This design has remained almost constant throughout the lifetime of the GDP in an attempt to keep

[1]Source: `https://www.aoml.noaa.gov/phod/gdp/bib/bibliography_chronological.php`

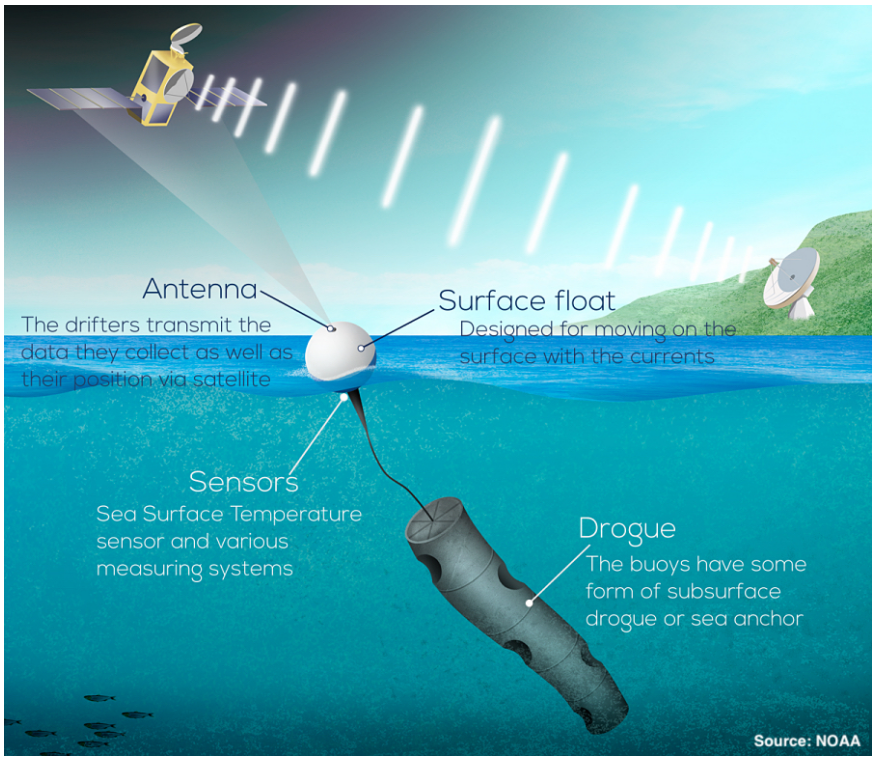

Figure 1.1.1: Diagram of a drifter.

the dataset homogeneous. The surface float contains the battery and various other sensors, and is designed to stay at or near the surface so it can communicate with satellites.

An important aspect to the drifter design is the drogue labelled in Figure 1.1.1. If an arbitrary floating object was just to drift at the surface, without anything attached, the currents which move that float would be a primarily a mix of three components:

- **Surface drift**. The surface currents in absence of wind.
- **Stokes drift**. Currents caused by wind waves.
- **Direct wind forcing**. Wind pushing directly on the object, causing it to move with the direction of the wind.

The main aim of the GDP is to understand and observe the surface drift as accurately and widely as possible. This is where the *drogue* sock comes into effect. The drogue sock, which is attached to the drifter by a tether, ranges from around $10m$ to $20m$ depth such that it is centered at $15m$. At these depths, Stokes drift and wind driven currents have less of an effect. For a more detailed description of the design and history of the GDP we refer the reader to Lumpkin et al. [2017].

As of July 2021, the GDP database consists of about $26,000$ drifters. Each drifter is dropped from a ship, then it is left to freely float in the ocean. Over the life of the drifter two key events may occur: first the drogue may detach and second the drifter "dies". The drogue falling off means the drifter is now just a float excluding the drogue as shown in Figure 1.1.1. When this occurs the drifter motions will be more influenced by wind and wave driven currents—but this data is still informative and is recorded and studied in its own right. There is a sensor which indicates if the drogue is still attached or not. After the drogue falls off we refer to the drifter as an *undrogued* drifter. The drifter dying can be caused by various phenomena, the most common categories being: washing ashore, stopping transmission due to the battery dying, and being picked up by a ship [Lumpkin et al., 2012]. The average lifetime of a drifter is about 410 days resulting in a total of 10.6 million days of observations when combining all the drifters.

Up until around 2016, the locations of most drifters were recorded by a satellite system called ARGOS. ARGOS typically has a standard deviation on location observations of around 500m with a larger standard deviation in the longitudinal direction than the latitudinal. In modern day, all drifters are tracked by GPS which has an

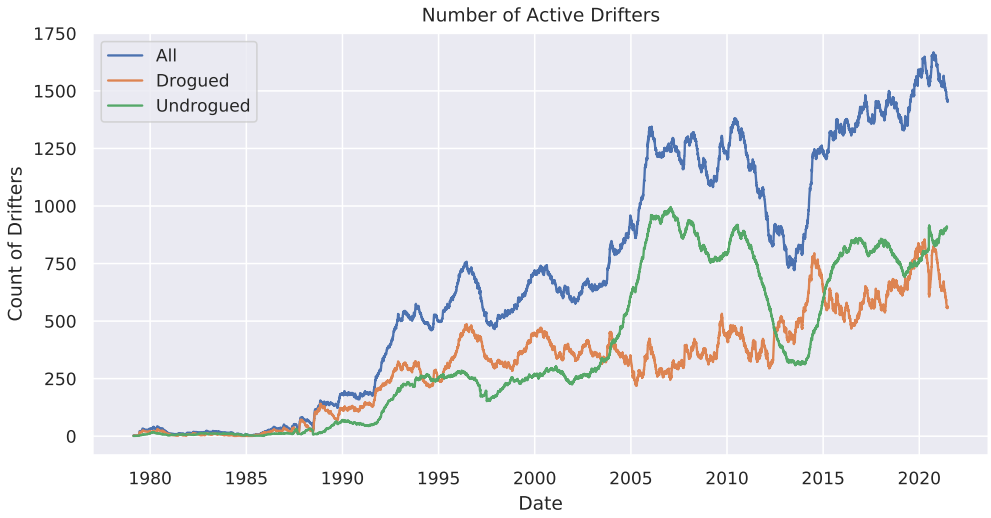

Figure 1.1.2: The time series of how many drifters are active at any given time. The "All" line is the sum of the drogued and undrogued data.

error on the order of a few meters, making the positioning error negligible for most applications, although outliers do still occur [Early and Sykulski, 2020]. Each drifter is fit with a number of sensors, one of the most popular sensors being a thermometer to measure sea surface temperature. Data recorded from the sensors are temporarily stored, then also transmitted over the ARGOS system at the same time as the location. Elipot et al. [2016] give a detailed comparison of ARGOS fixes to GPS.

In Figure 1.1.2, we show how many drifter observations are available at any point in time since 1980. One thing to note from this plot is how much of the data comes from undrogued drifters. Typically, around 50% of any of the data gathered is from undrogued drifters. To give further insight as to what these numbers translate to in terms of spatial coverage, we display maps in Figure 1.1.3 showing the active drifters at a few snapshots in time. The combination of these two figures shows the growth

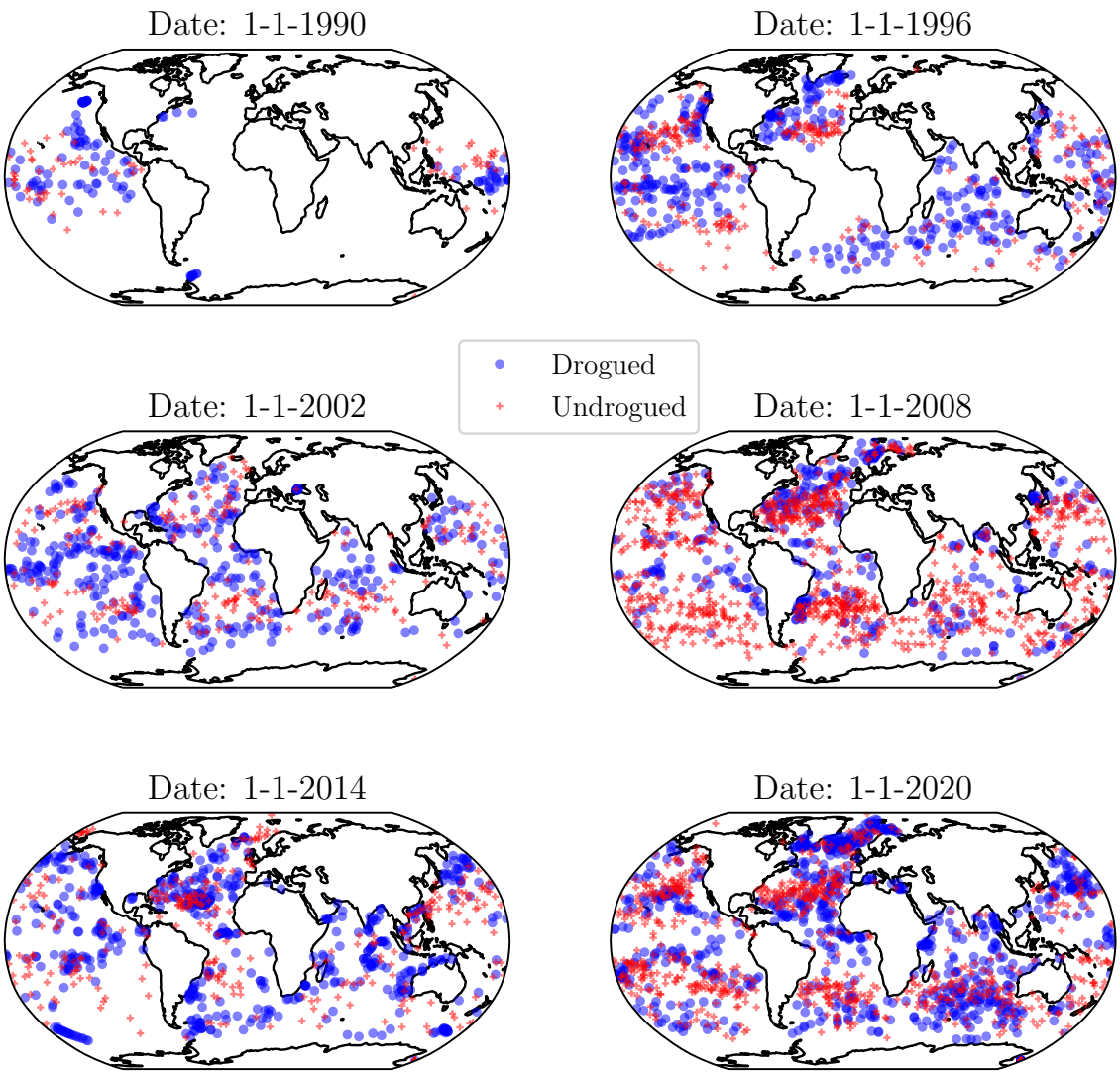

Figure 1.1.3: The locations of all active drifters at different snapshots of time. Each map shows the locations of all active drifters on the date listed. As per the legend, blue circles are drogued drifters, and red pluses are undrogued drifters.

and coverage of the GDP and how it has grown to a global scale.

Other Drifter-like Deployments. We note that the GDP is not the only source of surface current observations, it is just one of the largest. Other notable mentions include:

- **SOFAR** have a global array of thousands of drifters.
- **ARGO floats** [Lebedev et al., 2007] are designed to profile sea temperature and salinity at depths of 1000m. These floats rise to the surface to transmit data and remain at the surface for a number of days ($\circ$10 days) before sinking again, therefore their positions can be used as ocean surface observations in these periods at the surface.
- Regional focused deployments over a short time frame: **GLAD** [Özgökmen, 2016] and **LASER** [D'Asaro et al., 2017] in the Gulf of Mexico, or **LatMix** [Shcherbina et al., 2015] in the Sargasso Sea.

These deployments have a different design to the drifters tracked by the GDP, hence they need to be treated differently when modelled. For further reading, see for example Rio et al. [2014] for differences between ARGO and GDP drifter behaviour.

1.2 Movement Models for Drifter Trajectories

Ultimately, the sparsity of observations in both time and space limits the questions which we can answer with raw drifter data alone. However, we can fit statistical

models to make inferences based on the predictions and parameters. Here we give insight on two models which will be used in future chapters.

First, in Section 1.2.1 we introduce the reader to the Eulerian and Lagrangian specification of motion. In particular, this relationship leads to how we can use Lagrangian observations (which the drifter data are) to fit Eulerian models, then in turn how we can use these Eulerian models to simulate new synthetic Lagrangian observations. In particular, we give insight on how these models are ultimately used to aid decision making.

Secondly, by making the simplifying assumption that ocean currents are statistically stationary in time, we can make interesting observations directly from the data, as we shall investigate in Chapters 3 and 4. For this purpose, in Section 1.2.2 we therefore provide an introduction to discrete-time discrete-space Markov models for the GDP.

1.2.1 Continuous Position Models

What we observe when a drifter moves is the position of a particle which is influenced by some sort of fluid dynamics. Here we give a very brief introduction to a topic from fluid dynamics, which is the relationship between Eulerian and Lagrangian observations. In particular, we outline how this relationship leads to the core concept of how we can simulate new particles. Here we give all positions in $\mathbb{R}^2$ as the data we are interested in modelling are two-dimensional in space. These concepts also extend to work in higher dimensions, e.g., if the drifters could also sink to different depths.

On a practical note, when relating velocity and position in this chapter, we make an

assumption that the space in which we are working in is not affected by the curvature of the earth. For simplicity, references to locations will be referred to as easting and northing pairs from a fixed origin at $(0, 0)$. For example, $(1, 3)$ would be 1 meter east and 3 meters north of the origin. The easting-northing pairs will have an equivalent representation in a longitude-latitude coordinate pair. Outside of this chapter, the location data are in standard longitude-latitude pairs and velocities are in m/s. In practice, when doing computations with these velocities we first take a local projection from longitude-latitude (degrees) to easting-northing (km) then do the calculations in the eastern-northing space and translate the new location of interest to longitude-latitude space. Such tasks are easily implemented using packages such as proj [PROJ contributors, 2022].

A Lagrangian specification is when we follow a particle as it moves around space. We consider the position of a particle at time t, which starts at location $\mathbf{r}_0 \in \mathbb{R}^2$, as $\mathbf{r}(t, \mathbf{r}_0) \in \mathbb{R}^2$. A quantity which is of interest is the velocity, which is the time derivative of the particle's location:

$$\mathbf{v}(t, \mathbf{r}_0) = \frac{\partial}{\partial t}\mathbf{r}(t, \mathbf{r}_0). \tag{1.2.1}$$

This velocity and position definition are in the Lagrangian specification of motion. In other words, the position and velocity of the particle are both functions of the start point and the current time. In practice, the start point $\mathbf{r}_0$ is chosen by the deployment team on board the deploying ship and is sometimes selected to study specific regions of interest or ensure a good spread of global coverage.

The alternative definition of motion is a Eulerian specification which is defined as

the spatial-temporal function returning the velocity at position $\mathbf{x} \in \mathbb{R}^2$ and time t:

$$\mathbf{u}[\mathbf{x}, t]. \tag{1.2.2}$$

The relationship between the Lagrangian and Eulerian specifications is $\mathbf{u}[\mathbf{r}(t, \mathbf{r_0}), t] = \mathbf{v}(t, \mathbf{r}_0)$, i.e., the Lagrangian velocity of trajectory at time t is the Eulerian velocity at the position of the object at time t.

Estimating Lagrangian Velocity from the data. The velocity which a particle moves at is often a sought after quantity. However, in practice drifters only record and transmit their positions at irregular intervals. Furthermore, these location fixes are error prone. Hence, we have noisy observations of $\mathbf{r}(t, \mathbf{r}_0)$ for a range of sampled t and we want to infer the continuous function $\mathbf{r}$ for all t and the derivative of that function with respect to t. This is a well studied problem; the approach which is commonly used for the GDP is the Kriging method first introduced for drifter data by Hansen and Poulain [1996]. Kriging (equivalently known as Gaussian Process Regression) is an interpolation method which allows one to predict the value of a function at an unsampled point in time using the temporally close sampled points. These predictions are provided in the GDP dataset to provide positions and velocities (with their associated estimated variance) at 00:00, 06:00, 12:00 and 18:00 to form the 6-hourly product which we use throughout this thesis [Lumpkin and Centurioni, 2020].

In recent years, due to technological advancements, the positioning errors have become smaller and the location readings are more frequent, warranting a more up to date product. Elipot et al. [2016] provide an alternative interpolation method

to Kriging. A weighted linear model with t-distributed errors is fit to the nearest four points (two before and two after the time of interest). Due to higher sampling frequencies in recent years, this product is provided hourly, allowing higher frequency phenomena to be observed. In present day, a normal drifter provides a GPS location fix occurring near the start of every hour, motivating the hourly resolution. However, for the purposes of this thesis the 6-hourly data is sufficient, and has the advantage of going back further in time to the GDP's inception in 1979.

Both of these interpolation methods model the Lagrangian locations directly. The Lagrangian velocity is provided by computing or estimating the derivative of the interpolation function. We shall use this velocity estimate in Chapter 2 as the target of our model.

Going from Lagrangian to Eulerian. One of the applications of the GDP is to use these Lagrangian observations to create Eulerian velocity maps. In particular, we aim to learn a function $\mathbf{u}(\mathbf{x}, t)$ which explains the Lagrangian velocities. The estimation of the spatial-temporal function $\mathbf{u}$ is often considered as a statistical challenge. That is we aim to minimize the difference between the Eulerian and Lagrangian products

$$\hat{\mathbf{u}} = \arg\min_{\mathbf{u}} \sum_i d(\mathbf{u}(\mathbf{x}_i, t_i), \hat{\mathbf{v}}_i),$$

where $\hat{\mathbf{v}}_i$ is the Lagrangian velocity estimate from drifters at point $\mathbf{x}_i$ and time t_i; and d is a distance function; typically this is taken as mean squared error (MSE).

A common approach is to fit a model for the observed Lagrangian velocities in a sliding window fashion. We take all drifters observed in a spatial longitude-latitude

$x^\circ \times x^\circ$ box then fit parameters for that location. The collection of all of these models is then taken as the spatial model. As a straightforward example, we consider a *mean-flow* model. We create a purely spatial mean flow map in Figure 1.2.1 by taking the mean in each box. Such a map is often referred to as a pseudo-Eulerian [Lumpkin et al., 2017] or quasi-Eulerian [Fratantoni, 2001] velocity field. The map resolves all large-scale major oceanographic features, such as the Equatorial and Antarctic Circumpolar currents which are primarily longitudinally oriented (top panel). However, this method assumes that currents are constant over time: this is not the case.

The use of a $x^\circ \times x^\circ$ longitude-latitude box for spatial gridding is generally flawed. We give a detailed discussion of the use of these gridding systems in Chapter 3 along with an alternative option. In particular, one of the main flaws is that the area of each grid becomes smaller as the absolute value of latitude increases.

To try to account for variations over time, we can consider the relationship of drifter velocities to various sources of remote sensed information relating to wind and sea surface height, such as those used in Rio et al. [2014], Mulet et al. [2021]. These models are particularly interesting as they can be used to compare the different effects of the variables on drogued and undrogued drifters [Lumpkin et al., 2013].

In Chapter 2 we propose a gradient boosting model which aims to relate remotely sensed information to the velocity data obtained from drifters. We do this by using covariate information relating to the main drivers of surface drift (sea surface height) and wind-driven currents. Due to the non-parametric nature of the model, it could also be used to test other covariates such as those related to Stokes drift.

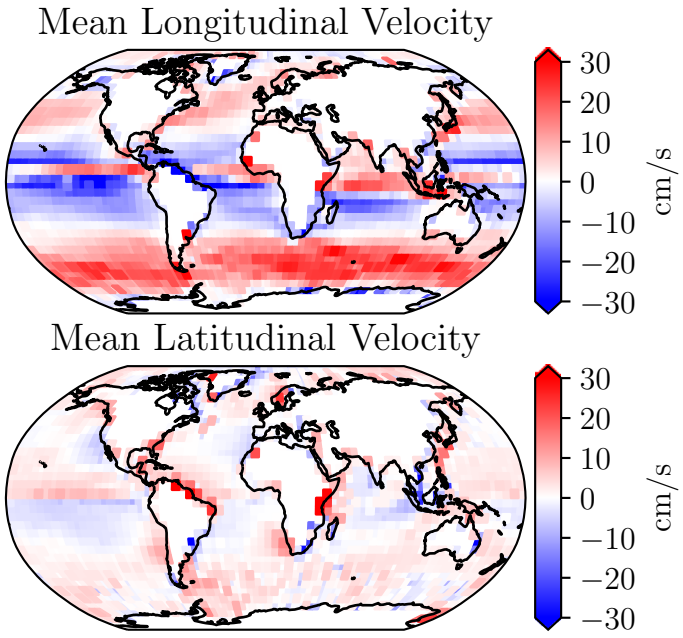

Figure 1.2.1: A plot of the mean current derived from all the driver observations. The mean of the longitudinal or latitudinal velocities are taken in $5^\circ \times 5^\circ$ bin. The resolution is chosen for visual purposes.

Going from Eulerian to Lagrangian Often the analysis of ocean currents starts from a Eulerian product in the form of either a spatial-temporal model or a dataset providing ocean velocity estimates over a grid of points in space and time. The proposed impact of these models is that they can be used to track where objects or masses of liquid end up. However, simply analysing the currents and flows on these spatial-temporal maps will likely not produce actionable information.

For example, if we were trying to decide where to search for ocean plastics, how do we use a Eulerian model to figure this out? One way to utilise the Eulerian model is to produce thousands of synthetic Lagrangian particles and then observe where these particles end up. By then analysing the distribution of where all these particles end up, decisions can be made. We introduce how one can simulate these particles here.

A common way to use an ocean current product is to generate a new artificial trajectory, where $y(t)$ denotes the position of the trajectory at time t. The simplest model to simulate such a trajectory is to solve the initial value problem:

$$y(t) = y_0 \quad \nabla y(t) = \mathbf{u}(y(t), t), \tag{1.2.3}$$

where we want to find $y(t + h)$.

The integration in Equation (1.2.3) can be solved by a numerical integration method, the simplest of which is using Euler's method:

$$y(t + h) = y(t) + h\mathbf{u}(y(t), t). \tag{1.2.4}$$

The ocean current products are generally provided as gridded products, hence $\mathbf{u}$ will usually require interpolation both in time and space. A review of the analysis and the usefulness of synthetic Lagrangian particles can be found in van Sebille et al. [2018].

The view presented in this section only provides a brief review on this method of analysing drifters. We note that this is just one of the many methods to model trajectories. We refer the reader to LaCasce [2008] for a more complete overview, which includes statistical methods to estimate diffusivity, and analysing multiple trajectories starting from similar points in space and time (see also Beron-Vera and LaCasce [2016] and Oscroft et al. [2021] for more recent analyses).

1.2.2 Markov Chain

Another common model to analyse trajectories is to form a discrete-time discrete-space Markov chain. Unlike the aforementioned continuous time movement models, we do not directly estimate a measurement of velocity. The Markov chain models the probability of being in position y given it was in position x one time epoch ago.

In summary, a discrete-time discrete-space (DTDS) Markov chain gives a density on the movements of an object in a state space, $s \in \mathcal{S}$, where $\mathcal{S} = \{1, ..., N\}$ is a set of all N possible states. Let $\{x_i\}_{i=1}^{t}$ be a stochastic process where $x_t \in \mathcal{S}$ is the state of the process at time t. The Markov property means that if we are interested in predicting the future state (x_{t+1}), we have the following property:

$$P(x_{t+1} = y | x_t, \ldots, x_1) = P(x_{t+1} = y | x_t).$$

In other words, the future state of the system is independent of the past states, given we know the current state of the system.

The probability $P(x_t = i | x_{t-1} = j)$ is a function of only two variables ($i, j \in \mathcal{S}$) hence we can store the model in a so called *transition matrix*. The transition matrix

is an $N \times N$ matrix with entries:

$$T_{ij} = P(x_t = i | x_{t-1} = j).$$

An issue occurs as we are trying to model the motion of a continuous-time continuous-space model by a discrete-time discrete-space model. Therefore, we must first create functions to discretize space and time.

The space is often discretized as a $l^\circ \times l^\circ$ longitude-latitude grid system, where l is a variable which must be specified in advance. The time is discretized by selecting the number of days k and setting k days as the physical time difference between x_t and x_{t-1}. Therefore, the resulting interpretation of the Markov chain is that: If the process is in box i on day 0, the process will move to box j on day k with probability T_{ij}. The results which a Markov chain will infer are heavily influenced by both the physical time difference k and the grid size l, as we shall show in Chapter 3.

Markov models have proven useful for various applications, likely due to how easy they are to deploy and explain. One can store a model which summarizes the entire ocean by storing only approximately 10MB of data. Some notable applications using drifter data include: finding the destinations of ocean debris [van Sebille et al., 2012, Maximenko et al., 2012], tracking debris from the MH370 airliner to aid search and rescue [Miron et al., 2019], investigating the effect of ocean currents in plankton genomics [Laso-Jadart et al., 2021], identifying main pathways of ocean currents [Drouin et al., 2022], and analyzing how well connected oceanic regions are [Froyland et al., 2014]. This method is also used for the website `http://plasticadrift.org/`, which is an excellent visual tool showing where an object floating on the surface of the ocean

might travel to.

The Markov model approach presented here and which will be used in Chapter 3 assumes that the domain is stationary over time, which is seldom realistic. There are time trends due to changes in ocean dynamics and seasonal effects. Approaches have been proposed to account for this which generally involve taking windowed estimates of the transition matrix (e.g., see van Sebille et al. [2012] for a seasonal example). In the future work of this thesis, Section 5.1.3, we propose to use methodology from the animal tracking literature to estimate transition rates using covariate information. Such a method could be used to account for seasonal effects and even be fit to remotely sensed observations.

1.3 Contributions of this Thesis

Chapter 2

In Chapter 2, we propose a method that can create a spatial-temporal model predicting Lagrangian ocean velocities using the same observational data sources as Rio et al. [2014]. The model utilises a framework known as natural gradient boosting which predicts a distribution rather than a single point. This methodology is provided in a very general context such that it can be used in other applications. The software related to this chapter is provided as such.

One of the intended usages of the model would be to use the predictions as the Eulerian velocity field $\mathbf{u}$ to then simulate new trajectories. The model predicts a conditional distribution of the velocity estimates provided by the 6-hourly GDP product.

We could use the conditional distribution provided by the predictions of the model when simulating synthetic trajectories. As the model is specifically trained to replicate drifter behaviour, the synthetic trajectories should be a close match with the statistical properties of drifter trajectories.

This chapter is currently under review under the title *Multivariate Probabilistic Regression with Natural Gradient Boosting* by Michael O'Malley, Adam M. Sykulski, Rick Lumpkin and Alejandro Schuler. A preprint is available on arXiv [O'Malley et al., 2021]. The related code to the paper is available as part of the open source package `https://github.com/stanfordmlgroup/ngboost`. A sample usage of the package can be seen in Section A.D.4.

Chapter 3

In Chapter 3, we develop a novel method for computing the most likely path taken by drifters between two arbitrary points in the ocean. We also provide an estimate of the travel time associated with this path. We show results of the method on various location pairs, showing that it works on global scales.

Chapter 3 has been published in the Journal of Oceanic and Atmospheric Technology: "Estimating the travel time and the most likely path from Lagrangian drifters" O'Malley et al. [2021]. Additionally, an open source Python package is available for accessible application at `https://github.com/MikeOMa/DriftMLP`. A demonstration of the package is advertised on the NOAA website at `https://www.aoml.noaa.gov/phod/gdp/derived_pathways.php`.

Additionally, we supplied data to the co-authors of the paper O'Malley et al.

[2021] to aid a study titled *"How marine currents and environment shape plankton genomic differentiation: a mosaic view from Tara Oceans metagenomic data"* by Laso-Jadart, Romuald; O'Malley, Michael; Sykulski, Adam; Wincker, Patrick; Ambroise, Christophe; Madoui, Mohammed-Amin. The paper is currently under revision and available on bioRxiv at `https://doi.org/10.1101/2021.04.29.441957`. The details of the supplied data are given in the Appendix of Chapter B.J.

Chapter 4

In Chapter 4, we provide an alternative approach to estimating how long it takes to travel between two arbitrary points in the ocean. A direct approach is taken in that we look for drifter trajectories which have travelled between the two points. This method initially produces estimates which are prone to bias, noise, and missingness. We propose the use of a distance matrix modelling approach known as multidimensional scaling to reduce these effects. Results are shown on a simulated and real example. This chapter is distinct from Chapters 2 and 3 in that it does not fit a model for drifter motion, and thus provides a complementary analysis to the earlier chapters. Chapter 4 is an original work to this thesis.

Chapter 2

Multivariate Probabilistic Regression with Natural Gradient Boosting

2.1 Introduction

The standard regression problem is to predict the value of some continuous target variable $\mathbf{Y}$ based on the values of an observed set of features $\mathbf{X}$. Under mean squared error loss, this is equivalent to estimating the conditional mean $\mathbb{E}[\mathbf{Y}|\mathbf{X}]$. Often, however, the user is also interested in a measure of predictive uncertainty about that prediction, or even the probability of observing any particular value of the target. In other words, one seeks to estimate $p(\mathbf{Y}|\mathbf{X})$. This is called *probabilistic* regression.

Probabilistic regression has been approached in multiple ways; for example, deep distribution regression [Li et al., 2021], quantile regression forests [Meinshausen and

Ridgeway, 2006], and generalized additive models for *Location, shape, scale* [Rigby et al., 2019]. All the listed methods focus on producing some form of outcome distribution. Both Li et al. [2021] and Meinshausen and Ridgeway [2006] rely on binning the output space, an approach which will work poorly in higher output dimensions. Here we focus on a method closer to Rigby et al. [2019] where we prespecify a form for the desired distribution, and then fit a model to predict the parameters which quantify this distribution. Such a parametric approach allows one to specify a full multivariate distribution while keeping the number of outputs which the learning algorithm needs to predict relatively small. This is in contrast to deep distribution regression [Li et al., 2021] for example, where the number of outputs which a learning algorithm needs to predict becomes infeasibly large as the dimension of the target data $\mathbf{Y}$ grows.

A common approach for flexible probabilistic regression is to use a standard multivariate regression model to parameterize a probability distribution function. This approach has been extensively used with neural networks [Williams, 1996, Sützle and Hrycej, 2005, Rasp and Lerch, 2018]. Recently Duan et al. [2020] proposed an algorithm called Natural Gradient Boosting (NGBoost) which uses a gradient boosting based learning algorithm to fit each of the parameters in a distribution using an ensemble of weak learners. A notable advantage of this approach is that it does not require extensive tuning to attain state-of-the-art performance. As such, NGBoost works out-of-the-box and is accessible without machine learning expertise.

Our contribution is to construct and test a similar method that works for multivariate probabilistic regression. The standard approach for multi-output regression is to either fit each dimension entirely separately, or assume that the residual errors

in both output dimensions are independent, allowing the practitioner to factor the objective function. In reality, however, output dimensions are often highly correlated if they can be predicted using the same input data. When the task requires a measure of uncertainty, we must therefore model the *joint* uncertainty in the predictions.

As a simple illustrative example of where a joint uncertainty prediction is useful, consider the two joint distributions of rainfall and temperature as in Figure 2.1.1. Suppose these are the forecasts we have predicted and we are interested in the probability that it is going to rain at most 2mm, while the temperature does not go higher than 12°. Using the independent forecast we get a probability of 0.14; whereas using the fit which allows correlations between output dimensions, we get a probability of 0.25 based on the joint forecast. The benefits of joint probabilistic regression are clear: (1) the distribution allows one to make more accurate decisions based on both outcomes, and (2) the correlation measure is often a quantity of interest in itself (e.g., in forecasting stock prices for portfolio selection).

The motivating application of this chapter is to predict two-dimensional velocity outputs. Such outputs are commonplace in environmental applications such as ocean currents [Maximenko et al., 2009, Sinha and Abernathey, 2021] and wind [Lei et al., 2009]. In these approaches, a diagonal covariance matrix is often assumed; which will seldom be the case in practice. In Section 2.4, we provide a real-world large scale example of using our approach to predict the multivariate distribution of ocean currents using remotely-sensed satellite data. Ocean current models are often used in practice by decision makers for applications such as predicting the locations of debris, plastics and oil spills. Therefore by modelling the joint two-dimensional stochasticisity

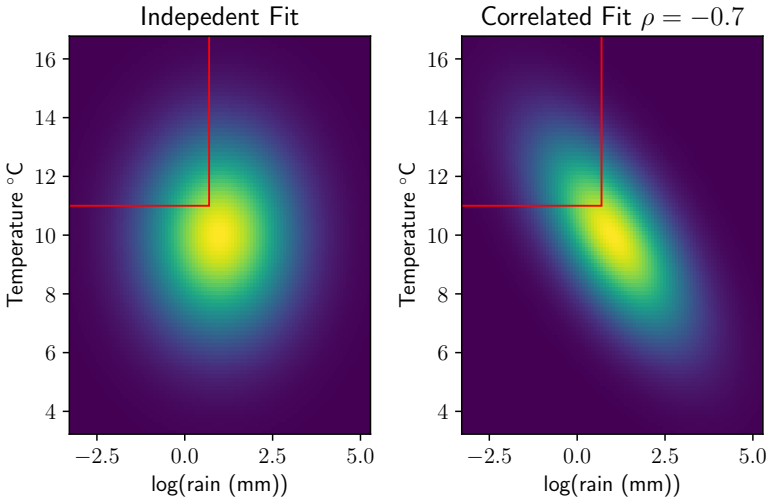

Figure 2.1.1: An example distribution of log rainfall and temperature for a day which we know will have rain. Both distributions are Gaussian and have the same mean vector and marginal variances. On the right plot the correlation between the two dimensions is set to -0.7, whereas on the left plot it is 0.

of directional currents more accurately, this will in turn translate to better forecasts and decisions being made, as well as better measures of associated uncertainty.

In summary, in this chapter we present an NGBoost algorithm that allows for multivariate probabilistic regression. Our key innovation is to place multivariate probabilistic regression in the NGBoost framework, which allows us to model all parameters of a *joint* distribution with flexible regression models. At the same time, we inherit the ease-of-use and performance of the NGBoost framework. In particular, we demonstrate the use of a multivariate Gaussian NGBoost parametrization in simulation and in the use case described above. The results show that: (1) joint multivariate probabilistic regression is more effective than naively assuming independence between the outcomes, and (2) our multivariate NGBoost approach outperforms a state-of-the-art neural network approach. Our code is freely available as part of the `NGBoost` Python package.

2.1.1 Notation

Let the input space be denoted $\mathcal{X} = \mathbb{R}^d$ and the output space be denoted $\mathcal{Y} = \mathbb{R}^P$. The aim of this chapter is to learn a conditional distribution from the training dataset $\mathcal{D} = \{\mathbf{X}_i, \mathbf{Y}_i | 1 \leq i \leq N \ \mathbf{X}_i \in \mathcal{X}, \mathbf{Y}_i \in \mathcal{Y}\}$. Let $p(\mathbf{Y}|\boldsymbol{\theta})$ be a probability density of the observations $\mathbf{Y} \in \mathcal{Y}$ parameterized by $\boldsymbol{\theta} \in \mathbb{R}^M$, where M is the dimension of the parameters. In this work we aim to learn a mapping such that the parameter vector varies for each data-point, i.e., a function $f : \mathcal{X} \to \mathbb{R}^M$, inducing the distribution function of $\mathbf{Y}$, $p\left(\mathbf{Y}|\boldsymbol{\theta} = f(\mathbf{X})\right)$. For convenience we shall henceforth adopt a shorter notation: $p\left(\mathbf{Y}|\boldsymbol{\theta} = f(\mathbf{X})\right) = p\left(\mathbf{Y}|\mathbf{X}\right)$.

2.2 Approach

Rather than attempting to produce a single point estimate for the response ($\mathbb{E}[\mathbf{Y}|\mathbf{X}]$) in this paper we aim to learn a distribution of $\mathbf{Y}$ conditional on $\mathbf{X}$. When fitting the conditional distribution we propose the use of the Natural Gradient Boosting (NGBoost) framework [Duan et al., 2020]. NGBoost is a method for conditional distribution regression. The approach is based on combining the well-proven approach of boosting [Friedman, 2001] with Natural Gradients [Amari, 1998] in place of ordinary gradients. In 2.2.1, we give a brief introduction to both of these concepts before introducing NGBoost in 2.2.1. Finally, we introduce and motivate the use of the Multivariate Gaussian Distribution in section 2.2.3.

2.2.1 Preliminaries

Gradient Boosting

In the classical setting as described in Friedman [2001] gradient boosting is an algorithm which aims to learn a function $\mathcal{X} \to \mathbb{R}$ in a non-parametric fashion, which minimizes some loss function L. Typically, this loss function is the mean squared error loss. The extension to the higher dimensional case is relatively straightforward, we aim to learn a function mapping from $\mathcal{X}$ to $\mathbb{R}^M$, which minimizes a loss function $L : (\mathcal{Y} \times \mathbb{R}^M) \to \mathbb{R}$. In the typical setting we would have $M = P$ and the output would be the expectation of $\mathbf{Y}$. Note that in the following formulation we re-obtain the classical gradient descent setting by taking $M = 1$. We aim to fit an additive

model of the following form:

$$f(\mathbf{X}) = \sum_{b=1}^{B} \beta_b \mathbf{h}(\mathbf{X}, \mathbf{a}_b) \tag{2.2.1}$$

where $\beta_b \in \mathbb{R}$ is the additive weight. The function $\mathbf{h}(\mathbf{X}, \mathbf{a}_b)$ is known as a base learner, weak learner or basis functions with parameters $\mathbf{a}_b$. The output of $\mathbf{h}$ is M dimensional. In practice this can be any model however decision trees are the usual choice and also what we use here.

Gradient boosting aims to find parameters β_b and $\mathbf{a}_b$ which minimize the loss:

$$\{\beta_b, \mathbf{a}_b\}_{b=1}^{B} = \underset{\mathbf{a}_b, \beta_b}{\arg\min} \underset{b \in \{1,\ldots,B\}}{} \sum_{i=1}^{N} L\left(\sum_{b=1}^{B} \beta_b \mathbf{h}(\mathbf{X}_i, \mathbf{a}_b).\mathbf{Y}_i\right). \tag{2.2.2}$$

However such an optimization problem is typically very difficult so instead a greedy stage-wise heuristic is used which is based on functional gradient descent [Friedman, 2001]. We start with a flat estimate for the ensemble $f^{(0)} = 0$ and fit β_b, $\mathbf{a}_b$ sequentially, defining $f^{(b)}(\mathbf{X}) = f^{(b-1)}(\mathbf{X}) - \beta_b h(\mathbf{X}, \mathbf{a}_b)$.

This sequential fitting is done by doing gradient descent in the function space. We aim to take a step, in the direction of steepest descent under the constraint that the direction in which we go can be predicted by $\mathbf{h}$. We define the functional gradient as

$$g_m^{(b)}(\mathbf{X}_i) = \left[\frac{\partial L(\mathbf{y}_i, f(\mathbf{X}_i))}{\partial f(\mathbf{X}_i)_m}\right]_{f(\mathbf{X}_i)=f^{(b-1)}(\mathbf{X}_i)} \qquad m \in \{1, \ldots, M\}.$$

The next iteration of the model starts by finding the parameters $\mathbf{a}$ which minimize the mean squared error between the fitted base learner and the functional gradients:

$$\mathbf{a}_b = \arg\min_{\mathbf{a}} \sum_{i=1}^{N} \sum_{m=1}^{M} \left(g_m^{(b)}(\mathbf{X}_i) - \mathbf{h}(\mathbf{X}_i, \mathbf{a})_m\right)^2. \tag{2.2.3}$$

Then β_b is found via a line search, minimizing the following:

$$\beta_b = \arg\min_{\beta} \sum_{i=1}^{N} L\left(\sum_{k=1}^{b-1} \beta_k \mathbf{h}(\mathbf{X}_i, \mathbf{a}_k) - \beta \mathbf{h}(\mathbf{X}_i, \mathbf{a}_b), \mathbf{Y}_i\right). \tag{2.2.4}$$

This is found the same way as in Duan et al. [2020] by starting with $\beta = 1$, then successively halving β until the objective in (2.2.4) is seen to increase. We then stop the halving and take the value which was seen before the increase.

Base Learner We use a decision tree as the base learner as this is by far the most popular for gradient boosting in general. We use the implementation in the python package scikit-learn [Pedregosa et al., 2011], however any other base learner could easily be used. The decision tree partitions the input dataset based on the values of the features $\mathbf{X}$ then predicts a constant mean within each partition, we refer to these partitions as leafs. We refer to the depth of a leaf as the number of splits which led to that leaf.

We control 3 hyper-parameters to control fitting. The maximum depth of the tree, the maximum number of leafs, and the minimum amount of data points required in each split. The fitting algorithm grows in a best leaf manner. It picks the leaf which results in the best error reduction at that time. The algorithm will not allow a split if the maximum number of leafs has been reached, the depth of the new leaf will be higher than the max depth, or if the number of data-points in one of the leafs after splitting will be lower than the minimum specified.

As we need a base learner which predicts a value in $\mathbb{R}^M$, we fit M independent decision trees, each tree being one of the entries in the vector-valued output. In practice this allows us to factorize the objective 2.2.3 and therefore just do M independent fits of a univariate base learner. We concatenate the relevant parameters for each tree to form what we refer to as $\mathbf{a}_b$ in the previous text.

Early Stopping

In gradient boosting the number of base learners B in 2.2.1, has a large impact on how complicated the model can be. Generally having too large of an B will lead to overfitting, and too small of an B will lead to underfitting. The greedy nature of how we fit a gradient boosting model introduces a computationally efficient way to pick this value M. The strategy is known as early stopping. We randomly partition the training data $\mathcal{D}$ into a training set and validation set. Each $\mathbf{X}_i, \mathbf{Y}_i$ pair will appear in only one of the training and validation sets.

When finding the parameters for the function β_b and $\mathbf{a}_b$ we only use the training set. Then in each boosting iteration, after finding the parameters we evaluate and record the loss function on the validation set given the current model. The validation set is an unseen dataset, hence when the loss starts to increase on the validation set we can interpret this as the generalization abilities of the model getting worse. This is where we introduce a patience parameter k. Suppose we see the loss at iteration B_{min} is lower than $B_{min} + 1, \ldots, B_{min} + k$, then we simply stop the algorithm and revert the function estimate back to the state at B_{min} iterations and this becomes the final model. Alternatively, one can fix B_{min} and then retrain the boosting model from scratch without early stopping and using the entire dataset (validation and train combined) to fit the model.

Natural Gradients

Suppose $L : \mathbb{R}^M \rightarrow \mathbb{R}$, one way to intuitively explain the gradient of L is by the solution to:

$$\nabla L(\boldsymbol{\theta}) \propto \lim_{\psi \rightarrow 0} \underset{||\boldsymbol{\epsilon}|| < \psi}{\arg\min,} L(\boldsymbol{\theta} + \boldsymbol{\epsilon}). \tag{2.2.5}$$

for $\psi \in \mathbb{R}$, $\boldsymbol{\epsilon} \in \mathbb{R}^M$ and $\boldsymbol{\theta} \in \mathbb{R}^M$. It is the direction of steepest descent in a infinitesimally small neighbourhood of the current point. However, when optimizing a distribution this constraint $||\boldsymbol{\epsilon}|| < \psi$ often does not make sense. For example should the difference between two Gaussian distributions with equal variance and with means 1 and 2 be 1? These two distribution would be almost identical if they both had a variance of 10000 but would be extremely different if they had a variance of 0.1.

Instead, a so called natural gradient [Amari, 1998, Martens, 2020] is a sensible representation where we replace the Euclidean distance in 2.2.5 with an alternative way to measure how distant two points in the parameter space. In this chapter we use the KL divergence. We denote the so called natural gradient as $\tilde{\nabla} L$, and it takes the following form:

$$\tilde{\nabla} L(\boldsymbol{\theta}) \propto \lim_{\psi \rightarrow 0} \underset{||KL(\boldsymbol{\theta}, \boldsymbol{\theta} + \boldsymbol{\epsilon})|| < \psi}{\arg\min} L(\boldsymbol{\theta} + \boldsymbol{\epsilon}). \tag{2.2.6}$$

Where $KL(\boldsymbol{\theta}, \boldsymbol{\theta} + \boldsymbol{\epsilon})$ denotes the KL divergence between the distribution with parameters $\boldsymbol{\theta}$ and the distribution resulting from using parameters $\boldsymbol{\theta} + \boldsymbol{\epsilon}$. This has a straightforward solution which is just to multiplying the ordinary gradient by the inverse fisher information $\tilde{\nabla} L(\boldsymbol{\theta}) = \mathcal{I}(\boldsymbol{\theta})^{-1} \nabla L(\boldsymbol{\theta})$ [Amari, 1998].

For demonstration on how this relatively simple adaptation can help, we give a simple example below with a Gaussian distribution. We simulate 1000 points from

a $\mathcal{N}(\mu = 0, \sigma^2 = 16)$ distribution, then we run a gradient descent algorithm to find the maximum likelihood estimate for both the mean and variance parameters. We re-parameterize $\sigma = \exp(a)$, then optimize a as it is unconstrained. A step size of 0.1 is used. We show the results in Figure 2.2.1.

With the ordinary gradient the variance parameter quickly goes around the value 4.2 which is roughly the root mean squared error (RMSE) between the samples and the value 2. Then the mean parameter estimate very slowly moves to 0, the standard deviation estimate follows the RMSE. With natural gradient descent the variance estimate learns at a much slower rate, eventually going higher than the RMSE then converges to the sample RMSE. Even for this simple example after 300 iterations ordinary gradient descent has not learned a good estimate of the true mean or variance whereas natural gradient descent did so in around 50 iterations.

2.2.2 Natural Gradient boosting

NGBoost [Duan et al., 2020] is a boosting algorithm developed to fit conditional distributions. The natural gradient is used in place of the ordinary gradient and the algorithm fits a gradient boosting model where the objective is a *Scoring rule*. In place of the typical task of learning a single value (or vector in the multi-output setting) for the expectation of the response, we instead learn a function from the features $\mathbf{X}$ which output the parameters for a probability distribution function. We then assess the goodness of fit of these parameters with a scoring rule rather than the MSE.

The first task required for natural gradient boosting is to pick the type of distribution which will be used to predict $\mathbf{Y}$, $p(\mathbf{Y}|\boldsymbol{\theta})$. For example if it is continuous

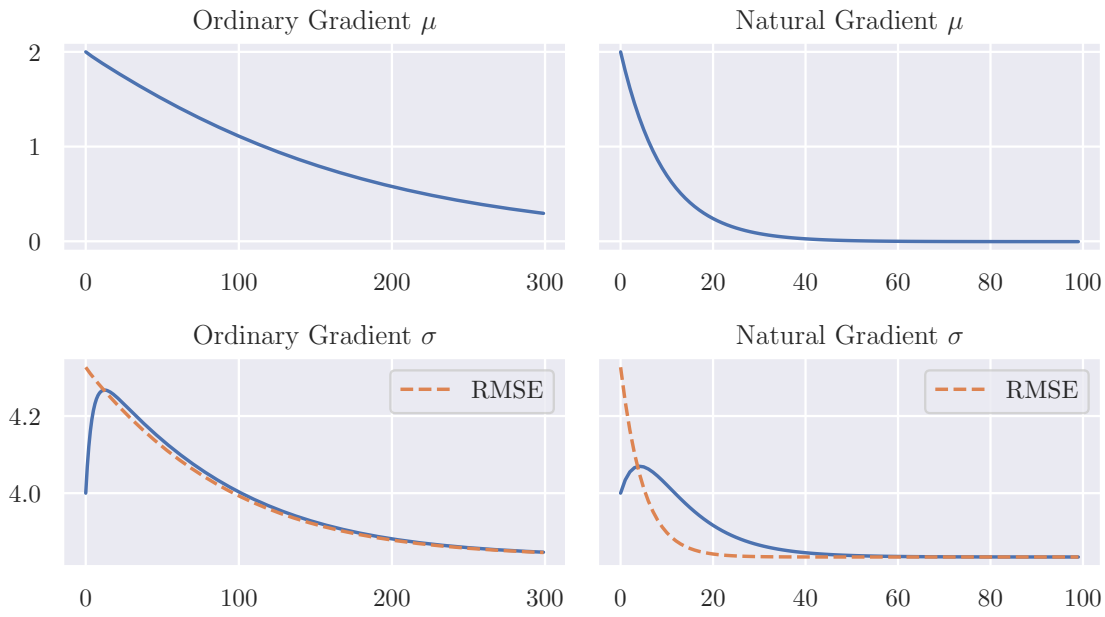

Figure 2.2.1: Gradient Descent using the ordinary gradient and the natural gradient with a stepsize of 0.1. The objective is taken to be the log-liklihood for 500 samples from a $\mathcal{N}(0, 4)$ starting from $\mu = 2, \quad \sigma = 4$. The RMSE is plotted on the variance plot at the current mean estimate. Natural gradient descent converged after 100 iterations so the x axis has a smaller range.

and univariate ($P = 1$) we can choose a Gaussian distribution, or alternatively if we believe it has heavier tails we could specify a t distribution. In Section 2.2.3 we introduce the multivariate Gaussian distribution which will be used in this Chapter.

When using gradient boosting we use the output of the function approximation f as the parameters for the pre-specified probability distribution in $\mathbb{R}^M$ which will typically have $M > P$ (P being the dimension of $\mathcal{Y}$). For example in the univariate Gaussian case we have $M = 2$, a parameter for the mean and variance and $P = 1$. To assess the goodness of fit we use a class of loss functions known as scoring rules [Gneiting and Raftery, 2007]. A scoring rule is a function $\mathcal{S} : \mathbb{R}^M \times \mathcal{Y} \to \mathbb{R}$ which measures the accuracy of a predicted distribution. The log-likelihood is an example of such a rule which is what we use here as $\mathcal{S}$. Similar to above $\tilde{\nabla}\mathcal{S}$ is used to refer to the natural gradient of $\mathcal{S}$. The only difference from gradient boosting described in Section 2.2.1 is that we use the natural gradient in place of the ordinary gradient as in 2.2.3.

In summary, after the natural gradient boosting algorithm which is summarized in algorithm 1, we obtain the following conditional distribution:

$$p\left(\mathbf{Y}|\boldsymbol{\theta}(\mathbf{X})\right),$$

where

$$\boldsymbol{\theta}(\mathbf{X}) = \boldsymbol{\theta}^{(0)} - \sum_{b=1}^{B} \eta\beta_b \mathbf{h}(\mathbf{X}, \mathbf{a}_b). \tag{2.2.7}$$

We initialize $\boldsymbol{\theta}^{(0)}$ as the maximum likelihood estimate to all the training data $\mathbf{Y}$, hence it does not depend on $\mathbf{X}$. Note we also use a learning rate parameter $\eta > 0$ which is typically chosen to be lower than 1. A low value for η is often found to provide

superior results at the cost of requiring a higher value for B, resulting in the model fitting more slowly. In the algorithm, we refer to $\boldsymbol{\theta}_i^{(b)}$ as the parameter estimate for $\mathbf{X}_i$ at stage b. Note we do not include early stopping in the algorithm as written here but we do use it in the application of the algorithm.

Algorithm 1 NGBoost for probabilistic prediction.

Data: Dataset $\mathcal{D} = \{\mathbf{X}_i, \mathbf{Y}_i\}_{i=1}^N$.

Input: Boosting iterations B, Learning rate η, Probability distribution with parameter $\boldsymbol{\theta}$, Scoring rule $\mathcal{S}$, Base learner $\mathbf{h}$.

Output: Scalings and base learner parameters $\{\beta_b, \mathbf{a}_b\}_{b=1}^B$.

$\boldsymbol{\theta}_i^{(0)} \leftarrow \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^N \mathcal{S}(\boldsymbol{\theta}, \mathbf{Y}_i)$ {initialize to marginal for all i}

for $b \leftarrow 1, \ldots, B$ **do**

  for $i \leftarrow 1, \ldots, N$ **do**

    for $m \leftarrow 1, \ldots, M$ **do**

      $g_m^{(b)}(\mathbf{X}_i) \leftarrow \tilde{\nabla}\mathcal{S}\left(\boldsymbol{\theta}_i^{(b-1)}, \mathbf{Y}_i\right)_m$

    end

  end

  $\mathbf{a}_b = \arg\min_{\mathbf{a}} \sum_{i=1}^N \sum_{m=1}^M \left(g_m^{(b)}(\mathbf{X}_i) - \mathbf{h}(\mathbf{X}_i, \mathbf{a})_m\right)^2$.

  $\beta_b \leftarrow \arg\min_{\beta} \sum_{i=1}^N \mathcal{S}\left(\boldsymbol{\theta}_i^{(b-1)} - \beta\mathbf{h}(\mathbf{X}_i, \mathbf{a}_b), \mathbf{Y}_i\right)$

  for $i \leftarrow 1, \ldots, N$ **do**

    $\boldsymbol{\theta}_i^{(b)} \leftarrow \boldsymbol{\theta}_i^{(b-1)} - \eta\left(\beta_b\mathbf{h}(\mathbf{X}_i, \mathbf{a}_b)\right)$

  end

end

2.2.3 A Multivariate Extension

The original NGBoost paper Duan et al. [2020] briefly noted that NGBoost could be used to jointly model multivariate outcomes, but did not provide details. Here we show how NGBoost extends to multivariate outcomes and provide a detailed investigation of one useful parametrization, namely the multivariate Gaussian.

In univariate NGBoost ($P = 1,\ Y \in \mathbb{R}$), the predicted distribution is parametrized with $p(Y|\{\boldsymbol{\theta}_m = f_m(\mathbf{X})\})$ where Y is a univariate outcome. For example, $Y|\mathbf{X}$ may be assumed to follow a univariate Gaussian distribution where μ and $\log \sigma$ are taken to be the parameter vector $\boldsymbol{\theta}$ (i.e., the output of NGBoost is $\boldsymbol{\theta}(\mathbf{X}) = (\mu(\mathbf{X}), \log \sigma(\mathbf{X}))$). Therefore to model a *multivariate* outcome, all that is necessary is to specify a parametric distribution that has multivariate support, as we shall now show.

Multivariate Gaussian NGBoost The multivariate Gaussian is a commonly used distribution and a natural choice for many applications. The distribution is a generalization of the univariate Gaussian distribution. The generalization allows us to define a joint distribution on a vector valued function. If we could predict this distribution then we can then investigate the conditional correlations between output dimensions.

In particular the suitability comes from the widespread approach of MSE being used for optimization metrics for various areas including oceanography on the prediction of currents. The standard Gaussian distribution's log likelihood is proportional to MSE when the variance is fixed. Furthermore, ocean currents are often reported in standard axes - longitudinal and latitudinal, but in practice these two coordinates are highly correlated, motivating the choice to model the correlation between these axes.

This is within the scope of the multivariate Gaussian distribution hence our choice.

The rest of this chapter is focused on the development and evaluation of a probabilistic regression algorithm using the multivariate Gaussian. Other multivariate distributions are also trivially accommodated by our framework as long as the natural gradient can be calculated with respect to some parametrization in $\mathbb{R}^M$. We leave the development and testing of alternatives to future work.

The multivariate Gaussian distribution is commonly written in the *moment* parameterization as

$$\mathbf{Y}_i \sim \mathbb{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

where $\boldsymbol{\Sigma}$ is the covariance matrix (a $P \times P$ positive definite matrix), and $\boldsymbol{\mu}$ is the mean vector (a $P \times 1$ column vector). Note that we only consider positive definite matrices for $\boldsymbol{\Sigma}$ to ensure the inverse exists. We write the probability density function as

$$p(\mathbf{Y}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{|\boldsymbol{\Sigma}|^{-\frac{1}{2}}}{(2\pi)^{\frac{P}{2}}} \exp\left[-\frac{(\mathbf{Y}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{Y}_i - \boldsymbol{\mu})}{2}\right]. \tag{2.2.8}$$

We fit the parameters of this distribution conditional on the corresponding training data $\mathbf{X}_i$ such that $p(\mathbf{Y}_i|\mathbf{X}_i) = p(\mathbf{Y}_i|\boldsymbol{\mu}(\mathbf{X}_i), \boldsymbol{\Sigma}(\mathbf{X}_i))$. To perform unconstrained gradient based optimization for any distribution, we must have a parameterization for the multivariate Gaussian distribution where all parameters lie on the real line. The mean vector $\boldsymbol{\mu}$ already satisfies this. However, the covariance matrix does not; it lies in the space of positive definite matrices. We shall model the inverse covariance matrix which is also constrained to be positive definite. We leverage the fact that every positive definite matrix can be factorized using the Cholesky decomposition with

positive entries on the diagonals [Banerjee and Roy, 2014].

We opt to use an upper triangular representation of the square root of the inverse covariance matrix $\mathbf{\Sigma}^{-1} = \mathbf{L}^\top \mathbf{L}$, as used by Williams [1996], where the diagonal is transformed using an exponential to force the diagonal to be positive. As an example, in the two-dimensional case we have

$$\mathbf{L} = \begin{bmatrix} \exp(a_{11}) & a_{12} \\ 0 & \exp(a_{22}) \end{bmatrix}, \tag{2.2.9}$$

which yields the inverse covariance matrix

$$\mathbf{\Sigma}^{-1} = \begin{bmatrix} \exp(2a_{11}) & \exp(a_{11})a_{12} \\ \exp(a_{11})a_{12} & a_{12}^2 + \exp(2a_{22}) \end{bmatrix}.$$

This parameterization for $\mathbf{\Sigma}^{-1}$ ensures that the resulting covariance matrix $\mathbf{\Sigma} = (\mathbf{L}^\top \mathbf{L})^{-1}$ is positive definite for all $a_{ij} \in \mathbb{R}$. Hence, we can fit the multivariate Gaussian in an unconstrained fashion using the parameter vector $\boldsymbol{\theta} = (\mu_1, \mu_2, a_{11}, a_{22}, a_{12})$ as the output in the two-dimensional case. Note that the number of parameters grows quadratically with the dimension of the data. Specifically, the relation between M, the dimension of $\boldsymbol{\theta} \in \mathbb{R}^M$, and P, the dimension of $\mathbf{Y}_i$, is

$$M = \frac{P^2 + 3P}{2}. \tag{2.2.10}$$

For NGBoost using the log-likelihood scoring rule, we require both the gradient and the Fisher Information. The gradient calculations are given in Williams [1996], and the derivations for the Fisher information are given in Appendix A.B. We have used these derivations to add the multivariate Gaussian distribution to the open-source Python package NGBoost as part of this work. Note that the natural gradient is particularly advantageous for multivariate problems such as these. This is because, for

example, there are multiple equivalent parameterizations for the multivariate Gaussian distribution [Sützle and Hrycej, 2005, Salimbeni et al., 2018, Malagò and Pistone, 2015], but the choice of parameterization has been shown to be less important when using natural gradients than with classical gradients (see, e.g. Salimbeni et al. [2018]).

2.3 Simulation

We now demonstrate the effectiveness of our multivariate Gaussian NGBoost algorithm in simulation. Specifically, we show that (a) it outperforms a naive baseline where the target components are modeled independently, (b) it outperforms a state-of-the art neural network approach, and (c) the natural gradient is a key component in the effective training of distributional boosting models in the multivariate setting.

We simulate the data similarly to Williams [1996]. The nature of the simulation tests each algorithm's ability to uncover nonlinearities in each of the distributional parameter's relationship with the input. Specifically, we use a one-dimensional input and two-dimensional output, allowing us to illustrate the fundamental benefits of our approach in even the simplest multivariate extension. The data are simulated as follows:

$$\mathbf{X}_i \overset{\text{IID}}{\sim} \text{Uniform}(0, \pi) \quad i \in \{1, \ldots, N\}$$

$$\mathbf{Y}_i | \mathbf{X}_i \sim \mathbb{N}\left(\begin{bmatrix} \mu_1(\mathbf{X}_i) \\ \\ \mu_2(\mathbf{X}_i) \end{bmatrix}, \begin{bmatrix} \sigma_1^2(\mathbf{X}_i) & \sigma_1(\mathbf{X}_i)\sigma_2(\mathbf{X}_i)\rho(\mathbf{X}_i) \\ \\ \sigma_1(\mathbf{X}_i)\sigma_2(\mathbf{X}_i)\rho(\mathbf{X}_i) & \sigma_2^2(\mathbf{X}_i) \end{bmatrix} \right),$$

with the following functions:

$$\begin{aligned}
\mu_1(x) &= \sin(2.5x)\sin(1.5x) + x, \\
\mu_2(x) &= \cos(3.5x)\cos(0.5x) - x^2, \\
\sigma_1^2(x) &= 0.01 + 0.25[1 - \sin(2.5x)]^2, \\
\sigma_2^2(x) &= 0.01 + 0.25[1 - \cos(3.5x)]^2, \\
\rho(x) &= \sin(2.5x)\cos(0.5x).
\end{aligned} \tag{2.3.1}$$

Simulated data alongside the true parameters are shown in Figure 2.3.1.

We compare multiple methods, all of which predict a multivariate Gaussian. For all boosting models we use decision trees with a max depth of 3. We allow the maximum number of leafs for that depth (8) and allow a leaf to have one data point in it. This depth is sufficient as the underlying functions are univariate. Even a depth 1 decision tree would likely be good enough to learn these univariate function. Specifically, we consider five different comparison methods:

- **NGB:** The method proposed in this chapter: Natural gradient boosting to fit a multivariate Gaussian distribution.
- **Indep NGB:** Independent natural gradient boosting where a univariate Gaussian model is fitted for the two dimensions separately, i.e., $\mathbf{Y}_{i,1} \sim \mathbb{N}(\mu_1, \sigma_2)$, $\mathbf{Y}_{i,2} \sim \mathbb{N}(\mu_2, \sigma_2)$. Early stopping allows the number of trees used to predict each dimension to differ.
- **skGB:** scikit-learn's [Pedregosa et al., 2011] implementation of gradient boosting (skGB) is used as a point prediction approach. To turn the skGB predictions

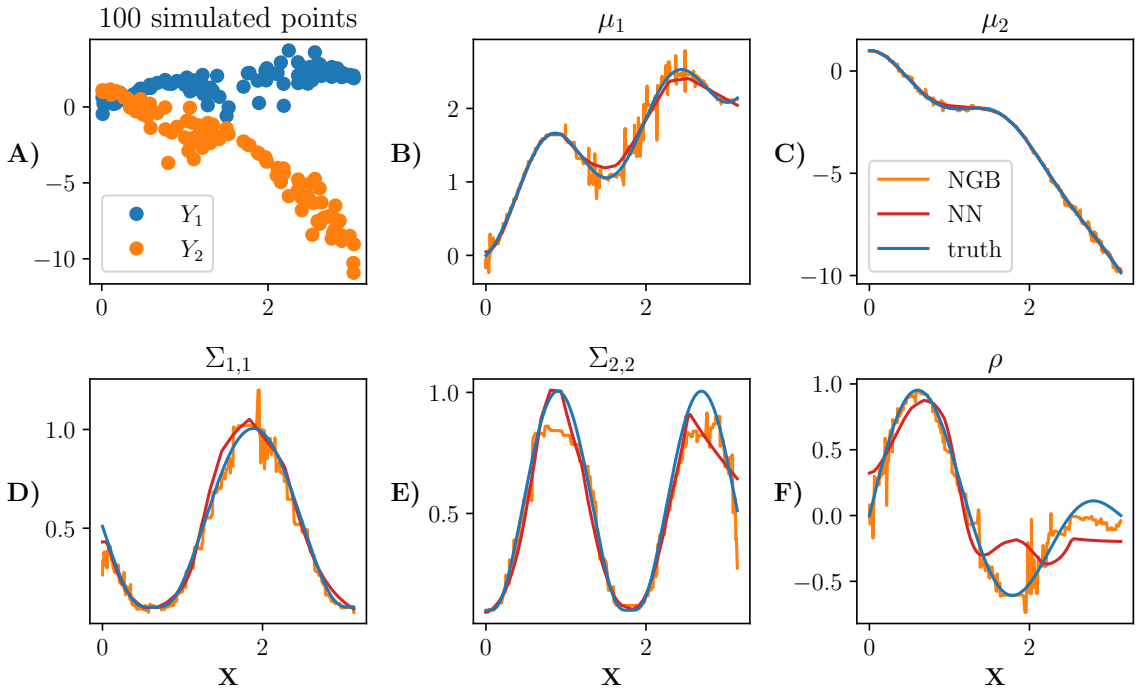

Figure 2.3.1: Simulated data from Equation (2.3.1). A sample of 100 points is shown in A). We plot each parameter in plots B) to F). The true parameters of the distribution are shown in blue, the NGBoost fit is shown in orange, and the neural network fit with one hidden layer (100 neurons in the hidden layer) is shown in red. Both model fits are trained on $N = 5000$ training points.

into a multivariate Gaussian, we estimate a constant diagonal covariance matrix based on the residuals of the training data. For metric computation we assume that each $\mathbf{Y}_i$ follows a multivariate Gaussian with mean from a model fit for each dimension, and constant covariance matrix. Similar to Indep NGB we allow a different number of trees for each dimension.

- **GB:** A multi-parameter gradient boosting algorithm, where the only change from NGB is that the gradients are not multiplied by $\mathcal{I}(\boldsymbol{\theta})^{-1}$.

- **NN:** The neural network approach from [Williams, 1996]. We fit the multivariate Gaussian using tensorflow [Abadi et al., 2016]. More details about the model and the grid search carried out on the network structure are explained in the supplementary information.

Experimental Evaluation To prevent overfitting, we employ early stopping for all methods on the held-out validation set, where we use a patience of 50 for all methods. For all boosting approaches we use the estimators up to the best iteration that was found, and for neural networks we restore the weights back to the best epoch. The base learner for all boosting approaches are run using the default parameters specified in Section 2.2.1. For each N considered in the experiment, the neural network structure with the best log-likelihood metric averaged over all replications is chosen from the grid search. Note that we do not retrain the models after selecting the early stopping epochs/number of learners. We found that doing so gave a large advantage to the boosting based approaches. A learning rate of 0.01 is used for all methods, for NN we use the Adam optimizer [Kingma and Ba, 2015].

We train the the model used the experiment with $N \in \{500, 1000, 3000, 5000, 8000, 10000\}$ with 50 replications for each value of N. For each simulation, N values were simulated as the training dataset. In addition to the N values a further 300 values were simulated as the validation set for early stopping, and another 1000 values were used as the test set. We then use the KL divergence from the true distribution at the test-set locations as the main comparison metric, however we report an extended table of results in Table A.D.1.

The per-model results on the test data points are shown in Table 2.3.1. The average Kullback-Leibler (KL) divergence from the predicted distribution to true distribution is used as a metric. The results show that the NGB method is best for all N despite not being tuned in any way besides early stopping. The KL divergence metric gets close to zero as N grows, showing that the predictions from NGB seems to converging to the true underlying distribution as N increases. NN performs worse than NGB with higher KL divergence for all values of N. The KL divergence metric for Indep NGB seems to be converging to a non-zero quantity as N increases, likely because the target distribution in Equation (2.3.1) cannot be captured by a multivariate normal with a diagonal covariance matrix. skGB has large KL divergences which get worse as N increases, likely because of the homogeneous variance fit. We note that GB performs significantly worse than NGB showing that the natural gradient is necessary to fit the multivariate Gaussian effectively with boosting. Further metrics can be seen in the supplementary information, in particular they show GB fits the mean $\boldsymbol{\mu}$ very poorly, which is then compensated for by a large covariance estimate, explaining the large KL divergence.

N	NGB	Indep NGB	skGB	GB	NN
500	**0.564±0.016**	1.633±0.043	17.194±0.301	126.227±2.579	1.285±0.547
1000	**0.257±0.004**	1.150±0.020	17.963±0.270	114.113±1.622	0.320±0.018
3000	**0.106±0.002**	0.884±0.015	19.609±0.248	97.682±1.397	0.149±0.004
5000	**0.081±0.008**	0.878±0.015	20.308±0.169	90.101±1.291	0.128±0.005
8000	**0.053±0.004**	0.866±0.013	20.614±0.170	79.006±1.168	0.103±0.004
10000	**0.043±0.001**	0.831±0.010	20.554±0.150	74.799±1.191	0.130±0.004

Table 2.3.1: Average KL divergence in the test set (to 3 decimal places) from the predicted distribution to the true distribution as the number of training data points N varies. Standard error estimated from 50 replications reported after ± to 3 decimal places. Lower values are better, the result with the lowest mean is emboldened in each row. An extended table showing additional metrics is shown in the supplementary information.

The modification we made from the simulation in Williams [1996] is that we added x and $-x^2$ terms to $\mu_1(x)$ and $\mu_2(x)$ respectively in Equation (2.3.1). This modification is to highlight where our method excels. We also ran the original simulation without the modification, where the results are given in the supplementary information. As a summary, we note the major differences: (1) the gaps between NGB, GB and NN are smaller, and (2) the NN method does best for $N \in \{500, 1000, 3000\}$, NGB does best for $N \in \{5000, 8000, 10000\}$.

2.4 Ocean Drifters Application

The motivating application of this chapter is to predict two-dimensional oceanographic velocities from satellite data on a large spatial scale. Typically, this is done through physics-inspired parametric models fitted with least-squares or similar metrics, treating the directional errors as independent [Mulet et al., 2021].

2.4.1 Data

Here we introduce the datasets used which shall define $\mathbf{Y}_i \in \mathbb{R}^2$, $\mathbf{X}_i \in \mathbb{R}^9$. The data for all sources is available from 1992 to 2019 inclusive. All data sources are publicly available.

For the model output $\mathbf{Y}_i$, we seek to predict two-dimensional oceanic near-surface velocities as functions of global remotely-sensed climate data. The dataset used to train, validate and test our model comes from the Global Drifter Program, which contains transmitted locations of freely drifting buoys as they drift in the ocean and measure near-surface ocean flow. The quality controlled 6-hourly product is used [Lumpkin and Centurioni, 2020] to construct the velocity observations. We drop observations which have a high location noise estimate, and we apply a low-pass filter the velocities at 1.5 times the inertial period (a timescale determined by the Coriolis effect), following previous similar works [Laurindo et al., 2017]. We only use data from buoys which still have a drogue (sea anchor) attached and thus more accurately follow ocean near-surface flow. We use the inferred two-dimensional velocities of these drifting buoys as our outputs $\mathbf{Y}_i$.

The longitude-latitude locations of these observations and the time of year (percentage of 365) are used as three of the nine features in $\mathbf{X}_i$ to account for spatial and seasonal effects. For the remaining six features in $\mathbf{X}_i$, we use two-dimensional longitude-latitude measurements of *geostrophic velocity* (ms^{-1}), *surface wind stress* (Pa) and *wind speed* (ms^{-1}). These variables are used as they jointly capture geophysical effects known as geostrophic currents, Ekman currents, and wind forcing, which are known to drive oceanic near-surface velocities. We obtain geostrophic velocity and wind measurements from data products Thematic Assembly Centers [2020a] and Thematic Assembly Centers [2020b] respectively, the data products are interpolated to the longitude-latitude locations of interest. We further preprocess the data as explained in Appendix A.A.

To allow us to run multiple model fits, we subset the data to only include data points which are spatially located in the North Atlantic Ocean as defined by IHO marine regions Flanders Marine Institute [2018] and between 83°W and 40°W longitude. We use a temporal gridding of the buoy data of one day. This results in 414697 observations for each input and output variable in the combined data set used for training, validating and testing our probabilistic regression model.

2.4.2 Metrics

We cannot use KL divergence as in the simulation of Section 2.3 because the true distribution of these data is unknown. Therefore we use a series of performance metrics that diagnose model fit listed here.

Negative Log-Likelihood (NLL): All methods are effectively minimizing the

negative log-likelihood; therefore we use negative log-likelihood as one of our metrics:

$$1/N \sum_i \log p(\mathbf{Y}_i|\mathbf{X}_i).$$

RMSE: To compare the point prediction performance of the models we also report an average root mean squared error (RMSE):

$$\left[1/(NP)\sum_{i=1}^{N}\sum_{j=1}^{P}(\mathbf{Y}_{i,j}-\hat{\mathbf{Y}}_{i,j})^2\right]^{1/2}, \text{ where } \hat{\mathbf{Y}}_i = \mathbb{E}[\mathbf{Y}|\mathbf{X}_i].$$

Region Coverage and Area: As this chapter focuses on a measure of probabilistic prediction, we also report metrics related to the prediction region. The prediction region is the multivariate generalization of the one-dimensional prediction interval. The two related summaries of interest are: (1) the percentage of $\mathbf{Y}_i$ covered by the $\alpha\%$ prediction region, and (2) the area of the prediction region.

A $\alpha\%$ prediction region can be defined for the multivariate Gaussian distribution as the set of values $\mathbf{Y}$ which satisfy the following inequality:

$$(\mathbf{Y}-\boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{Y}-\boldsymbol{\mu})^\top \leq \chi^2_{P,\alpha}, \tag{2.4.1}$$

where $\chi^2_{P,\alpha}$ is the quantile function of the χ^2 distribution with P degrees of freedom evaluated at $\alpha\%$. This prediction region forms a hyper-ellipse which has an area given by

$$\frac{(2\pi)^{P/2}}{P\Gamma(\frac{P}{2})}(\chi^2_{P,\alpha})^{P/2}|\boldsymbol{\Sigma}|^{1/2}. \tag{2.4.2}$$

We report the percentage of data points which satisfy Equation (2.4.1), and we report the average area of the prediction region over all points in the test set.

Metric	NGB	Indep NGB	skGB	GB	NN
NLL	**7.73±0.02**	7.74±0.02	8.17±0.02	8.79 ± 0.01	7.81±0.02
RMSE $cm\ s^{-1}$	14.53±0.14	14.45±0.12	**14.31±0.13**	24.14±0.36	16.63 ± 0.19
90% PR cov	0.87±0.00	0.87±0.00	0.89±0.00	0.94±0.00	0.89±0.00
90% PR area $cm^2\ s^{-2}$	2482 ± 51	2568±41	2714±8	8070±22	3670±75

Table 2.4.1: Average test set metrics defined in Section 2.4.2. Standard error estimated from 10 replications reported after ±. PR stands for prediction region. Coverage has been shortened to cov. All numbers rounded to 2 decimal places, aside from the 90% PR Area row which is rounded to the nearest integer. Lower values are better for NLL and RMSE, best value is emboldened in both rows.

2.4.3 Results

We compare the same five models considered in Section 2.3. To compare the models we randomly split the dataset, keeping each individual buoy record within the same set. We put 10% of records into the test set, 9% of the records into the validation set, and 81% into the training set. The model is fitted to the training set with access to the validation set for early stopping, and then the metrics from Section 2.4.2 are evaluated on the test set. This procedure is repeated 10 times and the results are shown in Table 2.4.1.

We note that in practice, selecting the number of boosting iterations then training the tree-based models with the training set and validation set would likely result in better performance. However, we found this approach worked poorly for the neural

network so, we do not retrain in an attempt to keep the comparison fair, that is that all models are trained using 81% of the data and early stopping uses the 9%. The metrics are reported on the remaining 10%.

For this example we also use a grid search for all of the boosting approaches, in addition to the neural network approach. We found that with the default parameters used in the simulation[1] the boosting methods generally did not ever early stop. They would reach the maximum we set of 1000 estimators. This behavior suggests that the base learner may not be flexible enough. For each boosting method, a grid search is carried out over number of leafs, and minimum data in leafs, as outlined in the supplementary material. The max-depth is fixed at 15. The best hyper-parameters from the grid search for each method are selected by evaluating the test-set negative log-likelihood. As in the simulation we use a patience of 50 for early stopping.

The aggregated results of the model fits are shown in Table 2.4.1. NGB and Indep NGB perform very similarly in terms of NLL, RMSE and 90% PR coverage in this example. However, NGB provides a smaller 90% PR area, which is expected as correlation will reduce $|\mathbf{\Sigma}|$ in Equation (2.4.2). To highlight the differences further, we show the spatial differences in negative log-likelihood between these two methods in Figure 2.4.1B). In Figures 2.4.1C) and D) we show the averaged held out spatial predictions for ρ and μ from the NGB model. The results shown in B) and C) suggest that using a combination of these approaches may be suitable to this application. For example, we could use the NGB model in geographic areas with high anticipated

[1]Default parameters being a max depth of 3, minimum samples per split 2 and minimum samples per leaf 1

correlation; otherwise we can use Indep NGB as it will likely do better.

In Table 2.4.1 we see that the NN and GB approaches do poorly overall. Both methods have large root mean-squared error, which is compensated for by a larger prediction region on average, as can be concluded from the large average PR area. This behavior agrees with the univariate example given in Figure 4 of the original NGBoost paper [Duan et al., 2020], and the behavior of GB in the simulation of Section 2.3.

One of the novelties of this work is that we model the dependence between the two output dimensions. Any region where ρ is close to zero in Figure 2.4.1C) likely means that this additional modeling of the dependence will not make a large difference to the predictions. In contrast, any region where the absolute value of ρ is large implies that the multivariate NGB predictions will be significantly different. In particular multivariate NGB predicts large values of $|\rho|$ around the South Equatorial and Gulf Stream currents.

In Figure 2.4.2 we show a sample trajectory on the Gulf Stream alongside the predicted means and 70% prediction region from both indepedent NGB and multivariate NGB. In Figure 2.4.2A) the main factor to note is that the elliptical prediction regions take various orientations over the trajectory, most of which have the major axis aligned with the flow. In contrast the predictions from independent NGB in Figure 2.4.2B) show the ellipses' major axes are always either aligned with the longitudinal or latitudinal axis, as must be the case. This difference is particularly evident around longitudes $78°W$ to $74°W$ where independent NGB's predictions show a larger minor axis in the ellipses in comparison to multivariate NGBoost predictions. This is likely

the key reason why multivariate NGB does better on average around the Gulf Stream as seen in Figure 2.4.1B).

2.5 Conclusions

This chapter has demonstrated the accuracy of our NGBoost method when focusing on bivariate outcomes with a multivariate Gaussian distribution. The derivations of the natural gradient and implementation in NGBoost are supplied for any dimensionality, but empirical proof of performance in higher dimensions is left to future work. Due to the quadratic relationship between P and the number of parameters used to parameterize the multivariate Gaussian, the complexity of the learning algorithm greatly increases with larger values of P. Investigating a reduced rank form of the covariance matrix may be of interest for higher P to reduce the number of parameters that need to be learned. We also leave the development and testing of alternative multivariate distributions to future work. The modular nature of our implementation makes it easy for users to experiment and add their own multivariate distributions as long as they can supply the relevant gradient and Riemannian metric functions.

The difference between multivariate NGBoost (NGB) and independent NGBoost (Indep NGB) was overall relatively small in our real-world application across the entire spatial domain studied, but NGB showed significant improvements around major currents such as the Gulf Stream where directional currents (that are not aligned with the x/y axis) are most prominent. This improvement was corroborated in simulations where NGB performed relatively much better than Indep NGB overall. Generally, one

should anticipate NGB to excel in cases with high correlation between the outcomes, whereas a method assuming independence should suffice when that assumption is warranted. Deep learning approaches are warranted in cases where a neural network naturally handles the input space such as images, speech or text.

Although this approach is shown to work sensibly and captures correlation between the longitudinal and latitudinal directions, the model which we have fit is not incredibly interpretable. The final model allowed trees with depth up to 15, ideally we could use a much simpler model such as depth 1 or 2 decision trees. In the future work of this thesis, Section 5.1.1, we propose a varying coefficient model which takes into account the relationship between the covariates. We believe in this case a much more intuitive and explainable model could be learned.

Final Remarks In this chapter we have proposed a technique for performing multivariate probabilistic regression with natural gradient boosting. We have implemented software in the NGBoost Python package which makes joint probabilistic regression easy to do with just a few lines of code and little to no tuning. Our simulation and case study show that multivariate NGBoost meets or exceeds the performance of existing methods for multivariate probabilistic regression.

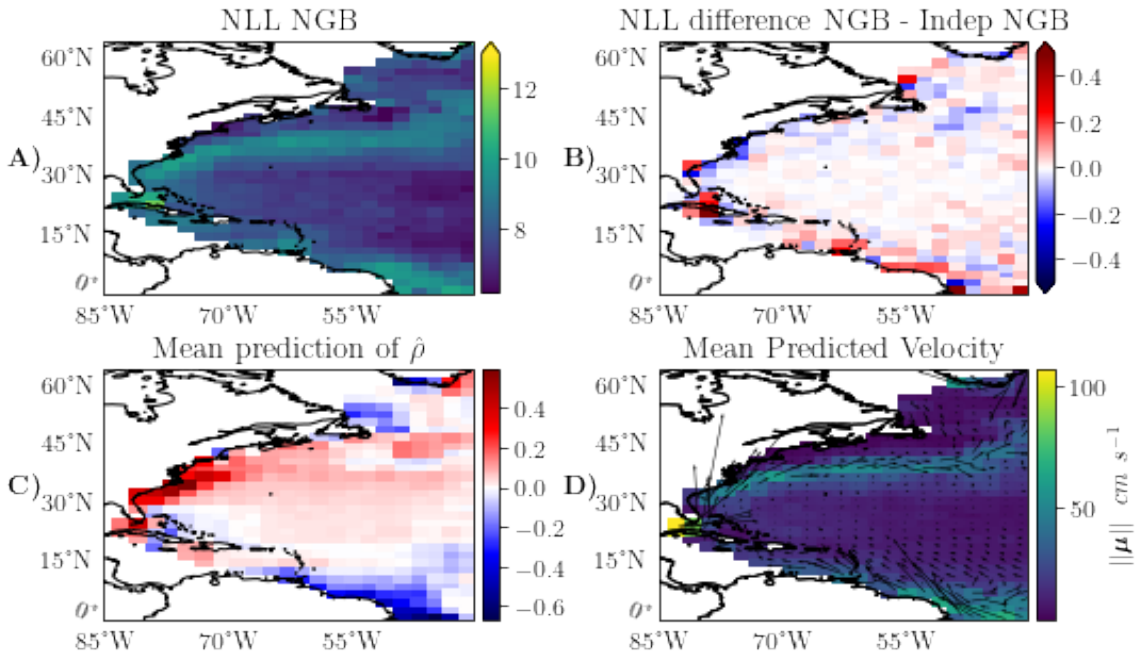

Figure 2.4.1: Summaries of test set results within $2° \times 2°$ latitude-longitude bins for the North Atlantic Ocean application. A) shows the spatial distribution of the negative log-likelihood for NGB, B) shows the difference between the negative log-likelihood spatially between NGB and Indep NGB; negative values (blue) implying NGB is better than Indep NGB (with vice versa in red). C) shows the average prediction of ρ where $\rho = \boldsymbol{\Sigma}_{0,1}/\sqrt{\boldsymbol{\Sigma}_{0,0}\boldsymbol{\Sigma}_{1,1}}$ is extracted from the predicted covariance matrix in the held out set from NGB. D) shows the mean currents estimated by NGB. All major ocean features are captured by the model [Lumpkin and Johnson, 2013]

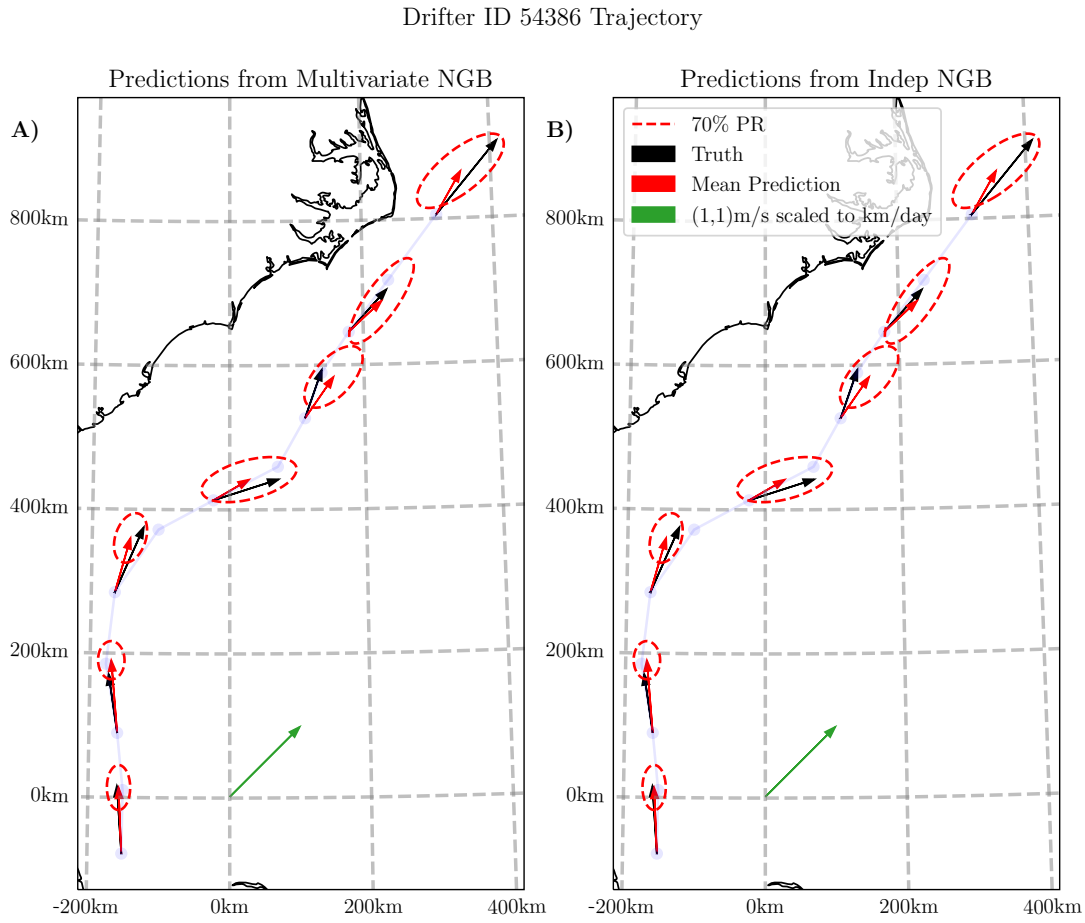

Figure 2.4.2: Plots showing the predictions from both multivariate and independent NGB for the first 12 days of drifter ID 54386 starting from the 23rd of September 2005. Predictions are plotted every 2 days for visualization purposes. The model used for prediction did not see this trajectory when trained. The faded blue line shows the trajectory of the drifter with a point plotted every day. The 70% prediction region is the boundary from Equation (2.4.1). The lines and confidence regions are scaled from m/s to be km/day. Conversion to easting-northing computed using a Transverse Mercator with central longitude-latitude at $(-78°, 28°)$.

Chapter 3

Most Likely Path

3.1 Introduction

The Lagrangian study of transport and mixing in the ocean is of fundamental interest to ocean modellers [van Sebille et al., 2018, 2009, LaCasce, 2008]. In particular, the analysis of data obtained from Lagrangian drifting objects greatly contribute to our knowledge of ocean circulation, e.g. through analysing the accuracy of numerical and stochastic models [Huntley et al., 2011, Sykulski et al., 2016], or the use of drifter data to better understand various pathways and where to search for marine debris [Miron et al., 2019, van Sebille et al., 2012, McAdam and van Sebille, 2018].

Meehl [1982] used shipdrift data to create a surface velocity data set on a $5^\circ \times 5^\circ$ grid. These velocities were used to simulate the Lagrangian drift of floating objects in Wakata and Sugimori [1990]. More recent works focus on using drifting buoys to derive Lagrangian models to discover areas where floating debris tends to end up [van Sebille, 2014, van Sebille et al., 2012, Maximenko et al., 2012]. Advances in technology

have resulted in much better data quality, which now permits the use of more detailed methodology. The newer models provide densities of where debris ends up on grid scales as small as $0.5^\circ \times 0.5^\circ$.

A tool which predicts travel times is of practical use in many fields. For example in ecological studies of marine species, genetic measurements are taken at various locations in the ocean [Watson, 2018]. Euclidean distance is often used as a measure of separability and isolation-by-distance [Becking et al., 2006, Ellingsen and Gray, 2002] to find correlations with diversity metrics or genetic differentiation between communities or populations of organisms. Due to various currents and land barriers, we expect Euclidean distance to often be a poor measure of how 'distant' or dissimilar communities or populations sampled in two locations are. The method proposed in this work would use the estimated travel times to supply a matrix containing a *Lagrangian distance* measure between all pairs of locations. This matrix can then be contrasted with a pairwise genetic distance matrix between these locations and will yield new insights. In many instances the Lagrangian distance matrix will be more correlated with genetic relatedness than a Euclidean distance matrix. This observation was already made in the Mediterranean Sea when studying plankton [Berline et al., 2014], and off the coast of California for a species of sea snail [White et al., 2010]. Both of the works by Berline et al. [2014] and White et al. [2010] rely on simulating trajectories from detailed ocean current data sets to estimate the Lagrangian distance. Such approaches do not scale globally and rely on simulated trajectories from currents rather than real observations.

In Figure 3.1.1, we show seven locations plotted on a map with ocean currents.

We use these locations as a proof-of-concept example throughout this chapter. The exact coordinates are given in Table 3.5.1. The aim is to introduce and motivate a method which provides an estimate as to how long it would take to drift between any two of these locations. For example, the travel time from location 2 to location 3 in the South Atlantic Ocean should be smaller than the return journey due to the Brazil current. We choose to include locations in both the North and South Atlantic as we wish to demonstrate that the method successfully finds pathways linking points which are extremely far apart.

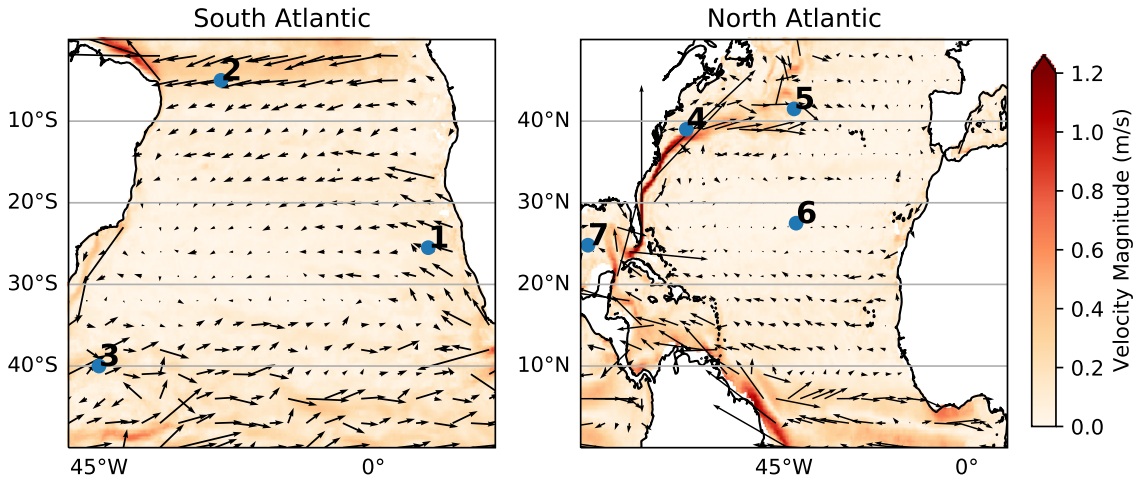

Figure 3.1.1: Locations of interest from Table 3.5.1. Annual mean values of the near-surface currents derived from drifter velocities [Laurindo et al., 2017] are plotted. Arrows drawn on a $3° \times 3°$ grid to show current direction.

3.1.1 Comparison with Related Works

In this section we give a brief overview of techniques that have used the Global Drifter Program to achieve a similar or related task. The work by Rypina et al. [2017] proposes

a statistical approach for obtaining travel times. A source area is defined such that at least 100 drifters pass through the source area. The method focuses on obtaining a spatial probability map and a mean travel time for every $1^\circ \times 1^\circ$ bin outside of the source area. This method successfully combines many trajectories, however for multiple locations one would have to decide on a varying grid box for each location of interest. Such a grid box must be manually chosen by the practitioner meaning that the method does not scale well with multiple locations. Rypina et al. [2017] also focus on estimating a mean travel time, where our method provides a travel time associated with the most likely path, and is hence more akin to estimating a mode or median travel time.

The method by van Sebille et al. [2011], which proposes the use of Monte Carlo Super Trajectories (MCST), could naturally be used to estimate travel times. This method simulates new trajectories as sequences of unique grid indices along with corresponding travel time estimates for each part of that journey. The method is purely data driven i.e. they only use real trajectories to fit the model. The travel time and pathway we supply here should be similar to the most likely MCST to occur between the two points. The advantage of our methodology is that we do not base the analysis on a simulation, such that the results from the method described in Section 3.3 are not subject to any randomness due to simulation.

Various other works have made attempts to compute Lagrangian based distances. For example, Berline et al. [2014] used numerically simulated trajectories to estimate *Mean Connection Times* within the Mediterranean Sea. Smith et al. [2018] used MCST to estimate various statistics of how seagrass fragments could drift from the

South East coast of Australia to Chile. Specifically, Smith et al. [2018] simulated 10 million MCST starting from the SE coast of Australia and only 264 (0.00264%) of the simulated trajectories were found to travel roughly to the Chilean coast.

The approach by Jönsson and Watson [2016] uses simulated drifter data to construct connectivity matrices between locations in the ocean. As the matrix is sparse, Dijkstra's algorithm is used to connect arbitrarily distant locations in the ocean to measure Lagrangian distance. Although this method may at first glance bare similarities with our method (specifically in the use of Dijkstra's algorithm), there are in fact many differences. First of all, the method uses simulated trajectories whereas we use real drifter trajectories. Secondly, Dijkstra's algorithm is performed by Jönsson and Watson [2016] on the *connectivity* matrix (which finds minimum connection times between locations), whereas our approach uses Dijkstra's algorithm on the *transition* matrix which describes a probabilistic framework for drifter movement. We found the latter approach to perform much better with real data. Finally, we cannot directly implement the approach described in Jönsson and Watson [2016] as only connectivity values higher than one year are used by the algorithm. For real data such a step would result in a very sparse connectivity matrix making the method infeasible. An initial analysis we conducted using similar methodology achieved poor results.

There are a variety of works which use Markov transition matrices for different aims to this work. Ser-Giacomi et al. [2015b] and Miron et al. [2019] look at probable paths, where both of these works find a path going between two points in a certain number of days using a dynamic program. Froyland et al. [2014] and Miron et al. [2017] study ocean dynamics by analysing eigenvalues of the transition matrix. Other

methods in the literature include characterizing dispersion and mixing [Ser-Giacomi et al., 2015a], identification coherent regions [Froyland et al., 2007, Ser-Giacomi et al., 2015a], forward integration of tracers [van Sebille et al., 2012, Maximenko et al., 2012], and guiding drifter deployments [Lumpkin et al., 2016]. We differ from these works as we ultimately aim to find travel times, as well as pathways, between multiple fixed locations.

Our proposed algorithm for computing travel times and pathways will also use the aforementioned Markov transition matrix approach. Our key novelty is that we build on this conceptual approach by implementing and demonstrating the benefits of using the (H3) spatial indexing system for discretization, and by supplying uncertainty quantification guidelines by applying grid rotations and data bootstrapping. The steps outlined in Algorithm 2 are individually known across disparate literature, however, this is the first work to our knowledge that effectively combines these steps to solve the problem of interest. We provide numerous examples to show how our methodology robustly outperforms state-of-the-art alternative approaches. In addition, we supply freely-available software in the form of a Python package, of which all parameters in the model can easily be customized to suit the needs of the practitioner.

In summary, the novel contributions of this work are: *a)* the combination of the steps in Section 3.3 to form a computationally-efficient algorithm which applies directly to transition matrices to find most likely paths and travel times simultaneously, *b)* computation of uncertainty from discretization error and data sampling (Section 3.4), and *c)* the demonstration of the method showing it successfully obtains robust measures of connectivity between both very distant and closely located points

(Section 3.5). The key outcome is that we obtain oceanographic travel times and most likely paths requiring no simulated trajectories.

We believe our method is preferable to Rypina et al. [2017] as we do not require custom treatment to different source areas. Jönsson and Watson [2016] requires the simulation of many very long and expensive to compute trajectories which obtain spurious results on real data. Using MCST's as in Smith et al. [2018] relies on simulation. The estimation of a full pairwise travel time matrix of the locations in Table 3.5.1 requires 42 travel time estimations. With MCSTs this would likely require the simulation of millions of trajectories and manual analysis of each location pair. Our method, in contrast, can produce such a travel time matrix in a matter of seconds given that the transition matrix needs to be estimated just once *a priori*. In a similar manner, global travel time maps can be made in a matter of minutes, such as those that we will be showing in Section 3.5.

3.2 Background and Notation

3.2.1 Global Drifter Program

The Global Drifter Program (GDP) is a database managed by the National Oceanographic and Atmospheric Administraction (NOAA) [Lumpkin and Centurioni, 2020, Lumpkin and Pazos, 2007]. This data set contains over 20,000 free-floating buoys temporally spanning from February 15, 1979 through to the current day. These buoys are referred to as *drifters*. The drifter design comprises of a sub-surface float and a drogue sock. Often this drogue sock detaches. We refer to the drifters which have

lost their drogue sock as non-drogued drifters, and drogued for those which still have the drogue attached.

Here we use the drifter data recorded up to July, 2020. We use data which has been recorded from drogued drifters only. This results in a total of 23461 drifters being used, where the spatial distribution of observations is shown in Figure 3.2.1. Only using drogued drifters is not a restriction, it would be straightforward to simply use the data from non-drogued drifters if a practitioner was interested in a species or object which experiences a high wind forcing, or a combination of both if it is believed that the species followed a mixture of near surface and wind-forced currents. The dataset is quality controlled and interpolated to six hourly intervals using the methodology from Hansen and Poulain [1996]. These interpolated values do contain some noise due to both satellite error and interpolation, however, the magnitude of this noise is negligible in comparison to the size of grid we use in Section 3.3. Hence, we ignore this noise and treat the interpolated values as observations. For the same reason we note that the interpolation method used is not important here, instead of the six hourly product we could use the hourly product produced by methodology proposed by Elipot et al. [2016], or drifter data smoothed by splines as proposed by Early and Sykulski [2020].

The value of using the Global Drifter Program is we obtain a true model-free representation of the ocean. All phenomena which act on the drifters are accounted for in the data set. The other common approach is to first obtain an estimate of the underlying velocity field, then simulate thousands of trajectories using the velocity field. While this simulation approach is often satisfactory in some applications, the

models generally do not agree completely with the actual observations.

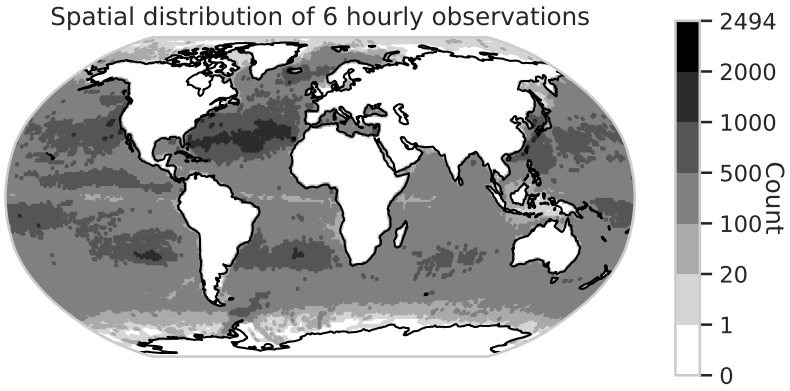

Figure 3.2.1: Number of observations from the Global Drifter Program in each $1^\circ \times 1^\circ$ longitude-latitude box.

3.2.2 Notation

Here we use x°, y° to be a geographic coordinate corresponding to latitude and longitude respectively. We refer to the longitude-latitude grid system using the notation $x^\circ \times y^\circ$, which means each grid box goes x° along the longitude axis and y° along the latitude axis. We use bold font for any data which is in longitude-latitude pairs; i.e $\mathbf{r} = (r_{lon}, r_{lat})$, and non-bold text for either a grid index or a single number. We use $\mathcal{S}$ to denote the set of all possible grid indices. A full table of notation is given in Appendix B.B.

3.2.3 Capturing Drifter Motion

We define the drifter's probability density function as

$$P(\mathbf{r}_1, t|\mathbf{r_0}, t_0)$$

where the drifter started at $\mathbf{r_0} \in \mathbb{R}^2$ at time t_0 and moved to position $\mathbf{r}_1 \in \mathbb{R}^2$ at time t, where $\mathbf{r}_0$ and $\mathbf{r}_1$ are longitude-latitude pairs. In the absence of a model, this probability density cannot be estimated continuously from data alone. Therefore, we follow previous works which spatially discretize the problem [Maximenko et al., 2012, van Sebille et al., 2011, Miron et al., 2019, Rypina et al., 2017, Lumpkin et al., 2016]. Instead of considering $\mathbf{r}_0 \in R^2$, we consider $r_0 \in \mathcal{S}$ where $\mathcal{S}$ is some set of states which correspond to a polygon in space; we will define how these are formed in Section 3.3.2. Often these states are simply $1^\circ \times 1^\circ$ degree boxes (e.g. as used in Figure 3.2.1). As in Maximenko et al. [2012], we assume that the process driving the drifter's movement is temporally stationary. That is:

$$P(r_1, t+k|r_0, t_0+k) = P(r_1, t|r_0, t_0), \quad r_0, r_1 \in \mathcal{S},$$

for any constant time increment k, i.e. the probability of going from r_0 to r_1 depends only on the time increment. The probability does not depend on the start or finish time.

Furthermore, given that we are using data which is observed at regular and discrete times, we shall only consider discrete values of time. Let $\mathbf{s} = \{s_0, s_1, s_2, \ldots, s_n\}$ be a sequence of locations equally spaced in time where each entry s_i can take the value of anything within $\mathcal{S}$. For time $0 \leq i \leq n-1$ define the probability $p(s_{i+1} = q|s_i = k)$ as the probability that the position at time $i+1$ is q given that the state at time i was k where $q, k \in \mathcal{S}$.

In some cases a location fix is missing from the dataset, usually due to a lack of location fixes or drifter malfunction. When this occurs we split the trajectory into

two at the point missing data occurs. For all practical purposes we then treat these as two independent trajectories.

A Lagrangian decorrelation time causes the drifter to 'forget' its history [LaCasce, 2008]. We aim to choose a constant time difference which is globally higher than the Lagrangian decorrelation time. The reasoning behind using this time is that if we consider a sequence of observations, which are at least the Lagrangian decorrelation time apart then the following Markov property is satisfied:

$$\begin{aligned} &p(s_{i+1} = q_{i+1} | s_i = q_i, s_{i-1} = q_{i-1}, \cdots, s_0 = q_0) \\ =&p(s_{i+1} = q_{i+1} | s_i = q_i), \end{aligned} \tag{3.2.1}$$

where q_i is just some fixed state and s_i is the random process. In other words, the Markov property states that probability of transition to state s_{i+1} is independent of all the past states at times $i-1$ and earlier, given the state at time i is known. In this case the physical time difference between s_{i+1} and s_i is a constant. This constant time difference being larger than the chosen Lagrangian decorrelation time validates the use of the Markov assumption.

For the rest of this chapter we assume that the time between discrete time observations is equal to $\mathcal{T}_L$. We call this quantity the Lagrangian cut off time. Setting $\mathcal{T}_L$ higher than the decorrelation time allows us to use the Markov property from Equation (3.2.1) freely. In so doing, alongside the simplification of discretizing locations, this allows the problem to be treated as a discrete time Markov chain. Here we fix $\mathcal{T}_L = 5$ days as this matches previous similar works [Maximenko et al., 2012, Miron et al., 2019]. The estimated decorrelation time for the majority of the surface of the

Ocean is likely to be lower than 5 days (e.g. see Zhurbas and Oh [2004] for the Pacific and Lumpkin et al. [2002] for regions in the Atlantic). In Section B.F we conduct a sensitivity analysis to show our results are not overly sensitive to the choice of $\mathcal{T}_L$ as long as $\mathcal{T}_L > 2$ days.

Ideally, we could use different values of $\mathcal{T}_L$ for different regions, however, using the discrete-time discrete-space models in this chapter, require an equal spacing between points. Choosing $\mathcal{T}_L = 5$ is conservative for all areas of the ocean. We conduct a sensitivity analysis around this in Appendix B.F.

3.3 Method for Computing the Most Likely Path and Travel Time

Maximenko et al. [2012] and van Sebille et al. [2012] focus on the use of a transition matrix estimated from drifters to discover points where drifters are likely to end up. In this section we build on such an approach by providing a method to take such a matrix and provide an ocean pathway and travel time.

In Section 3.3.1, we explain in detail how the transition matrix is formed. As a grid system is needed to form the discretization of data we introduce our chosen system in Section 3.3.2. Then in Section 3.3.3, we describe how we estimate the most likely path of a drifter to have taken. Finally, in Section 3.3.4 we explain how we turn the most likely path and transition matrix into an estimate of travel time. We give a summary of how this articulates in the pseudo-code in Algorithm 2.

3.3.1 Transition Matrix

The location of a drifter at any given time is a continuous vector in $\mathbb{R}^2$, the longitude and latitude of the point. We define an injective map which maps this continuous process onto a discrete set of states which are indexed by integers in $\mathcal{S}$. We define the map as follows:

$$f : \mathbb{R}^2 \to \mathcal{S}. \tag{3.3.1}$$

We aim to make a Markov transition matrix T of size n_{states} rows and columns, where $T_{s,q}$ denotes, the probability of moving from s to q in one time step. Similarly to the approach of Maximenko et al. [2012], we form our transition matrix using a gap method. In each drifter trajectory we only consider observations as a pair of points $\mathcal{T}_L$ days apart. When using this method for other applications we advise using $\mathcal{T}_L$ to be higher than the decorrelation time of velocity to justify the Markov assumption.

Consider a trajectory as a sequence of positions $\mathbf{y}_j = \{\mathbf{y}_{i,j}\}_{i=1}^{n_j}$ where j is the j^{th} out of N trajectories, n_j is the number of location observations in the trajectory, and $\mathbf{y}_{i,j} \in \mathbb{R}^2$ are the longitude-latitude positions. First, we map each trajectory into observed discrete states. We will denote these states as follows,

$$g_{i,j} = f(\mathbf{y}_{i,j}). \tag{3.3.2}$$

For each $s, p \in \mathcal{S}$ we estimate the relevant entry of our transition matrix T through using the following empirical estimate:

$$T_{s,p} = \frac{\sum_{j=1}^{N} \sum_{i=1}^{n_j - 4\mathcal{T}_L} \mathbb{I}[g_{i+4\mathcal{T}_L,j} = p]\mathbb{I}(g_{i,j} = s)}{\sum_{j=1}^{N} \sum_{i=1}^{n_j - 4\mathcal{T}_L} \mathbb{I}[g_{i,j} = s]}, \tag{3.3.3}$$

Where $\mathbb{I}$ is the indicator function, such that it takes the value 1 if the statement inside it is true, and zero otherwise. Note that we take gaps of $4\mathcal{T}_L$ as observations are every 6 hours in the GDP application and $\mathcal{T}_L$ is in days. The estimation of the transition matrix, using the discretization of trajectories in Equation (3.3.2), in combination with Equation (3.3.3), is commonly referred to as Ulam's approach [Ulam, 1960]. We expect that states in $\mathcal{S}$ which are not spatially close will have non-zero entries such that the matrix T will be very sparse, but this is not a problem for the methodology to work over large distances as we shall see.

3.3.2 Spatial Indexing

Clearly the resulting transition matrix described in Section 3.3.1 strongly depends on the choice of grid function in Equation (3.3.1). Most previous works [van Sebille et al., 2012, Maximenko et al., 2012, Rypina et al., 2017, McAdam and van Sebille, 2018, Miron et al., 2019] use longitude-latitude based square grids where all grid boxes typically vary between $0.5^\circ \times 0.5^\circ$ and $1^\circ \times 1^\circ$. A $1^\circ \times 1^\circ$ grid cell around the equatorial region will be approximately equal area to a 111.2km $\times$ 111.2km square box. However, if we take such a grid above 60° latitude, e.g. the Norwegian sea, the grid cell will be approximately equal area to a 55.6km $\times$ 111.2km square box.

There are a few other choices which we argue are more suitable for tracking moving data on the surface of the Earth. Typically three types of grids exist for tessellating the globe: triangles, squares, or a mixture of hexagons and pentagons. Here we choose to use hexagons and pentagons as they have the desirable property that every neighbouring shape shares precisely two vertices and an edge. This is different to say a

square grid where only side-by-side neighbours share two vertices and an edge, whereas diagonal neighbours share only a vertex. This equivalence of neighbors property for hexagons and pentagons is clearly desirable for the tracking of objects as this will result in a smoother transition matrix.

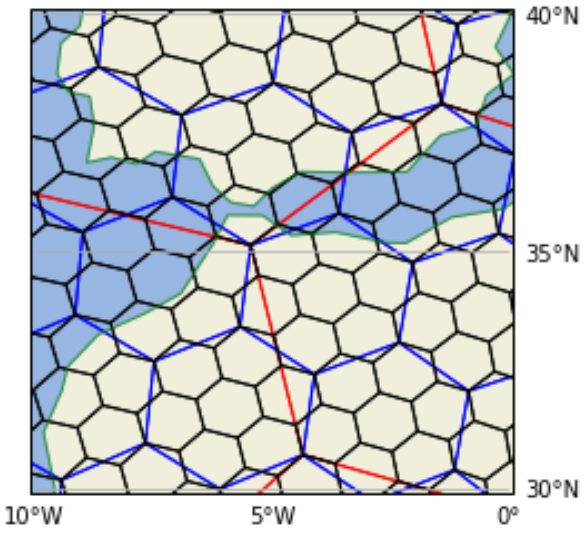

Figure 3.3.1: A small area around the Strait of Gibraltar which is tessellated using the H3 spatial index. We show resolutions 1, 2 and 3 in red, blue and black respectively. Black is the resolution used in this work.

We specifically use the grid system called *H3* by UBER [UBER, 2019]. This system divides the globe such that any longitude and latitude coordinate is mapped to a unique hexagon or pentagon. This shape will have a unique *geohash* which we can use to keep track of grid index. The benefit of using such a spatial indexing system is that attention is paid to ensuring that each hexagon is approximately equal area. We use the *resolution 3* index where each hexagon has an average area of $12,392\text{km}^2$. A square box of size $111.32\text{km} \times 111.32\text{km}$ has roughly the same area

as this which is very similar to the size of a $1^\circ \times 1^\circ$ grid cell near the equator. An example of an area tessellated by H3 is shown in Figure 3.3.1. Other potential systems which could be used include S2 by Google which is a square system, or simply using a longitude-latitude system as various other works do. We show some example pathways using different grid systems and resolutions in the Supplementary Information Figure B.H.1. The longitude-latitude system results in pathways that unrealistically follow long block-wise vertical or horizontal straight line motions, in contrast to the more realistic and meandering pathways produced by the hexagonal-pentagonal *H3* grid system.

3.3.3 Most Likely Path

For our analysis, the first step is to define a most likely path. A path is simply a sequence of states such that the first element is the origin and the last element is the destination. We also require that each state in the path is unique.

Definition 3.3.1 (Path)**.** *We define the space of possible paths $\mathcal{P}_{o,d}$, between the origin $o \in \mathcal{S}$ and destination $d \in \mathcal{S}$, as the following:*

$$\mathcal{P}_{o,d} = \{\mathbf{p} = (p_1 = o, p_2, p_2, \ldots, p_{n-1}, p_n = d) : n \in \mathbb{N}, p_i \in \mathcal{S},$$
$$i \neq j \Rightarrow p_i \neq p_j\}.$$

With a cardinality operator $|\mathbf{p}| = n$ which denotes the length of the path.

Given the transition matrix T we define the probability of such a path:

$$P(\mathbf{p}) = \prod_{i=1}^{|\mathbf{p}|-1} P(s_{i+1} = p_{i+1} | s_i = p_i) = \prod_{i=1}^{|\mathbf{p}|-1} T_{p_i, p_{i+1}}. \tag{3.3.4}$$

Definition 3.3.2 (Most likely path)**.** *Consider any path* $\mathbf{p} \in \mathcal{P}_{o,d}$*. By the most likely path* $\hat{\mathbf{p}}$ *we mean the path which maximizes the probability of observing that path.*

$$\hat{\mathbf{p}} = \underset{\mathbf{p} \in \mathcal{P}_{o,d}}{\arg\max} \{P(\mathbf{p})\} = \underset{\mathbf{p} \in \mathcal{P}_{o,d}}{\arg\max} \left\{ \prod_{i=1}^{|\mathbf{p}|-1} T_{p_i, p_{i+1}} \right\}. \tag{3.3.5}$$

Optimising Equation (3.3.5) appears intractable at first glance. However, this can easily be solved with shortest path algorithms such as Dijkstra's algorithm [Dijkstra, 1959]. We give precise details on how to find this pathway in Appendix B.C.

3.3.4 Obtaining a travel time estimate

The most likely path is often a quantity of interest in itself, however we can also obtain a travel time estimate of this path. The method should be fast and efficient as it should be able to run for large sets of locations quickly. We achieve this by giving a formula to estimate the travel time based directly on the transition matrix.

Consider the path, $\mathbf{p} = \{p_1, \ldots, p_n\}$, from which we aim to estimate the expected travel time. The key consideration this section addresses is: the path is a sequence of unique states, whereas when simulating from a discrete time Markov chain, the chain has a probability of remaining within the same state for multiple time steps. We therefore aim to obtain an estimate of how long the Markov chain takes, on average, to jump between p_i and p_{i+1}, and then aggregate this over the path to form a travel time estimate.

We assume that the only possibility is that the drifter follows the path we are interested in. So p_i must be followed by p_{i+1}. Now we use t to index the time of the Markov chain and suppose $s_t = p_i$. In this case we do not require that $i = t$ as we allow repeated states in the random process s_t. We are then interested in the random variable k where $t + k$ is the first time that the process transitions from p_i to p_{i+1}. Note that the only possibility for states $\{s_{t+l}\}_{l=1}^{k-1}$ is that they are all p_i, otherwise the object would not be following the path of interest. Therefore, we obtain the distribution of k as follows (proof in Appendix B.E):

$$\begin{aligned} &P(s_{t+k} = p_{i+1}, \{s_{t+l} = p_i\}_{l=1}^{k-1} | s_t = p_i, \mathbf{p}\}) \\ &= \frac{T_{p_i,p_{i+1}} T_{p_i,p_i}^{k-1}}{(T_{p_i,p_i} + T_{p_i,p_{i+1}})^k}. \end{aligned} \tag{3.3.6}$$

Note that if we set $a = \dfrac{T_{p_i,p_i}}{T_{p_i,p_i} + T_{p_i,p_{i+1}}}$ in Equation (3.3.6) we get:

$$P(s_{t+k} = p_{i+1} | s_t = p_i, \mathbf{p}) = a^{k-1}(1-a), \tag{3.3.7}$$

which is the probability distribution function of a geometric distribution with success probability $1 - a$. We denote the random variable for the travel time between p_i and p_{i+1} as k_i. As the geometric distribution corresponds to the time until a failure, we are interested in taking one time increment longer than this as we require k_i to be the time that we move from p_i to p_{i+1} i.e. the time of the failure. Therefore the distribution of k_i exactly follows $k_i - 1 \sim \text{Geom}(1-a)$. Also, note that k_i is in units of the chosen Lagrangian cutoff time $\mathcal{T}_L$.

To get the expectation of the total Lagrangian travel time we consider the sum of

all the individual parts of the travel times $\mathbf{k} = \sum_{i=1}^{|\mathbf{p}|-1} k_i$, such that we obtain:

$$\mathbb{E}[\mathbf{k}] = \sum_{i=1}^{|\mathbf{p}|-1} \mathbb{E}[k_i] = \sum_{i=1}^{|\mathbf{p}|-1} \left(\frac{T_{p_i,p_i}}{T_{p_i,p_{i+1}}} + 1 \right), \tag{3.3.8}$$

where we have used that the expectation of the negative binomial is $\mathbb{E}[x \sim \mathrm{NB}(1, a)] = \frac{a}{1-a}$.

We could attempt to obtain a simple variance estimate for the estimate $\mathbb{E}[\mathbf{k}|\mathbf{p}]$ and $\mathrm{var}\,[\mathbf{k}|\mathbf{p}]$ easily with classical statistics. However, we would only be able to account for variability within the estimates of the entries T, as we would have to assume $\mathbf{p}$ is known. In our case we are interested in the time of $\hat{\mathbf{p}}$, which is itself an estimate as it depends on T. Obtaining any analytical uncertainty in the estimation of the most likely path would be intractable due to the complexity of the shortest path algorithm. Therefore, we propose to address the issue of uncertainty in $\mathbb{E}[\mathbf{k}|\mathbf{p}]$ and $\hat{\mathbf{p}}$ due to data randomness in Section 3.4.2 using the non-parametric bootstrap. To finish this section, we provide the pseudo-code for our approach in Algorithm 2.

One aspect to note which makes the pathways very difficult to check is that the most likely path's travel time is not necessarily the most likely travel time. It could be the case that there are many alternative, almost as likely pathways all which have a different travel time. Therefore, the travel time of these alternative pathways may have a higher contribution to the total travel time distribution than the most likely path. This is intractable to validate unless we just simulate many pathways and travel times. We do some brief comparisons to the real observed modal travel times in 4.5.1.

Algorithm 2 Pseudo-code which summarizes how Section 3.3 is used to turn drifter data and a spatial index function into most likely paths and travel time estimates.

Input: Drifter data set $\mathbf{y}$, a set of locations $\mathbf{x}$, Lagrangian cutoff time $\mathcal{T}_L$

Map all the drifter locations $\mathbf{y}$ to their grids $g_{j,i} = f(\mathbf{y}_{j,i})$ using the map from Equation (3.3.1). Map all the locations of interest to their grids $g^{x_i} = f(x_i)$.

Form transition matrix T using Equation (3.3.3). **for** *each unique pair o and d in* $\{g^{x_i}\}_{x_i \in \mathbf{x}}$ **do**

  Find and store the most likely path $\hat{\mathbf{p}}_{o,d}$ by optimizing Equation (3.3.5). Using this path, find and store the expected travel time $\mathbb{E}[\hat{\mathbf{k}}_{o,d} | \mathbf{p} = \hat{\mathbf{p}}_{o,d}]$ using Equation (3.3.8).

end

Result: Travel times $\mathbb{E}[\hat{\mathbf{k}}_{o,d} | \mathbf{p} = \hat{\mathbf{p}}_{o,d}]$ for every pair of locations in $\mathbf{x}$ and a corresponding path $\hat{\mathbf{p}}_{o,d}$ given as a sequence of grid indices in $\mathcal{S}$.

3.4 Stability and Uncertainty

3.4.1 Random Rotation

A key consideration is that the final results of the algorithmic approach may strongly rely on the precise grid system f chosen in Equation (3.3.1). To address the uncertainty due to the discretization we propose to *randomly sample* a new grid system then run the algorithm for the new grid system. In a simple 2d square grid one could simply sample a new grid system by sampling two numbers between 0 and the length of a side of the square, then shifting the square by these sampled amounts in the x and y direction. In global complicated grid systems such as the one we consider here proposing uniform random shifting is not trivial.

Rather than trying to reconfigure the grid system, instead we suggest a more universal alternative. We suggest randomly rotating the longitude-latitude locations of all the relevant data using random rotations. Such a strategy will work for any spatial grid system as it just involves a prepossessing step of transforming all longitude-latitude coordinates[1]. Note that for each rotation we are required to re-assign the points to the grid and re-estimate the transition matrix. These are the two most computationally expensive procedures of the method. To generate the random rotations we use the method suggested by Shoemake [1992]. In summary, it amounts to generating 4 random numbers on a unit 4 dimensional hypersphere as the quaternion

[1] Conditional on the grid system having a reasonable minimum area. This method rotates the poles to a random point, which would give spurious results in a longitude-latitude grid. Thus providing another reason why the H3 system is more suitable.

representation of the 3 dimensional rotation, which can equivalently be represented as a rotation matrix M. Then we apply this rotation to the Cartesian representation of longitude and latitude.

To obtain travel times which remove bias effects from discretization, we sample n_{rot} rotation matrices $M^{(i)}$. We then run Algorithm 2, however as a prepossessing step we rotate all locations of the drifter trajectories and locations of interest. For each rotation matrix this will result in a set of travel times $\hat{d}^{(i)}$. The sample mean of these rotations will be more stable than the vanilla method. The sample standard deviation will inform us about uncertainty in travel times due to discretization.

3.4.2 Bootstrap

If we required a rough estimate of uncertainty we could consider that $\hat{\mathbf{p}}$, the most likely path, is fixed and then estimate Var $[\hat{\mathbf{k}}]$. However, this would be a poor estimate because such an estimate would assume that: (1) the transition matrix entries follow a certain distribution, and (2) the path $\hat{\mathbf{p}}$ is the true most likely path. In reality neither of these are true, they will both just be estimates. The transition matrix elements are estimated from limited data and the shortest path strongly depends on the estimated transition matrix, e.g. a small change in the transition matrix could result in a significantly different path. Therefore, we obtain estimates of uncertainty by bootstrapping [Efron, 1993].

Bootstrapping is a method to automate various inferential calculations by resampling. Here the main goal is to estimate uncertainty around $\hat{\theta} = \mathbb{E}[\hat{\mathbf{k}}]$. The bootstrap involves first resampling from the original drifters to obtain a new data set. We call

$\mathbf{y}^* = \{\mathbf{y}_j^*\}_{j=1,\dots N}$ a bootstrap sample, where $\mathbf{y}_j^*$ is a drifter trajectory which has been sampled with replacement from the original N drifters. Then we use $\mathbf{y}^*$ as the input dataset to Algorithm 2.

We do this resampling B times to obtain B estimates of $\hat{\theta} = \mathbb{E}[\hat{\mathbf{k}}]$, we denote these bootstrap estimates as $\{\hat{\theta}^{(b)}\}_{b=1}^{B}$. We then estimate our final bootstrapped mean and standard deviation estimates as the following:

$$sd_{boot}^2 = \left[\frac{\sum_{b=1}^{B}\left(\hat{\theta}^{(b)} - \hat{\theta}^{(\cdot)}\right)^2}{B-1}\right],$$
$$\text{where} \quad \hat{\theta}^{(\cdot)} = \sum_{b=1}^{B} \hat{\theta}^{(b)}/B. \tag{3.4.1}$$

In addition to the uncertainty measure in travel time that both the bootstrap and rotation methodology provide, these methods also supply a collection of sample most likely paths. These paths can be used to investigate various phenomena, such as why the uncertainty is high. We can plot the paths for a fixed origin-destination pair and may see for example that many paths follow one current where another collection of paths follow a different current. We give numerous examples of this in Sections 3.5.2 and 3.5.3.

3.5 Application

We use the locations given in Table 3.5.1 for the demonstration of the method described in this chapter. These locations were chosen for multiple reasons; (1) they were placed on or near ocean currents, such as the South Atlantic current, the Equatorial current and the Gulf Stream; the magnitudes of which can be seen in Figure 3.1.1,

	Longitude	Latitude
1	9.0	-25.5
2	-25.0	-5.0
3	-45.0	-40.0
4	-69.0	39.0
5	-42.5	41.5
6	-42.0	27.5
7	-93.2	24.8

Table 3.5.1: Table of station locations

and (2) stations were placed in both the North and South Atlantic to show how the method can find pathways which are not trivially connected. First we go over an application of the vanilla method from Section 3.3, then we provide brief results using the adaptations using bootstrap and rotations from Section 3.4 in Section 3.5.2 and Section 3.5.3 respectively. We supply a link to a Python package and code used to create these results in Appendix B.A. Prior to our analysis we take a practical step to improve the reliability of the method. We find the states corresponding to $-79.7°, 9.07°$, $-80.73°, 8.66°$ (two points on the Panama land mass), $-5.6°, 36°$ and $-5.61°, 35.88°$ (two points on the Strait of Gibraltar), then remove the corresponding rows and columns from T. If this step is not taken the method often uses pathways crossing the Panama land mass, resulting in impossibly short connections to the Pacific Ocean. The reasoning for removing the points on the Strait of Gibraltar is

data-driven, further details are in the supplementary information, particularly how one can adapt the method to specify travel times into and out of the Mediterranean Sea.

Figure 3.5.1 shows the pathways between a representative sample of the stations. First we note what features are observed in the most likely path. The Gulf Stream is used on almost every path trying to access locations 4, 5 or 6 in Figure 3.5.1. Observe in Figure 3.5.1 c) when going from location 3 to 5 that the method chooses to enter the Gulf of Mexico and then uses the Gulf Stream to access location 5, even though the actual geodesic distance of this path is long. Other examples which use the Gulf Stream include d) and h). Generally, any of the paths leaving location 1 and attempting to travel northwest use the Benguela Current, for example Figure 3.5.1 a), i) and g).

The travel times obtained between the sample stations in Figure 3.5.1 show interesting results regarding the lack of symmetry when reversing the direction of the path between two stations. When going from location 2 to location 4 we estimate a long most likely path in terms of physical distance. However, the resulting travel time of this path (0.7 years) is smaller than the travel time of the more direct path from location 4 to location 2 (4.8 years) - which is much shorter in distance. This is because the path going from location 2 to location 4 follows strong currents such as the North Equatorial current and the Gulf Stream. Another interesting result is that going from 3 to 5 and vice versa are relatively close in terms of travel time even though 3 to 5 uses the Gulf Stream but the return does not. In the most likely path from 3 to 5, up until around -16° latitude the travel time is 5.2 years, which we

expect as the pathway seems to be going against the Brazil current. After this point the rest of the path takes the remaining 3 years despite the remainder being over half the actual physical distance of the pathway. We expect this short time is due to the method finding a pathway along the North Brazil current, followed by the Caribbean current, followed by the Gulf Stream.

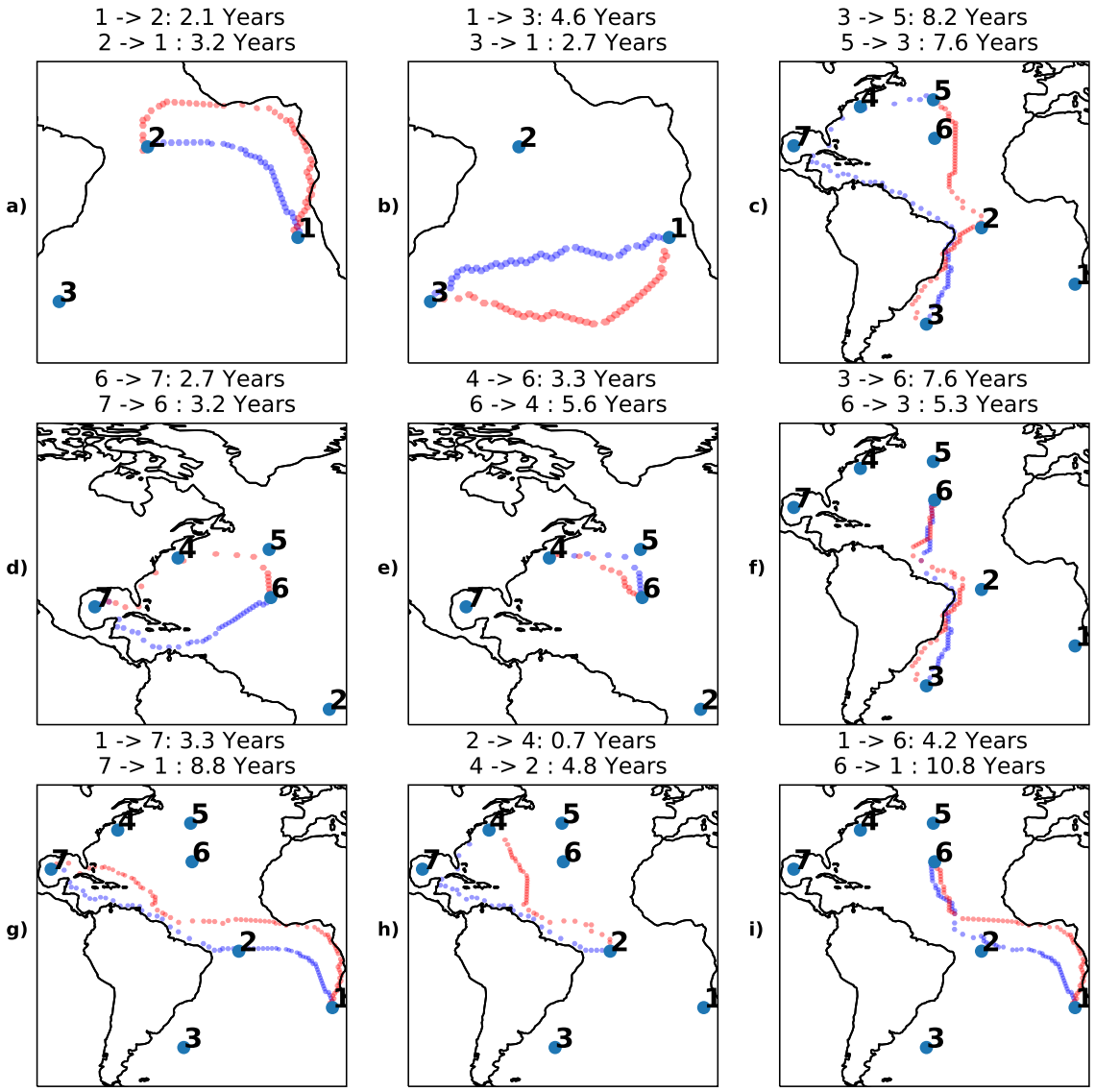

Figure 3.5.1: Example pathways found from the method. Sequences of blue hexagons are going from the lower number to the higher number. Sequences of red hexagons are going from the higher number to the lower number. Numbered locations are as in Table 3.5.1. The expected travel time of the most likely path is given in the title of each plot. Similar plots can be provided for every location pair using the online code, however these are not presented here owing to page length considerations.

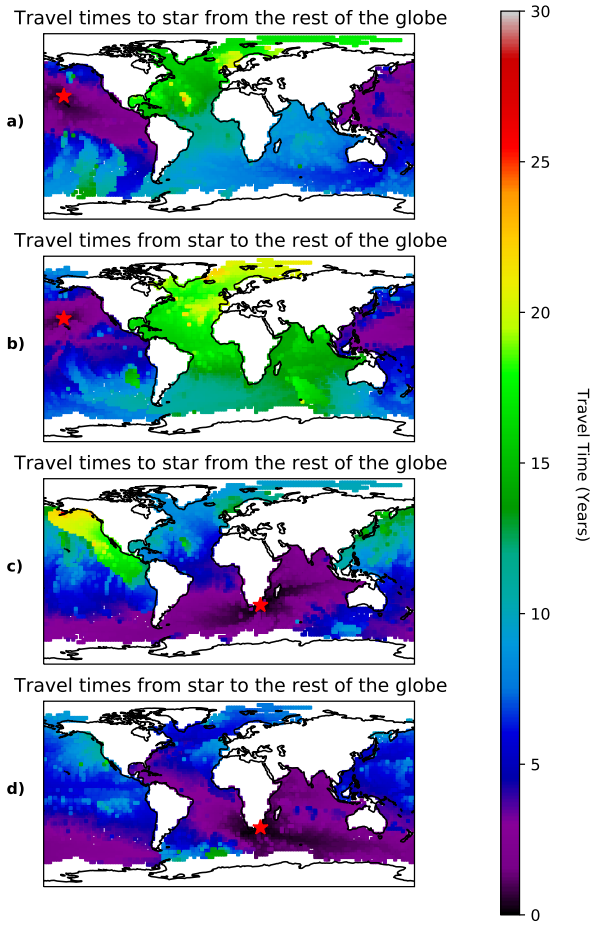

Figure 3.5.2: Travel times of the most likely path originating from the red stars and going to or from (indicated by the title) the centroid of a 2.5° × 2.5° square grid system. Figure setup and locations taken to match Figure 2 of Jönsson and Watson [2016].

3.5.1 Global Travel Times

Figure 3.5.2 shows the travel time distribution to and from two fixed locations, taken to match the studied locations of Jönsson and Watson [2016], to the entire globe. We note that the travel time map is less smooth than the one shown in Jönsson and Watson [2016]. The black and purple areas however (up to 5 years travel time) are similar to those found in Jönsson and Watson [2016], showing agreement over short spatial scales. When it comes to larger distances we generally find the maps are markedly different. For example the yellow patch in the north east Pacific in Figure 3.5.2c is not seen in Jönsson and Watson [2016]. Such discrepancies can be attributed to many reasons such as: (1) they reflect the difference in methods, where we use a transition matrix approach, and Jönsson and Watson [2016] use a connectivity matrix; (2) Jönsson and Watson [2016] aim to find the shortest path in time, whereas we aim to find the expected time of the most likely path; and (3) the results shown here are derived from real data, whereas Jönsson and Watson [2016] use simulated trajectories.

We show an example in Figure 3.5.3 which explains the lack of spatial smoothness in Figure 3.5.2, where we show two pathways both originating from a fixed point and ending at two distinct points only 1° latitude apart. The points are on either side of the discontinuity in the north-east Pacific seen in Figure 3.5.2c. The pathways become visibly different after they have both reached the south Pacific. Such a phenomenon results in the lack of spatial smoothness of travel time distributions. This demonstrates that the travel times do not necessarily obey the triangle inequal-

ity. If smoothness is desired we show an alternative approach in the supplementary information, where instead a minimum travel time is the objective, which is then more analogous to the Jönsson and Watson [2016] approach. We argue however that the expected travel time of the most likely path, rather than the minimum travel time, is a more relevant metric for estimating connectivity and Lagrangian distance in applications measuring spatial dependence between points in the ocean.

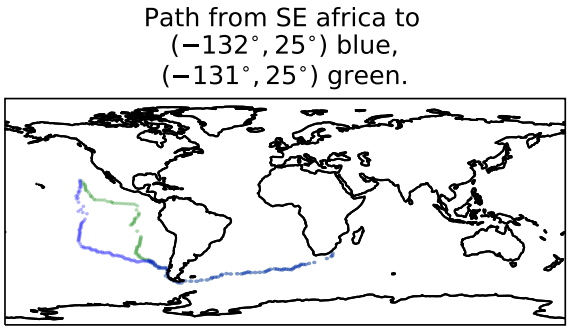

Figure 3.5.3: The most likely path from two points in the North Pacific to the south-east coast of Africa. The green and blue pathways are almost identical as they cross the south Atlantic. The pathways differ greatly however as they cross the Pacific, even though the two starting points in the north Pacific are only 1 degree apart. The path going from $-131°, 25°$ has an expected travel time of 21.2 years, the path going from $-132°, 25°$ has an expected travel time of 11.4 years.

3.5.2 Bootstrap

To show the value of the bootstrap we show the results for one particular pair of stations, the pathway going from location 1 to location 3 and back. The pathways which result from the bootstrap are shown in the bottom panel of Figure 3.5.4. The

darker lines on the figure imply that that this transition is used more often. We see that for most of the journey the darker lines closely follow the original path. The bootstrap discovers some slightly different paths, for example around $-20°$ Longitude the path going from 3 to 1 occasionally seems to find that going further south is a more likely path. Also, around the beginning of the path going from 1 to 3, we see that the most likely path taken most frequently by the bootstrap samples often does not follow the most likely path from the full data.

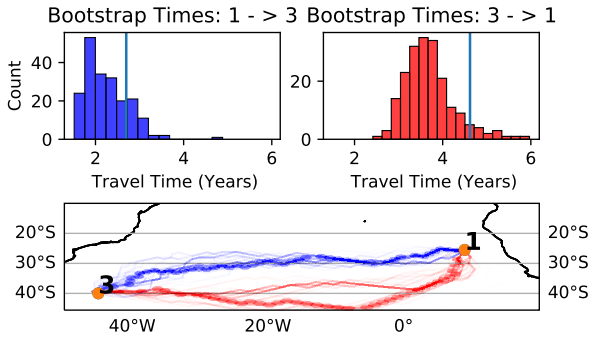

Figure 3.5.4: Two bootstrap distributions of travel times are shown in the top row resulting from 200 bootstrap samples. The vertical line is the travel time if the full data is used to estimate the path and time. The corresponding bootstrapped paths are shown in the bottom figure. Blue lines and hexagons are for going from 1 to 3, red lines and hexagons are for going from 3 to 1. The lines connect the centroids of the spatial index of the bootstrapped paths. Darker lines mean that path is taken more often. The light hexagons are the pathway taken if the full data is used with no resampling i.e. the pathway shown in Figure 3.5.1.

The main goal of the bootstrap is that we obtain an estimate of the standard errors. In this case we get standard error estimates using Equation (3.4.1) of 0.5

years for going from 3 to 1 and 0.6 years for going from 1 to 3. In general, we found that the standard error was lower when the path follows the direction of flow. The top row of plots in Figure 3.5.4 appears to show that there is a slight bias between the bootstrap mean and the vanilla method travel time. We believe this is due to the variance within the paths. The mean estimated from the bootstrap samples are close to the estimates from the rotation method we will shortly present in Figure 3.5.6. The rotation mean estimates are within 0.4 years of the bootstrap means in both cases shown here.

3.5.3 Rotation

If we consider two points in the same H3 Index, for example location 1 $(9°, -25.5°)$ and a new point $9°, -26.2°$ (as shown in Figure 3.5.5), then using the original grid system the method will simply produce a travel time of 0. To solve this problem, we consider using 100 rotations as explained in Section 3.4.1. For each rotation we estimate the travel time both back and forth. In 22 of the rotations the two points ended up in the same hexagon, hence resulting in a zero travel time. We plot the distribution of the other 78 travel times in the bottom row of Figure 3.5.5. The mean of all the entries including the zeros is 20.5 days for going from the new point to location 1, and 22.2 days for going from location 1 to the new point.

The second benefit of performing rotations is to make estimates less dependent on the grid system. We use the same 100 rotations as with the previous example, and compute the most likely path and the mean travel times. In Figure 3.5.6 we plot the pathways with the mean and standard deviation of the travel times resulting from

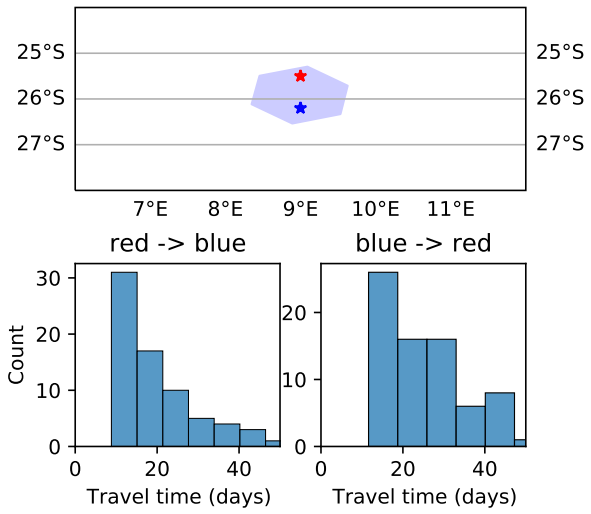

Figure 3.5.5: Plot of location 1 from Table 3.5.1 and the point $9°, -26.2°$, which is $0.7°$ south of location 1. The relevant H3 hexagon is plotted over the points. In the bottom row we plot the histogram and density estimate of the travel times in each direction from applying 100 rotations. The 22 zeros for when the two locations are in the same hexagon are not included in the histogram.

these 100 rotations. The travel times and paths shown in this figure are comparable to those given in Figure 3.5.1. In most of the pathways we see that the darkest, most popular paths match up with the pathways in Figure 3.5.1.

One of the more interesting results from this analysis is the path going from 2 to 1 in Figure 3.5.6 a). Most of the paths go up closer to the Equator, then use the Equatorial Counter current, followed by the Guinea and Gulf of Guinea currents as in the original vanilla application of the methodology. A small number of the rotations result in pathways that end up crossing the South Atlantic, to the south of location 2, then follows the South Atlantic current over to location 1.

In general, the travel times from the rotation and original method can be signifi-

cantly different, which supports the need for this rotation methodology. If we compare Figure 3.5.1 and Figure 3.5.6, most of the distances stay close to what they were in the original results using no rotations. We see that going from 6 to 4 drops from 5.6 years in Figure 3.5.1e) to 3.8 years in Figure 3.5.6e) and 4 to 6 increases from 3.3 years to 5.4 years. This causes the ordering of the distances to change as 6 to 4 is now the shorter travel time. We believe the case in e) is mainly due to 4 being located just north west of the stronger currents of the Gulf Stream, which makes it sensitive to the grid system. However, the high standard errors in Figure 3.5.6 suggest we are uncertain about this travel time.

3.6 Discussion and Conclusion

In contrast to van Sebille [2014], our methodology as presented does not take into account seasonality. We have a few ideas for how seasonality could be incorporated in future work. An obvious adaptation, if the aim was to obtain a short travel time which is expected to lie in a small 3 month window, is to just estimate T using drifter observations which are in that time window. Alternatively, we could use $\mathcal{T}_L$ to be a certain jump such as a gap of two months, then we estimate 6 transition matrices say $T^{(k)}$, where the entries $T^{(k)}_{i,j}$ are probabilities of going from the previous time period at state i to state j at the current time. Such a set up could still be solved using our shortest path algorithm. We justify our approach in the same way as Maximenko et al. [2012]: we aim to provide a global view and a simple general concept explaining the pattern of potential pathways and travel times. The base method can then be

adapted by practitioners to account for local spatial or temporal considerations.

More results demonstrating the robustness of our method, and motivation of parameter choices, can be found in the supplementary information. A key finding we discuss here, is that we found the size of the grid system affects the estimated travel times significantly, regardless of whether the lat-lon or the *H3* grid system is used. Therefore we do not recommend comparing travel times obtained from two different grid sizes. Generally the results are correlated in an order comparison sense, however, their magnitudes change. Typically a smaller grid system results in shorter travel times. Due to this we would only advise the results to be used in relative comparison to each other, for example by saying that the travel time from a to b is twice that than from b to c, where both times are obtained with the same grid system. The choice to show resolution 3 in this chapter was found to perform robustly (balancing the error from discretisation and data sparsity), and follows grid sizes that approximately match previous works where $1^\circ \times 1^\circ$ grids are used, but this can be changed easily in the online package.

More detailed grid systems could also greatly benefit the Markov transition matrix approach in general. For example, in Section B.H.5 we discuss how the method can take pathways across land-masses and ocean barriers if a grid covers both sides. A very useful adaptation would be to make a grid system that never crosses a land mass. This would stop artificial leakage which we had to deal with by removing links in the transition matrix. If we could also allow the grid cells to adaptively resize based on the ocean currents this could result in more realistic results which take into account currents which act as barriers.

The use of the bootstrap and rotations are relatively easy methods to implement, each of which provides effective estimates of uncertainty from data uncertainty and discretisation respectively. However, combining these procedures into one requires careful consideration. If we wanted to run n_{rot} rotations and B bootstraps for each rotation, we still require a method to combine these estimates of travel times. We could treat every rotation equivalently, so say that our bootstrap sample in Equation (3.4.1) is all $n_{rot} \times B$ samples to obtain an estimate of uncertainty in travel time due to the combination of grid discretization and data randomness. Additionally, we could decompose the uncertainty and provide a standard error for just the data randomness by estimating a standard error for each rotation using just the B samples in each rotation, and then taking the average of all n_{rot} standard error estimates.

Our choice of the Lagrangian decorrelation time of 5 days may be too low in some instances. Previous works have found correlations in the velocity data lasting longer than 5 days in certain regions [Lumpkin et al., 2002, Zhurbas and Oh, 2004, Elipot et al., 2010]. This may suggest that using a larger value for $\mathcal{T}_L$ may be needed to justify the Markov assumption. The tradeoff however is resolution, where shorter timescales allow pathways and distances to be computed with more detail. Our methodology is designed flexibly such that the practitioner can pick the most appropriate timescale for the spatial region and application of interest.

In general some unexpected features of the method do occur such as the discontinuity discussed in Section 3.5.1. We expect there would be less of a discontinuity if these times were computed with the rotation methodology, however we argue that the discontinuities with travel times of most likely pathways should always exist. If

smoothness of travel times was a major requirement, then one could consider the *shortest* path in travel time rather than the *most likely* path. We briefly show this adaptation in the supplementary information. We expect the results would require more careful checking in such an approach, as the shortest path would be more likely to use any glitches in the grid system such as if there was a connection over Panama.

To summarize, in this chapter we have created a novel method to estimate Lagrangian pathways and travel times between oceanic locations, thus offering a new, fast and intuitive tool to improve our knowledge of the dynamics of marine organisms and oceanic transport and global circulation.

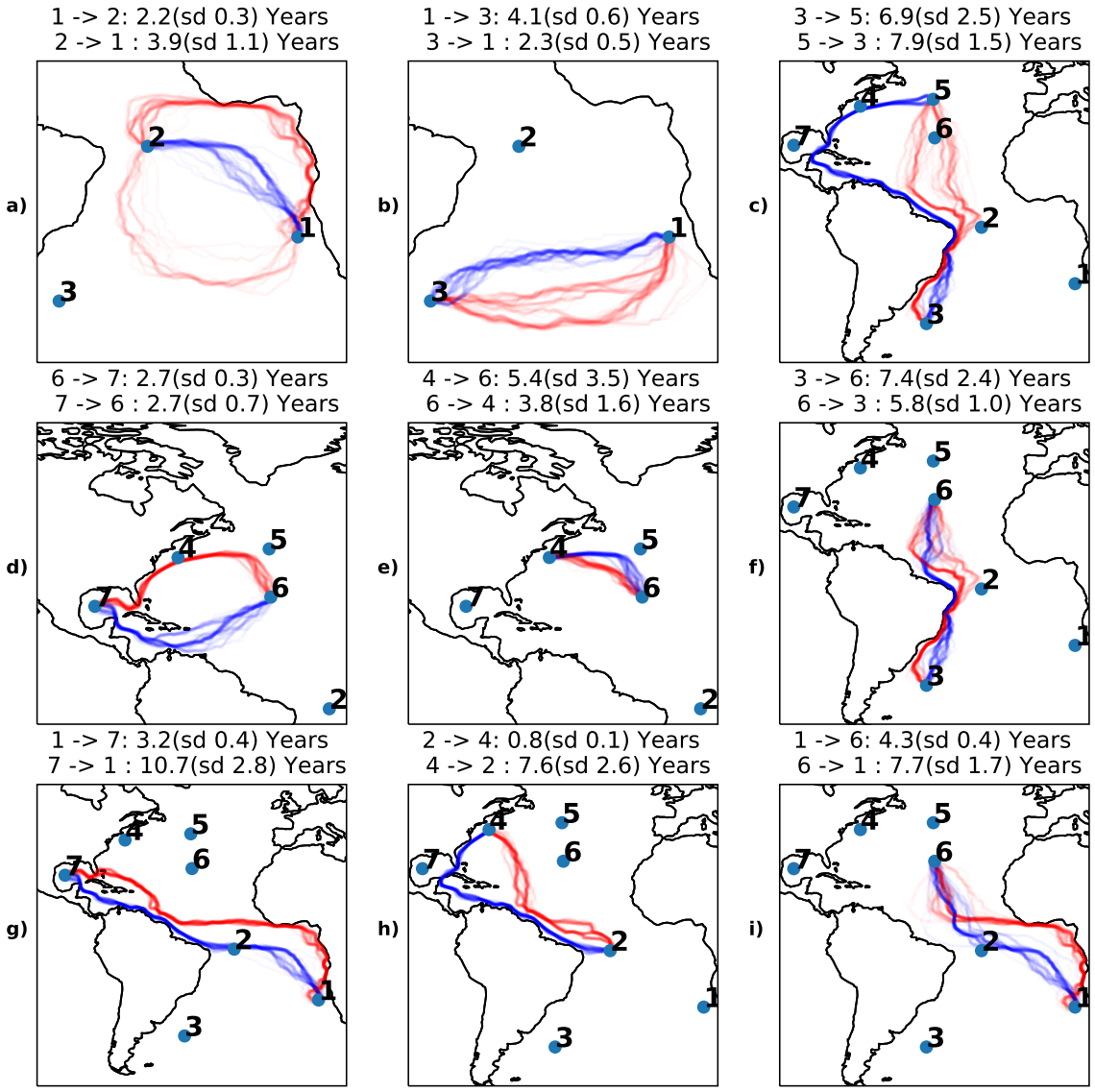

Figure 3.5.6: This figure layout is the same as in Figure 3.5.1, except here we plot paths resulting from 100 random rotations. Each line connects the centroid of each hexagon within the path. Note that the hexagons now come from rotated grid systems, so the centroids could be at any location hence the smooth continuous looking lines. The lines are plotted with transparency, when multiple lines overlap these lines will look darker. Standard deviations of the travel times of the 100 paths are reported in the title of each figure.

Chapter 4

Multidimensional Scaling for Travel Time Matrix Completion

4.1 Introduction

The motivation and aims of this chapter overlap with those of Chapter 3, hence we refer the reader to Section 3.1 for further background. As a brief summary, we aim to take S locations of interest in the ocean $\mathbf{p}_s \in \mathbb{R}^2$, $s \in \{1, \ldots, S\}$ which we will refer to as stations, and produce a measure of travel time between any pair of stations. The travel time will be derived from a large set of trajectories $\{\mathbf{y}_j\}_{j=1}^{N}$ where $\mathbf{y}_j = \{\mathbf{y}_{i,j}\}_{i=1}^{n_j}$. Unlike the approach in Chapter 3, the method introduced in this chapter does not focus on obtaining a candidate pathway, only a travel time.

In the previous approach introduced in Chapter 3, we assumed a model for the motion of drifters, and then extracted a quantity which we use as a travel time estimate from the estimated model. In practical terms this works very well: it is fast, global

and provides sensible results with no hand tuning required. However, the method also has weaknesses including:

- The Markov model assumption is required to make the problem tractable; it is not necessarily realistic.
- The results are heavily dependent on the discretization step. For example, the travel times are seen to change significantly when a finer resolution for spatial discretization is used.

This chapter proposes a more direct approach. We aim to obtain an estimate of the travel time between two points by using the observed travel times from individual drifters that have visited both points. Specifically, we estimate how long it takes a trajectory to travel within a radius of the two locations of interest.

In contrast to the most likely path approach in Chapter 3, we rely on a model-free approach, and we make no assumptions about how the trajectories are generated. The approach in this chapter provides a much more tangible travel time estimate, as we can say the estimates which we provide can be directly estimated from trajectory data. Furthermore, the estimate provided in this chapter is a symmetric matrix, whereas the method proposed in Chapter 3 is typically not symmetric. However, the method introduced in this chapter does not scale globally, which is why we provide both chapters as separate contributions to the broad problem of interest. We will further discuss the advantages and disadvantages of the two approaches in Section 4.6.2.

4.1.1 Brief summary of the approach

Multidimensional Scaling (MDS) [Torgerson, 1952, Kruskal, 1964, Cox, 2001, Borg, 2005] is a family of models to analyse a set of pairwise distances or dissimilarities between a set of objects. MDS aims to reconstruct a representation of these objects in a latent space where the distances in the latent space closely matches the original pairwise dissimilarities. A common illustrative example of MDS is to take flight times between cities to recreate a spatial map of where the cities are located, using only these flight times. However, the flights example would usually have complete data. A similar application to the problem we have here is wireless sensor localization [Biswas et al., 2006]. Wireless sensors receive signals which the strength of can be recorded. However, signals are only received from stations which are close or neighbouring. MDS may then be used to create an embedding of these sensor locations, which can then estimate signal strength between all stations [Drineas et al., 2006, Shang and Ruml, 2004], even those which are distant.

The second example related to wireless sensors is more similar to the problem at hand, where the measure of dissimilarity is travel time. We have finite drifters which travel for a finite amount of time, hence there will be pairs of locations which have no observed travel times. We aim to use MDS to complete these travel times.

The approach we take has two distinct stages. The first is to obtain an estimate of travel time. Unlike the aforementioned examples, there is no one clear metric which we can record as a dissimilarity. We have thousands of trajectories of finite length from which we aim to estimate a set of pairwise travel times directly. We aim to

extract the most likely travel time, the approach for this is outlined in Section 4.2.

Secondly, the algorithm we propose yields travel times which can be either: missing, biased (due to a practical step taken), and have unwanted error due to finite sampling. In Section 4.3, we introduce a multidimensional scaling model to account for these issues.

In Section 4.4, we then implement both steps of the algorithm to a simulated example where we have an idea of what the results should be. We also investigate the sensitivity of the method to how much data is missing. In Section 4.5, we apply the method directly to the Global Drifter Program using a set of locations of interest in the North Atlantic. Finally, we discuss the results and propose various areas of future research in Section 4.6.

4.2 Travel Time Estimation

To simplify the remainder of the chapter, we outline the processing of the dataset. The first step undertaken on the trajectories is to check which stations each trajectory goes to, then record the travel times from and to these stations; this step is explained in Section 4.2.1. Then we extract a single travel time estimate for each station pair. We choose to extract an estimate of the mode of the travel time distribution, the method for this is explained in Section 4.2.2. Finally, in Section 4.2.3 we describe how the former steps are used to create the symmetric pairwise travel time matrix $\mathbf{D}$.

4.2.1 Extracting times

One of the first questions we need to answer is what is a travel time. For a road or train network this is usually a straightforward question to answer. The operators of the vehicles move between nodes on a known graph and generally try to take a direct path. In the ocean we don't have an obvious graph, drifters often loop back to a similar location, and are extremely unlikely to visit any specific point. To simplify this problem we focus on the time of the most direct path any drifter takes to travel from station i to station j.

We record travel times based on which stations a trajectory visits. The problem is that the chance of a trajectory visiting the exact point of the station is extremely unlikely. Instead, we consider an r km radius around the station. If the trajectory is seen within this r km radius of the station, we say it has visited the station, and once the drifter exits that radius we say it has exited the station.

We consider a single trajectory $\{\mathbf{y}_i\}_{i=1}^{n}$; we assume that the trajectory is regularly sampled so the physical time difference between $\mathbf{y}_i$ and $\mathbf{y}_{i+1}$ is taken to be a unit of time.

The algorithm to extract distances is straightforward, we iterate through the trajectory and check which stations the trajectory has visited. For each station, we record two times: the arrival time and the departure time. The arrival time is the first time which we see the trajectory visit the station. The departure time is the last time we class the trajectory as visiting the station, *before* visiting another station. If the trajectory leaves the station but then returns immediately and then leaves again,

Station	Arrival Time	Departure Time
0	2019-12-17 06:00:00	2019-12-18 18:00:00
1	2020-01-12 06:00:00	2020-01-14 18:00:00
5	2020-03-19 00:00:00	2020-03-19 12:00:00

Table 4.2.1: Station sequence for drifter id 68055200 and the corresponding arrival times and the time which the drifter exited the 100km radius. the trajectory can be seen in Figure 4.2.1. Although not clearly seen, the drifter briefly enters the radius of Station 5.

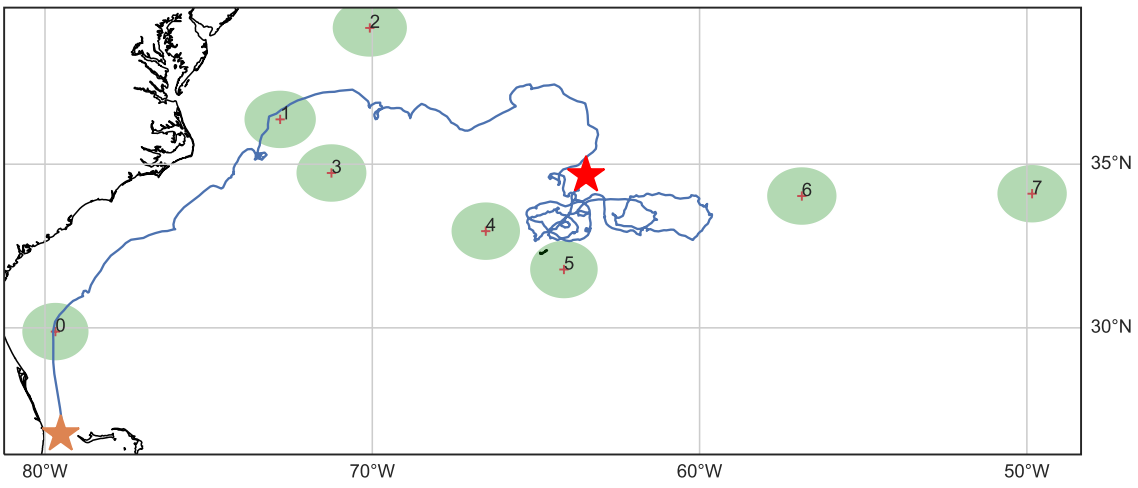

Figure 4.2.1: Trajectory of drifter id 68055200 in the North Atlantic Ocean. The trajectory starts in the south west, at the orange star in the plot and ends at the red star. Considered stations plotted as '+'. A circle with radius 100km drawn around each station.

we would take the time which it leaves for the second time as the departure time. When applied to the trajectory shown in Figure 4.2.1 this process results in a table such as the one shown in Table 4.2.1.

After creating this table, extracting the travel times from station i to station j is straightforward. We take the arrival time at station j minus the departure time from station i. This results in a travel time for a subset of the stations. We iterate over each station i visited in the trajectory, where for each station i we iterate over future stations j and record the time. As an example, in Table 4.2.1 we would record a time for 0 to 1, 0 to 5, and 1 to 5.

If the same station is visited by a trajectory on two occasions, this will usually result in the above algorithm recording two travel times for some of the from-to station pairs. In these cases, we only record the shortest for any given from-to station pair for that trajectory.

When we apply this algorithm to all N trajectories for each station pair $i, j \in \{1, \ldots, S\}$ we obtain a list of travel times $M^{ij} = \{M_k^{ij}\}_{k=1}^{|M^{ij}|}$. We note that for some station pairs no travel times will be observed. This could be for various reasons such as stations being situated extremely far apart, the area being poorly sampled by drifters, or the stations being situated either side of a boundary current. In these cases, we will set $|M_{ij}| = 0$, i.e., no travel time observations. We also force $|M^{ii}| = 0$ as the travel time from a station to itself should be zero.

4.2.2 Positive Kernel Density estimation

The next step is to turn each collection of travel times M^{ij} from all the trajectories into a single estimate of travel time between station i and j. There are many options which may be taken, the most straightforward of which is to take a summary statistic, e.g., minimum, mean, median, mode. In the application we expect the travel time estimates to have a large number of outliers and potentially multiple modes in the travel time distribution caused by multiple ocean current pathways connecting stations. Considering these features of the distributions we aim to estimate the location of the most prominent mode of the distribution of travel times, and use this as our single travel time estimate between station i and j.

For ease of notation, in this section we consider a univariate dataset $\mathbf{x} = \{x_i\}_{i=1}^n$ where $x_i \sim_{IID} X$ where X is a random variable with unknown distribution. In practical terms this dataset $\mathbf{x}$ is the travel time list M^{ij}. We aim to form an estimate of the density function of X using the data $\mathbf{x}$, then extract a mode from this density estimate. This mode will be taken as the travel time. Extracting the mode is a trivial task in the univariate setting where the function is known, it can be done via evaluating the density on a fine grid of points then taking the argument giving the highest density. The rest of this section focuses on the density estimation aspect of the problem.

We choose to use kernel density estimation.

Definition 4.2.1. *A Kernel density estimate is any estimate for a probability density*

function f defined as follows [Bowman, 1984]:

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} K\left(x, x_i, h\right). \tag{4.2.1}$$

Where K is referred to as a kernel function, and h is referred to as the bandwidth parameter.

A Kernel Density Estimate (KDE) is a non-parametric estimate of the probability density function. The estimate heavily depends on the chosen form of the kernel K and bandwidth parameter h. For this chapter we shall use the Gaussian kernel:

$$K(x, x_i, h) = \frac{1}{\sqrt{2\pi}h} \exp\left[\frac{(x - x_i)^2}{2h^2}\right], \quad x \in \mathbb{R}. \tag{4.2.2}$$

The application here applies to strictly positive data, hence the resulting density estimate should not have mass below zero. The approach we take is as in Wand et al. [1991]: we transform the data into a space $g : (0, \infty) \to (-\infty, \infty)$, then do a typical kernel density estimate in this transformed space. Then we transform the density estimate back into the original space. We note that this is not the only possible approach for this non-negative kernel density estimation problem. Other approaches exist such as truncating the distribution below zero then re-normalizing the density to integrate to one [Gasser and Müller, 1979], or reflecting mass below zero back onto the positive domain [Schuster, 1985].

Here we provide a derivation of the formula given by Wand et al. [1991] to get a kernel density estimate in the original space. We start with a density function for a transformed random variable $Y = g(X)$ denoted as $f_Y(y)$. Denote the cumulative density function

$$F_Y(y) = \int_{-\infty}^{y} f_Y(y) dy.$$

Now for any monotonically increasing function g we have that

$$P(X \leq x) = P(Y \leq g(x))$$

$$F_X(x) = F_Y(g(x)),$$

then if we differentiate both sides w.r.t. x we obtain the change of variable formula:

$$\frac{d}{dx}F_X(x) = \frac{d}{dx}F_Y(g(x))$$

$$f_X(x) = g'(x)f_Y(g(x)). \tag{4.2.3}$$

Now consider a kernel density estimate $\hat{f}_Y(y)$ as the density estimate from the transformed data $y_i = g(x_i)$. We can apply Equation (4.2.3) to back transform the data obtaining the result from Wand et al. [1991]:

$$\hat{f}_X(x) = \frac{1}{n}\sum_{i=1}^{n} g'(x)K(g(x), g(x_i), h). \tag{4.2.4}$$

We use the Gaussian kernel with the log transform $g(x) = \log(x)$, with derivative $g'(x) = \frac{1}{x}$. We therefore obtain the component within the sum using:

$$g'(x)K(g(x), g(x_i), h) = \frac{1}{\sqrt{2\pi}xh}\exp\left[\frac{(\log(x) - \log(x_i))^2}{2h^2}\right]. \tag{4.2.5}$$

We note that Equation (4.2.5) is similar in form to the log-Gaussian distribution. This combination is what we use to obtain strictly positive kernel density estimates.

Bandwidth Estimation

Kernel density estimates are largely dependent on the bandwidth parameter h from Equation (4.2.1). A naive option would be to select the bandwidth in the log space; hence carrying out the bandwidth selection to select the optimal $\hat{f}_Y(y)$, then use that

estimated bandwidth parameter to infer the density $\hat{f}_X(x)$. Scott's rule [Scott, 1992] is commonly used and selects a bandwidth according to:

$$h = (4/3)^{1/5}\sigma n^{-1/5},$$

where σ is the sample standard deviation of the data we are fitting a KDE to. This rule is the default option for many software packages. However, Scott's rule is derived by assuming the underlying distribution is Gaussian. Moreover, such an approach would optimize the fit in the log-space, not the original constrained space.

We have no prior expectation for what distribution the data comes from in the drifter application. The family of distributions is likely a mixture, with the number of components being related to the number of prominent pathways in the ocean. Deriving rules of thumb in the same way as Scott's rule would require us to make an assumption on the underlying distribution. Hence, we choose a more data driven approach to bandwidth selection in place of a rule of thumb.

We choose to select the bandwidth based on cross validation. One option is to optimize the held-out log-likelihood using leave one out cross validation. The left out log-likelihood function is defined as:

$$\sum_{j=1}^{n} \log \hat{f}_{h,-j}(x_j), \tag{4.2.6}$$

where $\hat{f}_{h,-j}$ is the kernel density estimate in the original data space excluding the kernel centered on x_j using bandwidth h. The above estimate is inconsistent [Bowman, 1984], and favours very large bandwidths when outliers are present. Hence, we choose to use an alternative discussed in Bowman [1984], Rudemo [1982]; this approach is often referred to as unbiased cross validation or least-squares cross validation in KDE

software packages. The estimate is:

$$l^*_{LOO}(h) = \int_{-\infty}^{\infty} \hat{f}_h^2(x)dx - \frac{2}{n}\sum_{j=1}^{n} \hat{f}_{h,-j}(x_j), \tag{4.2.7}$$

where $\hat{f}_h$ is the kernel density estimate using all of the data and bandwidth h. We minimize Equation (4.2.7) to select the bandwidth.

Is the strictly positive adaptation necessary?

In this application, we are using the kernel density estimation for mode extraction. So now we investigate the ability of the method to extract modes if we model the distribution in an unconstrained space, in comparison to modelling the distribution in the log transformed space.

As an extreme example, we consider a mixture of two gamma distributions, the density for which is below:

$$f_X(x) = 0.4\frac{1}{\Gamma(4.5)0.5^{4.5}}x^{3.5}\exp\left(\frac{-x}{0.5}\right) + 0.6\frac{1}{\Gamma(15)}x^{14}\exp(-x). \tag{4.2.8}$$

The density of this distribution is shown in Figure 4.2.2a). We simulate datasets of size 10, 30 and 100 from this distribution. This choice of sample size is representative of the samples which we shall encounter in Section 4.5.

We then estimate the KDE using a traditional estimate with a Gaussian Kernel labelled as "kde" in the plot, then "log kde" uses the transformed KDE. For each dataset, we select the bandwidth to maximize the cross validation function l^*_{LOO} from Equation (4.2.7). We plot the resulting density estimates in Figure 4.2.2. We can see that the Gaussian KDE places mass below zero resulting in the first mode being underestimated. In all three cases for the Gaussian KDE shown the second mode

would be selected as the most prominent one. Whereas the strictly positive KDE correctly puts more mass on the first mode in every case shown. A similar observation was made by Charpentier and Flachaire [2015] on UK income distributions, where the constrained KDE is less sensitive to bandwidth choice when picking modes of distributions in comparison to the unconstrained KDE.

To make this argument more robust, we simulate 100 datasets from the distribution for each candidate n. We fit a kernel density estimate selecting the bandwidth according to l^*_{LOO} for each dataset. First, in Figure 4.2.3, we plot point-wise 90% intervals and the median.

For the same kernel density estimates we plot the modes in Figure 4.2.4, here the contrast is very clear; the log Gaussian KDE almost always gets closer to the mode around 5. Whereas the unconstrained KDE selects the less prominent mode around 15 in almost half of the simulations. In addition, the log Gaussian KDE selects the smaller mode less often as n increases. The unconstrained KDE does not noticeably get better between $n = 30$ and $n = 100$.

In addition, as shown in Figures 4.2.3 a) and b) and Figure 4.2.2 b) and c), the log Gaussian KDE often places too much mass in the upper tail, resulting in an under-estimate of the second mode. Suggesting that if the second mode was more prominent log Gaussian KDE may miss that and pick the first mode. In general for the application at hand the underestimation of the mode at the higher value is less of an issue as typically practitioners apply minimum travel times for the reasons explained in Jönsson and Watson [2016].

As a practical note, there is a chance that the mode of this kernel density estimate

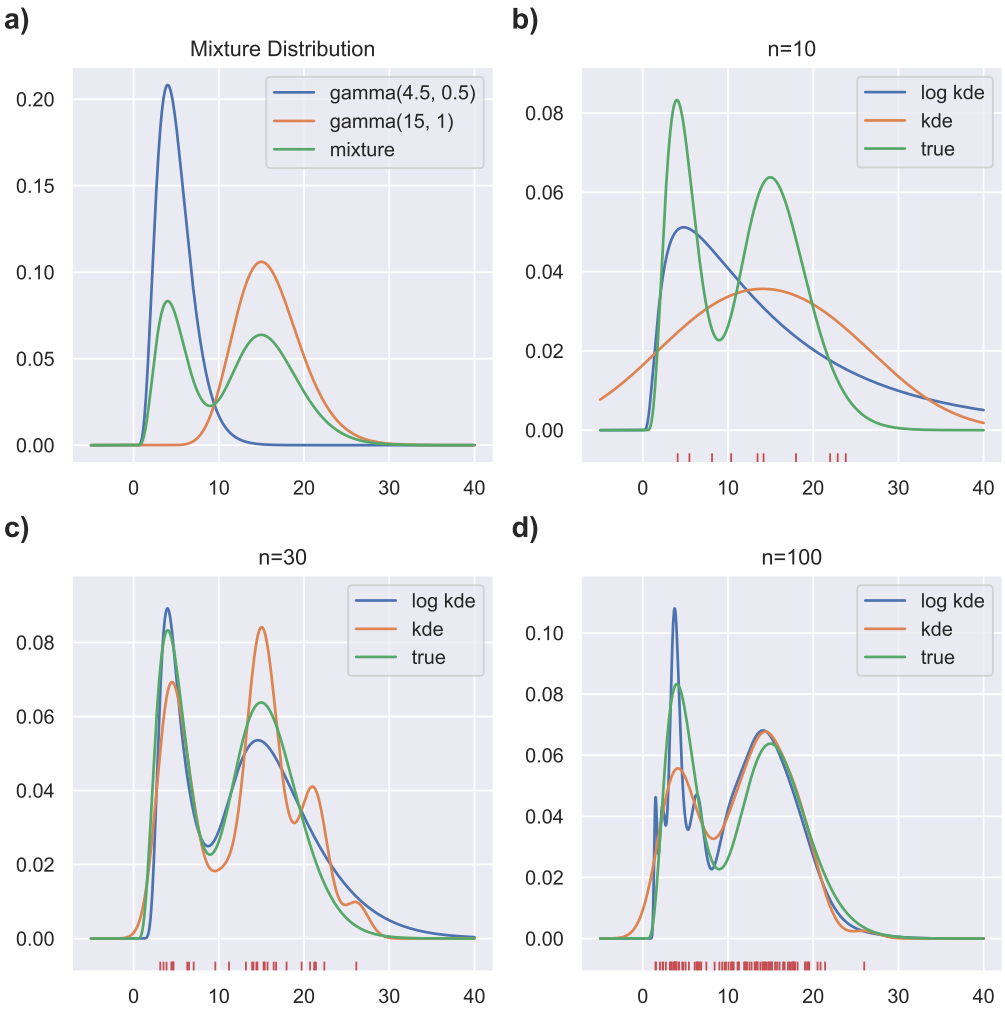

Figure 4.2.2: Plots of the kernel density estimation and target density. a) shows the probability density of each component from Equation (4.2.8). The density labelled mixture in a) is the one we wish to estimate. b), c) and d) show kernel density estimates where we estimate the bandwidth with least squares cross validation. "log kde" uses the kernel specified in Equation (4.2.5), "kde" uses the Gaussian kernel.

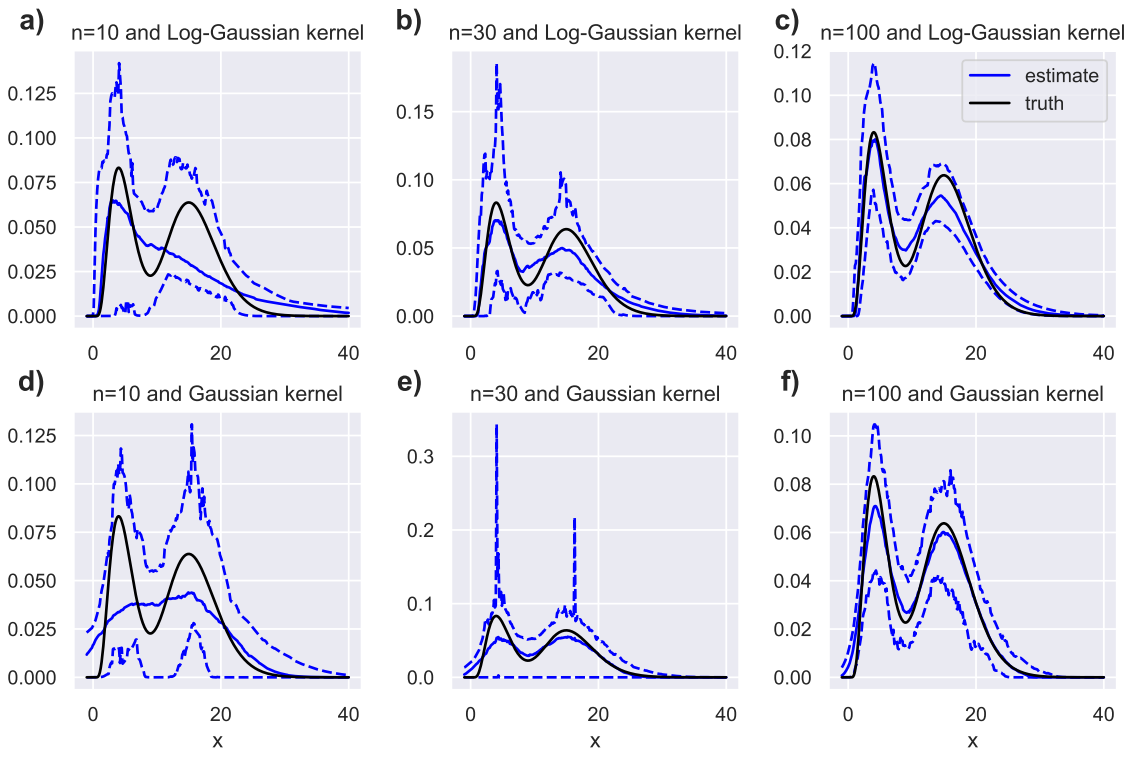

Figure 4.2.3: This plot of pointwise 90% intervals of kernel density estimates over 100 simulations of sample size n from the mixture distribution (true density shown in black). 95% and 5% quantiles plotted as blue dotted lines, 50% quantile plotted as the solid blue line. The top row uses the kernel from Equation (4.2.5), the bottom row is using the Gaussian kernel i.e., it is not constrained to only have mass above zero.

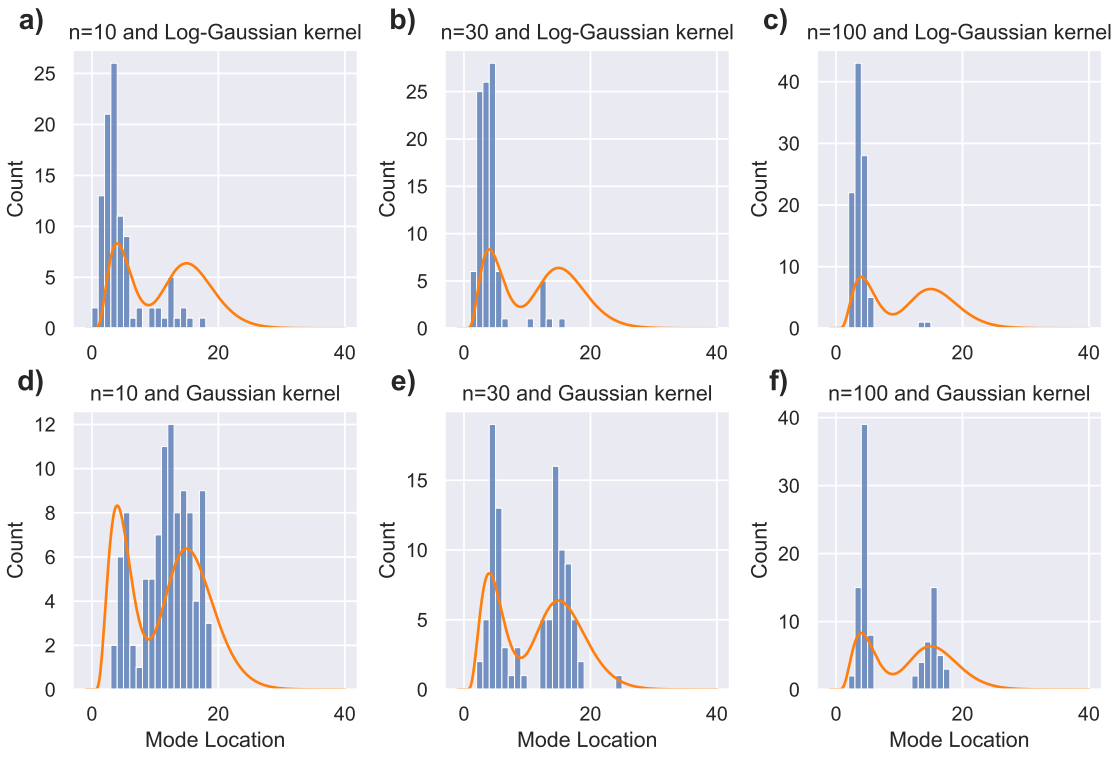

Figure 4.2.4: Histogram of the mode for the same 100 kernel density estimates used in Figure 4.2.3. We extract the mode of the KDE for each simulation, then plot the count in the histogram. True density plotted for reference.

falls lower than the minimum of the data. In these cases, the mode will be taken as the minimum of the data. We only carry out the above procedure if $n > 5$, in all other cases we record the travel time as missing. The reasoning for this is because the estimates seen in the application with only a very small number of samples ($n \leq 5$) are often a result of very indirect pathways. We found that the method we will shortly introduce to estimate missing travel times (via matrix completion) is better than the mode estimates in these cases.

4.2.3 Creating a symmetric travel times matrix

After doing the process mentioned in Section 4.2.1, many of the station pairs had very few observations. 11 out of 42 directed pairs had less than 5 observations. It's likely that the points with few observations will result in unreliable estimates of travel time. So we decided to work with symmetric matrices where the travel time will be interpreted as the travel time in the most likely direction. Working with non-symmetric matrices is an area of future research, potentially using methods from Chapter 20 of Borg [2005]. We also considered just concatenating the travel times from i to j and j to i then estimating a modal based on this new data. However, assuming the travel times for the reverse of the journey come from the same distribution as the travel time for the journey is not realistic in most cases.

We briefly outline the steps on how the symmetric travel time matrix $\mathbf{D}$ is formed. First we run the algorithm described in Section 4.2.1, producing the observed travel times between each i, j pair for M^{ij}. We then take each M^{ij} as the vector $\mathbf{x}$ as described in Section 4.2.2 choose the bandwidth to minimize l^*_{LOO} from Equation

(4.2.7) and form a density estimate for the travel times $f_X(x)$. We then evaluate $f_X(x)$ on a regular grid of 1000 points starting from the minimum of the data and up to the maximum of the data. We store the value of $x \in \mathbb{R}$ which maximizes $\hat{f}_X(x)$ as entry T_{ij}. In the case that $n = |M^{ij}| \leq 5$ the mode is not estimated, the entry is treated as missing.

For the completion method which will be introduced in Section 4.3 we require a symmetric matrix, we denote the symmetric matrix as $\mathbf{D}$ with entries δ_{ij} to follow notation from Borg [2005]. We form a symmetric matrix according to the following rules:

1. If $|M_{ij}| > |M_{ji}|$ set $\delta_{ij} = \delta_{ji} = T_{ij}$
2. If $|M_{ij}| < |M_{ji}|$ set $\delta_{ij} = \delta_{ji} = T_{ji}$.
3. If $|M_{ij}| = |M_{ji}|$ $\delta_{ij} = \delta_{ji} = \mathbf{minimum}(T_{ji}, T_{ij})$.
4. If both T_{ij} and T_{ji} are missing then δ_{ij} and δ_{ji} are treated as missing.

The rules are such that the more popular direction is chosen (more trajectories are seen), and if there is a tie then the lower of the two times is chosen.

4.3 Distance Matrix Completion

Using the method described in Section 4.2, we obtain a symmetric travel time matrix where we may have observations missing. In this section first we introduce classical multidimensional scaling (MDS). Then we introduce a broader more flexible class of

MDS known as metric MDS which can handle missing data. We then introduce an extension of metric MDS which introduces a monotonic transform to the dissimilarities. This monotonic transformation is what we use to correct the additive bias introduced by the radius r.

We use the following notation: let $\mathbf{D}$ be a symmetric $S \times S$ dissimilarity matrix, with entries δ_{ij} in the i^{th} row and the j^{th} column. To match the notation in Borg [2005] we denote the multidimensional embedding as $\mathbf{X}$ which is an $S \times p$ matrix where p is the dimension of the embedding and S is the number of objects. We use $\mathbf{x}_i$ to refer to the i^{th} row of $\mathbf{X}$, and we also refer to $\mathbf{x}_i$ as the embedding point for object i.

In Sections 4.3.1 and 4.3.2 we give a background on MDS and the optimisation framework used. We then introduce a monotonic transformation known as I-Splines in Section 4.3.3. Finally, in Section 4.3.4 we introduce Procrustes analysis which is used to measure the similarity of two MDS embeddings.

4.3.1 Multidimensional Scaling

Multidimensional scaling (MDS) is a model generally used for visualization purposes. MDS takes a set of dissimilarities or similarities; then aims to provide an embedding for each object, where similar objects are close within this embedding. Mathematically we aim to find a set of points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{S}$ where $\mathbf{x}_i \in \mathbb{R}^p$. We abuse notation and use $\mathbf{x}_i$ to refer to a row of the matrix $\mathbf{X}$ which is an $S \times p$ matrix. The points $\mathbf{x}_i$ are referred to as the MDS embedding points and in a good embedding the pairwise distance between point $\mathbf{x}_i$ and $\mathbf{x}_j$ closely resembles an observed distance δ_{ij}. We

measure distance with the L_2 norm, for $\mathbf{y}_i \in \mathbb{R}^p$:

$$||\mathbf{y}|| = \left(\sum_{i=1}^{p} y_i^2 \right)^{1/2}.$$

The dimension of $\mathbf{x}_i$, p, is what is known as the embedding dimension. Typically, we want p to be as low as possible to keep interpretability and stop the model from being too flexible. However, we want it to be high enough to accurately represent the data. Generally, being able to plot the MDS solution is desired which restricts us to $p = \{1, 2, 3\}$. In this chapter we only consider $p = 2$ however we keep the definition general.

In our application the S objects are stations in the ocean and our dissimilarities are the travel times outlined in Section 4.4. We have missing travel times which we aim to fill in using MDS. We now introduce three topics in MDS, the first being classical MDS as it is often what is thought of as the default MDS method. We point out why classical MDS does not work for our problem, then we introduce Metric and Transformed MDS which are more suited to the missing data problem.

Classic Multidimensional Scaling

The original and one of the most numerically efficient MDS algorithms is known as classical MDS as proposed in Torgerson [1952]. Suppose $\mathbf{D}^{(2)}$ is the $S \times S$ squared distance matrix with entries d_{ij}^2 where $d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$. In matrix notation $\mathbf{D}^{(2)}$ can be written as:

$$\mathbf{D}^{(2)} = \boldsymbol{\alpha}\mathbf{1}^\top + \mathbf{1}\boldsymbol{\alpha}^\top - 2\mathbf{X}\mathbf{X}^\top, \tag{4.3.1}$$

where $\boldsymbol{\alpha}$ is an $S \times 1$ vector with entries $||\mathbf{x}_i||^2$ and $\mathbf{1}$ is an $S \times 1$ vector of ones. We now assume that $\mathbf{X}$ is unknown and we wish to re-obtain $\mathbf{X}$ directly from $\mathbf{D}^{(2)}$ where we would use the observed dissimilarities as $\mathbf{D}^{(2)}$. The following process is classical MDS.

For the derivation of classical MDS we need to introduce the centering matrix:

$$\mathbf{J} = \mathbb{I}_S - \frac{1}{S}\mathbf{1}\mathbf{1}^\top,$$

where $\mathbb{I}_S$ is the $S \times S$ identity matrix. Suppose we have an arbitrary $S \times k,\ k \in \mathbb{N}$ matrix $\mathbf{Y}$. The matrix $\tilde{\mathbf{Y}} = \mathbf{J}\mathbf{Y}$ has entries:

$$\tilde{\mathbf{Y}}_{ij} = \mathbf{Y}_{ij} - \frac{1}{S}\sum_{s=1}^{S}\mathbf{Y}_{sj}.$$

So, each element of $\tilde{\mathbf{Y}}$ is the original element minus the sample mean of the data in that column. The centering matrix has the properties that $\mathbf{J1} = 0$, $\mathbf{1}^\top\mathbf{J} = 0$ and $\mathbf{J}^\top = \mathbf{J}$

In order to solve for $\mathbf{X}$ we pre and post-multiply both sides of Equation (4.3.1) by the centering matrix resulting in:

$$\mathbf{J}\mathbf{D}^{(2)}\mathbf{J} = -2\mathbf{J}\mathbf{X}\mathbf{X}^\top\mathbf{J}. \tag{4.3.2}$$

We assume that $\mathbf{X}$ has each column mean equal to zero. So we have that $\mathbf{J}\mathbf{X} = \mathbf{X}$, thus allowing simplification of Equation (4.3.2):

$$\mathbf{X}\mathbf{X}^\top = -\frac{1}{2}\mathbf{J}\mathbf{D}^{(2)}\mathbf{J}. \tag{4.3.3}$$

This assumption that $\mathbf{X}$ has column means equal to zero is not a restriction as we will show in Section 4.3.4. This can be undone after the process through a rigid shift.

We can then obtain $\mathbf{X}$ through a matrix square root, e.g., an eigendecomposition. Set

$$-\frac{1}{2}\mathbf{J}\mathbf{D}^{(2)}\mathbf{J} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\top},$$

where $\mathbf{Q}$ is the matrix with each column being the eigenvectors of the matrix on the LHS, and $\mathbf{\Lambda}$ is a diagonal matrix with the eigenvalues on the diagonal. We have that the eigenvalues on the diagonal are decreasing (the largest eigenvalue is in $\mathbf{\Lambda}_{1,1}$). As we want a p dimensional MDS embedding we set the matrix $\mathbf{\Lambda}_{+}$ as a diagonal matrix $p \times p$ with the square root of the first p positive eigenvalues and $\mathbf{Q}_{+}$ as the $S \times p$ matrix containing the corresponding eigenvectors as columns. Then the coordinate matrix $\mathbf{X} = \mathbf{Q}_{+}\mathbf{\Lambda}_{+}$ is the classical MDS embedding.

To assume that there is this true embedding which generated $\mathbf{D}^{(2)}$ in Equation (4.3.1) is seldom realistic. If it is realistic then $\mathbf{\Lambda}$ will only contain p positive eigenvalues. In the cases where it is not realistic Λ will likely have more positive eigenvalues. When only taking the first p positive eigenvalues we minimize the objective function [Wang, 2012]:

$$\sum_{i=1}^{S}\sum_{j=i+1}^{S}\left(d_{ij}^{2} - ||\mathbf{x}_i - \mathbf{x}_j||^2\right), \tag{4.3.4}$$

with the constraint that the embedding is in $\mathbb{R}^p$.

We do not use classical MDS as it is not flexible enough, namely: (1) We need to make considerable adaptations to deal with missing data. Often this is done by first estimating the missing distances through a naive method such as a shortest path algorithm, then computing the embedding coordinates. (2) We wish to model the additive effect introduced by the radius, making such an adaptation is more difficult

in this framework. We instead choose a more general MDS approach *metric MDS*.

Metric MDS

A more flexible approach is to consider MDS as a optimisation problem [Torgerson, 1952]. The aim is to finding a configuration $\mathbf{X}$ that minimizes an objective function we specify. We denote the pairwise distances between the embedding as $d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$. Commonly with MDS the goal is to minimize stress as in Kruskal [1964][1]:

$$\sigma(\mathbf{X}) = \sum_{i<j} w_{ij}(d_{ij} - \delta_{ij})^2, \tag{4.3.5}$$

where w_{ij} is used as a weighting term, typically always set to 1 and $\sum_{i<j}$ is shorthand for $\sum_{i=1}^{S}\sum_{j=i+1}^{S}$. We find the optimal embedding through minimization of the stress

$$\hat{\mathbf{X}} = \arg\min_{\mathbf{X}} \sigma(\mathbf{X}).$$

We optimize the above with a gradient descent approach, which is outlined in Section 4.3.2. We note that the objective in Equation 4.3.5 is different in form to the function which classical MDS minimizes shown in Equation 4.3.2.

A common usage of the weights w_{ij} is to specify them to deal with missing data [Borg, 2005]. We set the weights as follows to deal with missing data:

$$w_{ij} = \begin{cases} 0 & \text{if } \delta_{ij} \text{ is missing} \\ 1 & \text{Otherwise} \end{cases} \tag{4.3.6}$$

which will remove the dependence of Equation (4.3.5) on the missing data. After fitting MDS we can then estimate the missing δ_{ij} using distances inferred from the embedding as the distances between point i and j.

[1]Originally referred to raw stess in Kruskal [1964], we just define this metric as stress for brevity.

Transformed MDS

For the application we form the estimate of the distance matrix with the method described in Section 4.5. We only measure the travel times after leaving a radius r then entering a radius r of the two stations. This radius introduces a negative bias into the travel times. Here we propose the use of transforming the dissimilarities to correct the bias.

A solution to this is to introduce a monotonic function m which has a strictly positive intercept $m(0)$ which estimates the bias. We apply the transformation to the δ_{ij} and then fit MDS to the transformed quantity. For example, we could use a linear transformation $m(x) = b_0 + b_1 x$ constrained such that $b_0 > 0,\ b_1 > 0$. We set $\hat{d}_{ij} = m(\delta_{ij})$ then we alter the stress function to use these transformed $\hat{d}_{ij}$ in place of the δ_{ij}. Equation (4.3.5) becomes

$$\sigma(\mathbf{X}, \mathbf{b}) = \sum_{i<j} w_{ij}(d_{ij} - \hat{d}_{ij})^2, \tag{4.3.7}$$

where $\mathbf{b} = \{b_0, b_1, ...\}$ contains the parameters related to the monotonic function m. Now the optimization problem has an extra variable

$$\arg\min_{\mathbf{X},\mathbf{b}} \sigma(\mathbf{X}, \mathbf{b}).$$

The algorithm we use to find the optimal $\mathbf{X}$ and $\mathbf{b}$ is described in Section 4.3.2.

In Section 4.4 and Section 4.5, we consider various forms of monotonic function $m(x)$. We refer to the types of MDS with respect to the form of the function $m(x)$. The titles of commonly used MDS transforms are given in Table 4.3.1. Absolute, is the standard MDS model with no transformation. The spline transformation basis functions $I_k(x)$ used for Spline MDS will be introduced in Section 4.3.3.

Absolute	$m(x) = x$
Ratio	$m(x) = b_0 x$
Interval	$m(x) = b_0 + b_1 x$
Spline	$m(x) = b_0 + \sum_{k=1}^{K} b_k I_k(x)$

Table 4.3.1: Titles for metric MDS configurations based on the monotonic transform used as in Borg [2005]. I_k is a monotonic basis function. $b_0,\ \{b_k\}$ are parameters to be estimated or specified in the MDS procedure. Typically, we constrain $b_0 \geq 0,\ b_k \geq 0$ to constrain $m(x)$ to be a monotonically increasing function.

Travel Time Estimate If we had two stations with the circles of radius r we use to obtain travel times touching, we would likely obtain a travel time very close to 0. Therefore $m(0)$ would solely be an estimate of how long it takes an object to travel over the two radii r. $||\mathbf{x}_i - \mathbf{x}_j||$ is an estimate of $\hat{d}_{ij} = m(\delta_{ij})$ as the model is fit to make these quantities match. Hence, we take the inverse $m^{-1}(||\mathbf{x}_i - \mathbf{x}_j||)$ to estimate δ_{ij}. Applying this reasoning the MDS estimate of the unbiased travel time is the sum of the estimate of δ_{ij} and the bias introduced by the radius:

$$m^{-1}(||\mathbf{x}_i - \mathbf{x}_j||) + m(0). \tag{4.3.8}$$

This is the quantity which we recommend to be used from the model when supplying an estimate of travel time between locations i and j. This formula also highlights the relationship between m and the radius r. As we increase r in the formation of the dataset the model should learn that $m(0)$ is larger.

4.3.2 Optimization of Metric MDS

Once we have specified an MDS model the next challenge is to find the parameters which minimize the stress function. We denote the parameters for the monotonic function as $\mathbf{b}$. When referring to the parameters for the MDS configuration we use the notation $\mathbf{X}_f$ as a vector of length Sp, the subscript f denotes that we have flattened the coordinates into a vector, which can be trivially reversed. As a shorthand notation we refer to the stress resulting from the unflattened configuration as $\sigma(\mathbf{X}_f, \mathbf{b})$. As previously mentioned above the goal is to solve the optimization problem

$$\hat{\mathbf{X}}, \hat{\mathbf{b}} = \arg\min_{\mathbf{X},\mathbf{b}} \sigma(\mathbf{X}, \mathbf{b}),$$

where stress is defined as in Equation (4.3.7).

We obtain the gradients $\frac{\partial}{\partial \mathbf{X}_f}\sigma(\mathbf{X}_f, \mathbf{b})$ and $\frac{\partial}{\partial \mathbf{b}}\sigma(\mathbf{X}_f, \mathbf{b})$ using auto-differentiation implemented in the Python package jax [Bradbury et al., 2018]. The use of auto-differentiation makes future extensions to the model easily to implement (e.g. changing the equation for stress σ or monotonic transform m).

Trivial Solution One of the challenges in optimizing transformed MDS with flexible monotonic transforms is that we have a trivial solution: picking a set of parameters $\mathbf{b}$ such that $m(\delta_{ij}) = 0 \ \forall i, j$ and setting $\mathbf{x}_i = \mathbf{x}_j$ will result in $\sigma(\mathbf{X}, \mathbf{b}) = 0$.

This trivial undesired solution is typically avoided by standardizing the transformed distances at each step of the optimization algorithm such that the following is satisfied:

$$\sum_{i<j} \mathbb{I}(w_{ij} > 0)\hat{d}_{ij}^2 = s,$$

for some s which is fixed before the optimization. The optimization algorithm outlined in [Borg, 2005, p. 204] set this value to be:

$$\sum_{i<j} \mathbb{I}(w_{ij} > 0)\hat{d}_{ij}^2 = \frac{S(S-1)}{2},$$

causing the mean transformed dissimilarity to be 1 if there is no missing data. Here we set it such that the mean squared transformed dissimilarity matches the original dissimilarities δ_{ij}. That is we set

$$\sum_{i<j} \mathbb{I}(w_{ij} > 0)\hat{d}_{ij}^2 = \sum_{i<j} \mathbb{I}(w_{ij} > 0)\delta_{ij}^2. \tag{4.3.9}$$

This choice of s is such that in absolute MDS ($m(x) = x,\ \hat{d}_{ij} = \delta_{ij}$) the standardization has no effect.

In addition to the above change in standardization from the framework described in Borg [2005], we also decided to change the quantity which we normalize (the l.h.s. of Equation (4.3.9)). We exclude the intercept of the monotonic transform ($m(0)$) from the standardisation. We standardize $\hat{d}_{ij}$ such that

$$\sum_{i<j} \mathbb{I}(w_{ij} > 0)(\hat{d}_{ij}^2 - m(0)) = \sum_{i<j} \mathbb{I}(w_{ij} > 0)\delta_{ij}^2. \tag{4.3.10}$$

The reasoning for this is because we want

$$m^{-1}(\hat{d}_{ij}) - m(0)$$

to be an estimate of δ_{ij}, and we want $m(0)$ to be an estimate of the bias which is not contained within δ_{ij}. By excluding $m(0)$ from the standardization we attempt to keep the $m(0)$ in the same units at δ_{ij}. We also retain that the standardization avoids the trivial solution $m(x) = 0$ and it has no effect on Absolute MDS as $m(0) = 0$ in that case.

From a practical point of view, using a model which avoids the trivial solution is preferable. For example, using $m(x) = b + x,\ b > 0$ rather than $m(x) = b_0 + b_1 x \quad b_0, b_1 > 0$, the former would be preferable as we are not required to rescale the transformed dissimilarities. Proposing such an adaptation for more flexible transformations like the one that will be introduced in Section 4.3.3 is difficult. Hence, the optimization framework here is designed to work with a monotonic transformation which can cause the trivial solution. By using the same optimization framework for all models we keep the comparisons fair.

We follow a similar procedure to the one described by [Borg, 2005, p. 204]. However, instead of using a method known as majorization [de Leeuw, 2005] to optimise $\mathbf{X}$ we opt to use a gradient based algorithm BFGS. We acknowledge that using majorization in place of BFGS would likely converge faster. We choose to use BFGS because, as with auto-differentiation, no model specific derivations are required.

The optimization is carried out as follows:

1. Initiate the configuration $\mathbf{X}^{[0]}$ either randomly or via a simple MDS procedure.

   For example

   $$\mathbf{X}^{[0]} = \arg\min_{\mathbf{X}} \sum_{i<j} \left(\delta_{ij} - ||\mathbf{x}_i - \mathbf{x}_j||_2^2\right).$$

   Initiate $k = 0$.

2. **Optimize parameters for the transform** $m(x)$**.** Set

   $$\mathbf{b}^{[k+1]} = \arg\min_{\mathbf{b}} \sigma\left(\mathbf{X}^{[k]}, \mathbf{b}\right).$$

3. **Predict transformed dissimilarities**. Using the function $m(x)$ and the fitted

$\mathbf{b}^{[k+1]}$. Temporarily store

$$\hat{d}^*_{ij} = m\left(\delta_{ij}\right)$$

for all i, j. Rescale to obtain $\hat{d}_{ij}$:

$$\hat{d}_{ij} = \frac{s}{s_{\hat{d}}}(\hat{d}^*_{ij} - m(0)) + m(0), \tag{4.3.11}$$

where

$$s_{\hat{d}} = \sum_{i<j} \mathbb{I}(w_{ij} > 0)(\hat{d}^*_{ij} - m(0))^2 \quad \text{and } s = \sum_{i<j} \mathbb{I}(w_{ij} > 0)\delta^2_{ij}.$$

Use these standardised $\hat{d}_{ij}$ in computations for σ.

4. **Update $\mathbf{X}^{[k+1]}$.**

   (a) First compute the gradient $\mathbf{g}^{[k]} = \dfrac{\partial}{\partial \mathbf{X}_f}\sigma(\mathbf{X}^{[k]}_f, \mathbf{b}^{[k+1]})$ using auto differentiation.

   (b) Update the BFGS hessian estimate $\mathbf{H}$ based on $\mathbf{X}^{[k]}, \mathbf{X}^{[k-1]}, \mathbf{g}^{[k]}\mathbf{g}^{[k-1]}$.

   (c) Solve for $\mathbf{g}'$

   $$\mathbf{H}(\mathbf{g}') = \mathbf{g}.$$

   (d) **Line Search.** Select ρ:

   $$\rho = \arg\min_{\rho} \sigma(\mathbf{X}^{[k]} + \rho\mathbf{g}', \mathbf{b}^{[k+1]}), \tag{4.3.12}$$

   We approximate this optimization as a grid search over $\rho \in [0.001, 0.01, 0.1, 1, 10]$.

   (e) Set $\mathbf{X}^{[k+1]} = \mathbf{X}^{[k]} + \rho\mathbf{g}'$.

5. **Check convergence conditions**. If:

$$|\sigma(\mathbf{X}^{[k+1]}, \mathbf{b}^{[k+1]}) - \sigma(\mathbf{X}^{[k]}, \mathbf{b}^{[k]})| < \epsilon,$$

stop the algorithm, otherwise set $k = k + 1$ return to step 2.

Due to the presence of local minima, we fit each MDS model a number of times using different initial configurations. For one of the initial configurations we use Absolute MDS where the initial solution is chosen at random. The points in the remaining initial configurations are generated independently according to:

$$x_{i,j} \sim_{IID} \text{Uniform}(0, u) \text{ for } i \in \{1, ...S\},\ j \in 1, ..., p.$$

For each initial configuration, we fit the MDS model then we only keep the model which results in the lowest stress evaluation. The results are not sensitive to u as long as the resulting pairwise distances of the initialization are roughly on same order of magnitude as the average dissimilarity:

$$\left(\sum_{i<j} \mathbb{I}(w_{ij} > 0) \delta_{ij} \right) / \sum_{i<j} \mathbb{I}(w_{ij} > 0).$$

It is not discussed further in this work, but this step of starting from various points is essential, as MDS is known to have many local minima [Borg, 2005].

4.3.3 Monotonic Regression

One of the transforms we consider for MDS is a semi-parametric estimate for m in MDS, where the main requirement is that the function is monotonic. The goal of monotonic regression is to estimate a strictly increasing function on an interval $[L, U]$. We use monotonic spline regression to achieve this. The method takes a basis expansion of $x \in \mathbb{R}$ such that the resulting regression function is $f(x) = b_0 + \sum_{k=1}^{K} b_k I_k(x)$, where each component of the sum $b_k I_k(x)$ is a monotonically increasing

function. To achieve this monotonicity constraint, we construct basis functions $I_k(x)$ which are monotonically increasing by definition, and constrain $b_k > 0 \ \forall k$. Here we give a brief summary of how these basis functions are defined by Ramsay [1982].

The idea behind how we form the I-spline basis expansion is we take a non-negative basis function of degree d, then integrate that basis function. The integration of this nonnegative function results in a degree $d+1$ function which is monotonically increasing. Here we use B-splines as the nonnegative function.

Prior to setting up a B-spline of degree d, we must specify a knot vector $\mathbf{t}$. This requires defining k interior knot points. Often these knots are either equally spaced between L and U, or placed at equally spaced quantiles of the data. In this work, we always specify the internal knots at equally spaced quantiles. We set the knot vector to be

$$\mathbf{t} = \{\underbrace{L, \ldots, L}_{\text{length } d}, t_0, \ldots, t_k, \underbrace{U, \ldots, U}_{\text{length } d}\}.$$

Let $B_{i,d}$ denote the B-spline basis function of degree d. B-spline basis functions of degree 0, are defined in the following way

$$B_{i,0} = \left\{ \begin{array}{ll} 1, & \text{if } t_i \leq x < t_{i+1} \\ 0, & \text{otherwise} \end{array} \right\}.$$

$B_{i,0}$ results in a piecewise 0 degree polynomial, x taking the value 1 between one pair of knots. Higher order B-spline basis functions d are defined recursively as follows:

$$B_{i,d}(x) = \frac{x - t_i}{t_{i+d} - t_i} B_{i,d-1}(x) + \frac{t_{i+d+1} - x}{t_{i+d+1} - t_{i+1}} B_{i+1,d-1}(x).$$

Under this definition, B-splines are scaled to satisfy $\sum B_{i,d}(x) = 1$. For monotonic regression, a preferable scaling would be to have a basis function $M_{i,d}$ such that

$\int M_{i,d}(x)dx = 1, \quad \forall i$. The most common variant of such a spline is known as the M-splines which are a set of scaled B-splines. It can be shown that $\int B_{i,d}(x)dx = \frac{d}{t_{i+d}-t_i}$, leading to a transformation to M-splines:

$$M_{i,d}(x) = \frac{t_{i+d} - t_i}{d} B_{i,d}(x).$$

To form the degree d I-spline we simply integrate the M-spline of degree $d-1$ from L to the point being evaluated at x.

$$I_{i,d}(x) = \int_L^x M_{i,d-1}(x). \tag{4.3.13}$$

Ramsay [1988] gives a convenient formula to compute this using a weighted sum of M splines of degree d.

The reasoning behind choosing to integrate M-splines over B-splines is the interpretability of coefficients. The value of each I-spline basis function will have a maximum of 1 rather than the fairly arbitrary value integrating a B-spline would result in. An example of the M-, B- and I-spline basis functions are shown in Figure 4.3.1.

This is not the only approach in monotonic regression. Other potential approaches for monotonic regression we could consider are:

- Consider the function $f(x) = \sum_{j=1}^{K} \beta_j B_j(x)$, where $B_j(x)$ are B-splines, Wood [2017] shows that if β_j form a non-decreasing sequence, then $f'(x) \geq 0$. Thus, setting $\beta_j = \gamma_1 + \sum_{i=2}^{j} \exp(\gamma_i)$ results in $f'(x) > 0$, with γ_i unconstrained. We expect the fitting time and the results of this would be similar to the I-spline approach.

- Convex C splines which integrate I-splines to form basis functions that are both increasing and convex, resulting in a monotonic and convex function.

- The approach described in Ramsay [1998]. This involves modelling an unconstrained function w, say with B-splines or Gaussian process regression. Call D^m the operation of taking the derivative of order m for $m > 0$ and the integral $D^m f(x) = \int_L^x f(s)ds$ for $m = -1$. The function

$$f(x) = \beta_0 + \beta_1 D^{-1}\{\exp(D^{-1}w(x))\}$$

is shown to be a monotonic function under certain conditions on the function w. These conditions are not that restricting and are easy to enforce.

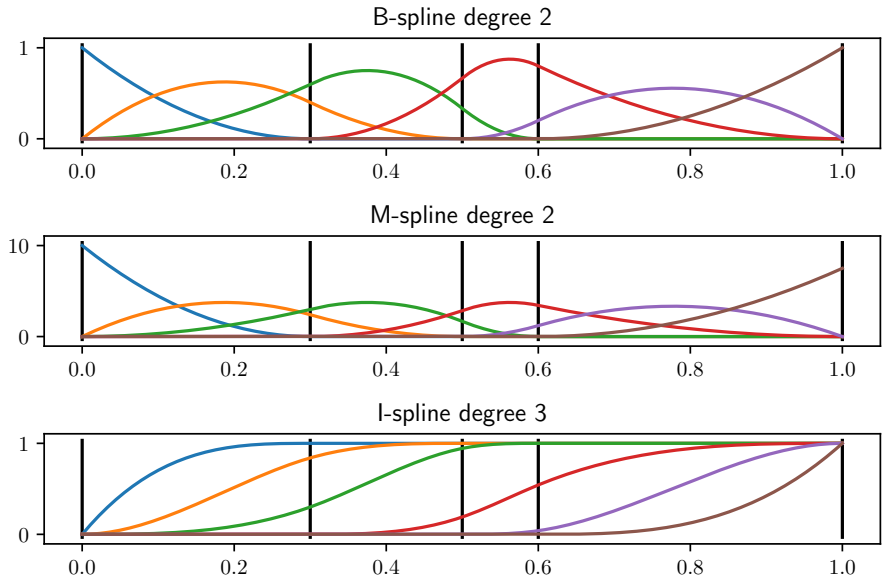

Figure 4.3.1: Example of the three types of basis functions explained in Section 4.3.3. Internal knots displayed as black lines defined at 0.3, 0.5, 0.6.

4.3.4 Procrustes Analysis

One of the core steps in MDS is when we calculate the distance between embedding points $\mathbf{x}_i \in \mathbb{R}^p \ \forall i$ as $d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$. One of the issues with MDS is that there are infinitely many equivalent solutions as the inferred distances will remain the same under rigid shifts, rotations and reflections. Let $\mathbf{c} \in \mathbb{R}^p$ be an arbitrary constant vector. Let $A \in \mathbb{R}^{p \times p}$ be a $p \times p$ matrix with the property that $||A|| = 1$. An example of such a matrix would be a rotation or reflection matrix. If we consider $\mathbf{y}_i = A\mathbf{x}_i + \mathbf{c} \ \forall i$ we get that

$$||\mathbf{y}_i - \mathbf{y}_j|| = ||\mathbf{x}_i - \mathbf{x}_j||.$$

Thus, there are infinitely many equivalent MDS embeddings, making comparisons between two solutions nontrivial.

We address this issue using Procrustes analysis. Suppose we have two solutions or coordinates for the objects $\mathbf{X}$ with rows $\{\mathbf{x}_i\}_{i=1}^S$ and $\mathbf{Y}$ with rows $\{\mathbf{y}_i\}_{i=1}^S$. We aim to find a rigid shift vector $\mathbf{c}$, rotation and reflection matrix $A \in \mathbb{R}^{p \times p}$ with norm 1 and scaling scalar $\rho \in \mathbb{R}$ such that

$$d(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{n} ||\rho A\mathbf{x}_i + \mathbf{c} - \mathbf{y}_i|| \tag{4.3.14}$$

is minimized. Note that the scaling scalar does change the distances inferred, however this is useful as it allows us to compare two embeddings with difference scales.

Now we state the results which we use and are in Chapter 5.2 Cox [2001]. To obtain the rotation and reflection matrix, we take the singular value decomposition:

$$\mathbf{Y}^T\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}.$$

Setting $\mathbf{A} = \mathbf{V}\mathbf{U}^T$ gives the minimizer of $d(\mathbf{X}, \mathbf{Y})$. The optimal scaling is

$$\rho = \text{tr}(\mathbf{X}\mathbf{A}\mathbf{Y}^T)/\text{tr}(\mathbf{X}\mathbf{X}^T)$$

and the rigid shift $\mathbf{b} = \bar{\mathbf{y}} - \rho\mathbf{A}^T\bar{\mathbf{x}}$, where $\bar{x} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$ and similarly $\bar{y} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i$.

In the following sections, we use these optimal values to report the Procrustes distance $d(\mathbf{X}, \mathbf{Y})$. However, we also use the rotated configurations as a plotting tool.

4.4 Simulation

The aim of this section is to generate a test bed for the algorithm where we know the ground truth. First, we introduce the simulation setup that we will be using. Second, we analyse the travel times from this simulation to empirically justify why transformed MDS is necessary in Section 4.4.1. We show sample MDS model fits on this simulation in Section 4.4.2. Finally, we investigate the sensitivity to these simulation parameters in Section 4.4.3 by changing the parameters and refitting MDS. Finally, we recap on our findings in Section 4.4.4.

We create a map and place the stations. We specify the size of a grid having the $x-$axis ranging between x_{min} and x_{max}, and $y-$axis ranging from y_{min} to y_{max}. We pick a sensible radius, r, we then place $n_{stations}$ stations over the grid by randomly sampling n points $s_i \in \mathbb{R}^2$, at least $3r$ away from each other using Euclidean distance. This placement is carried out by randomly sampling a point, then checking if it satisfies the $3r$ constraint. If the constraint is satisfied then it is placed, otherwise a new point is generated until we have $n_{stations}$ points.

To simulate the trajectories, we generate velocities from an Autoregressive (AR)

order 1 process. This is motivated by the common assumption that the background velocities of drifter motions approximately follow an Ornstein-Uhlenbeck process [Sykulski et al., 2016], after ignoring oscillatory effects. The discrete time version of this is the AR(1) process. We simulate the trajectories at integer timesteps using the following relationship:

$$\mathbf{y}_t = \mathbf{y}_{t-1} + \mathbf{v}_t \quad t \in \{i \in \mathbb{N} : 1 \leq i \leq \frac{T_{max}}{dt}\} \tag{4.4.1}$$

where $\mathbf{y}_t,\ \mathbf{v}_t \in \mathbb{R}^2$ and $\mathbf{v}_t$ is simulated as a bivariate AR process: $\mathbf{v}_t = \phi \mathbf{v}_{t-1} + \sqrt{dt}\boldsymbol{\epsilon}_t$, where $\boldsymbol{\epsilon}_t \sim N(\mathbf{0}, \mathbb{I}_2)$, $\phi \in (-1, 1)$, such that we have assumed independence and isotropy between the lat-lon components. The term dt specifies the true time increment between observations and T_{max} is the time-length of the simulation. To initialise we sample $\mathbf{y}_1$ from a 2d uniform distribution over the grid of interest, and set $\mathbf{v}_1 = [0, 0]$. We generate n_{traj} trajectories as above and use this as the dataset to apply the algorithm to. All travel time are reported in units of the number of discrete time steps $t \in \{0, 1, \ldots, \frac{T_{max}}{dt}\}$ for this section.

As a summary of the setup explained above, we provide a brief summary of the variables which define the simulation in Table 4.4.1. Throughout the section, we fix $x_{min}, y_{min}, x_{max}, y_{max}$ to $-5, -5, 5, 5$ respectively. Initially, we set the simulation parameters $n_{traj} = 10000,\ \phi = 0.3,\ T_{max} = 10,\ n_{stations} = 7,\ r = 1,\ dt = 0.1$. These parameters are chosen as they result in a dense travel time matrix. We note that this configuration does not result in a dataset similar to the application, as instead we have deliberately simulated to observe more direct travel times between stations, to check the method scales as it should. Specifically, each station pair has an average

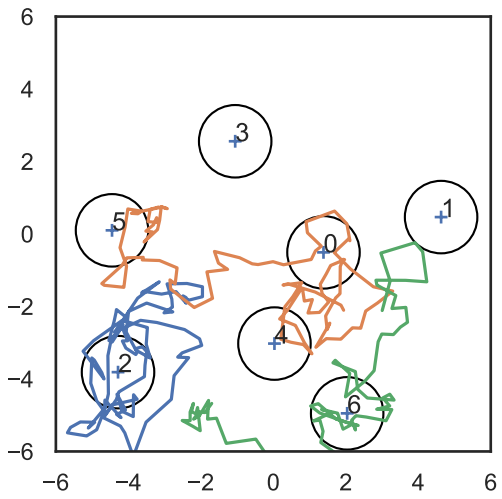

Figure 4.4.1: Station locations and three sample trajectories for the simulation described in Section 4.4. A radius of 1 is drawn around each station to show when the trajectory visits that station.

of 220.0 observations in the resulting simulation, whereas in the real data application the average count is 21.2.

4.4.1 Why Transformed MDS

Here we make the case of why transformed MDS is necessary. The first and easier point to motivate transformed MDS is that the intercept term is necessary. When calculating the travel times from station i to station j we observe when the object leaves a radius of r of station i then enters a radius r of station j. We argue that by having an intercept in the $m(x)$ function we allow the algorithm to model this directly.

The less obvious point to motivate is why we wish to consider nonlinear transforms. As the velocities are stationary, the optimal 2d embedding would be to recreate

n_{traj}	The number of trajectories.
ϕ	The AR Coefficient.
T_{max}	The maximum time hence selecting how long the trajectories are.
dt	The time increment between observations.
$n_{stations}$	The number of stations.
r	The radius around the stations.
$x_{min}, y_{min}, x_{max}, y_{max}$	The edges of the domain we are interested in.

Table 4.4.1: Decision variables for the simulation.

the original station locations from the pairwise travel times. We show the relationship of pairwise station distances resulting from the simulation and the raw travel times in Figure 4.4.2. It appears that a linear regression where a square root transform is applied to the estimated travel times has the best fit. However, we have no *a priori* reasoning to use this transformation. This is why we consider I-spline regression, as the regression may automatically learn such a transformation. We can then use the I-spline transformation to justify the choice of parametric transformation (e.g., square root) if the two functions look similar.

4.4.2 Dense Matrix Results

Here we use the dense travel time matrix which has no missing observations. We set $w_{ij} = 1 \; \forall i, j \; i \neq j$ as there is no missing data. We consider four different MDS models. When fitting interval MDS we consider both a linear transform $m(x) = b_0 + b_1 x$ and a square-root transform $m(x) = b_0 + b_1\sqrt{x}$. For ease of reference, we shorten the names

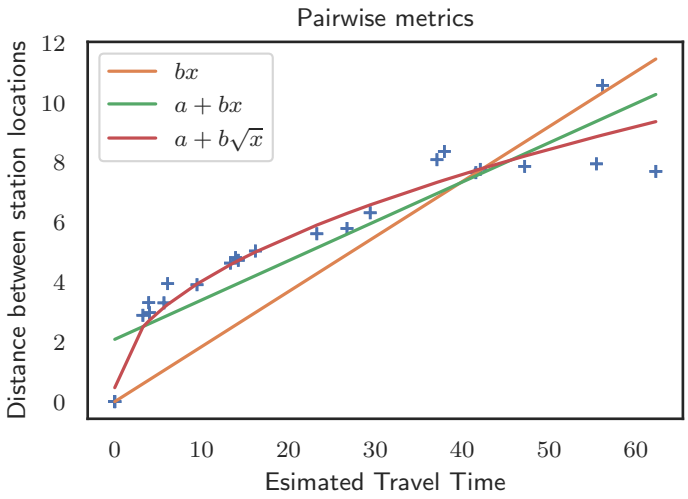

Figure 4.4.2: The estimated travel times from mode extraction on the x-axis and the pairwise euclidean distance between stations on the y-axis. The trajectory and station simulation setup is described in Section 4.4.

of each method list to the following four:

- **Spline:** I-Spline MDS
- **Interval:** Interval MDS (Linear transform)
- **S-Interval:** Interval MDS (Square root transform)
- **Absolute:** Absolute MDS $m(x) = x$

When fitting Spline MDS, a decision needs to be made about the number of internal knots and the degree of the spline. We fix the degree of the spline to 4, then focus on choosing the number of knots. To decide on the number of knots, we fitted an MDS model with 0 to 8 internal knots, then selected the number of internal knots which resulted in the MDS model having the lowest Procrustes distance to the station locations. This resulted in 1 internal knot placed at the median.

We fit an MDS model for each model type. We start the optimization from 20 different initializations, setting $u = 20$ as the maximum of the uniform distribution. We show the MDS solution for each of these configurations in Figure 4.4.3. Generally, every model does similarly. If we compare the Procrustes distance to the original station locations, S-Interval does the best as shown in Table 4.4.2. Absolute has the worst fit, which suggests that modelling the bias improves the fit as expected.

The transformation plots shown in the bottom plot of Figure 4.4.3 may appear very different at first glance, but the main difference is the intercepts, as they are excluded from the standardization. This scaling makes the comparison of S-Interval and Spline visually difficult. When we scale the monotonic functions differently (not shown), the transforms from Spline MDS and S-Interval MDS look very similar within the range of the data.

Model Name	Average Proscrustes	$m(0)$
S-Interval	0.21	3.28
Spline	0.22	19.12
Interval	0.25	24.03
Absolute	0.33	0

Table 4.4.2: Average Proscrustes distance from each model for the fits shown in Figure 4.4.3. We also show the estimates of $m(0)$

We note that the value of $m(0)$ for Spline MDS and Interval MDS is higher than S-Interval. The estimate is high for Interval MDS because the linear model is not as good of a fit, and we can tell this because the ability for Interval MDS to recreate

the station locations is slightly worse than S-Interval and Spline MDS in Table 4.4.2. The high estimate from Spline MDS is due to its lack of extrapolation ability. The I-Spline component of the model predicts 0 below the minimum of the δ_{ij} data, so below the minimum of the data we predict a constant. If we instead extrapolated in a more natural way such as to preserve the gradient of the monotonic function at the minimum of the data, we would get a more sensible estimate. We carried out a naive linear extrapolation to the Spline MDS function which resulted in an estimate of $m(0)$ as 10.07, which is still higher than the estimate from the S-Interval MDS model but getting closer.

4.4.3 Sensitivity to parameters

We now run a sensitivity analysis to n_{traj}, r, $n_{stations}$ and ϕ using the four MDS models. $dt = 0.1$ and $T_{max} = 10$ are left fixed. We test an equally spaced grid of 10 points for each parameter, the minima and maxima of this grid are listed in Table 4.4.3. To measure the sensitivity, we fix the other three simulation parameters at the default values which are shown in the Table 4.4.3, then vary the parameter of interest. For each simulation parameter set, we obtain 5 realisations.We randomly generate the station locations each time.

The default starting point shown in Table 4.4.3 was chosen such that the parameters result in roughly 15% sparsity in the symmetric matrix. 15% sparsity is close to the data sparsity we have in the applied example. Through some trial and error, we found the default combination shown in Table 4.4.3 achieves around 15% sparsity on average.

We record the average Procrustes distance between the stations and the inferred configuration to assess the goodness of fit of the configurations. Let $\tilde{\mathbf{x}}_i$ denote the optimally transformed point to match the set of station locations $\mathbf{s}_i$, the metric we measure is the average Procrustes distance defined as follows:

$$P = \frac{1}{S}\sum_{i=1}^{S} ||\tilde{\mathbf{x}}_i - \mathbf{s}_i||.$$

$P = 0$ implies the embedding matches the station locations exactly, where we normalize by S to allow us to compare the metric when varying the number of stations. For the remainder of this section, we will refer to this metric as the score of an embedding.

As previously mentioned, the MDS optimization procedure typically has many local minima. As an attempt to avoid these local minima, we fit each MDS model using 20 different initializations and retain the fit which has the minimum stress $\sigma(\mathbf{X}, \mathbf{b})$ at the optimal value. We fix the number of internal knots for Spline MDS as 1 as that was the best for the dense matrix results.

Parameter	Minimum of range	Maximum of range	Default
n_{traj}	1000	6000	1500
r	0.3	1.3	1
ϕ	-0.9	0.9	0.3
$n_{stations}$	4	13	7

Table 4.4.3: Default parameters and the ranges used for the sensitivity analysis.

The results of the sensitivity analysis are displayed in Figure 4.4.4. We can see that generally Spline, S-Interval, and Interval all follow the same trend in contrast to

Absolute MDS. Absolute MDS tends to do better than the other models when the sparsity is relatively high (at least 15%). Whereas, the models with transformations tend to do better at low sparsity.

To investigate the performance of each model as we alter sparsity, we reuse the sensitivity simulation results. Over all 200 (40 parameter sets, 5 replications for each set) simulated datasets, we extract the sparsity and the score for each model. It is reasonable to expect that increasing sparsity will generally result in a higher score (i.e., worse fit), hence we fit an I-Spline monotonic regression for each model and plot this fit in Figure 4.4.5. We also provide a rolling window estimate to show the relationship is close to monotonic even when modelled without the constraint. It can be seen that at around 14% sparsity it becomes better to use Absolute MDS than S-Interval or Interval MDS. Around 16% sparsity it becomes better to use Absolute MDS than Spline MDS. For lower than 13% sparsity generally S-Interval and Spline MDS do very similarly and better than Interval MDS.

4.4.4 Summary of Simulation Findings

In this simulation section, we first proposed a framework to simulate the trajectories in. We then applied the methodology introduced in Sections 4.2 to extract a travel times matrix, which potentially has sparsity. Then we use the methodology from 4.3 to recreate the original station locations seeing only the travel times. We now give a brief summary of the main findings discovered from the simulation.

- In Section 4.4.1 we gave empirical evidence of why we consider nonlinear trans-

formations. In particular, we showed that the relationship between the modal travel times and the distance between physical station locations is nonlinear.

- In Section 4.4.2 we found that when using a dense travel time matrix, transformed MDS does better than Absolute MDS. We found that using nonlinear transforms are beneficial.

- In Section 4.4.3, we varied the parameters of the simulation to introduce sparsity in a realistic way. We found that when sparsity gets higher than around about 14% absolute MDS does better than transformed MDS for the examples considered. When sparsity is lower than 14% the transformed MDS model generally does better than the Absolute MDS model.

We use these findings to justify decisions made in the application to the real dataset in the next section.

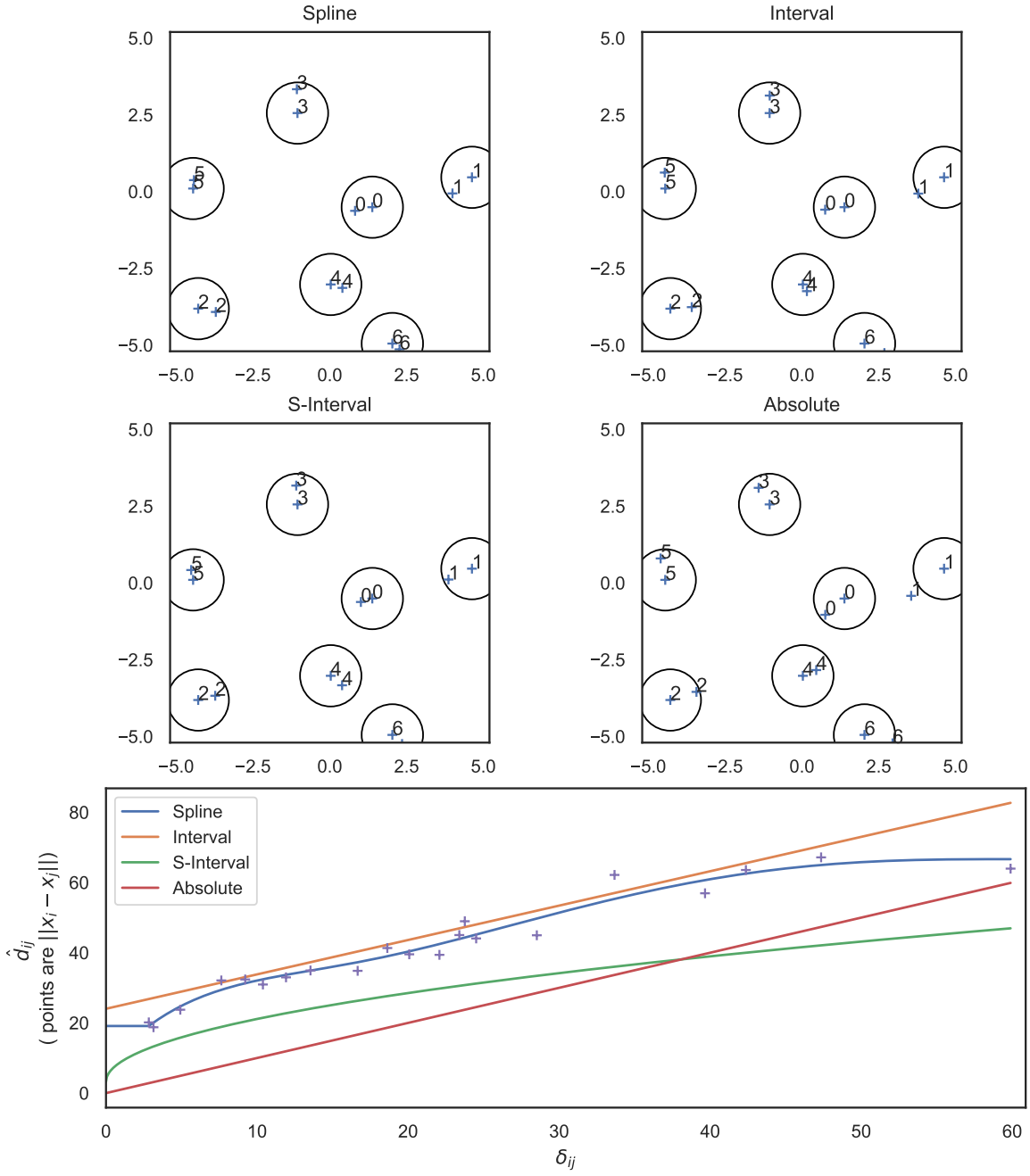

Figure 4.4.3: The embedding from 4 MDS configurations using a dense travel time matrix. The transformation functions are plotted in the bottom panel.

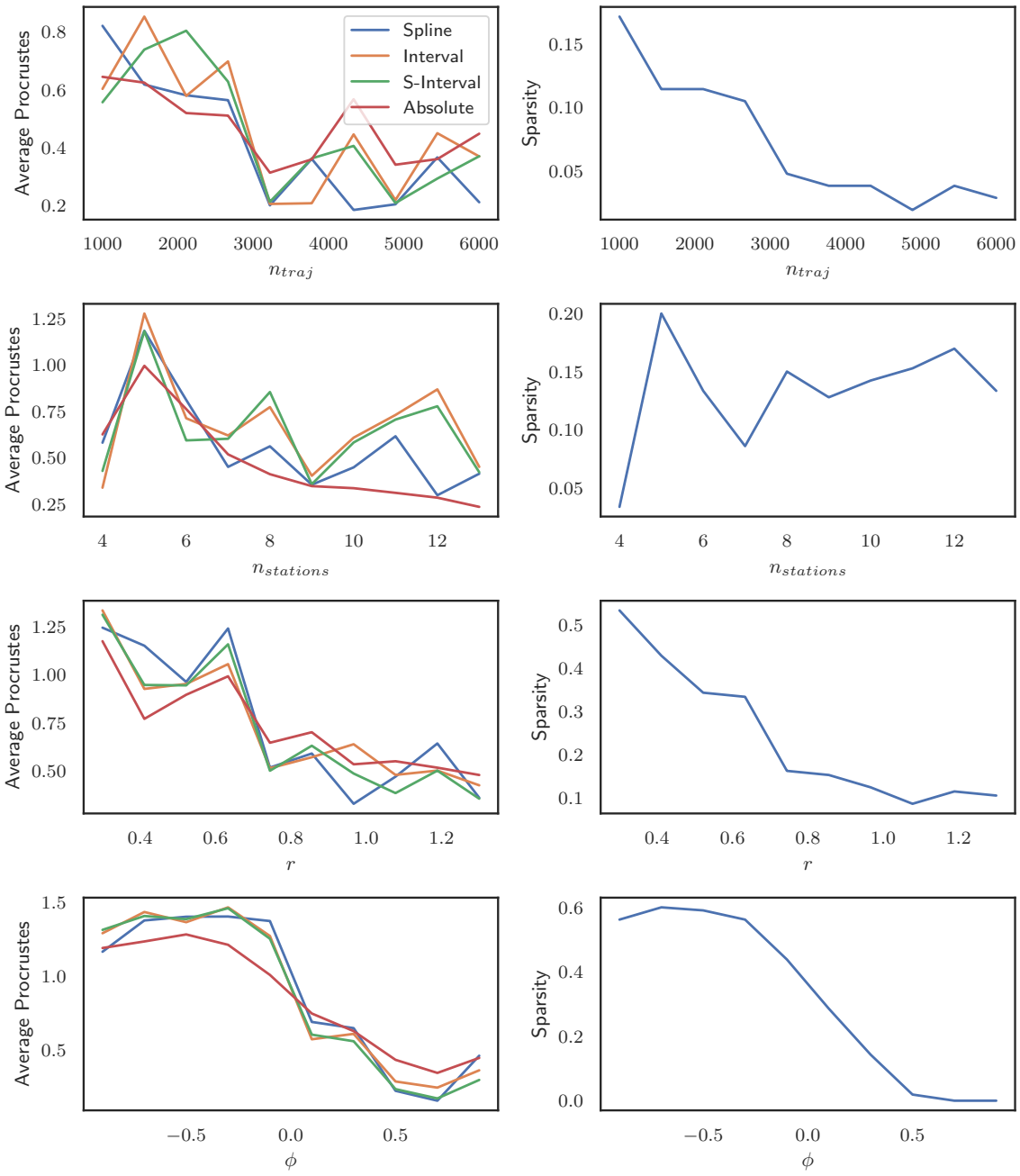

Figure 4.4.4: Sensitivity to each parameter for the ranges specified in Table 4.4.3. Each point on the line is the mean of 5 replications.

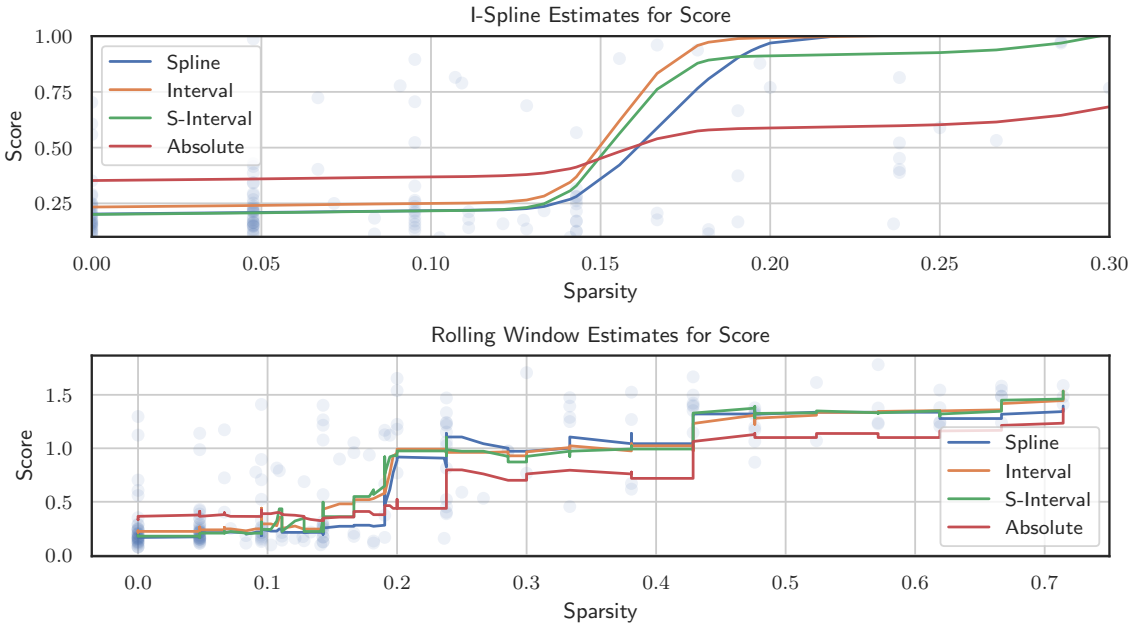

Figure 4.4.5: I-Spline and rolling window estimates to predict the score defined in section 4.4.3 given sparsity of the input dissimilarity matrix. In the I-Spline model we restrict the x-axis to highlight where Absolute MDS starts to do better. The partially transparent points are the scores from the Spline MDS model. The I-Spline models are fit with a mean absolute error objective and 20 internal knots. The rolling window estimate uses a median to summarize and has a window with 30 data points. The objective of I-Spline regression and summary method of the rolling window were chosen to be to lessen the influence of outliers.

4.5 Application

The application considered here is to estimate the dissimilarities between sampling points of the Tara Oceans Expedition [Pesant et al., 2015]. We call these sampling points the stations. We focus on an area in the North Atlantic Ocean. This area has 8 Tara stations and the locations of these are shown in Figure 4.2.1. The currents are shown in Figure 4.5.1. The strongest current seen is the Gulf Stream, which stations 0, 1 are on and 2 and 3 are close to.

We choose to use $r = 100km$ as the radius. This radius was chosen by trial and error looking at what values for the radius resulted around 10% sparsity. This radius choice resulted in 3 missing travel times which translates to 10.7% sparsity in the travel times. This target sparsity is informed by the simulation results presented in Section 4.4.3 as this is around the point where the results are better using transformed MDS.

A sample of kernel density estimates which are used to obtain δ_{ij} are shown in Figure 4.5.2. We observe that generally these density estimates decay to zero density slowly, often outside the maximum of the data, however this seems to be a good feature. For example, in the plots from station 1 to station 3 and station 4 to station 5, we see relatively large outliers around 170 and 300 respectively. In theory, this could happen for any station pair and the heavy tailed nature of the density estimates accounts for this. In this application, all we use is the modes of these distributions, however we discuss a potential follow-on work on better modelling of these distributions in Section 4.6.2.

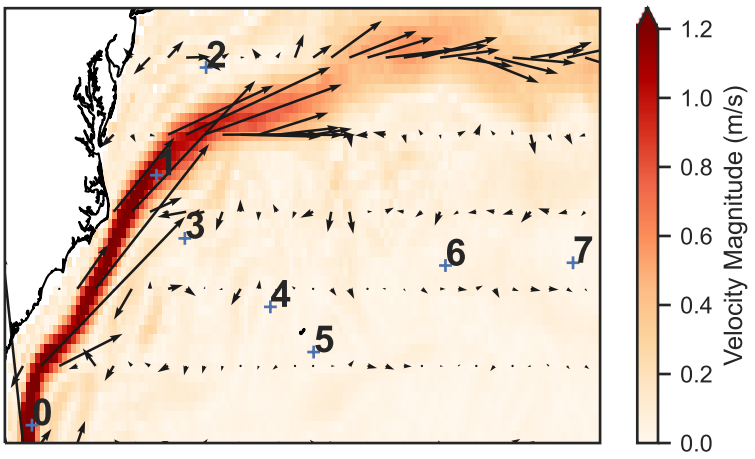

Figure 4.5.1: The stations plotted on a near surface mean current map supplied by Laurindo et al. [2017]. The locations are shown in Figure 4.2.1 with a sample trajectory also.

For Spline MDS again we need to choose the degree and number of knots. We choose to use degree 4 I-Splines as they result in a smooth function. We control the complexity by choosing the number of internal knots. We fit I-Spline MDS for a range of number of internal knots from 1 to 8 and inspect the transform for each fit. The figure is not included, but all transforms take a very similar shape, suggesting the results are not very sensitive to this parameter. We choose to take 4 internal knots as it is the last value where the line has no visible wiggles; it looks almost identical to the function using 3 internal knots.

We choose to represent the travel times in days for MDS. To choose the initializations for MDS we set the maximum of the uniform distribution to 20. Similar to the simulation we generate 20 different initializations and retain the fit which has the minimum stress after optimization.

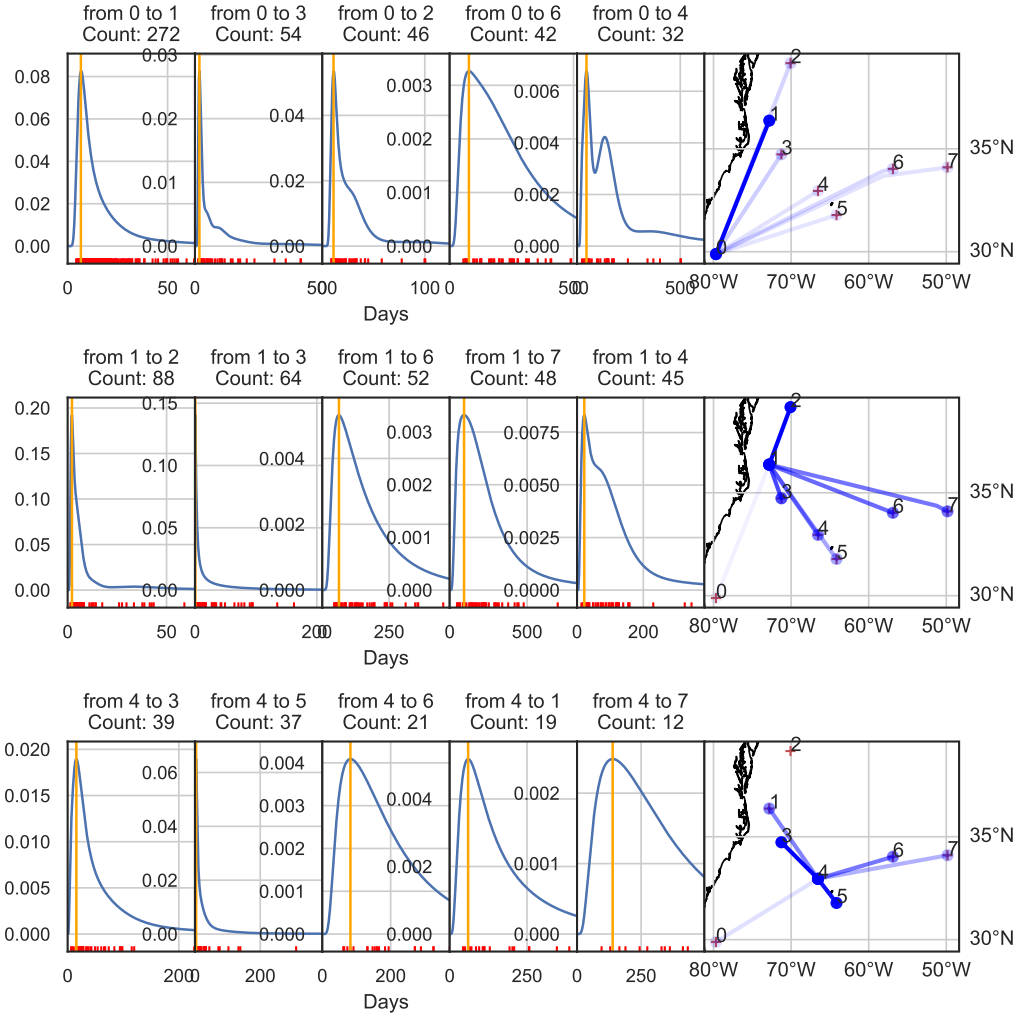

Figure 4.5.2: Positive constrained KDE fits to the travel times data originating from stations 0, 1 and 4. The numbers on the top of a KDE plot indicates the origin and destination station, alongside how many drifters we observed for this origin-destination pair. We plot the travel time data as red ticks on each plot. The mode of the KDE is indicated in orange. The 5 pairs with the highest number of observations are shown in each row. Plots on the right are provided to display where these connections are going to. Darker lines indicate more observations, lighter for fewer observations.

To match the output configuration to the stations, we conducted a Procrustes analysis on the MDS solution to rotate them to the longitudes and latitudes numbers of the original stations as the target. We note that a more accurate Procrustes analysis could be carried out by minimising geodetic distance between the embedding points and the stations rather than Euclidean distance, however we are only using the rotation as a visual tool hence it would not be worth the extra complexity. The rotated solutions of each model are shown in Figure 4.5.3.

The transformation functions $m(x)$ for Spline, Interval and S-Interval MDS are all very similar; resulting in very similar looking configurations. There is a slightly higher agreement between S-Interval and Spline MDS; likely meaning that S-Interval is the best parametric model to use as Spline MDS learns a similar shape. The main difference between the models are the estimates for $m(0)$. Spline MDS and Interval MDS estimate $m(0)$ around 40 days whereas S-Interval estimates $m(0)$ around 10 days. As the results are relatively similar, we just analyze the Spline MDS fit.

Unlike the simulation example in this case the MDS embeddings do not appear in locations close to their original locations. For example, station 0 is distant from the original location. This is expected due to the currents. The MDS procedure correctly shrinks the distance in places where strong currents are present, such as going from 0 to 1, 2 and 3; all of these stations are located on or near the Gulf Stream current shown in Figure 4.5.1. These are placed considerably closer in the embedding than in the original station locations. Whereas stations 4, 5, 6 and 7 remain spread out as the currents are not as strong.

In Figure 4.5.4 a) we show the Spline MDS inferred travel times from Equation

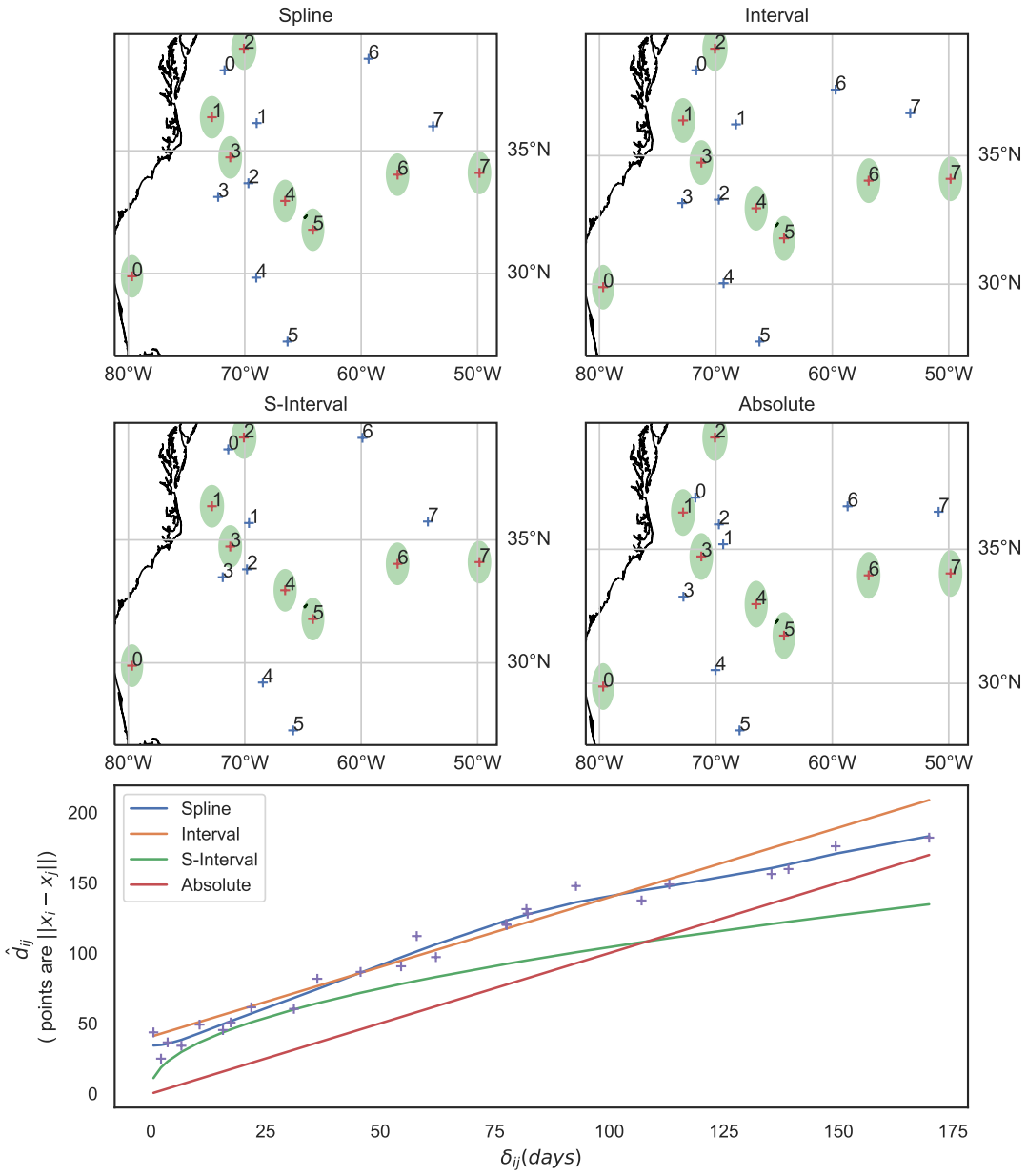

Figure 4.5.3: Results from each of the MDS models in 2 dimensions. The fitted model for $m(x)$ is shown for each model on the bottom plots. the blue '+' points are the embedding distances inferred from Spline MDS, excluding points which are missing in the original matrix.

(4.3.8) and the original travel times. Overall, Spline MDS recreates the original distances very well. This is why we do not consider going to $p = 3$ or higher. The extra complexity would not be worth the loss of interpretability. There is one outlier which is a travel time estimate between 3 and 6. The original estimate of $\delta_{3,6}$ was 235.5 days which resulted from 9 drifter trajectories, so it was likely unreliable. The travel time $\delta_{3,7}$ was 169.8. We expect that the travel time between stations 3 and 7 would be longer than the travel time between stations 3 and 6. The original $\mathbf{D}$ did not have this feature; the estimates distances from MDS do. Hence, MDS effectively smoothed out this outlier between stations 3 and 6.

4.5.1 Comparison with Most Likely Paths

We now analyze the completed Spline MDS's travel times with comparison to the most likely path travel times from Chapter 3. In this application, the three missing distances were between (2,3); (2,4) and (2,5). The estimates we obtain with Equation (4.3.8) using Spline MDS are 34.82, 40.75 and 67.7 days for each pair respectively. We show these results in Figure 4.5.4b) and c). Although positively correlated, the two methods from each chapter provide travel times of a very different scale, particularly at larger travel times.

The Most Likely Path (MLP) methodology has travel times as high as 1962 days; whereas Spline MDS has a maximum estimate of 201 days. There are numerous possible reasons for this. First, if we consider the shorter travel times we estimated in this work (< 75 days) and compare them to the MLP travel times, the scales are comparable; the difference in scales gets considerably worse with longer travel times.

This could be due to censorship in the data. Drifters live on average 410 days hence, a journey with a larger travel time is less likely to be observed. We are restricted to observing shorter trajectories. This censorship will have the effect of making the estimates of the modal travel time smaller. The MLP methodology should not be effected by this as it works with 5 day snap shots of drifter trajectories.

Second, these two travel times are fundamentally estimating different phenomena so we do not expect them to match. In Chapter 3, we take the most likely path fix, then estimate the expected travel time of that pathway given that the object follows the path. Here, we consider the mode of the travel time distribution of all observed drifters. They are very similar concepts, however, they are not the exact same solution.

Finally, the scale of the MLP travel times is largely dependant on the resolution of the discretization as shown in the supplementary information for Chapter 4. A broader scale has the effect of smoothing out currents in space. In this case, we used the resolution 3 grid system which has hexagons around the same size as a $100km \times 100km$ square which likely has the effect of smoothing out currents like the gulf stream. Using a finer scale (resolution 4) means the travel times are often smaller. This can be observed from the supplementary results presented with Chapter 3 by inspecting Figure B.H.1 in isolation, or by comparing Figures B.H.3 and 3.5.6. These figures show that using resolution 3 estimates are as much as $3\times$ as large as resolution 4 estimates. Only minor smoothing occurs with the MDS distances as we consider the radius around the stations.

One point of note is the outlierness of observations which were treated as missing in

MDS. In particular, station pairs 2 to 3, 2 to 4 and 2 to 5 which are labelled in Figure 4.5.4 c). The estimates from Spline MDS in all three of these cases are extremely low, whereas the most likely path methodology puts these times much higher. Potential reasons for this disagreement are discussed in Section 4.6.2, the main point being that the Gulf Stream acts as a barrier.

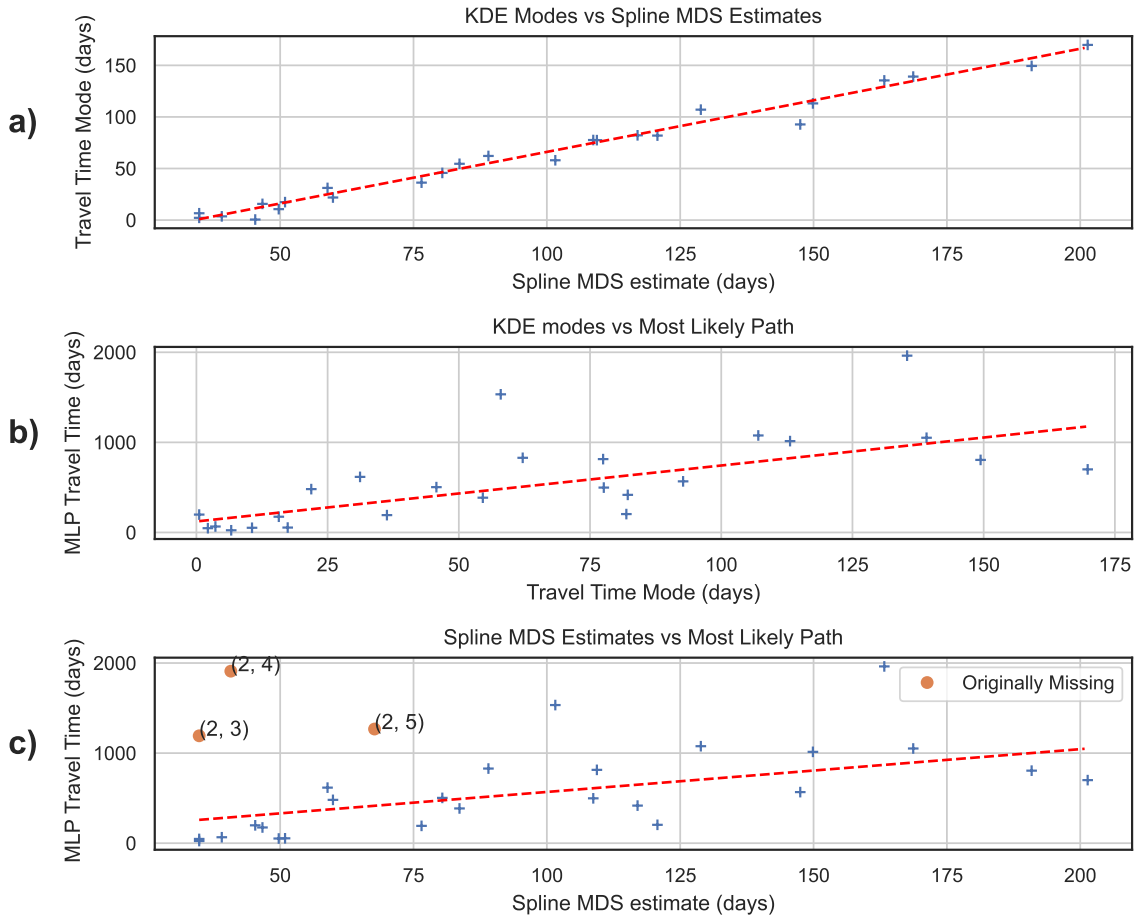

Figure 4.5.4: Plots comparing results. a) Compares the data contained in $\mathbf{D}$ to the travel time estimates produced by Spline MDS from Equation 4.3.8. b) Compares the data in $\mathbf{D}$ to the expected travel times from the Most Likely Path methodology from Chapter 3. **c)** Compares the Spline MDS travel time estimates to the Most Likely Path methodology, the points which were missing in $\mathbf{D}$ are shown in orange and labelled. The best fit lines shown in red are fit to all plotted points by linear regression with a Huber loss function.

4.6 Discussion and Future Work

4.6.1 Summary of Findings

In this chapter, we have proposed a framework to first extract the modal travel time and then model the travel times using MDS to fill in missing values and smooth outliers. We also proposed an estimate to correct bias in the modal travel times using MDS. We analyzed the model fits and sensitivity to sparsity on a simulated example in Section 4.4 using four variants of the model (Spline, Interval, S-Interval and Absolute). We then fit the same four models to trajectories from the Global Drifter Program in Section 4.5 and compared with the work from Chapter 3.

In both the application and the simulation, Spline MDS predicted a transformation similar to S-Interval MDS transformation. We also found that in the simulation experiment that S-Interval typically did better by comparing the results with the true station locations. Therefore, in application, using S-Interval MDS may be preferable as it has fewer parameters than Spline MDS and typically gives very similar results. However, Spline MDS is useful as an exploratory tool as it helps justify the choice of the square-root transform used in S-Interval. In the simulation, we found that there was a benefit of using Spline MDS over Absolute MDS up until the sparsity in $\mathbf{D}$ is around 16% sparsity in the travel times matrix.

In Section 4.5.1, we compared the results which we obtained in this chapter to the results from Chapter 3. In particular, we found that there seemed to be a positive correlation in travel times found between the two methods. However, the scales of the observations were typically higher with the MLP methodology from Chapter 3, and

we gave potential explanations for this difference in scales. We also noted that the MDS based method fails to account for the Gulf Stream which acts as a boundary current, whereas the MLP methodology gives high travel times for stations which are located on either side of the boundary current. We suggest a potential fix for this in the MDS method as future work in Section 4.6.2.

4.6.2 Future Work

There are many ways in which the work in this chapter could be improved. This work was an initial study and an introduction to how one can apply the MDS framework to trajectory data. Here we discuss extensions to the models which could be carried out to increase the inferential capabilities of the method.

One could remove the point where a mode estimate is taken. Specifically, rather than having dissimilarities δ_{ij} one could directly use the observed travel times stored in M^{ij}. A stress function could be used in a similar fashion to repeated observation MDS, where multiple observations exist for each object. For an MSE-like stress for absolute MDS, the objective would take the form:

$$\sum_{i<j} \sum_{k=1}^{|M^{ij}|} (M_k^{ij} - ||\mathbf{x}_i - \mathbf{x}_j||)^2.$$

This naturally gives a higher weight to pairs with more observations as such pairs make a larger contribution to the objective. We emphasise that this would not be a difficult adaptation, it was excluded for a number of reasons: (1) it makes result interpretation more difficult as we do not observe a single δ_{ij}, (2) the question of which form of stress to use becomes more important as we need to give less weight to

outliers, e.g., do we use an MSE-like stress as above or a robust alternative.

The monotonic models assumed that the bias introduced by the radius was constant for all stations. The plot in Figure 4.5.1 clearly shows that the current is not the same around all stations. We could relax this assumption by modelling the bias as a varying effect, for example, by changing the model such that

$$\hat{d}_{ij} = b_{0,i} + b_{0,j} + b_1\delta_{ij},$$

where $b_{0,i}$ and $b_{0,j}$ are estimates of the bias introduced by the radius around stations i and j respectively. This would add an extra $S - 1$ parameters to the model which may not be desirable, hence, we could instead consider fitting these bias terms as a function of the velocity field surround the station such as

$$b_{0,i} = b_{0,0} + b_{0,1}|\mathbf{v}_i|,$$

where $|\mathbf{v}_i|$ would be an estimate of the average magnitude of ocean current velocity in the neighbourhood of station i. Such a model would only add one extra parameter. The magnitude of the ocean current velocities could be easily extracted from a product such as the climatology derived from the GDP [Laurindo et al., 2017] used to create Figure 4.5.1.

As we only considered embeddings in $\mathbb{R}^2$ in this work, an improvement could be seen by constraining the two-dimensional (2d) embedding to be somewhat sensible. Rather than using multidimensional scaling to create the embedding points in an arbitrary frame of reference, one could use a 2d function to transform longitude and latitude. Such models have previously been used to model non-stationary spatial

dependence [Sampson and Guttorp, 1992, Richards and Wadsworth, 2021]. By constraining this 2d function, one could ensure the end results are in a similar rotation to the original station locations. We would expect the model to stretch out areas with weaker currents and shrink areas which have stronger currents.

On a separate line of work, the pathways that drifters take between these stations themselves are often of interest. We could instead only focus on a single pair of stations and model the travel times shown in Section 4.5.2 directly using a mixture distribution, and then identifying how many mixture components in the distribution exist. It would be of even greater interest to cluster the trajectories directly. These models could be used to answer many interesting questions in a direct way such as:

- How many clusters exist?
- What does the central trajectory of each cluster look like?
- Is the travel time over these clusters significantly different?
- Do drogued and undrogued drifters tend to take different pathways?

The development of such a method could facilitate the creation of a tool where a practitioner could input two locations and a visualisation answering all these questions directly from the data. This would complement and generalize regional analyses in the literature such as the pathways studied in Drouin et al. [2022] in the North Atlantic.

Travel time near boundary currents

The ability of our method to accurately fill in missing distances can be very poor. One issue present in the application is the relatively short distance for pairs $(2, 3)$; $(2, 4)$;

and $(2, 5)$ in comparison to the MLP methodology. In Figure 4.5.2 we see that 2 and 3 are relatively close in all four configurations. The actual travel time between these two sites should be very large as they are situated on different sides of the Gulf Stream which acts as an oceanographic barrier. The MDS models place these stations close as there is a relatively short travel time from both stations 1 and 0 to both stations 2 and 3. Here we discuss two reasons for this downfall: a) We do not account for the non-missing at random nature of these travel times and b) The model produces a distance matrix. We now discuss both of these issues in more detail.

The application to complete the distance matrices here assumes that the missing distances are missing at random. However, there is a causal link between the number of observations and the travel time. Observations with a large travel time are less likely to be seen, hence these missing distances inform us that travel time is large. Addressing this issue by modelling the not missing at random nature of the data in the methodology may result in a more sensible distance completion between stations 2 and 3.

The method introduced here relies on ultimately creating an $S \times S$ travel time matrix with entries t_{ij}. One of the oversights in this chapter is the question of if the travel time matrix should be a Euclidean distance matrix (which MDS produces). That is that the following properties hold for all $i,\ j,\ k \in \{1, \ldots, S\}$:

1. The matrix is symmetric $t_{ij} = t_{ji}$.

2. The triangle inequality holds $t_{ij} \leq t_{ik} + t_{kj}$.

3. The off-diagonals are all positive $i \neq j \Rightarrow t_{ij} \geq 0$.

4. The diagonals are zero $i = j \Rightarrow t_{ij} = 0$.

We now go through each of the four properties and argue for or against each property being realistic in practical applications.

The goal of this chapter was to build a matrix which could be used as a measure of oceanographic separation, which in turn could be used to e.g. investigate the correlations between genetic species at various sampling stations. This application to genetic species typically uses a symmetric matrix, hence, we aim to produce a symmetric matrix containing the more likely path of the two directions. Therefore, the first property holding seems reasonable. The third and fourth property are clearly desirable in practice as a travel time should not be zero if the two locations are not the same, and the travel time from something to itself should be zero.

The point which is not as realistic is if the resulting matrix should satisfy the triangle inequality. To see this, we make the argument using stations 1, 2 and 3 in the application. If we want station 1 to be close to both stations 2 and 3 say $t_{1,2} = t_{1,3} = 0.5$ years. However, we want 2 and 3 to be distant as they are on either side of the barrier say $t_{2,3} = x$ where x is some large number, then we must have that $t_{2,3} \leq t_{2,1} + t_{1,3} \Rightarrow x \leq 1$, therefore with the triangle inequality satisfied we put a relatively low upper-bound on this travel time.

A potential solution to this would be to reformulate the model to account for anomalies which would automatically detect barriers (e.g., land or a strong current). The distances would take the form $t_{ij} = \hat{t}_{ij} + \hat{a}_{ij}$ Where the matrix containing $\hat{t}_{ij}$ is estimated from the MDS solution proposed in this chapter, resulting in the matrix

being a Euclidean distance matrix. $\hat{a}_{ij}$ estimates an anomaly component. When fitting the model we would heavily penalise non-zero $\hat{a}_{ij}$ in the optimization. The matrix containing $\hat{a}_{ij}$ would only be constrained to be symmetric with a zero diagonal and positive elements. A non-zero component in the $\hat{a}_{ij}$ matrix would signify an anomaly where the triangle inequality is not met; suggesting a barrier of some sort is present.

Under this reformulation it would also be natural to account for the missingness of observations. This could be done by changing the MDS objective function from stress to a likelihood based objective. Then we could propose that a missing observation in the matrix is due to censorship in how long the drifters live. Such an MDS objective would mean stations which have a missing travel time are placed at large distances from one another in the embedding.

Chapter 5

Conclusions

This thesis has developed and tested models which have been proposed to answer challenging questions using the Global Drifter Program data. The main issue which we aimed to make progress on in Chapter 2 is to show that modelling the joint distribution of northward and eastward velocities can lead to a sensible and informative model. We have also provided methodology showing that implementation is feasible and generalisable. Chapters 3 and 4 then provide methods to answer the question how long does it take to get from point A to point B, where A and B are any two points on the surface of the Ocean.

In Chapter 2, we expanded an existing framework known as NGBoost to allow for the prediction of multivariate outcomes using boosting. We drew comparisons between the algorithm when it uses the natural gradient and the ordinary gradient, from this we concluded that the natural gradient was crucial to the NGBoost fit and efficiency. Additionally, we compared with the alternative neural network approach which was proposed by Williams [1996] and found that NGBoost performed better in

most cases considered.

As emphasised in the introduction of Chapter 2, the benefits of probabilistic regression generally come from the decision making aspect. The model output from Chapter 2 has many potential uses which could be independent works in themselves. The prediction provides a measurement of the likelihood of an observation. This could be used to identify outliers by finding observations with a low likelihood under the prediction.

In the application considered, further investigation of these outliers could be used to identify faults or changes in either the drifters or the remotely-sensed datasets. For example, in the mid-2000s there was a problem with the drogue sensors which resulted in drifter observations being recorded as drogued for much longer than they should have been. The work by Lumpkin et al. [2013] proposed an automatic drogue detection method to fix this. The method, which is essentially a model comparison between an undrogued model, where the effect of wind on the observed velocity is higher, and a drogued model. Similarly to Lumpkin et al. [2013], the predictions from Chapter 2 could be used to identify the point at which the motions are most likely to start following undrogued behavior.

In Chapter 3, we propose an algorithm based on the established Markov Chain model to find the most likely path between two points in the ocean. We then proposed the use of rotations and bootstrap to obtain the discretization error and estimation error. We have shown example pathways for a set of real locations and found that the results match previous studies based on less generalisable methodology. Our method has since been used to identify the role that marine currents have in plankton genomic

differentiation [Laso-Jadart et al., 2021].

In Chapter 4, we proposed a more direct approach to the most likely path problem using density estimation and multidimensional scaling (MDS). In particular, we focused on the ability of MDS to fill in missing distances. We found that this worked well on a simulated example, however when using real data the model needs to be further adapted. Namely, in the real data example, the MDS model recreated the observed distances well, however, the estimates for the missing distances seemed to be somewhat spurious.

Although it may initially seems like 10.6 million drifter days provides sufficiently enough points, there are in fact ample sampling gaps in both space and time to prevent learning a spatial-temporal model directly. In Chapter 2, we worked around this by using external covariate information which explains some of the spatial and temporal variations. In Chapter 3 we justified the stationary-in-time assumption as "we aim to provide a global view and a simple general concept explaining the pattern of potential pathways and travel times;" this statement also holds for the work in Chapter 4. In Section 5.1.3 we shall propose a future area of research which would allow the Markov models to vary in time.

This thesis has also resulted in software contributions. The work from Chapter 2 has been added to the NGBoost open-source Python package to make the fitting of similar models a matter of a few lines of code. The most likely path work from Chapter 3 is available for future use through the package DriftMLP. We include a default transition matrix which is a product of the Global Drifter Program data, so one can compute pathways without even downloading the trajectory data. These software

contributions make it straightforward for one to utilise the methods for future applications. Moreover, as all of the software is readily available on GitHub, this facilitates the rapid development of new similar methodologies within the same framework.

5.1 Future Work

5.1.1 Gradient Boosting Varying Coefficient Model

One of the areas of future work is to make the model presented in Chapter 2 more interpretable. Non-parametric or machine learning models are often too flexible which allow them to fit the data well, but we lose interpretability of the model. A promising area of future work would be to specify a parametric linear model for the same application as in Chapter 2, but allow the coefficients of the parametric model to vary non-parametrically. Thus, obtaining an interpretable flexible model which remains explainable.

Related to this idea, Mulet et al. [2021] aimed to estimate a *mean dynamic topography product* at higher resolution than existing satellite data can provide. This is done by removing the effects of wind-driven and temporally varying sea surface height from the velocity observations of drifting objects. In the study, the Global Drifter Program is used as one of the datasets. Most of the effects within this model are written as a physics-inspired equation which relates the observed drifter velocities to the covariates. The parameters of the equations are then assumed to vary according to auxiliary variables. Mulet et al. [2021] fit these spatially varying covariates by gridding domain of the auxiliary variables, then fitting a model within each grid cell.

A promising avenue of future research would be to improve on the gridding step by instead using varying coefficient models. A varying coefficient model is a model where the prediction η takes the following form [Hastie and Tibshirani, 1993]:

$$\eta = \beta_0 + \sum_{i=1}^{p} X_i \beta_i(R_i), \tag{5.1.1}$$

where $X_i \in \mathbb{R}$ are the covariates and $R_i \in \mathbb{R}^{d_i}$ are the auxillary variables which the *varying coefficient* $\beta_i : \mathbb{R}^{d_i} \to \mathbb{R}$ changes over. Wang and Hastie [2014] use gradient boosted trees to fit the varying coefficient models β_i. One of the main benefits of doing so in place of gridding is that we may then consider a higher dimensional function for the varying coefficients.

As an example of one of the physics-inspired equations, we consider the effect of wind stress. In Mulet et al. [2021] the effect due wind stress fit is specified in complex notation:[1]

$$\mathbf{u}_w = \beta \exp(i\theta)\boldsymbol{\tau}, \tag{5.1.2}$$

where β and θ quantify the magnitude and angle respectively, of the combination of Ekman and Stokes currents, and $\boldsymbol{\tau}$ is the wind stress as the surface in complex notation. Both θ and β are assumed to be functions of mixed layer depth and latitude (which are both known). The model is fit independently in 5 metre $\times$ 1°latitude bins to account for non-stationary behaviour.

Varying coefficient models require the model be linear, hence we must simplify

[1]We omit an exponent of $\boldsymbol{\tau}$ in Equation (5.1.2). The exponent is known hence reintroducing it would be a simple transform to the data $\boldsymbol{\tau}$.

Equation (5.1.2). By expanding $\exp(i\theta)$ and converting to array notation we obtain:

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = \beta \begin{bmatrix} \tau_x \cos(\theta) - \tau_y \sin(\theta) \\ \tau_x \sin\theta + \tau_y \cos(\theta) \end{bmatrix} = \beta\cos(\theta) \begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix} + \beta\sin(\theta) \begin{bmatrix} -\tau_y \\ \tau_x \end{bmatrix}. \quad (5.1.3)$$

This effect can now be written as part of a linear equation by considering the equivalent parameterization: $\alpha_1 = \beta\cos(\theta) \quad \alpha_2 = \beta\sin(\theta)$.

As an initial model the data used would be a subset of those used in Mulet et al. [2021]:

- g_x, g_y the longitudinal and latitudinal geostrophic velocity anomalies.
- τ_x, τ_y the longitudinal and latitudinal wind stress at the surface.
- w_x, w_y the longitudinal and latitudinal wind speed.
- Mixed Layer Depth, the product supplied by Guinehut et al. [2012].

The proposed approach inspired by Mulet et al. [2021] and Equation 5.1.3 is to fit the

following multi-output varying coefficient model:

$$\begin{aligned}\mathbf{Y} =& \boldsymbol{\beta}_0\left(\text{Longitude}, \text{Latitude}\right) + \\ &\begin{bmatrix} g_x \\ g_y \end{bmatrix} \odot \boldsymbol{\beta}_1\left(\text{Longitude}, \text{Mixed Layer Depth}\right) + \\ &\begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix} \beta_2\left(\text{Latitude}, \text{Mixed Layer Depth}\right) + \\ &\begin{bmatrix} -\tau_y \\ \tau_x \end{bmatrix} \beta_3\left(\text{Latitude}, \text{Mixed Layer Depth}\right) + \\ &\begin{bmatrix} w_x \\ w_y \end{bmatrix} \beta_4\left(\text{Longitude}, \text{Latitude}\right) + \\ &\boldsymbol{\epsilon}, \end{aligned} \tag{5.1.4}$$

where $\odot$ is the element wise product, $\boldsymbol{\beta}_0, \boldsymbol{\beta_1}$ are functions $\mathbb{R}^2 \to \mathbb{R}^2$, $\beta_2, \beta_3, \beta_4$ are functions $\mathbb{R}^2 \to \mathbb{R}$. $\boldsymbol{\epsilon} \sim \mathbb{N}\left(0, \boldsymbol{\Sigma}\right)$ and $\boldsymbol{\Sigma}$ is a covariance matrix. $\boldsymbol{\Sigma}$ could be fit as a conditional variance to allow it to vary spatially using an adaptation of the work in Chapter 2.

We note that the unknowns in Equation (5.1.4) are primarily made up of two dimensional functions which may motivate the use of a more explainable model over gradient boosting. Figure 3 in Mulet et al. [2021] shows their model fits for β_2 and β_3 which appear smooth over the domain. This may mean using a model with smoothness constraints for all β_i may produce a better fit. Considering this, instead of using gradient boosting, one could use a two-dimensional basis function such as splines [Wood, 2017], a Gaussian process approach [Dambon et al., 2021, Heaton et al., 2019,

Nychka et al., 2015], or Voronoi tessellations [Zhang et al., 2014].

5.1.2 The Most Likely Path Uncertain Weights

From the work in Chapter 3, the proposed most likely path methodology did not consider that the transition probabilities are estimates of an unknown transition matrix. A straightforward improvement would be to consider that these quantities are estimates while estimating the most likely path. The aim would be that the new method will discourage the use of connections which have been estimated from few drifter observations.

Gao [2011] proposes a solution to the most likely path with uncertain weights problem. The proposed solution is to replace the weights in the network with a $\alpha\%$ quantile of the network weight estimate. By using a high quantile, the algorithm will penalise the use of weights with high uncertainty. We could estimate these quantiles using the bootstrapped networks.

5.1.3 Continuous Time Discrete Space Markov Models

To date, we do not know of a work analysing ocean trajectories using alternative forms of discrete-space Markov Chains to the Discrete-Time Discrete-Space (DTDS) model we used in Chapter 3. In particular, one of the undesired properties is that in the most likely paths shown in Chapter 3, often two subsequent hexagons on the path do not share an edge. The paths regularly feature large jumps in space.

A desirable property would be that a neighbor in a pathway shares an edge.

A straightforward solution to this would be to instead model the motion using a Continuous-Time Discrete-Space (CTDS) Markov model. A potential pathway from the CTDS model would be a sequence of neighbouring hexagons, causing the pathway to appear visually smooth with no jumps in space.

The application would be relatively straightforward, for example similar CTDS models have already been applied in the animal tracking literature [Hanks et al., 2015]. This could solve many known problems with the discrete-time discrete space model such as the choice of physical time to discretize time. Additionally, the covariate extensions proposed in Hanks et al. [2015] could be used to model the movements using the same data sources used in Chapter 2. This would allow the transition rates of the CTDS model to vary over time and capture effects such as seasonality.

Appendix A

Chapter 2 - Appendix

A.A Drifter Data Processing

Name	unit	Resolution (spatial, temporal)
Drifter Speed	u-v component $cm\ s^{-1}$	irregular, 6 hourly
Wind Speed	u-v component $m\ s^{-1}$	0.25 degrees, hourly
Surface wind stress	u-v component Pa	0.25 degrees, hourly
Geostrophic Velocity Anomaly	u-v component $m\ s^{-1}$	0.25 degrees, daily
Position	location-longitude $degrees$	irregular, 6 hourly
day of year	$days$	6 hourly

Table A.A.1: Summary of data used in the application. Drifter speed is used to define the 2 dimensional response $\mathbf{Y}$. The rest of the variables listed defined the 9 features used for $\mathbf{X}$.

For reproducibility, we give instructions on how the application dataset used in

Section 2.4 is created. In its raw form, the drifter dataset is irregularly sampled in time, however the product used here is processed to be supplied on a 6-hourly scale Lumpkin and Centurioni [2020] and includes location uncertainties. A high uncertainty would be caused by a large gap in the sampling of the raw data or a large satellite positioning error. In our study, we dropped all drifter observations with a positional error higher than 0.5° in either the longitude or latitudinal coordinates.

The six features we used relate to wind stress, geostrophic velocity, and wind speed, and are all available on longitude-latitude grids, with spatial and temporal resolution specified in Table A.A.1; we refer to these as gridded products. Prior to using these products to predict the drifter observations, we spatially interpolated[1] the gridded products to the drifter locations. To interpolate the gridded products, an inverse distance weighting interpolation was used. We only used the values at the $n = 4$ corners which define the spatial box containing the longitude-latitude location of the drifter location of interest, where the following estimate is used:

$$\frac{\sum_{i=1}^{n} w_i g_i}{\sum_{i=1}^{n} w_i}, \tag{A.A.1}$$

where g_i is the gridded value for the i^{th} corner (e.g., a 0.5 $m\ s^{-1}$ east-west geostrophic velocity at 30° longitude, 25° latitude), w_i is the inverse of the Haversine distance[2] between the drifter's longitude-latitude and the longitude-latitude location of the gridded value g_i.

If two or more of the grid corners which are being interpolated from did not have a value recorded in the gridded product (e.g., if two corners were on a coastline in the

[1]The temporal resolutions already match.

[2]The Haversine distance is the greater circle distance between two longitude-latitude pairs.

case of wind stress and geostrophic velocity), we did not interpolate and treated that point as missing. If only one corner was missing, we use $n = 3$ in Equation (A.A.1), omitting the missing point.

We low-pass filtered the drifter velocities, wind speed, and wind stress series to remove effects caused by inertial oscillations and tides; this process is similar to previous works [Laurindo et al., 2017]. A critical frequency of 1.5 times the inertial period (inertial period is a function of latitude) was used. Due to there being some missing or previously dropped data due to pre-processing steps, some time series have gaps in time larger than 6 hours. In such cases, we split that time series into individual continuous segments. We applied a fifth-order Butterworth lowpass filter to each continuous segment. If any of these segments were shorter than 18 observations (4.5 days), the whole segment was treated as missing in the dataset. The Butterworth filter was applied in a rolling fashion to account for the changing fashion of the inertial period which defines the critical frequency.

The geostrophic data are available on a daily scale, therefore we decimated all data to daily, only keeping observations at 00:00. No further preprocessing was carried out on the geostrophic velocities (after interpolation) as inertial and tidal motions are not present in the geostrophic velocity product.

Finally, to remove poorly sampled regions from the dataset, we partitioned the domain into $1^\circ \times 1^\circ$ longitude-latitude non-overlapping grid boxes. We counted the number of daily observations contained in each box, if this count was lower than 25, then the observations within that box were dropped from the dataset. A link to download the processed data can be found at `https://doi.org/10.5281/zenodo.`

5644972.

We only considered complete collections for $\mathbf{X}_i$, $\mathbf{Y}_i$; if any of the data were recorded as missing in the process explained above, that daily observation was dropped from the dataset. Prior to fitting the models, we scaled the training data $\mathbf{X}_i$ to be in the range $[0, 1]$ for all variables, this step was taken to make the neural network fitting more stable.

A.B Multivariate Normal Natural Gradient Derivations

To fit the NGBoost model, we require three elements, the log-likelihood, the derivative of the log-likelihood, and the Fisher information matrix. We use the parametrization for the multivariate Gaussian given in Section 2.2.3.

We shall derive results for the general case where P is the dimension of the data $\mathbf{Y} \in \mathbb{R}^P$. We use $Y_i \in \mathbb{R},\ i \in \{1, \ldots, P\}$ to denote the value of the i^{th} dimension of $\mathbf{Y}$ in this section, a similar notation is used for $\boldsymbol{\mu}$ and μ_i. The probability density function can be written as:

$$p(\mathbf{Y}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{P}{2}} |\boldsymbol{\Sigma}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{Y} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{Y} - \boldsymbol{\mu})\right]. \tag{A.B.1}$$

As mentioned in Section 2, the optimization of $\boldsymbol{\mu}$ is relatively straightforward as it lives entirely on the real line unconstrained. $\boldsymbol{\Sigma}$ is an $N \times N$ positive definite matrix, which is a difficult constraint. Therefore, optimizing this directly is a challenge. Instead, if we consider the Cholesky decomposition of $\boldsymbol{\Sigma}^{-1} = \mathbf{L}^\top \mathbf{L}$ where $\mathbf{L}$ is an

upper triangular matrix, we only require that the diagonal be positive to ensure that $\mathbf{\Sigma}$ is positive definite. We choose to model the inverse of $\mathbf{\Sigma}$ as in Williams [1996].

Therefore, we form an unconstrained representation for $\mathbf{L}$, where we denote a_{ij} as the element in the i^{th} row and j^{th} column as follows:

$$a_{ij} = \begin{cases} \exp(\nu_{ij}) & \text{If } i = j \\ \nu_{ij} & \text{If } i < j \\ 0 & \text{Otherwise} \end{cases} \quad \text{where } i, j \in \{1, \ldots, P\}.$$

Hence, we can fit $\mathbf{L}$ by doing gradient based optimization on $\{\nu_{ij} \in \mathbb{R} : 1 \leq i \leq j \leq P\}$ as all values live on the real line. As an example, for $P = 2$ we can fit a multivariate Gaussian using unconstrained optimization through the parameter vector $\boldsymbol{\theta} = (\mu_1,\ \mu_2, \nu_{11}, \nu_{12}, \nu_{22}) \in \mathbb{R}^M$.

As in Williams [1996], we simplify the parameters to:

$$\begin{aligned} z_i &= \mu_i - Y_i && i \in \{1, \ldots, P\} \\ \eta_i &= (\mathbf{Lz})_i = \sum_{j=i}^{p} a_{ij} z_j && i \in \{1, \ldots, P\}. \end{aligned}$$

where we have denoted $\mathbf{z}$ as the column vector of $z_i \in \mathbb{R}$. Throughout the following derivations we will interchangeably use $\{\mathbf{\Sigma}, \mathbf{L}, a_{ij}, \boldsymbol{\mu}, \mathbf{z}, z_i, \eta_i\}$ without noting that these parameters have a mapping between them (e.g. $a_{11} = \exp(\nu_{11})$).

Noting that $\log|\mathbf{\Sigma}| = \log|\mathbf{L}^\top \mathbf{L}|^{-1} = -2\log|\mathbf{L}| = -2\log\left(\prod_{i=1}^{P} a_{ii}\right)$, the negative

log-likelihood may be simplified as follows:

$$\begin{aligned} -\log p(\mathbf{Y}|\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\nu})) &= -\frac{P}{2}\log(2\pi) + \frac{1}{2} \times \log|\boldsymbol{\Sigma}| + \frac{1}{2}(\mathbf{Y} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{Y} - \boldsymbol{\mu}) \\ &= c - \sum_{i=1}^{P} \log a_{ii} + \frac{1}{2}\mathbf{z}^\top \mathbf{L}^\top \mathbf{L}\mathbf{z} \\ &= \sum_{i=1}^{P} \left\{ \frac{1}{2}\eta_i^2 - \nu_{ii} \right\} + c, \end{aligned}$$

where c is a constant independent of $\boldsymbol{\mu}$ and ν_{ij}. For shorter notation, we will write $l = -\log p(\mathbf{Y}|\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\nu}))$.

The first derivatives are stated in Williams [1996] and we also state them here for reference:

$$\frac{dl}{d\mu_i} = \sum_{j=1}^{i} \eta_j a_{ji} \qquad i \in \{1, \ldots, P\} \tag{A.B.2}$$

$$\frac{dl}{d\nu_{ii}} = \eta_i z_i a_{ii} - 1 \qquad i \in \{1, \ldots, P\} \tag{A.B.3}$$

$$\frac{dl}{d\nu_{ij}} = \eta_i z_j \qquad 1 \leq i < j \leq P. \tag{A.B.4}$$

The final element we need for NGBoost to work is the $M \times M$ Fisher information matrix. We derive the Fisher Information from the following:

$$\mathcal{I}_{ij} = \mathbb{E}\left[\frac{d^2 l}{d\theta_i d\theta_j} \right], \quad i,\ j \in \{1, \ldots, M\}.$$

Note that $\mathcal{I}_{ij}$ is a symmetric matrix, such that $\mathcal{I}_{ij} = \mathcal{I}_{ji}$. For convenience, we denote the entries of the matrix with the subscripts of the parameter symbols, e.g. $\mathcal{I}_{\mu_i, \nu_{kq}}$. We use the letters $i, j, k, q \in \{1, \ldots, P\} \subset \mathbb{N}$ to index the variables. The following two expectations are frequently used: $\mathbb{E}[z_i z_j] = \boldsymbol{\Sigma}_{ij}$ and $\mathbb{E}[z_i] = 0$.

The derivatives of η_j with respect to the other parameters are repeatedly used, so

we give them here:

$$\frac{d\eta_k}{d\mu_i} = \begin{cases} 0 & \text{if } k > i \\ a_{ki} & \text{if } k \leq i \end{cases} \qquad i,\ k \in \{1, \ldots, P\}$$

$$\frac{d\eta_i}{d\nu_{kq}} = \frac{d}{d\nu_{kq}} \sum_{j=i}^{n} a_{ij} z_j \qquad 1 \leq i \leq p, \quad 1 \leq k \leq q \leq P$$

$$= \begin{cases} 0 & \text{if } i \neq k \\ z_q & \text{else if } q > k \\ a_{ii} z_i & \text{else if } q = k \end{cases} .$$

We start with the Fisher information for μ_i, μ_j, using the existing derivation of $\frac{dl}{d\mu_k}$ in Equation (A.B.2):

$$\begin{aligned} \frac{d^2 l}{d\mu_i d\mu_k} &= \frac{d}{d\mu_i} \sum_{j=1}^{k} \eta_j a_{jk} \qquad i,\ k \in \{1, \ldots, P\} \\ &= \sum_{j=1}^{k} a_{ji} a_{jk} \\ &= \left[\mathbf{L}^\top \mathbf{L}\right]_{ik} . \end{aligned}$$

We therefore have that:

$$\mathcal{I}_{\mu_i, \mu_j} = \mathbf{\Sigma}_{ij} \quad i,\ j \in \{1, \ldots, P\}. \tag{A.B.5}$$

Next we consider the mean differentiated with respect to ν_{kq} again starting from

Equation (A.B.2):

$$\begin{aligned}\frac{d^2 l}{d\mu_i d\nu_{kq}} &= \frac{d}{d\nu_{kq}} \sum_{j=1}^{i} \eta_j a_{ji} && i \in \{1, \ldots, P\},\ 1 \leq k \leq q \leq P \\ &= \sum_{j=1}^{k} \left[\eta_j \frac{d}{d\nu_{kq}} a_{ji} + a_{ji} \frac{d}{d\nu_{kq}} \eta_j \right].\end{aligned}$$

As the expectation of both terms inside the sum is zero for every valid combination of i, k, q, we conclude that:

$$\mathcal{I}_{\mu_i, \nu_{kq}} = 0 \quad i \in \{1, \ldots, P\},\ 1 \leq k \leq q \leq P. \tag{A.B.6}$$

Finally, the last elements we require for the Fisher information matrix are the entries for ν_{ij}, ν_{kq}. First, we consider diagonals ($i = j$) w.r.t. all ν_{kq} with $k \leq q$. We start by using Equation (A.B.3) for $\frac{dl}{d\nu_{ii}}$:

$$\begin{aligned}\frac{d^2 l}{d\nu_{ii} d\nu_{kq}} &= \frac{d}{d\nu_{kq}} \eta_i z_i a_{ii} && i \in \{1, \ldots, P\},\ 1 \leq k \leq q \leq P \\ &= a_{ii} z_i \frac{d}{d\nu_{kq}} \eta_i + \eta_i z_i \frac{d}{d\nu_{kq}} a_{ii} \\ &= \begin{cases} a_{ii} z_i z_q & \text{if } k = i \text{ and } k < q \\ a_{ii}^2 z_i z_i & \text{if } i = k = q \\ 0 & \text{if } i \neq k \end{cases} \\ &+ \begin{cases} z_i \sum_{j=i}^{P} a_{ij} z_j a_{ii} & \text{if } i = k = q \\ 0 & \text{Otherwise.} \end{cases}\end{aligned}$$

Taking the expectation we get:

$$\mathcal{I}_{\nu_{ii},\nu_{kq}} = \begin{cases} a_{ii}\boldsymbol{\Sigma}_{iq} & \text{if } k = i \text{ and } q > k \\ a_{ii}^2\boldsymbol{\Sigma}_{ii} + a_{ii}\sum_{j=i}^{P} a_{ij}\boldsymbol{\Sigma}_{ij} & \text{if } i = k = q \\ 0 & \text{if } i \neq k. \end{cases} \quad , i \in \{1, \ldots, P\},\ 1 \leq k \leq q \leq P \tag{A.B.7}$$

Finally, we derive the Fisher information for the off diagonals with respect to the off diagonals ($i < j$ and $k < q$). We start from the expression for $\frac{dl}{v_{ij}}$ in Equation (A.B.4):

$$\begin{aligned} \frac{d^2 l}{d\nu_{kq} d\nu_{ij}} &= \frac{d}{d\nu_{kq}}\eta_i z_j \qquad\qquad 1 \leq i < j \leq P,\ 1 \leq k < q \leq P \\ &= z_j \frac{d}{d\nu_{kq}}\eta_i \\ &= \begin{cases} z_j z_q & \text{if } k = i \\ 0 & \text{if } i \neq k. \end{cases} \end{aligned}$$

Hence,

$$\mathcal{I}_{\nu_{ij},\nu_{kq}} = \begin{cases} \Sigma_{jq} & \text{If } k = i \\ 0 & \text{if } k \neq i. \end{cases} \qquad 1 \leq i < j \leq P,\ 1 \leq k < q \leq P. \tag{A.B.8}$$

Equations (A.B.5), (A.B.6), (A.B.7), (A.B.8) give the full specification of the Fisher Information.

A.C Practical Adjustment for Singular Inverse Covariance Matrices

A small adjustment is made to $\mathbf{L}$ to improve numerical stability in the implementation. In particular, if the diagonal entries of $\mathbf{L}$ from Equation (2.2.9) ($\exp(\nu_{ii})$) get close to zero the resulting matrix $\mathbf{\Sigma}^{-1}$ has a determinant close to zero, causing numerical problems to occur when inverting $\mathbf{\Sigma}^{-1}$. The fix we propose for this is to add a small perturbation of 10^{-6} to the diagonal elements of $\mathbf{L}$. For most datasets, once the model is fitted, the predicted value for the diagonal elements will be much larger than 10^{-6} so this slight adjustment has a negligible effect on the predictions.

This practical adjustment will become an issue in cases where the determinant of the covariance matrix is very large. As an extreme example, suppose the predicted covariance without the adaptation is $\mathbf{\Sigma} = \mathbb{I}_2 \times 10^{12}$ where $\mathbb{I}_2$ is the identity matrix. The practical adjustment results in a marginal variance four times larger in this case, which is clearly not negligible. Problems like this can typically be mitigated by scaling the output data $\mathbf{Y}$ (e.g., using a min-max scaling strategy).

A.D Supplementary Information

This supplementary information provides the following:

1. Extra details on the setup of the probabilistic neural network.

2. Details of the grid search which we used for both the application and simulation.

3. Extra details on the timings of all the methods.

4. Extended simulation results, where for the simulations we show the metrics introduced for the application. We also show the unmodified simulation as it was presented in Williams [1996].

5. A sample code snippet using the package.

A.D.1 Probabilistic Neural Networks

In this chapter, an existing neural network approach (referred to as NN) is used to fit conditional multivariate Gaussian distributions. The method fits a neural network taking inputs $\mathbf{X} \in \mathcal{X}$ where the output layer has M units, which are used as $\boldsymbol{\theta}$ in the probability density function parameterization in Section 2.2.3 [Williams, 1996, Sützle and Hrycej, 2005]. The loss function used when optimizing the neural network parameters is the negative log-likelihood.

Instead of fitting M independent models as in Section 2.2.2, when using probabilistic neural networks we fit one model which has M outputs, one for each parameter of the probability density function. Throughout the chapter, a fully connected neural network structure is used such as the one in Figure A.D.1. We describe the structure search in Section A.D.2.

Note that, unlike NGBoost, the natural gradient is not used in the optimization of the probabilistic neural networks to account for the geometry of the distribution space.

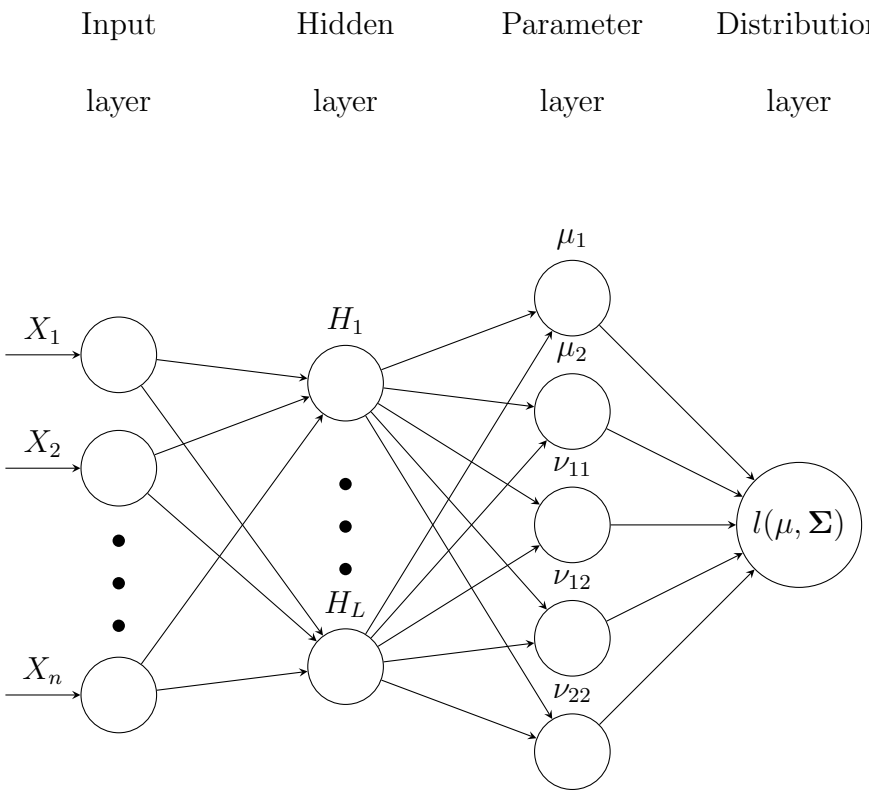

Figure A.D.1: Neural Network structure with one hidden layer with L neurons and a 2 dimensional output.

A.D.2 Machine Learning Model Details

We cease the model fit when the validation score has not improved for 50 iterations. For the neural network approach, an iteration is one *epoch* which is a full pass of the training dataset. For boosting approaches, an iteration is the growth of one base learner $f^{(m)}$ as explained in Section 2.2. If early stopping does not occur, we specify a maximum of 1000 iterations, however in our results this was only reached in the application section when using the GB method. The validation score used is the same as the scoring rule or loss for the method, i.e., root mean squared error for skGB, negative log-likelihood for all other methods.

Boosting Grid Search In our reported results, the boosting methods (NGB, GB, skGB, and Indep NGB) all used the same base learner, an sklearn decision tree, resulting in a direct mapping between hyper-parameters. Hence, we carried out the same grid search for these four models. We conducted a grid search over the following sets:

- Max depth in $\{8, 15, 31, 64\}$.
- Minimum data in leaf in $\{1, 15, 32\}$.

which resulted in 12 hyper-parameter sets for each method, and all other parameters were left at the default other than the learning rate. We did not tune the learning rate, we used a learning rate of 0.01 for the simulation and 0.1 for the application. The larger learning rate was chosen for the application to allow us to conduct replicated fits in a reasonable amount of time.

Neural Network grid search We carried out the following grid search for the neural network architecture in both the simulation example and application in Sections 2.3 and 2.4. We considered the following structures for the hidden layers in the grid search:

- A 20 unit layer
- A 50 unit layer
- A 100 unit layer (best for simulation when $N \in \{1000, 3000, 5000, 10000\}$)
- A 20 unit layer followed by another 20 unit layer
- A 50 unit layer followed by another 20 unit layer. (best for simulation when $N \in \{500, 8000\}$)
- 100 units followed by another 20 unit layer (best for the application).

After each layer, we applied a RELU activation on each node aside from the output nodes. A fixed batch batch size of 256 was used. For the simulation, a learning rate of 0.01 worked well for the Adam optimizer. In the application, we added an extra parameter into the grid search, a learning rate of 0.001 and 0.01. Adam's other parameters were left as the defaults in tensorflow. For the application, the best learning rate and structure pair was 0.001 and a 100 unit layer followed by a 20 unit layer respectively.

Timings & Computation All model fits were run on an internal computing cluster using only CPUs. We note that computational time was not a key consideration in

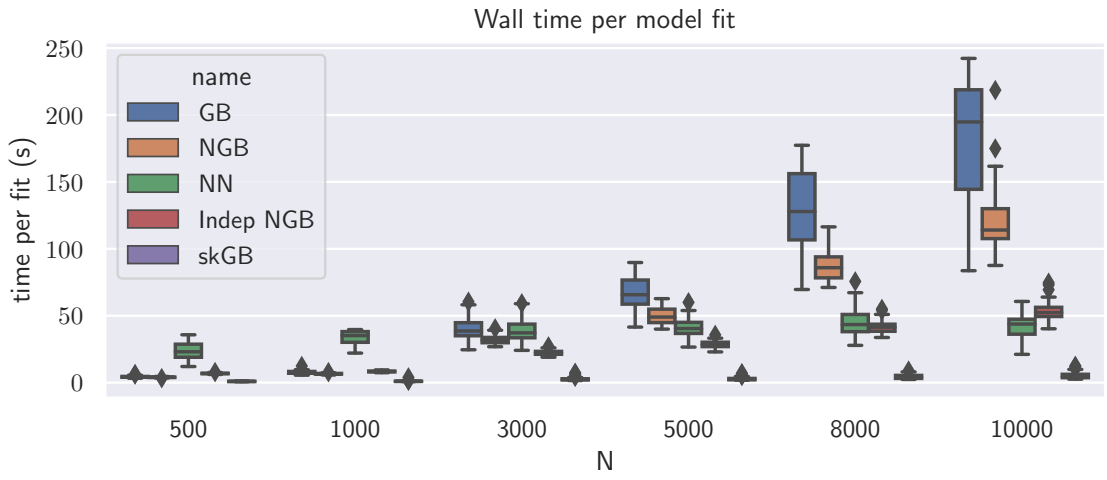

Figure A.D.2: Wall Times per model fit over the 50 replications of Section 2.3. Box plot drawn in the standard way (boxes being a 50% inter-quartile range, with lines drawn at median and at the minimum or maximum excluding "outliers"). NN model fits were given access to 10 threads, all other methods ran on a single thread. The plotting times are for a single model fit. It does not show the times relating to the grid search carried out for the NN method.

this work. If it was of key importance, using a more efficient base learner for all boosting based approaches would be advisable for large N, such as LightGBM [Ke et al., 2017] or XGBoost [Chen and Guestrin, 2016].

Simulation We show the wall times for the simulation which we ran for Section 2.3 in Figure A.D.2. Generally, NGB and GB are the slowest and skGB is fastest.

Application We did not explicitly time each model fit in the application, hence we only give rough estimates based on the total running time for the best selected models from the grid search. The neural networks were given access to 15 CPU threads. All other methods were limited to 1 CPU thread, the difference between wall time and

CPU time was negligible for all boosting approaches, hence we only report the wall time for those here. The estimated fitting times are shown below (reported as total time for the 10 random replications divided by 10):

- NGB: 1.45 hours
- GB: 6 hours
- skGB: 1.3 hours
- NN: 0.7 hours wall time, 2.1 hours CPU time.

The GB method was the slowest method. This is because early stopping did not occur, meaning the algorithm would run for the full 1000 boosting iterations. In contrast, for NGB, early stopping usually occurred around 250 boosting iterations, which reduced the run time.

We reiterate that these times are rough estimates. Both the simulation and the application were run on a cluster with various nodes with shared resources which the model fitting would be assigned to. For the timings shown in the list above, the NN approach was fit on a node with a 2.6GHz AMD Opteron Processor 6238 and the NGB, GB and skGB models were fit on a 2.30GHz Intel Xeon CPU E5-2699 v3.

A.D.3 Extended Simulation Results

We now report the metrics introduced in Section 2.4 for the simulation study in Section 2.3. These were omitted from the main manuscript due to page length considerations. These are shown in Table A.D.1. We can see that the RMSE metric is very poor for

the GB method, which explains the poor NLL and KL divergence metrics for GB. With all methods, NLL and KL divergence show similar patterns, however, if we used the NLL metric it would choose NN for lower values of N over NGB.

We give the results for the unaltered simulation setup of Williams [1996] in Table A.D.2. As noted in the main body of the chapter, we see that NN does best for $N \in \{500, 1000, 3000\}$ and NGB does best for $N \in \{5000, 8000, 10000\}$ according to the KL divergence from the truth metric. Moreover, we note GB does not do as poorly as in Table A.D.1. We believe this is because the mean is fit better as can be seen in the RMSE metric. Otherwise, the general patterns seen in both Tables A.D.1 and A.D.2 are very similar.

A.D.4 Sample Code Snippet

To emphasize how straightforward it is to use the model we show a minimal code snippet below to show how the package can be used:

```
# pandas is used to read the data
import pandas as pd

# NGBoost is the package for prediction
import ngboost

data = pd.read_hdf("path to dataset")

# Split into covariates X:
coordinates = data[["lon", "lat"]]
# and response Y:
velocities = data[['u', 'v']]

# Initiate the model and specify the response is 2 dimensional
# n_estimators should be larger in practice.
multivariate_model = ngboost.NGBoost(
```

```
    Dist=ngboost.distns.MultivariateNormal(2),
    n_estimators=10
    )
#Fit the ensemble
multivariate_model.fit(X=coordinates, Y=velocities)

# Obtain the predicted distribution at the coordinates
predicted_distributions = multivariate_model.pred_dist(coordinates)

# The predicted mean and covarince arrays respectively:
predicted_distributions.loc # Shape N x 2
predicted_distributions.cov # Shape N x 2 x 2
```

Metric	N	NGB	Indep NGB	skGB	GB	NN
KL div	500	**0.56 ± 0.02**	1.63 ± 0.04	17.19 ± 0.30	126.23 ± 2.58	1.28 ± 0.55
	1000	**0.26 ± 0.00**	1.15 ± 0.02	17.96 ± 0.27	114.11 ± 1.62	0.32 ± 0.02
	3000	**0.11 ± 0.00**	0.88 ± 0.01	19.61 ± 0.25	97.68 ± 1.40	0.15 ± 0.00
	5000	**0.08 ± 0.01**	0.88 ± 0.01	20.31 ± 0.17	90.10 ± 1.29	0.13 ± 0.01
	8000	**0.05 ± 0.00**	0.87 ± 0.01	20.61 ± 0.17	79.01 ± 1.17	0.10 ± 0.00
	10000	**0.04 ± 0.00**	0.83 ± 0.01	20.55 ± 0.15	74.80 ± 1.19	0.13 ± 0.00
NLL	500	1.35 ± 0.02	1.27 ± 0.01	2.05 ± 0.01	2.49 ± 0.01	**1.05 ± 0.02**
	1000	1.02 ± 0.01	1.10 ± 0.01	1.92 ± 0.01	2.25 ± 0.01	**0.89 ± 0.01**
	3000	0.84 ± 0.01	1.02 ± 0.01	1.90 ± 0.01	2.01 ± 0.01	**0.83 ± 0.01**
	5000	**0.79 ± 0.01**	0.98 ± 0.01	1.87 ± 0.01	1.89 ± 0.01	0.81 ± 0.01
	8000	**0.78 ± 0.01**	0.97 ± 0.01	1.87 ± 0.01	1.80 ± 0.01	0.80 ± 0.01
	10000	**0.77 ± 0.01**	0.97 ± 0.01	1.88 ± 0.01	1.75 ± 0.01	0.82 ± 0.01
RMSE	500	0.66 ± 0.00	0.65 ± 0.00	0.65 ± 0.00	1.91 ± 0.01	**0.65 ± 0.00**
	1000	0.63 ± 0.00	0.63 ± 0.00	0.63 ± 0.00	1.85 ± 0.01	**0.62 ± 0.00**
	3000	0.63 ± 0.00	0.62 ± 0.00	0.63 ± 0.00	1.76 ± 0.01	**0.62 ± 0.00**
	5000	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	1.70 ± 0.01	**0.62 ± 0.00**
	8000	0.62 ± 0.00	**0.62 ± 0.00**	0.62 ± 0.00	1.65 ± 0.01	0.62 ± 0.00
	10000	0.62 ± 0.00	**0.62 ± 0.00**	0.62 ± 0.00	1.64 ± 0.01	0.62 ± 0.00
90% PR area	500	1.90 ± 0.02	2.59 ± 0.03	4.27 ± 0.05	8.57 ± 0.10	3.02 ± 0.07
	1000	1.97 ± 0.01	2.52 ± 0.02	4.54 ± 0.04	7.74 ± 0.07	2.64 ± 0.02
	3000	2.22 ± 0.01	2.60 ± 0.01	4.96 ± 0.03	7.21 ± 0.06	2.58 ± 0.02
	5000	2.30 ± 0.01	2.65 ± 0.01	5.09 ± 0.02	6.89 ± 0.05	2.61 ± 0.03
	8000	2.36 ± 0.01	2.69 ± 0.01	5.14 ± 0.02	6.49 ± 0.05	2.65 ± 0.03
	10000	2.38 ± 0.01	2.69 ± 0.01	5.19 ± 0.02	6.37 ± 0.05	2.63 ± 0.03
90% PR cov	500	0.76 ± 0.00	0.83 ± 0.00	0.81 ± 0.00	0.84 ± 0.00	0.88 ± 0.00
	1000	0.80 ± 0.00	0.85 ± 0.00	0.83 ± 0.00	0.86 ± 0.00	0.89 ± 0.00
	3000	0.85 ± 0.00	0.86 ± 0.00	0.85 ± 0.00	0.88 ± 0.00	0.89 ± 0.00
	5000	0.87 ± 0.00	0.87 ± 0.00	0.86 ± 0.00	0.89 ± 0.00	0.90 ± 0.00
	8000	0.87 ± 0.00	0.88 ± 0.00	0.87 ± 0.00	0.90 ± 0.00	0.90 ± 0.00
	10000	0.88 ± 0.00	0.88 ± 0.00	0.86 ± 0.00	0.90 ± 0.00	0.90 ± 0.00

Table A.D.1: Metrics used in Section 2.4, on the simulation run in Section 2.3. We also include the KL divergence from Table 2.3.1 for comparison (with one less decimal point shown here to allow the table to fit on the page). The average over the 50 replications is reported. Standard error estimates reported after ±.

Metric	N	NGB	Indep NGB	skGB	GB	NN
KL div	500	0.96 ± 0.03	2.27 ± 0.06	19.84 ± 0.39	2.16 ± 0.06	**0.40 ± 0.04**
	1000	0.46 ± 0.01	1.44 ± 0.03	20.08 ± 0.29	1.28 ± 0.03	**0.26 ± 0.04**
	3000	0.18 ± 0.01	1.06 ± 0.02	20.43 ± 0.19	0.75 ± 0.01	**0.13 ± 0.02**
	5000	**0.11 ± 0.00**	0.96 ± 0.02	20.65 ± 0.20	0.65 ± 0.01	0.13 ± 0.02
	8000	**0.07 ± 0.00**	0.91 ± 0.01	20.75 ± 0.17	0.55 ± 0.01	0.11 ± 0.02
	10000	**0.06 ± 0.00**	0.92 ± 0.02	21.10 ± 0.18	0.54 ± 0.01	0.13 ± 0.02
NLL	500	1.24 ± 0.01	1.25 ± 0.01	1.96 ± 0.01	1.27 ± 0.01	**0.93 ± 0.02**
	1000	1.03 ± 0.01	1.12 ± 0.01	1.91 ± 0.01	1.12 ± 0.01	**0.86 ± 0.01**
	3000	0.86 ± 0.01	1.02 ± 0.01	1.88 ± 0.01	0.98 ± 0.01	**0.82 ± 0.01**
	5000	0.82 ± 0.01	1.00 ± 0.01	1.86 ± 0.01	0.94 ± 0.01	**0.81 ± 0.01**
	8000	**0.78 ± 0.01**	0.97 ± 0.01	1.86 ± 0.01	0.90 ± 0.01	0.80 ± 0.01
	10000	**0.78 ± 0.01**	0.97 ± 0.01	1.86 ± 0.01	0.90 ± 0.01	0.81 ± 0.01
RMSE	500	0.64 ± 0.00	0.64 ± 0.00	0.63 ± 0.00	**0.63 ± 0.00**	0.64 ± 0.00
	1000	0.63 ± 0.00	0.63 ± 0.00	0.63 ± 0.00	**0.62 ± 0.00**	0.63 ± 0.00
	3000	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	**0.62 ± 0.00**	0.62 ± 0.00
	5000	0.61 ± 0.00	0.61 ± 0.00	0.61 ± 0.00	**0.61 ± 0.00**	0.62 ± 0.00
	8000	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	**0.61 ± 0.00**	0.62 ± 0.00
	10000	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	**0.62 ± 0.00**	0.62 ± 0.00
90% PR area	500	2.03 ± 0.02	2.61 ± 0.03	4.68 ± 0.06	2.72 ± 0.03	2.79 ± 0.05
	1000	2.11 ± 0.02	2.53 ± 0.02	4.86 ± 0.05	2.65 ± 0.02	2.79 ± 0.04
	3000	2.31 ± 0.01	2.65 ± 0.01	5.10 ± 0.03	2.69 ± 0.01	2.71 ± 0.03
	5000	2.35 ± 0.01	2.67 ± 0.01	5.14 ± 0.02	2.70 ± 0.01	2.69 ± 0.02
	8000	2.41 ± 0.01	2.70 ± 0.01	5.20 ± 0.02	2.71 ± 0.01	2.73 ± 0.03
	10000	2.42 ± 0.01	2.72 ± 0.01	5.24 ± 0.02	2.72 ± 0.01	2.70 ± 0.03
90% PR cov	500	0.78 ± 0.00	0.83 ± 0.00	0.84 ± 0.00	0.84 ± 0.00	0.88 ± 0.00
	1000	0.81 ± 0.00	0.84 ± 0.00	0.85 ± 0.00	0.85 ± 0.00	0.89 ± 0.00
	3000	0.86 ± 0.00	0.87 ± 0.00	0.86 ± 0.00	0.88 ± 0.00	0.89 ± 0.00
	5000	0.86 ± 0.00	0.87 ± 0.00	0.87 ± 0.00	0.88 ± 0.00	0.90 ± 0.00
	8000	0.88 ± 0.00	0.88 ± 0.00	0.87 ± 0.00	0.89 ± 0.00	0.90 ± 0.00
	10000	0.87 ± 0.00	0.88 ± 0.00	0.87 ± 0.00	0.89 ± 0.00	0.90 ± 0.00

Table A.D.2: Same as Table A.D.1 except we use the same simulation as Williams [1996], i.e. without adding the $+x$ and $-x^2$ terms of Equation (2.3.1). Note that the difference between NGB and GB or NN is not as large as in the original table. All values rounded to two decimal places.

Appendix B

Chapter 3 - Appendix

B.A Data Availability

The drifter data were provided by the Global Drifter Program [Lumpkin and Centurioni, 2020]. The currents used for visualisation purposes in Figure 3.1.1 are V3.05 of the dataset supplied on the Global Drifter Program website [Laurindo et al., 2017].

Code to reproduce all figures related to the method is available at `https://github.com/MikeOMa/MLTravelTimesFigures` which depends on the Python package implementing all of the above methods in this chapter at `https://github.com/MikeOMa/DriftMLP`. The package takes roughly 3 minutes total to go from raw data to a pairwise travel time matrix for the locations shown in Table 3.5.1 using Algorithm 2.

$P(x\|y)$	denotes the probabilities of event(s) x given that y occurs.
$\mathbb{E}[x]$	The expectation of x.
$f(s)$	The discretization function i.e. H3.
$\mathbb{I}(x)$	Indicator function giving 1 if x is true and 0 otherwise.
$\arg\max_{x \in S}$	An operator which gives the input value, which maximizes the function q, restricted to the set S.
T, $T_{i,j}$	T denotes transition matrix, with entries $T_{i,j}$. $i,\ j \in \mathcal{S}$, denoting the probability of moving from state i to j in $\mathcal{T}_L$ days.
$x^\circ \times y^\circ$	refers to a longitude-latitude grid system, x degrees in the longitudinal direction, y degrees in the latitudinal direction.
$\mathcal{T}_L$	Lagrangian cut off time.
$\mathcal{S}$	The set of all possible spatial indices.
$\mathcal{P}_{o,d}$	The set of all possible paths going from o to d.
$\mathbf{p} = \{p_i\}_{i=1}^n$	A pathway of length n. Indicates a sequence $p_1,\ p_2, \ldots, p_n$. All $p_i \in \mathcal{S}$.
$\mathbf{k}$	The expected travel time of a path $\mathbf{p}$.
$\hat{\mathbf{p}}$, $\hat{\mathbf{k}}$	Hat notation implies we are considering the most likely path and travel time of that path respectively.
s_t	used to index the state of the Markov chain after t steps.

Table B.B.1: Table of mathematical notation.

B.B Table of Notation

We include a table of mathematical notation for reader reference in Table B.B.1.

B.C Finding the shortest path

To solve the optimization of Equation (3.3.5), we can equivalently consider the *log* of $P(\mathbf{p})$:

$$\log P(\mathbf{p}) = \sum_{i=0}^{n-1} \log T_{p_i,p_{i+1}}.$$

Then we use the fact that:

$$\begin{aligned}\hat{\mathbf{p}} &= \arg\max_{\mathbf{p}\in\mathcal{P}_{o,d}} \{\log P(\mathbf{p})\} = \arg\min_{\mathbf{p}\in\mathcal{P}_{o,d}} \{-\log P(\mathbf{p}\}) \\ &= \arg\min_{\mathbf{p}\in\mathcal{P}_{o,d}} \left\{ -\sum_{i=0}^{n-1} \log T_{p_i,p_{i+1}} \right\}. \end{aligned} \tag{B.C.1}$$

Now in this form this can be solved using the vast literature on shortest path algorithms.

B.D Shortest Path Algorithms

Shortest path algorithms [Gallo and Pallottino, 1988, Dijkstra, 1959], such as Dijksta's algorithm, are popular algorithms which find the so called *s*hortest path within a graph. In our case the graph is formed such that the vertices or nodes uniquely correspond to a grid system index, i.e. a row/column in the transition matrix T. If there is a non-zero probability in $T_{i,j}$ we add an edge denoted $e_{i,j}$, where the weight on this edge is denoted $w(e_{i,j}) = -\log(T_{i,j})$ between the vertex i and going to the vertex j. Note that $T_{i,j}$ is not necessarily the same as $T_{j,i}$, hence we have a *directed graph*. Given a start vertex o and an end vertex d, shortest path algorithms will find

the path $P = \{v_1 \ldots, v_n\}$ such that P minimizes the following

$$\sum_{i=1}^{n-1} w(e_{v_i, v_{i+1}}),$$

hence it solves the problem in Equation (B.C.1). The algorithm used is exact, hence if no path is found then no path exists given the current network.

B.E Derivation of Equation (3.3.6)

The derivation uses the Markov property, the conditional probability definition, and the fact that $P(x \in \{a, b\}) = P(x = a) + P(x = b)$.

$$\begin{aligned}
&P(s_{t+k} = p_{i+1}, \{s_{t+l} = p_i\}_{l=1}^{k-1} | s_t = p_i, \mathbf{p}\}) \\
&= P(s_{t+k} = p_{i+1} | s_{t+k-1} = p_i, s_{t+k} \in \{p_i, p_{i+1}\}) \\
&\quad \times \prod_{l=1}^{k-1} P(s_{t+l} = p_i | s_{t+l-1} = p_i, s_{t+l} \in \{p_i, p_{i+1}\}) \\
&= \frac{P(s_{t+k} = p_{i+1} | s_{t+k-1} = p_i)}{P(s_{t+k} \in \{p_i, p_{i+1}\} | s_{t+k-1} = p_i)} \\
&\quad \times \prod_{l=1}^{k-1} \frac{P(s_{t+l} = p_i | s_{t+l-1} = p_i)}{P(s_{t+l} \in \{p_i, p_{i+1}\} | s_{t+l-1} = p_i)} \\
&= \frac{P(s_{t+k} = p_{i+1} | s_{t+k-1} = p_i)}{P(s_{t+1} \in \{p_i, p_{i+1}\} | s_t = p_i)^k} \\
&\quad \times \prod_{l=1}^{k-1} P(s_{t+l} = p_i | s_{t+l-1} = p_i) \\
&= \frac{T_{p_i, p_{i+1}} T_{p_i, p_i}^{k-1}}{(T_{p_i, p_i} + T_{p_i, p_{i+1}})^k}
\end{aligned}$$

where the first equality follows from the explanation given in Section 3.3.4.

B.F Brief Sensitivity Analysis to cut off time

The main tuning parameter which we have fixed in this chapter is the Lagrangian cut off time used when estimating the transition matrix T. The method is not especially sensitive to this choice as we shall now demonstrate. To show the sensitivity we ran an experiment where for a grid of values for $\mathcal{T}_L$ we estimated a pairwise travel time matrix for the locations in Table 3.5.1, then we estimated the Spearman correlation coefficient between the non-diagonal entries of each matrix to the corresponding entry of the travel time matrix generated from $\mathcal{T}_L = 5$. Results are shown in Figure B.F.1. The experiment shows that the distances change but overall the matrices are very strongly correlated, particularly for $\mathcal{T}_L > 2$. For comparison the average correlation value between the the pairwise travel time matrix $\mathcal{T}_L$ and the travel time matrices generated from the 100 rotations used in Section 3.5.3 is 0.8. A similar analysis, considering sensitivity to grid sizes is given in the online supplementary information.

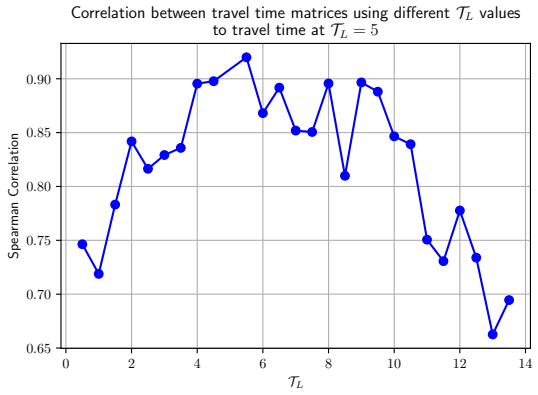

Figure B.F.1: Spearman Correlation coefficient between the non-diagonal elements of the travel time matrix generated by $\mathcal{T}_L = 5$ and the matrices generated by the values of $\mathcal{T}_L$ on the x-axis.

B.G Supplementary Information

B.H Grid Size Sensitively

We ran an experiment using a **resolution 3** and **resolution 4** H3 grid and three different sized longitude-latitude grids. The resolution 4 grid breaks down each resolution 3 cell into roughly seven polygons (in the manner seen in Figure 3.3.1 of the main text), resulting in an average area of $1,770km^2$, hence a much finer grid. The transition probabilities from the smaller grid sizes generally have higher uncertainty, due to less data which then results in even more variable pathways. In Figure B.H.1, we can see that the variance across grid sizes is far larger than the difference in results from changing $\mathcal{T}_L$.

One of the more concerning features of a longitude-latitude grid system is that the pathways tend to choose transitions either directly east, west, north or south

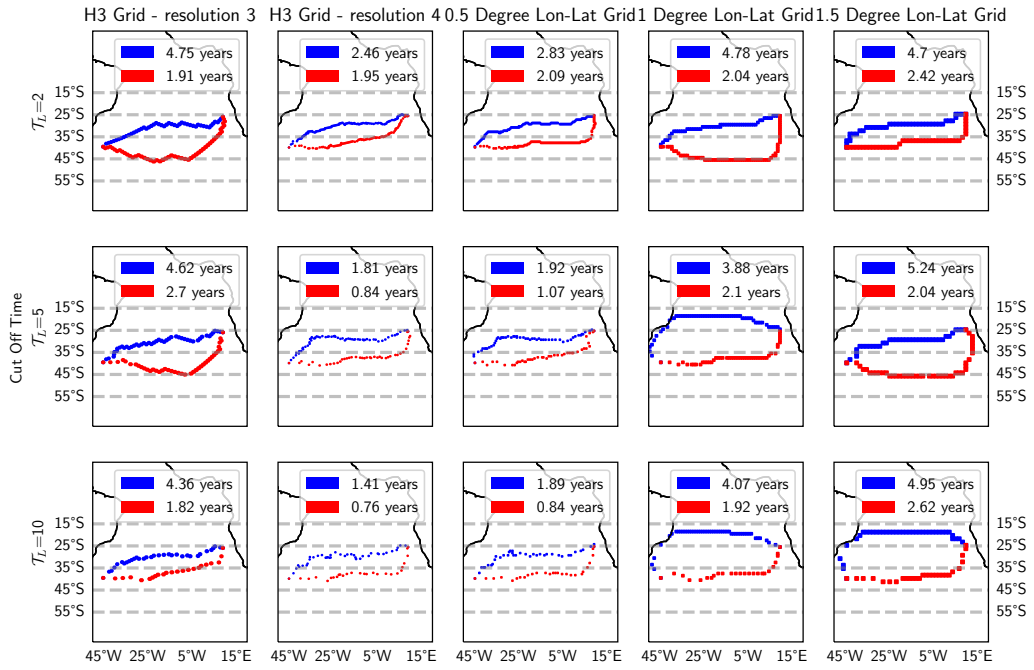

Figure B.H.1: Paths and travel times between location 1 and 3 in Table 3.5.1 of the main text. Results given for $\mathcal{T}_l$ = 2, 5, 10 days and with various grid systems. Grid systems are the same within a column, indicated by the title of that column. Lagrangian cut-off times are altered by row.

and rarely do diagonal transitions. This is particularly visible with the larger 1.5 degree grid size. In contrast, for the H3 grids we do not see this sort of pattern, and trajectories are more naturally resembling meandering pathways. This is a key benefit of the H3 grid system, in addition to the more constant cell sizes it produces across the globe.

B.H.1 Comparison to Smith (2018)

Smith et al. [2018] studied long-distance dispersal events through a number of methods. An example of pathways are given in Figure 2 of Smith et al. [2018], about how an object could drift from the south-east coast of Australia over to the south-west coast of Brazil. The two pathways given are from simulated particles under the HYbrid Coordinate Ocean Model (HYCOM) [Chassignet et al., 2007] and using Monte Carlo Super Trajectories [van Sebille et al., 2011], which we qualitatively compare to our Most Likely Path method here. Figure B.H.2 provides the results which the most likely path method found using 60 rotations. We show results from both resolution 3 and resolution 4 in the figure.

The pathways shown in Figure B.H.2 look very similar to those of the route of simulated HYCOM particles. The travel times reported in Figure B.H.2 are also comparable, where the minimum time reported in Smith et al. [2018] is 2.4 years using MCSTs. The resolution 3 travel time results in a close match, which is expected as the grid size is similar to what Monte Carlo Super Trajectories are created with.

B.H.2 Bootstrap and Rotation Pathways

In Figure 3.5.6 we showed example pathways for rotations in resolution 3 of the H3 grid system. Here we show the same results in three different flavors.

- A similar plot using resolution 4 of the H3 grid system in Figure B.H.3 with 60 rotations.
- A bootstrap version in Figure B.H.4 with 100 bootstrap samples.

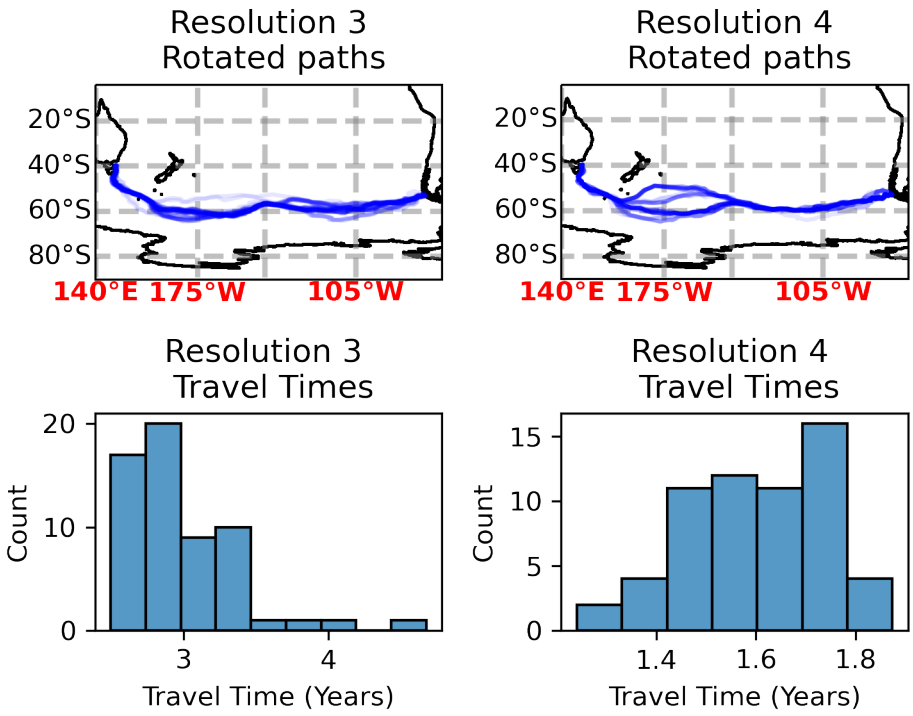

Figure B.H.2: 60 Pathways of particles going from the south-east coast of Australia to the south-west coast of Brazil. Both resolution 3 and resolution 4 H3 grid results shown. Pathway axis and grid lines set to match Figure 2 of Smith et al. [2018].

- A bootstrap version using resolution 4 in Figure B.H.5 with 100 bootstrap samples.

Due to computational restrictions with the resolution 4 networks we use 60 rotations instead of the 100 used in the main document for resolution 3.

We see that there is less variance in the pathways due to rotations in resolution 4 compared to resolution 3, however the variability in the pathways from the bootstrap is considerably larger. This is due to the classic bias-variance trade-off in statistical estimation. The results are less biased at resolution 4 due to less discretization, but

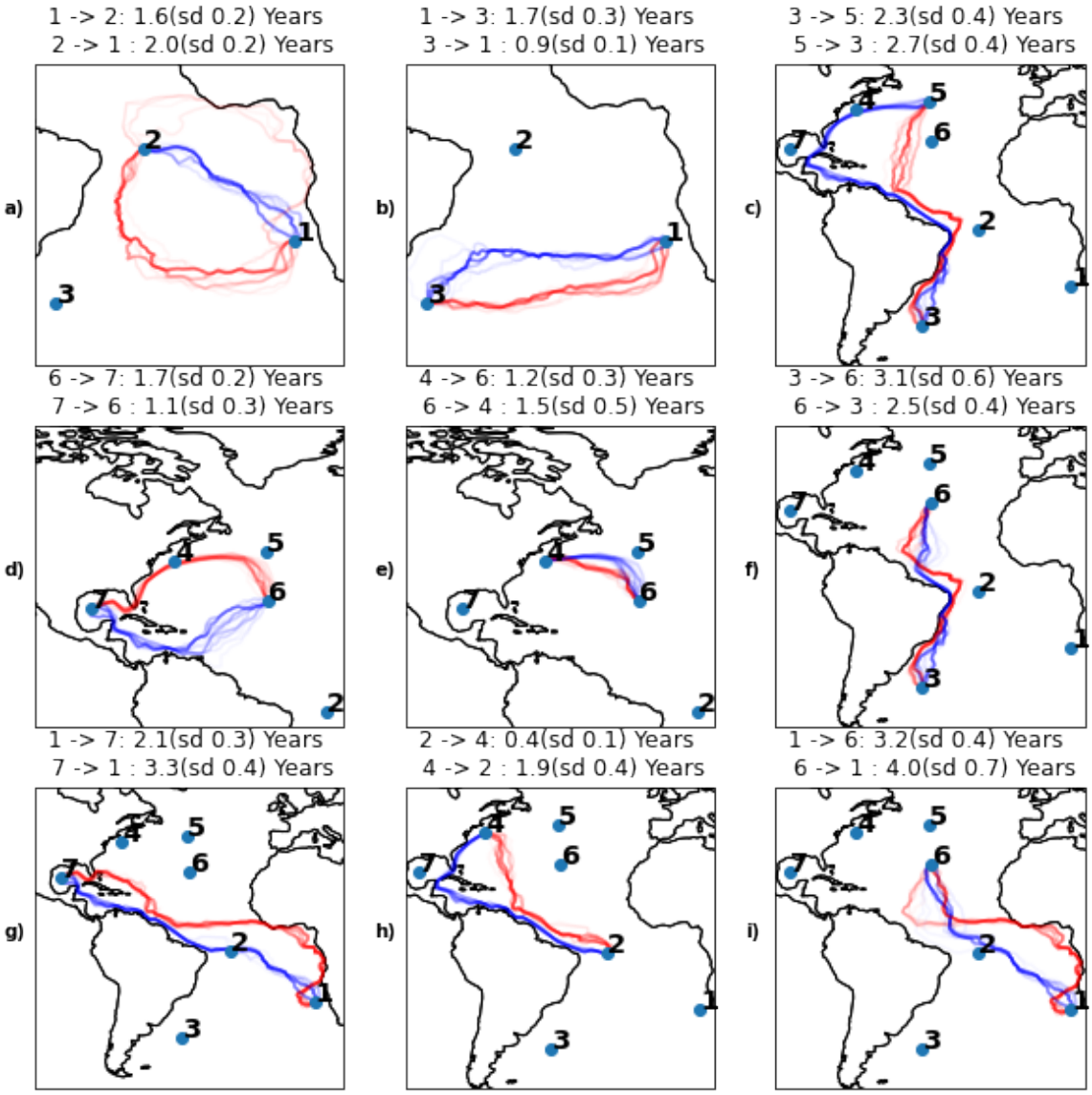

Figure B.H.3: This figure is similar to Figure 3.5.6 of the main text, however here we only use 60 rotations and use resolution 4 of the *H3* grid system. Each line connects the centroid of each hexagon within the path.

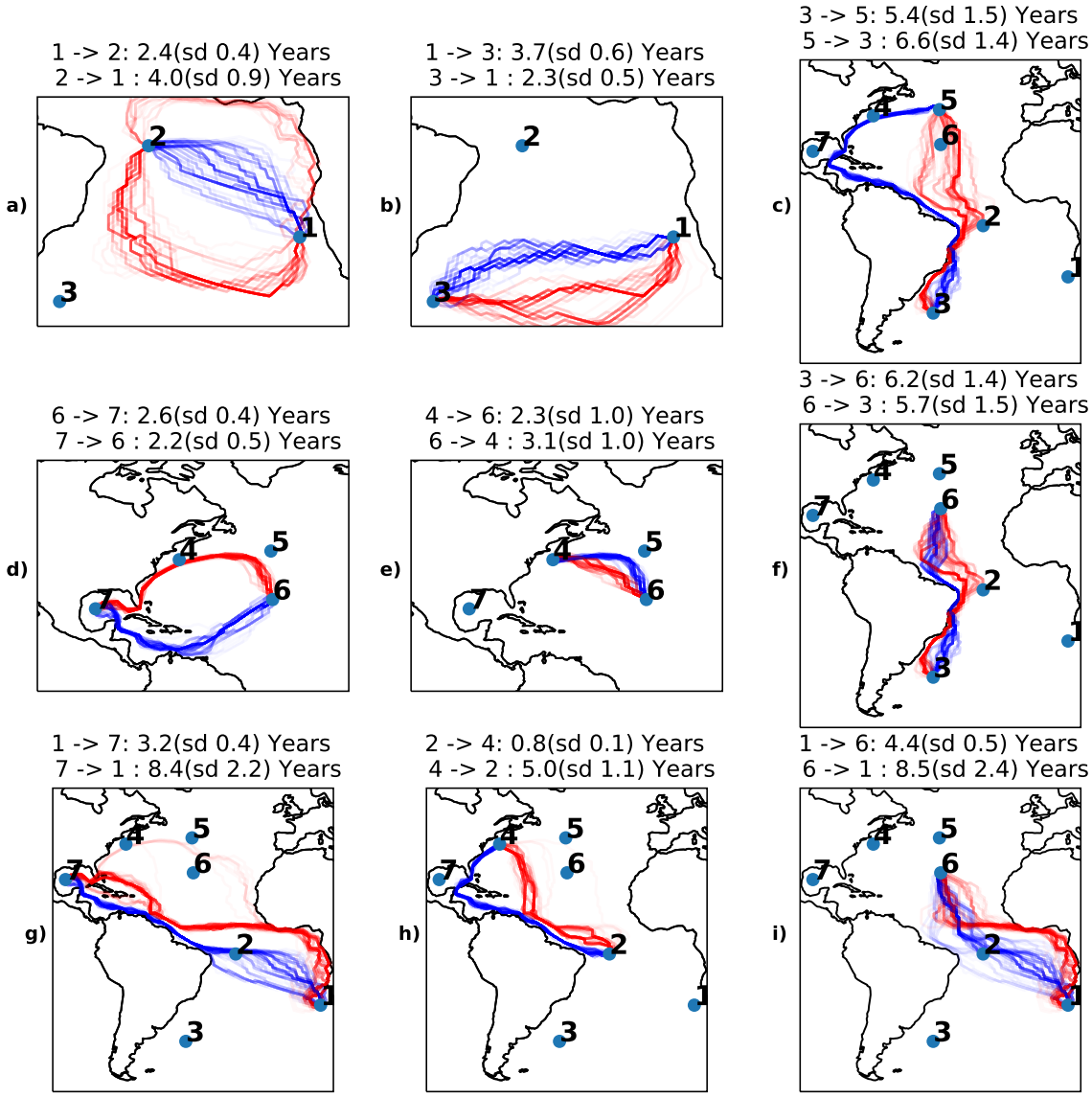

Figure B.H.4: This figure is similar to Figure 3.5.6 of the main text. Here we only use 100 bootstrap samples instead of rotations and use resolution 3 of the *H3* grid system. Each line connects the centroid of each hexagon within the path. Note all paths are in the same non-rotated grid system here.

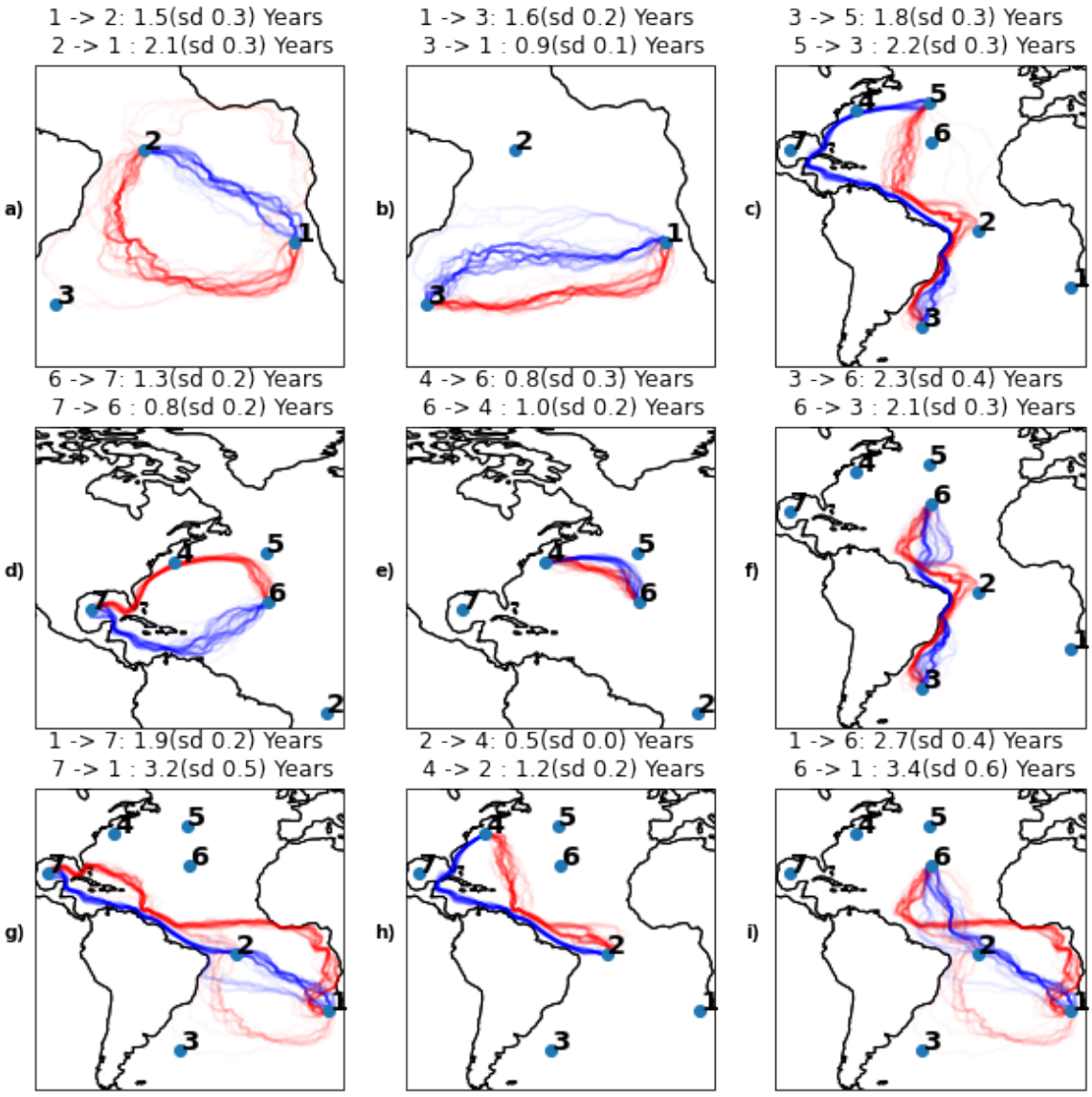

Figure B.H.5: This figure is similar to Figure 3.5.6 of the main text. Here we only use 100 bootstrap samples instead of rotations and use resolution 4 of the *H3* grid system. Each line connects the centroid of each hexagon within the path. Note all paths are in the same non-rotated grid system here.

the variance of the transition matrix entries increases due to less data being available to estimate transition probabilities. In the main body we present resolution 3 as this seems to balance this trade-off well in the global dataset, however in regional studies with high data density (such as data from numerous recent clustered drifter deployments), we imagine that resolution 4 might optimally balance this trade-off and reduce uncertainty in these regional studies.

B.H.3 Shortest Travel Time Map

We recreate the travel time map shown in Figure 3.5.2 of the main document, however this time we use the expected travel time as the objective of the shortest path algorithm. The new objective we use to find the optimal path and travel time is:

$$\hat{\mathbf{p}} = \arg\min_{\mathbf{p}\in\mathcal{P}} \sum_{i=0}^{n-1} \left(\frac{T_{p_i,p_i}}{T_{p_i,p_{i+1}}} + 1 \right), \tag{B.H.1}$$

which is the minimum of Equation (9) in the main text. We show the result of this in Figure B.H.6. As we are now directly looking for a minimum, this map is more directly comparable to the results shown in Figure 2 in Jönsson and Watson [2016]. The map shown in Figure B.H.6 is smoother than the results of Jönsson and Watson [2016].

The travel times from this map may be preferable to the most likely travel times map; for example if distances which obey the triangle inequality are desired. These are not used as part of the main text as we believe the expected travel time of the most likely path is a more suitable measure for most applications. Essentially, if there was a 'true' and non-Gaussian travel time distribution, the minimum travel time is

an estimate of the minimum of that distribution, whereas the travel time of the most likely path is more akin to a mode. This explains, in part, the larger travel times we obtain in Figure 3.5.2 of the main document, versus what we see here in Figure B.H.6 and in Figure 2 of Jönsson and Watson [2016].

B.H.4 Grid Size and Lagrangian cut off time sensitivity

To investigate the relationship between the assumed decorrelation time ($\mathcal{T}_L$), and the grid size, we show an extended version of the sensitivity analysis shown in Section B.F. Under each resolution we estimate the travel time matrix for a grid of values for $\mathcal{T}_L$. Then we estimate the Spearman and Pearson correlation (of the off-diagonal entries) to the travel times created at $\mathcal{T}_L = 5$ for that grid system. The results are shown in Figure B.H.7. We show the same correlation metric between each pair of grid systems in Figure B.H.8. We see the correlation values are at worst 0.68 which would still be interpreted as a moderate to strong positive correlation.

We also show the mean and variance of the travel times in each matrix produced under the five grid systems and a grid of values for $\mathcal{T}_L$ in Figure B.H.9. It can clearly be seen that the smaller grid systems (*H3* resolution 4 and $0.5° \times 0.5°$) are less robust to changes in value of $\mathcal{T}_L$. The mean and standard deviation show a downward trend as we increase the value of $\mathcal{T}_L$. This shows the added robustness of resolution 3 and motivates this choice in the main body of the chapter for this dataset (the Global Drifter Program). We recommend repeating such sensitivity analyses for use with different datasets, especially if applied to simulated trajectories or regional studies.

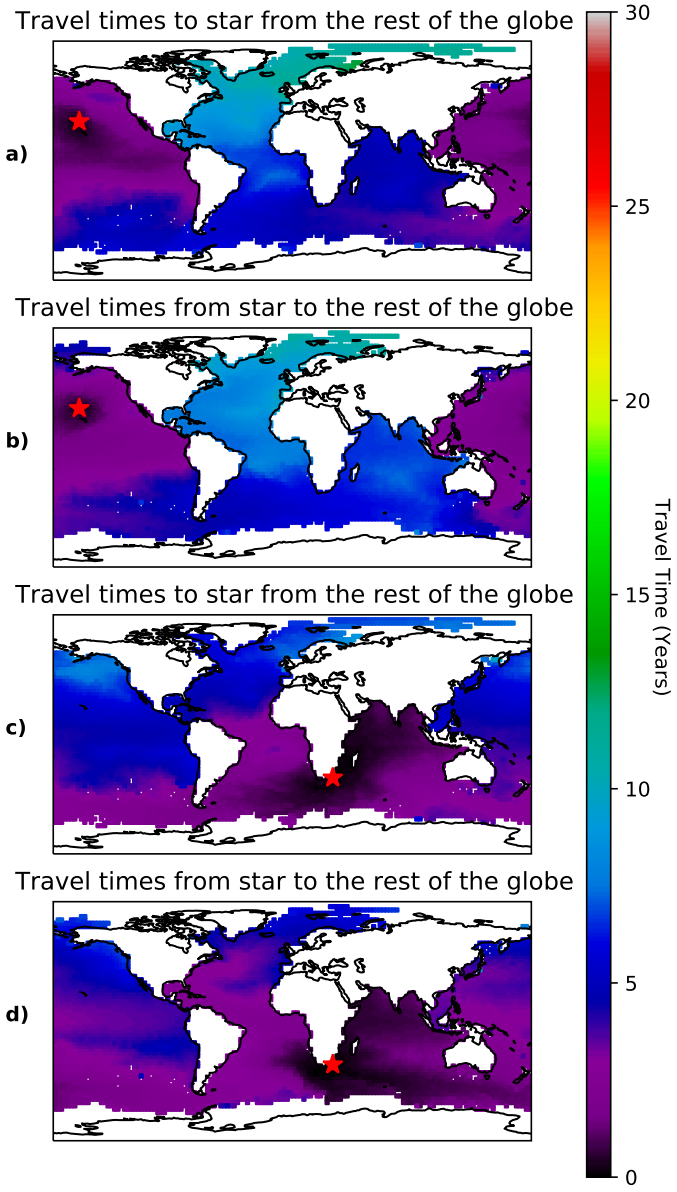

Figure B.H.6: Similar to Figure 3.5.2 of the main text, however rather than the travel time of the most likely path, this shows the shortest expected travel time.

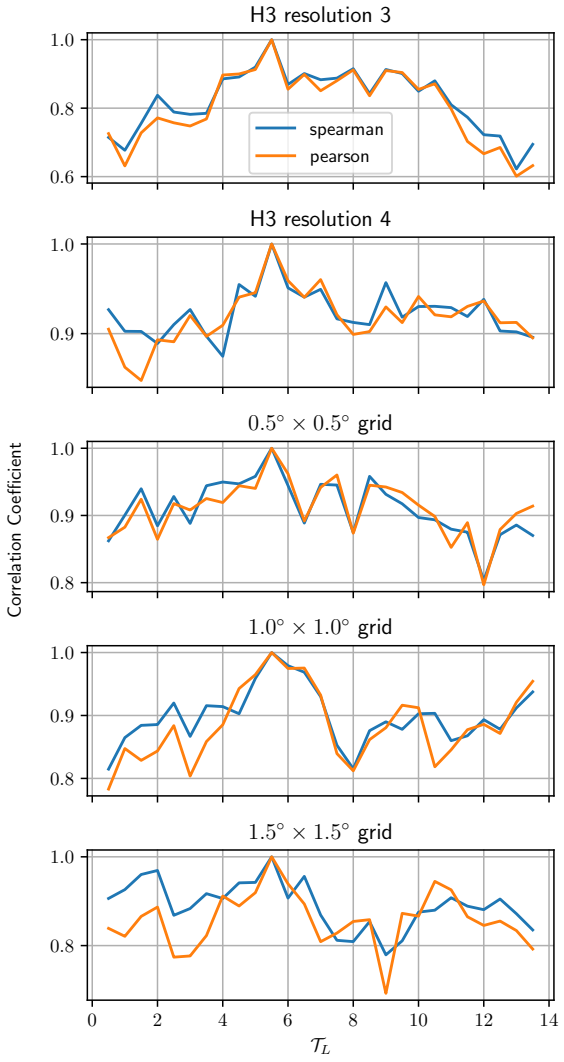

Figure B.H.7: A repeat of the experiment shown in Figure B.F.1 of the main text, however we show the results under both Spearman and Pearson correlations and under 5 different grid systems.

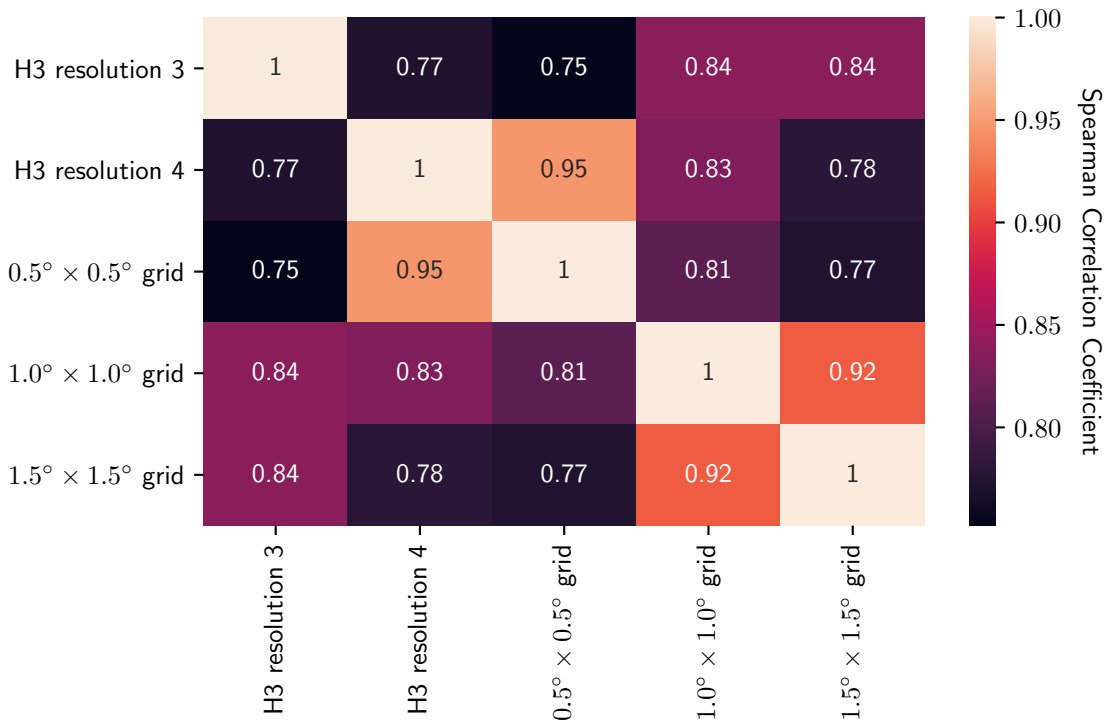

Figure B.H.8: The pairwise Spearman correlation value between the 42 travel time estimates produced by each grid system. We fix $\mathcal{T}_L = 5$ days.

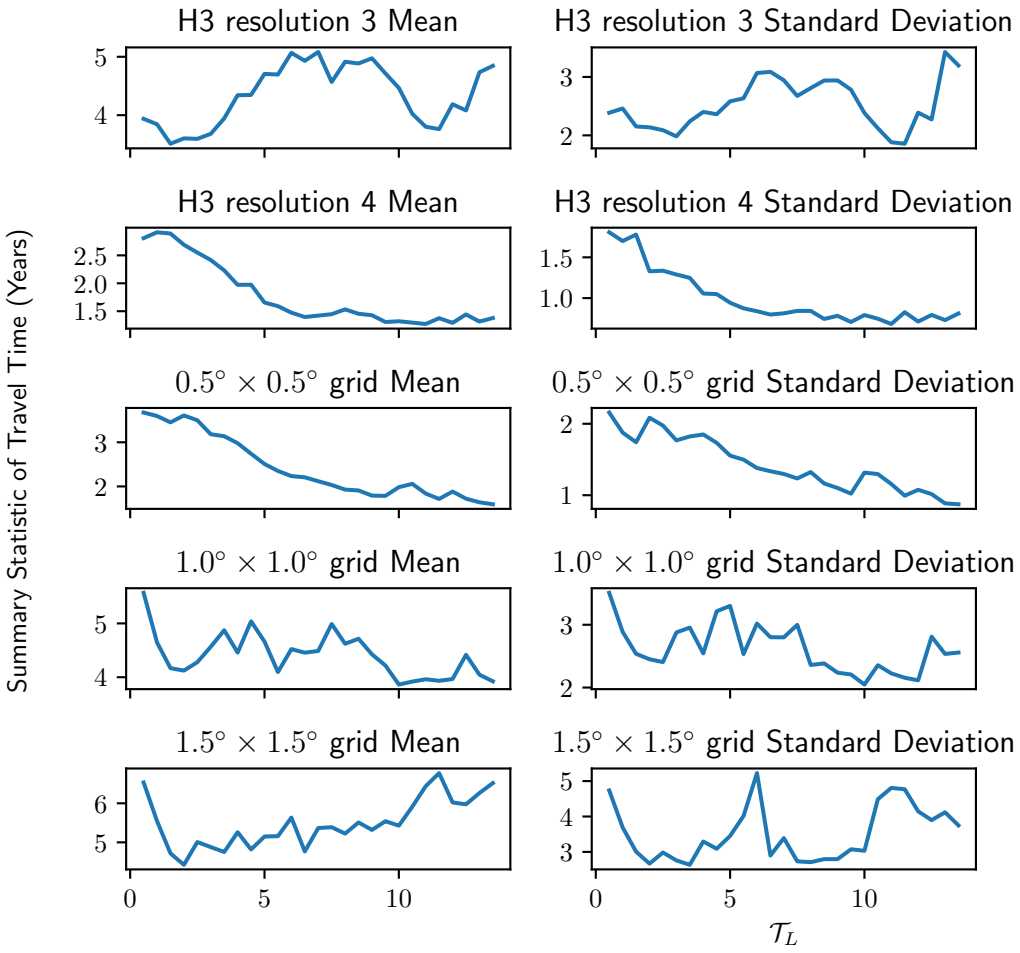

Figure B.H.9: The mean and standard deviation of the 42 travel time estimates used throughout the chapter. We show the mean and variance of these 42 travel time estimates in years for each combination of the 5 grid systems and 28 decorrelation times (0.5, 1, 1.5,..., 14 days).

B.H.5 Artificial connections

When running the analysis for the rotations of Section 3.5.3, if we do not take the preprocessing step of removing the two points on the Strait of Gibraltar, we find that some rotations allow this connection. In 71 of the 100 rotations we were unable to obtain a travel time estimate from the Atlantic into the Mediterranean and in 95 we were unable to find a travel time estimate from the Mediterranean to the Atlantic. When we do not do a rotation we are able to obtain an estimate into the Mediterranean, this is due to the way the grid aligns as shown in Figure 3.3.1. Even if only one of the 100 rotations are unable to provide an estimate it would be advisable to not use the estimate from this method. Therefore, using the vanilla method on its own to estimate travel times into the Mediterranean is not a good option.

Overall, the method provided depends on the availability of drifter data making a connection at some point. Connections such as going across the Strait of Gibraltar are in practice highly unlikely; any pathway which crosses it is due to a grid covering both the east and west of the Strait of Gibraltar. One potential way to adapt the method to approximate travel times across the Strait is, either adding artificial simulated trajectories as in van Sebille et al. [2012], or simply add a very small probability to the transition matrix crossing from the west to the east of the Strait of Gibraltar (and vice versa). For example, take two locations, one west and one east of the Strait of Gibraltar, say these correspond to states w and e respectively. If we wanted the crossing time to be 100 days into the Mediterranean sea, set $T_{e,w}$ such that $19 \times T_{e,w} = T_{e,e}$, the transition matrix will no longer be valid as the e row no longer sums to one

but the method will still work as intended, giving a 100 day crossing time from state e to w. Such an adaptation would require the removal of the state which covers the Strait of Gibraltar to force the algorithm to take the artificial 100 day crossing.

This example where the method detects the Mediterranean Sea's artificial connection is an interesting bonus feature of the rotation methodology, however, it is not as easily applicable to the Panama land mass problem. In the case of Panama, we will still obtain a travel time estimate from the Gulf of Mexico to the Pacific if a grid cell can cover both sides, but the times which are permitted to skip over the Panama land mass will be much shorter. An automatic detection could be achieved by looking at a large sample of rotations then running a test for multi modality. If it finds that there are two modes which are very far apart then this would be a sign that the method is finding some shortcut which is only present under some rotations. If such a method worked to detect the Panama land mass, we could then use it to search for more subtle surface transport barriers. In general it is preferable to preprocess the transition matrix T such that rows/columns corresponding to unwanted links such at the Panama Canal and the Strait of Gibraltar are simply removed, as we performed in our analysis. Visual detection of pathways will generally solve any issues.

B.I DriftMLP package documentation

As previously mentioned as part of this project we also developed a Python package named DriftMLP (Drift Most Likely Path). The package version is currently v1.2. The documentation for the package with examples can be found at `https://driftmlp.`

readthedocs.io/.

B.J Data Supplied for Laso-Jadart et al [2021]

The original goal and aims of this project was to supply a measure of Lagrangian separation between arbitrary locations in the ocean. In particular, we were interested in the Tara Oceans stations.

For a description of the final supplied product we quote the relevant paragraph from Laso-Jadart et al. [2021]: We extracted the data for both drogued and undrogued drifters (i.e. drifters that lost their sock) to maximize the information used by the method. No drifters have ever been observed to get out of the Mediterranean Sea through the Strait of Gibraltar, therefore to avoid missing data, we arbitrarily added 100 years to the travel times of pathways out of the Mediterranean Sea over the Strait of Gibraltar and added 1 year to the pathways going into the Mediterranean Sea, based on previous models on surface water. We used 450 rotations within the method to reduce the reliance of travel times on the grid system used. Two travel times are obtained by the method for each pair of stations: back and forth, resulting in an asymmetric travel time matrix between all possible station pairings. For our analyses, we retained only the minimum of these two travel times in the matrix, as this then accounts for the direction of currents between stations.

Bibliography

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

S.-I. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10 (2):251–276, 1998.

S. Banerjee and A. Roy. *Linear Algebra and Matrix Analysis for Statistics*. Chapman and Hall/CRC, London, 2014. ISBN 1420095382.

L. E. Becking, D. F. Cleary, N. J. de Voogd, W. Renema, M. de Beer, R. W. van Soest, and B. W. Hoeksema. Beta diversity of tropical marine benthic assemblages in the Spermonde Archipelago, Indonesia. *Marine Ecology*, 27(1):76–88, 2006.

F. Bentivegna, F. Valentino, P. Falco, E. Zambianchi, and S. Hochscheid. The relationship between loggerhead turtle (caretta caretta) movement patterns and mediterranean currents. *Marine Biology*, 151(5):1605–1614, 2007.

L. O. Berline, A. M. Rammou, A. Doglioli, A. Molcard, and A. Petrenko. A connectivity-based eco-regionalization method of the Mediterranean Sea. *PLoS ONE*, 9(11):1–9, 2014. ISSN 19326203.

F. J. Beron-Vera and J. LaCasce. Statistics of simulated and observed pair separations in the Gulf of Mexico. *Journal of Physical Oceanography*, 46(7):2183–2199, 2016.

P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Transactions on Sensor Networks (TOSN)*, 2(2):188–220, 2006.

F. Bonjean and G. S. E. Lagerloef. Diagnostic model and analysis of the surface currents in the tropical pacific ocean. *Journal of Physical Oceanography*, 32(10): 2938–2954, 2002.

I. Borg. *Modern multidimensional scaling : theory and applications*. Springer Series in Statistics. Springer, New York, 2005. ISBN 9780387251509.

A. W. Bowman. An alternative method of cross-validation for the smoothing of density estimates. *Biometrika*, 71(2):353–360, 1984. ISSN 0006-3444.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs. `http://github.com/google/jax`, 2018.

A. Charpentier and E. Flachaire. Log-transform kernel density estimation of income distribution. *Actualité économique*, 91(1-2):141–159, 2015. ISSN 0001-771X.

E. P. Chassignet, H. E. Hurlburt, O. M. Smedstad, G. R. Halliwell, P. J. Hogan, A. J. Wallcraft, R. Baraille, and R. Bleck. The HYCOM (hybrid coordinate ocean model) data assimilative system. *Journal of Marine Systems*, 65(1-4):60–83, 2007.

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

T. F. Cox. *Multidimensional scaling*. Monographs on Statistics and Applied Probability. Chapman and Hall/CRC, Boca Raton, Fla., 2001. ISBN 1584880945.

J. A. Dambon, F. Sigrist, and R. Furrer. Maximum likelihood estimation of spatially varying coefficient models for large data with an application to real estate price prediction. *Spatial Statistics*, 41:100470, 2021.

E. D'Asaro, C. Guigand, A. Haza, H. Huntley, G. Novelli, T. Özgökmen, and E. Ryan. [Dataset] Lagrangian submesoscale experiment (LASER) surface drifters, interpolated to 15-minute intervals. *Distributed by: Gulf of Mexico Research Initiative Information and Data Cooperative (GRIIDC), Harte Research Institute, Texas A&M University–Corpus Christi*, 2017. doi: https://doi:10.7266/N7W0940J.

J. de Leeuw. Applications of convex analysis to multidimensional scaling. *eScholarship, University of California*, 2005.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, pages 269–271, 1959. ISSN 0945-3245.

P. Drineas, A. Javed, M. Magdon-Ismail, G. Pandurangan, R. Virrankoski, and A. Savvides. Distance matrix reconstruction from incomplete distance information for sensor network localization. In *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, pages 536–544, 2006. ISBN 1424406269.

K. L. Drouin, M. S. Lozier, F. J. Beron-Vera, P. Miron, and M. J. Olascoaga. Surface pathways connecting the South and North Atlantic Oceans. *Geophysical Research Letters*, 49(1), 2022.

T. Duan, A. Anand, D. Y. Ding, K. K. Thai, S. Basu, A. Y. Ng, and A. Schuler. Ngboost: Natural gradient boosting for probabilistic prediction. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 2690–2700. PMLR, 2020.

J. J. Early and A. M. Sykulski. Smoothing and interpolating noisy GPS data with smoothing splines. *Journal of Atmospheric and Oceanic Technology*, 37(3):449–465, 2020.

B. Efron. *An introduction to the bootstrap*. Monographs on Statistics and Applied Probability. Chapman & Hall, New York, 1993. ISBN 0412042312.

S. Elipot, R. Lumpkin, and G. Prieto. Modification of inertial oscillations by the mesoscale eddy field. *Journal of Geophysical Research: Oceans*, 115(9):1–20, 2010. ISSN 21699291.

S. Elipot, R. Lumpkin, R. C. Perez, J. M. Lilly, J. J. Early, and A. M. Sykulski. A

global surface drifter data set at hourly resolution. *Journal of Geophysical Research: Oceans*, 121(5):2937–2966, 2016.

S. Elipot, A. Sykulski, R. Lumpkin, L. Centurioni, and M. Pazos. A dataset of hourly sea surface temperature from drifting buoys. *arXiv preprint arXiv:2201.08289*, 2022.

K. Ellingsen and J. Gray. Spatial patterns of benthic diversity: Is there a latitudinal gradient along the Norwegian continental shelf? *Journal of Animal Ecology*, 71: 373 – 389, 2002.

Flanders Marine Institute. IHO sea areas, version 3, 2018. Available online at `https://www.marineregions.org/`.

D. M. Fratantoni. North Atlantic surface circulation during the 1990's observed with satellite-tracked drifters. *Journal of Geophysical Research: Oceans*, 106(C10): 22067–22093, 2001.

J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

G. Froyland, K. Padberg, M. H. England, and A. M. Treguier. Detection of coherent oceanic structures via transfer operators. *Physical review letters*, 98(22):224503, 2007.

G. Froyland, R. M. Stuart, and E. van Sebille. How well-connected is the surface of the global ocean? *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 24 (3):033126, 2014.

G. Gallo and S. Pallottino. Shortest path algorithms. *Annals of Operations Research*, 13(1):1–79, 1988.

Y. Gao. Shortest path problem with uncertain arc lengths. *Computers & Mathematics with Applications*, 62(6):2591–2600, 2011.

T. Gasser and H.-G. Müller. Kernel estimation of regression functions. In *Smoothing techniques for curve estimation*, pages 23–68. Springer, 1979.

T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

S. Guinehut, A.-L. Dhomps, G. Larnicol, and P.-Y. Le Traon. High resolution 3-D temperature and salinity fields derived from in situ and satellite observations. *Ocean Science*, 8(5):845–857, 2012.

E. M. Hanks, M. B. Hooten, and M. W. Alldredge. Continuous-time discrete-space models for animal movement. *The Annals of Applied Statistics*, 9(1):145–165, 2015. ISSN 1932-6157.

D. V. Hansen and P.-M. Poulain. Quality control and interpolations of WOCE-TOGA drifter data. *Journal of Atmospheric and Oceanic Technology*, 13(4):900–909, 1996.

T. Hastie and R. Tibshirani. Varying-coefficient models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 55(4):757–796, 1993. ISSN 00359246.

M. J. Heaton, A. Datta, A. O. Finley, R. Furrer, J. Guinness, R. Guhaniyogi, F. Gerber, R. B. Gramacy, D. Hammerling, M. Katzfuss, et al. A case study competition

among methods for analyzing large spatial data. *Journal of Agricultural, Biological and Environmental Statistics*, 24(3):398–425, 2019.

H. S. Huntley, B. Lipphardt Jr, and A. Kirwan Jr. Lagrangian predictability assessed in the East China Sea. *Ocean Modelling*, 36(1-2):163–178, 2011.

B. F. Jönsson and J. R. Watson. The timescales of global surface-ocean connectivity. *Nature Communications*, 7(1):1–6, 2016.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 3146–3154, 2017.

J. J. Kennedy, N. Rayner, C. Atkinson, and R. Killick. An ensemble data set of sea surface temperature change from 1850: The Met Office Hadley Centre HadSST. 4.0.0.0 data set. *Journal of Geophysical Research: Atmospheres*, 124(14):7719–7763, 2019.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, 2015.

J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

J. H. LaCasce. Statistics from Lagrangian observations. *Progress in Oceanography*, 77(1):1–29, 2008.

R. Laso-Jadart, M. O'Malley, A. M. Sykulski, C. Ambroise, and M.-A. Madoui. How marine currents and environment shape plankton genomic differentiation: a mosaic view from tara oceans metagenomic data. *bioRxiv*, 2021. doi: 10.1101/2021.04.29.441957.

L. C. Laurindo, A. J. Mariano, and R. Lumpkin. An improved near-surface velocity climatology for the global ocean from drifter observations. *Deep Sea Research Part I: Oceanographic Research Papers*, 124:73–92, 2017.

K. V. Lebedev, H. Yoshinari, N. A. Maximenko, and P. W. Hacker. Velocity data assessed from trajectories of Argo floats at parking level and at the sea surface. *IPRC Technical Note*, 4(2), 2007.

M. Lei, L. Shiyan, J. Chuanwen, L. Hongling, and Z. Yan. A review on the forecasting of wind speed and generated power. *Renewable and Sustainable Energy Reviews*, 13(4):915–920, 2009.

R. Li, B. J. Reich, and H. D. Bondell. Deep distribution regression. *Computational Statistics & Data Analysis*, 159:107203, 2021. ISSN 0167-9473.

R. Lumpkin and L. Centurioni. Global Drifter Program quality-controlled 6-hour interpolated data from ocean surface drifting buoys. NOAA National Centers for Environmental Information., 2020. Dataset. Accessed 22-01-2022.

R. Lumpkin and G. C. Johnson. Global ocean surface velocities from drifters: Mean, variance, El Niño-Southern Oscillation response, and seasonal cycle. *Journal of Geophysical Research. Oceans*, 118(6):2992–3006, 2013. ISSN 2169-9275.

R. Lumpkin and M. Pazos. Measuring surface currents with surface velocity program drifters: The instrument, its data, and some recent results. *Lagrangian Analysis and Prediction of Coastal and Ocean Dynamics*, 2:39–67, 2007.

R. Lumpkin, A.-M. Treguier, and K. Speer. Lagrangian eddy scales in the Northern Atlantic Ocean. *Journal of Physical Oceanography*, 32(9):2425–2440, 2002.

R. Lumpkin, N. Maximenko, and M. Pazos. Evaluating where and why drifters die. *Journal of Atmospheric and Oceanic Technology*, 29(2):300–308, 2012.

R. Lumpkin, S. A. Grodsky, L. Centurioni, M.-H. Rio, J. A. Carton, and D. Lee. Removing spurious low-frequency variability in drifter velocities. *Journal of Atmospheric and Oceanic Technology*, 30(2):353–360, 2013.

R. Lumpkin, L. Centurioni, and R. C. Perez. Fulfilling observing system implementation requirements with the global drifter array. *Journal of Atmospheric and Oceanic Technology*, 33(4):685–695, 2016.

R. Lumpkin, T. Özgökmen, and L. Centurioni. Advances in the application of surface drifters. *Annual Review of Marine Science*, 9:59–81, 2017.

L. Malagò and G. Pistone. Information geometry of the Gaussian distribution in view of stochastic optimization. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, pages 150–162, 2015.

J. Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020. URL `http://jmlr.org/papers/v21/17-678.html`.

N. Maximenko, P. Niiler, L. Centurioni, M.-H. Rio, O. Melnichenko, D. Chambers, V. Zlotnicki, and B. Galperin. Mean dynamic topography of the Ocean derived from satellite and drifting buoy data using three different techniques. *Journal of Atmospheric and Oceanic Technology*, 26(9):1910–1919, 2009.

N. Maximenko, J. Hafner, and P. Niiler. Pathways of marine debris derived from trajectories of Lagrangian drifters. *Marine Pollution Bulletin*, 65(1-3):51–62, 2012.

R. McAdam and E. van Sebille. Surface connectivity and interocean exchanges from drifter-based transition matrices. *Journal of Geophysical Research: Oceans*, 123(1): 514–532, 2018.

G. A. Meehl. Characteristics of surface current flow inferred from a global ocean current data set. *Journal of Physical Oceanography*, 12(6):538–555, 1982.

N. Meinshausen and G. Ridgeway. Quantile regression forests. *Journal of Machine Learning Research*, 7(6), 2006.

P. Miron, F. J. Beron-Vera, M. J. Olascoaga, J. Sheinbaum, P. Pérez-Brunius, and G. Froyland. Lagrangian dynamical geography of the Gulf of Mexico. *Scientific Reports*, 7(1):1–12, 2017.

P. Miron, F. J. Beron-Vera, M. J. Olascoaga, and P. Koltai. Markov-chain-inspired search for MH370. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29 (4), 2019.

S. Mulet, M.-H. Rio, H. Etienne, C. Artana, M. Cancet, G. Dibarboure, H. Feng,

R. Husson, N. Picot, C. Provost, et al. The new CNES-CLS18 global mean dynamic topography. *Ocean Science Discussions*, pages 1–31, 2021.

D. Nychka, S. Bandyopadhyay, D. Hammerling, F. Lindgren, and S. Sain. A multiresolution Gaussian process model for the analysis of large spatial datasets. *Journal of Computational and Graphical Statistics*, 24(2):579–599, 2015.

M. O'Malley, A. M. Sykulski, R. Lumpkin, and A. Schuler. Multivariate probabilistic regression with natural gradient boosting. *arXiv preprint arXiv:2106.03823*, 2021.

S. Oscroft, A. M. Sykulski, and J. J. Early. Separating mesoscale and submesoscale flows from clustered drifter trajectories. *Fluids*, 6(1):14, 2021.

T. Özgökmen. [Dataset]Grand Lagrangian deployment (GLAD) experiment CODE-style and flat surface drifter trajectories. *Distributed by: Gulf of Mexico Research Initiative Information and Data Cooperative (GRIIDC), Harte Research Institute, Texas A&M University–Corpus Christi.*, 2016. doi: https://doi:10.7266/N7086388.

M. O'Malley, A. M. Sykulski, R. Laso-Jadart, and M.-A. Madoui. Estimating the travel time and the most likely path from Lagrangian drifters. *Journal of Atmospheric and Oceanic Technology*, 38(5):1059–1073, 2021.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

S. Pesant, F. Not, M. Picheral, S. Kandels-Lewis, N. Le Bescot, G. Gorsky, D. Iudicone, E. Karsenti, S. Speich, R. Troublé, et al. Open science resources for the discovery and analysis of tara oceans data. *Scientific Data*, 2(1):1–16, 2015.

PROJ contributors. *PROJ coordinate transformation software library*. Open Source Geospatial Foundation, 2022. URL `https://proj.org/`.

J. O. Ramsay. Some statistical approaches to multidimensional scaling data. *Journal of the Royal Statistical Society. Series A (General)*, 145(3):285, 1982. ISSN 00359238.

J. O. Ramsay. Monotone regression splines in action. *Statistical Science*, 3(4):425–441, 1988.

J. O. Ramsay. Estimating smooth monotone functions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(2):365–375, 1998.

S. Rasp and S. Lerch. Neural networks for postprocessing ensemble weather forecasts. *Monthly Weather Review*, 146(11):3885–3900, 2018.

J. Richards and J. L. Wadsworth. Spatial deformation for nonstationary extremal dependence. *Environmetrics*, 2021.

R. A. Rigby, M. D. Stasinopoulos, G. Z. Heller, and F. De Bastiani. *Distributions for modeling location, scale, and shape: Using GAMLSS in R*. CRC press, 2019.

M.-H. Rio, S. Mulet, and N. Picot. Beyond GOCE for the ocean circulation estimate: Synergetic use of altimetry, gravimetry, and in situ data provides new insight into

geostrophic and Ekman currents. *Geophysical Research Letters*, 41(24):8918–8925, 2014.

M. Rudemo. Empirical choice of histograms and kernel density estimators. *Scandinavian Journal of Statistics*, 9:65–78, 1982.

I. I. Rypina, D. Fertitta, A. MacDonald, S. Yoshida, and S. Jayne. Multi-iteration approach to studying tracer spreading using drifter data. *Journal of Physical Oceanography*, 47(2):339–351, 2017. ISSN 15200485.

H. Salimbeni, S. Eleftheriadis, and J. Hensman. Natural gradients in practice: Non-conjugate variational inference in gaussian process models. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, volume 84 of *Proceedings of Machine Learning Research*, pages 689–697, 2018.

P. D. Sampson and P. Guttorp. Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, 87(417):108–119, 1992.

E. F. Schuster. Incorporating support constraints into nonparametric estimators of densities. *Communications in Statistics-Theory and methods*, 14(5):1123–1136, 1985.

D. W. Scott. *Multivariate density estimation : theory, practice, and visualization.* Wiley series in probability and mathematical statistics. Wiley, New York, 1992. ISBN 0471547700.

E. Ser-Giacomi, V. Rossi, C. López, and E. Hernández-García. Flow networks: A characterization of geophysical fluid transport. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 25(3):036404, 2015a.

E. Ser-Giacomi, R. Vasile, E. Hernández-García, and C. López. Most probable paths in temporal weighted networks: An application to ocean transport. *Physical review E*, 92(1):012818, 2015b.

Y. Shang and W. Ruml. Improved MDS-based localization. In *IEEE INFOCOM 2004*, volume 4, pages 2640–2651. IEEE, 2004. ISBN 0780383559.

A. Y. Shcherbina, M. A. Sundermeyer, E. Kunze, E. D'Asaro, G. Badin, D. Birch, A.-M. E. Brunner-Suzuki, J. Callies, B. T. Kuebel Cervantes, M. Claret, et al. The LatMix summer campaign: Submesoscale stirring in the upper ocean. *Bulletin of the American Meteorological Society*, 96(8):1257–1279, 2015.

K. Shoemake. Uniform random rotations. In *Graphics Gems III (IBM Version)*, pages 124–132. Elsevier, 1992.

A. Sinha and R. Abernathey. Estimating ocean surface currents with machine learning. *Frontiers in Marine Science*, 8, 2021. ISSN 2296-7745.

T. M. Smith, P. H. York, B. R. Broitman, M. Thiel, G. C. Hays, E. van Sebille, N. F. Putman, P. I. Macreadie, and C. D. Sherman. Rare long-distance dispersal of a marine angiosperm across the Pacific Ocean. *Global ecology and biogeography*, 27 (4):487–496, 2018. ISSN 1466-822X.

E. A. Sützle and T. Hrycej. Numerical method for estimating multivariate conditional distributions. *Computational statistics*, 20(1):151–176, 2005.

A. M. Sykulski, S. C. Olhede, J. M. Lilly, and E. Danioux. Lagrangian time series models for ocean surface drifter trajectories. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 65(1):29–50, 2016.

Thematic Assembly Centers. [Dataset] Global ocean gridded L4 sea surface heights and derived variables reprocessed. E.U. Copernicus Marine Service Information. Available at: `https://resources.marine.copernicus.eu/?option=com_csw&view=details&product_id=SEALEVEL_GLO_PHY_L4_REP_OBSERVATIONS_008_047`, Accessed: 2nd December 2020, 2020a.

Thematic Assembly Centers. [Dataset] Global ocean wind L4 reprocessed 6 hourly observations. E.U. Copernicus Marine Service Information. Available at: `https://resources.marine.copernicus.eu/?option=com_csw&view=details&product_id=WIND_GLO_WIND_L4_REP_OBSERVATIONS_012_006`, Accessed: 2nd December 2020, 2020b.

W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952. ISSN 0033-3123.

UBER. H3 spatial index. `https://eng.uber.com/h3/`, 2019. Accessed: 2020-01-08.

S. M. Ulam. *A collection of mathematical problems.* Interscience Tracts in Pure and Applied Mathematics. Interscience Publishers, New York, 1960.

E. van Sebille. Adrift.org.au - A free, quick and easy tool to quantitatively study planktonic surface drift in the global ocean. *Journal of Experimental Marine Biology and Ecology*, 461:317–322, 2014.

E. van Sebille, P. J. van Leeuwen, A. Biastoch, C. Barron, and W. de Ruijter. Lagrangian validation of numerical drifter trajectories using drifting buoys: Application to the Agulhas system. *Ocean Modelling*, 29(4):269–276, jan 2009. ISSN 1463-5003.

E. van Sebille, L. M. Beal, and W. E. Johns. Advective time scales of Agulhas leakage to the North Atlantic in surface drifter observations and the 3D OFES model. *Journal of Physical Oceanography*, 41(5):1026–1034, 2011.

E. van Sebille, M. H. England, and G. Froyland. Origin, dynamics and evolution of ocean garbage patches from observed surface drifters. *Environmental Research Letters*, 7(4), 2012. ISSN 17489326.

E. van Sebille, S. M. Griffies, R. Abernathey, T. P. Adams, P. Berloff, A. Biastoch, B. Blanke, E. P. Chassignet, Y. Cheng, C. J. Cotter, E. Deleersnijder, K. Döös, H. F. Drake, S. Drijfhout, S. F. Gary, A. W. Heemink, J. Kjellsson, I. M. Koszalka, M. Lange, C. Lique, G. A. MacGilchrist, R. Marsh, C. G. M. Adame, R. McAdam, F. Nencioli, C. B. Paris, M. D. Piggott, J. A. Polton, S. Rühs, S. H. Shah, M. D. Thomas, J. Wang, P. J. Wolfram, L. Zanna, and J. D. Zika. Lagrangian ocean analysis: Fundamentals and practices. *Ocean Modelling*, 121:49–75, 2018. ISSN 1463-5003.

Y. Wakata and Y. Sugimori. Lagrangian motions and global density distributions of floating matter in the ocean simulated using shipdrift data. *Journal of Physical oceanography*, 20:125–138, 1990.

M. P. Wand, J. S. Marron, and D. Ruppert. Transformations in density estimation. *Journal of the American Statistical Association*, 86(414):343–353, 1991.

J. Wang. *Geometric Structure of High-Dimensional Data and Dimensionality Reduction*. Springer Berlin Heidelberg, 2012. ISBN 9783642274961.

J. C. Wang and T. Hastie. Boosted varying-coefficient regression models for product demand prediction. *Journal of Computational and Graphical Statistics*, 23(2):361–382, 2014.

J. R. Watson. The geography of the World's oceans explains patterns of planktonic diversity. In *2018 Ocean Sciences Meeting*. AGU, 2018.

C. White, K. A. Selkoe, J. Watson, D. A. Siegel, D. C. Zacherl, and R. J. Toonen. Ocean currents help explain population genetic structure. *Proceedings of the Royal Society B: Biological Sciences*, 277(1688):1685–1694, 2010.

P. M. Williams. Using neural networks to model conditional multivariate densities. *Neural computation*, 8(4):843–854, 1996.

S. N. Wood. *Generalized additive models: an introduction with R*. Chapman and Hall/CRC, 2017.

Z. Zhang, C. Y. Lim, and T. Maiti. Analyzing 2000–2010 childhood age-adjusted

cancer rates in Florida: a spatial clustering approach. *Statistics and Public Policy*, 1(1):120–128, 2014.

V. Zhurbas and I. S. Oh. Drifter-derived maps of lateral diffusivity in the Pacific and Atlantic oceans in relation to surface circulation patterns. *Journal of Geophysical Research: Oceans*, 109(C5), 2004.