Forecasting Complex Systems Using Stochastic Models for Low Dimensional Approximations

Paul Smith

The University of Leeds

School of Mathematics

Submitted in accordance with the requirements for the degree of

Doctor of Philosophy

August 2020

To Mum and Dad, for all your support.

Acknowledgements

I would like to thank my supervisors, Jochen Voss and Elena Issoglio, who have supported me throughout my PhD. You have been selfless with your time and knowledge, and have made my research an enjoyable experience. I could not have asked for better people to help me through my time at Leeds. Thank you. I would also like to thank Tiziano De Angelis for the many enjoyable conversations we have had, and for being gracious in defeat on the squash court.

The friends I have met in Leeds have made my time here very special. You allowed me to switch off from work, explore the countryside, and enjoy your company. Thank you for this.

Finally, I would like to give a special thanks to Eva, for your infectious smile, outlook on life, and for putting up with me over these last few months.

Abstract

Computer simulators are often used to model complex systems, which by their very nature are in general expensive to run and hard to condition. Statistical methods can be used to simplify and help analyse these simulator outputs, with the aim of increasing understanding of the underlying complex system and producing simplified forecasts of this system. This work investigates some statistical methods that can be used, and applies these methods to some output from the HadCM3 climate simulator.

In Part I of this thesis, the statistical framework is introduced, and can be split broadly into the areas of *dimension reduction*, *stochastic modelling*, and *forecasting*. Within the dimension reduction section, the fastICA independent component analysis method is examined with some definciencies highlighted. A novel independent component analysis method is introduced, called clusterICA, which uses clustering in the projective space and Householder reflections to obtain independent directions.

Modelling of the components found using dimension reduction is then examined. This includes using a block-average Ornstein-Uhlenbeck process, where the pointwise Ornstein-Uhlenbeck process is integrated over disjoint time intervals. As the first step of modelling involves removing seasonality, a novel spline function is introduced that can be used to ensure seasonal means between pointwise and block-average Ornstein-Uhlenbeck processes remain equal. Forecasts using the distributional properties of the block-average Ornstein-Uhlenbeck are also examined.

In Part II of this thesis, methods introduced in Part I are applied to output obtained from the HadCM3 climate simulator. Two different forecasting methods are used to obtain forecasts of the sets of

low dimensional components, and these forecasts are used to reconstruct the full high dimensional simulator output. These forecasted reconstructions are compared to the reconstructions using the true low dimensional components, and the full HadCM3 simulator output.

Contents

Introduction

This thesis describes the makings of an approach to simplify simulators of high dimensional, complex systems, by the use of statistical methods. A *complex system* is a system that contains many interacting variables, with these interactions having the ability to alter the state of the system in a highly sensitive, potentially chaotic way. Examples of complex systems include the Earth's climate, urban transport networks, and the brain. Complex systems can be approximated by the use of *simulators*, where a mathematical or physical model is used to attempt to mimic the behaviour of the associated system.

Throughout this thesis, we use the example of a *climate simulator* that aims to approximate some aspects of the Earth's climate, for example the temperature or atmospheric pressure. These simulators obtain some output through the use of a combination of physical laws, heuristics rules and expert judgements, alongside actual measurements (for example, historic CO_2 levels extracted from ice-cores). There is a huge amount of uncertainty in these kind of large-scale simulators, from the spatial and temporal uncertainty in the input measurements, the uncertainty in the expert judgements, and the uncertainty in the interactions of the equations within the model, to give just a few. Here, the difficulties with dealing with output from these high dimensional simulators are discussed, with the overarching aim being to find methods that allow for simple forecasts of the simulator to be calculated.

The climate simulator examined in the latter part of this thesis is the HadCM3 climate simulator. These simulators are continually being improved, although a by-product of this improvement is often an increase in the model complexity with the addition of more interacting systems, or an increase in resolution with smaller grid cells and more atmospheric stratification. This can result in climate simulators diving further into the “black box” territory and therefore becoming increasingly difficult to interpret whilst at the same time taking longer to perform simulator runs.

In this thesis, the use of statistical methods to enable the simplification of simulator runs are explored, with the aim of increasing understandability, allowing for more flexible simulator runs, and for simple simulator forecasting. The statistical methods used here can be split into three supplementary themes: *dimensional reduction*, *stochastic modelling*, and *forecasting*. The techniques introduced in this thesis within each of these themes build on methods established in the previous theme. That is, the dimension reduction techniques can be used to obtain some low dimensional output from some high dimensional simulator run, which can then be approximated using the stochastic modelling techniques. Then, forecasting techniques can be applied using these stochastic models to obtain forecasts for the low dimensional output. In all of these areas, this thesis examines interpretable methods that can be performed on high dimensional simulator outputs, both from classical statistics and more novel ideas.

In Part I of this thesis, we introduce the three main themes that are used to enable simulator output simplification and forecasting. In addition to this, we critique some established statistical methods and examine their weaknesses, leading to the foundations of some novel methods that attempt to counteract these weaknesses. The underlying behaviour of the climate simulator output used in Part II is kept in mind throughout Part I, most notably that the output is not given pointwise but as averages over disjoint time periods. This leads to the proposal of stochastic processes and associated forecasts that take this behaviour into account.

In Part I, Chapter 1, we examine methods, collectively called dimension reduction techniques, that reduce the complexity of some high dimensional simulator output. This is broadly split into the methods of *principal component analysis* and *independent component analysis*. Within the independent component analysis discussion, we consider the popular *fastICA* method and find it to be deficient in certain areas. This leads to the idea of a novel method called *clusterICA*, with the aim of removing some of the weaknesses found in *fastICA*. In Chapter 2 we introduce stochastic processes that can be applied to the outputs obtained from the dimension reduction techniques, using the idea that some of the information lost by applying the methods from Chapter 1 can be replaced with stochasticity. Here we introduce block-average Ornstein-Uhlenbeck processes which merge classical techniques with more novel ones that explicitly use the averaging behaviour of the climate simulator outputs seen in Part II. In Chapter 3 we obtain forecasts that can be applied to the outputs from the dimension reduction methods, using the models introduced in Chapter 2, including the block-average Ornstein-Uhlenbeck process.

Part II applies the framework introduced in Part I to two climate simulator runs, obtained from the HadCM3 simulator, with different outputs. This allows comparisons both within each simulator run – judging how the different statistical models compare to one-anther – and across the simulator runs. In Chapter 4 the HadCM3 simulator output consists of the mean sea-level air pressure over (a discretised form of) the Earth's surface. In Chapter 5 the HadCM3 simulator output consists of the mean sea-level air pressure with the addition of two-dimensional (horizontal) wind velocity.

The main finding of this thesis is that a subset of the methods considered result in forecasts that closely mirror the "true" observations. In these successful cases, a large proportion of the forecasting accuracy comes from some very simple, easily applicable techniques.

All the code used throughout this thesis has been written in `R`, with the most important functions available from `https://github.com/pws3141/thesisCode`.

Part I

Theory

Chapter 1

Dimension Reduction

1.1 Introduction to Dimension Reduction

The first chapter of this thesis introduces methods to reduce the dimensionality of a simulator output or a data set. Here a *simulator* is short for a *computer* simulator that attempts to use mathematical modelling to approximate some behaviour or event. These simulators are often used for forecasting, for example a weather simulator could be used to obtain a prediction for the temperature in Leeds next week. This section is motivated by an initial question that arises from a large simulator output: *what parts of this output are "important", and what can be ignored with relatively little loss?* This question can be reversed to ask, *where is computational time being used wisely in the model, and where is it being wasted?* These are the two questions that we attempt to be answered in this chapter, using some classical dimension reduction methods.

The simulator can be represented by the (deterministic) function,

$$f\colon \mathcal{M} \to \mathbb{R}^p, \tag{1.1}$$

where $\mathcal{M}$ is the $\underline{\text{m}}$odel input parameter space, and for any $m \in \mathcal{M}$, the simulator gives a vector output in $\mathbb{R}^p$. In general it is assumed that the deterministic simulator has been run $n \in \mathbb{N}$ times, either independently or as a causal time-series such that the input of the t^{th} run, $m_t \in \mathcal{M}$, is a function of the previous inputs $m_{t-1}, m_{t-2}, \ldots \in \mathcal{M}$ and of the previous outputs $f(m_{t-1}), f(m_{t-2}), \ldots \in \mathbb{R}^p$. This set of simulator outputs, each in $\mathbb{R}^p$, can either be represented by a matrix in $\mathbb{R}^{n\times p}$, where each row of the matrix corresponds to $f(m)$ for some $m \in \mathcal{M}$, or as a set of vectors $\{x_1, x_2, \ldots, x_n\}$ where each $x_i = f(m_i)$ for some $m_i \in \mathcal{M}$, $i = 1, \ldots, n$.

The overall plan for the thesis is to obtain forecasts of the simulator by modelling some low dimensional approximation by some stochastic process. That is, the information omitted in the low dimensional approximation is replaced by some "noise" term. A more detailed version of the plan is given in Panel 1.1.

Panel 1.1: Thesis plan: overview of the mathematical concept

1. Simulator gives deterministic output $x_i = f(x_{i-1}, t_i)$ for $i = 1, 2, \ldots, n$. Here, x_i is in a high dimensional space $\mathbb{R}^p$, and f is from (1.1) with a constrained input $m_t = (x_{i-1}, t_i)$ for simplicity.

2. Some transformation is initially applied to give a suitable output x_i that the following steps can be applied to. These transformations are discussed in detail in the relevant sections.

3. A dimension reduction method is applied to split the simulator output into the form
$$x_i = y_i + z_i,$$
where $y_i \in V \subseteq \mathbb{R}^p$, and $z_i \in V^\perp$, with $V^\perp$ the orthogonal complement of V. Here z_i is the discarded information from the high dimensional simulator output, and thus is chosen to be "un-interesting" in some way, for example such that $\sum_i z_i^\top z_i$ is small;

4. With z_i discarded, set $\tilde{y}_i = \mathrm{P}_V f(y_{i-1}, t)$, where $\mathrm{P}_V f \colon \mathbb{R}^p \times [0, \infty) \to V$ is the orthogonal projection of f onto V.

5. Approximate the deterministic function $\mathrm{P}_V f$ by some $g \colon V \times [0, \infty) \to V$ and some noise to obtain an approximation to $\tilde{y}_i$, given by the stochastic evolution equation
$$\hat{y}_i = g(\hat{y}_{i-1}, t_i) + \varepsilon_i,$$
where ε_i is some noise.

6. Use $\hat{y}_i$ in replacement of x_i to obtain approximations of the simulator

output. Continuous-time stochastic evolution is also considered in detail in Section 2.3.

Suppose the simulator output is given by the set of vectors $\mathcal{X} := \{x_1, x_2, \ldots, x_n\}$ with each $x_i \in \mathbb{R}^p$, $i = 1, \ldots, n$, with $\operatorname{span}(\mathcal{X}) = r$, $r \leq p$. The dimension reduction methods described in this chapter attempt to find projections of the data that explain some "important" behaviour of the simulator. That is, the dimension reduction methods decompose a set of data $\mathcal{X} = \{x_1, \ldots, x_n\}$ into some basis of the span of $\mathcal{X}$, $\{w_1, w_2, \ldots, w_r\}$, with $w_i \in \mathbb{R}^p$. This basis is ordered such that the projections $y_i = (\langle x_1, w_i\rangle, \langle x_2, w_i\rangle, \ldots, \langle x_n, w_i\rangle) \in \mathbb{R}^n$, $i = 1, \ldots, r$, are of decreasing "interest". Here, $\langle \cdot, \cdot \rangle$ is the standard Euclidean inner product such that $\langle a, b\rangle = a^\top b$ for $a, b \in \mathbb{R}^p$, and the use of the word *projection* is a slight abuse of terminology, used to describe the *scalar projection* of a simulator output onto some vector with unit norm. It is remarked here that not all r basis vectors are kept in practice. In fact, often only a small subset of the basis vectors that describe the most "interesting" $l < r$ projections, $y_1, \ldots, y_l$, are used.

In general, throughout this chapter the simulator outputs are arranged row-by-row in the matrix $X = (x_1\, x_2 \cdots x_n)^\top \in \mathbb{R}^{n\times p}$, and projections of the set of simulator outputs are given by $y = Xw$, for $w \in \mathbb{R}^p$.

The two dimension reduction methods discussed in this chapter are principal component analysis and independent component analysis. There are three major differences between these techniques, which are,

(i) whether *whitening* of the simulator output is required. This is a linear transformation that transforms a matrix $X \in \mathbb{R}^{n\times p}$ into a matrix $Y = XA$ such that the sample covariance matrix $Y^\top Y/(n-1)$ is the identity matrix. The whitening transform is unique only up to orthogonal transformations, and therefore there are infinitely many possible whitening transformations A. It is remarked here for future reference that the matrix A is not orthogonal;

(ii) the *contrast function* used to assess whether a projection is "interesting". A contrast function has the property that maximising this function over the directions $w \in \mathbb{R}^p$ gives projections $y = Xw$ that are deemed to be of interest;

(iii) whether approximations to the contrast function are required in practice to obtain the directions.

Naïvely, the entirety of the space $\mathbb{R}^p$ needs to be examined to find the directions $w_i \in \mathbb{R}^p$, $i = 1, \dots, r$, that maximise the contrast function whilst satisfying any other criteria. An exhaustive search of $\mathbb{R}^p$ is clearly extremely computational expensive for p large, and thus it is first noted here that the search space to obtain the directions $w_1, \dots, w_r$ can be immediately reduced in size to just the associated projective space, which is defined below. The intuitive idea on why projective spaces (rather than Euclidean spaces) are used is that when picking vectors for dimension reduction, only the direction of the vector is interesting, and antipodal directions give the same results (in terms of the respective contrast functions discussed in this chapter). The contrast functions discussed in this chapter are based on the densities of the projected data onto some direction, and are invariant to reflection of the densities. The reduced search space is formalised mathematically in the definition of real projective space.

Definition 1.1.1 (Projective space)**.** *The projective space on $\mathbb{R}^p$, denoted by $\mathbb{R}^{p-1}P$, is the set of all equivalence classes $[u] \in \mathbb{R}^{p-1}P$, where for all $u, w \in \mathbb{R}^p \setminus \{0\}$, u and w are equivalent if there exists some $\lambda \in \mathbb{R}$ such that $u = \lambda w$. That is, the equivalence class is given by $[u] = \{w \colon \lambda w = u \text{ for some } \lambda \in \mathbb{R}\backslash\{0\}\}$.*

Throughout this chapter there is a slight abuse of notation, due to writing $u \in \mathbb{R}^{p-1}\mathrm{P}$ to denote the vector $u \in \mathbb{R}^p$ that belongs in the equivalence class $[u] \in \mathbb{R}^{p-1}\mathrm{P}$, such that $u^\top u = 1$ (i.e. u lies on the $(p-1)$-dimensional sphere S^{p-1}).

This chapter is split into two main parts. In Section 1.2, principal component analysis is introduced by describing the underlying mathematical theory in Section 1.2.1, with the method used in practice to obtain the principal directions given in Section 1.2.2. For principal component analysis, the method can be applied directly using standard computational techniques, and therefore principal component analysis can be implemented in a standard way on a set of simulator outputs. In Section 1.2.3 we give a brief discussion on how the theory and computation behind principal component analysis is affected by the size of the data it is being applied to. Section 1.3 introduces independent component analysis, with the mathematical theory described in Section 1.3.1. Section 1.3.2 we consider the how the size of the data affects the theory and computation behind independent component analysis (in the same way as Section 1.2.3). Following this and in reference to Item (iii) above, we introduce some competing methods to obtain approximate independent component directions in Section 1.4. One method that we discuss at length in Section 1.4.1 is fastICA, which is a popular algorithm but with some important deficiencies. We discuss how problems arise within the fastICA method and how they are intrinsically connected to the approximations that form the basis of the computational efficiency of fastICA. This examination of the fastICA contrast function is new work and forms part of a paper that has been published in the Journal of Multivariate Analysis Smith *et al.* (2020). To overcome some of the drawbacks encountered in fastICA, in Section 1.4.2 we introduce a novel independent component analysis method, called clusterICA. This method uses a clustering method applied in a novel way in the projective space. In Section 1.4.3 we compare clusterICA to some established algorithms – including fastICA – on test data.

Remark 1.1: Notation

Some notation is given here to avoid ambiguity going forward. We will clearly state when any notation differs from that described below.

1. Random variables are given by upper-case sans-serif typeface. For example, $\mathsf{X} \in \mathbb{R}^m$ would be a random vector of size m, with components X_i, $i = 1, \ldots, m$;
2. Lower case letters are realisation from some random variable, that is, $x \in \mathbb{R}^m$ is an observation from some random variable, say $\mathsf{X} \in \mathbb{R}^m$;
3. Matrices are given by upper-case standard LaTeX typeface, i.e. computer modern Roman, such that an $n \times m$ matrix is written $X \in \mathbb{R}^{n\times m}$.

Suppose that data from a simulator is given by the set of vectors $\{x_1, \ldots, x_n\}$, with each $x_i \in \mathbb{R}^p$. In this case, the matrix representation of this set can be split into three scenarios with associated notations. These are,

1. $X = (x_1\, x_2 \cdots x_n)^\top \in \mathbb{R}^{n\times p}$ is the matrix representation of the raw simulator output, given row-wise. In this representation, $\tilde{X}$ is used to denote the centred-column version of X. That is,

   $$\tilde{X} = \big((x_1 - \bar{x})\ (x_2 - \bar{x}) \cdots (x_n - \bar{x})\big)^\top \in \mathbb{R}^{n\times p},$$

   with $\bar{x} = (1/n)\sum_{i=1}^{n} x_i$.

2. $Y = \tilde{X}A \in \mathbb{R}^{n\times r}$, $r = \text{rank}(\tilde{X}) \leq \min\{n-1,\, p\}$, is a *whitened* form of X, such that the covariance of Y is the identity matrix $I_r \in \mathbb{R}^{r\times r}$, where $A \in \mathbb{R}^{p\times r}$ is a whitening matrix. This is discussed in Section 1.3.1.

3. $Z = YQ = \tilde{X}AQ \in \mathbb{R}^{n\times r}$ describes some orthogonal transformation of the whitened matrix Y, with $Q \in R^{r\times r}$ an orthogonal matrix (*i.e.* $Q^\top Q = QQ^\top = I_r$).

1.2 Principal Component Analysis

The first dimension reduction method considered is known as principal component analysis (often referred to as 'PCA') and was originally described by Pearson (1901) with a practical description given later by Hotelling (1933). This section starts with a brief introduction to principal component analysis and the main theoretical points highlighted. Then, we describe the method of applying principal component analysis in practice. All proofs here are omitted, and further reading can be found in many undergraduate multivariate statistical textbooks, including Jackson (2005); Manly (1994); Mardia *et al.* (1979).

1.2.1 Introduction to principal component analysis

As explained above, principal component analysis obtains an orthonormal basis $\{w_1, \ldots, w_r\}$ of the space spanned by the simulator outputs $\{x_1, \ldots, x_n\}$. In this section we give a motivation for, and describe mathematically, the metric used for choosing this basis.

To obtain each basis vector sequentially, the variance of the associated projection of the simulator output is maximised with the constraint that the basis vectors must be orthonormal. One motivation for maximising the variance of the projection is that the associated basis vector here "separates" the data in some maximal way. First, define the (sample) variance of the projection of some simulator output by the mapping $V\colon \mathbb{R}^{p-1}\mathrm{P} \to [0, \infty)$ from the basis vector $w \in \mathbb{R}^{p-1}\mathrm{P}$ by,

$$V(w : \{x_1, \ldots, x_n\}) := \mathrm{Var}\{\langle x_1, w\rangle, \ldots, \langle x_n, w\rangle\}. \tag{1.2}$$

This will be shortened to $V(w)$ if the choice of the dependent simulator output is obvious.

We first give a motivating example for using principal component analysis. Suppose each $x_i \in \mathbb{R}^p$ is a vector of exam scores obtained by student i, $i = 1, \ldots, n$. Then finding a projection of the set of student scores, $\langle x_1, w\rangle, \ldots, \langle x_n, w\rangle$, where $w \in \mathbb{R}^{p-1}\mathrm{P}$ is chosen such that $V(w)$ is maximum, gives a way to rank the students by separating them maximally using a linear combination of their exam scores (Mardia *et al.*, 1979). A motivation to using principal component analysis on climate simulators with the aim of reducing their complexity is that many of the input parameters $m \in \mathcal{M}$ may be very closely correlated. Thus the input space of the mapping $f\colon \mathcal{M} \to \mathbb{R}^p$ could be reduced by taking a set of $l < p$ linear combination of these correlated parameters and a new mapping found $\tilde{f}\colon \tilde{\mathcal{M}} \to \mathbb{R}^p$, with $|\tilde{\mathcal{M}}| < |\mathcal{M}|$, where very little "important" information (with respect to the second moment) has been lost in the new simulator compared to the original.

Panel 1.2 describes the theoretical procedure used in principal component analysis to obtain the orthogonal basis $\{w_1, w_2, \ldots, w_r\}$, $r = \mathrm{rank}(\tilde{X})$, from simulator output $\{x_1, x_2, \ldots, x_n\}$.

Panel 1.2: Principal component analysis procedure

The principal component analysis procedure for obtaining the orthonormal basis from simulator output $\{x_1, \ldots, x_n\}$, is as follows. Each vector $w_i \in \mathbb{R}^{p-1}\mathrm{P}$, $i = 1, \ldots, r = \operatorname{rank}(\tilde{X})$ that forms the basis is defined in the following way, using the mapping V given in (1.2).

First, $w_1 \in \mathbb{R}^p$ satisfies,

$$w_1 = \operatorname*{argmax}_{w \in \mathbb{R}^{p-1}\mathrm{P}} V(w).$$

Then, for each $j = 2, 3, \ldots, r$ the basis vectors are found sequentially, satisfying,

$$w_j = \operatorname*{argmax}_{\substack{w \in \mathbb{R}^{p-1}\mathrm{P} \\ w \perp \{w_1, \ldots, w_{j-1}\}}} V(w).$$

1.2.2 Principal component analysis method

In this section we give a practical computational method for applying principal component analysis to a simulator output. Note that unlike independent component analysis introduced later in Section 1.3, only one method is given here as this is computationally simple and gives the (approximate) orthonormal basis $\{w_1, \ldots, w_r\}$ that we described above in Panel 1.2.

It can be shown that the orthogonal basis found in Panel 1.2 corresponds to the eigenvectors of the sample covariance matrix $C_X \in \mathbb{R}^{p \times p}$ of the matrix $X = (x_1 \cdots x_n)^\top \in \mathbb{R}^{n \times p}$. The sample covariance matrix C_X is given by

$$C_X = \frac{1}{n-1} \tilde{X}^\top \tilde{X}, \tag{1.3}$$

where $\tilde{X}$ is the centred-column version of X (following the notational convention given in Remark 1.1). The eigenvalue γ_j of C_X gives the variance of the corresponding eigenvector w_j. That is, $\gamma_j = V(w_j)$, and these are ordered such that $\gamma_i \geq \gamma_j$ for all $i > j$, and for all $j = r+1, \ldots, n$, $\gamma_j = 0$. The sum of the r eigenvalues is equal to the total variance of the simulator output and therefore it is often remarked that "principal component j explains $\gamma_j / \sum_{i=1}^{r} \gamma_i$ of the variation in the data".

The dimension of the simulator, p, can be large and in this case calculating the covariance matrix C_X directly is computationally expensive. Therefore, in practice, *singular value decomposition* of $\tilde{X}$ is used for finding the eigenvalues and eigenvectors of C_X, without calculating C_X explicitly. We now describe how the eigenvectors and eigenvalues of C_X can be found directly, by applying singular value decomposition to $\tilde{X}$.

In the real matrix setting, singular value decomposition decomposes a matrix $M \in \mathbb{R}^{a\times b}$ into the form $U\Lambda V^\top$, where $U \in \mathbb{R}^{a\times r}$, $\Lambda \in \mathbb{R}^{r\times r}$ and $V \in \mathbb{R}^{b\times r}$ with $r \leq \min\{a, b\}$ the rank of M. Here, U and V have orthogonal columns, and Λ is a diagonal matrix. Assume that the decomposition is such that the diagonal elements of Λ are ordered $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_r$.

Consider the centred simulator output matrix $\tilde{X} \in \mathbb{R}^{n\times p}$ with $r = \text{rank}(\tilde{X}) \leq \min\{n-1, p\}$, and let $\tilde{X} = U\Lambda V^\top$ be a singular value decomposition with $U \in \mathbb{R}^{n\times r}$, $\Lambda \in \mathbb{R}^{r\times r}$ and $V \in R^{p\times r}$. Now, the covariance matrix can be written,

$$C_X = \frac{1}{n-1}(U\Lambda V^\top)^\top(U\Lambda V^\top) = V\Big(\frac{1}{n-1}\Lambda^2\Big)V^\top,$$

as $U^\top U = I_n$ and where $\Lambda^2 = \text{diag}(\lambda_i^2)$. Thus, C_X can be written in eigen-decomposition form with the eigenvectors given by the columns of V and the associated eigenvalues $\lambda_i^2/(n-1)$. Using this knowledge, principal component analysis can be applied in practice to some data as described in Panel 1.3.

Panel 1.3: Principal component analysis computational method

Suppose the simulator output is given by the matrix $X = (x_1 \cdots x_n)^\top \in \mathbb{R}^{n\times p}$. Let $\tilde{X}$ be the matrix X with centred columns. Then, the principal components are found in the following way:

1. Perform singular value decomposition on $\tilde{X}$ to obtain matrices $\Lambda = \text{diag}(\lambda_i) \in \mathbb{R}^{r\times r}$ and $V = (v_1 \cdots v_r) \in \mathbb{R}^{p\times r}$;

2. Set $w_i = v_i$, $m_i = \tilde{X}w_i$, and $\gamma_i = \lambda_i^2/(n-1)$, $i = 1, \ldots, r$;

Then, w_i corresponds to the i^{th} principal direction with associated principal component given by m_i. The variance of the i^{th} principal component is given by $\gamma_i = V(w_i \colon \{x_1, \ldots, x_n\})$.

Note that in practice, often many of the γ_i with $i = l+1, \ldots, r$ for some $l < r$, found in Panel 1.3 are relatively small, as the associated principal components account for very little variation of the original data. In this case, often only l of the principal components are kept, where the number l can be chosen such that projections onto the set of principal directions, $\{w_1, \ldots, w_l\}$, account for some high percentage of the total variance of the simulator output $\{x_1, \ldots, x_n\}$. That is, l is chosen such that

$$\frac{\sum_{i=1}^{l} \gamma_i}{\sum_{i=1}^{r} \gamma_i} \in [1-\varepsilon, 1],$$

for some $\varepsilon << 1$.

To end this section, some terminology is given. The term *loading* is often used to refer to the eigenvectors scaled by the square root of their respective eigenvalue. That is, the i^{th} loading is given by the vector $\sqrt{\gamma_i} \cdot w_i \in \mathbb{R}^p$. We note here that there is often some confusion about this term, possibly[1] stemming from `R` documentation – for example in '`princomp`' (R Core Team, 2019) – that describes the 'loadings' as being the eigenvectors without scaling. In general, the eigenvectors given by the columns of V are known as principal directions (or axes), with the data projected onto them known as the principal components.

1.2.3 Principal component analysis scenarios: simulator and output size

In this section we present a short discussion on how principal component analysis is affected by the size of the simulator output, both in terms of the dimension of the simulator and how many iterations of the simulator are available. Note that in this thesis, only "case 1" is used in practice (see Figure 1.1).

Figure 1.1 gives a overview of the four scenarios that can be encountered, and how this affects principal component analysis. These four scenarios will be discussed in greater detail now.

[1]at least, personally

Case 1: "Standard" PCA
$X = (x_1, \dots, x_n)^\top \in \mathbb{R}^{n\times p}$
Aim: eigenvalues and eigenvectors of
$C_X = \frac{1}{n-1} X^T X$

Case 3: Functional PCA
$x = (x_1, \dots, x_n),\ x_i \in \mathrm{L}^2([0,T])$
Aim: eigenvalues and eigenfunctions of
$C_x(s,t) = \frac{1}{n-1}\sum_{i=1}^{n}\big(x_i(s) - \bar{x}(s)\big)\big(x_i(t) - \bar{x}(t)\big)$

$$\begin{tikzcd}
n < \infty,\ p < \infty \arrow[r] \arrow[d] & n < \infty,\ p = \infty \arrow[d] \\
n = \infty,\ p < \infty \arrow[r] & n = \infty,\ p = \infty
\end{tikzcd}$$

Case 2: "Standard" PCA (with perfect data)
The distribution of random vector $\mathsf{X} \in \mathbb{R}^p$ is known.
Aim: eigenvalues and eigenvectors of covariance, K_{XX}

Case 4: Functional PCA (with perfect data)
We know distribution of random function $\big(\mathsf{X}(t),\ t \in [0,T]\big)$.
Aim: eigenvalues and eigenfunctions of
$C_{\mathsf{X}}(s,t) = \mathsf{E}\Big(\big(\mathsf{X}(s) - \mu(s)\big)\big(\mathsf{X}(t) - \mu(t)\big)\Big)$

Figure 1.1: Four principal component analysis 'cases'

1.2.3.1 Case 1: $n < \infty$, $p < \infty$

This is the case of the output of all simulators in practice, as data must be saved discretely. However, in the case where the dimension of the simulator, p, is much larger than the number of iterations, n, available, it could be beneficial to analyse the output using functional principal component analysis, as is discussed in Section 1.2.3.3.

When the data consists of n sets of p variables, $x_1, x_2, \dots, x_n \in \mathbb{R}^p$, with $X = (x_1, x_2, \dots, x_n)^\top \in \mathbb{R}^{n\times p}$, then the theory of principal component analysis is as given in Section 1.2.1 and Section 1.2.2 above.

1.2.3.2 Case 2: $n = \infty$, $p < \infty$

This case is equivalent to knowing the distribution of the random variable $\mathsf{X} \in \mathbb{R}^p$ exactly. Then also the covariance K_{XX} is known (as opposed to the sample covariance C_X) and thus the corresponding eigenvalues and eigenvectors can be found by eigen-decomposition, $K_{\mathsf{XX}} = VDV^T$. This gives the principal components exactly (by the columns of V), and the respective variance accounted for by each principal direction by the diagonal elements of D.

1.2.3.3 Case 3: $n < \infty$, $p = \infty$

This case is referred to as *functional principal component analysis* in the literature, see for example Hall & Hosseini-Nasab (2006). In scenarios where the dimension of the simulator is much larger than the number of iterations ($i.e$ $p >> n$), then functional principal component analysis could be used to analyse the data, treating each simulator output $\mathbb{R}^p$-vector as some set of observations from a random function. Random functions are introduced in Chapter 2 in terms of stochastic differential equations, and are used to model the principal components obtained from simulation data.

Let $(\mathbb{R}, \mathrm{B}(\mathbb{R}))$ be a measurable space and $(\Omega, \mathcal{F}, \mathbb{P})$ a probability space. Assume that we have obtained n independent realisations from an $\mathbb{R}$-valued stochastic process $(\mathsf{X}(t))_{t\in[0,T]}$, with $\mathsf{E}\mathsf{X}(t) = \mu(t) < \infty$ for $t \in [0, T]$. Explicitly, we have

$$\begin{aligned} x(t) &= \big(x_1(t), x_2(t), \ldots, x_n(t)\big) \\ &= \big(x(t,\omega_1), x(t,\omega_2), \ldots, x(t,\omega_n)\big) \end{aligned} \tag{1.4}$$

for $t \in [0, T]$ and $\omega_i \in \Omega$, $i = 1, 2, \ldots, n$.

In this scenario the covariance function $C_{\mathsf{X}}(s,t) = \mathsf{E}\big(\mathsf{X}(s) - \mu(s)\big)\big(\mathsf{X}(t) - \mu(t)\big)$ needs to be approximated, as does the corresponding covariance operator

$$(\mathcal{C}_{\mathsf{X}} f)(t) := \int_{\mathrm{T}} C_{\mathsf{X}}(s,t) f(s)\,\mathrm{d}s \quad \text{for } f \in \mathrm{L}^2([0,T]).$$

Note that $\mathsf{X}(t)$ can be written in terms of the eigenvalues and eigenfunctions of the covariance operator using the Karhunen-Loève expansion, which gives

$$\mathsf{X}(t,\omega) = \mu(t) + \sum_{j=1}^{\infty} v_j^{1/2}\, \phi_j(t)\, \xi_j(\omega). \tag{1.5}$$

Here, $\xi_j(\omega) = \upsilon_j^{-1/2}\langle \mathsf{X}(t,\omega) - \mu(t), \phi_j(t)\rangle_{\mathrm{L}^2([0,T])}$ and $\{\upsilon_j, \phi_j\}$ are the eigenvalues and eigenfunctions of the covariance operator, ordered such that $\upsilon_1 \geq \upsilon_2 \geq \ldots \geq 0$ (Lord *et al.*, 2014). The eigenfunctions satisfy $\langle \phi_j, \mathcal{C}\phi_k\rangle = \delta_{\{j=k\}}\, \upsilon_k$, and thus the covariance function can be written,

$$C_{\mathsf{X}}(s,t) = \sum_{j=1}^{\infty} \upsilon_j \phi_j(s)\phi_j(t).$$

The use of functional principal component analysis is motivated by (1.5), as here some approximation to X can be found by truncating the number of eigenfunctions used, such that a new random function is obtained via

$$\mathsf{X}_l(t,\omega) = \mu(t) + \sum_{j=1}^{l} \upsilon_j^{1/2}\, \phi_j(t)\, \xi_j(\omega).$$

which has associated covariance function

$$C_{(\mathsf{X},l)}(s,t) = \sum_{j=1}^{l} \upsilon_j \phi_j(s)\phi_j(t).$$

As the υ_j, $j = 1, 2, \ldots$ are in descending order, we can choose the value of l such that the truncated version of X retains the vast majority of the variance that is present in the full process. See Lord *et al.* (2014) for the proof that $C_{(\mathsf{X},l)}(s,t) \to C_{\mathsf{X}}(s,t)$ and $\mathsf{E}\big(\mathsf{X}(t) - \mathsf{X}_J(t)\big) \to 0$ uniformly for $s, t \in [0,T]$.

For the sample x, as in (1.4), the approximation to $C_{\mathsf{X}}(s,t)$ can be found in the standard empirical way, which gives

$$C_x(s,t) = \frac{1}{n-1}\sum_{i=1}^{n} \big(x_i(s) - \bar{x}(s)\big)\big(x_i(t) - \bar{x}(t)\big),$$

where $\bar{x} = 1/n \sum_{i=1}^{n} x_i$. As before, this may be written

$$\begin{aligned} C_x(s,t) &= \sum_{j=1}^{\infty} \hat{v}_j \hat{\phi}_j(s) \hat{\phi}_j(t) \\ &= \sum_{j=1}^{n-1} \hat{v}_j \hat{\phi}_j(s) \hat{\phi}_j(t). \end{aligned} \tag{1.6}$$

Here, $\{\hat{v}_j, \hat{\phi}_j\}$ are the eigenvalues and eigenfunctions of the operator $\mathcal{C}_x$, ordered such that $\hat{v}_1 \geq \hat{v}_2 \geq \ldots$, where $\mathcal{C}_x$ is given by

$$(\mathcal{C}_x f)(t) = \int_0^T C_x(s,t) f(s) \, \mathrm{d}s \quad \text{for } f \in \mathrm{L}^2([0,T]).$$

The second equality (1.6) holds as C_x maps onto the space spanned by the n functions $(x_i - \overline{x})$, $i = 1, 2, \ldots, n$, of which $n-1$ are independent. Therefore $\hat{v}_n = \hat{v}_{n+1} = \ldots = 0$.

Applying functional principal component analysis to the n observed functions gives $n-1$ loadings $\big(\hat{\phi}_1(t), \hat{\phi}_2(t), \ldots, \hat{\phi}_{n-1}(t)\big)$, each of which accounts for $\big(\sum_{j=1}^{n-1} \hat{v}_j\big)^{-1} \cdot \hat{v}_k$, $k = 1, 2, \ldots, n-1$ of the total variation in the original data.

1.2.3.4 Case 4: $n = \infty$, $p = \infty$

In this case the full expansion of the stochastic process $(\mathsf{X}(t))_{t \in [0,T]}$ in terms of the eigenvalues and eigenfunctions of the covariance operator as in (1.5) is known, so

$$\mathsf{X}(t, \omega) = \mu(t) + \sum_{j=1}^{\infty} v_j^{1/2} \, \phi_j(t) \, \xi_j(\omega),$$

and thus the exact principal components are available.

This case could be applicable to simulator output data which is high dimensional with many iterations Here, the limit p, $n \to \infty$ with $n/p \to \lambda \in (0, \infty)$ could be considered. This is similar to the situation in Hall & Hosseini-Nasab (2006).

1.3 Independent Component Analysis

This section introduces independent component analysis (also referred to as 'ICA'), a well-established and popular dimension reduction technique that builds on the principal component analysis method introduced in Section 1.2. The main motivation behind independent component analysis differs from principal component analysis in one important aspect. The use of independent component analysis supposes that the simulator output is some *mixture of independent processes*, and we want to obtain the basis that "de-mixes" the observations into the original independent processes. For example, the simulator output $f(m)$ for some $m \in \mathcal{M}$ that attempts to model some climate or weather behaviour could be taken to be some mixture of more fundamental Earth system processes, and independent component analysis is used to discover these processes. The first mention of independent component analysis came from within signal processing research – as signals are an obvious candidate for the above framework, for example, microphones picking up different speech signals – and was originally called INCA, for $\underline{\text{in}}$dependent $\underline{\text{c}}$omponent $\underline{\text{a}}$nalysis (Jutten & Herault, 1991).

Further motivation for using independent component analysis is given in Richman (1986), where the aim is to obtain some interpretable low dimensional set of data from some climate simulator output. In this case, although principal component analysis is advantageous when pure data reduction is sought or for extracting some components with maximum variance, the lack of interpretability of these components and the mixing of different physical phenomena into single components are negative side-effects. Richman (1986) showed by way of climate simulator examples where rotation of the principal components resulted in increased interpretability of the resulting components. It turns out that independent component analysis is exactly a rotation of the components found by principal component analysis.

As with principal component analysis, we give a short theoretical introduction here, with the reader referred to Comon (1994) for a more detailed mathematical introduction and Hyvärinen (1999); Hyvärinen *et al.* (2004); Stone (2004) for a comprehensive overview of the principles underlying independent component analysis and its applications in a wide variety of practical examples. For examples of independent component analysis applied to climate systems see Aires *et al.* (2000); Ilin *et al.* (2006); Lotsch *et al.* (2003)

This section is arranged as follows. In Section 1.3.1 we introduce the method by the use of *mutual information* as a contrast function. In Section 1.3.2 we consider how the size of the simulator affects the theory and computation behind independent component analysis (which mirrors Section 1.2.3 in the principal component analysis case). The main contribution of this chapter comes from Section 1.4, where we describe independent component analysis methods that use approximations to a contrast function related to mutual information. Initial we introduce a popular independent component analysis method known as *fastICA*, and discuss some of its drawbacks (Section 1.4.1). Then in Section 1.4.2 we describe a novel method called *clusterICA*, that uses clustering on the projective space to help find independent components. Finally, in Section 1.4.3 the methods introduced in Section 1.4 are compared to one-another and to other established independent component analysis methods on a set of test data.

1.3.1 Introduction to independent component analysis

In this section, we introduce the use of mutual information as a contrast function and the motivation for the choice of the basis $\{w_1, \ldots, w_r\}$. Then, we describe transformations of the simulator output that simplify the mutual information and thus allow it to be (approximately) computed by using approximations to entropy. It is shown here how independent component analysis builds on principal component analysis through the use of the *whitening* transform. The basis $\{w_1, \ldots, w_r\}$ is chosen to ensure linear independence of the associated projections, which differs to the aim of uncorrelated projections used in principal component analysis, such that in this case the basis vectors are not required to be orthogonal. See Remark 1.2 for a recap on the differences between independent, uncorrelated and orthogonal vectors.

Remark 1.2: Independent or uncorrelated

The definition of linearly independent and uncorrelated vectors are often confused with one-another (and with that for orthogonal), and so they are repeated here for clarity. See Rodgers *et al.* (1984) for more information.

Suppose $x \in \mathbb{R}^p$ is an observation from the random vector X, and $y \in \mathbb{R}^p$ is an observation from the random vector Y. Then,

1. Linearly independent: x and y are linearly independent if and only if $ax - y \neq 0$ for all $a \in \mathbb{R}$;

2. Orthogonal: x and y are orthogonal if and only if $x^\top y = 0$;

3. Uncorrelated: x and y are uncorrelated if and only if $(x - \bar{x}\mathbb{1})^\top (y - \bar{y}\mathbb{1}) = 0$, where $\bar{x}$ (respectively, $\bar{y}$) is the arithmetic mean of x (y), and every element of the vector $\mathbb{1} \in \mathbb{R}^p$ is 1.

Note that all uncorrelated and orthogonal vectors are linearly independent, but the converse is not true. See Rodgers *et al.* (1984, Figure 1.) for a simple example showing the relationships between the three definitions.

As before, let the simulator output be given by $\{x_1, \ldots, x_n\}$. Then, the basic assumption underpinning independent component analysis is that each x_i is a linear combination of some unknown sources $s_1, \ldots, s_n \in \mathbb{R}^p$ that are statistically independent. That is, with $x_i = (x_{i1}, \ldots, x_{ip})$, $s_i = (s_{i1}, \ldots, s_{ip})$, we assume

$$x_{ij} = \sum_{k=1}^{p} s_{ik} b_{kj},$$

for $i = 1, \ldots, n$, and $j = 1, \ldots, p$, where $B = (b_{ij}) \in \mathbb{R}^{p \times p}$ is known as the *mixing matrix*. Here, it is assumed that the number of sources is equal to the dimension of the simulator output, such that $B = (b_1\, b_2 \cdots b_p) \in \mathbb{R}^{p \times p}$ and $S = (s_1\, s_2 \cdots s_n)^\top \in \mathbb{R}^{n \times p}$. Then, $x_i = \sum_{k=1}^{p} s_{ik} b_k$ and $X = SB^\top$. For the more general case where $S \in \mathbb{R}^{n \times N}$, $N \neq p$, see Comon (1994); Ilin *et al.* (2006). In the climate case, the vectors $s_i \in \mathbb{R}^p$ can be thought of as some set of fundamental climate system states varying in time, with the elements of the vectors being statistically independent from one-another.

With this set-up, the motivation behind independent component analysis is to be able to obtain some estimate of the matrix S from the mixed observations X. That is, independent component analysis attempts to find the de-mixing matrix W such that $XW = S$. Here it is remarked that by extracting statistically independent components, independent component analysis makes use of higher-order statistics compared to principal component analysis, which only uses second-order statistics for obtaining components (Aires *et al.*, 2000). From this motivation, we introduce a contrast function that results in the statistically independent sources being found in theory, which is achieved by finding a de-mixing matrix W. In Section 1.3.1.1 we discuss the theory underpinning the contrast function of independent component analysis in terms of random variables, and then in terms of realisations from these random variables (for example, output from a climate simulator) in Section 1.3.1.2.

1.3.1.1 Contrast function in terms of random variables

Following Comon (1994), let $\mathsf{X} \in \mathbb{R}^p$ be a random variable with probability density function $h\colon \mathbb{R}^p \to [0, \infty)$. Then, X has mutually independent components if and only if the probability density function can be written as $h(x) = \prod_{i=1}^p h_i(x_i)$, for any $x = (x_1, \dots, x_p) \in \mathbb{R}^p$, where $h_i\colon \mathbb{R} \to [0, \infty)$, $i = 1, \dots, p$, are the probability density functions of each element of X. This motivates using Kullback–Leibler divergence (Kullback & Leibler, 1951), and more specifically, the *mutual information*, to measure the "distance" between $h(x)$ and $\prod_{i=1}^p h_i(x_i)$. Mutual information is zero if and only if the components of X are statistically independent and positive otherwise, and is given by

$$D\left[h, \prod_{i=1}^{p} h_i\right] = \int_{\mathbb{R}^p} h(x) \log\left(\frac{h(x)}{\prod_{i=1}^p h_i(x)}\right) \mathrm{d}x.$$

Square brackets are used here to indicate that D is a functional, taking the functions h and $\prod_{i=1}^p h_i$, as its arguments.

The next step involves writing $D\big[h, \prod_{i=1}^{p} h_i\big]$ in terms of a different functional and the covariance matrix of X. Then, we will explain how to simplify the mutual information in terms of this new functional by transforming the random variable X.

First, we define the *differential entropy* of X as,

$$H[h] := -\int_{\mathbb{R}^p} h(x) \log h(x) \, \mathrm{d}x, \tag{1.7}$$

whenever this integral exists (Cover & Thomas, 2012). This is referred to as 'entropy' going forward, even though it is well known that differential entropy is not the limit of the discrete random variable version of entropy (known as Shannon's entropy). In the special case of a Gaussian random variable with covariance matrix $\Sigma \in \mathbb{R}^{p\times p}$ the entropy can be calculated explicitly and it takes the value $\eta(\Sigma)$ given by

$$\eta(\Sigma) := \frac{1}{2}\big(p + p\log(2\pi) + \log(\det \Sigma)\big). \tag{1.8}$$

It is known that this is an upper bound for entropy, namely the entropy of any random variable with first two moments finite and (invertible) covariance matrix K will belong to the interval $(-\infty, \eta(K)]$. In the one-dimensional case for a Gaussian random variable with variance σ^2, (1.8) becomes

$$\eta(\sigma^2) = \frac{1}{2}\big(1 + \log(2\pi\sigma^2)\big). \tag{1.9}$$

The *negentropy* J is defined as

$$J[h] := \eta(\Sigma) - H[h], \tag{1.10}$$

where $\eta(\Sigma)$ is given by (1.8). Therefore, $J[h] \in [0, \infty)$, and negentropy is zero when the distribution with density h is Gaussian, and strictly greater than zero otherwise.

The Kullback-Leibler divergence can be written is terms of the negentropy and the covariance matrix of X, given by K_{XX}, as

$$D\Big[h, \prod_{i=1}^{p} h_i\Big] = J[h] - \sum_{i=1}^{p} J[h_i] + \frac{1}{2}\log\frac{\prod_{i=1}^{p}(K_{\mathsf{XX}})_{ii}}{\det K_{\mathsf{XX}}}. \tag{1.11}$$

It is shown in Comon (1994), that the last term of (1.11) is zero for a *standardised* random variable where the covariance matrix is the diagonal matrix I_p. That is, for the standardised version of X, given by Y with probability density function g, (1.11) becomes,

$$D\Big[g, \prod_{i=1}^{p} g_i\Big] = J[g] - \sum_{i=1}^{p} J[g_i]. \tag{1.12}$$

Now, let some orthogonal transformation of the standardised random variable Y be given by Z and have probability density function f. Then, the equality $J[g] = J[f]$ is satisfied (Comon, 1994), such that we have $D\big[f, \prod_{i=1}^{p} f_i\big] = D\big[g, \prod_{i=1}^{p} f_i\big]$. Therefore, for a standardised random variable in $\mathbb{R}^p$, minimising the Kullback-Leibler divergence (1.12) over the set of orthogonal transformations is equivalent to maximising $\sum_{i=1}^{p} J[f_i]$.

This motivates both the use of standardising the data and of negentropy as a contrast function to use in independent component analysis, which is now presented in terms of realisations from a random variable in Section 1.3.1.2.

1.3.1.2 Contrast function in terms of realisations from random variables

Let $X = (x_1\, x_2 \cdots x_n)^\top \in \mathbb{R}^{n\times p}$ be a set of n observations from the random variable $\mathsf{X} \in \mathbb{R}^p$. Then, the standardisation procedure is known as *whitening* and is a form of principal component analysis described in Section 1.2. Let C_X be the sample covariance matrix of $\tilde{X}$, such that

$$C_X = \frac{1}{n-1}\tilde{X}^\top \tilde{X},$$

where $\tilde{X}$ is X with centred columns with rank $r \leq \min\{n-1, p\}$, as described in Section 1.2.2. Then, the singular value decomposition of $\tilde{X}$ is denoted $U\Lambda V^\top$ where

$U \in \mathbb{R}^{n\times r}$, $V \in \mathbb{R}^{p\times r}$ are column-orthogonal, and $\Lambda \in \mathbb{R}^{r\times r}$ is diagonal with elements λ_i, $i = 1, \dots, p$. Following notational convention as described in Remark 1.1, set $Y = \tilde{X}A \in \mathbb{R}^{n\times r}$ with $A = \sqrt{n-1}V\Lambda^{-1}$, where $\Lambda^{-1} = \operatorname{diag}(1/\lambda_i) \in \mathbb{R}^{r\times r}$. Then, the sample covariance matrix of Y is

$$\begin{aligned} C_Y &= \frac{1}{n-1}A^\top \tilde{X}^\top \tilde{X} A \\ &= \frac{1}{n-1}(\sqrt{n-1}V\Lambda^{-1})^\top V\Lambda V^\top (\sqrt{n-1}V\Lambda^{-1}) \\ &= I_r \end{aligned}$$

and Y is known as a whitened version of X. Here, V is exactly the set of principal directions obtained by principal component analysis, and thus whitening can be considered as just projecting the data onto the principal directions, each scaled by $\sqrt{n-1}/\lambda_i = 1/\sqrt{\gamma_i}$, where γ_i gives the variance of the i^{th} principal component.

Now, in terms of random variables, the aim of maximising $\sum_{i=1}^{p} J[f_i]$ as discussed in Section 1.3.1.1 becomes: *find the orthogonal matrix $Q \in \mathbb{R}^{r\times r}$ such that the transformation $Z = YQ \in \mathbb{R}^{n\times r}$ of the whitened data Y maximises*

$$\sum_{i=1}^{r} \hat{J}(z_i).$$

Here, $Z = (z_1\, z_2 \cdots z_r)$ is such that z_i is the i^{th} *column* of Z, and $\hat{J}$ is some approximation to the one-dimensional negentropy $J[f_i] = \int_{\mathbb{R}} f_i(x)\log f_i(x)\,\mathrm{d}x$.

Once the orthogonal matrix $Q \in \mathbb{R}^{r\times r}$ has been determined, then the de-mixing matrix W such that $XW = S$, as discussed at the beginning of this section, is given by,

$$W = \sqrt{n-1}V\Lambda^{-1}Q.$$

Note that W is not an orthogonal matrix in general, as the whitening matrix $A = \sqrt{n-1}V\Lambda^{-1}$ is not orthogonal. In practice, often we determine the orthogonal matrix $Q = (q_1\, q_2 \cdots q_p)$ column-wise, where initially, $q_1 \in \mathbb{R}^{r-1}\mathrm{P}$ is found such that $\hat{J}(z_1)$ is maximised. Then for each subsequent iteration $j = 2, 3, \dots, p$, the vector $q_j \in \mathbb{R}^{r-1}\mathrm{P}$ is found that maximises $\hat{J}(z_j)$ with the constraint $q_j^\top q_i = 0$ for

$i = 1, \ldots, j-1$. The independent component analysis method for data $X \in \mathbb{R}^{n\times p}$ is summarised in Panel 1.4.

Panel 1.4: Independent component analysis computational method

Suppose the simulator output is given by the matrix $X = (x_1 \cdots x_n)^\top \in \mathbb{R}^{n\times p}$. Let $\tilde{X}$ be the matrix X with centred columns. Then, the independent components are found in the following way:

1. Perform singular value decomposition on $\tilde{X}$ to obtain matrices $\Lambda = \operatorname{diag}(\lambda_i) \in \mathbb{R}^{r\times r}$ and $V = (v_1 \cdots v_r) \in \mathbb{R}^{p\times r}$, where $r = \operatorname{rank}(\tilde{X})$. Note that this is the same step as in principal component analysis as described in Panel 1.3;

2. Set $A = \sqrt{n-1} V\Lambda^{-1}$ and perform whitening on X to give a whitened matrix $Y = \tilde{X}A \in \mathbb{R}^{n\times r}$;

3. Find the orthogonal matrix $Q \in \mathbb{R}^{r\times r}$ such that the orthogonal transform of Y, given by $Z = YQ \in \mathbb{R}^{n\times r}$ maximises $\sum_{i=1}^{r} \hat{J}(z_i)$, where z_i, $i = 1, \ldots r$, are the columns of Z and $\hat{J}$ is some approximation to (one-dimensional) negentropy (1.10). The matrix $Q = (q_1\, q_2 \cdots q_r)$ is often found column-wise in the following way:

   (a) Find $q_1 \in \mathbb{R}^{r-1}\mathrm{P}$ that satisfies

   $$q_1 = \operatorname*{argmax}_{q\in\mathbb{R}^{r-1}\mathrm{P}} \hat{J}(Yq);$$

   (b) For $j = 2, \ldots, r$, find $q_j \in \mathbb{R}^{r-1}\mathrm{P}$ that satisfies

   $$q_j = \operatorname*{argmax}_{\substack{q\in\mathbb{R}^{r-1}\mathrm{P} \\ q\perp\{q_1,\ldots,q_{j-1}\}}} \hat{J}(Yq);$$

4. Set $W = AQ = \sqrt{n-1} V\Lambda^{-1}Q \in \mathbb{R}^{p\times r}$.

Then, the columns of W form a basis, $\{w_1, w_2, \ldots, w_r\}$, such that w_i, $i = 1, \ldots, r$, is the i^{th} independent direction.

One method for finding an approximation of entropy and negentropy of a one-dimensional random variable, called M-spacing (neg)entropy, is discussed below in Section 1.3.1.3.

Similar to principal component analysis, when performing sequential independent component analysis in practice often only $l < r$ vectors from the basis $\{w_1, \dots, w_r\}$ are found. When following the sequential method such that the orthogonal matrix $Q \in \mathbb{R}^{r \times r}$ is found column-wise, the first basis vector w_1 describes the direction onto which the projection of Y is most non-Gaussian (*i.e.* obtains maximum negentropy), with subsequent basis vectors giving projections with decreasing negentropy. Often, finding the columns of Q is computationally intensive and therefore l is chosen to be small so only the few most "interesting" projections of Y are found.

Remark 1.3: Choosing l in independent component analysis methods

We make a short remark here to help address the confusion that can arise around selecting the value of l and of r in independent component analysis.

Note that in the above description of independent component analysis, only l is chosen by the user to give independent component directions $\{w_1, \dots, w_l\}$, and the value of r is given by the rank of the centred matrix $\tilde{X}$. This is because, in theory, the whitening step (*i.e.* the principal component analysis step) should only be used for transforming X to have covariance matrix equal the identity matrix. Ideally, it should not be used for dimension reduction prior to finding the orthogonal matrix Q. When the whitening matrix also reduces the dimension of the data below the rank of $\tilde{X}$, i.e. $Y = \tilde{X}A \in \mathbb{R}^{n \times \tilde{r}}$ in Step 2. of Panel 1.4 with $\tilde{r} < r$, then this is equivalent to choosing the first $\tilde{r}$ principal component directions, which inherently favour Gaussian projections. This goes against the motivation of independent component analysis, which aims for non-Gaussian projections. However, in practice often one must choose an $\tilde{r} < r$, as optimising negentropy over the projective space $\mathbb{R}^{r-1}\mathrm{P}$ is a computationally intensive task, which is made simpler by restricting the space of directions to $\mathbb{R}^{\tilde{r}-1}\mathrm{P}$. In addition, a choice of $\tilde{r} < r$ in the whitening step reduces the likelihood of numerical issues that can occur in Step 3. (Panel 1.4).

In some computational implementations of independent component analysis methods, the choice of l is confused with that of the number r. For example, in `‘fastICA’` (Marchini *et al.*, 2013), the user specifies l, but this number is also chosen as the number of principal component directions used in Step 2. (Panel 1.4). Whilst often a choice of $\tilde{r} < r$ is required to be made, defaulting this choice to $\tilde{r} = l$ will in general result in a suboptimal outcome. For the `‘fastICA’` function, the matrix found in Step 2. is $Y \in \mathbb{R}^{n\times l}$, and then the orthogonal matrix $Q \in \mathbb{R}^{l\times l}$ is found is found in Step 3. such that $Z = YQ \in \mathbb{R}^{n\times l}$ maximises $\sum_{i=1}^{l} \hat{J}(z_i)$. The use of only l principal component directions in the whitening step fundamentally changes the basis vectors $\{w_1, \dots, w_l\}$ obtained.

1.3.1.3 Entropy approximation: M-spacing

In this section we describe an approximation of (one-dimensional) differential entropy and negentropy, denoted by $\hat{J}$ in Section 1.3.1 above, using the M-spacing method. If the random variable $\mathsf{X} \in \mathbb{R}$ has probability density function f and variance $\sigma^2 \in (0, \infty)$, the differential entropy $H[f]$ as defined in (1.7) can take any value on the interval $(-\infty, \eta(\sigma)]$, where $\eta(\sigma) = \frac{1}{2}\big(1 + \log(2\pi\sigma^2)\big)$ is the entropy of a one-dimensional Gaussian random variable with variance σ^2, (give in (1.9)). As the definition of entropy involves the integral of a density, efficient estimation of entropy from data is not trivial. For a survey of different methods to estimate entropy from data, see Beirlant *et al.* (1997). Here, we only consider the M-spacing estimator, originally given in Vasicek (1976).

From Beirlant *et al.* (1997), the entropy approximation is based on sample spacings. Suppose $d_1, d_2, \dots, d_n \in \mathbb{R}$ is a sample of one-dimensional points from a distribution with density f, and $d_{(1)}, d_{(2)}, \dots, d_{(n)}$ is the ordering such that $d_{(1)} \leq d_{(2)} \leq \cdots \leq d_{(n)}$. Define the M-spacing difference to be $\Delta_M d_i = d_{(i+M)} - d_{(i)}$ for

$M \in \{3, \ldots, n-1\}$ and $i \in \{1, 2, \ldots, n-M\}$. Then the M-spacing approximation for entropy $H[f]$ for a sample $d = (d_1, d_2, \ldots, d_n)$ is denoted by $H_{M,n}(d)$ and is given by

$$H_{M,n}(d) = \frac{1}{n}\sum_{i=1}^{n-M} \log\left(\frac{n}{M}\Delta_M d_i\right) - \psi(M) + \log(M), \tag{1.13}$$

where $\psi(x) = -\frac{\mathrm{d}}{\mathrm{d}x}\Gamma(x)$ is the Digamma function. This is a realisation of the general M-spacing formula given in Hall (1984). Algorithm 1.1 gives the pseudo-code for calculating M-spacing entropy in practice.

Algorithm 1.1: M-spacing entropy

```
input: data vector, $d$; M-spacing parameter, $M$
output: entropy approximation, $h$
begin
    $d = (d_1, d_2, \ldots, d_n)^\top \in \mathbb{R}^n$
    % first need to order d by a permutation such that $d_{(1)} \leq d_{(2)} \leq \cdots \leq d_{(n)}$
    $d_\pi \leftarrow (d_{(1)}, d_{(2)}, \ldots, d_{(n)})$,
    % calculate M-spacing difference of d
    for $i$ in $1:(n-M)$
        $\Delta d_i \leftarrow d_{(M+i)} - d_{(i)}$
    end for
    % calculate M-spacing entropy of d
    $h \leftarrow (1/n) * \sum_{j=1}^{n-M}(\log(n * \Delta d_j / M)) - \mathrm{digamma}(M) + \log(M)$
    return $h$
end
```

It is worth noting that the M-spacing approximation of differential entropy does not require a density estimation step. Also, in general the objective function is not very smooth, although increasing spacing size M does result in a smoother function. A method to attempt to overcome the non-smoothness (and the resulting local extrema, which can cause numerical optimisation issues) is given in Learned-Miller & Fisher III (2003) and involves replicating the data with some added Gaussian noise. The M-spacing method also requires sorting of the data, which has computational cost of $\mathcal{O}(n \log n)$.

Derivative of M-spacing entropy approximation. Let $y_1, y_2, \dots, y_n \in \mathbb{R}^p$ such that $Y = (y_1, y_2, \dots, y_n)^\top \in \mathbb{R}^{n\times p}$ is a whitened data matrix. Finding a (one-dimensional) projection of the data in independent component analysis is equivalent to finding $z = Yq$ for some direction $q \in \mathbb{R}^{r-1}\mathrm{P}$. Given any q, entropy can be calculated for this projection using the M-spacing algorithm described above, $H_{M,n}(Yq) = H_{M,n}(z)$. As the aim of independent component analysis here is to obtain a projection $z = Yq$ that maximises negentropy over all $q \in \mathbb{R}^{p-1}\mathrm{P}$, in practice the M-spacing approximation to entropy can be minimised using some optimisation procedure. That is, we need to find the optimal direction q^* that minimises entropy, $q^* = \operatorname{argmin}_q H_{M,n}(Yq)$ for some fixed $M > 2$.

It is well known that minimisation is significantly more efficient if the derivative of the function to be optimised is known, and therefore we give the derivation of M-spacing here. Furthermore, the derivative for a special case of the direction $q = R\tilde{v} \in \mathbb{R}^{r-1}\mathrm{P}$ is given, where $R \in \mathbb{R}^{r\times r}$ is some symmetric orthogonal matrix and $\tilde{v} = (0, \dots, 0, v)^\top \in \mathbb{R}^{r-1}\mathrm{P}$ with $v \in \mathbb{R}^{k-1}\mathrm{P}$ for some $k < r$. This special case is required for the clusterICA method introduced in Section 1.4.2, and is used to ensure that the independent component analysis loadings found sequentially (as in Step 3. of Panel 1.4) are orthogonal. This is described in more detail later.

Lemma 1.3.1. *(i) Let $Y = (y_1, y_2, \dots, y_n)^\top \in \mathbb{R}^{n\times r}$ and $q \in \mathbb{R}^{r-1}P$. Let $\{\pi(1), \pi(2), \dots, \pi(n)\}$ be a permutation of $\{1, 2, \dots, n\}$ such that $(Yq)_{\pi(1)} \le (Yq)_{\pi(2)} \le \dots \le (Yq)_{\pi(n)}$. Then, with $H_{M,n}$ as in (1.13),*

$$\frac{\partial}{\partial q_j} H_{M,n}(Yq) = \frac{1}{n} \sum_{i=1}^{n-M} \frac{Y_{\pi(i+M),j} - Y_{\pi(i),j}}{(Yq)_{\pi(i+M)} - (Yq)_{\pi(i)}},$$

where $Y_{i,j} \in \mathbb{R}$ is the $(i, j)^{th}$ element of the matrix Y. [1]

(ii) In the special case where $q = R\tilde{v}$, with $R = (r_{ij}) \in \mathbb{R}^{r\times r}$ a given symmetric

[1] The π notation for the order permutation is introduced here to emphasise that the ordering is of the values $(Yq)_{\pi(i)}$, and this ordering is constant throughout the lemma. That is, whereas $Y_{(i)j}$ would be the i^{th} smallest value from the j^{th} column of Y, we have $Y_{\pi(i),j}$, which is the row number of the j^{th} column that corresponds to the i^{th} smallest value of the vector Yq.

orthogonal matrix and $\tilde{v} = (0, 0, \ldots, 0, v)^\top \in \mathbb{R}^r$ *with* $v \in \mathbb{R}^k$, $k < r$, *then*

$$\nabla_v H_{M,n}(YR\tilde{v}) = \nabla_q H_{M,n}(Yq)_{|q=R\tilde{v}} \frac{dR\tilde{v}}{dv},$$

and,

$$\frac{dR\tilde{v}}{dv} = \frac{d}{dv} R \begin{pmatrix} 0 \\ \vdots \\ 0 \\ v \end{pmatrix} = \begin{pmatrix} r_{1,r-k} & r_{1,r-k+1} & \cdots & r_{1,r} \\ r_{2,r-k} & r_{2,r-k+1} & \cdots & r_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ r_{r,r-k} & r_{r,r-k+1} & \cdots & r_{r,r} \end{pmatrix}.$$

Proof. Part (i). We have $(Yq)_{\pi(i)} = \sum_{l=1}^{r} Y_{\pi(i),l} q_l$, and therefore $\frac{\partial}{\partial q_j}(Yq)_{\pi(i)} = Y_{\pi(i),j}$ for $j = 1, 2, \ldots, p$. Now,

$$\begin{aligned} \frac{\partial}{\partial q_j} H_{M,n}(Yq) &= \frac{1}{n} \sum_{i=1}^{n-M} \frac{1}{\frac{n}{M}\big((Yq)_{\pi(i+M)} - (Yq)_{\pi(i)}\big)} \cdot \frac{n}{M}\big(Y_{\pi(i+M),j} - Y_{\pi(i),j}\big) \\ &= \frac{1}{n} \sum_{i=1}^{n-M} \frac{Y_{\pi(i+M),j} - Y_{\pi(i),j}}{(Yq)_{\pi(i+M)} - (Yq)_{\pi(i)}}, \end{aligned}$$

as required.

Part (ii). This can be easily calculated using the chain-rule and part *(i)*. □

1.3.1.4 Invariance to scaling of the independent component analysis method

The result obtained from independent component analysis is independent to affine transformations of the data relating to each dimension. That is, for simulator output $X \in \mathbb{R}^{n\times p}$ and any diagonal matrix $D \in \mathbb{R}^{p\times p}$, then independent component analysis applied to X is equivalent to that applied to XD. To prove this, recall that independent component analysis consists of two main steps. The first is whitening, where a matrix $A \in \mathbb{R}^{p\times r}$ is found such that $Y = XA$ satisfies $Y^\top Y/(n-1) = I_r$. The second step of independent component analysis is in finding an orthogonal matrix $Q \in R^{r\times r}$ such that $Z = XAQ$ maximises the contrast function.

Therefore, it is sufficient to show that whitening of X is equal, up to an orthogonal transformation, to whitening of XD. This is shown in Lemma 1.3.3. Note that invariance to scaling does not transfer to principal component analysis where pre-processing in this manner affects the result and is used often by, for example, scaling the data such that the columns of XD have unit variance.

Lemma 1.3.2. *Let $X \in \mathbb{R}^{n\times p}$, $A \in \mathbb{R}^{p\times q}$ and XA, such that* $\operatorname{rank}(X) = \operatorname{rank}(XA)$. *Then, there exists a matrix $B \in \mathbb{R}^{q\times p}$ such that*

$$XAB = X.$$

Proof. As $r := \operatorname{rank}(X) = \operatorname{rank}(XA)$, the column-space, $\operatorname{colsp}(XA) = \operatorname{colsp}(X)$. Let $W \in \mathbb{R}^{n\times r}$ be a matrix such that the columns form a basis of $\operatorname{colsp}(X)$. Then, there exists some matrices $P \in \mathbb{R}^{r\times p}$, $Q \in \mathbb{R}^{r\times q}$ such that $X = WP$ and $XA = WQ$.

Now, $\operatorname{rank}(WQ) = r = \operatorname{rank}(W)$ and W has full column-rank (as the columns form a basis). Therefore, Q has full row-rank (linearly independent rows) and therefore there exists a Moore–Penrose inverse which can be written as $Q^+ = Q^\top(QQ^\top)^{-1}$, and satisfies $QQ^+ = I_r$.

From this,

$$XAQ^+P = WQQ^+P = WP = X,$$

and setting $B = Q^+P$ concludes the proof.[1] □

Lemma 1.3.3. *Let $X \in \mathbb{R}^{n\times p}$ with* $\operatorname{rank}(X) = r$ *and, for $i \in \{1, 2\}$ let $A_i \in \mathbb{R}^{p\times r}$ such that $Y_i := XA_i$ satisfies $Y_i^\top Y_i/(n-1) = I_r$. Then there exists an orthogonal matrix $Q \in \mathbb{R}^{r\times r}$ such that $Y_1 = Y_2Q$.*

Proof. Using Lemma 1.3.2 as $\operatorname{rank}(X) = r$, set $B \in \mathbb{R}^{r\times p}$ such that $Y_2B = XA_2B = X$. Let $Q = BA_1$. Then $Y_1 = XA_1 = XA_2BA_1 = Y_2Q$ as required.

[1]Thanks goes to stackexchange user ‘user1551’ for help with this proof: https://math.stackexchange.com/questions/3711293.

Furthermore,

$$\begin{aligned}
Q^\top Q &= A_1^\top B^\top B A_1 \\
&= A_1^\top B^\top \frac{Y_2^\top Y_2}{n-1} B A_1 \\
&= A_1^\top \frac{X^\top X}{n-1} A_1 \\
&= \frac{Y_1^\top Y_1}{n-1} \\
&= I_r,
\end{aligned}$$

and thus Q is orthogonal. □

1.3.2 Independent component analysis scenarios: simulator and output size

This section discusses the same four cases introduced in Section 1.2.3, but now in terms of independent component analysis. Figure 1.2 gives an overview of the four cases that can be encountered within the independent component analysis framework.

1.3.2.1 Case 1: $n < \infty$, $p < \infty$

As discussed in Section 1.2.3 with regards to principal component analysis, this is the case of the output of all simulators in practice.

When the data consists of n sets of p variables, $x_1, x_2, \ldots, x_n \in \mathbb{R}^p$, with $X = (x_1, x_2, \ldots, x_n)^\top \in \mathbb{R}^{n\times p}$, then the theory of independent component analysis is as given in Section 1.3.1.2 above.

1.3.2.2 Case 2: $n = \infty$, $p < \infty$

In this case the distribution of $\mathsf{X} \in \mathbb{R}^p$ is known. Therefore the true distribution of the standardised random variable Y is known, and thus so is the density of $\mathsf{Y}q$ for

Case 1: "Standard" ICA
$X = (x_1, \dots, x_n)^\top \in \mathbb{R}^{n\times p}$, whitened to $Y = (y_1, \dots, y_n)^\top \in \mathbb{R}^{n\times r}$
Aim: $q \in \mathbb{R}^{r-1}\mathrm{P}$ such that the (estimated) density of Yq maximises negentropy.

Case 3: "Functional ICA"
$x = (x_1, \dots, x_n)$, $x_i \in \mathrm{L}^2([0,T])$,
Aim: standardise x to $y = (y_1, \dots, y_n)$, $y_i \in \mathrm{L}^2([0,T])$,
and find a function $q\colon [0,T] \to \mathbb{R}$ such that the (estimated) density of $\int y_i q \, \mathrm{d}t$, $i = 1, \dots, n$ maximises negentropy

$$\begin{tikzcd}
n < \infty,\, p < \infty \arrow[r] \arrow[d] & n < \infty,\, p = \infty \arrow[d] \\
n = \infty,\, p < \infty \arrow[r] & n = \infty,\, p = \infty
\end{tikzcd}$$

Case 2: "Standard" ICA (with perfect data)
The distribution of random vector $\mathsf{X} \in \mathbb{R}^p$ is known.
Standardise to get random vector $\mathsf{Y} \in \mathbb{R}^p$.
Aim: $q \in \mathbb{R}^{p-1}\mathrm{P}$ such that the (true, known) density of Yq maximises negentropy.

Case 4: "Functional ICA" (with perfect data)
The distribution of the random function $(\mathsf{X}(t))_{t\in[0,T]}$ is known.
Standardised to, $(\mathsf{Y}(t))_{t\in[0,T]}$.
Aim: Find a function $q\colon [0,T] \to \mathbb{R}$ such that the distribution of $\int \mathsf{Y} q \, \mathrm{d}t$ maximises negentropy

Figure 1.2: Four ICA Cases
,

$q \in \mathbb{R}^{p-1}\mathrm{P}$. However, in general the negentropy will still need to be approximated, for instance using M-spacing as described in Section 1.3.1.3.

1.3.2.3 Case 3: $n < \infty$, $p = \infty$

Let $(\mathsf{X}(t))_{t\in[0,T]}$ be an $\mathbb{R}$-valued stochastic process, with $\mathsf{E}\mathsf{X}(t) = \mu(t) < \infty$ for $t \in [0,T]$, and suppose that we have n independent realisations of this random function. That is,

$$x(t) = \big(x_1(t), x_2(t), \dots, x_n(t)\big) \quad \text{for } t \in [0,T].$$

In the context of functional independent component analysis, we required $w \in \mathrm{L}^2([0,T])$ such that $\int_0^T w(t)^2 \, \mathrm{d}t = 1$ (notated $w \in \mathbb{R}^\infty\mathrm{P}$) minimises the mutual information of the density estimated from the points

$$\Big(\int_0^T x_1(t)w(t)\,\mathrm{d}t,\ \int_0^T x_2(t)w(t)\,\mathrm{d}t, \ldots,\ \int_0^T x_n(t)w(t)\,\mathrm{d}t\Big). \tag{1.14}$$

One question that arises immediately is whether it is possible to whiten the data x, as this is the motivation for using negentropy as the contrast function as discussed in Section 1.3.1.1. First, consider the standard empirical approximation to $C_x(s,t)$ for the sample x, given by

$$C_x(s,t) = \frac{1}{n}\sum_{i=1}^{n}\big(x_i(s) - \overline{x}(s)\big)\big(x_i(t) - \overline{x}(t)\big),$$

where $\bar{x} = 1/n\sum_{i=1}^{n} x_i$.

Now, for some $w \in \mathbb{R}^\infty\mathrm{P}$, the covariance function of $x_i(t)w(t)$, $i = 1, \ldots, n$ becomes

$$\begin{aligned} C_x^w(s,t) &= \frac{1}{n-1}\sum_{i=1}^{n}\big(x_i(s)w(s) - \overline{x}(s)w(s)\big)\big(x_i(t)w(t) - \overline{x}(t)w(t)\big) \\ &= \frac{1}{n-1}\,w(s)w(t)\sum_{i=1}^{n}\big(x_i(s) - \overline{x}(s)\big)\big(x_i(t) - \overline{x}(t)\big). \end{aligned}$$

For the whitening process when $p < \infty$ the aim is for the covariance matrix of the whitened data to be the identity matrix, and thus the variance of the projected whitened data for any projection $q \in \mathbb{R}^{r-1}\mathrm{P}$, $r \leq \min\{n-1, p\}$, is one. Naïvely expanding this to the $p = \infty$ case, the aim would be for the covariance function of the transformed data y to be equal to $C_y(s,t) = \delta_{\{s=t\}}$. However in this case,

$$\big(\mathcal{C}_y f\big)(t) = \int_0^T \delta_{\{s=t\}} f(s)\,\mathrm{d}s = f(t),$$

and thus $\mathcal{C}_y$ is not trace class (as $\sum_{i=1}^{\infty} v_i = \sum_{i=1}^{\infty} 1 = \infty$) and so is not a valid covariance operator.

One (computationally expensive) option that avoids the need for whitening is to scale the projected data (1.14) for every projection $w \in \mathbb{R}^\infty\mathrm{P}$ such that it has unit variance. Therefore, the negentropy calculated for each projection will be comparable. In this case, the steps for functional independent component analysis will be:

1. Choose $w \in \mathrm{S}^\infty$;
2. Project data to $\mathbb{R}$, $x^{(w)} = \left(\int_0^T x_1(t)w(t)\,\mathrm{d}t,\, \int_0^T x_2(t)w(t)\,\mathrm{d}t, \ldots,\, \int_0^T x_n(t)w(t)\,\mathrm{d}t\right)$;
3. Standardise the data $y^{(w)}$ such that $\hat{\sigma}_y^{(w)} := \frac{1}{n-1}(y^{(w)})^\top y^{(w)} = 1$;
4. Estimate the negentropy of the standardised data;
5. Repeat Steps 1. - 4. to obtain the $w \in \mathbb{R}^\infty\mathrm{P}$ that maximises negentropy.

1.3.2.4 Case 4: $n = \infty$, $p = \infty$

In this case the random function $\mathsf{X}(t)$, $t \in [0, T]$, is known exactly and thus so is $\mathsf{X}(t)w(t)$ for $w \in \mathbb{R}^\infty\mathrm{P}$. However, in general the negentropy of each projection will need to be approximated and maximised over all $w \in \mathrm{S}^\infty$. On top of this, the issue of whitening the random function $X(t)$, $t \in [0, T]$ still needs to be considered.

The limit of the size of some data, n, $p \to \infty$ is covered in Diaconis & Freedman (1984), where it is shown that in many cases virtually all one-dimensional projections of high-dimensional data are Gaussian. The case where $p = \infty$ and $n \to \infty$ is covered in von Weizsäcker (1997). These results are motivation for using independent component analysis in the high dimensional setting, as projections that result in non-Gaussian distributions are scarce and thus can be considered "interesting".

1.4 Methods For Independent Component Analysis

1.4.1 fastICA

This section has formed part of the paper titled 'On the Estimation of Entropy in the FastICA Algorithm', written by the author, Elena Issoglio and Jochen Voss, and published in the Journal of Multivariate Analysis (Smith *et al.*, 2020). In this section we introduce the fastICA method and establish some notational consistency. We then examine the approximations used to obtain the contrast function, and give a new understanding of some of the theoretical issues in the method.

A widely used method to perform independent component analysis in higher dimensions is fastICA (Hyvärinen & Oja, 2000). This method has found applications in areas as wide ranging as facial recognition (Draper *et al.*, 2003), epileptic seizure detection (Yang *et al.*, 2015) and fault detection in wind turbines (Farhat *et al.*, 2017). Recent works on extensions of the algorithm can be seen in Miettinen *et al.* (2014), Ghaffarian & Ghaffarian (2014) and He *et al.* (2017). The fastICA method uses a series of substitutions and approximations of the projected density and its entropy. It then applies an iterative scheme for optimising the resulting contrast function (which is an approximation to negentropy). Because of its popularity in many areas, analysis and evaluation of the strengths and weaknesses of the fastICA algorithm is crucially important. In particular, we need to understand both how well the contrast function estimates entropy and the performance of the optimisation procedure.

The main strength of the fastICA method is its speed, which is considerably higher than many other methods. Furthermore, if the data is a mixture of a small number of underlying factors, fastICA is often able to correctly identify these factors. However, fastICA also has some drawbacks, which have been pointed out in the literature. Learned-Miller & Fisher III (2003) use test problems from Bach & Jordan (2002) with performance measured by the Amari error (Amari *et al.*, 1996) to compare fastICA to other independent component analysis methods. They find that these perform better than fastICA on many examples. Focussing on a different aspect, Wei (2014) investigates issues with the convergence of the iterative scheme employed by fastICA to optimise the contrast function. In Wei (2017) it is shown that the two most common fastICA contrast functions fail to de-mix certain bimodal distributions with Gaussian mixtures, although some other contrast function choices (related to classical kurtosis estimation) may give reliable results within the fastICA framework.

In this section we identify and discuss a more fundamental problem with fastICA. We demonstrate that the approximations used in fastICA can lead to a contrast function where the optimal points no longer correspond to directions of low entropy.

1.4.1.1 The fastICA algorithm.

The exposition starts by describing the fastICA method from Hyvärinen & Oja (2000). The aim of this method is to efficiently find a projection of given (whitened) data which minimises entropy. The theory behind this method was originally introduced in Hyvärinen (1998), although here we adjust the notation to match the rest of this section. We will mention explicitly where our notation differs from Hyvärinen (1998) and Hyvärinen & Oja (2000). We will write 'fastICA' when we are discussing the theoretical method, and '`fastICA`' when we are discussing the `R` implementation from the `fastICA` CRAN package (Marchini *et al.*, 2013).

As the definition of entropy (given in Section 1.3.1) involves the integral of the density, the estimation of entropy or negentropy from data is not trivial. The M-spacing technique described in Section 1.3.1.3 is a good, albeit slow, approximation. The fastICA method provides a more efficient way to estimate negentropy $J[f]$ (given in (1.10)) by using a series of approximations and substitutions both for f and for $J[\cdot]$ to obtain a surrogate for negentropy $J[f]$ which is then subsequently maximised. The reason behind these substitutions is to reduce computational cost, but the drawback is that the resulting approximation may be very different from the true contrast function.

The fastICA method to obtain the first loading from data $X \in \mathbb{R}^{n\times p}$ follows the steps given below.

(i) Whiten the data to obtain $Y \in \mathbb{R}^{n \times r}$ with $r = \text{rank}(\tilde{X})$, where $\tilde{X}$ is column-wise centred version of X (see Remark 1.1), such that $C_Y = I_r$ (Hyvärinen & Oja, 2000, Section 5.2);

(ii) Iteratively find the optimal direction q^*, given by

$$q^* = \underset{q \in \mathbb{R}^{r-1}\text{P}}{\text{argmax}}\, \hat{J}^*(Yq), \tag{1.15}$$

where $\hat{J}^*$ is an approximation to negentropy, given in (1.22) below.

If more than one loading is required, then Step (ii) is repeated for each subsequent new direction, with the added constraint that q must be orthogonal to the previously found directions. This can be implemented within the fastICA framework using Gram-Schmidt orthogonalisation (Hyvärinen & Oja, 2000, Section 6.2). This is known as the *deflation* fastICA method. There is also a *parallel* fastICA method that finds all loadings concurrently, although we do not consider this here.

In the literature regarding fastICA it is often the convergence of the iterative method to solve (1.15) that is examined. It can be shown, for example in Wei (2014), that in certain situations this iterative step fails to find a good approximation for q^*. In contrast, here we consider the mathematical substitutions and approximations used in the derivation of $\hat{J}^*(Yq)$.

Assumption 1.4.1 given below introduces the technical assumptions given in Hyvärinen (1998, Sections 4 and 6). The notation used in this section has been slightly adjusted from Hyvärinen (1998) to aid understanding.

Assumption 1.4.1. *Let G_i, $i = 1, 2, \ldots, I$ be functions that do not grow faster than quadratically. Let $\varphi(\cdot)$ denote the density of a standard Gaussian random variable and assume that there are $\alpha_i, \beta_i, \gamma_i, \delta_i$, $i = 1, 2, \ldots, I$, such that the functions*

$$K_i(x) := \frac{G_i(x) + \alpha_i x^2 + \beta_i x + \gamma_i}{\delta_i}, \tag{1.16}$$

satisfy

$$\int_{\mathbb{R}} K_i(x)K_j(x)\varphi(x)\,dx = \mathbb{1}_{\{i=j\}};\ \textit{and,} \tag{1.17a}$$

$$\int_{\mathbb{R}} K_i(x)x^k\varphi(x)\,dx = 0,\ \textit{for}\ k = 0, 1, 2, \tag{1.17b}$$

for i, $j = 1, 2, \ldots, I$, *where* $\mathbb{1}_{\{i=j\}} = 1$ *if* $i = j$ *and zero otherwise.*

The functions G_i are given as $\bar{G}_i$ in Hyvärinen (1998) and as G_i in Hyvärinen & Oja (2000). The functions K_i are described in Hyvärinen (1998, Section 6) and are called G_i there.

The `fastICA` algorithm only implements the case $I = 1$. In this case, the function G_1 can be chosen nearly arbitrarily so long as it does not grow faster than quadratically: It is easy to show that for every G which is not exactly equal to a second order polynomial, a function K_1 can be found that satisfies the conditions given in (1.17) by choosing suitable α_1, β_1, γ_1 and δ_1. For general $I \in \mathbb{N}$, specific G_i, $i = 1, 2, \ldots, I$ must be chosen for the conditions (1.17) to hold. With $I = 2$, the functions $G_1(x) = x^3$ and $G_2(x) = x^4$ are proposed in the literature (Hyvärinen, 1998, Section 7) and seem to be useful in practice, even though these functions violate the growth condition from Assumption 1.4.1. We have not found any examples of specific functions G_i that satisfy (1.17) for $I > 2$ in the fastICA literature.

Let $q \in \mathbb{R}^{r-1}\mathrm{P}$ and $z = (z_1, z_2, \ldots, z_n) = Yq \in \mathbb{R}^n$ be the data projected onto q. Since the data has been whitened, z has sample mean 0 and sample variance 1. Further, let $f\colon \mathbb{R} \to \mathbb{R}$ be the unknown density of the population-level-whitened and projected data. Then f satisfies $\int f(x)\,dx = 1$, $\int x\,f(x)\,dx = 0$ and $\int x^2\,f(x)\,dx = 1$. We need to estimate the negentropy $J[f]$ using the data $z_1, \ldots, z_n$. Define

$$c_i := \mathsf{E}_f K_i(\mathsf{Z}) = \int f(x)K_i(x)\,\mathrm{d}x, \tag{1.18}$$

where Z is a random variable with density f, for all $i \in \{1, \ldots I\}$. For $I = 1$, setting $K(x) := K_1(x)$, $G(x) := G_1(x)$ and $c := c_1$, the derivation of the contrast function used in the fastICA method then consists of the steps given in Panel 1.5 below.

Panel 1.5: FastICA steps

1. Replace f by a density f_0 given by

$$f_0(x) = A \exp\Big(\kappa x + \zeta x^2 + aK(x)\Big), \tag{1.19}$$

for all $x \in \mathbb{R}$. The constants A, κ, ζ and a are chosen to minimise negentropy (maximise entropy) under the constraints $\int f_0(x)K(x)\,\mathrm{d}x = c$. In Proposition 1.4.3 we will show that $J[f_0] \leq J[f]$.

2. Approximate f_0 by $\hat{f}_0$ defined as

$$\hat{f}_0(x) = \varphi(x)\big(1 + cK(x)\big) \tag{1.20}$$

for all $x \in \mathbb{R}$. In Theorem 1.4.7 we will show $J[\hat{f}_0] \approx J[f_0]$.

3. Approximate $J[\hat{f}_0]$ by second order Taylor expansion,

$$\hat{J}[\hat{f}_0] = \frac{1}{C}\big(\mathsf{E}_f G(\mathsf{Z}) - \mathsf{E}_\varphi G(\mathsf{N})\big)^2, \tag{1.21}$$

where Z is a random variable with density f, $\mathsf{N} \sim \mathcal{N}(0,1)$, and C some constant. Note that, maybe surprisingly, Z has density f, not f_0. In Lemma 1.4.8 and Remark 1.7 we will show that $\hat{J}[\hat{f}_0] \approx J[\hat{f}_0]$.

4. Use Monte-Carlo approximation for the expectations in (1.21), *i.e.* use

$$\hat{J}^*(z) = \Big(\frac{1}{n}\sum_{j=1}^{n} G(z_j) - \frac{1}{L}\sum_{j=1}^{L} G(n_j)\Big)^2, \tag{1.22}$$

where $n_1, \ldots, n_L$ are samples from a standard Gaussian and L is large. Here $\hat{J}^*(z) \approx C\hat{J}[\hat{f}_0]$.

The steps in this chain of approximations are illustrated in Figure 1.3. The restriction to $I = 1$ here removes a summation from Step 1. and Step 2., therefore simplifying Step 3. and the associated estimations in Step 4. Theoretically these steps can be completed for arbitrary $I \in \mathbb{N}$, although in this case a closed-form

$$\begin{tikzcd}
f \arrow[r] & J[f] \arrow[d, phantom, "\rotatebox{90}{$\leq$}"] \\
f_0 \arrow[r] \arrow[d, phantom, "\wr"] & J[f_0] \arrow[d, phantom, "\wr"] \\
\hat{f}_0 \arrow[r] & J[\hat{f}_0] \arrow[d, phantom, "\wr"] \\
 & \hat{J}[\hat{f}_0] = \frac{1}{C}\big(\mathsf{E}_f G(\mathsf{Z}) - \mathsf{E}_\varphi G(\mathsf{N})\big)^2 \arrow[d, phantom, "\wr"] \\
 & \frac{1}{C}\,\hat{J}^*(y), \text{ where } \hat{J}^*(z) = \big(\frac{1}{n}\sum_{j=1}^{n} G(z_j) - \frac{1}{L}\sum_{j=1}^{L} G(n_j)\big)^2
\end{tikzcd}$$

Figure 1.3: Approximations used in fastICA: The fastICA contrast function $\hat{J}^*(y)$ is used in place of negentropy $J[f]$. Note that the first step involves an inequality rather than an approximation.

version equivalent to Step 3. is much more complicated.

Before these approximations are investigated more formally in the next section, this section is concluded with a few simple observations: The approximation to the negentropy used in fastICA dramatically decreases the computational time needed to find independent components. Unlike the M-spacing estimator introduced in Section 1.3.1.3, the approximation $\hat{J}^*(Yq) = \hat{J}^*(Xw)$ is a simple Monte-Carlo estimator and does not require sorting of the data. The algorithm to solve (1.15) also benefits from the fact that an approximate derivative of $q \mapsto \hat{J}^*(Yq)$ can be derived analytically.

In Step 1. of the procedure, we do not obtain a proper approximation, but have an inequality instead: f is replaced with a density f_0 such that $J[f_0] \leq J[f]$. As a result, the q which maximises $J[f_0]$ can be very different from the one which maximises $J[f]$. In contrast, Steps 2. and 3. are proper approximations and convergence of $\hat{f}_0$ to f_0 for Step 2. and of $\hat{J}[\hat{f}_0]$ to $J[f_0]$ for Step 3. in the limit $\|c\| \to 0$, where $c = (c_1, \ldots, c_I)$ is proved in Theorem 1.4.7 and Lemma 1.4.8, Remark 1.7 respectively. Step 4. is a simple Monte-Carlo approximation exhibiting well-understood behaviour. From the above discussion, it seems sensible to surmise that the loss of accuracy in fastICA is due to the surrogate used in Step 1. in

Panel 1.5.

We conclude this section with a few simple observations: Using (1.17), (1.16) and the fact that Z and N are standardized we find

$$\begin{aligned}
c_i &= \mathsf{E}_f K_i(\mathsf{Z}) \\
&= \mathsf{E}_f K_i(\mathsf{Z}) - \mathsf{E}_\varphi K_i(\mathsf{N}) \\
&= \mathsf{E}_f \left(\frac{G_i(\mathsf{Z}) + \alpha_i \mathsf{Z}^2 + \beta_i \mathsf{Z} + \gamma_i}{\delta_i} \right) - \mathsf{E}_\varphi \left(\frac{G_i(\mathsf{N}) + \alpha_i \mathsf{N}^2 + \beta_i \mathsf{N} + \gamma_i}{\delta_i} \right) \\
&= \frac{\mathsf{E}_f G_i(\mathsf{Z}) + \alpha_i 1 + \beta_i 0 + \gamma_i}{\delta_i} - \frac{\mathsf{E}_\varphi G_i(\mathsf{N}) + \alpha_i 1 + \beta_i 0 + \gamma_i}{\delta_i} \\
&= \frac{\mathsf{E}_f G_i(\mathsf{Z}) - \mathsf{E}_\varphi G_i(\mathsf{N})}{\delta_i}.
\end{aligned}$$

Thus, the fastICA objective function (ignoring the final Monte Carlo approximation) satisfies $\hat{J}[\hat{f}_0] \propto c^2$ for the case $I = 1$, considered above, and $\hat{J}[\hat{f}_0] \propto \sum_{i=1}^{I} c_i^2$ in the general case. Thus, fastICA can only see the data through the c_i. If the data are approximately Gaussian, we have $\mathsf{E}_f G_i(\mathsf{Z}) \approx \mathsf{E}_\varphi G_i(\mathsf{N})$ and $c_i \approx 0$ for all i and thus $\hat{J}[\hat{f}_0] \approx 0$, but the opposite implication does not hold. This is in contrast to the true negentropy, which satisfies $J[f] = 0$ if and only if f is Gaussian.

A first consequence of this argument is that projections where the true distribution is Gaussian will look ‘uninteresting’ to fastICA: for these directions, q, the objective function $\hat{J}^*(Yq)$ will be small and the search for the maximum in (1.15) will be driven away from these directions. This is particularly relevant since for high dimensional data, where the search volume is vast, projections along most directions are close to Gaussian (Diaconis & Freedman, 1984; von Weizsäcker, 1997), so fastICA will be able to exclude much of the search volume. Conversely, if $\hat{J}[\hat{f}_0]$ and thus $\|c\|$ is large, the projected density f is not Gaussian and by maximising (an approximation to) $\hat{J}[\hat{f}_0]$, the fastICA method will find directions which are ‘interesting’. But the above discussion also shows that optima can be missed when $\hat{J}[\hat{f}_0]$ is small, but the projected density f is still far from Gaussian. This is the case we are concerned with in this section and thus we assume $\|c\| \approx 0$ when we consider the fastICA approximations in detail in the next section.

1.4.1.2 Approximations used in the fastICA method

In this section, we investigate the validity of the approximation given in Section 1.4.1.1. We consider Step 1. from Panel 1.5 in Proposition 1.4.3, Step 2. in Theorem 1.4.7, and Step 3. in Proposition 1.4.9. Throughout this section, we consider arbitrary $I \in \mathbb{N}$ for completeness. This section is all novel work which increases understanding of the behaviour of the fastICA contrast function.

We first introduce some assumptions, in addition to Assumption 1.4.1, that are required for the mathematics in this section to hold.

Assumption 1.4.2. *There exists $\varepsilon > 0$ such that for all $h \in \mathbb{R}^I$ with $h^\top h < \varepsilon$, we have*

$$h^\top K(x) \geq -\frac{1}{2} \tag{1.23}$$

for all $x \in \mathbb{R}$, where $K(x) = \big(K_1(x), K_2(x), \dots, K_I(x)\big)$. In addition, there exists a function $M \colon \mathbb{R} \to \mathbb{R}$ such that

$$\sum_{i=1}^{I}\sum_{j=1}^{I}\sum_{k=1}^{I} |K_i(x)K_j(x)K_k(x)| \leq M(x) \text{ for all } x \in \mathbb{R}, \text{ and} \tag{1.24a}$$

$$\int_{\mathbb{R}} \varphi(x)M(x)\,dx =: \tilde{M} < \infty. \tag{1.24b}$$

Note that under the condition that each G_i does not grow faster than quadratically (given in Assumption 1.4.1), we can always find some positive constants B_i, $i = 1, 2, \dots, I$ such that

$$|K_i(x)| \leq B_i(1 + x^2), \tag{1.25}$$

for all $x \in \mathbb{R}$. Note also that for $I = 1$ the condition given by (1.23) that there exists an $\varepsilon > 0$ such that for all $h \in [0, \varepsilon)$, we have $hK(x) \geq -1/2$ is satisfied as follows. Let $\alpha, \beta, \gamma, \delta$ be parameters for which (1.17) holds. Then, (1.17) holds also for $\alpha, \beta, \gamma, -\delta$. Moreover, since G does not grow faster than quadratically, αx^2 is the dominant term in $K(x)$ as $x \to \pm\infty$. Therefore, to ensure that (1.23) holds it is enough to choose δ or $-\delta$ such that the sign is the same as that of α.

Step 1. (Panel 1.5) We start our discussion by considering Step 1. of the approximations. We prove that the distribution which maximises entropy for given values of $c_1, \dots, c_I$ is of the form (1.19) (where the last term is replaced by a sum, for arbitrary $I \in \mathbb{N}$) and thus that we indeed have $J[f_0] \le J[f]$.

Proposition 1.4.3. *Let f be the true density of the whitened data projected in some direction (thus with zero mean and unit variance). Recall c_i is defined by (1.18). The density f_0 that maximises entropy in the set*

$$\left\{g\colon \mathbb{R} \to \mathbb{R}\,;\ g \text{ is a density function, and } \int_{\mathbb{R}} g(x)K_i(x)\,dx = c_i,\ i = 1, 2, \dots, I\right\},$$

is given by,

$$f_0(x) = A\exp\left(\kappa x + \zeta x^2 + \sum_{i=1}^{I} a_i K_i(x)\right), \tag{1.26}$$

for some constants κ, ζ, A and a_i, $i = 1, 2, \dots, I$ that depend on c_i, $i = 1, 2, \dots, I$. It follows from this that $J[f_0] \le J[f]$.

Proof. We use the method of Lagrange multipliers in the calculus of variations (see, for example, Evans, 1998) to find a necessary condition for the density that maximises entropy given the constraints on mean and variance, and in (1.18). See Remark 1.4 for a short introduction to Lagrange multipliers.

Remark 1.4: Lagrange multipliers

The method of Lagrange multipliers is a common method used in optimisation to obtain saddle points of a function given some constraints. Intuitively, the method uses the fact that the gradient at level curves (which can be thought of as the contour lines) is perpendicular to the curve, and that at the maximum of the function given some constraint, the gradient of the constraint function is in the same direction as that of the level curve. That is, with $f\colon \mathbb{R}^2 \to \mathbb{R}$ some function that maps from the two dimensional space, and the aim to find the optimum of f given some constraint $g(x,y) = c$ where $c \in \mathbb{R}$. At this optimum, $\nabla f(x^*, y^*) - \lambda \nabla g(x^*, y^*) = 0$, for some $\lambda \in \mathbb{R}$.

Let $F[\cdot]\colon C^2 \to \mathbb{R}$ be a functional of the function $g\colon \mathbb{R} \to \mathbb{R}$, with $g \in C^2$, where C^2 is the set of all twice continuously differentiable functions. Then, the functional derivative $\delta F/\delta g : \mathbb{R} \to \mathbb{R}$ is explicitly defined by

$$\int_{\mathbb{R}} \frac{\delta F}{\delta g}(x)\phi(x)\,\mathrm{d}x := \frac{\mathrm{d}}{\mathrm{d}\varepsilon} F[g+\varepsilon\phi]\Big|_{\varepsilon=0} = \lim_{\varepsilon \downarrow 0}\left(\frac{F[g+\varepsilon\phi]-F[g]}{\varepsilon}\right), \tag{1.27}$$

for any function $\phi \in C^2$. The right-hand side of (1.27) is known as the Gâteaux differential $\mathrm{d}F(g;\phi)$ (Berger, 1977). Define the inner product of two functions by $\langle g, h\rangle := \int_{\mathbb{R}} g(x)h(x)\,\mathrm{d}x$, with norm $\|g\|_{\mathrm{L}^2} := \langle g, g\rangle^{\frac{1}{2}} = \left(\int_{\mathbb{R}} g(x)^2\,\mathrm{d}x\right)^{\frac{1}{2}}$. Now, the following system of equations needs to be solved:

$$\begin{cases} U[g](x) & := \frac{\delta}{\delta g}H[g] + \lambda_1 \frac{\delta}{\delta g}V[g] + \lambda_2 \frac{\delta}{\delta g}P[g] + \lambda_3 \frac{\delta}{\delta g}Q[g] + \sum_{i=1}^{I} \nu_i \frac{\delta}{\delta g}R_i[g] = 0; \\ V[g] & = 0; \\ P[g] & = 0; \\ Q[g] & = 0; \\ R_i[g] & = 0, \end{cases}$$

where $\lambda_1, \lambda_2, \lambda_3, \nu_i, i = 1, \dots, I$ are some real numbers, $H[g]$ is entropy as given in (1.7), and

$$\begin{aligned} V[g] &:= \mathrm{Var}[g] - 1 = \int_{\mathbb{R}} g(x)x^2\,\mathrm{d}x - \left(\int_{\mathbb{R}} g(x)x\,\mathrm{d}x\right)^2 - 1; \\ P[g] &:= \int_{\mathbb{R}} g(x)\,\mathrm{d}x - 1; \\ Q[g] &:= \int_{\mathbb{R}} g(x)x\,\mathrm{d}x; \\ R_i[g] &:= \int_{\mathbb{R}} g(x)K_i(x)\,\mathrm{d}x - c_i. \end{aligned}$$

Using (1.7) and (1.27) the term with H gives,

$$\begin{aligned}
\langle \frac{\delta H}{\delta g}, \phi \rangle &= -\frac{\mathrm{d}}{\mathrm{d}\varepsilon} \int \big(g(x) + \varepsilon\phi(x)\big) \log\big(g(x) + \varepsilon\phi(x)\big)\, \mathrm{d}x \Big|_{\varepsilon=0} \\
&= -\int \Big(g(x) \frac{\phi(x)}{g(x) + \varepsilon\phi(x)} + \phi(x) \log\big(g(x) + \varepsilon\phi(x)\big) + \varepsilon\phi(x) \frac{\phi(x)}{g(x) + \varepsilon\phi(x)} \Big)\, \mathrm{d}x \Big|_{\varepsilon=0} \\
&= -\int \big(1 + \log g(x)\big)\phi(x)\, \mathrm{d}x \\
&= \langle -1 - \log g(x), \phi \rangle.
\end{aligned}$$

Now, looking at $V[g]$ and using the constraint $Q[g] = 0$, the inner product,

$$\begin{aligned}
\langle \frac{\delta V}{\delta g}, \phi \rangle &= \frac{\mathrm{d}}{\mathrm{d}\varepsilon} \Big(\int \big(g(x) + \varepsilon\phi(x)\big) x^2\, \mathrm{d}x - \Big(\int \big(g(x) + \varepsilon\phi(x)\big) x\, \mathrm{d}x \Big)^2 - 1 \Big) \Big|_{\varepsilon=0} \\
&= \int \phi(x) x^2\, \mathrm{d}x - 2\Big(\int \phi(x) x\, \mathrm{d}x \cdot \int g(x) x\, \mathrm{d}x \Big) \\
&= \langle x^2, \phi \rangle - 2\langle x, \phi \rangle \cdot Q[g] \\
&= \langle x^2, \phi \rangle.
\end{aligned}$$

Let $L[\cdot]\colon C^2 \to \mathbb{R}$ be of the form $L[g] = \int g(x) l(x)\, \mathrm{d}x - k$ for some function $l\colon \mathbb{R} \to \mathbb{R}$, and some constant $k \in \mathbb{R}$. Then it is easy to check that $\langle \frac{\delta L}{\delta g}, \phi \rangle = \langle l, \phi \rangle$ and therefore

$$\frac{\delta P}{\delta g} = 1, \quad \frac{\delta Q}{\delta g} = x \quad \text{and} \quad \frac{\delta R_i}{\delta g} = K_i.$$

Putting this into the equation for $U[g]$ gives

$$U[g](x) = -1 - \log g(x) + \lambda_1 + \lambda_2 x^2 + \lambda_3 x + \sum_{i=1}^{I} \nu_i K_i(x).$$

Setting $U[g] = 0$ and solving for g gives, $g(x) = f_0(x) = \exp[\lambda_1 - 1 + \lambda_2 x^2 + \lambda_3 x + \sum_{i=1}^{I} \nu_i K_i(x)]$ which is (1.26) with $A = \exp(\lambda_1 - 1)$, $\kappa = \lambda_3$, $\zeta = \lambda_2$ and $a_i = \nu_i$, $i = 1, \dots, I$. Note that the constants A, κ, ζ, and a_i depend on c_i indirectly through the constraints on the K_i expressed as $R_i[g] = 0$. □

Remark 1.5: Supremum of $|J[f] - J[f_0]|$

It is possible to specify a density f such that in some limit, $H[f] \to \infty$ whilst $H[f_0]$ remains bounded, and thus $\big|J[f] - J[f_0]\big| \to \infty$, with f_0 the density given in (1.19). That is, in Step 1. of the fastICA method given in Panel 1.5, the difference between the true negentropy and the surrogate negentropy can be arbitrarily large. For example, set the density f to be a mixture of two independent uniform densities, i.e.

$$f(x) = \frac{1}{2}\big(g(x;\, -1-\varepsilon, -1) + g(x;\, 1, 1+\varepsilon)\big)$$

where $\varepsilon \in \mathbb{R}$ and $g(\cdot\,;\, a, b)$ is the density function of a uniform distribution in the interval $[a, b]$. Then the expectation and variance is given by

$$\mathsf{E}_f \mathsf{Z} = 0; \quad \mathrm{Var}_f\, \mathsf{Z} = 1 + \varepsilon + \frac{\varepsilon^2}{3}.$$

As the support of $g(\cdot\,;\, -1-\varepsilon, -1)$ is disjoint from that of $g(\cdot\,;\, 1, 1+\varepsilon)$, the entropy is,

$$H[f] = \frac{1}{2}\big(H[g(\cdot\,;\, -1-\varepsilon, -1)] + H[g(\cdot\,;\, 1, 1+\varepsilon)]\big) - \log(2).$$

Then, $H[f] \to -\infty$ as $\varepsilon \to 0$, since f tends to a pair of Dirac deltas. Also,

$$\mathsf{E}_f K_i(\mathsf{Z}) =: c_i \to \frac{1}{2}\big(K_i(-1) + K_i(1)\big), \tag{1.28}$$

as $\varepsilon \to 0$. With f_0 as in (1.19), then

$$c_i = \int K_i(x) f_0(x)\, \mathrm{d}x, \tag{1.29}$$

and,

$$\begin{aligned} H[f_0] &= \int f_0(x)\log(A)\,\mathrm{d}x + \int f_0(x)\big(\eta x + \kappa x^2 + \sum_{i=1}^{I} a_i K_i(x)\big)\,\mathrm{d}x \\ &= \log(A) + \eta \mathsf{E}_{f_0}\mathsf{X} + \kappa \mathsf{E}_{f_0}\mathsf{X}^2 + \sum_{i=1}^{I} a_i \mathsf{E}_{f_0} K_i(\mathsf{X}) \\ &= \log(A) + \kappa + \sum_{i=1}^{I} a_i c_i. \end{aligned}$$

Therefore, for $H[f_0]$ to be unbounded below as $\varepsilon \to 0$ it would be required for either $\kappa \to -\infty$, $a_i \to -\infty$ or $A \to 0$, as c_i is bounded by (1.28) and Assumption 1.4.2. However, this can not occur whilst f_0 satisfies (1.29).

Step 2. (Panel 1.5) We now switch our attention to Step 2. of the approximations. As discussed in Section 1.4.1, we consider the case where $c \to 0$. The first step of our analysis is to identify the behaviour of the constants in the definition of f_0 as $c \to 0$. We then prove a few auxiliary results before concluding our discussion of Step 2. in Theorem 1.4.7.

Proposition 1.4.4. *Suppose Assumption 1.4.1 is satisfied, and let $A, \kappa, \zeta, a_1, \ldots, a_I$ be defined as in Proposition 1.4.3, as functions of c. Then*

$$\begin{aligned} A - \frac{1}{\sqrt{2\pi}} &= \mathcal{O}(\|c\|^2) \\ \kappa &= \mathcal{O}(\|c\|^2) \\ \zeta + \frac{1}{2} &= \mathcal{O}(\|c\|^2) \\ a_i - c_i &= \mathcal{O}(\|c\|^2), \quad i = 1, 2, \ldots, I, \end{aligned}$$

as $\|c\| \to 0$.

Remark 1.6: Implicit function theorem

The proof of Proposition 1.4.4 uses the implicit function theorem, which allows solutions of a system of non-linear equations to be found. Under some conditions, it allows a system of non-linear equations (represented by some function $F\colon \mathbb{R}^{n+k} \to \mathbb{R}^k$) to be reduced to a system of linear equations, in some local area around a point (which is in the function's co-domain). That is, under some conditions omitted here, if $F(x_1, y_1) = 0$, then for every $x \in \mathbb{R}^n$ "close to" $x_1 \in \mathbb{R}^n$ there exists a $y \in \mathbb{R}^k$ "close to" $y_1 \in R^k$, such that $F(x, y) = 0$, and $y = g(x)$ for some $g\colon \mathbb{R}^n \to \mathbb{R}^k$. The function g is continuous and the derivative may be found by differentiating $F(x, g(x)) = 0$. From this derivative and by using Taylor series, an approximation for $g(x)$ around the point a can be found. This forms the basis of the proof of Proposition 1.4.4.

Proof. Define $x = (c_1, \dots, c_I)^\top \in \mathbb{R}^I$ and $y = (A, \kappa, \zeta, a_1, \dots, a_I)^\top \in \mathbb{R}^{I+3}$. Furthermore, let $F\colon \mathbb{R}^I \times \mathbb{R}^{I+3} \to \mathbb{R}^{I+3}$ be given by

$$F(x, y) = \begin{pmatrix} \int f_0(x)\,\mathrm{d}x - 1 \\ \int f_0(x)x\,\mathrm{d}x \\ \int f_0(x)x^2\,\mathrm{d}x - 1 \\ \int f_0(x)K_1(x)\,\mathrm{d}x - c_1 \\ \vdots \\ \int f_0(x)K_I(x)\,\mathrm{d}x - c_I \end{pmatrix},$$

where f_0 is given in (1.26) and K_i in (1.16). Then, for $x_1 = (0, \dots, 0)^\top$ and $y_1 = (\frac{1}{\sqrt{2\pi}}, 0, -\frac{1}{2}, 0, \dots, 0)^\top$, $F(x_1, y_1) = 0$.

Assuming F is twice differentiable, we use the Implicit Function Theorem (see, for example, de Oliveira, 2014) around (x_1, y_1). First, we need to show $D_yF(x_1, y_1)$

is invertible. We have

$$D_yF(x_1, y_1) = \begin{pmatrix} M & 0 \\ 0 & -I_I \end{pmatrix}, \quad \text{with, } M = \begin{pmatrix} \sqrt{2} & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 4 \end{pmatrix}.$$

Therefore, $D_yF(x_1, y_1)$ is non-singular, and so the Implicit Function Theorem holds. There exist some open set $\mathcal{U} \subset \mathbb{R}^I$ and a unique continuously differentiable function $g\colon \mathcal{U} \to \mathbb{R}^{I+3}$ such that $g(x_1) = y_1$ and $F\big(x, g(x)\big) = 0$ for all $x \in \mathcal{U}$. Then,

$$Dg(x) = -D_yF\big(x, g(x)\big)^{-1} D_xF\big(x, g(x)\big). \tag{1.30}$$

As g is continuous in the set $\mathcal{U}$, there exists some $\varepsilon > 0$, such that for all $c \in \mathcal{U}$ with $\|c\| < \varepsilon$, $g(x_1 + c) = y_1 + d$ for some $d \in \mathbb{R}^{I+3}$. Using Taylor series we can expand g around $x_1 = 0 \in \mathbb{R}^I$ to obtain $g(x_1 + c) = g(x_1) + Dg(x_1)\, c + \mathcal{O}(\|c\|^2)$, and

$$Dg(x_1) = \frac{d + \mathcal{O}(\|c\|^2)}{c}.$$

Putting this together with (1.30) at $x = x_1$ and rearranging gives,

$$d = -D_yF(x_1, y_1)^{-1} D_xF(x_1, y_1)\, c + \mathcal{O}(\|c\|^2).$$

Now, since

$$D_xF(x_1, y_1) = \begin{pmatrix} 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ & I_I & \end{pmatrix} \in \mathbb{R}^{(I+3)\times I},$$

one easily obtains that,

$$d = \begin{pmatrix} 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ & I_I & \end{pmatrix} c + \mathcal{O}(\|c\|^2),$$

and so,

$$y_1 + d = \begin{pmatrix} \frac{1}{\sqrt{2\pi}} \\ 0 \\ -\frac{1}{2} \\ c_1 \\ \vdots \\ c_I \end{pmatrix} + \mathcal{O}(\|c\|^2), \text{ as } c \to 0.$$

This completes the proof. □

We now define the following functions $y(\cdot)$ and $r(\cdot)$ for future use. Let $y\colon \mathbb{R} \to \mathbb{R}$ be given by

$$y(x) := \kappa x + (\zeta + \frac{1}{2})x^2 + \sum_{i=1}^{I} a_i K_i(x), \tag{1.31}$$

and $r\colon \mathbb{R} \to \mathbb{R}$ given by

$$r(x) := e^x - 1 - x. \tag{1.32}$$

Using these definitions, we can write f_0, given in Proposition 1.4.3, as

$$f_0(x) = \varphi(x) \cdot \sqrt{2\pi} A e^{y(x)}. \tag{1.33}$$

The following lemmas are two technical results needed in the proof of Theorem 1.4.7.

Lemma 1.4.5. *Let $g\colon \mathbb{R} \to \mathbb{R}$ and $l\colon \mathbb{R} \to \mathbb{R}$ be any functions and $h\colon \mathbb{R} \to \mathbb{R}_+$ be* *convex with $h(0) = 0$. Then,*

$$\sup_{x\in\mathbb{R}} \big| l(x)h(\varepsilon g(x)) \big| \leq \varepsilon \sup_{x\in\mathbb{R}} \big| l(x)h(g(x)) \big|$$

for all $\varepsilon \in [0,1]$.

Proof. As h is convex, for all $\lambda \in [0,1]$ and for all $x, y \in \mathbb{R}$, then $h\big(\lambda x + (1-\lambda)y\big) \leq \lambda h(x) + (1-\lambda)h(y)$. Let $\varepsilon \in [0,1]$. Substituting $\lambda = \varepsilon$, $x = g(x)$ and $y = 0$, $h\big(\varepsilon\, g(x)\big) \leq \varepsilon\, h\big(g(x)\big)$ for all $g(x) \in \mathbb{R}$, as $h(0) = 0$. Noticing that h maps to the positive real line concludes the proof. □

Lemma 1.4.6. *Let $r\colon \mathbb{R} \to \mathbb{R}_+$ be given as in (1.32). Then,*

$$r(\varepsilon\, y) \leq \varepsilon^2 r(y), \text{ for all } y \geq 0, \text{ and for all } \varepsilon \in [0,1]. \tag{1.34}$$

Moreover, for any function $l\colon \mathbb{R} \to \mathbb{R}$, we have

$$\sup_{x\in\mathbb{R}} \Big| l(x) r\big(\varepsilon(1+x^2)\big) \Big| \leq \varepsilon^2 \sup_{x\in\mathbb{R}} \Big| l(x) r(1+x^2) \Big|.$$

Proof. We will use the Taylor expansion of the exponential around 0 for both the left-hand and right-hand side of (1.34). The left-hand side gives,

$$\begin{aligned} r(\varepsilon\, y) &= \exp(\varepsilon\, y) - 1 - \varepsilon\, y \\ &= \sum_{n=0}^{\infty} \frac{\varepsilon^n}{n!} y^n - 1 - \varepsilon\, y, \quad \text{absolutely convergent for all } \varepsilon y \in \mathbb{R} \\ &= \varepsilon^2 \left(\sum_{n=2}^{\infty} \frac{\varepsilon^{n-2}}{n!} y^n \right) \end{aligned}$$

and the right-hand side of (1.34) gives,

$$\varepsilon^2\, r(y) = \varepsilon^2 \Big(\sum_{n=0}^{\infty} \frac{1}{n!}\, y^n - 1 - y\Big) = \varepsilon^2 \Big(\sum_{n=2}^{\infty} \frac{1}{n!}\, y^n\Big).$$

Putting these two results together,

$$r(\varepsilon\, y) - \varepsilon^2\, r(y) = \varepsilon^2 \Big(\sum_{n=2}^{\infty} \frac{1}{n!}\, y^n (\varepsilon^{n-2} - 1)\Big) \le 0,$$

as $\varepsilon^n - 1 \le 0$ for all $\varepsilon \in [0,1]$ and $n \in \mathbb{N}_+$. This proves (1.34).

Let $l\colon \mathbb{R} \to \mathbb{R}$ be some function. Then, as r maps to the positive real line and using (1.34) with $y = 1 + x^2$, we have

$$|l(x) r\big(\varepsilon(1+x^2)\big)| \le \varepsilon^2 |l(x) r(1+x^2)|,$$

for all $x \in \mathbb{R}$. Taking the supremum over the real line we conclude. $\square$

We now consider the error term between the density f_0 that maximises entropy, and its estimate $\hat{f}_0$.

Theorem 1.4.7. *Suppose we have functions K_i, $i = 1, 2, \dots, I$ that satisfy Assumptions 1.4.1 and 1.4.2. Let f_0 be given as in Proposition 1.4.3, and $\hat{f}_0$ be given by*

$$\hat{f}_0(x) = \varphi(x)\Big(1 + \sum_{i=1}^{I} c_i K_i(x)\Big).$$

Then,

$$\sup_{x \in \mathbb{R}} \Big| e^{\delta x^2} \big(f_0(x) - \hat{f}_0(x)\big)\Big| = \mathcal{O}(\|c\|^2) \text{ as } c \to 0,$$

for all $\delta < 1/2$.

Proof. Let $\varphi(x) = (2\pi)^{1/2} e^{-x^2/2}$ be the density of a standard Gaussian random variable and let the function $g\colon \mathbb{R} \to \mathbb{R}$ be defined by

$$g(x) := \frac{f_0(x) - \hat{f}_0(x)}{\varphi(x)}.$$

Then, with $y\colon \mathbb{R} \to \mathbb{R}$ as defined in (1.31) and using (1.33), we get,

$$\begin{aligned} g(x) &= \sqrt{2\pi}A\exp\big(y(x)\big) - \big(1 + \sum_{i=1}^{I} c_i K_i(x)\big) \\ &= \sqrt{2\pi}A\Big(\exp\big(y(x)\big) - 1 - y(x)\Big) + \sqrt{2\pi}A\big(1 + y(x)\big) - (1 + \sum_{i=1}^{I} c_i K_i(x)) \\ &\quad + \sqrt{2\pi}A\big(\sum_{i=1}^{I} c_i K_i(x) - \sum_{i=1}^{I} c_i K_i(x)\big). \end{aligned}$$

Rearranging this using the function $r\colon \mathbb{R} \to \mathbb{R}$ given in (1.32) and by expanding $y(x)$ gives,

$$\begin{aligned} g(x) &= \sqrt{2\pi}A \cdot r\big(y(x)\big) + \sqrt{2\pi}A \cdot \Big(\kappa x + \big(\zeta + \frac{1}{2}\big)x^2\Big) \\ &\quad + \big(\sqrt{2\pi}A - 1\big)\sum_{i=1}^{I} c_i K_i(x) + \sqrt{2\pi}A\sum_{i=1}^{I}(a_i - c_i)K_i(x) + (\sqrt{2\pi}A - 1). \end{aligned}$$

Note that the absolute value of $g(x)$ can be bounded by the following terms,

$$\begin{aligned} |g(x)| &\le \sqrt{2\pi}A\,|r\big(y(x)\big)| + \sqrt{2\pi}A\,|\kappa x| + \sqrt{2\pi}A\,|\zeta + \frac{1}{2}|x^2 \\ &\quad + \sqrt{2\pi}A\,\big|\sum_{i=1}^{I}(a_i - c_i)K_i(x)\big| + |\sqrt{2\pi}A - 1|\,\big|\sum_{i=1}^{I} c_i K_i(x)\big| + |\sqrt{2\pi}A - 1|. \end{aligned}$$

We have,

$$\begin{aligned}|f_0(x) - \hat{f}_0(x)| &= |\varphi(x) \cdot g(x)| \\ &= \varphi(x)\Big|\sqrt{2\pi}Ar\big(y(x)\big) + \sqrt{2\pi}A\Big(\kappa x + \big(\zeta + \frac{1}{2}\big)x^2\Big) \\ &\qquad + (\sqrt{2\pi}A - 1)\sum_{i=1}^{I} c_i K_i(x) \\ &\qquad\quad + \sqrt{2\pi}A\sum_{i=1}^{I}(a_i - c_i)K_i(x) + (\sqrt{2\pi}A - 1)\Big|.\end{aligned}$$

Multiplying both sides by $e^{\delta x^2}$ and setting $\tilde{\delta} = \frac{1}{2} - \delta$ so that $e^{\delta x^2}\varphi(x) = (2\pi)^{-1/2}e^{-\tilde{\delta}x^2}$, gives

$$\begin{aligned}&\big|e^{\delta x^2}\big(f_0(x) - \hat{f}_0(x)\big)\big| \\ &\quad = (2\pi)^{-1/2}e^{-\tilde{\delta}x^2}\Big|\sqrt{2\pi}Ar\big(y(x)\big) + \sqrt{2\pi}A\Big(\kappa x + \big(\zeta + \frac{1}{2}\big)x^2\Big) \\ &\qquad + (\sqrt{2\pi}A - 1)\sum_{i=1}^{I} c_i K_i(x) + \sqrt{2\pi}A\sum_{i=1}^{I}(a_i - c_i)K_i(x) \\ &\qquad + (\sqrt{2\pi}A - 1)\Big| \\ &\quad \leq T_1(x) + T_2(x) + \frac{1}{\sqrt{2\pi}} \cdot T_3(x) + T_4(x) + \frac{1}{\sqrt{2\pi}} \cdot T_5(x), \qquad (1.35)\end{aligned}$$

where,

$$\begin{aligned}
T_1(x) &:= \big|Ae^{-\tilde{\delta}x^2} r\big(y(x)\big)\big|;\\
T_2(x) &:= \big|Ae^{-\tilde{\delta}x^2}\big(\kappa x + (\zeta + \frac{1}{2})x^2\big)\big|;\\
T_3(x) &:= \big|(\sqrt{2\pi}A - 1)e^{-\tilde{\delta}x^2}\sum_{i=1}^{I} c_i K_i(x)\big|;\\
T_4(x) &:= \big|Ae^{-\tilde{\delta}x^2}\sum_{i=1}^{I}(a_i - c_i)K_i(x)\big|;\\
T_5(x) &:= |e^{-\tilde{\delta}x^2}(\sqrt{2\pi}A - 1)|.
\end{aligned}$$

If we show that $\|T_i\|_\infty$ is at least of order $\|c\|^2$ as $c \to 0$ for $i = 1, \ldots, 5$, then we can conclude the proof by taking the supremum of (1.35) over $x \in \mathbb{R}$, which gives,

$$\sup_{x\in\mathbb{R}}\big|e^{\delta x^2}(f_0(x) - \hat{f}_0(x))\big| = \mathcal{O}(\|c\|^2),$$

as $c \to 0$. This is how the proof proceeds, by showing that the norm of each term is at least of order $\|c\|^2$.

Term T_1. First, note that

$$\big|e^{-\tilde{\delta}x^2} r(y(x))\big| \leq \max_{\sigma\in\{-1,1\}}\big|e^{-\tilde{\delta}x^2} r(\sigma \cdot |y(x)|)\big|, \text{ for all } x \in \mathbb{R},$$

and thus,

$$\sup_{x\in\mathbb{R}}|T_1(x)| \leq A \cdot \sup_{\substack{x\in\mathbb{R}\\ \sigma\in\{-1,1\}}} |e^{-\tilde{\delta}x^2} r(\sigma \cdot |y(x)|)|. \tag{1.36}$$

Next we choose γ such that,

$$m_1 := \sup_{\substack{x\in\mathbb{R}\\ \sigma\in\{-1,1\}}} \big|e^{-\tilde{\delta}x^2} r\big(\sigma \cdot \gamma(1 + x^2)\big)\big| < \infty. \tag{1.37}$$

This is always possible for some $\gamma \in (-\tilde{\delta}, \tilde{\delta})$, as $r(0) = 0$, and since $e^{-\tilde{\delta}x^2} r\big(\pm\gamma(1 + x^2)\big)$ is continuous and $r\big(\pm\gamma(1 + x^2)\big)$ grows no faster that $e^{\gamma x^2}$ as $x \to \pm\infty$, it is beaten by $e^{-\tilde{\delta}x^2}$ in the tails.

For $y(x)$ as given in (1.31) and using (1.25) we can find an upper bound by

$$\begin{aligned} |y(x)| &\leq |\kappa| \cdot \Big(\frac{1 + x^2}{2}\Big) + |\zeta + \frac{1}{2}| \cdot (1 + x^2) + \sum_{i=1}^{I} |a_i| B_i (1 + x^2) \\ &= \gamma(1 + x^2) \cdot \frac{1}{\gamma}\Big(\frac{1}{2}|\kappa| + |\zeta + \frac{1}{2}| + \sum_{i=1}^{I} |a_i| B_i\Big) \\ &= \gamma(1 + x^2) \cdot \varepsilon_1, \end{aligned} \tag{1.38}$$

where γ is such that (1.37) holds, and $\varepsilon_1 := 1/\gamma \cdot \big(|\kappa|/2 + |\zeta + 1/2| + \sum_{i=1}^{I} |a_i| B_i\big)$. As $c \to 0$ we have by Proposition 1.4.4, $\kappa \to 0$, $\zeta \to -1/2$ and $a_i \to c_i$. Therefore, we can choose c small enough (and depending on γ) such that $\varepsilon_1 \in [0, 1]$. Now, from (1.36), (1.38), the fact that r is convex with a minimum at zero, and by Lemma 1.4.6, we get

$$\begin{aligned} \sup_{x \in \mathbb{R}} \big|T_1(x)\big| &\leq A \sup_{\substack{x \in \mathbb{R} \\ \sigma \in \{-1,1\}}} \big|e^{-\tilde{\delta}x^2} r\big(\sigma\gamma(1 + x^2)\varepsilon_1\big)\big| \\ &\leq \varepsilon_1^2 A \sup_{\substack{x \in \mathbb{R} \\ \sigma \in \{-1,1\}}} \big|e^{-\tilde{\delta}x^2} r\big(\sigma\gamma(1 + x^2)\big)\big| \\ &= \varepsilon_1^2 A m_1. \end{aligned}$$

By Proposition 1.4.4, we have $A \to 1/\sqrt{2\pi}$ as $c \to 0$ and,

$$\varepsilon_1 = \frac{1}{\gamma}\Big(\frac{1}{2}|\kappa| + |\zeta + \frac{1}{2}| + \sum_{i=1}^{I} |a_i| B_i\Big) = \mathcal{O}(\|c\|), \text{ as } c \to 0,$$

and therefore $\varepsilon_1^2 = \mathcal{O}(\|c\|^2)$ as $c \to 0$, and $\|T_1\|_\infty = \mathcal{O}(\|c\|^2)$ as $c \to 0$.

Term T_2. We proceed similarly as for T_1, and look for some $\varepsilon_2 \in [0, 1]$ such that $|\kappa x + (\zeta + \frac{1}{2})x^2| \leq \varepsilon_2(1 + x^2)$. Then, we have

$$\begin{aligned}\left|\kappa x + (\zeta + \frac{1}{2})x^2\right| &\leq |\kappa|(\frac{1 + x^2}{2}) + |\zeta + \frac{1}{2}|(1 + x^2)\\ &= \Big(\frac{1}{2}|\kappa| + |\zeta + \frac{1}{2}|\Big)(1 + x^2).\end{aligned}$$

Setting $\varepsilon_2 := (\frac{1}{2}|\kappa| + |\zeta + \frac{1}{2}|)$, by Proposition 1.4.4, $\varepsilon_2 = \mathcal{O}(\|c\|^2)$ as $c \to 0$, and thus we can choose c sufficiently small such that $\varepsilon_2 \leq 1$. Let,

$$m_2 := \sup_{x\in\mathbb{R}}\left|e^{-\tilde{\delta}x^2}(1 + x^2)\right| < \infty,$$

where $m_2 < \infty$ since $e^{-\tilde{\delta}x^2}(1 + x^2)$ is continuous and tends to zero in the tails. From this, for $\varepsilon_2 \in [0, 1]$ as above, we can apply Lemma 1.4.5 and get

$$\begin{aligned}\sup_{x\in\mathbb{R}}\left|e^{-\tilde{\delta}x^2}(\kappa x + (\zeta + \frac{1}{2})x^2)\right| &\leq A \sup_{x\in\mathbb{R}}\left|e^{\tilde{\delta}x^2}\varepsilon_2(1 + x^2)\right|\\ &\leq \varepsilon_2 \sup_{x\in\mathbb{R}}\left|e^{-\tilde{\delta}x^2}(1 + x^2)\right| = \varepsilon_2\, m_2.\end{aligned}$$

Then,

$$\sup_{x\in\mathbb{R}}\left|T_2(x)\right| = A \sup_{x\in\mathbb{R}}\left|e^{\tilde{\delta}x^2}(\kappa x + (\zeta + \frac{1}{2})x^2)\right| \leq A\,\varepsilon_2\, m_2.$$

Therefore, we have $\|T_2\|_\infty = \mathcal{O}(\|c\|^2)$, as $c \to 0$.

Term $\boldsymbol{T_3}$. As with the T_2 term, we want an $\varepsilon_3 \in [0, 1]$ such that $|\sum_{i=1}^{I} c_i K_i(x)| \leq \varepsilon_3(1 + x^2)$, so that we can apply Lemma 1.4.5 to show

$$\sup_{x\in\mathbb{R}}\left|e^{-\tilde{\delta}x^2}\sum_{i=1}^{I} c_i K_i(x)\right| \leq \varepsilon_3 \sup_{x\in\mathbb{R}}\left|e^{-\tilde{\delta}x^2}(1 + x^2)\right| < \infty.$$

First, note that by (1.25),

$$\Big|\sum_{i=1}^{I} c_i K_i(x)\Big| \le \Big|\sum_{i=1}^{I} c_i B_i (1+x^2)\Big|,$$
$$= \Big|\sum_{i=1}^{I} c_i B_i\Big| \cdot (1+x^2),$$

and thus we set $\varepsilon_3 := \big|\sum_{i=1}^{I} c_i B_i\big|$. Clearly, $\varepsilon_3 = \mathcal{O}(\|c\|)$ as $c \to 0$. Now, with c sufficiently small such that $\varepsilon_3 \in [0,1]$, we have by Lemma 1.4.5,

$$\sup_{x\in\mathbb{R}} \Big|e^{-\tilde{\delta}x^2} \sum_{i=1}^{I} c_i K_i(x)\Big| \le \sup_{x\in\mathbb{R}} \Big|e^{-\tilde{\delta}x^2} \big(\Big|\sum_{i=1}^{I} c_i B_i\Big|\big)(1+x^2)\Big|$$
$$\le \Big|\sum_{i=1}^{I} c_i B_i\Big| \cdot \sup_{x\in\mathbb{R}} \Big|e^{-\tilde{\delta}x^2}(1+x^2)\Big|$$
$$\le \varepsilon_3\, m_2.$$

Therefore,

$$\sup_{x\in\mathbb{R}} \big|T_3(x)\big| \le |\sqrt{2\pi}A - 1|\varepsilon_3\, m_2$$

Thus, $\|T_3\|_\infty = \mathcal{O}(\|c\|^3)$, as $c \to 0$, since $|\sqrt{2\pi}A - 1| = \mathcal{O}(\|c\|^2)$ and $\varepsilon_3 = \mathcal{O}(\|c\|)$ as $c \to 0$.

Term $\boldsymbol{T_4}$. Similar to the T_2 and T_3 terms, we want an $\varepsilon_4 \in [0,1]$ such that $\sum_{i=1}^{I}(a_i - c_i)K_i(x) \le \varepsilon_4(1+x^2)$. Note that

$$\Big|\sum_{i=1}^{I}(a_i - c_i)K_i(x)\Big| \le \Big|\sum_{i=1}^{I}(a_i - c_i)B_i\Big| \cdot (1+x^2),$$

by (1.25) and thus we set $\varepsilon_4 := |\sum_{i=1}^{I}(a_i - c_i)B_i|$, and by Proposition 1.4.4, $\varepsilon_4 = \mathcal{O}(\|c\|^2)$ as $c \to 0$. Choose c small enough such that $\varepsilon_4 \in [0,1]$. Then, by Lemma 1.4.5,

$$\begin{aligned}\sup_{x\in\mathbb{R}}|T_4(x)| &\le A\sup_{x\in\mathbb{R}}\Big|e^{-\tilde{\delta}x^2}\sum_{i=1}^{I}(a_i-c_i)B_i(1+x^2)\Big|\\ &= \varepsilon_4\,A\,m_2,\end{aligned}$$

and since $\varepsilon_4 = \mathcal{O}(\|c\|^2)$ as $c\to 0$, we have $\|T_4\|_\infty = \mathcal{O}(\|c\|^2)$, as $c\to 0$.

Term T_5. Here we can use the inequality $e^{\tilde{\delta}x^2}\le 1$ for all $x\in\mathbb{R}$ and from Proposition 1.4.4 we have

$$T_5(x)\le |\sqrt{2\pi}A-1| = \mathcal{O}(\|c\|^2),\ \text{ as } c\to 0.$$

This completes the proof. □

We have therefore shown that for sufficiently small c, the approximation $\hat{f}_0$ for the density that maximises entropy given the constraints in (1.18) is 'close to' f_0. We have also shown that the speed of convergence is of order $\|c\|^2$.

Step 3. (Panel 1.5) We now turn our attention to Step 3. of the approximations, where we find approximations for the entropy and negentropy of $\hat{f}_0$. For these proofs we require that $\hat{f}_0(x)\ge 0$ for all $x\in\mathbb{R}$, and therefore $\hat{f}_0$ is a density.

Lemma 1.4.8 (Approximation of Entropy). *Suppose Assumptions 1.4.1 and 1.4.2 hold, and let $\hat{f}_0$ be given as in Theorem 1.4.7. Suppose also that $\hat{f}_0(x)\ge 0$ for all $x\in\mathbb{R}$. Then the entropy of $\hat{f}_0$ satisfies*

$$H[\hat{f}_0] = \hat{H}[\hat{f}_0] + R(\hat{f}_0),$$

where,

$$\hat{H}[\hat{f}_0] := \eta(1) - \frac{1}{2}\|c\|^2, \tag{1.39}$$

with $\eta(\cdot)$ given in (1.8), $c = (c_1, c_2, \dots, c_I)^\top$, with the c_i defined in Proposition 1.4.3 and the remainder term bounded by

$$|R(\hat{f}_0)| \le C\,\tilde{M} \cdot \|c\|^3,$$

for some constant $C \in \mathbb{R}\backslash\{-\infty, \infty\}$, and $\tilde{M}$ given in Assumption 1.4.2.

Proof. Set $K(x) = (K_1(x), K_2(x), \dots, K_I(x))^\top$, for $x \in \mathbb{R}$. Now, with $\hat{f}_0$ as in Theorem 1.4.7, expanding $H[\hat{f}_0]$ gives,

$$\begin{aligned}
H[\hat{f}_0] &= -\int \hat{f}_0(x) \log \hat{f}_0(x)\,\mathrm{d}x \\
&= -\int \varphi(x)\Big(1 + c^\top K(x)\Big)\Big(\log\varphi(x) + \log\big(1 + c^\top K(x)\big)\Big) \\
&= -\int \varphi(x)\log\varphi(x)\,\mathrm{d}x - \int \varphi(x) c^\top K(x) \log\varphi(x)\,\mathrm{d}x \\
&\qquad - \int \varphi(x)\big(1 + c^\top K(x)\big)\log\big(1 + c^\top K(x)\big)\,\mathrm{d}x \\
&= \eta(1) - \int \varphi(x) c^\top K(x)\big(-\frac{1}{2}\log(2\pi) - \frac{1}{2}x^2\big)\,\mathrm{d}x \\
&\qquad - \int \varphi(x)\big(1 + c^\top K(x)\big)\log\big(1 + c^\top K(x)\big)\,\mathrm{d}x \\
&= \eta(1) - 0 - \int \varphi(x)\big(1 + c^\top K(x)\big)\log\big(1 + c^\top K(x)\big)\,\mathrm{d}x,
\end{aligned}$$

using the constraints given in (1.17). To obtain the approximation $\hat{H}[\hat{f}_0]$ and remainder $R(\hat{f}_0)$ terms, we consider the expansion of $\big(1+c^\top K(x)\big)\log\big(1+c^\top K(x)\big)$ around $c = 0$ using the Taylor series. Let $q(y) = y\log(y)$, $y \in \mathbb{R}$. Then,

$$q'(y) = \log(y) + 1; \quad q''(y) = \frac{1}{y}; \quad \text{and } q'''(y) = -\frac{1}{y^2}.$$

and thus using Taylor series around y_0 gives $q(y_0 + h) = h + \frac{1}{2}h^2 + R_1(y_0, h)$, where

$R_1(y_0, h)$ is the remainder term given by

$$\begin{aligned} R_1(y_0, h) &= \int_{y_0}^{y_0+h} \frac{(y_0 + h - \tau)^2}{2} \Big(\frac{-1}{\tau^2}\Big)\, \mathrm{d}\tau \\ &= -h^3 \int_0^1 \frac{(1-t)^2}{2(1+th)^2}\, \mathrm{d}t \end{aligned}$$

with the change of variables $\tau = (y_0 + th)$.

Now let us pick $y_0 = 1$ and $h = c^\top K(x)$ and denote by $R_2(x)$ the corresponding remainder $R_2(x) = R_1(1,\, c^\top K(x))$. Then,

$$H[\hat{f}_0] = \eta(1) - \int \varphi(x)\Big(c^\top K(x) + \frac{1}{2}\big(c^\top K(x)\big)^2 + R_2(x)\Big)\, \mathrm{d}x, \tag{1.40}$$

where the remainder term $R_2(x)$ is given explicitly by

$$R_2(x) = -\big(c^\top K(x)\big)^3 \int_0^1 \frac{(1-t)^2}{2\big(1 + tc^\top K(x)\big)^2}\, \mathrm{d}t.$$

Now using (1.17) and setting

$$R(\hat{f}_0) := -\int_{\mathbb{R}} \varphi(x)\, R_2(x)\, \mathrm{d}x \tag{1.41}$$

we get from (1.40),

$$\begin{aligned} H[\hat{f}_0] &= \eta(1) + 0 - \frac{1}{2}\sum_{i=1}^{I} c_i^2 + R(\hat{f}_0) \\ &= \hat{H}[\hat{f}_0] + R(\hat{f}_0), \end{aligned}$$

as needed to be shown. It remains to prove the bound for $R(\hat{f}_0)$.

From Assumption 1.4.2 there exists some $\varepsilon > 0$ such that $c^\top K(x) \geq -1/2$, $K(x) = (K_1(x), K_2(x), \ldots, K_I(x))^\top$, for all c with $c^\top c \leq \varepsilon$ for all $x \in \mathbb{R}$, and therefore,

$$\begin{aligned}
|R_2(x)| &= \left|\left(c^\top K(x)\right)^3 \int_0^1 \frac{(t-1)^2}{2\cdot(1+tc^\top K(x))^2}\,\mathrm{d}t\right| \\
&\le \left|\left(c^\top K(x)\right)\right|^3 \cdot \left|\int_0^1 \frac{(t-1)^2}{2\cdot(1-t/2)^2}\,\mathrm{d}t\right| \\
&= C\cdot\left|\left(c^\top K(x)\right)\right|^3,
\end{aligned}$$

where $C \in \mathbb{R}$, as the integral is of a continuous function over a compact set.

Now, there exists some $\delta > 0$ such that for all $c^\top c \le \delta$, $c_i \le \|c\|$ for $i = 1, 2, \ldots, I$. Then, with $c^\top c \le \min(\varepsilon, \delta)$, the absolute value of $R_2(x)$ can be bounded above by

$$\begin{aligned}
|R_2(x)| &\le C \sum_{i,j,k=1}^{I} \left|K_i(x)K_j(x)K_k(x)\right| \cdot \|c\|^3 \\
&\le C \cdot M(x) \cdot \|c\|^3
\end{aligned}$$

having used (1.24a) from Assumption 1.4.2. Putting this all together we obtain the bound for $R(\hat{f}_0)$,

$$\left|R(\hat{f}_0)\right| \le \int_{\mathbb{R}} \varphi(x)|R_2(x)|\,\mathrm{d}x \le C\tilde{M}\|c\|^3,$$

where $\tilde{M}$ is given in (1.24b), as required. $\square$

Remark 1.7: Approximation of negentropy

Note that the density $\hat{f}_0$ has unit variance. Indeed, by (1.17),

$$\begin{aligned}\int \hat{f}_0(x)x^2\,\mathrm{d}x &= \int \varphi(x)\big(1+\sum_{i=1}^{I} c_i K_i(x)\big)\,\mathrm{d}x \\ &= \int \varphi(x)x^2\,\mathrm{d}x + \sum_{i=1}^{I} c_i \int \varphi(x)K_i(x)x^2\,\mathrm{d}x = 1.\end{aligned}$$

Therefore, the negentropy equivalent of the entropy approximation given in Lemma 1.4.8 is $J[\hat{f}_0] = \hat{J}[\hat{f}_0] + R(\hat{f}_0)$ with $R(\hat{f}_0)$ given as in (1.41) and

$$\hat{J}[\hat{f}_0] = \frac{1}{2}\|c\|^2 = \frac{1}{2}\sum_{i=1}^{I} c_i^2.$$

Proposition 1.4.9. *With the same assumptions as in Lemma 1.4.8. Set $I = 1$. Then,*

$$\hat{J}[\hat{f}_0] \propto \big(\mathsf{E}_f G(\mathsf{Z}) - \mathsf{E}_\varphi G(\mathsf{N})\big)^2,$$

where Z is a random variable with density f and $\mathsf{N} \sim \mathcal{N}(0,1)$.

Proof. By the constraints that need to be satisfied by K (given in Assumption 1.4.2), we have $\int \varphi(x)K(x)x^k\,\mathrm{d}x = 0$ for $k = 0,1,2$. Substituting the form of $K(x)$ as in (1.16) into (1.17) and solving these three equations gives an explicit expression for α, β, γ in terms of G,

$$\begin{aligned}\alpha &= \frac{1}{2}\Big(\int \varphi(x)G(x)\,\mathrm{d}x - \int \varphi(x)G(x)x^2\,\mathrm{d}x\Big);\\ \beta &= -\int \varphi(x)G(x)x\,\mathrm{d}x;\\ \gamma &= \frac{1}{2}\Big(\int \varphi(x)G(x)x^2\,\mathrm{d}x - 3\int \varphi(x)G(x)\,\mathrm{d}x\Big).\end{aligned} \tag{1.42}$$

Recall that $c = \mathsf{E}K(\mathsf{Z}) = \frac{1}{\delta}\big(\mathsf{E}G(\mathsf{Z}) + \alpha\mathsf{E}\mathsf{Z}^2 + \beta\mathsf{E}\mathsf{Z} + \gamma\big)$. Now using (1.42) and the fact that $\mathsf{E}\,\mathsf{Z} = 0$ and $\mathsf{E}\,\mathsf{Z}^2 = 1$ (since Z has density f), we get $c = \frac{1}{\delta}\big(\mathsf{E}G(\mathsf{Z}) - \mathsf{E}G(\mathsf{N})\big)$. From the entropy approximation (1.39) and as the density f_0 has unit variance, we have $\hat{J}[\hat{f}_0] = \frac{1}{2}c^2$. Therefore,

$$\hat{J}[\hat{f}_0] = \frac{\big(\mathsf{E}G(\mathsf{Z}) - \mathsf{E}G(\mathsf{N})\big)^2}{2\,\delta^2}.$$

This completes the proof, with $C = 2\,\delta^2$ in Step 3. of Panel 1.5. The value of δ can be found by solving the additional constraint $\int \varphi(x)K(x)^2\,\mathrm{d}x = 1$. $\square$

This concludes our discussion of the approximations used in fastICA. We have shown that under certain conditions, the approximations given in Steps 2., 3. and 4. (Panel 1.5) are "close" to the true values. In the next section we give an example where these approximations are indeed close to one-another, but the surrogate density of the projections, f_0 from Step 1., is not close to the true density f.

1.4.1.3 Toy example: fastICA failing to find the optimum projection

We now highlight the approximation steps as explained in Panel 1.5 on a toy example. In this section we use example data as illustrated in Figure 1.4, which was intentionally created in a very simplistic manner to further emphasise the ease at which false optima are found using the contrast function $\hat{J}^*(y)$. The data was obtained by pre-selecting vertical columns where no data points are allowed. An iterative scheme was then employed, as explained in Panel 1.6.

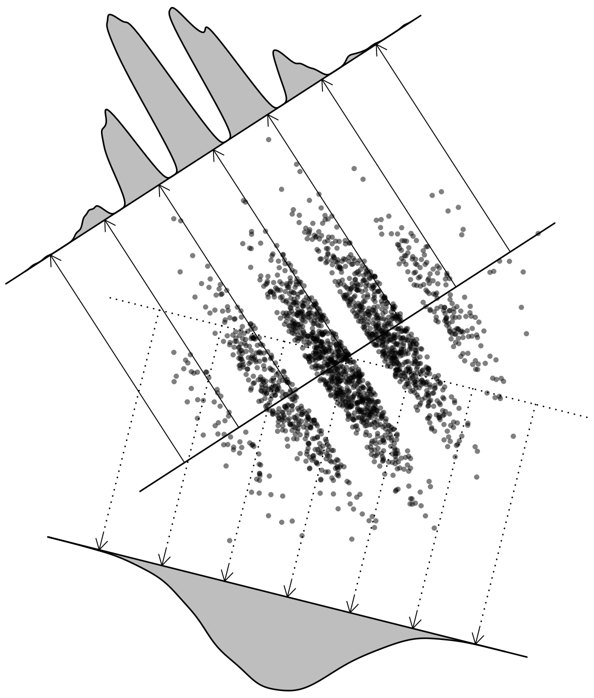


Figure 1.4: Scatter plot of original data with densities of the projected data in the direction obtained by M-spacing independent component analysis (solid line) and fastICA (dotted line). Kernel density estimation was used to obtain the marginal densities shown.

Panel 1.6: Toy example procedure

1. Sample n points from a standard two-dimensional Gaussian distribution;
2. Remove all points that lie in the pre-specified columns;
3. Whiten the remaining $\tilde{n}$ points;
4. Sample $n - \tilde{n}$ points from a standard two-dimensional Gaussian distri-

bution.

Repeat 2. - 4. until we have a sample of size n with no points lying in the pre-specified columns. No optimisation was done to the distribution of these points to attempt to force the fastICA contrast function to have a false optimum.

Referring back to Section 1.3.1.3, we use the M-spacing entropy approximation here to obtain a contrast function that can by compared to the fastICA contrast function (1.22). Following Learned-Miller & Fisher III (2003), we chose $M = \sqrt{n}$, where $n \in \mathbb{N}$ is the number of observations. This was chosen so that the condition $M/n \to 0$ as $n \to \infty$ is satisfied (Beirlant *et al.*, 1997; Vasicek, 1976). This approximation to entropy is a direct approximation to $H[f]$, and therefore does not involve an equivalent Step 1. (Panel 1.5) where f is substituted by a new density f_0.

Using the M-spacing method to find the first independent component, we want to obtain the direction $q^* := \operatorname{argmin}_{q \in \mathbb{R}^{r-1}\mathrm{P}} H_{M,n}(Yq)$. In this example, numerical optimisation is used to obtain q^* and the associated projection Yq^*. The contrast function to compare against the fastICA contrast function (1.22) is given by the M-spacing negentropy approximation, $J_{M,n}(z) = \eta(1) - H_{M,n}(z)$ for directions $q \in \mathbb{R}^{r-1}\mathrm{P}$. Note that $q^* = \operatorname{argmax}_{q \in \mathbb{R}^{r-1}\mathrm{P}} J_{M,n}(Yq)$.

To illustrate the kind of problems which can occur during the approximation from f to $\hat{f}_0$ and from $J[f]$ to $\hat{J}^*(y)$, we construct an example where the density f in the direction of maximum negentropy is significantly different to $\hat{f}_0$ in the same direction. This results in `fastICA` selecting a sub-optimal projection, as shown below. Here we just consider the case $I = 1$ in Assumption 1.4.1, with one $G = G_1$ and thus one $K = K_1$. Moreover, in `fastICA` there is a choice of two functions to use, $G(x) := (1/\alpha) \log \cosh(\alpha x)$, $\alpha \in [1, 2]$, and $G(x) := -\exp(-x^2/2)$.

We have considered these two functions with varying alpha, as well as the fourth moment contrast function given in Miettinen *et al.* (2015) and which we briefly discuss in Remark 1.8 below. In the example here all choices give very similar results and thus we only show the fastICA contrast function resulting from $G(x) = (1/\alpha) \log \cosh(\alpha x)$, with $\alpha = 1$, and the fourth moment contrast function, for simplicity.

Remark 1.8: Fourth moment contrast function in fastICA

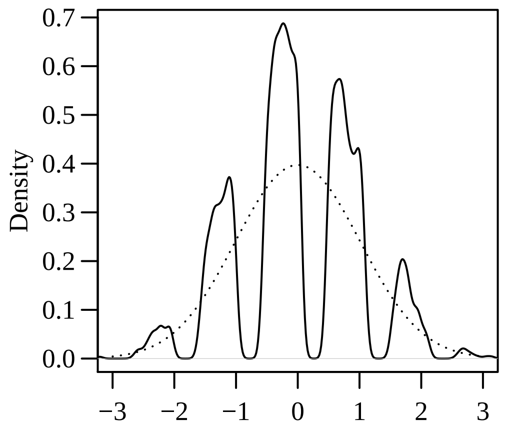
The contrast function used in Miettinen *et al.* (2015) is $|\mathsf{E}_f(q^\top \mathsf{Y})^4 - 3|$ where $\mathsf{Y} \in \mathbb{R}^r$ is some (standardised) random variable, $q \in \mathbb{R}^{r-1}$P and $q^\top \mathsf{Y} \sim f$. For a given (whitened) set of observations $Y = (y_1, \ldots, y_n)^\top \in \mathbb{R}^{n \times r}$ the empirical approximation for the expectation is used, such that the approximate contrast function is $\left|\frac{1}{n}\sum_{i=1}^n (q^\top y_i)^4 - 3\right|$.

Relating this back to Panel 1.5, the fourth moment contrast function used in Step 3. is $\hat{J}_{\text{fourth}}[f] = |\mathsf{E}_f \mathsf{Z}^4 - 3|$, where here a specific function G has been chosen to obtain this form. For Step 4. the approximate contrast function is $\hat{J}^*_{\text{fourth}}(z) = |\frac{1}{n}\sum_{i=1}^n z_i^4 - 3|$.

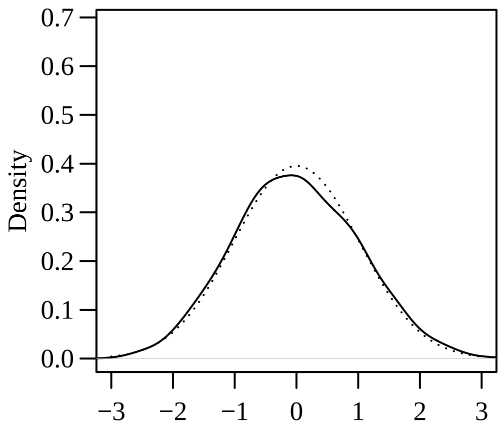
Figure 1.4 shows the distribution of one realisation obtained when we apply the procedure given in Panel 1.6. The projection of the data onto the solid line corresponds to the projection that maximises approximate negentropy found using the M-spacing method as described in Section 1.3.1.3. The dotted line gives the direction found when applying `fastICA` to the points. The two densities shown in Figure 1.4 are the sample densities of the projected data Yq onto the directions given by the solid and dotted arrows (which are approximations of the true densities f), corresponding to maximum M-spacing negentropy $J_{M,n}(Yq)$ and maximum `fastICA` contrast function $\hat{J}^*(Yq)$ respectively.

Figure 1.5 shows again the same sample densities as given in Figure 1.4, with the addition of the densities f_0 as described in Step 1. (Panel 1.5) onto the two directions found by maximising M-spacing negentropy and by `fastICA`. This highlights the large amount of information lost at Step 1. of the fastICA procedure, with f in the direction of maximum M-spacing negentropy showing clear peaks and troughs, whilst f_0 in the same direction is very similar to a standard Gaussian density, and does not correspond to the behaviour of the data in any meaningful way. Note here that the density $\hat{f}_0$ has not been shown. This is due to f_0 and $\hat{f}_0$ being very close to each other, and thus is omitted here for reasons of plot clarity.

With the data distributed as in Figure 1.4, the negentropy over projections in



(a) Densities onto the optimal direction found using the M-spacing method



(b) Densities onto the optimal direction found using the `fastICA` method

Figure 1.5: Plots showing the density f of the projected data (solid line), and the surrogate density f_0 used in the fastICA method (dotted line), for two different projections of the data. Sub-figure 1.5a corresponds to the direction of highest entropy, found using M-spacing, and Sub-figure 1.5b corresponds to the direction found by fastICA. These two directions are shown by the solid line and dotted line respectively in Figure 1.4.

the directions $w_\vartheta = (\sin(\vartheta), \cos(\vartheta))$, with $\vartheta \in [0, \pi)$, found by the M-spacing approximation and used in the `fastICA` method is shown in Figure 1.6. The contrast function obtained by approximating $J[f_0]$ directly is also included as the dashed line, and the contrast function $\hat{J}^*(y)$ given by the dotted line. The three contrast functions have been placed below one-another in the order of approximations given in Figure 1.3 and so the y-axis is independent for each. The equivalent approximate fourth moment contrast function as in Miettinen *et al.* (2015) is shown in Figure 1.7 (dotted line), given below the M-spacing approximation which is the same function as in Figure 1.6. The search is only performed on the half unit circle, as projections in directions $w_1 = (\sin(\vartheta), \cos(\vartheta))$ and $w_2 = (\sin(\vartheta + \pi), \cos(\vartheta + \pi))$ for any $\vartheta \in [0, \pi)$ have a reflected density with the same entropy. It is clear from Figure 1.6 that the `fastICA` result $\hat{J}^*$ is poor, with the `fastICA` contrast function missing the peak of negentropy that appears when using M-spacing. The contrast function used in the `fastICA` method clearly differentiates between the direction of the maximum and other directions, and thus in this example it is both confident and wrong (since there is a clear and unique peak). This is also true of the direct approximation to $J[f_0]$, showing that issues occur at the first step of approximations, when $J[f_0]$ is used instead of $J[f]$. This is further highlighted by the same issue occurring when the fourth moment contrast function is used (as shown in Figure 1.7), where the same incorrect direction is (confidently) found.

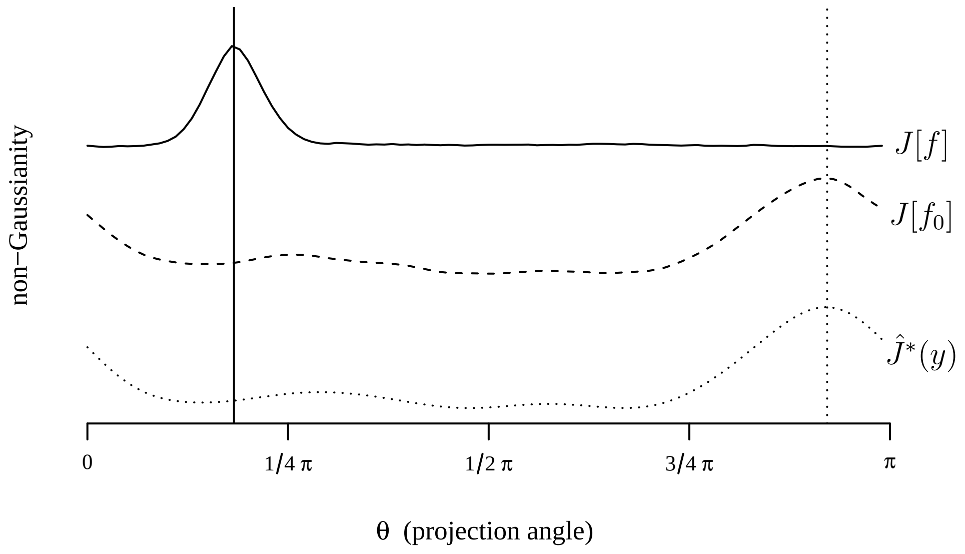


Figure 1.6: Objective functions of M-spacing (solid line), $J[f_0]$ (dashed line) and `fastICA` method (dotted line) for projections of the data given in Figure 1.4 in the directions $\vartheta \in [0, \pi)$. These correspond to $J[f]$, $J[f_0]$ and $\hat{J}^*(y)$ in Figure 1.3. The vertical lines give the directions which maximise the contrast functions for M-spacing (solid line) and `fastICA` (dotted line).

As is shown in Theorem 1.4.7, for sufficiently small c, the approximation for the density $\hat{f}_0$ (given in (1.20)) is "close to" f_0 (given in (1.19)), and the speed of convergence is of order $\|c\|^2$ for $c \to 0$. Therefore, it is our belief (backed up by computational experiments) that the majority of the loss of accuracy occurs in the approximation step where the surrogate f_0 is used instead of f, rather than in the later estimation steps for $J[\hat{f}_0]$ and $\hat{J}^*(y)$. This can be seen by comparing numerically the contrast functions $J[f]$, $J[f_0]$ and $\hat{J}^*(y)$ (shown in Figure 1.6), and by

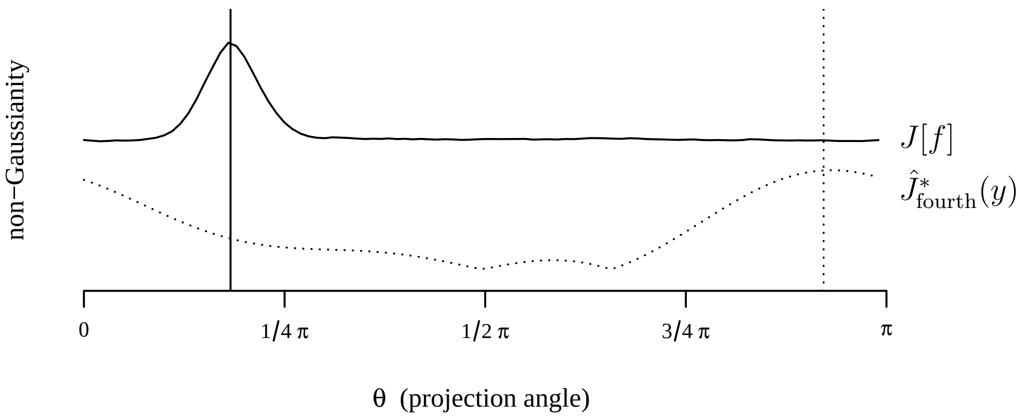


Figure 1.7: Objective functions of M-spacing (solid line), and the fastICA fourth moment method described in Remark 1.8 (dotted line) for projections of the data given in Figure 1.4 in the directions $\vartheta \in [0, \pi)$. The vertical lines give the directions which maximise the contrast functions for M-spacing (solid line) and `fastICA` (dotted line). This figure is comparable to Figure 1.7 and is presented here to strengthen the argument regarding the step in Panel 1.5 where the main errors are introduced in the fastICA method. Clearly, using the fourth moment contrast function still results in the incorrect direction being chosen.

comparing the densities f, f_0 and $\hat{f}_0$. Here, $J[f_0]$ and $\hat{J}^*(y)$ give similar directions for the maximum, and these differ significantly from the location of the maximum of $J[f]$. This is a fundamental theoretical problem with the fastICA method, and is not a result of computational or implementation issues with `fastICA`. In particular, the fact that the dotted vertical line in Figure 1.6 is at the maximum of $\hat{J}^*(y)$ indicates that the effect is not a convergence problem in the `fastICA` implementation. Finally, comparing $\hat{J}^*(z)$ in Figure 1.6 with $\hat{J}^*_{\text{fourth}}(z)$ in Figure 1.7 highlights that the issue is more fundamental than the choice of the function G in Step 3. of Panel 1.5, as both of these approximate contrast function give similar, wrong results.

1.4.1.4 Concluding remarks on fastICA.

In this section we have given an example where the fastICA method misses structure in the data that is obvious to the naked eye. Since this example is very simple, the fastICA result is concerning, and this concern is magnified when working in high dimensions as visual inspection is no longer easy. There is clearly some issue with the contrast function (surrogate negentropy) used in fastICA. Indeed, this surrogate has the property of being an approximation of a *lower bound* for negentropy, and this does not necessarily capture the actual behaviour of negentropy over varying projections, since we want to *maximise negentropy.* To strengthen the claim that accuracy is lost when substituting the density with the surrogate, we have shown convergence results for all the approximation steps used in the method.

To conclude this section, we ask the following questions which could make for interesting future work: *Is there a way, a priori, to know whether fastICA will work?* This is especially pertinent when fastICA is used with high dimensional data. The trade-off in accuracy for the fastICA method comes at the point where the density f is substituted with f_0. Therefore one could also ask: *Are there other methods similar to that of fastICA but that use a different surrogate density which more closely reflects the true projection density?*

If these two options are not possible, then potentially a completely different method for “fast” independent component analysis is needed, one that either gives a “good” approximation for all distributions, or where it is known when it breaks down. An initial step in this direction is introduced in the Section 1.4.2. In this next section we propose a new independent component analysis method, known as clusterICA, that uses the M-spacing approximation for entropy (introduced above), combined with a clustering procedure.

1.4.2 ClusterICA

In this section we introduce a novel algorithm for implementing independent component analysis, that uses entropy as the function to optimise in a way that allows the user to sensibly balance the trade-off between computational cost and accuracy. For each independent component, this algorithm — called 'clusterICA' — uses random directions on the projective space to initiate, then clusters a subset of these before performing optimisation on a single projection from each of these clusters. The clustering step allows only materially different directions to be examined, thus decreasing computational time and the likelihood of getting stuck in a local minima of the entropy function. The projection associated with the best direction, calculated using M-spacing approximation of differential entropy, is then output as the independent component. Additional components are found in the same way, with rotations of the search-space used to ensure orthogonality of the matrix $Q \in \mathbb{R}^{r \times r}$, which transforms the whitened matrix $Y \in \mathbb{R}^{n \times r}$. As discussed in Section 1.3.1, the unmixing matrix $W \in \mathbb{R}^{p \times r}$, which transforms the original matrix $X \in \mathbb{R}^{n \times p}$, (given in Step 4. of Panel 1.4) is not required to be orthogonal.

This section is structured as follows. Initially we introduce the clusterICA algorithm and detail the mathematical tools used in the algorithm. This is split into two main parts, arranged such that the more novel work is described first. In particular, we describe clustering in projective spaces (which here is the set of points on the r-dimensional half-sphere) after introducing a specific distance metric on the space; then, we give a description of the rotation of the points using the Householder reflection method to ensure orthogonality of the matrix $Q \in \mathbb{R}^{r \times r}$, and to allow a smaller search space for optimisation. After the constituent parts have been introduced, we give the full clusterICA algorithm. All relevant associated pseudo-code is included throughout this section, and we conclude with a tip on how to merge the fastICA method (described in Section 1.4.1) with the clusterICA method, arguing that this obtains advantages from both. After we introduce the method, an example is given to highlight where clustering prior to optimisation increases the computational speed of clusterICA.

In Section 1.4.3, clusterICA is included in an example comparing it to established methods on various two-dimensional samples, similar to those used in Learned-Miller & Fisher III (2003), using a standard comparison metric introduced in Amari *et al.* (1996) that highlights differences in the unmixing matrices.

The methods are also compared based on minimising the “true” entropy of the projections.

1.4.2.1 The clusterICA method.

Here we introduce the clusterICA method, which is a novel independent component analysis method that utilises clustering in projective spaces. The main idea behind this method is to cluster together different (initial random) directions in order to reduce the number of optimisations and, at the same time, ensure that enough directions have been investigated. Before we examine the details of the method, the main steps are listed in Panel 1.7. As discussed in Section 1.3.1.2, the first step of any independent component analysis method is to whiten the original data. Assuming the data is whitened, the following Steps 1. – 5. given in Panel 1.7 are performed to find a single independent component direction, and then repeated to obtain the desired number of orthogonal directions.

Panel 1.7: ClusterICA: condensed overview

1. Choose N (initial random) directions;
2. Select $\tilde{N} < N$ best directions;
3. Separate the best $\tilde{N}$ directions into $K \ll \tilde{N}$ clusters;
4. Optimise the best direction of each cluster (K optimisations in total);
5. Choose the best optimised direction.

When implementing the steps above, several issues require mathematical clarification. Two important questions which have not yet been addressed in this chapter will be expanded on below. A delicate question relates to how clustering is actually implemented. The idea is to cluster together similar directions, so the answer will in turn depend on what is meant by ‘similar directions’. To define this concept the problem is set in a projective space and a suitable metric on this space is introduced. A second issue is hidden in the definition of independent component directions, in particular in the fact that each direction that the whitened data is projected onto must be orthogonal, so when repeating the procedure to find the next direction orthogonality with all previous loadings must be ensured. This is achieved via Householder transformations, which are explained in full below. The full algorithm for clusterICA will be given after the above points have been addressed.

1.4.2.2 Clustering

We commence the mathematical description of clusterICA by explaining in detail the main novelty of this algorithm, which is projective clustering. First the process of clustering in Euclidean spaces is described. In this section the notation $d(u, v)$, $u, v \in \mathbb{R}^r$ is used to mean the standard Euclidean metric. The method and the pseudo-code can then be applied to the projective space setting by simply changing the Euclidean metric with a suitable metric d. Setting the problem in projective spaces is the tool that allows similar directions to be identified, or in other words that allows directions which are materially different to be differentiated, thus allowing effective clustering of directions.

Clustering in Euclidean spaces has been done in the literature, see for example Mardia *et al.* (1979). A specific method in the Euclidean setting, called k-means clustering (Lloyd, 1982) is now reviewed. The k-means clustering method splits data into K clusters in the following way. The data is first initialised into K clusters, the centroid of each of these clusters is found and then each point is assigned to the respective cluster depending on which centroid is closest. Once the new clusters have been assigned, the new centroid is calculated in each cluster and the process of assigning points to clusters is repeated with the new centroids. In Algorithm 1.2, k-means clustering pseudo-code is given, where the number of points is given as $\tilde{N}$ to be consistent with Panel 1.7. Note that in the Euclidean case finding the centroid reduces to calculating the mean of the elements in the cluster, which involves summing vectors in the Euclidean space. This however is more complicated in projective spaces, as we will see later, because 'summing directions' in projective spaces is not a well-defined action.

Algorithm 1.2: k-means.

```
input: data $u_1, u_2, \dots, u_{\tilde{N}}$,
        initial clustering $c^{(0)} = (c_1^{(0)}, c_2^{(0)}, \dots, c_{\tilde{N}}^{(0)})$
        number of clusters $K$
output: vector, length $\tilde{N}$
begin
    for $t$ in $1, 2, 3, \dots$
        % compute centroids: for $i = 1, \dots, K$, let
              $m_i^{(t)} \leftarrow$ sample mean of $\{u_j | c_j^{(t-1)} = i\}$
        % assign points to new clusters: for $j = 1, \dots, \tilde{N}$, let
              $c_j^{(t)} \leftarrow \underset{i=1,\dots,K}{\operatorname{argmin}}\, d(u_j, m_i^{(t)})$
        if $c^{(t)} = c^{(t-1)}$, stop
    end for
    return $c^{(t)} = (c_1^{(t)}, c_2^{(t)}, \dots, c_{\tilde{N}}^{(t)})$
end
```

We initialise the k-means method using the k-means++ method, introduced by Arthur & Vassilvitskii (2007). This gives K initial clusters for data $u_1, u_2, \dots, u_{\tilde{N}} \in \mathbb{R}^r$, $K \ll \tilde{N}$, in the following way. In the first step, a point u_i, $i \in \{1, 2, \dots, \tilde{N}\}$ is chosen as a cluster centroid at random. The next step involves picking another cluster centroid u_s at random with probabilities proportional to the distance $\mathrm{d}(u_j, u_i)$ for all $j = 1, 2, \dots, \tilde{N}$. For each subsequent step, a centroid is chosen from the set of remaining points, with probability of picking each point proportional to the minimum distance from all centroids currently chosen. This step is repeated until K centroids (and thus K clusters) have been picked. The output of the algorithm is a set of integers $c = (c_1, c_2, \dots, c_{\tilde{N}})$, with $c_i \in \{1, 2, \dots, K\}$ being the cluster that the point u_i belongs to. We give pseudo-code for k-means++ in Algorithm 1.3.

Algorithm 1.3: k-means++.

```
input:  data, $u_1, u_2, \dots, u_{\tilde{N}}$;
        number of clusters, $K$
output: vector, length $\tilde{N}$
begin
    sample $s$ uniformly from $1, 2, \dots, \tilde{N}$
    $m_1 \leftarrow u_s$
    for $i = 2, \dots, K$
        % compute minimum distance from selected centroids:
        for $j = 1, \dots, \tilde{N}$; $d_j \leftarrow \min_{k=1,\dots,i-1} d(u_j, m_k)$
        sample $s$ from $1, \dots, \tilde{N}$ with probability proportional to $d_1, d_2, \dots, d_{\tilde{N}}$
```

```
            $m_i \leftarrow u_s$
        end for
        % assign points to clusters:
        for $i = 1, \ldots, \tilde{N}$; $c_i \leftarrow \underset{j=1,\ldots,K}{\operatorname{argmin}}\ d(u_i, m_j)$
        return $c^{(0)} = (c_1^{(0)}, c_2^{(0)}, \ldots, c_{\tilde{N}}^{(0)})$
    end
```

Finally, note that the k-means clustering method can also be performed without *a priori* specifying the number of clusters K, but instead using hierarchical divisive clustering. This method is here called divisive k-means. In this method, k-means clustering is run on a loop, splitting a specified cluster into two using the k-means method with $K = 2$ (and k-means++ initialisation) until some tolerance is reached. The tolerance is given as a relative change in total root mean-squared error (RMSE) between successive loops. After loop $t - 1$, the divisive k-means method will have found t clusters, and then the root mean-squared error is given by,

$$\mathrm{RMSE}(U, c) = \left(\frac{1}{t} \sum_{i=1}^{t} d(u_i, \bar{u}^{(c_i)})^2 \right)^{1/2},$$

where $\bar{u}^{(c_i)}$ is the mean of all points u_k belonging to cluster c_i. At each step the cluster chosen to be split into two is that which has the largest within-cluster root mean-squared error. The procedure stops if the difference in root mean-squared error between successive steps is smaller than a specified value. The pseudo-code for divisive k-means is given in Algorithm 1.4. This is the method used in clusterICA, but set in projective spaces which is now introduced.

Algorithm 1.4: Divisive k-means.

```
required:  k-means++, Algorithm 1.3;
           k-means, Algorithm 1.2
input:     data, $u_1, u_2, \ldots, u_{\tilde{N}}$;
           tolerance, tol
output:    vector, length $\tilde{N}$
begin
    % calculate centroid of all points and the associated sum-of-squares
    $m \leftarrow$ sample mean of $\{u_1, \ldots, u_{\tilde{N}}\}$
    $T \leftarrow$ total RMSE between all $u_i$ and $\bar{u}$
    $t_1^{(0)} \leftarrow T$ % within-cluster RMSE for current clustering c
    $t^{(0)} \leftarrow T$ % total RMSE for current clustering c
    %%
    % initialise loop
    %%
    % all points currently in a single cluster
    $c \leftarrow (1, \dots, 1) \in \mathbb{R}^{\tilde{N}}$
    for $i = 1, 2, 3, \dots$
        % select cluster number with the largest within-cluster RMSE
        $\sigma \leftarrow \operatorname{argmin}_{j=1,\dots,i} t_j^{(i-1)}$
        % select the points that belong to cluster $\sigma$
        $v \leftarrow \{u_j : c_j = \sigma\}$
        % apply k-means (with k-means++ initialisation to v
        $c^{(0)} \leftarrow$ k-means++: data, $v$;
                                number of clusters $K = 2$
        $c^{(v)} \leftarrow$ k-means: data, $v$;
                              number of clusters $K = 2$;
                              initial clustering, $c^{(0)}$

        % update clustering
        % take into account new clusterings of points $v = \{u_j : c_j = \sigma\}$
        $c \leftarrow$ merge: $\{c: c_j \neq \sigma\}$ with $c^{(v)}$

        % recalculate RMSE:
        % within-cluster RMSE: $m_j$ is the centroid of cluster $j$
        for $j = 1, \dots, i+1$; $t_j^{(i)} \leftarrow \mathrm{RMSE}(\{u_i | c_i = j\}, m_j)$
        % total RMSE
        $t^{(i)} = \sum_{j=1}^{i+1} t_j^{(i)}$

        % calculate relative change to total RMSE
        % used to determine whether another split is computationally worthwhile
        $\Delta \leftarrow (t^{(i)} - t^{(i-1)}) / T$
        if $\Delta <$ tol: break
    end if
    return c
end
```

Given a set of directions (*i.e.* elements in the projective space), the aim is to apply divisive k-means to cluster this set into 'similar' directions. As has been shown above in Algorithms 1.2, 1.3 and 1.4, this requires a metric $d(u, v)$ defined for any two points on the projective space, $u, v, \in \mathbb{R}^{r-1}\mathrm{P}$. Moreover the algorithms require the computation of the centroid of a subset of directions (i.e. the mean in the Euclidean case). Both issues will be investigated below.

First a metric on the projective space needs to be defined, which should represent the 'distance' between the two directions identified by u and v. For any $u, v \in \mathbb{R}^{r-1}\mathrm{P}$ the metric $d_p \colon \mathbb{R}^{r-1}\mathrm{P} \times \mathbb{R}^{r-1}\mathrm{P} \to \mathbb{R}$ is defined by,

$$d_p(u, v) := \sqrt{1 - (u^\top v)^2}. \tag{1.43}$$

It can be shown that d_p is indeed a metric on $\mathbb{R}^{r-1}\mathrm{P}$. The choice of this metric is informally motivated by the reasoning given in Remark 1.9.

Remark 1.9: Motivation of choice of projective metric

For given directions $u, v \in \mathbb{R}^{r-1}\mathrm{P}$, let $\vartheta \in [0, \pi]$ be the smallest angle between them. If $\vartheta \leq \pi/2$ then ϑ can be used to describe the 'distance' between two equivalence classes $u, v \in \mathbb{R}^{r-1}\mathrm{P}$. However if the vectors are (nearly) opposite of each other (i.e. $\vartheta \approx \pi$) then ϑ is not suitable to measure the distance between directions, because we required the distance between opposite directions to be zero. A good measure instead is $|\sin \vartheta|$ which is zero at 0, π and 2π and can be defined for all $\vartheta \in [0, 2\pi]$. Recall now that $u^\top u = v^\top v = 1$, and thus by the Euclidean dot product formula, $u^\top v = \cos \vartheta \cdot \|u\| \cdot \|v\| = \cos \vartheta$, (as $\|u\| = \|v\| = 1$ since these vectors lie on S^{r-1}). Therefore $|\sin \vartheta| = \sqrt{1 - (u^\top v)^2} \in [0, 1]$, which is the distance between u and v given by (1.43).

Secondly we examine the issue of calculating the centroid of a set of directions. In Euclidean spaces, the standard formula to find the centroid of a set of points $u_1, u_2, \ldots, u_{\tilde{N}} \in \mathbb{R}^r$ is given by the mean $\bar{u} = (1/\tilde{N}) \sum_{i=1}^{\tilde{N}} u_i$. Clearly this formula cannot be used in projective spaces because directions cannot be "summed" together. Instead, it is easy to check that the mean $\bar{u}$ is the point where a specific function f attains its minimum, with f defined by

$$f(v) := \sum_{i=1}^{\tilde{N}} |u_i - v|^2.$$

The important feature of this characterisation of the mean is that f can be defined also for $v \in \mathbb{R}^{r-1}\mathrm{P}$ simply by replacing the Euclidean metric $|\cdot|$ with the metric d_p given by (1.43). This is done in Lemma 1.4.10 below, where in addition the minimum of f is calculated. The lemma motivates the definition of the centroid of a set of directions in projective spaces, given in Definition 1.4.11 below.

Lemma 1.4.10. *Let $u_1, u_2, \cdots, u_{\tilde{N}} \in \mathbb{R}^{r-1}P$. Let the function $f \colon \mathbb{R}^{r-1}P \to \mathbb{R}$ be defined by*

$$f(v) = \sum_{i=1}^{\tilde{N}} d_p(u_i, v)^2, \tag{1.44}$$

where $d_p(u, v)$ is the distance metric given in (1.43). Set $U = (u_1, u_2, \cdots, u_{\tilde{N}})^\top \in \mathbb{R}^{\tilde{N} \times r}$. Then

$$w := \underset{\|v\|=1}{\operatorname{argmax}} \{v^\top U^\top U v\}.$$

is a minimum of f.

Proof. The function f can be written,

$$f(w) = \sum_{i=1}^{\tilde{N}} \left(1 - (u_i^\top v)^2\right) = \tilde{N} - \sum_{i=1}^{\tilde{N}} v^\top u_i \, u_i^\top v,$$

for all $v \in \mathbb{R}^{r-1}\mathrm{P}$. Now, note that

$$\sum_{i=1}^{\tilde{N}} u_i \, u_i^\top = \begin{pmatrix} \sum_i u_{i1}u_{i1} & \sum_i u_{i1}u_{i2} & \cdots & \sum_i u_{i1}u_{i\tilde{N}} \\ \sum_i u_{i2}u_{i1} & \sum_i u_{i2}u_{i2} & \cdots & \sum_i u_{i2}u_{i\tilde{N}} \\ \vdots & \vdots & & \vdots \\ \sum_i u_{i\tilde{N}}u_{i1} & \sum_i u_{i\tilde{N}}u_{i2} & \cdots & \sum_i u_{i\tilde{N}}u_{i\tilde{N}} \end{pmatrix} = U^T U.$$

Therefore,

$$\begin{aligned} f(v) &= \tilde{N} - v^\top U^\top U v \\ &\geq \tilde{N} - w^\top U^\top U w = f(w), \end{aligned}$$

since $w^\top U^\top U w \geq v^\top U^\top U v$ for any $v \in \mathbb{R}^{r-1}\mathrm{P}$ (with $\|v\| = 1$) by definition of z, as required. □

From Lemma 1.4.10 and noting that the vector w that minimises the function f as in (1.44) is the Fréchet mean, Definition 1.4.11 follows.

Definition 1.4.11. *The centroid of a set of directions $u_1, u_2, \ldots, u_{\tilde{N}} \in \mathbb{R}^{p-1}P$ is denoted by $\bar{u}$, or mean$(u_1, u_2, \ldots, u_{\tilde{N}})$, and is given by*

$$\bar{u} = \underset{\|w\|=1}{\operatorname{argmax}}\{v^\top U^\top U v\},$$

where $U = (u_1, u_2, \cdots, u_{\tilde{N}})^\top$.

Notice that the mean $\bar{u}$ in projective spaces is effectively given by the eigenvector associated with the largest eigenvalue of the matrix $U^\top U$. This can be found as the first singular vector of U using singular value decomposition. From this, the function f as in (1.44) with argument $\bar{u}$ can be written as,

$$\begin{aligned} f(\bar{u}) &= \sum_{i=1}^{\tilde{N}} d_p(u_i, \bar{u})^2 \\ &= \sum_{i=1}^{\tilde{N}} (1 - (u_i^\top \bar{u})^2) \\ &= \tilde{N} - \bar{u}^\top U^\top U \bar{u} \\ &= \tilde{N} - \lambda_{\max}, \end{aligned}$$

as $\bar{u}^\top \bar{u} = 1$, where $\lambda_{\max}$ is the largest eigenvalue of $U^\top U$. This is the sum of the distance between all points and the centroid.

1.4.2.3 Using rotations to ensure orthogonality of subsequent loadings

Here, it is explained how we ensure orthonormality of the directions in which the whitened data is projected. A method is introduced that is in general more computationally stable than the popular Gram-Schmidt process and allows sequentially smaller search spaces to be employed in the optimisation stage, hence reducing computational time.

Suppose l independent components from data $X \in \mathbb{R}^{n\times p}$ are desired, and $Y \in \mathbb{R}^{n\times r}$ is the whitened version of X, $r = \text{rank}(\tilde{X})$ (following notation as in Remark 1.1). Since finding the loadings in clusterICA is a sequential procedure (as described in Step 3. of Panel 1.4), we assume here that $k-1$ previous directions $q_1, q_2, \ldots, q_{k-1} \in \mathbb{R}^{r-1}\text{P}$ have been found, which are orthonormal and such that the loadings Yq_i, $i = 1, \ldots, k-1$ (sequentially) minimise entropy. In particular the directions q_j, $j = 1, \ldots, k-1$, are such that the data projected onto each q_j minimises entropy in the space $V_j = \{q \in \mathbb{R}^{p-1}\text{P} \,|\, q \perp q_i,\ i = 1, 2, \ldots, j-1\}$.

Aim: To find the kth direction, $q_k \in \mathbb{R}^{r-1}\text{P}$ (associated with the kth independent component Yq_k).

Requirement: To explore the subspace of directions in $\mathbb{R}^{r-1}\text{P}$ orthonormal to all previous $k-1$ directions, $q_1, \ldots, q_{k-1}$.

Main idea of this section: Use rotation matrices to ease the search of the subspace of directions in $\mathbb{R}^{r-1}\text{P}$ orthonormal to all previous directions. Firstly, the idea and method to help obtain the direction q_k is presented, and then the inductive method to obtain the specific rotation matrix is described.

Set $R^{(0)} = I_r \in \mathbb{R}^{r\times r}$, and assume that the rotation matrices $R^{(j)}$, $j = 1, 2, \ldots, k-1$, are available. Here, each matrix $R^{(j)}$ rotates the space such that $R^{(j)}q_i = e_i^{(r)}$, for $j = 1, 2, \ldots, k-1$, and $i = 1, 2, \ldots, j$, where $e_i^{(r)} \in \mathbb{R}^r$ is the r dimensional vector with a 1 in the ith element and zeros elsewhere. These rotations $R^{(j)}$ are effectively aligning the axis so that the first j dimensions of the space are the directions associated with the first j independent component loadings. The aim here is to find the kth optimal direction $q_k \in \mathbb{R}^{p-1}\text{P}$ and the kth rotation matrix $R^{(k)}$.

After the rotation $R^{(k-1)}$ is performed, ensuring that q_k is orthogonal to all previous directions is equivalent to asking that q_k is of the form $q_k = R^{(k-1)}\tilde{v}$, with $\tilde{v} = (0, 0, \dots, 0, v)$, where the first $k-1$ components of $\tilde{v}$ are zero and $v \in \mathbb{R}^{r-k}\mathrm{P}$.

That is, with $m := r - k + 1$, we explore the space $\{0\} \times \cdots \times \{0\} \times \mathbb{R}^{m-1}\mathrm{P}$ (where the first $k-1$ dimensions only include the zero element). The direction $v_m \in \mathbb{R}^{m-1}\mathrm{P}$ is found, such that the projection of the data Y onto $R^{(k-1)}\tilde{v}_m$, $\tilde{v}_m = (0, \cdots, 0, v_m)^\top \in \mathbb{R}^{r-1}\mathrm{P}$, minimises entropy. Therefore, at each step the dimension of the search space is effectively reduced by one.

In Panel 1.8 below we describe how the rotation matrix is updated to obtain $R^{(k)}$.

Panel 1.8: Independent component rotation matrix

Suppose that we have found the optimal direction $q_k = R^{(k-1)}\tilde{v}_m$ and that the previous rotation matrix $R^{(k-1)}$ is available. In this panel we describe how the kth rotation matrix is calculated, which has the property $R^{(k)}q_j = e_j^{(r)}$, $j = 1, 2, \dots, k$. It turns out that the matrix $R^{(k)}$ is an update of the previous rotation matrix $R^{(k-1)}$, more precisely

$$R^{(k)} = R^{(k-1)} \begin{pmatrix} I_{k-1} & 0 \\ 0 & P \end{pmatrix}.$$

Here, $P \in \mathbb{R}^{m \times m}$ is known as the Householder matrix (Householder, 1958) and it is defined as $P = I_m - 2v\,v^\top$ where $v = \frac{u}{(u^\top u)}$ and $u = v_m - e_1^{(m)}$. The Householder matrix P as defined above reflects the vector v_m onto $e_1^{(m)}$, *i.e.* through the hyperplane that bisects v_m and $e_1^{(m)}$ and goes through the origin. That is, $Pv_m = e_1^{(m)}$ (using the fact that $v_m^\top v_m = 1$) and as P is symmetric (which is clear from the definition), $Pe_1^{(m)} = v_m$. By Lemma 1.4.12 (given below), $R^{(k)}q_j = e_j^{(r)}$ for all $j < k$, and also $R^{(k)}q_k = e_k^{(r)}$, as required. Pseudo-code for calculating the rotation matrix $R^{(k)}$ is given in Algorithm 1.5.

The procedure explained in Panel 1.8 is performed for $k = 1, 2, \dots, l$. Once l steps have been completed the result is an orthogonal matrix $R^{(l)} \in \mathbb{R}^{r \times r}$, of which the first l columns are kept to give the set $\{q_1, q_2, \dots, q_l\}$, where each $q_i \in \mathbb{R}^{r-1}\mathrm{P}$ is the ith column of $R^{(l)}$. The l independent components are then found by projecting the whitened data onto each direction $q_1, \ldots, q_l$. The l directions are given column-wise in the (column orthonormal) matrix $Q = (q_1\, q_2 \cdots q_l) \in \mathbb{R}^{r\times l}$, and the independent component directions are given by the columns of $AQ \in \mathbb{R}^{p\times l}$, where $A \in \mathbb{R}^{p\times r}$ is the whitening matrix as introduced in Step 2. (Panel 1.4) and discuss in Section 1.3.1.2.

Algorithm 1.5: Householder transformation.

```
input:  vector $v_m \in \mathbb{R}^{r-k}$P;
        rotation matrix $R^{(k-1)} = (r_1^{(k-1)}\, r_2^{(k-1)} \cdots r_p^{(k-1)}) \in \mathbb{R}^{r\times r}$
output: matrix $R^{(k)} \in \mathbb{R}^{r\times r}$
begin
        % calculate Householder matrix $P \in \mathbb{R}^{m\times m}$, with $m = r - k + 1$
        $u \leftarrow v_m - \mathbb{1}_m$, where $\mathbb{1}_m = (1, 0, \ldots, 0) \in \mathbb{R}^m$
        $v \leftarrow u \,/\, (u^\top u)$
        $P \leftarrow \mathcal{I}_m - 2v\,v^\top \in \mathbb{R}^{m\times m}$
        % calculate new rotation matrix $R^{(k)}$
        % matrix 'P' only acts on last m columns of $R^{(k-1)}$
        $\tilde{R} \leftarrow (r_k^{(k-1)}\, r_{k+1}^{(k-1)} \cdots r_r^{(k-1)}) \in \mathbb{R}^{r\times r}$
        $R \leftarrow \tilde{R}\,P$
        $R^{(k)} \leftarrow (r_1^{(k-1)}\, r_2^{(k-1)} \cdots r_{k-1}^{(k-1)}\, R) \in \mathbb{R}^{r\times r}$
        return $R^{(k)}$
end
```

Lemma 1.4.12. *With $R^{(k)} \in \mathbb{R}^{r\times r}$ and vectors $q_i \in \mathbb{R}^{r-1}P$, $i = 1, 2, \ldots, k$ as defined above,*

$$R^{(k)} q_j = e_j^{(r)},$$

for $j \in \{1, 2, \ldots, k\}$.

Proof. Suppose $j \in \{1, \ldots, k-1\}$. Here,

$$\begin{aligned}
R^{(k)} q_j &= R^{(k-1)} \begin{pmatrix} I_{k-1} & 0 \\ 0 & P \end{pmatrix} q_j \\
&= \begin{pmatrix} I_{k-1} & 0 \\ 0 & P \end{pmatrix} R^{(k-1)} q_j, \quad \text{as both matrices are symmetric} \\
&= \begin{pmatrix} I_{k-1} & 0 \\ 0 & P \end{pmatrix} e_j^{(r)}, \quad \text{by the assumption on } R^{(j)},\ j \in \{1, \ldots, k-1\} \\
&= e_j^{(p)}.
\end{aligned}$$

Now, for $j = k$, it is required to show $R^{(k)} q_k = e_k^{(r)}$. Here, with $\tilde{v}_k = (0, \ldots, 0, v_k)^\top$,

$$\begin{aligned}
R^{(k)} q_k &= R^{(k-1)} \begin{pmatrix} I_{k-1} & 0 \\ 0 & P \end{pmatrix} q_k \\
&= \begin{pmatrix} I_{k-1} & 0 \\ 0 & P \end{pmatrix} R^{(k-1)} q_k, \quad \text{as both matrices are symmetric} \\
&= \begin{pmatrix} I_{k-1} & 0 \\ 0 & P \end{pmatrix} \tilde{v}_m, \quad \text{as } R^{(k-1)} \text{ is orthogonal} \\
&= e_k^{(r)}, \quad \text{as } P v_m = e_1^{(m)}.
\end{aligned}$$

Thus $R^{(k)} q_j = e_j^{(r)}$ for all $j \in \{1, 2, \ldots, k\}$ as required. □

1.4.2.4 ClusterICA: The full algorithm

We now describe the full clusterICA algorithm, with details regarding clustering in projective spaces, entropy estimation and ensuring orthogonality of the directions $q_1, \ldots, q_l$. The aim is to perform independent component analysis on the (unwhitened) data $X \in \mathbb{R}^{n \times p}$, such that l independent components are found.

The data X must first be whitened to give matrix $Y \in \mathbb{R}^{n\times r}$, where $r = \text{rank}(\tilde{X}) \leq \min\{n-1, p\}$, with $\tilde{X}$ the column-centred version of X. Note that in practice, the value of r can be specified by the user, although the discussion in Remark 1.3 is pertinent here as if r is significantly smaller than $\text{rank}(\tilde{X})$, a lot of potentially interesting information (in the non-Gaussian sense) may be lost.

The clusterICA algorithm to obtain the k^{th} independent component loading follows the same 5 steps indicated in Panel 1.7, which are repeated in Panel 1.9 with extra details. For the steps below, suppose that the first $k-1$ directions $\{q_1, q_2, \ldots, q_{k-1}\}$ and rotation matrix $R^{(k-1)}$ are known. The pseudo-code is given in Algorithm 1.6.

Panel 1.9: ClusterICA: detailed overview

1. Initialise N random vectors on $\mathbb{R}^{r-k}\mathrm{P}$, prepend them with $k-1$ zeros and pre-multiply with the rotation matrix $R^{(k-1)}$ to give N random directions in $\mathbb{R}^{r-1}\mathrm{P}$.
2. Project data Y onto the $\mathbb{R}^{r-1}\mathrm{P}$ random directions and calculate entropy using M-spacing (Algorithm 1.1). Sort these directions in increasing order of entropy of the projected data onto these directions and keep the $\tilde{N}$ best directions, *i.e.* the $\tilde{N}$ directions where the projected data have the lowest entropy.
3. Using the divisive clustering function (Algorithm 1.4), put the $\tilde{N}$ best directions into K clusters, where $K \ll \tilde{N}$.
4. Do K optimisations of the directions, one for each cluster, initialising with the best directions (in terms of lowest entropy) in each cluster. The optimisation is done in the $\mathbb{R}^{r-k}\mathrm{P}$ space, with the directions then prepended by zeros and pre-multiplied by $R^{(k-1)}$ such that they belong to $\mathbb{R}^{r-1}\mathrm{P}$ before calculating the M-spacing entropy value.
5. Keep the best direction post-optimisation, denoted by $q_k = R^{(k-1)}\tilde{v}_k$ for step k, where $\tilde{v}_k$ is the optimal $\mathbb{R}^{r-k}\mathrm{P}$ vector prepended by zeros. Calculate also the rotation matrix $R^{(k)}$ using Householder transformations (described in Panel 1.8 and Algorithm 1.5) to ensure orthogonality for the next loading.

Algorithm 1.6: clusterICA

```
required:  $M$-spacing entropy, Algorithm 1.1;
           Divisive k-means, Algorithm 1.4;
           Householder transformation, Algorithm 1.5
input:     data matrix $X \in \mathbb{R}^{n\times p}$;
           number of components, $l$
           (Optional) size of whitened matrix, $r$
output:    whitened matrix, $Y$;
           whitened directions $Q$;
           independent component directions, $AQ$;
           independent components, $Z = XAQ = YQ$;
begin
    % whiten matrix X: $r = \mathrm{rank}(\tilde{X})$, unless specified
% must have the number of components $l \leq r$
$A \leftarrow$ whitening matrix: $X \in \mathbb{R}^{n \times p}$
$Y \leftarrow \tilde{X}A$
% initialise rotation matrix
$R^{(0)} \leftarrow \mathfrak{I}_r$
for $k$ in $1:l$
    % set the size of the search space
    $m \leftarrow r - k + 1$
    %
    %% Step 1: random directions
    %
    $V \leftarrow (v_1, v_2, \ldots, v_N)^\top \in \mathbb{R}^{N \times m}$,
        with each $v_j \in \mathbb{R}^{m-1}\mathrm{P}$ uniformly distributed over $\mathbb{R}^{m-1}\mathrm{P}$
    set $\tilde{v}_j \leftarrow (0, 0, \ldots, 0, v_j) \in \mathbb{R}^r$
    set $q_j \leftarrow R^{(k-1)} \cdot \tilde{v}_j$
    %
    %% Step 2: calculate M-spacing of random directions
    %
    set $e_j \leftarrow$ $M$-spacing entropy; $d = Yq_j$,
                                        $M = \sqrt{n}$
    % put directions into increasing order with respect to entropy
    order the rows of $V$ such that $e_{(1)} \leq e_{(2)} \leq \cdots \leq e_{(N)}$
            i.e. increasing order of $M$-spacing entropy
    % keep the best $\tilde{N}$ directions
    $U \leftarrow (v_{(1)}, v_{(2)}, \ldots, v_{(\tilde{N})})$, which are associated with $e_{(j)}$, given by
            $e_{(j)} \leftarrow$ $M$-spacing entropy: $d = YR^{(k-1)}\tilde{v}_{(j)}$,
                                            $M = \sqrt{n}$
    %
    %% Step 3: clustering of the best $\tilde{N}$ directions
    %
    % sort $U$ into $K$ clusters using divisive k-means;
    $c \leftarrow$ divisive k-means: data, $v_{(1)}, \ldots, v_{(\tilde{N})}$,
                            tolerance, tol
    %   gives $K$ clusters, $c_j \in \{1, \ldots, K\}$, $j \in \{1, \ldots, \tilde{N}\}$
    $U = (U_1, U_2, \ldots, U_K)$ where $U_i = \{v_{(j)} : c_j = i\}$
    %   thus, $U_i \cap U_j = \varnothing$ and $|U_i| = \tilde{N}_i$, with $\sum_i \tilde{N}_i = \tilde{N}$
    %
    %% Step 4: optimisation within each cluster
    %
    % optimise the direction that minimises entropy within each cluster,
    % to obtain K directions,
            for i in 1:K
                % find best direction within cluster U_i
            $j \leftarrow \underset{i=1,\dots,K}{\text{argmin}}\ e_i^{(u)}$, with $\tilde{u}_i := (0,\dots,0,u_i)^\top \in \mathbb{R}^{r-1}\text{P}$ and,
                $v^{(i)} \leftarrow \text{argmin}_{v \in U_i} e^{(v)}$, where $\tilde{v} = (0,\dots,0,v)^\top \in \mathbb{R}^{r-1}\text{P}$, and
                        $e^{(v)} \leftarrow$ M-spacing entropy: $z = YR^{(k-1)}\tilde{v}$
                                                    $M = \sqrt{n}$
                % optimise with initial value given by best direction within cluster U_i
                $u_i \leftarrow$ optimisation over $v \in \mathbb{R}^{m-1}\text{P}$, with $\tilde{v} = (0,\dots,0,v)^\top \in \mathbb{R}^{r-1}\text{P}$:
                        initial value $v^{(i)}$
                        minimise M-spacing entropy: $z = YR^{(k-1)}\tilde{v}$,
                                                    $M = \sqrt{n}$
            end for
            % select best direction post-optimisation
            $j \leftarrow \underset{i=1,\dots,K}{\text{argmin}}\ e_i^{(u)}$, with $\tilde{u}_i := (0,\dots,0,u_i)^\top \in \mathbb{R}^{r-1}\text{P}$ and,
                    $e_i^{(u)} \leftarrow$ M-spacing entropy: $z = YR^{(k-1)}u_i$,
                                                    $M = \sqrt{n}$
            $\hat{u}_k \leftarrow u_j$
            $\hat{q}_k \leftarrow R^{(k-1)}\tilde{u}_k = R^{(k-1)} \cdot (0,\dots,0,\hat{u}_k)^\top$
            %
            %% Step 5: rotation matrix using Householder transformation
            %
            % update direction matrix by calculating Householder matrix
            $R^{(k)} \leftarrow$ Householder rotation; $v_r = \hat{u}_k$,
                                                    $R^{(k-1)} = R^{(k-1)}$
        end for
        $Q \leftarrow \{(R^{(l)})_{ij}, i = 1,2,\dots,p,\ j = 1,2,\dots,l\}$
        $Z \leftarrow Y \cdot Q$
        return Y; Q; AQ; Z
    end
```

1.4.2.5 ClusterICA with fastICA initialisations

As discussed briefly in Section 1.3.2, in large dimensions it can be shown that in the vast majority of cases, projections of data onto random directions are approximately Gaussian distributed (Diaconis & Freedman, 1984; von Weizsäcker, 1997). This result implies that as the rank $r = \text{rank}(\tilde{X})$ of the data increases (and thus so does the size of the whitened matrix $Y \in \mathbb{R}^{n\times r}$), the efficiency of the cluster-ICA initialisation method – where random directions are chosen to obtain a set of "good" starting projections – is reduced.

With fastICA (Hyvärinen & Oja, 2000) – as described in Section 1.4.1 – the contrast function that is optimised is an approximation to negentropy of a density that maximises Gaussianity given some constraints, described in Panel 1.5. This approach can result in the contrast function indicating that projections are much more Gaussian that they actually are, as was shown by way of example in Section 1.4.1.3. However, for higher dimensional data where the majority of projections are indeed close-to Gaussian, the fastICA objective function might be smooth and flat, with spikes in the (sparse) directions where projections of the data are not Gaussian. See the short discussion at the end of Section 1.4.1.1 on how the fastICA contrast function could display this behaviour.

Following this, in higher dimensions clusterICA can be set to include the direction that optimises the fastICA contrast function alongside the random directions in the initialisation step. That is, to find the k^{th} direction onto which to project the whitened matrix Y, also include in the initialisation set of random directions (described in Step 1., Panel 1.9), the vector $v_m \in \mathbb{R}^{m-1}\mathrm{P}$, $m = r - k + 1$, that maximises the function $\hat{J}[\hat{f}_0]$ (seen in Step 3, Panel 1.5). In the specific case of clusterICA, the vector v_m maximises,

$$\big(\mathsf{E}G(\mathsf{Y}^\top R^{(k-1)}\tilde{v}_m) - \mathsf{E}G(\mathsf{N})\big)^2, \tag{1.45}$$

where $G(x) = \log\big(\cosh(x)\big)$, $R^{(k-1)} \in \mathbb{R}^{r\times r}$ is the rotation matrix as described in Panel 1.8, and $\tilde{v}_m = (0, \dots, 0, v_m)^\top \in \mathbb{R}^{r-1}\mathrm{P}$, $\mathsf{Y} \in \mathbb{R}^r$ is a random vector of which the whitened matrix $Y \in \mathbb{R}^{n\times r}$ is n realisations, and $\mathsf{N} \sim \mathcal{N}(0, 1)$. In clusterICA, the expectations in (1.45) are approximated by the arithmetic mean.

1.4.2.6 Clustering example

One main advantage of the clusterICA method is the clustering of the directions. In this section a toy example is presented that demonstrates a case where clustering is useful. Here, 'useful' means that fewer directions need to be optimised post-clustering compared to pre-clustering to end up in the global minima of (M-spacing) entropy.

The plot of the toy example is shown in Figure 1.8. This is formed from a two dimensional projection of three independent sources. Figure 1.8 also contains the sample densities of two projections of the data – one at a global optimum of entropy and one at a local optimum. The plot of the (M-spacing) entropy of projections in directions from 0 to π is given in Figure 1.9. As we can see, the global optimum is a narrow well around $\vartheta = 0$, and when optimising from random starting directions many of these will find the two other local optima. In this example we show that by clustering the random directions, the number of directions needed to obtain the global optimum reduces.

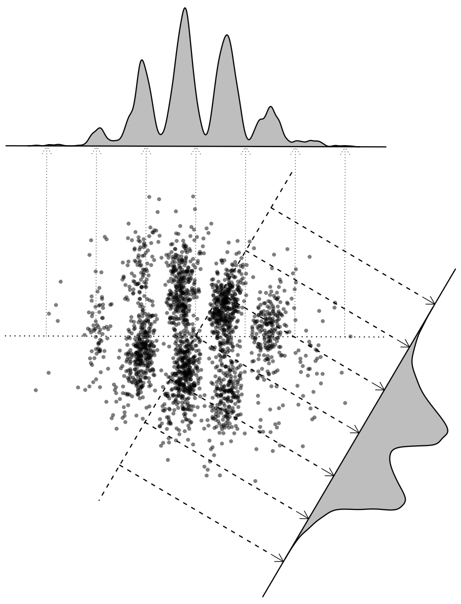


Figure 1.8: Distribution of a two dimensional projection of three independent sources. The approximate densities are estimated from the one dimensional projection onto the lines, and relate to a local and global minimum (at approximately $\theta = 0.3\pi$ and $\theta = 0$ respectively) of the entropy plot in Figure 1.9.

On $\mathbb{R}^{r-1}\mathrm{P}$ with $p = r = 2$ here, we initialise $N = 20$ random directions (following Step 1., Panel 1.9), and the $\tilde{N} = 10$ directions that result in the lowest M-spacing entropy are chosen (Step 2.). These are shown as the vertical lines in Figure 1.9. These directions are sorted into $K = 3$ clusters using k-means clustering (Step 3.), and the best direction from each cluster is shown by the

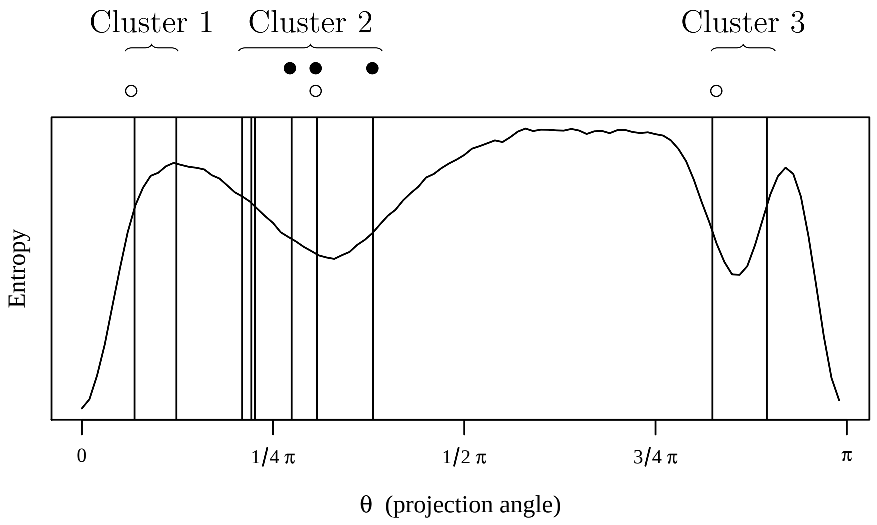


Figure 1.9: Entropy of projections of the data shown in Figure 1.8 over ten directions from 0 to π. Here, the three vertical lines with the '•' symbol above represent the best three random directions. The three lines with the '∘' symbol above represent the best random direction from each of the $K = 3$ clusters. The line with both the '•' and '∘' symbol above is a best direction overall and the best direction of one of the clusters.

symbol '∘' directly above the vertical line in Figure 1.9. The best three directions (irrespective of clustering) are shown by the '•' symbol. The line with both the '∘' and '•' symbol above is the best direction overall and thus also the best direction from one of the clusters. From Figure 1.9 it is clear that optimising the best three directions ('•') using standard gradient descent optimisation methods all result in finding a local optimum that is not the global optimum. However, after clustering and optimising the best direction that belongs to each cluster ('∘'), we find the global optimum for one of the optimised directions. The estimated density of the global minimum is shown by the dotted lines in Figure 1.8, with the local minimum found by naïvely optimising the best three directions shown by the dashed lines.

Thus it is clear that, in this example, clustering before optimisation results in obtaining the true minimum with fewer directions optimised. Here, the direction which results in the global minimum after optimisation is the fifth best (pre-optimisation) direction overall. Therefore by clustering, the number of directions that need to be optimised to obtain the global minimum reduces from five to three.

1.4.3 Comparing independent component analysis methods

In this section we compare differing independent component analysis methods to one-another. The methods include fastICA (described in Section 1.4.1), clusterICA (introduced above in Section 1.4.2), and two other established methods; Infomax ICA (Bell & Sejnowski, 1995) and Jade ICA (Cardoso & Souloumiac, 1993). The `R` code used to implement the respective independent component analysis methods are given by Marchini *et al.* (2013) for fastICA, and Nathaniel (2018) for Infomax and Jade ICA. The implementation of clusterICA in `R` is the work of the author.

In this example we use two dimensional samples similar to those used in Bach & Jordan (2002). Here the samples are from 18 distributions, whose densities are shown in Figure 1.10, and given by $f_1(\cdot), f_2(\cdot), \ldots, f_{18}(\cdot)$.

Here we sample n two-dimensional points, with each dimension sampled independently from $f_j(\cdot)$, where $j \in \{1, \ldots, 18\}$ to give a matrix $S \in \mathbb{R}^{n\times 2}$. For each sample a random rotation matrix is created by sampling $\theta \sim \mathcal{U}[0, 2\pi]$, and setting $R \in \mathbb{R}^{2\times 2}$ to

$$R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

The data is then transformed from S to X, where $X = S\,R^\top \in \mathbb{R}^{n\times p}$ has rank $r = 2$. The various independent component analysis methods are applied to the data X and the resulting unmixing matrices W, such that XW is an estimate of S, are compared to the original rotation matrix R using Amari error introduced in Bach & Jordan (2002). The Amari error for two matrices A, $B \in \mathbb{R}^{d\times d}$, is given by

$$d_A(A, B) = \frac{1}{2d}\left(\sum_{i=1}^{d}\left(\frac{\sum_{j=1}^{d}|c_{ij}|}{\max_j|c_{ij}|} - 1\right) + \sum_{j=1}^{d}\left(\frac{\sum_{i=1}^{d}|c_{ij}|}{\max_i|c_{ij}|} - 1\right)\right), \tag{1.46}$$

where $c_{ij} = (A\,B^{-1})_{ij}$. The Amari error $d_A(A, B) \in [0, 1]$ and equals zero when $A = B$. When A and B are rotation matrices, then the Amari error is also equal to zero when the rotations have a difference of $\pi/2$.

In this example, we sample a "training" set of $n = 1000$ points from each of the 18 two-dimensional distributions, each of which is rotated by a random rotation matrix. This process is repeated independently 5000 times. Therefore, there are 5000 mixed matrices for each of the 18 distributions to which we can apply the independent component analysis methods to. The Amari error is calculated for each of the independent component analysis methods on each sample, where A in (1.46) is the unmixing matrix found by the respective independent component analysis method, and B is the original rotation matrix. Therefore for each of the two-dimensional distributions, 5000 Amari errors are obtained, and the mean of these errors is calculated. The M-spacing entropy (see Section 1.3.1.3) calculated from the projection of each sample of size $n = 1000$ in the directions given by the independent component analysis methods is also calculated, with $M = \sqrt{1000}$. In this case, clusterICA is expected to perform well (that is, obtain a lower M-spacing entropy approximation) as the contrast function used in this method is based directly on the M-spacing entropy approximation.

The M-spacing entropy approximation tends to the true value of the differential entropy as the size of the samples increases, and therefore for a large enough sample size the M-spacing approximation should be close to true entropy. Therefore, 5000 independent "test" sets each of size $n = 10000$ are also considered for each two-dimensional distribution. These sets are projected in the directions found by the independent component analysis methods performed on the smaller original training sets, and M-spacing entropy is calculated for these projections of the test set, with $M = \sqrt{10000}$. These M-spacing values are theoretically close to the true value of differential entropy and thus can be used to help judge how successful the independent component analysis methods have been at finding the direction of the data that minimises differential entropy (*i.e.* maximises non-Gaussianity).

The mean of the Amari errors of each independent component analysis method for each distribution (1) - (18) is given in Table 1.11. Here, we see that clusterICA performs well against the established methods, and beats these methods when unmixing 6 of the 18 distributions. It can be seen that fastICA performs very poorly here, as is expected from the discussion in Section 1.4.1.

Table 1.12 gives the means for each two-dimensional distribution of the M-spacing entropy approximation of the 5000 training sets, projected in the directions found by each independent component analysis method. These same directions are used to project the larger test data sets and we calculate the M-spacing entropy of these projections. The means of these are shown in Table 1.13. As expected, Table 1.12 shows clusterICA outperforming all other methods. Table 1.13 shows clusterICA outperforming other methods for 9 of the 18 distributions, with Infomax performing best for 6 and Jade ICA the remaining 3.

These results suggests that clusterICA does indeed do well at finding directions that "minimise entropy" when the data is projected onto them, and that clusterICA is a valid independent component analysis method in practice. However, it is also significantly slower. Whilst the fastICA, Jade ICA and Infomax have a similar computing time in this example, clusterICA is more than 60 times slower, taking minutes rather than seconds to run. Also, as discussed in Section 1.4.2.5, clusterICA becomes much less efficient as the dimensions increase. This is a candidate for further research, for instance merging clusterICA with Jade ICA initialisations, or using the projective clustering part of clusterICA to improve Jade ICA.

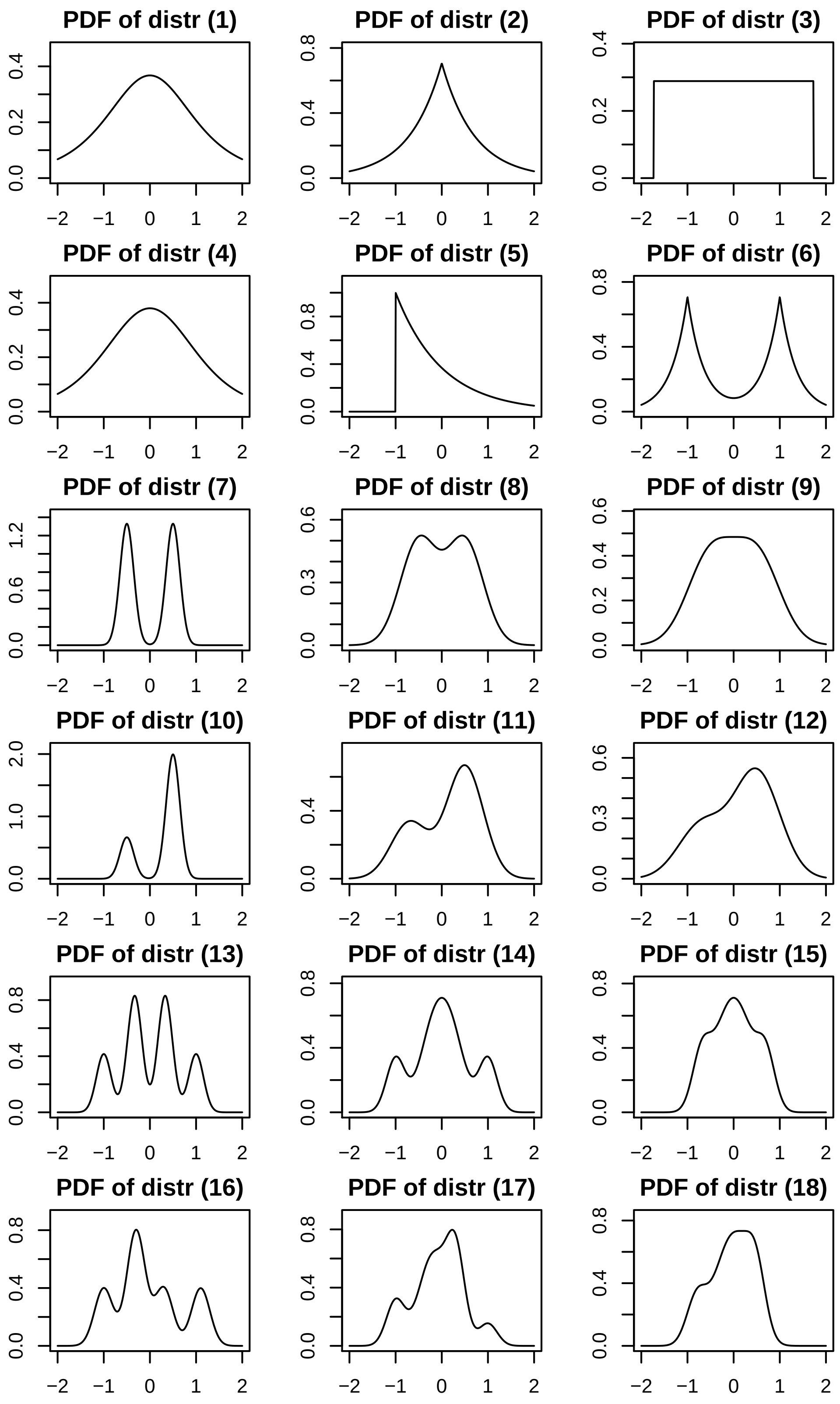


Figure 1.10: Densities of the 18 distributions used in Bach & Jordan (2002), created using the ‘ica’ package (Nathaniel, 2018)

Distribution	fastICA	Imax	JadeICA	clusterICA
1	25.2725	**14.5259**	15.3130	15.7769
2	24.9559	**13.4863**	14.6431	14.7753
3	23.5455	12.6209	**12.4017**	13.0182
4	25.0867	**14.0513**	15.0320	17.5280
5	25.8960	13.4429	13.7905	**12.2654**
6	23.2423	**12.3258**	12.8929	12.4370
7	22.8274	11.0457	**11.0007**	11.5811
8	24.5662	**13.7188**	13.8858	17.4829
9	26.7776	**16.0531**	16.1626	28.0554
10	49.9740	45.7964	14.1616	**12.3483**
11	25.4151	14.3189	**14.1775**	14.8379
12	26.9233	17.4029	**16.8607**	18.3334
13	24.8045	13.8418	12.8619	**12.3718**
14	30.1633	22.0310	13.7546	**13.4634**
15	25.1011	14.3988	**13.8151**	17.1314
16	25.9454	14.3873	12.6026	**12.3471**
17	43.6762	43.6602	21.6598	**14.9587**
18	25.1120	14.6767	**14.0712**	16.3141

Figure 1.11: Table showing mean of the Amari Errors related to each distribution (1) - (18) for 5000 samples of 1000 points for each independent component analysis method. Lower errors are better, with the best mean for each distribution highlighted in bold. Here, the Amari error means have been multiplied by 100.

Distribution	fastICA	Imax	JadeICA	clusterICA
1	1.0550	1.0531	1.0537	**1.0501**
2	1.2031	1.2023	1.2030	**1.1990**
3	1.1993	1.1984	1.1981	**1.1959**
4	1.2245	1.2241	1.2243	**1.2202**
5	0.9359	0.9227	0.9252	**0.9092**
6	1.0289	1.0274	1.0293	**1.0239**
7	0.7916	0.7908	0.7904	**0.7874**
8	1.2802	1.2804	1.2805	**1.2766**
9	1.2967	1.2969	1.2972	**1.2914**
10	0.9628	0.9480	0.7967	**0.7823**
11	1.2428	1.2425	1.2424	**1.2384**
12	1.2805	1.2802	1.2804	**1.2754**
13	1.1849	1.1783	1.1745	**1.1692**
14	1.2701	1.2666	1.2547	**1.2508**
15	1.2832	1.2832	1.2831	**1.2796**
16	1.2014	1.1959	1.1911	**1.1872**
17	1.2912	1.2918	1.2769	**1.2656**
18	1.2722	1.2720	1.2719	**1.2681**

Figure 1.12: Table showing mean of the m-spacing entropy of 5000 training data sets of size 1000 each, projected in the directions found by the respective independent component analysis method, for distribution (1) - (18). Lower entropy is better, with the best entropy mean for each distribution highlighted in bold.

Distribution	fastICA	Imax	JadeICA	clusterICA
1	1.1335	**1.1311**	1.1318	1.1315
2	1.2747	**1.2733**	1.2742	1.2737
3	1.2286	1.2264	1.2258	**1.2255**
4	1.2966	**1.2956**	1.2959	1.2963
5	0.9831	0.9672	0.9705	**0.9519**
6	1.0943	1.0926	1.0944	**1.0922**
7	0.8295	0.8287	**0.8283**	0.8286
8	1.3374	**1.3371**	1.3372	1.3379
9	1.3539	**1.3536**	1.3537	1.3545
10	1.0079	0.9898	0.8328	**0.8219**
11	1.3004	1.2996	1.2995	**1.2994**
12	1.3384	**1.3376**	1.3377	1.3377
13	1.2274	1.2199	1.2158	**1.2129**
14	1.3194	1.3151	1.3026	**1.3017**
15	1.3354	1.3349	**1.3348**	1.3356
16	1.2485	1.2421	1.2370	**1.2357**
17	1.3465	1.3460	1.3288	**1.3204**
18	1.3261	1.3252	**1.3251**	1.3256

Figure 1.13: Table showing mean of the m-spacing entropy of 5000 test data sets of size 10000, projected in the directions found by each independent component analysis method on the original training data set of size 1000, for distribution (1) - (18). Lower entropy is better, with the best entropy mean for each distribution highlighted in bold.

Chapter 2

Modelling Using Stochastic Processes

In the previous chapter, we discussed various methods to reduce the dimension of a simulator output to a set of principal or independent components. The overall aim here is to forecast some aspects of the original simulator using these lower dimensional components. This leads appropriately to the requirement to model these components in some way. The aim of this chapter is to introduce stochastic processes that can be used as models for these components. This procedure explicitly assumes that the information lost in the dimension reduction stage can be replaced by some "randomness", as was discussed in Panel 1.1. Here, the type of "randomness" present, and the type of stochastic processes, is constrained to allow for these processes to be applied to the components obtained in Part II of this thesis.

First, some preliminary theory must be introduced.

2.1 Preliminaries

The naming of this chapter as 'stochastic process models' refers to models describing some process over time that includes random elements. To obtain a definition of a stochastic process, we first give some probability and measure theory and introduce our notation.

Following notation as in Øksendal (2013), let $(\Omega, \mathcal{F}, \mathsf{P})$ be a probability space for some set Ω, $\mathcal{F}$ a σ-algebra on Ω and P a probability measure. The definitions of σ-algebra and probability measures are given below.

Definition 2.1.1 (σ-algebra)**.** *The σ-algebra $\mathcal{F}$ on Ω is a family of subsets of Ω such that:*

i. $\emptyset \in \mathcal{F}$;

ii. If $F \in \mathcal{F}$, then the complement, $\Omega \backslash F \in \mathcal{F}$;

iii. If $F_1, F_2, \ldots \in \mathcal{F}$, then $\bigcup_{i=1}^{\infty} F_i \in \mathcal{F}$.

Definition 2.1.2 (Probability measure)**.** *A probability measure associated with the set Ω and σ-algebra $\mathcal{F}$ is a function $\mathsf{P}\colon \mathcal{F} \to [0,1]$ such that:*

i. $\mathsf{P}(\emptyset) = 0$ *and* $\mathsf{P}(\Omega) = 1$;

ii. For $F_1, F_2, \ldots \in \mathcal{F}$ such that $F_i \cap F_j = \emptyset$, $i \neq j$,

$$\mathsf{P}\Big(\bigcup_{i=1}^{\infty} F_i\Big) = \sum_{i=1}^{\infty} \mathsf{P}(F_i).$$

Definition 2.1.3 (Random variable in $\mathbb{R}^p$)**.** *Let X be an $\mathcal{F}$-measurable function $\mathsf{X}\colon \Omega \to \mathbb{R}^p$, where $\mathcal{F}$-measurable means that for all open sets $U \in \mathbb{R}^n$, $\mathsf{X}^{-1}(U) := \{\omega \in \Omega;\, \mathsf{X}(\omega) \in U\} \in \mathcal{F}$. Then, X is a random variable in $\mathbb{R}^p$ on the probability space $(\Omega, \mathcal{F}, \mathsf{P})$ with distribution $\mu_{\mathsf{X}}(B) = \mathsf{P}(\mathsf{X}^{-1}(B))$.*

Definition 2.1.4 (Stochastic process)**.** *A stochastic process is a family of random variables (as given in Definition 2.1.3) defined on the probability space $(\Omega, \mathcal{F}, \mathsf{P})$, and is given by $(\mathsf{X}_t)_{t \in T}$, where T is the ordered parameter space and for each $t \in T$,*

$\omega \to \mathsf{X}_t(\omega)$ is a random variable. The function $t \to \mathsf{X}_t(\omega)$ for some fixed $\omega \in \Omega$, will be referred to interchangeably as the trajectory or path of X_t.

In this chapter, upper-case letters in sans serif typeface are used to notate random variables and stochastic processes, following on from the notation used in Chapter 1 and discussed in Panel 1.1. The only departure from this is the "white-noise" process introduced in Section 2.2, which is notated $(\varepsilon_t)_{t\in\mathbb{Z}}$ by convention. Observations from a stochastic process will be given by lower-case letters in computer modern typeface.

This chapter is arranged into two major parts. The first part is given in Section 2.2 and refers to classical *time series models*, which are defined to be causal stochastic models in discrete time such that T in Definition 2.1.4 is the set of integers $\mathbb{Z}$. Therefore, the process is a (countable) family $(\mathsf{X}_t)_{t\in T} = \{\ldots, \mathsf{X}_1, \mathsf{X}_2, \mathsf{X}_3, \ldots\}$. *Causal* means that the value at time t of a time series can only depend on observations or functions up to time t, and no future information is used.

Section 2.3 discusses processes obtained from stochastic differential equations defined on continuous time such that T in Definition 2.1.4 is the positive real line $[0, \infty)$, and the output is given by $(\mathsf{X}_t)_{t\geq 0}$. Clearly in practice the observed process must be a sample from the continuous output and thus a countable family $\{x_{t_1}, x_{t_2}, x_{t_3}, \ldots\}$ for some $t_1 < t_2 < t_3 < \ldots$. This section includes a novel stochastic process defined by the integral of a stochastic differential equation over disjoint time intervals, called block-average and described in Section 2.3.2.1. An associated parameter fitting procedure for this process is described in Section 2.3.3.3. We also discuss a novel spline function that can be used to ensure that seasonality in pointwise stochastic differential equations is mirrored in the respective block-average process (Section 2.3.4).

In this chapter we only consider univariate processes, such that $(\mathsf{X}_t)_{t\in T}$ only take values in $\mathbb{R}$. Here, the processes only depend on their own previous trajectory and some stochastic "noise" to determine the future state, and there is no dependency on other explanatory variables or processes. This choice of only considering univariate processes was done for three main reasons, one of them being the inevitable time-constraint whilst doing post-graduate research. The second reason was due to the original motivation of this thesis, where simpler models are desired to help with interpretability, computational intensiveness and forecasting of some high dimensional climate simulator. Purely focussing on univariate processes reduces the possibility of an increasing creep of "black box" modelling, which is to be avoided for tractability where possible. The third reason was related to the dimension reduction techniques discussed in Chapter 1. With independent component analysis, introduced in Section 1.3, the independent components are theoretically estimating some *independent* source signals and indeed are linearly independent by design. For multivariate theory for time series models see Priestley (1981, Volume II) and for related stochastic differential equation theory see Øksendal (2013) and Kessler *et al.* (2012).

2.2 Time Series Models

In this section we discuss some classes of discrete causal processes in time that contain a random element. Although no novel work is introduced in this section, the mathematics discussed here gives a good background knowledge to "classical" time series, and these processes will be used for the initial modelling of the components obtained by the dimension reduction techniques in Chapter 1, on simulator output given in Part II of this thesis. This will allow comparisons between the classical models given in this section and the more advanced models given in Section 2.3.

For notation throughout, the upper-case X_t describes an individual random variable from a process whose distribution is given by the t^{th} step of a time series model. An observation or realisation of a time series model at step t is denoted by a lower-case x_t.

The first properties to examine in, and remove from, a process are *trend* and *seasonal effects*. Using the same loose definition as in Chatfield (2016), a trend is given to be some long-term change in the underlying mean level, where neither "long-term" nor "mean level" is well defined. The simplest way to represent a trend in the local mean level is by the deterministic linear equation $\mu(t) = \alpha + \beta t$, where $\mu(t)$ is the local level and β is the trend term. The seasonal effects are cyclic variation arising in the data with some periodicity, such that $\mu(t) = \mu(t + k \cdot N)$ for some $N > 0$, and for all $k \in \mathbb{Z}$. Often, seasonal effects are described as having an annual period (that is, the N above is one year) with cyclic effects having any other period, but here no differentiation is made between them.

The seasonal effect is an important effect to remove from a climate simulator if month-to-month variability due to the Earth's orbital eccentricity and obliquity (axial tilt) want to be accounted for. An example of a mean function that displays some seasonal behaviour is given in Figure 2.1. In this case the solid black line gives the mean function of, say, some climate simulator output run over a number of years, displaying strong seasonality. This seasonality needs to be removed to obtain a mean function given by the dashed line, before time series modelling using the methods discussed in this section can commence.

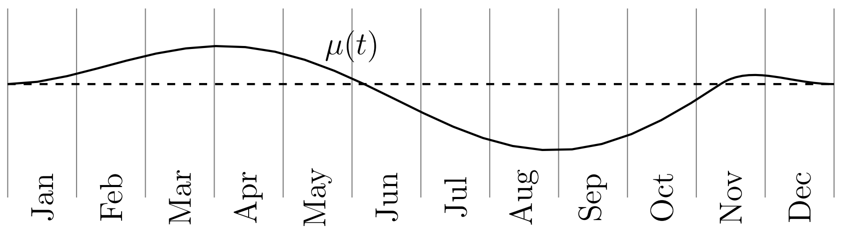


Figure 2.1: Example of removing seasonality from a time series from some climate simulator output that has yearly repeating behaviour. The solid black line gives some mean function of the simulator output over the year, which exhibits some strong seasonal affect. This function is required to be removed from the output to give a constant mean function given by the dashed line, before modelling the output using time series models.

The arrangement of this section corresponds to *stationary* and *non-stationary* time series models, and thus we introduce this concept here. Suppose the process $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ is from a time series model. Then, $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ is a (second-order) stationary time series process if the first moment is finite and does not depend on time, and the second moment is finite and only depends on the lag between the two variables. That is,

$$\begin{aligned} \mathsf{E}\mathsf{X}_t &= \mu(t) =: \mu \quad \text{for all } t \in \mathbb{Z} \\ \operatorname{Cov}(\mathsf{X}_t, \mathsf{X}_{t+k}) &= \mathsf{E}(\mathsf{X}_t - \mu)(\mathsf{X}_{t+k} - \mu) =: \gamma(k), \end{aligned} \tag{2.1}$$

for any t, $t+k \in \mathbb{Z}$. Here, $\mu = \mu(t)$, $t \in \mathbb{Z}$, is constant throughout time, so that any trend and seasonality as discussed above has been removed. The *autocovariance function* $\gamma \colon \mathbb{Z} \to \mathbb{R}$ as in (2.1) is often standardised to give the *autocorrelation function* $\rho \colon \mathbb{Z} \to [-1, 1]$,

$$\rho(k) = \frac{\gamma(k)}{\gamma(0)} = \frac{\gamma(k)}{\sigma^2},$$

where $\sigma^2 = \operatorname{Var} \mathsf{X}_t$ for all $k \in \mathbb{Z}$.

The autocovariance function (autocorrelation function) is a function of the autocovariance (autocorrelation) against the lags k, and is a useful analysis tool for time series models. These two functions are now considered for a specific time series process, which is introduced in Definition 2.2.2. First, the *white noise* process must be defined.

Definition 2.2.1 (White noise process)**.** *The process $(\varepsilon_t)_{t\in\mathbb{Z}}$, with each $\varepsilon_t \in \mathbb{R}$, is known as the white-noise process in $\mathbb{R}$ exactly when the mean and autocovariance are given by*

$$\begin{aligned} \mathsf{E}\, \varepsilon_t &= 0; \;\; \textit{and,} \\ \gamma(k) &= \mathsf{E}\, \varepsilon_t \varepsilon_{t+k} = \sigma_\varepsilon^2 \mathbb{1}_{k=0}, \end{aligned}$$

for all $t \in \mathbb{Z}$, $k \in \mathbb{Z}$ and for some $\sigma_\varepsilon^2 \in (0, \infty)$. That is, $(\varepsilon_t)_{t\in\mathbb{Z}}$ is a sequence of uncorrelated zero-mean random variables, each with variance σ_ε^2.

Definition 2.2.2 (General linear process (Box & Jenkins, 1976))**.** *Let $(\varepsilon_t)_{t\in\mathbb{Z}}$ be a white noise process, and B the backwards shift operator such that $B^d \mathsf{X}_t = \mathsf{X}_{t-d}$ for some process $(\mathsf{X}_t)_{t\in\mathbb{Z}}$. Then, $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ is a general linear process exactly when*

$$\begin{aligned} \mathsf{X}_t &:= (1 + \sum_{j=1}^{\infty} \zeta_j B^j) \varepsilon_t \\ &= \zeta(B) \varepsilon_t, , \end{aligned} \tag{2.2}$$

with $\zeta(B) = (1 + \sum_{j=1}^{\infty} \zeta_j B^j)$.

A general linear process $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ is stationary if $\zeta(x) = (1+\sum_{j=1}^{\infty}\zeta_j x^j) < \infty$ for $|x| \leq 1$. The autocovariance function for a stationary general linear process can be found by substituting X_t as in (2.2) into the autocovariance definition (2.1) to give, with $\zeta_0 = 1$,

$$\begin{aligned}\gamma(k) &= \mathsf{E}\mathsf{X}_t\mathsf{X}_{t+k}\\ &= \mathsf{E}\Big(\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\zeta_i\zeta_j\varepsilon_{t-i}\varepsilon_{t+k-j}\Big)\\ &= \sigma_\varepsilon^2\sum_{j=0}^{\infty}\zeta_j\zeta_{j+k},\end{aligned}$$

where the last equality is due to the covariance of the white noise process, $\mathsf{E}\varepsilon_t\varepsilon_{t+k} = \sigma_\varepsilon^2\mathbb{1}_{k=0}$. With $k=0$ we obtain the variance $\sigma_{\mathsf{X}}^2 = \gamma(0) = \sigma_\varepsilon^2\sum_{j=0}^{\infty}\zeta_j^2$.

For a set of observations $\{x_1, x_2, \ldots, x_n\}$ with $\bar{x} := \frac{1}{n}\sum_{t=1}^{n}x_t = 0$, the sample autocovariance for lag k is given by,

$$c(k) = \frac{1}{n}\sum_{t=1}^{n-k}x_t x_{t+k}, \tag{2.3}$$

and the sample autocorrelation at lag k is

$$r(k) = \frac{c(k)}{c(0)}. \tag{2.4}$$

The plot of the function $r(k)$ against the lag k is known as the *correlogram*. If the sample autocorrelation function is calculated using observations from a non-stationary process, then the decay will be very slow as a function of the lag.

In this section a function known as the *partial autocorrelation* is used in addition to the autocorrelation to help understand the behaviour of a stationary process or a set of observations. A definition is not given here but instead the readers are referred to Box & Jenkins (1976, Section 3.2.5 and 3.3.2) for the specific partial autocorrelation functions for the time series models that are discussed in this section.

Remark 2.10: Spectral density function

An equivalent function to the autocovariance function is the spectral density function, also known as the power spectrum. This is the Fourier transform of the autocovariance function, and describes the distribution of the variance of a process over its frequency. In general, analysis using the autocovariance or autocorrelation function is known as time domain analysis, whereas using the spectral density it is known as frequency domain or spectral analysis. See, *e.g.* Jenkins & Watts (1968, Chapter 6) for a detailed description of spectral analysis on a time series.

Another important concept related to time series models is *invertibility*. First note that the general linear process can be expressed as an infinite sum of all past values of X_j, $j \leq t$, *i.e.*

$$\pi(B)\mathsf{X}_t = \varepsilon_t, \tag{2.5}$$

where $\pi(B) := 1 - \sum_{j=1}^{\infty} \pi_j B^j$ is the generating function of the weights π_j. Then, from Box & Jenkins (1976), $\pi(B) = \phi^{-1}(B)$ and the process is invertible if $\pi(x) = 1 - \sum_{j=1}^{\infty} \pi_j x^j$ converges for all $|x| \leq 1$. That is, the weights must satisfy $\sum_{j=1}^{\infty} |\pi_j| < \infty$.

This section is arranged as follows. In Section 2.2.1 we introduce some common stationary time series models; the autoregressive, moving average, and mixed autoregressive-moving average processes. In Section 2.2.2 we introduce the integrated autoregressive-moving average process which describes a non-stationary time series model.

2.2.1 Time series models for stationary processes

In this section we assume that the process $(\mathsf{X}_t)_{t\in\mathbb{Z}} = \{\ldots, \mathsf{X}_1, \mathsf{X}_2, \mathsf{X}_3, \ldots\}$ is in stationarity, with mean $\mu := \mathsf{E}\mathsf{X}_t = 0$ (assumed without loss of generality) and autocovariance function $\gamma(k)$ for all $k \in \mathbb{Z}$.

In Section 2.2.1.1 we introduce the autoregressive process (notated AR(p) with p parameters) and the moving average process (notated MA(q) with q parameters) is introduced in Section 2.2.1.2. In Section 2.2.1.3 we introduce a mixture of the two, known as autoregressive-moving average processes and notated ARMA(p, q).

2.2.1.1 Autoregressive process

Let $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ be a process of the form,

$$\mathsf{X}_t = \varphi_1\mathsf{X}_{t-1} + \varphi_2\mathsf{X}_{t-2} + \cdots + \varphi_p\mathsf{X}_{t-p} + \varepsilon_t, \tag{2.6}$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ independent for all $i \in \mathbb{Z}$. Then, $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ is an autoregressive process of order p, notated AR(p). Using the backwards shift operator, (2.6) becomes $\varphi(B)\mathsf{X}_t = \varepsilon_t$, where $\varphi(B) = 1 - \varphi_1 B - \ldots - \varphi_p B^p$ is a polynomial of order p. From this it can be seen that (2.6) can be rewritten as a general linear process by $\mathsf{X}_t = \varphi^{-1}(B)\varepsilon_t$, which is in the form of (2.2) with $\zeta(B) = \varphi^{-1}(B)$. That is, an AR($p$) process can be rewritten as an infinite sum of weighted independent Gaussians (*i.e.* a moving average process with infinite parameters, which we discuss in the next section). As discussed in Section 2.2, the process $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ is stationary if $\zeta(x)$ converges for all $|x| \leq 1$. This is equivalent to the requirement that the roots of $\varphi(x) = 0$ must satisfy $|x| > 1$, *i.e.* lie outside the unit circle.

The autocorrelation function for an autoregressive process does not have a cut-off point for which it is zero after, but instead decays in an exponential manner. On the other hand the partial autocorrelation function of an AR(p) process is zero for all lags greater than p.

Suppose that the observations $\{x_1, x_2, \ldots, x_n\}$ are known to be from an AR(p) process with p to be determined. It is clear that the value of p can not be estimated by only looking at the sample autocorrelation function, due to the exponential decay. However, the sample partial autocorrelation function can be used to estimate the value of p by examining the lag for which the function is near zero afterwards. This is shown by way of an example below, and is discussed in more detail in Section 3.1.1.

Example 2.2.3 (Autoregressive process, order 1)**.** *In this example* $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ *is an* $AR(1)$ *process which has the form,*

$$\mathsf{X}_t = \varphi \mathsf{X}_{t-1} + \varepsilon_t,$$

with $\varphi = 0.7$ *and* $\varepsilon_j \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ *for* $j \in \mathbb{Z}$, $\sigma_\varepsilon^2 = 0.4$*. A realisation* $\{x_1, x_2, \dots, x_{250}\}$ *of the process* $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ *is shown in Figure 2.2.*

Using this realisation, the sample autocorrelation function is calculated and given in Figure 2.3, and shows an exponential decay with no clear cut off. However, the sample partial autocorrelation function shown in Figure 2.4 is close to zero for all lags greater than one. As discussed above, this is the expected behaviour of an $AR(1)$ *process.*

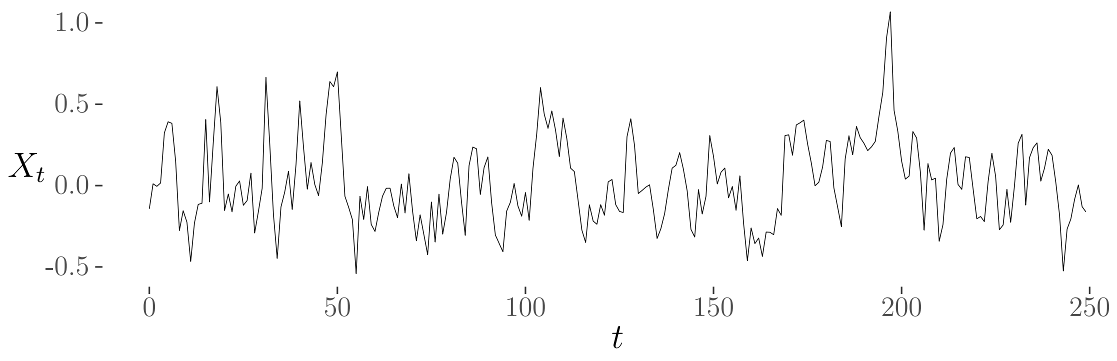


Figure 2.2: Time series of a realisation $\{x_1, x_2, \dots, x_{250}\}$ of an autoregressive process of order 1, $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ of the form $\mathsf{X}_t = \varphi\mathsf{X}_{t-1} + \varepsilon_t$, with $\varphi = 0.7$ and $\varepsilon_j \sim \mathcal{N}(0, 0.4)$.

2.2.1.2 Moving average process

Let $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ be a general linear model with a finite number of non-zero parameters,

$$\mathsf{X}_t = \varepsilon_t + \vartheta_1 \varepsilon_{t-1} + \cdots + \vartheta_q \varepsilon_{t-q}, \tag{2.7}$$

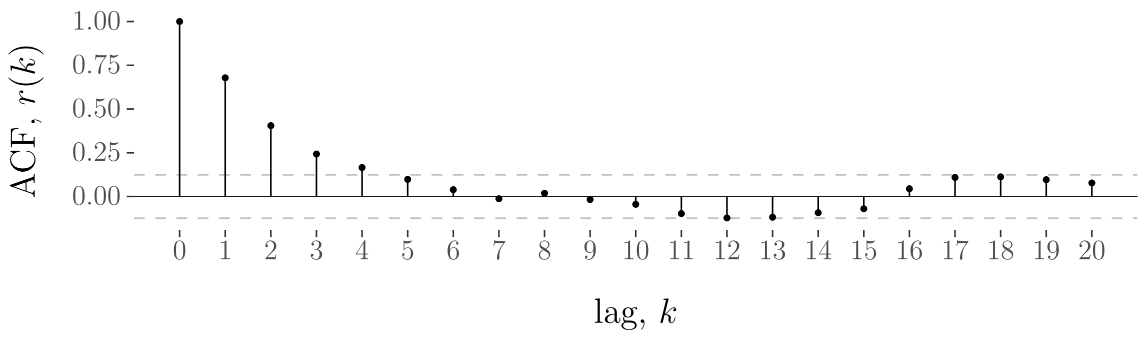


Figure 2.3: Sample autocorrelation function of a realisation $\{x_1, x_2, \ldots, x_{250}\}$ of an autoregressive process of order 1, $(\mathsf{X}_t)_{t\in\mathbb{Z}}$, of the form $\mathsf{X}_t = \varphi\mathsf{X}_{t-1} + \varepsilon_t$, with $\varphi = 0.7$ and $\varepsilon_j \sim \mathcal{N}(0, 0.4)$. Note that here there is no clear cut off point K for which $r(k) \approx 0$ for all $k \geq K$.

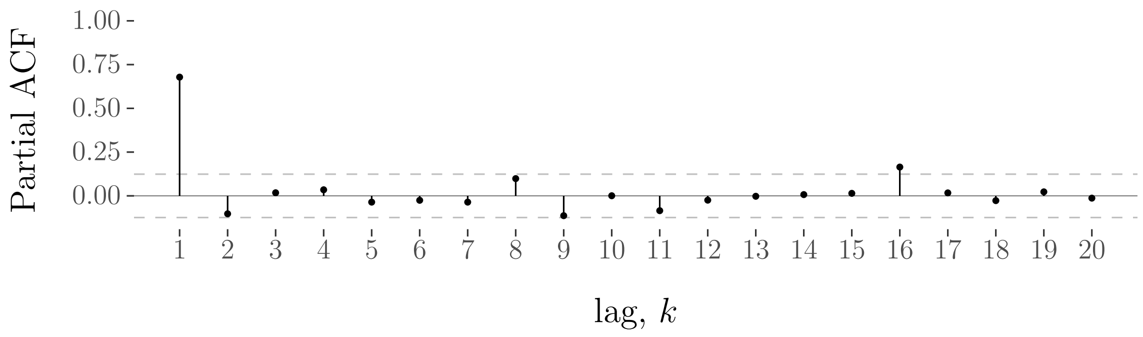


Figure 2.4: Sample partial autocorrelation function of a realisation $\{x_1, x_2, \ldots, x_{250}\}$ of an autoregressive process of order 1, $(\mathsf{X}_t)_{t\in\mathbb{Z}}$, of the form $\mathsf{X}_t = \varphi\mathsf{X}_{t-1} + \varepsilon_t$, with $\varphi = 0.7$ and $\varepsilon_j \sim \mathcal{N}(0, 0.4)$. Note that for all lags greater than 1 the partial autocorrelations are all close to zero (apart from at lag 16, which lies just outside the 95% confidence interval), as expected for an AR(1) process.

where $\varepsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ independent for all $i \in \mathbb{Z}$. Then, $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ is a moving average process of order q, denoted MA(q). The form given in (2.7) can be rewritten in terms of the backwards shift operator, $\mathsf{X}_t = \vartheta(B)\varepsilon_t$, where $\vartheta(B) = 1 + \vartheta_1 B + \ldots + \vartheta_q B^q$ is a polynomial of order q. This process is stationary for all parameter values, and is invertible if (2.7) can be written in the form $\pi(B)\mathsf{X}_t = \varepsilon_t$ (as in (2.5)) with $\pi(x) < \infty$ for all $|x| \leq 1$. The invertibility condition is equivalent to the roots of $\vartheta(x) = 0$ satisfying $|x| > 0$, *i.e.* the roots lie outside the unit circle. If an MA(q) process is invertible, then it can be written as an infinite sum of the past values of the process (that is, in the form of an AR(∞) process).

Many authors use a different form for autoregressive processes such that the signs on the right-hand side of (2.7) are negative, for instance in Box & Jenkins (1976). Although this makes no difference to the theory, in practise one needs to be wary as to which form is being used, as there is no standard used across computer software and functions.

The autocorrelation function of a moving average process of order q is (see, *e.g.* Chatfield, 2000),

$$\rho(k) = \begin{cases} 1, & \text{for } k = 0 \\ \frac{\sum_{i=0}^{q-k} \vartheta_i \vartheta_{i+k}}{\sum_{i=0}^{q} \vartheta_i^2}, & \text{for } k = 1, 2, \dots, q \\ 0, & \text{for } k > q, \end{cases} \tag{2.8}$$

where $\vartheta_0 = 1$. From this it is clear that for a realisation of an MA(q) process, $\{x_1, x_2, \dots, x_n\}$, the sample autocorrelation should be close to zero for all lags greater than q. The sample partial autocorrelation will not exhibit a clear cut-off, but should tail off to zero (Box & Jenkins, 1976, Section 6.2). This insight is used for model identification and forecasting in Section 3.1.1.

Example 2.2.4 (Moving average process, order 1)**.** *In this example* $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ *is an MA*(1) *process which has the form,*

$$\mathsf{X}_t = \varepsilon_t + \vartheta\varepsilon_{t-1},$$

with $\vartheta = 0.8$ *and* $\varepsilon_j \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ *for* $j \in \mathbb{Z}$, $\sigma_\varepsilon^2 = 0.4$. *A realisation of the process is given by* $\{x_1, x_2, \dots, x_{250}\}$ *and is shown in Figure 2.5.*

Using this realisation, the sample autocorrelation function is calculated and given in Figure 2.6, and shows a very quick delay. After lag 1*, the autocorrelation is close to zero, which we expect from the theory for a moving average process of order* 1*. From* (2.8) *the true lag* 1 *autocorrelation of the moving average process* *is* $\rho(1) = (1 \times 0.8)/(1^2 + 0.8^2) \approx 0.4878$*, which is comparable to the lag* 1 *sample autocorrelation from this realisation of* $r(1) = 0.4786$*. The sample partial autocorrelation shown in Figure 2.7 does not show a cut-off after a certain lag, but instead tails off in an alternating positive-negative manner.*

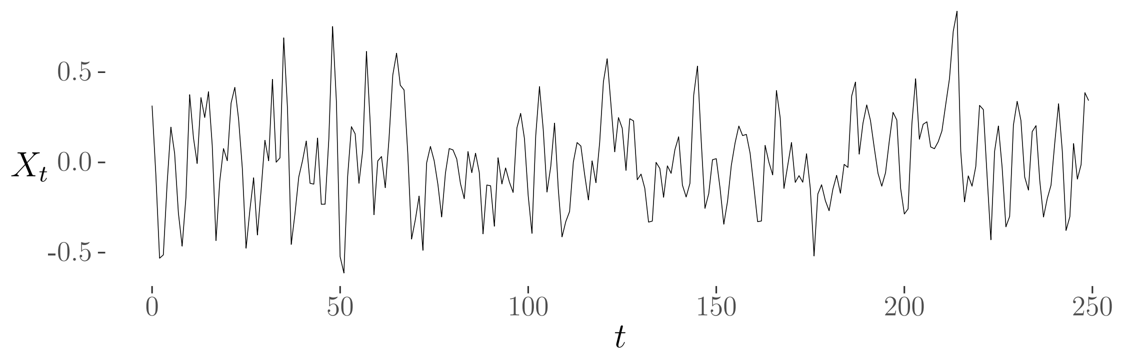


Figure 2.5: Time series of a realisation $\{x_1, x_2, \ldots, x_{250}\}$ of a moving average process of order 1, $(\mathsf{X}_t)_{t\in\mathbb{Z}}$, of the form $\mathsf{X}_t = \varepsilon_t + \vartheta\varepsilon_{t-1}$ with $\vartheta = 0.8$ and $\varepsilon_j \sim \mathcal{N}(0, 0.4)$.

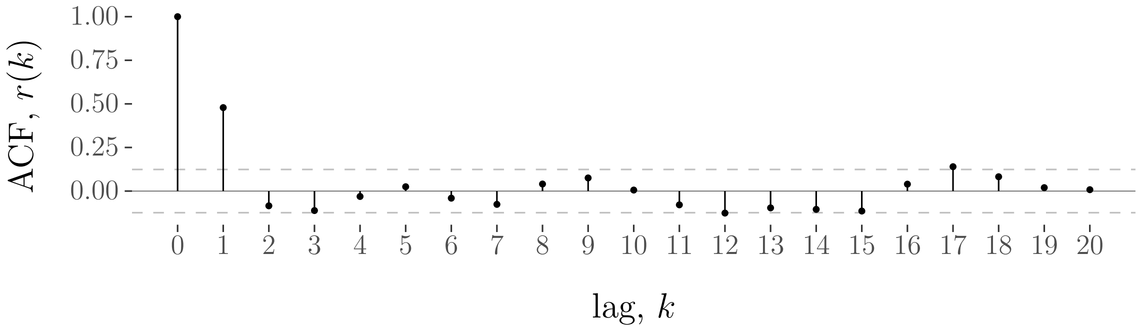


Figure 2.6: Sample autocorrelation function of a realisation $\{x_1, x_2, \ldots, x_{250}\}$ of an autoregressive process of order 1, $(\mathsf{X}_t)_{t\in\mathbb{Z}}$, of the form $\mathsf{X}_t = \varepsilon_t + \vartheta\varepsilon_{t-1}$ with $\vartheta = 0.8$ and $\varepsilon_j \sim \mathcal{N}(0, 0.4)$. Note that here there is a clear cut off after lag 1, and all other sample autocorrelations are close to zero.

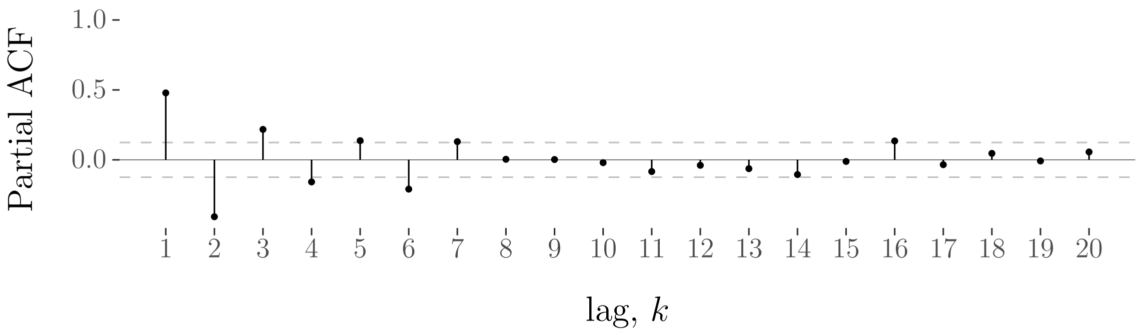


Figure 2.7: Sample partial autocorrelation function of a realisation $\{x_1, x_2, \ldots, x_{250}\}$ of an autoregressive process of order 1, $(\mathsf{X}_t)_{t\in\mathbb{Z}}$, of the form $\mathsf{X}_t = \varepsilon_t + \vartheta\varepsilon_{t-1}$ with $\vartheta = 0.8$ and $\varepsilon_j \sim \mathcal{N}(0, 0.4)$. This function exhibits alternating positive-negative decay with no clear cut-off.

2.2.1.3 Autoregressive-moving average process

Here we introduce the autoregressive-moving average process, which is a mixture of the autoregressive and moving average processes that have been discussed in the sections above. The general form of an ARMA(p, q) process, $(\mathsf{X}_t)_{t\in\mathbb{Z}}$, is given by

$$\mathsf{X}_t = \varphi_1\mathsf{X}_{t-1} + \ldots + \varphi_p\mathsf{X}_{t-p} + \varepsilon_t + \vartheta_1\varepsilon_{t-1} + \ldots \vartheta_q\varepsilon_{t-q},$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ independent for all $t \in T$. This can be written using the backwards shift operator as

$$\varphi(B)\mathsf{X}_t = \vartheta(B)\varepsilon_t, \tag{2.9}$$

where φ and ϑ are polynomial operators of degree p and q respectively. As with the stationary autoregressive process, the autoregressive-moving average process is stationary if the roots of $\varphi(x) = 0$ lie outside the unit circle. The invertibility condition is the same as that of the moving average process, in that the autoregressive-moving average process is invertible if the roots of $\vartheta(x) = 0$ lie outside the unit circle.

A stationary and invertible ARMA(p, q) process can be written as an infinite order autoregressive process

$$\pi(B)\mathsf{X}_t = \varepsilon_t$$

with $\pi(B) = \vartheta^{-1}(B)\varphi(B)$, and as an infinite order moving average process,

$$\mathsf{X}_t = \zeta(B)\varepsilon_t$$

with $\zeta(B) = \varphi^{-1}(B)\vartheta(B)$.

As the autoregressive-moving average process exhibits autoregressive behaviour, the autocorrelation function should have no clear cut-off, but instead tail off to zero relatively quickly. Conversely, as the process includes moving average behaviour, the partial autocorrelation function should also have no clear cut-off point. For an ARMA(p, q) process, the autocorrelation function is a mixture of exponentials and damped sine waves after the $q - p$ lag, and the partial autocorrelation function is dominated by a mixture of exponentials and damped sine waves after the $p - q$ lag (Box & Jenkins, 1976, Section 6.2.1).

An initial assumption for the principal and independent components found in Part II on the various simulator outputs (using the theory introduced in Chapter 1) is that these are realisations from ARMA(p, q) processes. After removing the trend and seasonality from each realisation and checking stationarity, the values of p and q are estimated alongside the associated parameter values. This is explained in detail in Section 3.1.1. A method to model non-stationary realisations and to help modify a non-stationary realisation into a stationary one is given below in Section 2.2.2.1.

Example 2.2.5 (Autoregressive-moving average process, order $(1, 1)$)**.** *In this example* $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ *is a process which has the form,*

$$\mathsf{X}_t = \varphi\mathsf{X}_{t-1} + \varepsilon_t + \vartheta\varepsilon_{t-1},$$

with $\varphi = 0.7$, $\vartheta = 0.8$ *and* $\varepsilon_j \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ *for* $j \in \mathbb{Z}$, $\sigma_\varepsilon^2 = 0.4$. *A realisation* $\{x_1, x_2, \ldots, x_{250}\}$ *of the process is shown in Figure 2.8.*

Using this realisation, the sample autocorrelation function is calculated and given in Figure 2.9, and shows a (relatively fast) delay with no clear cut off. This suggests that the process has some autoregressive behaviour. The sample partial autocorrelation function calculated using the realisation and shown in Figure 2.10 also displays no clear cut off. This suggests that the observations are from an ARMA(p, q) *process, where p and q need to be determined. However, determining the autoregressive and moving average orders is non-trivial, as it is a difficult skill for most to tell when the autocorrelation and partial autocorrelation functions become dominated by a mixture of exponentials and sine waves. This difficulty is highlighted in Example 3.1.1 in Section 3.1.1. For an ARMA*$(1, 1)$ *process, the autocorrelation function should decay exponentially from the first lag, and the partial autocorrelation function should be dominated by exponential decay from the first lag.*

This example is presented here to give an indication of the difficulty in using the autocorrelation and partial autocorrelation functions to identify the structure of a process. This example is referred back to in Chapter 3 to show how model criterion could be used to help identify the correct model.

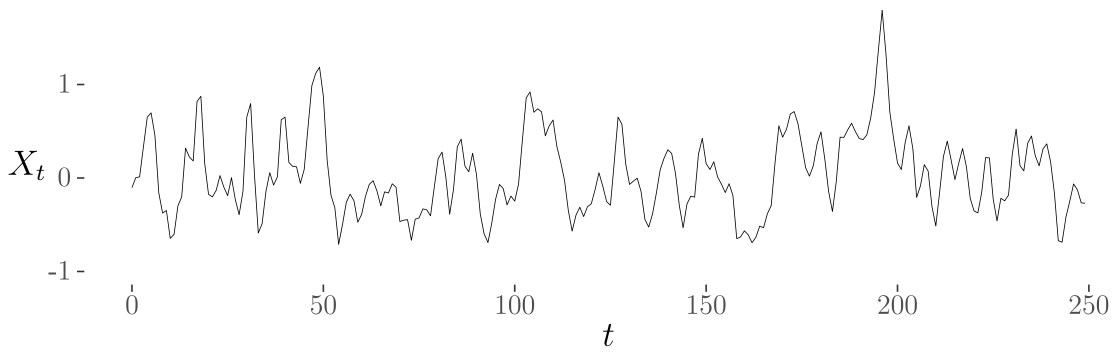


Figure 2.8: Time series of a realisation $\{x_1, x_2, \ldots, x_{250}\}$ of an autoregressive-moving average process of order $(1, 1)$, $(\mathsf{X}_t)_{t\in\mathbb{Z}}$, of the form $\mathsf{X}_t = \varphi\mathsf{X}_{t-1} + \varepsilon_t + \vartheta\varepsilon_{t-1}$ with $\varphi = 0.7$, $\vartheta = 0.8$ and $\varepsilon_j \sim \mathcal{N}(0, 0.4)$.

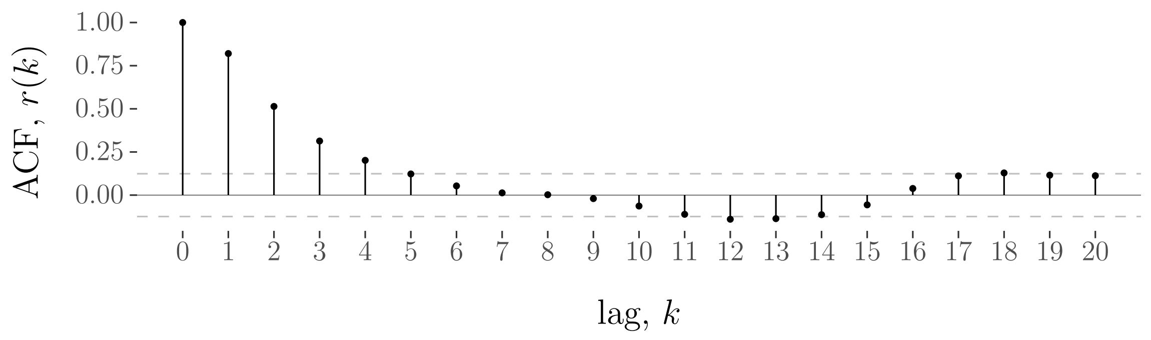


Figure 2.9: Sample autocorrelation function of a realisation $\{x_1, x_2, \ldots, x_{250}\}$ of an autoregressive-moving average process of order $(1,1)$, $(\mathsf{X}_t)_{t\in\mathbb{Z}}$, of the form $\mathsf{X}_t = \varphi\mathsf{X}_{t-1} + \varepsilon_t + \vartheta\varepsilon_{t-1}$ with $\varphi = 0.7$, $\vartheta = 0.8$ and $\varepsilon_j \sim \mathcal{N}(0, 0.4)$. Note that here there is no clear cut-off point, suggesting that autoregressive behaviour is present in the observations.

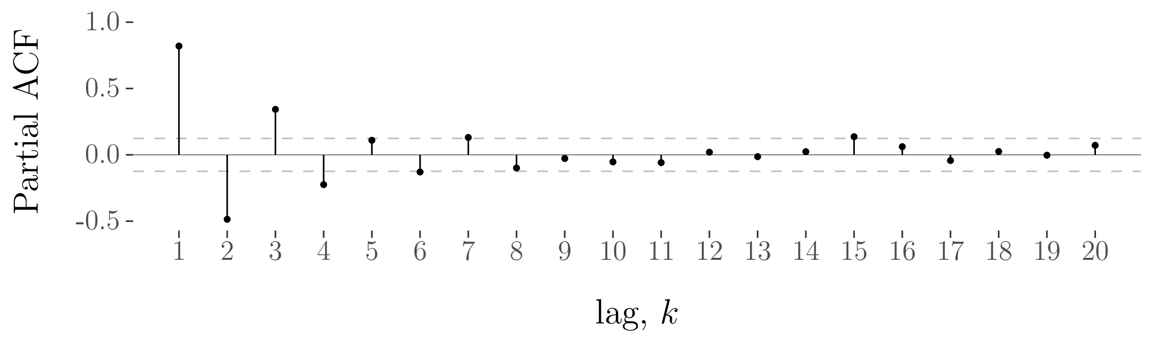


Figure 2.10: Sample partial autocorrelation function of a realisation $\{x_1, x_2, \ldots, x_{250}\}$ of an autoregressive-moving average process of order $(1,1)$, $(\mathsf{X}_t)_{t\in\mathbb{Z}}$, of the form $\mathsf{X}_t = \varphi\mathsf{X}_{t-1} + \varepsilon_t + \vartheta\varepsilon_{t-1}$ with $\varphi = 0.7$, $\vartheta = 0.8$ and $\varepsilon_j \sim \mathcal{N}(0, 0.4)$. Note that here there is no clear cut-off point, suggesting that moving average behaviour is present in the observations.

2.2.2 Time series models for non-stationary processes

Suppose that the process $(\mathsf{Y}_t)_{t\in\mathbb{Z}}$ is not stationary, such that the autocovariance (and autocorrelation) depends not just on the lag, but also on the time. The motivation behind the model introduced in this section is that, in some cases, applying differencing to $(\mathsf{Y}_t)_{t\in\mathbb{Z}}$ can result in a stationary process. In this case, to obtain a stationary process consider $\mathsf{Y}_t - \mathsf{Y}_{t-1} = (1-B)\mathsf{Y}_t$, or more generally the d^{th} difference $(1-B)^d\mathsf{Y}_t$. The model considered here is known as the integrated autoregressive-moving average process, and is equivalent to the autoregressive-moving average process discussed in Section 2.2.1 after taking d differences of the process.

2.2.2.1 Integrated autoregressive-moving average process

In this section we introduce a process with the same form as the autoregressive-moving average process discussed in Section 2.2.1, but which is not stationary due to the roots of the polynomial operators. That is, an ARIMA(p, d, q) process $(\mathsf{Y}_t)_{t\in\mathbb{Z}}$ has the form

$$\varrho(B)\mathsf{Y}_t = \vartheta(B)\varepsilon_t,$$

where $\varrho(x) = 0$ has d roots that lie on the unit circle. Alternatively the form of $(\mathsf{Y}_t)_{t\in\mathbb{Z}}$ can be given as

$$\varphi(B)(1-B)^d\mathsf{Y}_t = \vartheta(B)\varepsilon_t, \tag{2.10}$$

where for stationarity $\varphi(x) = 0$ has roots outside the unit circle, and for invertibility $\vartheta(x) = 0$ has roots outside the unit circle. Then, $(\mathsf{X}_t)_{t\in\mathbb{Z}}$, with $\mathsf{X}_t = (1-B)^d\mathsf{Y}_t$, is a stationary, invertible ARMA(p, q) process of the form $\varphi(B)\mathsf{X}_t = \vartheta(B)\varepsilon_t$.

Example 2.2.6 (Integrated autoregressive-moving average process, order $(1, 1, 1)$)**.** *In this example $(\mathsf{Y}_t)_{t\in\mathbb{Z}}$ is an integrated autoregressive-moving average process which has the form,*

$$(1-B)\mathsf{Y}_t = \varphi(1-B)\mathsf{Y}_{t-1} + \varepsilon_t + \vartheta\varepsilon_{t-1},$$

with $\varphi = 0.7$, $\vartheta = 0.8$ and $\varepsilon_j \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ for $j \in \mathbb{Z}$, $\sigma_\varepsilon^2 = 0.4$. The first difference $\mathsf{X}_t = (1-B)\mathsf{Y}_t$ is an ARMA$(1, 1)$ process. A realisation $\{y_1, y_2, \ldots, y_{250}\}$ of the *process is shown in Figure 2.11.*

Using this realisation, the sample autocorrelation function is calculated and given in Figure 2.12, and shows a very slow delay. This suggests that the process is not stationary, and therefore differences should be taken. When first differences are taken the realisation $\{x_1, x_2, \ldots, x_{250}\}$ *of the process* $(\mathsf{X}_t)_{t\in T}$ *is obtained, which is the realisation we discussed in Example 2.2.5 and is shown in Figure 2.8. Looking at the sample autocorrelation of the differenced process in Figure 2.9 the decay is quite fast, which suggests that the differenced process is stationary.*

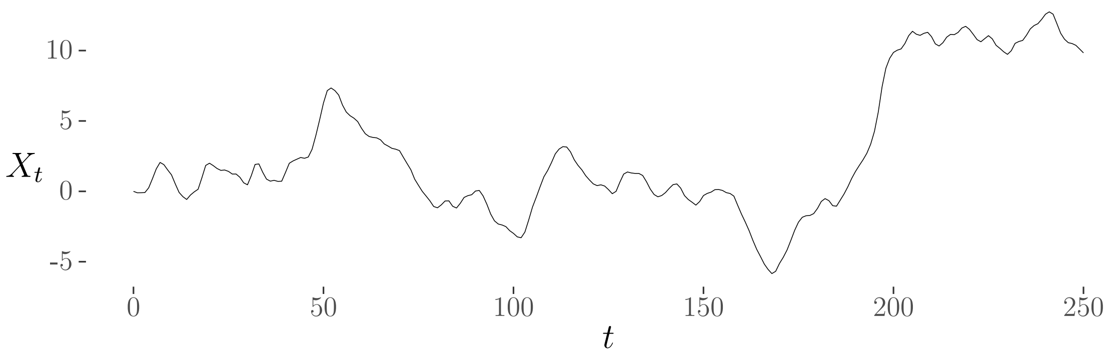


Figure 2.11: Time series of a realisation $\{y_1, y_2, \ldots, y_{250}\}$ of an integrated autoregressive-moving average process of order $(1,1,1)$, $(\mathsf{Y}_t)_{t\in\mathbb{Z}}$, of the form $(1-B)\mathsf{Y}_t = \varphi(1-B)\mathsf{Y}_{t-1} + \varepsilon_t + \vartheta\varepsilon_{t-1}$ with $\varphi = 0.7$, $\vartheta = 0.8$ and $\varepsilon_j \sim \mathcal{N}(0, 0.4)$.

Now that some discrete causal classical time series models have been introduced, more novel work is given in Section 2.3 below. This includes stochastic processes defined on continuous time and stochastic processes integrated over a set of time intervals.

2.3 Ornstein-Uhlenbeck Processes

We now consider stochastic differential equations, which have solutions that are a case of continuous time stochastic processes. We give a short introduction to

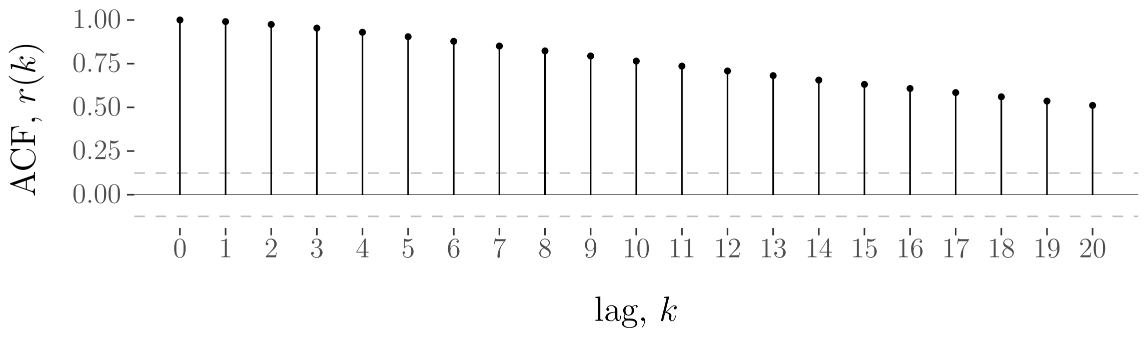


Figure 2.12: Sample autocorrelation function of a realisation $\{y_1, y_2, \ldots, y_{250}\}$ of an autoregressive process of order 1, $(\mathsf{Y}_t)_{t\in\mathbb{Z}}$, of the form $(1-B)\mathsf{Y}_t = \varphi(1-B)\mathsf{Y}_{t-1} + \varepsilon_t + \vartheta\varepsilon_{t-1}$ with $\varphi = 0.7$, $\vartheta = 0.8$ and $\varepsilon_j \sim \mathcal{N}(0, 0.4)$. Here there is very slow decay of autocorrelation as a function of the lag, which suggests that the process is not stationary. Taking first differences gives the process shown in Figure 2.8 with sample autocorrelation function and sample partial autocorrelation function shown in Figure 2.9 and Figure 2.10 respectively.

stochastic differential equations in Section 2.3.1. We then define the Ornstein-Uhlenbeck process in Section 2.3.2, and by averaging over some disjoint time intervals, we obtain the block-average Ornstein-Uhlenbeck process as introduced in Section 2.3.2.1.

Once these stochastic differential equations have been described, we introduce some parameter estimation methods. These are split into parameter estimations methods for pointwise stochastic differential equations (Section 2.3.3.1), and for block-average stochastic differential equations (Section 2.3.3.3).

2.3.1 Introduction to stochastic differential equations

A brief introduction to stochastic differential equations is now given. For a more comprehensive overview, the reader is referred to Da Prato & Zabczyk (2014); Iacus (2009); Lindsey (2004); Øksendal (2013). The most common example of a stochastic process in continuous time is Brownian motion. Let $x \in \mathbb{R}^n$ and,

$$p(t, x, y) = \frac{1}{(2\pi t)^{n/2}} \exp\left(-\frac{|x-y|}{2t}\right),$$

for $y \in \mathbb{R}^n$ and $t > 0$. Then there exists a probability space $(\Omega, \mathcal{F}, \mathsf{P}^x)$ and a stochastic process $(\mathsf{B}_t)_{t \geq 0}$ on Ω such that, for all $F_1, \ldots, F_k \in \mathcal{F}$,

$$\mathsf{P}^x(\mathsf{B}_{t_1} \in F_1, \mathsf{B}_{t_2} \in F_2, \ldots, \mathsf{B}_{t_k} \in F_k)$$
$$= \int_{F_1 \times F_2 \times \cdots \times F_k} p(t_1, x, x_1)\, p(t_2 - t_1, x_1, x_2) \ldots p(t_k - t_{k-1}, x_{k-1}, x_k)\, \mathrm{d}x_1\, \mathrm{d}x_2 \ldots \mathrm{d}x_k.$$

Here, $(\mathsf{B}_t)_{t \geq 0}$ is known as Brownian motion, and satisfies the following conditions,

1. $\mathsf{B}_0 = 0$ almost surely;
2. Increments: Let $0 \leq s_1 < t_1 \leq s_2 < t_2 \leq \ldots \leq s_k < t_k$ such that $[s_j, t_j]$, $j = 1, 2, \ldots, k$ are non-overlapping intervals. Then,
   (a) Independent increments: $\{(\mathsf{B}_{t_j} - \mathsf{B}_{s_j}); j = 1, 2, \ldots, k\}$ are independent random variables;
   (b) Stationary increments: Let $t_0 \in [0, \infty)$ and $0 \leq s < t < \infty$. Then the distribution of the random variable $\big(\mathsf{B}_{t_0+t} - \mathsf{B}_{t_0+s}\big)$ is independent of t_0;
   (c) Mean and variance: For $0 \leq s < t < \infty$, $\mathsf{E}\big(\mathsf{B}_t - \mathsf{B}_s\big) = 0$ and $\mathrm{Var}\big(\mathsf{B}_t - \mathsf{B}_s\big) < \infty$.
3. Continuous path: the path $t \to \mathsf{B}_t$ is continuous almost surely.

Let the stochastic differential equation take the form

$$\mathrm{d}\mathsf{X}_t = b(t, \mathsf{X}_t)\, \mathrm{d}t + a(t, \mathsf{X}_t)\, \mathrm{d}\mathsf{B}_t \tag{2.11}$$

which has the solution $\mathsf{X}_t = \mathsf{X}_0 + \int_0^t b(s, \mathsf{X}_s)\, \mathrm{d}s + \int_0^t a(s, \mathsf{X}_s)\, \mathrm{d}\mathsf{B}_s$, where $\int_0^t a(s, \mathsf{X}_s)\, \mathrm{d}\mathsf{B}_s$ is a stochastic integral. A strong solution is a solution X_t that is adapted to the filtration and constructed from a version of Brownian motion B_t that is given in advance (Øksendal, 2013, Section 5.3).

The transition density of a stochastic process is given by the function $p\colon [0, \infty) \times [0, \infty) \times \mathbb{R}^2 \to [0, \infty)$ where,

$$p(s, t, x, y) = \langle \text{density of } \mathsf{X}_t \text{ at } y, \text{ given } \mathsf{X}_s = x \rangle.$$

If the equality $p(s,t,x,y) = p(0,t-s,x,y)$ holds for all $s,t \in \mathbb{R}^+$, $t \geq s$, then the stochastic process is time-homogeneous and for simplicity the notation $p(\Delta t, x, y)$, where $\Delta t = t - s$, is used for the transition density. In the case of Brownian motion described above, the transition density $p(\Delta t, x, y)$ is given by the density of a Gaussian with variance Δt, such that non-overlapping increments are independently Gaussian distributed, $(\mathsf{B}_t - \mathsf{B}_s) \sim \mathcal{N}(0, t-s)$, for $s < t$.

We now focus our attention on a specific stochastic differential equation known as the Ornstein-Uhlenbeck process. This process and the variations described in Section 2.3.2 are used to model the components obtained by dimension reduction on the HadCM3 simulator outputs given in Part II.

2.3.2 Introduction to the Ornstein-Uhlenbeck process

The main stochastic process we consider in this section is the Ornstein-Uhlenbeck process (Uhlenbeck & Ornstein, 1930). Here, the process $\mathsf{X}_t^{\mathrm{OU}} \colon t \to \mathbb{R}$ is given by the stochastic differential equation,

$$\mathrm{d}\mathsf{X}_t^{\mathrm{OU}} = -\gamma \mathsf{X}_t^{\mathrm{OU}}\,\mathrm{d}t + \sigma\,\mathrm{d}\mathsf{B}_t, \tag{2.12}$$

where $\gamma \in \mathbb{R}$, $\sigma \in \mathbb{R}^+$, and $\mathsf{B}_t \colon t \to \mathbb{R}$ is Brownian motion.

The Ornstein-Uhlenbeck process is a random process with γ known as the drift parameter and σ^2 the diffusion parameter, and has solution (Karatzsas & Shreve, 1988),

$$\mathsf{X}_t^{\mathrm{OU}} = \mathsf{X}_0^{\mathrm{OU}} e^{-\gamma t} + \sigma \int_0^t e^{-\gamma(t-s)}\,\mathrm{d}\mathsf{B}_s, \quad t \in [0, \infty),$$

where $\mathsf{X}_0^{\mathrm{OU}}$ is the initial random variable of the process. If $\mathsf{E}\big((\mathsf{X}_0^{\mathrm{OU}})^2\big) < \infty$ then from Karatzsas & Shreve (1988) the mean, variance and covariance functions are given by

$$\mu(t) := \mathsf{E}\mathsf{X}_t^{\mathrm{OU}} = \mathsf{E}\mathsf{X}_0^{\mathrm{OU}}\, e^{-\gamma t}; \tag{2.13a}$$

$$V(t) := \mathrm{Var}(\mathsf{X}_t^{\mathrm{OU}}) = \frac{\sigma^2}{2\gamma} + \Big(\mathrm{Var}\,\mathsf{X}_0^{\mathrm{OU}} - \frac{\sigma^2}{2\gamma}\Big)e^{-2\gamma t}; \tag{2.13b}$$

$$\rho(s,t) := \mathrm{Cov}(\mathsf{X}_s^{\mathrm{OU}}, \mathsf{X}_t^{\mathrm{OU}}) = \Big(\mathrm{Var}\,\mathsf{X}_0^{\mathrm{OU}} + \frac{\sigma^2}{2\gamma}(e^{2\gamma\{t\wedge s\}} - 1)\Big)e^{-\gamma(t+s)}. \tag{2.13c}$$

The stationary distribution of an Ornstein-Uhlenbeck process is $\mathcal{N}\big(0, \frac{\sigma^2}{2\gamma}\big)$. If $\mathsf{X}_0^{\mathrm{OU}} = x$ almost surely, then the transition density $p(t, x, \cdot)$ is from the Gaussian distribution $\mathcal{N}\big(xe^{-\gamma t}, \frac{\sigma^2}{2\gamma}(1 - e^{-2\gamma t})\big)$.

2.3.2.1 The block-average Ornstein-Uhlenbeck process

In our application in Part II, data will be reported as monthly averages of a stochastic process. In order to get a model for this situation, we consider the behaviour of averages of an Ornstein-Uhlenbeck process within certain disjoint time intervals. In this case, the integral of the Ornstein-Uhlenbeck process can be used to model the observations. The output of the HadCM3 simulator in Part II is given as monthly means of some variable over (a discretised version of) the Earth's surface, and therefore this is the motivation for considering the behaviour of these integrals. It is worth nothing here that the behaviour of some similar Ornstein-Uhlenbeck integral-type processes have been previously described in Lindsey (2004); Taylor *et al.* (1994), although the author was not aware of these references until after the work on this section had been completed, and indeed these references have very difference aims and focuses then those obtained using the motivation here.

Consider the integral of an Ornstein-Uhlenbeck process within some partition, where $\mathsf{X}^{\mathrm{OU}} \in \mathrm{L}^2([0,T], \mathbb{R})$ is the (random) Ornstein-Uhlenbeck process, and $0 = t_1 < t_2 < t_3 < \cdots < t_n = T$ is the time partition. That is, for each $i \in \{2, 3, \ldots, n\}$, the quantity of interest is,

$$\mathsf{X}_{[t_{i-1},t_i)}^{\mathrm{baOU}} := \frac{1}{t_i - t_{i-1}} \int_{t_{i-1}}^{t_i} \mathsf{X}_t^{\mathrm{OU}}\,\mathrm{d}t = \frac{1}{t_i - t_{i-1}} \langle \mathbb{1}_{[t_{i-1},t_i)}, \mathsf{X}^{\mathrm{OU}} \rangle. \tag{2.14}$$

We now investigate the distribution of the random variables $\mathsf{X}^{\mathrm{baOU}}_{[t_{i-1},t_i)}$, $i = 2, 3, \dots, n$. First note that the standard Ornstein-Uhlenbeck process is a Gaussian process and therefore for any set of time-points $\tau_1, \tau_2, \dots, \tau_K \in [0, T]$ the random vector $(\mathsf{X}^{\mathrm{OU}}_{\tau_1}, \mathsf{X}^{\mathrm{OU}}_{\tau_2}, \dots, \mathsf{X}^{\mathrm{OU}}_{\tau_K})^\top$ has a Gaussian distribution with some mean m and covariance matrix Σ. Here we give the definition of a Gaussian measure on the Hilbert space, and then we use a lemma from Rajput & Cambanis (1972) to show that the law of $\mathsf{X}^{\mathrm{OU}} \in \mathrm{L}^2([0, T], \mathbb{R})$ is Gaussian. This result, alongside (2.15) given below, is used to give the distribution of the block-average Ornstein-Uhlenbeck process.

Definition 2.3.1 (Gaussian Measure on the Hilbert Space (Da Prato & Zabczyk, 2014))**.** *A probability measure μ on $(H, \mathcal{B}(H))$ is called Gaussian if for any $h \in H$, there exists an $m \in \mathbb{R}$, $q \geq 0$ such that*

$$\mu\big\{x \in H\colon \langle h, x\rangle \in A\big\} = \mathcal{N}(m, q)(A),$$

for any $A \in \mathcal{B}(\mathbb{R})$, where $\mathcal{N}(a, b)(C)$ is the integral of the density function over C of a Normal distribution with mean a and variance b.

Let $(\mathrm{L}^2([0, T]), \mathcal{B}(\mathrm{L}^2([0, T], \mathbb{R})), \mathrm{Pr})$ be a probability space and let $\mathsf{X} \in \mathrm{L}^2([0, T], \mathbb{R})$ be a random process. Then, the law of X is Gaussian if

$$\mathrm{Pr}\big(\mathsf{X}^{-1}(B)\big) = \mu(B),$$

for any $B \in \mathcal{B}(\mathrm{L}^2([0, T], \mathbb{R}))$, where μ is a Gaussian measure as defined in Definition 2.3.1. In this case, for any function $f \in \mathrm{L}^2([0, T], \mathbb{R})$,

$$\langle \mathsf{X}, f\rangle \sim \mathcal{N}(m_f, q_f) \tag{2.15}$$

for some $m_f \in \mathbb{R}$ and $q_f \geq 0$.

Lemma 2.3.2. *Let $\mathsf{X}^{OU}_t\colon [0, T] \to \mathbb{R}$ be an Ornstein-Uhlenbeck process as in* (2.12)*. Then the law of the random process $\mathsf{X}^{OU} \in L^2([0, T], \mathbb{R})$ is Gaussian.*

Proof. See Rajput & Cambanis (1972) for proof. □

Now, writing the block-average Ornstein-Uhlenbeck process as an inner product, as in the second equality of (2.14), and using Lemma 2.3.2, from (2.15) the expectation and covariance of the block-average process is given by

$$\mathsf{E}\big(\langle \mathsf{X}^{\mathrm{OU}}, \mathbb{1}_{[a,b]}\rangle\big) = \mathsf{E}\Big(\int_a^b \mathsf{X}^{\mathrm{OU}}_t \,\mathrm{d}t\Big) = 0; \tag{2.16}$$

$$\mathrm{Cov}\big(\langle \mathsf{X}^{\mathrm{OU}}, f\rangle, \langle \mathsf{X}^{\mathrm{OU}}, g\rangle\big) = \langle f, \mathcal{C}g\rangle, \tag{2.17}$$

for any $a, b \in [0,T]$ and $f, g \in \mathrm{L}^2([0,T],\mathbb{R})$. Here, the covariance operator is given by

$$\big(\mathcal{C}g\big)(s) = \int_0^T c(s,t)g(t)\,\mathrm{d}t,$$

for any $g \in \mathrm{L}^2([0,T],\mathbb{R})$. For a stationary Ornstein-Uhlenbeck process (*i.e.* X^{OU} as in (2.12) with $\mathsf{X}^{\mathrm{OU}}_0 \sim \mathcal{N}(0, \frac{\sigma^2}{2\gamma})$), the covariance function $c\colon [0,T]\times[0,T] \to \mathbb{R}$ is

$$c(s,t) = \frac{\sigma^2}{2\gamma} e^{-\gamma|t-s|},$$

for any $s, t \in [0,T]$.

Consider (2.15), Lemma 2.3.2 and the expectation and covariance of an block-average Ornstein-Uhlenbeck process as in (2.16) and (2.17) respectively. Then, the vector $(\mathsf{X}^{\mathrm{baOU}}_{[t_1,t_2)}, \mathsf{X}^{\mathrm{baOU}}_{[t_2,t_3)}, \ldots, \mathsf{X}^{\mathrm{baOU}}_{[t_{n-1},t_n)}) \in \mathbb{R}^{n-1}$, with each $\mathsf{X}^{\mathrm{baOU}}_{[t_{i-1},t_i)}$ given as in (2.14), is a realisation from $\mathcal{N}(0,\Sigma)$, where $\Sigma = (\sigma_{ij})_{i,j=1,2,\ldots,n-1}$, and

$$\sigma_{ij} = \langle \mathbb{1}_{[t_{i-1},t_i)}, \mathcal{C}\mathbb{1}_{[t_{j-1},t_j)}\rangle = \int_{t_{i-1}}^{t_i}\int_{t_{j-1}}^{t_j} \frac{\sigma^2}{2\gamma} e^{-\gamma|t-s|}\,\mathrm{d}t\,\mathrm{d}s.$$

Note that, for $a, b, c, d \in \mathbb{R}$ with $a < b \leq c < d$,

$$\begin{aligned}
\langle \mathbb{1}_{[a,b)}, \mathcal{C}\mathbb{1}_{[c,d)} \rangle &= \int_a^b \int_c^d \frac{\sigma^2}{2\gamma} e^{-\gamma|t-s|} \, \mathrm{d}t \, \mathrm{d}s \\
&= \int_a^b \int_c^d \frac{\sigma^2}{2\gamma} e^{-\gamma(t-s)} \, \mathrm{d}t \, \mathrm{d}s, \quad \text{as } a < b \leq c < d \\
&= \frac{\sigma^2}{2\gamma} \int_a^b e^{\gamma s} \, \mathrm{d}s \int_c^d e^{-\gamma t} \, \mathrm{d}t \\
&= -\frac{\sigma^2}{2\gamma^3} \big(e^{-\gamma d} - e^{-\gamma c}\big)\big(e^{\gamma b} - e^{\gamma a}\big).
\end{aligned}$$

Only considering the interval $[a, b)$ for some $a, b \in \mathbb{R}$ gives,

$$\begin{aligned}
\langle \mathbb{1}_{[a,b)}, \mathcal{C}\mathbb{1}_{[a,b)} \rangle &= \frac{\sigma^2}{2\gamma} \Big(\int_a^b \int_a^s e^{-\gamma(s-t)} \, \mathrm{d}t \, \mathrm{d}s + \int_a^b \int_s^b e^{-\gamma(t-s)} \, \mathrm{d}t \, \mathrm{d}s \Big) \\
&= \frac{\sigma^2}{2\gamma} \Big(\int_a^b 1 - e^{-\gamma(s-a)} \, \mathrm{d}s - \int_a^b e^{-\gamma(b-s)} - 1 \, \mathrm{d}s \Big) \\
&= \frac{\sigma^2}{2\gamma} \Big(b + \frac{1}{\gamma} e^{-\gamma(b-a)} - a - \frac{1}{\gamma} - \big(\frac{1}{\gamma} - b - \frac{1}{\gamma} e^{-\gamma(b-a)} + a \big) \Big) \\
&= \frac{\sigma^2}{\gamma^3} \big(e^{-\gamma(b-a)} + \gamma(b-a) - 1 \big).
\end{aligned}$$

For the case of equal length time intervals being used such that the time-steps are given by $t_j = j\Delta t$ for all $j \in \mathbb{Z}^+$ and $\Delta t > 0$, the elements of the covariance matrix becomes,

$$\sigma_{ij} = \begin{cases} \dfrac{\sigma^2}{2\gamma^3} e^{-\gamma|j-i|\Delta t} \big(e^{-\gamma\Delta t} + e^{\gamma\Delta t} - 2 \big) & \text{if } j \neq i; \\ \dfrac{\sigma^2}{\gamma^3} \big(e^{-\gamma\Delta t} + \gamma\Delta t - 1 \big) & \text{if } j = i. \end{cases}$$

Therefore the covariance matrix of the process $(\mathsf{X}^{\mathrm{baOU}}_{[(j-1)\Delta t, j\Delta t)})_{j\in\mathbb{Z}}$ is a symmetric

Toeplitz matix of the form $\Sigma = (\sigma_{ij})_{i,j}$ with $\sigma_{ij} = c\, a^{|j-i|} + b\mathbb{1}_{i=j}$, for

$$c = \frac{\sigma^2}{2\gamma^3}\big(e^{-\gamma\Delta t} + e^{\gamma\Delta t} - 2\big); \tag{2.18a}$$

$$a = e^{-\gamma\Delta t}; \tag{2.18b}$$

$$b = \frac{\sigma^2}{2\gamma^3}\big(e^{-\gamma\Delta t} - e^{\gamma\Delta t} + 2\gamma\Delta t\big). \tag{2.18c}$$

Now that we have introduced the main stochastic differential equation used in this thesis alongside its block-averaged form, our focus moves on to estimation of the Ornstein-Uhlenbeck parameters given some realisations. These realisations can either be from the pointwise Ornstein-Uhlenbeck process, or integrated over some time intervals as discussed above.

2.3.3 Parameter fitting procedures

Here we consider estimating the drift and diffusion terms using some realisations from a stochastic differential equation of the form given in (2.11). This section predominantly considers the Ornstein-Uhlenbeck process and its block-averaged form as described in Section 2.3.2, although first a more general case is examined.

This section is split into two main parts, mirroring the split between pointwise Ornstein-Uhlenbeck processes discribed in Section 2.3.2 and the block-averaged form in Section 2.3.2.1. In Section 2.3.3.1, we introduce parameter estimation methods for pointwise realisations from a stochastic differential equation. This is split into three main methods; the "partitioning" method, maximum likelihood estimation, and method of moments. These methods will be compared here on toy examples, before being selectively applied to the simulator outputs in Part II. Although the assumptions on the realisations in this section are not satisfied by the HadCM3 simulator outputs in Part II (as these realisations are averages), parameter fitting of pointwise processes are considered here for completeness. In Section 2.3.3.3 we introduce novel parameter estimation using maximum likelihood for realisations from a block-average Ornstein-Uhlenbeck process as described in Section 2.3.2.1. The simulator outputs given in Part II are indeed some time-averaged data and therefore the methods given in Section 2.3.3.3 can be applied in Part II.

First we give a brief remark on the difference and interdependence between model structure and parameter estimation.

Remark 2.11: Model structure and parameter estimation

Before looking into the parameter estimation methods, it is worth making a short remark on the difference between the *structure* and the *parameter estimation* of a stochastic differential equation. In this section parametric stochastic differential equations are considered, and therefore prior to any estimation it is required to specify the structure of the process. For example, the Ornstein-Uhlenbeck process given in (2.12) is a specific choice of a stochastic differential equation which exhibits specific behaviours. A modeller is required to choose the model before any parameter estimation can be carried out, and this choice should be justifiable.

On top of this, it is important to distinguish the type of observations that one has and the way they are modelled. For the simulated climate data that is discussed in Part II, the observations are *monthly means*. This is an important consideration to take into account, as it is the difference between observing a $\mathsf{X}^{\mathrm{OU}}_{t_i}$, where t_i is the beginning of the month (say), and observing the monthly mean $\mathsf{X}^{\mathrm{baOU}}_{[t_{i-1},t_i)}$ as in (2.14). The two processes give observations $\{\mathsf{X}^{\mathrm{OU}}_{t_1}, \mathsf{X}^{\mathrm{OU}}_{t_2}, \ldots\}$ and $\{\mathsf{X}^{\mathrm{baOU}}_{[t_1,t_2)}, \mathsf{X}^{\mathrm{baOU}}_{[t_2,t_3)}, \ldots\}$, with different distributions to optimise to obtain the parameter estimations. An example of misspecifying the model in this way is given at the end of this chapter in Example 2.3.10.

2.3.3.1 Parameter estimation: pointwise observations

In this section we consider estimation methods for when pointwise observations of a stochastic process are available, such that the observation at time t is exactly the current state of the process.

In general, there are two different limits in which estimated parameters can approach the true values. Given some set of realisations $\{x_{t_1}, x_{t_2}, \ldots, x_{t_n}\}$, with $t_i - t_{i-1} = \Delta t$, $i = 2, \ldots, n$, and $t_n = T$, the first considers the case $\Delta t \to 0$ for some fixed T, whilst the second considers $n \to \infty$ for some fixed Δt (*i.e.* $T \to \infty$). If we assume the principal and independent components obtained from the simulator climate data in Part II are sparse observations from a stochastic differential equation, then the focus here is on the estimation methods that are consistent in the limit of $n \to \infty$ with Δt constant. However, both scenarios will be discussed for completeness.

This section is arranged as follows. We initially consider a simple method, here called the partitioning method, that finds piecewise constant approximations for the drift and diffusion terms in (2.11). This method is most effective when it is known that the true drift and diffusion terms are both functions of the form $f(x) = \sum_{i=1}^{P} f_i \mathbb{1}_{x \in \lambda_i}$, and $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_P\}$ is a known partitioning of the state space of the process. The partitioning method can then be used to find approximations of the numbers f_i, $i = 1, 2, \ldots, P$.

Next, we discuss estimation methods to determine the parameters of an Ornstein-Uhlenbeck process $\mathsf{X}_t^{\mathrm{OU}}$ given in (2.12). In this case the drift and diffusion terms, γ and σ^2, are unknown parameter values to be determined.

Lastly, we simulate a test set of realisations of Ornstein-Uhlenbeck stochastic differential equations with varying γ and σ^2 values, and the Ornstein-Uhlenbeck parameter estimation methods are compared to each-other by considering the approximations $\hat{\gamma}$ and $\hat{\sigma}^2$.

Partitioning method

One of the simplest method for approximating the functions b and a in (2.11) is by partitioning the state space and modelling b and a as piecewise functions that are constant within each partition. These approximations are found by simple arithmetic formula that describe the behaviour of neighbouring observations.

Suppose the observations $\{x_0, x_{\Delta t}, x_{2\Delta t}, \ldots, x_{n\Delta t}\}$ are a realisation of the process $(\mathsf{X}_t)_{t \geq 0}$ with image $\mathcal{X}$. Let $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_P\}$ be a partition of $\mathcal{X}$ such that

$\bigcup_{j=1}^{P} \lambda_j = \mathcal{X}$ and $\lambda_j \cap \lambda_k = \emptyset$ for all $j, k \in \{1, 2, \ldots, P\}$ with $j \neq k$, ensuring that there is at least one observation within each partition.

To calculate the approximate drift and diffusion terms, we consider each partition λ_j, $j \in \{1, 2, \ldots, P\}$, separately and examine the Δt-time behaviour of observations belonging to λ_j at time $k\Delta t$ for $k \in \{0, 1, \ldots, n-1\}$. The approximation to the drift function b is given by $\hat{b}(x) = \sum_{j=1}^{P} \mathbb{1}_{\{x \in \lambda_j\}} \hat{b}_j$, where

$$\hat{b}_j = \frac{\sum_{i=0}^{n-1} \mathbb{1}_{\{x_{i\Delta t} \in \lambda_j\}} (x_{(i+1)\Delta t} - x_{i\Delta t})}{\Delta t \sum_{i=0}^{n-1} \mathbb{1}_{\{x_{i\Delta t} \in \lambda_j\}}}. \tag{2.19}$$

The approximation to the diffusion function a is the quadratic variation approximation of the observations and is given by $\hat{a}(x) = \sum_{j=1}^{P} \mathbb{1}_{\{x \in \lambda_j\}} \hat{a}_j$, where

$$\hat{a}_j^2 = \frac{\sum_{i=0}^{n-1} \mathbb{1}_{\{x_{i\Delta t} \in \lambda_j\}} (x_{(i+1)\Delta t} - x_{i\Delta t})^2}{\Delta t \sum_{i=0}^{n-1} \mathbb{1}_{\{x_{i\Delta t} \in \lambda_j\}}}.$$

This method clearly suffers from the 'curse of dimensionality', as for even quite low dimensions it becomes impossible to partition the space such that there are enough observations in each partition to estimate $\hat{b}$ and $\hat{a}$ to a good degree of accuracy. However, it is possible to apply the method and examine its accuracy easily for lower dimensional examples. This is done below. First, we introduce the Euler-Maruyama method in Remark 2.12, which is used throughout this thesis to approximate trajectories of stochastic differential equations.

Remark 2.12: Euler-Maruyama method

The Euler-Maruyama method is one of the simplest and most common methods used to obtain a time discrete approximation of a stochastic differential equation. Suppose the stochastic differential equation has the form given in (2.11), *i.e.*

$$\mathrm{d}\mathsf{X}_t = b(t, \mathsf{X}_t)\,\mathrm{d}t + a(t, \mathsf{X}_t)\,\mathrm{d}B_t,$$

with the initial value $\mathsf{X}_0 = \mathsf{X}_{t_1}$.

Let $t_1 < t_2 < \cdots < t_n = T$ be some discretisation of the time interval $[t_1, T]$. Then, the Euler-Maruyama approximation is a continuous time stochastic process $(\mathsf{Y}_t)_{t\in[t_1,T]}$ which satisfies the iterative scheme,

$$\mathsf{Y}_{t_{i+1}} = \mathsf{Y}_{t_i} + b\big(t_i, \mathsf{Y}_{t_i}\big)(t_{i+1} - t_i) + a\big(t_i, \mathsf{Y}_{t_i}\big)(B_{t_{i+1}} - B_{t_i}),$$

for $i = 0, 1, \ldots, n-1$ with $\mathsf{Y}_0 = \mathsf{X}_{t_1}$ (Kloeden & Platen, 2013).

This method is used throughout this section in examples where stochastic differential equations are approximated.

Example 2.3.3 (Partitioning method in 2-dimensions). *In this example the partitioning method is applied to a two dimensional trajectory to estimate the drift term of a stochastic process. Let X_t be a stochastic process given by the differential equation,*

$$d\mathsf{X}_t = \big(R - \varphi'(\sqrt{\mathsf{X}_t^\top \mathsf{X}_t})\big)\mathsf{X}_t\, dt + \sigma\, dB_t \tag{2.20}$$

where,

$$R = \begin{pmatrix} \cos(\vartheta) & -\sin(\vartheta) \\ \sin(\vartheta) & \cos(\vartheta) \end{pmatrix}; \quad \varphi(x) = x + \frac{1}{\gamma x}$$

for some $\vartheta \in [0, \pi), \gamma \in \mathbb{R}$ and $\sigma^2 \in (0, \infty)$.

An approximate trajectory at times $0 = t_1 < t_2 < \cdots < t_n = T$, with $t_i - t_{i-1} = \Delta t$, $i \in \{2, \ldots, n\}$, is obtained using Euler-Maruyama approximation. That is, the stochastic differential equation given by $(\mathsf{X}_t)_{t\in[0,T]}$ is approximated by the iterative scheme given by

$$\mathsf{Y}_{t_{i+1}} = \mathsf{Y}_{t_i} + \big(R - \varphi'(\sqrt{\mathsf{Y}_t}^\top \mathsf{Y}_{t_i})\big)\mathsf{Y}_{t_i}\Delta t + \sigma \cdot (\mathsf{B}_{t_{i+1}} - \mathsf{B}_{t_i}), \tag{2.21}$$

for $i = 1, \ldots, n-1$ and for some initial Y_{t_1}.

Figure 2.13 gives an approximate trajectory of the stochastic differential equa-

tion (2.20) using the Euler-Maruyama approximation (2.21), with $\vartheta = \frac{\pi}{2}$, $\gamma = 3$ and $\sigma^2 = 1$. The space is partitioned into 16 equal sized squares and the drift function $\big(R - \varphi'(\sqrt{\mathsf{X}_t^\top \mathsf{X}_t})\big)\mathsf{X}_t$ is approximated using (2.19). The estimated size and direction of the drift $\hat{b}_j$ at each partition $j \in \{1, 2, \ldots, 16\}$ are shown by the black arrows in Figure 2.13, with the true drift values at the centre of each partition given by the grey arrows. In general these approximations are relatively good, although this accuracy depends strongly on the amount of data available in each partition. This is clearly an issue for higher dimensional stochastic processes.

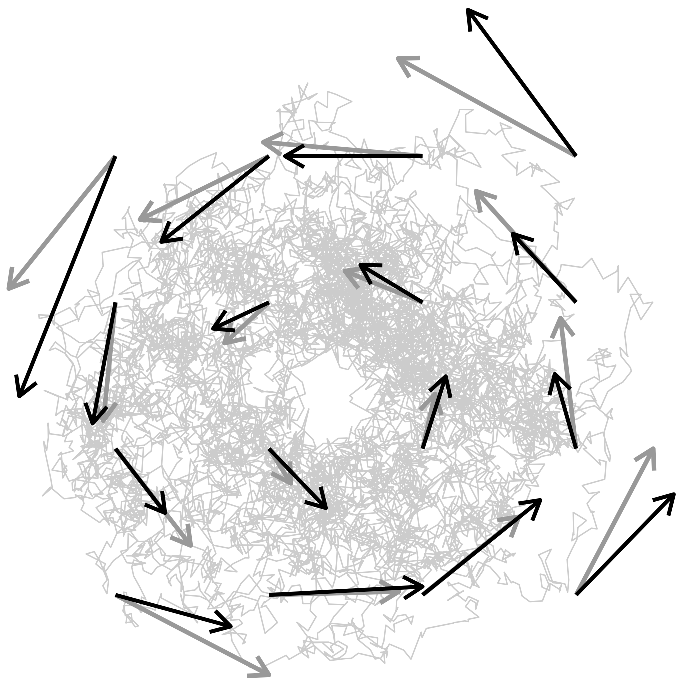


Figure 2.13: Figure showing approximate stochastic differential equation given by the Euler-Maruyama approximation (2.21) with $\vartheta = \pi/2$, $\gamma = 3$ and $\sigma^2 = 1$. The drift function $\big(R - \varphi'(\sqrt{\mathsf{X}_t^\top \mathsf{X}_t})\big)\mathsf{X}_t$ is approximated using the partitioning method by partitioning the space into 16 and calculating $\hat{b}$ as in (2.19). The grey arrows show the true drift at that point, and the black show the estimated $\hat{b}_j$ for each partition $j = 1, \ldots, 16$.

Assessing the quality of the partitioning method approximation. To judge the quality of the partitioning method as described above, consider trajectories of stochastic processes with piecewise constant drift and diffusion. That is, consider the process $(\mathsf{X}_t)_{t\geq 0}$ given by the stochastic differential equation,

$$\mathrm{d}\mathsf{X}_t = b(\mathsf{X}_t)\,\mathrm{d}t + a(\mathsf{X}_t)\,\mathrm{d}B_t$$

with $b(x) = \sum_{i=1}^{P} b_i \mathbb{1}_{x\in\lambda_i}$ and $a(x) = \sum_{i=1}^{P} a_i \mathbb{1}_{x\in\lambda_i}$, and obtain estimates $\hat{b}_j$, $\hat{a}_j$, of b_j and a_j respectively, $j \in \{1, 2, \ldots, P\}$. Here $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_P\}$ is a (known) partition of the image of X_t, $\mathcal{X}$, such that $\bigcup_{i=1}^{P} \lambda_i = \mathcal{X}$ and $\lambda_i \cap \lambda_j = \emptyset$ for all $i \neq j$.

Two examples are given below, both of which highlight issues with using this method. Example 2.3.4 underscores an issue when trajectories do not properly span the state space, resulting in poor approximations in these partitions. This error is magnified when the state space is high dimensional (where 'high' in this case can mean anything as low as three dimensions). Example 2.3.5 emphasises a deficiency of the partitioning method when the time step of the observations is too large.

Example 2.3.4 (Exploring the state space)**.** *In this example the state space is partitioned into six by*

$$\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_6\} = \{(-\infty, -8),\ [-8, -4),\ [-4, 0),\ [0, 4),\ [4, 8),\ [8, \infty)\},$$

with $b = (b_1, \ldots, b_6) = (1, 0.4, -0.5, 0, -0.3, -1)$ *the drift coefficients on this partition. The diffusion function is piecewise constant with only one jump at* $x = 0$ *and takes the values* $a^2 = (a_1^2, \ldots, a_6^2) = (4, 4, 4, 2, 2, 2)$ *in the partition* Λ*.*

The following procedure is repeated 100 *times:*

1. *Euler-Maruyama approximation is used to simulate a realisation of the process described above, from time* $t_1 = 0$ *to* $t_n = 100$ *with* $\Delta t = t_j - t_{j-1} = 10 \times 10^{-4}$, $j = 2, 3, \ldots, n$. *This gives the observations* $\{x_0, x_{\Delta t}, \ldots, x_{n\Delta t}\}$*.*

2. *The approximations for b_j and a_j^2, $j \in \{1, 2, \dots, 6\}$ are obtained from this realisation using the partitioning method.*

In Figure 2.14 we show ten of the 100 *sample trajectories of the process, along with the partition $\Lambda = \{\lambda_1, \dots, \lambda_6\}$ (where the boundaries are given by the dotted lines), and the true drift and diffusion values in each partition.*

The means and standard deviations of the approximations for b_j and a_j^2, $j \in \{1, 2, \dots, 6\}$ for all iterations are shown in Table 2.15. Here it can be clearly seen how the accuracy of the drift approximations decrease in the partitions where the trajectory spend less time, for example $(-\infty, -8)$.

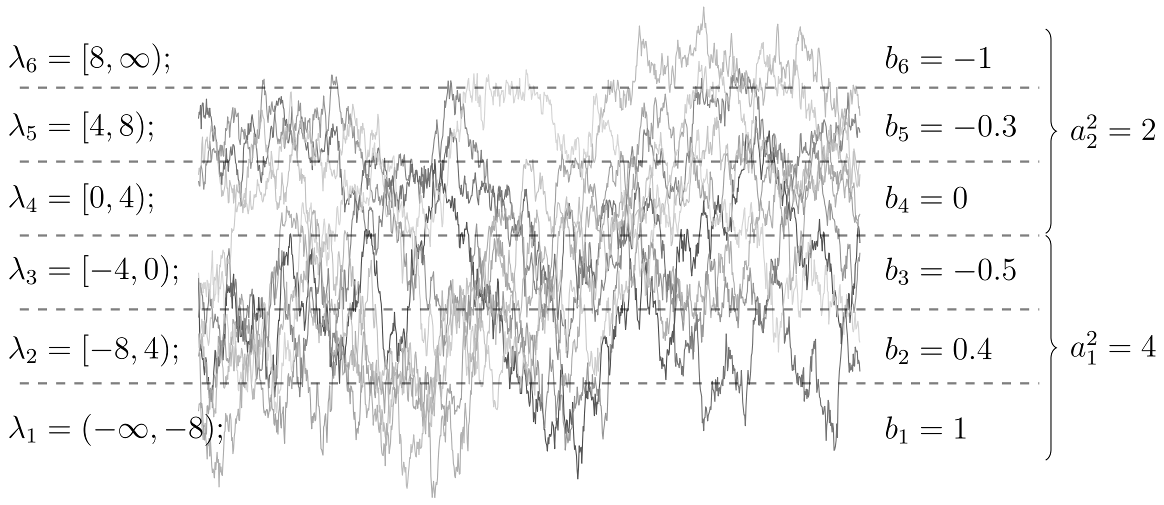


Figure 2.14: A selection of the 100 trajectories of a stochastic process with piecewise constant drift and diffusion functions that take values b_j and a_j on the partition λ_j, $j \in \{1, 2, \dots, 6\}$.

Example 2.3.5 (Varying Δt)**.** *A known weakness of the partitioning method is in the approximation of the diffusion coefficient when the time steps between observations are "large". In this case, the drift term becomes dominant and the*

Partition	b_j	$\bar{b}_j$	$\mathrm{sd}(\bar{b}_j)$	a_j^2	$\bar{a}_j^2$	$\mathrm{sd}(\bar{a}_j^2)$
$(-\infty,-8)$	1.0	2.5992	6.4744	4	4.0573	0.3418
$[-8,-4)$	0.4	0.4809	0.5937	4	4.0063	0.0557
$[-4,0)$	-0.5	-0.4364	0.5346	4	3.9960	0.0493
$[0,4)$	0.0	-0.0894	0.3757	2	1.9981	0.0208
$[4,8)$	-0.3	-0.6159	0.9880	2	2.0079	0.0399
$[8,\infty)$	-1.0	-3.5948	8.1162	2	2.0346	0.2535

Figure 2.15: Table showing the true values b_j, a_j^2 and the mean of the approximations obtained using the partitioning method – along with the standard deviation from the 100 trajectories – that the drift and diffusion functions respectively take on the partition λ_j, $j \in \{1,2,\dots,6\}$.

approximation of quadratic variation that is used to obtain the diffusion term deteriorates.

In this example the state space is partitioned

$$\Lambda = \{\lambda_1,\lambda_2,\lambda_3,\lambda_4\} = \{(-\infty,-4),[-4,0),[0,4),[4,\infty)\},$$

with drift coefficients $b = (b_1,\dots,b_4) = (1, 0.5, -0.5, -1)$ *and diffusion coefficients* $a^2 = (a_1^2,\dots,a_4^2) = (4,4,2,2)$.

Here, the following procedure is repeated 100 *times:*

1. *Euler-Maruyama approximation is used to simulate a trajectory of the process described above, from time* $t_1 = 0$ *to* $t_n = 100$ *with* $\Delta t = t_j - t_{j-1} = 10\times 10^{-4}$, $j = 2,3,\dots,n$. *This gives the observations* $\{x_0, x_{\Delta t},\dots,x_{n\Delta t}\}$.

2. *The dense realisations are thinned such that five trajectories of varying sparsity are obtained, with time steps,* $\Delta t_1 = 0.1$, $\Delta t_2 = 0.6$, $\Delta t_3 = 1.1$, $\Delta t_4 = 1.6$, $\Delta t_5 = 2.1$.

3. *The sparse realisations with time steps* $\Delta t_1, \Delta t_2, \Delta t_3$ *and* Δt_4 *are truncated such that they all have the same length as the observations with time step*

Δt_5. *Thus, the realisations are*

$$\{x_0, x_{\Delta t_i}, x_{2\Delta t_i}, \dots, x_{\tilde{n}\Delta t_i}\}, \quad i \in \{1, 2, \dots, 5\}$$

where $\tilde{n} = n{\cdot}(\Delta t/\Delta t_5)$*, and it is assumed that* $n{\cdot}(\Delta t/\Delta t_5) \in \mathbb{N}$ *for simplicity.*

4. *The partitioning method is applied to each of these five "observed" trajectories independently to give drift and diffusion approximations* $\hat{b}_{ij}$ *and* $\hat{a}^2_{ij}$*,* $j = 1, \dots, 4$ *respectively, for each trajectory with time step* Δt_i*,* $i = 1, \dots, 5$*.*

Figures 2.16 and 2.17 give box plots showing the quality of the drift and diffusion estimates for each thinned observation. The lower and upper hinges of the box plots correspond to the 25th *and* 75th *percentiles of the data, and the whiskers show* 1.5 *times the interquartile range.*

For the thinned observation with time step Δt_i *the associated box plot in Figure 2.16 shows* $|\hat{b}_{ij} - b_j|$ *for* $j = 1, \dots, 4$ *and for each of the* 100 *iterations, giving* 400 *data points. The larger number of outliers when the time step is* $\Delta t_1 = 0.1$ *compared to the larger time steps is most likely due to the time spanned by this trajectory is shorter than the others, and thus for some realisations the state space is not explored as thoroughly.*

For each Δt_i*,* $i = 1, \dots, 5$*, Figure 2.17 shows the box plots for* $|\hat{a}^2_{ij} - a^2_j|$*,* $j = 1, \dots, 4$ *over all iterations. It is clear from these figures that the size of* Δt *is a significant factor in the accuracy of the diffusion approximation when using the partitioning method.*

The principal and independent components obtained on the simulator output in Part II, using the methods discussed in Chapter 1, are one dimensional time series and thus theoretically the partitioning method will work for modelling these components. However, the lack of *a priori* knowledge of any partition of the state space, and the lack of justification for piecewise drift and diffusion means that in

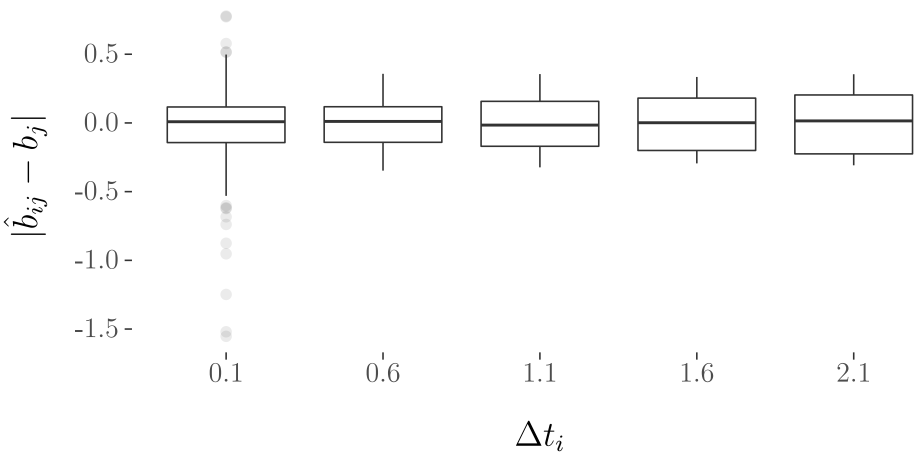


Figure 2.16: Box plot showing the difference in the drift approximations $\hat{b}_{ij}$ and the true values b_j, $j = 1, \ldots, 4$, for each Δt_i value, $i = 1, \ldots, 5$. That is, for each Δt_i, the box plot displays $|\hat{b}_{ij} - b_j|$ for $j = 1, \ldots, 4$ for all 100 iterations. The lower and upper hinges of the boxes correspond to the 25th and 75th percentiles of the data, and the whiskers show 1.5 times the interquartile range.

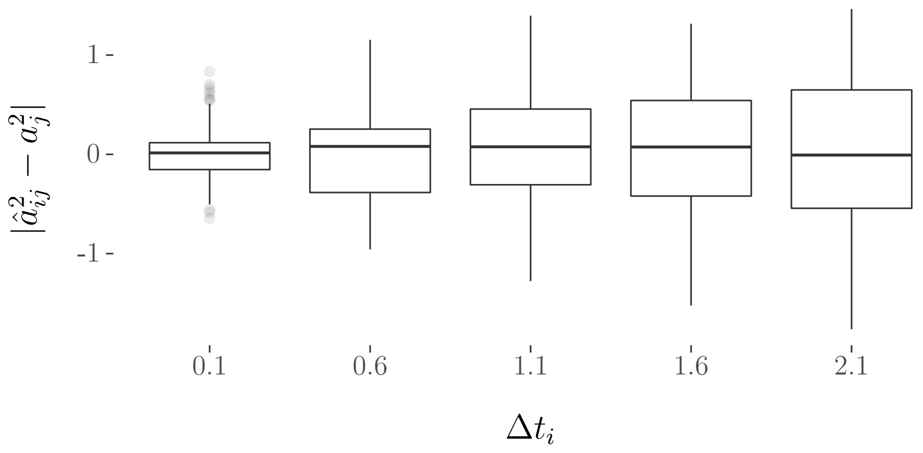


Figure 2.17: Box plot showing the difference in the diffusion approximations $\hat{a}^2_{ij}$ and the true values a^2_j, $j = 1, \ldots, 4$, for each Δt_i value, $i = 1, \ldots, 5$. That is, for each Δt_i, the box plot displays $|\hat{a}^2_{ij} - a^2_j|$ for $j = 1, \ldots, 4$ for all 100 iterations. The lower and upper hinges of the boxes correspond to the 25th and 75th percentiles of the data, and the whiskers show 1.5 times the interquartile range.

reality the partitioning method is not suitable for use in Part II. This leads to a different stochastic differential equation form and parameter estimation method, known as maximum likelihood estimation.

Maximum Likelihood Estimator

In this section we introduce a potentially more useful method for parameter estimation of stochastic differential equations, which will be applied to the simulator outputs in Part II. First we give a brief theoretical introduction behind maximum likelihood estimation on stochastic differential equations. Then, we discuss various methods for utilising maximum likelihood estimation in practice. This includes methods that are consistent in both the $\Delta t \to 0$ case, and the $T \to \infty$ case.

Consider for the moment a more general stochastic differential equation. Let $\{\Omega, \mathcal{F}, \mathbb{P}\}$ be a probability space with filtration $\{\mathcal{F}_t,\, t \geq 0\}$ and $(\mathsf{B}_t)_{t\geq 0}$ the standard Brownian process on Ω. Define the process $(\mathsf{X}_t)_{t\geq 0}$ on Ω by,

$$\mathrm{d}\mathsf{X}_t = b(\vartheta, \mathsf{X}_t)\,\mathrm{d}t + a(\mathsf{X}_t)\,\mathrm{d}B_t, \tag{2.22}$$

with X_0 $\mathcal{F}_0$-measurable and parameter of interest $\vartheta \in \Theta \subset \mathbb{R}$. The functions $b\colon \mathbb{R}^2 \to \mathbb{R}$ and $a\colon \mathbb{R} \to \mathbb{R}^+$ are known and satisfy the following conditions for any $\vartheta \in \Theta$ (see Kutoyants, 2013),

1. The function $b(\vartheta, \cdot)$ is locally bounded, the function $a(\cdot)^2$ is continuous and positive and for some $A > 0$ the condition,

$$x\, b(\vartheta, x) + a(x)^2 \leq A(1 + x^2),$$

   holds.

2. The functions satisfy,

$$\begin{aligned} V(\vartheta, x) &= \int_0^x \exp\Big(-2\int_0^y \frac{b(\vartheta, v)}{a(v)^2}\,\mathrm{d}v\Big)\,\mathrm{d}y \to \pm\infty, \quad \text{as } x \to \pm\infty; \text{ and,} \\ G(\vartheta) &= \int_{-\infty}^{\infty} a(y)^2 \exp\Big(2\int_0^y \frac{b(\vartheta, v)}{a(v)^2}\,\mathrm{d}v\Big)\,\mathrm{d}y < \infty. \end{aligned} \tag{2.23}$$

Suppose also that the true value of the parameter is given by ϑ_0.

Here two methods to obtain a maximum likelihood estimator are introduced. Maximum likelihood estimators are such that the approximations of the true parameter ϑ_0 are chosen such that the probability of the observed data occurring maximises a likelihood function. Let $\{x_{t_1}, x_{t_2}, \dots, x_{t_n}\}$ be some set of observations, and $\Delta t = t_i - t_{i-1}$ for $i = 2, 3, \dots, n$. The first method introduced is consistent in the limit $n \to \infty$ with t_n constant and $\Delta t \to 0$ whilst the second method assumes Δt constant and $t_n \to \infty$ as $n \to \infty$. First the theory behind both methods is introduced, and then the methods are applied on sample data.

The motivation behind the first maximum likelihood method is given by the behaviour of the full continuous path $(\mathsf{X}_t)_{t\in[0,T]}$. It is shown in Durrett (2018, Section 6) that $(\mathsf{X}_t)_{t\in[0,T]}$, as given in (2.22) with functions $b(\cdot,\cdot)$ and $a(\cdot)$ satisfying the conditions above has the invariant distribution with density function,

$$f(x;\,\vartheta) = \frac{1}{G(\vartheta)\,a(x)^2}\exp\Big(2\int_0^x \frac{b(\vartheta, v)}{a(v)^2}\,\mathrm{d}v\Big),$$

for $x \in \mathbb{R}$, where G is given in (2.23).

In the infinite dimensional space the Lebesgue measure does not exist. The likelihood function can instead be given by the Radon-Nikodyn derivative,

$$L(\vartheta;\mathsf{X}^T) := \frac{\mathrm{d}\mathsf{P}^T}{\mathrm{d}\mathsf{P}_0^T}(\mathsf{X}^T) = \exp\Big(\int_0^T \frac{b(\vartheta,\mathsf{X}_t)}{a(\mathsf{X}_t)^2}\,\mathrm{d}\mathsf{X}_t - \frac{1}{2}\int_0^T \frac{b(\vartheta,\mathsf{X}_t)^2}{a(\mathsf{X}_t)^2}\,\mathrm{d}t\Big),$$

where P^T and P_0^T are the probability measures induced in $(\mathcal{C}[0,T], \mathcal{B}[0,T])$ by $\mathsf{X}^T = (\mathsf{X}_t)_{t\in[0,T]}$ and by $\mathsf{B}^T = (\mathsf{B}_t)_{t\in[0,T]}$ respectively. As the measure P^T is absolutely continuous with respect to the Wiener measure P_0 the likelihood function exists Lipster & Shiryaev (1977). The log-likelihood is given by

$$l(\vartheta;\mathsf{X}^T) = \log\big(L(\vartheta,\mathsf{X}^T)\big) = \int_0^T \frac{b(\vartheta,\mathsf{X}_t)}{a(\mathsf{X}_t)^2}\,\mathrm{d}\mathsf{X}_t - \frac{1}{2}\int_0^T \frac{b(\vartheta,\mathsf{X}_t)^2}{a(\mathsf{X}_t)^2}\,\mathrm{d}t. \tag{2.24}$$

Then, the maximum likelihood estimator is given by $\hat{\vartheta}^T_{\mathrm{ML}}$ exactly when

$$\hat{\vartheta}^T_{\mathrm{ML}} := \underset{\vartheta\in\Theta}{\operatorname{argmax}}\, L(\vartheta;\,\mathsf{X}^T).$$

Note that as $\log(\cdot)\colon (0,\infty) \to \mathbb{R}$ is a monotonically increasing function, then also $\hat{\vartheta}^T_{\mathrm{ML}} = \max_{\vartheta\in\Theta} l(\vartheta; \mathsf{X}^T)$.

In practice only a set of n observations from the full trajectory will be known, and thus only an approximation, $\hat{\vartheta}^{T,n}_{\mathrm{ML}}$, can be found for $\hat{\vartheta}^T_{\mathrm{ML}}$. The first maximum likelihood method uses the theory detailed above to obtain $\hat{\vartheta}^{T,n}_{\mathrm{ML}}$, assuming that the integrals in (2.24) can be approximated from the observations. Thus, for this estimator to be accurate, dense observations from the process $(\mathsf{X}_t)_{t\in[0,T]}$ are required for integral approximation.

For the second maximum likelihood method we assume instead that we have sparse observations from the process and therefore the integral approximation technique is not justifiable. In this case, for times $0 = t_1 < t_2 < \ldots < t_n = T$, the sparse path, $\mathsf{X}_{t_1}, \mathsf{X}_{t_2}, \ldots, \mathsf{X}_{t_n}$ is a Markov chain with likelihood function given by,

$$L_n(\vartheta\colon \mathsf{X}^{T,n}) := L_n(\vartheta\colon \mathsf{X}_{t_1}, \mathsf{X}_{t_2}, \ldots, \mathsf{X}_{t_n}) = \prod_{i=2}^{n} p(\Delta t_i, \mathsf{X}_{t_{i-1}}, \mathsf{X}_{t_i}),$$

where $\Delta t_i = t_i - t_{i-1}$ and $p\colon [0,\infty) \times \mathbb{R}^2 \to \mathbb{R}^+$ is the transition density of the time-homogeneous process. The log-likelihood is given by

$$l_n(\vartheta; \mathsf{X}^{T,n}) = \sum_{i=2}^{n} \log\big(p(\Delta t_i, \mathsf{X}_{t_{i-1}}, \mathsf{X}_{t_i})\big), \tag{2.25}$$

and the maximum likelihood estimator is given by $\hat{\vartheta}^{T,n}_{\mathrm{sML}}$ and satisfies, $\hat{\vartheta}^{T,n}_{\mathrm{sML}} = \max_{\vartheta\in\Theta} l_n(\vartheta; \mathsf{X}^{T,n})$. Here, the notation 'sML' is used for this '$\underline{\text{s}}$parse $\underline{\text{m}}$aximum $\underline{\text{l}}$ikelihood' method.

Billingsley (1961) shows that the maximum likelihood estimator for a general Markov chain exists with probability going to one, and converges in probability to the true value ϑ_0 as the length of the chain increases to infinity. In Dacunha-Castelle & Florens-Zmirou (1986) convergence in probability of the maximum likelihood estimator $\hat{\vartheta}^{T,n}_{\mathrm{sML}}$ is considered for Δt constant and $n \to \infty$. It is also shown that consistent estimators of ϑ are obtained for $\Delta t = \Delta t_n \to 0$ as $n \to \infty$, with $n\Delta t_n \to \infty$. In this latter case, asymptotic efficiency with speed $\sqrt{n\Delta t_n}$ is obtained if $n\Delta t_n = \mathcal{O}(1)$.

Dense maximum likelihood estimation. First, a method given in Bishwal (2007, Section 7) is introduced, which approximates the integrals in (2.24) to obtain an estimator $\hat{\vartheta}^{T,n}_{\mathrm{ML}}$, consistent in the $\Delta t \to 0$ sense. Here, approximations of the Itô and Stratonovich integral are given and used to approximate the log-likelihood function to find the corresponding parameters estimates. First, we reduce the size of the class of drift functions in (2.22) such that they are constrained to be of the form,

$$b(\vartheta, \mathsf{X}_t) = \vartheta\, \tilde{b}(\mathsf{X}_t)$$

for some square integrable function $\tilde{b}\colon \mathbb{R} \to \mathbb{R}$, where ϑ is the parameter value of interest.

Suppose that dense observations of the process $(\mathsf{X}_t)_{t\geq 0}$ as in (2.22) are given over the time interval $[0,T]$, denoted $x^{T,n} = \{x_{t_1}, x_{t_2}, \dots, x_{t_n}\}$, with $0 = t_1 < t_2 < \cdots < t_n = T$. Let $\mathsf{X}^T := (\mathsf{X}_t)_{t\in[0,T]}$, $f\colon \mathbb{R}^+ \times \mathbb{R} \to \mathbb{R}$ be square integrable and $\int f(t,\mathsf{X}_t) \circ \mathrm{d}\mathsf{X}_t$ be the Stratonovich integral (Stratonovich, 1966). The Stratonovich integral is related to the Itô integral by

$$\int_0^T f(t,\mathsf{X}_t) \circ \mathrm{d}\mathsf{X}_t = \int_0^T f(t,\mathsf{X}_t)\,\mathrm{d}\mathsf{X}_t + \frac{1}{2}\int_0^T \frac{\partial}{\partial x} f(t,\mathsf{X}_t)\,\mathrm{d}t \quad \text{a.s.},$$

and so the maximum likelihood estimator (2.24) can be rewritten,

$$\begin{aligned}\hat{\vartheta}^T_{\mathrm{ML}} &= \operatorname*{argmax}_{\vartheta\in\Theta} l(\vartheta\colon \mathsf{X}^T)\\ &= \frac{\int_0^T \tilde{b}(\mathsf{X}_t)\,\mathrm{d}\mathsf{X}_t}{\int_0^T \tilde{b}^2(\mathsf{X}_t)\,\mathrm{d}t}\\ &= \frac{\int_0^T \tilde{b}(\mathsf{X}_t)\circ \mathrm{d}\mathsf{X}_t - \frac{1}{2}\int_0^T \frac{\mathrm{d}}{\mathrm{d}x}\tilde{b}(\mathsf{X}_t)\,\mathrm{d}t}{\int_0^T \tilde{b}^2(\mathsf{X}_t)\,\mathrm{d}t} \quad \text{a.s.}.\end{aligned} \tag{2.26}$$

Then, from Bishwal (2007), $\lim_{\Delta t\to 0} S_n = \int_0^T f(t,\mathsf{X}_t)\circ \mathrm{d}\mathsf{X}_t$, where,

$$S_n := \frac{1}{2}\sum_{i=2}^{n}\big(f(t_{i-1},\, x_{t_{i-1}}) + f(t_i,\, x_{t_i})\big)\big(x_{t_i} - x_{t_{i-1}}\big),$$

and this Stratonovich integral approximation S_n is substituted into (2.26) to give the approximation

$$\hat{\vartheta}_{\mathrm{ML}}^{T,n} = \frac{\frac{1}{2}\sum_{i=2}^{n}\big(\tilde{b}(x_{t_{i-1}}) + \tilde{b}(x_{t_i})\big)(x_{t_i} - x_{t_{i-1}}) - \frac{1}{2}\sum_{i=1}^{n}\frac{\mathrm{d}}{\mathrm{d}x}\tilde{b}(x_{t_{t-1}})(t_i - t_{i-1})}{\sum_{i=1}^{n}\tilde{b}^2(x_{t_{i-1}})(t_i - t_{i-1})}. \tag{2.27}$$

Now, consider an Ornstein-Uhlenbeck process as introduced in Section 2.3.2. Suppose that the drift parameter σ^2 is known, and maximum likelihood estimation is used to obtain an estimate of the diffusion parameter γ. With the notation above, the Ornstein-Uhlenbeck process as in (2.12) has drift function $\tilde{b}(x) = -x$ and drift parameter $\vartheta = \gamma$. For the specific case of the diffusion being $\sigma^2 = 1$, the log-likelihood (2.24) becomes

$$l(\gamma\colon \mathsf{X}^T) = -\int_0^T \gamma \mathsf{X}_t\,\mathrm{d}\mathsf{X}_t - \frac{1}{2}\int_0^T (\gamma \mathsf{X}_t)^2\,\mathrm{d}t.$$

The maximum likelihood estimator (2.26) is then

$$\hat{\gamma}_{\mathrm{ML}}^{T} = -\frac{\int_0^T \mathsf{X}_t\,\mathrm{d}\mathsf{X}_t}{\int_0^T \mathsf{X}_T^2\,\mathrm{d}t} = -\frac{\int_0^T \mathsf{X}_t \circ \mathrm{d}\mathsf{X}_t + \frac{1}{2}(T-0)}{\int_0^T \mathsf{X}_T^2\,\mathrm{d}t},$$

with the approximation using dense observations, $x^{T,n} = \{x_{t_1}, \dots, x_{t_n}\}$, given by

$$\hat{\gamma}_{\mathrm{ML}}^{T,n} = -\frac{\frac{1}{2}\sum_{i=2}^{n}\big(x_{t_{i-1}} + x_{t_i}\big)(x_{t_i} - x_{t_{i-1}}) + \frac{1}{2}T}{\sum_{i=2}^{n} x_{t_{i-1}}^2 (t_i - t_{i-1})}.$$

Sparse maximum likelihood estimation. In this section the log-likelihood (2.25) and associated maximum likelihood estimator for sparse observations from an Ornstein-Uhlenbeck process are examined. Here, we desire estimates for both the unknown parameters γ and σ^2 of the Ornstein-Uhlenbeck process (2.12).

Let $x^{T,n} = \{x_{t_1}, x_{t_2}, \dots, x_{t_n}\}$ be a set of observations from the Ornstein-Uhlenbeck process (2.12), with $t_n = T$ and $t_i - t_{i-1} = \Delta t$ for $i = 2, \dots, n$. Assuming that $\mathsf{X}_{t_1} = x_{t_1} \in \mathbb{R}$ almost surely, then from Section 2.3.2, for any $\Delta t > 0$ and $y \in \mathbb{R}$

the transition density of an Ornstein-Uhlenbeck process is,

$$p(\Delta t, x_{t_1}, y) = \sqrt{\frac{\gamma}{\pi\sigma^2(1-e^{-2\gamma\Delta t})}}\exp\Big(-\frac{\gamma}{\sigma^2}\cdot\frac{(y-x_{t_1}e^{-\gamma\Delta t})^2}{1-e^{-2\gamma\Delta t}}\Big).$$

Therefore, discrete observations from an Ornstein-Uhlenbeck process form a Markov chain with log-likelihood in the form of (2.25) given by,

$$\begin{aligned} l_n(\gamma,\sigma^2\colon x^{T,n}) = &= \frac{n}{2}\big(\log\gamma - \log(\pi\sigma^2(1-e^{-2\gamma\Delta t})\big) \\ &\qquad - \frac{\gamma}{\sigma^2(1-e^{-2\gamma\Delta t})}\sum_{i=1}^{n}(x_{t_i}-x_{t_{i-1}}e^{-\gamma\Delta t})^2. \end{aligned} \tag{2.28}$$

Differentiating (2.28) with respect to γ and σ^2 gives,

$$\begin{aligned} &\frac{\partial}{\partial\gamma}l_n(\gamma,\sigma^2\colon x^{T,n}) \\ &\quad = \frac{n}{2}\Big(\frac{1}{\gamma} - \frac{2\Delta t e^{-2\gamma\Delta t}}{1-e^{-2\gamma\Delta t}}\Big) \\ &\qquad - \frac{1}{\sigma^2(1-e^{-2\gamma\Delta t})}\Bigg(\frac{\big(1-(1-2\gamma\Delta t)e^{-2\gamma\Delta t}\big)\sum_{i=2}^{n}(x_{t_i}-x_{t_{i-1}}e^{-\gamma\Delta t})^2}{1-e^{-2\gamma\Delta t}} \\ &\qquad\qquad - 2\gamma\Delta t e^{-2\gamma\Delta t}\sum_{i=2}^{n}x_{t_{i-1}}(x_{t_i}-x_{t_{i-1}}e^{-\gamma\Delta t})\Bigg); \\ &\frac{\partial}{\partial\sigma^2}l_n(\gamma,\sigma^2\colon x^{T,n}) = \frac{-n}{2\sigma^2} + \frac{\gamma}{(\sigma^2)^2(1-e^{-2\gamma\Delta t})}\sum_{i=2}^{n}(x_{t_i}-x_{t_{i-1}}e^{-\gamma\Delta t})^2. \end{aligned}$$

The sparse maximum likelihood estimators $\hat{\gamma}_{\mathrm{sML}}$ and $\hat{\sigma}^2_{\mathrm{sML}}$ therefore satisfy

$$\frac{\partial}{\partial\gamma}l_n(\hat{\gamma}_{\mathrm{sML}},\hat{\sigma}^2_{\mathrm{sML}}\colon x^{T,n}) = 0; \text{ and } \frac{\partial}{\partial\sigma^2}l_n(\hat{\gamma}_{\mathrm{sML}},\hat{\sigma}^2_{\mathrm{sML}}\colon x^{T,n}) = 0.$$

When $\frac{\partial}{\partial\sigma^2}l_n(\gamma,\sigma^2\colon x^{T,n}) = 0$, σ^2 can be written as a function of γ in the following way,

$$\sigma^2(\gamma) = \frac{2\gamma}{n(1-e^{-2\gamma\Delta t})}\sum_{i=2}^{n}(x_{t_i}-x_{t_{i-1}}e^{-\gamma\Delta t})^2. \tag{2.29}$$

Therefore, the maximum likelihood estimators can be written $\hat{\gamma}_{\mathrm{sML}}$ and $\hat{\sigma}^2_{\mathrm{sML}} = \sigma^2(\hat{\gamma}_{\mathrm{sML}})$. This dependency simplifies any optimisation required to obtain the estimators.

Example 2.3.6 (Maximum likelihood estimation: Ornstein-Uhlenbeck process)**.** *In this example, the following process is repeated* 100 *times.*

1. *Euler-Maruyama approximation is used to simulate a trajectory of an Ornstein-Uhlenbeck process with* $\gamma = 1$ *and* $\sigma^2 = 100$*, from time* $t_1 = 0$ *to* $t_n = 100$ *with* $\Delta t = 10 \times 10^{-4}$*;*

2. *The Ornstein-Uhlenbeck trajectory is thinned such that the new time step is* $\Delta t_{sparse} = 0.1$*;*

3. *Estimated parameter values* $\hat{\gamma}_{sML}$ *and* $\hat{\sigma}^2_{sML}$ *are found by maximising the log-likelihood as in* (2.28)*.*

For iterations $i = 1, 2, \ldots, 100$ *we obtain parameter estimations* $\hat{\gamma}_{sML,\,i}$*, which have mean* $\bar{\gamma}_{sML} := (1/100)\sum_{i=1}^{100}\hat{\gamma}_{sML,\,i} = 1.0012$ *and have standard deviation* $sd(\{\hat{\gamma}_{sML,\,1}, \ldots, \hat{\gamma}_{sML,\,100}\}) = 0.1548$*. For* σ^2 *the procedure above results in estimates* $\hat{\sigma}^2_{sML,\,i}$*,* $i = 1, \ldots, 100$*, with mean* $\bar{\sigma}^2_{sML} := (1/100)\sum_{i=1}^{100}\hat{\sigma}^2_{sML,\,i} = 99.9382$ *and standard deviation* $sd(\{\hat{\sigma}^2_{sML,\,1}, \ldots, \hat{\sigma}^2_{sML,\,100}\}) = 4.7781$*.*

The grey path in Figure 2.18 shows an Ornstein-Uhlenbeck trajectory found in Step 1.. The black line gives the sparse "observation" $\{x_{t_1}, x_{t_2}, \ldots, x_{t_n}\}$ *with* $\Delta t_{sparse} = t_j - t_{j-1} = 0.1$ *for all* $j \in \{2, 3, \ldots, n\}$*. For this specific sparse trajectory, maximising the log-likelihood* (2.28) *gives maximum likelihood estimates* $\hat{\gamma}_{sML} = 0.8617$ *and* $\hat{\sigma}^2_{sML} = 103.1348$*.*

Example 2.3.7 (Unstability of the sparse maximum likelihood method)**.** *For some realisations of Ornstein-Uhlenbeck processes with certain* γ *and* σ^2 *values, the sparse maximum likelihood method (where* (2.28) *is maximised) becomes unstable.*

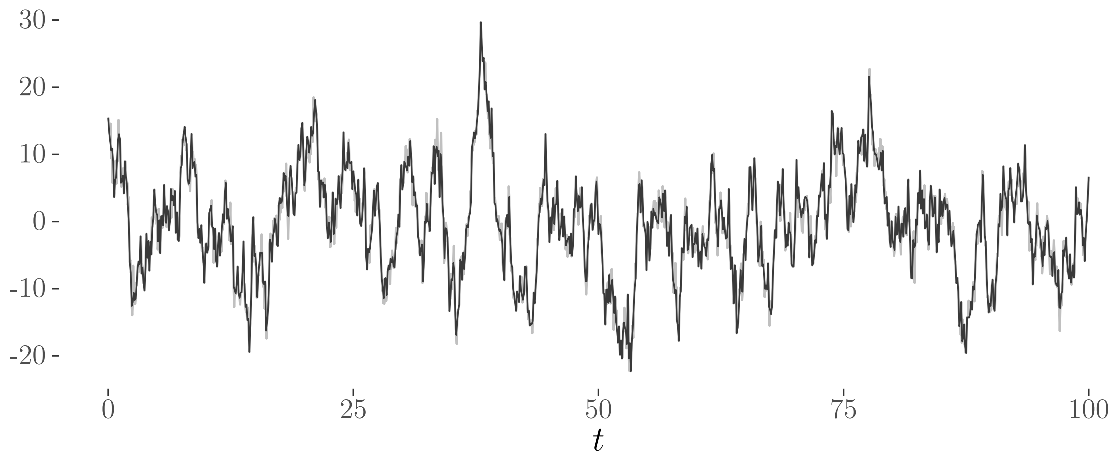


Figure 2.18: Here, an Ornstein-Uhlenbeck trajectory found using Euler-Maruyama approximation with $\Delta t = 10 \times 10^{-4}$ (grey path) is thinned such that the "observed" trajectory has time steps $\Delta t_{\text{sparse}} = 0.1$ (black line) By maximising the log-likelihood (2.28) using the sparse observations, the maximum likelihood parameter estimates are $\hat{\gamma}_{\text{sML}} = 0.8617$ and $\hat{\sigma}^2_{\text{sML}} = 103.1348$.

This behaviour is especially prevalent when the time step of the sparse realisation is "large". This is explained most clearly by examining the log-likelihood plot along the line $(\gamma, \sigma^2(\gamma))$, where $\sigma^2(\cdot)$ is given in (2.29).

Here, the true parameter values are $\gamma = 5$, $\sigma^2 = 10$ and Euler-Maruyama approximation is used to simulate an Ornstein-Uhlenbeck process from time $t_1 = 0$ to $t_n = n\Delta t = 1000$. Then, the chain is thinned such that the observations are given by $\{x_0, x_1, \ldots, x_{1000}\}$, i.e. $\Delta t_{sparse} = 1$.

For this set of sparse observations, the estimated parameter values found using sparse maximum likelihood are $\hat{\gamma}_{sML} = 3.7798$ and $\hat{\sigma}^2_{sML} = \sigma^2(\hat{\gamma}_{sML}) = 7.8191$. Figure 2.19 shows the log-likelihood along the line $(\gamma, \sigma^2(\gamma))$ for γ from 1 to 20 (solid black line), which is nearly flat for γ values from around 3 to 20. The vertical dotted grey line gives the maximum likelihood estimator $\hat{\gamma}_{sML}$, and the horizontal dotted grey line shows the maximum log-likelihood value, $l_n(\hat{\gamma}_{sML}, \sigma^2(\hat{\gamma}_{sML})) = -1436.99$.

From this it is clear that any small permutation in this line could easily result in a very different $(\gamma, \sigma^2(\gamma)$ combination being chosen as the "optimal". These small permutations could be due to many factors, such as numerical issues, or certain outliers in the trajectory.

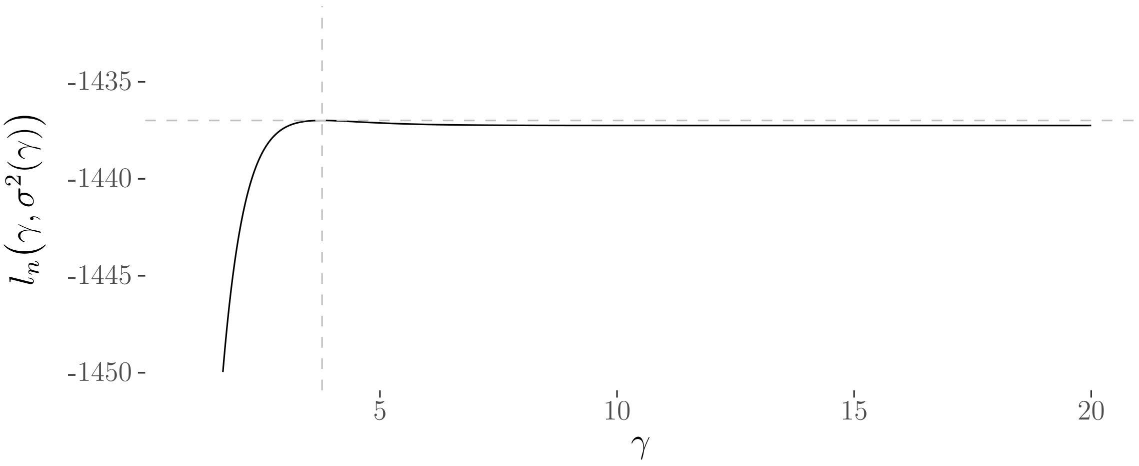


Figure 2.19: Log-likelihood (solid black line) for a sparse realisation from an Ornstein-Uhlenbeck process with parameters $\gamma = 5$, $\sigma^2 = 10$ and time steps $\Delta t_{\text{sparse}} = 1$. The log-likelihood function is calculated along the line $(\gamma, \sigma^2(\gamma))$ for γ from 1 to 20, and $\sigma^2(\cdot)$ given in (2.29). The maximum value of the log-likelihood is found at $\hat{\gamma}_{\text{sML}} = 3.7798$ (vertical dotted grey line) and $\sigma^2(\hat{\gamma}_{\text{sML}}) = 7.8191$. Here, $l_n(\hat{\gamma}_{\text{sML}}, \sigma^2(\hat{\gamma}_{\text{sML}})) = -1436.99$ (horizontal dotted grey line).

Method of Moments Estimator

We now introduce another parameter estimation method for obtaining drift and diffusion estimates of an Ornstein-Uhlenbeck process. The motivation behind this parameter estimation method is as follows. Suppose monthly observations of an output from an Earth system simulation are available, which is believed to be a realisation of a sparse Ornstein-Uhlenbeck process (2.12). The sample variance and lag-1 correlation (i.e. the correlation between successive months) is known or has been calculated from the trajectory. Using only this information, is it possible to estimate the parameters γ and σ^2?

Recall from Section 2.3.2 that for an Ornstein-Uhlenbeck process, $\mathsf{dX}_t^{\text{OU}} = -\gamma \mathsf{X}_t^{\text{OU}}\, \mathrm{d}t + \sigma\, \mathsf{dB}_t$, with $\mathsf{X}_0^{\text{OU}} \sim \mathcal{N}(0, \frac{\sigma^2}{2\gamma})$, $(\mathsf{X}_t)_{t\geq 0}$ is a stationary zero-mean Gaussian process with covariance function,

$$\rho(s,t) = \rho(t-s) = \frac{\sigma^2}{2\gamma} e^{-\gamma|t-s|}. \tag{2.30}$$

From this, the true variance and lag-1 correlation of an Ornstein-Uhlenbeck process is $\mathrm{V} := \mathrm{Var}(\mathsf{X}_t^{\text{OU}}) = \rho(0) = \frac{\sigma^2}{2\gamma}$ and $r(1) = \frac{\rho(1)}{\mathrm{Var}(\mathsf{X}_t^{\text{OU}})} = e^{-\gamma}$. If we know value of V and $r(1)$, then by rearranging the variance and lag-1 correlation we obtain,

$$\gamma = -\log r(1); \quad \sigma^2 = -2\mathrm{V}\log r(1). \tag{2.31}$$

If the sample variance and lag-1 correlation have been calculated from an observed trajectory, then by replacing the true variance and lag-1 correlation in (2.31), the method of moments estimates $\hat{\gamma}_{\text{MM}}$ and $\hat{\sigma}^2_{\text{MM}}$ for γ and σ^2 can be found.

Method of moments parameter estimation is only valid when the lag-1 sample correlation is positive, which is an obvious condition that can be seen by examining (2.31). As γ increases, the respective Ornstein-Uhlenbeck process begins to more resemble white-noise (which has zero correlation), and therefore the likelihood of a negative lag-1 sample correlation increases. This limitation restricts when method of moments estimation can be applied to a realisation in practise.

Example 2.3.8 (Method of moments estimation)**.** *In this example, method of moments estimation is applied to both the dense and the sparse trajectories shown in Figure 2.18 (from Example 2.3.6).*

That is, the dense trajectory is from a stationary Ornstein-Uhlenbeck process with $\gamma = 1$ and $\sigma^2 = 100$, approximated by Euler-Maruyama method with time steps $\Delta t = 10 \times 10^{-4}$. The sparse trajectory is found by taking every 100 observations of the dense chain, such that the sparse time step is $\Delta t_{sparse} = 0.1$.

From the discussion in Section 2.3.2, the two trajectories theoretically have variance $V = 100/2$ and lag-1 correlation $r(1) = e^{-1} \approx 0.368$. Using the dense *trajectory, the sample variance and correlation are calculated as* $\hat{V} = 60.1213$ *and* $\hat{r}(1) = 0.4060$ *respectively. Putting these sample moments into* (2.31) *gives Ornstein-Uhlenbeck parameter estimates* $\hat{\gamma}_{MM} = 0.9015$ *and* $\hat{\sigma}^2_{MM} = 108.3935$.

The sparse trajectory with $\Delta t_{sparse} = 0.1$ *gives sample variance and correlation* $\hat{V}_{sparse} = 60.9216$ *and* $\hat{r}_{sparse}(1) = 0.4017$ *respectively. This gives estimated parameters* $\hat{\gamma}_{sMM} = 0.9120$ *and* $\hat{\sigma}^2_{sMM} = 111.1197$.

2.3.3.2 Comparing parameter estimation methods for pointwise observations

In this section we compare the sparse maximum likelihood and method of moments parameter estimation methods described in Section 2.3.3 to each other. These methods are applied to a set of Ornstein-Uhlenbeck processes, simulated using the Euler-Maruyama approximation in the following way.

1. Simulate dense trajectories of the Ornstein-Uhlenbeck process (2.12) using the Euler-Maruyama approximation as introduced in Remark 2.12;

2. Thin the simulated trajectories by only taking every $N \gg 1$ points, giving sparse trajectories similar to that used to motivate the sparse maximum likelihood estimation method.

Each dense Ornstein-Uhlenbeck realisation simulated in Step 1. above gives observations $\{x_0,\, x_{\Delta t},\, x_{2\Delta t}, \ldots,\, x_{n\Delta t}\}$, with x_0 a realisation from a $\mathcal{N}(0, \frac{\sigma^2}{2\gamma})$ distribution and $\Delta t \ll 1$. Then, removing all but every N point (assuming N is a factor of n) as in Step 2. above gives the sparse trajectory $\{x_0, x_{N\Delta t},\, x_{2N\Delta t}, \ldots,\, x_{n\Delta t}\}$.

Using the sparse observations found in Step 2., the drift and diffusion estimates obtained by maximising the sparse log-likelihood (2.28) are compared to those obtained by method of moments from (2.31).

To compare the parameter estimation methods, $M \in \mathbb{Z}$ trajectories are simulated for various drift and diffusion combinations (γ, σ^2). For every combination of $(\gamma,\, \sigma^2)$, the approximations $\{(\hat{\gamma}_i,\, \hat{\sigma}^2_i) \colon i = 1, 2, \ldots, M\}$ are found for both maximum likelihood estimation and method of moments, and the sample mean and standard deviation of these sets of parameter estimates are calculated.

The estimations from the sparse maximum likelihood which maximises the log-likelihood (2.28) are notated by the subscript 'sML' and method of moments by 'MM'. To ensure that a method of moments estimator can be obtained sample trajectories that have a negative lag-1 sample correlation are omitted. Because of this, the number of valid method of moments estimators is less than the total number of trajectories simulated.

Sparse Ornstein-Uhlenbeck process: comparing maximum likelihood and method of moments

In this section, we simulate Ornstein-Uhlenbeck trajectories with varying drift and diffusion values, and the associated parameter estimates $\hat{\gamma}$ and $\hat{\sigma}^2$ are compared for each estimation method as explained in the section above.

In this case, the values for the drift are given by $\gamma \in \{0.5, 1, 1.5, 2, 2.5, 3\}$ and those for the diffusion term are $\sigma^2 \in \{50, 100, 150, 200, 250, 300\}$. For every drift and diffusion combination (of which there are 36), we simulate $M = 100$ realisations of a dense Ornstein-Uhlenbeck trajectory on $\mathbb{R}$ using the Euler-Maruyama method as introduced in Remark 2.12, from time $t_1 = 0$ to $t_n = 100$ and with time steps $\Delta t = 10 \times 10^{-4}$. These trajectories are then thinned by only selecting every $N = 100$ time points to create sparse trajectories with time steps $\Delta t_{\text{sparse}} = 0.1$. The two parameter estimation methods – sparse maximum likelihood and method of moments – are then applied to the observations $\{x_0, x_{\Delta t_{\text{sparse}}}, x_{2\Delta t_{\text{sparse}}}, \dots, x_{1000\Delta t_{\text{sparse}}}\}$.

Table 2.20 shows the mean and standard deviation of the two parameter fitting procedures for each value of $\gamma \in \{0.5, 1, 1.5, \dots, 3\}$. For every value of γ, 100 trajectories are simulated for each value of $\sigma^2 \in \{50, 100, 150, \dots, 300\}$, and $\#\{50, 100, \dots, 300\} = 6$, giving 600 trajectories to obtain parameter estimates from. Thus, for each γ value, $\bar{\gamma}_{\text{sML}}$ and $\bar{\gamma}_{\text{MM}}$ are the means of $\hat{\gamma}_{\text{sML},1}, \dots, \hat{\gamma}_{\text{sML},600}$ and $\hat{\gamma}_{\text{MM},1}, \dots, \hat{\gamma}_{\text{MM},600}$ respectively.

Table 2.21 shows the equivalent means and standard deviations for each value of $\sigma^2 \in \{50, 100, \dots, 300\}$. For every value of σ^2, 100 trajectories are simulated for each value of $\gamma \in \{0.5, 1, 1.5, \dots, 3\}$, and $\#\{0.5, 1, \dots, 3\} = 6$, giving 600 trajectories to obtain parameter estimates from. Thus, for each σ^2 value, $\bar{\sigma}^2_{\text{sML}}$ and $\bar{\sigma}^2_{\text{MM}}$ are the means of $\hat{\sigma}^2_{\text{sML},1}, \dots, \hat{\sigma}^2_{\text{sML},600}$ and $\hat{\sigma}^2_{\text{MM},1}, \dots, \hat{\sigma}^2_{\text{MM},600}$ respectively.

Figure 2.22 shows a selection of maximum likelihood estimators $\hat{\gamma}_{\text{sML}}$ for $\gamma \in \{0.5, 1, 1.5, \dots, 3\}$. The step function (black line) gives the true value of γ as a function of realisation number, given on the x-axis. The grey circles (●) shows the $\hat{\gamma}_{\text{sML}}$ estimates when the Ornstein-Uhlenbeck process has true diffusion parameter $\sigma^2 = 50$. The black triangles ($\triangle$) give the $\hat{\gamma}_{\text{sML}}$ estimates when the true diffusion parameter is $\sigma^2 = 300$.

Figure 2.23 shows a selection of maximum likelihood estimators $\hat{\sigma}^2_{\text{sML}}$ for $\sigma^2 \in \{50, 100, \dots, 300\}$. The step function (black line) gives the true value of σ^2 as a function of realisation number, given on the x-axis. The grey circles (●) shows the $\hat{\sigma}^2_{\text{sML}}$ estimates when the Ornstein-Uhlenbeck process has true drift parameter $\gamma = 0.5$. The black triangles ($\triangle$) give the $\hat{\sigma}^2_{\text{sML}}$ estimates when the true drift parameter is $\gamma = 3$.

Figures 2.24 and 2.25 show the equivalent method of moments estimators in the same way as Figures 2.22 and 2.23 respectively.

Studying Figure 2.24 and Figure 2.25 it is clear that method of moments produces some very spurious results with a large variance in the estimations. One reason for this is due to the fact that this method only works for trajectories with positive lag-1 sample correlations, and therefore favours correlation estimates that are skewed upwards. For example, suppose that two trajectories both have a true lag-1 correlation of 0.1, but have lag-1 sample correlations of -0.1 and 0.3. Then, the method of moments estimator will only be able to be applied to the realisation with positive lag-1 sample correlation. Repeating this process with multiple trajectories, it is clear that the method "favours" trajectories where the lag-1 sample correlation is larger than the true lag-1 correlation.

γ	0.50	1.00	1.50	2.00	2.50	3.00
$\bar{\gamma}_{\mathrm{sML}}$	0.5173	1.0266	1.5396	2.0266	2.5387	3.0157
$\mathrm{sd}(\{\hat{\gamma}_{\mathrm{sML},1}, \ldots, \hat{\gamma}_{\mathrm{sML},600}\})$	0.0116	0.0247	0.0340	0.0526	0.0641	0.0810
$\bar{\gamma}_{\mathrm{MM}}$	0.5499	1.0833	1.6505	2.2427	2.7050	3.0442
$\mathrm{sd}(\{\hat{\gamma}_{\mathrm{MM},1}, \ldots, \hat{\gamma}_{\mathrm{MM},600}\})$	0.0164	0.0601	0.1824	0.5329	0.8168	0.9020

Figure 2.20: Table showing mean and standard deviation for the maximum likelihood and method of moments parameter estimators for $\gamma \in \{0.5, 1, 1.5, \ldots, 3\}$. For each value of γ, 100 approximate Ornstein-Uhlenbeck trajectories are simulated using Euler-Maruyama approximation for each σ^2 value, where $\sigma^2 \in \{50, 100, 150, \ldots, 300\}$. These 600 trajectories are then thinned to create sparse observations to perform the parameter estimation on.

σ^2	50	100	150	200	250	300
$\bar{\sigma}^2_{\mathrm{sML}}$	50.18	99.89	149.96	200.03	250.31	301.23
$\mathrm{sd}(\{\hat{\sigma}^2_{\mathrm{sML},1}, \ldots, \hat{\sigma}^2_{\mathrm{sML},600}\})$	2.54	4.86	7.59	10.21	12.27	14.90
$\bar{\sigma}^2_{\mathrm{MM}}$	52.43	105.58	158.36	209.49	261.38	318.02
$\mathrm{sd}(\{\hat{\sigma}^2_{\mathrm{MM},1}, \ldots, \hat{\sigma}^2_{\mathrm{MM},600}\})$	12.76	27.57	36.09	51.73	71.37	84.20

Figure 2.21: Table showing mean and standard deviation for the maximum likelihood and method of moments parameter estimators for $\sigma^2 \in \{50, 100, 150, \ldots, 300\}$. For each value of σ^2, 100 approximate Ornstein-Uhlenbeck trajectories are simulated using Euler-Maruyama approximation for each γ value, where $\gamma \in \{0.5, 1, 1.5, \ldots, 3\}$. These 600 trajectories are then thinned to create sparse observations to perform the parameter estimation on.

2.3.3.3 Parameter estimation: block-average Ornstein-Uhlenbeck observations

In this section we detail a novel method to obtain drift and diffusion parameter estimates of an Ornstein-Uhlenbeck process (2.12) given observations that are averaged over some time periods. That is, instead of observing pointwise realisations of an Ornstein-Uhlenbeck process – as was the case in the sections above – we observe realisations of the block-average process given by (2.14). In the case of the HadCM3 climate simulator in Part II, we obtain outputs that are monthly means.

Let $\{x^{\mathrm{baOU}}_{[t_1,t_2)}, x^{\mathrm{baOU}}_{[t_2,t_3)}, \ldots, x^{\mathrm{baOU}}_{[t_{n-1},t_n)}\}$ be a realisation from a block-average Ornstein-Uhlenbeck process as in (2.14), with drift and diffusion parameters γ and σ^2 respectively. As shown in Section 2.3.2.1, this vector has a density function from a

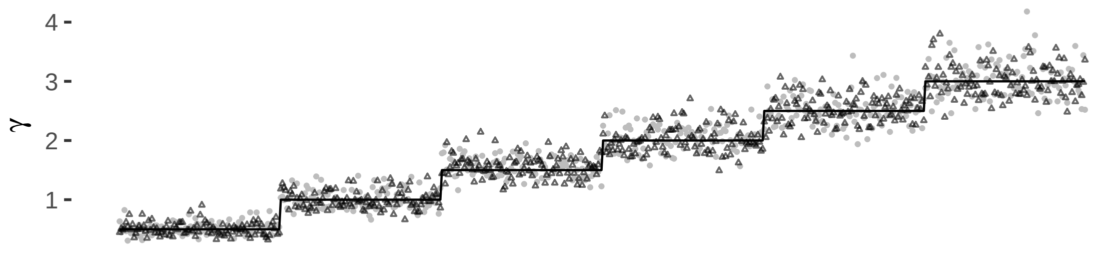


Figure 2.22: Ornstein-Uhlenbeck parameter estimation for $\hat{\gamma}_{\mathrm{sML}}$ for $\gamma \in \{0.5, 1, 1.5, \ldots, 3\}$. Grey circles ($\bullet$) are $\hat{\gamma}_{\mathrm{sML}}$ estimates when $\sigma^2 = 50$ and black triangles ($\triangle$) are $\hat{\gamma}_{\mathrm{sML}}$ estimates for $\sigma^2 = 300$. The x-axis gives the realisation number, and the step function gives the true γ value for each realisation. For each combination of (γ, σ^2), 100 Ornstein-Uhlenbeck trajectories are simulated from time 0 to 10 with $\Delta t = 10 \times 10^{-4}$, and then thinned such that the sparse trajectories have $\Delta t_{\mathrm{sparse}} = 10 \times 10^{-2}$.

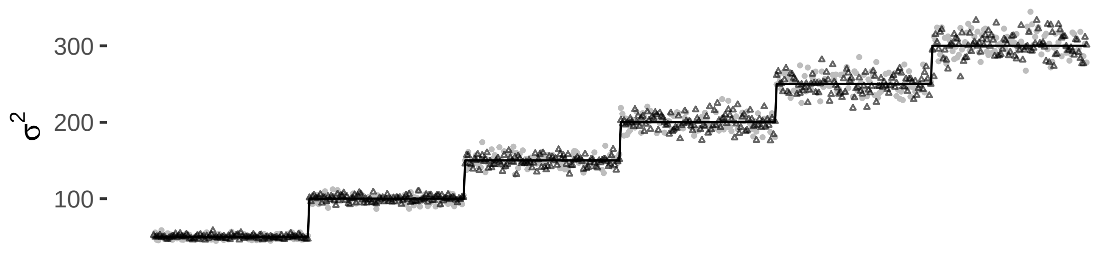


Figure 2.23: Ornstein-Uhlenbeck parameter estimation for $\hat{\sigma}^2_{\mathrm{sML}}$ for $\sigma^2 \in \{50, 100, 150, \ldots, 300\}$. Grey circles ($\bullet$) are $\hat{\sigma}^2_{\mathrm{sML}}$ estimates when $\gamma = 0.5$ and black triangles ($\triangle$) are $\hat{\sigma}^2_{\mathrm{sML}}$ estimates for $\gamma = 3$. The x-axis gives the realisation number, and the step function gives the true γ value for each realisation. For each combination of (γ, σ^2), 100 Ornstein-Uhlenbeck trajectories are simulated from time 0 to 10 with $\Delta t = 10 \times 10^{-4}$, and then thinned such that the sparse trajectories have $\Delta t_{\mathrm{sparse}} = 10 \times 10^{-2}$.

zero mean Gaussian distribution. The aim now is to find estimates for γ and σ^2 of the original Ornstein-Uhlenbeck process. This is done by maximum likelihood estimation, described below.

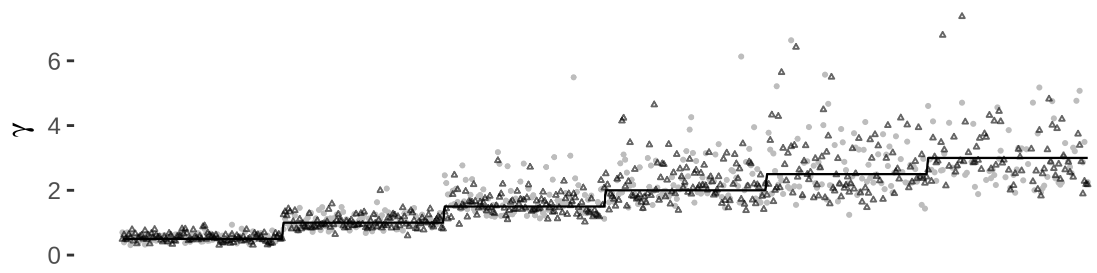


Figure 2.24: Ornstein-Uhlenbeck parameter estimation for $\hat{\gamma}_{\text{MM}}$ for $\gamma \in \{0.5, 1, 1.5, \dots, 3\}$. Grey circles (●) are $\hat{\gamma}_{\text{MM}}$ estimates when $\sigma^2 = 50$ and black triangles (△) are $\hat{\gamma}_{\text{MM}}$ estimates for $\sigma^2 = 300$. The x-axis gives the realisation number, and the step function gives the true γ value for each realisation. For each combination of (γ, σ^2), 100 Ornstein-Uhlenbeck trajectories are simulated from time 0 to 10 with $\Delta t = 10 \times 10^{-4}$, and then thinned such that the sparse trajectories have $\Delta t_{\text{sparse}} = 10 \times 10^{-2}$.

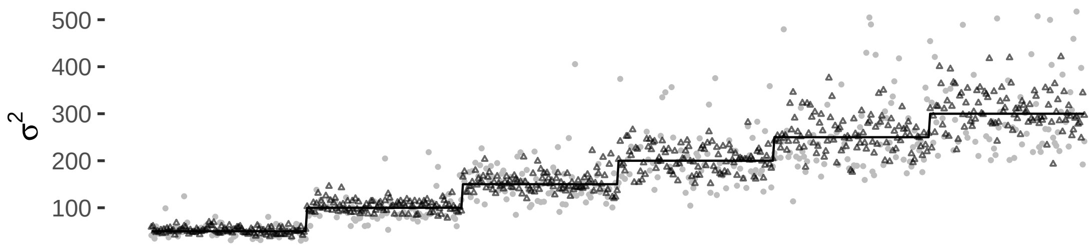


Figure 2.25: Ornstein-Uhlenbeck parameter estimation for $\hat{\sigma}^2_{\text{MM}}$ for $\sigma^2 \in \{1, 21, 41, 61, 81, 101\}$. Grey circles (●) are $\hat{\sigma}^2_{\text{MM}}$ estimates when $\gamma = 0.5$ and black triangles (△) are $\hat{\sigma}^2_{\text{MM}}$ estimates for $\gamma = 3$. The x-axis gives the realisation number, and the step function gives the true γ value for each realisation. For each combination of (γ, σ^2), 100 Ornstein-Uhlenbeck trajectories are simulated from time 0 to 10 with $\Delta t = 10 \times 10^{-4}$, and then thinned such that the sparse trajectories have $\Delta t_{\text{sparse}} = 10 \times 10^{-2}$.

Maximum likelihood estimator

Here we suppose that the Ornstein-Uhlenbeck process is integrated over equal-sized time intervals, such that $t_i - t_{i-1} = \Delta t$ for $i = 2, \dots, n$. Let the elements of the vector $x^{\text{baOU}} = (x^{\text{baOU}}_{[0,\,\Delta t)}, x^{\text{baOU}}_{[\Delta t,\,2\Delta t)}, \dots, x^{\text{baOU}}_{[(n-1)\Delta t,\,n\Delta t)}) \in \mathbb{R}^n$ be realisations from the block-average Ornstein-Uhlenbeck process as given in (2.14). As discussed in Section 2.3.2.1, x^{baOU} is therefore a realisation from a multivariate Gaussian distribution with mean zero and covariance matrix $\Sigma_{\gamma,\sigma^2} = (\sigma_{ij}) \in \mathbb{R}^{n\times n}$, where $\sigma_{ij} = ca^{|j-i|} + b\mathbb{1}_{i=j}$ with c, a and b given in (2.18). Here the covariance matrix is notated with γ and σ^2 in the subscript to make explicit the dependence on the Ornstein-Uhlenbeck parameter values.

From above and Section 2.3.2.1, the vector form of a block-average Ornstein-Uhlenbeck process, $\mathsf{X}^{\text{baOU}} = (\mathsf{X}^{\text{baOU}}_{[0,\,\Delta t)}, \mathsf{X}^{\text{baOU}}_{[\Delta t,\,2\Delta t)}, \dots, \mathsf{X}^{\text{baOU}}_{[(n-1)\Delta t,\,n\Delta t)})$ has a distribution with density,

$$p(x; \Sigma_{\gamma,\sigma^2}) = (2\pi)^{-n/2}\,|\Sigma_{\gamma,\sigma^2}|^{-1/2}\exp(-\frac{1}{2}x^\top\Sigma^{-1}_{\gamma,\sigma^2}x),$$

for all $x \in \mathbb{R}^n$, and therefore the log-likelihood function is given by,

$$l(\Sigma_{\gamma,\sigma^2} : \mathsf{X}^{\text{baOU}}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma_{\gamma,\sigma^2}| - \frac{1}{2}(\mathsf{X}^{\text{baOU}})^\top\Sigma^{-1}_{\gamma,\sigma^2}\mathsf{X}^{\text{baOU}}. \tag{2.32}$$

The aim is now to maximise (2.32) over all covariance matrices Σ_{γ,σ^2}, which depend on γ and σ^2. For the log-likelihood function to be optimised directly over all γ and σ^2 values, we require the inverse of the covariance matrix. Although in general this is computationally intensive for large n, by exploiting the structure of the covariance matrix (which is a symmetric Toeplitz matrix), the inverse can be found using Trench's Algorithm in order n^2 time (Trench, 1964). See Golub & Van Loan (2012) for a description of the algorithm, and McLeod *et al.* (2007) for an `R` implementation. However, Trench's Algorithm is not useful for obtaining the determinant of the covariance matrix (as required in (2.32)) and therefore in general Cholesky decomposition has to be used instead.

For the observation x^{baOU}, the maximum likelihood estimates $\hat{\gamma}_{\text{baML}}$ and $\hat{\sigma}^2_{\text{baML}}$ satisfy,

$$\begin{pmatrix}\hat{\gamma}_{\text{baML}}\\ \hat{\sigma}^2_{\text{baML}}\end{pmatrix} = \underset{(\gamma,\sigma^2)\in\mathbb{R}\times(0,\infty)}{\text{argmax}} \quad l(\Sigma_{\gamma,\sigma^2} : \bar{x}), \tag{2.33}$$

where $l\colon \mathbb{R}^{n\times n} \to \mathbb{R}$ is given by (2.32) and Σ_{γ,σ^2} has elements $\sigma_{ij} = c\,a^{|j-i|} + b\mathbb{1}_{i=j}$ with c, a and b given in (2.18). Here, the subscript 'baML' is used to show that the log-likelihood being maximised is that given by (2.32), and related to <u>b</u>lock-<u>a</u>verage Ornstein-Uhlenbeck <u>m</u>aximum <u>l</u>ikelihood estimation.

Remark 2.13: Using maximum likelihood estimation with one observation

As described above, the maximum likelihood estimation method only uses one observation from a multivariate Gaussian distribution $\mathcal{N}(0, \Sigma_{\gamma,\sigma^2})$, which initially seems unsuitable for the parameter estimation task. To encourage the reader that this method has potential, consider the case of n independent (centred) observations $x_1, x_2, \ldots, x_n$ from a univariate normal distribution with variance σ^2. Here, the maximum likelihood estimator for σ^2 is $\hat{\sigma}^2_{\mathrm{ML}} = (1/n)\sum_{i=1}^n x_i^2$. Now, suppose instead these n observations are a single observation $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ from a multivariate normal with covariance matrix given by $\sigma^2 I_n$, where $I_n \in \mathbb{R}^n$ is the identity matrix. Here, the multivariate Normal density becomes,

$$\begin{aligned} p(x; \sigma^2 I_n) &= (2\pi)^{-n/2}(\sigma^2)^{-n/2} \exp\big(-\frac{1}{2}\sum_{i=1}^n \frac{x_i^2}{\sigma^2}\big) \\ &= \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\big(-\frac{x_i^2}{2\sigma^2}\big). \end{aligned}$$

Therefore the likelihood of a single realisation in $\mathbb{R}^n$ assumed from a multivariate Gaussian with a diagonal covariance matrix is equivalent to the likelihood obtained with n realisations from a univariate Gaussian.

Example 2.3.9 (Maximum likelihood estimation: block-average Ornstein-Uhlenbeck process)**.** *This example is directly comparable to Example 2.3.6 as the pointwise Ornstein-Uhlenbeck trajectories that are used to perform maximum likelihood estimation are the same.*

In this example, the following process is repeated 100 *times. Note that Steps 1. and 2. can be replaced with simulating values directly from a multivariate Gaussian as discussed in Section 2.3.2. Here the average over an Ornstein-Uhlenbeck process* *is used to simulate how in practice a block-average Ornstein-Uhlenbeck trajectory may arise. For example, a simulator may be designed to average over some time intervals whilst it is being run to save on computational memory.*

1. *Euler-Maruyama approximation is used to simulate a trajectory of an Ornstein-Uhlenbeck process with $\gamma = 1$ and $\sigma^2 = 100$, from time $t_0 = 0$ to $t_n = 100$ with $\Delta t = 10 \times 10^{-4}$;*

2. *The average of the observations in each time interval $T_i = [i-1, i)$, $i = 1, 2, \ldots, 100$ is calculated to give an approximate block-average Ornstein-Uhlenbeck process, $\bar{x}_{[0,1)}, \bar{x}_{[1,2)}, \ldots, \bar{x}_{[99,100)}$;*

3. *Estimated parameter values $\hat{\gamma}_{baML}$ and $\hat{\sigma}^2_{baML}$ are found by maximising the log-likelihood as in (2.33).*

Over the $i = 1, 2, \ldots, 100$ iterations, parameter estimations $\hat{\gamma}_{baML,\,i}$ are obtained, and the mean is given by $\bar{\gamma} := (1/100)\sum_{i=1}^{100} \hat{\gamma}_{baML,\,i} = 1.0015$, with standard deviation $sd(\{\hat{\gamma}_{baML,\,1}, \ldots, \hat{\gamma}_{baML,\,100}\}) = 0.2271$. For σ^2 the estimates $\hat{\sigma}^2_{baML,\,i}$ are obtained with mean given by $\bar{\sigma}^2 := (1/100)\sum_{i=1}^{100} \hat{\sigma}^2_{baML,\,i} = 99.9489$ and standard deviation $sd(\{\hat{\sigma}^2_{baML,\,1}, \ldots, \hat{\sigma}^2_{baML,\,100}\}) = 20.1745$.

The grey path in Figure 2.26 gives one Ornstein-Uhlenbeck realisation used in the example. This is the same trajectory as shown in Figure 2.18 and as discussed in Example 2.3.8. The grey vertical lines show the partitioning of time into the intervals $T_i = [i-1, i)$, $i = 1, 2, \ldots, 100$. The value of the block-average process $\bar{x}_{[i-1,i)}$ in interval T_i is given by the black horizontal lines. For this realisation, the maximum likelihood method as given in (2.33) gives estimated Ornstein-Uhlenbeck parameter values $\hat{\gamma}_{baML} = 0.9133$ and $\hat{\sigma}^2_{baML} = 112.4258$.

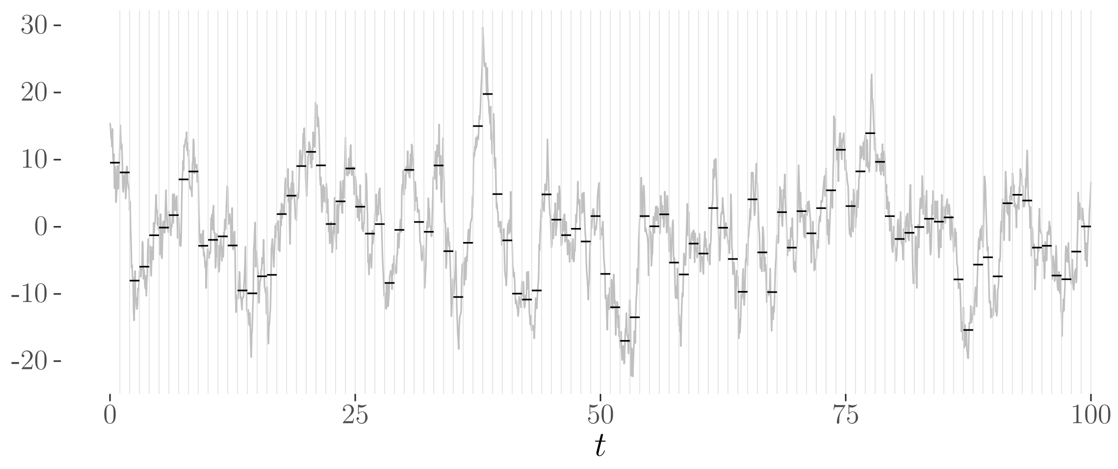


Figure 2.26: Here, an Ornstein-Uhlenbeck trajectory (grey path) is split and averaged over the intervals $T_i = [i-1, i)$, $i = 1, 2, \ldots, 100$ (grey vertical lines), to give an approximate block-average Ornstein-Uhlenbeck process (black horizontal lines). Using the block-average Ornstein-Uhlenbeck maximum likelihood method as given in (2.33), this realisation gives estimates $\hat{\gamma}_{\text{baML}} = 0.9133$ and $\hat{\sigma}^2_{\text{baML}} = 112.4258$.

2.3.4 Ornstein-Uhlenbeck process with seasonal effects

In this section we introduce block-average Ornstein-Uhlenbeck processes that exhibit some seasonal behaviour, alongside some mean functions that can be used to help approximately simulate the underlying (pointwise) Ornstein-Uhlenbeck processes. This is motivated by the general behaviour of climate systems where a sensible *a priori* assumption of the output of a simulation would include some month-by-month variability that repeats with a yearly cycle.

This section is structured as follows. In Section 2.3.4.1 we first introduce the motivation behind this section, and discuss the structure of a seasonal block-average Ornstein-Uhlenbeck process. Then, in Section 2.3.4.2 we introduce two different mean functions that can be used to give pointwise Ornstein-Uhlenbeck processes with the same seasonal means as some block-average Ornstein-Uhlenbeck process. The second seasonal mean function introduced in Section 2.3.4.2 use a novel spline approach, where the function is constrained by the average behaviour within each disjoint time interval.

2.3.4.1 Introduction to seasonal effects

The motivation to estimate and remove seasonality is highlighted in Panel 2.10. Briefly, we observe some block-average Ornstein-Uhlenbeck process that exhibits seasonality. From this, we want to estimate the seasonality, and produce dense realisations from some pointwise Ornstein-Uhlenbeck processes that have the same seasonal expectations as the block-average Ornstein-Uhlenbeck process. In Part II of this thesis, the "observed" time series that are modelled as block-average Ornstein-Uhlenbeck processes with some seasonality are the principal and independent components obtained by dimension reduction as described in Chapter 1.

Panel 2.10: Motivation for considering seasonal effects

We want to be able to estimate the drift and diffusion parameters of an underlying Ornstein-Uhlenbeck process, given observations from some block-average Ornstein-Uhlenbeck process shifted with some seasonal means. That is, we obtain $\hat{\gamma}_{\text{baML}}$ and $\hat{\sigma}^2_{\text{baML}}$ in the following way:

1. Observe some seasonal block-average Ornstein-Uhlenbeck process;
2. Estimate the seasonal means and remove them from the observations, to obtain an approximate realisation from a standard block-average Ornstein-Uhlenbeck process as in (2.14). This is described in more detail in Remark 2.15;
3. Estimate the underlying Ornstein-Uhlenbeck process drift and diffusion parameters, γ and σ^2, using the maximum likelihood estimator (2.33);

Now, using the estimated drift and diffusion parameters, a realisation from a pointwise Ornstein-Uhlenbeck process can be simulation such that the mean of the realisation during each season is equivalent to the estimated seasonal means of the original (seasonal) block-average Ornstein-Uhlenbeck realisation observed in Step 1.. That is,

(i) Simulate a pointwise Ornstein-Uhlenbeck process as in (2.12) with parameter values $\hat{\gamma}_{\text{baML}}$ and $\hat{\sigma}^2_{\text{baML}}$;

(ii) Add some spline function to the simulated Ornstein-Uhlenbeck process such that the expected value of the process in each season is equal to the seasonal means estimated from the original block-average Ornstein-Uhlenbeck process observations.

These new pointwise realisations can be used for forecasting and analysing the original observations.

Step 2. can be achieved simply by finding the arithmetic mean for each season and subtracting it from the associated points, and we described Step 3. in Section 2.3.3.3. Therefore, we can obtain the parameter estimates $\hat{\gamma}_{\text{baML}}$ and $\hat{\sigma}^2_{\text{baML}}$. Now, Step (i) can be done using Euler-Maruyama approximation as discussed in Remark 2.12. Therefore, in this section we focus on the novel Step (ii) of Panel 2.10.

Here we define some mean function $m(t)$ using the estimated seasonal means from a block-average Ornstein-Uhlenbeck realisation. First, we give some conditions that this function is required to satisfy. Let the process $(\mathsf{Y}_t)_{t\geq 0}$ be given by,

$$\mathsf{Y}_t = \mathsf{X}_t^{\text{OU}} + m(t), \tag{2.34}$$

where X_t^{OU} is the Ornstein-Uhlenbeck process given by (2.12). Differentiating this we obtain the stochastic differential equation,

$$\begin{aligned} \mathrm{d}\mathsf{Y}_t &= \mathrm{d}\mathsf{X}_t^{\text{OU}} + \frac{\mathrm{d}}{\mathrm{d}t} m(t)\,\mathrm{d}t \\ &= -\gamma\big(\mathsf{Y}_t - m(t)\big)\,\mathrm{d}t + \sigma\,\mathrm{d}\mathsf{B}_t + \frac{\mathrm{d}}{\mathrm{d}t} m(t)\,\mathrm{d}t \\ &= -\gamma\Big(\mathsf{Y}_t - m(t) - \frac{1}{\gamma}\frac{\mathrm{d}}{\mathrm{d}t} m(t)\Big)\,\mathrm{d}t + \sigma\,\mathrm{d}\mathsf{B}_t. \end{aligned} \tag{2.35}$$

One comment on the form of this stochastic differential equation is given below in Remark 2.14.

Remark 2.14: Derivative of mean function

Notice that the stochastic process given by $\mathrm{d}\mathsf{Y}_t$ in (2.35) includes the derivative of the function $m(t)$. Intuitively, this derivative prevents the mean of the process lagging behind the expected mean given by $m(t)$. For example, suppose the mean function is linear such that $m(t) = a\,t + b$ for some $a, b \in \mathbb{R}$, and compare the two stochastic processes,

$$\mathrm{d}\mathsf{Y}_t^{(1)} = -\gamma\big(\mathsf{Y}_t - at - b\big)\,\mathrm{d}t + \sigma\,\mathrm{d}\mathsf{B}_t \tag{2.36}$$

$$\mathrm{d}\mathsf{Y}_t^{(2)} = -\gamma\Big(\mathsf{Y}_t - at - b - \frac{a}{\gamma}\Big)\,\mathrm{d}t + \sigma\,\mathrm{d}\mathsf{B}_t. \tag{2.37}$$

One realisation of these two processes is given in Figure 2.27, highlighting $\mathsf{Y}_t^{(1)}$ lagging behind the true mean function. That is, $\mathsf{E}\mathsf{Y}_t^{(1)} < a\,t + b$ for all $t > 0$.

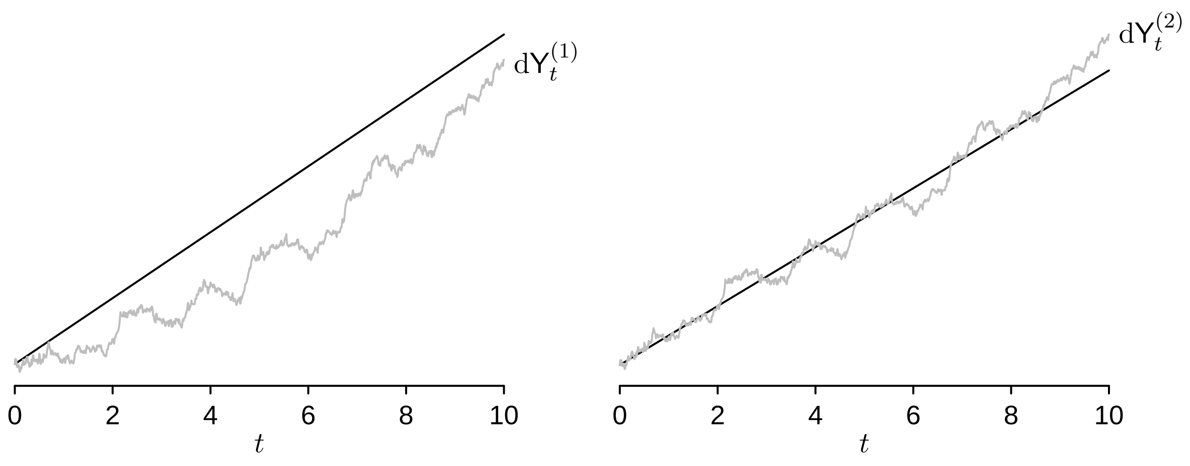


Figure 2.27: An example of a realisation from two stochastic processes, $\mathsf{Y}_t^{(1)}$ and $\mathsf{Y}_t^{(2)}$ as in (2.36) and (2.37) respectively, showing how the derivative of the mean function is required in the differential equation to prevent the stochastic process from lagging behind the true mean.

If the mean function $m(t)\colon [0,\infty) \to \mathbb{R}$ is some differentiable function that satisfies Lipschitz continuity, then the process Y_t as in (2.34) is a strong solution of $\mathrm{d}\mathsf{Y}_t$ given in (2.35). Integrating Y_t over some time interval $[t_{i-1}, t_i)$ gives the process $(\mathsf{Y}_{[t_{i-1},t_i)})_{i\in\mathbb{Z}}$, defined by,

$$\begin{aligned}
\mathsf{Y}_{[t_{i-1},t_i)} &:= \frac{1}{t_i - t_{i-1}} \int_{t_{i-1}}^{t_i} \mathsf{Y}_t \, \mathrm{d}t \\
&= \frac{1}{t_i - t_{i-1}} \int_{t_{i-1}}^{t_i} \mathsf{X}_t^{\mathrm{OU}} + m(t) \, \mathrm{d}t \\
&= \frac{1}{t_i - t_{i-1}} \Big(\langle \mathbb{1}_{[t_{i-1},t_i)}, \mathsf{X}^{\mathrm{OU}} \rangle + \int_{t_{i-1}}^{t_i} m(t) \, \mathrm{d}t \Big) \\
&= \mathsf{X}^{\mathrm{baOU}}_{[t_{i-1},t_i)} + \mu_i, \qquad (2.38)
\end{aligned}$$

where $\mu_i := 1/(t_i - t_{i-1}) \cdot \int_{t_{i-1}}^{t_i} m(t) \, \mathrm{d}t$. Removing the seasonal means μ_i from the process $(Y_{[t_{i-1},t_i)})_{i\in\mathbb{Z}}$ gives a zero mean block-average Ornstein-Uhlenbeck process.

The seasonal notation used throughout this section is now introduced. Let $(\mathsf{Y}_t)_{t\geq 0}$ be a process that exhibits seasonality, such that some aspect of the behaviour of the process repeats after $T_N \in \mathbb{R}^+$ time. The "yearly" interval $[0, T_N)$ is subdivided into seasons, $[T_0, T_1), [T_1, T_2), \ldots, [T_{N-1}, T_N)$, where $0 = T_0 < T_1 < \cdots < T_N$. Throughout this section, the expectation of the process over each season is the same for every year. That is, for every season seperated in time by T_N, the expectation of the process averaged over this season is equal, such that if $\mu_i := \mathsf{E}\{\mathsf{Y}_t \colon t \in [T_{i-1}, T_i)\}$, then also $\mu_i = \mathsf{E}\{\mathsf{Y}_t \colon t \in [kT_N + T_{i-1}, kT_N + T_i)\}$, for all $i \in \{1, 2, \ldots, N\}$ and $k \in \mathbb{N}$. Throughout the rest of this section, we assume that we know the mean values $\mu_1, \ldots, \mu_N$ exactly. In Remark 2.15 we explain how estimates of the seasonal means (as in Step 2. of Panel 2.10) can be obtained from a seasonal block-average Ornstein-Uhlenbeck realisation.

In the case of the HadCM3 climate simulator examined in Part II, it is shown that the seasonal behaviour has a yearly interval $[0, T_N)$, subdivided into twelve months, such that $N = 12$.

Remark 2.15: Estimating the seasonal mean

In general the exact seasonal means (given by μ_i in (2.38)) will not be known, and will instead have to be calculated from some realisation of the process $(\mathsf{Y}_{[t_{i-1},t_i)})_{i\in\mathbb{Z}}$ given in (2.38) (as we discussed in Step 2. of Panel 2.10). Here we show how this mean approximation is found and removed from the original seasonal block-average Ornstein-Uhlenbeck realisation given in Step 1. (Panel 2.10). Let $\{y_{t_1}, y_{t_2}, \ldots, y_{t_n}\}$, $0 = t_1 < t_2 < \cdots < t_n = T$, be some observation of the process $(\mathsf{Y}_t)_{t\geq 0}$ given in (2.34) with mean function $m(t)$. Let $0 = T_0 < T_1 < T_2 < \ldots < T_N \leq T$ be the seasons of the process such that it has the same expectation over the interval $[T_{i-1}, T_i)$ as over $[kT_N + T_{i-1}, kT_N + T_i)$, for any $k \in \mathbb{Z}$. Estimates of the mean values μ_i for each season $i = 1, 2, \ldots, N$, using the observations $\{y_{t_1}, y_{t_2}, \ldots y_{t_n}\}$, are given by the arithmetic means

$$\hat{\mu}_i = \frac{1}{n_i} \sum_{j=1}^{n} y_{t_j} \mathbb{1}_{t_j \in \bigcup_{k=0}^{\lfloor T/T_N \rfloor} [T_{i-1}+kT_N, T_i+kT_N)},$$

where $n_i := \#\{j \colon t_j \in \bigcup_{k=0}^{\lfloor T/T_N \rfloor} [T_{i-1} + k\,T_N, T_i + k\,T_N)\}$. For this estimation to be valid we require at least one observation in each season. An example of observations being put into bins to calculate the sample means is given in Figure 2.28.

2.3.4.2 Seasonal mean functions

This section proceeds as follows. First, we introduce a simple piecewise constant mean function. This is not used in the rest of this thesis, but is included here as a simple example to help the reader understand the process. Then, we describe a mean function that is Lipschitz and which takes the form of quadratic splines over the time intervals.

Piecewise constant mean

In this section we suppose the process $(\mathsf{Y}^{\mathrm{P}}_t)_{t\geq 0}$ satisfies

$$\mathsf{Y}^{\mathrm{P}}_t = \mathsf{X}^{\mathrm{OU}}_t + m^{\mathrm{P}}(t), \tag{2.39}$$

$[0, T_1)$	$[T_1, T_2)$	$[T_2, T_1)$	$\cdots$	$[T_{N-1}, T_N)$
$y_{t_1}\ y_{t_2}\ y_{t_3}$	$y_{t_4}\ y_{t_5}\ y_{t_6}\ y_{t_7}$	y_{t_8}	$\cdots$	$y_{t_{31}}\ y_{t_{32}}\ y_{t_{33}}$
$y_{t_{34}}\ y_{t_{35}}\ y_{t_{36}}$	$y_{t_{37}}\ y_{t_{38}}\ y_{t_{39}}\ y_{t_{40}}$	$y_{t_{41}}$	$\cdots$	$y_{t_{64}}\ y_{t_{65}}\ y_{t_{66}}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$y_{t_{n-4}}\ y_{t_{n-3}}\ y_{t_{n-2}}$	$y_{t_{n-1}}\ y_{t_n}$			
$\hat{\mu}_1$	$\hat{\mu}_2$	$\hat{\mu}_3$	$\cdots$	$\hat{\mu}_N$

Figure 2.28: An example showing binning of observations $y_{t_1}, y_{t_2}, \ldots, y_{t_n}$ into bins associated with the intervals $[0, T_1), [T_1, T_2), \ldots, [T_{N-1}, T_N)$, with $T_N \leq T$.

where

$$m^{\mathrm{P}}(t) = \sum_{n=1}^{N} \mu_n \mathbb{1}_{\{t \in \bigcup_{k \in \mathbb{Z}} [T_{j-1}+kT_N, T_j+kT_N)\}},$$

with $\mu_j \in \mathbb{R}$, $j = 1, \ldots, N$.

In this case, a realisation of the process $(\mathsf{Y}_t)_{t\in\mathbb{R}}$ as in (2.39) can be obtained by simulating an Ornstein-Uhlenbeck process $\mathsf{X}_t^{\mathrm{OU}}$ and adding the associated seasonal mean value μ_i whilst the process is in the respective seaon, $i = 1, \ldots, N$. Here the process will exhibit discontinuous jumps at each seasonal time-boundary.

This issue is the motivation for the mean function given below, where Lipschitz continuous splines are used for the $m(t)$ function in place of the piecewise continuous construction used here.

Mean function using quadratic splines

Following from the section above, here we consider modelling the mean function $m(t)$ using a specific form of periodic quadratic splines. As discussed in 2.3.4, let the mean value of the stochastic process $(\mathsf{Y}_t)_{t\geq 0}$ over the interval $[T_{i-1}, T_i)$, where

$0 = T_0 < T_1 < \cdots < T_N$, be given by

$$\mu_i := \mathsf{E}\Big\{\mathsf{Y}_t \colon t \in [T_{i-1}, T_i)\Big\}. \tag{2.40}$$

For the quadratic spline case, the process $(\mathsf{Y}_t)_{t\geq 0}$ has the form $\mathsf{Y}_t = \mathsf{X}_t^{\mathrm{OU}} + m(t)$, with $\mathsf{X}_t^{\mathrm{OU}}$ an Ornstein-Uhlenbeck process as in (2.12), and $m(t)$ some recurrent mean function given by

$$m(t) = \sum_{i=1}^{N} \big(a_i t^2 + b_i t + c_i\big)\,\mathbb{1}_{\{t \in \bigcup_{k\in\mathbb{Z}} [T_{i-1}+k\,T_N,\, T_i + k\,T_N)\}}, \tag{2.41}$$

with $a_i, b_i, c_i \in \mathbb{R}$, $i = 1, 2, \ldots, N$ chosen such that the following conditions are satisfied:

1. Expectation condition: For $i = 1, 2, \ldots, N$,

$$\frac{1}{T_i - T_{i-1}} \int_{T_{i-1}}^{T_i} (a_i t^2 + b_i t + c_i)\,\mathrm{d}t = \mu_i, \tag{2.42}$$

   with μ_i given in (2.40). Note that this expectation condition is different to the classical spline condition that requires the function to go through a specific point. This condition allows the spline to be calculated using information from a block-average process when pointwise values are not available.

2. Equal at boundaries with recurrent boundary condition:

$$\begin{cases} a_i\,T_i^2 + b_i\,T_i + c_i = a_{i+1}\,T_i^2 + b_{i+1}\,T_i + c_{i+1}, & \text{for } i \in \{1, \ldots, N-1\}; \\ a_i\,T_i^2 + b_i\,T_i + c_i = a_1\,T_i^2 + b_1\,T_i + c_1, & \text{for } i = N. \end{cases}$$

3. Derivative equal at boundaries with recurrent boundary condition:

$$\begin{cases} \frac{\mathrm{d}}{\mathrm{d}t}\big(a_i\,t^2 + b_i\,t + c_i\big)|_{t=T_i} = \frac{\mathrm{d}}{\mathrm{d}t}\big(a_{i+1}\,t^2 + b_{i+1}\,t + c_{i+1}\big)|_{t=T_i}, & \text{for } i \in \{1, \ldots, N-1\}; \\ \frac{\mathrm{d}}{\mathrm{d}t}\big(a_i\,t^2 + b_i\,t + c_i\big)|_{t=T_i} = \frac{\mathrm{d}}{\mathrm{d}t}\big(a_1\,t^2 + b_1\,t + c_1\big)|_{t=T_i}, & \text{for } i = N. \end{cases}$$

   Calculating the derivative gives, for $i \in \{1, 2, \ldots, N-1\}$, the condition

$2a_i T_i + b_i = 2a_{i+1} T_i + b_{i+1}$ and at the boundary, $2a_N T_N + b_N = 2a_1 T_N + b_1$.

Using these three conditions the values a_i, b_i, c_i, $i = 1, 2, \ldots, N$, can be calculated by a solving a linear system of equations. First, we note that the expectation condition given in (2.42) can be written,

$$\frac{1}{T_i - T_{i-1}} \left(\frac{1}{3} a_i (T_i^3 - T_{i-1}^3) + \frac{1}{2} b_i (T_i^2 - T_{i-1}^2) + c_i (T_i - T_{i-1}) \right) = \mu_i,$$

for $i = 1, \ldots, 12$. From this, the linear system of equations $Ax = b$ can be solved for

$$x = (a_1, b_1, c_1, a_2, b_2, c_2, \ldots, a_N, b_N, c_N)^\top \in \mathbb{R}^{3N},$$

with $A \in \mathbb{R}^{3N \times 3N}$ and $b \in \mathbb{R}^{3N}$ given by,

$$A = \begin{pmatrix} A_1^{(1)} & -A_1^{(2)} & 0 & 0 & \cdots & 0 & 0 \\ 0 & A_2^{(1)} & -A_2^{(2)} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & & A_{N-1}^{(1)} & -A_{N-1}^{(2)} \\ -A_0^{(2)} & 0 & 0 & \cdots & & 0 & A_N^{(1)} \end{pmatrix},$$

where

$$A_i^{(1)} = \begin{pmatrix} T_i^2 & T_i & 1 \\ 2T_i & 1 & 0 \\ \frac{T_i^3 - T_{i-1}^3}{3(T_i - T_{i-1})} & \frac{T_i^2 - T_{i-1}^2}{2(T_i - T_{i-1})} & 1 \end{pmatrix}; \quad \text{and } A_i^{(2)} = \begin{pmatrix} T_i^2 & T_i & 1 \\ 2T_i & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

An illustration of periodic quadratic splines satisfying the above conditions is given in Figure 2.29 for time from 0 to 2, split into the two intervals $[0, 1)$ and $[1, 2)$ (*i.e.* $T_0 = 0$, $T_1 = 1$, $T_2 = 2$).

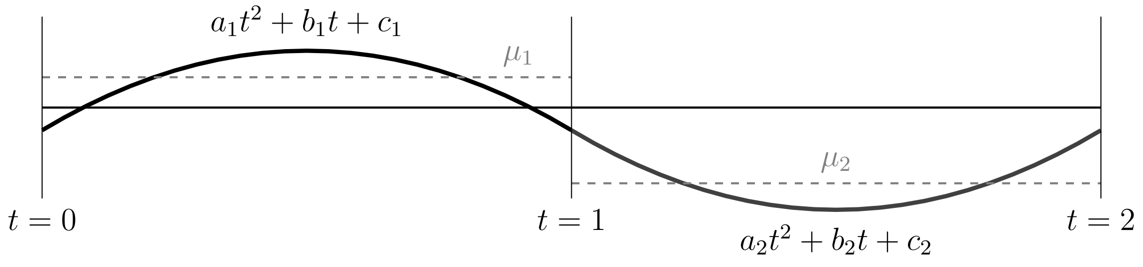


Figure 2.29: An example of quadratic splines for $t \in [0, 2]$ with the space partitioned into the intervals $\{[0, 1), [1, 2)\}$. Here a_i, b_i, c_i, $i = 1, 2$, is chosen such that the expectation and boundary conditions are satisfied.

2.3.5 Misspecified models and removing seasonal effects

This section considers Ornstein-Uhlenbeck processes with some seasonality, where it is assumed that only one observation per 'season' is available. Here it is shown by way of an example why it is important to specify the model that the observations are from correctly, and remove the seasonal mean in a sensible way. Compare the following mean removal and parameter estimation methods.

Take each season observation as being an observation of the block-average Ornstein-Uhlenbeck process (2.38) for that season. That is, the observations $\{y^{\text{baOU}}_{[0,1)}, y^{\text{baOU}}_{[1,2)}, \ldots, y^{\text{baOU}}_{[n-1,n)}\}$ are from a block-average Ornstein-Uhlenbeck process with some seasonal behaviour, with the seasons given by $T_0 = 0$, $T_1 = 1$, $T_2 = 2, \ldots, T_{12} = 12$. For simplicity, assume that the number of observations is divisible by 12 (*i.e.* $n/12 \in \mathbb{Z}$), so that there are an equal number of observations for each season.

The sample mean for the i^{th} season is calculated by

$$\hat{\mu}_i = \frac{1}{\tilde{n}} \sum_{j=0}^{\tilde{n}-1} y^{\text{baOU}}_{[i+12j-1,i+12j)},$$

for $i = 1, \ldots, 12$, where $\tilde{n} = n/12$.

Three different methods are now applied to the observations. In the first two methods the observations are misspecified such that they are assumed to be from a piecewise Ornstein-Uhlenbeck process taken at the midpoint of each season.

Case 1: In the first case, the midpoint of the quadratic splines calculated using the sample means $\hat{\mu}_j$, $j = 1, \ldots, 12$, for each season is subtracted from the observations, to give adjusted observations

$$x_{t_i}^{(1)} = y_{[i-1,i)}^{\text{baOU}} - m(t_i),$$

$i = 1, \ldots, n$, where $m(t)$ is as (2.41) and $t_i = (2i-1)/2 = i - 1/2$. Note that in this case, the adjusted observations will not have mean zero in each season, as in general the value of $m(t)$ at a season midpoint is not the same as the respective sample mean $\hat{\mu}_j$ (this is shown in Example 2.3.10 below). Therefore in this first case the observations are misspecified in two different ways.

Case 2: In the second case, the sample means are removed from the seasonal observations directly, giving adjusted observations,

$$x_{t_i}^{(2)} = y_{[i-1,i)}^{\text{baOU}} - \hat{\mu}_i,$$

for $i = 1, \ldots, n$. Here, the seasonal means of the adjusted observations will be zero, although the observations are still misspecified as being from a pointwise Ornstein-Uhlenbeck process. The differences between $x_{t_i}^{(1)}$ and $x_{t_i}^{(2)}$ are illustrated in Figure 2.30, which is part of Example 2.3.10 below.

In both Case 1. and 2., we apply sparse maximum likelihood estimation to the adjusted observations $\{x_{t_1}^{(k)}, x_{t_2}^{(k)}, \ldots, x_{t_n}^{(k)}\}$, $k = 1, 2$, by maximising the log-likelihood (2.28).

Case 3: In the third case the observations are correctly identified as from a block-average Ornstein-Uhlenbeck process, and the sample means for each season are removed. This gives adjusted observations,

$$x_{[i-1,i)}^{\text{baOU}} = y_{[i-1,i)}^{\text{baOU}} - \hat{\mu}_i,$$

with $i = 1, \ldots, n$, and where seasonal means for the realisation are now zero. We then calculate the parameter estimates using maximum likelihood estimation for block-average Ornstein-Uhlenbeck processes as in (2.33).

Example 2.3.10 (Wrongly specified "shifted" block-average Ornstein-Uhlenbeck process)**.** *In this example the following process is repeated over* 100 *iterations.*

1. *A set of observations* $\{x^{baOU}_{[0,1)}, x^{baOU}_{[1,2)}, \dots, x^{baOU}_{[119,120)}\}$ *from a block-average Ornstein-Uhlenbeck process with parameter values* $\gamma = 1$, $\sigma^2 = 100$, *is simulated by a multivariate Gaussian with mean zero and covariance* $\Sigma_{\gamma,\sigma^2} = (\sigma_{ij})_{i,j}$ *with* $\sigma^2_{ij} = ca^{|j-i|} + b\mathbb{1}_{i=j}$ *as in* (2.18)*. Note that this replaces the first two steps that were previously used in Example 2.3.9 to simulate a block-average Ornstein-Uhlenbeck process.*

2. *Seasonal behaviour is added to these observations by*

$$y^{baOU}_{[i-1+12k,i+12k)} = x^{baOU}_{[i-1+12k,i+12k)} + \mu_i,$$

for each $i \in \{1, \dots, 12\}$ *and* $k \in \{0, \dots, \tilde{n}\}$, *with* $\tilde{n} = 10$*. One realisation* $\{y^{baOU}_{[0,1)}, \dots, y^{baOU}_{[119,120)}\}$ *is shown in Figure 2.31.*

3. *Estimated seasonal means are calculated by*

$$\hat{\mu}_i = \frac{1}{10} \sum_{k=1}^{10} y^{baOU}_{[i-1+12k,i+12k)},$$

for $i = 1, \dots, 12$, *and the associated quadractic spline* $m(t)$ *is calculated following the procedure in Section 2.3.4.2. Figure 2.30 shows the quadratic spline function* $m(t)$ *for* $t \in [0, 12]$ *(black line), and the values at the seasonal midpoints* $\{m(t_i) \colon i = 1, \dots, 12\}$ *(black points) calculated using one set of seasonal means* $\hat{\mu}_1, \dots, \hat{\mu}_{12}$ *(shown by the horizontal lines).*

4. *Obtain "pointwise" Ornstein-Uhlenbeck observations for Case 1. and Case*

2.,

$$\begin{cases} x_{t_i}^{(1)} = y_{[i-1,i)}^{baOU} - m(t_i); \\ x_{t_i}^{(2)} = y_{[i-1,i)}^{baOU} - \hat{\mu}_{k(i)}, \end{cases}$$

$t_i = i - 1/2$, *for* $i \in \{1, \dots, 120\}$, *and with*

$$k(j) = \begin{cases} j \, mod(12) \quad for \; j = 1, 2, \dots, 11; \\ 12 \quad for \; j = 12. \end{cases}$$

5. *Obtain block-average Ornstein-Uhlenbeck observations* $y_{[i-1,i)}^{baOU} - \hat{\mu}_{k(i)}$, *for* $i \in \{1, \dots, 120\}$.

6. *Apply sparse maximum likelihood estimation as in* (2.28) *on the "pointwise" observations* $\{x_{t_1}^{(1)}, \dots x_{t_{120}}^{(1)}\}$ *and* $\{x_{t_1}^{(2)}, \dots x_{t_{120}}^{(2)}\}$ *for Case 1. and Case 2 respectively. For Case 3. apply block-average maximum likelihood estimation as in* (2.32) *on the observations* $\{y_{[0,1)}^{baOU} - \hat{\mu}_1, \dots, y_{[119,120)}^{baOU} - \hat{\mu}_{12}\}$. *These each give drift and diffusion parameter estimates* $\hat{\gamma}$ *and* $\hat{\sigma}^2$ *for the original Ornstein-Uhlenbeck process.*

Following this procedure, means and standard deviations for the parameter estimates found for each case over 100 iterations are given in Table 2.32. Recall that the true Ornstein-Uhlenbeck drift and diffusion here is $\gamma = 1$ and $\sigma^2 = 100$. From these results it is clear that misspecifying the observations as from a pointwise Ornstein-Uhlenbeck process results in inaccurate maximum likelihood estimates for γ and σ^2, whereas if the observations are (correctly) identified as from a block-average Ornstein-Uhlenbeck process then the associated maximum likelihood approach works relatively well in this case. Note that the sparse maximum likelihood method applied to the observations with either the seasonal mean, or the spline at the season midpoint gives very similar results. This suggests that, in this case, the error introduced from model misspecification was significantly larger than that introduced by removing the incorrect mean value.

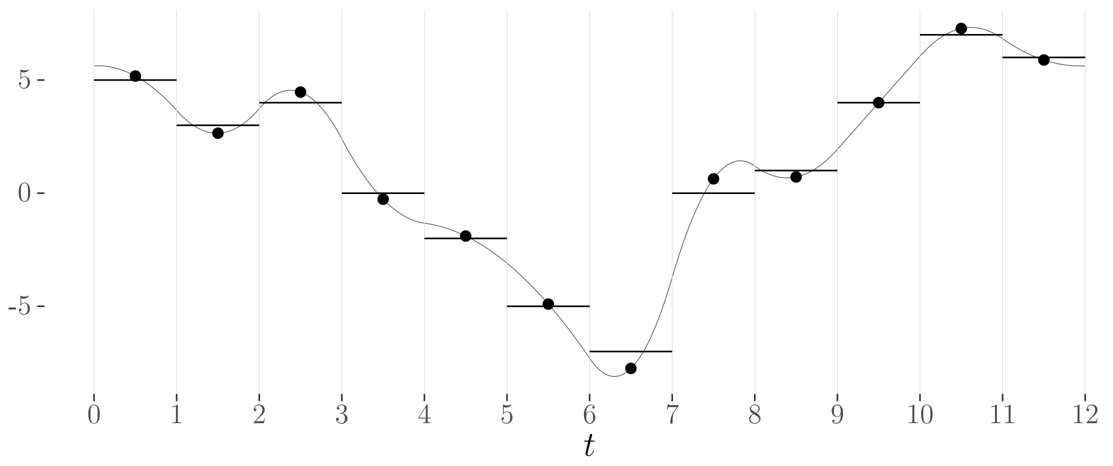


Figure 2.30: Quadratic spline (black line) calculated using the sample means in each time interval (black horizontal lines). The value of the quadratic splines at each season midpoint is shown by the black circles, with the grey vertical lines giving the boundary between seasons, $T_0 = 0, T_1 = 1, \ldots, T_{12} = 12$. The gaps between the horizontal lines and the black circles highlight the differences between the seasonal (sample) means and the midpoints of the quadratic splines.

In this chapter we introduced stochastic processes which we will use to model the principal and independent components obtained from some HadCM3 simulations in Part II. After the use of some stochastic process has been justified to model these principal or independent components, the next step involves obtaining forecasts of the components using these models. We now discuss forecasting techniques in Chapter 3.

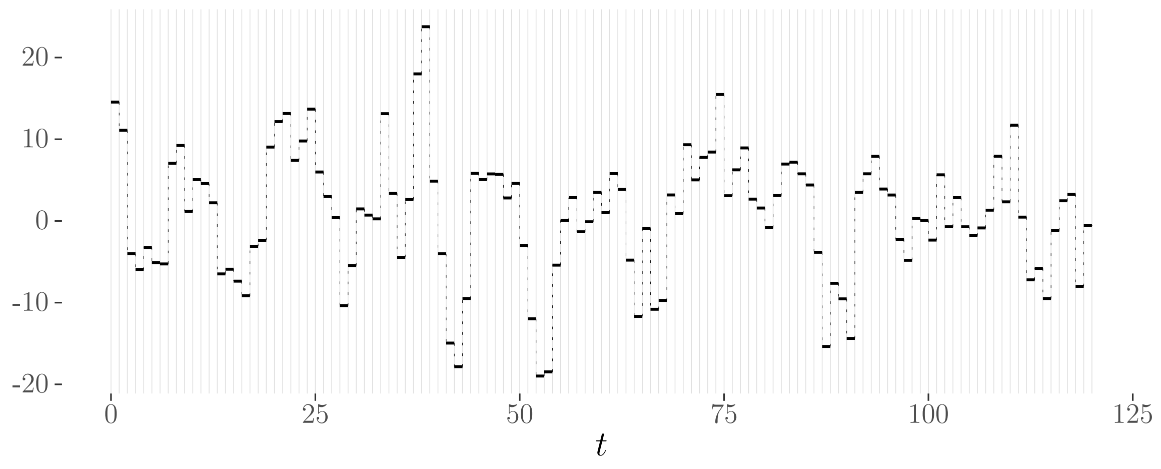


Figure 2.31: One realisation of the seasonal block-average Ornstein-Uhlenbeck process (solid black horizontal lines) with the seasons bounded by the grey vertical lines. That is, the horizontal lines show $\{y_{[0,1)}, y_{[1,2)}, \dots, y_{[n-1,n)}\}$, with $n = 120$, and seasonal behaviour that is split into twelve seasons with boundaries $T_j = j$, $j = 0, 1, \dots, 12$.

	Observations	$\bar{\gamma}$	$\mathrm{sd}(\bar{\gamma})$	$\bar{\sigma}^2$	$\mathrm{sd}(\bar{\sigma}^2)$
Case 1:	$\{x_{t_1}^{(1)}, \dots, x_{t_{120}}^{(1)}\}$	0.633	0.128	45.879	6.980
Case 2:	$\{x_{t_1}^{(2)}, \dots, x_{t_{120}}^{(2)}\}$	0.625	0.128	45.099	7.049
Case 3:	$\{x_{[0,1)}, \dots, x_{[119,120)}\}$	1.027	0.210	103.588	19.774

Figure 2.32: Table showing maximum likelihood parameter estimation mean and standard deviation over 100 iterations for the three cases. The true parameter values here are $\gamma = 1$ and $\sigma^2 = 100$. For Case 1. and Case 2., sparse maximum likelihood estimations as in (2.25) were applied to observations $\{x_{t_1}^{(1)}, \dots x_{t_{120}}^{(1)}\}$ and $\{x_{t_1}^{(2)}, \dots x_{t_{120}}^{(2)}\}$ respectively, for each iteration. For Case 3., observations $\{y_{[0,1)}^{\text{baOU}} - \hat{\mu}_1, \dots, y_{[119,120)}^{\text{baOU}} - \hat{\mu}_{12}\}$ were used, and the block-average Ornstein-Uhlenbeck log-likelihood (2.32) was maximised for each iteration.

Chapter 3

Forecasting

In Chapter 1, we introduced techniques to reduce the size of a large simulator output or data-set, to some set of one-dimensional time series. In Chapter 2 we introduced theory on modelling one-dimensional time series, either using classical techniques such as ARMA models or by assuming that the time series belonged to some class of stochastic differential equations. This chapter on forecasting is motivated by the following scenario that is the main thread that runs through this thesis and arises from the preceding two chapters.

Suppose we have an output from some large simulator and have applied the theory as discussed in Chapter 1 to reduce this output to a set of low dimensional projections. We have then modelled these projections using the stochastic process theory from Chapter 2, assuming random noise replaces the information lost from the projection into a smaller subspace. Now, ***we require forecasts of the low dimensional projections using these models****, and then use these forecasts to reconstruct some future behaviour of the high dimensional simulator output.*

In this chapter we focus on the step that is in **bold** in the above motivation.

That is, the aim is to forecast the value X_{n+h}, where h is known as the *forecast horizon*, using the observations $x^{(n)} = \{x_j \colon j \in \mathbb{Z}, j \leq n\}$. Here, X_{n+h} is an unknown random variable and $x^{(n)}$ is a set of known values. Following Chatfield (2000), the point forecast of X_{n+h} is notated $\hat{x}_n(h)$. Using this notation gives both the time up to which observations are avaliable by the subscript n, as well as the time of the forecast by $n + h$, in a non-ambiguous way.

This chapter is arranged as follows. In Section 3.1 we introduce some notation and discuss model specification and diagnostic techniques, including the Box-Jenkins method. In Section 3.2 we describe *point forecast* theory for ARMA processes (introduced in Section 2.2.1.3), and for Ornstein-Uhlenbeck processes (from Section 2.3.2). In Section 3.3, we define *forecast intervals* and interval estimation methods are introduced for both ARIMA and Ornstein-Uhlenbeck processes, as well as more general interval estimation methods based on simulation. In Section 3.2.2.1 we introduce novel one-step-ahead forecast theory using the distributional properties of the block-average Ornstein-Uhlenbeck process, with the associated forecast intervals given in Section 3.3. Lastly, In Section 3.4 we give some examples of the various estimated point and interval forecasts available, applied to ARMA and Ornstein-Uhlenbeck processes.

3.1 Model Suitability

The first remark made in this section is to distinguish between forecasting *models* and *methods*. As discussed in Chatfield (2000), a forecasting model is some mathematical construct that is (subjectively) chosen to represent the "true" behaviour of the time series being forecast. On the other hand, a forecasting method removes the requirement of mirroring a "true" model, and instead is just some rule or formula that is used to compute a forecast. Therefore, a forecasting method does not attempt to explicitly represent some underlying reality. In this chapter we will focus on using forecasting models and tie these models into the time series and stochastic process theory discussed in Chapter 2.

Mainly for simplicity, here we only consider univariate forecasting models, where the forecast of a time series in $\mathbb{R}$ is found only using present and past observations from that individual time series. This compares to the multivariate case, where there can be more than one time series, or explanatory variable, used to forecast a given variable. As well as simplicity, another reason for choosing only univariate forecasting models is so that we do not stray too far into an area that motivated the original dimension reduction of the high-dimensional simulator. This area includes; increased opportunity for mathematical errors and model misspecification, high parameter uncertainty, and hard-to-determine model conditioning. For an introduction to multivariate time series and forecasting models, see Chatfield (2000) and Priestley (1981, Volume 2).

In the univariate case a function f is required that depends on the time series model M and the present and past observation $x^{(n)}$, such that the forecast is given by $\hat{x}_n(h) = f(x^{(n)}; M)$ and satisfies some optimality criteria. One common approach is to choose the forecasts to minimise mean square error, given by $\mathsf{E}\big(X_{n+h} - \hat{x}_n(h)\big)^2$ which is minimised when $\hat{x}_n(h) = \mathsf{E}(X_{n+h}|x^{(n)})$ (Priestley, 1981). This choice of forecast is by no means unique, and in fact relates to the assumption of a quadratic loss function. A loss function is a function that gives some value to an event, representing the "cost" of the event occuring, with a smaller value representing a better outcome. A quadratic loss function uses a quadractic curve to penalise events further away from the optimal event proportionally more than those close to the optimal. In some scenarios this may not be a sensible choice, and a non-symmetric or $0-1$ loss function may be preferable. For example, in a medical setting it is often better for a patient to have a false positive diagnosis than a false negative, suggesting that the loss function should be larger for false negatives than false positives.

An obvious area that needs to be addressed before finding forecasts is whether the chosen model M is the correct one to describe a set of observations $x = \{x_0, x_1, \ldots, x_n\}$. One way to inform this decision is by using *model criterion*, such as adjusted-R^2 and Akaike Information Criterion (Akaike, 1969). The Akaike Information Criterion is given by,

$$\mathrm{AIC} = -2\log\Big(\max\big(L(\cdot\,; x_0, \ldots, x_n, M)\big)\Big) + 2p$$

where $L(\cdot\,; x_0, \ldots, x_n, M)$ is the likelihood of the observations given the model, and p is the number of independent model parameters. Models that result in a low Akaike Information Criterion are generally considered to be better. Often, the term $-2\log L(\cdot\,; x_0, \ldots, x_n, M)$ is approximated by $n\log(s/n)$, where s is the residual sum-of-squares. The Akaike Information Criterion bias-corrected version is frequently used, given by

$$\text{AIC}_c = \text{AIC} + \frac{2(p+1)(p+2)}{(n-p-2)},$$

which penalises complex models describing short time series.

3.1.1 Choosing ARIMA model structure: the Box-Jenkins method

This section describes the Box-Jenkins method, which is an iterative procedure to obtain a suitable model $M \in \mathcal{M}$ to explain some set of observations, where $\mathcal{M}$ is the ARIMA class of models. It is discussed in detail in Box *et al.* (2015, Part II.). The basic iterative approach can be described in three steps:

Panel 3.11: Box-Jenkins procedure

1. *Identification:* using the time series observations and prior knowledge of time series models, determine a model that might explain the behaviour of the observations. That is, identify a model $M \in \mathcal{M}$;
2. *Estimation:* using the time series observations and the selected model class, estimate the model parameters. That is, use parameter approximation techniques to obtain a model $\hat{M}$.
3. *Diagnostic checking:* check for model inadequacies by comparing the fitted model to the true model. That is, check whether the approximate model $\hat{M}$ describes the data sufficiently well.

Here, the data is assumed to belong to the class of ARIMA models, as described in Section 2.2.2.1. Then, the *identification* step simplifies to first differencing the time series to obtain stationarity (and thus an ARMA process), and then obtaining estimations for the number of autoregressive and moving average parameters, p and q respectively. That is, we suppose initially that the time series observations $\{y_1, y_2, \ldots, y_n\}$ can be modelled as being from the ARIMA(p, d, q) process $(\mathsf{Y}_t)_{t\in\mathbb{Z}}$, given by $\varphi(B)(1-B)^d\mathsf{Y}_t = \vartheta(B)\varepsilon_t$ (as in (2.10)), where φ and ϑ are polynomial operators of degree p and q respectively, and B the backwards shift operator such that $B^d \mathsf{Y}_t = \mathsf{Y}_{t-d}$.

The process $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ given by $\mathsf{X}_t = (1-B)^d \mathsf{Y}_t$ is then an ARMA(p,q) process, which is used to model the differenced observations $x_i = (1-B)^d\, y_{i+d}$, $i = 1, 2, \ldots, n-d$. Once the value of d has been estimated to give the observations $\{x_1, x_2, \ldots, x_n\}$, the degrees of the operators $\varphi(B)$ and $\vartheta(B)$ can be found. We now describe how the differencing and operator degrees can be estimated by examining the sample autocorrelation function and sample partial autocorrelation function.

To obtain the order of differencing, we examine the sample autocorrelation function. If this function does not die out quickly this suggests that the process might not be in stationarity. This was shown in Example 2.2.6. From Box *et al.* (2015), in general it is expected that the order of differencing, d, is low, normally 0, 1, or 2. Box *et al.* (2015) suggests only looking at around the first 20 autocorrelations of the observations and of the differenced observations of order 1 and 2, if required.

Once the observations have been differenced such that stationarity is obtained, the values of the autoregressive and moving average degrees are estimated. Note that for an autoregressive process with degree p, the sample autocorrelation function should decrease in some exponential (and potentially sinuous) manner with respect to the lag, and the sample partial autocorrelation function should be zero for all lags that are $(p+1)$ and above. For a moving average process with degree q, the sample autocorrelation function should be zero for for all lags that are $(q+1)$ and above. As discussed in Section 2.2.1.3, if the sample autocorrelation function appears to be decaying by a mixture of exponentials and damped sine waves only after a number of lags, this suggests that both p and q are greater than zero and the (differenced) observations $\{x_1, x_2, \ldots, x_n\}$ belong to an ARMA process. In this case the sample autocorrelation function is expected to behave like one calculated from a pure autoregressive process after lag $q-p$.

We remark here that this model identification step is non-trivial, and interpreting sample autocorrelation and partial autocorrelation functions is in general hard to do. This is highlighted in Example 3.1.1 below. One method that is sometimes used to avoid this step – and indeed to avoid one's subjective judgement being challenged – is to consider a range of models and choose the one that minimises some model criterion, for instance adjusted-R^2 or Akaike Information Criterion as described in Section 3.1.

Once the order (p, d, q) of the ARIMA model has been chosen, the *estimation* step involves finding estimates for the φ_i, $i = 1, 2, \ldots, p$ and ϑ_j, $j = 1, 2, \ldots, q$, which is often achieved using maximum likelihood estimation. For details of this step, see Box *et al.* (2015, Section 7) and Brockwell *et al.* (1991, Section 8).

In the *diagnostic checking* step, it is required to determine both whether the order (p, d, q) (*i.e.* the model M) and parameter estimates (*i.e.* $\hat{M}$) seem sensible, and if not, in what way they might be incorrect. This can be done by, for example, overfitting the observations by choosing a more complicated model and seeing if this gives better results (Box *et al.*, 2015, Section 8.1.2). Also, one can look at the residuals of the data to check whether these satisfy the properties of white noise, or apply common residual tests such as Shapiro-Wilk (Shapiro & Wilk, 1965), Ljung-Box (Ljung & Box, 1978) or Breusch–Godfrey (Godfrey, 1978).

Example 3.1.1 (Box-Jenkins method on Example 2.2.6)**.** *In this example the Box-Jenkins method is applied to the process* $(\mathsf{Y}_t)_{t\in T}$*, which is an integrated autoregressive-moving average process of the form,*

$$(1 - B)\mathsf{Y}_t = \varphi(1 - B)\mathsf{Y}_{t-1} + \varepsilon_t + \vartheta\varepsilon_{t-1},$$

with $\varphi = 0.7$*,* $\vartheta = 0.8$ *and* $\varepsilon_j \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ *for* $j \in T$*,* $\sigma_\varepsilon^2 = 0.4$*, and* B *the backwards shift operator. This is the same process as we considered in Example 2.2.6, and the first difference,* $\mathsf{X}_t = (1 - B)\mathsf{Y}_t$*, is the* ARMA$(1, 1)$ *process considered in Example 2.2.5.*

Here, we focus on the identification and diagnostic checking steps in Panel 3.11. This example will proceed as follows:

1. *Estimate the order of differencing,* d*, using the sample autocorrelation function;*

2. *Estimate the autoregressive order, p, and the moving average order, q, using the sample autocorrelation and sample partial autocorrelation functions;*

3. *Estimate the parameter values* $\varphi_1, \dots, \varphi_p$ *and* $\vartheta_1, \dots, \vartheta_q$*;*

4. *Use the Akaike information criterion, introduced in Section 3.1, and tests for Gaussianity of the residuals for the diagnostic checking step to determine whether the chosen model is a suitable fit for the data.*

These steps are repeated until the Akaike information criterion indicates that the chosen model and estimated model parameters are sensible in relation to the data. As the Akaike information criterion gives a sense of the trade-off between model complexity and goodness of fit, a "sensible" value is very much a reasoned choice that is needed to be made by the modeller.

This example is presented here to highlight some issues that can occur when applying the Box-Jenkins method to data, and how uncertainty over the "correct" way to proceed can arise.

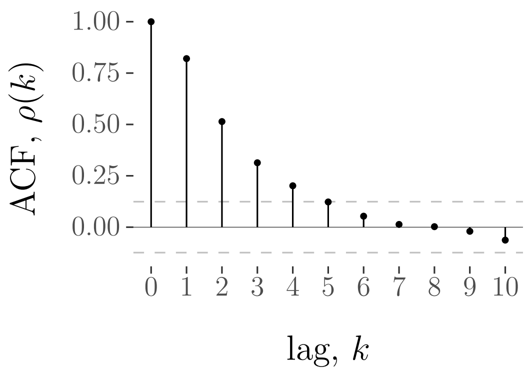
Suppose we have the realisation $\{y_1, y_2, \dots, y_{250}\}$ *of the process* $(Y_t)_{t\in T}$ *(which is the same realisation as seen in Example 2.2.6). The aim here – assuming these observations are from an* $ARIMA(p, d, q)$ *process – is to use these observations to estimate p, d and q. Using the autocorrelation and partial autocorrelation functions to estimate the order of an ARIMA process was briefly discussed in Section 2.2. Following the Box-Jenkins procedure (Panel 3.11), the order of differencing is estimated first.*

The realisation of the $ARIMA(1, 1, 1)$ *process,* $\{y_1, y_2, \dots, y_{250}\}$*, was shown in Figure 2.11 and the associated sample autocorrelation function in Figure 2.12. The sample autocorrelation function exhibits a very slow decay, indicating – as discussed in Section 2.2 – that the process is not stationary. Taking the first difference gives observations* $\{x_1, x_2, \dots, x_{250}\}$*, where* $x_t = (1 - B)y_t$*,* $t = 1, \dots, 250$*,* *shown in Figure 2.8 (Example 2.2.5). The sample autocorrelation function for the differenced observations was given in Figure 2.9 (reproduced here in Figure 3.1a) and exhibits fast decay. This speed of decay suggests that the order of differencing is $d = 1$ and the process of which $\{x_1, \ldots, x_{250}\}$ is a realisation is stationary. With this information, it is now assumed that the observations $\{x_1, \ldots, x_{250}\}$ are from an ARMA(p, q) process, where p and q are to be determined.*

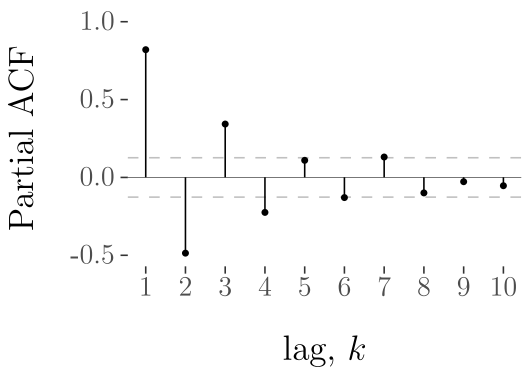
Recall from Section 2.2.1.3, the autocorrelation function is a mixture of exponentials and damped sine waves after the $(q-p)^{th}$ lag, and the partial autocorrelation function is dominated by a mixture of exponentials and damped sine waves after the $(p-q)^{th}$ lag.

Using this information, consider the sample autocorrelation and partial autocorrelation functions as shown in Figure 2.9 and Figure 2.10 respectively. They are reproduced here in Figure 3.1 up to lag 10 for convenience. In this case, to anyone bar an expert in this field, it is often challenging to determine when the sample autocorrelation function (Figure 3.1a), becomes a mixture of exponentials and damped sine waves, and when the sample partial autocorrelation function (Figure 3.1b), becomes dominated by a mixture of exponentials and damped sine waves. It can quite convincingly be argued that both the sample autocorrelation (barring lag 1) and partial autocorrelation functions are wholly dominated by exponentials and damped sine waves. That is, $q - p = 0$ (as $\rho(0) = 1$ by definition, so that the exponential and damped sine wave mixture starts at lag 1) and $p - q = 0$.

However, here suppose the observations x_t, $t = 1, \ldots, 250$, are initially (mis)-specified as being from an ARMA$(2, 2)$ process, so that $q - p = p - q = 0$. Here the value of $p = q = 2$ is chosen to illustrate one way the observations could be specified as to an incorrect underlying process. Using these observations the



(a) Reproduction of sample autocorrelation function, Figure 2.9, Example 2.2.5.



(b) Reproduction of sample partial autocorrelation function, Figure 2.10, Example 2.2.5.

Figure 3.1: Reproduction of the sample autocorrelation and partial autocorrelation functions from Example 2.2.5, Figure 2.9 and Figure 2.10 respectively.

following parameter estimates are found,

$$\hat{\varphi}_1 = -0.1869;\ \hat{\varphi}_2 = 0.5250;\ \hat{\vartheta}_1 = 1.7513;\ \hat{\vartheta}_2 = 0.8063.$$

The Akaike information criterion for this process order and parameter value estimates is $AIC = -111.5432$, with the bias-corrected version $AIC_c = -111.3799$.

The steps are now repeated to check whether the initial model and parameter estimates are a good fit in relation to the data. Looking back at the sample autocorrelation and partial autocorrelation functions in Figure 3.1 (originally from Example 2.2.5) and referring to Table 6.1 in Box & Jenkins (1976), notice that the behaviour of the two functions follows that of a $ARIMA(1, d, 1)$ process. That is, the sample autocorrelation function decays exponentially from the first lag, and the sample partial autocorrelation function is dominated by exponential decay from the first lag.

Using this information, suppose the (differenced) observations x_t, $t = 1, \ldots, 250$

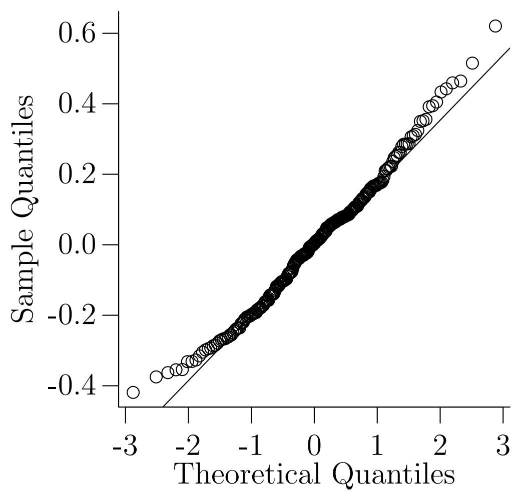
are assumed to be from an $ARMA(1, 1)$ *process. Then, the following parameter estimates are found,*

$$\hat{\varphi} = 0.6761; \ \hat{\vartheta} = 0.8036.$$

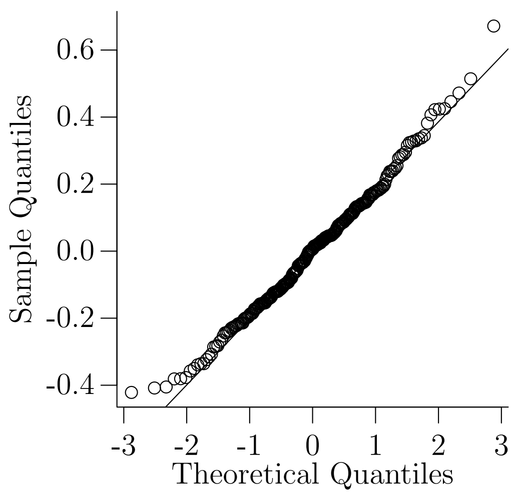
Recall that the true values are $\varphi = 0.7$ *and* $\vartheta = 0.8$*. From this choice of model and estimated model parameters, the Akaike information criterion is* $AIC = -107.7376$ *and bias-corrected version,* $AIC_c = -107.6404$*.*

From the Akaike information criterion alone, it appears that choice of the $ARIMA(2, 1, 2)$ *process gives an estimated model that fits the data better than the choice of* $ARIMA(1, 1, 1)$*, which is the true model. However, it is always prudent to look at the residuals as part of the diagnostic checking step. Applying the Shapiro-Wilk test for normality (Shapiro & Wilk, 1965) to the residuals from the* $ARIMA(2, 1, 2)$ *and* $ARIMA(1, 1, 1)$ *models gives p-values of* 0.0569 *and* 0.165 *respectively. With the "standard" significance level of* 0.05*, the Shapiro-Wilk test suggest that both of these models have residuals that are Gaussian distributed. Note that in this case one cannot compare the two p-values obtained to one-another, as under the null hypothesis the distribution of the p-value is uniform and thus a p-value of* 0.165 *is not "stronger proof" of residual Gaussianity than one of* 0.0569*. However, Royston (1995) suggests that the default significance level for a Shapiro-Wilk test should be set at a p-value of* 0.1*, in which case the tests on the two models suggest that the choice of* $ARIMA(1, 1, 1)$ *is preferable. Looking at the Q–Q plot of the residuals, given in Figure 3.2, the residuals from the choice of* $ARIMA(1, 1, 1)$ *model (Figure 3.2b) seem to be a slightly better fit than those from the* $ARIMA(2, 1, 2)$ *models (Figure 3.2a). This, along with the result from the Shapiro-Wilk test, indicates that the observations can be modelled as a realisation from an* $ARIMA(1, 1, 1)$ *process. A different conclusion would be reached if only the Akaike information criterion results were used.*

As explained at the beginning of this example, we applied the Box-Jenkins



(a) Q-Q plot of the residuals, assuming the observations are from an ARIMA$(2, 1, 2)$ process and estimating the parameter values.



(b) Q-Q plot of the residuals, assuming the observations are from an ARIMA$(1, 1, 1)$ process and estimating the parameter values.

Figure 3.2: Q-Q plots of the residuals when assuming the observations come from either a ARIMA$(2, 1, 2)$ process or a ARIMA$(1, 1, 1)$ process and estimation the associated parameter values.

method for a realisation from a simple ARIMA$(1, 1, 1)$*, demonstrating the ambiguity and somewhat arbitrary decisions that the lay-person can often face. In this example, different choices for model diagnostic checks and different choices within these checks can result in different models being selected.*

The model is now assumed to have been chosen, and thus the following sections consider forecasts based on the chosen model.

3.2 Calculating Point Forecasts

This section deals with calculating (approximate) point forecasts using a set of observations, assuming these observations can be modelled according to some underlying stochastic process.

Following the structure of Chapter 2, this section is split into point forecasts relating to the time series ARMA processes in Section 3.2.1, and the Ornstein-Uhlenbeck process (both pointwise and block-average) in Section 3.2.2.

3.2.1 Forecasting ARMA processes

Here we use an ARMA process (introduced in Section 2.2.1.3) to obtain some forecasts. Recall that a stationary ARMA(p, q) process is given by $\mathsf{X}_t = \varphi_1 \mathsf{X}_{t-1} + \ldots + \varphi_p \mathsf{X}_{t-p} + \varepsilon_t + \vartheta_1 \varepsilon_{t-1} + \ldots \vartheta_q \varepsilon_{t-q}$, where $\varphi_1, \ldots, \varphi_p$ and $\vartheta_1, \ldots, \vartheta_q$ are the model parameters, and $\varepsilon_j \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ independent for some $\sigma_\varepsilon^2 \in (0, \infty)$. With B the backwards shift operator such that $B^j \mathsf{X}_t = \mathsf{X}_{t-j}$ for $j = 1, 2, \ldots$, the ARMA process can be expressed as $\varphi(B)\mathsf{X}_t = \vartheta(B)\varepsilon_t$, where $\varphi(B)$ and $\vartheta(B)$ are polynomial operators in B of degrees p and q respectively.

As described in Section 2.2.1.3, the process $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ can be rewritten as an MA(∞) process, as $\mathsf{X}_t = \zeta(B)\varepsilon_t = \sum_{t=0}^{\infty} \zeta_t \varepsilon_t$, with $\zeta(B) = \varphi^{-1}(B)\vartheta(B)$. In this form, from Box *et al.* (2015), the forecast with horizon h that minimises mean square error is given by

$$\hat{x}_n(h) = \sum_{j=h}^{\infty} \zeta_j \varepsilon_{n+h-j}. \tag{3.1}$$

The forecast written in this way cannot be used in practice as it requires the parameters ζ_j, $j = h, h+1, \ldots$ and the countably large number of observations $x^{(n)}$ to be known exactly so that the innovations $\varepsilon_n, \varepsilon_{n-1}, \varepsilon_{n-2}, \ldots$ can be obtained.

We now explore an alternative way of finding the point forecast $\hat{x}_n(t)$, which also benefits from a simple approximation method. For general ARMA processes the point forecasts $\hat{x}_n(h)$ can be found recursively for horizons $h = 1, 2, 3, \ldots$ directly from the model equation. This is achieved by replacing all future values of the white noise term ε_t, $t > n$ by its expectation, which here is equal to zero, and substituting all future (unknown) values of X_t, $t > n$, by the expectation $\mathsf{E}(\mathsf{X}_t | x^{(n)})$. For example, an ARMA(1, 1) process is described by the equation $\mathsf{X}_t = \varepsilon_t + \varphi \mathsf{X}_{t-1} + \vartheta \varepsilon_{t-1}$, and therefore the recursive approach to finding the minimum mean squared error forecast using the observations $\{\ldots, x_1 \ldots, x_{n-1}, x_n\}$

is given by,

$$\hat{x}_n(h) = \begin{cases} \varphi x_n + \vartheta \varepsilon_n, & h = 1; \\ \varphi \hat{x}_n(h-1), & h \geq 2. \end{cases} \tag{3.2}$$

This forecast still requires knowledge of the parameters φ and ϑ, as well as the innovation ε_n, and thus assumes full knowledge of the model M. In practice, only estimates of the ARMA process parameter values are available, say $\hat{M}$, with estimated parameters $\hat{\varphi}$ and $\hat{\vartheta}$. Using the model $\hat{M}$ also gives estimated innovation $\hat{\varepsilon}_n$, and therefore (3.2) becomes,

$$\hat{x}_n(h) = \begin{cases} \hat{\varphi} x_n + \hat{\vartheta} \hat{\varepsilon}_n, & h = 1; \\ \hat{\varphi} \hat{x}_n(h-1), & h \geq 2. \end{cases} \tag{3.3}$$

3.2.2 Forecasting Ornstein-Uhlenbeck processes

In this section, we consider Ornstein-Uhlenbeck processes $(\mathsf{X}_t^{\mathrm{OU}})_{t \in \mathbb{Z}}$ with the form (2.12), as discussed in Section 2.3.2. We know (from (2.13)) the distribution of the process explicitly. Given observations $x^{(n)} = \{x_{t_1}, x_{t_2}, \ldots, x_{t_n}\}$ from a stationary Ornstein-Uhlenbeck process and using the knowledge of the distribution, the point forecast $\hat{x}_n(h)$ is given by,

$$\hat{x}_n(h) = \mathsf{E}(\mathsf{X}_{n+h}^{\mathrm{OU}} | x^{(n)}) = x_n e^{-\gamma h}. \tag{3.4}$$

As with the ARMA process above, in practice the point forecast for an Ornstein-Uhlenbeck process uses the estimated parameter values, so that (3.4) becomes

$$\hat{x}_n(h) = x_n e^{-\hat{\gamma} h}, \tag{3.5}$$

where $\hat{\gamma}$ is an estimate for the drift parameter γ. Note that the parameter σ^2 (and the estimate $\hat{\sigma}^2$) is not required for the point forecast of an Ornstein-Uhlenbeck process, although it will be required later to obtain forecast intervals in Section 3.3.

3.2.2.1 Forecasting block-average Ornstein-Uhlenbeck processes

Here we find point forecasts for the block-average Ornstein-Uhlenbeck process introduced in Section 2.3.2.1. First, recall that the block-average Ornstein-Uhlenbeck process has a multivariate Gaussian distribution, and therefore forecasts can be found by considering the conditional distribution. We first consider the conditional distribution of a general multivariate Gaussian, before looking at the specific block-average Ornstein-Uhlenbeck case.

Let $\mathsf{X} \in \mathbb{R}^n$ be a multivariate Gaussian random variable with some mean vector μ and covariance matrix Σ, $\mathsf{X} \sim \mathcal{N}(\mu, \Sigma)$, partitioned such that $\mathsf{X} = (\mathsf{X}_1^{(j)}, \mathsf{X}_2^{(j)})^\top$ with $\mathsf{X}_1^{(j)} \in \mathbb{R}^j$ and $\mathsf{X}_2^{(j)} \in \mathbb{R}^{n-j}$, for some $j < n$. Partition the mean vector and covariance matrix in the same way, such that $\mu = (\mu_1^{(j)}, \mu_2^{(j)})^\top$ with $\mu_1^{(j)} \in \mathbb{R}^j$, $\mu_2^{(j)} \in \mathbb{R}^{n-j}$, and

$$\begin{pmatrix} \Sigma_{11}^{(j)} & \Sigma_{12}^{(j)} \\ \Sigma_{21}^{(j)} & \Sigma_{22}^{(j)} \end{pmatrix},$$

where $\Sigma_{11}^{(j)} \in \mathbb{R}^{j \times j}$, $\Sigma_{12}^{(j)} \in \mathbb{R}^{j \times n-j}$, $\Sigma_{21}^{(j)} \in \mathbb{R}^{n-j \times j}$, and $\Sigma_{22}^{(j)} \in \mathbb{R}^{n-j \times n-j}$. Then, it is a well known result (Eaton, 1983, Prop. 3.13) that the conditional distribution of $\mathsf{X}_2 | \mathsf{X}_1 = x_1$ is also a multivariate Gaussian, with mean vector

$$\mu_2 + \Sigma_{21}^{(j)} (\Sigma_{11}^{(j)})^{-1} (x_1 - \mu_1), \tag{3.6}$$

and covariance matrix

$$\Sigma_{22}^{(j)} - \Sigma_{21}^{(j)} (\Sigma_{11}^{(j)})^{-1} \Sigma_{12}^{(j)}. \tag{3.7}$$

We now consider the conditional distribution of a block-average Ornstein-Uhlenbeck process. For finding the one-step-ahead point forecast at time n of the block-average Ornstein-Uhlenbeck process, $\left(\mathsf{X}_{[t_{i-1}, t_i)}^{\mathrm{baOU}}\right)_{i \in \mathbb{Z}}$ (described in Section 2.3.2.1), the conditional distribution of the $(n+1)$-vector, split at $j = n$ is required.

The vector $\left(\mathsf{X}_{[0,\Delta t)}^{\mathrm{baOU}}, \mathsf{X}_{[\Delta t, 2\Delta t)}^{\mathrm{baOU}}, \ldots, \mathsf{X}_{[n\Delta t, (n+1)\Delta t)}^{\mathrm{baOU}}\right) \in \mathbb{R}^{(n+1)}$, has a $\mathcal{N}(0, \Sigma)$ distribution, with $\Sigma \in \mathbb{R}^{(n+1) \times (n+1)}$ the Toeplitz matrix given in (2.18). Given the first n realisations from a block-average Ornstein-Uhlenbeck process and using (3.6), the one-step-ahead point forecast $\hat{x}_n(1)$ is given by,

$$\begin{aligned}\hat{x}_n(1) &= \mathsf{E}\Big(\mathsf{X}^{\text{baOU}}_{[n\Delta t,(n+1)\Delta t)} \,|\, x^{\text{baOU}}_{[(n-1)\Delta t,n\Delta t)}, \ldots, x^{\text{baOU}}_{[0,\Delta t)}\Big) \\ &= 0 + \Sigma^{(n)}_{21}(\Sigma^{(n)}_{11})^{-1} x^{(n)}_1, \end{aligned} \tag{3.8}$$

where

$$x^{(n)}_1 = (x^{\text{baOU}}_{[0,\Delta t)}, x^{\text{baOU}}_{[\Delta t,2\Delta t)}, \ldots, x^{\text{baOU}}_{[(n-1)\Delta t,n\Delta t)}) \in \mathbb{R}^k, \tag{3.9a}$$

$$\Sigma^{(n)}_{11} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_{nn} \end{pmatrix}, \quad \text{and;} \tag{3.9b}$$

$$\Sigma^{(n)}_{21} = (\sigma_{(n+1)\,1}\; \sigma_{(n+1)\,2}\; \cdots\; \sigma_{(n+1)\,n}), \tag{3.9c}$$

with $\sigma_{ij} = c\, a^{|j-i|} + b\mathbb{1}_{i=j}$ as in (2.18).

3.3 Interval Forecasts

In general, a point forecast is not as useful as it might first appear. Crucial questions that are required for a decision to be made using point forecasts are missing, for example; (1) *What does $\hat{x}_n(h) = y$ mean?* (2) *Is the process more likely to be less than or greater than y at time $n+h$?* (3) *How does one compare two models that give different point forecasts, say $\hat{x}^{(1)}_n(h) = y^{(1)}$ and $\hat{x}^{(2)}_n(h) = y^{(2)}$?* These questions all require interval forecasts for answers. In our scenario, where the forecasts are to be used to help reconstruct a high dimensional climate simulator output, interval forecasts are required to give some probability to the reconstructed scenarios, and to be able to compare differing models and forecasts.

This section describes some mathematical theory and common issues when calculating out-of-sample forecast intervals. That is, prescribing some probability for the interval that X_{n+h} could belong to, given observations $x^{(n)}$. Here, X_{n+h} is a random variable conditional on $x^{(n)}$, and the (deterministic) point forecast is given by $\hat{x}_n(h) = \mathsf{E}\big(\mathsf{X}_{n+h}|x^{(n)}\big)$ (described in Section 3.1). Following Chatfield (2000) these intervals are called *prediction intervals* to separate them from the theory of confidence intervals which give an interval for estimates of some fixed parameter values.

The conditional forecast error of a forecast with horizon h given observations $x^{(n)}$ is a random variable given by

$$e_n(h) = \mathsf{X}_{n+h} - \hat{x}_n(h).$$

Here, by convention, the upper-case sans-serif typeface is not used for the conditional forecast error random variable. The within-sample forecast errors are the residuals found from the fitted model, $x_t - \hat{x}_{t-1}(1)$ for $t = 2, 3, \ldots, n$.

To obtain a $100(1-\alpha)\%$ prediction interval for X_{n+h}, in this section it is assumed that the interval is symmetric about the point forecast $\hat{x}_n(h)$, the prediction mean square error is unbiased and that forecast errors are normally distributed. In this case, the $100(1-\alpha)\%$ prediction interval is,

$$\hat{x}_n(h) \pm z_{\alpha/2}\sqrt{\text{Var } e_n(h)}, \tag{3.10}$$

where z_γ is the value for which the standard Gaussian cumulative density function equals $1-\gamma$. Although this looks like a standard confidence interval, the motivation – to help predict the behaviour of a *random variable*, X_{n+h} – differs from the classic confidence interval motivation, which is used to obtain a random interval to help determine some *fixed parameter value*. This difference in motivation merits the different use of language.

For ARMA processes, the uncertainty of the forecast error, given by Var $e_n(h)$ in (3.10), can be simplified to the *prediction mean square error*, $\mathsf{E}\, e_n(h)^2$, in the following way. As the ARMA point forecast $\hat{x}_n(h)$ is unbiased (and thus $\mathsf{E}\, e_n(h) = 0$), the prediction mean square error becomes equal to the forecast error variance, $\mathsf{E}\, e_n(h)^2 = \text{Var } e_n(h)$.

Recall from Section 2.2.1.3 that an ARMA process expressed in MA(∞) form is $x_t = \zeta(B)\varepsilon_t = \sum_{j=0}^{\infty} \zeta_j \varepsilon_{t-j}$, where $\zeta(B) = \varphi^{-1}(B)\vartheta(B)$, with φ and ϑ the autoregressive and moving-average polynomial operators respectively. Using (3.1) the forecast error at horizon time h is $e_n(h) = \varepsilon_{n+h} + \sum_{j=1}^{h-1} \phi_j \varepsilon_{n+h-j}$. From this, the variance term in the prediction interval (3.10) becomes

$$\operatorname{Var} e_n(h) = \mathsf{E}\, e_n(h) = (1 + \phi_1^2 + \cdots + \phi_{h-1}^2)\sigma_\varepsilon^2, \tag{3.11}$$

as $\varepsilon_j \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ are independent for all $j \in \mathbb{Z}$. In practice the parameter values are replaced by their estimates $\hat{\phi}_1, \ldots, \hat{\phi}_{h-1}$ to obtain the forecast error variance.

Now, consider the Ornstein-Uhlenbeck process, with point forecast given by (3.4). The prediction interval here uses both the drift and diffusion parameters of the process, and the explicit distribution derived as a result of (2.13) given in Section 2.3.2. From this, the $100(1-\alpha)\%$ prediction interval (3.10) at horizon h becomes,

$$x_n(h)e^{-\gamma h} \pm z_{\alpha/2}\sqrt{\frac{\sigma^2}{2\gamma}(1 - e^{-2\gamma t})}, \tag{3.12}$$

where z_γ is the value for which the standard Gaussian cumulative density function equals $1 - \gamma$.

In the case of the block-average Ornstein-Uhlenbeck process, the one-step-ahead point forecast is given by (3.8). For the prediction interval, recall that for a multivariate Gaussian random variable $\mathsf{X} \sim \mathcal{N}(\mu, \Sigma)$, the covariance of the condition distribution of $\mathsf{X}_2 | \mathsf{X}_1 = x_1$ for some partition of X is given by (3.7). Now, given a realisation of the first n values from a block-average Ornstein-Uhlenbeck process, the variance of $\mathsf{X}^{\mathrm{baOU}}_{[n\Delta t,(n+1)\Delta t)} \,|\, x^{\mathrm{baOU}}_{[(n-1)\Delta t, n\Delta t)}, \ldots, x^{\mathrm{baOU}}_{[0,\Delta t)}$ is given by

$$\sigma_{(n+1)\,(n+1)} - \Sigma_{21}^{(n)}(\Sigma_{11}^{(n)})^{-1}\Sigma_{12}^{(n)},$$

with σ_{ij}, $i,j = 1,2,\ldots$, given in (2.18), $\Sigma_{11}^{(n)} \in \mathbb{R}^{n\times n}$ as in (3.9b), $\Sigma_{21}^{(n)} \in \mathbb{R}^{1\times n}$ as in (3.9c), and

$$\Sigma_{12}^{(n)} = (\sigma_{1\,(n+1)}, \sigma_{2\,(n+1)}, \ldots, \sigma_{n\,(n+1)})^\top \in \mathbb{R}^n.$$

Therefore, the $100(1-\alpha)\%$ one-step-ahead prediction interval for the block-average Ornstein-Uhlenbeck process is given by

$$\hat{x}_n(1) \pm z_{\alpha/2}\sqrt{\sigma_{(n+1)\,(n+1)} - \Sigma_{21}^{(n)}(\Sigma_{11}^{(n)})^{-1}\Sigma_{12}^{(n)}}, \tag{3.13}$$

with $\hat{x}_n(1)$ as in (3.8) and z_γ the value for which the standard Gaussian cumulative density function equals $1-\gamma$. Note that to obtain the point forecast $\hat{x}_n(1)$ and prediction interval in practice, the values of σ_{ij}, $i,j=1,2,\ldots,n+1$ are estimated using parameter estimations $\hat{\gamma}_{\text{baML}}$ and $\hat{\sigma}^2_{\text{baML}}$ obtained from the n realisations $x^{\text{baOU}}_{[0,\Delta t)},\ldots,x^{\text{baOU}}_{[(n-1)\Delta t,n\Delta t)}$.

3.3.1 Simulating prediction intervals

Approximate prediction intervals for a probabilistic model may be found using a simulation approach which involves using the (assumed "true") model and the (estimated) parameter values to simulate many time series trajectories. Using this set of trajectories, a $100(1-\alpha)\%$ prediction interval at time $n+h$ can be estimated by calculating the range of values that encompass $100(1-\alpha)\%$ of the simulated trajectories. As ARIMA processes and stochastic differential equations such as Ornstein-Uhlenbeck processes are probabilistic time series models, this method can be used on both.

Another simulation approach is called *resampling* and differs to the approach above by sampling from the empirical distribution of the past observed residuals as opposed to sampling using the distribution of the innovations assuming that a specific model is true (which, in our case, would be Gaussian). This empirical distribution can then be used to simulate time series trajectories and calculate prediction intervals in the same manner as above. One type of resampling technique is called *bootstrapping*. In this approach the simulated innovations are taken from the residuals with replacement. The general theory of bootstrapping relies on the observations that are used to sample from being independent, which is most often not the case within a time series environment (unless the time series observations are from a white noise process). However, time series bootstrapping relies on the residuals – as opposed to the observations – being independent, and thus can be used here. When the underlying model is wrongly specified then the residuals may not be independent and resampling approaches, which are highly dependent on model choice, will result in poor prediction intervals.

3.3.2 Comments on prediction intervals

In general, the prediction interval given by (3.10) is too narrow compared to the "true" range of future observations, so that more than $100\alpha\%$ of future observations lie outside a $100(1-\alpha)\%$ prediction interval. Two major reasons for this is that (3.10) assumes that, (i) the model chosen is the "true" model, and (ii) the model parameters are known exactly. For example in the case of ARMA processes, in practice, the parameter estimators $\hat{\phi}_i$, $i = 1, 2, \ldots, h-1$, are substituted directly into (3.10). For sufficiently long observed trajectories and low chosen model complexity, the added uncertainty from using estimated parameter values is normally small compared to uncertainty from other sources. Therefore, it is common practice to input the estimated parameters directly into (3.10).

The wrong model being identified from the observations can also be the cause of prediction intervals that underestimate the range of future values. One common example of a wrongly identified model is by overfitting the data with the assumption that better within-sample fit means better out-of-sample forecasts. This error can be avoided to a certain extent by using diagnostic checks on fitted models, for example the adjusted-R^2 and Akaike Information Criterion as discussed in Section 3.1. On the other hand, the correct model could be identified from the observations, but any prediction interval depends on the assumption that the future is similar to the past. If this is not the case and the model changes at some point then any prediction interval calculated using the original model will not be very useful.

As commented in Chatfield (2000), a way to mitigate against prediction intervals that are too narrow for useful interpretation is to decrease the percentage of future observations that one expects to belong to a prediction interval. That is, reducing a 95% prediction interval to 80% or 90% instead, so selecting an $\alpha = 0.2$ or $\alpha = 0.1$ respectively in (3.10).

3.4 Forecasting Examples

In this section, realisations from ARMA and Ornstein-Uhlenbeck processes are used to obtain point forecasts and prediction intervals. The process realisations are split into training and test sets, and the point forecasts and prediction intervals found using the training sets are compared to the "true" behaviour of the test sets.

Section 3.4.1 considers realisations from an ARMA process, and Section 3.4.2 considers the Ornstein-Uhlenbeck process. In both sections, prediction intervals found using the distribution of the respective process, and by using simulated realisations are considered. In addition, Section 3.4.1 contains an example of using bootstrapping for calculating the prediction intervals. We remark here that all point forecasts appear to reach stationarity very quickly (within the time-frame of interest), such that after a relatively small time horizon the point forecast is close to zero and the prediction intervals are stable.

3.4.1 ARMA(1, 1) process

For this section a ARMA$(1,1)$ process, given by $\mathsf{X}_t = \varphi \mathsf{X}_{t-1} + \varepsilon_t + \vartheta \varepsilon_{t-1}$ with $\varphi = 0.5$, $\vartheta = 0.8$ and $\varepsilon_j \sim \mathcal{N}(0, \sigma_\varepsilon^2 = 0.5)$, is simulated from $t = 0$ to $t = 1000$ giving observations $\{x_0, x_1, \dots, x_{1000}\}$. The realised trajectory is then split into a training set from $t = 0$ to $t = 750$, $x_{\text{training}} = \{x_0, \dots, x_{750}\}$ and a test set from $t = 751$ to $t = 1000$, $x_{\text{test}} = \{x_{751}, \dots, x_{1000}\}$.

From the training set, we obtain parameter approximations, $\hat{\varphi} = 0.5004$ and $\hat{\vartheta} = 0.8068$, and innovation variance estimate $\hat{\sigma}_\varepsilon^2 = 0.4711$.

Example 3.4.1 (Prediction intervals from the true distribution of the process)**.** *Here the known distribution of an ARMA*$(1,1)$ *process is used, alongside the training set, to obtain a prediction interval given by* (3.10)*. The training set is used to obtain approximate parameter values for the ARMA process, which we then input into the known distributional form. This example is split into three steps. The first step is to calculate the point forecasts* $\hat{x}_n(h)$ *for horizons* $h = 1, 2, \dots, H$*. Then, the parameters for the equivalent MA*(∞) *process need to be found. The third step involves plugging the MA*(∞) *parameter values into* (3.11) *and calculating the prediction interval* (3.10)*.*

Note that in this case the estimated parameter values and innovation variance are input directly into (3.10) ignoring any uncertainty at this stage. That is, (3.10) and (3.11) become

$$\hat{x}_n(h) \pm z_{\alpha/2}\sqrt{\hat{V}_e(h)}, \tag{3.14}$$

with $\hat{V}_e(h) = (1 + \hat{\phi}_1^2 + \cdots + \hat{\phi}_{h-1}^2)\hat{\sigma}_\varepsilon^2$.

For step one, the point forecast $\hat{x}_n(h)$ *at horizon* h *that minimises mean square error is found in a recursive manner by (3.3). This gives* $x_n(h)$ *for* $h = 1, 2, \ldots, H$ *where* $H = 250$ *mirrors the length of the test set.*

Now, parameter estimates for the equivalent MA(∞) *process are calculated. Using* $\hat{\varphi}$ *and* $\hat{\vartheta}$*, the MA*(∞) *form (truncated to order* $H-1$*) can be found, giving estimates* $\{\hat{\phi}_j,\ j = 1, 2, \ldots, H-1\}$.

Using the point forecasts $\{\hat{x}_n(h)\colon h = 1, 2, \ldots, H\}$ *and estimated MA*(∞) *parameter values, we find approximations to the mean square error given by* $\{\hat{V}_e(h)\colon h = 1, 2, \ldots, H\}$ *and thus the approximate prediction interval is given by (3.14).*

Figure 3.3 shows the ARMA$(1,1)$ *realisation* $\{x_0, x_1, \ldots, x_{1000}\}$ *for time* $t = 700$ *to* $t = 800$*, split into the training set* $x_{training}$ *(solid black line) and the test set* x_{test} *(dotted black line). The thick grey line gives the point forecasts* $\{\hat{x}_n(h)\colon h = 1, 2, \ldots, 50\}$ *and the dashed lines show the upper and lower values of the* 95% *prediction interval calculated using (3.14).*

For this realisation, 3.6% *of the test values are outside the* 95% *prediction interval, which is less than one would expect from the remarks made in Section 3.3.2. However, it must be noted that this is a contrived example where we have identified the "true" model exactly. Although the use of parameter estimates has not been reflected in the prediction interval, as the estimates have been made from a long time series (*$n = 750$*) with a simple model structure (number of parameters,* $p = 2$*) the added uncertainty should be small.*

Example 3.4.2 (Prediction intervals from simulated realisations)**.** *This is an ex-*

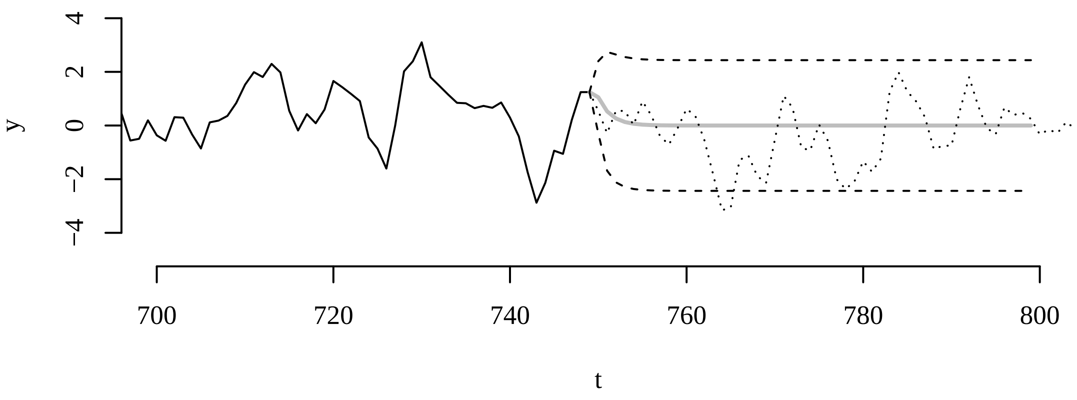


Figure 3.3: This figures shows a realisation of an ARMA$(1, 1)$ process, split into a training set (solid black line) and test set (dotted black line), with the training set used to estimate the model parameters. The estimated parameters have been used to produce a set of point forecasts by (3.3) and to obtain the equivalent MA(∞) process form that allows 95% model prediction intervals to be estimated by (3.14). The thick grey line gives the point forecasts $\{\hat{x}_n(h) \colon h = 1, 2, \ldots, 50\}$ and the dashed lines show the upper and lower values of a 95% prediction interval. Note that for this ARMA$(1, 1)$ realisation, 3.6% of the test set observations $\{x_{751}, \ldots, x_{1000}\}$ lie outside of the 95% prediction interval.

ample to highlight the simulation method discussed in Section 3.3.1. Here we use the estimated ARMA$(1, 1)$ *parameter values* $\hat{\varphi}$ *and* $\hat{\vartheta}$*, and the estimated innovation variance* $\hat{\sigma}^2_\varepsilon$ *to simulate* $N = 100$ *trajectories from an ARMA*$(1, 1)$ *process, all starting from the last observation of the training set,* x_{750}*. Each simulated trajectory has length* $H = 250$ *to mirror the length of the test set. This set of simulated trajectories is used to obtain a point forecast and a prediction interval in the following way.*

For the point forecast at horizon h*, the mean at time* h *of all the* 100 *simulated trajectories is taken. Repeating this for all horizons* $h = 1, 2, \ldots, H$ *gives the set of point forecasts* $\{\hat{x}_n(h) \colon h = 1, 2, \ldots, H\}$*. Here it is remarked that the same* *notation for the point forecasts is used in Example 3.4.1 even though it is defined differently. This is to highlight that both the point forecasts in this and the previous example are an attempt to answer the same fundamental forecasting question.*

To find the simulated $100(1-\alpha)\%$ *prediction interval at horizon* h*, the* $\alpha/2$ *and* $1-\alpha/2$ *quantiles of the simulated trajectories at time* h *are found, such that the most extreme* $(\alpha/2)\%$ *lower and upper values at the time* h *are discarded and the range of the remaining simulated values is taken as the prediction interval at that time.*

Figure 3.4 shows the last 50 *observations from the training set,* $\{x_{700},\dots,x_{750}\}$ *as the solid black line, and the first* 50 *observations from the test set,* $\{x_{751},\dots,x_{800}\}$ *by the dotted black line. The point forecasts calculated as the mean of the simulated values are shown by the thick grey line, with the prediction intervals calculated from the* $\alpha/2$, $1-\alpha/2$ *quantiles of the simulated trajectories given by the dashed lines. In this case* 6.4% *of the test set observations* x_{test} *were outside of the simulated* 95% *prediction interval.*

Example 3.4.3 (Prediction intervals from bootstrapping the residuals)**.** *This example uses the residuals found when assuming the training set realisation* $x_{training}$ *is a realisation of an* $ARMA(1,1)$ *process with the estimated parameter values* $\hat{\varphi}=0.5004$ *and* $\hat{\vartheta}=0.8068$.

The residuals are used in the following way to simulate potential future trajectories of the process. At each time step t the "innovation" used, e_t*, is chosen at random from the fitted values of the training set,* $\{\hat{\varepsilon}_0,\dots,\hat{\varepsilon}_{750}\}$*, sampled with replacement, to give an approximate realisation from the residuals' empirical distribution. This sampled innovation replaces the true innovation in the standard* $ARMA(1,1)$ *method to give the value of the trajectory at time* t *as* $x_t=\varphi x_{t-1}+\hat{e}_t+\vartheta\hat{e}_{t-1}$*. Note that this explicitly removes the assumption that the innovations are Gaussian, although in this example one would hope that this is the case.*

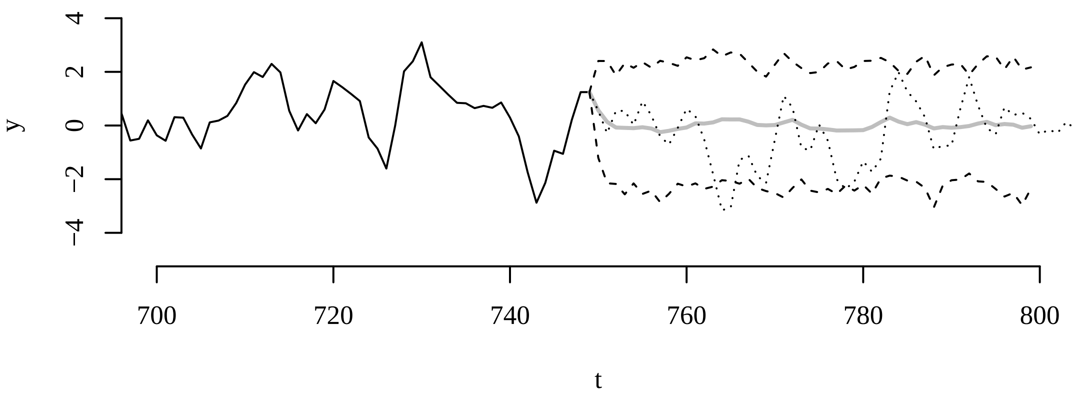


Figure 3.4: This figure shows a realisation of an ARMA$(1,1)$ process, split into a training set (solid black line) and test set (dotted black line), with the training set used to estimate the model parameters. With these estimates, 100 ARMA$(1,1)$ trajectories were simulated and this set of simulations was used to give point forecasts and prediction intervals. The thick grey line gives the point forecasts $\{\hat{x}_n(h) \colon h = 1, 2, \ldots, 50\}$ and the dashed lines show the upper and lower values of a 95% simulated prediction interval. Note that for this ARMA$(1,1)$ realisation, 6.4% of the test set observations $\{x_{751}, \ldots, x_{1000}\}$ lie outside of the 95% simulated prediction interval.

Here, $N = 100$ simulations from $t = 751$ to $t = 1000$, starting at the last observation from the training set x_{750}, are simulated by the bootstrapping method, and the simulated prediction intervals and point forecasts for horizons $h = 1, 2, \ldots H$, with $H = 250$ are calculated in the same way as in Example 3.4.2.

Figure 3.5 shows the observed ARMA$(1,1)$ trajectory $\{x_0, x_1, \ldots, x_{1000}\}$ from $t = 700$ to $t = 800$, split into the observations from the training set $x_{training}$ (solid black line) and the test set x_{test} (dotted black line). As before, the grey solid line shows the bootstrapping point forecasts, and the dashed black lines give the 95% bootstrapped prediction intervals. For this ARMA realisation and bootstrapping simulation approach, 6.8% of the values of the test set x_{test} lie outside of the 95%

prediction interval.

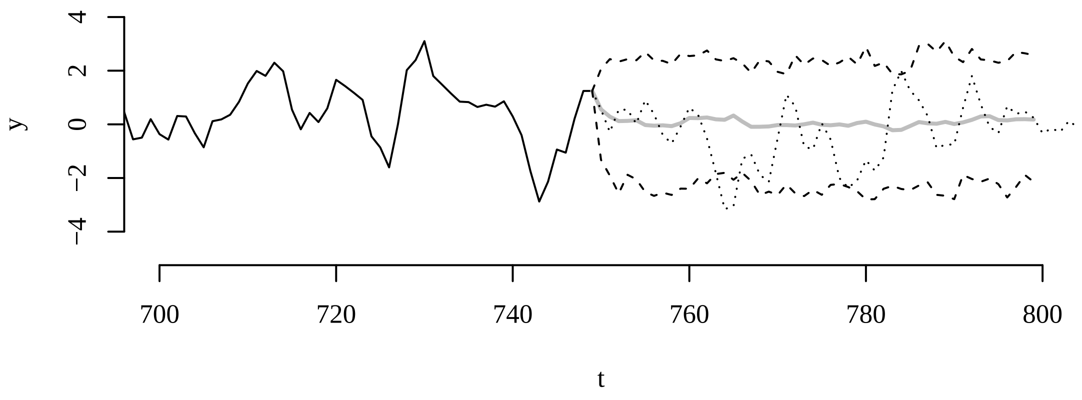


Figure 3.5: This figure shows a realisation of an ARMA$(1, 1)$ process, split into a training set (solid black line) and test set (dotted black line), with the training set used to estimate the residuals. These residuals were sampled with replacement to produce $N = 100$ simulations following the ARMA$(1, 1)$ form with the new "innovations", which were then used to give point forecasts and prediction intervals. The thick grey line gives the point forecasts $\{\hat{x}_n(h) \colon h = 1, 2, \ldots, 50\}$ and the dashed lines show the upper and lower values of a 95% simulated prediction interval. Note that for this ARMA$(1, 1)$ realisation, 6.8% of the test set observations $\{x_{751}, \ldots, x_{1000}\}$ lie outside of the 95% bootstrap prediction interval.

From these examples we can see that using different forecasting approaches gives slightly different results, although the underlying behaviour of the point forecasts and prediction intervals are similar between the three examples. That is, the forecasts reach stationarity quickly such that the point forecasts are close to zero and the prediction intervals are relatively constant. The forecasts obtained using simulated realisations (Example 3.4.2) tend to the forecasts obtained using the distribution properties of the ARMA process (Example 3.4.1) as the number of realisations used increases. The bootstrapping forecasts (Example 3.4.3) have different asymptotic behaviour, as here the residuals from the training sample are used to obtain new simulations.

3.4.2 Ornstein-Uhlenbeck process

In this section, a realisation of an Ornstein-Uhlenbeck process of the form,

$$\mathrm{d}\mathsf{X}_t^{\mathrm{OU}} = -\gamma \mathsf{X}_t^{\mathrm{OU}}\,\mathrm{d}t + \sigma\,\mathrm{d}B_t,$$

with $\gamma = 1$ and $\sigma^2 = 100$, is simulated from time $t = 0$ to $t = 1000$ using Euler-Maruyama approximation. The time step used in the approximation is $\Delta t_{\mathrm{EM}} = 10 \times 10^{-4}$, and then the trajectory is thinned to give an "observed" realisation, $\{x_0, x_{\Delta t}, x_{2\Delta t}, \dots, x_{n\Delta t}\}$, with $\Delta t = 10 \times 10^{-3}$ and $n\Delta t = 1000$.

As before, these observations are split into a training set, denoted $x_{\text{training}} = \{x_0, x_{\Delta t}, \dots, x_{n_t\Delta t}\}$, and a test set, $x_{\text{test}} = \{x_{(n_t+1)\Delta t}, x_{(n_t+2)\Delta t}, \dots, x_{n\Delta t}\}$, where $n_t\Delta t = 750$. Using the training set and maximising the log-likelihood as given in (2.28) (Section 2.3.3.1) gives parameter estimates $\hat{\gamma} = 1.0286$ and $\hat{\sigma}^2 = 101.1043$.

Example 3.4.4 (Prediction intervals from the true distribution of the process)**.** *In this example the prediction intervals are calculated from* (3.12) *using the distributional knowledge of an Ornstein-Uhlenbeck process as given in Section 2.3.2, with parameter estimates found using maximum likelihood estimation. Here, the parameter estimates found using maximum likelihood estimation,* $\hat{\gamma} = 1.0286$ *and* $\hat{\sigma}^2 = 101.1043$*, are used to give the point forecasts and prediction intervals. That is,* (3.12) *becomes*

$$x_n(h)e^{-\hat{\gamma}h} \pm z_{\alpha/2}\sqrt{\frac{\hat{\sigma}^2}{2\hat{\gamma}}(1 - e^{-2\hat{\gamma}h})}.$$

where $x_n(h) = x_{n_t\Delta t}\, e^{-\hat{\gamma}h}$*. The point forecasts and prediction intervals are calculated for horizons* $h = 1, 2, \dots, H$ *where* $H = 250$ *to mirror the length of the test set.*

Figure 3.6 shows a section of the realisation $\{x_0, x_{\Delta t}, \dots, x_{n\Delta t}\}$ *of an Ornstein-Uhlenbeck process from time* $t = 700$ *to* $t = 800$ *with parameters* $\gamma = 1$ *and* $\sigma^2 = 100$*, split into a training set (solid black line) and a test set (dotted black line). The solid grey line shows the point forecasts using the distributional properties given* *in* (2.13)*, and the* 95% *prediction interval is given by the dashed black lines. In this case,* 5.3% *of the realisations from the test set x_{test} are outside the* 95% *prediction interval.*

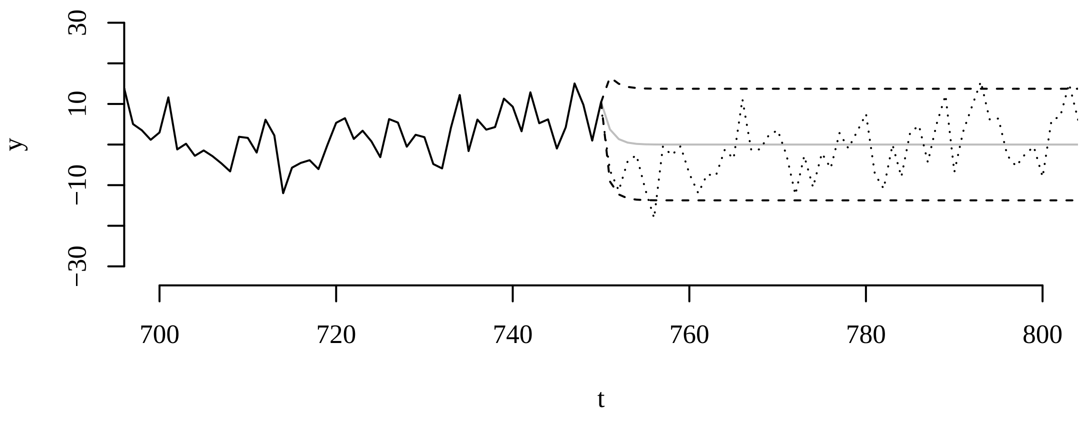


Figure 3.6: This figure shows a section of the realisation $\{x_0, x_{\Delta t}, \dots, x_{n\Delta t}\}$ of an Ornstein-Uhlenbeck process with parameters $\gamma = 1$ and $\sigma^2 = 100$, split into a training set (solid black line) and a test set (dotted black line). The solid grey line is the point forecasts using the distributional properties given in (2.13) and the 95% prediction interval is given by the dashed black lines. In this case, 5.3% of the realisations from the test set x_{test} are outside the 95% prediction interval.

Example 3.4.5 (Prediction intervals from simulated realisations)**.** *In this example, simulated prediction intervals and point forecasts are obtained from $N = 100$ realisations of an Ornstein-Uhlenbeck process with parameters given by those found by maximum likelihood estimation on $x_{training}$, i.e. $\hat{\gamma} = 1.0286$ and $\hat{\sigma}^2 = 101.1043$, and initial value given by the last value of the training set, $x_{n_t\Delta t}$.*

The prediction intervals and point forecasts are calculated in the same way as in Example 3.4.2. From the N simulated trajectories, the point forecast at horizon h is given by the mean of the trajectories at time h, and the $100(1-\alpha)\%$ prediction *interval at horizon h is given by the $\alpha/2$ and $1-\alpha/2$ quantiles of the trajectories at time h.*

Figure 3.7 shows a section of the realisation $\{x_0, x_{\Delta t}, \ldots, x_{n\Delta t}\}$ of an Ornstein-Uhlenbeck process from time $t = 700$ to $t = 800$ with parameters $\gamma = 1$ and $\sigma^2 = 100$, split into a training set (solid black line) and a test set (dotted black line). The solid grey line shows the simulated point forecasts, and the simulated 95% prediction interval is given by the dashed black lines. In this case, 7.1% of the realisations from the test set x_{test} are outside the 95% prediction interval.

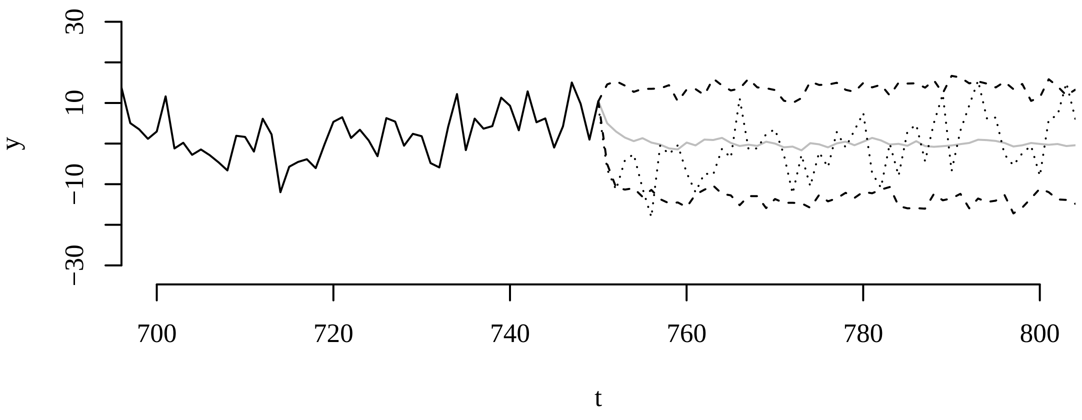


Figure 3.7: This figure shows a section of the realisation $\{x_0, x_{\Delta t}, \ldots, x_{n\Delta t}\}$ of an Ornstein-Uhlenbeck process with parameters $\gamma = 1$ and $\sigma^2 = 100$, split into a training set (solid black line) and a test set (dotted black line). The solid grey line is the simulated point forecast, and the simulated 95% prediction interval is given by the dashed black lines. In this case, 7.1% of the realisations from the test set x_{test} are outside the 95% prediction interval.

As with the forecasts for ARMA processes found in Section 3.4.1, here the two Ornstein-Uhlenbeck examples give forecasts with similar behaviour, such that they both reach stationarity quickly relative to the time-scale of interest. The simulated forecasts found in Example 3.4.5 will tend asymptotically towards the distributional forecasts shown in Example 3.4.4 as the number of simulations used increase.

In this section, different ways of producing point forecasts and prediction intervals have been examined, predominately to apply to observations from ARIMA and Ornstein-Uhlenbeck processes. The methods discussed here go some way to help with the aim stated at the beginning of this section, that is, *to use ARIMA and Ornstein-Uhlenbeck processes to forecast the low dimensional projections of high dimensional simulator output*.

This concludes Part I of the thesis, which has covered theory that can now be applied to the complex systems in Part II. To recap, the theory introduced so far has included:

1. Projecting some simulator output onto a set of basis vectors to reduce the dimension of the data;

2. Modelling these new dimensions (independently) as one-dimensional stochastic processes, assuming that the information lost in the projection step can be replaced with Gaussian noise;

3. Using these stochastic processes to (independently) forecast the projections.

In Part II of this thesis, the forecasts are used to reconstruct the simulator output (by "inverse" principal and independent component analysis, described in Chapter 4). Both the individual component forecasts, as well as the full reconstructions, can be compared to the "true" component and full simulator output values, by splitting the output into training and test sets. This analysis gives a sense of a goodness of fit, and allows comparisons of the forecasts from principal components to those from independent components.

Part II

Application

Introduction to Part II

The thesis so far has focussed on mathematical techniques: dimension reduction, (Chapter 1), stochastic modelling (Chapter 2), and Forecasting (Chapter 3). The overall aim of this project is to use dimension reduction methods in partnership with stochastic process and forecasting theory to model a high-dimensional simulation as a lower-dimensional simulation with some noise, and to use this model for forecasting. Part II of the thesis is a natural conclusion of the previous three chapters, where we now consider an output from a "real-life" simulator that is used to help inform actual policy decisions.

Chapter 4 and Chapter 5 focus on applying the techniques introduced in Chapters 1, 2 and 3 to some simulation output from the Hadley Centre Coupled Model (HadCM3), which is a specific general circulation model (GCM). GCMs are a type of climate simulator that describe the evolution of Earth systems using numerical schemes to approximate a system of partial differential equations such as the Navier-Stokes equations on a rotating sphere. A diagram explaining the stages of both of these chapters is presented in Figure 3.8. Briefly, reconstructions of the full simulator output using a set of forecasted principal or independent components are compared to the true simulator output. The number of components used in the forecasted reconstructions can be increased to reduce some error. The number of principal components needed to reduce the reconstruction error below some threshold can be compared to the number of independent components required, or the number of components can be fixed, and the principal component reconstruction error compared to the independent component error. We can also compare the difference in the reconstructions using the true principal or independent components, and the reconstructions using one-step-ahead forecasts for the respective components. Therefore, in this chapter the ability to forecast using principal components is compared directly to that using independent components, as well as the suitability of the models used to forecasts the components.

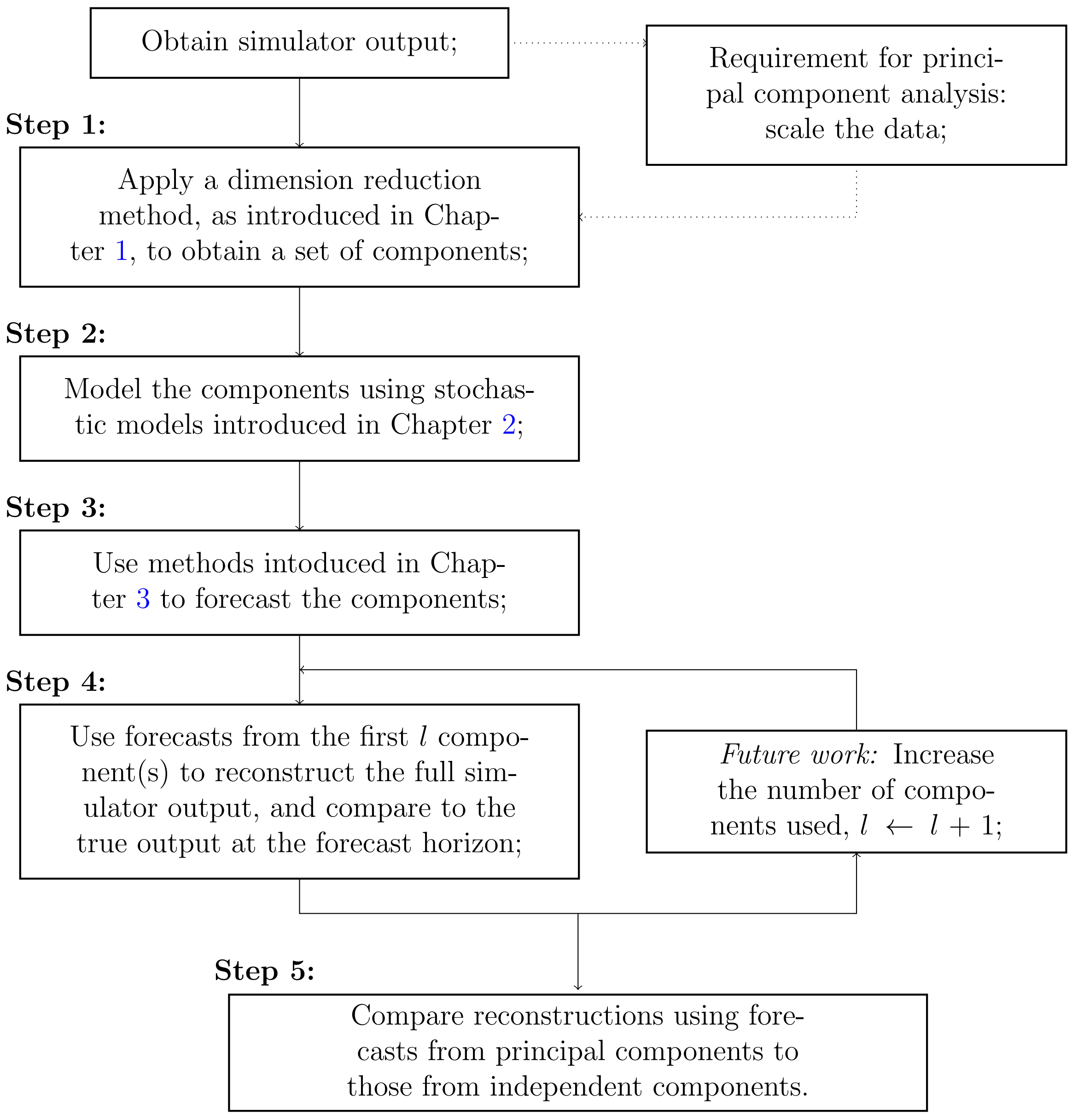

Obtain simulator output;
Requirement for principal component analysis: scale the data;
Step 1:
Apply a dimension reduction method, as introduced in Chapter 1, to obtain a set of components;
Step 2:
Model the components using stochastic models introduced in Chapter 2;
Step 3:
Use methods intoduced in Chapter 3 to forecast the components;
Step 4:
Use forecasts from the first l component(s) to reconstruct the full simulator output, and compare to the true output at the forecast horizon;
Future work: Increase the number of components used, $l \leftarrow l + 1$;
Step 5:
Compare reconstructions using forecasts from principal components to those from independent components.

Figure 3.8: Diagram showing stages invoked in Part II

In Chapter 4, we analyse the HadCM3 simulator when the output is of the mean sea-level air pressure over the Earth's surface. This chapter also contains a brief introduction to the HadCM3 simulator and a discussion on preprocessing the simulator output before applying principal component analysis. In Chapter 5, we analyse the HadCM3 simulator mean sea-level air pressure with two-dimensional horizontal wind speed output.

Chapter 4

Dimension Reduction, Forecasting and Reconstruction: Mean Sea-Level Air Pressure Climate Simulation Output

As discussed in the introduction above, here the HadCM3 simulator output is given by the mean sea-level air pressure. This chapter is structured as follows. We introduce the HadCM3 simulator in Section 4.1, with a brief history and some additional reading for the interested party. In Section 4.2 we describe some preprocessing of the simulator output that is required for principal component analysis (as introduced in Section 1.2) to be applied in Section 4.2.2. We apply independent component analysis (from Section 1.3) to the output in Section 4.3. Then, in Section 4.4 we apply the stochastic theory (introduced in Chapter 2) to the principal components and independent components such that we can model these as stochastic processes and obtain estimated parameters values. In Section 4.5 we use these stochastic process models to obtain point and interval forecasts (introduced in Chapter 3). Finally, in Section 4.6, we obtain reconstructions of the full simulator output, and analyse the quality of the forecasts. Note that the future work highlighted in Step 4. in Figure 3.8 is not considered in this section, due to time constraints.

Remark 4.16: Terminology – *model* or *simulator*

Throughout this section, following Rougier *et al.* (2013) the term "simulator" to describe the computer code that produces some output is preferred to the term "model". This prevents the over-use of the term "model", which is extremely easy in this setting. Also, this allows the terminology "model" to refer to the broader class of "statistical model", which includes the simulator outputs and the judgements about how these observations are related to the Earth system.

4.1 Introduction to the HadCM3 Climate Simulator

In this section we describe a specific climate simulator that is used throughout Part II, known as the HadCM3 simulator. This general circulation model simulates both the Earth's atmosphere and its oceans using two separate simulations that interact in some way. A climate simulator is described as *coupled* if the simulations for the atmosphere and for the ocean interact, which is done predominantly via the sea surface temperature and sea ice extents. The HadCM3 climate simulator as described in Gordon *et al.* (2000) is a coupled simulator, and was the first coupled simulator that did not require artificial adjustments – known as flux adjustments – to prevent unrealistic outputs. Prior to HadCM3, coupled models had to be prevented from experiencing significant climatic drift by flux adjustments. See Barthelet *et al.* (1998) for an example of when not applying flux adjustments in a coupled simulator (which was different to HadCM3) resulted in unrealistic drift. The atmospheric model component of HadCM3 is based on the United Kingdom Meteorological Office's (UKMO) unified forecast/ climate model examined in Cullen (1993) (also see Connolley & Cattle, 1994), and is calculated on a 2.5° × 3.75° grid with nineteen vertical levels (Pope *et al.*, 2000). The ocean model component of HadCM3 is based on the model described in Cox (1984), and is calculated on a 1.25° × 1.25° latitude-longitude grid with twenty vertical levels. The sea-ice model is the same as that found in HadCM2, described in Cattle & Crossley (1995), and uses the same grid size and number of vertical levels as the atmospheric component. The simulator is run on a time step of 30 minutes for the atmosphere model and one hour for the ocean model, with the atmosphere and ocean components coupled once a day. The HadCM3 simulator was a major simulator used in the Independent Panel on Climate Change (IPCC) Third and Fourth Assessments (Pachauri & Reisinger, 2008; Watson *et al.*, 2001). An example of output from the HadCM3 simulator is given in Figure 4.1. In this figure, the structure of the HadCM3 output is shown by a snapshot of the mean sea-level air pressure on a 2.5° × 3.75° grid, and thus at sea-level (ignoring the other vertical layers of the simulator output) the Earth is modelled on 96 longitudinal and 73 latitudinal cells.

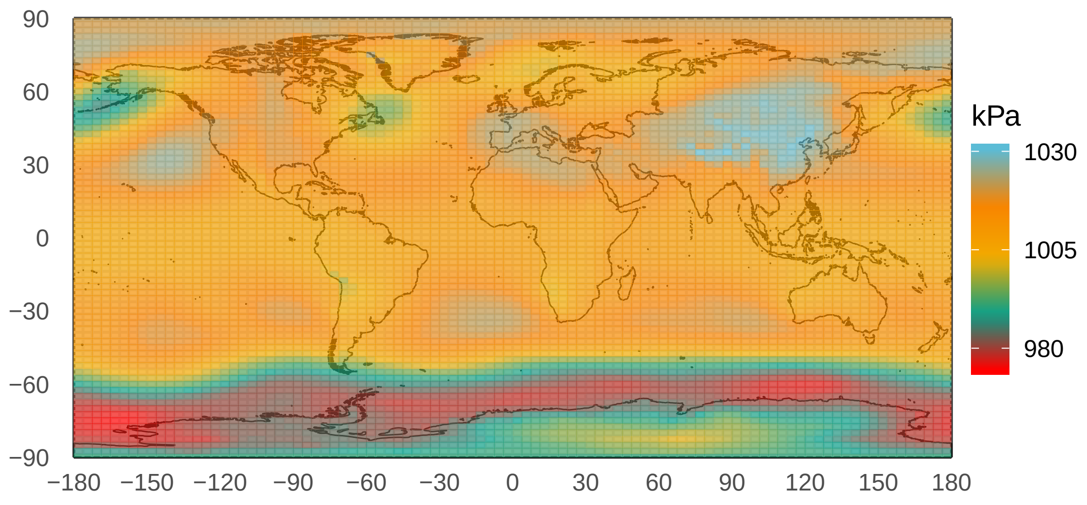


Figure 4.1: An example of a typical simulator output, with an output of mean sea-level air pressure. The simulator gives an output of values on a discretised representation of the Earth's surface, using a grid of 96 longitudinal cells and 73 latitudinal cells.

In this chapter, the output from the HadCM3 simulator is mean sea-level air pressure, and is given on a 96×73 grid that is used to represent the Earth's surface. Therefore, each time-step of the simulator can be given by the mapping

$$f_1 \colon \mathcal{M} \to \mathbb{R}^{96\times 73}, \tag{4.1}$$

for the mean sea-level air pressure output. Here, $96 \times 73 = 7008$ and each dimension of the mapping $f_1(m)$, $m \in \mathcal{M}$, represents the value of the simulator output on one grid cell. Although this mapping is represented in Figure 4.1 by equal sized grid cells, this is actually a miss-characterisation of the output, as these grid cells are on the surface of a sphere and therefore have differing area dependent on their position. We discuss this in detail in Section 4.2.1. Throughout this chapter, the simulator output that we analyse consists of $n = 6000$ time-steps, and can be represented as a matrix $X \in \mathbb{R}^{6000\times 7008}$ such that the i^{th} row of X is given by the mapping $x_i = f(m_i)$ for some input $m_i \in \mathcal{M}$, $i = 1, \ldots, 6000$.

4.2 Applying Principal Component Analysis to the Mean Sea-Level Air Pressure HadCM3 Simulator Output

In this section, principal component analysis as introduced in Section 1.2 is applied to output from the HadCM3 simulator. One important aspect to take into consideration for principal component analysis is the area of the grid cell, as this affects the variance of each output dimension which in turn affects the principal component basis found. Although the grid cells in Figure 4.1 appear to be of equal size, this is purely an artefact of the map projection used. As a general remark, it is worth noting that a sphere's surface cannot be represented on a plane without distortion (Gauss & Pesic, 2005), which in the case here involves distorting the area represented by equal grid sizes of Figure 4.1. As principal component analysis finds directions that maximise variance of the projections, scaling the simulator output by the grid cell size is equivalent to scaling the variance of each column of the simulator output X.

Using information given regarding the area of each cell, we can adjust the HadCM3 simulator output by scaling each element of the mapping $f_1(m)$, as in (4.1), by the area of the associated cell. That is, each element of the HadCM3 simulator output can be adjusted by a second function $g\colon \mathbb{R} \times S^2 \to \mathbb{R}$, in the following way. Assume that the sphere is divided into p_{lat} and p_{lon} latitudinal and longitudinal cells respectively, such that the cells have midpoints at the latitudes φ_i, $i = 1, \dots, p_{\text{lat}}$, and at the longitudes ϑ_j, $j = 1, \dots, p_{\text{lon}}$. For the simulator output $x \in \mathbb{R}$ at the grid cell with centre $s \in S^2$, the mapping that scales the output by the (square of the) respective grid cell area, g, is given by

$$g(x, s) = x \sum_{i=1}^{p_{\text{lat}}} \sum_{j=1}^{p_{\text{lon}}} \sqrt{c_{ij}}\, \mathbb{1}_{s \in \chi_{ij}}. \tag{4.2}$$

Here, $\chi_{ij} \in S^2$ refers to the grid cell with centre at latitude φ_i and longitude ϑ_j, and with area given by c_{ij}. That is, the cell χ_{ij} is bounded by $[\varphi_{i,S}, \varphi_{i,N}] = [\varphi_i - \varphi_{\text{cell}}/2, \varphi_i + \varphi_{\text{cell}}/2]$ in the latitude, and $[\vartheta_{j,W}, \vartheta_{j,E}] = [\vartheta_j - \vartheta_{\text{cell}}/2, \vartheta_j + \vartheta_{\text{cell}}/2]$ in the longitude, where φ_{cell}, and ϑ_{cell} are the latitudinal and longitudinal span of the cell. Note that the values c_{ij}, $i = 1, \dots, p_{\text{lat}}$, $j = 1, \dots, p_{\text{lon}}$, of the area of each cell needs to be calculated.

The method of finding the area c_{ij} in (4.2) that corresponds to each cell χ_{ij}, $i = 1, \dots, p_{\text{lat}}$, $j = 1, \dots, p_{\text{lon}}$, for the output given by the HadCM3 simulator is now discussed in Section 4.2.1.

4.2.1 Calculating the area of the grid cells

In this section we discuss the partition of the Earth that is used by the HadCM3 simulator. As described in Section 4.1, the atmospheric component of the simulator is run on a 2.5° × 3.75° grid, and the oceanic component is run on a 1.25° × 1.25° grid. The mean sea-level air pressure output used in this chapter is on a 2.5° × 3.75° grid. In this description, the non-Euclidean geometry of the simulator is not explicit, and the constant angles that determine the grid cell borders hide the fact that these cells lying on the surface of a sphere have differing areas.

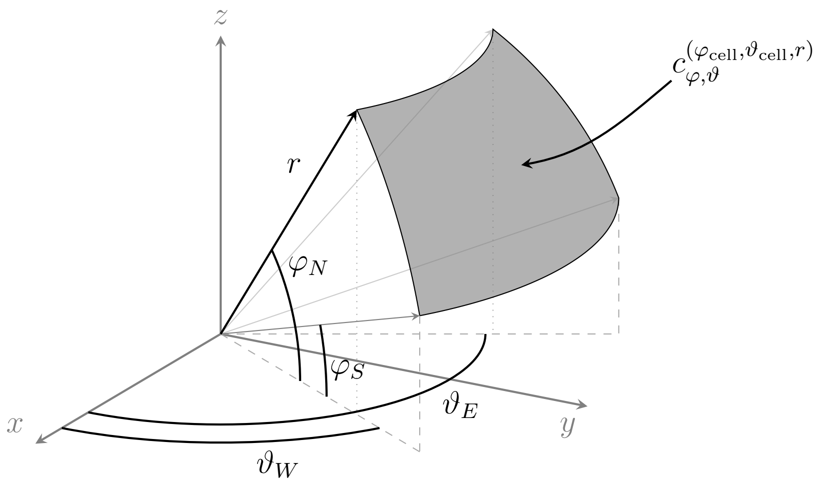

z
$c_{\varphi,\vartheta}^{(\varphi_{\text{cell}},\vartheta_{\text{cell}},r)}$
r
φ_N
φ_S
ϑ_E
x
y
ϑ_W

Figure 4.2: Surface area of a (spherical rectangular) segment of a sphere of radius r in relation to the latitude and longitude of the cell boundary. Here, φ and ϑ give the centre of the cell, and $\varphi_{\text{cell}} := \varphi_N - \varphi_S$, $\vartheta_{\text{cell}} := \vartheta_E - \vartheta_W$. Note that here the angles φ_S and φ_N are taken from the $x - y$ plane, instead of the convention of taking these angles from the z-axis. This is for comparability to the way that latitude is calculated (and shown in Figure 4.1), with latitude 0 at the equator and $\pi/2$ ($-\pi/2$) at the north (south) pole.

From Figure 4.2, each cell is defined by the radius from the origin, r, as well as the latitude and longitude that make up the border of the cell. These are given by φ_S and φ_N for the southern and northern latitudes that define the top and bottom of the cell, and ϑ_E and ϑ_W for the eastern and western longitudes that define the right and left cell boundary. Note that the angles φ_S and φ_N are defined from the $x - y$ plane as opposed to being defined from the z-axis (as is convention in the spherical coordinate system), to match the convention of latitude being 0 at the equator and $\pi/2$ (respectively, $-\pi/2$) at the north (south) pole.

The area of the cell is given by $c_{\varphi,\vartheta}^{(\varphi_{\text{cell}},\vartheta_{\text{cell}},r)}$, with the subscript giving the position of the cell centre and superscript giving the latitude and longitude span of the cell, and the radius of the ball whose surface the cell lies on.

Suppose now that the latitudinal and longitudinal span of the cell is constant, given by φ_{cell} and ϑ_{cell}, and the radius r is fixed, and define the grid cells by their centre point, with latitude φ_i, $i = 1, \ldots, p_{\text{lat}}$ and longitude ϑ_j, $j = 1, \ldots, p_{\text{lon}}$. We will calculate explicitly the area given by $c_{\varphi,\vartheta}^{(\varphi_{\text{cell}},\vartheta_{\text{cell}},r)}$, and thus find the area of the grid cells used in the HadCM3 output.

First note that the Cartesian coordinates (x, y, z) can be written in terms of (φ, ϑ) in the spherical coordinate system – with φ the angle from the $x-y$ plane and ϑ the angle from the x-axis – by,

$$\begin{cases} x = r\cos\varphi\cos\vartheta; \\ y = r\cos\varphi\sin\vartheta; \\ z = r\sin\varphi. \end{cases}$$

Then, the parametric representation $\rho\colon \Theta \times \Theta \to \mathbb{R}^3$ of the Cartesian space in terms of φ and ϑ is given by,

$$\rho(\varphi, \vartheta) = \begin{pmatrix} r\cos\varphi\cos\vartheta \\ r\cos\varphi\sin\vartheta \\ r\sin\varphi \end{pmatrix}.$$

Now the integral over the surface given in cell C with centre at latitude φ and longitude ϑ, latitudinal span given by φ_{cell} and longitudinal span ϑ_{cell}, can be written in this new coordinate system by,

$$\int_C \mathrm{d}S = \int_{\varphi-\varphi_{\text{cell}}/2}^{\varphi+\varphi_{\text{cell}}/2} \int_{\vartheta-\vartheta_{\text{cell}}/2}^{\vartheta+\vartheta_{\text{cell}}/2} \left\| \frac{\partial\rho(\varphi', \vartheta')}{\partial\varphi} \times \frac{\partial\rho(\varphi', \vartheta')}{\partial\vartheta} \right\| \mathrm{d}\vartheta'\,\mathrm{d}\varphi'.$$

Here,

$$\begin{aligned} \left\| \frac{\partial\rho(\varphi, \vartheta)}{\partial\varphi} \times \frac{\partial\rho(\varphi, \vartheta)}{\partial\vartheta} \right\| &= (r^4\cos^4\varphi\cos^2\vartheta + r^4\cos^4\varphi\sin^2\vartheta + r^4\sin^2\varphi\cos^2\varphi)^{1/2} \\ &= r^2\,|\cos\varphi| \\ &= r^2\cos\varphi, \quad \text{for } \varphi \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]. \end{aligned}$$

Therefore, we have

$$\int_C \mathrm{d}S = r^2 \cdot \vartheta_{\text{cell}} \cdot \Big(\sin\big(\varphi + \frac{\varphi_{\text{cell}}}{2}\big) - \sin\big(\varphi - \frac{\varphi_{\text{cell}}}{2}\big)\Big).$$

For fixed r, ϑ_{cell} and φ_{cell}, this integral only depends on the latitudinal position of the cell, φ and is independent of the longitudinal position, ϑ. In this case, $c_{\varphi,\vartheta}^{(\varphi_{\text{cell}},\vartheta_{\text{cell}},r)}$ can be written as a mapping $c\colon (-\pi/2, \pi/2) \to [0, 4\pi r^2]$, with

$$c(\varphi) = r^2\vartheta_{\text{cell}}\Big(\sin\big(\varphi + \frac{\varphi_{\text{cell}}}{2}\big) - \sin\big(\varphi - \frac{\varphi_{\text{cell}}}{2}\big)\Big). \tag{4.3}$$

Using this, for each cell of the HadCM3 output, a weight can be associated to the output based on the latitude of the cell's centre. For the HadCM3 output in this chapter, the value of c_{ij} in (4.2) is given by $c_{ij} = c_i = c(\varphi_{k(i)})$, with $c(\cdot)$ as in (4.3),

$$\varphi_m = -\pi/2 + (2s-1)\varphi_{\text{cell}}/4,$$

$m = 1, \ldots, p_{\text{lat}}$, and $k\colon \{1, \ldots, p\} \to \{1, \ldots, p_{\text{lat}}\}$ the function that gives the latitude subscript associated with the i^{th} element of x. Note that the cell centres for $m = 1$ and $m = p_{\text{lat}}$ are at $\varphi_1 = -\pi/2 + \varphi_{\text{cell}}/4$ and $\varphi_{p_{\text{lat}}} = -\pi/2 + (2p_{\text{lat}} - 1)\varphi_{\text{cell}}/4 = \pi/2 - \varphi_{\text{cell}}/4$, and define a cell with latitudinal span $\varphi_{\text{cell}}/2$. However, for $m = 2, \ldots, p_{\text{lat}} - 1$ the cells have latitudinal span φ_{cell}. This is a particularity of the HadCM3 simulator output, and is discussed in Remark 4.17.

Now we have described the specific mapping $g : \mathbb{R} \times S^2 \to \mathbb{R}$ as in (4.3) for each element of the HadCM3 simulator output $x = f_1(m) \in \mathbb{R}^p$. The output given as the $\mathbb{R}^p$ vector can also be scaled by the matrix $G = \text{diag}(\sqrt{c_i}) \in \mathbb{R}^{p\times p}$, to give scaled output $x^{(g)} = Gx \in \mathbb{R}^p$.

Remark 4.17: Area of the southern-most and northern-most cells in the HadCM3 simulator output

The output of the HadCM3 simulator treats the northern-most and southern-most cells differently to all the others. In this remark, only the northern cells are considered, as the southern cells can be treated in a similar manner. That is, suppose the cell of interest have centre at latitude $\varphi_{p_{\text{lat}}}$ and longitude ϑ_j for some $j = 1, 2, \ldots, p_{\text{lon}}$.

In the HadCM3 simulator the latitude span for these cells is half that of all the other grid cells, such that the latitude-longitude span is $[0.5\varphi_{\text{cell}}, \vartheta_{\text{cell}}]$, where φ_{cell} (respectively, ϑ_{cell}) is the latitude (longitude) span of the general HadCM3 cell output. These cells are (spherical) triangular in shape, as the longitudinal borders $\vartheta_j - \vartheta_{\text{cell}}/2$ and $\vartheta_j + \vartheta_{\text{cell}}/2$ converge to a point at latitude $\pi/2 = \varphi_{p_{\text{lat}}} + \vartheta_{\text{cell}}/4$. Note the divisor of 4 instead of 2 for the latitude, as discussed above.

4.2.2 Applying principal component analysis to the weighted simulator output

In this section we apply principal component analysis to the mean sea-level air pressure output from the HadCM3 simulator, represented by the mapping given by f_1 in (4.1), and shown at a single time-step in Figure 4.1. Let the output be given by the matrix $X \in \mathbb{R}^{n \times p}$, with $n = 6000$ and $p = 7008$. Then, the first step involves scaling the columns of the matrix X with regards to the area of the cell that each element of the matrix is associated with, such that we have $X^{(g)} = XG$ with $G = \text{diag}(\sqrt{c_i})$ as described in Section 4.2.1.

Recall that for some standard output $\{x_1, \dots, x_n\}$, the principal directions $\{w_1, \dots, w_l\}$ are found such they form the orthonormal basis that maximises $V(w_i : \{x_1, \dots, x_n\})$ for each w_i sequentially, where V is given in (1.2) and repeated here,

$$V(w : \{x_1, \dots, x_n\}) := \text{Var}\{\langle x_1, w\rangle, \dots, \langle x_n, w\rangle\}.$$

In the case of the HadCM3 output, as indicated by the dotted arrows at Step 1. in Figure 3.8, the vectors need to be weighted by the matrix G as described in Section 4.2.1. Therefore the principal components are selected to be the basis vectors that maximise,

$$\begin{aligned} V(w : \{x_1^{(g)}, \dots, x_n^{(g)}\}) &= \text{Var}\{\langle x_1^{(g)}, w\rangle, \dots, \langle x_n^{(g)}, w\rangle\} \\ &= \text{Var}\{\langle Gx_1, w\rangle, \dots, \langle Gx_n, w\rangle\}. \end{aligned}$$

In this case, the matrix X in the principal component analysis method (Panel 1.3) is replaced with $X^{(g)} = XG \in \mathbb{R}^{n\times p}$, which is then column-centred to give $\tilde{X}^{(g)}$. Then, the single value decomposition in Step 1. of Panel 1.3 is performed on $\tilde{X}^{(g)}$. That is, finding the eigenvectors and eigenvalues of the covariance matrix

$$C_X = \frac{1}{n-1}(\tilde{X}^{(g)})^\top \tilde{X}^{(g)},$$

is done in practice by decomposing the matrix $\tilde{X}^{(g)}$ into $\tilde{X}^{(g)} = U\Lambda V^\top$ and taking the columns of $V \in \mathbb{R}^{p\times r}$ to be the principal directions, with $\Lambda = \mathrm{diag}(\lambda_i) \in \mathbb{R}^{r\times r}$ (ordered such that $\lambda_1 \geq \cdots \geq \lambda_r$) and the associated variances given by $\gamma_i = \lambda_i^2/(n-1)$. Here, the rank of the $\tilde{X}^{(g)}$ is given by $r = \min\{n-1, p\} = 5999$.

Performing principal component analysis on the output, weighted by cell area, gives principal directions as the columns of $V \in \mathbb{R}^{p\times r}$. Following the notation as set out in Panel 1.2, the principal directions are notated $w_i := v_i \in \mathbb{R}^p$, $i = 1, \ldots, r$, where v_i is the i^{th} column of V. The principal components are given by the columns of the matrix $M = (m_1\ m_2\ \cdots\ m_r) = \tilde{X}^{(g)}V$, with the i^{th} principal component, $m_i \in \mathbb{R}^n$, having variance $\gamma_i = \lambda_i^2/(n-1)$.

The first six principal directions $w_1, \ldots, w_6$ are shown in Figure 4.3, with the elements of each principal direction plotted on the associated grid cell. The first principal direction contains some behaviour that could be related to atmospheric circulation resulting from the Hadley Cell, or the El Niño-Southern Oscillation. Hadley Cell circulation is the movement of air upwards at the equator and then towards the poles, sinking at a latitude of about ±30°. The El Niño-Southern Oscillation is the name given to the irregularly periodical variation in winds and sea surface temperatures over the tropical eastern Pacific ocean. The El Niño is related to Walker circulation, which is caused by the pressure gradient resulting from a high pressure system over the eastern Pacific ocean, and low pressure system over Indonesia. It also increases the size of the Western Hemisphere Warm Pool, a warm sea region that straddles North and South America, and is present in the first principal direction in Figure 4.3.

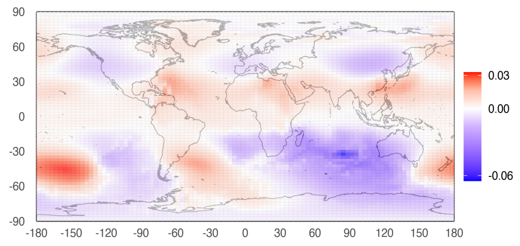
Projecting the weighted and column-centred simulator output $\tilde{X}^{(g)}$ along the principal directions $w_1, w_2, \ldots \in \mathbb{R}^{7008}$ gives the principal components, $m_1, m_2, \ldots \in \mathbb{R}^{6000}$. The first six principal components, corresponding to the data projected along the directions given in Figure 4.3, are shown in Figure 4.4. Throughout this chapter the principal components are assumed to be – and displayed as – causal time-series, for $t = 0, 1, \ldots, n-1$, where $n = 6000$. It is clear that the principal components exhibit strong seasonal behaviour with a periodicity of twelve, similar to the example in Figure 2.1, as discussed in Section 2.2 and in Section 2.3.4. The seasonal behaviour can also be seen by examining the correlogram for each principal component. This is done in Section 4.4.2, with the seasonal behaviour removed before modelling the modified principal components as causal processes using time series models and stochastic differential equations as introduced in Chapter 2.

The plot of the percentage of the variance that is accounted for by the first twenty principal components, given by $100 \cdot \gamma_i / \sum_{j=1}^{r} \gamma_j$, $i = 1, \ldots, 20$, where $r = 5999$ is the rank of $\tilde{X}^{(g)}$, is shown in Figure 4.5. The six principal components, as shown in Figure 4.4, obtained by projecting $\tilde{X}^{(g)}$ along the principal directions shown in Figure 4.3 account for 71.94% of the total variance, and the first twenty principal components account for 86.84% of the total variance. These two results are shown by the grey horizontal lines in Figure 4.5.

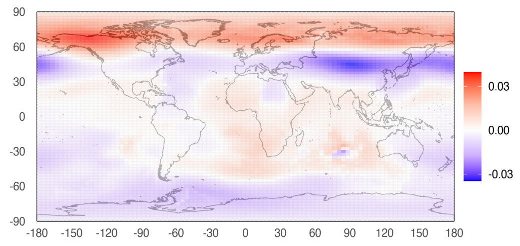
4.3 Applying Independent Component Analysis to the Mean Sea-Level Air Pressure HadCM3 Simulator Output

In this section, we apply independent component analysis (as introduced in Section 1.3) to the mean sea-level air pressure output obtained from the HadCM3 simulator. By Lemma 1.3.3, independent component analysis performed on the matrix $X \in \mathbb{R}^{n \times p}$ is invariant to any scaling of the columns of X. That is, independent component analysis theoretically gives the same results for X as it does for XD for any $D = \text{diag}(d_i) \in \mathbb{R}^{p \times p}$, and therefore, unlike principal component analysis, pre-processing the output by the area of the grid cells is not required.

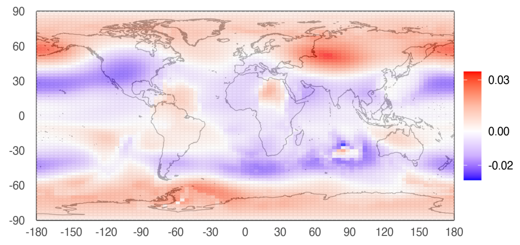
As in Section 4.2.2, here we consider the mean sea-level air pressure output from the HadCM3 simulator, with each time-step represented by the mapping f_1 as in (4.1). Here, we use the *fastICA* independent component analysis method (as



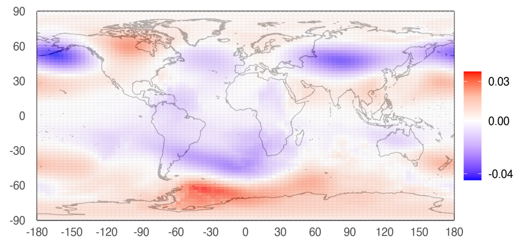
(a) First principal direction, $w_1 \in \mathbb{R}^{7008}$.



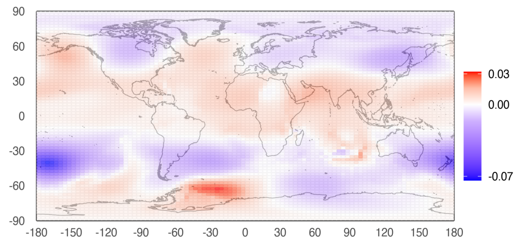
(b) Second principal direction, $w_2 \in \mathbb{R}^{7008}$.



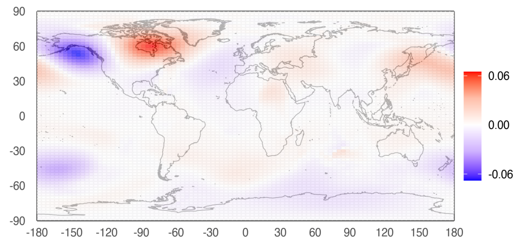
(c) Third principal direction, $w_3 \in \mathbb{R}^{7008}$.



(d) Fourth principal direction, $w_4 \in \mathbb{R}^{7008}$.

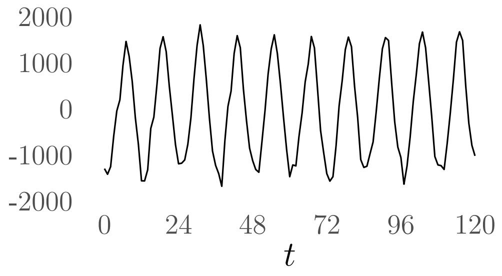


(e) Fifth principal direction, $w_5 \in \mathbb{R}^{7008}$.

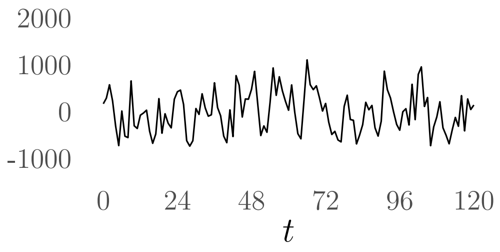


(f) Sixth principal direction, $w_6 \in \mathbb{R}^{7008}$.

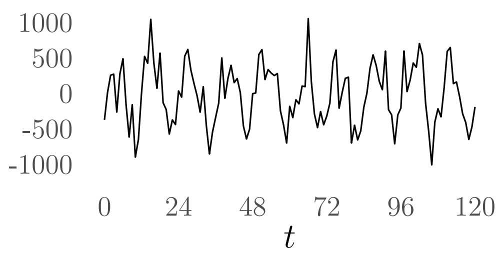
Figure 4.3: First six principal directions found by performing principal component analysis on mean sea-level air pressure output (scaled according to grid cell area) from the HadCM3 simulator. That is, the plots show $w_1, \ldots, w_6 \in \mathbb{R}^{7008}$, mapped onto the corresponding grid cells.



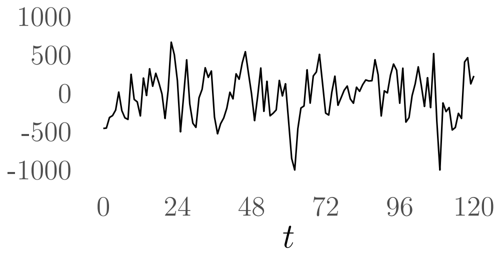
(a) First principal component, m_1, for time steps $t = 0, 1, \ldots, 119$.



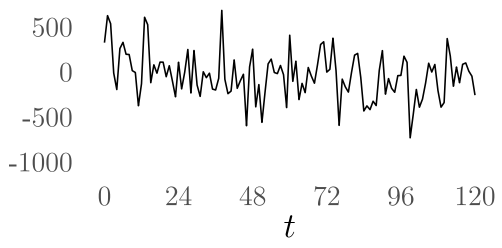
(b) Second principal component, m_2, for time steps $t = 0, 1, \ldots, 119$.



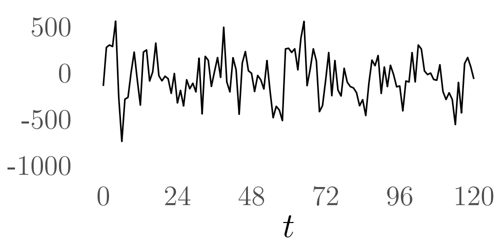
(c) Third principal component, m_3, for time steps $t = 0, 1, \ldots, 119$.



(d) Fourth principal component, m_4, for time steps $t = 0, 1, \ldots, 119$.



(e) Fifth principal component, m_5, for time steps $t = 0, 1, \ldots, 119$.



(f) Sixth principal component, m_6, for time steps $t = 0, 1, \ldots, 119$.

Figure 4.4: First six principal components for time steps $t = 0, 1, \ldots, 119$. That is, the first 120 elements of each vector $m_1, \ldots, m_6 \in \mathbb{R}^{6000}$ are shown. The vectors $m_1, \ldots, m_6$ are found by projecting the scaled (and column-centred) mean sea-level HadCM3 simulator output, $\tilde{X}^{(g)}$, along the first six principal directions, $w_1, \ldots, w_6 \in \mathbb{R}^{7008}$, as shown in Figure 4.3.

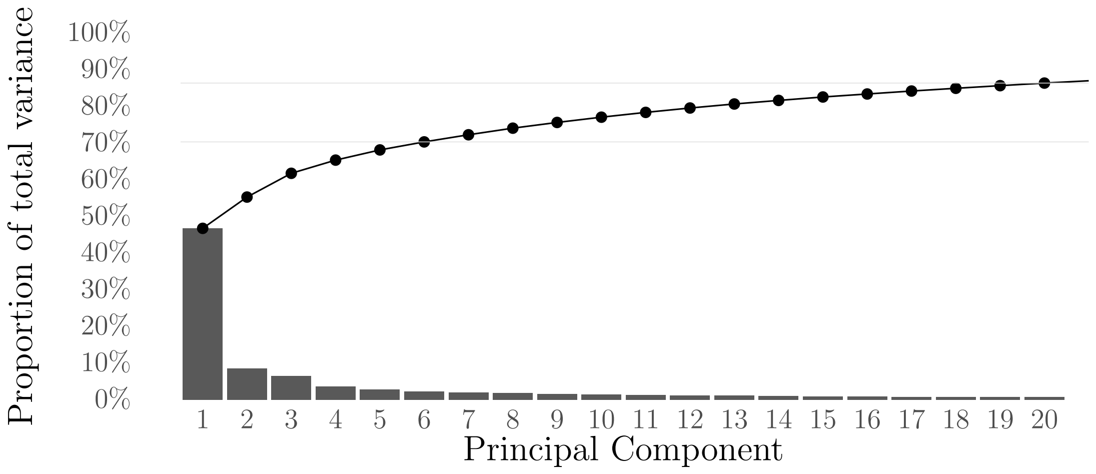


Figure 4.5: Bar chart showing the proportion of total variance accounted for by each principal component found by applying principal component analysis on mean sea-level air pressure output from the HadCM3 simulator. That is, each bar shows the value $100 \cdot \gamma_i / \sum_{j=1}^{r} \gamma_j$, $i = 1, \ldots, 20$, with $r = 5999$ the rank of $\tilde{X}^{(g)}$. The cumulative sum of these proportions is shown by the black points and line, such that each point shows the value $100 \cdot \sum_{k=1}^{i} \gamma_k / \sum_{j=1}^{r} \gamma_j$ for $i = 1, \ldots, 20$. The grey horizontal lines show the cumulative proportion of total variance accounted for by the first six, and the first twenty principal components. These account for 71.94% and 86.84% of the total variance respectively.

discussed in Section 1.4.1). The choice of this method is due to the dimension of the simulator being "large", and follows on from the argument given in Section 1.4.2.5. That is, the fastICA method is efficient when the vast majority of projections of the data are very close to Gaussian, with few sparse non-Gaussian directions. To reduce computational time and the likelihood of numerical errors, the whitening step of *fastICA* is applied using PCA-whitening with $\tilde{r} = 167$ principal components, (that is, $r = 167$ in Step 2. of Panel 1.4). This reduction of the dimension of the whitened matrix is commented on in Remark 1.3. The choice of $\tilde{r} = 167$ principal components is used as these account for more than 99% of the total variance in the model.

Figure 4.6 shows the first six independent directions, $w_1, \ldots, w_6 \in \mathbb{R}^p$, where each w_i, $i = 1, \ldots, \tilde{r}$, is the i^{th} column of the matrix $W = AQ \in \mathbb{R}^{p \times \tilde{r}}$, with $A \in \mathbb{R}^{p \times \tilde{r}}$ the whitening matrix, $Q \in \mathbb{R}^{\tilde{r} \times \tilde{r}}$ the orthogonal direction matrix as described in Panel 1.4, and $p = 7008$. It is interesting to note the prevalence of the Weddell Sea area (east of the Antarctic peninsula) in many of the independent directions. For example, this area is extremely prominent in the second, third and fifth independent directions.

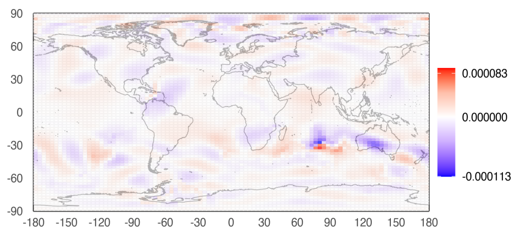
The first six independent components are shown in Figure 4.7. The independent components are given by the columns of $Z \in \mathbb{R}^{n\times\tilde{r}}$, $\tilde{r} = 167$ and $n = 6000$, where $Z = \tilde{X}W$ and $W = AQ \in \mathbb{R}^{p\times\tilde{r}}$ is the matrix with independent directions as the columns (as described above and shown in Figure 4.6). The respective M-spacing entropy estimation – as introduced in Section 1.3.1.3 – has been calculated for the first 6 independent components in Figure 4.7 and displayed in the corresponding captions, with $M = \lfloor\sqrt{6000}\rfloor$.

Now that principal and independent components have been obtained for the mean sea-level air pressure HadCM3 simulator output, the next step is to find parameter estimates assuming that these components are realisations from some suitable stochastic processes. This is discussed in Section 4.4.

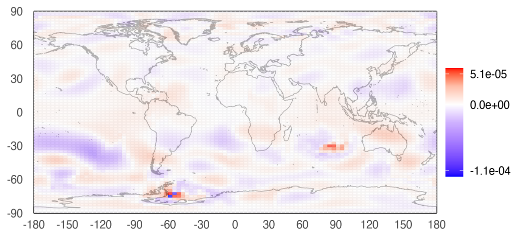
4.4 Modelling Principal and Independent Components by Stochastic Processes

In this section, we model the components obtained by principal component analysis and independent component analysis applied to output from the HadCM3 simulator, using techniques introduced in Chapter 2. This is shown as Step 2. in Figure 3.8.

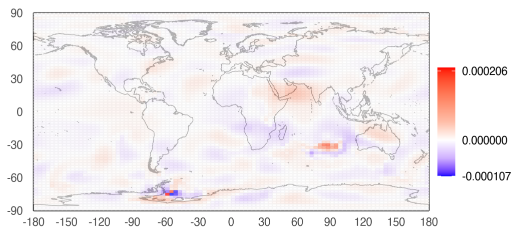
This section is arranged as follows. In Section 4.4.1 we introduce the procedure used later in this section, and some notation. In Section 4.4.2 we consider the principal components obtained from the mean sea-level air pressure HadCM3 simulator output. In Section 4.4.3 we consider the independent components obtained from applying fastICA to the mean sea-level air pressure HadCM3 simulator output.



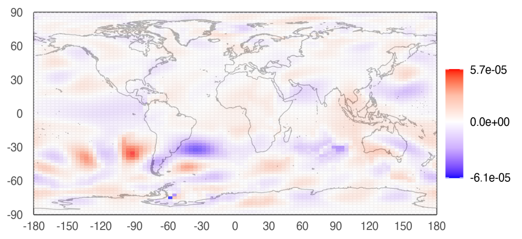
(a) First independent direction, $w_1 \in \mathbb{R}^{7008}$. The M-spacing entropy of the HadCM3 mean sea-level air pressure projection onto this direction is 1.115.



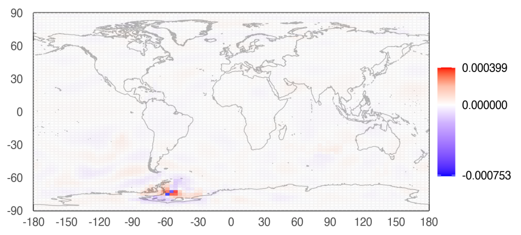
(b) Second independent direction, $w_2 \in \mathbb{R}^{7008}$. The M-spacing entropy of the HadCM3 mean sea-level air pressure projection onto this direction is 1.127.



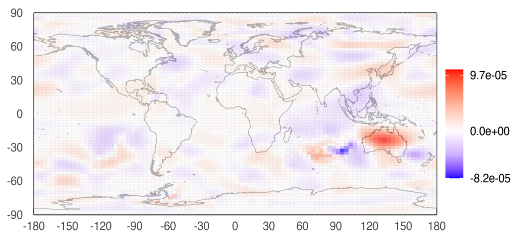
(c) Third independent direction, $w_3 \in \mathbb{R}^{7008}$. The M-spacing entropy of the HadCM3 mean sea-level air pressure projection onto this direction is 1.146.



(d) Fourth independent direction, $w_4 \in \mathbb{R}^{7008}$. The M-spacing entropy of the HadCM3 mean sea-level air pressure projection onto this direction is 1.164.

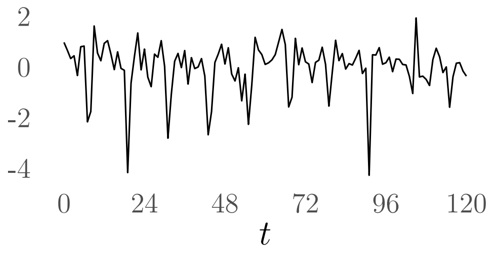


(e) Fifth independent direction, $w_5 \in \mathbb{R}^{7008}$. The M-spacing entropy of the HadCM3 mean sea-level air pressure projection onto this direction is 1.173

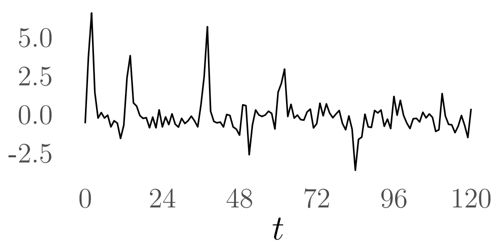


(f) Sixth independent direction, $w_6 \in \mathbb{R}^{7008}$. The M-spacing entropy of the HadCM3 mean sea-level air pressure projection onto this direction is 1.181

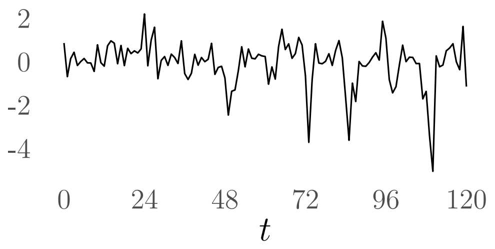
Figure 4.6: First six independent directions found by performing the fastICA method on mean sea-level air pressure output from the HadCM3 simulator. That is, the plots show $w_1, \ldots, w_6 \in \mathbb{R}^{7008}$. The M-spacing entropy associated with each independent direction is calculated as in Section 1.3.1.3, with $M = \lfloor\sqrt{n}\rfloor = \lfloor\sqrt{6000}\rfloor$.



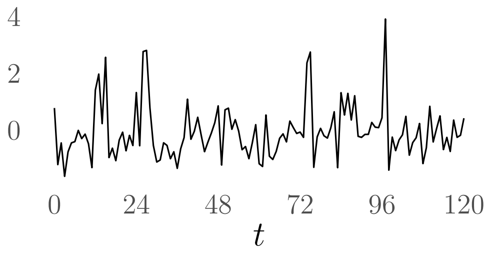
(a) First independent component, s_1, shown for time steps $t = 0, 1, \ldots, 119$. The M-spacing entropy of this independent component is 1.115.



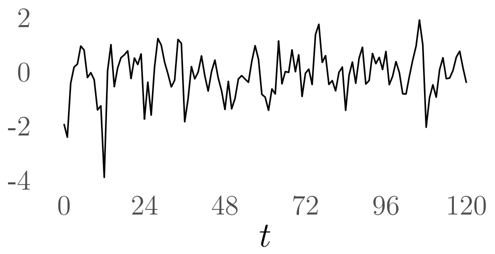
(b) Second independent component, s_2, shown for time steps $t = 0, 1, \ldots, 119$. The M-spacing entropy of this independent component is 1.127.



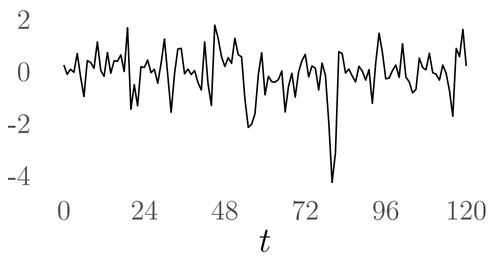
(c) Third independent component, s_3, shown for time steps $t = 0, 1, \ldots, 119$. The M-spacing entropy of this independent component is 1.146.



(d) Fourth independent component, s_4, shown for time steps $t = 0, 1, \ldots, 119$. The M-spacing entropy of this independent component is 1.164.



(e) Fifth independent component, s_5, shown for time steps $t = 0, 1, \ldots, 119$. The M-spacing entropy of this independent component is 1.173.



(f) Sixth independent component, s_6, shown for time steps $t = 0, 1, \ldots, 119$. The M-spacing entropy of this independent component is 1.181.

Figure 4.7: First six independent components, $s_1, \ldots, s_6 \in \mathbb{R}^{6000}$, found by projecting the (column-centred) mean sea-level air pressure HadCM3 simulator output, $\tilde{X}$, along the first six independent directions, $w_1, \ldots, w_6 \in \mathbb{R}^{7008}$, as shown in Figure 4.6. Here, the independent components are shown as a time-series for times $t = 0, \ldots, 119$. The M-spacing entropy is calculated for each independent component as in Section 1.3.1.3, with $M = \lfloor \sqrt{n} \rfloor = \lfloor \sqrt{6000} \rfloor$.

4.4.1 Introduction to modelling the principal and independent components

Each principal and independent component considered throughout this chapter belongs to $\mathbb{R}^n$, with $n = 6000$, and each is modelled as a causal stochastic process with time steps $t = 0, \Delta t, 2\Delta t, \dots, (n-1)\Delta t$ with $\Delta t = 1$ constant. Here, the information lost from only considering a low dimensional set of components – as opposed to the basis – is represented by way of stochasticity. This stochastic approximation of the components is used to obtain a stochastic evolution equation of the deterministic function $P_V f$ as described in Step 5. of Panel 1.1.

In the later forecasting stage, Section 4.5, the aim is to produce one-step-ahead forecasts of a selection of principal and independent components, and compare the one-step-ahead reconstructions from these forecasts to the "true" result obtained from the simulator output. To allow comparisons to the "truth", the simulator is split into "training" and "test" sets, such that the forecasts using the training set can be compared to the output in the test set.

Suppose a realisation from a one-dimensional causal process is given in vector form by $x = (x_0, x_1, \dots, x_{n-1}) \in \mathbb{R}^n$ for some $n \in \mathbb{N}$. Define the *training sample* of size $j < n$ to be $x^{(j)} := (x_0, x_1, \dots, x_{j-1}) \in \mathbb{R}^j$. The *training set* is given to be the set of all training samples for $j = \tilde{n}, \tilde{n}+1, \dots, n-1$, notated $\mathcal{X}_{\text{training}} = \{x^{(\tilde{n})}, x^{(\tilde{n}+1)}, \dots, x^{(n-1)}\}$, for some $\tilde{n} < n$. The *test sample* of the realisation is given by the remaining elements that are not included in the smallest training sample of the training set, *i.e.* $(x_{\tilde{n}}, x_{\tilde{n}+1}, \dots, x_n) \in \mathbb{R}^{n-\tilde{n}}$.

In this chapter – in an attempt to reduce computational time used – we only implement principal component analysis and independent component analysis on the full set of simulator outputs from time $t = 0$ to $(n-1)\Delta t$. Then, the resulting principal and independent components are each split into training samples of size $j = \tilde{n}, \tilde{n}+1, \dots, (n-1)$, for some $\tilde{n} < n$, with these training samples used to obtain suitable models for the components. The estimated models are checked using the respective test sample.

In Sections 4.4.2 and 4.4.3 below, we apply the following procedure. We assume that the time series of interest is given by $x = (x_0, x_1, \dots, x_{n-1}) \in \mathbb{R}^n$, and that it can be modelled by a stochastic process with some known parametric form and unknown "true" parameter ϑ_0. Then, with training set $\mathcal{X}_{\text{training}} = \{x^{(\tilde{n})}, x^{(\tilde{n}+1)}, \ldots, x^{(n-1)}\}$, the procedure applied in this section is given by the following steps, for $j = \tilde{n}, \tilde{n}+1, \ldots, n-1$:

1. Obtain the training sample $x^{(j)} = (x_0, x_1, \ldots, x_{j-1}) \in \mathbb{R}^j$ from the training set $\mathcal{X}_{\text{training}}$;

2. Use parameter fitting techniques as discussed in Chapter 2 to obtain an estimate of ϑ_0 using $x^{(j)}$, giving estimated parameter value $\hat{\vartheta}(j)$;

3. Compare the expected behaviour of the stochastic process with parameter $\hat{\vartheta}(j)$ to the "true" behaviour of the test sample from time j onwards $(x_j, x_{j+1}, \ldots, x_{n-1})$.

From this, parameter estimates of the stochastic process are found for each training sample from the training set $\mathcal{X}_{\text{training}}$, and this can be used in the one-step-ahead forecasting stage in Section 4.5. At each subsequent step, the new estimated parameter value found from the progressively larger training sample are used to give the one-step-ahead forecast.

In Sections 4.4.2 and 4.4.3 below, three different parametric families are considered as models for the training sample realisations $x^{(j)}$, $j = \tilde{n}, \tilde{n}+1, \ldots, n-1$, and are given by; the ARIMA process (introduced in Section 2.2.2.1), Ornstein-Uhlenbeck process (Section 2.3.2) and the block-average Ornstein-Uhlenbeck process (Section 2.3.2.1). If $x^{(j)}$ is assumed to belong to the ARIMA family, then the Box-Jenkins method is used to determine the initial differencing, the order of the autoregressive operator, and the order of moving-average operator, as described in Section 3.1.1. When $x^{(j)}$ is assumed to be an Ornstein-Uhlenbeck process, the parameter values γ and σ^2 as in (2.12) are estimated either by maximising the sparse log-likelihood (2.28) that arises from the sum of the transition densities of a Markov chain as in (2.25), or by using the method of moments where the parameter estimates are given by (2.31). For the block-average Ornstein-Uhlenbeck family, the γ and σ^2 parameter values are estimated from $x^{(j)}$ by maximising the log-likelihood from a multivariate Gaussian distribution as given in (2.32).

Note that it is known that the simulator output in this chapter is a collection of monthly means, and therefore the first two choices for the parametric family of the time series given by the principal and independent components (with seasonal effects removed, if required) are misspecifications. That is, the estimated ARIMA and Ornstein-Uhlenbeck processes are modelled on the assumption of pointwise observations, which is not the case here. An example of this type of misspecification has been shown in Example 2.3.10.

Before applying the techniques described in Chapter 2, any trend and seasonal effects must be removed from the data. The HadCM3 simulator output is monthly, such that the time-steps of the observations are constant and equal to $\Delta t = 1$. With this knowledge (and by looking at the correlogram of the components, which are shown below), the seasonal effects can be removed by subtracting the associated monthly mean from each observation, as discussed in Section 2.3.4. With the means of each month given by μ_j, $j = 1, \ldots, 12$, then the observation at time $t \in \mathbb{N}$ is assumed to be some random departure away from the mean value $\mu_{k(t)}$, where $k \colon \mathbb{Z} \to \{1, 2, \ldots, 12\}$ is the mapping such that

$$k(t) = \begin{cases} t \bmod(12) & \text{if } t \bmod(12) \in \{1, \ldots, 11\}, \\ 12 & \text{if } t \bmod(12) = 0. \end{cases} \tag{4.4}$$

4.4.2 Modelling the principal components

By examining the principal components given in Figure 4.4, it is clear that some of the principal components exhibit strong seasonal behaviour. This can also be seen by considering the correlogram, which is a plot of the sample autocorrelation (2.4) against the lag. Figure 4.8 gives the correlogram for the first six principal components, showing that seasonality is present in all these principal components and is especially strong in the first four. Due to this, the seasonal mean is removed from all the principal components before modelling. Figure 4.9 shows the first three principal components on the left (taken from Figure 4.4), with the grey line giving the quadratic spline function using the monthly sample means, as discussed in Section 2.3.4.2. On the right of Figure 4.9 are the first three principal components with the monthly means removed, giving a process with no seasonality. Here, the principal components are notated $m_1, m_2, \ldots$, and the principal components with monthly means removed – called *seasonally adjusted principal components* throughout this chapter – are notated $\tilde{m}_1, \tilde{m}_2, \ldots$.

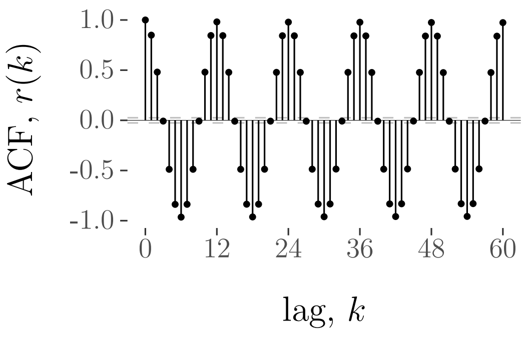
In this section we find parameter estimates for the stochastic models using the training sets from the first $l = 6$ seasonally adjusted principal components $\tilde{m}_1, \ldots, \tilde{m}_6$, where $\tilde{m}_i = (\tilde{m}_{i\,0}, \tilde{m}_{i\,1}, \ldots, \tilde{m}_{i\,n-1}) \in \mathbb{R}^n$, $i = 1, \ldots, 6$, and $n = 6000$ the number of time steps. The training set for the seasonally adjusted principal component $\tilde{m}_i$ is given by $\{\tilde{m}_i^{(4000)}, \tilde{m}_i^{(4001)}, \ldots, \tilde{m}_i^{(5999)}\}$, with each training sample given in vector form by $\tilde{m}_i^{(j)} = (\tilde{m}_{i\,0}, \tilde{m}_{i\,1}, \ldots, \tilde{m}_{i\,(j-1)}) \in \mathbb{R}^j$, $j = 4000, \ldots, 5999$.

We now model the seasonally adjusted principal components as realisations from ARIMA processes, (pointwise) Ornstein-Uhlenbeck processes and block-average Ornstein-Uhlenbeck processes in Sections 4.4.2.1, 4.4.2.2 and 4.4.2.3 respectively.

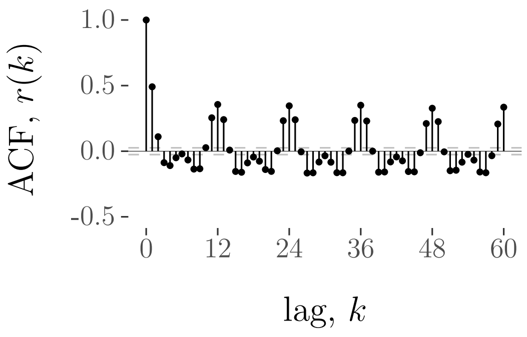
4.4.2.1 Modelling the principal components as observations from an ARIMA processes

In the first instance, we assume the seasonally adjusted principal components are realisations from ARIMA processes (introduced in Section 2.2.2.1) with the estimated model structure chosen using the Box-Jenkins method (as described in Section 3.1.1). That is, each seasonally adjusted principal component is assumed to be a realisation from an (independent) ARIMA(p, d, q) process, and the Box-Jenkins method is used to find estimates for the values of p, d and q. This gives the estimated order of differencing, $\hat{d}$, estimated order of the autoregressive operator, $\varphi(B)$, and the estimated order of the moving-average operator $\vartheta(B)$. Following this, estimates for the parameters of $\varphi(B)$, $\varphi_1, \ldots, \varphi_p$ and for the parameters of $\vartheta(B)$, $\vartheta_1, \ldots, \vartheta_q$, are found.

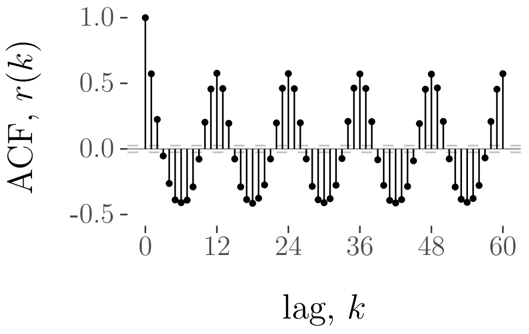
This procedure will be applied to the first six seasonally adjusted principal components, and split into two parts. For simplicity, the order of the ARIMA process is found for each seasonally adjusted principal component using the full component. Then, the parameter estimates are found for every training sample of the associated seasonally adjusted principal component. That is, for each seasonally adjusted principal component, $\tilde{m}_i$, $i = 1, \ldots, 6$. We proceed as follows:



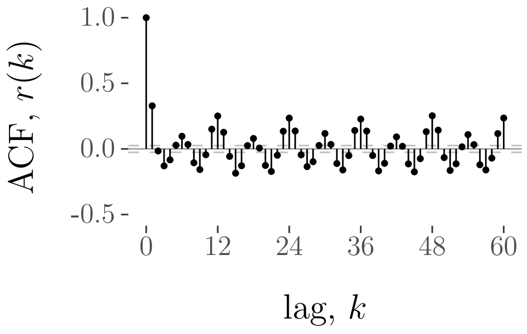
(a) Autocorrelation function (correlogram) for the first principal component, m_1, for lags $k = 0, 1, \ldots, 60$.



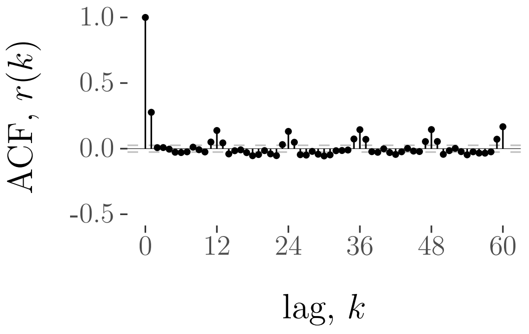
(b) Autocorrelation function (correlogram) for the second principal component, m_2, for lags $k = 0, 1, \ldots, 60$.



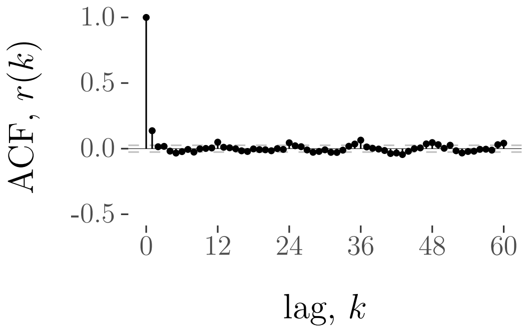
(c) Autocorrelation function (correlogram) for the third principal component, m_3, for lags $k = 0, 1, \ldots, 60$.



(d) Autocorrelation function (correlogram) for the fourth principal component, m_4, for lags $k = 0, 1, \ldots, 60$.

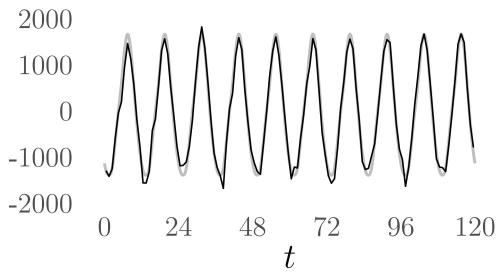


(e) Autocorrelation function (correlogram) for the fifth principal component, m_5, for lags $k = 0, 1, \ldots, 60$.

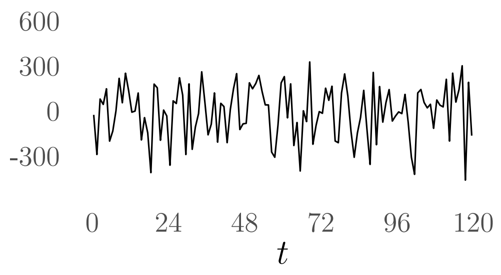


(f) Autocorrelation function (correlogram) for the sixth principal component, m_6, for lags $k = 0, 1, \ldots, 60$.

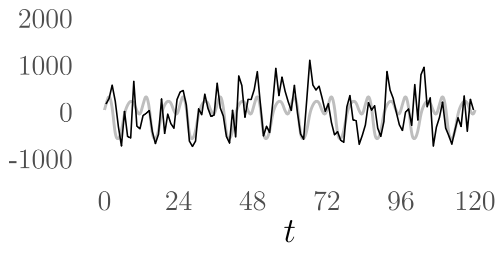
Figure 4.8: Autocorrelation functions (correlogram) for the first six principal component, $m_1, \ldots, m_6$, obtained from the mean sea-level air pressure HadCM3 output, for lags $k = 0, 1, \ldots, 60$.



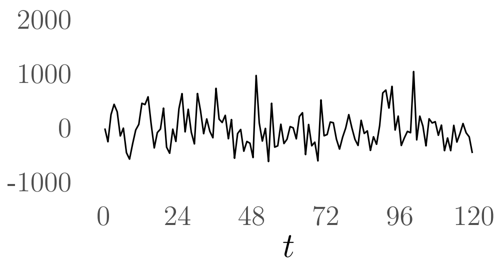
(a) First 120 points of the first principal component, $m_1 \in \mathbb{R}^{6000}$, with the grey line giving the associated quadratic spline calculated from the monthly means.



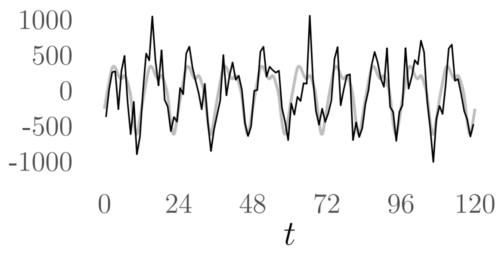
(b) First 120 points of the first principal component with monthly means removed, notated $\tilde{m}_1 \in \mathbb{R}^{6000}$.



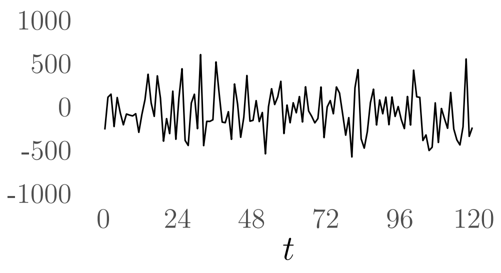
(c) First 120 points of the second principal component, $m_2 \in \mathbb{R}^{6000}$, with the grey line giving the associated quadratic spline calculated from the monthly means.



(d) First 120 points of the second principal component with monthly means removed, notated $\tilde{m}_2 \in \mathbb{R}^{6000}$.



(e) First 120 points of the third principal component, $m_3 \in \mathbb{R}^{6000}$, with the grey line giving the associated quadratic spline calculated from the monthly means.



(f) First 120 points of the third principal component with monthly means removed, notated $\tilde{m}_3 \in \mathbb{R}^{6000}$.

Figure 4.9: First six principal components, $m_1, \ldots, m_6 \in \mathbb{R}^{6000}$, found by projecting the weighted (and column-centred) mean sea-level HadCM3 simulator output, $\tilde{X}^{(g)}$, along the first six principal directions, $w_1, \ldots, w_6 \in \mathbb{R}^{7008}$, as shown in Figure 4.3. Here, only the time steps $t = 0, \ldots, 119$ of the principal components are shown (out of a total of $n = 6000$ time steps).

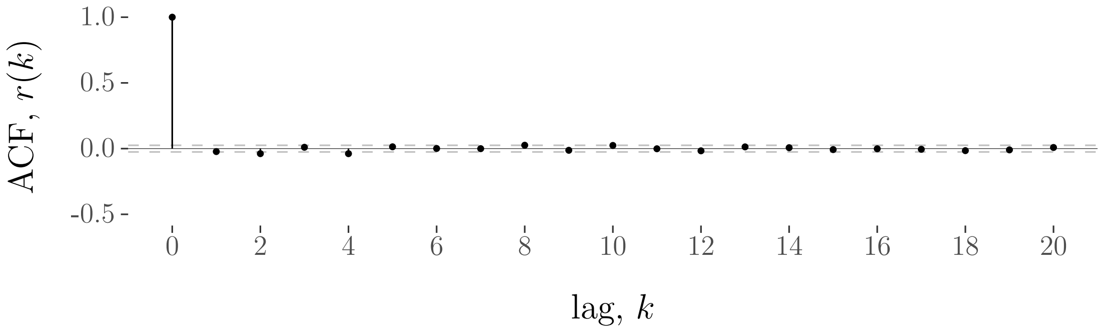
1. We estimate the orders of the ARIMA(p, d, q) process, $\hat{p}_i$, $\hat{d}_i$, and $\hat{q}_i$, by applying the Box-Jenkins method to the whole seasonally adjusted principal component $\tilde{m}_i$;

2. Using each of the training samples $\tilde{m}_i^{(j)} = (\tilde{m}_{i\,0}, \tilde{m}_{i\,1}, \ldots, \tilde{m}_{i\,(j-1)}) \in \mathbb{R}^j$, $j = 4000, 4001, \ldots, 5999$, we take the $\hat{d}_i$-differences, and then we obtain the respective parameter estimates $\varphi_1^{(j)}, \ldots, \varphi_{\hat{p}_i}^{(j)}$ and $\vartheta_1^{(j)}, \ldots, \vartheta_{\hat{q}_i}^{(j)}$.

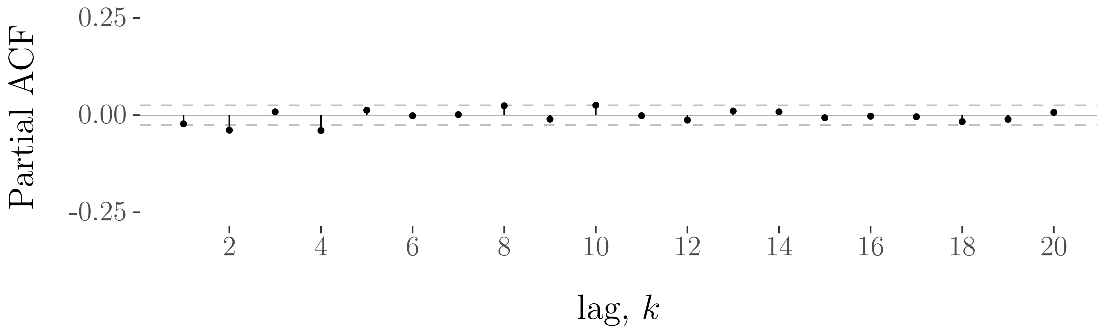
This allows one-step-ahead forecasting to be applied in the later forecasting section (Section 4.5) to obtain point forecasts and prediction intervals to compare against the test samples $\tilde{m}_{i\,4000}, \tilde{m}_{i\,4001}, \ldots, \tilde{m}_{i\,6000}$, $i = 1, \ldots, 6$. Below, detailed steps of the Box-Jenkins method are shown for the first seasonally adjusted principal component, mirroring the steps used in Example 3.1.1. We also briefly discuss modelling the remaining 5 seasonally adjusted principal components.

Box-Jenkins method on the first seasonally adjusted principal component. Here, the sample autocorrelation and partial autocorrelation functions of the seasonally adjusted principal component, $\tilde{m}_1$, are used to estimate the orders of the ARIMA process, p, d and q. Figure 4.10 shows the sample autocorrelation and partial autocorrelation functions for $\tilde{m}_1$, up to lag 20. From the rapid decay of the sample autocorrelation function we estimate that the chain is already in stationarity, such that $d = 0$. That is, we assume that the seasonally adjusted principal component $\tilde{m}_1$ is a realisation from an ARMA(p, q) process. As only the autocorrelation at zero lag is significantly away from zero, the sensible process to choose to model the first seasonally adjusted principal component is the white noise process (as given in Definition 2.2.1). The realisation $\tilde{m}_1$ is assumed to be from the process $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ with $\mathsf{X}_t = \varepsilon_t$ and $\varepsilon_t \sim \mathcal{N}(0, \sigma_\varepsilon^2)$, where only the value σ_ε^2 is needed to be estimated. The autocorrelation of the white-noise process satisfies $\gamma(k) = \sigma_\varepsilon^2 \mathbb{1}_{k=0}$, and thus the estimate of the variance can be given by the zero lag sample autocovariance $\hat{\sigma}_\varepsilon^2 = c(0)$ (with the function c given in (2.3)). It must be remarked here that it is known that this is a misspecification of the HadCM3 simulator output due to assuming pointwise observations instead of monthly averages.

Very similar sample autocorrelation and partial autocorrelation results are found for all the first six seasonally adjusted principal components. Therefore, estimates for the variance of the white-noise process are found for the training sets $\{\tilde{m}_i^{(\tilde{n})}, \tilde{m}_i^{(\tilde{n}+1)}, \dots, \tilde{m}_i^{(n-1)}\}$, $i = 1, \dots, 6$ and with $\tilde{n} = 4000$. For each seasonally adjusted principal component and training sample $m_i^{(j)}$, an estimate $\hat{\sigma}^2_{\varepsilon,i}(j)$ is obtained corresponding to the zero lag sample autocovariance of $m_i^{(j)}$. Figure 4.11 shows the estimates of the standard deviation obtained from training samples $\tilde{m}_i^{(4000)}, \tilde{m}_i^{(4001)}, \dots, \tilde{m}_i^{(5999)}$ of the first four seasonally adjusted principal components, given by $\sigma_{\varepsilon,i}(4000), \sigma_{\varepsilon,i}(4001), \dots, \sigma_{\varepsilon,i}(5999)$, $i = 1, 2, 3, 4$, and shown by the solid, dotted, dot-dashed and dashed lines respectively.



(a) Sample autocorrelation function (correlogram) for the first seasonally adjusted principal component, $\tilde{m}_1$.



(b) Sample partial autocorrelation function for the first seasonally adjusted principal component, $\tilde{m}_1$.

Figure 4.10: Sample autocorrelation (correlogram) and partial autocorrelation functions for the first seasonally adjusted principal component, $\tilde{m}_1$.

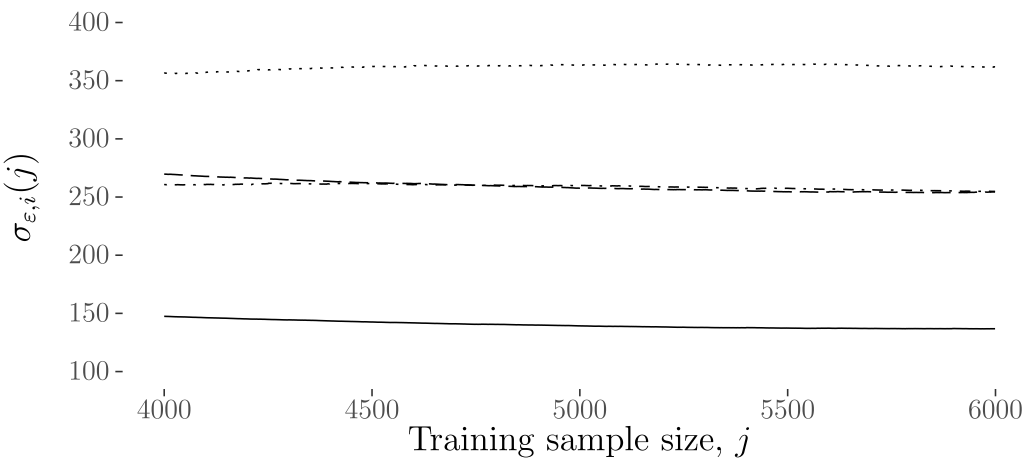


Figure 4.11: Estimation of the standard deviation of the white-noise process, $\hat{\sigma}_{\varepsilon,i}(j)$ for seasonally adjusted principal components $i = 1, 2, 3, 4$ and for training samples $\tilde{m}_i^{(j)}$ of size $j = 4000, 4001, \ldots, 5999$. The standard deviation estimations using training samples from the first, second, third and fourth seasonally adjusted principal components are given by the solid, dotted, dot-dashed and dotted lines respectively.

4.4.2.2 Modelling the principal components as sparse observations from an Ornstein-Uhlenbeck processes

In this section, we assume that the seasonally adjusted principal components, $\tilde{m}_1, \ldots, \tilde{m}_6$, are realisations from some independent Ornstein-Uhlenbeck processes. Here, we obtain estimates for the Ornstein-Uhlenbeck drift and diffusion parameters, γ and σ^2 respectively, by two different methods, both of which have been discussed in Section 2.3.3.1. First, we use the sparse maximum likelihood estimation method, where estimates are found to maximise the log-likelihood (2.28). Then, we examine the method of moments, where the sample lag-1 correlation and variance are used in (2.31) to obtain parameter estimations.

Using sparse maximum likelihood estimation

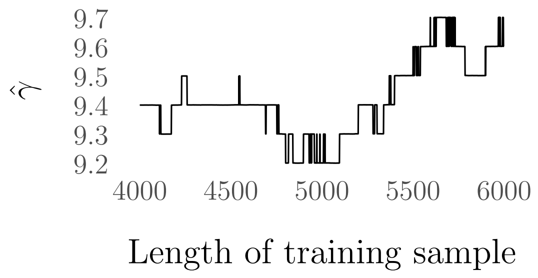
Here we consider the maximum likelihood technique for sparse observations. That is, we assume that each principal component is some Markov chain with respect to time, and thus the log-likelihood function becomes the sum of transition densities of adjacent time steps, as in (2.25).

For each seasonally adjusted principal component, maximum likelihood estimates are obtained from the training samples of size $\tilde{n}, \tilde{n}+1, \ldots, n-1$ and are given by $\hat{\gamma}_{\mathrm{sML},\,i}(j)$ and $\hat{\sigma}^2_{\mathrm{sML},\,i}(j)$, $i = 1, \ldots, l$ and $j = \tilde{n}, \ldots, n-1$, $\tilde{n} = 4000$. The maximum likelihood estimates $\hat{\gamma}_{\mathrm{sML},\,1}(j)$ and $\hat{\sigma}_{\mathrm{sML},\,1}(j)$ for the first seasonally adjusted principal component are shown in Figure 4.12, found using the training set $\{\tilde{m}_1^{(4000)}, \tilde{m}_1^{(4001)}, \ldots, \tilde{m}_1^{(5999)}\}$, where $\tilde{m}_1^{(j)} = (\tilde{m}_{11}, \tilde{m}_{12}, \ldots, \tilde{m}_{1j})$. It is clear from Figure 4.12 that the sparse maximum likelihood estimation technique is producing some spurious results, jumping between different combinations of $\big(\hat{\gamma}_{\mathrm{sML}}, \sigma(\hat{\gamma}_{\mathrm{sML}})\big)$ with the addition of a singular observation to the training sample. This suggests that the log-likelihood may have multiple peaks, or several plateaus where the log-likelihood has near zero gradient. Indeed, Figure 4.13, which gives the log-likelihood for $\big(\gamma, \sigma^2(\gamma)\big)$, with $\gamma = 1, 2, \ldots, 1000$ and $\sigma^2(\cdot)$ as in (2.29), shows such a plateau. This is similar to the log-likelihood plot in Example 2.3.7, and shows the instability of the sparse maximum likelihood method when the observational time steps are too large. Because of this instability, the sparse maximum likelihood estimation technique is disregarded here, and not considered in the rest of this chapter.

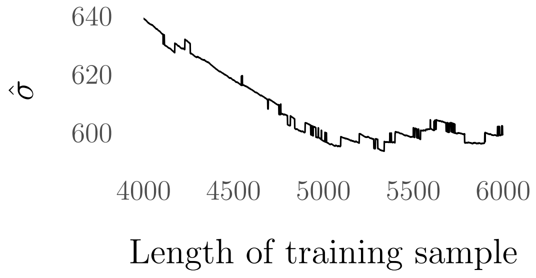
Using method of moments parameter estimation

We now consider the method of moments parameter estimation – as introduced in Section 2.3.3.1 – where for an Ornstein-Uhlenbeck realisation, the sample lag-1 correlation and variance is found and used in (2.31) to obtain estimates for the drift and diffusion.

Recall that this method is only valid for realisations that have positive sample lag-1 correlation. Figure 4.14 shows the sample lag-1 correlation for the training samples of the first six seasonally adjusted principal components. For each seasonally adjusted principal component $\tilde{m}_i$, $i = 1, \ldots, 6$, the sample lag-1 correlation is calculated for every training sample $\tilde{m}_i^{(j)}$, $j = 4000, \ldots, 5999$. From this, it is clear that only the fourth seasonally adjusted principal component has positive lag-1 sample correlation for all training samples, shown by the solid thick line in Figure 4.14. For this seasonally adjusted principal component, Ornstein-Uhlenbeck



(a) Sparse maximum likelihood estimator $\hat{\gamma}_{\mathrm{sML},1}(j)$.



(b) Sparse maximum likelihood estimator $\hat{\sigma}_{\mathrm{sML},1}(j)$.

Figure 4.12: Sparse maximum likelihood estimators $\hat{\gamma}_{\mathrm{sML},1}(j)$ and $\hat{\sigma}_{\mathrm{sML},1}(j)$ for the first seasonally adjusted principal component, for training samples of size $j = 4000, 4001, \ldots, 5999$. That is, each estimator pair $(\hat{\gamma}_{\mathrm{sML},1}(j), \hat{\sigma}_{\mathrm{sML},1}(j))$ is found using the corresponding training sample $\tilde{m}_1^{(j)} = (\tilde{m}_{11}, \tilde{m}_{12}, \ldots, \tilde{m}_{1j})$. It is clear from this figure that the sparse maximum likelihood estimator technique is sub-optimal for modelling the principal components arising from the HadCM3 mean sea-level air pressure output.

parameter estimates can be found using the method of moments for all training samples. The thick dotted, dot-dashed and dashed lines show the sample lag-1 correlations corresponding to training samples of the second, third and sixth seasonally adjusted principal components respectively. The sample lag-1 correlations obtained from the training samples from these three seasonally adjusted principal components have varying signs, and therefore for some of these training samples, the method of moments can be used to obtain Ornstein-Uhlenbeck parameter estimates. For the first and fifth seasonally adjusted principal components, all training samples have negative sample lag-1 correlation – as shown by the thin solid and dashed-long dashed lines respectively in Figure 4.14 – and therefore method of moments cannot be used here. The grey lines in Figure 4.14 give the 95% confidence interval for the sample lag-1 correlation of white-noise realisations. As all the training samples here have sample lag-1 correlation within this interval, training samples for which the sample lag-1 correlation is negative could be modelled as realisations from a white-noise instead of an Ornstein-Uhlenbeck process. We discuss this in more detail in Section 4.4.2.1.

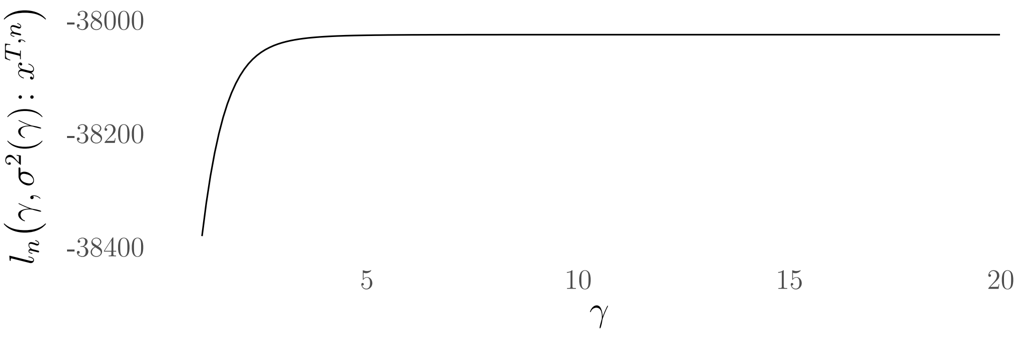


Figure 4.13: Log-likelihood (2.28) for varying $(\gamma, \sigma^2(\gamma))$ combinations, for the first seasonally adjusted principal component, $\tilde{m}_1$. As can be seen, the gradient of the log-likelihood for this set of observations is near zero for all γ values greater than around 5. This is similar behaviour to that seen in Example 2.3.7.

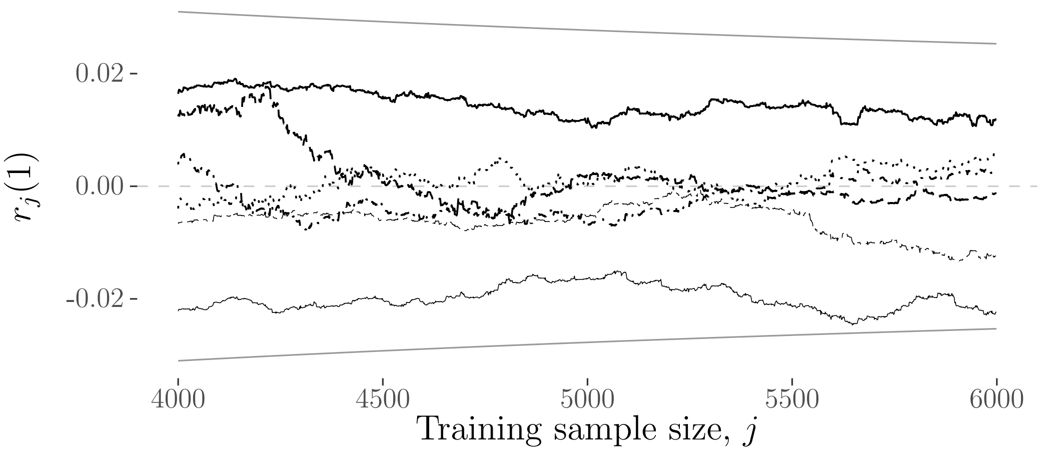


Figure 4.14: Sample lag-1 correlations for training samples $\tilde{m}_i^{(4000)}, \dots, \tilde{m}_i^{5999}$, $i = 1, \dots, 6$. The thick dotted, dot-dashed, solid and dashed line gives the sample lag-1 correlation corresponding to training samples of the second, third, fourth and sixth seasonally adjusted principal components respectively, for which for some training samples these are positive. The sample lag-1 correlations for the training samples of the first and fifth seasonally adjusted principal components are given by the thin solid and long dashed-dashed lines respectively, and are negative for all training samples. Recall that the method of moments Ornstein-Uhlenbeck parameter estimation can only be applied to training samples with positive sample lag-1 correlation. The grey lines give an approximate 95% confidence interval for the sample lag-1 correlation of a white noise realisation.

From the analysis of the sample lag-1 correlation, shown for the first six seasonally adjusted principal components in Figure 4.14, the method of moments can be applied to some of the training samples from the second, third, fourth and sixth seasonally adjusted principal components. For seasonally adjusted principal components $\tilde{m}_i$, $i = 2, 3, 4, 6$, and training samples $\tilde{m}_i^{(j)}$, $j = 4000, \ldots, 5999$ such that the sample lag-1 correlation of $\tilde{m}_i^{(j)}$ is positive, the method of moments is used to obtain Ornstein-Uhlenbeck parameter estimates $\hat{\gamma}_{\mathrm{MM},\,i}(j)$ and $\hat{\sigma}^2_{\mathrm{MM},\,i}(j)$. Figure 4.15a gives the drift estimates $\hat{\gamma}_{\mathrm{MM},\,i}(j)$ using training samples of the second, third, fourth and sixth seasonally adjusted principal components by the dotted, dot-dashed, solid and dashed lines respectively. Figure 4.15b gives the corresponding square-root of the diffusion estimate, $\hat{\sigma}_{\mathrm{MM},\,i}(j)$. The gaps in some of the lines in Figure 4.15 are at training samples of size j for which the sample lag-1 correlation is non-positive and thus the drift and diffusion estimates can not be found at these points using method of moments.

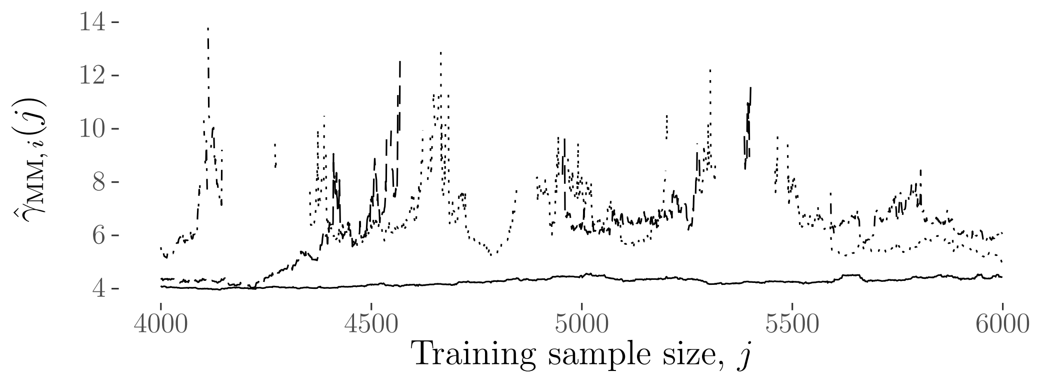
4.4.2.3 Modelling the principal components as block-average Ornstein-Uhlenbeck processes

In this section we model the observations from each seasonally adjusted principal component as realisations from some block-average Ornstein-Uhlenbeck processes, with parameter estimates found using the associated maximum likelihood estimator (introduced in Section 2.3.3.3).

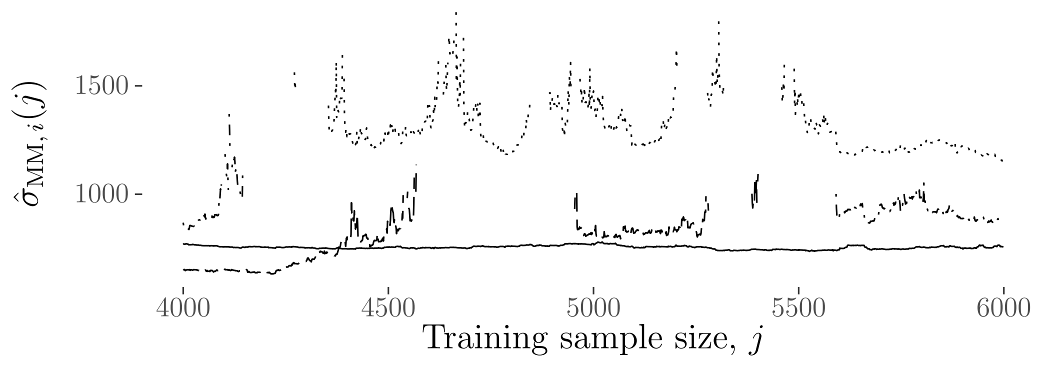
In this case each principal component (with seasonally effects still present) can be modelled as a realisation from some stochastic process $(\mathsf{Y}_{[t_{i-1},t_i)})_{i\in\mathbb{Z}}$ given by

$$\mathsf{Y}_{[t_{i-1},t_i)} = \mathsf{X}^{\mathrm{baOU}}_{[t_{i-1},t_i)} + \mu_{k(t_i)}, \tag{4.5}$$

where $\mathsf{X}^{\mathrm{baOU}}_{[t_{i-1},t_i)}$ is the block-average Ornstein-Uhlenbeck process as described in Section 2.3.2.1, $\mu_j \in \mathbb{R}$, $j = 1, \ldots, 12$, are some seasonal means, and $k(\cdot)$ is the seasonal index mapping given in (4.4). In other words, we assume that the principal components are realisations from some (piecewise) seasonal Ornstein-Uhlenbeck process that has been averaged over each season to give (4.5). That is, the process $\mathsf{Y}_t = \mathsf{X}^{\mathrm{OU}}_t + m(t)$ is averaged over each season, where here $m(t)$ is some



(a) Ornstein-Uhlenbeck parameter estimates $\hat{\gamma}_{\mathrm{MM},i}(j)$ for $i = 2,3,4,6$ and $j \in \{4000, 4001, \ldots, 5999\}$ for which the training sample $\tilde{m}_i^{(j)}$ has positive sample lag-1 correlation.



(b) Ornstein-Uhlenbeck parameter estimates $\hat{\sigma}_{\mathrm{MM},i}(j)$ for $i = 2,3,4,6$ and $j \in \{4000, 4001, \ldots, 5999\}$ for which the training sample $\tilde{m}_i^{(j)}$ has positive sample lag-1 correlation.

Figure 4.15: Ornstein-Uhlenbeck parameter estimates $\hat{\gamma}_{\mathrm{MM},i}(j)$ and $\hat{\sigma}_{\mathrm{MM},i}(j)$ for $i = 2,3,4,6$ and $j \in \{4000, 4001, \ldots, 5999\}$ for which the training sample $\tilde{m}_i^{(j)}$ has positive sample lag-1 correlation. The dotted, dot-dashed, solid and dashed line gives the Ornstein-Uhlenbeck parameter estimates corresponding to training samples of the second, third, fourth and sixth seasonally adjusted principal components respectively.

seasonal mean function with monthly mean values $\mu_1, \ldots, \mu_{12}$. We assume that the mean function is a quadratic spline function as described in Section 2.3.4.2 and shown in Figure 4.9 for the first three principal components.

As before, the seasonally adjusted principal components are given by the set $\{\tilde{m}_1, \tilde{m}_2, \ldots, \tilde{m}_l\}$ for some $l \leq r$, where $r \in \mathbb{N}$ is the rank of $\tilde{X}^{(g)}$. Here, as in the previous sections we choose $l = 6$.

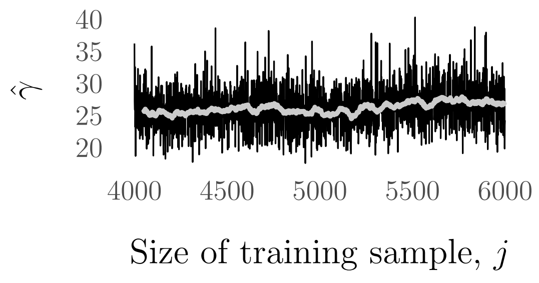
As in the sparse Ornstein-Uhlenbeck case (given in Section 4.4.2.2), the parameter estimates are found for the training samples of each seasonally adjusted principal component. That is, for each seasonally adjusted principal component, $\tilde{m}_i$, the estimates $\hat{\gamma}_{\text{sML},\,i}(j)$ and $\hat{\sigma}^2_{\text{sML},\,i}(j)$ are obtained from the associated training sample $\tilde{m}_i^{(j)} = (\tilde{m}_{i\,0}, \tilde{m}_{i\,1}, \ldots, \tilde{m}_{i\,(j-1)})$, $i = 1, \ldots, 6$, with $j = 4000, 4001, \ldots, 5999$. The parameter estimates $\hat{\gamma}_{\text{baML},\,i}(j)$ and $\hat{\sigma}_{\text{baML},\,i}(j)$ for seasonally adjusted principal components $i = 1, 2, 3$ and training sample size $j = 4000, 4001, \ldots, 5999$ are shown in Figure 4.16 by the black lines, with the thick grey lines giving the 50 step moving average. The gap in the parameter estimation for the second seasonally adjusted principal component is where the maximum likelihood optimisation failed to converge for a subset of the training samples. In this case, the parameter estimates for the largest training sample before this point is used instead.

4.4.3 Modelling the independent components

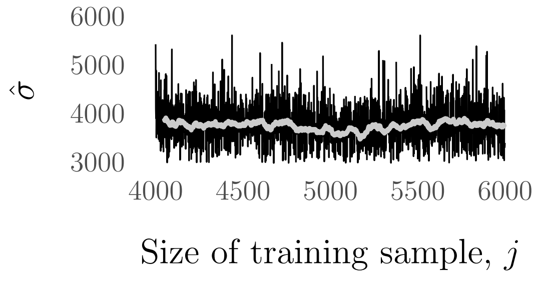
This section mirrors the layout of Section 4.4.2. First, we check for (and remove) seasonality of the independent components. Then, we assume that the seasonally adjusted independent components are realisations from a selection of stochastic processes and implement parameter estimation techniques on the associated training samples. As in Section 4.4.2, here we consider the ARIMA, Ornstein-Uhlenbeck, and block-average Ornstein-Uhlenbeck processes.

Figure 4.17 shows the correlograms (calculated from (2.4)) for the first six independent components. From this, as with the principal components discussed in Sections 4.4.2, we judge that seasonality is present and thus this needs to be removed before the modelling stage. As before, this is achieved by calculating and removing the seasonal means discussed in Section 4.2.

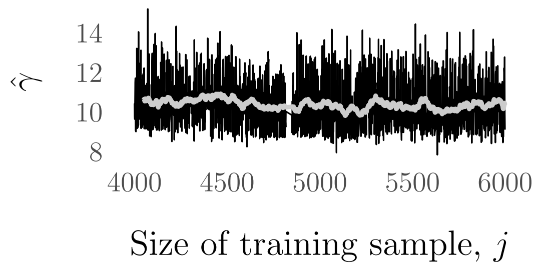
Figure 4.18 shows the first three independent components (plotted as a time-series) in the left-hand column, with the periodic quadratic spline calculated as in Section 2.3.4.2 using the seasonal means given by the grey line. The right-hand columns show the first three seasonally adjusted independent components. In this section, we apply stochastic modelling to these seasonally adjusted independent components. We note here that the seasonal effects are smaller for the independent



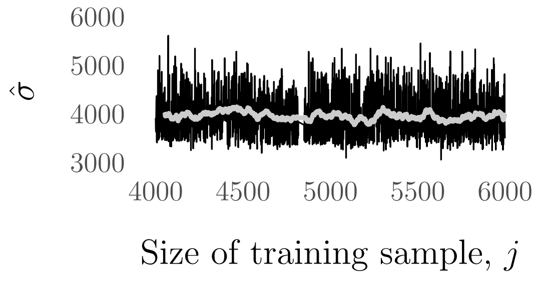
(a) Parameter estimates $\hat{\gamma}_{\text{baML},1}(j)$ using training samples of size $j = 4000, 4001, \ldots, 5999$.



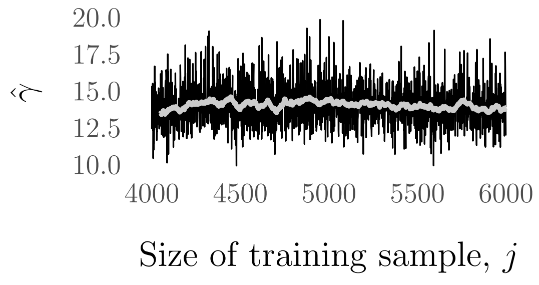
(b) Parameter estimates $\hat{\sigma}_{\text{baML},1}(j)$ using training samples of size $j = 4000, 4001, \ldots, 5999$.



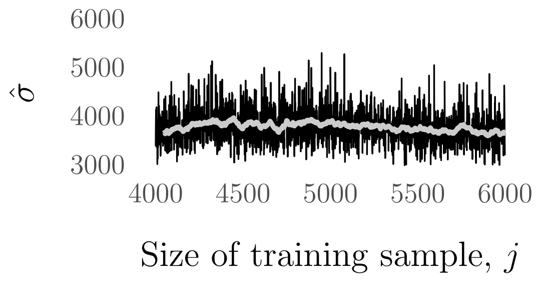
(c) Parameter estimates $\hat{\gamma}_{\text{baML},2}(j)$ using training samples of size $j = 4000, 4001, \ldots, 5999$.



(d) Parameter estimates $\hat{\sigma}_{\text{baML},2}(j)$ using training samples of size $j = 4000, 4001, \ldots, 5999$.

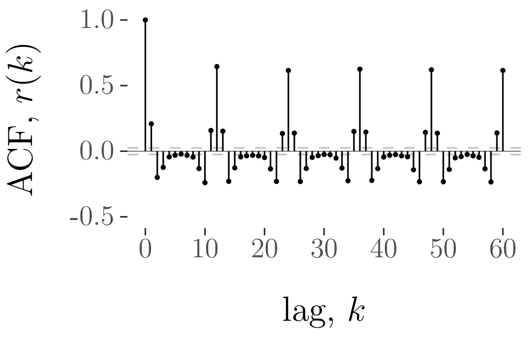


(e) Parameter estimates $\hat{\gamma}_{\text{baML},3}(j)$ using training samples of size $j = 4000, 4001, \ldots, 5999$.

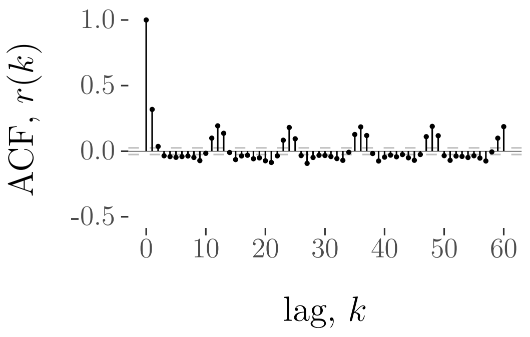


(f) Parameter estimates $\hat{\sigma}_{\text{baML},3}(j)$ using training samples of size $j = 4000, 4001, \ldots, 5999$.

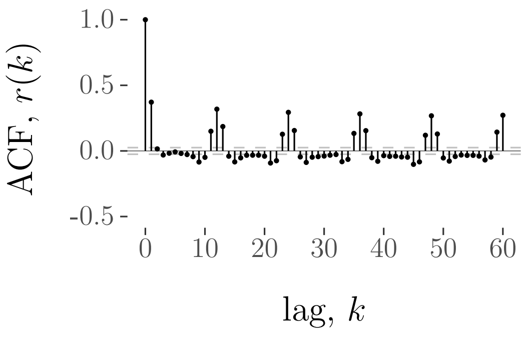
Figure 4.16: Parameter estimates $\hat{\gamma}_{\text{baML},i}(j)$ and $\hat{\sigma}_{\text{baML},i}(j)$ using block-average Ornstein-Uhlenbeck maximum likelihood estimation for the first three seasonally adjusted principal components, for increasing training sample size $j = 4000, 4001, \ldots, 5999$. That is, each estimator pair $\left(\hat{\gamma}_{\text{baML},i}(j), \hat{\sigma}_{\text{baML},i}(j)\right)$ is found using training sample $\tilde{m}_i^{(j)} = (\tilde{m}_{i1}, \tilde{m}_{i2}, \ldots, \tilde{m}_{ij})$, $i = 1, 2, 3$. The grey line shows the 50 step moving average.



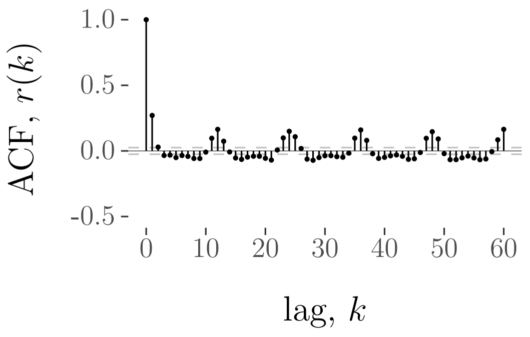
(a) Autocorrelation function (correlogram) for the first independent component, s_1, for lags $k = 0, 1, \ldots, 60$.



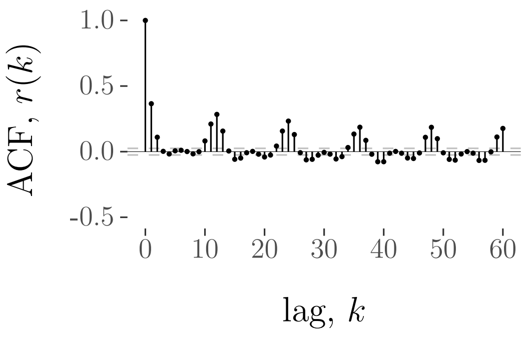
(b) Autocorrelation function (correlogram) for the second independent component, s_2, for lags $k = 0, 1, \ldots, 60$.



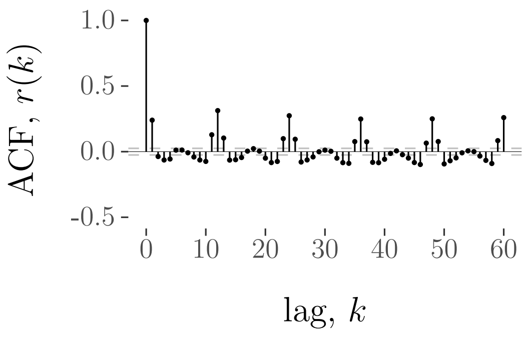
(c) Autocorrelation function (correlogram) for the third independent component, s_3, for lags $k = 0, 1, \ldots, 60$.



(d) Autocorrelation function (correlogram) for the fourth independent component, s_4, for lags $k = 0, 1, \ldots, 60$.



(e) Autocorrelation function (correlogram) for the fifth independent component, s_5, for lags $k = 0, 1, \ldots, 60$.



(f) Autocorrelation function (correlogram) for the sixth independent component, s_6, for lags $k = 0, 1, \ldots, 60$.

Figure 4.17: Autocorrelation functions (correlogram) for the first six independent component, $s_1, \ldots, s_6$, for lags $k = 0, 1, \ldots, 60$.

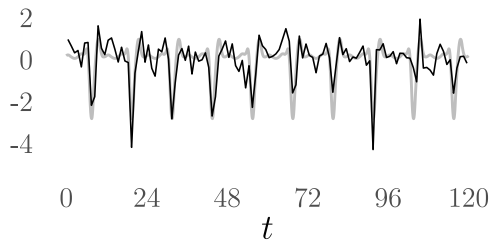
components than they were for the principal components. As principal component analysis aims to find projections of the data with large variances, this method often results in the first few components capturing some simple seasonal behaviour, and these will inherently have a large variance and thus a pronounced seasonality. For example, if we applied principal component analysis to some data from a simulator that gave an output of hourly temperatures, then we would expect the first principal component to capture the temperature variation arising from day and night. It is hoped that applying independent component analysis to some climate simulator results in components that explain some fundamental physical phenomena (Richman, 1986), and therefore it is probable that the components will exhibit seasonality. However, the seasonality present will be less pronounced than in the principal components as a large variance in the components is not the focus.

4.4.3.1 Modelling the independent components as observations from ARIMA processes

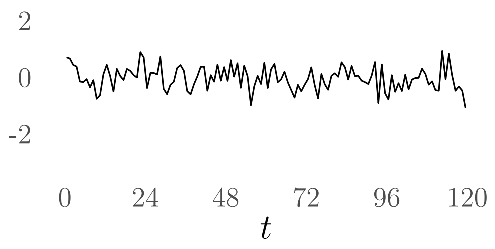
The seasonally adjusted independent components are assumed to be from some ARIMA process, and the model structure is found using the Box-Jenkins method.

Here we follow the procedure as described in Section 4.4.2.1, with seasonally adjusted independent components replacing the seasonally adjusted principal components. That is, the Box-Jenkins method is applied to the first six seasonally adjusted independent components, $\tilde{s}_1, \ldots, \tilde{s}_6 \in \mathbb{R}^p$, $p = 7008$, to obtain the estimates $\hat{p}_i$, $\hat{d}_i$ and $\hat{q}_i$, $i = 1, \ldots, 6$. Then, the training samples $\tilde{s}_i^{(j)} = (\tilde{s}_{i\,0}, \tilde{s}_{i\,1}, \ldots, \tilde{s}_{i\,(j-1)}) \in \mathbb{R}^j$, $j = 4000, 4001, \ldots, 5999$, are used to find the parameter estimates $\hat{\varphi}_{i,1}(j), \ldots, \hat{\varphi}_{i,p}(j)$ and $\hat{\vartheta}_{i,1}(j), \ldots, \hat{\vartheta}_{i,q}(j)$.

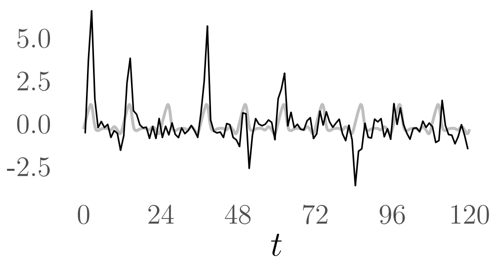
Here, detailed steps of the Box-Jenkins method are shown for the fifth seasonally adjusted independent component, mirroring the steps used in Example 3.1.1. We choose the fifth seasonally adjusted independent component as for the other five seasonally adjusted independent components, the sample autocorrelation and partial autocorrelation imply that these can only be modelled using white-noise processes (in the same was as the seasonally adjusted principal components were modelled). The estimated standard deviations for the training samples of the first,



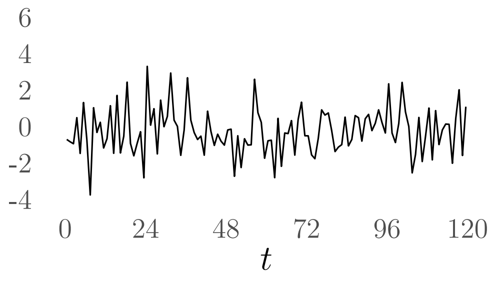
(a) First independent component, s_1 for times $t = 0, 1, \dots, 119$.



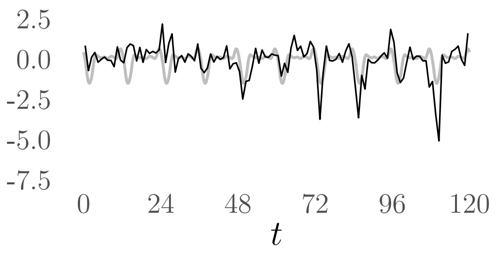
(b) First seasonally adjusted independent component, $\tilde{s}_1$ for times $t = 0, 1, \dots, 119$.



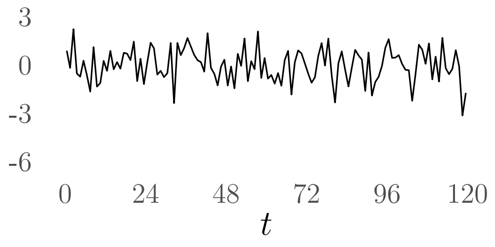
(c) Second independent component, s_2 for times $t = 0, 1, \dots, 119$.



(d) Second seasonally adjusted independent component, $\tilde{s}_2$ for times $t = 0, 1, \dots, 119$.



(e) Third independent component, s_3 for times $t = 0, 1, \dots, 119$.



(f) Third seasonally adjusted independent component, $\tilde{s}_3$ for times $t = 0, 1, \dots, 119$.

Figure 4.18: The left-hand column gives the first three independent components, s_1, s_2, s_3 for times $t = 0, 1, \dots, 119$, by the black lines, found by projecting the (column-centred) mean sea-level air pressure HadCM3 simulator output, $\tilde{X}$, along the first three independent directions, $w_1, w_2, w_3 \in \mathbb{R}^{7008}$, as shown in Figure 4.6. The grey lines give the quadratic spline calculated using the associated seasonal means. The right-hand column gives the first three seasonally adjusted independent components, $\tilde{s}_1, \tilde{s}_2, \tilde{s}_3$ for times $t = 0, 1, \dots, 119$.

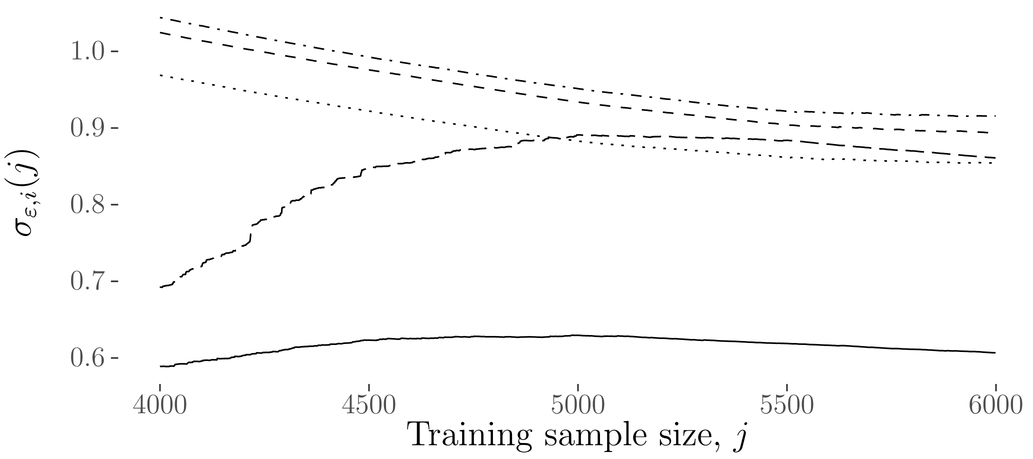


Figure 4.19: Estimates of the standard deviations of the white-noise process, $\hat{\sigma}_{\varepsilon,i}(j)$ for seasonally adjusted independent components $i = 1, 2, 3, 4, 6$ and for training samples $\tilde{s}_i^{(j)}$ of size $j = 4000, 4001, \ldots, 5999$. The standard deviation estimates using training samples from the first, second, third, fourth and sixth seasonally adjusted independent components are given by the solid, dashed, dotted, dot-dashed and long-dashed lines, respectively.

second, third, fourth and sixth seasonally adjusted independent components, are shown in Figure 4.19 by the solid, dashed, dotted, dot-dashed and long-dashed lines respectively.

Box-Jenkins method on the fifth seasonally adjusted independent component. Following the Box-Jenkins method, first we find an estimate for the order of difference d, followed by estimates for the autoregressive order, p, and the moving average order q.

As the autocorrelation function (shown in Figure 4.20a) has a quick decay, it is assumed that the fifth seasonally adjusted independent component is already in stationarity such that $\hat{d}_1 = 0$. Now, we judge that the autocorrelation function decays exponentially from the first lag, and the partial autocorrelation function is dominated by exponential decay from the first lag. From this and the discussion in Section 3.1.1, we obtain the estimates of the order of the autoregressive and moving-average parts of the process as $\hat{p}_1 = 1$ and $\hat{q}_1 = 1$ respectively. That is, the realisation is assumed to be from an ARMA(1, 1) process $(\mathsf{X}_t)_{t\in\mathbb{Z}}$, given by

$$\mathsf{X}_t = \varphi_1 \mathsf{X}_{t-1} + \varepsilon_t + \vartheta_1 \varepsilon_{t-1},$$

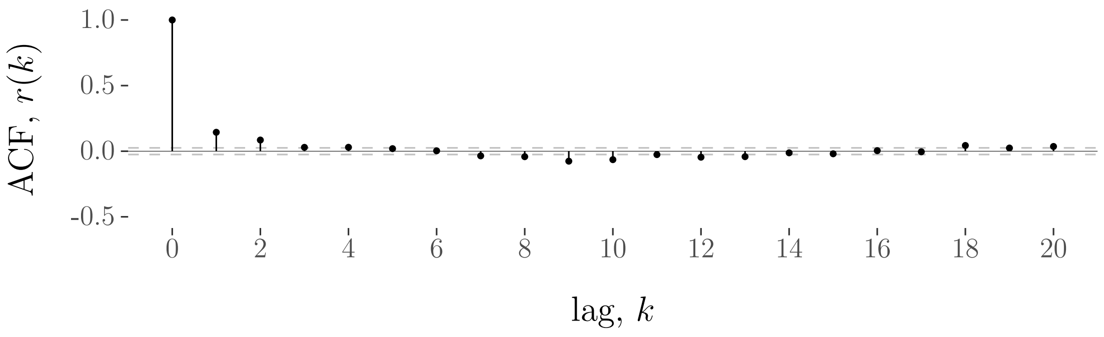
where $\varepsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ independent for all $t \in \mathbb{Z}$.

Now, we obtain estimates for the parameters φ_1 and ϑ_1, and for the variance of the white-noise term ε_t for each sample from the training set $\{\tilde{s}_5^{(4000)},\ \tilde{s}_5^{(4001)}, \dots,\ \tilde{s}_5^{(5999)}\}$. This gives the autoregressive parameter estimates $\hat{\varphi}_{5,1}(4000), \dots, \hat{\varphi}_{5,1}(5999)$, moving-average estimates $\hat{\vartheta}_{5,1}(4000), \dots, \hat{\vartheta}_{5,1}(5999)$, and the estimates of the variance of the white-noise $\sigma^2_{\varepsilon,5}(4000), \dots, \sigma^2_{\varepsilon,5}(5999)$. These are shown in Figure 4.21. From studying these estimates it appears that the model might change after around time $t = 5500$, as the parameter estimates change quite rapidly after this point. This has not been examined here, but should be kept in mind for future research.

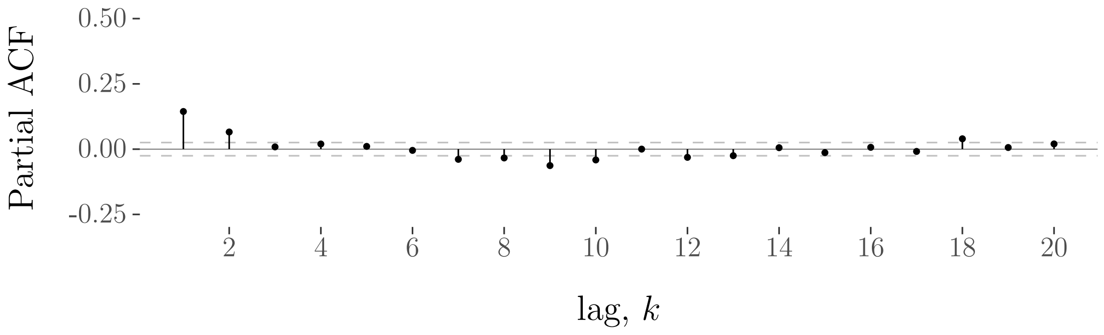
4.4.3.2 Modelling the independent components as sparse observations from Ornstein-Uhlenbeck processes

In this section we model the independent components – obtained by applying fastICA to the mean sea-level air pressure output from the HadCM3 simulator – as realisations from Ornstein-Uhlenbeck processes (after the seasonality has been removed). Note that in Section 4.4.2.2 it was determined that the sparse maximum likelihood method (which maximises the Markov chain log-likelihood (2.28)) is not suitable for estimating Ornstein-Uhlenbeck parameters here, and therefore in this section only the method of moments is considered.

First, the sample lag-1 correlation is calculated for the training sets of the first six seasonally adjusted independent components, $\{\tilde{s}_i^{(4000)}, \tilde{s}_i^{(4001)}, \dots, \tilde{s}_i^{(5999)}\}$, $i = 1, \dots, 6$. These are shown in Figure 4.22. The thick solid, dashed, dotted and dot-dashed lines correspond to the sample lag-1 correlations for training samples of the first, third, fifth and sixth seasonally adjusted independent components respectively. These four seasonally adjusted independent components have positive sample lag-1 correlation for all training samples, and therefore the method of moments can be applied here. The thin solid and long dashed-dashed lines correspond to the sample lag-1 correlations for training samples of the second



(a) Sample autocorrelation function (correlogram) up to lag 20 for the fifth seasonally adjusted independent component, $\tilde{s}_5$.

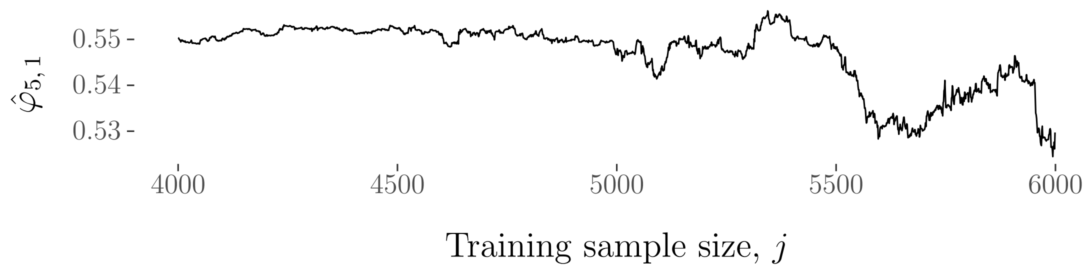


(b) Sample partial autocorrelation function up to lag 20 for the fifth seasonally adjusted independent component, $\tilde{s}_5$.

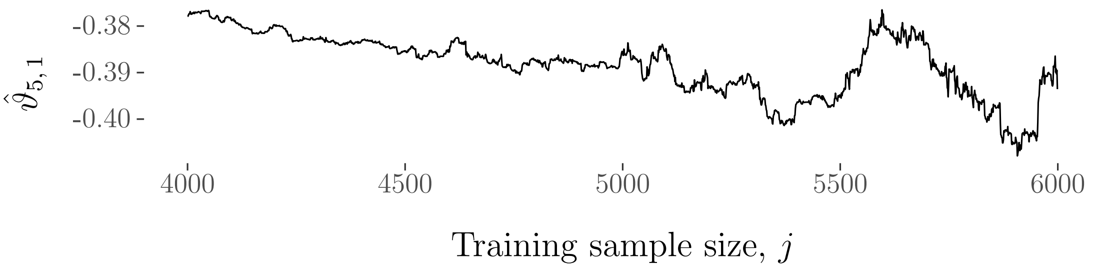
Figure 4.20: Sample autocorrelation and partial autocorrelation functions up to lag 20 for the fifth seasonally adjusted independent component, $\tilde{s}_5$.

and fourth seasonally adjusted independent components respectively. The second seasonally adjusted independent component has no training samples which have positive sample lag-1 correlation, and the fourth seasonally adjusted independent component has only a subset of the training set which has positive sample lag-1 correlations.

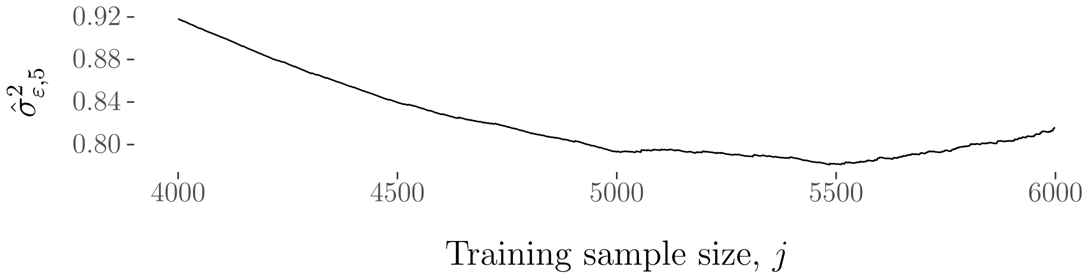
For training samples of the first, third, fourth, fifth and sixth seasonally adjusted independent components with positive sample lag-1 correlation, the parameter estimates obtained using method of moments are given by $\hat{\gamma}_{\mathrm{MM},i}(j)$ and $\hat{\sigma}_{\mathrm{MM},i}(j)$, $i = 1, 3, 4, 5, 6$, $j = 4000, \ldots, 5999$, and are shown in Figure 4.23 by the solid, dashed, long dashed-dashed, dotted and dot-dashed lines respectively. These



(a) Parameter estimate $\hat{\varphi}_5(4000), \ldots, \hat{\varphi}_5(5999)$ using the training set obtained from the fifth seasonally adjusted independent component, $\tilde{s}_5$.



(b) Parameter estimate $\hat{\vartheta}_5(4000), \ldots, \hat{\vartheta}_5(5999)$ using the training set obtained from the fifth seasonally adjusted independent component, $\tilde{s}_5$.



(c) Innovation variance estimate $\hat{\sigma}^2_{\varepsilon,5}(4000), \ldots, \hat{\sigma}^2_{\varepsilon,5}(5999)$ using the training set obtained from the fifth seasonally adjusted independent component, $\tilde{s}_5$.

Figure 4.21: Parameter estimates assuming that the fifth seasonally adjusted independent component, $\tilde{s}_5$ can be modelled by an ARMA$(1,1)$ process. Here, the parameter estimates are found using the training samples $\tilde{s}_5^{(4000)}, \ldots, \tilde{s}_t^{(5999)}$.

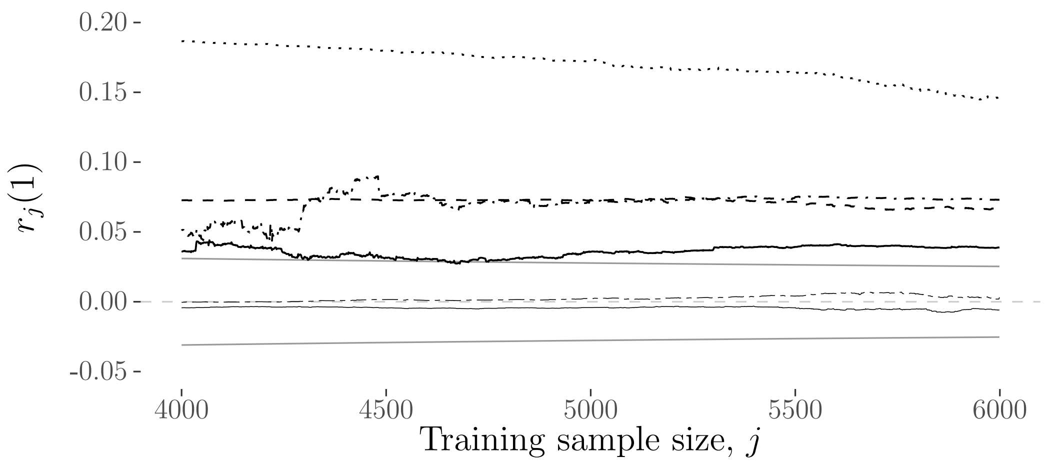


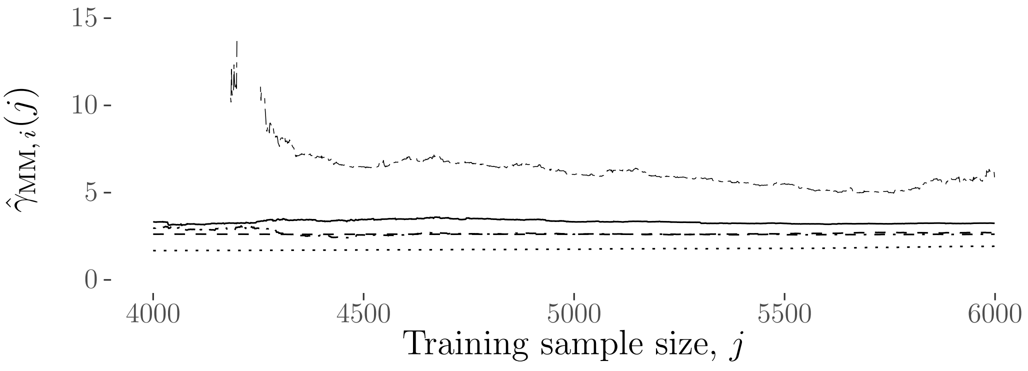
Figure 4.22: Sample lag-1 correlations for training samples of the first six seasonally adjusted independent components, $\tilde{s}_i^{(4000)}, \ldots, \tilde{s}_i^{5999}$, $i = 1, \ldots, 6$. The thick solid, dashed, dotted and dot-dashed lines correspond to the sample lag-1 correlation for training samples of the first, third, fifth and sixth seasonally adjusted independent components respectively. These four seasonally adjusted independent components have positive sample lag-1 correlation for all training samples. The thin solid and long dashed-dashed lines correspond to the sample lag-1 correlation for training samples of the second and fourth seasonally adjusted independent components respectively. The second seasonally adjusted independent component has no training samples with positive sample lag-1 correlation, and the fourth seasonally adjusted independent component has positive sample lag-1 correlation for only a subset of the training set.

parameter estimates all have asymptotic behaviour. This suggests that there are no sudden changes in the behaviour of these seasonally adjusted independent components that results in more than one stochastic process being required to model it satisfactorily.

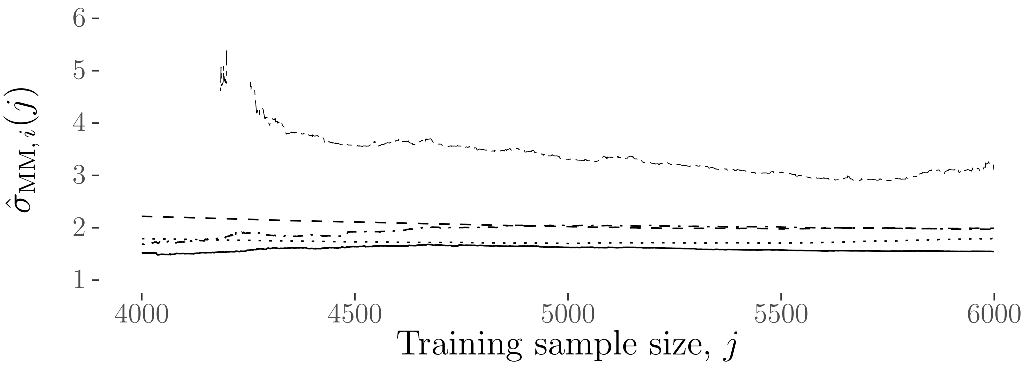
4.4.3.3 Modelling the independent components as observations from block-average Ornstein-Uhlenbeck processes

In this section we assume the seasonally adjusted independent components are realisations from some block-average Ornstein-Uhlenbeck processes, and use maximum likelihood estimation to find the drift and diffusion estimates.

Following the procedure used in Section 4.4.2.3, for each seasonally adjusted



(a) Estimates of $\hat{\gamma}_{\text{MM},i}(j)$ for seasonally adjusted independent components $i = 1, 3, 4, 5, 6$ and training samples of size $j = 4000, \ldots, 5999$.



(b) Estimates of $\hat{\sigma}_{\text{MM},i}(j)$ for seasonally adjusted independent components $i = 1, 3, 4, 5, 6$ and training samples of size $j = 4000, \ldots, 5999$.

Figure 4.23: Ornstein-Uhlenbeck parameter estimates for the first, third, fourth, fifth and sixth seasonally adjusted independent components, using the method of moments on the training samples $\tilde{s}_i^{(j)}$, $i = 1, 3, 4, 5, 6$, $j = 4000, \ldots, 5999$. The estimates are only shown for the training sample of size $j \in \{4000, \ldots, 5999\}$ such that the sample lag-1 correlation of $\tilde{s}_i^{(j)}$ is positive. This is true for all training samples of the first, third, fifth and sixth seasonally adjusted independent component, where the parameter estimates are given by the thick solid, dashed, dotted and dot-dashed lines respectively. The fourth seasonally adjusted independent component has some training samples with a positive sample lag-1 correlation, with associated parameter estimates given by the thin long dashed-dashed lines.

independent component, $\tilde{s}_i$, $i = 1, \ldots, 6$, maximum likelihood estimation is performed for each training sample $\tilde{s}_i^{(j)} = (\tilde{s}_{i\,0}, \tilde{s}_{i\,1}, \ldots, \tilde{s}_{i\,(j-1)})$, $j = 4000, \ldots, 5999$, to obtain block-average Ornstein-Uhlenbeck parameter estimates $\hat{\gamma}_{\text{baML},i}(j)$ and

$\hat{\sigma}_{\text{baML},i}(j)$. The parameter estimates obtained in this way for the first three seasonally adjusted independent component are shown in Figure 4.24. In this case the estimates found using the training samples of the first and third seasonally adjusted independent components seem to not behave in an asymptotic manner, and there are quite sudden changes in the estimates, suggesting that either the Ornstein-Uhlenbeck process is not satisfactory for modelling these seasonally adjusted independent components, or that multiple models are required here. As with the ARMA estimates in Section 4.4.3.1, this requires further research which is not performed here.

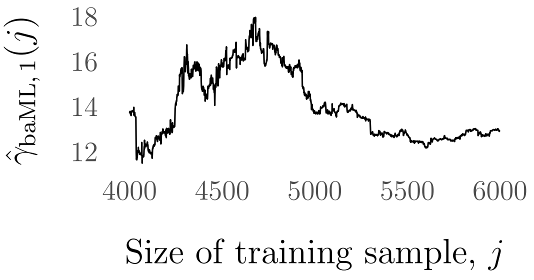
4.5 Forecasting Principal and Independent Components

Now, we use the stochastic models described in Section 4.4 to forecast the principal and independent components, with forecasting theory as introduced in Chapter 3.

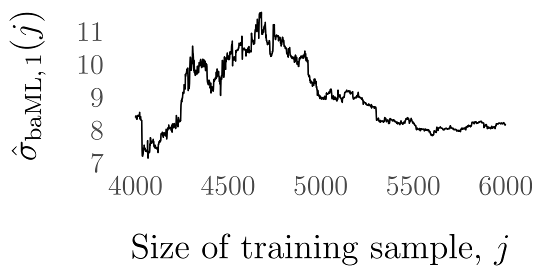
In Section 4.5.1 we introduce the forecasting approaches used, and one method to assess the forecasting quality. Mirroring the previous sections, the remainder of this section is split into two main subsections, relating to forecasts of the principal components and forecasts of the independent components obtained from applying principal component analysis and independent component analysis to the mean sea-level air pressure HadCM3 simulator output. In Section 4.5.2 we discuss forecasts for the principal components, and the forecasts for the independent components in Section 4.5.3.

4.5.1 Introduction to component forecasting

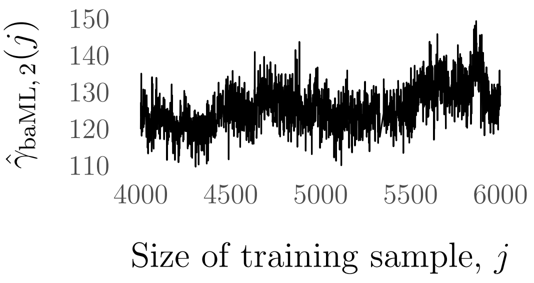
Let $x = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{R}^n$, assumed to be some realisation from a stochastic process, and $\{x^{(\tilde{n})}, x^{(\tilde{n}+1)}, \ldots, x^{(n-1)}\}$ the training set, where each training sample is given by $x^{(j)} = (x_0, \ldots, x_{j-1}) \in \mathbb{R}^j$, $j = \tilde{n}, \ldots, n-1$, and $\tilde{n} < n$. Recall from Chapter 3 that $\hat{x}_t(h)$ refers to a point forecast of the process at time $t+h$, using information $x_0, \ldots, x_t$ such that the forecast horizon is h. In this section we want to obtain *one-step-ahead* forecasts for the components, such that we use the



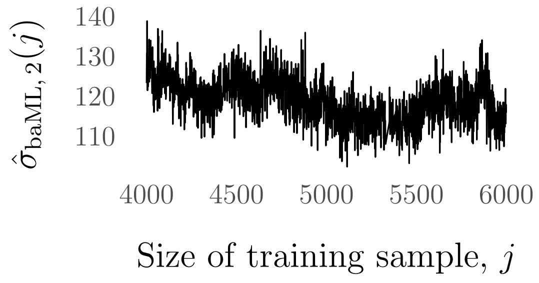
(a) Parameter estimates $\hat{\gamma}_{\text{baML},1}(j)$ using training samples of size $j = 4000, 4001, \ldots, 5999$.



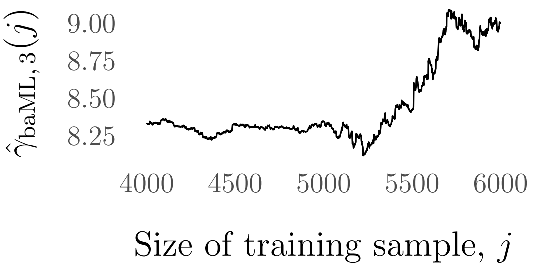
(b) Parameter estimates $\hat{\sigma}_{\text{baML},1}(j)$ using training samples of size $j = 4000, 4001, \ldots, 5999$.



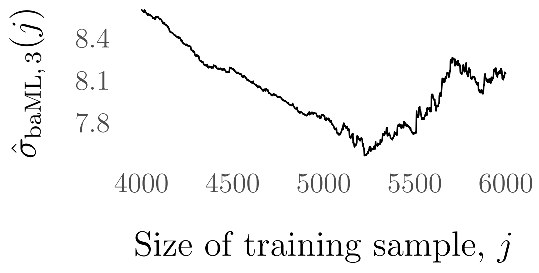
(c) Parameter estimates $\hat{\gamma}_{\text{baML},2}(j)$ using training samples of size $j = 4000, 4001, \ldots, 5999$.



(d) Parameter estimates $\hat{\sigma}_{\text{baML},2}(j)$ using training samples of size $j = 4000, 4001, \ldots, 5999$.



(e) Parameter estimates $\hat{\gamma}_{\text{baML},3}(j)$ using training samples of size $j = 4000, 4001, \ldots, 5999$.



(f) Parameter estimates $\hat{\sigma}_{\text{baML},3}(j)$ using training samples of size $j = 4000, 4001, \ldots, 5999$.

Figure 4.24: Parameter estimates $\hat{\gamma}_{\text{baML},i}(j)$ and $\hat{\sigma}_{\text{baML},i}(j)$ using block-average Ornstein-Uhlenbeck maximum likelihood estimation for the first three seasonally adjusted independent components, for increasing training sample size $j = 4000, 4001, \ldots, 5999$. That is, each estimator pair $\left(\hat{\gamma}_{\text{baML},i}(j), \hat{\sigma}_{\text{baML},i}(j)\right)$ is found using training sample $\tilde{s}_i^{(j)} = (\tilde{s}_{i1}, \tilde{s}_{i2}, \ldots, \tilde{s}_{ij})$, $i = 1, 2, 3$.

information from the training sample $x^{(t)} = (x_0, \dots, x_{t-1})$ to predict the value of the component at time t, for all times $t = \tilde{n}, \tilde{n}+1, \dots, n-1$. Therefore we need to calculate the point forecasts $\hat{x}_{\tilde{n}-1}(1), \hat{x}_{\tilde{n}}(1), \dots, \hat{x}_{n-2}(1)$.

Within both Section 4.5.2 and 4.5.3, two forecasting approaches are used. The first forecasting approach that we consider is motivated by the fact that the Ornstein-Uhlenbeck method of moments parameter estimation is only valid when the sample lag-1 correlation is positive. When the sample lag-1 correlation is positive, then the respective forecast can be obtained by considering the distribution of the Ornstein-Uhlenbeck with drift and diffusion estimates, as described in Chapter 3. However, when the sample lag-1 correlation is non-positive, the drift and diffusion parameters can not be estimated using method of moments. In this case (subject to the sample lag-1 correlation being within some confidence interval) we can model the observations as being a realisation from a white-noise process. We refer to this method as the *mixed forecasting* method, and it is described in detail in Panel 4.12

Panel 4.12: Mixed forecasting approach

Let $x = (x_0, x_1, \dots, x_{n-1}) \in \mathbb{R}^n$ be some observation (with no seasonality) with the associated training set $\{x^{(\tilde{n})}, x^{(\tilde{n}+1)}, \dots, x^{(n-1)}\}$, where each training sample is given by $x^{(j)} = (x_0, \dots, x_{j-1}) \in \mathbb{R}^j$, $j = \tilde{n}, \dots, n-1$. Here we describe the mixed forecasting method for obtaining one-step-ahead point forecasts $\hat{x}_{t-1}(1)$, $t = \tilde{n}, \tilde{n}+1, \dots, n-1$, and prediction intervals.

For $t = \tilde{n}, \dots, n-1$, we apply the following procedure:

1. Obtain the training sample $x^{(t)} = (x_0, \dots, x_{t-1}) \in \mathbb{R}^t$;
2. Calculate the sample lag-1 correlation of $x^{(t)}$, given by $r(1\colon x^{(t)})$;
3. If $r(1\colon \tilde{x}^{(t)})$ is positive, then the Ornstein-Uhlenbeck parameter estimates $\hat{\gamma}_{\mathrm{MM},\,i}(j)$ and $\hat{\sigma}_{\mathrm{MM},\,i}(j)$ are obtained via method of moments. Using these parameter estimates, we calculate the one-step-ahead point forecast $x_{t-1}(1)$ as in (3.5) and prediction interval as in (3.12).
4. If $r(1\colon \tilde{m}_i^{(t)})$ is non-positive, then use the white-noise standard deviation estimate $\sigma_{\varepsilon,\,i}(j)$ to obtain the one-step-ahead point forecast $x_{t-1}(1) = 0$ and prediction interval as in (3.10);

The second forecasting approach is called the *block-average Ornstein-Uhlenbeck forecasting approach* and assumes that every training sample of the observation $x = (x_0, x_1, \dots, x_{n-1})$ can be modelled as a realisation from a block-average Ornstein-Uhlenbeck process. In this case, for each training sample $x^{(j)}$, we calculate the associated parameter estimates $\hat{\gamma}_{\mathrm{baML}}(j)$ and $\hat{\sigma}^2_{\mathrm{baML}}(j)$. Using these we obtain the point forecast $\hat{x}_{j-1}(1)$ as given by (3.8) and the prediction interval using (3.13).

To help assess the validity of the forecasts, we calculate the rolling average of the root mean-squared error between the (one-step-ahead) point forecasts and the true values of the observation. That is, for observation $x = (x_0, x_1, \dots, x_{n-1})$ with training set $\{x^{(\tilde{n})}, x^{(\tilde{n}+1)}, \dots, x^{(n-1)}\}$, the rolling average of the root mean-squared error is given by

$$\sqrt{\frac{1}{j-\tilde{n}-1} \sum_{k=\tilde{n}}^{j} \big(x_k - \hat{x}_{k-1}(1)\big)^2}, \tag{4.6}$$

for $j = \tilde{n}, \tilde{n}+1, \dots, n-1$.

4.5.2 Forecasting principal components

Here, we obtain forecasts for the principal components relating to the mean sea-level air pressure HadCM3 simulator output, as described in Section 4.2.2. As discussed above, this section is split into forecasts found using the mixed forecasting approach (in Section 4.5.2.1), and using the block-average Ornstein-Uhlenbeck forecasting approach (in Section 4.5.2.2).

4.5.2.1 Mixed forecasting approach

We showed in Figure 4.15 that, of the first six seasonally adjusted principal components, only a subset of the training sets of the second, third, fourth and sixth components have positive sample lag-1 correlation. Only the training samples that have positive sample lag-1 correlation can be used to obtain Ornstein-Uhlenbeck drift and diffusion parameter estimates using method of moments. On the other hand, when we calculated the sample autocorrelation and partial autocorrelation of the first six seasonally adjusted principal components (in Section 4.4.2.1), we judged that these could all be modelled by white-noise processes. Amalgamating these two processes and parameter estimates, here we apply the mixed forecasting method as described in Panel 4.12 on the seasonally adjusted principal components $\tilde{m}_1, \ldots, \tilde{m}_6$. Each training set comprises of the training samples $\tilde{m}_i^{(4000)}, \tilde{m}_i^{(4001)}, \ldots, \tilde{m}_i^{(5999)}$, $i = 1, 2, \ldots, 6$. After we have calculated the point forecasts and prediction intervals for the seasonally adjusted principal components, we can add back the seasonal means to obtain forecasts for the principal components.

Figure 4.25 shows the point forecasts and prediction intervals for the first three principal components using the mixed forecasting approach for training samples of size $4000, \ldots, 4120$. In this case, for the first principal component, 1.5% of the test sample, $m_{1,4000}$, $m_{1,4001}, \ldots,$ $m_{1,5999}$, lie outside of the approximate 95% prediction intervals. For the second and third principal components, the respective proportions are 5.9% and 3.7%. Here we note that 1.5% seems like a low proportion to be outside the 95% prediction interval. This is much lower than all the toy examples given in Section 3.4. Also, the 95% prediction interval is an approximation using the approximate parameter values, and therefore by the comments made in Section 3.3.2 we would expect it to be narrower than a "true"

95% prediction interval, such that more than 5% of future observations lie outside of the interval. We discuss this more in the conclusion of this chapter.

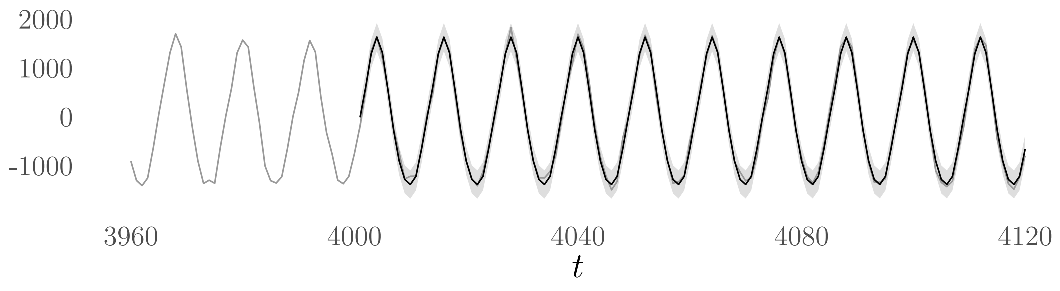
We now assess the validity of the predictions using the rolling average of the root mean-squared error as in (4.6). The solid, dashed and dotted lines in Figure 4.26 show the rolling averages of the root mean-squared error for principal components one, two and three respectively. In this case the point forecast of the first principal component gives the lowest root mean-squared rolling average. The most likely cause of this is due to strong seasonality in this principal component with low (relative) stochasticity. Therefore, the seasonal means account for a high proportion of the forecasting accuracy. The variability accounted for by the seasonality of the second and third principal components is less than the first (which can be seen visually by the plot of the first three seasonally adjusted principal components in Figure 4.9), which might be one cause for the larger rolling average of the root mean-squared error.

4.5.2.2 Block-average Ornstein-Uhlenbeck forecasting approach

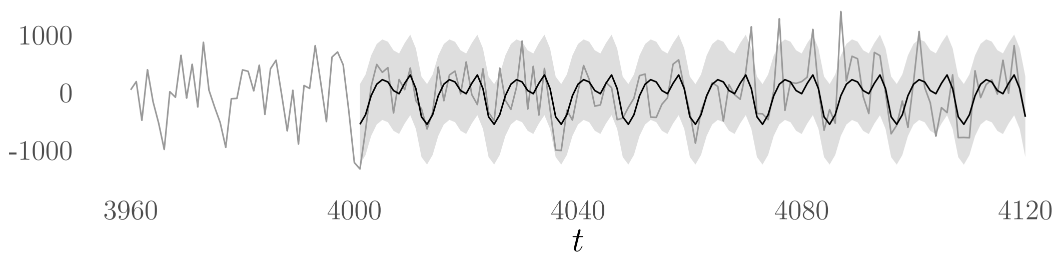
In this case, we assume that the training samples of the first six seasonally adjusted principal components, $\tilde{m}_i^{(j)}$, $i = 1, \ldots, 6$, $j = 4000, \ldots, 5999$, are realisations from independent block-average Ornstein-Uhlenbeck processes. Then, the drift and diffusion parameter estimates for each training sample are given by $\hat{\gamma}_{\mathrm{baML},\,i}(j)$ and $\hat{\sigma}^2_{\mathrm{baML},\,i}(j)$, and were calculated in Section 4.4.2.3. With these parameter estimates, we obtain the one-step-ahead point forecasts for the seasonally adjusted principal components using (3.8) and the prediction interval using (3.13).

In this case, the point forecasts and prediction intervals obtained are very similar to those obtained via the mixed forecasting approach (Section 4.5.2.1). Therefore, the following figures have been moved to Appendix A.1:

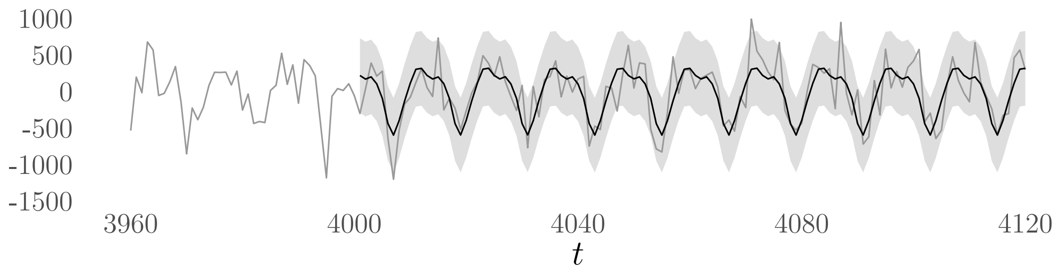
- The point forecasts and prediction intervals for the first three principal components, for training samples of size $j = 4000, \ldots, 4120$, shown in Figure A.1;
- The rolling average of the root mean-squared error as in (4.6), shown in Figure A.2.



(a) First principal component for $t = 3960, \ldots, 4120$ (grey solid line). The one-step-ahead point forecast is given by the solid black line, with the 95% prediction interval given by the shaded grey area.



(b) Second principal component for $t = 3960, \ldots, 4120$ (grey solid line). The one-step-ahead point forecast is given by the solid black line, with the 95% prediction interval given by the shaded grey area.



(c) Third principal component for $t = 3960, \ldots, 4120$ (grey solid line). The one-step-ahead point forecast is given by the solid black line, with the 95% prediction interval given by the shaded grey area.

Figure 4.25: First three principal components for $t = 3960, \ldots, 4120$ (grey solid line). The one-step-ahead point forecast found using the mixed forecasting method is given by the solid black line, with the 95% prediction interval given by the shaded grey area.

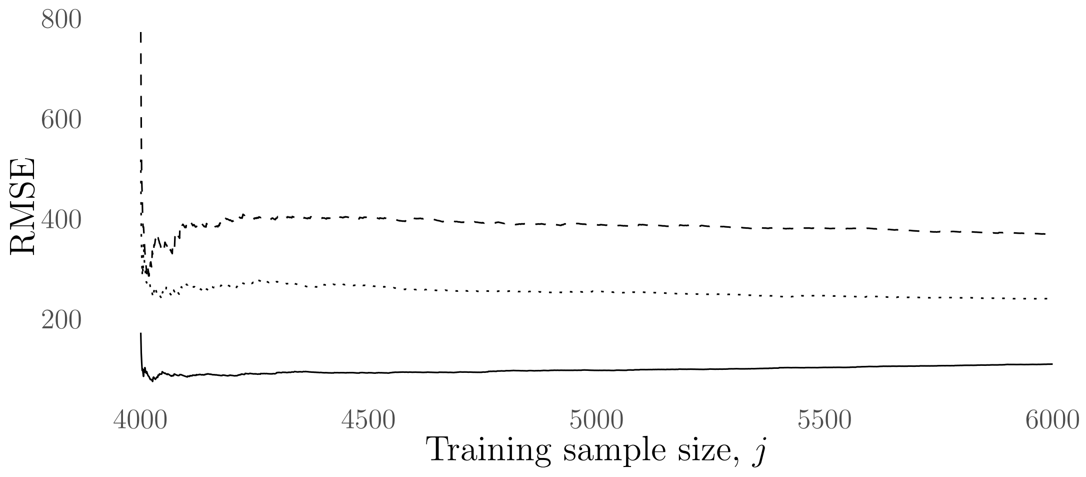


Figure 4.26: Rolling average of the root mean-squared error between the true principal components and the respective point forecasts, found using the mixed forecasting approach. The rolling average of the root mean-squared error between the first, second and third principal component and the respective one-step-ahead forecasts is given by the solid, dashed and dotted line respectively.

In this case, 1.5% of the realisation $m_{1\,4001}, \ldots, m_{1\,6000}$ lie outside the (approximate) 95% prediction interval. Similarly, for the second principal component, we have 5.6%, and for the third principal component we have 3.8%.

From considering the root mean-squared error, here we can surmise that the point forecasts obtained from the block-average forecasting approach are close to those obtained from the mixed forecasting approach. It is also clear that the prediction intervals using the different forecasting approaches were very similar to one-another, as the proportion of the test samples that lay outside of the 95% prediction intervals were comparable. This implies that misspecifying block-average realisations as pointwise realisations (as in the mixed forecasting approach), did not make a big difference to the resulting forecasts. This is in contrast to the toy example (Example 2.3.10), where misspecifying the realisations made a real difference to the result.

4.5.3 Forecasting independent components

Similarly to Section 4.5.2, here we find one-step-ahead forecasts for the first six independent components – obtained by applying the fastICA method to the mean sea-level air pressure HadCM3 simulator output – using the stochastic models and parameter estimates determined in Section 4.4.3.

The independent components are given by s_i, $i = 1, \ldots, 6$, with the seasonally adjusted independent components given by $\tilde{s}_i$. For each seasonally adjusted independent component, the associated training set is $\{\tilde{s}_i^{(4000)}, \tilde{s}_i^{(4001)}, \ldots, \tilde{s}_i^{(5999)}\}$, with the training samples given by $\tilde{s}_i^{(j)} = (\tilde{s}_{i\,0}, \tilde{s}_{i\,1}, \ldots, \tilde{s}_{i\,(j-1)}) \in \mathbb{R}^j$, $j = 4000, \ldots, 5999$. We use the notation $\hat{\tilde{s}}_{i\,t-1}(1)$ to give the one-step-ahead forecast of the i^{th} seasonally adjusted independent component at time $t-1$. By adding the seasonal mean to these forecasts we obtain $\hat{s}_{i\,t-1}(1)$, the associated one-step-ahead forecast for the i^{th} independent component at time $t-1$.

Below, we find the one-step-ahead forecasts for the independent components using the mixed forecasting approach as described in Panel 4.12, and the equivalent forecasts using the block-average Ornstein-Uhlenbeck forecasting approach.

4.5.3.1 Mixed forecasting approach

In this section, we apply the mixed forecasting approach (given in Panel 4.12) to the first six seasonally adjusted independent components. We have found (in Section 4.4.3.2) that only the second and fourth seasonally adjusted independent components have training samples with non-positive sample lag-1 correlation, and both of these correlations lie within the 95% confidence interval for white-noise sample correlation. This, along with examination of the sample autocorrelation and partial autocorrelation functions (which we did in Section 4.4.3.1), is used as justification for choosing white-noise for the training samples with non-positive sample lag-1 correlation.

In Figure 4.27 we show the one-step-ahead point forecasts for the independent components using the (seasonally adjusted) training sets $\tilde{s}_i^{(j)}$, for $i = 1, 2, 3$ and $j = 4000, \ldots, 4119$ by the black line, with the shading giving a 95% prediction interval. The grey lines give the true independent components, shown as a time series for times $t = 3960, \ldots, 4120$. In this case, 6.6% of the test sample for the first independent component, $s_{1\,4000}$, $s_{1\,4001}, \ldots$, $s_{1\,5999}$, lies outside of the 95% prediction interval. For the second and third independent component test sets, this proportion is 0.5% and 1.1% respectively. The values for the second and third independent components here are quite low, suggesting that the prediction intervals are too large, possible due to model misspecification.

Now we further assess the quality of the forecasts by looking at the rolling averages of the root mean-squared errors given by (4.6). Figure 4.26 gives the rolling average of the root mean-squared errors between independent component one, two and three, and the associated point forecasts, by the solid, dashed and dotted lines respectively. The increasing nature of the rolling average root mean-squared error for the second and third independent component forecasts after around $j = 5000$ implies that potentially these independent components need to be modelled by different stochastic processes (or with different parameter estimates) after these points.

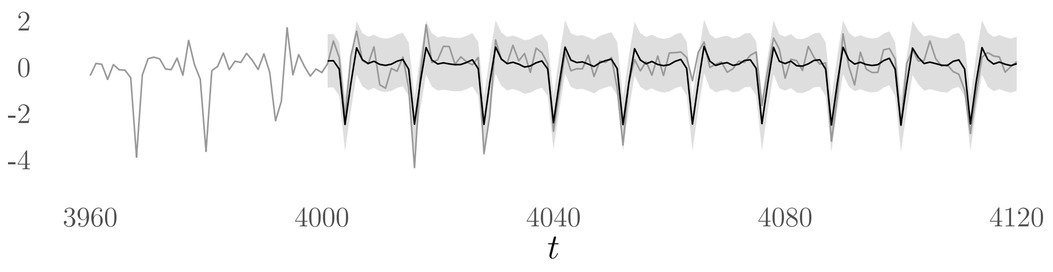
4.5.3.2 Block-average Ornstein-Uhlenbeck forecasting approach

Here we obtain forecasts for the independent components using the block-average Ornstein-Uhlenbeck forecasting approach, and the block-average Ornstein-Uhlenbeck drift and diffusion parameter estimates that were found (in Section 4.4.3.3) for the training samples $\tilde{s}_i^{(j)}$, $i = 1, \ldots, 6$, $j = 4000, \ldots, 5999$.

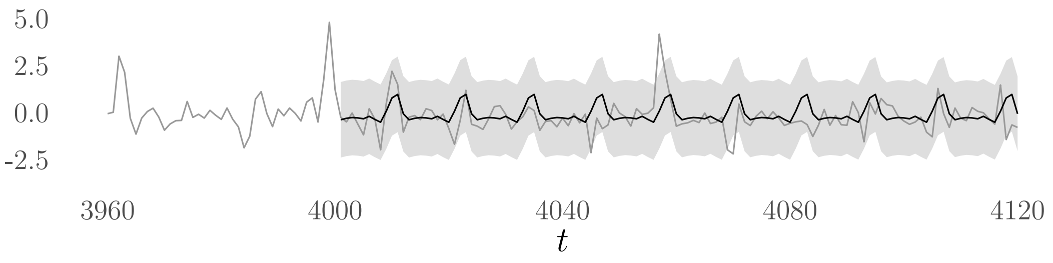
In this case, the point forecasts and prediction intervals obtained are very similar to those obtained via the mixed forecasting approach (Section 4.5.3.1), and therefore the following figures have been placed in Appendix A.1:

- The point forecasts and prediction intervals for the first three independent components at times $t = 4000, 4001, \ldots, 4119$, shown in Figure A.3;
- The rolling average of the root mean-squared error, shown in Figure A.4.

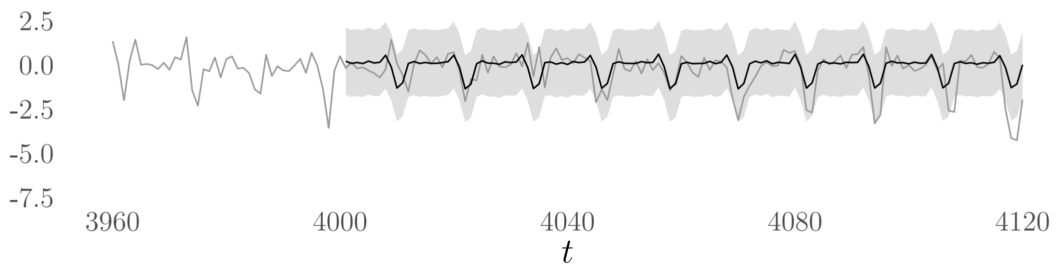
In this case, 6.6% of the test sample of the first independent component, $s_{1\,4000}$, $s_{1\,4001}, \ldots$, $s_{1\,5999}$, is outside of the 95% prediction interval. For the test samples of the second and third independent components, 0.5% and 1.1% of the values lie outside of the respective 95% prediction interval. Note that these are



(a) First independent component for $t = 3960, \ldots, 4120$ (grey solid line). The one-step-ahead point forecasts found using the mixed forecasting approach are given by the solid black line, with the 95% prediction interval given by the shaded grey area.



(b) Second independent component for $t = 3960, \ldots, 4120$ (grey solid line). The one-step-ahead point forecasts found using the mixed forecasting approach are given by the solid black line, with the 95% prediction interval given by the shaded grey area.



(c) Third independent component for $t = 3960, \ldots, 4120$ (grey solid line). The one-step-ahead point forecasts found using the mixed forecasting approach are given by the solid black line, with the 95% prediction interval given by the shaded grey area.

Figure 4.27: First three independent components for $t = 3960, \ldots, 4120$ (grey solid line). The one-step-ahead point forecasts found using the mixed forecasting approach are given by the solid black line, with the 95% prediction intervals shown by the shaded grey area.

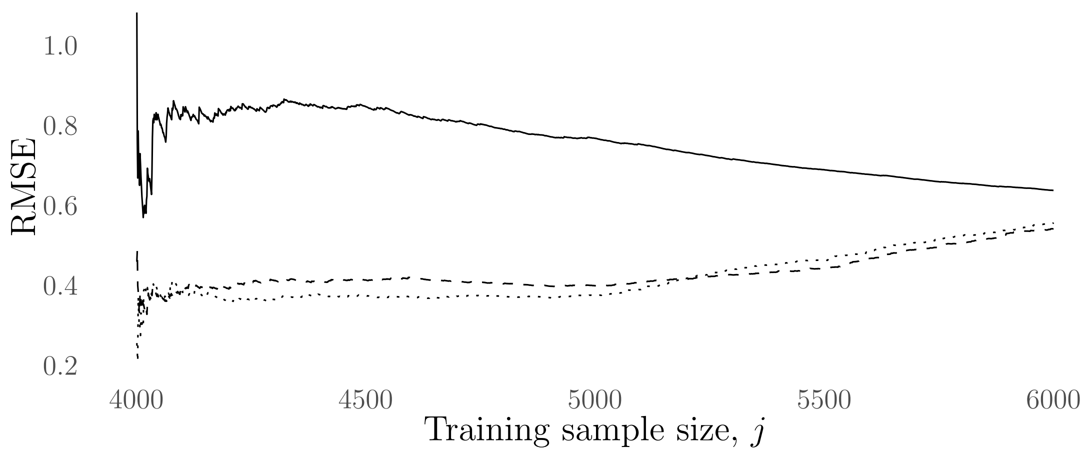


Figure 4.28: Rolling average of the root mean-squared error between the true independent components and the respective point forecasts, found using the mixed forecasting approach. The solid line corresponds to the first independent component, the dashed line to the second, and the dotted line to the third.

the exact same proportions as were found with the prediction intervals found using the mixed forecasting approach.

We conclude here that the forecasts of the independent components obtained using the block-average forecasting approach are very similar to those obtained using the pointwise mixed forecasting approach. That is, the misspecification of the independent components as pointwise realisations (as opposed to averaged realisations) does not make a substantial difference to the forecasts obtained. Recall that this was also the case in the principal component forecasts (seen in Section 4.5.2), such that both the mixed forecasting approach and the block-average Ornstein-Uhlenbeck forecasting approach gave very similar results.

4.6 Reconstruction of the Climate Simulator Output

In this section, we compare the full mean sea-level air pressure output of the HadCM3 climate simulator to the reconstruction found using $l = 6$ principal or independent components, and to the reconstruction found using the one-step-ahead point forecasts for the $l = 6$ principal and independent components.

In Section 4.6.1 we introduce the methods used to find the reconstructions, a way to analyse the quality of the reconstructions, and some terminology used throughout this section. After this introduction, the remainder of the section is split into analysing the reconstructions found using the principal components (Section 4.6.2), and analysing the reconstructions found using the independent components (Section 4.6.3).

4.6.1 Introduction to reconstructing the climate simulator output

Reconstructions from the principal or independent components (and the associated one-step-ahead forecasts) are obtained using *inverse principal component analysis* or *inverse independent component analysis*. These are described below in Panel 4.13 and Panel 4.14 respectively.

The comparisons in this section are done both by considering root mean-squared errors and by visual inspection of the reconstructions. One initial comment on the use of root mean-squared error to compare reconstructions found via inverse principal component analysis to those found using inverse independent component analysis is made below in Remark 4.18. For the visual comparisons the following plots are given at times $t = 4000$ and $t = 5000$:

1. The HadCM3 simulation mean sea-level air pressure output;
2. Reconstructions using $l = 6$ components;
3. Reconstructions using one-step-ahead point forecasts for the $l = 6$ components.

Panel 4.13: "Inverse" principal component analysis

For principal component analysis, reconstructions of the full simulator output are found by doing "inverse" principal component analysis in the following way. As in Section 4.2, let the true simulator output be given by $X \in \mathbb{R}^{n \times p}$, and the scaled output by $X^{(g)} = XG$ with $G \in \mathbb{R}^{p \times p}$ the diagonal matrix with elements corresponding to the respective grid cell area. Then letting $\tilde{X}^{(g)}$ be the column-centred version of $X^{(g)}$, the singular value decomposition of $\tilde{X}^{(g)}$ is given by $\tilde{X}^{(g)} = U \Lambda V^{\top}$. Here, the columns of V are the principal directions and the principal components are given by the columns of $M = \tilde{X}^{(g)} V$.
Let $M^{(l)} \in \mathbb{R}^{n \times l}$ be the matrix with columns the first l principal components, and let $V^{(l)} \in \mathbb{R}^{p \times l}$ be the matrix of the first l columns of V. Then, inverse principal component analysis using l components proceeds as follows.

1. Set $\hat{\hat{Z}}^{(l,G)} = M^{(l)}(V^{(l)})^{\top}$;
2. Add the column means of the original matrix X to $\hat{\hat{Z}}^{(l,G)}$ to give matrix $\hat{Z}^{(l,G)}$;
3. Set G^{-1} the diagonal matrix with diagonal elements $1/\sqrt{c_i}$, where c_i is the grid cell area for the cells $i = 1, \ldots, p$. We rescale $\hat{Z}^{(l,G)}$ by the inverse of the cell grid size to obtain an approximation to the original simulator output X, given by $Z^{(l)} = \hat{Z}^{(l,G)} G^{-1}$.

We also plot the differences at times $t = 4000$ and $t = 5000$ between:

(i) The true simulator output and the reconstruction using $l = 6$ components;

(ii) The true simulator output and the reconstruction using one-step-ahead point forecasts for the $l = 6$ components;

(iii) The reconstruction using $l = 6$ components and the reconstruction using one-step-ahead point forecasts for the $l = 6$ components.

Recall that the HadCM3 simulator output here runs for times $t = 0, 1, \ldots, 5999$, and we obtained forecasts for the principal and independent components in Section 4.5 for times $t = 4000, 4001, \ldots, 5999$. Therefore, the choice of $t = 4000$ and

Panel 4.14: "Inverse" independent component analysis

For independent component analysis, reconstructions of the full simulator output are found by doing "inverse" independent component analysis in the following way. As in Section 4.3 and following the notation introduced in Remark 1.1, let the true simulator output be given by $X \in \mathbb{R}^{n\times p}$, and the column-centred version by $\tilde{X} \in \mathbb{R}^{n\times p}$. Now, performing independent component analysis on $\tilde{X}$ using the method described in Panel 1.4 (Section 1.3.1.2), the independent components are given by the columns of the matrix $S \in \mathbb{R}^{n\times r}$, where $S = \tilde{X}AQ$, $A \in \mathbb{R}^{p\times r}$ is the whitening matrix such that the sample covariance of $Y = \tilde{X}A$ is equal to identity, and $Q \in \mathbb{R}^{r\times r}$ is an orthogonal matrix.
Assume the first l independent components have been kept and are given by columns of $S^{(l)} \in \mathbb{R}^{n\times l}$. Let $Q^{(l)} \in \mathbb{R}^{r\times l}$ be the first l columns of the Q matrix. Then, inverse independent component analysis using these components is as follows.

1. Set $B = (\tilde{X}AQ)^{+}\tilde{X}$, where '+' indicates the Moore–Penrose inverse. Then, from Lemma 1.3.2, $\tilde{X}AQB = \tilde{X}$.
2. Set $B^{(l)} \in \mathbb{R}^{l\times p}$ to be the first l rows of B, and set $\hat{\tilde{Z}}^{(l)} = S^{(l)}B^{(l)}$;
3. Add the column means of the original matrix X to $\hat{\tilde{Z}}^{(l)}$ to give matrix $\hat{Z}^{(l)}$, which is an approximation to the original simulator output X.

$t = 5000$ for visual comparison in this section is such that we get a start and mid-point reconstruction using the component forecasts. Figure 4.29 shows the true mean sea-level air pressure HadCM3 simulator output for these two time points.

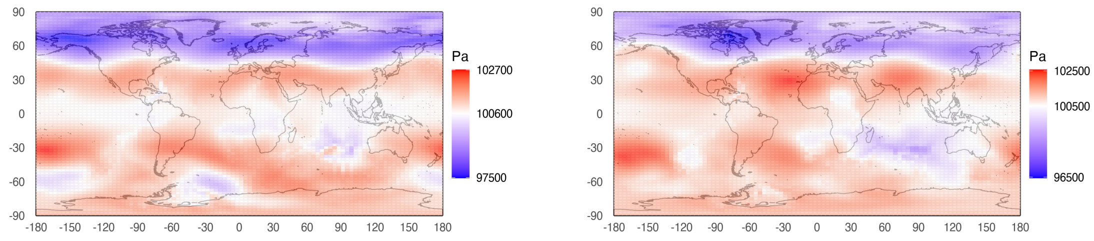
The root mean-squared error between the reconstructed simulator output using l principal components, given by $Z^{(l)}$ as in Panel 4.13, and the true output is,

$$e(t\colon l) := \left(\frac{1}{p}\sum_{i=1}^{p}(z_{ti}^{(l)} - x_{ti})^2\right)^{1/2}, \tag{4.7}$$

where here $Z^{(l)} = \left(z_0^{(l)} \cdots z_{n-1}^{(l)}\right)^\top \in \mathbb{R}^{n\times p}$ and $X = (x_0 \cdots x_{n-1})^\top \in \mathbb{R}^{n\times p}$ are such that z_{ij} (respectively, x_{ij}) is the $(i+1)^{\text{th}}$ row, j^{th} column element of $Z^{(l)}$

Remark 4.18: The use of root mean-squared error for comparison

Here it is remarked that principal component analysis – due to the maximal variance of the projections – obtains components that minimise root mean-squared error. Therefore, when comparing inverse principal component analysis using l principal components to inverse independent component analysis using l independent components, the root mean-squared error of the former will theoretically always be lower than the latter.
Due to this, here we give more weight to the comparison of the root mean-squared errors between the reconstructions using the true components and the reconstructions using the forecasted components. That is, we want to compare how well the reconstructions found using the forecasted components mirror those found using the true components, instead of how well the reconstructions found using the forecasted components mirror the true HadCM3 simulator output.



(a) Time, $t = 4000$.

(b) Time, $t = 5000$.

Figure 4.29: Mean sea-level air pressure HadCM3 simulator output at times $t = 4000, 5000$.

(respectively, X).

We now define the root mean-squared errors between the reconstructed climate simulator output using l principal or independent components, and the one-step-ahead forecasts for the l components.

First, define $\hat{M}_{\tilde{n}}^{(l)} \in \mathbb{R}^{n \times l}$ to be the matrix with columns given by the true principal components up to element $\tilde{n}$, followed by the one-step-ahead principal component forecast up to element n. That is, the i^{th} column of $\hat{M}_{\tilde{n}}^{(l)}$ is given by the n-vector

$$\big(m_{i\,0}, m_{i\,1}, \ldots, m_{i\,(\tilde{n}-1)}, \hat{m}_{i,\tilde{n}-1}(1), \hat{m}_{i,\tilde{n}}(1), \ldots, \hat{m}_{i,n-2}(1)\big). \tag{4.8}$$

Here, $\hat{m}_{i,j-1}(1) = \mathsf{E}\big(\mathsf{M}_{i\,j}|m_{i\,j-1}, \ldots, m_{i\,0}\big)$, is some one-step-ahead forecast found using the training set $m_i^{(j)} = \big(m_{i\,0}, \ldots, m_{i\,(j-1)}\big)$, where it is assumed that the principal component m_i is a realisation from some stochastic process $(\mathsf{M}_{i\,t})_{t\in\mathbb{Z}}$. In this section $n = 6000$, $\tilde{n} = 4000$ and the columns of $\hat{M}_{\tilde{n}}^{(l)}$ are known as the *one-step-ahead forecasted principal components.* This is a slight abuse of terminology, as actually only the latter elements $m_{i\,4000}, \ldots,\ m_{i\,5999}$ have been replaced with the one-step-ahead forecasts $\hat{m}_{i\,3999}(1), \ldots, \hat{m}_{i\,5998}(1)$.

Similarly, define $\hat{S}_{\tilde{n}}^{(l)} \in \mathbb{R}^{n\times l}$ to be the matrix with columns given by the true independent components up to element $\tilde{n}$, followed by the one-step-ahead independent component forecast up to element n, such that the i^{th} column of $\hat{S}_{\tilde{n}}^{(l)}$ is given by the n-vector

$$\big(s_{i\,0}, s_{i\,1}, \ldots, s_{i\,(\tilde{n}-1)}, \hat{s}_{i,\tilde{n}-1}(1), \hat{s}_{i,\tilde{n}}(1), \ldots, \hat{s}_{i,n-2}(1)\big). \tag{4.9}$$

As before, the columns of $\hat{S}_{\tilde{n}}^{(l)}$ are known as the *one-step-ahead forecasted independent components.*

Substituting $M^{(l)}$ for $\hat{M}_{\tilde{n}}^{(l)}$ in the inverse principal component analysis method (Panel 4.13) gives the one-step-ahead forecasted principal component reconstruction matrix $\hat{Z}_{\tilde{n}}^{(l)}$, whilst substituting $S^{(l)}$ for $\hat{S}_{\tilde{n}}^{(l)}$ in the inverse independent component analysis method (Panel 4.14) gives the equivalent one-step-ahead forecasted independent component reconstruction matrix.

Now, the root mean-squared error between the true HadCM3 simulator output and the reconstructed output using one-step-ahead forecasts of the l components is given by

$$\hat{e}(t\colon l,\, X) := \Big(\frac{1}{p}\sum_{i=1}^{p}(\hat{z}_{ti}^{(l)} - x_{ti})^2\Big)^{1/2}, \tag{4.10}$$

for $t = \tilde{n}, \tilde{n}+1, \ldots, n$. For $t = 0, \ldots, \tilde{n}-1$ this is equal to the root mean-squared error between the true output and the l components reconstruction, $e(t\colon l)$ in (4.7).

The root mean-squared error between the reconstructed simulator output using l components, $Z^{(l)} = (z_1^{(l)} \cdots z_n^{(l)})^\top \in \mathbb{R}^{n\times p}$, and the reconstructed output using one-step-ahead forecasts for the l components, $\hat{Z}_{\tilde{n}}^{(l)} = \left(\hat{z}_1^{(l)} \cdots \hat{z}_n^{(l)}\right)^\top \in \mathbb{R}^{n\times p}$, is given by

$$\hat{e}(t\colon l,\, Z) := \left(\frac{1}{p}\sum_{i=1}^{p}(\hat{z}_{ti}^{(l)} - z_{ti}^{(l)})^2\right)^{1/2}, \tag{4.11}$$

for $t = \tilde{n}, \tilde{n}+1, \dots, n$. For $t = 1, \dots, \tilde{n}-1$, this is equal to zero, as the reconstruction $Z_{\tilde{n}}^{(l)}$ is obtained using components given by (4.8) and (4.9), where the first $\tilde{n}$ elements are equal to the true components.

The rolling average of each root mean-squared error is also considered, where the average value of the reconstruction root mean-squared error of all previous reconstructions from $t = 4000$ onwards is taken. That is, the rolling averages of (4.7), (4.11) and (4.10) are

$$\frac{1}{j-3999}\sum_{t=4000}^{j} e(t\colon l); \quad \frac{1}{j-3999}\sum_{t=4000}^{j} \hat{e}(t\colon l,\, Z); \quad \frac{1}{j-3999}\sum_{t=4000}^{j} \hat{e}(t\colon l,\, X), \tag{4.12}$$

for $j = 4000, \dots, 5999$.

The rest of this section is split into two parts, with reconstruction of the mean sea-level air pressure output via inverse principal component analysis discussed in Section 4.6.2, and via inverse independent component analysis in Section 4.6.3. A short comment is made below in Remark 4.19 that explains the terminology in use in this section.

Remark 4.19: Terminology

Throughout this section, the following terminology is used for simplicity:

1. *Principal component reconstructions* refers to reconstructions obtained via inverse principal component analysis using a set of the first l principal components. Throughout this section we use $l = 6$ principal components in the reconstructions;

2. *One-step-ahead forecasted principal components* refers to the columns of $\hat{M}_{\tilde{n}}^{(l)}$ as given in (4.8). Throughout this section we set $l = 6$ and $\tilde{n} = 4000$;

3. *Forecasted principal component reconstructions* refers to reconstructions obtained via inverse principal component analysis using one-step-ahead point forecasts in lieu of the true principal components. That is, the matrix $\hat{M}_{\tilde{n}}^{(l)}$ is used instead of $M^{(l)}$ in the inverse principal component analysis method (described in Panel 4.13) with $l = 6$ and $\tilde{n} = 4000$;

4. *Independent component reconstructions* refers to reconstructions obtained via inverse independent component analysis using a set of the first l independent components. Throughout this section we use $l = 6$ independent components in the reconstructions;

5. *One-step-ahead forecasted independent components* refers to the columns of $\hat{S}_{\tilde{n}}^{(l)}$ as given in (4.9). Throughout this section we set $l = 6$ and $\tilde{n} = 4000$;

6. *Forecasted independent component reconstructions* refers to reconstructions obtained via inverse independent component analysis using one-step-ahead point forecasts in lieu of the true independent components. That is, the matrix $\hat{S}_{\tilde{n}}^{(l)}$ is used instead of $S^{(l)}$ in the inverse independent component analysis method (described in Panel 4.14) with $l = 6$ and $\tilde{n} = 4000$.

4.6.2 Reconstruction using principal components

In this section we find forecasted principal component reconstructions, and we compare these to the true HadCM3 simulator output, and to the principal component reconstructions.

In Section 4.6.2.1 we introduce principal component reconstructions and compare these to the true HadCM3 simulator output. In the remainder of this section we consider reconstructions calculated using two versions of the forecasted principal component matrix $\hat{M}^{(l)}_{\tilde{n}}$. In Section 4.6.2.2, we consider forecasts for the principal components obtained via the mixed forecasting approach. In Section 4.6.2.3, we consider forecasts obtained using the block-average Ornstein-Uhlenbeck forecasting approach. We remark that by the analysis performed in Section 4.6.2, where both forecasting approaches led to very similar point forecasts and prediction intervals, we expect the two reconstructions to be very similar to each other.

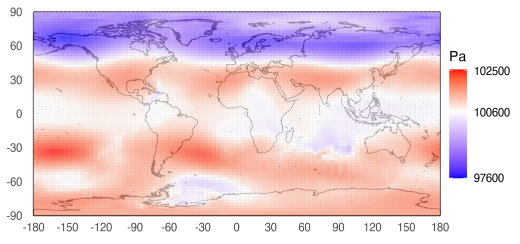
4.6.2.1 Principal component reconstructions

Before we consider the forecasted principal components reconstructions we introduce the principal component reconstructions calculated using $l = 6$ principal components. These principal component reconstructions are shown visually at time steps $t = 4000$ and $t = 5000$ in Figure 4.30. The difference between the HadCM3 simulator output and the principal component reconstructions at these times steps is shown in Figure 4.31. This plot follows from Item (i) discussed in the introduction above (Section 4.6.1). We note that the colour scheme used in both these figures mask that the underlying values and ranges are different, which are shown in the associated plot legends.

We also show the difference between the HadCM3 simulator output and the principal component reconstruction for times $t = 4000, 4001, \ldots, 5999$ by way of root mean-squared error in Figure 4.32. The grey circles give the root mean-squared error (as given in (4.7)), and the black line gives the rolling average of the root mean-squared error (as given in (4.12)). Here the y-axis shows the average root mean-squared difference *per grid cell* in Pascals. The approximate atmospheric pressure at sea-level is known as the *standard atmosphere (atm)*, with 1 atm being equal to $101,325$ Pa, and thus a difference of 250 Pa represents an error of less than 0.25%.

4.6.2.2 Reconstruction errors using the mixed forecasting approach

In this section, we analyse forecasted principal component reconstructions when the mixed forecasting approach is used to obtain one-step-ahead forecasts for the


(a) Time, $t = 4000$

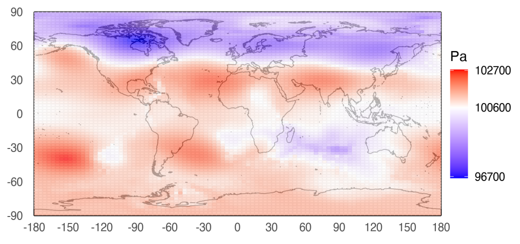
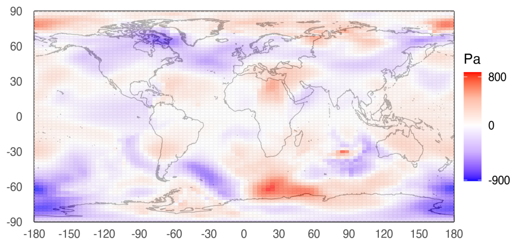

(b) Time, $t = 5000$

Figure 4.30: Principal component reconstructions with $l = 6$ principal components at times $t = 4000, 5000$.


(a) Time, $t = 4000$

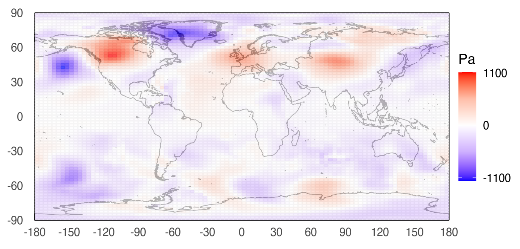

(b) Time, $t = 5000$

Figure 4.31: Difference between the true mean sea-level air pressure HadCM3 simulator output and the principal component reconstructions with $l = 6$ principal components at times $t = 4000, 5000$.

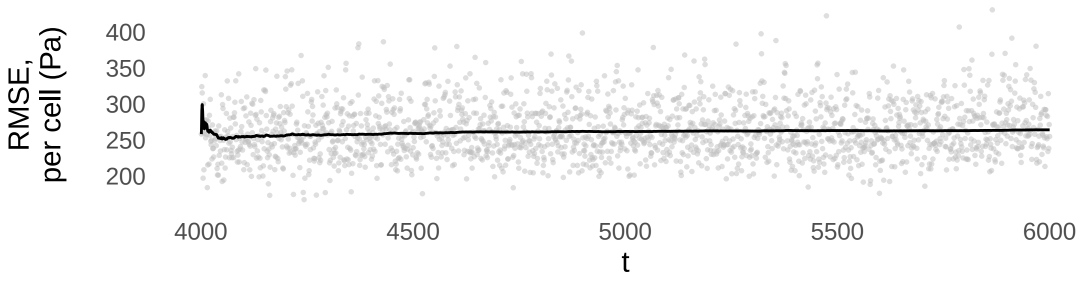
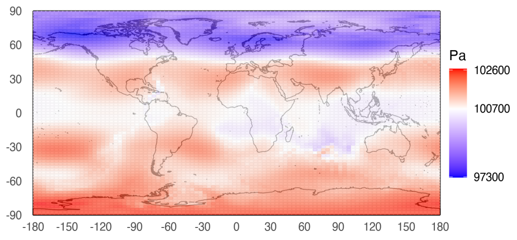


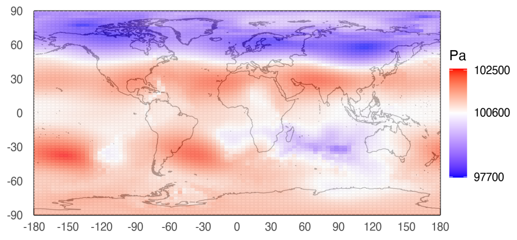
Figure 4.32: Root mean-squared error per grid cell between the true simulator output and the principal component reconstructions with $l = 6$ principal components, for times $t = 4000, \ldots, 5999$. The grey circles give the specific (per cell) root mean-squared errors (4.7), with the black line giving the running average of root mean-squared error over time (4.12).

principal components (as seen in Section 4.5.2). The forecasted principal component reconstructions at times $t = 4000$ and $t = 5000$ are shown in Figure 4.33.

Figure 4.34 shows the difference between the true HadCM3 mean sea-level air pressure output and the forecasted principal component reconstructions at times $t = 4000$ and $t = 5000$. Figure 4.35 shows the difference between the principal component reconstruction, and the forecasted principal component reconstruction using the mixed forecasting approach. These two figures show the differences as discussed in Items (ii) and (iii) in the introduction above (Section 4.6.1). As before, these plots of the differences have a much smaller range than the plots showing the forecasted principal component reconstructions, and are centred around zero.

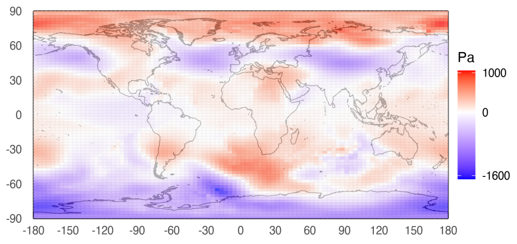


(a) Time, $t = 4000$.

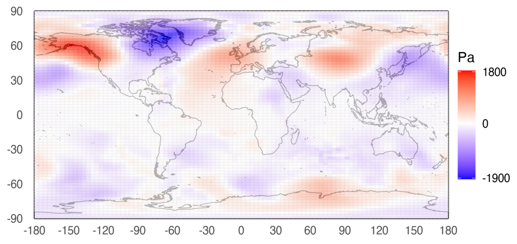


(b) Time, $t = 5000$.

Figure 4.33: Forecasted principal component reconstructions with $l = 6$ principal components using the mixed forecasting approach, at times $t = 4000, 5000$.

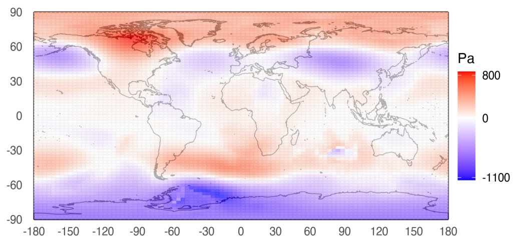


(a) Time, $t = 4000$.

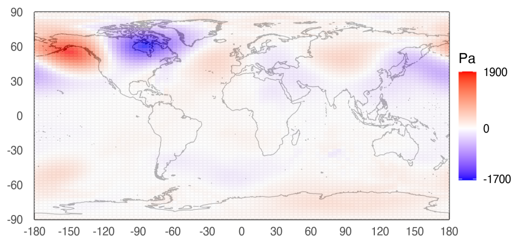


(b) Time, $t = 5000$.

Figure 4.34: Difference between the true mean sea-level air pressure HadCM3 simulator output and the forecasted principal component reconstruction with $l = 6$ principal components using the mixed forecasting approach, at times $t = 4000, 5000$.



(a) Time, $t = 4000$.



(b) Time, $t = 5000$.

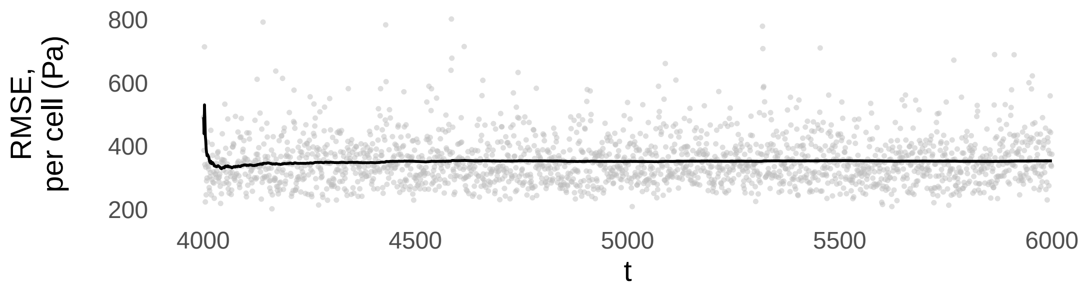
Figure 4.35: Difference between the principal component reconstructions of the mean sea-level air pressure with $l = 6$ principal components, and the forecasted principal component reconstructions using the mixed forecasting approach, at times $t = 4000, 5000$.

We assess the quality of these reconstructions using root mean-squared error. The grey circles in Figure 4.36a and Figure 4.36b show the root mean-squared errors between the forecasted principal component reconstructions (found using the mixed forecasting approach) and: *(i)* the true HadCM3 simulator output (as in (4.10)), and; *(ii)* the principal component reconstructions (as in (4.11)), respectively. The black line shows the associated rolling average as in (4.12).

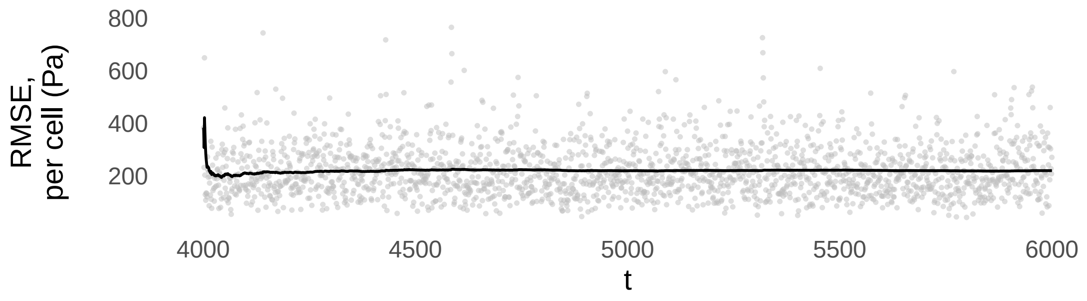
Recall that Figure 4.32 (Section 4.6.2) gives the root mean-squared errors for the differences between the HadCM3 simulator output and the principal component reconstructions (using the actual principal components). Comparing this to Figure 4.36a, we see that the root mean-squared errors when using the forecasted principal component reconstructions are larger than the errors when using the principal component reconstructions. This is expected, as here we are replacing the true principal components in the inverse principal component analysis step with the one-step-ahead point forecasts using the mixed forecasting approach.

4.6.2.3 Reconstruction errors using the block-average Ornstein-Uhlenbeck forecasting approach

In this section, we analyse the forecasted principal component reconstructions when the block-average Ornstein-Uhlenbeck forecasting approach is used to obtain one-step-ahead forecasts for the first $l = 6$ principal components. These principal



(a) Root mean-squared error per grid cell between the true HadCM3 simulator output and the forecasted principal component reconstructions with $l = 6$ principal components using the mixed forecasting approach.



(b) Root mean-squared error per grid cell between the principal component reconstructions using $l = 6$ principal components, and the forecasted principal component reconstructions using the mixed forecasting approach.

Figure 4.36: The grey circles give the root mean-squared error per grid cell (calculated using (4.10) and (4.11)) comparing the forecasted principal component reconstructions with $l = 6$ principal components using the mixed forecasting approach, to both: *(i)* the true HadCM3 simulator mean sea-level air pressure output (Figure 4.36a), and; *(ii)* the principal component reconstructions using the actual $l = 6$ principal components (Figure 4.36b). The black line gives the rolling average over time, as in (4.12).

component forecasts were given in Section 4.5.2.2. The block-average forecasting approach here produces very similar reconstructions to those found when the mixed forecasting approach was used, and therefore all figures for this section are included in Appendix A.2. The relevant figures here are:

- The forecasted principal component reconstructions at times $t = 4000$ and $t = 5000$, shown in Figure A.5;
- The difference between the HadCM3 simulator output and the forecasted

principal component reconstructions, at times $t = 4000$ and $t = 5000$, shown in Figure A.6;

- The difference between the principal component reconstructions using $l = 6$ principal components, and the forecasted principal component reconstructions at times $t = 4000$ and $t = 5000$, shown in Figure A.7;

- The root mean-squared errors between the forecasted principal component reconstructions and; *(i)* the HadCM3 simulator output, shown in Figure A.8a; and *(ii)* the principal component reconstruction with $l = 6$ principal components, shown in Figure A.8b.

To conclude the principal component forecasted reconstruction section, we comment that the reconstructions using the mixed forecasting approach and those using the block-average forecasting approach appear similar. This was expected as the principal component forecasts found in Section 4.5.2 were very similar to one-another. It is also worth remarking that the root mean squared errors between the forecasted principal component reconstructions and the true HadCM3 output are larger than those between the true principal component reconstructions and the true HadCM3 output. We would expect this as information is being lost when the true components are replaced by their forecasts. We have included the plots of the differences between the various reconstructions at times $t = 4000$ and $t = 5000$ so that it is possible for an expert to examine whether there are some systematic errors occurring in the reconstructions.

4.6.3 Reconstruction using independent components

In this section we find forecasted independent component reconstructions, and we compare these to the true HadCM3 simulator output, and to the independent component reconstructions. This section has the same layout as the previous section (Section 4.6.2), and all the plots produced in this section can be compared to the respective plots shown in that section.

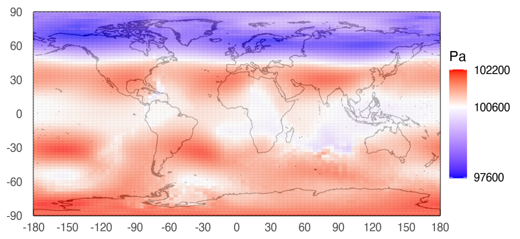
In Section 4.6.3.1 we introduce independent component reconstructions and compare these to the true HadCM3 simulator output. Then, in the remainder of this section we consider forecasted independent component reconstructions calculated using two versions of the forecasted independent component matrix $\hat{S}_{\tilde{n}}^{(l)}$ (with columns defined by (4.9)). In Section 4.6.3.2 we consider forecasts obtained using the mixed forecasting approach, and in Section 4.6.3.3 we consider forecasts obtained using the block-average Ornstein-Uhlenbeck forecasting approach. As with the forecasted principal component reconstructions, here we expect both versions of the forecasted independent component reconstructions to be very similar, as the point forecasts and prediction intervals for the independent components were seen to be very alike (in Section 4.5.3).

4.6.3.1 Independent component reconstructions

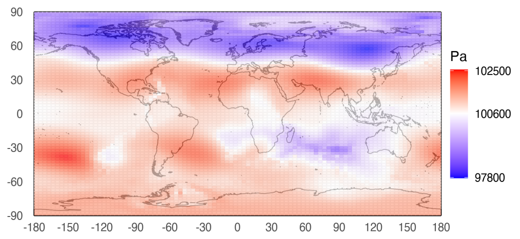
We first consider the independent component reconstructions, found using $l = 6$ independent components. The independent component reconstructions at times $t = 4000$ and $t = 5000$ are shown in Figure 4.37. The difference between the HadCM3 simulator output and the independent component reconstructions at these times is shown in Figure 4.38.

The quality is assessed by considering the root mean-squared errors between the independent component reconstructions and the true HadCM3 simulator output for times $t = 4000, 4001, \dots, 5999$. These are shown by the grey circles in Figure 4.39 with the black line giving the associated rolling average.

As expected (and commented on in Remark 4.18), the root mean-squared errors from the principal component reconstructions (shown in Figure 4.32) are smaller than those found using the independent component reconstructions. However, recalling that the average air pressure at sea-level is approximately $101,325$ Pa,

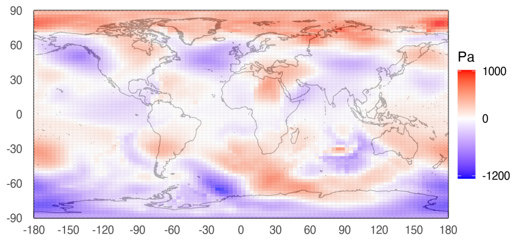


(a) Time, $t = 4000$.

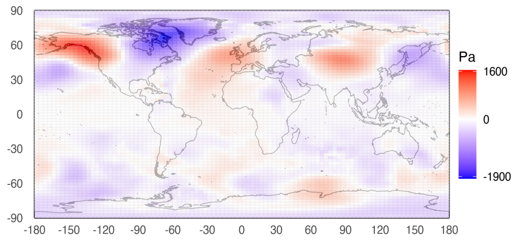


(b) Time, $t = 5000$.

Figure 4.37: Reconstructions of the mean sea-level air pressure HadCM3 simulator output via inverse independent component analysis with $l = 6$ independent components at times $t = 4000, 5000$.



(a) Time, $t = 4000$.



(b) Time, $t = 5000$.

Figure 4.38: Difference between the true mean sea-level difference HadCM3 simulator output and the independent component reconstructions with $l = 6$ independent components at times $t = 4000, 5000$.

the root mean-squared errors from the independent component reconstructions are still relatively small. The main interest in the rest of this section is on the differences between the forecasted independent component reconstructions and the independent component reconstructions.

4.6.3.2 Reconstruction errors using the mixed forecasting approach

In this section we consider forecasted independent component reconstructions when the mixed forecasting approach is used to obtain one-step-ahead forecasts for the first $l = 6$ independent components (as seen in Section 4.5.3.1). The forecasted independent component reconstructions using this approach at times $t = 4000$ and $t = 5000$ are shown in Figure 4.40.

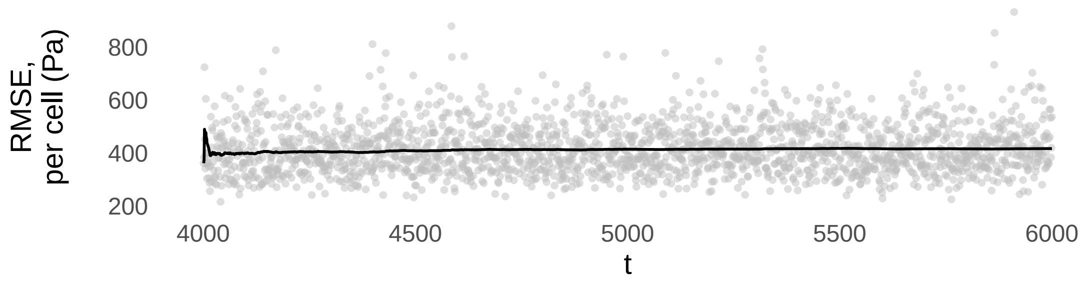
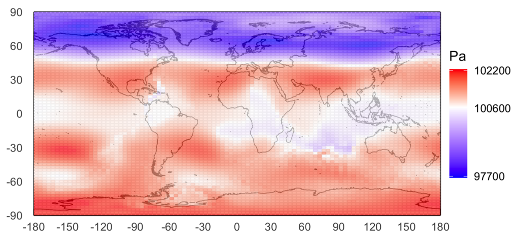


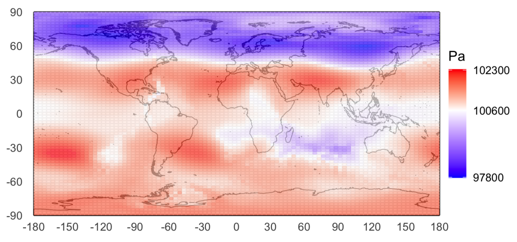
Figure 4.39: The grey circles give the root mean-squared error per grid cell between the true HadCM3 simulator output and the independent component reconstructions with $l = 6$ independent components (calculated using (4.7)). The black line gives the rolling average (4.12).

The differences between the forecasted independent component reconstruction and: *(i)* the true mean sea-level air pressure HadCM3 simulator output, and; *(ii)* the independent component reconstruction (using $l = 6$ independent components), are shown in Figure 4.41 and Figure 4.42 respectively.

One initial remark that we make here is that the differences between the forecasted independent component reconstructions and the independent component reconstructions (shown in Figure 4.42) seem to be lower than the equivalent principal component differences (shown in Figure 4.35, Section 4.6.2.2). This suggests that the white-noise, Ornstein-Uhlenbeck models could be a better fit for the seasonally adjusted independent components than for the seasonally adjusted principal components.

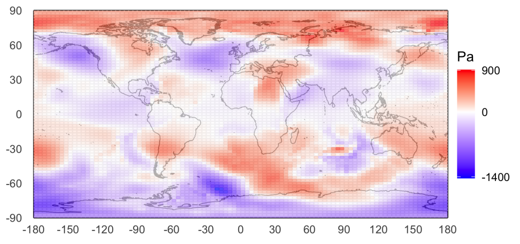


(a) Time, $t = 5000$.

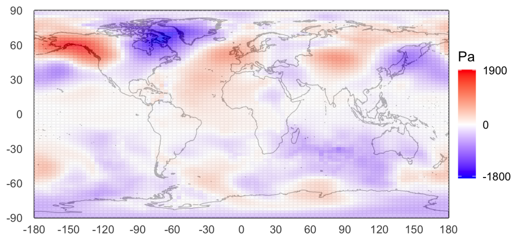


(b) Time, $t = 5000$.

Figure 4.40: Forecasted independent component reconstructions using the mixed forecasting approach for the first $l = 6$ independent components, at times $t = 4000, 5000$.

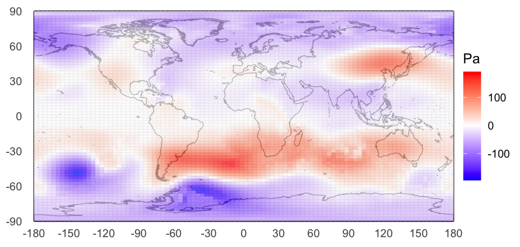


(a) Time, $t = 4000$.

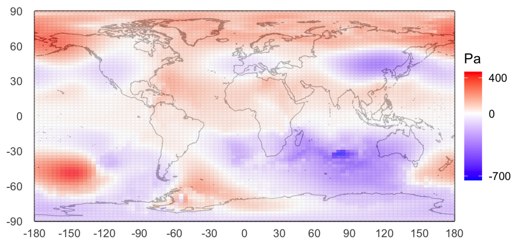


(b) Time, $t = 5000$.

Figure 4.41: Difference between the true mean sea-level air pressure HadCM3 simulator output and the forecasted independent component reconstructed using the mixed forecasting approach, at times $t = 4000, 5000$.



(a) Time, $t = 4000$.

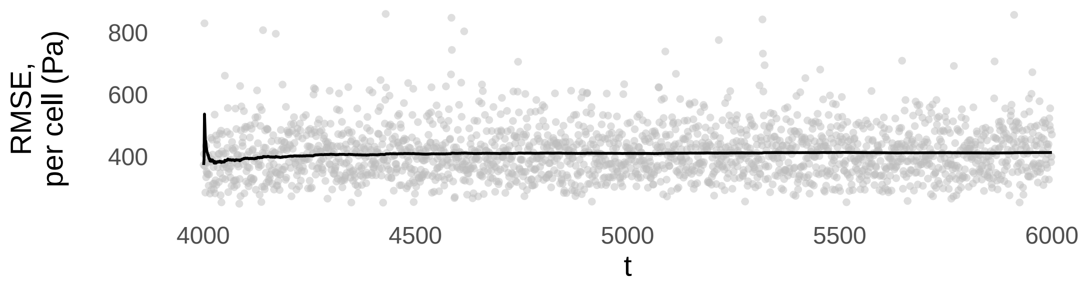


(b) Time, $t = 5000$.

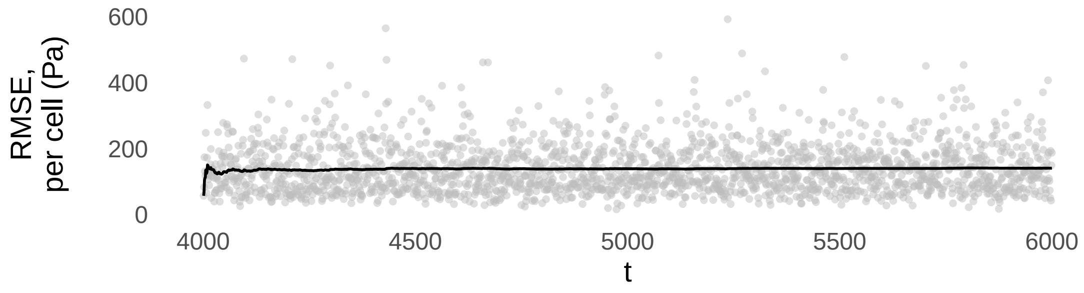
Figure 4.42: Difference between the independent component reconstructions, and the forecasted independent component reconstructions using the mixed forecasting approach, at times $t = 4000, 5000$.

As before, we assess the quality of the forecasted independent component reconstructions by considering the root mean-squared errors. The grey circles in Figures 4.43a and 4.43b are the root mean-squared errors between the forecasted independent component reconstructions using the mixed forecasting approach, and: *(i)* the true mean sea-level air pressure HadCM3 output, and; *(ii)* the independent component reconstruction, respectively. The black lines give the respective root mean-squared error rolling average as in (4.12).

In this case, the root mean-squared errors between the forecasted independent component reconstructions and the independent component reconstructions (shown in Figure 4.43b) are lower than the equivalent forecasted principal component reconstruction root mean-squared errors (shown in Figure 4.36b). This strengthens the initial judgement – originally made when examining Figure 4.42 – that the mixed forecasting model is a better fit for the seasonally adjusted independent components than the seasonally adjusted principal components. We will comment on this better fit in more detail in the conclusion, Section 4.7.



(a) Root mean-squared error per grid cell between the forecasted independent component reconstructions using the mixed forecasting approach, and the true HadCM3 simulator output.



(b) Root mean-squared errors per grid cell between the forecasted independent component reconstructions using the mixed forecasting approach, and the independent component reconstructions (using $l = 6$ independent components).

Figure 4.43: The grey circles give the root mean-squared errors per grid cell between the forecasted independent component reconstructions using the mixed forecasting approach, and: *(i)* the true HadCM3 simulator output (Figure 4.43a); *(ii)* the independent components reconstructions with $l = 6$ independent components (Figure 4.43b). The black lines give the rolling average of the root mean-squared errors over time.

4.6.3.3 Reconstruction errors using the block-average Ornstein-Uhlenbeck forecasting approach

In this section we analyse the forecasted independent component reconstructions when the block-average Ornstein-Uhlenbeck forecasting approach is used to obtain one-step-ahead forecasts for the first $l = 6$ independent components. These independent component forecasts were introduced in Section 4.5.3.2. The reconstructions found in this section are similar to those found when the mixed forecasting approach was used (discussed in Section 4.6.3.2), and therefore all figures here have been placed in Appendix A.2. The relevant figures here are:

- The forecasted independent component reconstructions at times $t = 4000$ and $t = 5000$, shown in Figure A.9;
- The differences between the HadCM3 simulator output and the forecasted independent component reconstructions at times $t = 4000$ and $t = 5000$, shown in Figure A.10;
- The difference between the independent component reconstructions using $l = 6$ independent components, and the forecasted independent component reconstructions independent component analysis using the block-average Ornstein-Uhlenbeck forecasting approach at times $t = 4000$ and $t = 5000$, shown in Figure A.11.
- The root mean-squared errors between the forecasted independent component reconstructions using the block-average Ornstein-Uhlenbeck forecasting approach for $l = 6$ independent components and; *(i)* the true mean sea-level air pressure HadCM3 output, shown in Figure A.12a; *(ii)* the independent component reconstructions with $l = 6$ independent components, shown in Figure A.12b.

As with the forecasted independent component reconstruction when the mixed forecasting approach was used (Section 4.6.3.2), the difference between the forecasted independent component reconstructions using the block-average Ornstein-Uhlenbeck forecasting approach and the (true) independent component reconstructions is smaller than the equivalent principal component reconstructions difference (shown in Figure A.7). A similarity with the principal component reconstructions here is that the choice of using the block-average Ornstein-Uhlenbeck forecasting approach or the mixed forecasting approach seems to make little overall difference to the forecasted independent component reconstructions.

To conclude this section we note that the root mean-squared errors found using the block-average Ornstein-Uhlenbeck forecasting approach are comparable to the errors found using the mixed forecasting approach. Therefore, similarly to the forecasted principal component reconstructions, it appears that here, misspecifying the independent components as piecewise observations and using the mixed forecasting method does not seem to be any worse than correctly specifying them as block-average observations and modelling them as realisations from block-average Ornstein-Uhlenbeck processes. It is also worth noting that the root mean-squared errors of the difference between the forecasted independent component reconstructions and the true HadCM3 output are on average the same as those between the true independent component reconstructions and the HadCM3 output. This implies that using the forecasted independent components instead of the true components does not affect the total reconstruction error.

4.7 Concluding remarks

We now highlight some of the main conclusions arising from the analysis in this chapter. First, we will discuss the stochastic modelling of the components (as seen in Section 4.4), followed by the component forecasts (as in Section 4.5), and finally the forecasted reconstructions (as in Section 4.6).

One key point that was made evident during the stochastic modelling was that the first six principal and six independent components contained strong seasonality, which was clearly seen from the autocorrelation functions (given in Figures 4.8 and 4.17 respectively). This was expected for the principal components, as it is well known that principal component analysis often results in components that emphasize seasonality (as seasonal behaviour is often the cause of large variation in data). For the independent components, this is less expected. However, as discussed in this chapter (and in Chapter 1), one motivation for using independent component analysis within a climate setting is to try to obtain fundamental Earth system processes. These fundamental processes are likely to contain some seasonality.

A second point that we make here in relation to the stochastic modelling part of this chapter, is that both of the pointwise Ornstein-Uhlenbeck parameter estimation methods were deficient in some way. The sparse maximum likelihood estimation method resulted in some spurious solutions, and thus this method was only applied to the principal components (seen in Figure 4.12). On the other hand, the requirement of the method of moments on having a positive lag-1 sample correlation meant that for many of the training samples of the principal and independent components, it was impossible to obtain the drift and diffusion estimates (as seen in Figure 4.15 and Figure 4.23 for the principal and independent components, respectively).

We saw in the forecasting section (Section 4.5) that for both the principal and independent component forecasts, the use of the mixed forecasting approach or the block-average Ornstein-Uhlenbeck forecasting approach gave very similar results. This was shown visually, and by the proportion of the test samples that lay outside of the 95% prediction intervals. As the mixed forecasting approach misspecified the components as pointwise realisations (as opposed to averaged over the seasons), we would have expected, *a priori*, that the block-average Ornstein-Uhlenbeck forecasting approach would give better quality forecasts (as this was shown in Example 2.3.10). This however, was not the case, and further analysis on these forecasts is needed here to find a reason for this.

For some of the components, the prediction intervals (found using both forecasting approaches) were much wider than expected, such that only a very low proportion of the test samples were outside of these intervals. This was the case for the first principal component, and for the second and third independent component. This suggest that either the model chosen for these components was incorrect, or that the parameter estimation methods converged to a local maximum and not a global one. However, the latter reason seems to be less likely than the former, as the behaviour of the prediction intervals was consistent across the two forecasting methods (which used different underlying models, and different parameter estimation methods). We therefore judge that the models used here for these components are suboptimal, and further work is required to find more suitable models.

The main conclusion arising from the reconstruction section (Section 4.6) is that the stochastic models and associated forecasts used here seem to be more appropriate when applied to the independent components than to the principal components. This can be seen by comparing the reconstructions found using the one-step-ahead point forecasts to those found using the test samples. That is, in general the difference between the forecasted independent component reconstructions and the independent component reconstruction (measured using root mean-squared error) seems to be smaller than the equivalent difference between the forecasted principal component reconstructions and the principal component reconstructions. This can be seen for the mixed forecasting method by comparing Figure 4.36b to Figure 4.43b, and for the block-average Ornstein-Uhlenbeck forecasting method by comparing Figure A.8b to Figure A.12b. This is slightly surprising, as these independent components are chosen for their non-Gaussianity, and therefore modelling them using processes with Gaussian transition densities seems counter-intuitive. One explanation for this could be that the non-Gaussianity is present within the seasonal effects of the independent components, such that the seasonally adjusted independent components can be successfully modelled using Gaussian processes. This would be something to consider in any future work.

It can also be seen that the difference between the HadCM3 simulator output and the independent component reconstructions are larger than between the HadCM3 simulator output and the principal component reconstructions. This is what we would in general expect, as by the maximal variance property of principal components we theoretically obtain reconstructions that minimise root mean-squared error (as discussed in Remark 4.18).

However, for the independent components, the root mean-squared errors between the HadCM3 simulator output and the forecasted independent component reconstructions are very similar to the errors between the HadCM3 simulator output and the independent component reconstructions. That is, here the overall error from using the forecasts for the independent components in place of the true components is small. This can be seen by comparing Figure 4.39 to Figure 4.43a (using the mixed forecasting approach) or to Figure A.12a (using the block-average Ornstein-Uhlenbeck forecasting approach). For the principal components, there is a marked difference between using the forecasted components and using the true components, when comparing the reconstructions to the true HadCM3 simulator output. This can be seen by comparing Figure 4.32 to Figure 4.36a (using the mixed forecasting approach) or Figure A.8a (using the block-average Ornstein-Uhlenbeck forecasting approach). We therefore suggest that the reconstructions found using independent components can be replaced by the forecasts with little overall loss of error.

Chapter 5

Dimension Reduction, Forecasting and Reconstruction: Mean Sea-Level Pressure With Two-Dimensional Wind Climate Simulation Output

This chapter follows the same structure as Chapter 4. However, here we perform analysis on a HadCM3 simulator run with output representing the mean sea-level air pressure with horizontal (two dimensional) wind velocity over a discretised version of the Earth's surface. The mean sea-level air pressure output is given on a 96×73 grid (as in Chapter 4), and the wind velocity output is given on a 96×72 grid. Then (similarly to the mapping given by (4.1) in Chapter 4) the simulator for which the output here is obtained from can be represented by the mapping

$$f_2 \colon \mathcal{M} \to \mathbb{R}^{(96\times73)+2\times(96\times72)}. \tag{5.1}$$

Here, $(96 \times 73) + 2 \times (96 \times 72) = 20832$, and each dimension in this mapping $f_2(m)$, $m \in \mathcal{M}$, represents the value of one of the output parameters on one grid cell. The size of the grid cell is dependent on whether the output is mean sea-level air pressure or wind velocity, and we discussed how the area of the corresponding cell can be found in Section 4.2.1. In this chapter the HadCM3 simulator output is represented by the matrix $X \in \mathbb{R}^{n\times p}$, where $p = 20832$ and $n = 2400$.

5.1 Applying Principal Component Analysis to the Mean Sea-Level Air Pressure with Wind Velocity HadCM3 Simulator Output

In this section, principal component analysis (given in Section 1.2) is applied to output from the HadCM3 simulator as represented by the mapping f_2 in (5.1). As seen previously, we need to account for the size of the grid cells that represent the Earth's surface.

Let the output here be given by the matrix $X \in \mathbb{R}^{n\times p}$, with $n = 2400$ and $p = (96 \times 73) + 2 \times (96 \times 72) = 20832$, such that the i^{th} row of X is given by $f_2(m_i)$ for some $m_i \in \mathcal{M}$, $i = 1, 2, \ldots, 2400$. We scale the columns of the matrix X to give scaled matrix $X^{(g)} = XG$, where $G = \text{diag}(\sqrt{c_i}) \in \mathbb{R}^{p\times p}$ and each c_i, $i = 1, \ldots, p$ gives the area of each grid cell (as introduced in Section 4.2.1.

Now, we perform principal component analysis on the scaled matrix by centring the columns of $X^{(g)}$ and applying singular value decomposition (as described in Panel 1.3). With $\tilde{X}^{(g)}$ the column-centred version of $X^{(g)}$, we obtain singular value decomposition $\tilde{X}^{(g)} = U\Lambda V^\top$, with $\Lambda = \text{diag}(\lambda_i) \in \mathbb{R}^{r\times r}$, $V \in \mathbb{R}^{p\times r}$ where $r = \min\{n-1, p\} = 2399$ is the rank of $\tilde{X}^{(g)}$.

Following the notational convention throughout this thesis, the principal directions are given by $w_i := v_i$, $i = 1, 2, \ldots, r$, where $v_i \in \mathbb{R}^p$ is the i^{th} column of the matrix $V \in \mathbb{R}^{p\times r}$. The principal components are given by the projection of the data along the respective principal directions, $M = (m_1\, m_2 \ldots m_r) = \tilde{X}^{(g)}V$, with the i^{th} principal component, $m_i \in \mathbb{R}^n$, having variance $\gamma_i = \lambda_i^2/(n-1)$.

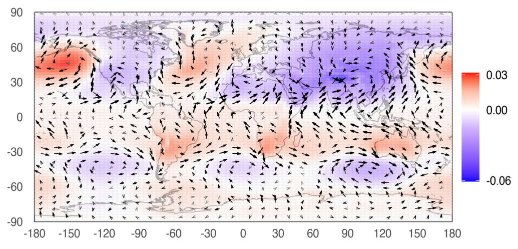
The first six principal directions $w_1, \dots, w_6 \in \mathbb{R}^p$, $p = 20832$, are shown in Figure 5.1. In this case, the mean sea-level air pressure part of each principal component is represented by the blue-red gradient (exactly as before in Chapter 4). The arrows represent the two-dimensional wind velocity part of each principal component, with each arrow giving the direction and a representation of the magnitude of the velocity at the base of the arrow. The magnitude of the wind is represented by the arrow changing colour from grey to black, changing length and changing thickness, according to the plot in Figure 5.2.

We calculate the principal components $m_1, \dots, m_r \in \mathbb{R}^n$ by projecting the scaled and centred matrix $\tilde{X}^{(g)}$ along the principal directions $w_1, \dots, w_r$. Throughout this chapter we model the principal components as causal time-series over the times $t = 0, 1, \dots, n-1$, and the elements of each principal component are given by $m_{i\,0}$, $m_{i\,1}, \dots, m_{i\,2399}$, $i = 1, \dots, r$, where $r = 2399$ is the rank of the $\tilde{X}^{(g)}$. The first six principal components are shown in Figure 5.3 for times $t = 0, 1, \dots, 239$.

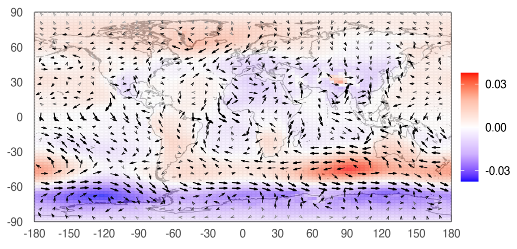
The percentage of the total variance that is accounted for by each of the first twenty principal components is given by $100 \cdot \gamma_i / \sum_{j=1}^{r} \gamma_j$, $i = 1, \dots, 20$, $r = 2399$, and is shown by the bar chart in Figure 5.4. The black line gives the cumulative sum of the variance, $100 \cdot \sum_{k=1}^{i} \gamma_k / \sum_{j=1}^{r} \gamma_i$, $k = 1, \dots, 20$. The first six principal components account for 73.50% of the total variance of the data, and the first twenty principal components account for 87.98% of the total variance.

5.2 Applying Independent Component Analysis to the Mean Sea-Level Air Pressure with Wind Velocity HadCM3 Simulator Output

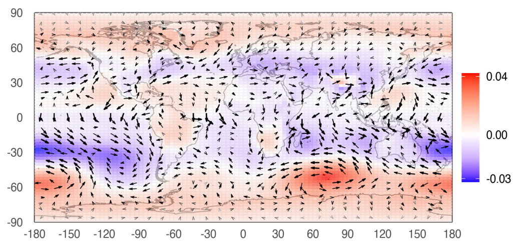
In this section, we apply independent component analysis (introduced in Section 1.3) to the mean sea-level air pressure with wind velocity output obtained from the HadCM3 simulator. As discussed in the previous chapter, we can apply independent component analysis without the need to scale the output with respect to the grid cell area. We perform independent component analysis using the fastICA method (in the same way as in Section 4.3) on the matrix $X \in \mathbb{R}^{n \times p}$. In this



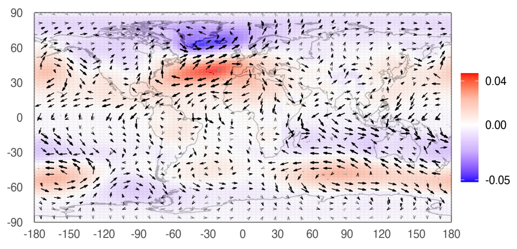
(a) First principal direction, $w_1 \in \mathbb{R}^p$.



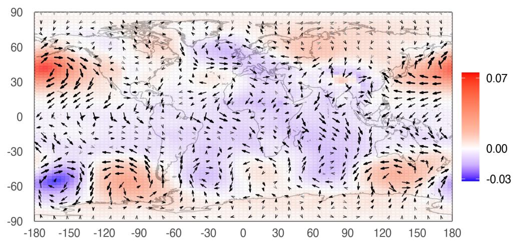
(b) Second principal direction, $w_2 \in \mathbb{R}^p$.



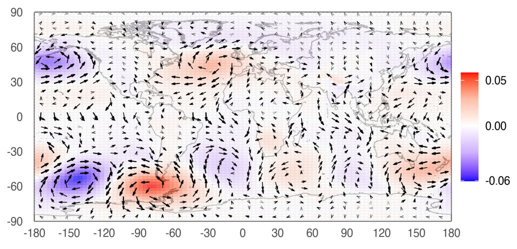
(c) Third principal direction, $w_3 \in \mathbb{R}^p$.



(d) Fourth principal direction, $w_4 \in \mathbb{R}^p$.



(e) Fifth principal direction, $w_5 \in \mathbb{R}^p$.



(f) Sixth principal direction, $w_6 \in \mathbb{R}^p$.

Figure 5.1: First six principal directions found by performing principal component analysis on the mean sea-level air pressure with wind velocity output (scaled according to grid cell area) from the HadCM3 simulator. That is, the plots show $w_1, \ldots, w_6 \in \mathbb{R}^p$, $p = 20832$, mapped onto the corresponding grid cells. Here, the mean sea-level air pressure output is given by the red-blue gradient, with the arrows representing the wind velocity, following the behaviour as in Figure 5.2.

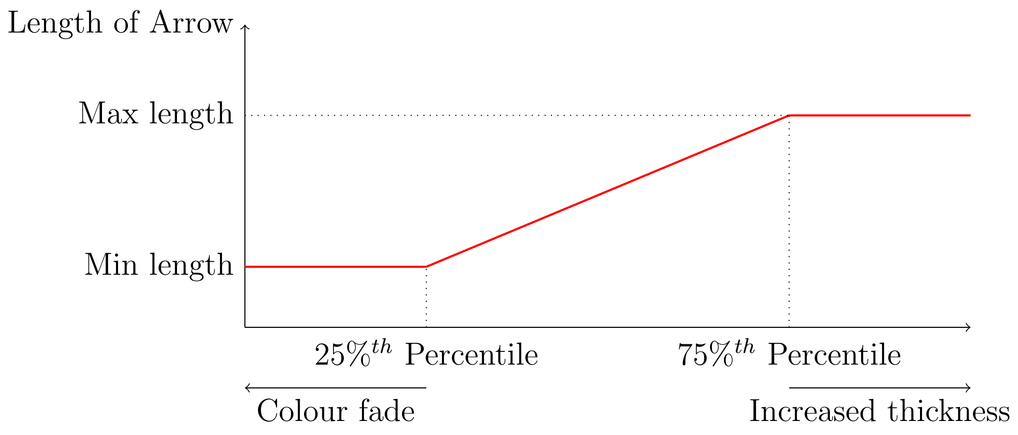
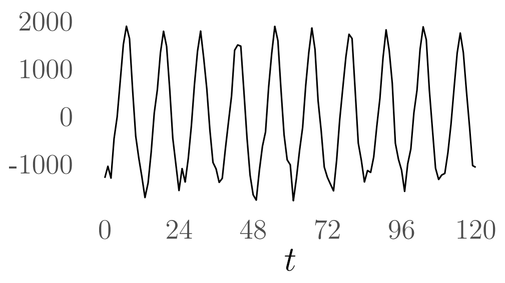


Figure 5.2: Function to determine length, thickness and colour of arrows in Figure 5.1.

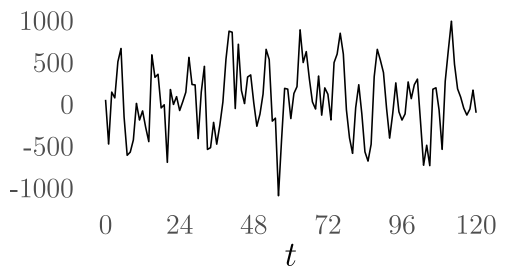
case, we apply the PCA-whitening step using $\tilde{r} = 148$ principal components (such that $r = 148$ in Step 2. of Panel 1.4) These 148 principal components account for more than 99% of the variance of the total output.

Each independent direction, w_i, $i = 1, \ldots, \tilde{r}$, with $\tilde{r} = 148$ as discussed above, is the i^{th} column of the matrix $W = AQ \in \mathbb{R}^{p \times \tilde{r}}$, with $A \in \mathbb{R}^{p \times r}$ the whitening matrix, and $Q \in \mathbb{R}^{\tilde{r} \times \tilde{r}}$ the orthogonal direction matrix as described in Panel 1.4. The first six independent directions, $w_1, \ldots, w_6 \in \mathbb{R}^p$, $p = 20832$, are shown in Figure 5.5.

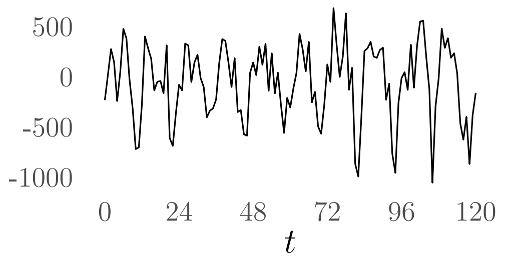
The independent components are given by the columns of $Z \in \mathbb{R}^{n \times \tilde{r}}$, $n = 2400$, where $Z = \tilde{X}W$ and $W = AQ \in \mathbb{R}^{p \times \tilde{r}}$ is the matrix with independent directions as the columns. The first six independent components are shown in Figure 5.6, for time steps $t = 0, \ldots, 119$. The M-spacing entropy approximations (described in Section 1.3.1.3) for the independent components here are given in the respective captions in Figure 5.6, with $M = \lfloor \sqrt{2400} \rfloor$.



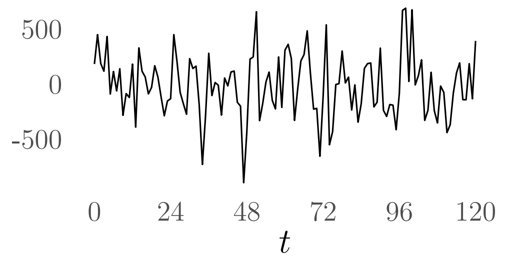
(a) First principal component for time steps $t = 0, 1, \ldots, 119$.



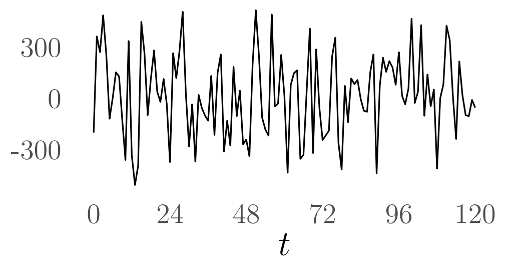
(b) Second principal component for time steps $t = 0, 1, \ldots, 119$.



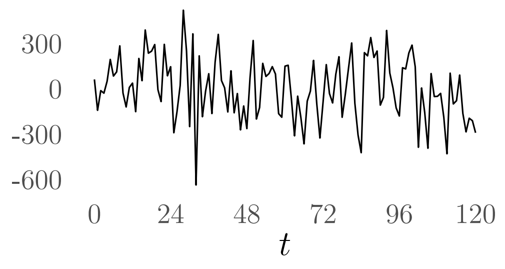
(c) Third principal component for time steps $t = 0, 1, \ldots, 119$.



(d) Fourth principal component for time steps $t = 0, 1, \ldots, 119$.



(e) Fifth principal component for time steps $t = 0, 1, \ldots, 119$.



(f) Sixth principal component for time steps $t = 0, 1, \ldots, 119$.

Figure 5.3: First six principal components for time steps $t = 0, 1, \ldots, 119$, such that the first 120 elements of each vector $m_1, \ldots, m_6 \in \mathbb{R}^{2400}$ is shown. The vectors $m_1, \ldots, m_6$ are found by projecting the weighted (and column-centred) mean sea-level air pressure with wind velocity HadCM3 simulator output, $\tilde{X}^{(g)} \in \mathbb{R}^{2400 \times 20832}$, along the first six principal directions, $w_1, \ldots, w_6 \in \mathbb{R}^{20832}$ (shown in Figure 5.1).

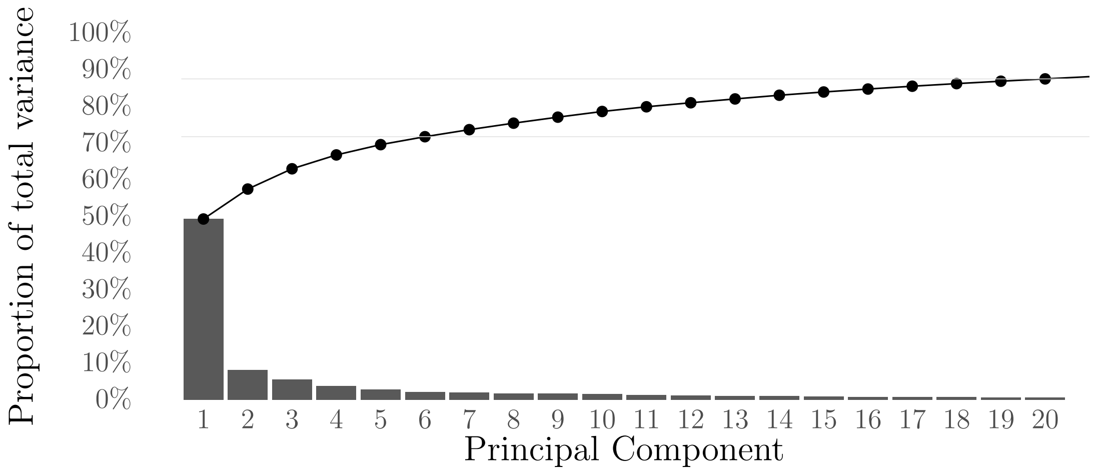


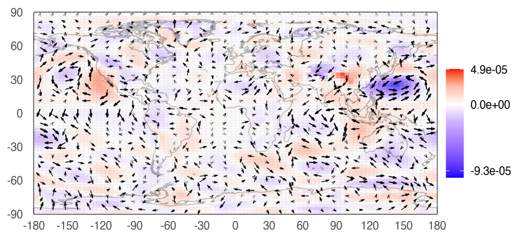
Figure 5.4: Bar chart showing the proportion of total variance accounted for by each principal component, found by applying principal component analysis on the mean sea-level air pressure with wind velocity HadCM3 simulator output. Each bar shows the value $100 \cdot \gamma_i / \sum_{j=1}^{r} \gamma_j$, $i = 1, \ldots, 20$. The cumulative sum of these proportions is shown by the black points and line, such that each point gives the value $100 \cdot \sum_{k=1}^{i} \gamma_k / \sum_{j=1}^{r} \gamma_j$ for $i = 1, \ldots, 20$. The grey horizontal lines show the cumulative proportion of total variance accounted for by the first six, and the first twenty principal components. These account for 73.50% and 87.98% of the total variance respectively.

5.3 Modelling Principal and Independent Components by Stochastic Processes

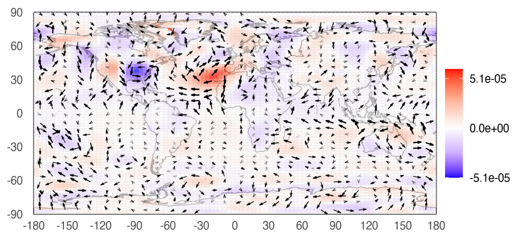
In this section, we model the components obtained by principal component analysis and independent component analysis applied to the mean sea-level air pressure with wind velocity output from the HadCM3 simulator, using techniques introduced in Chapter 2. This is shown as Step 2. in Figure 3.8. This section mirrors Section 4.4.

Each principal and independent component considered throughout this chapter belong to $\mathbb{R}^n$ with $n = 2400$, and each is modelled as an causal stochastic process with time steps $t = 0, 1, 2, \ldots, (n-1)$ such that $\Delta t = 1$ constant.

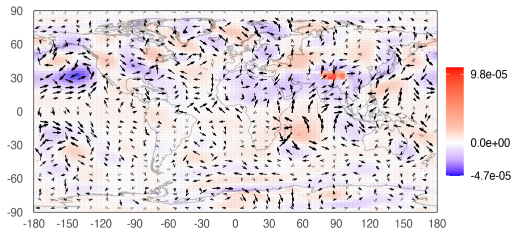
As in the previous chapter, here we split the components into a "training"



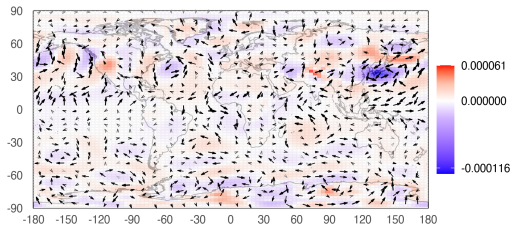
(a) First independent direction, $w_1 \in \mathbb{R}^{20832}$.



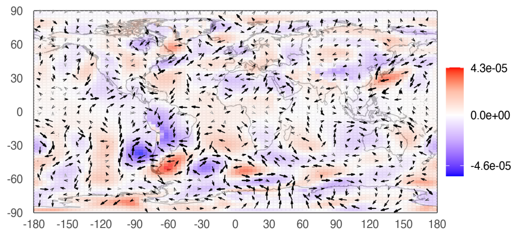
(b) Second independent direction, $w_2 \in \mathbb{R}^{20832}$.



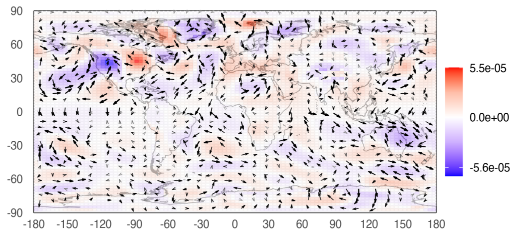
(c) Third independent direction, $w_3 \in \mathbb{R}^{20832}$.



(d) Fourth independent direction, $w_4 \in \mathbb{R}^{20832}$.

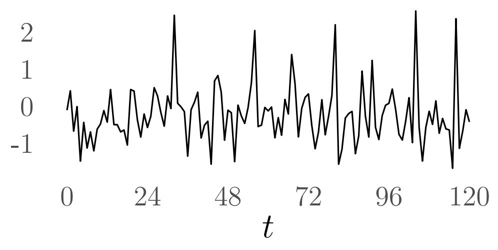


(e) Fifth independent direction, $w_5 \in \mathbb{R}^{20832}$.

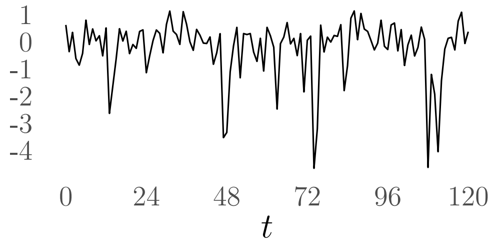


(f) Sixth independent direction, $w_6 \in \mathbb{R}^{20832}$.

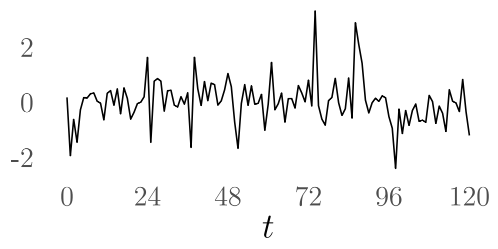
Figure 5.5: First six independent directions, $w_1, \ldots, w_6 \in \mathbb{R}^{20832}$, from the HadCM3 simulator with mean sea-level air pressure with wind velocity output, mapped onto the corresponding grid cells. Here, the mean sea-level air pressure output is given by the red-blue gradient, with the arrows representing the wind velocity, following the behaviour as in Figure 5.2.



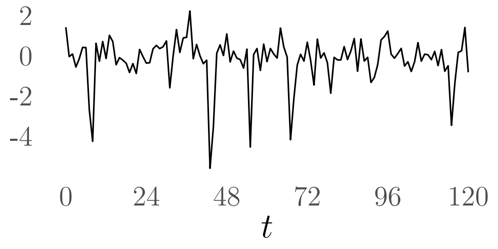
(a) First independent component, $s_1 \in \mathbb{R}^{20832}$, for time steps $t = 0, 1, \ldots, 119$. The M-spacing entropy of this independent component is 1.043.



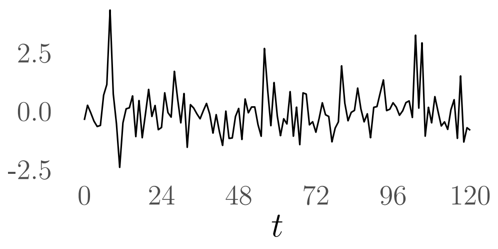
(b) Second independent component, $s_2 \in \mathbb{R}^{20832}$, for time steps $t = 0, 1, \ldots, 119$. The M-spacing entropy of this independent component is 1.046.



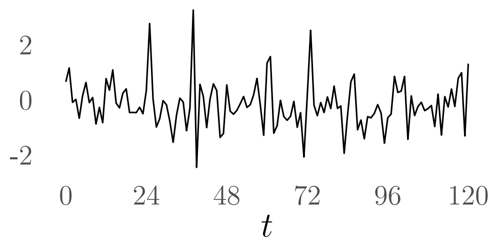
(c) Third independent component, $s_3 \in \mathbb{R}^{20832}$, for time steps $t = 0, 1, \ldots, 119$. The M-spacing entropy of this independent component is 1.106.



(d) Fourth independent component, $s_4 \in \mathbb{R}^{20832}$, for time steps $t = 0, 1, \ldots, 119$. The M-spacing entropy of this independent component is 1.107.



(e) Fifth independent component, $s_5 \in \mathbb{R}^{20832}$, for time steps $t = 0, 1, \ldots, 119$. The M-spacing entropy of this independent component is 1.132.



(f) Sixth independent component, $s_6 \in \mathbb{R}^{20832}$, for time steps $t = 0, 1, \ldots, 119$. The M-spacing entropy of this independent component is 1.156.

Figure 5.6: First six independent components, $s_1, \ldots, s_6 \in \mathbb{R}^{20832}$, found by projecting the (column-centred) mean sea-level air pressure and two-dimensional wind HadCM3 simulator output, $\tilde{X} \in \mathbb{R}^{2400 \times 20832}$, along the first six independent directions, $w_1, \ldots, w_6 \in \mathbb{R}^{20832}$, as shown in Figure 5.5. Here, the independent components are shown as a time-series for times $t = 0, \ldots, 119$. The M-spacing entropy is calculated for each independent component as in Section 1.3.1.3, with $M = \lfloor \sqrt{n} \rfloor = \lfloor \sqrt{2400} \rfloor$.

set and "test" sample. With $x = (x_0, x_1, \ldots, x_{n-1})$ some observation, then the training set is given by $\{x^{(\tilde{n})}, x^{(\tilde{n}+1)}, \ldots, x^{(n-1)}\}$ with each element the training sample $x^{(j)} = (x_0, \ldots, x_{j-1}) \in \mathbb{R}^j$, $j = \tilde{n}, \ldots, n-1$. The test sample here is $(x_{\tilde{n}}, x_{\tilde{n}+1}, \ldots, x_n) \in \mathbb{R}^{n-\tilde{n}}$. In this chapter we have $n = 2400$ and set $\tilde{n} = 1600$.

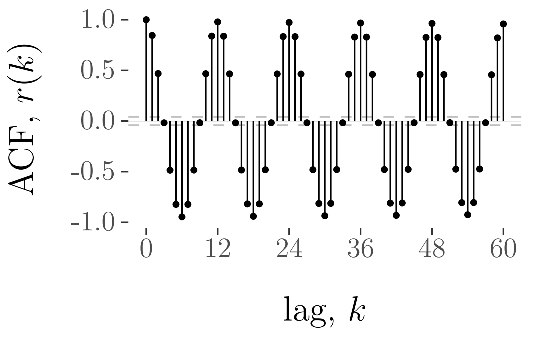
We consider the principal components obtained from the mean sea-level air pressure with wind velocity HadCM3 simulator output in Section 5.3.1, and the respective independent components in Section 5.3.2, after removing any seasonality from the components. As in the previous chapter, the stochastic processes used to model the seasonally adjusted principal and independent components considered here are the ARIMA, Ornstein-Uhlenbeck and block-average Ornstein-Uhlenbeck processes.

5.3.1 Modelling the principal components

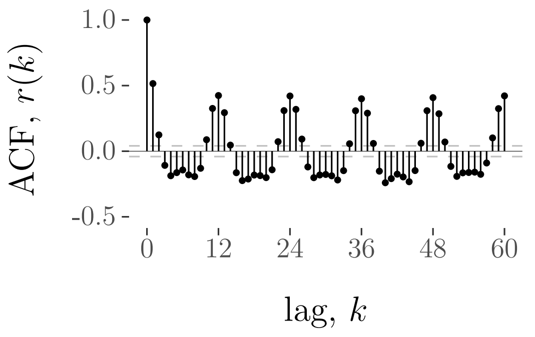
First, we examine the correlograms for the principal components to determine whether seasonality (in the form of (4.4)) needs to be removed. The correlograms for the first six principal components, $m_1, \ldots, m_6$, are shown in Figure 5.7, which shows that seasonality is present. Removing this seasonality gives the set of seasonally adjusted principal components, $\tilde{m}_i \in \mathbb{R}^n$, $i = 1, \ldots, 6$. Figure 5.8 shows the first three principal components on the left-hand side (black line) and the associated periodic quadratic mean function (grey line). The right-hand side shows the respective seasonally adjusted principal components.

In this section we find parameter estimates for the stochastic models using the training sets from the first $l = 6$ seasonally adjusted principal components, $\tilde{m}_1, \ldots, \tilde{m}_6$, where $\tilde{m}_i = (\tilde{m}_{i\,0}, \tilde{m}_{i\,1}, \ldots, \tilde{m}_{i\,n-1}) \in \mathbb{R}^n$, $i = 1, \ldots, 6$ and $n = 2400$. The training set for the seasonally adjusted principal component $\tilde{m}_i$ is given by $\{\tilde{m}_i^{(1600)}, \tilde{m}_i^{(1601)}, \ldots, \tilde{m}_i^{(2399)}\}$, with each training sample given in vector form by $\tilde{m}_i^{(j)} = (\tilde{m}_{i\,0}, \tilde{m}_{i\,1}, \ldots, \tilde{m}_{i\,j-1}) \in \mathbb{R}^j$.

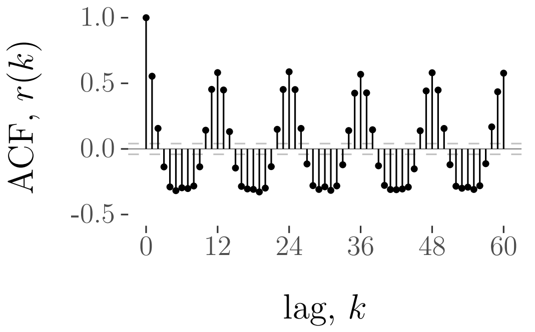
We now model the seasonally adjusted principal components as realisations from ARIMA processes, (pointwise) Ornstein-Uhlenbeck processes and block-average Ornstein-Uhlenbeck processes in Sections 5.3.1.1, 5.3.1.2 and 5.3.1.3 respectively.



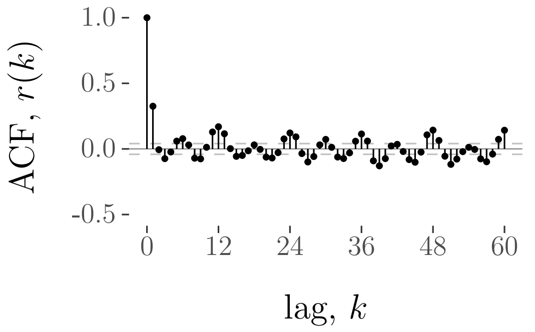
(a) Autocorrelation function (correlogram) for the first principal component, m_1, for lags $k = 0, 1, \ldots, 60$.



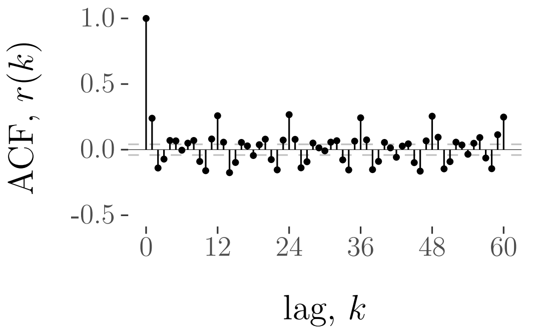
(b) Autocorrelation function (correlogram) for the second principal component, m_2, for lags $k = 0, 1, \ldots, 60$.



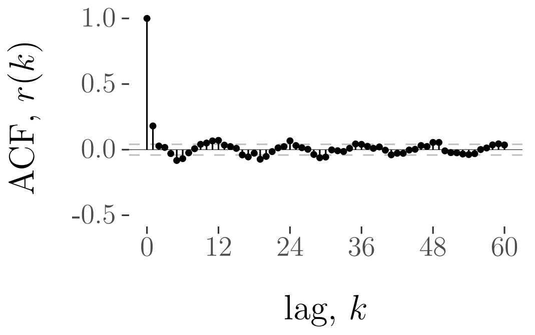
(c) Autocorrelation function (correlogram) for the third principal component, m_3, for lags $k = 0, 1, \ldots, 60$.



(d) Autocorrelation function (correlogram) for the fourth principal component, m_4, for lags $k = 0, 1, \ldots, 60$.

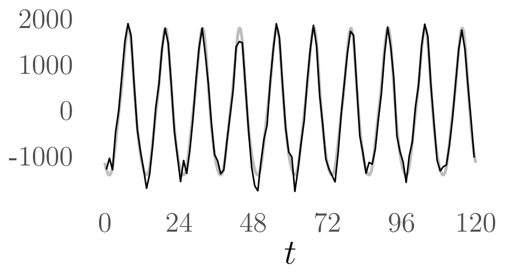


(e) Autocorrelation function (correlogram) for the fifth principal component, m_5, for lags $k = 0, 1, \ldots, 60$.

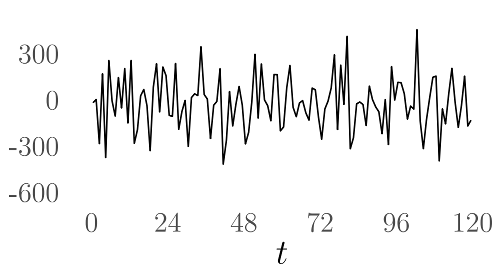


(f) Autocorrelation function (correlogram) for the sixth principal component, m_6, for lags $k = 0, 1, \ldots, 60$.

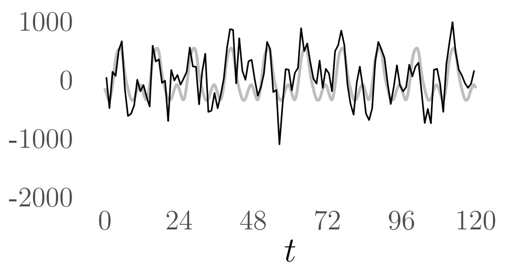
Figure 5.7: Autocorrelation functions (correlogram) for the first six principal component, $m_1, \ldots, m_6$, obtained from the HadCM3 simulator, mean sea-level pressure with wind velocity output, for lags $k = 0, 1, \ldots, 60$.



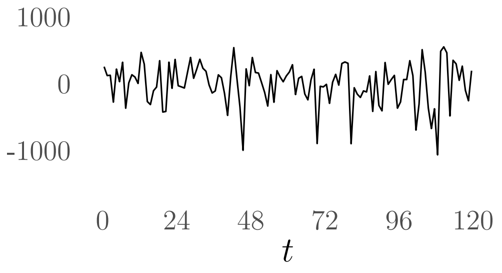
(a) First 120 points of the first principal component, $m_1 \in \mathbb{R}^{2400}$, with the grey line giving the associated quadratic spline calculated from the monthly means.



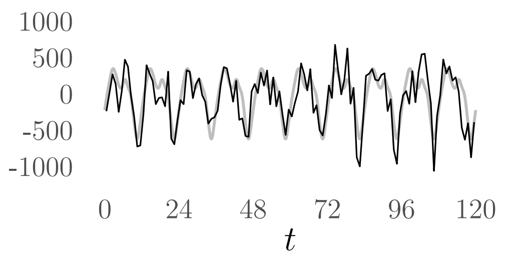
(b) First 120 points of the first seasonally adjusted principal component, $\tilde{m}_1 \in \mathbb{R}^{2400}$.



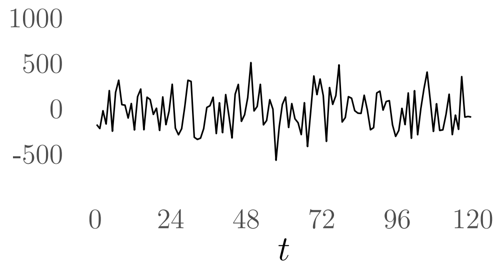
(c) First 120 points of the second principal component, $m_2 \in \mathbb{R}^{2400}$, with the grey line giving the associated quadratic spline calculated from the monthly means.



(d) First 120 points of the second seasonally adjusted principal component, $\tilde{m}_2 \in \mathbb{R}^{2400}$.



(e) First 120 points of the third principal component, $m_3 \in \mathbb{R}^{2400}$, with the grey line giving the associated quadratic spline calculated from the monthly means.



(f) First 120 points of the third seasonally adjusted principal component, $\tilde{m}_3 \in \mathbb{R}^{2400}$.

Figure 5.8: First six principal components, $m_1, \ldots, m_6 \in \mathbb{R}^{6000}$, found by projecting the weighted (and column-centred) HadCM3 simulator mean sea-level air pressure with wind velocity output, $\tilde{X}^{(g)} \in \mathbb{R}^{2400 \times 20832}$, along the first six principal directions, $w_1, \ldots, w_6 \in \mathbb{R}^{20832}$ (shown in Figure 5.1).

5.3.1.1 Modelling the principal components as observations from an ARIMA processes

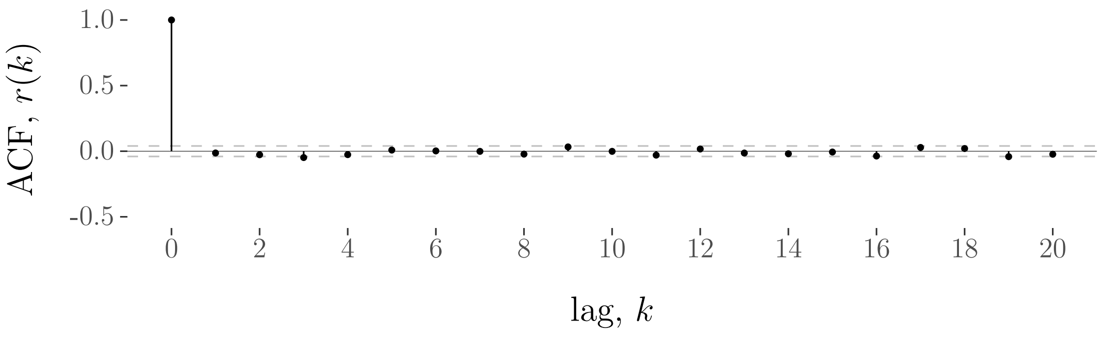
Here we assume the seasonally adjusted principal components are realisations from ARIMA processes. The Box-Jenkins method is used to estimate the model structure, and then parameter estimates are obtained (following the same procedure as in Section 4.4.2.1).

We apply this procedure to the first six seasonally adjusted principal components. For each seasonally adjusted principal component $\tilde{m}_i$, $i = 1, \ldots, 6$, we estimate the model structure $\hat{p}_i$, $\hat{d}_i$ and $\hat{q}_i$. Then we use the associated training samples $\tilde{m}_i^{(j)} = (\tilde{m}_{i\,0}, \tilde{m}_{i\,1}, \ldots, \tilde{m}_{i\,(j-1)}) \in \mathbb{R}^j$, to obtain the parameter estimates $\hat{\varphi}_{i,1}(j), \ldots, \hat{\varphi}_{i,p_i}(j)$ and $\hat{\vartheta}_{i,1}(j), \ldots, \hat{\vartheta}_{i,q_i}(j)$, $j = 1600, 1601, \ldots, 2399$. We describe the Box-Jenkins method for the first seasonally adjusted principal component in detail below. We then briefly discuss the remaining 5 seasonally adjusted principal components.

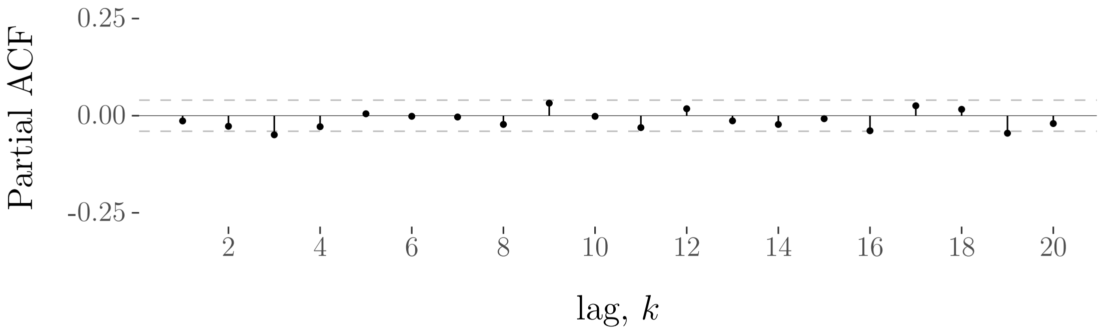
Box-Jenkins method on the first seasonally adjusted principal component. Here we estimate the model structure p_1, d_1 and q_1 using the sample autocorrelation and partial autocorrelation functions of the seasonally adjusted principal component, $\tilde{m}_1$. The sample autocorrelation and partial autocorrelation functions for $\tilde{m}_1$, up to lag 20, are shown in Figure 5.9

The sample autocorrelation and partial autocorrelation functions in Figure 5.9 indicate that the first seasonally adjusted principal component can be modelled as a realisation from a white-noise process. That is, we assume that $\tilde{m}_1$ is a realisation from the process $(\mathsf{X}_t)_{t\in\mathbb{Z}}$ with $\mathsf{X}_t = \varepsilon_t$ and $\varepsilon_t \sim \mathcal{N}(0, \sigma^2_{\varepsilon,1})$, where only the value $\sigma^2_{\varepsilon,1}$ is needed to be estimated. The estimate of the variance can be given by the zero lag sample autocovariance $\hat{\sigma}^2_{\varepsilon,1} = c(0)$.

Very similar sample autocorrelation and partial autocorrelation results are found for the remaining five seasonally adjusted principal components, and therefore we judge that they are all realisations from white-noise processes. Estimates for the variance of the white-noise process are found using the training sets $\{\tilde{m}_i^{(\tilde{n})}, \tilde{m}_i^{(\tilde{n}+1)}, \ldots, \tilde{m}_i^{(n-1)}\}$, $i = 1, \ldots, 6$ and with $\tilde{n} = 1600$, $n = 2400$, giving $\hat{\sigma}^2_{\varepsilon,i}(j)$, $j = 1600, \ldots, 2399$. The estimates for the standard deviation obtained



(a) Sample autocorrelation function (correlogram) for the first seasonally adjusted principal component, $\tilde{m}_1$



(b) Sample partial autocorrelation function for the first seasonally adjusted principal component, $\tilde{m}_1$

Figure 5.9: Sample autocorrelation (correlogram) and partial autocorrelation functions for the first seasonally adjusted principal component, $\tilde{m}_1$, obtained from the HadCM3 simulator mean sea-level air pressure with wind velocity output.

using training sets of the first four seasonally adjusted principal components are shown in Figure 5.10 by solid, dotted, dot-dashed and dashed lines respectively.

5.3.1.2 Modelling the principal components as sparse observations from Ornstein-Uhlenbeck processes

In this section we assume that the seasonally adjusted principal components are realisations from some Ornstein-Uhlenbeck processes, and the method of moments is used to obtained drift and diffusion parameter estimates for the training samples.

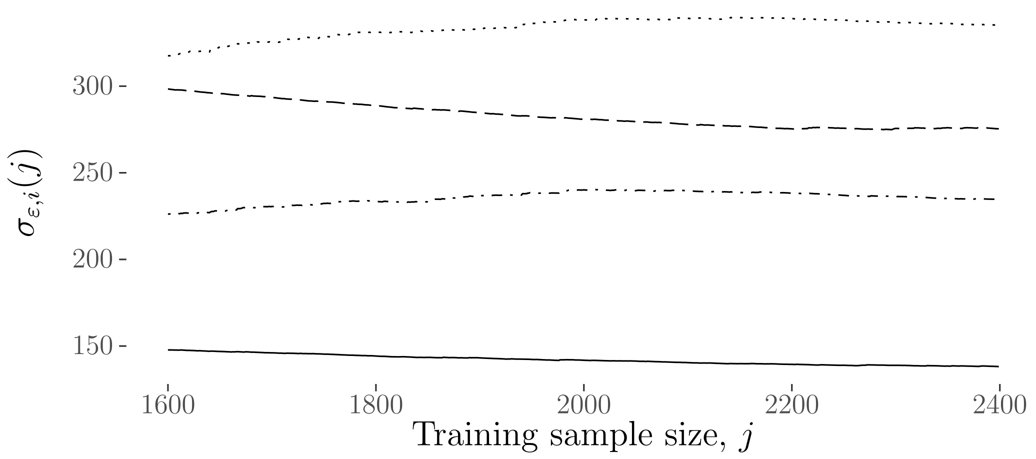


Figure 5.10: Estimation of the standard deviation of the white-noise process, $\hat{\sigma}_{\varepsilon,i}(j)$ for seasonally adjusted principal components $i = 1, 2, 3, 4$, and for training samples $\tilde{m}_i^{(j)}$ of size $j = 1600, 1601, \ldots, 2399$. The standard deviation estimations using training samples from the first, second, third and fourth seasonally adjusted principal component are given by the solid, dotted, dot-dashed and dashed lines respectively.

Here we obtain parameter estimates for the training sets from the first six seasonally adjusted principal components, $\{\tilde{m}_i^{(1600)}, \tilde{m}_i^{(1601)}, \ldots, \tilde{m}_i^{(2399)}\}$, $i = 1, \ldots, 6$, giving $\hat{\gamma}_{\text{sML},i}(j)$ and $\hat{\sigma}^2_{\text{sML},i}(j)$, $j = 1600, \ldots, 2399$.

Using method of moments parameter estimation

The method of moments parameter estimation (as introduced in Section 2.3.3.1) is used here.

First, we check which training samples have positive sample lag-1 correlation, as this is required to be able to apply method of moments. The sample lag-1 correlation of the training samples $\tilde{m}_i^{(1600)}, \tilde{m}_i^{(1601)}, \ldots, \tilde{m}_i^{(2399)}$, $i = 1, \ldots, 6$, is shown in Figure 5.11.

From this it is clear that for every seasonally adjusted principal component, there is at least one training sample that results in a negative sample lag-1 correlation. Thus, the method of moments can not be applied to the full training set of any of the seasonally adjusted principal components. The second, fourth, fifth and sixth seasonally adjusted principal components all have training samples for which the sample lag-1 correlation is positive (which are shown by the thick dotted, dot-dashed, solid and long-dashed lines in Figure 5.11, respectively). All of the training samples from the first and third seasonally adjusted principal components have negative sample lag-1 correlation, and these are shown in Figure 5.11 by the thin solid and long dashed-dashed lines respectively. We note that all training samples have sample lag-1 correlations that lie within the approximate 95% confidence interval for sample lag-1 correlation of white noise, given by the grey lines.

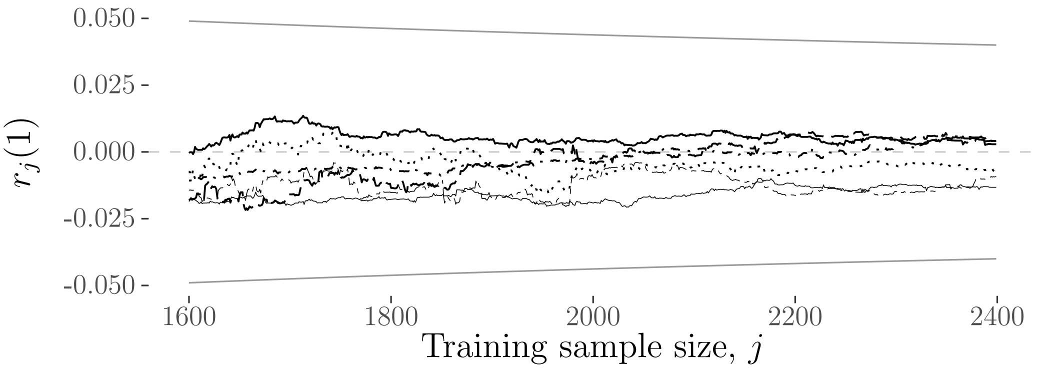
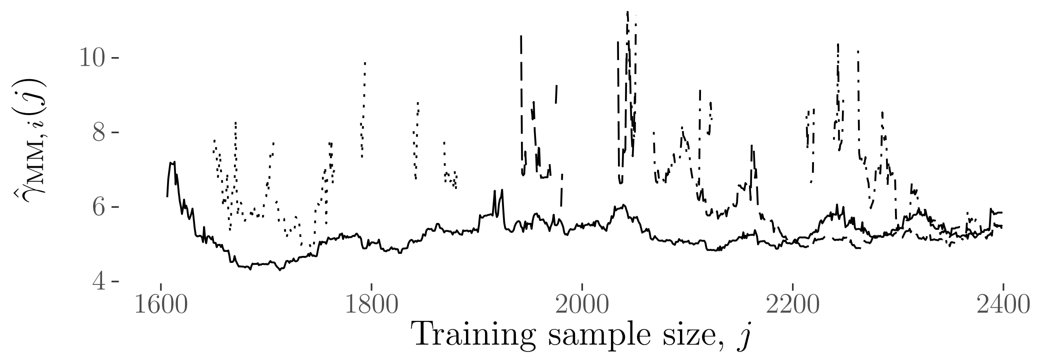
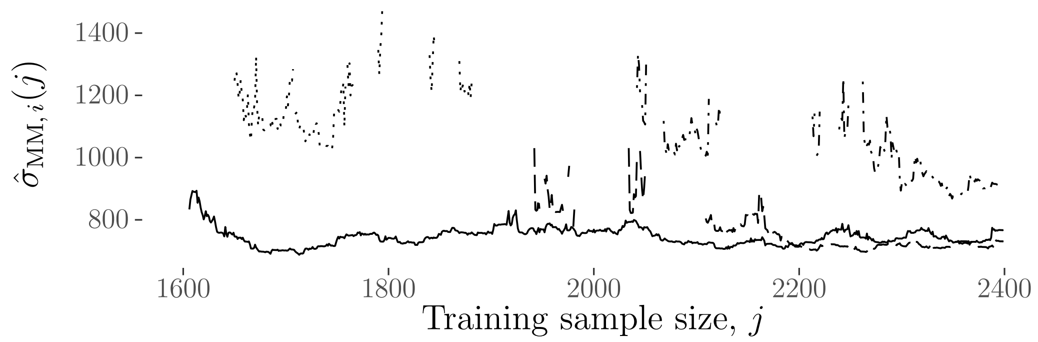


Figure 5.11: Sample lag-1 correlations for training samples of the first six seasonally adjusted principal component, $\tilde{m}_i^{(1600)}, \ldots, \tilde{m}_i^{2399}$, $i = 1, \ldots, 6$. The thick dotted, dot-dashed, solid and long-dashed lines correspond to the sample lag-1 correlation for training samples of the second, fourth, fifth and sixth seasonally adjusted principal components respectively. These four seasonally adjusted principal components have, for some training samples, a positive sample lag-1 correlation. The thin solid and long dashed-dashed lines correspond to the sample lag-1 correlations for training samples of the first and third seasonally adjusted principal components, which are all negative. The grey solid lines give the approximate 95% interval for sample lag-1 correlation of white noise.

Now, we apply method of moments on the training samples of the second, fourth, fifth and sixth seasonally adjusted principal components which have positive sample lag-1 correlation. From this, we obtain parameter estimates for the drift and diffusion terms, $\hat{\gamma}_{\mathrm{MM},i}(j)$, and $\hat{\sigma}^2_{\mathrm{MM},i}(j)$, for all $i \in \{2, 4, 5, 6\}$ and $j \in \{1600, 1601, \ldots, 2399\}$ where the training sample $\tilde{m}_i^{(j)}$ has positive sample lag-1 correlation. These estimates are shown in Figure 5.12, with the dotted, dot-dashed, solid and long-dashed lines corresponding to the sample lag-1 correlations of the second, fourth, fifth and sixth seasonally adjusted principal components, respectively. The gaps in the plots correspond to training samples with non-positive sample lag-1 correlation.



(a) Estimates $\hat{\gamma}_{\mathrm{MM},i}(j)$ for seasonally adjusted principal components $i = 2, 4, 5, 6$ and training samples of size $j \in \{1600, \ldots, 2399\}$, where $\tilde{m}_i^{(j)}$ has a positive sample lag-1 correlation, given by the dotted, dot-dashed, solid and long-dashed lines respectively.



(b) Estimates $\hat{\sigma}_{\mathrm{MM},i}(j)$ for seasonally adjusted principal components $i = 2, 4, 5, 6$ and training samples of size $j \in \{1600, \ldots, 2399\}$, where $\tilde{m}_i^{(j)}$ has a positive sample lag-1 correlation, given by the dotted, dot-dashed, solid and long-dashed lines respectively.

Figure 5.12: Ornstein-Uhlenbeck parameter estimates for the second, fourth fifth and sixth seasonally adjusted principal components, using the method of moments on the training samples $\tilde{m}_i^{(j)}$, $i = 2, 4, 5, 6$, $j \in \{1600, \ldots, 2399\}$ which have positive sample lag-1 correlation. Here, estimates using the second, fourth fifth and sixth seasonally adjusted principal component are given by the dotted, dot-dashed, solid and long-dashed lines respectively

5.3.1.3 Modelling the principal components as observations from block-average Ornstein-Uhlenbeck processes

In this section we assume that the seasonally adjusted principal components, obtained from the mean sea-level air pressure with wind velocity HadCM3 simulator output, are realisations from some some block-average Ornstein-Uhlenbeck processes (similar to Section 4.4.2.3 for the mean sea-level air pressure output).

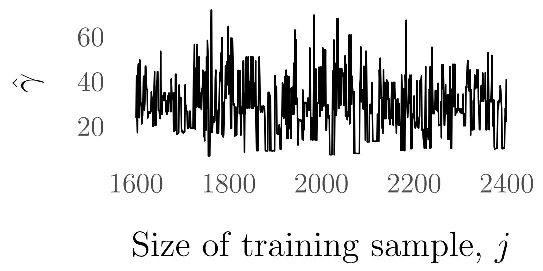
We find parameter estimates for the training samples of the first 6 seasonally adjusted principal component using the block-average Ornstein-Uhlenbeck maximum likelihood estimator (introduced in Section 2.3.3.3).

For each seasonally adjusted principal component, we obtain estimates $\hat{\gamma}_{\mathrm{sML},\,i}(j)$ and $\hat{\sigma}^2_{\mathrm{sML},\,i}(j)$ by applying the maximum likelihood method on the training sample $\tilde{m}_i^{(j)} = (\tilde{m}_{i\,1}, \tilde{m}_{i\,2}, \ldots, \tilde{m}_{i\,j})$, $i = 1, \ldots, 6$, with $j = 1600, 1601, \ldots, 2399$. The parameter estimates $\hat{\gamma}_{\mathrm{baML},\,i}(j)$ and $\hat{\sigma}_{\mathrm{baML},\,i}(j)$ for training samples of the first three seasonally adjusted principal components are shown in Figure 5.13. The somewhat disjointed nature of these estimates is due to optimisation issues, as here the inverse of large matrices need to be found and optimised over to maximise the log-likelihood (given in (2.32)). Due to the non-convergence of the optimisation here, modelling of the seasonally adjusted principal components by block-average Ornstein-Uhlenbeck processes is not used in the forecasting stage later in this chapter.

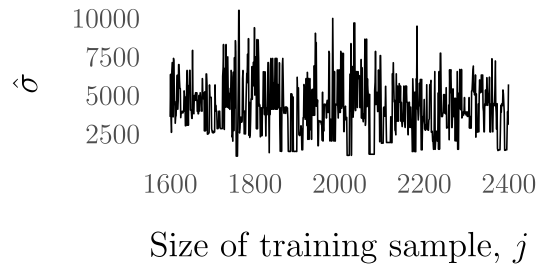
5.3.2 Modelling the independent components

This section mirrors the layout of Section 5.3.1 above. Before modelling the independent components (that were obtained in Section 5.2), we check for seasonality, and remove this if present. Then, we assume that the training samples of the seasonally adjusted independent components are realisations from ARIMA, Ornstein-Uhlenbeck, and block-average Ornstein-Uhlenbeck processes.

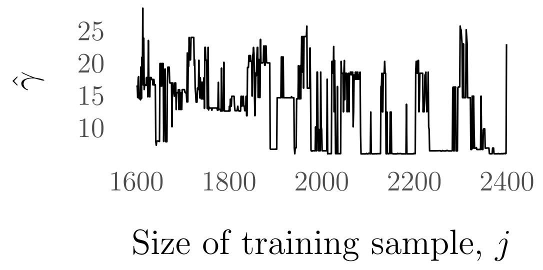
Figure 5.14 shows the correlograms for the first six independent components. As with the principal components in the section above, here seasonality is present and therefore we remove it by calculating the seasonal means.



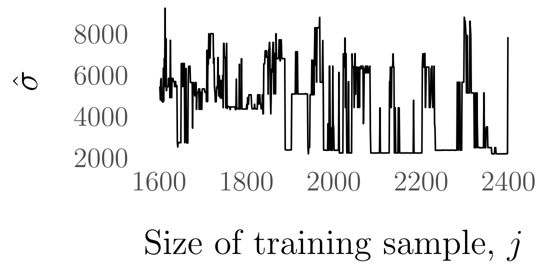
(a) Maximum likelihood estimator $\hat{\gamma}_{\text{baML},1}(j)$ found using the training samples of size $j = 1600, \ldots, 2399$ of the first seasonally adjusted principal component.



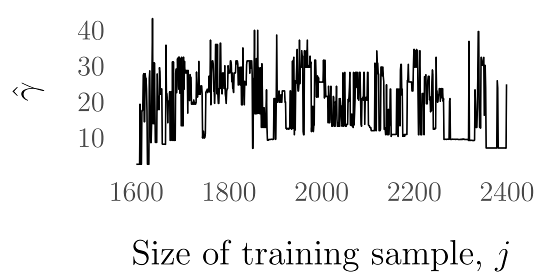
(b) Maximum likelihood estimator $\hat{\sigma}_{\text{baML},1}(j)$ found using the training samples of size $j = 1600, \ldots, 2399$ of the first seasonally adjusted principal component.



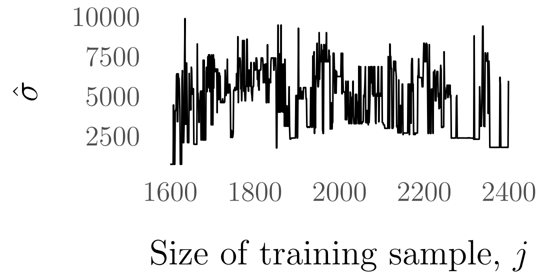
(c) Maximum likelihood estimator $\hat{\gamma}_{\text{baML},2}(j)$ found using the training samples of size $j = 1600, \ldots, 2399$ of the second seasonally adjusted principal component.



(d) Maximum likelihood estimator $\hat{\sigma}_{\text{baML},2}(j)$ found using the training samples of size $j = 1600, \ldots, 2399$ of the second seasonally adjusted principal component.

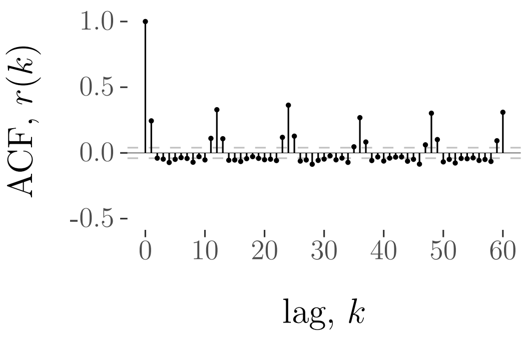


(e) Maximum likelihood estimator $\hat{\gamma}_{\text{baML},3}(j)$ found using the training samples of size $j = 1600, \ldots, 2399$ of the third seasonally adjusted principal component.

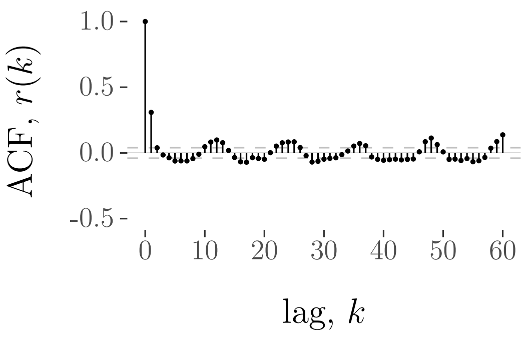


(f) Maximum likelihood estimator $\hat{\sigma}_{\text{baML},3}(j)$ found using the training samples of size $j = 1600, \ldots, 2399$ of the third seasonally adjusted principal component.

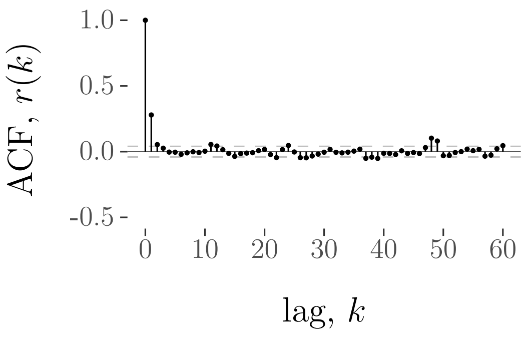
Figure 5.13: Maximum likelihood estimators $\hat{\gamma}_{\text{baML},i}(j)$ and $\hat{\sigma}_{\text{baML},i}(j)$ found using training samples of size $j = 1600, \ldots, 2399$ of the first three seasonally adjusted principal components.



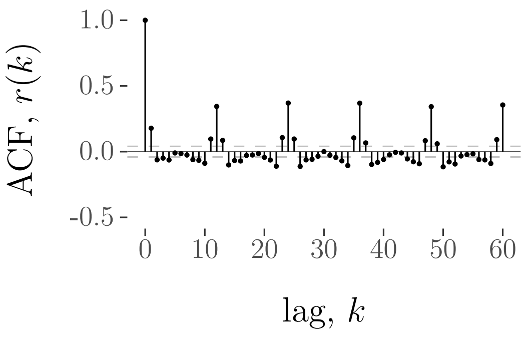
(a) Autocorrelation function (correlogram) for the first independent component, s_1, for lags $k = 0, 1, \ldots, 60$.



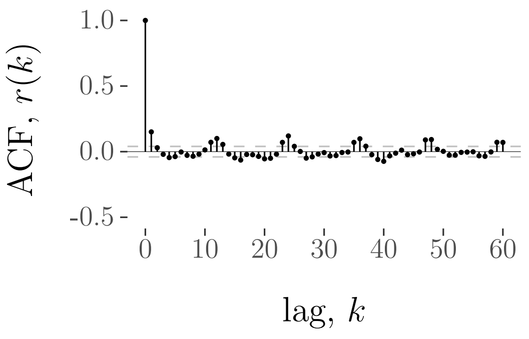
(b) Autocorrelation function (correlogram) for the second independent component, s_2, for lags $k = 0, 1, \ldots, 60$.



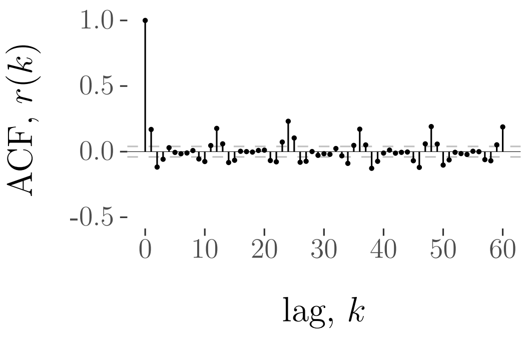
(c) Autocorrelation function (correlogram) for the third independent component, s_3, for lags $k = 0, 1, \ldots, 60$.



(d) Autocorrelation function (correlogram) for the fourth independent component, s_4, for lags $k = 0, 1, \ldots, 60$.



(e) Autocorrelation function (correlogram) for the fifth independent component, s_5, for lags $k = 0, 1, \ldots, 60$.



(f) Autocorrelation function (correlogram) for the sixth independent component, s_6, for lags $k = 0, 1, \ldots, 60$.

Figure 5.14: Autocorrelation functions (correlogram) for the first six independent component, $s_1, \ldots, s_6$, obtained by applying fastICA to the mean sea-level air pressure with wind velocity HadCM3 simulator output, for lags $k = 0, 1, \ldots, 60$.

On the left-hand of Figure 5.15 we show the first three independent components (black line), with the periodic quadratic spline (calculated as in Section 2.3.4.2) given by the grey line. The right-hand column shows the first three seasonally adjusted independent components. From this figure, we can see that both the seasonality and stochasticity of the independent components are much smaller than in the principal components (shown in Figure 5.8). We have previously commented on this behaviour in Section 4.4.3, for the principal and independent components found on the mean sea-level air pressure HadCM3 simulator output.

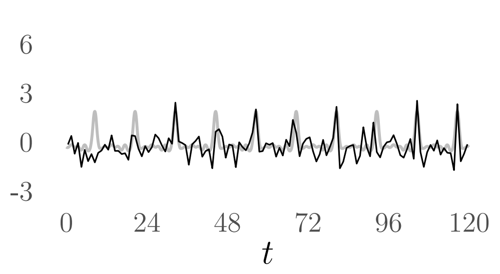
5.3.2.1 Modelling the independent components as observations from ARIMA processes

Here we assume that the seasonally adjusted independent components are realisations from ARIMA processes. As before, the model structure is found using the Box-Jenkins method and then estimates for the parameters $\varphi_1, \ldots, \varphi_p$ and $\vartheta_1, \ldots, \vartheta_q$ are calculated.

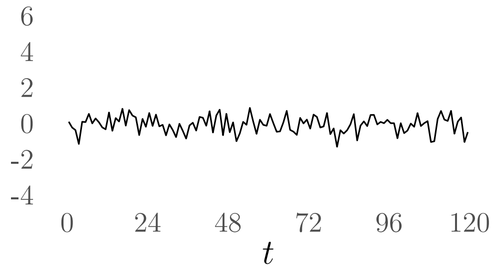
We do not give detailed steps of the Box-Jenkins method here, as for all of the first six seasonally adjusted independent components, the sample autocorrelation and partial autocorrelation plots suggest that the white-noise process can be used to model these seasonally adjusted independent components. Then, the white-noise variance is estimated for the training sets of the six seasonally adjusted independent components, $\{\tilde{s}_i^{(1600)}, \ldots, \tilde{s}_i^{(2399)}\}$, such that we obtain, $\sigma^2_{\varepsilon,i}(1600), \sigma^2_{\varepsilon,i}(1601), \ldots, \sigma^2_{\varepsilon,i}(2399)$, $i = 1, \ldots, 6$. The estimated standard deviations for the training sets of the first four seasonally adjusted independent components are shown in Figure 5.16.

5.3.2.2 Modelling the independent components as sparse observations from Ornstein-Uhlenbeck processes

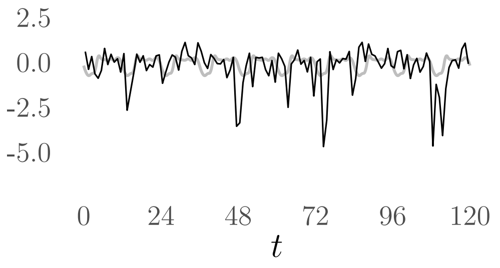
In this section we model the seasonally adjusted independent components (obtained by applying fastICA on the mean sea-level air pressure with wind velocity HadCM3 simulator output) as realisations from Ornstein-Uhlenbeck processes. As



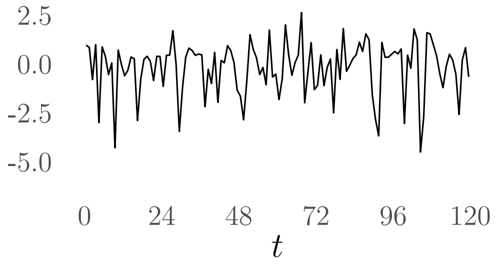
(a) First independent component, s_1, for times $t = 0, 1, \ldots, 119$. The grey line gives the associated quadratic spline.



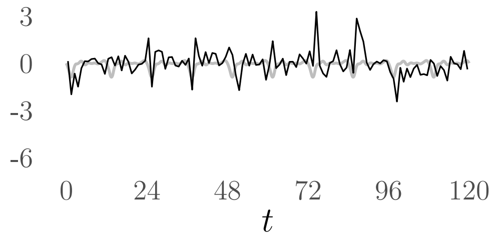
(b) First seasonally adjusted independent component, $\tilde{s}_1$, for times $t = 0, 1, \ldots, 119$.



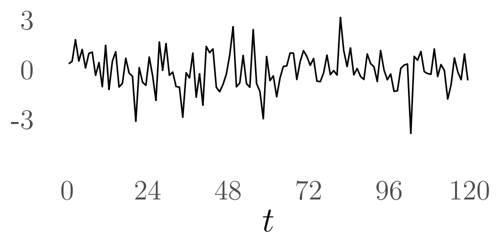
(c) Second independent component, s_2, for times $t = 0, 1, \ldots, 119$. The grey line gives the associated quadratic spline.



(d) Second seasonally adjusted independent component, $\tilde{s}_2$, for times $t = 0, 1, \ldots, 119$.



(e) Third independent component, s_3, for times $t = 0, 1, \ldots, 119$. The grey line gives the associated quadratic spline.



(f) Third seasonally adjusted independent component, $\tilde{s}_3$, for times $t = 0, 1, \ldots, 119$.

Figure 5.15: The left-hand column gives the first three independent components, s_1, s_2, s_3 for times $t = 0, \ldots, 119$, found by projecting the (column-centred) mean sea-level air pressure with wind velocity HadCM3 simulator output, $\tilde{X}$, along the first three independent directions (shown in Figure 5.5), by the black lines. The grey lines give the quadratic mean function. The right-hand column gives the first three seasonally adjusted independent components, $\tilde{s}_1, \tilde{s}_2, \tilde{s}_3$ for times $t = 0, \ldots, 119$.

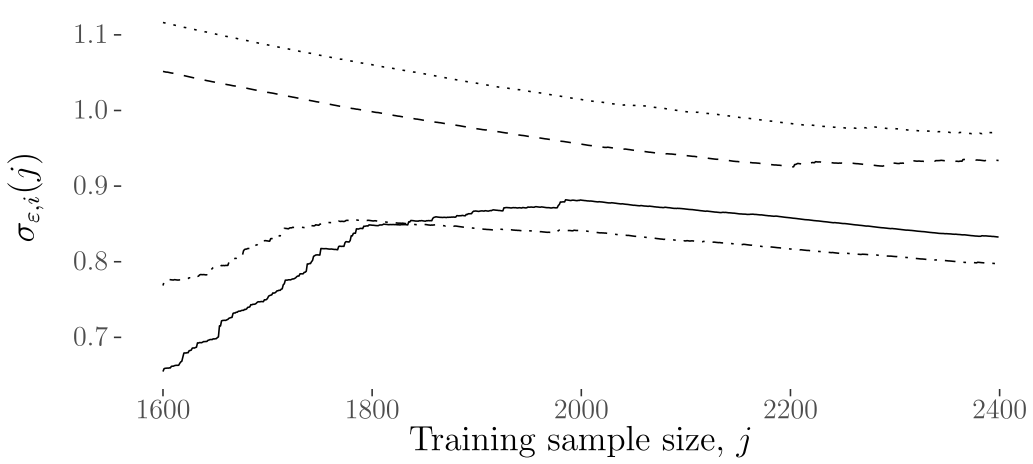


Figure 5.16: Estimation of the standard deviation of the white-noise process, $\hat{\sigma}_{\varepsilon,i}(j)$, for training samples of the first four seasonally adjusted independent components, $\tilde{s}_i^{(j)}$, $i = 1, \ldots, 4$, $j = 1600, \ldots, 2399$. The estimates using training samples from the first, second, third and fourth seasonally adjusted independent components are given by the solid, dashed, dotted and dot-dashed lines respectively.

in previous sections, here we only use the method of moments to obtain estimates of the drift and diffusion parameters γ and σ^2.

As before, we calculate the sample lag-1 correlations for the training sets $\{\tilde{s}_i^{(4000)}, \tilde{s}_i^{(4001)}, \ldots, \tilde{s}_i^{(5999)}\}$, $i = 1, \ldots, 6$, as we require positive sample lag-1 correlation to apply the method of moments. The sample lag-1 correlations for these training sets are shown in Figure 5.17. The thick solid and dashed lines correspond to the sample lag-1 correlation for training samples of the first, and fifth seasonally adjusted independent components respectively. These two seasonally adjusted independent components have positive sample lag-1 correlation for some training samples. The thin dotted, dot-dashed, long-dashed and solid lines correspond to the sample lag-1 correlation for training samples of the second, third, fourth and sixth seasonally adjusted independent components respectively. None of the training samples from these four seasonally adjusted independent components have positive sample lag-1 correlation. The grey solid lines in Figure 5.17 give the approximate 95% interval for sample lag-1 correlation of white noise. Note that some of the training samples for the fourth and sixth seasonally adjusted independent components have sample lag-1 correlations that are outside of this 95% confidence interval. For the fourth seasonally adjusted independent components, the number of training samples where this happens is small, and for the sixth seasonally adjusted independent components, the sample lag-1 correlation is always very close to the lower boundary of the prediction interval for white-noise sample correlation.

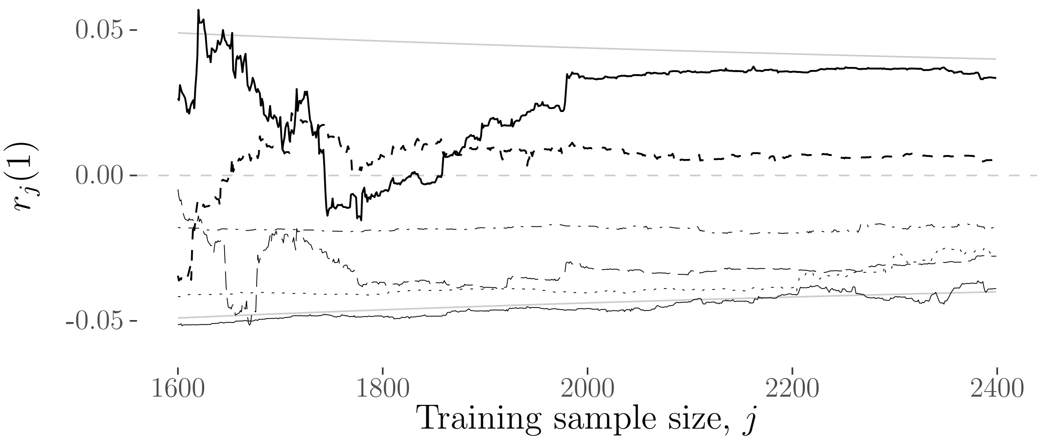
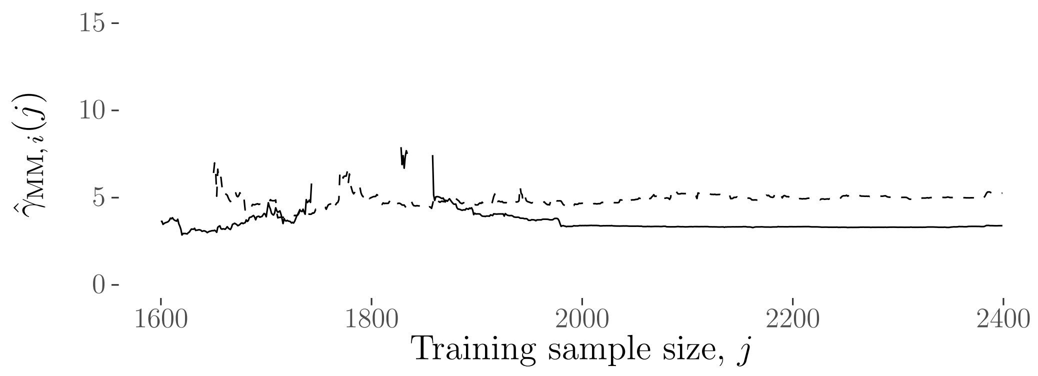
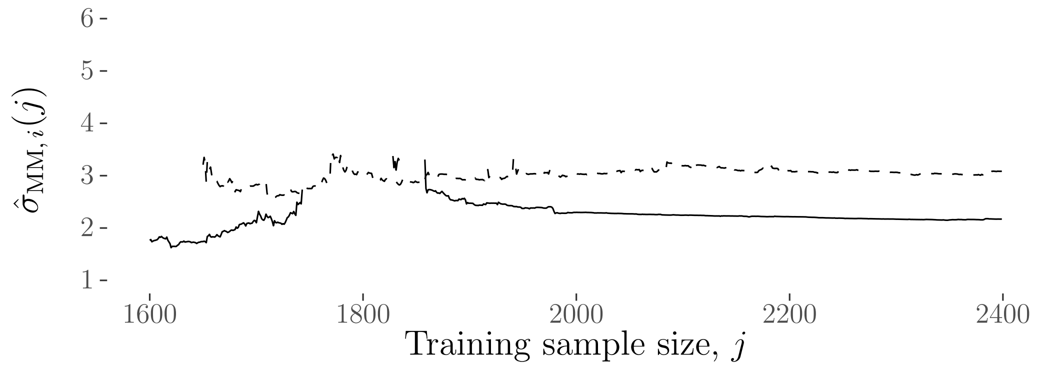


Figure 5.17: Sample lag-1 correlations for training samples of the first six seasonally adjusted independent components, $\tilde{s}_i^{(4000)}, \ldots, \tilde{s}_i^{(5999)}$, $i = 1, \ldots, 6$. The thick solid and dashed lines correspond to the sample lag-1 correlations for training samples of the first, and fifth seasonally adjusted independent components respectively. The thin dotted, dot-dashed, long-dashed and solid lines correspond to the sample lag-1 correlations for training samples of the second, third, fourth and sixth seasonally adjusted independent components. The grey solid lines give the approximate 95% interval for sample lag-1 correlation of white noise.

For training samples of the first and fifth seasonally adjusted independent components that have positive sample lag-1 correlation, we obtain the parameter estimates $\hat{\gamma}_{\mathrm{MM},i}(j)$ and $\hat{\sigma}_{\mathrm{MM},i}(j)$, $i = 1, 5$, for $j \in \{1600, \ldots, 2399\}$. These are shown in Figure 5.18 by the solid and dashed lines respectively. The gaps in the lines correspond to training samples that have a non-positive sample lag-1 correlation.



(a) Estimates $\hat{\gamma}_{\mathrm{MM},i}(j)$ for seasonally adjusted independent components $i = 1, 5$ and training samples of size $j = 1600, \ldots, 2399$.



(b) Estimates $\hat{\sigma}_{\mathrm{MM},i}(j)$ for seasonally adjusted independent components $i = 1, 5$ and training samples of size $j = 1600, \ldots, 2399$.

Figure 5.18: Ornstein-Uhlenbeck parameter estimates for the first and fifth seasonally adjusted independent components, given by the solid and dashed lines respectively, using the method of moments on the training samples $\tilde{s}_i^{(j)}$, $i = 1, 5$, $j = 1600, \ldots, 2399$. The estimates are only shown for the training samples of size $j \in \{1600, \ldots, 2399\}$ which have positive sample lag-1 correlation.

5.3.2.3 Modelling the independent components as observations from block-average Ornstein-Uhlenbeck processes

In this section we assume the seasonally adjusted independent component are realisations from block-average Ornstein-Uhlenbeck processes, and use maximum likelihood to find estimates for the drift and diffusion parameters.

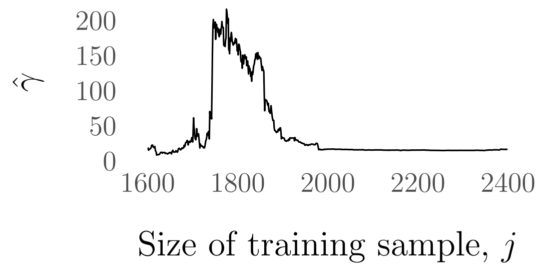
Here, for each seasonally adjusted independent component, $\tilde{s}_i$, $i = 1, \ldots, 6$ we apply the maximum likelihood estimation method to the training samples given by $\tilde{s}_i^{(j)} = (\tilde{s}_{i\,0}, \tilde{s}_{i\,1}, \ldots, \tilde{s}_{i\,(j-1)})$, for $j = 1600, \ldots, 2399$. This give the estimates $\hat{\gamma}_{\text{baML},\,i}(j)$ and $\hat{\sigma}^2_{\text{baML},\,i}(j)$, $i = 1, \ldots, 6$, $j = 1600, \ldots, 2399$. The parameter estimates obtained in this way for the first three seasonally adjusted independent component are shown in Figure 5.19. The parameter estimates obtained using the training samples of the first seasonally adjusted independent component seem to behave in a discontinuous manner, with a large step-change of the estimates at approximately $j = 1750$. These estimates then appear to tend back to their original values in an asymptotic manner. This could indicate that two separate stochastic processes are required to model this seasonally adjusted independent component properly, although we have not investigated this further due to time constraints. Note also the gap in the parameter estimates for some of the training samples of the second seasonally adjusted independent component, resulting from optimisation issues. In this case, the parameter estimates found using the largest training sample prior to this gap are used in the forecasting stage later in this chapter.

5.4 Forecasting Principal and Independent Components

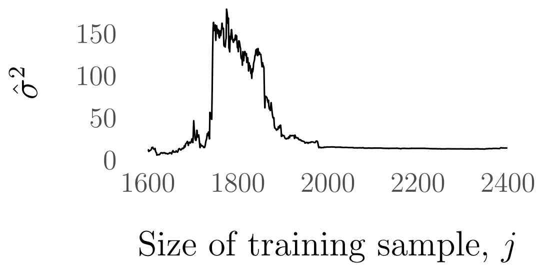
Now, we use the stochastic models (described in Section 5.3) to forecast the principal and independent components, using the forecasting theory introduced in Chapter 3.

This section follows the same principals as discussed in Section 4.5, such that we want to obtain one-step-ahead forecasts for the components. That is, with $x = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{R}^n$ some realisation from a stochastic process, and the training set $\{x^{(\tilde{n})}, x^{(\tilde{n}+1)}, \ldots, x^{(n-1)}\}$, we calculate the point forecasts $\hat{x}_{\tilde{n}-1}(1), \hat{x}_{\tilde{n}}(1), \ldots, \hat{x}_{n-2}(1)$ and associated prediction interval.

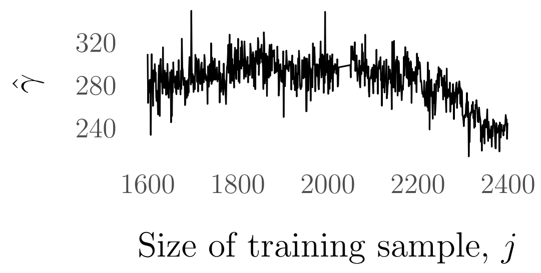
In Section 5.4.1, we find one-step-ahead forecasts for the first six principal components, and in Section 5.4.2 we find one-step-ahead forecasts for the first six independent components. Each of these sections are subdivided into subsections



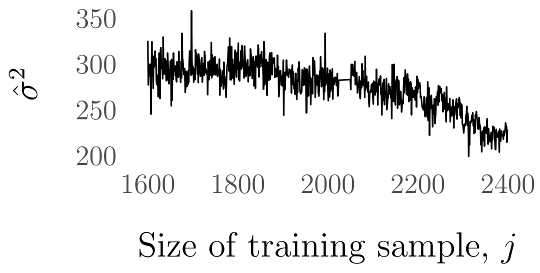
(a) Parameter estimates $\hat{\gamma}_{\text{baML},1}(j)$ using training samples of size $j = 1600, 1601, \ldots, 2399$.



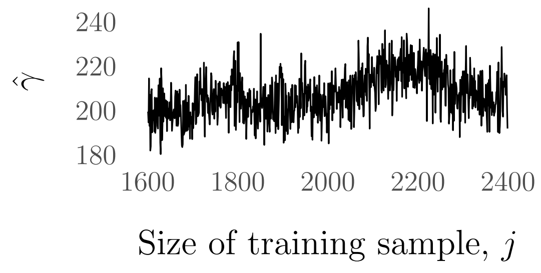
(b) Parameter estimates $\hat{\sigma}_{\text{baML},1}(j)$ using training samples of size $j = 1600, 1601, \ldots, 2399$.



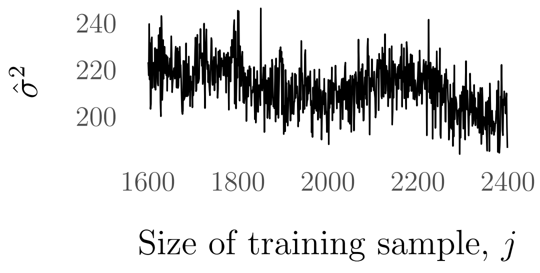
(c) Parameter estimates $\hat{\gamma}_{\text{baML},2}(j)$ using training samples of size $j = 1600, 1601, \ldots, 2399$.



(d) Parameter estimates $\hat{\sigma}_{\text{baML},2}(j)$ using training samples of size $j = 1600, 1601, \ldots, 2399$.



(e) Parameter estimates $\hat{\gamma}_{\text{baML},3}(j)$ using training samples of size $j = 1600, 1601, \ldots, 2399$.



(f) Parameter estimates $\hat{\sigma}_{\text{baML},3}(j)$ using training samples of size $j = 1600, 1601, \ldots, 2399$.

Figure 5.19: Parameter estimates $\hat{\gamma}_{\text{baML},i}(j)$ and $\hat{\sigma}_{\text{baML},i}(j)$, $i = 1, 2, 3$, found on training samples of the first three seasonally adjusted independent components, of size $j = 1600, 1601, \ldots, 2399$.

corresponding to the mixed forecasting approach, and the block-average Ornstein-Uhlenbeck forecasting approach (both discussed earlier in Section 4.5.1).

5.4.1 Forecasting principal components

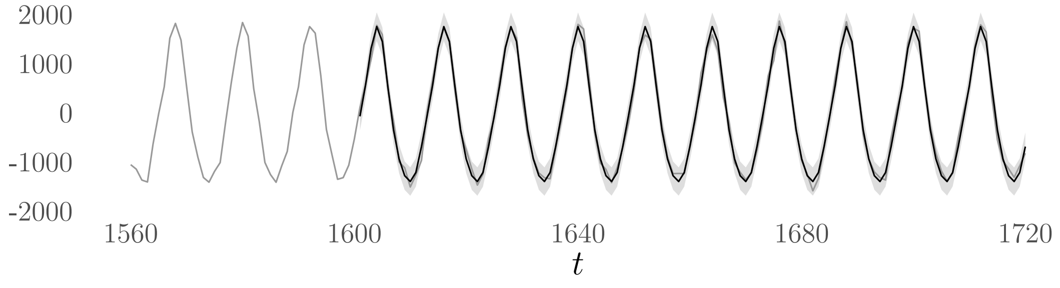
In this section we obtain forecasts for the principal components relating to the mean sea-level air pressure with wind velocity HadCM3 simulator output (described in Section 5.1). In Section 5.4.1.1, forecasts are found using the mixed forecasting approach (as described in Panel 4.12). Due to the block-average Ornstein-Uhlenbeck parameter estimation issue (seen in Section 5.3.1.3), we do not consider the block-average Ornstein-Uhlenbeck forecasting approach here.

5.4.1.1 Mixed forecasting approach

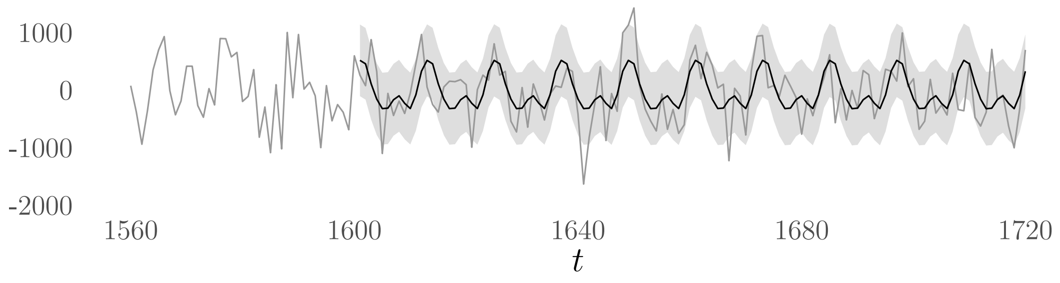
We have shown (in Section 5.3.1.2) that for the second, fourth, fifth and sixth seasonally adjusted principal components, some of the training samples have positive sample lag-1 correlation. In these cases, method of moments can be used to obtain estimates of the Ornstein-Uhlenbeck process drift and diffusion parameters. For all other training samples, the (non-positive) sample lag-1 correlations were shown to be within the approximate 95% confidence interval for sample lag-1 correlation of a realisation from a white-noise processes. Therefore, we judge that the mixed forecasting approach is a suitable method to obtain point forecasts and prediction intervals of the principal components.

We show the point forecasts (black line) and prediction intervals (grey shading) for the first three principal components in Figure 5.20. The grey line shows the true values of the principal components. Using the mixed forecasting approach, 1.6% of the test sample of the first principal component lies outside the 95% prediction interval. The proportion seems quite low, suggesting that the prediction interval is too wide, potentially due to model misspecification. For the test samples of the second and third principal components, the proportion is 6.5% and 6.2% respectively.

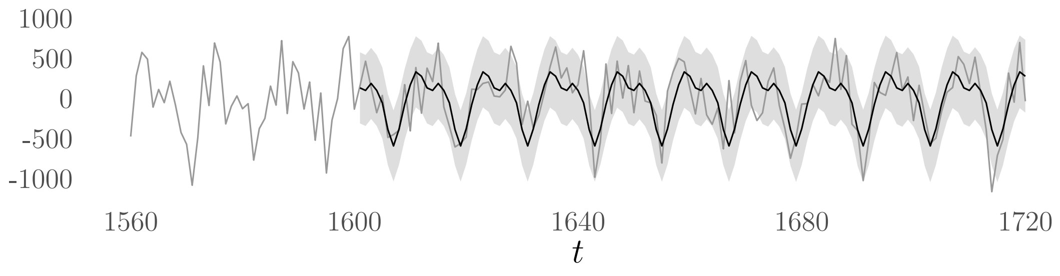
As seen previously, we assess the validity of the forecasts using root mean-squared error (as introduced in Section 4.5.1). The solid, dashed and dotted lines



(a) First principal component for $t = 1560, \ldots, 1720$ (grey solid line). The one-step-ahead point forecast for is given by the solid black line, with the 95% prediction interval given by the shaded grey area.



(b) Second principal component for $t = 1560, \ldots, 1720$ (grey solid line). The one-step-ahead point forecast for is given by the solid black line, with the 95% prediction interval given by the shaded grey area.



(c) Third principal component for $t = 1560, \ldots, 1720$ (grey solid line). The one-step-ahead point forecast for is given by the solid black line, with the 95% prediction interval given by the shaded grey area.

Figure 5.20: First, second and third principal components plotted as a time series for time $t = 1560, \ldots, 1720$ (grey solid line). The one-step-ahead point forecast found using the mixed forecasting method is given by the solid black line, with the 95% prediction interval given by the shaded grey area.

in Figure 5.21 show the rolling averages of the root mean-squared error between the point forecasts for the first, second and third principal component, and the actual principal components, respectively.

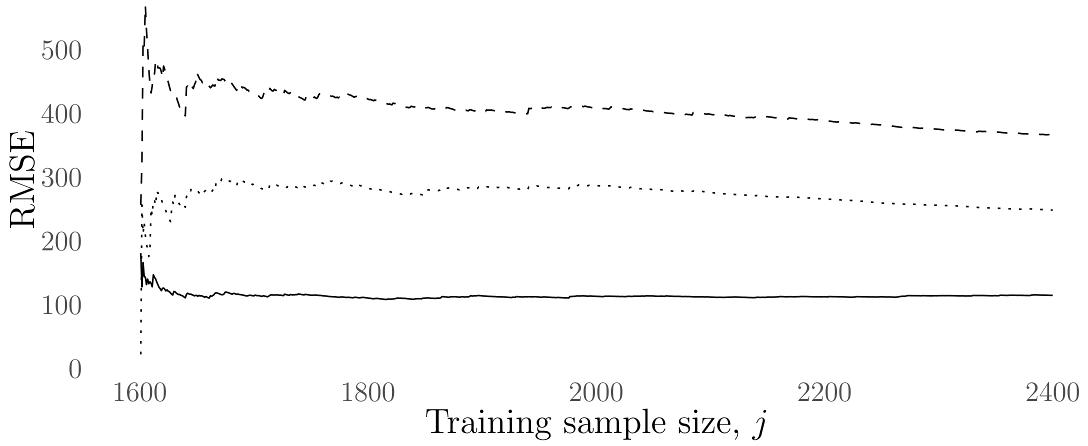


Figure 5.21: Rolling average of the root mean-squared error between the true principal components and the respective point forecasts, found using the mixed forecasting approach. The rolling average of the root mean-squared error between the first, second and third principal component and the associated one-step-ahead forecasts is given by the solid, dashed and dotted line respectively.

5.4.2 Forecasting independent components

Similarly to Section 5.4.1, here one-step-ahead forecasts are found for the first six independent components obtained by applying the fastICA method to the mean sea-level air pressure with wind velocity HadCM3 simulator output. Here we obtain forecasts using both the mixed forecasting approach, and the block-average Ornstein-Uhlenbeck forecasting approach (as discussed in Section 4.5.1).

5.4.2.1 Mixed forecasting approach

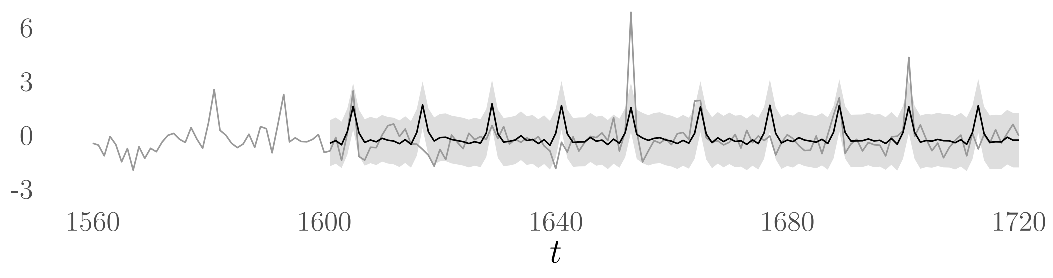
From Section 5.3.2.2 it was shown that for a subset of the training sets from the first and fifth seasonally adjusted independent components, the method of moments parameter estimation could be used to obtain estimates assuming an underlying Ornstein-Uhlenbeck process. All other training samples from the first six seasonally adjusted independent component have non-positive sample lag-1 correlation. A subset of these (from the fourth and sixth seasonally adjusted independent components) are below the lower bound of the 95% confidence interval for sample lag-1 correlation of white-noise realisations (seen in Figure 5.17). However, only a very small number of training samples from the fourth seasonally adjusted independent component have sample lag-1 correlations that are below this bound, and the training samples from the sixth seasonally adjusted independent component all have sample lag-1 correlations that are very close to this lower bound. Due to this, we judge that the mixed forecasting approach can be used here.

In Figure 5.22 we show the first three independent components obtained by applying fastICA to the mean sea-level air pressure with wind velocity HadCM3 simulation output by the grey line. The black lines give the one-step-ahead forecast and the grey shading the approximate 95% prediction interval, using the mixed forecasting approach. Here, 12.0% of the test set for the first independent component, $s_{1\,1600}, s_{1\,1601}, \ldots, s_{1\,2399}$, lie outside of the associated approximate 95% prediction interval. The proportion of the test sets of the second and third principal components that lie outside of the approximate 95% prediction interval is 2.1% and 1.1% respectively.

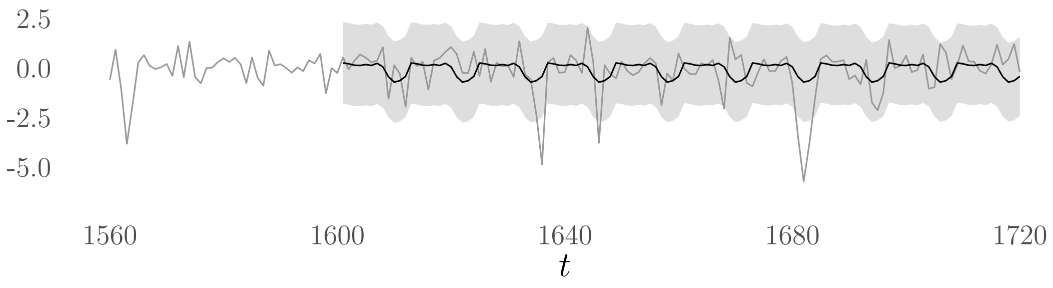
We now consider the root mean-squared error of these forecasts. In Figure 5.23 we show the rolling average of the root mean-squared error between the point forecast found using mixed forecasting approach for the first three independent components, and the true independent components. Examining Figure 5.23, and recalling that 12% of the test set of the first independent component lies outside of the approximate 95% prediction interval, suggests that this independent component is not modelled very well. The rolling root mean-squared error for the first independent component is consistently above that for the second and third independent component.

5.4.2.2 Block-average Ornstein-Uhlenbeck forecasting approach

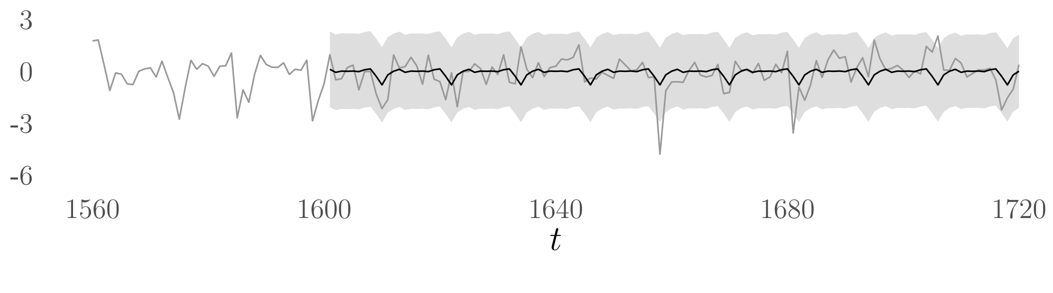
Here we obtain forecasts for the independent components using the block-average Ornstein-Uhlenbeck forecasting approach, with the block-average Ornstein-Uhlenbeck



(a) First independent component for $t = 1560, \ldots, 1720$ (grey solid line). The one-step-ahead point forecast for is given by the solid black line, with the 95% prediction interval given by the shaded grey area.



(b) Second independent component for $t = 1560, \ldots, 1720$ (grey solid line). The one-step-ahead point forecast for is given by the solid black line, with the 95% prediction interval given by the shaded grey area.



(c) Third independent component for $t = 1560, \ldots, 1720$ (grey solid line). The one-step-ahead point forecast for is given by the solid black line, with the 95% prediction interval given by the shaded grey area.

Figure 5.22: First, second and third independent components plotted as a time series for time $t = 1560, \ldots, 1720$ (grey solid line). The black line gives the one-step-ahead point forecast, and the grey shading gives the 95% prediction interval, when the mixed forecasting approach is used.

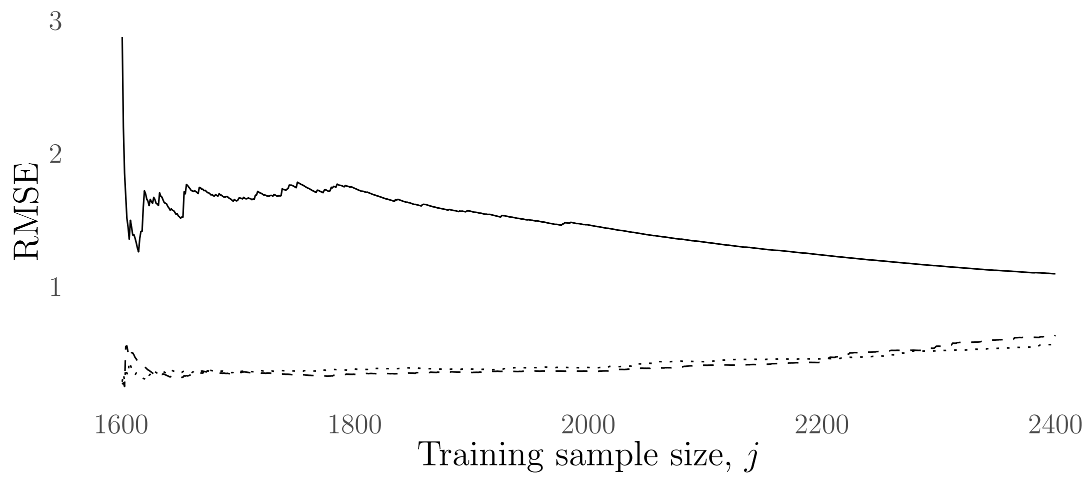


Figure 5.23: Rolling average of the root mean-squared error between the true independent components and the respective point forecasts using the mixed forecasting approach. The rolling average of the root mean-squared errors between the first, second and third independent component and the associated one-step ahead forecasts are given by the solid, dashed and dotted lines respectively.

parameter estimates found in Section 5.3.1.3. The point forecasts and prediction intervals here are all very similar to those seen when the mixed forecasting approach was used (Section 5.4.2.1), and therefore have been placed in Appendix B.1. These figures are:

- The point forecasts and 95% prediction intervals for the first three independent components, shown in Figure B.1;
- The rolling average of the root mean-squared error between the point forecasts for the first three independent components, and the true independent components, shown in Figure B.2;

Using the block-average Ornstein-Uhlenbeck forecasting approach, 12.0% of the test set for the first independent component, $s_{1\,1600}$, $s_{1\,1601}, \ldots,$ $s_{1\,2399}$, lie outside of the associated approximate 95% prediction interval. The proportion of the test sets of the second and third principal components that lie outside of the approximate 95% prediction interval is 2.1% and 1.1% respectively. For the first three independent components, the same proportion of the test samples are outside of the respective 95% prediction intervals here as they were for the prediction interval found using the mixed forecasting approach (Section 5.4.2.1). This suggests that these prediction intervals are extremely similar to one-another.

The similarity of the root mean-squared errors here to those found using the mixed forecasting approach implies that the point forecasts of the two forecasts methods are very similar, and that the misspecification of the model in the mixed forecasting approach does not greatly affect the outcome. This was also the case for the mean sea-level air pressure HadCM3 simulator output examined in Chapter 4.

In this section we have calculated point forecasts and prediction intervals for the independent components, using both the mixed forecasting approach and the block-average forecasting approach. We have found that both these forecasting approaches give extremely similar point forecasts and prediction intervals. This is shown by the comparability of the root mean-squared errors between the true components and the point forecasts, and by the equal proportion of test samples that lie outside the prediction intervals.

5.5 Reconstruction of the Climate Simulator Output

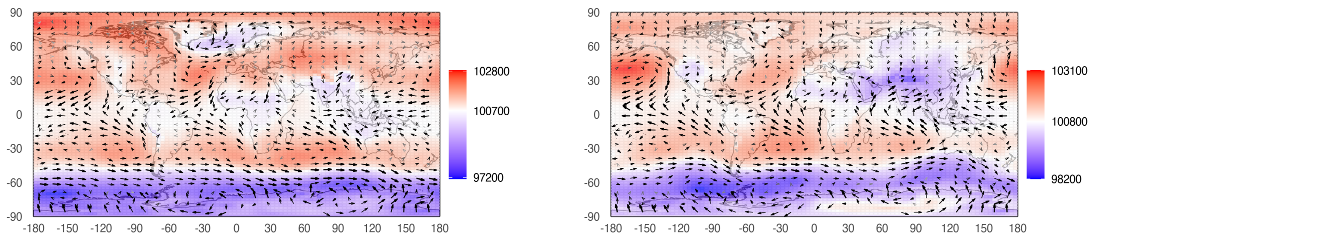
We now compare the full mean sea-level air pressure with wind velocity HadCM3 simulator output to the principal and independent component reconstructions, and to the forecasted principal and independent component reconstructions. The reconstructions are obtained using inverse principal component analysis (introduced in Panel 4.13) and inverse independent component analysis (introduce in Panel 4.14).

In Section 5.5.1 we give an introduction to these reconstructions. Then, we analyse the forecasted principal component reconstructions in Section 5.5.2, and the forecasted independent component reconstructions in Section 5.5.3.

5.5.1 Introduction to reconstructing the climate simulator output

Throughout this section, we use visual comparisons of the reconstructions at times $t = 1600$ and $t = 2000$. Recall that the forecasts for the components (found in Section 5.4 above) are found for times $t = 1600, 1601, \ldots, 2399$, such that here we look at reconstructions are the beginning and at the midpoint of the forecasts.

The mean sea-level air pressure and wind velocity HadCM3 simulator output at times $t = 1600$ and $t = 2000$ is shown in Figure 5.24. As before (and throughout this section) the arrows represent the wind velocity and have the properties given in Figure 5.2.



(a) Time, $t = 1600$.   (b) Time, $t = 2000$.

Figure 5.24: Mean sea-level air pressure and wind velocity HadCM3 simulator output at times $t = 1600$ and $t = 2000$.

We assess the validity of the forecasts by calculating the root mean-squared errors for the reconstructions. These errors are calculated for the mean sea-level air pressure and for the wind velocity separately in the following way.

First, recall that the HadCM3 simulator output here is given by the matrix $X \in \mathbb{R}^{2400\times((73\times96)+2(72\times96))}$, such that the first $73 \times 96 = 7008$ columns of X relate to the mean sea-level air pressure output, and the last $2\cdot(72\times96) = 13824$ columns of X relate to the wind velocity output. Therefore the mean sea-level air pressure output can be given by the matrix $X_{\text{mslp}} \in \mathbb{R}^{2400\times7008}$, and the wind velocity output by $X_{\text{wind}} \in \mathbb{R}^{2400\times13824}$, such that the full output satisfies $X = (X_{\text{mslp}}\, X_{\text{wind}})$.

This notation is mirrored with the principal and independent component reconstructions, given by the matrix $Z^{(l)} \in \mathbb{R}^{n\times p}$ with $n = 2400$ and $p = (73 \times 96) +$

$2 \times (72 \times 96) = 20832$. That is, we split $Z^{(l)}$ into the mean sea-level air pressure component, $Z^{(l)}_{\text{mslp}} \in \mathbb{R}^{2400\times 7008}$, by taking the first 7008 column of $Z^{(l)}$, and the wind velocity component $Z^{(l)}_{\text{wind}} \in \mathbb{R}^{2400\times 13824}$, by taking the last 13824 columns of $Z^{(l)}$.

In this case, the root mean-squared error between the mean sea-level air pressure part of the principal or independent component reconstructions (using l components), given by $Z^{(l)}_{\text{mslp}}$, and the true mean sea-level air pressure output is,

$$\begin{aligned} e_{\text{mslp}}(t\colon l) &:= \left(\frac{1}{73\times 96}\sum_{i=1}^{73\times 96}(z^{(l)}_{ti} - x_{ti})^2\right)^{1/2} \\ &= \left(\frac{1}{7008}\sum_{i=1}^{7008}(z^{(l)}_{ti} - x_{ti})^2\right)^{1/2}. \end{aligned} \tag{5.2}$$

Similarly, the equivalent root mean-squared error for the wind velocity principal or independent component reconstruction is,

$$\begin{aligned} e_{\text{wind}}(t\colon l) &:= \left(\frac{1}{2\times(72\times 96)}\sum_{i=(73\times 96)+1}^{(73\times 96)+2(72\times 96)}(z^{(l)}_{ti} - x_{ti})^2\right)^{1/2} \\ &= \left(\frac{1}{13824}\sum_{i=7009}^{20832}(z^{(l)}_{ti} - x_{ti})^2\right)^{1/2}, \end{aligned} \tag{5.3}$$

where as before, $Z^{(l)} = \left(z^{(l)}_0 \cdots z^{(l)}_{n-1}\right)^\top \in \mathbb{R}^{n\times p}$ and $X = (x_0 \cdots x_{n-1})^\top \in \mathbb{R}^{n\times p}$ are such that z_{ij} (respectively, x_{ij}) is the $(i+1)^{\text{th}}$ row, j^{th} column element of $Z^{(l)}$ (respectively, X).

Let $\hat{M}^{(l)}_{\tilde{n}} \in \mathbb{R}^{2400\times l}$ be the matrix with the first l principal components as columns, but with the test samples of the principal components, given by the elements $m_{i\,\tilde{n}}, m_{i\,\tilde{n}+1}, \ldots, m_{i\,n-1}$, are replaced by some one-step-ahead forecasts $\hat{m}_{i\,\tilde{n}-1}(1), \ldots, \hat{m}_{i\,n-2}(1)$. This is the same definition as (4.8) (Section 4.6.1). Similarly, $\hat{S}^{(l)}_{\tilde{n}} \in \mathbb{R}^{2400\times l}$ has columns given by the first l independent components, where the test samples of the independent components are replaced by the one-step-ahead point forecasts (as introduced in (4.9)). Throughout this section we have $n = 2400$ and we set $\tilde{n} = 1600$ and $l = 6$.

Forecasted principal or independent component reconstructions are given by the matrix $\hat{Z}_{\tilde{n}}^{(l)}$. As seen above, this can be split into the mean sea-level air pressure part and the wind velocity part to give $\hat{Z}_{\tilde{n},\text{mslp}}^{(l)} \in \mathbb{R}^{2400\times 7008}$ and $\hat{Z}_{\tilde{n},\text{wind}}^{(l)} \in \mathbb{R}^{2400\times 13824}$ respectively, such that $\hat{Z}_{\tilde{n}}^{(l)} = (\hat{Z}_{\tilde{n},\text{mslp}}^{(l)}\ \hat{Z}_{\tilde{n},\text{wind}}^{(l)})$.

In this case, the root mean-squared errors between the forecasted principal or independent component reconstructions, and the true HadCM3 output, is given by

$$\hat{e}_{\text{mslp}}(t\colon l,\, X) := \Big(\frac{1}{7008}\sum_{i=1}^{7008}(\hat{z}_{ti}^{(l)} - x_{ti})^2\Big)^{1/2}, \tag{5.4}$$

and,

$$\hat{e}_{\text{wind}}(t\colon l,\, X) := \Big(\frac{1}{13824}\sum_{i=7009}^{20832}(\hat{z}_{ti}^{(l)} - x_{ti})^2\Big)^{1/2}, \tag{5.5}$$

for the mean sea-level air pressure, and the wind velocity, respectively, with $\hat{Z}_{\tilde{n}}^{(l)} = \big(\hat{z}_0^{(l)} \cdots \hat{z}_{n-1}^{(l)}\big)^\top \in \mathbb{R}^{n\times p}$.

Similarly, the root mean-squared errors between the forecasted principal or independent component reconstructions and the principal or independent component reconstructions is given by

$$\hat{e}_{\text{mslp}}(t\colon l,\, Z) := \Big(\frac{1}{7008}\sum_{i=1}^{7008}(\hat{z}_{ti}^{(l)} - z_{ti}^{(l)})^2\Big)^{1/2}, \tag{5.6}$$

using the mean sea-level air pressure part, and,

$$\hat{e}_{\text{wind}}(t\colon l,\, Z) := \Big(\frac{1}{13824}\sum_{i=7009}^{20832}(\hat{z}_{ti}^{(l)} - z_{ti}^{(l)})^2\Big)^{1/2}, \tag{5.7}$$

for the wind velocity part.

The rolling average of the root mean-squared errors is considered. For the root mean-squared errors between the principal or independent component reconstructions and the true HadCM3 output ((5.2) and (5.3)), this is given by

$$\frac{1}{j-1599}\sum_{t=1600}^{j} e_{\text{mslp}}(t\colon l);\ \text{and};\ \frac{1}{j-1599}\sum_{t=1600}^{j} e_{\text{wind}}(t\colon l); \tag{5.8}$$

for $j = 1600, \ldots, 2399$. The rolling averages are similarly defined for the root mean-squared errors (5.6), (5.7), (5.4), and (5.5).

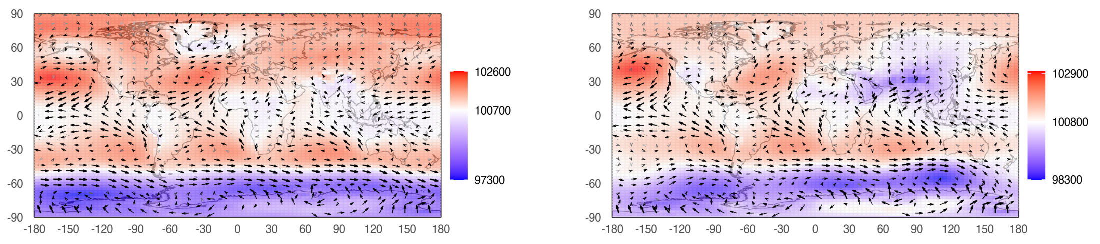
5.5.2 Reconstruction using principal components

In this section we find forecasted principal component reconstructions, and we compare these to the true HadCM3 simulator output, and to the principal component reconstructions with $l = 6$ principal components.

We first introduce the principal component reconstructions in Section 5.5.2.1, and compare these to the true HadCM3 simulator output. In Section 5.5.2.2 we consider forecasted principal component reconstructions using the mixed forecasting approach to obtain the matrix $\hat{M}_{\tilde{n}}^{(l)}$. Recall that the block-average Ornstein-Uhlenbeck forecasting approach did not give adequate results in Section 5.3.1.3 and is therefore not considered here.

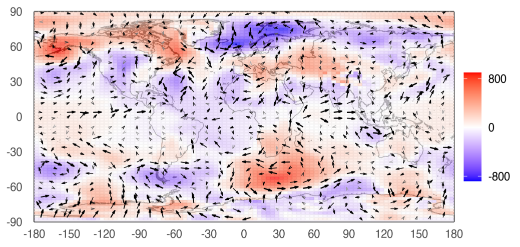
5.5.2.1 Principal component reconstructions

Here we give principal component reconstructions for $l = 6$ principal components, and compare them to the HadCM3 simulator output. We show the principal component reconstructions for times $t = 1600$ and $t = 2000$ in Figure 5.25. The differences between the HadCM3 simulator output (given in Figure 5.24) and the principal component reconstructions at these times are shown in Figure 5.26.



(a) Time, $t = 1600$. (b) Time, $t = 2000$.

Figure 5.25: Principal component reconstructions using $l = 6$ principal components of the mean sea-level air pressure with wind velocity output, at times $t = 1600$ and $t = 2000$.


(a) Time, $t = 1600$.

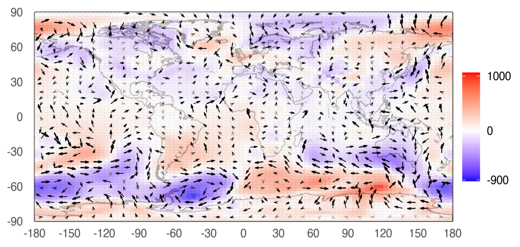

(b) Time, $t = 2000$.

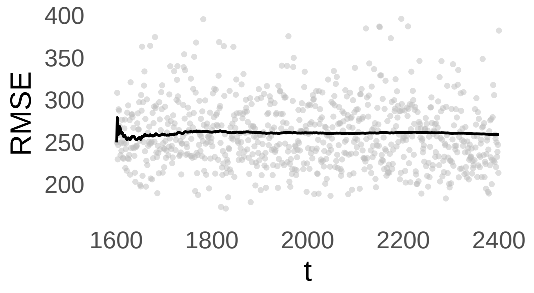
Figure 5.26: Difference between the HadCM3 mean sea-level air pressure with wind velocity HadCM3 simulated output, and the principal component reconstructions using $l = 6$ principal components, at times $t = 1600$ and $t = 2000$.

We calculate the root mean-squared errors for the differences between the HadCM3 simulator output and the principal component reconstructions at times $t = 1600, 1601, \ldots, 2399$, as shown by the grey circles in Figure 5.27. The black lines show the associated rolling average (as given by (5.8)). As we discussed in the introduction above (Section 5.5.1), the root mean-squared error calculations are split between the mean sea-level air pressure output and the wind velocity output. The range of the mean sea-level air pressure values in the HadCM3 simulator output is 95600 Pa to 104800 Pa and the standard deviation is 1261. This compares to the range of the wind velocity which is -14.8 to 19.7 ms^{-1} with standard deviation 3.3.

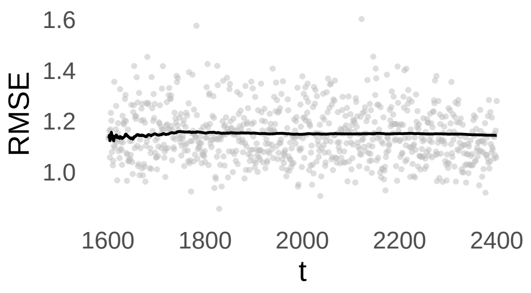
5.5.2.2 Reconstruction errors using the mixed forecasting approach

In this section we analyse the forecasted principal component reconstructions when the mixed forecasting approach is used for $l = 6$ principal components (as shown in Section 5.4.1.1). Figure 5.28 shows the forecasted principal component reconstructions at times $t = 1600$ and $t = 2000$.

The differences between the true HadCM3 simulator output and the forecasted principal component reconstructions using the mixed forecasting approach at times $t = 1600$ and $t = 2000$ is shown in Figure 5.29. Figure 5.30 gives the differences between the principal component reconstructions and the forecasted principal component reconstructions at the same time points.

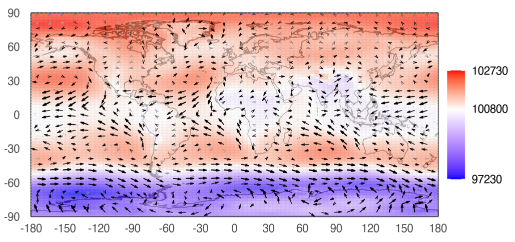


(a) Root mean-squared error per grid cell for the mean sea-level air pressure outputs (calculated using (5.2)).

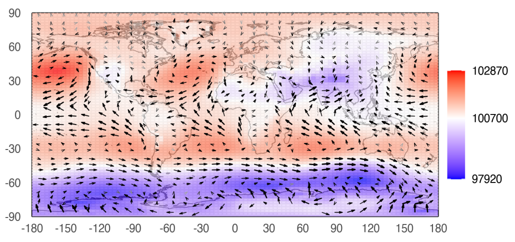


(b) Root mean-squared error per grid cell for the wind outputs (calculated using (5.3)).

Figure 5.27: The grey circles give the root mean-squared error per grid cell between the true HadCM3 simulator output and the principal component reconstructions with $l = 6$ principal components. The black lines give the rolling average of the root mean-squared errors.



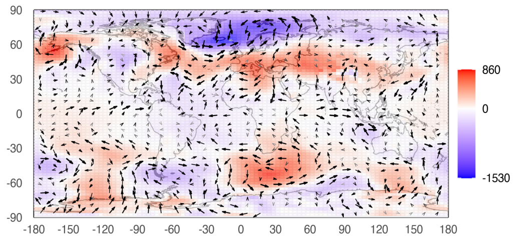
(a) Time, $t = 1600$.



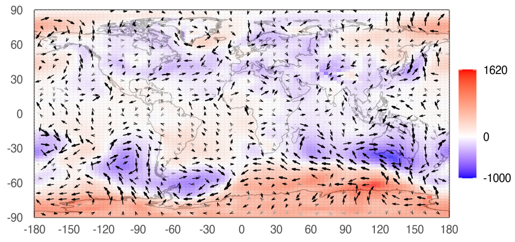
(b) Time, $t = 2000$.

Figure 5.28: Forecasted principal component reconstruction of the mean sea-level air pressure with wind velocity HadCM3 simulator output using the mixed forecasting approach, at times $t = 1600$ and $t = 2000$.

Now we use root mean-squared errors to assess the validity of the forecasted principal component reconstructions. As described in this section's introduction (Section 5.5.1), the reconstructions are split into the mean sea-level air pressure part, and the wind velocity part. For both of these parts, we calculate the root mean-squared errors between the forecasted principal component reconstructions and: *(i)* the true HadCM3 simulator output (calculated using (5.4) and (5.5)), and; *(ii)* the principal component reconstructions (calculated using (5.6) and (5.7)). These are shown by the grey circles in Figure 5.31 and Figure 5.32 respectively.

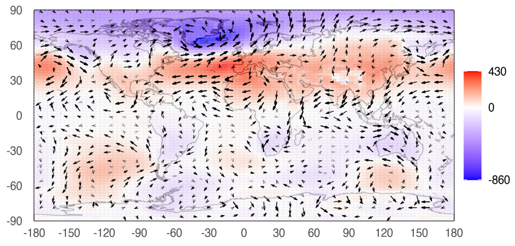


(a) Time, $t = 1600$.

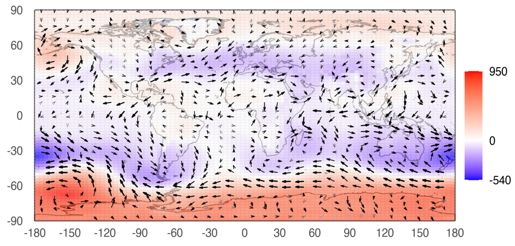


(b) Time, $t = 2000$.

Figure 5.29: Difference between the true HadCM3 simulator output, and the forecasted principal component reconstructions using the mixed forecasting approach, at times $t = 1600, 2000$.



(a) Time, $t = 1600$.

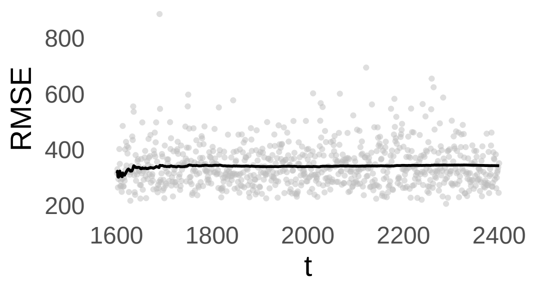


(b) Time, $t = 2000$.

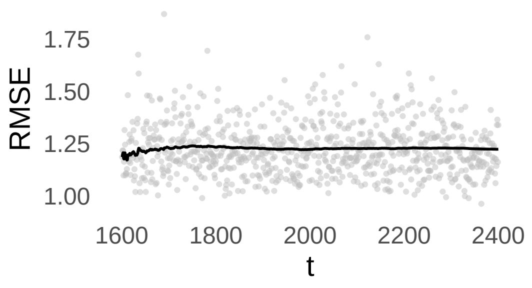
Figure 5.30: Difference between the principal component reconstructions (using $l = 6$ principal components) and the forecasted principal component reconstructions using the mixed forecasting approach, at times $t = 1600, 2000$.

The black lines give the associated rolling averages.

In this section we have analysed reconstructions of the HadCM3 simulator output using forecasted principal components found using the mixed forecasting approach. We have shown that the root mean-squared errors of the difference between the forecasted principal component reconstructions and the true HadCM3 simulator are slightly larger than those of the difference between the true principal component reconstructions and the HadCM3 simulator. We have also looked at the differences between the forecasted principal component reconstructions, the principal component reconstructions, and the HadCM3 simulator output at times $t = 1600$ and $t = 2000$. These are included to allow an expert to judge whether there are any systematic errors occurring in the reconstructions.

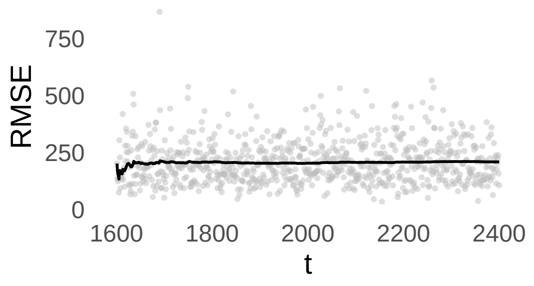


(a) Root mean-squared error per grid cell of the mean sea-level output as in (5.4).

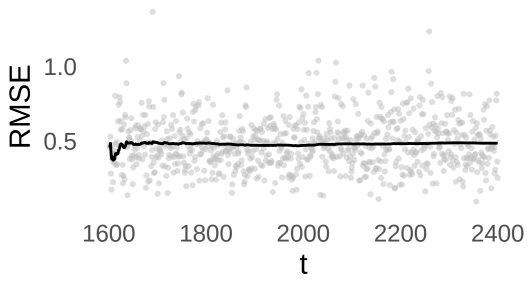


(b) Root mean-squared error per grid cell of the wind outputs as in (5.5).

Figure 5.31: The grey circles give the root mean-squared errors per grid cell between the forecasted principal component reconstructions, found using the mixed forecasting approach, and the true HadCM3 simulator output. This is split into the root mean-squared error of the mean sea-level air pressure values given by (5.4), and the root mean-squared error of the wind values given by (5.5). The running average of root mean-squared error over time (5.8) is given by the black line.



(a) Root mean-squared error per grid cell of the mean sea-level output as in (5.6).



(b) Root mean-squared error per grid cell of the wind outputs as in (5.7).

Figure 5.32: The grey circles give the root mean-squared errors per grid cell between the forecasted principal component reconstructions, found using the mixed forecasting approach, and the principal component reconstructions, using $l = 6$ principal components. This is split into the root mean-squared error of the mean sea-level air pressure values given by (5.6), and the root mean-squared error of the wind values given by (5.7). The running average of root mean-squared error over time (5.8) is given by the black line.

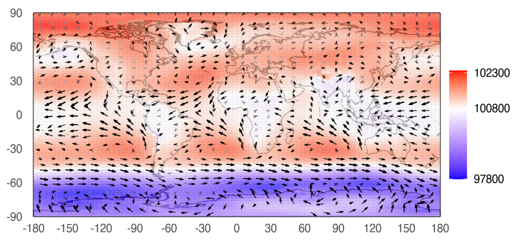
5.5.3 Reconstruction using independent components

In this section we find forecasted independent component reconstructions, and we compare these to the true HadCM3 simulator output, and to the independent component reconstructions.

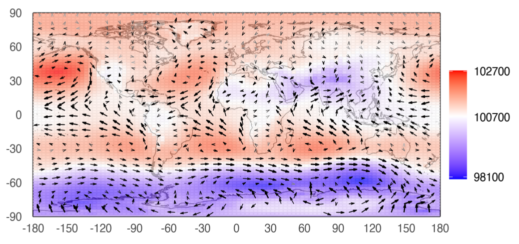
In Section 5.5.3.1 we introduce the independent component reconstructions and compare these to the HadCM3 simulator output. Then, we consider forecasted independent component reconstructions using the mixed forecasting approach in Section 5.5.3.2, and using the block-average Ornstein-Uhlenbeck forecasting approach in Section 5.5.3.3. This section has a similar layout as Section 5.5.2 (with the inclusion of the block-average Ornstein-Uhlenbeck forecasting approach), and all the plots produced here can be compared to the respective plots shown earlier.

5.5.3.1 Independent component reconstructions

The independent component reconstructions at times $t = 1600$ and $t = 2000$ are given in Figure 5.33. The differences between the HadCM3 simulator output and the independent component reconstructions at times $t = 1600$ and $t = 2000$, are shown in Figure 5.34.



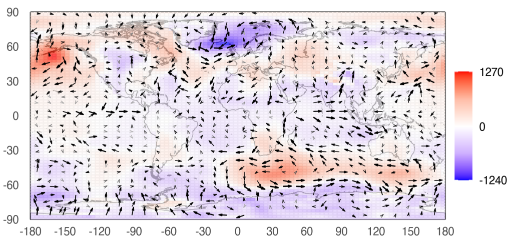
(a) Time, $t = 1600$.



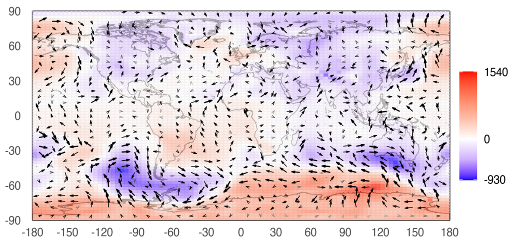
(b) Time, $t = 2000$.

Figure 5.33: Mean sea-level air pressure with wind velocity HadCM3 simulator independent component reconstructions (with $l = 6$ independent components), at times $t = 1600$ and $t = 2000$.

We now consider the root mean-squared errors between the true HadCM3 simulator output and the independent component reconstructions. The grey circles in Figure 5.35 give the root mean-squared errors for the independent component reconstructions, split into the mean sea-level air pressure component, and wind



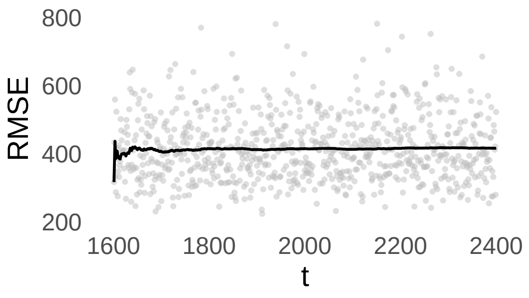
(a) Time, $t = 1600$.



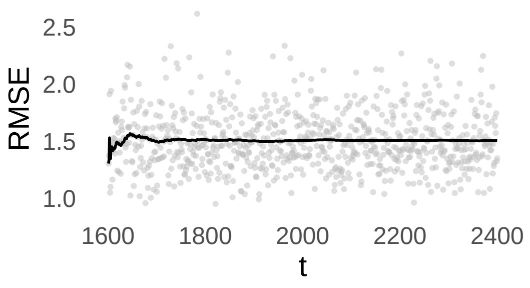
(b) Time, $t = 2000$.

Figure 5.34: Difference between the true mean sea-level difference pressure with wind velocity HadCM3 simulator output and the independent component reconstructions using $l = 6$ independent components at times $t = 1600$ and $t = 2000$.

velocity component (calculated using (5.2) and (5.3), respectively). The black lines show the rolling average of the root mean-squared error. As expected (see Remark 4.18), the root mean-squared errors are larger here than for the principal component reconstructions (shown in Figure 5.27).



(a) Root mean-squared error per grid cell of the mean sea-level output as in (5.2).



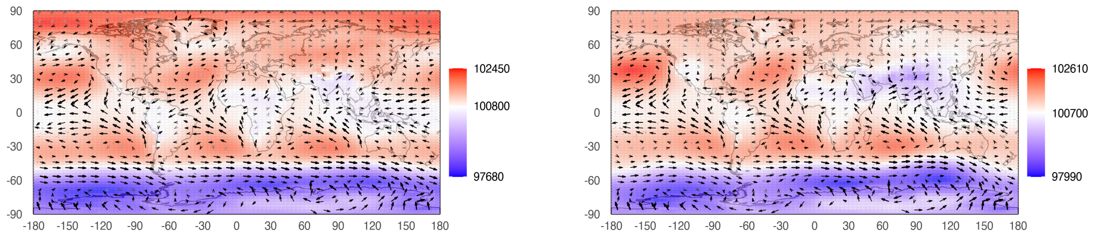
(b) Root mean-squared error per grid cell of the wind outputs as in (5.3).

Figure 5.35: Root mean-squared error per grid cell between the true HadCM3 simulator output and the independent component reconstructions, using $l = 6$ independent components, split into the mean sea-level air pressure output and the wind velocity output. The running average of root mean-squared error over time (calculated using (5.8)) is given by the black line.

5.5.3.2 Reconstruction errors using the mixed forecasting approach

In this section we analyse the forecasted independent component reconstructions when the mixed forecasting approach is used to obtain one-step-ahead forecasts for the first $l = 6$ independent components (as seen in Section 5.4.2.1). The forecasted independent component reconstructions using the mixed forecasting approach are shown in Figure 5.36 for times $t = 1600$ and $t = 2000$.

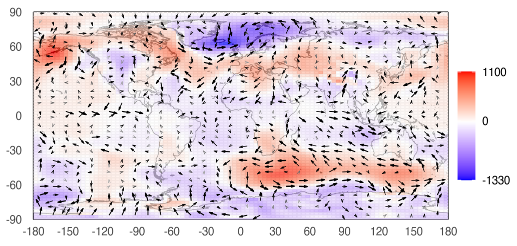
The difference between the true mean sea-level air pressure with wind velocity HadCM3 simulator output and the forecasted independent component reconstructions using the mixed forecasting approach is shown for times $t = 1600$ and $t = 2000$ in Figure 5.37. The difference between the independent component reconstructions using $l = 6$ independent components, and the forecasted independent component reconstructions using the mixed forecasting approach is shown for times $t = 1600$ and $t = 2000$ in Figure 5.38.



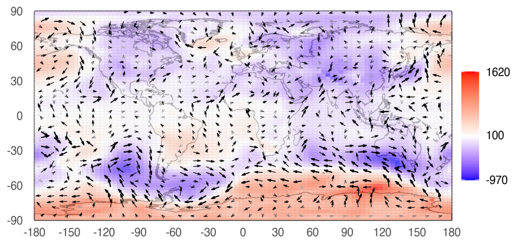
(a) Time, $t = 1600$. (b) Time, $t = 2000$.

Figure 5.36: Forecasted independent component reconstructions of the mean sea-level air pressure with wind velocity using the mixed forecasting approach, at times $t = 1600, 2000$.

We now consider the root mean-squared errors for the reconstructions, split into the mean sea-level air pressure component and the wind velocity component. The grey circles in Figure 5.39 and Figure 5.40 give the root mean-squared error between the forecasted independent component reconstructions using the mixed forecasting approach, and: *(i)* the true HadCM3 simulator output, and; *(ii)* the independent component reconstructions using $l = 6$ independent components, respectively. The black lines give the rolling average of the root mean-squared errors.

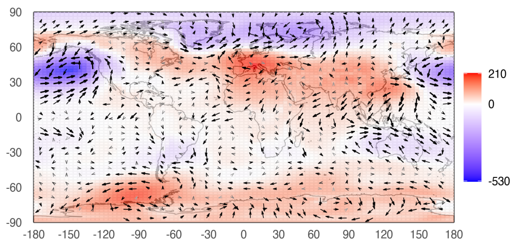


(a) Time, $t = 1600$.

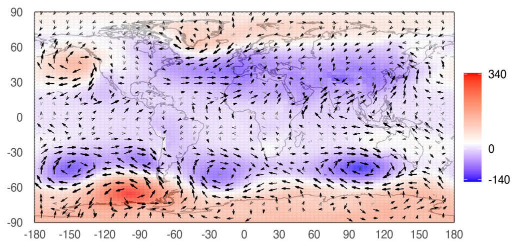


(b) Time, $t = 2000$.

Figure 5.37: Difference between the true mean sea-level air pressure with wind velocity HadCM3 simulator output and the forecasted independent component reconstructions using the mixed forecasting approach, at times $t = 1600, 2000$.



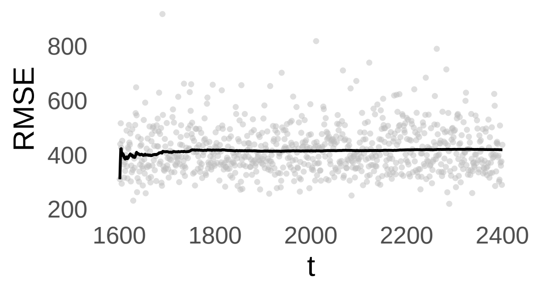
(a) Time, $t = 1600$.



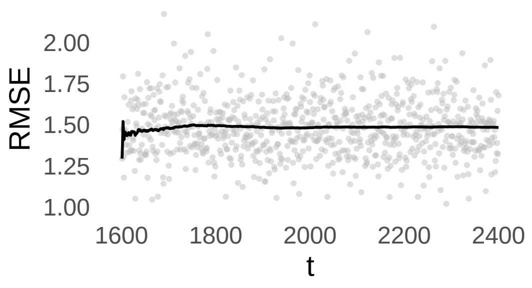
(b) Time, $t = 2000$.

Figure 5.38: Difference between the independent component reconstruction (with $l = 6$ independent components) of the mean sea-level air pressure with wind velocity HadCM3 simulator output, and the forecasted independent component reconstructions using the mixed forecasting approach, at times $t = 1600, 2000$.

One comment to make here is that the rolling average of the root mean-squared error between the true HadCM3 simulator output and the forecasted independent component reconstructions is very similar to the rolling average of the root mean-squared error between the true HadCM3 output and the independent component reconstructions (shown in Figure 5.35). For the equivalent principal component reconstructions, there was a clear difference in quality between the reconstructions found using the forecasts and using the true principal components (which can be seen by comparing Figure 5.31 and Figure 5.27).

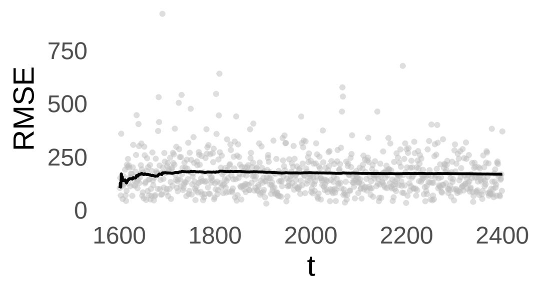


(a) Root mean-squared error per grid cell of the mean sea-level pressure.

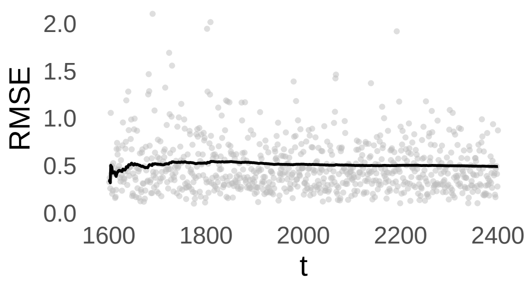


(b) Root mean-squared error per grid cell of the wind.

Figure 5.39: The grey circles give the root mean-squared error per grid cell between the true HadCM3 simulator output and the forecasted independent component reconstructions using the mixed forecasting approach. The black lines give the associated rolling average of the root mean-squared errors. The root mean-squared errors are split into those from the mean sea-level air pressure component (calculated using (5.4)), and from the wind velocity component (calculated using (5.5)).



(a) Root mean-squared error per grid cell of the mean sea-level pressure.



(b) Root mean-squared error per grid cell of the wind.

Figure 5.40: The grey circles give the root mean-squared error per grid cell between the independent component reconstructions and the forecasted independent component reconstructions using the mixed forecasting approach. The black lines give the associated rolling average of the root mean-squared errors. The root mean-squared errors are split into those from the mean sea-level air pressure component (calculated using (5.6)), and from the wind velocity component (calculated using (5.7)).

5.5.3.3 Reconstruction errors using the block-average Ornstein-Uhlenbeck forecasting approach

In this section we consider the forecasted independent component reconstructions when the block-average Ornstein-Uhlenbeck forecasting approach is used to obtain one-step-ahead forecasts for the first $l = 6$ independent components. We described these independent component forecasts in Section 5.4.2.2. Here the reconstructions are very similar to those seen in Section 5.5.3.2 and therefore the figures have been placed in Appendix B.2. These figures include:

- The forecasted independent component reconstructions using the block-average Ornstein-Uhlenbeck forecasting approach, at times $t = 1600$ and $t = 2000$, shown in Figure B.3;
- The difference between the true HadCM3 simulator output and the forecasted independent component reconstructions at times $t = 1600$ and $t = 2000$, shown in Figure B.4;
- The difference between the independent component reconstructions (using $l = 6$ independent components), and the forecasted independent component reconstructions, at times $t = 1600$ and $t = 2000$, shown in Figure B.5;
- The root mean-squared errors between the forecasted independent component reconstructions using the block-average Ornstein-Uhlenbeck forecasting approach, and: *(i)* the true HadCM3 simulator output, shown in Figure B.6, and; *(ii)* the independent component reconstructions using $l = 6$ independent components, shown in Figure B.7.

There is very little difference between the root mean-squared errors found using the mixed forecasting approach (seen in Figures 5.39 and 5.40), and found using the block-average Ornstein-Uhlenbeck forecasting approach. This suggests that the two forecasting approaches give very similar reconstructions, and the pointwise misspecification of the mixed forecasting approach does not significantly affect the resulting forecasts.

In this section we considered reconstructions of the HadCM3 simulator output using forecasted independent components found with the mixed forecasting approach and the block-average forecasting approach. As was expected from the similarities seen in the component point forecasts and prediction intervals (Section 5.4.2), both of the forecasting approaches gave similar reconstructions with similar root mean-squared errors (when compared to the true HadCM3 simulator output). We also note that the forecasted independent component reconstructions have a similar root mean squared error to the true independent component reconstructions, when these are compared to the true HadCM3 simulator output. This behaviour is the same as that found for the independent component reconstructions (Section 4.6.3) and suggests that there is very little increase in the total reconstruction error when the true independent components are replaced by forecasted components.

5.6 Concluding remarks

We conclude this chapter with some remarks on the applications of the techniques introduced in Part I on the mean sea-level air pressure with wind velocity HadCM3 simulator output. These remarks are split into those relating to the stochastic modelling of the components (as seen in Section 5.3), those relating to the component forecasts (as in Section 5.4), and relating to the forecasted reconstructions (as in Section 5.5). We will also comment on similarities and differences between the results seen in this chapter to those seen in Chapter 4.

As with the principal and independent components seen in Chapter 4, those found in this chapter also contained strong seasonality. This was shown for the principal and independent components in the plots of the autocorrelation functions, Figure 5.7 and Figure 5.14, respectively. The seasonality of the principal components here is much stronger than that of the independent components, which was also the case in Chapter 4 (and that we discussed in Section 4.7).

In this chapter we did not model the components using the sparse maximum likelihood approach as this method was shown to be deficient in the previous chapter (Section 4.4.2.2). When we applied the method of moments to the training samples of the components, we encountered the same issue with regards to non-positive sample lag-1 correlations as in the previous chapter. This is a fundamental issue with the method of moments, and resulted in us being unable to obtain drift and diffusion estimates for many of the training samples.

Modelling the principal components as realisations from block-average Ornstein-Uhlenbeck processes resulted in some non-convergence of the maximum likelihood method (as seen in Figure 5.13). This was due to numerical errors when inverting the covariance matrix. As discussed when the block-average Ornstein-Uhlenbeck maximum likelihood estimation method was introduced (in Section 2.3.3.3), we use the Cholesky decomposition to obtain the inverse and determinant of the covariance matrix, as Trench's algorithm cannot obtain the latter. Replacing this with a more efficient algorithm should help avoid numerical issues here, and this is considered as useful future research. We note that this issue did not arise to the same extent when looking at the principal and independent components obtained from the mean sea-level air pressure HadCM3 output in Chapter 4.

In the forecasting section (Section 5.4), we used the mixed forecasting approach to obtain forecasts for the principal components, and both the mixed forecasting and the block-average Ornstein-Uhlenbeck forecasting approaches to obtain forecasts for the independent components. The block-average Ornstein-Uhlenbeck forecasting approach was not used on the principal components due to the issues with the maximum likelihood estimation discussed above. The two approaches used for forecasting the independent components gave very similar point forecasts and prediction intervals (as can be seen by comparing Figures 5.22 to B.1). This was also seen for the independent components in Chapter 4. As we discussed there, future research is required here to determine the reason why the misspecification in the mixed forecasting approach did not greatly affect the forecasts.

Also, similar to the prediction intervals seen in Chapter 4, some of the intervals found here were much wider than expected. Most notably, the prediction intervals for the first principal component (Section 5.4.1), and for the second and third independent components (Section 5.4.2). The prediction intervals for the first independent component (found using both the mixed and the block-average Ornstein-Uhlenbeck forecasting approaches) appears to be too narrow, as 12% of the test sample lay outside of these approximate 95% prediction intervals. As we discussed in the previous chapter, this suggests that the models we used on these components were sub-optimal, and we recommend some further analysis on different models that can be applied here.

In the reconstruction section (Section 5.5), we saw that the root mean-squared errors for the principal component reconstructions compared to the true HadCM3 simulator output (seen in Figure 5.27) were in general smaller than for the independent component reconstructions (Figure 5.35). This is expected due to the properties of principal component analysis (discussed in Remark 4.18), and was also seen in Chapter 4.

Another similarity with the analysis in Chapter 4, is that the rolling average of the root mean-squared errors between the forecasted independent component reconstructions (using both forecasting approaches) and the HadCM3 simulator output are very similar to the rolling average of the errors between the independent component reconstructions and the HadCM3 simulator output. This can be seen be comparing Figures 5.39 (using the mixed forecasting approach) or B.6 (using the block-average Ornstein-Uhlenbeck forecasting approach) to Figure 5.35. This implies that the forecasts can successfully replace the independent components with very little additional error being introduced. This is not the case for the principal component forecasts, where the forecasted principal component reconstructions introduce additional error above the principal component reconstruction, which can be seen by comparing Figure 5.29 (where the mixed forecasting approach is used) to Figure 5.27.

Concluding Remarks

In this thesis we have attempted to find an approach to simplify the high-dimensional HadCM3 simulator output. Initial steps involved using dimension reduction techniques to obtain a basis vector of the span of the simulator output. Then, taking a selection of these basis vectors and the associated projections, we have modelled these projections using various stochastic models, assuming the information lost by only choosing a subset of the basis vectors can be replaced by some randomness. Using forecasts based on these stochastic models, one-step-ahead forecasts can then be found for the projections obtained in the dimension reduction step.

In Part II of this thesis, the theory discussed in Part I was applied to some data obtained from the HadCM3 climate simulator. Comparisons were made on the reconstructed simulator output using the various forecasts obtained from the stochastic process assumptions. In addition to this, some issues regarding applying the theoretical methods discussed in Part I came to light, most notably with respect to the sparse maximum likelihood, and the block-average Ornstein-Uhlenbeck maximum likelihood parameter estimation methods.

In these last pages, an attempt is made to tie together some of the loose strings from the previous chapters, and to give some ideas for further research stemming from this thesis. First, some comments are made on the successes and failures present in this document.

With regards to forecasting the reconstructions of the HadCM3 simulator output in Part II, using the one-step-ahead forecasts for the principal and independent components and the respective inverse methods gave reconstructions that were, in general, quite similar to the true output. We also found that when the independent components were replaced with their forecasts, the resulting reconstructions had similar errors when compared to the true HadCM3 simulator output. This suggests that these forecasts can replace the true components with minimal addition error introduced to the reconstructions. For both the principal and independent components, the misspecification of these components by pointwise stochastic models did not show up in the end result, and indeed the two forecasting methods examined produced extremely similar reconstructions. One hypothesis made here is that the stochasticity of the components has a much smaller effect on the resulting reconstruction forecasts than the deterministic seasonal effects do. Investigating the impacts of the seasonal effects and the stochastic effects on the reconstructions separately is grounds for further research here.

In Part I of this thesis, the clusterICA method introduced some interesting novel techniques for obtaining independent components. Most notably, the approach of clustering the initial directions in projective spaces could be a useful technique for improving established independent component analysis methods. This novel clustering of the search space could be added to some of these established methods to produce an independent component analysis method that results in a superior algorithm, as discussed below. However, in the algorithm's current form there are many scenarios where established independent component analysis methods are both quicker and result in better independent components (in terms of having a lower M-spacing entropy).

The introduction of block-average Ornstein-Uhlenbeck processes in this thesis was novel in the context of simulators that give some averaged output. The use of these processes (and the periodic mean functions) have many applications in modelling climate data, alongside the associated maximum likelihood parameter estimation method and the forecasting theory. However, the convergence of this maximum likelihood parameter estimation method needs further work. This was seen when we modelled the principal components obtained from the mean sea-level pressure with wind velocity HadCM3 simulator output, where some spurious estimates were found. One issue here could be due to the fact that the optimisation is performed over the full covariance matrix (as in (2.18)), which can be extremely large. The symmetric Toeplitz form of this matrix could be examined to find more efficient methods for finding its determinant, which is currently the most intensive part of the optimisation stage.

Further Research

Examining the component forecasts obtained in Part II in detail would be, I believe, fruitful further research. The hypothesis here – that the seasonal structure in the components dominant the stochastic part when considering the errors in the reconstructions – could be closely examined. It is also interesting that in general, the stochastic processes were better at modelling the independent components than they were the principal components (with regards to the forecasted reconstructions), as the latter should theoretically be better suited to the Gaussian models used in this thesis. Further work could be done on considering the seasonality of the independent components as a source for the non-Gaussianity of these components. This leads to the question: once seasonality is removed from the independent components, are the remaining residuals suitable to be modelled by Gaussian stochastic processes? If so, is this some more general behaviour of independent components or is it only applicable to our specific examples?

As noted above, one major part of any further research could be to improve the clusterICA method such that it consistently beats other established independent component analysis methods. The obvious way forward for this research would be to merge the clustering step in an established method. For example, with random directions in the initialisation, and then after clustering these random directions, optimisation using the established methods could be performed on this set of (hopefully diverse) directions. Looking at it from the other direction, another way to improve the clusterICA algorithm could be to find initial directions using the various contrast functions found in the established methods, alongside a set of random directions. Then, these directions could be put into clusters before optimising. This was briefly discussed with respect to fastICA in Section 1.4.2.5.

Finally, another avenue for further research is in the block-average Ornstein-Uhlenbeck maximum likelihood estimation. It is noted in this thesis that Trench's Algorithm (Trench, 1964) is efficient for obtaining the inverse of the associated covariance matrix. However, the determinant can not be obtained in this way, and we have not been able to find similar methods to do so. This lack of an efficient method for finding the determinant slows the optimisation step considerably and potentially results in the numerical issues that were seen in Section 5.3.1.3.

Research into whether it is possible to exploit the symmetrical Toeplitz structure of the covariance matrix associated with the block-average Ornstein-Uhlenbeck process would be extremely useful here.

Appendices

Appendix A

Figures from Chapter 4

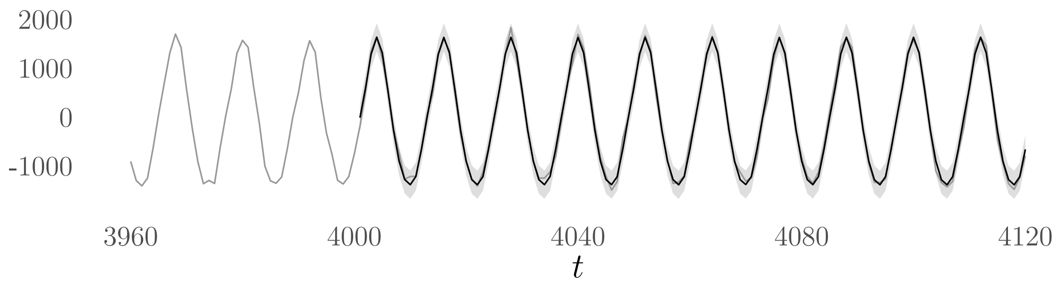
This appendix includes additional figures from Chapter 4 that are omitted from the main body of the thesis to aid readability and conciseness. They are included here for completeness.

A.1 Forecasting

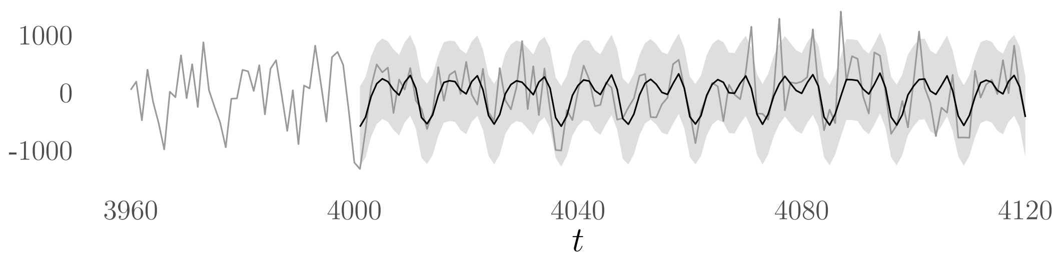
In this section we show the additional figures that are referenced in Section 4.5. These figures all relate to the block-average Ornstein-Uhlenbeck forecasting approach, and are excluded from the main body of the thesis due to them being very similar to equivalent figures obtained when the mixed forecasting approach was used.

In this section, the block-average Ornstein-Uhlenbeck forecasts for the first three principal components are shown in Figure A.1. Here, the point forecasts are given by the black line, with the grey shading giving the approximate 95% prediction intervals. In Figure A.2 we show the rolling average of the root mean-squared errors between the true principal components and the one-step-ahead point forecasts obtained using the block-average Ornstein-Uhlenbeck forecasting approach.

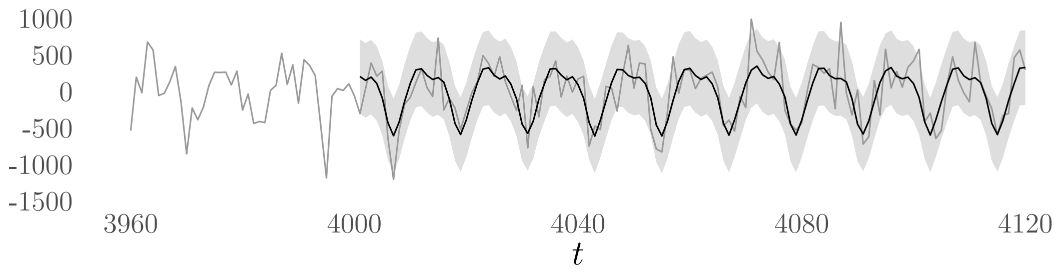
In Figure A.3 and Figure A.4 we show the block-average Ornstein-Uhlenbeck forecasts for the first three independent components, and the rolling average root mean-squared error between the true independent components and these point forecasts, respectively.



(a) First principal component for $t = 3960, \dots, 4120$ (grey solid line). The one-step-ahead point forecast found is given by the black line, with the 95% prediction interval given by the shaded grey area.



(b) Second principal component for $t = 3960, \dots, 4120$ (grey solid line). The one-step-ahead point forecast is given by the black line, with the 95% prediction interval given by the shaded grey area.



(c) Third principal component for $t = 3960, \dots, 4120$ (grey solid line). The one-step-ahead point forecast is given by the black line, with the 95% prediction interval given by the shaded grey area.

Figure A.1: First three principal components for time-steps $t = 3960, \dots, 4120$ (grey solid line). The one-step-ahead point forecasts found using the block-average Ornstein-Uhlenbeck forecasting approach are given by the black lines, with the approximate 95% prediction intervals given by the shaded grey areas.

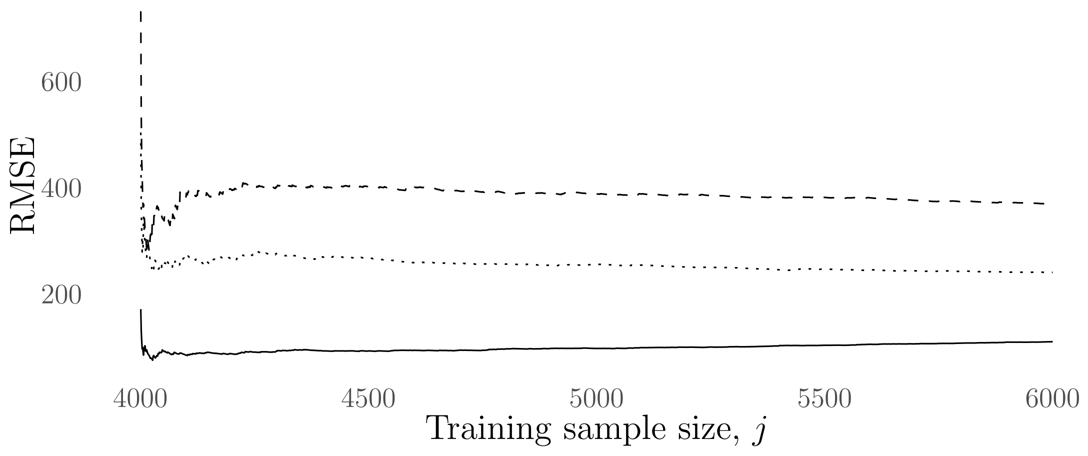
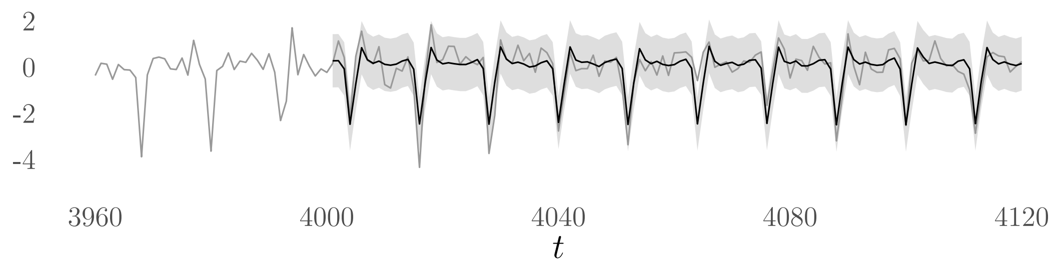
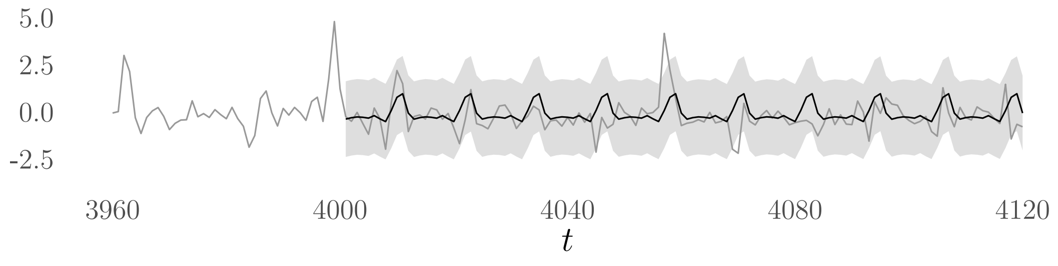


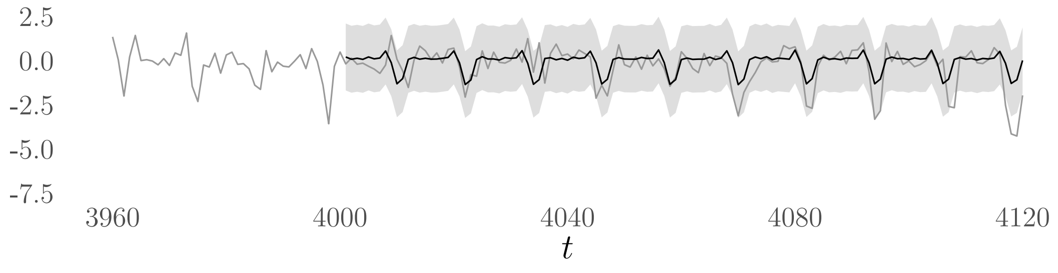
Figure A.2: Rolling average of the root mean-squared error between the true principal components and the respective point forecasts, found using the block-average Ornstein-Uhlenbeck forecasting approach. The rolling averages of the root mean-squared error between the first, second and third principal components, and the associated one-step ahead forecasts are given by the solid, dashed and dotted lines respectively.



(a) First independent component for $t = 3960, \ldots, 4120$ (grey solid line). The one-step-ahead point forecast found using the block-average Ornstein-Uhlenbeck forecasting approach is given by the solid black line, with the 95% prediction interval given by the shaded grey area.



(b) Second independent component for $t = 3960, \ldots, 4120$ (grey solid line). The one-step-ahead point forecast found using the block-average Ornstein-Uhlenbeck forecasting approach is given by the solid black line, with the 95% prediction interval given by the shaded grey area.



(c) Third independent component for $t = 3960, \ldots, 4120$ (grey solid line). The one-step-ahead point forecast found using the block-average Ornstein-Uhlenbeck forecasting approach is given by the solid black line, with the 95% prediction interval given by the shaded grey area.

Figure A.3: First three independent components for time-steps $t = 3960, \ldots, 4120$ (grey solid line). The one-step-ahead point forecast found using the block-average Ornstein-Uhlenbeck forecasting approach is given by the solid black line, with the approximate 95% prediction interval given by the shaded grey area.

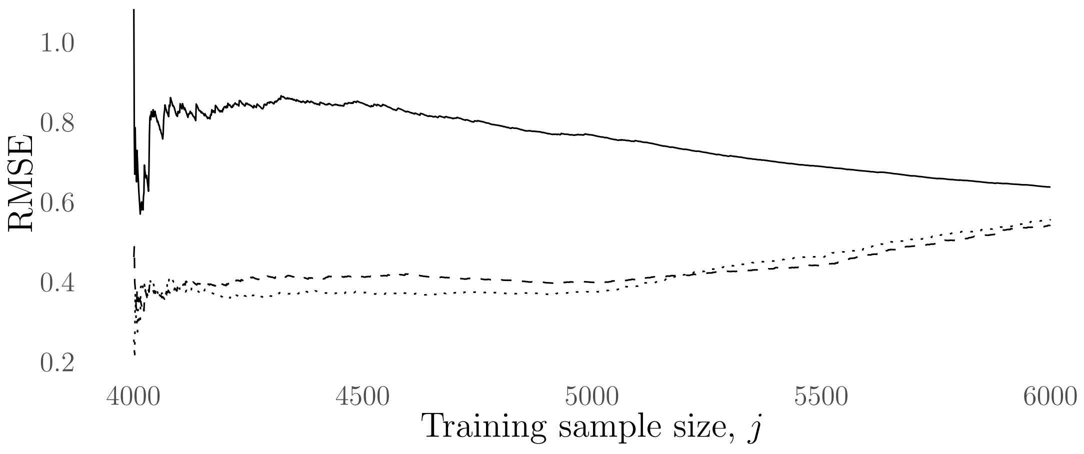


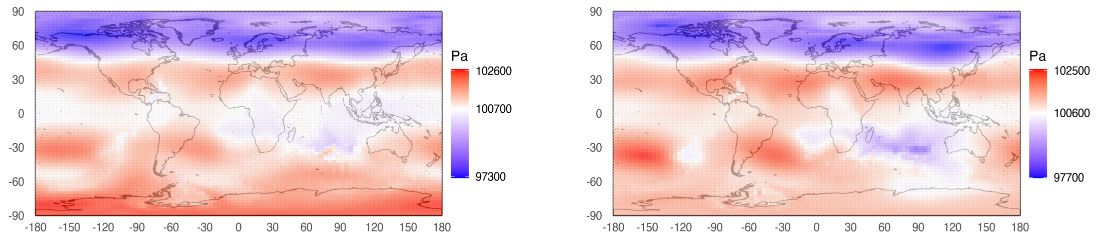
Figure A.4: Rolling average of the root mean-squared error between the true independent components and the respective point forecasts, found using the block-average Ornstein-Uhlenbeck forecasting approach. The rolling averages of the root mean-squared error related to the first, second and third independent components are given by the solid, dashed and dotted lines respectively.

A.2 Reconstructing

In this section we include figures that are omitted from Section 4.6. These figures are all related to the reconstructions obtained using the block-average Ornstein-Uhlenbeck forecast approach, and are very similar to the equivalent figures obtained using the mixed forecasting approach (and which are included in the main body of the thesis).

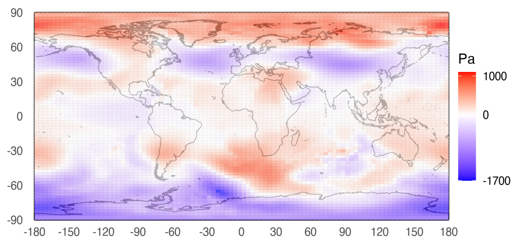
In Figure A.5 we show the forecasted principal component reconstruction of the mean sea-level pressure HadCM3 simulator output when the block-average Ornstein-Uhlenbeck forecasting approach was used. Figure A.6 shows the difference between the HadCM3 simulator output and these forecasted reconstructions, and Figure A.7 shows the difference between the principal component reconstructions and the forecasted principal component reconstructions. In Figure A.8a and Figure A.8b we show the root mean-squared error between the forecasted principal component reconstructions and: *(i)* the HadCM3 simulator output, and; *(ii)* the principal component reconstructions, respectively.

The equivalent reconstruction plots when the block-average Ornstein-Uhlenbeck forecasting approach was used to obtain forecasted independent component reconstructions are shown in Figure A.9, A.10 and A.11. The root mean-squared errors are shown in Figure A.12.

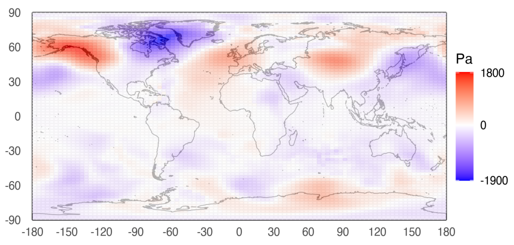


(a) Time, $t = 4000$. (b) Time, $t = 5000$.

Figure A.5: Forecasted principal component reconstruction of the mean sea-level air pressure HadCM3 simulator output using the block-average Ornstein-Uhlenbeck forecasting approach with $l = 6$ principal components, at times $t = 4000$ and $t = 5000$.

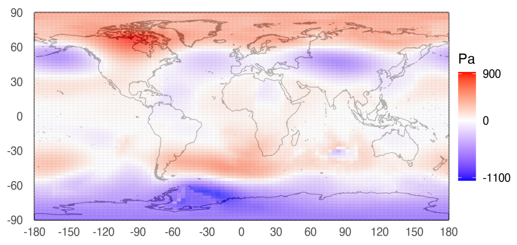


(a) Time, $t = 4000$.

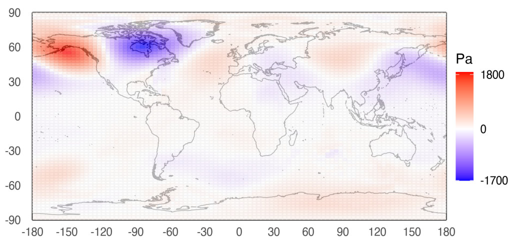


(b) Time, $t = 5000$.

Figure A.6: Differences between the true HadCM3 simulator output, and the forecasted principal component reconstructions using block-average Ornstein-Uhlenbeck forecasting approach with $l = 6$ principal components, at times $t = 4000$ and $t = 5000$.

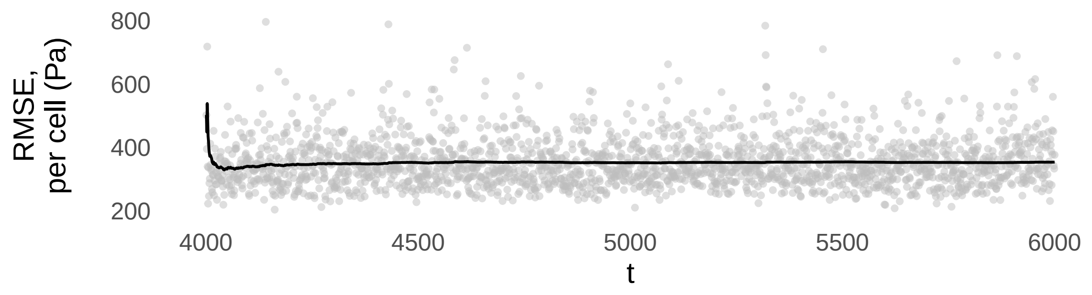


(a) Time, $t = 4000$.

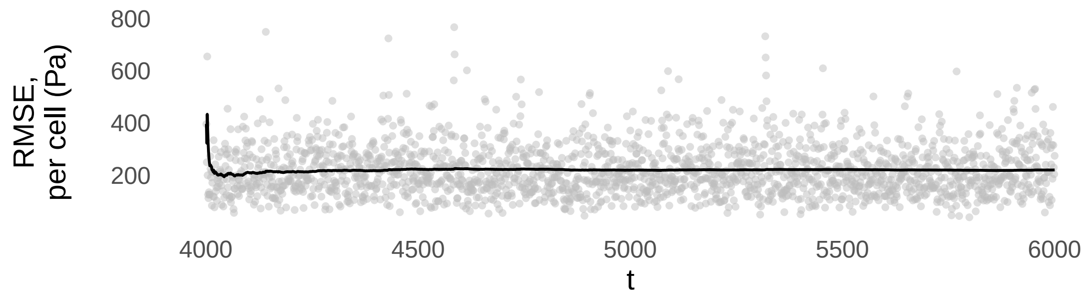


(b) Time, $t = 5000$.

Figure A.7: Differences between the principal component reconstructions with $l = 6$ principal components, and the forecasted principal component reconstructions using the block-average Ornstein-Uhlenbeck forecasting approach, at times $t = 4000$ and $t = 5000$.

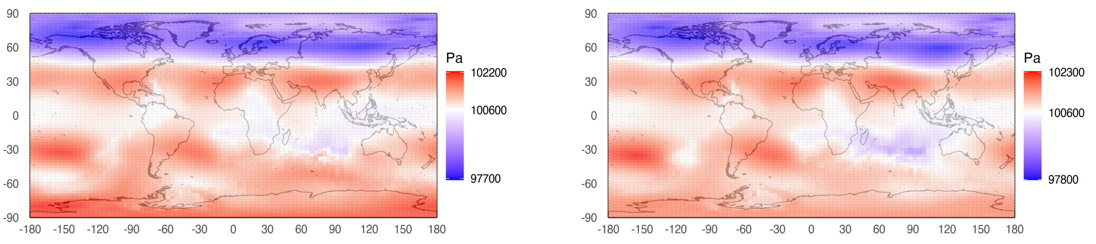


(a) Root mean-squared error per grid cell between the actual HadCM3 simulator output and forecasted principal component reconstructions.



(b) Root mean-squared error per grid cell between the principal component reconstructions and the forecasted principal component reconstructions.

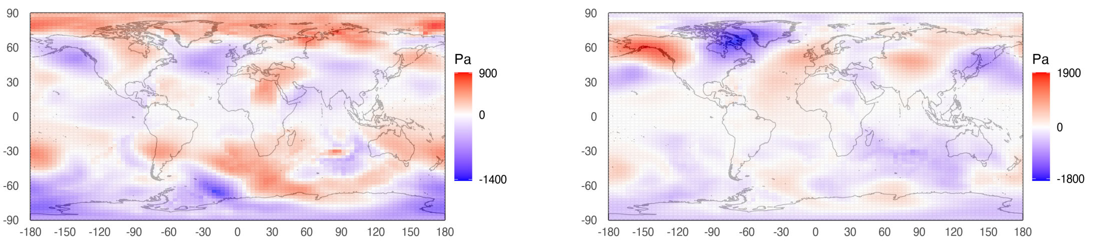
Figure A.8: The grey circles give the root mean-squared error per grid cell (calculated using (4.10) and (4.11)) between forecasted principal component reconstructions using the block-average Ornstein-Uhlenbeck forecasting approach for $l = 6$ principal components, and *(i)* the actual mean sea-level air pressure HadCM3 simulator output (Figure A.8a), and; *(ii)* the principal component reconstructions using $l = 6$ principal components (Figure A.8b). The black lines give the running average of the root mean-squared errors over time.



(a) Time, $t = 4000$.

(b) Time, $t = 5000$.

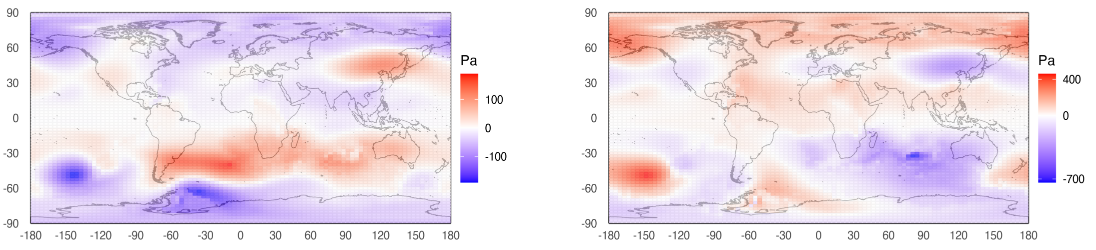
Figure A.9: Forecasted independent component reconstructions of the mean sea-level air pressure HadCM3 simulator output using the block-average Ornstein-Uhlenbeck forecasting approach with $l = 6$ independent components, at times $t = 4000$ and $t = 5000$.



(a) Time, $t = 4000$.

(b) Time, $t = 5000$.

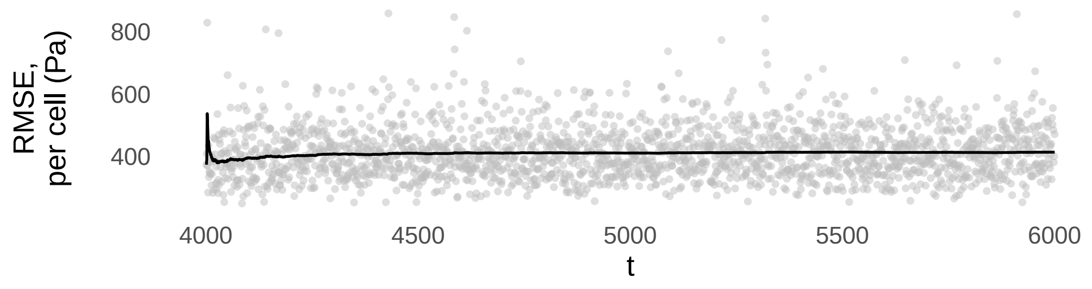
Figure A.10: Difference between the true mean sea-level air pressure HadCM3 simulator output and the forecasted independent component reconstructions using the block-average Ornstein-Uhlenbeck forecasting approach, at times $t = 4000, 5000$.



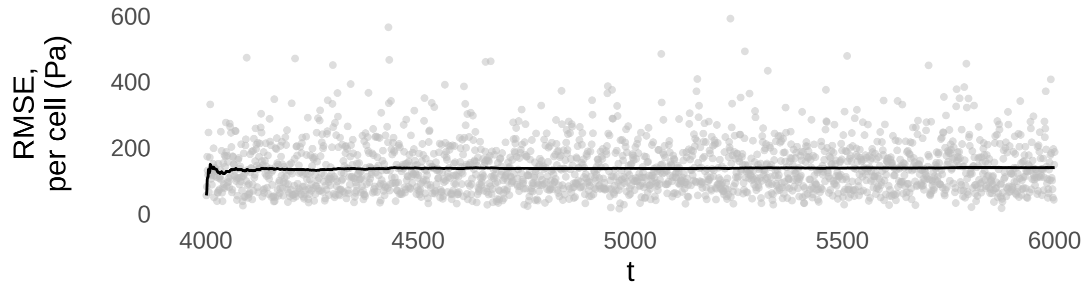
(a) Time, $t = 4000$.

(b) Time, $t = 4000$.

Figure A.11: Difference between the independent component reconstructions of the mean sea-level air pressure (with $l = 6$ independent components), and the forecasted independent component reconstructions using the block-average Ornstein-Uhlenbeck forecasting approach, at times $t = 4000, 5000$.



(a) Root mean-squared errors per grid cell between the forecasted independent component reconstructions using the block-average Ornstein-Uhlenbeck forecasting approach, and the true HadCM3 simulator output.



(b) Root mean-squared errors per grid cell between the forecasted independent component reconstructions using the block-average Ornstein-Uhlenbeck forecasting approach, and the (true) independent component reconstructions with $l = 6$ independent components.

Figure A.12: The grey circles give the root mean-squared errors per grid cell between the forecasted independent component reconstructions found using the block-average Ornstein-Uhlenbeck forecasting approach (with $l = 6$ independent components), and: *(i)* the true HadCM3 simulator output (Figure A.12a); *(ii)* the independent component reconstructions using $l = 6$ independent components (Figure A.12b). The black lines give the rolling average of root mean-squared errors over time.

Appendix B

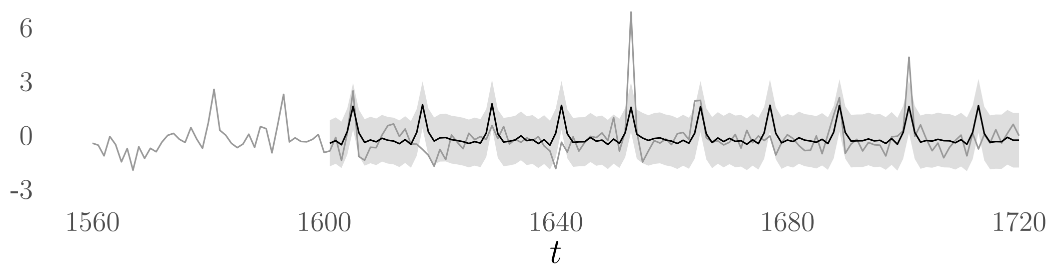
Figures from Chapter 5

This appendix includes additional figures from Chapter 5 that are omitted from the main body of the thesis to aid readability and conciseness. They are included here for completeness.

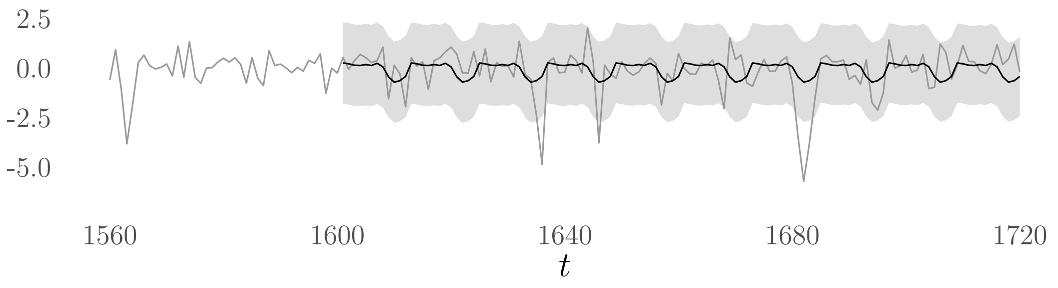
B.1 Forecasting

In this section we show the additional figures that are referenced in Section 5.4. These figures all relate to the block-average Ornstein-Uhlenbeck forecasting approach, and are excluded from the main body of the thesis due to them being very similar to equivalent figures obtained when the mixed forecasting approach was used.

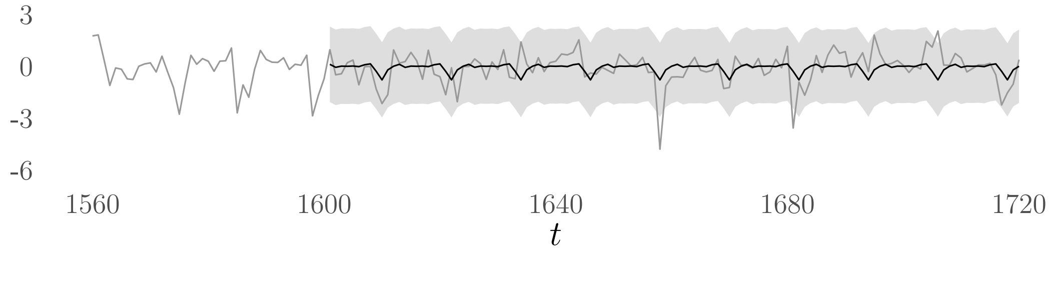
FIgure B.1 shows the block-average Ornstein-Uhlenbeck forecasts for the first three independent components. The point forecasts are given by the black line, with the grey shaded area giving the approximate 95% prediction intervals. The rolling average of the root mean-squared errors between these point forecasts and the true independent components are shown in Figure B.2.



(a) First independent component for $t = 1560, \ldots, 1720$ (grey solid line). The one-step-ahead point forecast for is given by the solid black line, with the 95% prediction interval given by the shaded grey area.



(b) Second independent component for $t = 1560, \ldots, 1720$ (grey solid line). The one-step-ahead point forecast for is given by the solid black line, with the 95% prediction interval given by the shaded grey area.



(c) Third independent component for $t = 1560, \ldots, 1720$ (grey solid line). The one-step-ahead point forecast for is given by the solid black line, with the 95% prediction interval given by the shaded grey area.

Figure B.1: First, second and third independent components plotted as a time series for time $t = 1560, \ldots, 1720$ (grey solid line). The one-step-ahead point forecast (black line) is found using the block-average Ornstein-Uhlenbeck forecasting approach, with shaded grey area showing the 95% prediction interval.

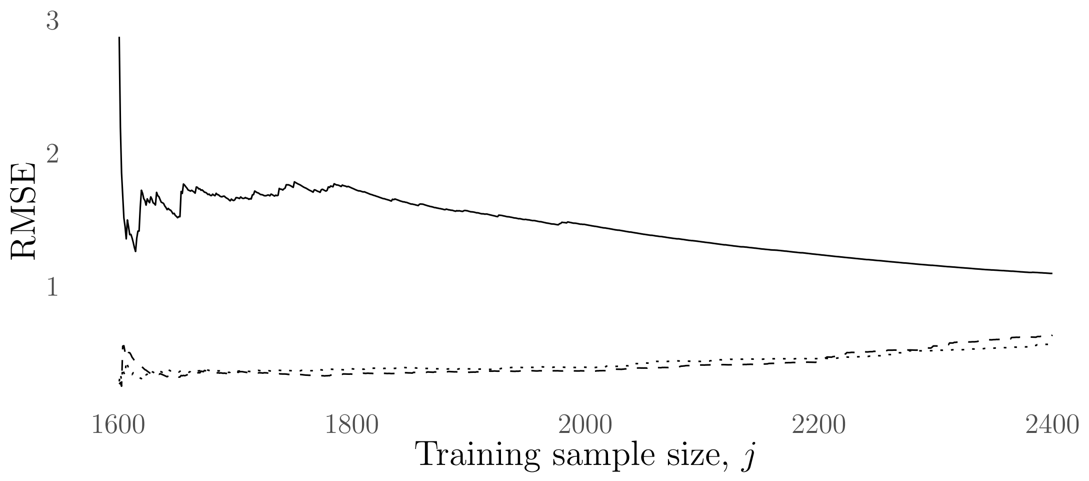
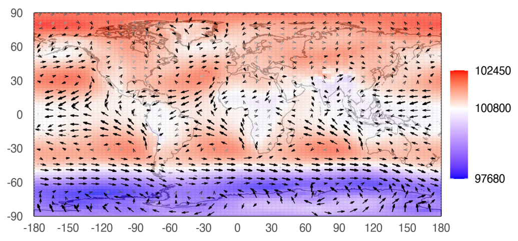


Figure B.2: Rolling average of the root mean-squared error between the true independent components and the respective point forecasts using the block-average Ornstein-Uhlenbeck forecasting approach The rolling average of the root mean-squared errors between the first, second and third independent component and the associated one-step ahead forecasts are given by the solid, dashed and dotted lines respectively.

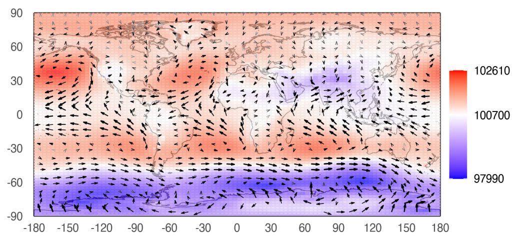
B.2 Reconstructing

In this section we include figures that are omitted from Section 5.5. These figures are all related to the reconstructions obtained using the block-average Ornstein-Uhlenbeck forecast approach, and are very similar to the equivalent figures obtained using the mixed forecasting approach (and which are included in the main body of the thesis).

In Figure B.3 we show the forecasted independent component reconstruction of the mean sea-level pressure with wind HadCM3 simulator output when the block-average Ornstein-Uhlenbeck forecasting approach was used. Figure B.4 shows the difference between the HadCM3 simulator output and these forecasted reconstructions, and Figure B.5 shows the difference between the independent component reconstructions and the forecasted independent component reconstructions. In Figure B.6 and Figure B.7 we show the root mean-squared error between the forecasted independent component reconstructions and: *(i)* the HadCM3 simulator output, and; *(ii)* the independent component reconstructions, respectively.

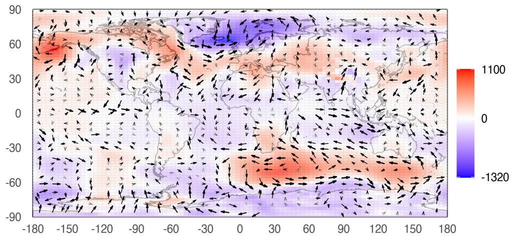


(a) Time, $t = 1600$.

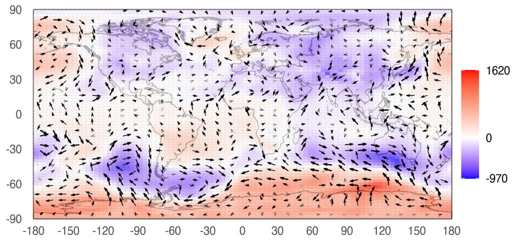


(b) Time, $t = 2000$.

Figure B.3: Forecasted independent component reconstructions using the block-average Ornstein-Uhlenbeck forecasting approach, at times $t = 1600, 2000$.

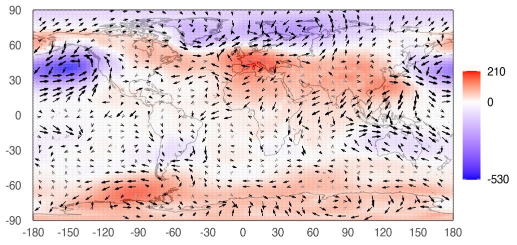


(a) Time, $t = 1600$.

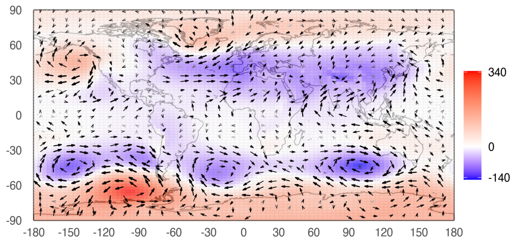


(b) Time, $t = 2000$.

Figure B.4: Difference between the true mean sea-level air pressure with wind velocity HadCM3 simulator output and the forecasted independent component reconstructions using the block-average Ornstein-Uhlenbeck forecasting approach, at times $t = 1600, 2000$.

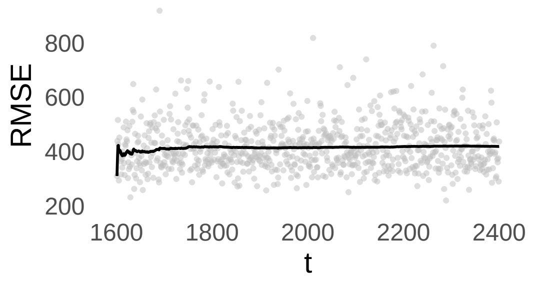


(a) Time, $t = 1600$.

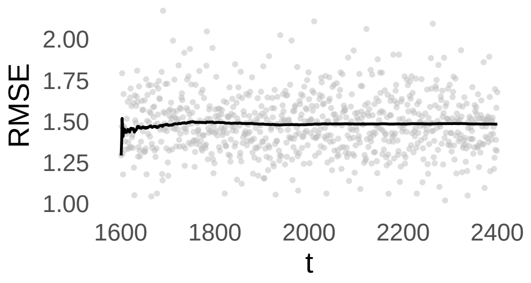


(b) Time, $t = 2000$.

Figure B.5: Difference between the independent component reconstruction of the mean sea-level air pressure with wind velocity, and the forecasted independent component reconstructions using the block-average Ornstein-Uhlenbeck forecasting approach, at times $t = 1600, 2000$.

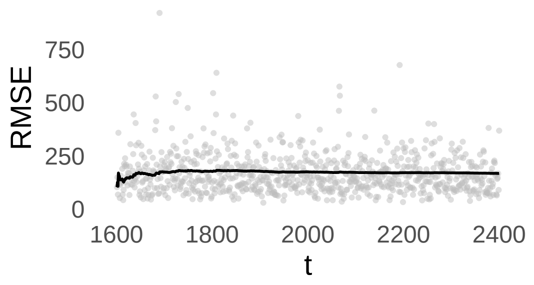


(a) Root mean-squared error per grid cell of the mean sea-level pressure.

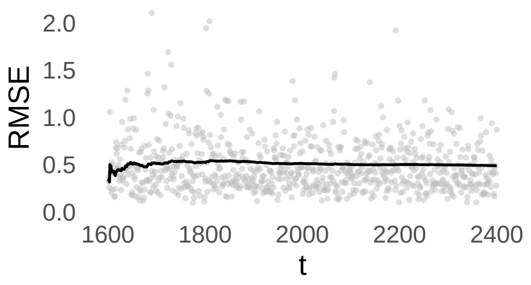


(b) Root mean-squared error per grid cell of the wind.

Figure B.6: The grey circles give the root mean-squared error per grid cell between the true HadCM3 simulator output and the forecasted independent component reconstructions using the block-average Ornstein-Uhlenbeck forecasting approach. The black lines give the associated rolling average of the root mean-squared errors. The root mean-squared errors are split into those from the mean sea-level air pressure component (calculated using (5.4)), and from the wind velocity component (calculated using (5.5)).



(a) Root mean-squared error per grid cell of the mean sea-level pressure.



(b) Root mean-squared error per grid cell of the wind.

Figure B.7: The grey circles give the root mean-squared error per grid cell between the independent component reconstructions (using $l = 6$ independent components), and the forecasted independent component reconstructions using the block-average Ornstein-Uhlenbeck forecasting approach. The black lines give the associated rolling average of the root mean-squared errors. The root mean-squared errors are split into those from the mean sea-level air pressure component (calculated using (5.6)), and from the wind velocity component (calculated using (5.7)).

References

AIRES, F., CHÉDIN, A. & NADAL, J.P. (2000). Independent component analysis of multivariate time series: Application to the tropical sst variability. *Journal of Geophysical Research: Atmospheres*, **105**, 17437–17455. 22, 24

AKAIKE, H. (1969). Fitting autoregressive models for prediction. *Annals of the institute of Statistical Mathematics*, **21**, 243–247. 177

AMARI, S., CICHOCKI, A. & YANG, H. (1996). A new learning algorithm for blind signal separation. In *Advances in neural information processing systems*, 757–763. 39, 76

ARTHUR, D. & VASSILVITSKII, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 1027–1035, Society for Industrial and Applied Mathematics. 79

BACH, F. & JORDAN, M. (2002). Kernel independent component analysis. *Journal of machine learning research*, **3**, 1–48. 39, 96, 99

BARTHELET, P., TERRAY, L. & VALCKE, S. (1998). Transient co2 experiment using the arpege/opaice non flux corrected coupled model. *Geophysical Research Letters*, **25**, 2277–2280. 212

BEIRLANT, J., DUDEWICZ, E., GYÖRFI, L. & VAN DER MEULEN, E. (1997). Nonparametric entropy estimation: An overview. *International Journal of Mathematical and Statistical Sciences*, **6**, 17–39. 30, 70

BELL, A. & SEJNOWSKI, T. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, **7**, 1129–1159. 96

BERGER, M. (1977). *Nonlinearity and functional analysis: lectures on nonlinear problems in mathematical analysis*, vol. 74. Academic press. 48

BILLINGSLEY, P. (1961). Statistical methods in markov chains. *The Annals of Mathematical Statistics*, 12–40. 142

BISHWAL, J. (2007). *Parameter estimation in stochastic differential equations*. Springer. 143

BOX, G. & JENKINS, G. (1976). *Time series analysis: forecasting and control*. 108, 109, 110, 114, 117, 183

BOX, G., JENKINS, G., REINSEL, G. & LJUNG, G. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons. 178, 179, 180, 186

BROCKWELL, P., DAVIS, R. & FIENBERG, S. (1991). *Time series: theory and methods: theory and methods*. Springer Science & Business Media. 180

CARDOSO, J.F. & SOULOUMIAC, A. (1993). Blind beamforming for non-gaussian signals. In *IEE proceedings F (radar and signal processing)*, vol. 140, 362–370, IET. 96

CATTLE, H. & CROSSLEY, J. (1995). Modelling arctic climate change. *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences*, **352**, 201–213. 213

CHATFIELD, C. (2000). *Time-series forecasting*. Chapman and Hall/CRC. 114, 175, 176, 177, 189, 193

CHATFIELD, C. (2016). *The analysis of time series: an introduction*. CRC press. 106

COMON, P. (1994). Independent component analysis, a new concept? *Signal processing*, **36**, 287–314. 21, 23, 24, 26

Connolley, W. & Cattle, H. (1994). The antarctic climate of the ukmo unified model. *Antarctic Science*, **6**, 115–122. 212

Cover, T. & Thomas, J. (2012). *Elements of information theory*. John Wiley & Sons. 25

Cox, M. (1984). A primitive equation, 3-dimensional model of the ocean. *GFDL Ocean Group Technical Report No 1, GFDL, Princeton University*. 213

Cullen, M. (1993). The unified forecast/climate model. *Meteorological Magazine*, **122**, 81–94. 212

Da Prato, G. & Zabczyk, J. (2014). *Stochastic equations in infinite dimensions*. Cambridge university press. 122, 126

Dacunha-Castelle, D. & Florens-Zmirou, D. (1986). Estimation of the coefficients of a diffusion from discrete observations. *Stochastics: An International Journal of Probability and Stochastic Processes*, **19**, 263–284. 142

de Oliveira, O. (2014). The implicit and the inverse function theorems: Easy proofs. *Real Analysis Exchange*, **39**, 207–218. 52

Diaconis, P. & Freedman, D. (1984). Asymptotics of graphical projection pursuit. *The annals of statistics*, 793–815. 38, 45, 92

Draper, B., Baek, K., Bartlett, M. & Beveridge, J. (2003). Recognizing faces with PCA and ICA. *Computer vision and image understanding*, **91**, 115–137. 39

Durrett, R. (2018). *Stochastic calculus: a practical introduction*. CRC press. 141

Eaton, M. (1983). Multivariate statistics: a vector space approach. *John Wiley & Sons, inc.*. 188

Evans, L. (1998). *Partial Differential Equations*. American Mathematical Society. 47

Farhat, M., Gritli, Y. & Benrejeb, M. (2017). Fast-ICA for mechanical fault detection and identification in electromechanical systems for wind turbine applications. *International Journal of Advanced Computer Science and Applications (IJACSA)*, **8**, 431–439. 39

Gauss, K. & Pesic, P. (2005). *General investigations of curved surfaces*. Courier Corporation. 214

Ghaffarian, S. & Ghaffarian, S. (2014). Automatic building detection based on purposive fastICA (PFICA) algorithm using monocular high resolution google earth images. *ISPRS Journal of Photogrammetry and Remote Sensing*, **97**, 152–159. 39

Godfrey, L. (1978). Testing against general autoregressive and moving average error models when the regressors include lagged dependent variables. *Econometrica: Journal of the Econometric Society*, 1293–1301. 180

Golub, G. & Van Loan, C. (2012). *Matrix computations*, vol. 3. JHU press. 156

Gordon, C., Cooper, C., Senior, C., Banks, H., Gregory, J., Johns, T., Mitchell, J. & Wood, R. (2000). The simulation of sst, sea ice extents and ocean heat transports in a version of the hadley centre coupled model without flux adjustments. *Climate dynamics*, **16**, 147–168. 212

Hall, P. (1984). Limit theorems for sums of general functions of m-spacings. In *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 96, 517–532, Cambridge University Press. 31

Hall, P. & Hosseini-Nasab, M. (2006). On properties of functional principal components analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **68**, 109–126. 18, 20

He, X., He, F. & Zhu, T. (2017). Large-scale super-Gaussian sources separation using fast-ICA with rational nonlinearities. *International Journal of Adaptive Control and Signal Processing*, **31**, 379–397. 39

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, **24**, 417. 12

Householder, A. (1958). Unitary triangularization of a nonsymmetric matrix. *Journal of the ACM (JACM)*, **5**, 339–342. 86

Hyvärinen, A. (1998). New approximations of differential entropy for independent component analysis and projection pursuit. 40, 41, 42

Hyvärinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE transactions on Neural Networks*, **10**, 626–634. 21

Hyvärinen, A. & Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural networks*, **13**, 411–430. 39, 40, 41, 42, 93

Hyvärinen, A., Karhunen, J. & Oja, E. (2004). *Independent component analysis*, vol. 46. John Wiley & Sons. 21

Iacus, S. (2009). *Simulation and inference for stochastic differential equations: with R examples*. Springer Science & Business Media. 122

Ilin, A., Valpola, H. & Oja, E. (2006). Exploratory analysis of climate data using source separation methods. *Neural Networks*, **19**, 155–167. 22, 23

Jackson, J. (2005). *A user's guide to principal components*, vol. 587. John Wiley & Sons. 12

Jenkins, G. & Watts, D. (1968). Spectral analysis and its applications. 110

Jutten, C. & Herault, J. (1991). Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal processing*, **24**, 1–10. 21

Karatzsas, I. & Shreve, S. (1988). Brownian motion and stochastic calculus. *Graduate texts in Mathematics*, **113**. 124

Kessler, M., Lindner, A. & Sorensen, M. (2012). *Statistical methods for stochastic differential equations*. Chapman and Hall/CRC. 106

KLOEDEN, P. & PLATEN, E. (2013). *Numerical solution of stochastic differential equations*, vol. 23. Springer Science & Business Media. 133

KULLBACK, S. & LEIBLER, R. (1951). On information and sufficiency. *The annals of mathematical statistics*, **22**, 79–86. 24

KUTOYANTS, Y. (2013). *Statistical inference for ergodic diffusion processes*. Springer Science & Business Media. 140

LEARNED-MILLER, E. & FISHER III, J. (2003). ICA using spacings estimates of entropy. *Journal of machine learning research*, **4**, 1271–1295. 31, 39, 70, 76

LINDSEY, J. (2004). *Statistical analysis of stochastic processes in time*, vol. 14. Cambridge University Press. 122, 125

LIPSTER, R. & SHIRYAEV, A. (1977). *Statistics of Random Processes: General Theory*. 141

LJUNG, G. & BOX, G. (1978). On a measure of lack of fit in time series models. *Biometrika*, **65**, 297–303. 180

LLOYD, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, **28**, 129–137. 78

LORD, G., POWELL, C. & SHARDLOW, T. (2014). *An introduction to computational stochastic PDEs*. 50, Cambridge University Press. 19

LOTSCH, A., FRIEDL, M.A. & PINZÓN, J. (2003). Spatio-temporal deconvolution of ndvi image sequences using independent component analysis. *IEEE Transactions on Geoscience and Remote sensing*, **41**, 2938–2942. 22

MANLY, B. (1994). *Multivariate statistical methods: a primer*. CRC Press. 12

MARCHINI, J., HEATON, C. & RIPLEY, B. (2013). *fastICA: FastICA Algorithms to perform ICA and Projection Pursuit*. R package version 1.2-0. 30, 40, 96

MARDIA, K., KENT, J. & BIBBY, J. (1979). *Multivariate analysis*. Academic Press. 12, 13, 78

McLeod, A., Yu, H. & Krougly, Z. (2007). Algorithms for linear time series analysis: With r package. *Journal of Statistical Software*, **23**. 156

Miettinen, J., Nordhausen, K., Oja, H. & Taskinen, S. (2014). Deflation-based fastICA with adaptive choices of nonlinearities. *IEEE Transactions on Signal Processing*, **62**, 5716–5724. 39

Miettinen, J., Taskinen, S., Nordhausen, K., Oja, H. *et al.* (2015). Fourth moments and independent component analysis. *Statistical science*, **30**, 372–390. 70, 71, 72

Nathaniel, E. (2018). *ica: Independent Component Analysis*. R package version 1.0-2. 96, 99

Øksendal, B. (2013). *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media. 104, 106, 122, 123

Pachauri, R. & Reisinger, A. (2008). Climate change 2007. synthesis report. contribution of working groups i, ii and iii to the fourth assessment report. *Cambridge University Press, Cambridge*. 213

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, **2**, 559–572. 12

Pope, V., Gallani, M., Rowntree, P. & Stratton, R. (2000). The impact of new physical parametrizations in the hadley centre climate model: Hadam3. *Climate Dynamics*, **16**, 123–146. 213

Priestley, M. (1981). *Spectral analysis and time series: probability and mathematical statistics*. 04; QA280, P7. 106, 177

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. 16

Rajput, B. & Cambanis, S. (1972). Gaussian processes and gaussian measures. *The Annals of Mathematical Statistics*, 1944–1952. 126, 127

Richman, M. (1986). Rotation of principal components. *Journal of climatology*, **6**, 293–335. 21, 245

Rodgers, J., Nicewander, W. & Toothaker, L. (1984). Linearly independent, orthogonal, and uncorrelated variables. *The American Statistician*, **38**, 133–134. 23

Rougier, J., Goldstein, M. & House, L. (2013). Second-order exchangeability analysis for multimodel ensembles. *Journal of the American Statistical Association*, **108**, 852–863. 212

Royston, P. (1995). Remark as r94: A remark on algorithm as 181: The w-test for normality. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, **44**, 547–551. 184

Shapiro, S. & Wilk, M. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, **52**, 591–611. 180, 184

Smith, P., Voss, J. & Issoglio, E. (2020). On the estimation of entropy in the fastICA algorithm. *Journal of Multivariate Statistics*. 11, 39

Stone, J. (2004). *Independent component analysis: a tutorial introduction*. MIT press. 21

Stratonovich, R. (1966). A new representation for stochastic integrals and equations. *SIAM Journal on Control*, **4**, 362–371. 143

Taylor, J., Cumberland, W. & Sy, J. (1994). A stochastic model for analysis of longitudinal aids data. *Journal of the American Statistical Association*, **89**, 727–736. 125

Trench, W. (1964). An algorithm for the inversion of finite toeplitz matrices. *Journal of the Society for Industrial and Applied Mathematics*, **12**, 515–522. 156, 343

Uhlenbeck, G. & Ornstein, L. (1930). On the theory of the brownian motion. *Physical review*, **36**, 823. 124

Vasicek, O. (1976). A test for normality based on sample entropy. *Journal of the Royal Statistical Society. Series B (Methodological)*, 54–59. 30, 70

von Weizsäcker, H. (1997). Sudakov's typical marginals, random linear functionals and a conditional central limit theorem. *Probability theory and related fields*, **107**, 313–324. 38, 45, 92

Watson, R., Albritton, D. & Dokken, D. (2001). *Climate change 2001: synthesis report*. Cambridge University Press Cambridge, UK. 213

Wei, T. (2014). On the spurious solutions of the fastICA algorithm. In *Statistical Signal Processing (SSP), 2014 IEEE Workshop on*, 161–164, IEEE. 40, 41

Wei, T. (2017). A study of the fixed points and spurious solutions of the deflation-based fastica algorithm. *Neural Computing and Applications*, **28**, 13–24. 40

Yang, C.H., Shih, Y.H. & Chiueh, H. (2015). An 81.6 μW fastICA processor for epileptic seizure detection. *IEEE transactions on biomedical circuits and systems*, **9**, 60–71. 39