The University of Reading
Department of Computer Science

Fast retrieval of weather analogues in a multi-petabyte meteorological archive

PhD

Baudouin Raoult

February 2020

Declaration

I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

Baudouin Raoult

Preface

Why embark on this PhD so late in life? I have always been passionate about Computer Science, and I did consider a research career on that subject. But then life decided otherwise, and I had the privilege of working for ECMWF, first as a software developer and then as a software architect, designing and implementing systems that would serve the meteorological community. One of this system is called MARS, and is the largest archive of weather data in the world, holding several hundreds of petabytes, and delivering data to thousands of users every day. Such developments were very fulfilling, but I still had research in the back of my mind. Then came a moment when all my children had left home, and I suddenly had a lot more time. I therefore decided to do this PhD, with the idea of combining my work experience with everything I would learn. And here I am.

Acknowledgements

There are many people I would like to thank and I genuinely hope that I did not forget anyone: first of all, my supervisor Dr Giuseppe Di Fatta for his support throughout this PhD; Dr Florence Rabier for giving me the opportunity to start this endeavour; Dr Erik Anderson and Dr François Bouttier for helping me in my application to the University of Reading; Dr Jean-Baptiste Filippi for putting the idea of searching for analogues in the MARS archive in my mind; Prof Bryan Lawrence for introducing me to Zotero, this was a life changer when it came to managing citations; Prof Florian Pappenberger and Dr Tiago Quintino for listening to me over coffee and giving me useful pointers; and of course my wife Christine, and my children Nina, Camille, Justine and Maxime for providing moral and logistical support, as well as proofreading this document.

Abstract

The European Centre for Medium-Range Weather Forecasts (ECMWF) manages the largest archive of meteorological data in the world. At the time of writing, it holds around 300 petabytes and grows at a rate of 1 petabyte per week. This archive is now mature, and contains valuable datasets such as several reanalyses, providing a consistent view of the weather over several decades.

Weather analogue is the term used by meteorologists to refer to similar weather situations. Looking for analogues in an archive using a brute force approach requires data to be retrieved from tape and then compared to a user-provided weather pattern, using a chosen similarity measure. Such an operation would be very long and costly.

In this work, a wavelet-based fingerprinting scheme is proposed to index all weather patterns from the archive, over a selected geographical domain. The system answers search queries by computing the fingerprint of the query pattern and looking for close matched in the index. Searches are fast enough that they are perceived as being instantaneous.

A web-based application is provided, allowing users to express their queries interactively in a friendly and straightforward manner by sketching weather patterns directly in their web browser. Matching results are then presented as a series of weather maps, labelled with the date and time at which they occur.

The system has been deployed as part of the *Copernicus Climate Data Store* and allows the retrieval of weather analogues from *ERA5*, a 40-years hourly reanalysis dataset.

Some preliminary results of this work have been presented at the *International Conference on Computational Science 2018* (Raoult et al. (2018)).

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Aim of the research 2

1.3 Goals 4

1.3.1 Fingerprinting 4

1.3.2 Query by example 5

1.4 Preliminary work 5

1.4.1 Space partitioning 5

1.4.2 Space-filling curves 6

1.4.3 Dimensionality reduction 7

1.5 Related work 7

1.5.1 Weather analogues 7

1.5.2 Wavelet-based retrieval systems 8

1.5.3 Query by example 10

1.5.4 Input of meteorological fields 10

1.6 Contributions 11

2 Meteorological data **13**

2.1 Nature of archived data 13
2.1.1 Observations 13
2.1.2 Model outputs 14
2.1.3 Analyses 15
2.1.4 Reanalyses 15
2.2 Data used in this research 16
2.3 Portrayal of fields 17

3 Wavelets **23**
3.1 An intuitive approach to wavelets 23
3.1.1 Decomposition 25
3.1.2 Compression 27
3.1.3 Conservation of energy 27
3.1.4 Discrete wavelet transform 30
3.1.5 2D wavelets decomposition 31
3.2 Wavelets and multi-resolution analysis 34

4 Fingerprinting **41**
4.1 Problem definition 41
4.2 Effectiveness of the mapping 42
4.3 Study of distances between fields 43
4.3.1 Working set 43
4.3.2 Distance matrix 44
4.3.3 Distribution of distances 44
4.3.4 Minimum distance threshold 53

4.4 Wavelet-based fingerprinting 57
4.5 Choice of the compression factor C 62
4.6 Definition of a fingerprinting scheme 65

5 Experimental validation **67**

5.1 Choice of the distance between fingerprints 67
5.2 Fingerprint-based distances 75
5.2.1 Distribution of fingerprint distances 75
5.2.2 Clusters 75
5.2.3 Worst cases 82
5.3 Memory footprint of fingerprints 82

6 Query by example **91**

6.1 A web-based user interface 92
6.2 Interactive user input 92
6.2.1 A canvas for “painting” fields 92
6.2.2 Input tools 94
6.2.3 Predefined patterns 96
6.2.4 Predefined regimes 98
6.2.5 User-provided date 99
6.3 Two fields queries 99
6.3.1 Selection of two query fields 99
6.3.2 Matching of two query fields 100
6.4 Realistic inputs 102
6.4.1 Climatologies 102

6.4.2 Constraints . . . 103
6.4.3 Other constraints . . . 106
6.4.4 Minimisation in an interactive environment . . . 106
6.5 Presenting results . . . 109
6.5.1 Presentation of results . . . 109
6.5.2 Issue with persistent weather . . . 110

7 Implementation **115**
7.1 Software used . . . 115
7.2 Software architecture . . . 118
7.2.1 Frontend . . . 118
7.2.2 Backend . . . 119
7.3 Deployment . . . 122

8 Conclusion **125**
8.1 Results . . . 125
8.1.1 On user perception . . . 126
8.2 Future work . . . 128
8.2.1 Refine the fingerprinting method . . . 128
8.2.2 Implement an efficient data structure to represent the index 128
8.2.3 Consider climatological gradients in constraints . . . 129
8.2.4 Extend to the globe . . . 129
8.2.5 Consider climate projections . . . 131
8.2.6 Add a keyword or textual search . . . 131
8.2.7 Analogues of well-known events . . . 131

8.2.8 Consider time evolution 132
8.2.9 More localised search 133
8.2.10 Fingerprints as proxies for meteorological fields 133
8.2.11 A possible application: finding cyclones 134

Chapter 1

Introduction

1.1 Motivation

The European Centre for Medium-Range Weather Forecasts (ECMWF) has been collecting meteorological information since 1980, and its archive has recently reached over 300 petabytes of primary data. These data consist of collected observations, model outputs from operational runs, research experiments as well as projects from the World Meteorological Organization (WMO) and projects funded by the European Commission.

ECMWF's archive is referred to as the Meteorological Archiving and Retrieval System (MARS) (Raoult et al. (1995); Raoult (1997, 2002); Woods (2006)). This archive is now mature and provides datasets that cover several decades at hourly temporal resolutions.

The primary motivation for this research is to identify and implement a novel way to exploit this wealth of information: the fast retrieval of weather analogues, i.e. given a weather situation, find efficiently all the past weather situations that exhibit the same pattern.

Weather analogue is the term used by meteorologists to refer to similar weather situations. Before computer simulations were available, weather analogues were one of the main tools available to forecasters. Nowadays, finding analogues in an archive still requires performing a brute force search: this entails retrieving data from the archive and compare them to a user-provided input, using a chosen similarity measure such as the Euclidean distance. Such an operation is time-consuming and costly on large archive systems as data will typically have to be recalled from a tape system.

With the advent of the Internet and commercial search engines such as *Google* or *Bing*, users are now expecting their queries to be answered immediately, or at least at a speed that allows interactive use without any delay (Nielsen (2009)). Furthermore, as these search engines automatically correct spelling mistakes and consider synonyms in their algorithms, users also expect results from approximate queries. The purpose of this work is to implement a system that can, given a user query describing a specific weather pattern, return the list of dates at which a similar weather situation happened.

Typical queries would be, for example:

- a cold spell or a heatwave;
- heavy precipitations over a given location;
- strong winds;
- a low pressure system.

Queries should therefore consist of which meteorological phenomenon to consider, its intensity and location. There are many use cases for which such a system would be useful:

- a forecaster would like to know, given today's meteorological situation, when did similar weather occur, and how did it evolve;
- a researcher would extract from the archive all the wind storms as input to their research;
- a model developer studying model errors would be interested in knowing if their model exhibits the same behaviour for similar weather regimes (e.g. systematic biases);
- a journalist would like the list of past heatwaves for an article that they are writing.

1.2 Aim of the research

This research aims to implement an efficient system to identify weather analogues in ECMWF's archive, with the following requirements:

- the system should be queryable: given a user-provided query, the system should return the most similar weather situation from the archive;

- the system should be fast: replies should be perceived by users as instantaneous, allowing interactive use;
- newly archived data should be added to the index, without the need to retune/retrain the system.

To achieve this aim, we will be defining a fingerprinting-based algorithm to index weather patterns. Queries would be done by computing the fingerprint of the query pattern, then comparing them to the index. The fingerprinting scheme should be such that:

- fingerprints are several orders of magnitude smaller than the original data, so it can be stored in an efficient data structure in memory and/or disk;
- similar weather patterns must have similar fingerprints: queries will be run by comparing fingerprints, and distance between fingerprints must reflect distances between weather situations.

This process is similar to what the popular music identification program "Shazam" is doing (Wang (2003)). This program uses the microphone of a mobile device to captures a few seconds of the music being played in a room, then computes a fingerprint from this sample, which is in turn used to look up the song name in a database.

For this to work, the fingerprint must be small, so that the index can be kept on disk, and the fingerprints must capture the broad features of the pattern while being insensitive to small changes. Preferably, there should be a similarity metric (distance) between the fingerprints that preserves the distance between the weather patterns. As for "Shazam", it should be possible to query the system by providing examples.

The objectives of this project are therefore the following:

- *Find an efficient fingerprinting system.* Efficiency here means that the computation of fingerprints is fast, the resulting fingerprints are small, they can be compared quickly and they can be stored in an efficient data structure.
- *Define a measure of the effectiveness of the fingerprinting method.* Part of the research is to find a fingerprinting method that is as accurate as possible, i.e. that returns the "closest" matching weather according to some agreed similarity measure. An objective metric must be defined that can be used to compare various fingerprinting schemes and their parametrisation. This metric must consider, for each fingerprinting scheme, how the distance between two fingerprints relates to the distance between two weather situations according to the agreed similarity measure.

- *Implement an interactive query system.* Users should be able to query the system interactively. This poses the challenge as to how users can effectively and easily describe a weather pattern. This query system should be a web-based graphical user interface allowing users to "paint" meteorological fields and visualise resulting matches.

1.3 Goals

The work described in this report is two-fold: first the definition of an efficient fingerprinting scheme, then the implementation of a *query by example* user interface allowing users to interact with the system.

The term *field* is used to describe a state of a meteorological variable (e.g. wind or temperature) at a given date and time, over a geographical domain. This concept is described in details in section 2.1.2. It is introduced here as the term is used in the following subsections.

1.3.1 Fingerprinting

The first part of this work consists of defining a fingerprinting method that fulfils the requirements introduced in section 1.2. This research will be looking for a scheme that can extract some information (a *fingerprint*) from each meteorological fields. This fingerprint should be small but should carry enough information about the original field so that when compared to other fingerprints, it preserves the relative order of the respective fields.

We will then define a measure of the *effectiveness* of the fingerprinting scheme. Such a measure will represent the average retrieval error. We use this measure to compare various schemes and how they can be parametrised so that the error is minimised.

Most of the meteorological fields are the results of multi-scale waves patterns, and their binary representation, when written to a file, compresses well when encoded in JPEG-2000, a wavelet-based compression scheme. Intuitively, wavelets seem to be able to capture the main features of fields. This work will therefore be based on the results presented by Jacobs et al. (1995) and Baluja and Covell (2008), who use multi-resolution wavelet-based algorithms for image querying and audio fingerprinting respectively.

1.3.2 Query by example

The second part of this research will study how users can query the system to retrieve analogues from the archive.

The fingerprinting method previously described can be tested using existing fields, for example, today's weather according to the most recent output of a numerical model. The fingerprint of the field is computed and matched against the fingerprints of the fields in the archive.

Although this use case is useful in itself, it is more likely that users of the system will want to provide their own fields as queries. This can be done by allowing them to upload the query field, and then having the system returning the nearest matching analogues. If a user wants to know what are the days in the archive with a given type of weather, a heatwave or a famous storm, for example, they will have to know of a date with such a weather situation, retrieve the corresponding field from an archive and upload it as a query. There are limits to this method, as it requires that the user has prior knowledge of a date at which a similar type of weather occurred to the one he or she is interested in.

As part of this work, a method that allows users to describe types of weather in an interactive fashion will be considered. Users will be provided with a tool to "paint" the field they are looking for. The painted pattern will be used as a query to the system, and similar fields will be returned. One of the main challenges of this method will be to ensure that the user's input is realistic from a meteorological point of view, while allowing for sketchy inputs.

Other input methods, such as selecting patterns from a predefined set, or the provision of a catalogue of weather regimes, will also be discussed.

1.4 Preliminary work

Finding weather analogues in an archive is akin to solving the *nearest neighbour* problem in a high-dimensional space. We started our work by looking at how to address that matter, avoiding a brute force approach.

1.4.1 Space partitioning

We first studied space-partitioning trees (Vaněček Jr (1991), Aref and Ilyas (2001)), which are data structures that recursively split a multi-dimensional space, usually

using hyper-planes of that space. The trees are often binary trees, where each node points to two subtrees, one containing the points that are on one side of an hyperplane, and the other containing the points in space that are on the other side of that hyperplane. The root node of the tree represents the whole space.

There are several flavours of such trees, depending on the algorithm used to select the hyperplanes. We studied amongst others, kd-trees (Bentley (1975)) and bisecting k-mean (Leskovec et al. (2014)) and PCA-based trees (McNames (2001)).

Searching the nearest neighbours of a point q within a radius r in a space-partitioning tree consists of visiting the tree starting from the root node and following the nodes that correspond to the side of the hyperplanes q belongs. When the ball centred on q of radius r intersects the hyperplane, both subtrees are visited. Searches are very fast and the number of comparisons is of the order of the dimensionality of the space considered.

The partitioning algorithms studied varied in complexity and time to build, the speeds of the lookups, and how balanced and deep the resulting trees are. Although kd-trees are often deeper than the other trees, but they are the fastest, especially at high dimensions.

PCA based algorithms are computationally very expensive, especially at creation time, but create shallow trees.

1.4.2 Space-filling curves

We also considered space-filling curves (e.g. Hilbert curve) as a means to measure the similarity between two fields. The advantage of these curves is that they offer a mapping between high dimensional spaces and a 1D number line, therefore providing a total ordering of the high-dimensional objects, as well as providing a very simple and fast metric for comparisons (Wang and Shan (2005), Lawder and King (2000), Liao et al. (2001), Hungershöfer and Wierum (2002)).

The main shortcomings of both space-partitioning trees and space-filling curves is that they are often expensive to build and contain all the points of the search space considered, and thus can be very large data structures.

Furthermore, both approaches suffer from the curse of dimensionality (Marimont and Shapiro (1979)). One way to overcome that curse is to reduce the dimensionality of the data considered.

1.4.3 Dimensionality reduction

There are many techniques to reduce the dimensions of a dataset (Sorzano et al. (2014),Van Der Maaten et al. (2009)) such as Principal Component Analysis (Pearson (1901)), Singular Value Decomposition (Golub and Reinsch (1971)) or Locally Sensitive Hashing (Indyk (2000)) to name a few, to which we can add autoencoder neural networks (DeMers and Cottrell (1993), Hinton and Salakhutdinov (2006)).

Wavelets are also used to reduce the dimensionality of data (Bruce et al. (2002), Gupta and Jacobson (2006), Bruce et al. (2002)).

All these methods introduce a mapping between a high dimensional space into a much smaller one, by retraining the most important information and discarding the less important one. This is the property we are looking for when computing fingerprints and the target space of the mapping can be considered to be the set of fingerprints of the elements of the original space. Consequently, any of the dimensionality reduction techniques can be considered to implement a fingerprinting scheme.

1.5 Related work

This research spans several disciplines: wavelet-based retrieval systems and query by example on the one hand, weather analogues and input of meteorological fields on the other. The amount of literature available for these topics varies a lot.

1.5.1 Weather analogues

Before computer simulations were available, weather analogues were the main tool available to forecasters. The chaotic nature of the atmosphere made this technique unreliable, as small differences between two similar weather patterns may lead to large differences in the following days (Lorenz (1969)). Furthermore, it was estimated that at a hemispheric scale, similar states of the atmosphere would be observed every 10^{30} years (Van den Dool (1994)). Nevertheless, analogues can be useful on a smaller scale ($\approx 900\,\text{km}$ in radius, Van den Dool (1989)).

Weather analogues have many usages. They are used for downscaling model outputs (Zorita and Von Storch (1999)). Grenier et al. (2013) assess various dissimilarity metrics to select climate analogues at geographically distant point locations (spatial analogues). Delle Monache et al. (2013) complement a current forecast with past analogues to create probabilistic weather prediction. Evans and Murphy

(2014) propose a tool to assess the risks of severe weather based on historical analogues. Most of these tools are considering single location analogues, and are based on climate time-series (e.g. past weather) at that location.

1.5.2 Wavelet-based retrieval systems

As the world is generating more and more data, efficient information retrieval has become a major challenge, and is therefore a very active field of research. Information is not only limited to text, but also comprises of images, movies and sounds. In this research, we will focus on wavelet-based retrieval system.

Walker (2005) provides an in-depth primer on the use of wavelets for scientific applications. He first introduces the mathematics of wavelets and multi-resolution analysis. He then presents the use of wavelets for audio and image compression and denoising, as well as other image processing techniques such as edge detection and pattern recognition.

Stollnitz et al. (1995a,b, 1996) also present a comprehensive primer of the use of wavelets for computer graphics. They also introduce the mathematics behind wavelets and their use for image compression. They then define b-spline based wavelets that can be used to capture the details of objects such as 2D curves and 3D surfaces. These objects can then be manipulated at a given level of detail without affecting details at other resolutions: a user could change the general sweep of a 2D curve while preserving smaller variations.

Shapiro (1993) explains how wavelets can be used to compress and encode images so that they can be reconstructed incrementally from a stream of bits, e.g. decode and uncompress on the fly as the image is received from a network.

Do and Vetterli (2002) show how to compute a generalised Gaussian distribution of the wavelets coefficients of an image. Image registration is then performed by comparing distributions with the Kullback–Leibler distance. Patrikalakis et al. (2006) use that work for the retrieval of seafloor sonar images.

Regentova et al. (2000) decompose images into sub-images from which edge and texture information are extracted using wavelet analysis. Similarities between sub-images are done by comparing edge and texture for each matching sub-images, resulting in a bit vector (0 if sub-images do not match, 1 if they match). Images are considered similar if the Hamming distance between these vectors is above a given threshold.

Wavelets have been used to retrieve medical images: Traina et al. (2003) ap-

ply multi-resolution wavelet decomposition to magnetic resonance medical images (MRI) to extract a *feature vector*, composed of the various wavelet coefficients; a database of MRI images can then be queried for similarities by comparing these feature vectors, using a normalised Euclidean distance. Pauly et al. (2009) also consider using multi-resolution wavelet decomposition for image registration of medical images, their purpose being to align images coming from different sources. Wavelet decomposition is used to extract a *wavelet energy map* from each image, which is then compared when performing the registration, and is robust to noise.

Marsolo et al. (2006) propose a wavelet-based system to find similar proteins. For each protein, they compute a distance matrix between every atom composing it, thus reducing a 3D structure to a 2D one. They then apply a 2D wavelet decomposition on the distance matrix, which can be considered as a greyscale image. From this decomposition, they only retain the top coefficients. They name the resulting vector *global descriptor* of the protein. Queries are then performed on these vectors. They show that this system outperforms a traditional kd-tree lookup by two orders of magnitude and around 90% accuracy, and also allows the matching of sub-structures.

Jacobs et al. (1995) propose a system that extracts a small *signature* from images before storing them into a database. This signature is computed using multi-resolution wavelet analysis. To search for an image, a user will provide an approximate sketch of it by "painting" a rough example of the requested image. The system will extract the signature from the user input and compare it with the stored signatures. This study shows that this scheme is more performant, both in speed and success rate, than traditional image retrieval systems based on the Euclidean distance on colour histograms. The work presented in this thesis is strongly inspired by this paper, as it introduces a system that fulfils all of the requirements outlined in section 1.2.

Baluja and Covell (2008) use wavelets to extract fingerprints from audio data, to identify songs of music pieces from short snippets of recordings. Spectrograms are then extracted from the audio data, at regular time intervals and for different bandwidths, and then converted into images. Multi-resolution wavelet analysis is then performed on the images, to extract the top coefficients, which are then encoded in bit-strings signatures as proposed by Jacobs et al. (1995). These signatures are hashed using a Locality-Sensitive-Hashing (Slaney and Casey (2008)), for fast retrievals.

1.5.3 Query by example

As previously mentioned, Jacobs et al. (1995) implement a query by example system in which users can "paint" a rough outline of the queried image, but this aspect is secondary to their paper.

Chang and Fu (1980) propose a *query by pictorial example*: before being indexed in a database, features such as roads or rivers are extracted from satellite images using a series of pattern recognition functions. These patterns are then stored in the database as a series of polygons. Users will then express queries by describing polygons, using a query language, which will be matched with the stored polygons.

1.5.4 Input of meteorological fields

The literature is very sparse concerning the interactive systems allowing the input or modification of meteorological fields. This is not due to such systems being rare: most forecasters' workstations provide tools to adjust model outputs interactively. It is mostly because no publications were made describing these systems.

Ruth (1993) proposes such a system for the Advanced Weather Interactive Processing System (AWIPS), the forecaster's workstation of the National Weather Service, in the USA. The paper describes software that allows forecasters to interactively edit contour lines of an underlying gridded field, before this field is used as input to a text generation software. The author introduces a method called the Systematic Interpolative Radial Search (SIRS) to reconstruct a field, i.e. a matrix of values, from isolines, by radially scanning for the nearest isolines from a given grid point, and then interpolating. Such a system is used by forecasters to adjust gridded fields produced by a numerical weather prediction model, in order to take into account sub-grid features missed by the model, such as the effect of a local valley.

Carroll (1997); Carroll and Hewson (2005) propose a method to interactively modify meteorological fields such as surface pressure, geopotential or precipitations. Users can draw an arrow on the screen to indicate how meteorological features must be adjusted. The system will modify the underlying data in such a way that the physical consistency between several fields is preserved, e.g. it will adjust the wind fields in response to changes in the pressure fields, and the humidity field according to modifications done on the location of precipitations. It considers how changes affect the fields in three dimensions, as well as in time. This system is at the heart of the forecaster's workstation at the MetOffice.

1.6 Contributions

In the first part of this research, we show that wavelet-based fingerprints which have been successfully used for image and audio retrieval, can be used to find weather analogues in a meteorological archive containing hourly fields for a 40 years period, over a predefined geographical domain.

In the process, we introduce a method to estimate the significant distance between two meteorological fields using hierarchical clustering. We also show that fingerprint-based distances can themselves be used to perform some clustering on the underlying fields.

We show that the distributions of distances between fingerprint are qualitatively the same as the distributions of distances between fields according to the Euclidian distance.

In the second part of this work, we propose ways for a user to interactively search for weather analogues in the archive, using the fingerprints. We introduce a web-based query by example user interface and discuss various input methods.

We propose a system by which users can "paint" meteorological fields on the screen, and be presented with matching results. A series of constraints can be applied to ensure that the user input represents climatologically realistic fields, in a fashion that does not affect the responsiveness of the user interface. We show that clustering of results can be used to ensure that results are spread throughout the whole archive.

Finally, we described the system that had been implemented, its software components and its deployment in a cloud infrastructure. The web page developed in the course of this research will be made available to expert users, such as forecasters and researchers.

Chapter 2

Meteorological data

Meteorological data are very varied in type, format, precision, etc. These data represent a particular weather phenomenon, at a given time (past or future), at a given location.

Handling this variety is difficult, and processes have been put in place over the years by the World Meteorological Organization to encode, annotate, exchange and archive these data. In this study, we will consider data that are already archived and can be retrieved by attributes, such as date or location. This is the case of all data available in ECMWF's MARS archive.

In this chapter, we will introduce the notion of *meteorological variable* as well as the notion of *meteorological field*, and how they are traditionally represented graphically.

2.1 Nature of archived data

The MARS archive contains two main types of data: observations and model outputs. Although each of them represents similar information, they are very different in the way they are created and in their structure.

2.1.1 Observations

The state of the atmosphere is described using several *variables* that represent physical quantities or dynamical properties, such as *air temperature*, *air pressure*, *wind speed*, *precipitations*, *cloud cover*, etc. Observations are collected in real-time

from different instruments (i.e. wind gauges, thermometers, barometers, etc.), located on different platforms (ships, drifting buoys, satellites, aircraft, balloons…). Observations are always given within a spatiotemporal reference system (date, time, location and height).

Every day, hundreds of millions of observations are captured, amounting to hundreds of GB of data. They are exchanged globally for use by numerical weather prediction systems (NWP), which are models running on supercomputers that forecast future weather. Observations are also stored in archives for later use, for example, to study the climatologies of given locations.

Observations are very varied in type, quality, frequency, etc. The unstructured nature of observations (uneven spatiotemporal distribution), as well as their erroneous nature (accuracy of the captor, measurement errors, instrument failures, etc.), make them difficult to use.

2.1.2 Model outputs

The second type of data is the output of NWP models; they are collections of *fields*, one for each variable represented, for a given time and horizontal layer. At large scales (greater than 10 km), the interactions between the different layers of the atmosphere are small compared to the effects of large structures and can be ignored. This is why traditionally meteorologists tend to consider fields as being 2D, their vertical coordinate being an attribute of the field, as is time.

Fields are matrices of floating-point values geographically distributed according to a mesh (called grid), either global or regional. The grids are sets of regularly distributed points (e.g. one grid point every 5 km) over a given area. These grids can be very large, several millions of points per field. The number of fields produced by an NWP system is also considerable: hundreds of different variables, hundreds of vertical levels and time steps.

Furthermore, in order to capture the predictability of the weather, these models are often run as *ensembles* (Molteni et al. (1996)), which means that the same model is run many times with slightly different initial conditions, and then the results are considered in a probabilistic fashion.

As a consequence, meteorological models produce a massive amount of outputs (many TB per day), much larger than the volume of corresponding observations. On the other hand, the structured nature of the fields makes it easier to use in computer programs.

2.1.3 Analyses

The process that links observations and fields is called *data assimilation* (Bengtsson et al. (1981); Ghil and Malanotte-Rizzoli (1991)): starting from a set of fields representing a "first guess" of the state of the atmosphere (usually a previous forecast), the process of data assimilation consists in adjusting the values at every grid point of these fields, to take into account the observed values within a given spatial and temporal window, as well as possible errors in the observations; this is done by minimising a cost function, and requires vast amounts of computing resources.

The output of the data assimilation is then considered the best approximation of the "truth", i.e. the best description of the state of the atmosphere based on the values provided by the observations. This output is called *analysis* and is used as the initial step of a weather forecast.

2.1.4 Reanalyses

The data assimilation described previously can be seen as a process that transforms observations, which are difficult to use in computer programs, into analysis fields, which are much easier to work with.

Reprocessing past observations in this way will produce a dataset which is known as a *reanalysis*. Observations are retrieved from the archives, then put through a data assimilation system, and the resulting fields are added back to the archive. Such datasets are very well structured and can be easily processed.

Because the data assimilation software used is unchanged during the creation of the reanalysis it produces a consistent set of fields representing the state of the atmosphere over long periods. This is used for studies linked to climate change (Santer et al. (2004); Frauenfeld et al. (2005)).

Several reanalyses have been produced by ECMWF, including ERA15 (Gibson et al. (1997)), ERA40 (Uppala et al. (2005)), ERA-Interim (Dee et al. (2011, 2014); Berrisford et al. (2011)), and now ERA5 (Hersbach and Dee (2016); Hersbach et al. (2019)).

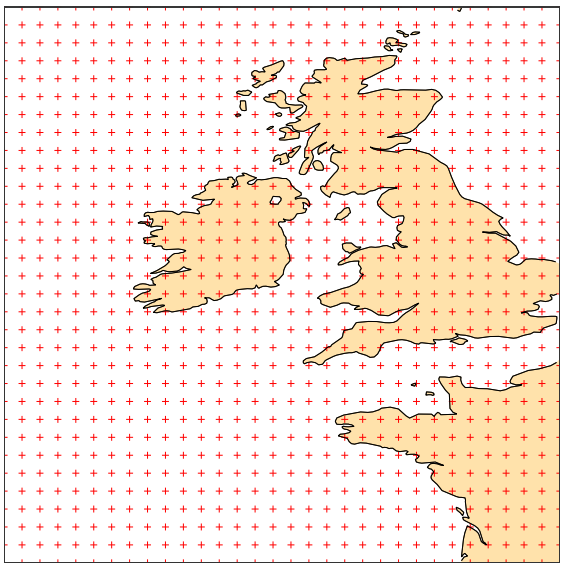


Figure 2.1: *Grid and area considered in this research.*

2.2 Data used in this research

This research is based on ECMWF's reanalyses. As described above, reanalyses provide an easy to use, consistent view of the weather over a very long period.

The data used in this work were originally daily data selected from the *ERA-Interim* dataset, a reanalysis covering the period 1979 to 2014, at 00 UTC (13 149 fields per variable). ERA-Interim is a global dataset on an N128 gaussian grid (≈ 80 km horizontal resolution).

During the course of this research, ECMWF produced a newer reanalysis, *ERA5*, covering 1979 to present. Fields are available hourly, the dataset considered (1979-2018) contains therefore 349 176 fields per variable. ERA5 is a global dataset on an N320 gaussian grid (≈ 30 km horizontal resolution). We decided to use this dataset instead of *ERA-Interim* as it is available at higher spatial and temporal resolutions.

The work presented focuses on a regular latitude/longitude grid of 0.5°×0.5°(≈ 55 km×55 km) on the domain 60°N 14°W 44.5°N 1.5°E that covers the British Isles (≈ 1700 km×1700 km, see figure 2.1), which agrees with the radius of 900 km presented in Van den Dool (1989) within which analogues can be considered. The size of the domain will capture synoptic scales weather patterns.

A typical NWP system will produce hundreds of meteorological variables. In this study, we will focus on *surface air temperature*, *snow depth*, *snowfall*, *10m zonal wind*, *10m meridional wind*, *10m wind speed*, *mean sea level pressure*, *total cloud*

cover, *total precipitations*, *geopotential at 500 hPa*, *geopotential at 850 hPa* and *significant wave height*. They have been selected because most of them relate to attributes of weather that a user is likely to look for when searching for analogues.

The *10m zonal wind* is the west-east component of the wind vector at a height of 10 m, while the *10m meridional wind* is its south-north component. The *mean sea level pressure* is the pressure as it would be at sea level if the effect of altitude was removed (as the altitude increases, the pressure of the atmosphere decreases). *Total precipitations* are the amount of water falling from the sky in all its forms: rain, snow, hail, sleet, etc. The geopotential represents the height of a given pressure level, multiplied by the constant of gravity g. For example, a value of 54 000 for *geopotential at 500 hPa* means that the isobar 500 hPa is at around 5504 m of altitude; this variable is one of the most important for the meteorologist as it captures the large scale weather patterns. Other variables are self-explanatory.

This report will contain many tables, maps and graphs. In order to keep it concise and readable, their number has been kept to a minimum, and not all tables, maps and graphs are shown for every selected meteorological variable, but only for one or two of them, for the purpose of illustrating the discussion.

2.3 Portrayal of fields

It is essential for the reader to understand the difference between the fields and their pictorial representation, as searching for analogues considers the former, while queries and results are presented to users according to the latter.

Meteorological fields are vectors of floating-point values, each value representing the intensity of that field at a corresponding grid point. The mapping between the index of the value in the vector to its corresponding grid point is called a *projection*. In this work, we are considering fields on a regular latitude-longitude grid, with an *equidistant cylindrical* projection (also known as *plate carrée* projection); in this case, it is customary to represent the values as a 2D matrix:

$$F = \begin{bmatrix} f_{11} & f_{12} & \cdots \\ \vdots & \ddots & \\ f_{K1} & & f_{KK} \end{bmatrix} \tag{2.1}$$

The action of representing a meteorological field graphically is called *portrayal* (Papathomas et al. (1988)). Usually, this is done by contouring the data at pre-defined regular intervals and then drawing the resulting isolines, labelling them

with the value they represent. Sometimes the space between isolines is filled with a uniform colour, according to a given palette mapping ranges of values (this is called *shading*).

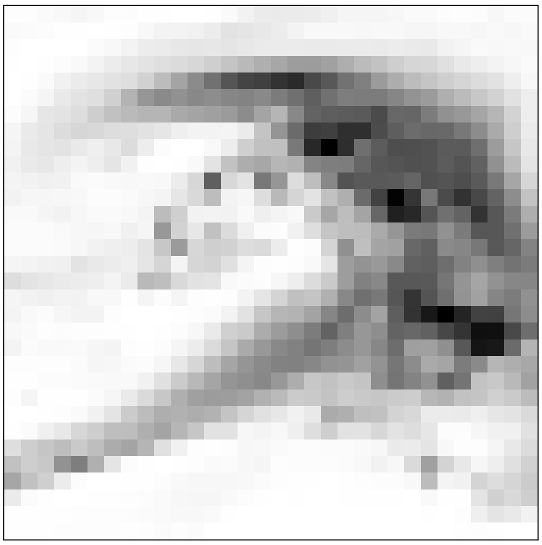
Traditionally parameters such as geopotential and pressure are portrayed with contour lines, while fields like precipitations or temperature are portrayed using shading. In the case of temperature, a palette ranging from blue (cold values) to red (warm values) is used. Figure 2.2 on the next page shows a field of precipitation and its usual portrayal using shaded contours following a logarithmic scale.

The figure demonstrates that a given portrayal (e.g. contouring and shading) can hide a lot of details of the underlying field. This will affect the perception users will have when visually comparing two fields. Other fields are shown in figure 2.3 on page 20.

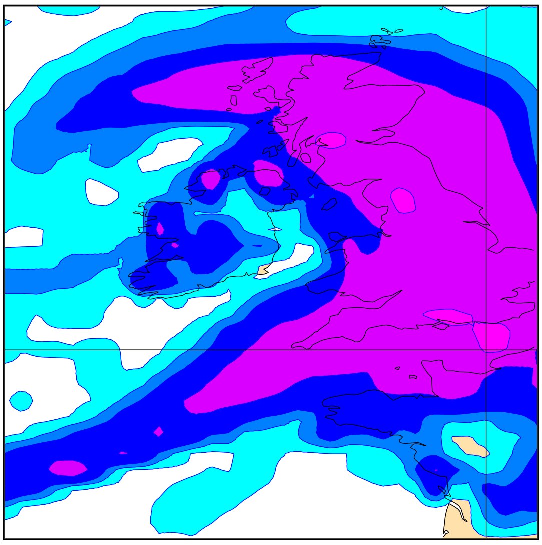
In this report, all *mean sea level pressure* fields are plotted with the fixed set of contouring intervals (one isoline every 5 hPa). *Geopotential at 850 hPa* fields and *geopotential at 500 hPa* fields are plotted with one isoline every 50 m. Other variables are plotted using colour palettes representing ranges of values. The mappings between colours and field values are shown in figures 2.4 and 2.5 on page 22.

.	.	0.1	0.1	0.2	0.1	.	0.1	.	.	.	.	.	0.1	0.3	0.4	0.4	0.4	0.4	0.3	0.2	0.2	0.2	0.2	0.1	0.1	0.1	.	0.1	0.1	0.1	0.1
0.1	0.1	0.1	.	.	.	.	.	0.1	0.2	0.2	0.3	0.4	0.6	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.1	0.1	0.2	0.1	0.2	0.1
.	.	.	.	.	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.5	0.5	0.5	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.3	0.2	0.1	0.1	0.1	0.1
.	.	.	0.1	0.1	0.2	0.4	0.5	0.6	0.6	0.7	0.8	0.9	1.1	1.4	1.6	1.7	1.9	1.9	1.8	1.6	1.3	1.2	1.0	0.8	0.8	0.6	0.4	0.3	0.2	0.2	0.1
.	.	0.1	0.3	0.4	0.5	0.6	0.6	0.9	1.3	1.8	2.3	2.8	3.3	3.7	3.7	4.0	4.2	3.6	3.3	2.8	2.5	2.0	1.6	1.3	1.1	1.0	0.8	0.7	0.6	0.3	0.2
.	0.1	0.2	0.4	0.5	0.6	0.9	1.5	2.1	2.3	2.2	2.3	2.2	2.4	2.4	2.3	2.6	2.8	3.1	2.8	2.5	2.6	2.6	2.4	2.1	2.0	1.6	1.3	1.1	0.9	0.6	0.3
0.1	0.2	0.5	0.5	0.7	1.0	1.2	1.4	1.4	1.4	1.7	1.8	1.9	2.0	2.3	1.8	1.5	2.3	3.1	3.2	3.2	3.2	3.5	3.0	3.0	2.6	2.2	2.0	1.8	1.4	0.9	0.3
0.2	0.4	0.5	0.7	0.8	0.8	0.7	0.6	0.6	0.5	0.3	0.2	0.2	0.2	0.4	1.2	2.1	2.9	3.8	3.7	4.1	4.1	3.5	3.1	2.9	3.0	2.9	2.7	2.3	1.7	1.0	0.4
0.2	0.4	0.5	0.6	0.4	0.3	0.4	0.7	0.4	0.1	.	.	0.1	0.4	0.9	1.2	1.2	2.6	4.2	5.0	4.1	2.8	3.2	3.5	3.5	3.4	3.2	3.2	2.8	1.9	1.1	0.4
0.3	0.4	0.5	0.3	0.2	0.3	0.5	0.3	.	.	.	0.1	0.6	1.2	1.6	1.7	1.4	1.5	2.4	2.5	2.5	3.0	3.3	3.4	3.4	3.4	3.2	3.3	3.1	2.2	1.3	0.5
0.2	0.3	0.3	0.3	0.1	0.1	0.1	0.1	0.1	0.1	0.4	0.8	3.1	0.8	0.6	2.7	2.1	0.7	1.3	2.3	3.4	3.3	3.3	3.2	3.0	3.4	3.3	3.5	3.2	2.6	1.7	0.8
0.2	0.2	0.2	0.2	0.1	0.1	0.1	0.1	0.2	0.2	0.5	0.7	1.3	0.2	0.2	0.7	1.4	0.8	0.6	1.2	1.6	2.9	3.7	4.9	4.0	2.6	3.5	4.0	3.6	3.1	2.2	1.2
0.1	0.1	0.2	0.3	0.2	0.1	0.1	0.1	0.2	1.3	0.9	0.3	0.2	0.3	0.1	0.3	0.2	0.1	1.0	1.7	1.3	1.8	3.0	4.4	4.2	3.3	2.8	3.1	4.0	3.3	2.6	1.6
0.1	0.1	0.1	0.1	0.2	0.2	0.1	0.3	0.2	2.0	0.9	0.5	1.2	0.7	0.3	0.1	0.1	0.3	0.6	1.1	1.3	1.3	1.7	2.1	2.6	2.6	2.2	2.6	3.3	3.3	2.8	2.0
0.1	0.1	0.1	0.1	0.2	0.3	0.3	0.4	0.5	1.1	2.0	0.8	0.9	0.9	0.7	0.7	0.4	.	0.1	0.8	2.2	1.3	1.8	1.8	3.0	3.1	2.3	2.4	2.6	3.3	3.0	2.4
0.2	0.2	0.3	0.3	0.6	0.4	0.4	0.3	0.5	0.8	0.2	0.8	0.9	0.6	0.2	.	.	.	0.1	0.5	1.9	2.1	2.0	2.3	3.1	3.2	2.8	2.5	2.2	2.5	2.8	2.6
0.7	0.5	0.5	0.4	0.4	0.3	0.2	0.4	1.4	1.3	0.7	0.6	0.5	0.1	0.1	0.1	0.1	0.3	0.7	1.0	1.9	2.5	2.3	3.3	3.3	3.2	2.7	2.1	1.8	2.1	2.4	2.4
0.2	0.3	0.3	0.2	0.3	0.1	0.1	.	0.2	0.1	0.2	.	.	.	0.2	0.5	0.8	1.2	1.2	1.5	2.3	2.8	2.6	3.3	4.0	3.4	2.8	2.1	1.8	2.4	2.6	2.2
0.2	0.1	0.1	0.2	0.2	0.2	.	.	.	.	.	.	0.1	0.2	0.6	1.1	1.4	1.5	1.8	2.4	2.9	2.3	1.8	3.2	4.0	4.5	5.0	4.3	3.7	3.7	2.7	2.2
0.1	0.1	.	0.1	.	.	.	.	.	.	0.1	0.1	0.4	0.9	1.0	1.6	1.9	2.2	2.6	3.1	2.5	2.0	2.2	3.1	3.0	3.1	3.6	4.0	4.7	4.7	3.6	2.9
0.2	0.1	0.1	0.1	0.1	0.2	0.2	.	.	0.1	0.2	0.4	0.7	1.1	1.7	2.0	2.3	2.5	2.7	2.5	2.3	2.0	2.3	2.9	1.9	1.9	2.2	3.3	4.6	4.5	2.9	1.5
.	.	.	.	0.1	0.2	0.1	0.1	0.1	0.2	0.5	0.8	1.3	1.8	2.1	2.3	2.5	2.5	2.4	2.1	2.1	1.8	2.1	2.7	2.3	1.7	1.8	2.6	2.3	1.4	1.6	1.7
.	.	0.1	0.1	0.1	0.1	0.1	0.1	0.3	0.6	1.0	1.4	1.8	2.1	2.2	2.3	2.0	1.6	1.1	1.2	1.1	1.2	2.2	2.7	2.4	2.2	3.1	2.7	1.2	1.1	1.4	1.8
.	0.2	.	.	.	0.1	0.2	0.4	0.7	1.1	1.6	1.9	1.9	1.9	1.8	1.5	1.3	1.0	1.1	1.2	1.0	1.1	1.1	1.0	1.0	1.4	1.5	1.2	1.1	0.9	1.0	1.3
.	.	.	.	0.1	0.3	0.5	1.0	1.3	1.6	1.5	1.6	1.5	1.2	0.8	0.6	0.5	0.3	0.4	1.6	1.6	1.3	1.2	0.9	0.8	0.4	0.4	0.4	0.3	0.5	0.5	0.7
.	0.1	0.3	0.5	0.8	1.0	1.3	1.3	1.6	1.8	1.4	1.1	0.6	0.6	0.2	0.2	0.1	0.3	0.6	1.0	0.6	0.6	0.8	0.5	0.7	0.5	0.1	0.1	0.2	0.3	0.3	0.6
0.6	1.0	1.1	1.3	1.1	1.3	1.6	1.8	1.5	0.6	0.3	.	.	.	0.2	0.1	0.1	0.1	.	0.1	0.1	0.1	0.2	0.4	0.4	0.5	0.1	.	.	0.2	0.4	0.7
1.2	1.0	1.1	2.1	2.5	1.5	0.5	0.1	.	.	.	.	0.1	0.1	0.2	0.2	.	0.1	0.1	0.1	0.1	0.1	0.3	0.2	0.7	1.9	0.9	0.5	0.2	0.3	0.6	0.9
1.7	1.2	0.9	0.4	0.1	.	.	.	.	0.1	0.1	0.2	0.2	0.2	0.1	0.1	.	.	.	.	.	.	0.1	0.1	0.1	1.3	0.2	0.3	0.8	0.7	0.8	1.0
0.4	0.1	.	.	.	.	.	.	0.1	0.1	0.2	0.2	0.2	0.1	0.1	.	.	.	.	.	.	.	.	.	.	0.1	.	.	0.5	0.9	0.8	1.0
.	.	.	.	.	.	.	0.1	0.1	0.1	0.1	0.1	0.1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	0.3	0.5	0.4	0.3
.	.	.	.	.	.	.	.	.	0.1	.	0.1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

(a) *Values of the field of one hour total precipitations for 24 October 1998, at* 12 UTC, *in* mm. *The zero values have been substituted with a dot for the sake of clarity.*

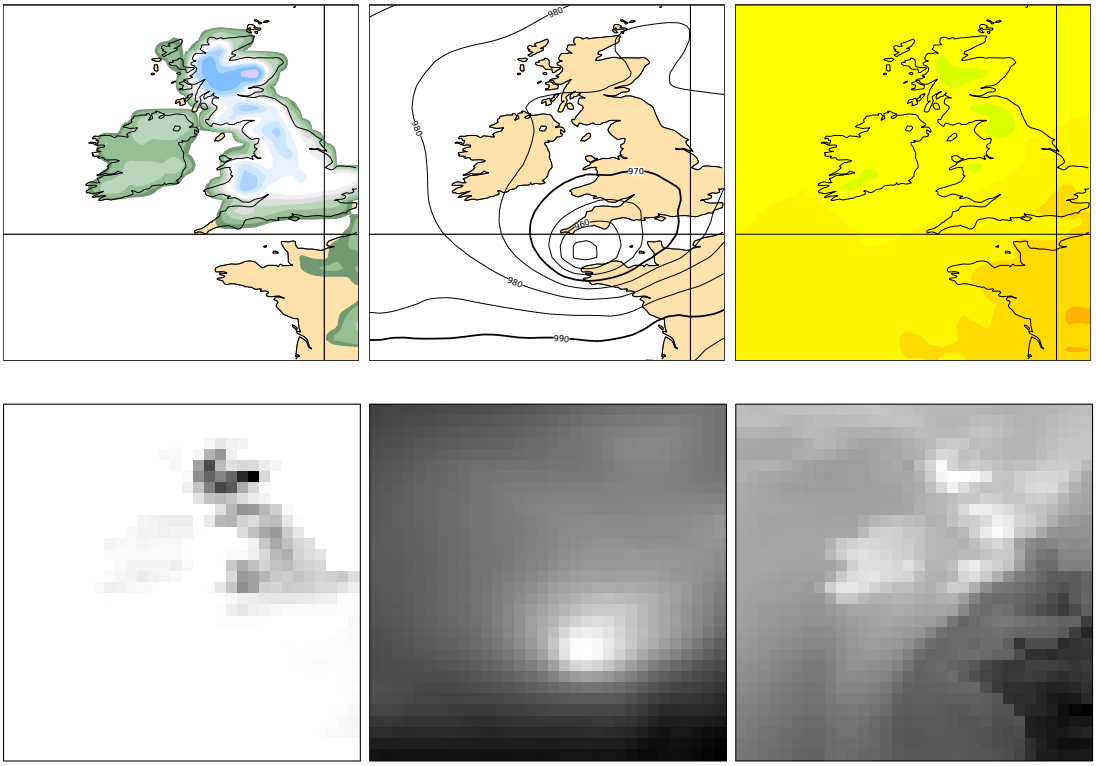


(b) *The matrix above plotted as a grey map. The values are first normalised to the interval* $[0, 255]$.



(c) *The precipitation field is plotted with contours and shading. The contours values follow a logarithmic scale.*

Figure 2.2: *This figure shows the grid point values of a* total precipitation *field (2.2a), a grey map representation of the same field (2.2b) and the traditional portrayal of such a field with contours and shading (2.2c). The figure clearly shows that contouring and shading may hide a lot of details, visible in the grey scale image. This will affect the perception a user may have when visually comparing two fields.*



(a) Snow depth. **(b)** Mean sea level pressure. **(c)** Surface air temperature.

Figure 2.3: *This figure shows some of the traditional portrayals of some meteorological fields (top row). They can use shading, contours or a mixture. In the case of the* snow depth *field, the shading is only present when there is snow on the ground. The bottom row shows the corresponding fields as a grey map, with their values normalised to the interval* $[0, 255]$*.* 0 *(white) represents the minimum value, while* 1 *(black) represents the maximum value*

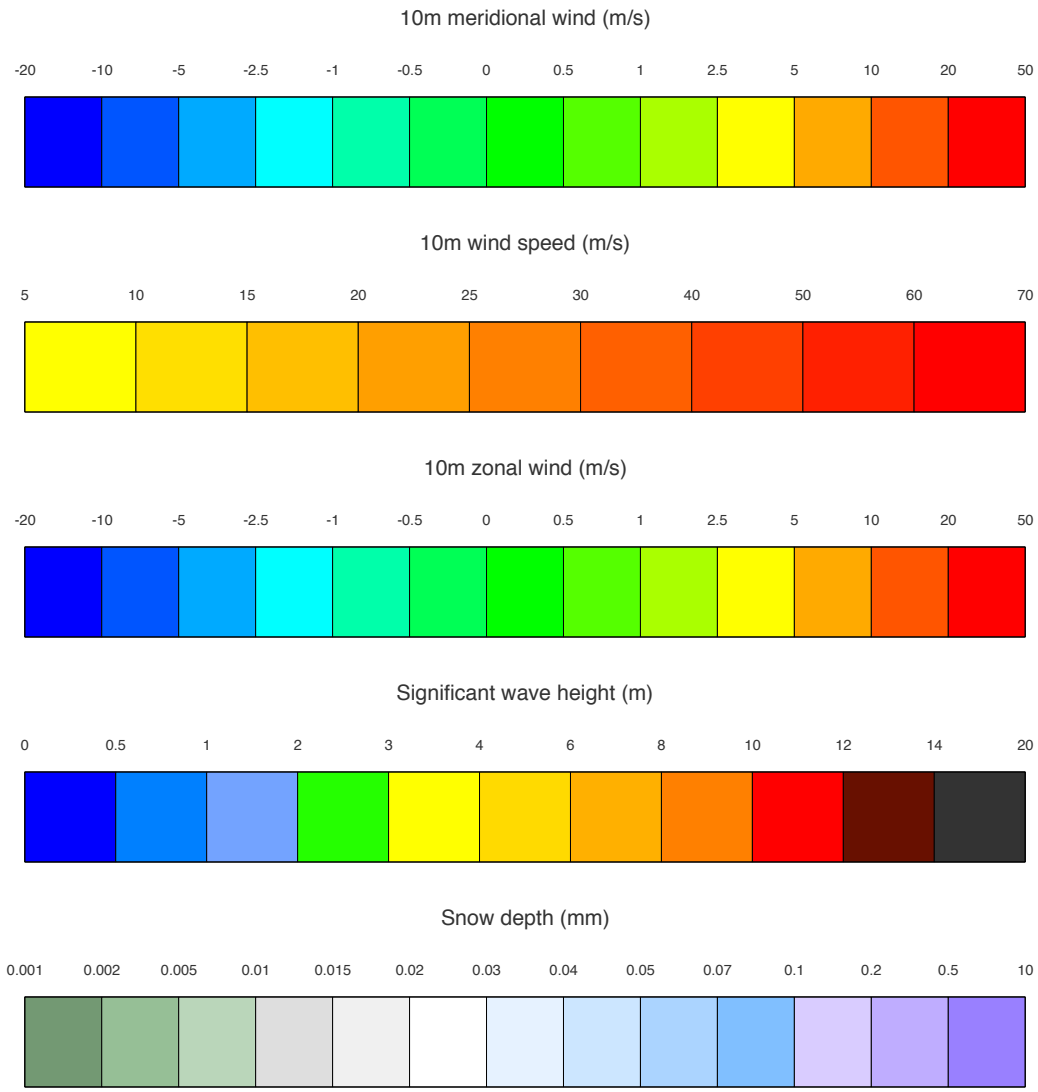


Figure 2.4: *Colour palettes used for the portrayal of meteorological variables.*

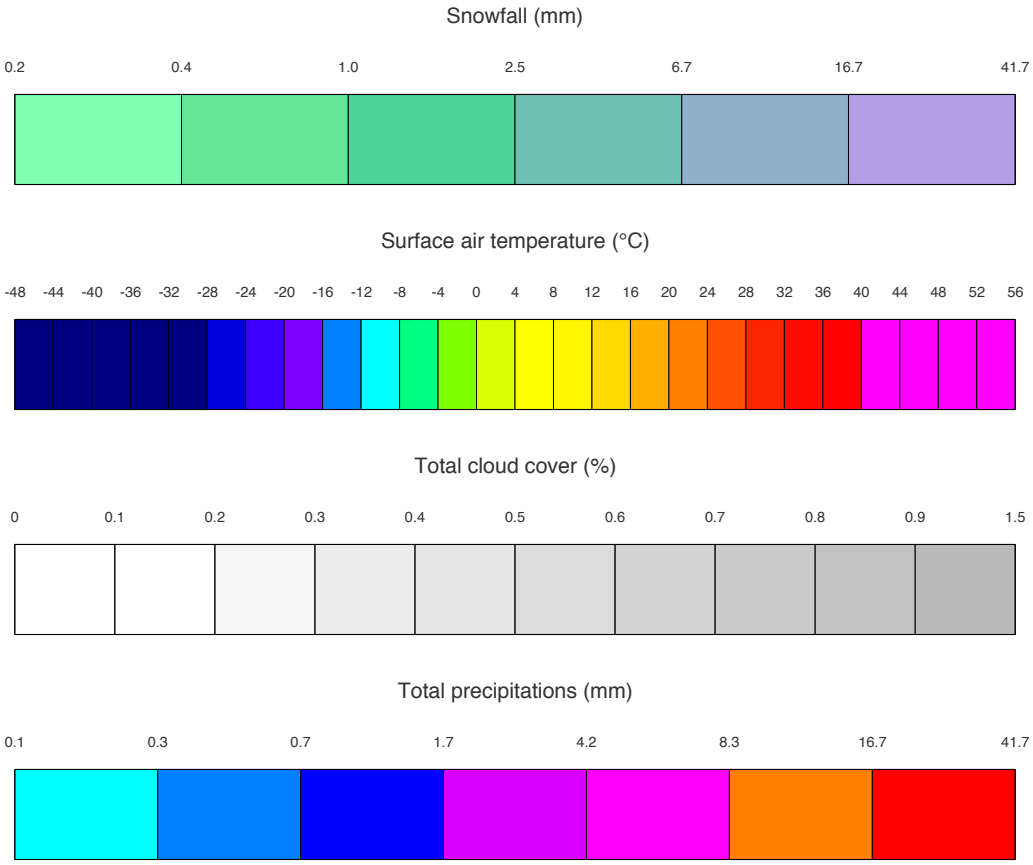


Figure 2.5: *Colour palettes used for the portrayal of meteorological variables.*

Chapter 3

Wavelets

The aim of this chapter is to provide the reader with an insight on how wavelet decompositions will compress the information contained in meteorological fields while preserving some of their spatial structure.

Wavelets and wavelet transformations are introduced in the literature in many fashions, depending on the indented audience (Daubechies (1988); Stollnitz et al. (1996); Mallat (1989); Keinert (2003); Walker (2005); Eckley (2001)). We do not intend to present here the mathematics behind wavelets and multi-resolution analysis, but instead, introduce them following an intuitive and algorithmic approach.

We will start by introducing how a 1D signal can be decomposed into a series of coefficients representing details at various resolutions, and the original signal can be retrieved from these coefficients. 2D wavelet decomposition is also touched upon.

3.1 An intuitive approach to wavelets

Considering the following discrete signal shown in figure 3.1 on the following page. The signal can be represented by the vector:

$$V = [10, 8, 9, 12, 7, 4, 3, 7, 5, 8, 12, 11, 10, 13, 14, 17]$$

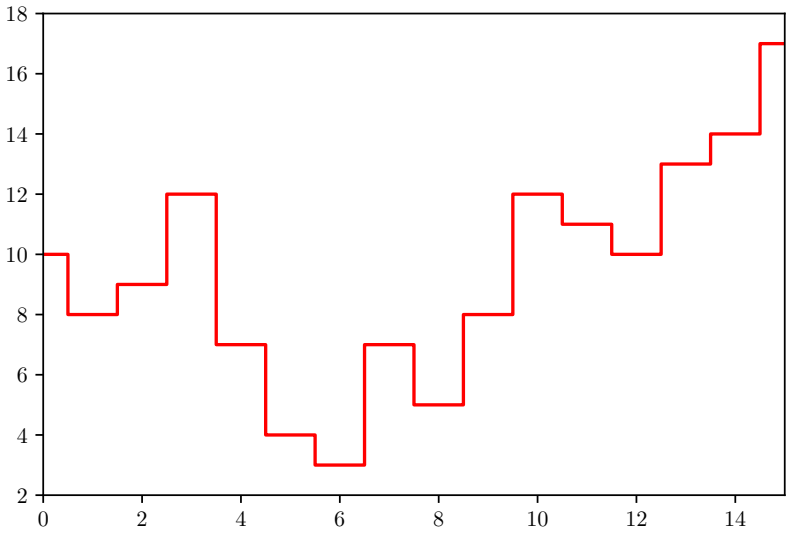


Figure 3.1: *Sample signal.*

3.1.1 Decomposition

Such a vector can be *decomposed* using the following procedure:

- values are averaged pairwise, and the averages are accumulated in a new vector A;
- the differences between averages and the original values are accumulated in a vector D. Because the difference between an average and the two values used to compute it are simply opposite of each other, only one difference is added to D.

The corresponding algorithm is:

Procedure *decompose(V)*
  $A \leftarrow [\,]$
  $D \leftarrow [\,]$
  for $i \leftarrow 1$ **to** $length(V)$ **by** 2 **do**
    $average \leftarrow (V[i] + V[i+1])/2$
    $A.append(average)$
    $D.append(V[i] - average)$
  end
  return (A, D)
end

Algorithm 3.1: *Decomposition of vector, first version.*

Which can be rewritten more elegantly as:

Procedure *decompose(V)*
  $A \leftarrow [\,]$
  $D \leftarrow [\,]$
  for $i \leftarrow 1$ **to** $length(V)$ **by** 2 **do**
    $A.append((V[i] + V[i+1])/2)$
    $D.append((V[i] - V[i+1])/2)$
  end
  return (A, D)
end

Algorithm 3.2: *Decomposition of vector, second version.*

Calling the procedure *decompose* with V as defined above leads to:

$$A = [9, 10.5, 5.5, 5, 6.5, 11.5, 11.5, 15.5]$$
$$D = [1, -1.5, 1.5, -2, -1.5, 0.5, -1.5, -1.5]$$

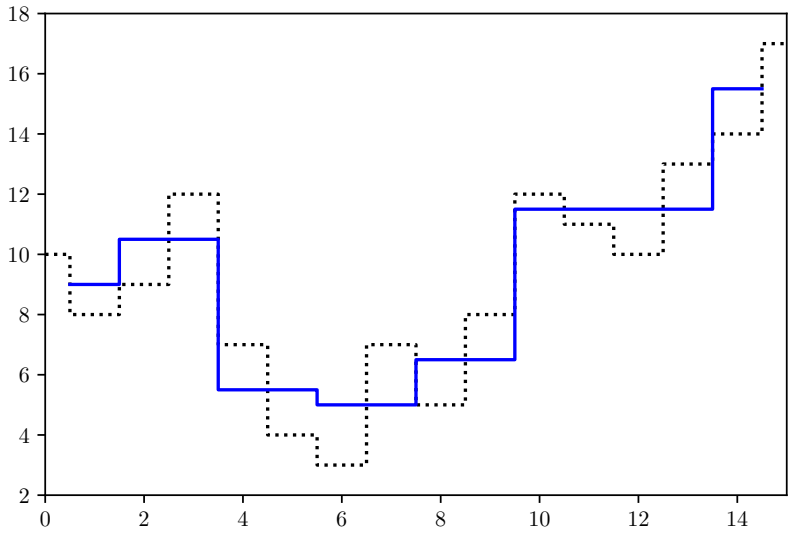


Figure 3.2: *Plot of the* approximation *A of the vector V (blue). The dotted black line illustrates the details that have been smoothed out.*

The vector A is an approximation of the vector V, which has been smoothed by removing the details found in D. The values of A and D are therefore called the *approximation coefficients* and *detail coefficients* respectively (see figure 3.2).

The number of values of A and D is half the number of values of V. The total number of values is therefore unchanged. As a consequence, authors often describe a version of the decomposition algorithm that modifies the input vector in place, setting the first half to the approximations and the second half to the details.

It should be noted that no information has been lost and the following procedure can be used to rebuild V from A and D:

Procedure *recompose(A, D)*
  $V \leftarrow [\,]$
  for $i \leftarrow 1$ **to** $length(A)$ **do**
    $V.append(A[i] + D[i])$
    $V.append(A[i] - D[i])$
  end
  return V
end

Algorithm 3.3: *Recomposition.*

The decomposition can then be applied recursively to approximation coefficients.

$$
\begin{aligned}
A_1 &= [9, 10.5, 5.5, 5, 6.5, 11.5, 11.5, 15.5] \\
D_1 &= [1, -1.5, 1.5, -2, -1.5, 0.5, -1.5, -1.5] \\
A_2 &= [9.75, 5.25, 9.0, 13.5] \\
D_2 &= [-0.75, 0.25, -2.5, -2.0] \\
A_3 &= [7.5, 11.25] \\
D_3 &= [2.25, -2.25] \\
A_4 &= [9.375] \\
D_4 &= [-1.875]
\end{aligned}
$$

This process is called *multilevel decomposition*, and assuming that the number of level is n, its output is written as:

$$(A_n, D_n, D_{n-1}, \ldots, D_2, D_1) \tag{3.1}$$

Using the example above, the full multilevel decomposition of V gives the following set of coefficients:

$$
\begin{aligned}
([9.375], [-1.875], [2.25, -2.25], [-0.75, 0.25, -2.5, -2.0], \\
[1, -1.5, 1.5, -2, -1.5, 0.5, -1.5, -1.5])
\end{aligned}
$$

3.1.2 Compression

Once the vector has been decomposed into approximation and detail coefficients, it can be compressed by zeroing some details and then performing recomposition (see figure 3.3 on the following page).

Zeroing the first level of details (the one computed by the first decomposition) gives:

$$([9.375], [-1.875], [2.25, -2.25], [-0.75, 0.25, -2.5, -2.0], \mathbf{[0, 0, 0, 0, 0, 0, 0, 0]})$$

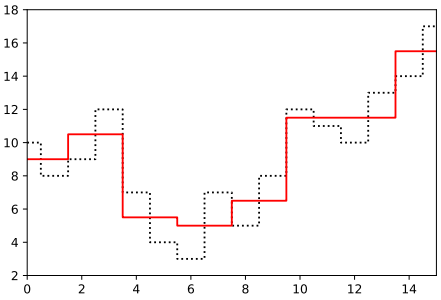
which is recomposed into the curve shown in figure 3.3a on the next page.

It should be noted that for the maximum compression level, only the mean value remains (figure 3.3d on the following page).

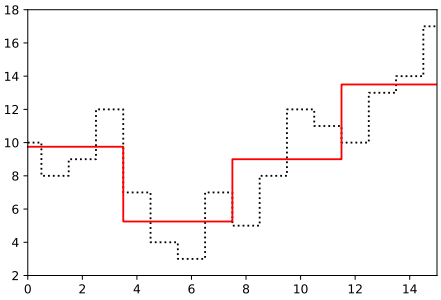
3.1.3 Conservation of energy

The energy of a signal f is defined as (Walker (2005)):

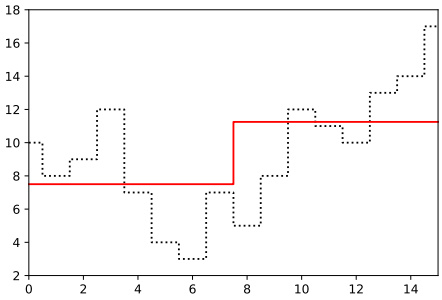
$$E = f_1^2 + f_2^2 + \ldots + f_n^2 \tag{3.2}$$



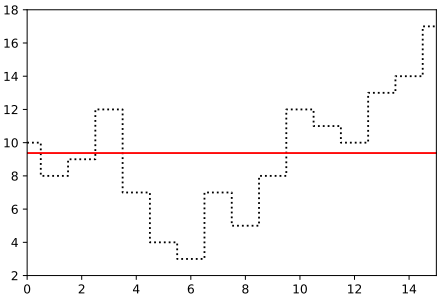
(a) *Compression level 1.*



(b) *Compression level 2.*



(c) *Compression level 3.*



(d) *Compression level 4.*

Figure 3.3: *Effect on various compression levels on vector V.*

The energy of the signal V is therefore:

$$\begin{aligned} E &= 10^2 + 8^2 + 9^2 + 12^2 + 7^2 + 4^2 + 3^2 + 7^2 + 5^2 + 8^2 + 12^2 + 11^2 + 10^2 \\ &\quad + 13^2 + 14^2 + 17^2 \\ &= 1620 \end{aligned}$$

Computing the energy using the approximation and detail coefficients will give

$$\begin{aligned} E_A &= 9^2 + 10.5^2 + 5.5^2 + 5^2 + 6.5^2 + 11.5^2 + 11.5^2 + 15.5^2 \\ &= 793.50 \\ E_D &= 1^2 + (-1.5)^2 + 1.5^2 + (-2)^2 + (1.5)^2 + 0.5^2 + (1.5)^2 + (1.5)^2 \\ &= 16.50 \\ E_A + E_D &= 810 \end{aligned}$$

The energy is now *half* of its original value. In order to conserve energy, a factor $\sqrt{2}$ is introduced during the decomposition. By simplifying $\sqrt{2}/2$ as $1/\sqrt{2}$, the decomposition algorithm becomes:

Procedure *decompose(V)*
  $A \leftarrow [\,]$
  $D \leftarrow [\,]$
  for $i \leftarrow 1$ **to** $length(V)$ **by** 2 **do**
    $A.append((V[i] + V[i+1])/\sqrt{2})$
    $D.append((V[i] - V[i+1])/\sqrt{2})$
  end
  return (A, D)
end

Algorithm 3.4: *Decomposition of vector with conservation of energy.*

And the recomposition algorithm becomes:

Procedure *recompose(A, D)*
  $V \leftarrow [\,]$
  for $i \leftarrow 1$ **to** $length(A)$ **do**
    $V.append((A[i] + D[i])/\sqrt{2})$
    $V.append((A[i] - D[i])/\sqrt{2})$
  end
  return V
end

Algorithm 3.5: *Recomposition.*

Using the new energy conserving decomposition:

$$A = [10\sqrt{2}, 21\sqrt{2}, 11\sqrt{2}, 10\sqrt{2}, 13\sqrt{2}, 23\sqrt{2}, 23\sqrt{2}, 31\sqrt{2}]$$
$$D = [2\sqrt{2}, -3\sqrt{2}, 3\sqrt{2}, -4\sqrt{2}, -3\sqrt{2}, \sqrt{2}, -3\sqrt{2}, -3\sqrt{3}]$$

and

$$E_A = 1587$$
$$E_D = 33$$
$$E_A + E_D = 1620$$

It should be noted that the approximation coefficients are half the size of the original signal but represent 98% of the original signal energy, while the detail coefficients only represent 2%.

3.1.4 Discrete wavelet transform

The process that has been illustrated in the preceding section is called the *Discrete Wavelet Transform* (DWT). The function *decompose* introduced previously is called recursively until a desired level of detail is reached:

Procedure *DWT(V, level)*
  if $level = 0$ **then**
    return V
  end
  $(A, D) \leftarrow decompose(V)$
  $R.append(DWT(A, level - 1))$
  $R.append(D))$
  return R
end

Algorithm 3.6: *Discrete wavelet transform.*

The level maximum is $log_2(length(V))$ as the number of approximation coefficient is halved at each iteration. The reverse process is called the *Inverse Discrete*

Wavelet Transform (IDWT).

Procedure *IDWT(V, level)*
  if $level = 0$ **then**
    return V
  end
  $L \leftarrow length(V)$
  $A \leftarrow V[1 \dots L/2]$
  $D \leftarrow V[L/2 + 1 \dots L]$
  return $recompose(IDWT(A, level - 1), D)$
end

Algorithm 3.7: *Inverse discrete wavelet transform.*

3.1.5 2D wavelets decomposition

DWT can easily be extended to 2 dimensions (Eckley (2001)), by performing the DWT on the first dimension, then on the second. If V is a 2D matrix, we will denote $V[i, *]$ the 1D vector representing the row i and $V[*, j]$ the 1D vector representing the column j. With this notation, the per-row wavelet decomposition can be written as:

Procedure *decompose-rows(V)*
  $R \leftarrow V$
  $L \leftarrow \sqrt{(length(V))}$
  for $i \leftarrow 1$ **to** L **do**
    $(A, D) \leftarrow decompose(V[i, *])$
    $R[1 \dots L/2, *] \leftarrow A$
    $R[L/2 + 1 \dots L, *] \leftarrow D$
  end
  return R
end

Algorithm 3.8: *Rows-wise decomposition of a 2D signal.*

and the per-column wavelet decomposition can be written as:

Procedure *decompose-columns(V)*
  $R \leftarrow V$
  $L \leftarrow \sqrt{(length(V))}$
  for $i \leftarrow 1$ **to** L **do**
    $(A, D) \leftarrow decompose(V[*, i])$
    $R[*, 1 \ldots L/2] \leftarrow A$
    $R[*, L/2 + 1 \ldots L] \leftarrow D$
  end
  return R
end

Algorithm 3.9: *Column-wise decomposition of a 2D signal.*

and a 2D decomposition simply becomes:

Procedure *decompose-matrix(V)*
  return *decompose-columns*(*decompose-rows*(V))
end

Algorithm 3.10: *Wavelet decomposition of a 2D signal.*

There are two algorithms to decompose a 2D signal at multiple levels (Stollnitz et al. (1995a)):

- standard decomposition: first decompose each row to the desired level, then decompose each column to the same level;
- non-standard decomposition: alternatively decompose each row and column to the desired level (see figure 3.5 on page 34).

The standard decomposition is given by the following algorithm, and is illustrated in figure 3.4 on the facing page:

Procedure *standard-2D-DWT(V, level)*
  for $i \leftarrow 1$ **to** *level* **do**
    $V \leftarrow$ *decompose-rows*(V)
  end
  for $i \leftarrow 1$ **to** *level* **do**
    $V \leftarrow$ *decompose-columns*(V)
  end
  return V
end

Algorithm 3.11: *Standard 2D decomposition.*

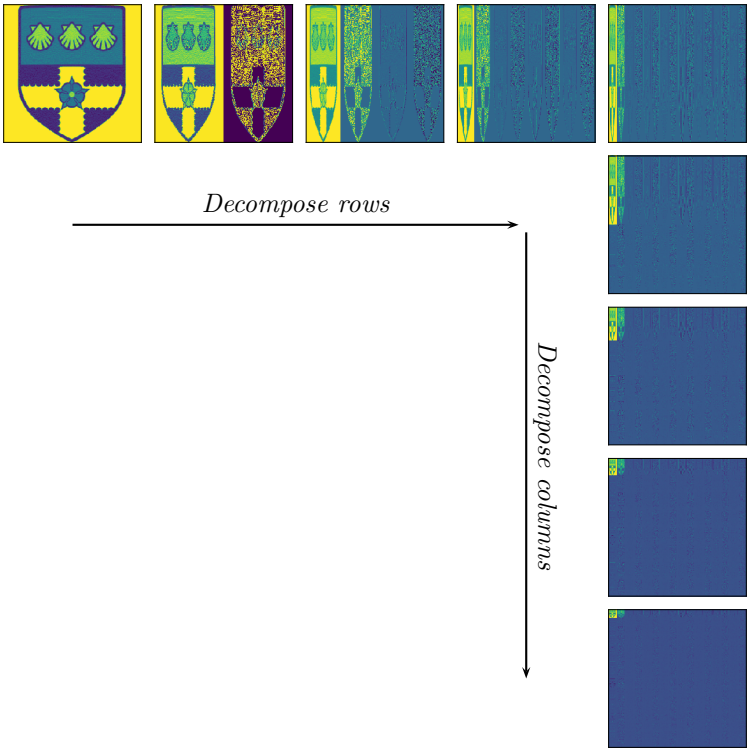


Figure 3.4: *2D wavelet decomposition of the University of Reading crest, using the standard method: decompose first by rows to the desired level, then decompose by columns to the same level. The colourmap used has been selected so that small details are visible.*

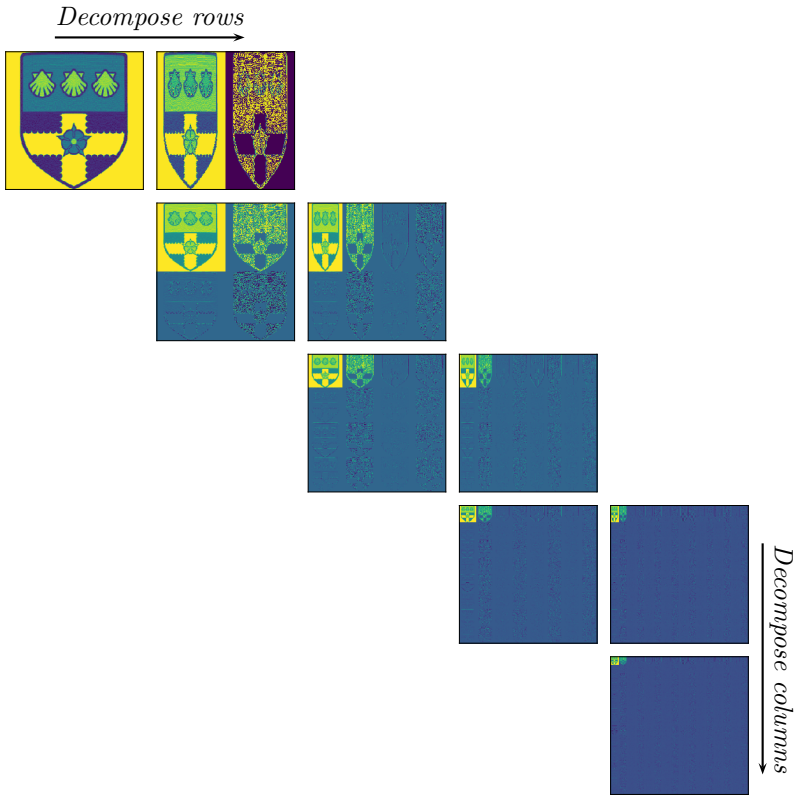


Figure 3.5: *2D wavelet decomposition of the University of Reading crest, using the non-standard method: decompose by row and by column alternatively.*

The non-standard decomposition is illustrated in figure 3.5, and can be written as:

Procedure *non-standard-2D-DWT(V, level)*
  for $i \leftarrow 1$ **to** *level* **do**
    $V \leftarrow$ *decompose-rows*(V)
    $V \leftarrow$ *decompose-columns*(V)
  end
  return V
end

Algorithm 3.12: *Non-standard 2D decomposition.*

The algorithm for the inverse 2D transform would be identical, and is omitted for the sake of conciseness.

3.2 Wavelets and multi-resolution analysis

There is ample literature about the mathematical foundations of wavelets (Keinert (2003); Jawerth and Sweldens (1994); Walker (2005)). Multi-resolution analysis

(MRA) provides the mathematical foundations for wavelet transforms. Intuitively, MRA considers how functions from $L^2(\mathbb{R})$, the space of square integrable functions, can be expressed as a linear combination of functions ψ_i that represent different scales of $L^2(\mathbb{R})$, and provides some constraints on how wavelets function can be defined.

Wavelet transforms is the process by which a function $f(t) \in L^2(\mathbb{R})$ can be expressed as a linear combination of wavelets functions $\psi_i(t)$:

$$f(t) = \sum_i a_i \psi_i(t) \tag{3.3}$$

The wavelets functions are chosen so that they form a *orthonormal basis* of $L^2(\mathbb{R})$:

$$\langle \psi_i(t), \psi_j(t) \rangle = 0 \qquad i \neq j \tag{3.4}$$

and

$$\langle \psi_i(t), \psi_i(t) \rangle = 1 \qquad \forall i \tag{3.5}$$

with $\langle g, h \rangle = \int g(x)h(x)dx$ being the inner product of functions g and h. If ψ_i form an orthonormal basis, the coefficients a_k can be calculated as:

$$a_i = \langle f(t), \psi_i(i) \rangle = \int f(t)\psi_i(t)dt \tag{3.6}$$

There are many wavelets that satisfy the definition given above, and they have been extensively studied (Daubechies (1988)).

The process described in the previous section is called the *discrete wavelet decomposition* of the signal V, using *Haar wavelets*. Haar wavelets are named after the Hungarian mathematician Alfréd Haar which introduced such a decomposition in 1909, which was then called *Haar sequence*. The term *wavelet* was introduced later. The decomposition given in section 3.1 implicitly uses the Haar wavelet. The Haar wavelet (figure 3.6 on the next page) defined by its *scaling function*:

$$\phi(x) = \begin{cases} 1 & 0 \leq t < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

and its *mother wavelet*:

$$\psi(x) = \begin{cases} 1 & 0 \leq t < \frac{1}{2}, \\ -1 & \frac{1}{2} \leq t < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

Figure 3.7 on the following page shows how various values of $\phi_{i,j}$ and $\psi_{i,j}$ covers $L^2(\mathbb{R})$ as the i and j vary.

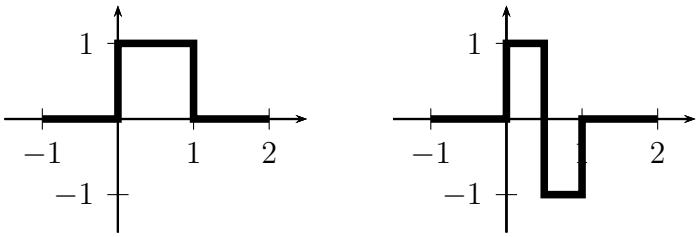
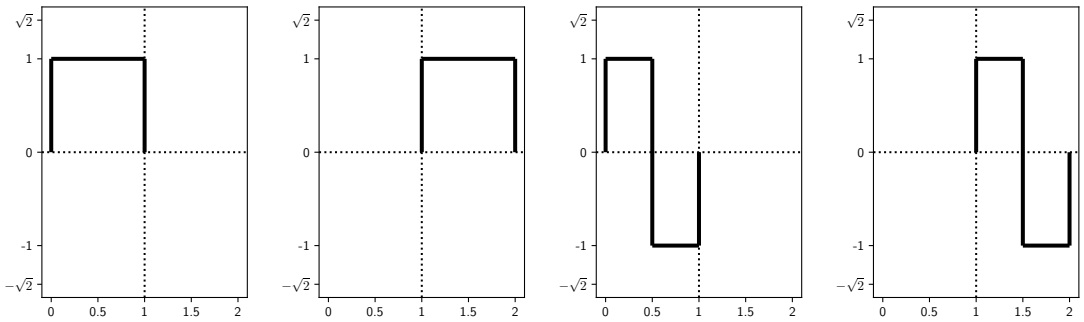
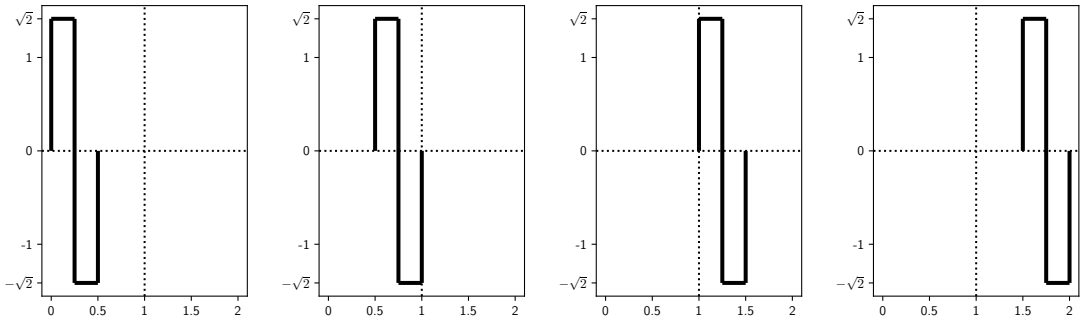


Figure 3.6: *Haar wavelets: left is the scaling function, right is the mother wavelet.*



(a) $\phi_{0,0}$ **(b)** $\phi_{0,1}$ **(c)** $\psi_{0,0}$ **(d)** $\psi_{0,1}$



(e) $\psi_{1,0}$ **(f)** $\psi_{1,1}$ **(g)** $\psi_{1,2}$ **(h)** $\psi_{1,3}$

Figure 3.7: *Haar wavelets: how different values of $\phi_{i,j}$ and $\psi_{i,j}$ covers part of $L^2(\mathbb{R})$ at various resolutions.*

So the signal:

$$V = [8, 6, 5, 7] \tag{3.9}$$

can be written as:

$$V = 8\phi_{0,0} + 6\phi_{0,1} + 5\phi_{0,2} + 7\phi_{0,3} \tag{3.10}$$

The summation is illustrated graphically in figure 3.8a on the next page, showing how V can be represented by the scaling functions $\phi_{i,j}$.

It can be decomposed once as (taking into account the scaling of $\sqrt{2}$):

$$V = 7\sqrt{2}\phi_{1,0} + 6\sqrt{2}\phi_{1,1} + \sqrt{2}\psi_{1,0} - \sqrt{2}\psi_{1,1} \tag{3.11}$$

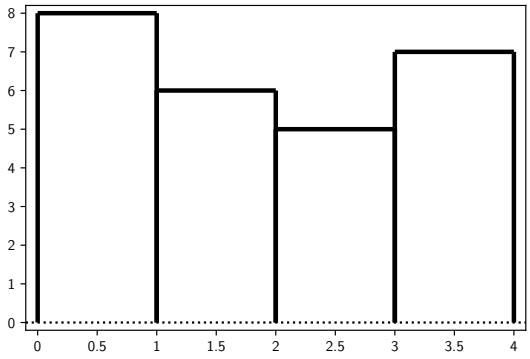
The factor $\sqrt{2}$ that was introduced to preserve the energy of the signal during its decomposition is also used to ensure that the norms of the $\phi_{i,j}$ and $\psi_{i,j}$ are 1.

Figures 3.8a and 3.8b show the respective contributions of the scaling functions $\phi_{i,j}$ and the wavelet functions $\psi_{i,j}$. The resulting sum is shown in figure 3.8c.

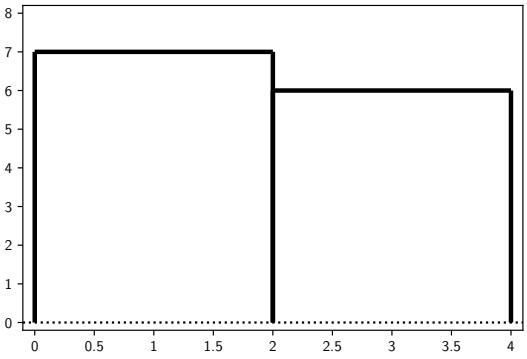
The signal can be further decomposed as:

$$V = 13\phi_{2,0} + \psi_{2,0} + \sqrt{2}\psi_{1,0} - \sqrt{2}\psi_{1,1} \tag{3.12}$$

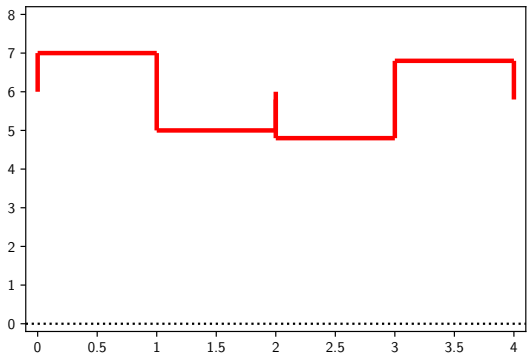
Figure 3.9 on page 39 shows that decomposition. The scaling function ϕ now captures the average value of the signal, while the first wavelet captures the first level of details. The second level of details is represented by the second level of wavelets.



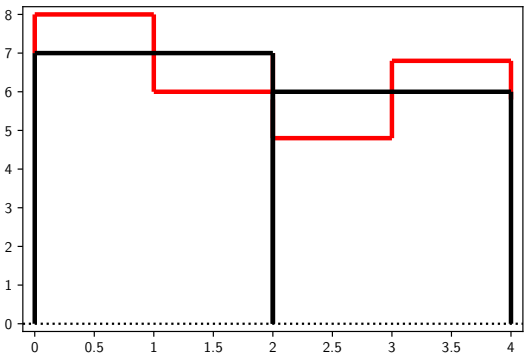
(a) $8\phi_{0,0} + 6\phi_{0,1} + 5\phi_{0,2} + 7\phi_{0,3}$



(b) $7\sqrt{2}\phi_{1,0} + 6\sqrt{2}\phi_{1,1}$

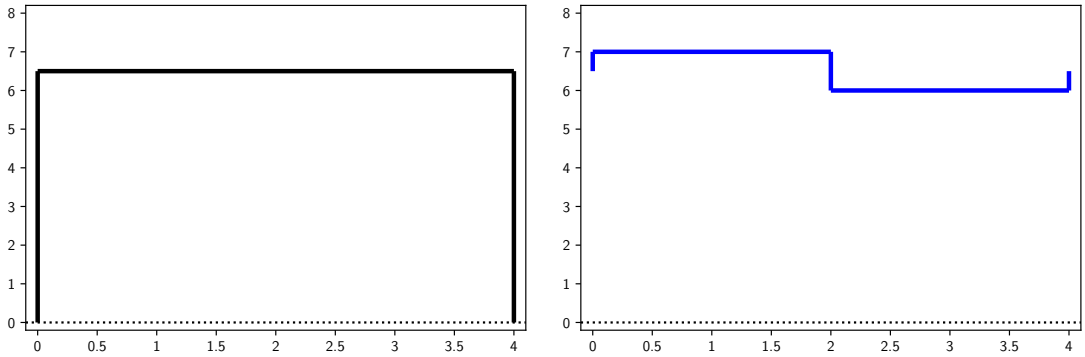


(c) $\sqrt{2}\psi_{1,0} - \sqrt{2}\psi_{1,1}$

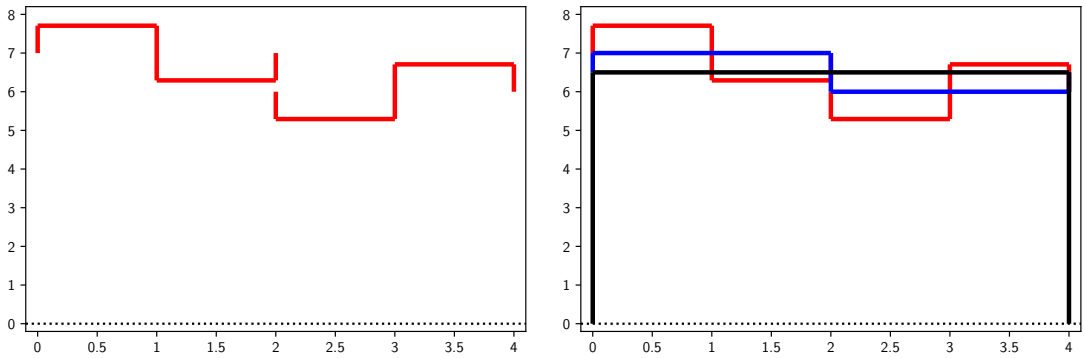


(d) $7\sqrt{2}\phi_{1,0} + 6\sqrt{2}\phi_{1,1} + \sqrt{2}\psi_{1,0} - \sqrt{2}\psi_{1,1}$

Figure 3.8: *Decomposition of the signal* $[8, 6, 5, 7]$. *Figure 3.8a represents the original signal as a linear combination of the Haar scaling functions* $\phi_{i,j}$. *Figure 3.8b is the first level approximation of the signal, as a linear combination of the Haar scaling functions. Figure 3.8c is the first level details of the signal as a linear combination of the Haar wavelets* $\psi_{1,j}$. *Although the wavelet functions* $\phi_{i,j}$ *are periodic and should therefore be around* $y = 0$, *they are vertically positioned on the plot in relation to the corresponding scaling functions, for illustration purposes. Figure 3.8d shows that the sum of the approximation and details captures fully the original signal.*



(a) $13\phi_{2,0}$ **(b)** $\psi_{2,0}$



(c) $\sqrt{2}\psi_{1,0} - \sqrt{2}\psi_{1,1}$ **(d)** $13\phi_{2,0} + \psi_{2,0} + \sqrt{2}\psi_{1,0} - \sqrt{2}\psi_{1,1}$

Figure 3.9: *Further decomposition of the signal* $[8, 6, 5, 7]$. *As for figure 3.8, the supporting scaling function* ϕ *(black) is plotted in figure 3.9a. It is a constant function that captures the average value of the signal. Figure 3.9b shows the wavelet function that represents the first level of details (blue). Figure 3.9c shows the wavelets representing the second level of details (red). Figure 3.9d show that the original signal can be fully reconstructed from the scaling function and the wavelet functions.*

Chapter 4

Fingerprinting

One of the primary goals of this research is to find a method to construct fingerprints from meteorological fields such that these fingerprints can be used as proxies to the original fields when looking for nearest neighbours. This means that distances between fingerprints should reflect the distance between the fields from which they were extracted.

Wavelets have been used successfully to implement fingerprint-based image or audio retrieval systems (see section 1.5.2). We will consider how this can be applied to meteorological fields.

We will first define the problem, then study the distances between the meteorological fields, in order to define a working set and establish a minimum distance threshold below which fields are considered to be "close enough" to be considered similar.

4.1 Problem definition

The problem we are trying to address can be formalised as:

- let v be a meteorological variable (e.g. *mean sea level pressure*, *10m wind speed*...);

- let $\mathcal{A}_v$ be the set of all meteorological fields in the archive for this variable. Assuming that all the fields are defined over the same grid (same geographical coverage, same resolution), $\mathcal{A}_v$ can be considered a subset of $\mathbb{R}^n$, with n being the number of grid points;

- let D be a distance function between the elements of A_v (typically the Euclidean distance, also known as the $L2$-norm);
- let $\mathcal{F}_v$ be the set of fingerprints for the variable v;
- let δ be a distance function between the elements of $\mathcal{F}_v$.

We are looking for a mapping $F_v \colon \mathcal{A}_v \mapsto \mathcal{F}_v$ such that:

$$D(f_1, f_2) \leq D(f_1, f_3) \Leftrightarrow \delta(F_v(f_1), F_v(f_2)) \leq \delta(F_v(f_1), F_v(f_3)) \quad \forall f_1, f_2, f_3 \in \mathcal{A}_v \ . \tag{4.1}$$

Intuitively, this means that F_v "preserves distances", i.e. if fields are close according to the distance D, their fingerprints must also be close according to the distance δ. Similarly, fields that are far apart must have fingerprints that are far apart. A study of distance preserving embeddings can be found in Indyk and Naor (2007).

The aim of this work is to find a mapping that *mostly satisfies* relation 4.1, i.e. a mapping for which the relation is true for most elements of $\mathcal{A}_v$.

4.2 Effectiveness of the mapping

As we are considering various fingerprinting schemes, we will compare how effective they are. We define the error ε_D of a mapping as a measure of the number of elements of $\mathcal{A}_v$ for which relation 4.1 does not hold.

A scheme is perfectly effective if, for every query q, we always find the field which is closest to q according to the distance D.

In order to measure the error of a fingerprinting scheme, we will define the $\varepsilon_D(q) = 0$ as the error of a single query q with respect to the distance D as:

$$\varepsilon_D(q) = D(NN_D(q), NN_\delta(q)) \tag{4.2}$$

where $NN_D(q)$ is q's nearest neighbour according to the distance D, and $NN_\delta(q)$ is q's nearest neighbour according to the distance between fingerprints δ. The error $\varepsilon_D(q)$ is therefore the distance between both nearest neighbours, and will be zero if they are the same. This is illustrated in figure 4.1 on the next page.

We consider a scheme to be valid if $\varepsilon_D(q)$ is negligibly small for a large number of values of q over a given working set. In the following sections, we will establish for which value $\varepsilon_D(q)$ is considered small.

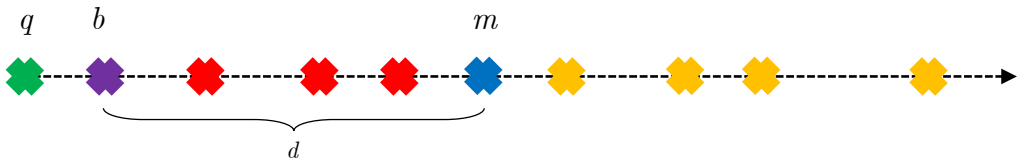


Figure 4.1: *This figure illustrates the computation of $\varepsilon_D(q)$. Each coloured cross shows a different field. The fields are organised from left to right according to their distance to the field q (green). Let us assume that when comparing the distances between fingerprints, the closest match is the field m (blue). This is not the closest match according to the distance D (it should have been the purple field). The error is proportional to distance $d = D(b, m)$.*

4.3 Study of distances between fields

Traditionally, distances between meteorological fields are computed using the root mean square deviation (RMSD), which is equivalent to the Euclidean distance. Other distances such, as the Pearson correlation coefficient (PCC) are also used. Mo et al. (2014) show the limitations of such metrics.

Since the fields that we are studying can be considered as vectors of $\mathbb{R}^{1024}$ (1024 being the number of grid points), we are affected by the curse of dimensionality: as the number of dimensions increases, the difference between the distance to the closest neighbours and the furthest neighbours tends to zero (Beyer et al. (1999)).

This means that even traditional metrics like the Euclidian distance are not always effective when comparing fields, especially when the weather patterns are slightly shifted in time or space. Nevertheless, as the Euclidean distance is the most commonly used metric in meteorology, we use it as the reference distance to which we will compare the distances between fingerprints.

4.3.1 Working set

The dataset we are considering is composed of hourly fields for the period 1979-2018 (see section 2.2 on page 16). This amount to over 350 thousand fields per variable. Because computing distances between every field would be prohibitive, we select a subset, called thereafter the *working set*, composed of one field per day over the chosen period. This lowers the total number of fields per variable to 14 610, which is much more manageable. All data from the working set are retrieved from the MARS archive and then interpolated to the desired grid and area.

4.3.2 Distance matrix

In order to study the distances between fields, we will first precompute the distance between every pair of fields in the working set, using Euclidean distance. This amounts to around 106 million distances per variable ($14610 \times (14610+1)/2$). On the hardware used for the project (see section 7.3 on page 122), we compute around 500 distances per second, including loading the fields in memory and decoding them, as well as persisting the results to disk. Computing all distances would last in the order of two and a half days to compute, using a single computation thread.

The results are collected in a distance matrix $\mathcal{D}$. For efficiency, this matrix is stored in a memory-mapped file.

Given a date *date*, the index of that date in the matrix $\mathcal{D}$ is:

$$index(date) = julian(date) - julian(\text{1979-01-01}) + 1 \tag{4.3}$$

where $julian()$ is a function returning the Julian day (Hatcher (1984)). Lookups in the distance matrix are then simply done using the index two dates:

$$distance(date_m, date_n) = \mathcal{D}_{i,j} \text{ with } i = index(date_m) \text{ and } j = index(date_n) \tag{4.4}$$

The distances from a $date_n$ can be looked up efficiently by reading the values of column $j = index(date_n)$:

$$\begin{bmatrix} 0 & \mathcal{D}_{1,2} & \dots & \mathcal{D}_{1,j} & \dots & \mathcal{D}_{1,n} \\ \mathcal{D}_{2,1} & 0 & \dots & \mathcal{D}_{2,j} & \dots & \mathcal{D}_{2,n} \\ \vdots & \vdots & & \vdots & & \vdots \\ \mathcal{D}_{i,1} & \mathcal{D}_{i,2} & \dots & \mathcal{D}_{i,j} & \dots & \mathcal{D}_{i,n} \\ \vdots & \vdots & & \vdots & & \vdots \\ \mathcal{D}_{n,1} & \mathcal{D}_{n,2} & \dots & \mathcal{D}_{n,j} & \dots & 0 \end{bmatrix} \tag{4.5}$$

The distance matrix $\mathcal{D}$ can then be used to compute the nearest neighbour of a field with index j:

$$NN(j) = \min(\{\mathcal{D}_{i,j} : i \in [1\,..\,n], i \neq j\}) \tag{4.6}$$

4.3.3 Distribution of distances

Using the distance matrix $\mathcal{D}$ defined previously, we plot the distributions of distances in figures 4.2 on page 46 and 4.3 on page 47.

With the exception of two fields (*snow depth* and *snowfall*), the distributions of distances have bell-shape that agrees with the findings of Thirey and Hickman (2015) on the study of distributions of distances in high dimensions.

We continue our study by looking at fields similarities, by clustering the data using the *k-means* algorithm. We arbitrarily set the number of clusters to nine, in order to get a feeling of the distribution of types of weather in the archive for each variable. The aim of this clustering is to assess if the weather patterns are evenly distributed, and therefore the clusters are of comparable sizes, or if some weather types are more frequent than others, and the clustering show unevenly sized clusters. For that purpose, the results of the clustering are plotted on a silhouettes graph (Rousseeuw (1987)).

The silhouettes are a measure of the quality of the clustering: assuming that an element i has been clustered in cluster C_i, we define $a(i)$ as the average distance between i and the other members of the same cluster:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i,\, i \neq j} D(i, j) \tag{4.7}$$

with $D(i, j)$ being the distance between i and j and $|C_i|$ being the number of elements in cluster C_i. The term $|C_i| - 1$ means that we exclude i when computing the average.

We then define $b(i)$ as the average distance between i and the members of the closest other cluster (the closest here meaning the cluster for which the average distance between i and every cluster members is the minimum):

$$b(i) = \min_{i \neq j} \frac{1}{|C_j|} \sum_{j \in C_j} D(i, j) \tag{4.8}$$

We can now define the silhouette $s(i)$ of i as:

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases} \tag{4.9}$$

For elements that are the single member of a cluster, $s(i) = 0$ by convention.

If $s(i)$ is close to 1, this means member i is clustered well; $s(i)$ is close to -1, this means that the i should have been clustered in another cluster (the closest one). The results of the clustering are shown in figures 4.4 on page 48 and 4.5 on page 49. The vertical dotted red line is the average value of the silhouette over all members.

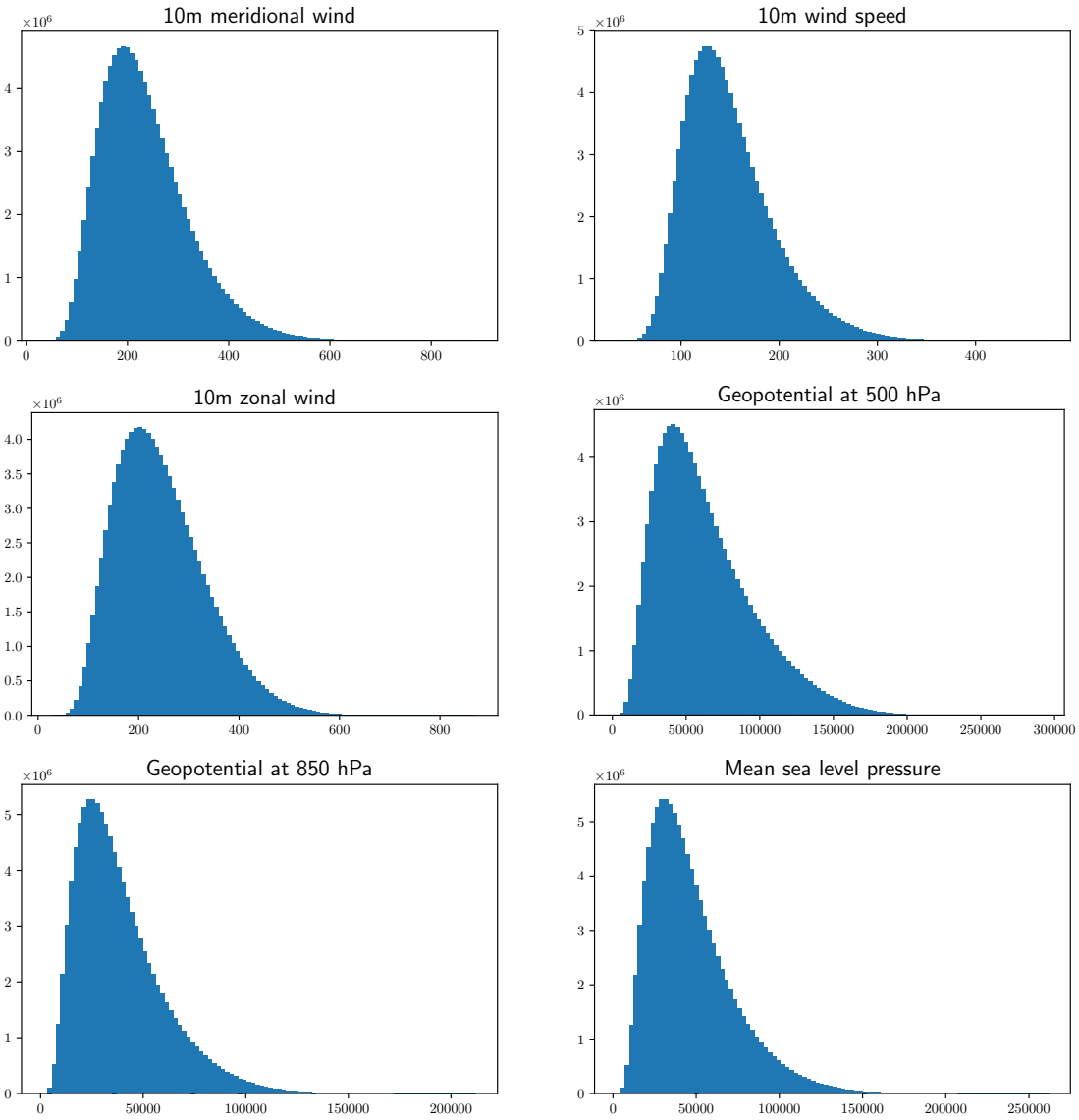


Figure 4.2: *Distribution of the distances between fields for each meteorological variable, with a number of bins set to 100. The number of fields considered is* 14 610*, which leads to approximately* 106 *million inter-field distances.*

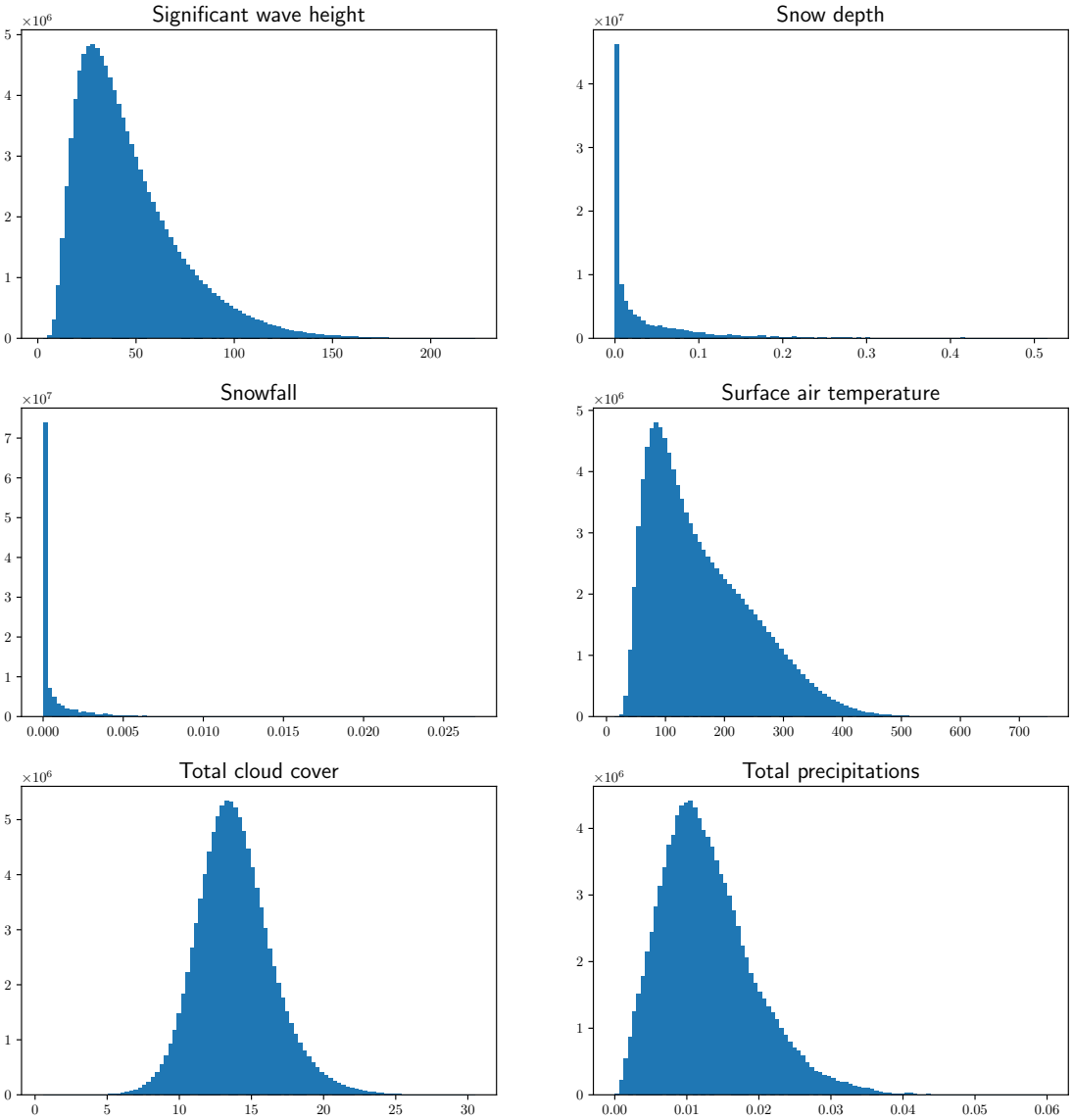


Figure 4.3: *Distribution of the distances between fields for each meteorological variable, with a number of bins set to 100.*

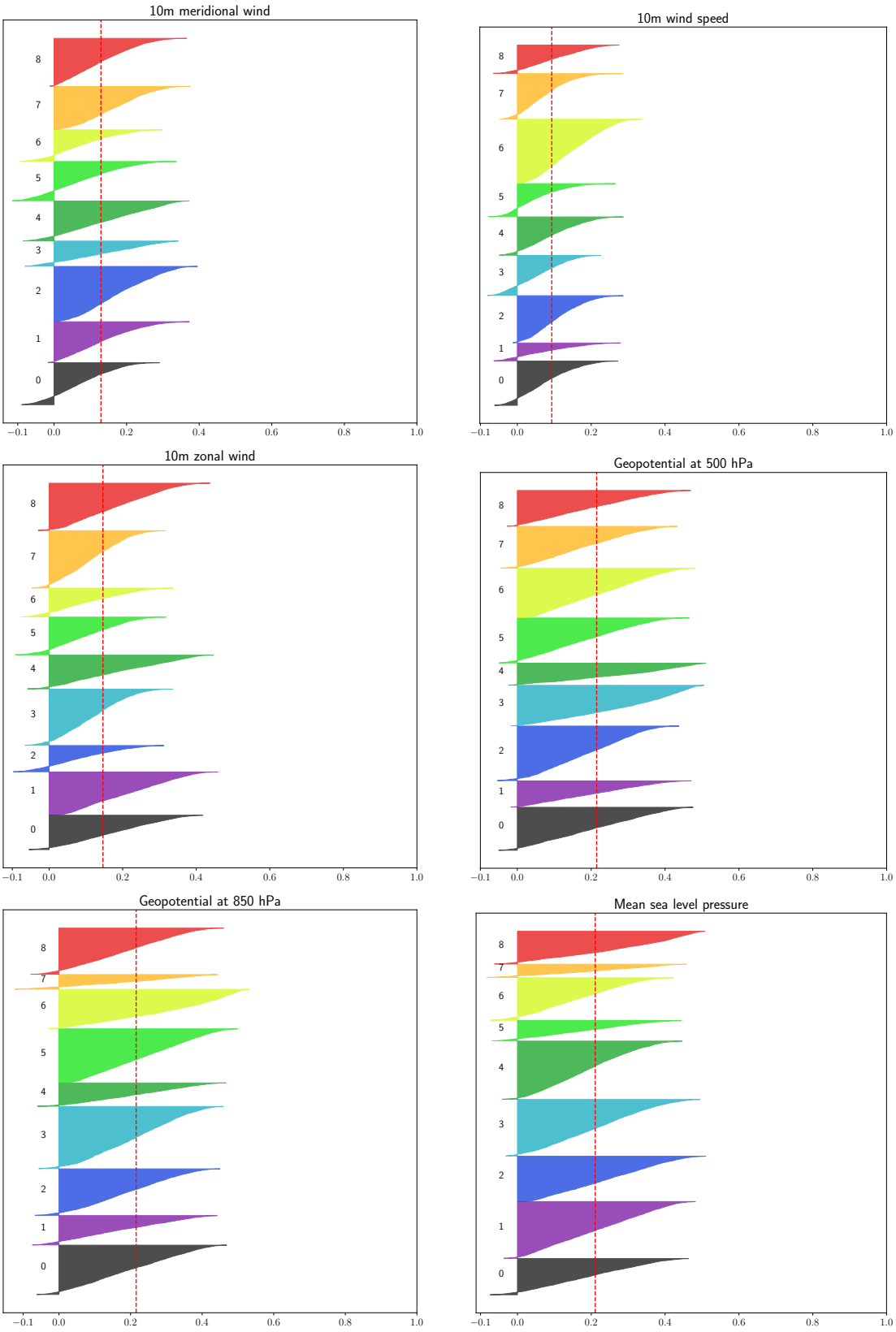


Figure 4.4: *Silhouettes plots. The horizontal axis represents the silhouette coefficient.*

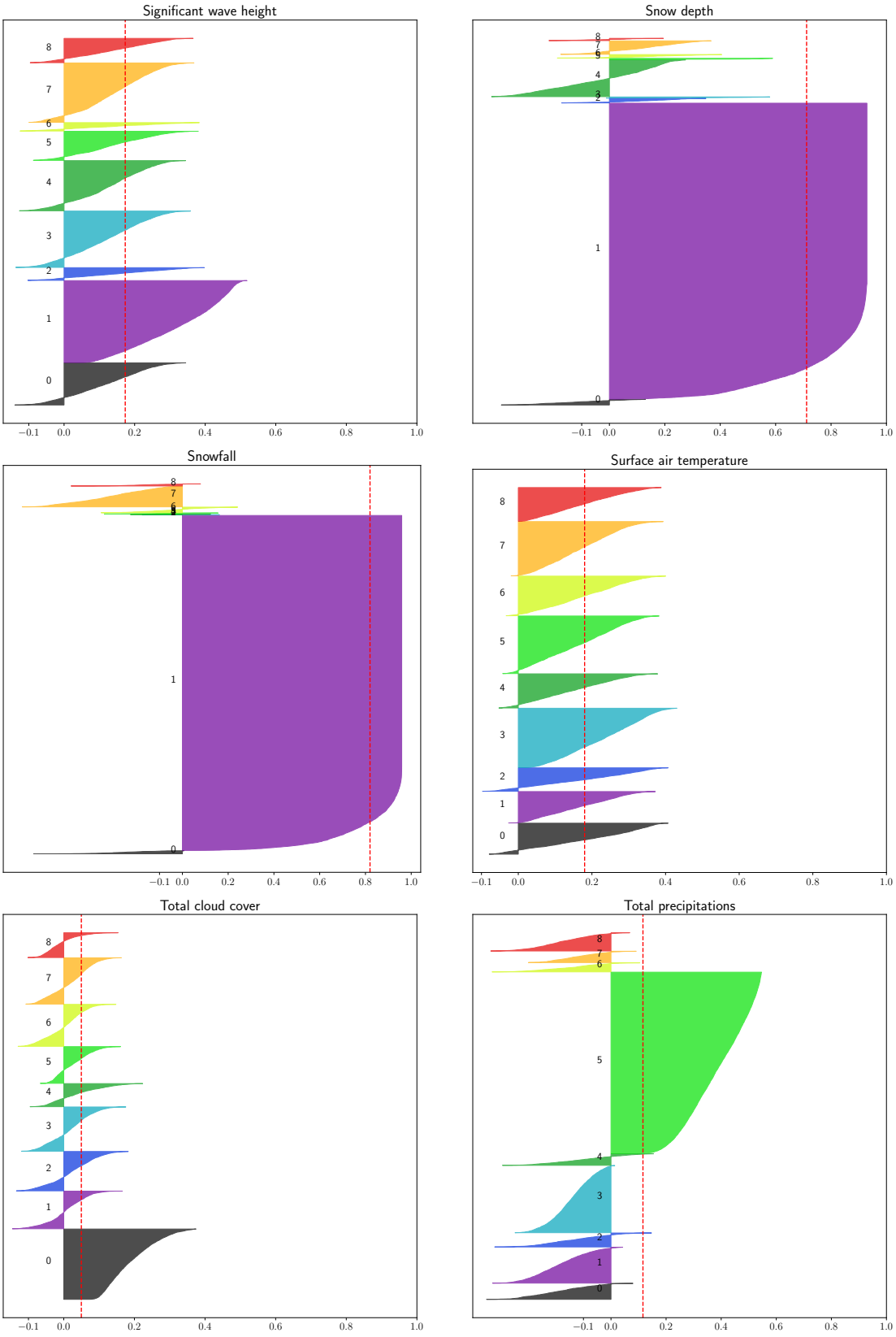


Figure 4.5: *Silhouettes plots. The horizontal axis represents the silhouette coefficient.*

The widths of the silhouettes represent the number of elements in each cluster. Plots showing silhouettes of approximately the same shape, such as for *geopotential at 500 hPa*, indicate that the variable can be clustered in evenly sized clusters and that its values are evenly distributed. When a plot shows a silhouette much wider than the others, this means that one cluster is much larger than the others and that the values represented by the variable are not evenly distributed. This is the case for *total precipitations*.

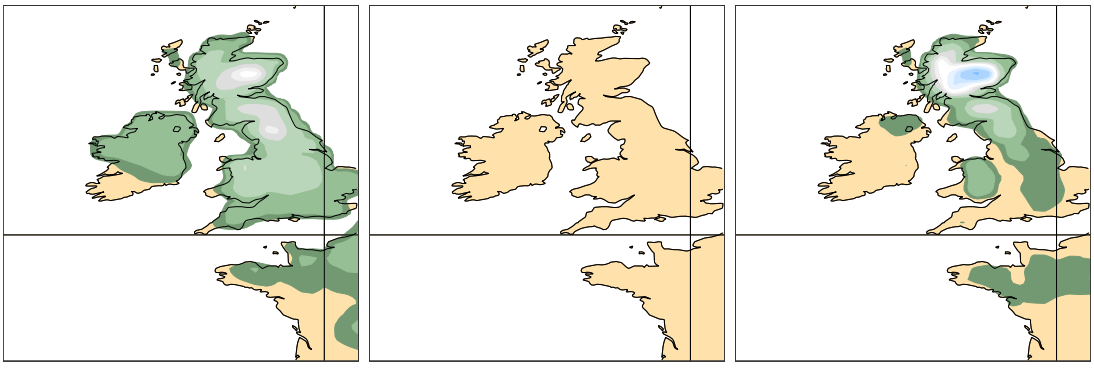
What these figure show is that they are two categories of meteorological variable: those like *10m wind speed*, *mean sea level pressure* or *surface air temperature* that can take a wide range of values, and therefore can be clustered in cluster of similar size, and those like *snowfall*, *snow depth* or *total precipitations* that have one cluster that is much larger than the other.

For *snowfall* and *snow depth* (figure 4.6 on the facing page), the large cluster represents days without snow, which is most of the days of the year. For *total precipitations* (figure 4.7 on page 52), these are the day with very little rain. For *significant wave height* these are the days with small waves. For *total cloud cover* these are the days that are overcast.

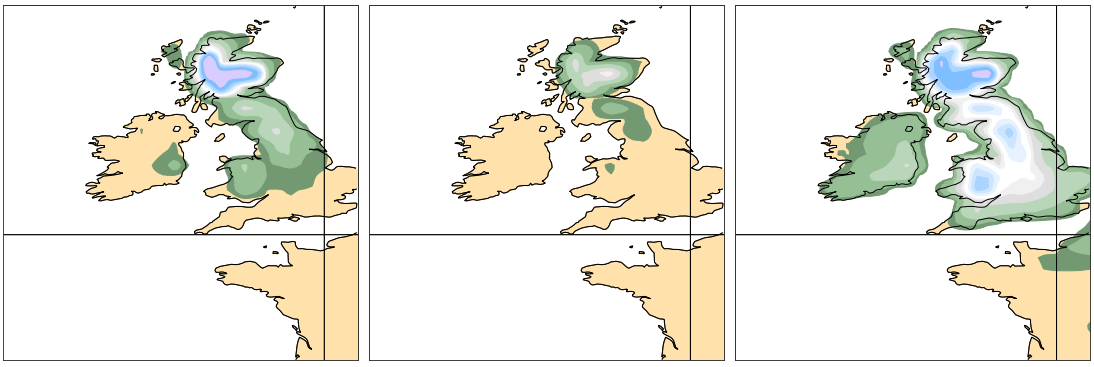
In order to ensure that the working set is composed of fields that are actually different from each other and that are representative of all kinds of weather pattern, we need to identify and remove the fields that are too similar to each other. For this, we will redo the clustering, this time using DBSCAN (Ester et al. (1996)), as this algorithm will automatically select the number of clusters given a maximum distance between cluster members: all members within that distance will be clustered together.

The algorithm is run with the smallest non-zero distance found in the distance matrix. A random member will then be selected from each cluster, to form the working set. This will ensure that the large ranges of constant fields, such as *snowfall* during summer days, are represented by a single instance in the working set.

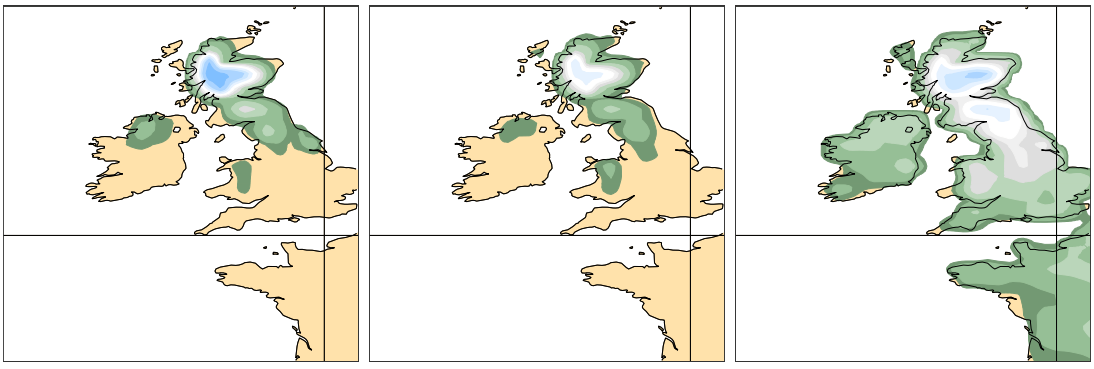
The effect of this process is to remove all constant fields but one from the working set. It does not affect fields such as *mean sea level pressure* or *surface air temperature*, but reduces greatly the number of fields for *snowfall*, *snow depth* and for a lesser extent, *total precipitations*. Figure 4.8 on page 53 shows the new distribution of distances between members of the working set for *snowfall*. Although the shape of the distribution is unchanged, the number of fields in the largest bin is greatly reduced.



(1) *2% of members* **(2)** *81% of members* **(3)** *1% of members*

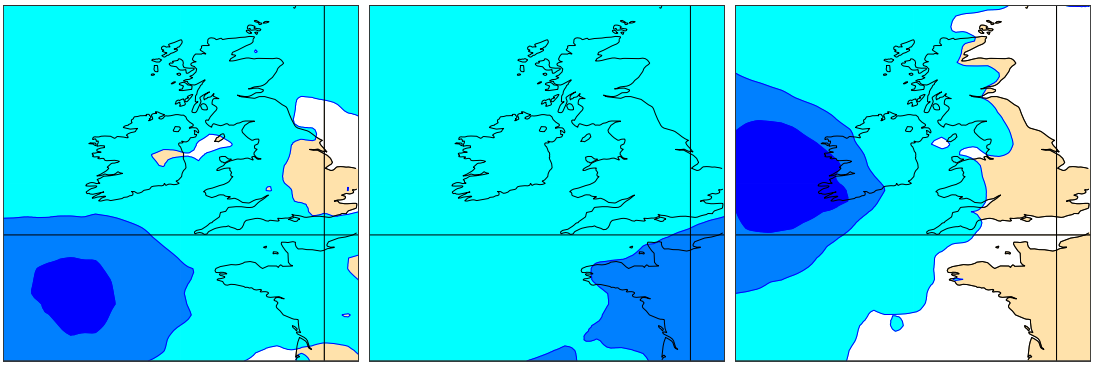


(4) *0.3% of members* **(5)** *10% of members* **(6)** *0.3% of members*

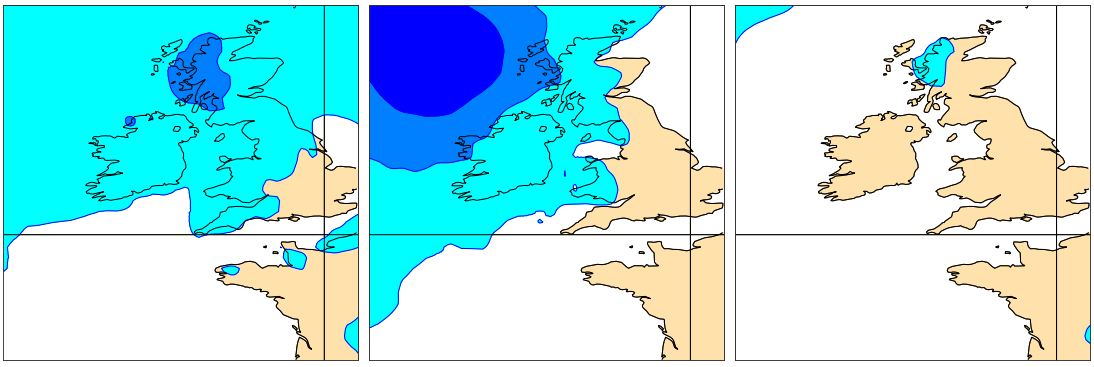


(7) *1% of members* **(8)** *4% of members* **(9)** *1% of members*

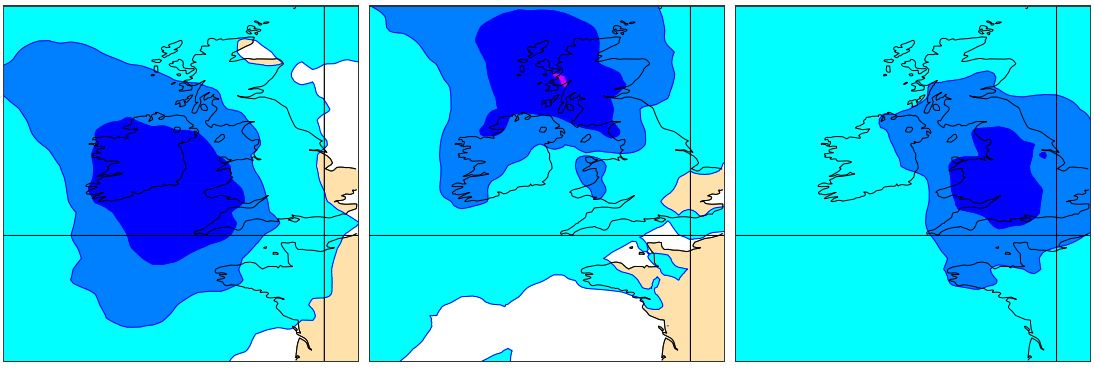
Figure 4.6: *K-Means centroids for* snow depth. *Cluster (2) corresponds to days without snow.*



(1) *4% of members* **(2)** *10% of members* **(3)** *4% of members*



(4) *18% of members* **(5)** *3% of members* **(6)** *50% of members*



(7) *2% of members* **(8)** *3% of members* **(9)** *5% of members*

Figure 4.7: *K-Means centroids for* total precipitations. *Cluster **(6)** corresponds to days of no or little rain.*

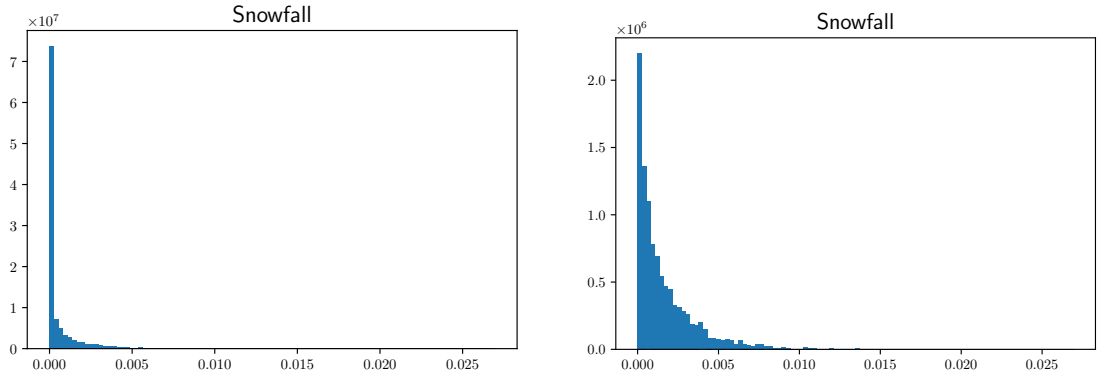


Figure 4.8: *Distribution of distances for the working set for the* snowfall *variable. On the left, the original distribution; on the right the distribution of the working set after selecting one member from each cluster. The number of bins is set to 100.*

4.3.4 Minimum distance threshold

In section 4.2, we have introduced the error ε_D as the distance between the field returned when searching for the nearest neighbour using the distance between fingerprints and the field returned when searching for the nearest neighbour using a distance D.

ε_D is therefore of the same nature as the output of D, i.e. has the same unit. As this study considers the Euclidean distance as D, ε_D is expressed in the same units as the fields themselves: for *10m wind speed*, the units will be $\mathrm{m\,s^{-1}}$.

We need to define a threshold, by which we consider the error ε_D to be "small enough", which means that a result of a query using the fingerprint distance is "close enough" to the actual nearest neighbour using Euclidean distance. To find this value, we use hierarchical clustering. In section 4.3.3 we performed k-means clustering with nine clusters to get a feel of how various types of weather were distributed in the working set. Using hierarchical clustering will allow us to find out the relationship between the number of clusters and the closeness of their members.

Hierarchical clustering (Lance and Williams (1967); Müllner (2011)) consists of starting with a set of single value clusters each containing a single element. Then the closest clusters are merged to form a new cluster. The newly created cluster is added to the set, and the merged clusters are discarded. The distance between the new cluster and the remaining clusters is computed according to a chosen *linkage method*.

The process is repeated iteratively until there is only one cluster left. Each cluster keeps track of the list of clusters that were combined to create it, thus creating

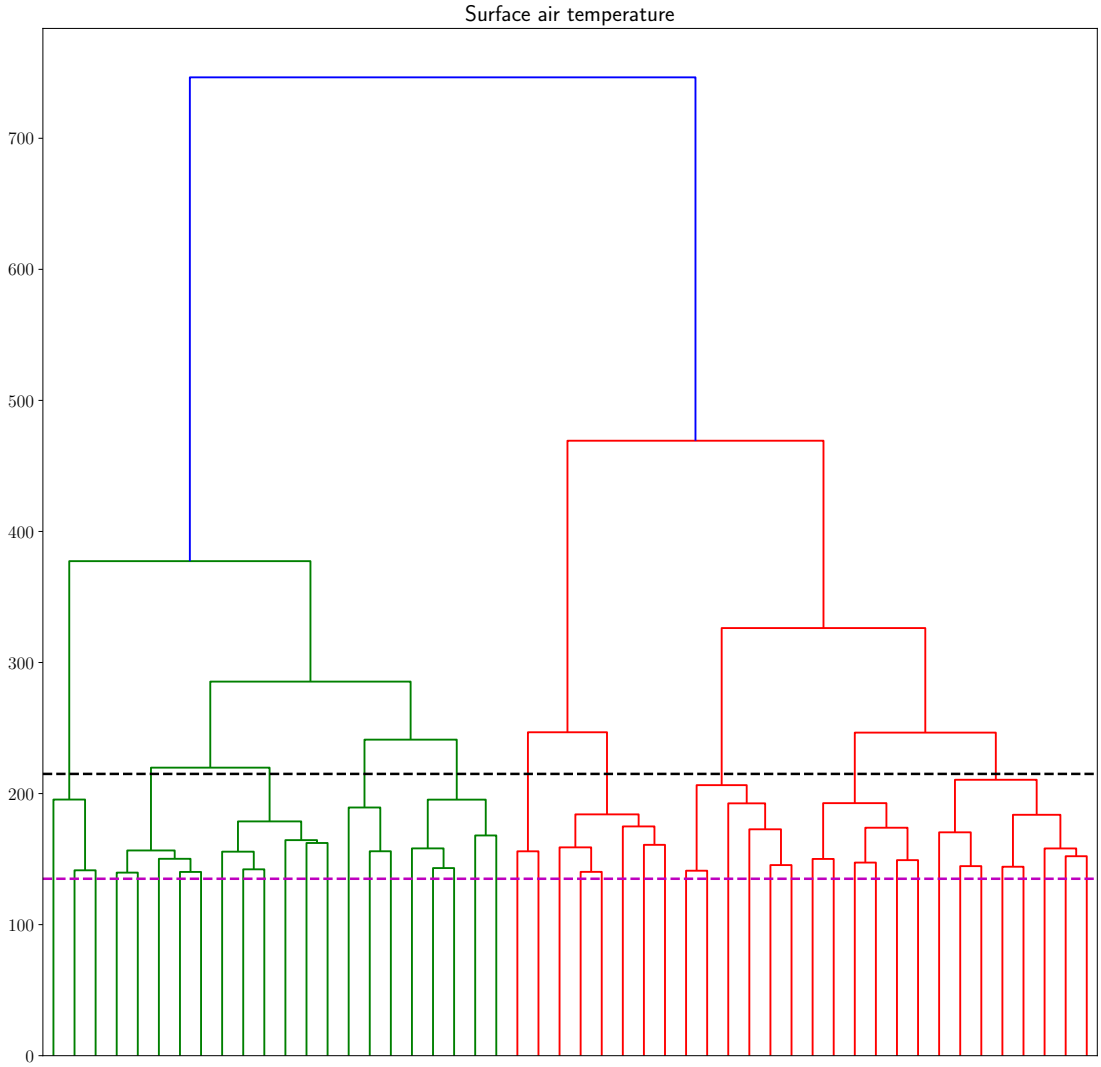


Figure 4.9: *Example of a dendrogram, showing the first few clustering levels of clusters for the* surface air temperature *variable, for the period 1979-2018. The horizontal lines represent the clusters. The vertical lines represent the link between a cluster and its children in the hierarchy. The y-axis represents the distance between child clusters. In that example, the root cluster is shown at* $y = 746$*, which means that the distance between its two children is* 746*. The two dashed lines illustrate two possible "cuts" of the tree. The black line is at* $y = 215$*, and crosses* 10 *vertical lines; this means that* 10 *clusters within which the inter-element distance is less than* 215*. The magenta line is at* $y = 135$*, and crosses* 50 *vertical lines. The clustering shown in this figure is done using the* complete-linkage method*, for the sake of clarity, as it produced a well-balanced tree.*

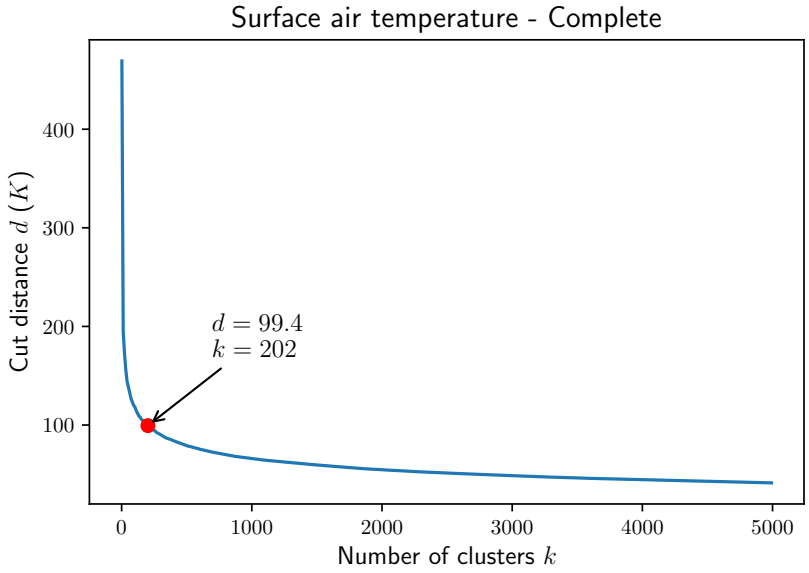


Figure 4.10: *Plot of the number of clusters against the dendrogram cutting distance for* surface air temperature*, using the* complete *linkage method. The red dot shows the position of the elbow found by the algorithm 4.1 on the following page. d is the cutting distance, k is the number of clusters.*

a hierarchy. Cluster hierarchies are plotted using a tree representation called a *dendrogram* (see figure 4.9 on the preceding page). One property of each cluster is the distance between its child clusters, which is shown on the y-axis of the dendrogram.

By "cutting" the dendrogram at a given $y = y_0$, i.e. a given distance, we define a number of clusters that have the characteristic that every element within each cluster have a distance between each other less than y_0. This is again illustrated in figure 4.9 on the facing page.

We then plot the number of clusters against the value of the "cut". This produces an L-shaped curve as shown in figure 4.10.

The curve shows that as the number of clusters increases, the distance between elements within the clusters decreases. Furthermore, the distance between cluster members drops rapidly and then the curve becomes flatter. After that, adding more clusters does not change the inter-member distance by a lot.

This is akin to the curve that is used to select the number of clusters when using the *k-means* clustering method, which plots the number of clusters against the total variance covered by the clusters. In that case, the "elbow method" (Thorndike (1953)) is used to choose the optimal number of clusters, as the point on the curve at which the curvature changes (the "elbow").

We will apply the same method to establish the number of clusters above which the distance between elements stops decreasing notably. We use the value of the distance at the cut as a minimum distance threshold, below which we will consider fields to be "close enough".

There are several algorithms to find the bend in the curve, such as checking the maximum curvature or checking the angle between points, amongst others (Satopaa et al. (2011)). We use the method described by Castellanos et al. (2002) which consists of finding the point that will generate the smallest angle (but not smaller than $\frac{7\pi}{8}$) between the last point on the curve and a third point that takes every position on the curve, as long as the points form a triangle with a clockwise orientation (see algorithm 4.1).

Procedure *find-elbow-in-curve(x, y)*
  $n \leftarrow length(x)$
  $C \leftarrow (x_n, y_n)$
  $best \leftarrow cos(\frac{7\pi}{8})$

  for $i \leftarrow 1$ **to** $n-2$ **do**
    $B \leftarrow (x_i, y_i)$
    for $j \leftarrow i+1$ **to** $n-2$ **do**
      $A \leftarrow (x_j, y_j)$

      $area \leftarrow \frac{det(\overrightarrow{BA}, \overrightarrow{AC})}{2}$

      $cosA \leftarrow \frac{\overrightarrow{CA} \cdot \overrightarrow{BA}}{\|\overrightarrow{CA}\| \|\overrightarrow{BA}\|}$

      if $area < 0$ *and* $cosA > best$ **then**
        $k \leftarrow j+1$
      end
    end
  end
  return k
end

Algorithm 4.1: *Triangle-based elbow finding algorithm as proposed by Castellanos et al. (2002).*

We then compute the cut distances using the different linkage methods described in Müllner (2011): *average*, *centroid*, *complete*, *median*, *single*, *Ward* and *weighted*. The resulting curves are plotted for each meteorological variables. See figure 4.11 on page 58 and figure 4.12 on page 59 for an example based on the *single* linkage method.

The results are listed in table 4.1. Each of the linkage methods leads to different values. As we are searching for a minimum distance threshold, we will select the values for the *single* linkage, as it gives the smallest values.

Variable	Average	Centroid	Complete	Median	Single	Ward	Weighted
10m meridional wind	157.1	132.2	197	134.5	107.7	498.1	161.2
10m wind speed	110.5	94.1	145.4	103.6	85.4	335.8	116.3
10m zonal wind	156.3	140	210.3	129.4	113.1	583.5	159.6
Geopotential at 500 hPa	22193	17516	34539	17532	12148	139245	22582
Geopotential at 850 hPa	14642	13135	21945	13273	9067	74760	16878
Mean sea level pressure	20742	15724	30223	15872	12059	81239	21106
Significant wave height	30.4	27	39.2	26.8	20.6	97.3	32.1
Snow depth	0.05	0.05	0.06	0.04	0.03	0.14	0.05
Snowfall	0.002	0.002	0.003	0.002	0.002	0.004	0.002
Surface air temperature	66	55.7	99.4	56.8	45.6	267.7	66.7
Total cloud cover	13.4	11.5	15.3	11	10.8	31.6	13.2
Total precipitations	0.02	0.02	0.02	0.02	0.02	0.03	0.02

Table 4.1: *Value of the cutting distance for the "elbow" point, for various linkage methods. The smallest values are highlighted.*

For *surface air temperature* for example, the minimum distance between two fields to be considered "close enough" would be 45.6 K. Please note that this value is computed using the Euclidean distance, e.g. the Euclidean distance, and the order of magnitude of the value depends on the number of dimensions, or grid points in the fields. We could consider normalising the value by dividing by the number of dimensions (1024) and obtain 0.044 K. We will continue by using the unnormalised values for the sake of readability of the various figures.

4.4 Wavelet-based fingerprinting

The first step is to look at how 2D Discrete Wavelet Transform (DWT) (see section 3.1.5) can be applied to meteorological fields. For that, we will consider a meteorological field as a 2D matrix (see section 2.3) and perform a DWT on it. The transform produces *approximation* and *detail* coefficients.

Figure 4.13 on page 60 illustrates the full decomposition process: 4.13b shows the original field to be transformed. After one transformation, the resulting approximation is half the size of the original field (top left of 4.13b) and three details coefficients are created: horizontal (top right of 4.13b), vertical (bottom left of 4.13b) and diagonal (bottom right of 4.13b). Further transformations are done by iteratively decomposing each resulting approximation until a single value is reached (4.13c, 4.13d, etc.).

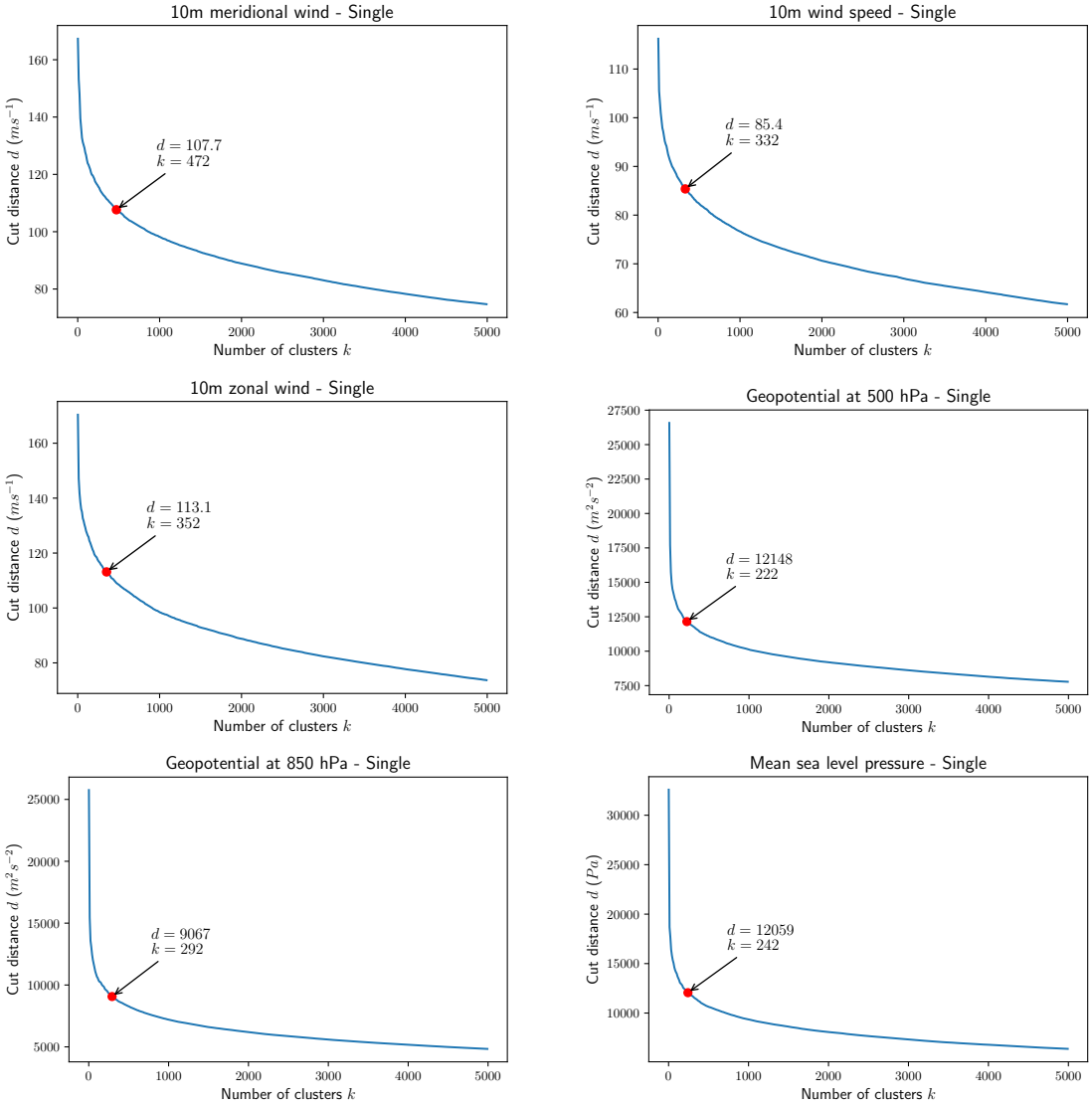


Figure 4.11: *Position of "elbow" points for each variable for* single *linkage method.*

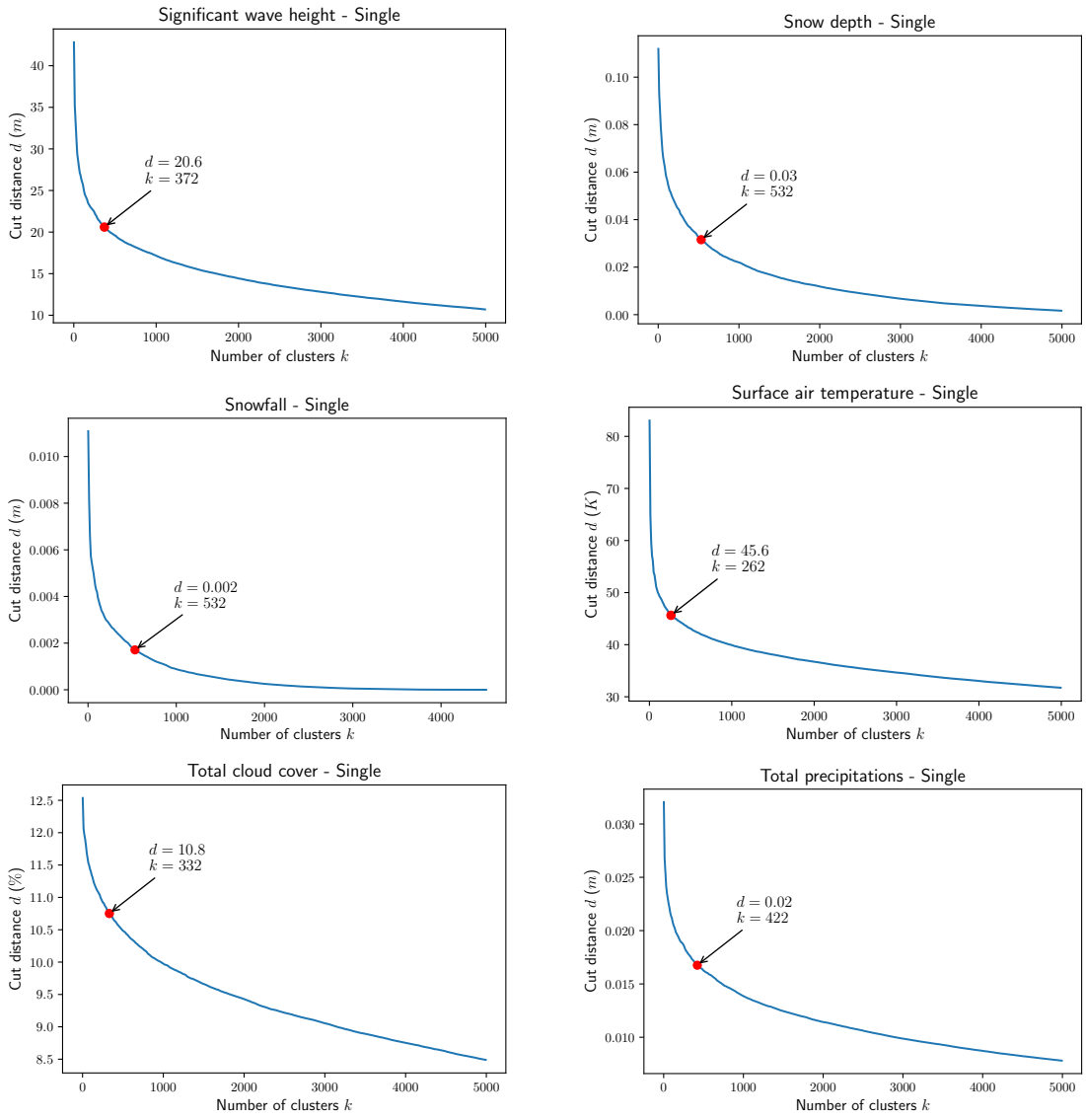
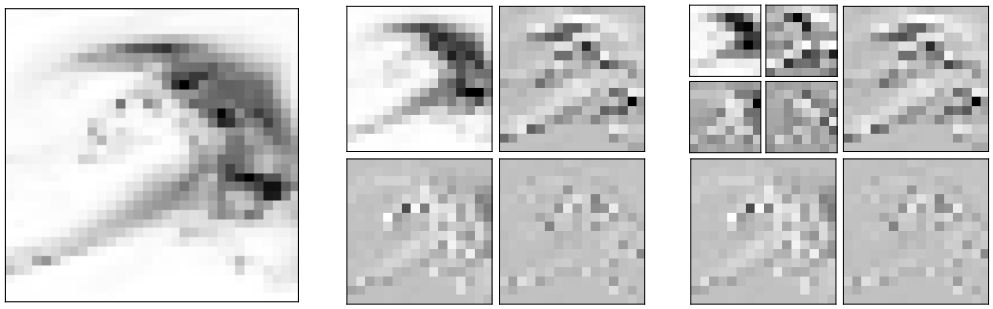
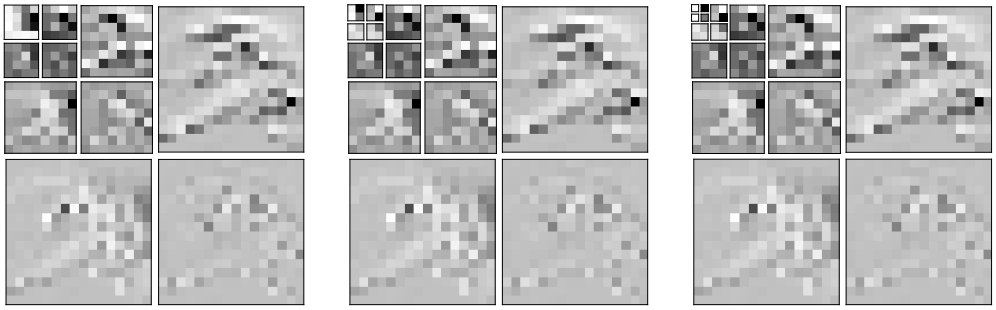


Figure 4.12: *Position of "elbow" points for each variable for* single *linkage method.*



(a) *Input field (level 0)* **(b)** *Decomposition level 1* **(c)** *Decomposition level 2*



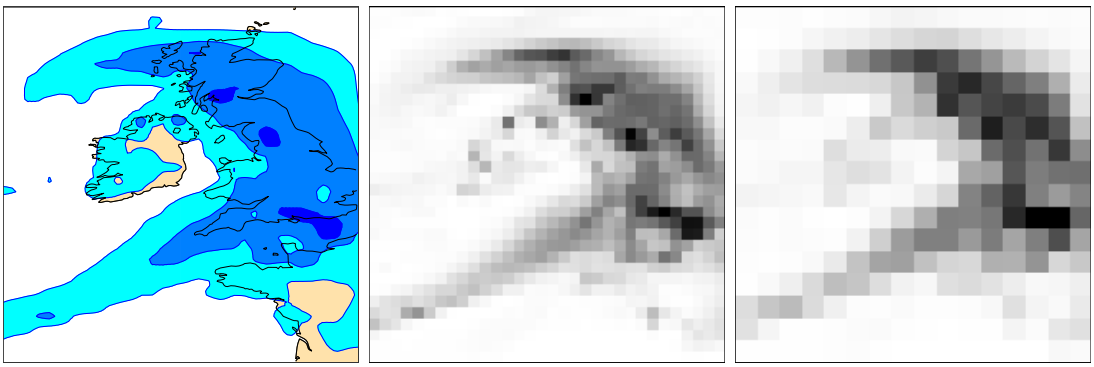
(d) *Decomposition level 3* **(e)** *Decomposition level 4* **(f)** *Decomposition level 5*

Figure 4.13: *Discrete Wavelet Transform for several levels. In each case, the top left image is the approximation coefficients, the other images are the details coefficients.*

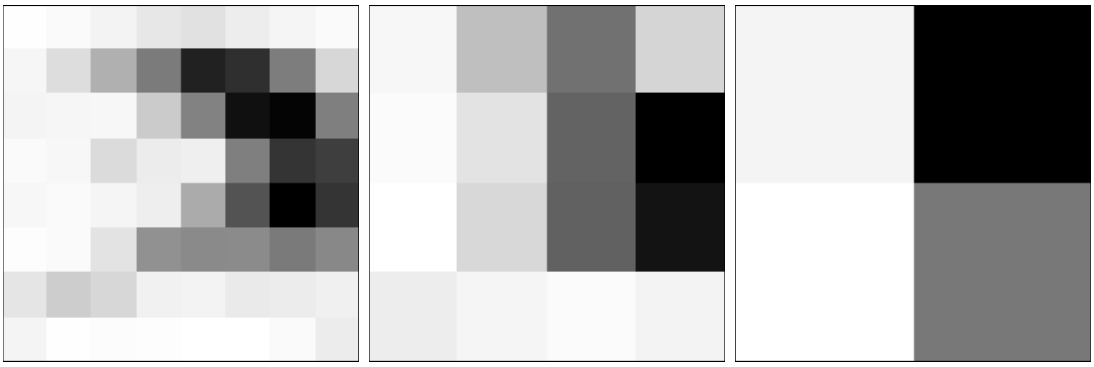
As described in chapter 3, the approximation is a smoothing of the signal, and captures large scale features, while details represent smaller variations around the approximation. The original signal can we reconstructed from all coefficients. Wavelet compression is performed by selecting the approximation coefficient of a given stage of the DWT and discarding the detail coefficients.

We will define the compression factor C as the level of the DWT. As C increases, the number of values in the compressed field is divided by 4 (figure 4.14 on the next page).

For this work, we need to understand how much compression we can apply to a field in order to retain enough information so that we can still compare two fields, and how can we encode the compressed data so that we create a fingerprint as small as possible.



(a) Total precipitations **(b)** $C = 0$, $N = 32^2 = 1024$ **(c)** $C = 1$, $N = 16^2 = 256$



(d) $C = 2$, $N = 8^2 = 64$ **(e)** $C = 3$, $N = 4^2 = 16$ **(f)** $C = 4$, $N = 2^2 = 4$

Figure 4.14: *Greyscale images showing the result of wavelet compression of a field of* total precipitations*. C is the compression factor, N is the number of data values remaining after compression.*

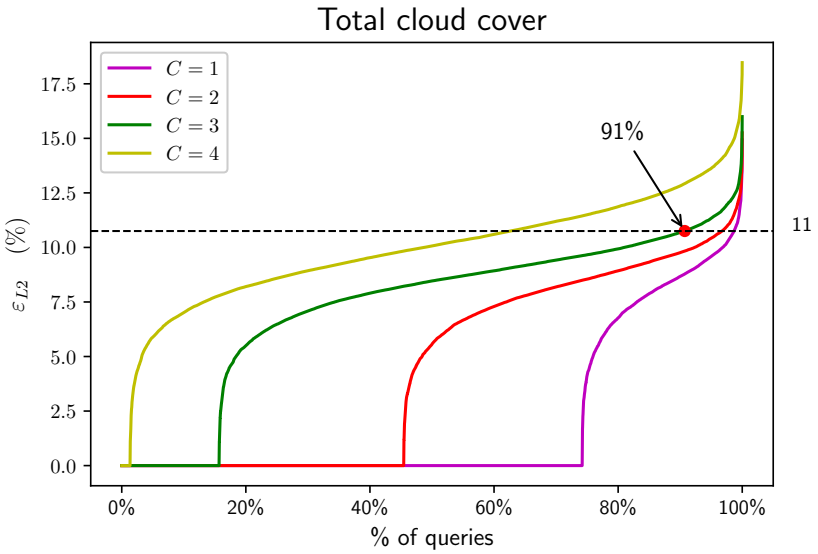


Figure 4.15: *Plot of the values of* $\varepsilon_{L2}(q)$ *for* total cloud cover*, for the compression factor* C *varying between* 1 *and* 4*.*

4.5 Choice of the compression factor C

The second step is to asses to which level the transform can be applied without too much loss of information. For that, we will consider a fingerprinting scheme that simply consists of retaining the approximation coefficient for various values of the compression factor C.

We then compute the error of each fingerprint as described in section 4.2. For each selected meteorological variable, $\varepsilon_{L2}(q)$ is computed for every field q in the working set, and for each value of C. The results are then sorted and plotted on a graph. The error is compared to the minimum distance threshold defined in section 4.3.4 and listed in table 4.1 on page 57.

Considering figure 4.15, we see that for $C = 3$, the value of ε_{L2} is 91% for the minimum distance threshold found earlier. This means that for only 9% of the queries, the field returned when querying the system is further away from the closest field according to the Euclidean distance than the minimum distance threshold. For $C = 4$ the value is just above 60% while for $C = 1$ and $C = 2$ it is close to 100%. It should also be noted that the portion of the curves that have value zero represents the proportion of queries when the nearest neighbour according to the fingerprinting distance and the nearest neighbour according to the Euclidean distance are the same.

Figure 4.16 on the facing page and figure 4.17 on page 64 show the same curves for all the selected meteorological variables. We can clearly see that the resulting

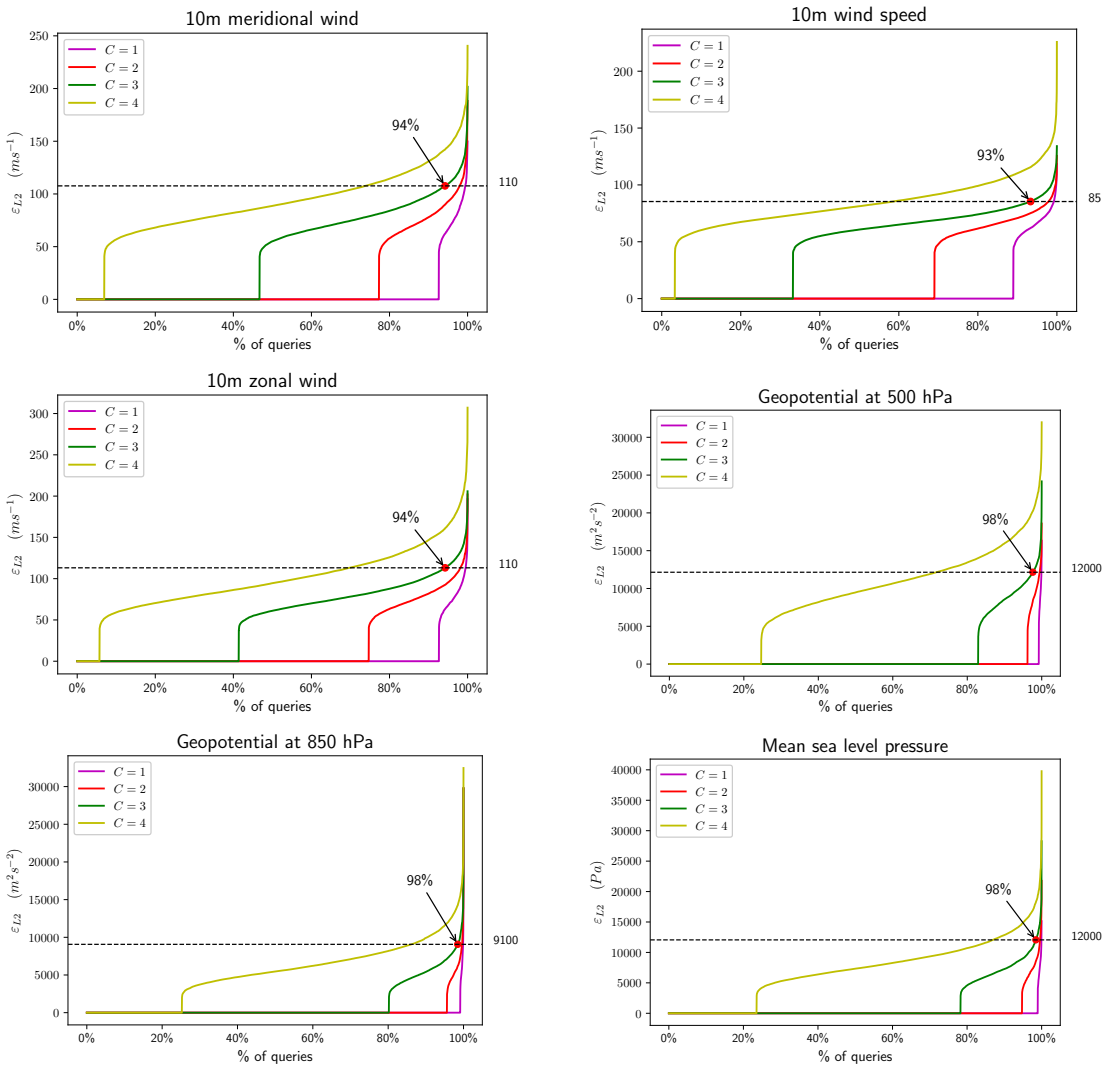


Figure 4.16: *Plot of the values of $\varepsilon_{L2}(q)$ for a subset of the selected meteorological variables, for various values of the compression factor C.*

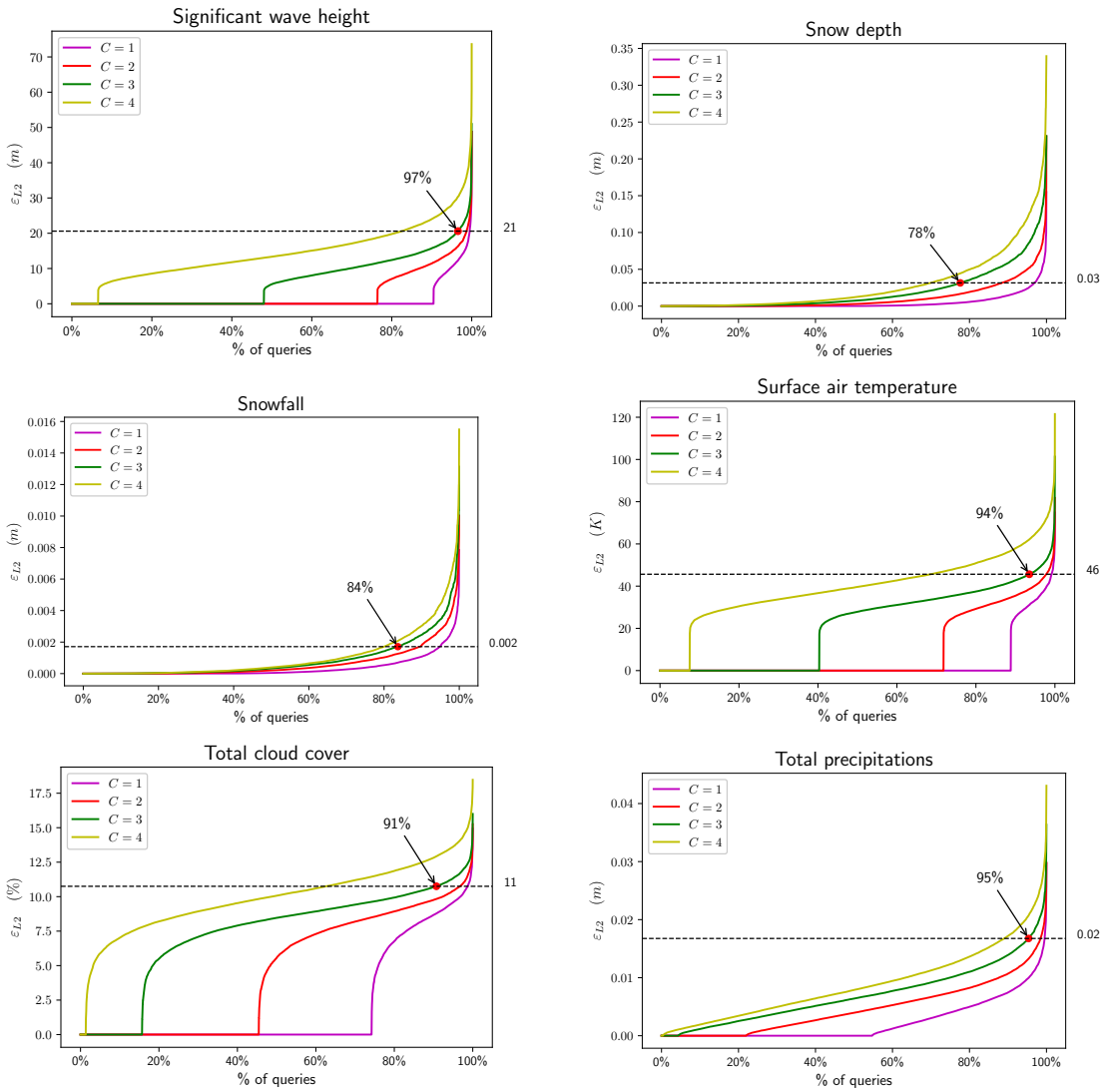


Figure 4.17: *Plot of the values of $\varepsilon_{L2}(q)$ for a subset of the selected meteorological variables, for various values of the compression factor C.*

graphs are similar, with the exception of *snow depth*, *snowfall* and, to a lesser extent, *total precipitations*, as it was the case for the distribution of distances in figure 4.2 on page 46 and figure 4.3 on page 47.

It is clear from the plots that various levels of compression affect the meteorological variables differently:

- fields that represent extensive properties, such as *total precipitations*, *snow depth* or *snowfall*, and that often noisy, lose a lot of their details with compression, and have higher values of ε_{L2};
- other fields that represent intensive properties, such as *10m wind speed*, *significant wave height* and *surface air temperature* are less compressible, but still show very low values of ε_{L2};
- fields that are "wave-like", i.e. pressure fields (*geopotential at 850 hPa*, *geopotential at 500 hPa* and *mean sea level pressure*), have a multi-resolution nature and are very smooth, are compressible without too much loss of information, and have very low values of ε_{L2} even for $C = 4$.

We will select $C = 3$ as the compression factor to use for the remainder of this work, as this provides enough information reduction so that generated fingerprints are small, while having a high effectiveness so that matching of fingerprints will provide good results.

4.6 Definition of a fingerprinting scheme

The third step is to define the fingerprinting scheme. We will do this by encoding the compressed field, which will reduce the size of the fingerprint even further.

The method proposed is to define the fingerprint F of a meteorological field f as:

$$F(f) = \langle s, r \rangle \tag{4.10}$$

where:

- s is a bit vector, representing the shape of f, and
- r is a reference value, capturing the intensity of the field f.

The fingerprinting method proposed is as follows:

1. the meteorological field is considered as a 2D grayscale image;

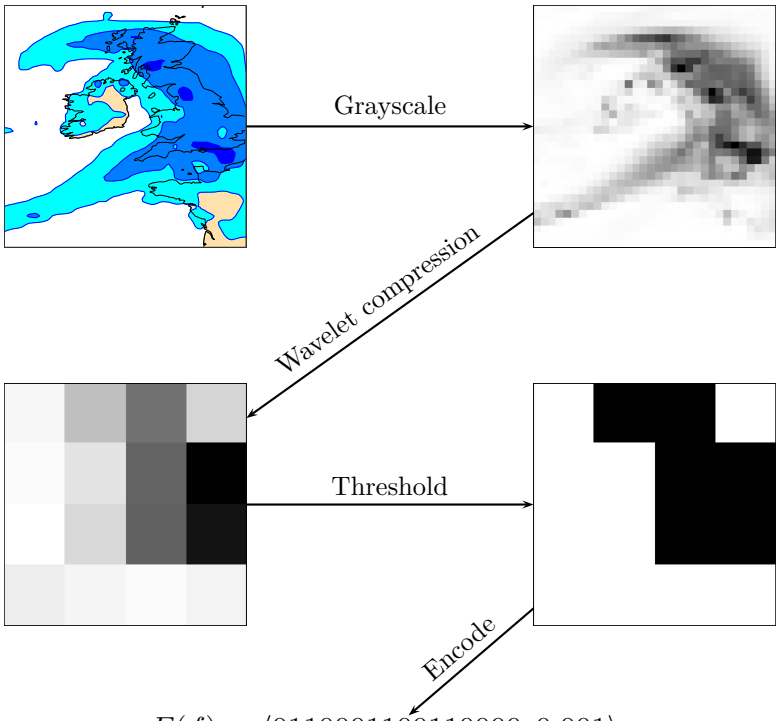


Figure 4.18: *Algorithm: field fingerprints are computed using wavelet compression and thresholding. In this example,* 0.001 *is the average value of the field.*

2. a reference value is selected (for example the mean, or the median of the field);
3. the field is compressed using wavelet compression;
4. the reference value is used as a threshold to convert the compressed image into a bitmap;
5. the bits that make the bitmap are extracted and form the shape part of the fingerprint.

Step 1 is described to stress that the algorithm expects the actual values of the field as input, and not a graphical representation (fields are not images, see section 2.3). In the case of this research, fields are already available in a binary form, so the first step is not necessary. In this example, the fingerprint is a tuple consisting of a 16 bits vector and a floating-point value. In a modern computer, this would use 48 bits of memory if the reference is in single precision. The method is illustrated in figure 4.18.

Chapter 5

Experimental validation

In the previous chapter, we proposed a scheme to extract small fingerprints from meteorological fields, so that they can be used as proxies when searching for analogues.

In order to validate that scheme, we will first define a distance, or metric, between fingerprints and find its optimal parametrisation.

We will then look at the distribution of distances between fingerprints and how they relate to the distribution of distances between fields. We will check if using distances between fingerprints to perform hierarchal clustering leads to meaningful results: showing that clustering fields and clustering fingerprints give comparable results would validate the effectiveness of the fingerprinting scheme.

Finally, we will have a look at cases for which the nearest neighbour according to fingerprints distances are very different from the nearest neighbour according to the Euclidean distance.

5.1 Choice of the distance between fingerprints

Querying the system will be done by comparing the fingerprint of a query field to the fingerprints of the fields in the archive. This entails solving the nearest-neighbour problem on a set of fingerprints, thus the need to have a distance between them.

In section 4.6, we define the fingerprint of f as $F(f) = \langle s, r \rangle$ where:

- s is a bit vector representing the shape of f, and

- r is a reference value, capturing the intensity of the field f.

We use the mean of the field for r, normalised to the interval $[0, 1]$:

$$r = \frac{\overline{F} - F_{min}}{F_{max} - F_{min}} \tag{5.1}$$

where F_{min} and F_{max} are respectively the minimum and maximum values for a field of that kind. F_{min} and F_{max} are computed once from the archive.

We then define the distance between the fingerprints $\langle s_1, r_1 \rangle$ and $\langle s_2, r_2 \rangle$ as a function of a distance D_s between s_1 and s_2, and a distance D_r between r_1 and r_2.

The Hamming distance (Hamming (1986)) between two bit vectors of the same length is the number of corresponding bits that are different. It will be used to compute the distance between s_1 and s_2:

$$D_s(s_1, s_2) = \frac{hamming(s_1, s_2)}{nbits} \tag{5.2}$$

where *hamming* is the Hamming distance and *nbits* is the maximum number of bits used to encode s_1 and s_2,

The distance between r_1 and r_2 is simply defined as:

$$D_r(r_1, r_2) = |r_1 - r_2| \tag{5.3}$$

Both D_s and D_r are in the interval $[0, 1]$. In order to establish a distance between fingerprints, they need to be combined into a single value. For this, we will study several possible combinations; these are listed in table 5.1 on the next page. Although the most obvious choice is to use method 0, i.e. a simple linear combination between D_s and D_r, we conduct an empirical experiment during which we compare all methods listed in the table. There is no mathematical motivation to the list of combinations; this is simply an exploratory work to assess if they would lead to significantly different results, that will require further investigations.

It should be noted that D_s and D_r are metrics in the mathematical sense of the terms. The distances listed in the table are linear combinations with positive coefficients, of monotonically increasing concave functions whose value at zero is 0, applied to either D_s and D_r. This ensures that the distances in the table are also metrics.

To select which of the methods listed in table 5.1, we then compute, for each of the selected meteorological variable v, the value of α that minimises:

$$\xi_{L2} = \sum_{q \in \mathcal{A}_v} \varepsilon_{L2}(q) \tag{5.4}$$

Method	Distance
0	$(1-\alpha)\ D_s + \alpha\ D_r$
1	$(1-\alpha)^2\ D_s + \alpha^2\ D_r$
2	$(1-\alpha)\ \sqrt{D_s} + \alpha\ \sqrt{D_r}$
3	$(1-\alpha)\ D_s + \alpha\ \sqrt{D_r}$
4	$(1-\alpha)\ \sqrt{D_s} + \alpha\ D_r$
5	$(1-\alpha)\ log(1+D_s) + \alpha\ log(1+D_r)$
6	$(1-\alpha)\ D_s + \alpha\ log(1+D_r)$
7	$(1-\alpha)\ log(1+D_s) + \alpha\ D_r$
8	$(1-\alpha)\ \sqrt{D_s} + \alpha\ log(1+D_r)$
9	$(1-\alpha)\ log(1+D_s) + \alpha\ \sqrt{D_r}$

Table 5.1: *List of methods used to combine D_s and D_r.*

Figures 5.1 on page 71 and 5.2 on page 72 show the curves of ξ_{L2} against values of α for a selected number of methods and meteorological variables. Not all combinations of methods and variables are shown for the benefit of space. The complete set of results can be seen in table 5.3 on the following page.

All curves exhibit a U-shape. On the left ($\alpha = 0$), the weight is given to D_s, i.e. the shape component of the fingerprint, while on the right side ($\alpha = 1$), the weight is given to D_r, i.e. the reference value component of the fingerprint.

Nevertheless, there is a caveat to that statement: because we use the minimum and maximum values of the fields and not the minimum and maximum values of the averages, in practice D_r does not reach the value 1, nor does D_s for some variables. Table 5.2 on the next page shows the maximum values for D_s and D_r. This means that the balance between shape and reference value is not 0.5.

For example, for *total precipitations*, the maximum value that D_r takes in the working set is 0.05, while the maximum for D_s is 1. For method 0, the middle point is the point for which $(1-\alpha) \times D_s = \alpha \times D_r$. Substituting D_s with 1 and D_r with 0.05 and simplifying, the middle point should be at $\alpha = 1/(1+0.05) = 0.95$. The table shows that for method 0, $alpha = 0.9$, which means that the weight of both components is actually balanced.

An interesting finding is that although the value of α is different for each method, the minimum of the curve is mostly the same (see table 5.3 on the following page). Not only the various methods have the same minimum value, but figures 5.3 and 5.4 show that the distribution of ε_{L2} is the same for every method.

This is certainly due to the fact that the ε_{L2} is defined as a distance between fields of the working set and therefore bear no relation to the fingerprints themselves.

Variable	D_s	D_r	$\sqrt{1+D_s}$	$\sqrt{1+D_r}$	$log(1+D_s)$	$log(1+D_r)$
10m meridional wind	1.0	0.48	1.0	0.69	0.69	0.39
10m wind speed	1.0	0.39	1.0	0.63	0.69	0.33
10m zonal wind	1.0	0.47	1.0	0.69	0.69	0.39
Geopotential at 500 hPa	1.0	0.69	1.0	0.83	0.69	0.53
Geopotential at 850 hPa	1.0	0.63	1.0	0.79	0.69	0.49
Mean sea level pressure	1.0	0.64	1.0	0.8	0.69	0.5
Significant wave height	0.75	0.32	0.87	0.57	0.56	0.28
Snow depth	0.56	0.02	0.75	0.13	0.45	0.02
Snowfall	0.88	0.04	0.94	0.2	0.63	0.04
Surface air temperature	1.0	0.32	1.0	0.56	0.69	0.27
Total cloud cover	1.0	0.92	1.0	0.96	0.69	0.65
Total precipitations	1.0	0.05	1.0	0.22	0.69	0.05

Table 5.2: *Maximum values of D_s and D_r for each of the meteorological variables, over the working set. This illustrates that although both values are in the interval $[0, 1]$, then do not always reach the value 1.*

Variable	0	1	2	3	4	5	6	7	8	9
10m meridional wind (ms^{-1})	96.0	96.0	96.9	96.3	96.2	96.0	96.0	96.0	96.2	96.3
10m wind speed (ms^{-1})	77.6	77.6	78.0	77.8	77.7	77.6	77.6	77.6	77.7	77.8
10m zonal wind (ms^{-1})	102.8	102.8	103.4	103.1	103.0	102.8	102.8	102.8	103.0	103.0
Geopotential at 500 hPa (m^2s^{-2})	15181	15181	15338	15269	15224	15185	15183	15183	15225	15269
Geopotential at 850 hPa (m^2s^{-2})	9423	9429	9520	9464	9450	9425	9425	9430	9451	9469
Mean sea level pressure (Pa)	11648	11649	11780	11724	11681	11646	11648	11646	11680	11721
Significant wave height (m)	18.2	18.2	18.2	18.2	18.2	18.2	18.2	18.2	18.2	18.2
Snow depth (m)	0.0283	0.0283	0.0283	0.0282	0.0283	0.0283	0.0283	0.0283	0.0283	0.0283
Snowfall (m)	0.0009	0.0009	0.0009	0.0009	0.0009	0.0009	0.0009	0.0009	0.0009	0.0009
Surface air temperature (K)	48.8	48.8	48.9	48.9	48.9	48.8	48.8	48.8	48.9	48.9
Total cloud cover (%)	9.2	9.2	9.2	9.2	9.2	9.2	9.2	9.2	9.2	9.2
Total precipitations (m)	0.0085	0.0085	0.0085	0.0085	0.0086	0.0085	0.0085	0.0085	0.0086	0.0085

Table 5.3: *Values of ξ_{L2} per method and per meteorological variable. It should be noted that this value is mostly the same for all methods.*

As a consequence, the sum ξ_{L2} is also not related to the fingerprints, but only to the fields in the working set, which is the same for all methods considered.

Furthermore, the vertical red dotted line in figures 5.3 on page 73 and 5.4 on page 74 represents the minimum distance threshold as defined in section 4.3.4. This shows that the peak of the distributions, i.e. the maximum error, is smaller than the minimum distance threshold in all the cases.

No real conclusion can be drawn from that empirical study. We will therefore focus on method 0, as using other methods of combining D_s and D_r makes little to no difference.

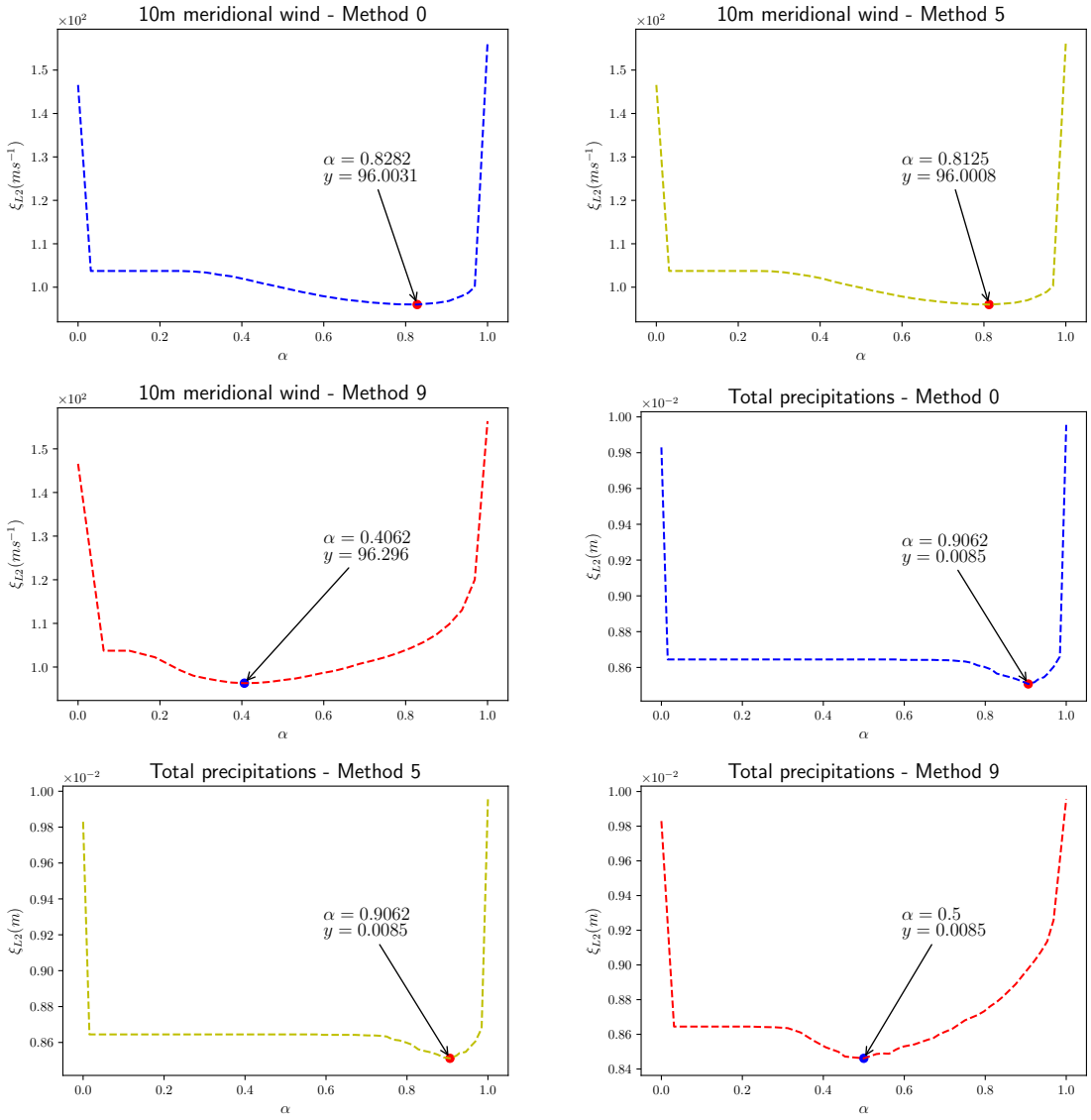


Figure 5.1: *Plots of the value for α that minimise the methods considered in table 5.1 on page 69. Only a selection of variables and methods are shown for conciseness. A complete set of results is given in table 5.3 on the facing page. It should be noted that all curves for the same meteorological variable have approximately the same minimum value.*

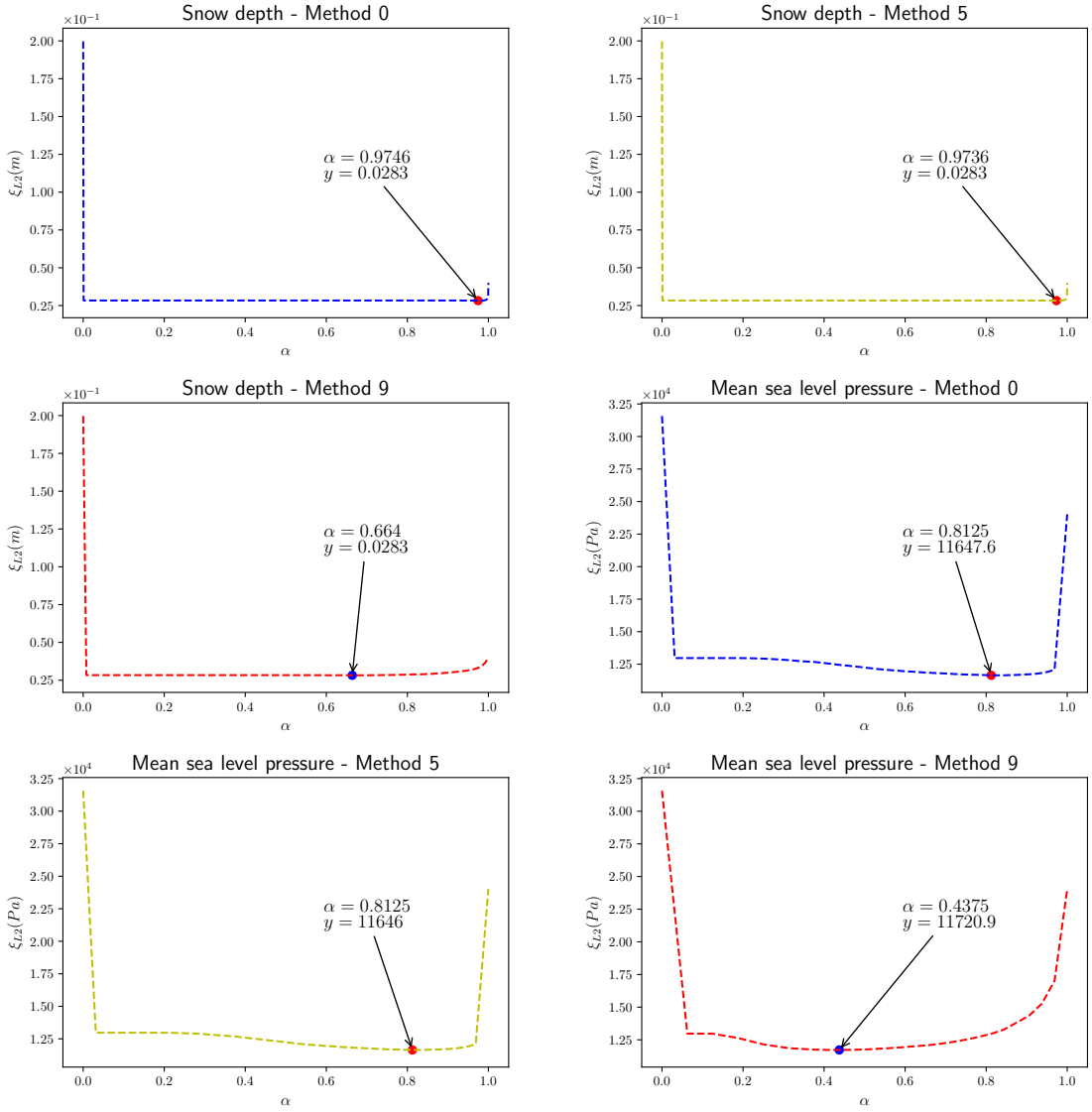


Figure 5.2: *Plots of the value for α that minimise the methods considered in table 5.1 on page 69. Only a selection of variables and methods are shown for conciseness. A complete set of results is given in table 5.3 on page 70. It should be noted that all curves for the same meteorological variable have approximately the same minimum value.*

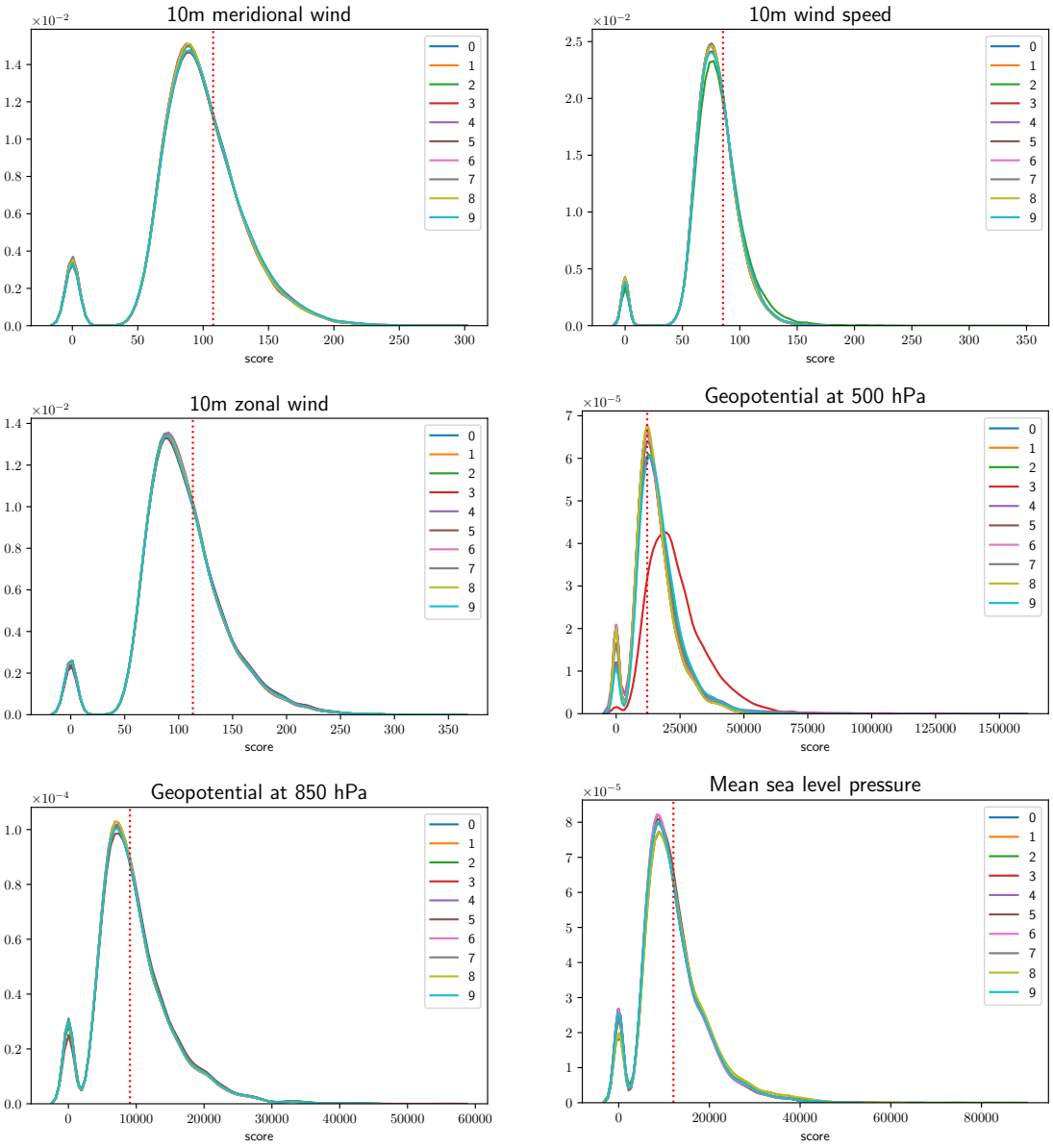


Figure 5.3: *Distribution of ε_{L2} for each of the ten methods. The vertical dotted line marks the minimum distance threshold as defined in section 4.3.4.*

.

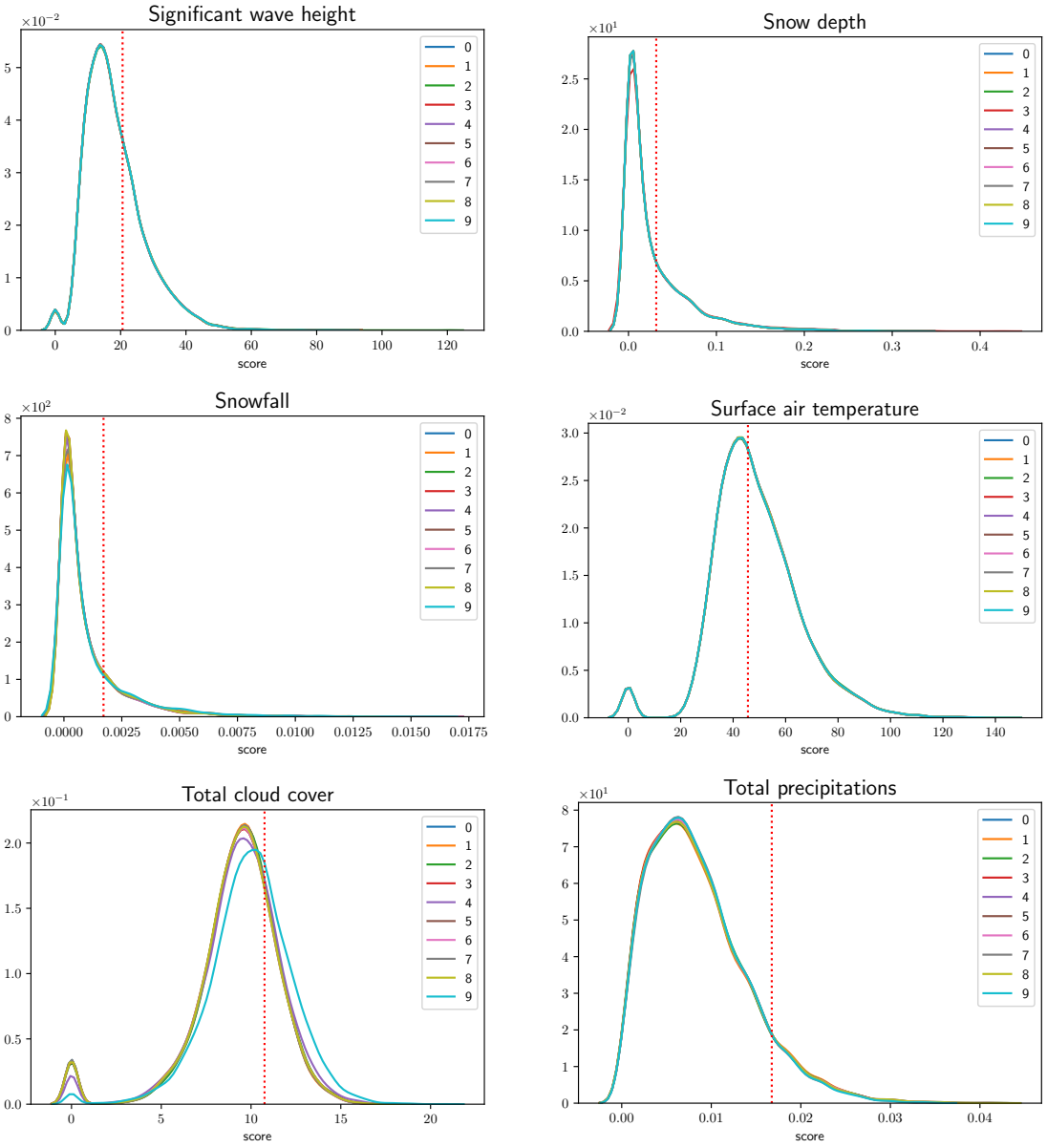


Figure 5.4: *Distribution of ε_{L2} for each of the ten methods. The vertical dotted line marks the minimum distance threshold as defined in section 4.3.4.*

.

5.2 Fingerprint-based distances

We now take a closer look at the fingerprint-based distance defined in this chapter, to check if they present the qualities of distances. For that, we compare the distribution of fingerprint distance with those of inter-field distances. We will also use the fingerprint distances to perform some clustering.

5.2.1 Distribution of fingerprint distances

Figures 5.5 and 5.6 show the distribution of fingerprint distances. Comparing it to the original distribution of distances based on the Euclidean distance (figures 4.2 on page 46 and 4.3 on page 47), we can see that with the exception of *snow depth* and *snowfall*, the shape of the bell-like curves are preserved.

Figures 5.7 and 5.8 show the distributions of distances for fingerprints that would only consist of a *shape* part (i.e. method 0 with $\alpha = 0$). The plots show clearly that the fingerprint can only take 16 different values, which is consistent with a choice of $C = 3$.

Figures 5.9 and 5.10 show the distributions of distances for fingerprints that would only consist of a *reference value* part (i.e. method 0 with $\alpha = 1$).

The distances in figures 5.5 and 5.6 are a combination of the two previous, and the "spikes" are the contribution of the *shape* component.

5.2.2 Clusters

We can build a distance matrix using the fingerprint distances between every pair of fields in the archive. We can then use this distance matrix as input to a hierarchical clustering algorithm, using the *complete* linkage method. We select that linkage method because it creates the most balanced clusters. We stop the clustering at nine clusters so that we can visually compare the results with the ones obtained using k-means clustering in section 4.3, such as figure 4.6 on page 51 and figure 5.12 on page 84.

Results are shown in figures 5.11 (*surface air temperature*), 5.12 (*snow depth*) and 5.13 on page 85 (*significant wave height*). A random member of each cluster is selected to be plotted in each figure. Of course, the choice of that member will affect the perception that the reader will have from the clustering, but in the three examples provided, it is clear that using fingerprint-based distances produces a

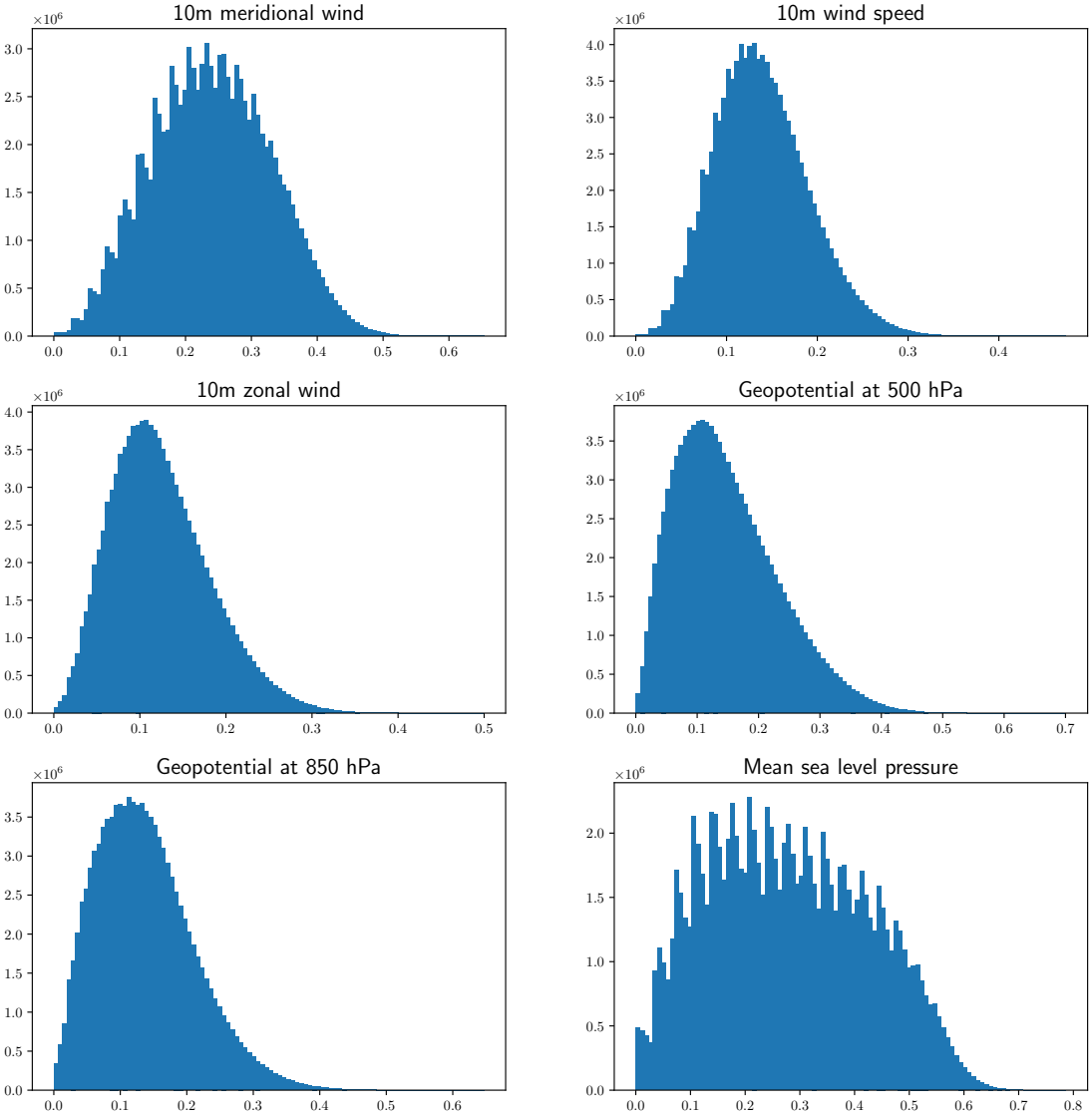


Figure 5.5: *Distribution of the fingerprint-bases distances between fields for each meteorological variable, with a number of bins set to 100. The shape of histograms are identical to the ones showing the distribution of distances between using the Euclidean distance (compare to figure 4.2 on page 46), albeit for their jagged edges.*

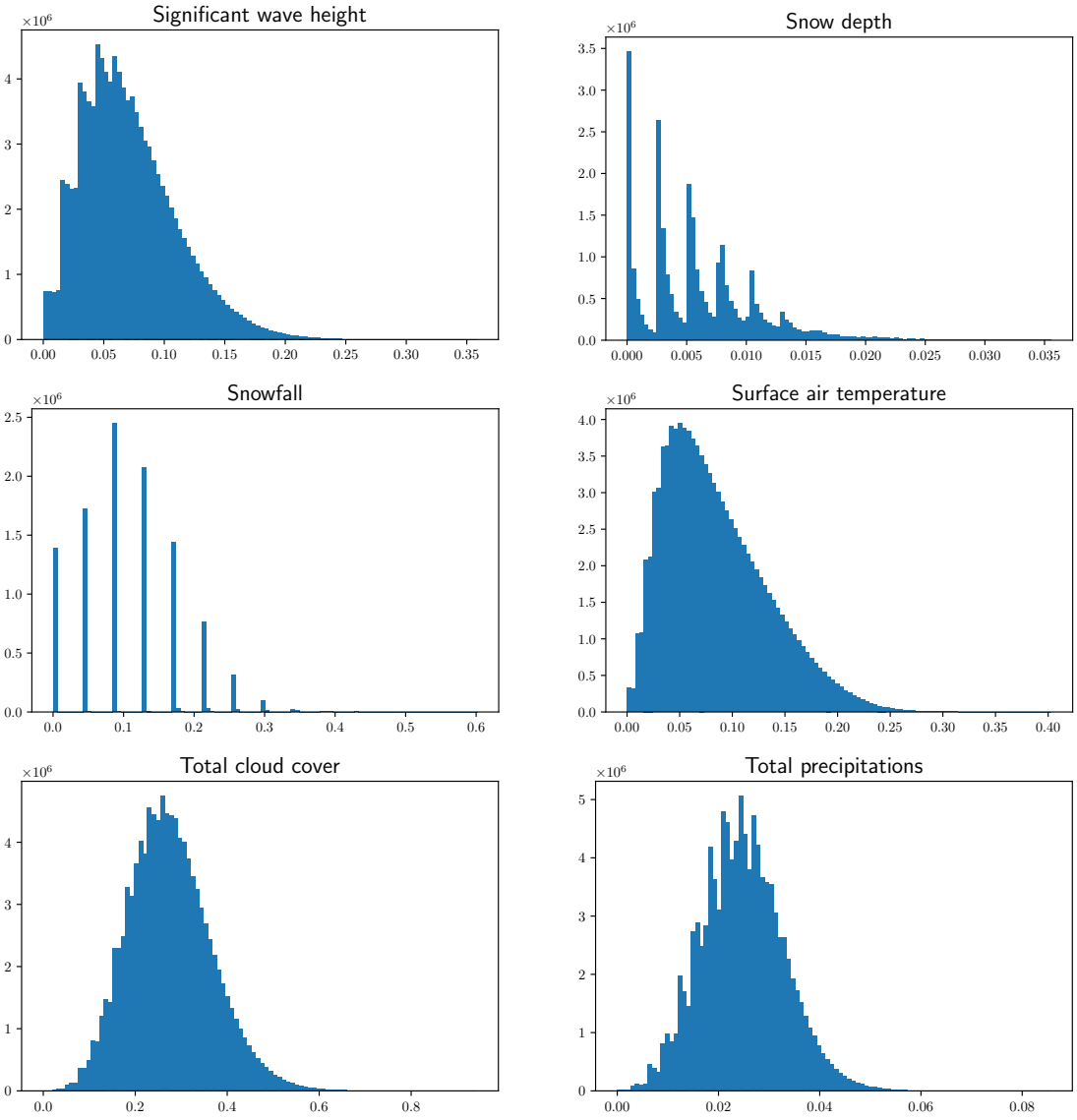


Figure 5.6: *Distribution of the fingerprint-bases distances between fields for each meteorological variable, with a number of bins set to 100. The shape of histograms are identical to the ones showing the distribution of distances between using the Euclidean distance (compare to figure 4.3 on page 47), albeit for their jagged edges.*

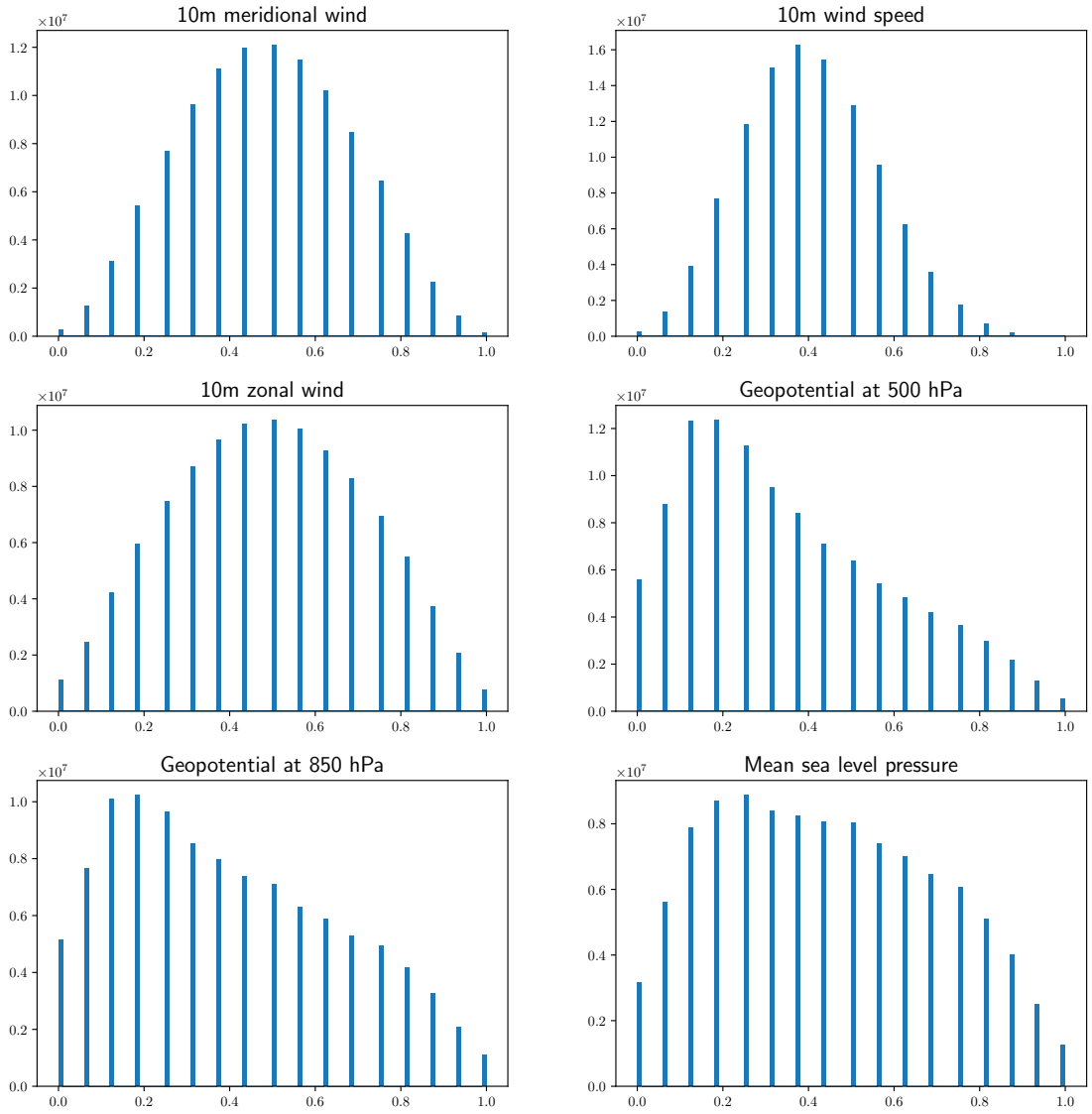


Figure 5.7: *Distribution of fingerprint-bases distances between fields using only the* shape *component of the fingerprint (number of bins set to 100).*

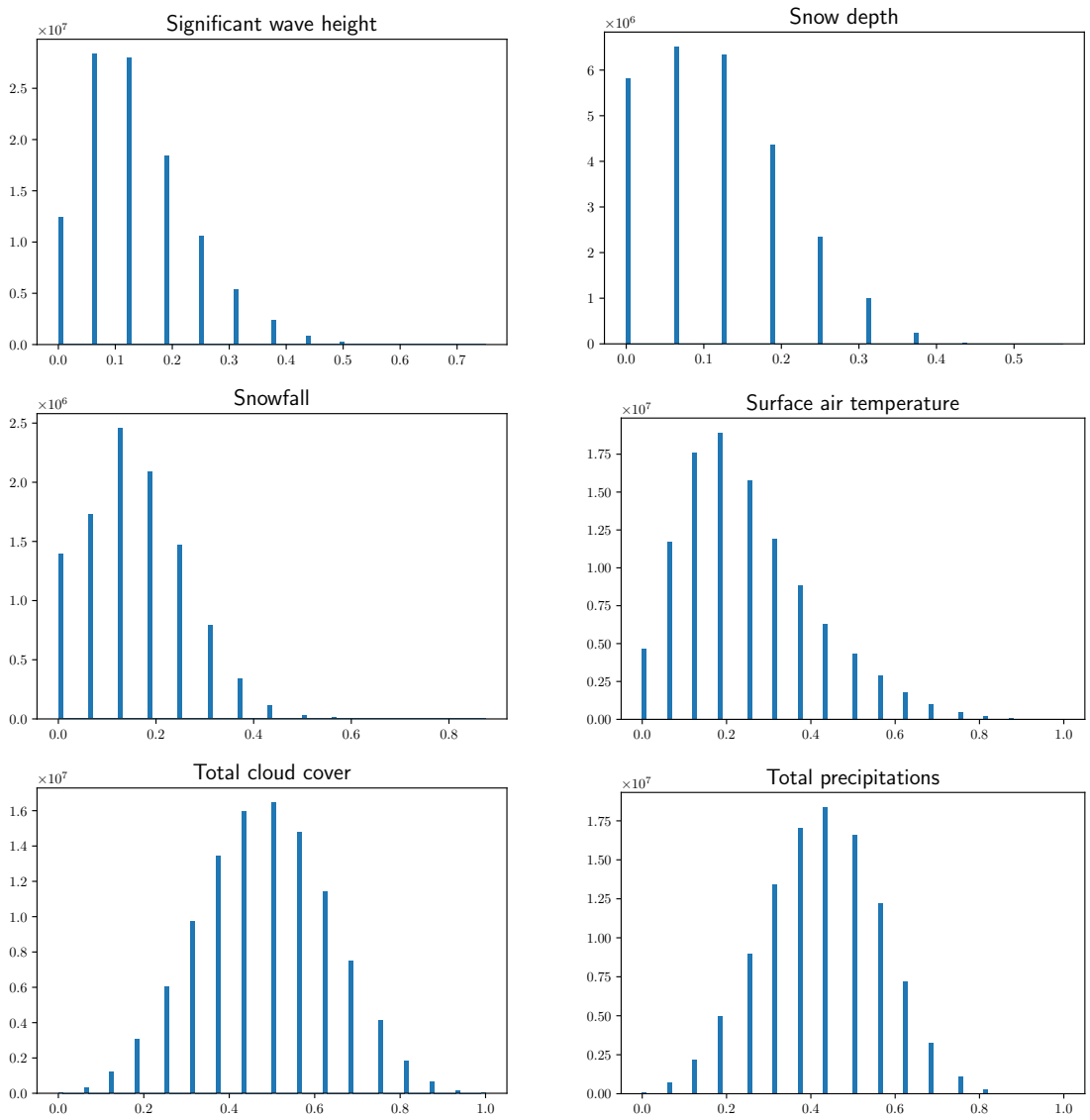


Figure 5.8: *Distribution of fingerprint-bases distances between fields using only the* shape *component of the fingerprint (number of bins set to 100).*

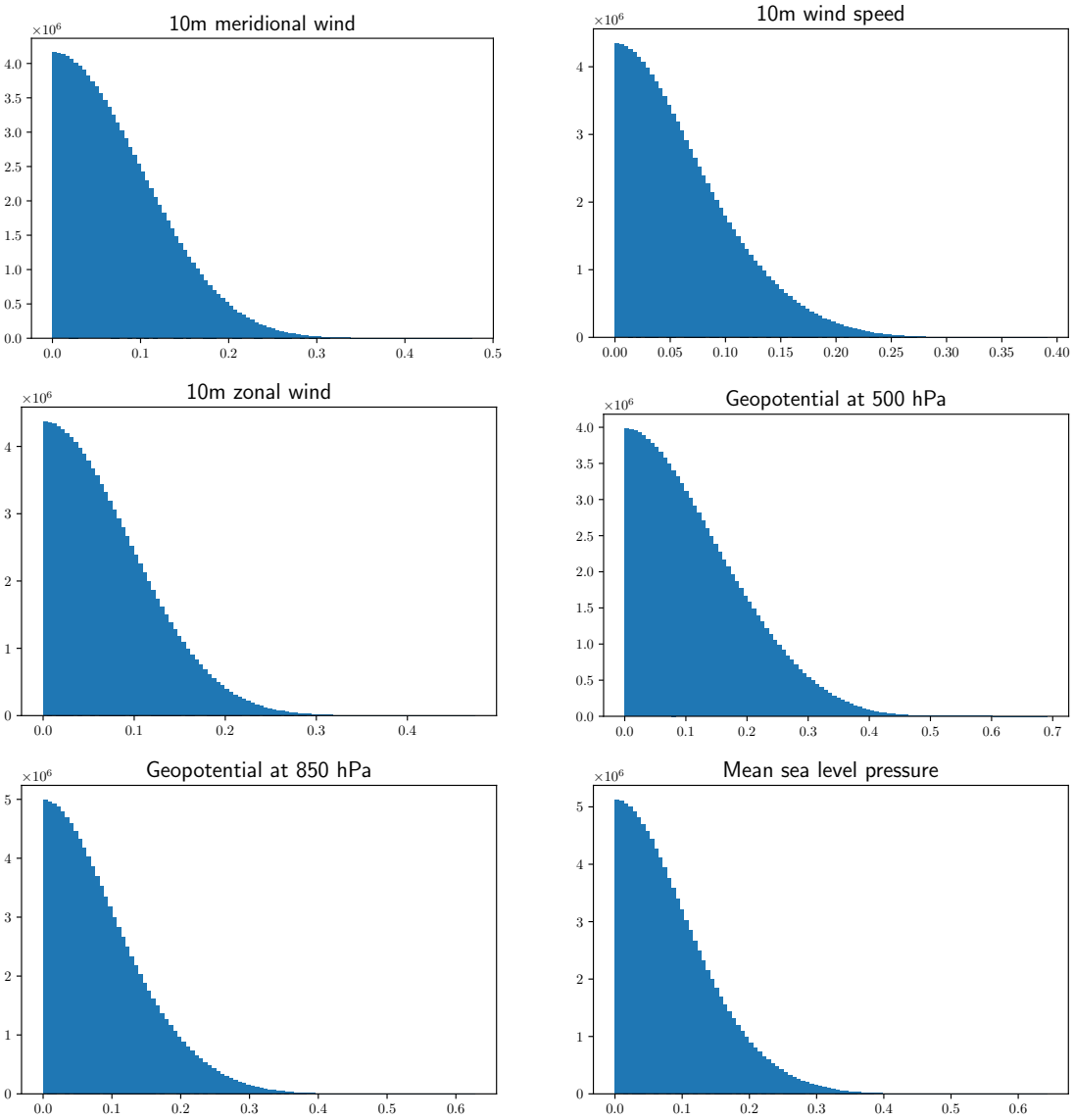


Figure 5.9: *Distribution of fingerprint-bases distances between fields using only the* reference value *component of the fingerprint (number of bins set to 100). The curves shown are half-normal distributions.*

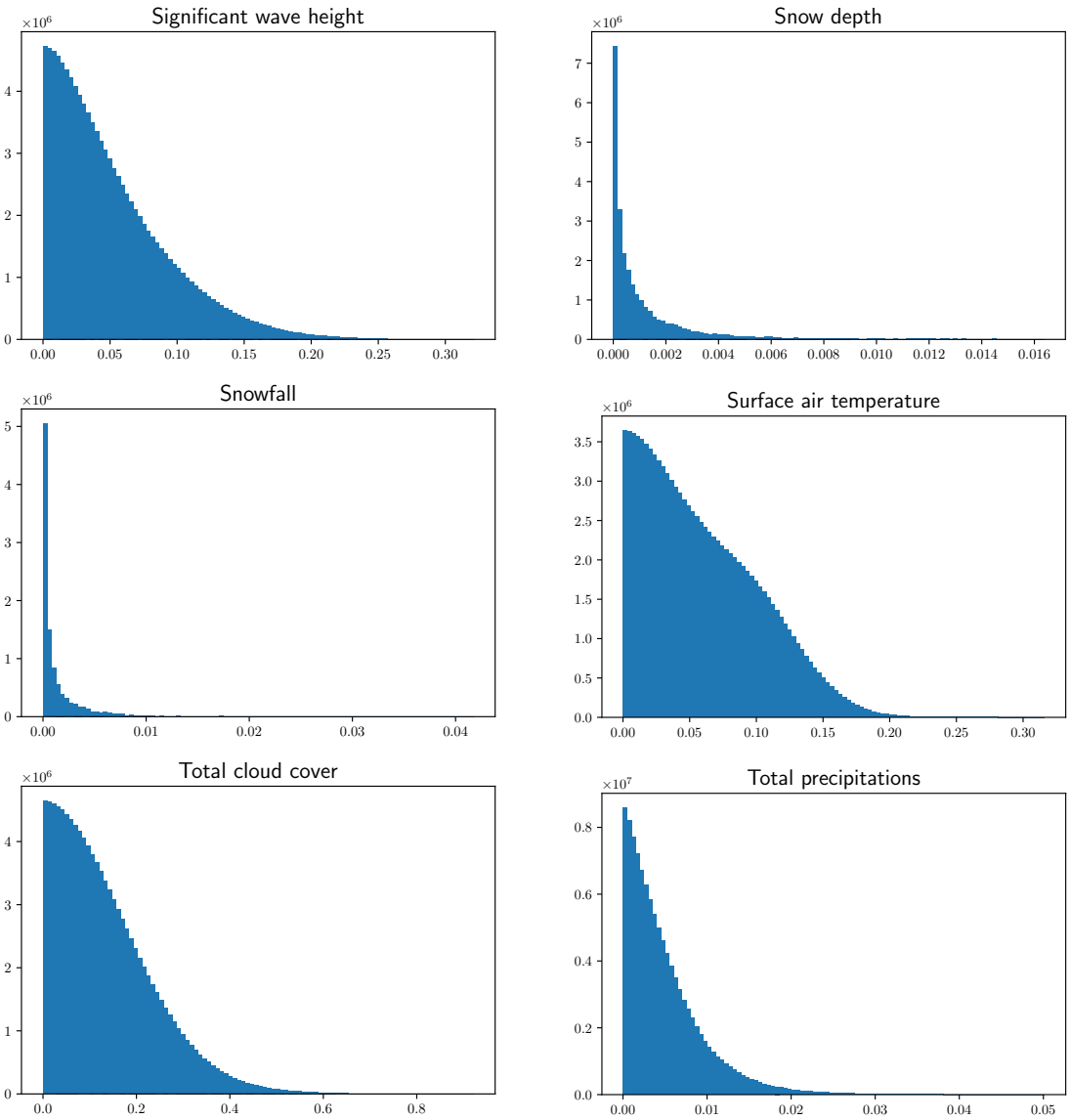


Figure 5.10: *Distribution of fingerprint-bases distances between fields using only the* reference value *component of the fingerprint (number of bins set to 100). The curves shown are half-normal distributions.*

satisfactory clustering.

The main advantage is that the hierarchical clustering is done without accessing the original data. This is possible because the algorithm only requires a distance matrix as input. In the case of an algorithm such as k-means, this needs to generate new data that are not in the starting set, i.e. the centroids, there is a need to produce fingerprints for these new data, by combining existing fingerprints. This could be considered as future work.

5.2.3 Worst cases

In the interest of completeness, we take a closer look at the worst cases, that is the queries for which ε_{L2} is maximum.

The results vary: in some cases, such as *snow depth* (figure 5.14 on page 86), *snowfall* (figure 5.15) and *10m wind speed* (figure 5.16), the worst cases can be perceived as more satisfactory than the actual closest fields according to the Euclidean distance (user perception is discussed later in section 8.1.1).

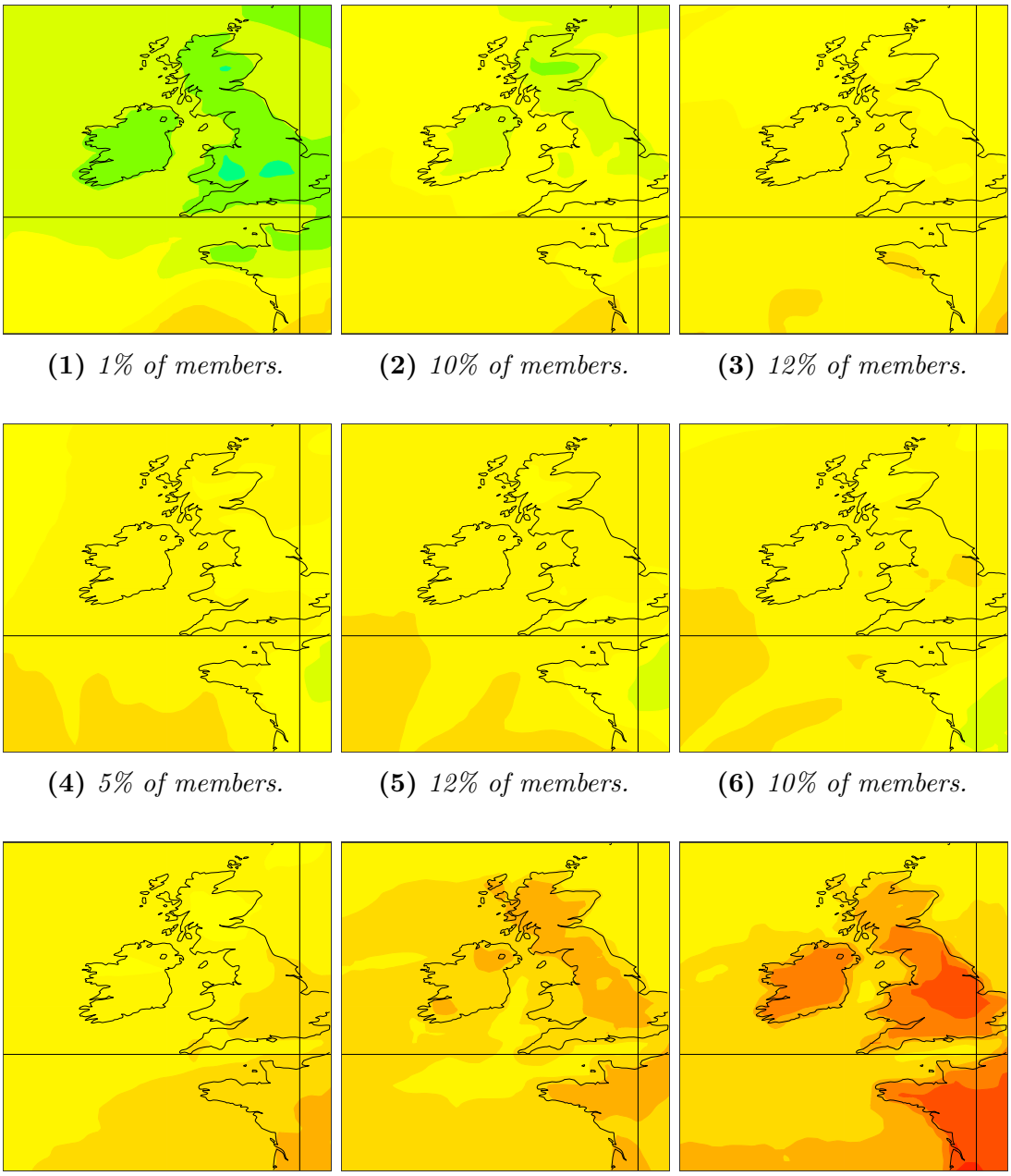
For *total precipitations* (figures 5.17 on page 87), *mean sea level pressure* (figure 5.18), *geopotential at 850 hPa* (figure 5.20) and *geopotential at 500 hPa* (figure 5.19), the worst cases can still be considered are somewhat similar to the original query, but are clearly worse than the actual closest fields according to the Euclidean distance.

For the remaining variables (figures 5.21-5.24) the worst cases are clearly different from the query field.

5.3 Memory footprint of fingerprints

The fingerprints are several orders of magnitude smaller than the original field: in our study, the fields have 1024 grid points, and therefore 1024 floating-point values. With the selected compression factor of $C = 3$, the fingerprints are composed of a 16 bits shape s and a reference value r which is a floating-point. Most of the meteorological fields are stored in the WMO GRIB format (World Meteorological Organization (2009)) using a 16 bits lossy packing method. This method can be used for the reference value r:

$$r_{16bits} = \left\lfloor 2^{16} \frac{(r - min_v)}{(max_v - min_v)} \right\rfloor \tag{5.5}$$

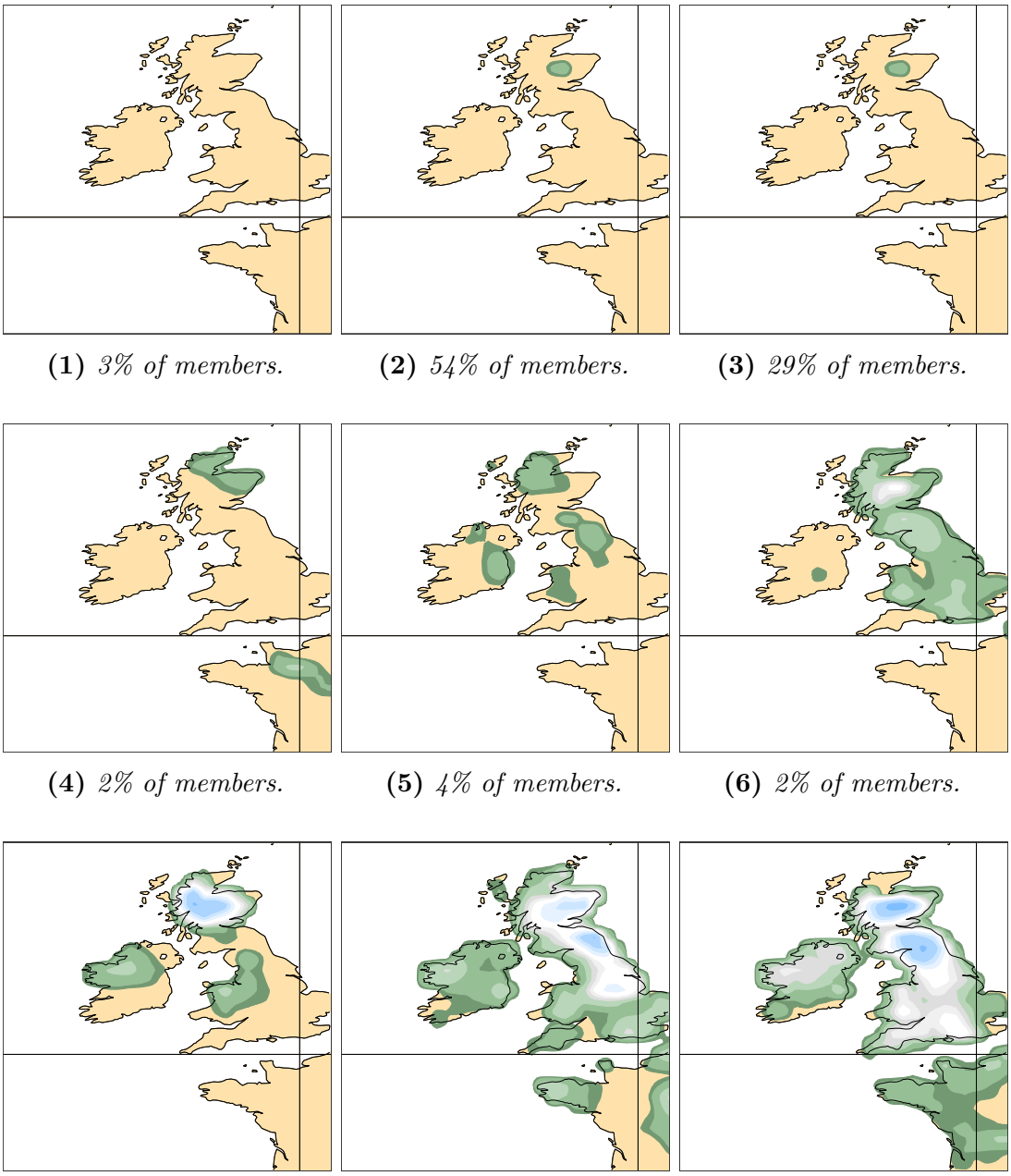


(1) *1% of members.* **(2)** *10% of members.* **(3)** *12% of members.*

(4) *5% of members.* **(5)** *12% of members.* **(6)** *10% of members.*

(7) *28% of members.* **(8)** *3% of members.* **(9)** *20% of members.*

Figure 5.11: *Hierarchical clustering of* surface air temperature *using the complete linkage method. A random member of each cluster is shown.*

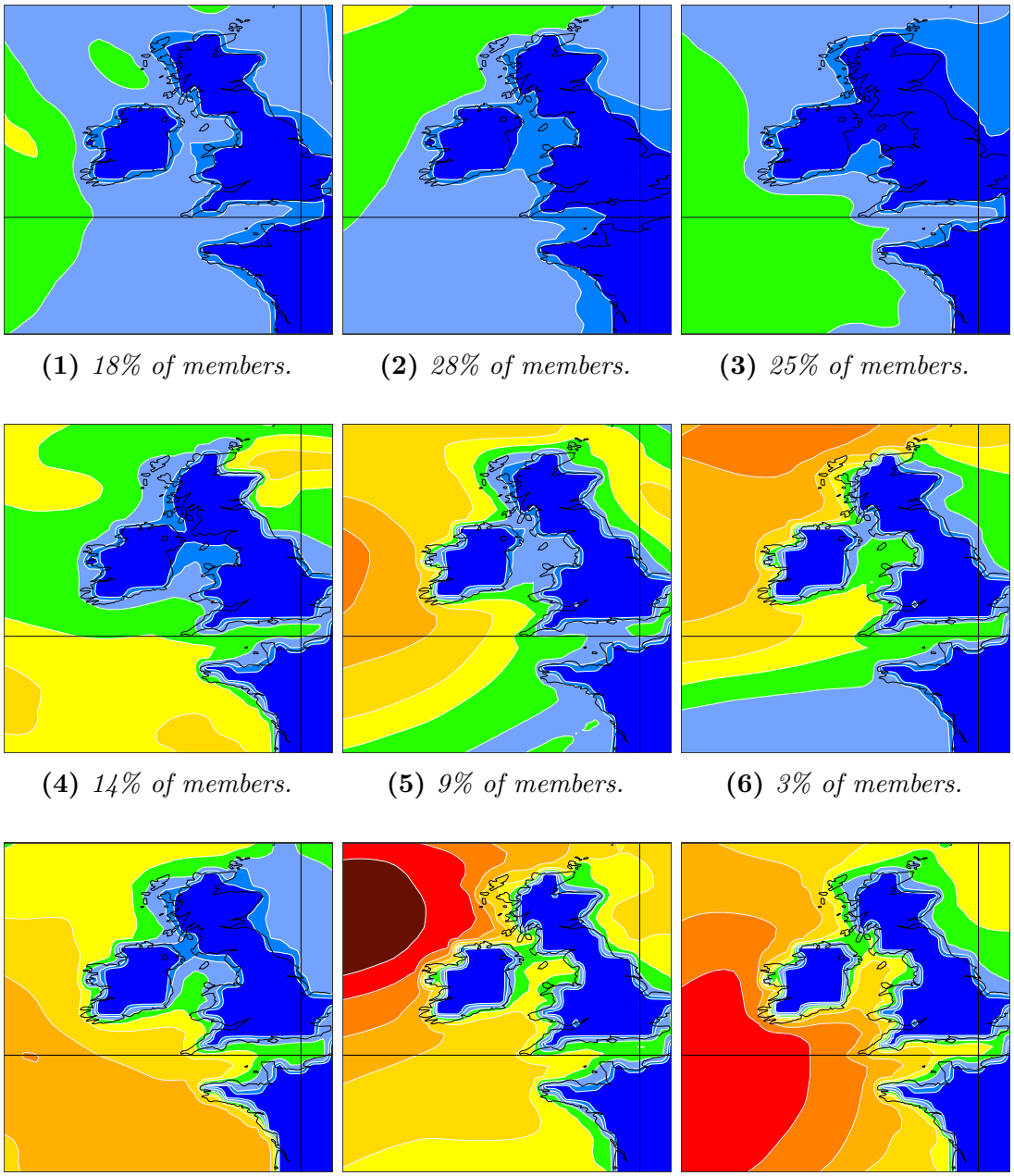


(1) *3% of members.* **(2)** *54% of members.* **(3)** *29% of members.*

(4) *2% of members.* **(5)** *4% of members.* **(6)** *2% of members.*

(7) *4% of members.* **(8)** *1% of members.* **(9)** *1% of members.*

Figure 5.12: *Hierarchical clustering of* snow depth *using the complete linkage method. A random member of each cluster is shown.*

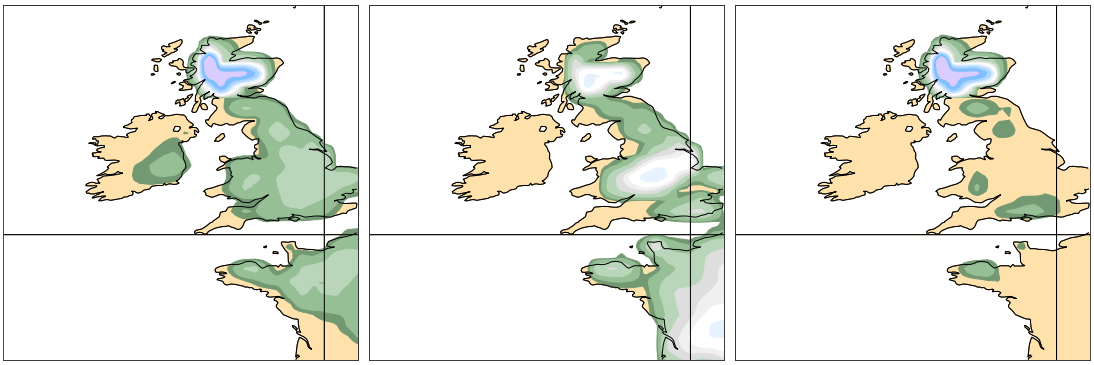


(1) *18% of members.* **(2)** *28% of members.* **(3)** *25% of members.*

(4) *14% of members.* **(5)** *9% of members.* **(6)** *3% of members.*

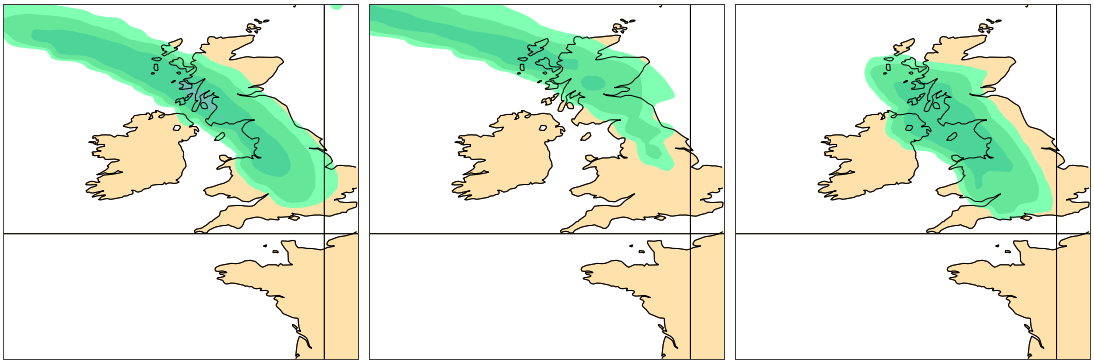
(7) *3% of members.* **(8)** *0.1% of members.* **(9)** *0.5% of members.*

Figure 5.13: *Hierarchical clustering of* significant wave height *using the complete linkage method. A random member of each cluster is shown.*



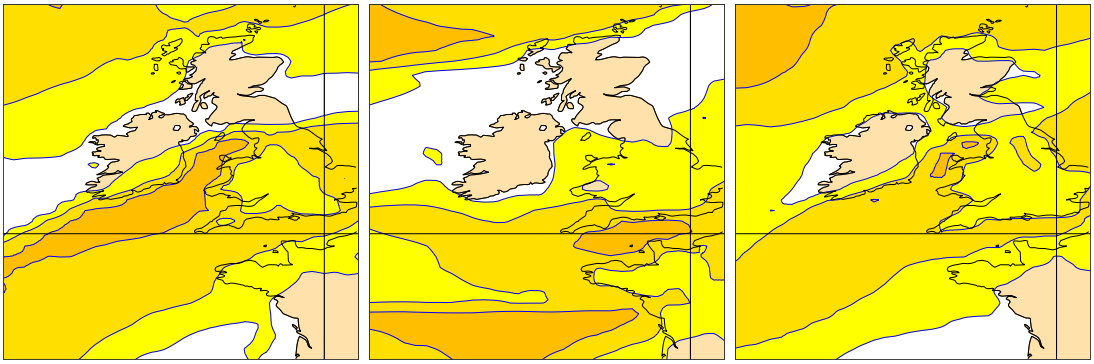
(a) *1979-03-16* **(b)** *1987-01-20* **(c)** *1979-03-15*

Figure 5.14: *Match with the highest error for* Snow depth. **(a)** *is the query field,* **(b)** *is the closest field according to the fingerprint distance,* **(c)** *is the closest field according to the Euclidean distance.*



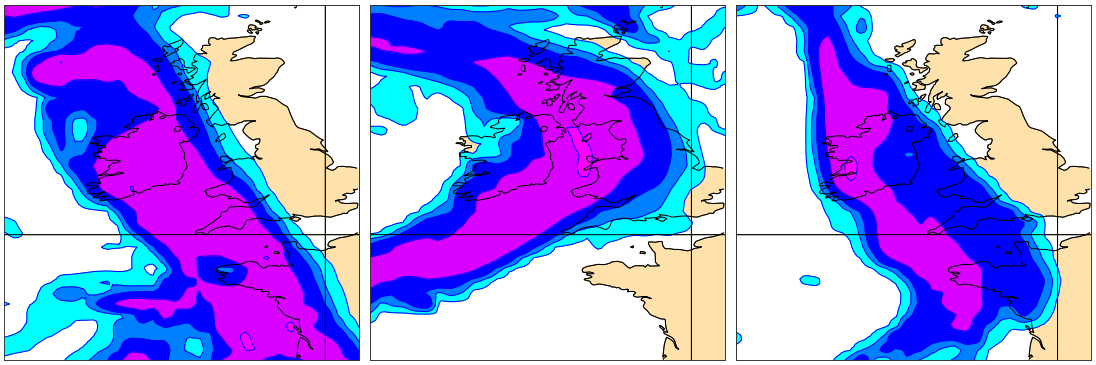
(a) *1981-01-16* **(b)** *2002-01-25* **(c)** *1987-03-07*

Figure 5.15: *Match with the highest error for* Snowfall. **(a)** *is the query field,* **(b)** *is the closest field according to the fingerprint distance,* **(c)** *is the closest field according to the Euclidean distance.*



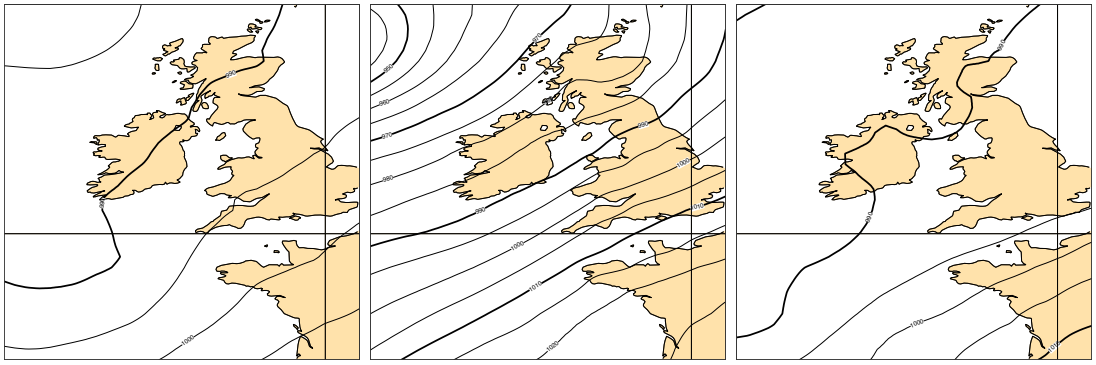
(a) *2011-02-06* **(b)** *2002-03-18* **(c)** *1980-12-24*

Figure 5.16: *Match with the highest error for* 10m wind speed. **(a)** *is the query field,* **(b)** *is the closest field according to the fingerprint distance,* **(c)** *is the closest field according to the Euclidean distance.*



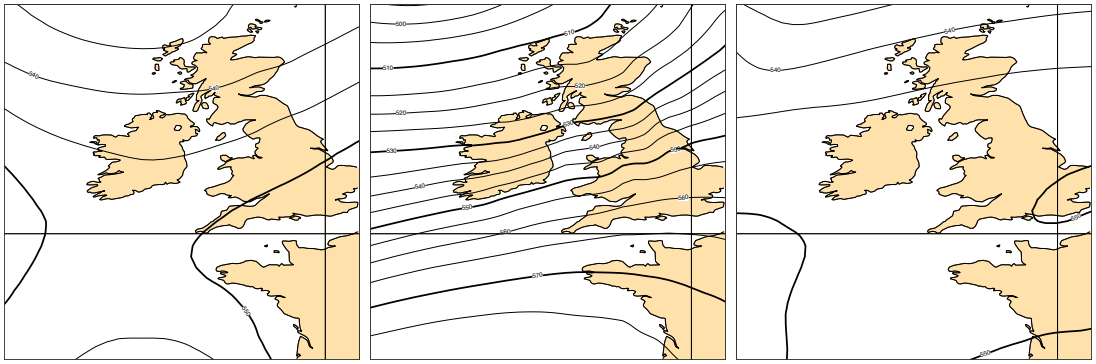
(a) *1981-12-13* **(b)** *1996-11-24* **(c)** *2003-02-28*

Figure 5.17: *Match with the highest error for* Total precipitations. **(a)** *is the query field,* **(b)** *is the closest field according to the fingerprint distance,* **(c)** *is the closest field according to the Euclidean distance.*



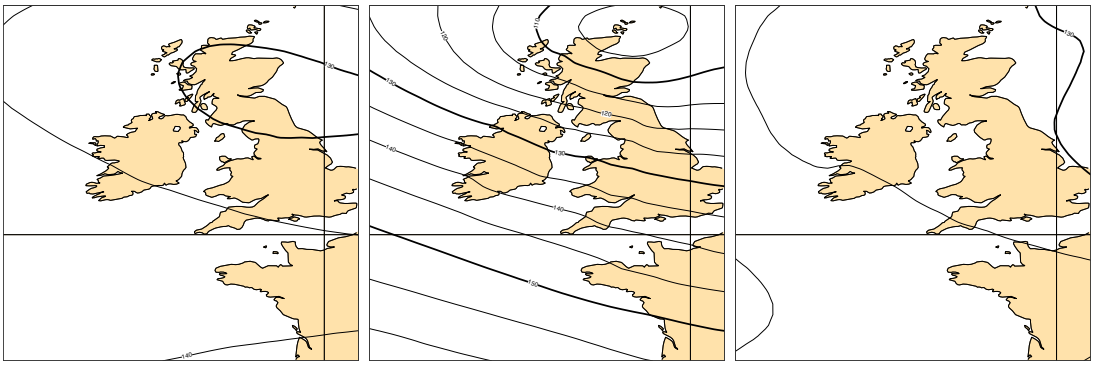
(a) *1981-09-25* **(b)** *2002-02-01* **(c)** *1988-01-05*

Figure 5.18: *Match with the highest error for* Mean sea level pressure. **(a)** *is the query field,* **(b)** *is the closest field according to the fingerprint distance,* **(c)** *is the closest field according to the Euclidean distance.*



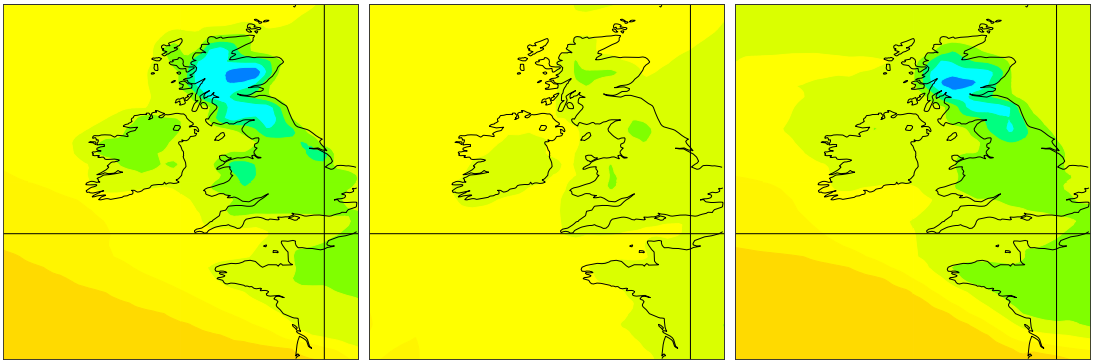
(a) *1988-01-15* **(b)** *1986-01-10* **(c)** *1982-03-28*

Figure 5.19: *Match with the highest error for* Geopotential at 500 hPa. **(a)** *is the query field,* **(b)** *is the closest field according to the fingerprint distance,* **(c)** *is the closest field according to the Euclidean distance.*



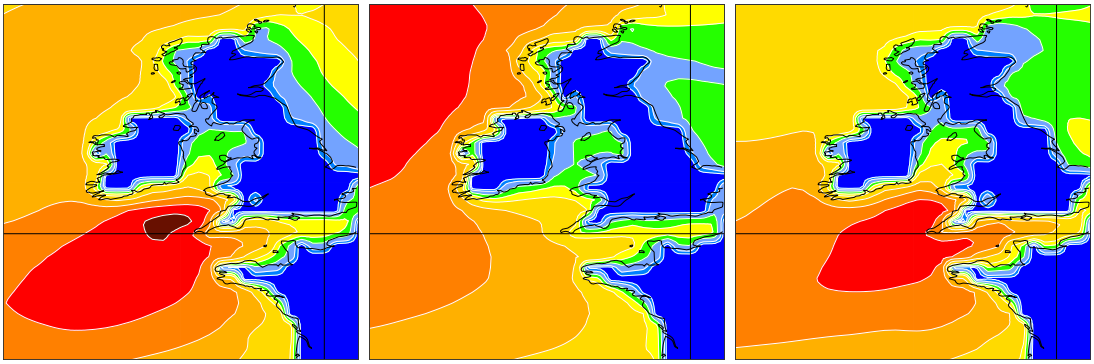
(a) *1980-01-24* **(b)** *1986-01-14* **(c)** *1992-10-29*

Figure 5.20: *Match with the highest error for* Geopotential at 850 hPa. **(a)** *is the query field,* **(b)** *is the closest field according to the fingerprint distance,* **(c)** *is the closest field according to the Euclidean distance.*



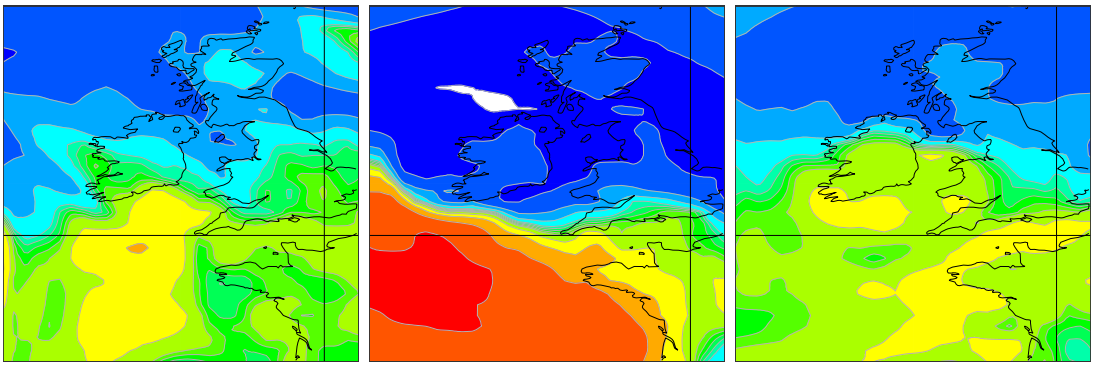
(a) *1995-12-28* **(b)** *1983-02-08* **(c)** *1982-01-07*

Figure 5.21: *Match with the highest error for* Surface air temperature. **(a)** *is the query field,* **(b)** *is the closest field according to the fingerprint distance,* **(c)** *is the closest field according to the Euclidean distance.*



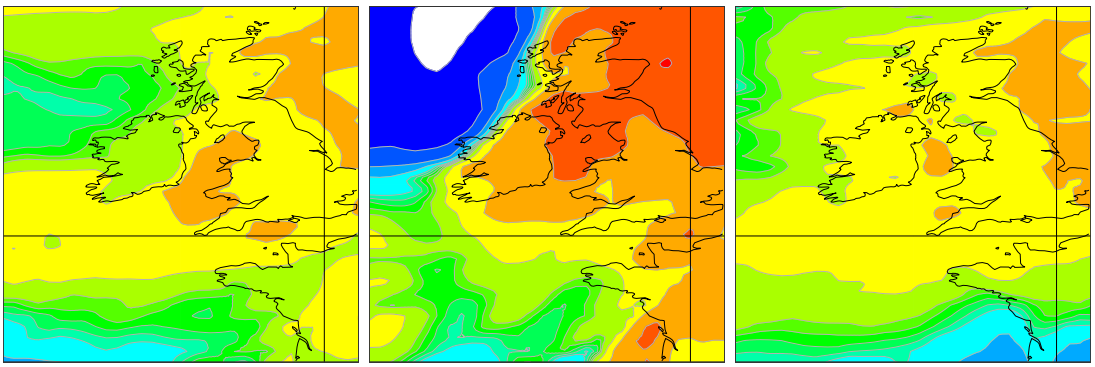
(a) *1998-01-04* **(b)** *1986-12-16* **(c)** *1990-01-25*

Figure 5.22: *Match with the highest error for* Significant wave height. **(a)** *is the query field,* **(b)** *is the closest field according to the fingerprint distance,* **(c)** *is the closest field according to the Euclidean distance.*



(a) *1982-11-04* **(b)** *1994-02-03* **(c)** *2002-09-08*

Figure 5.23: *Match with the highest error for* 10m zonal wind. **(a)** *is the query field,* **(b)** *is the closest field according to the fingerprint distance,* **(c)** *is the closest field according to the Euclidean distance.*



(a) *2016-02-20* **(b)** *2015-01-15* **(c)** *1983-01-07*

Figure 5.24: *Match with the highest error for* 10m meridional wind. **(a)** *is the query field,* **(b)** *is the closest field according to the fingerprint distance,* **(c)** *is the closest field according to the Euclidean distance.*

where $\lfloor x \rfloor$ is the nearest smaller integer from x (floor), and min_v and max_v is the minimum and maximum values possible for the meteorological variable v. In this case, the fingerprint can be as small as 32 bits.

Currently, the dataset considered in this study (see section 2.2) spans 40 years and has hourly fields. The number of fields for a single meteorological variable is therefore $14610 \times 24 = 350640$. As a consequence, the memory footprint of all the fingerprints will be $350640 \times (32/8) = 1402560$ bytes which is about 1.3 MiB.

Of course, this does not take into account the data structure needed to hold this information. Nevertheless, this volume is very small compared to the RAM available on modern computers and can therefore easily fit in memory, allowing for very fast lookups.

Chapter 6

Query by example

The fingerprinting system that has been described so far needs to be queryable. In this research, we consider how users can describe what they are looking for, in an interactive fashion. In the chapter we will therefore explore several ways of letting users perform queries:

- by "painting" a meteorological field directly on the screen;
- by selecting a predefined regime from a list (heatwave, cold-spell, etc);
- by using helper tools to input predefined patterns (high, low, trough, ridge, etc).

We will discuss the need for inputting physically realistic fields (e.g. no snow cover over the ocean). We will describe how constraints can be applied to adjust the user drawn input so it remains realistic (within the climatological values for that field: minimum, maximum, and possibly reasonable gradients).

We will also discuss the problem of allowing users to specify two fields (wind AND temperature).

We will also touch on the presentation of the results, and the need to cluster the results: for example, returning the best 5 matches for a heatwave may return 5 consecutive dates, while it will be better to return 5 dates from different years.

We consider a web-based user interface, with which users interact with the system using a mouse and a keyboard.

6.1 A web-based user interface

An experimental web-based user interface has been developed (see figure 6.1 on the facing page). The top row shows a map representing the current user input, followed by a canvas displaying a greyscale image of the same field, normalised to the interval $[0, 255]$, with 0 representing *white* and 255 representing *black*, and any values in between representing a corresponding level of grey. On the top right is a series of controls and tools that will be described in more details later. Below are three rows of results, labelled with the date at which the corresponding weather situation happens.

Both the map at the top left corner of the user interface and the greyscale image next to it, as well as the underlying matrix of values, are referred to in this chapter as the *query field*. The user can interact with the mouse either on the map or the greyscale image.

The user interface will automatically call the web backend for results 2 s after the last modification to the query field, allowing the user to perform several actions before a query is triggered.

6.2 Interactive user input

6.2.1 A canvas for "painting" fields

Unlike the SIRS system introduced by Ruth (1993) (see section 1.5.4) that allows users to interactively draw isolines in order to modify a meteorological field, in this research, we propose a way to interactively modify fields by modifying their values directly, by applying "paint" strokes with the mouse.

We use the term "paint" for the lack of a better word: in the same way that a physical paintbrush adds paint (i.e. matter) on a canvas, we introduce an analogy, in which a virtual brush will add some of pressure, precipitation or temperature to an existing field, according to the user's brush stroke. The reader is reminded that the graphical representation of a field does not always represent all details of the underlying data (see section 2.3 on the portrayal of meteorological data). The "painting" happens by adjusting the data values and not the chart. The chart is re-plotted automatically based on the modified values.

Below is an illustration of that concept. F is the current state of a meteorological field which is display on the user's screen. The user initiates a brush stroke gesture

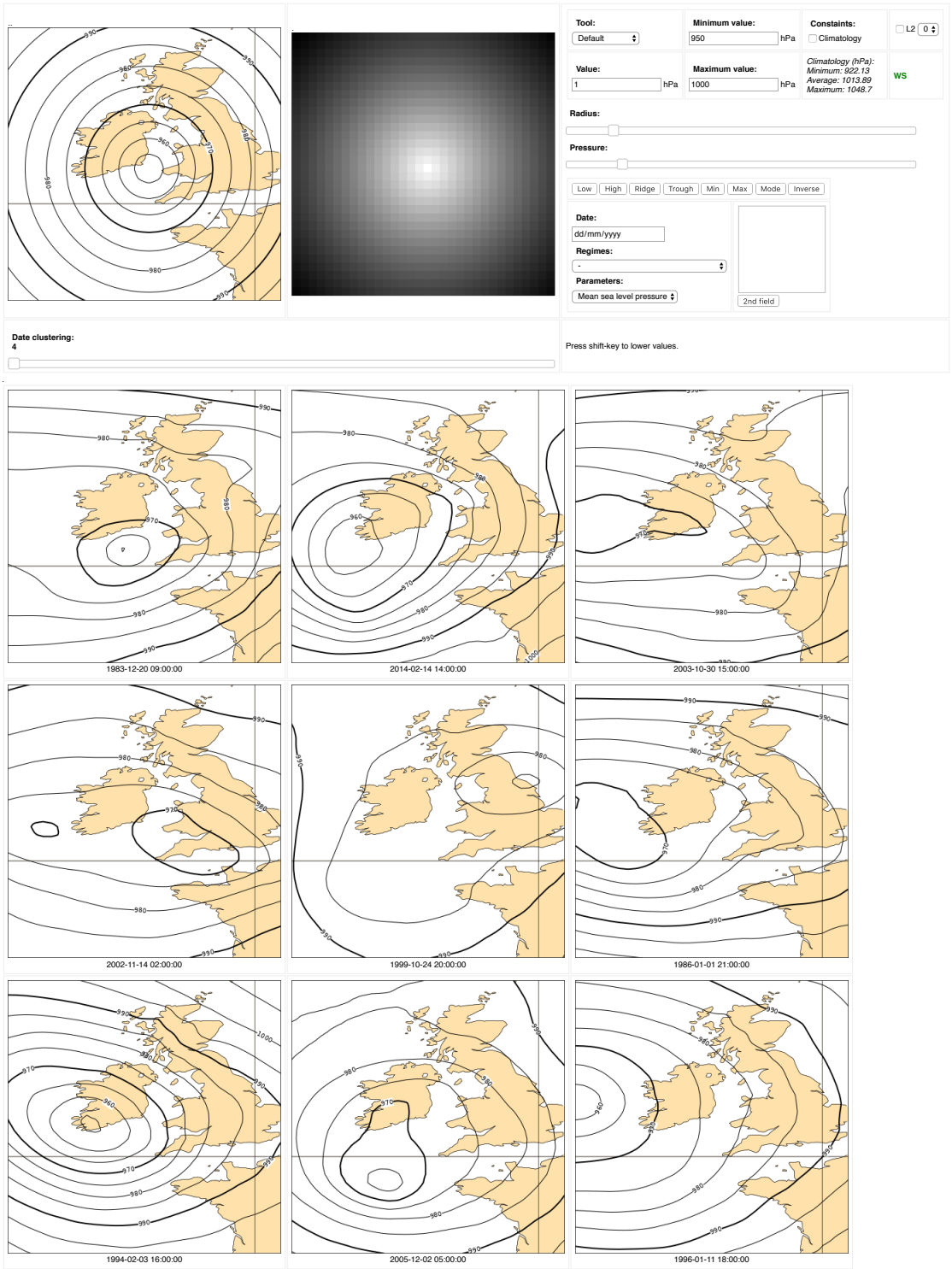


Figure 6.1: *The web-based user interface. The controls are in the top row. The results are shown below.*

starting from the bottom left of the canvas, to the top right of the canvas. The amount of "paint" added to the virtual canvas is recorded as ΔF. The content of ΔF will depend on the type of brush used, as well as the duration of the gesture (see section 6.2.2). ΔF is added to F to give F', the new state of the field displayed on the screen.

$$F = \begin{bmatrix} 3.1 & 4.2 & 5 & 5.3 \\ 2.9 & 3.5 & 2 & 4.2 \\ 2.7 & 2.7 & 3.5 & 4.1 \\ 1.4 & 2.8 & 3.1 & 3.9 \\ 1.5 & 2.2 & 2.8 & 3.2 \end{bmatrix} \quad \Delta F = \begin{bmatrix} . & . & . & 0.2 \\ . & . & . & 0.8 \\ . & . & 1.2 & . \\ 0.1 & 0.7 & . & . \\ . & . & . & . \end{bmatrix} \quad F' = \begin{bmatrix} 3.1 & 4.2 & 5 & \mathbf{5.5} \\ 2.9 & 3.5 & 2 & \mathbf{6.0} \\ 2.7 & 2.7 & \mathbf{4.7} & 4.1 \\ \mathbf{1.5} & \mathbf{3.5} & 3.1 & 3.9 \\ 1.5 & 2.2 & 2.8 & 3.2 \end{bmatrix}$$

6.2.2 Input tools

Tools are used to interact with the query field. They can be considered as virtual paint brushes, akin to the brushes of a software like *Photoshop*.

The tools are selected from a dropdown menu in the user interface. Some tools rely on the values of other widgets. Three tools have been implemented:

Default tool

The default tool is simulating an airbrush, adding an amount of "paint" at regular intervals (set to 0.1 s).

The amount of paint added at a grid point (i, j), at each interval, is defined by the Gaussian function:

$$\Delta F_{i,j} = P \times (F_{max} - F_{min}) \times e^{-d_{i,j}/R^2} \tag{6.1}$$

where $d_{i,j}$ is the distance between the mouse cursor and the grid point (i, j), P is the *pressure* of the brush and R its *radius*, both of which can be controlled by the user using the sliders on the user interface. The effect of changing these values is shown in figure 6.2 on the next page. The advantage of the Gaussian function is that it does not create strong gradients, keeping the query field realistic.

The user can decide to "remove paint", i.e. to change the sign of $\Delta F_{i,j}$ by depressing the shift-key while pressing the mouse button.

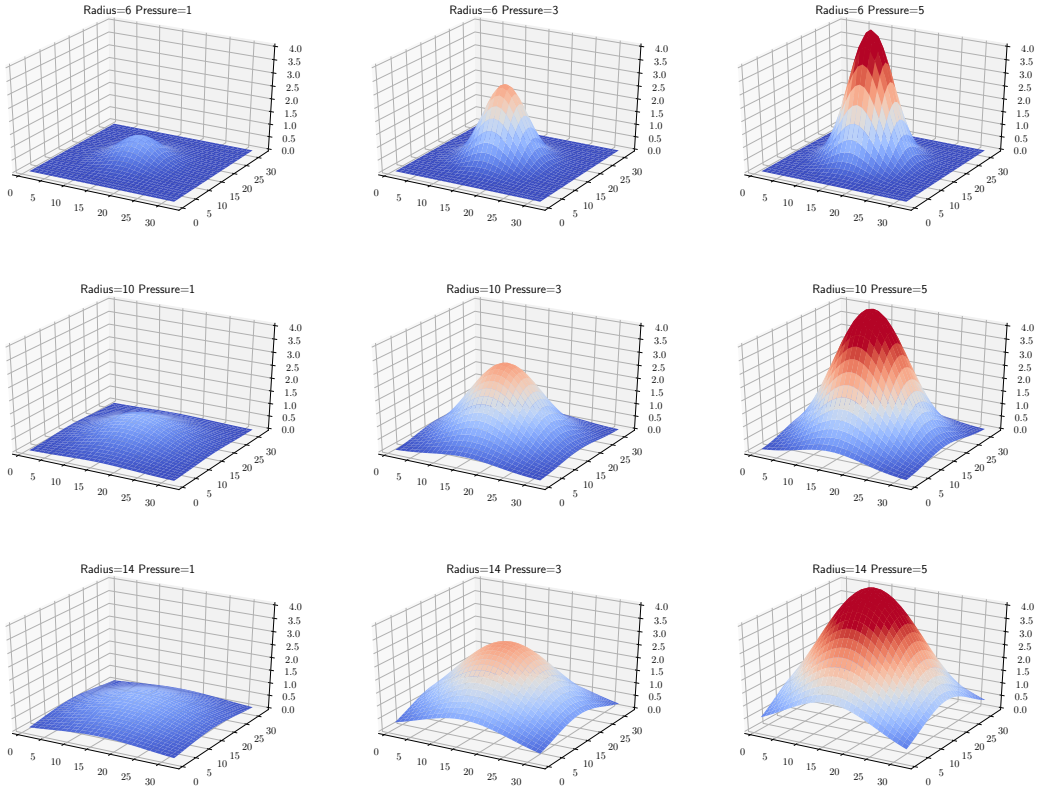


Figure 6.2: *Gaussian function for various values of radius and pressure.*

Levelling tool

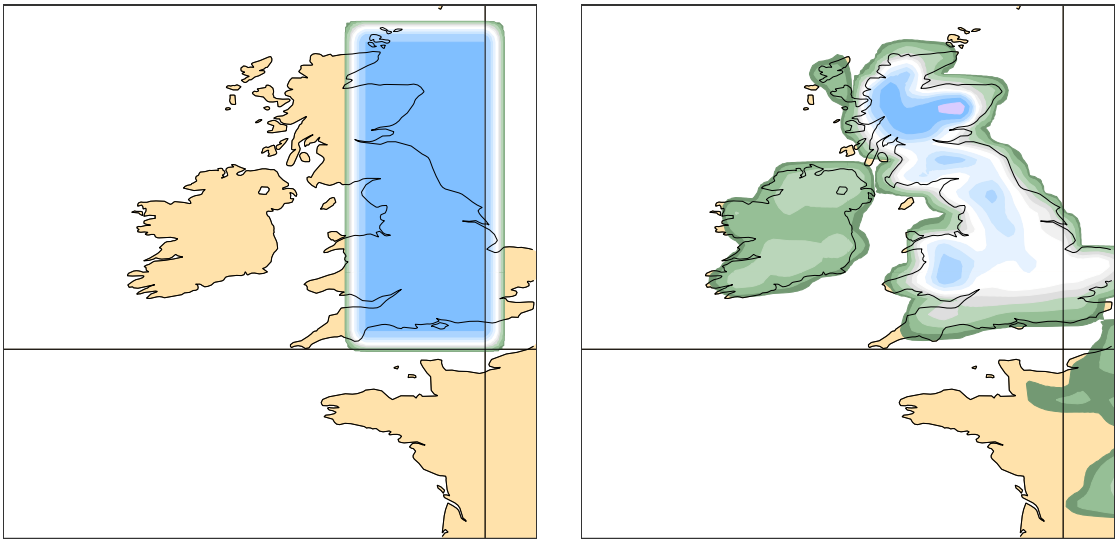
The levelling tool is a variation of the default tool, where the user provides a target value L (the level to reach, provided by the user in one of the user interface input fields). The tool uses a slightly modified Gaussian function:

$$\Delta F_{i,j} = P \times (L - F_{i,j}) \times e^{-d_{i,j}/R^2} \tag{6.2}$$

where L is the target value, Fi, j the value of the grid point (i, j) and P and R are as before. The value of $\Delta F_{i,j}$ is proportional to the difference between the current grid point value at the target value: if $L > F_{i,j}$, $\Delta F_{i,j}$ will be positive, as more "paint" is needed to reach L. Conversely, if $L < F_{i,j}$, $\Delta F_{i,j}$ will be negative, as "paint" needs to be removed to reach L. Once $L = F_{i,j}$, $\Delta F_{i,j}$ becomes zero.

The tool is also applied at regular intervals of 0.1 s. This means that if the user keeps pressing the mouse button over a given grid point, over time all the grid points with the selected radius will have the target value.

As for the default tool, using a Gaussian function allows for smooth transitions and gradients between grid points.



(a) *Query field.* **(b)** *Best match.*

Figure 6.3: *The system is queried for* snow depth *with a rectangular pattern with a constant value of* 35 cm*, and zero elsewhere. A match is returned for 29 January 1979. The green and white shapes surrounding the blue one are artefacts of the plotting software.*

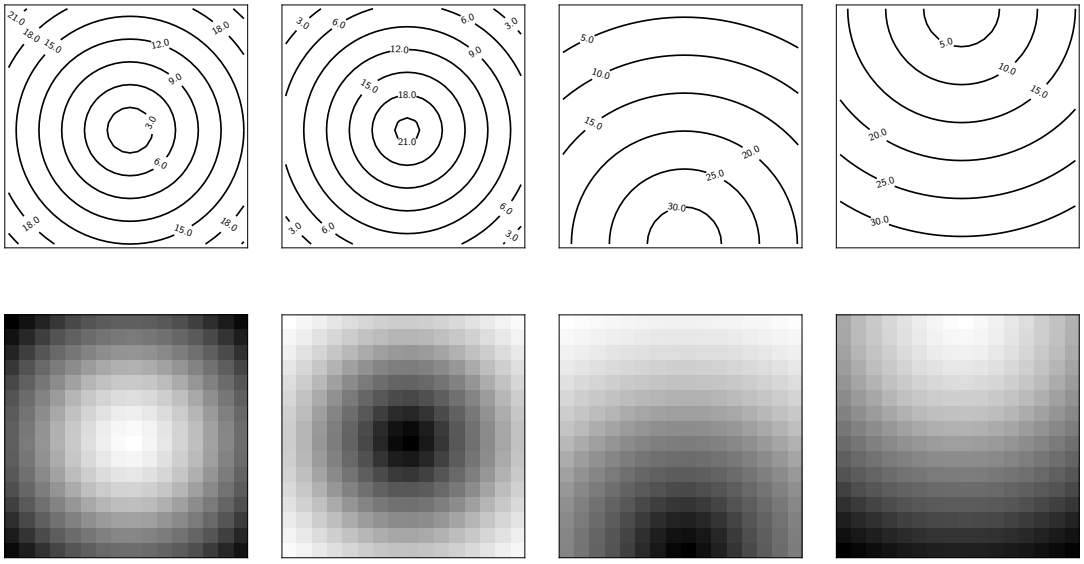
Rectangle tool

This tool functions differently to the previous two: it uses the *rubber band* paradigm, where the user clicks at one point, drags the mouse and releases at another point. The first and last points of that gesture define the two corners of a rectangle. All grid points within that rectangle are given a user-defined value. Unlike the previous tools, this will create unrealistic query fields, with very steep gradients. This is not an issue in itself, as the query can still return results that match the user's need (see figure 6.3).

The rectangle tool could easily be complemented with tools also based on the rubber band concept, which will draw other shapes, such as ovals.

6.2.3 Predefined patterns

Since "painting" on the query field may be difficult and inaccurate, a series of controls are provided that can applies a predefined pattern onto the canvas. The first four buttons correspond to the four major patterns that are exhibited by meteorological fields, in particular pressure and geopotential fields: *Low*, *High*, *Ridge* and *Through* (figure 6.4 on the next page). When users click on the corresponding button, the current field is replaced with the selected pattern, using the current



(a) *Low.* **(b)** *High.* **(c)** *Ridge.* **(d)** *Trough.*

Figure 6.4: *Predefined patterns, plotted for fields varying between* 0 *(F_{min}) and* 31 *(F_{max}). In should be noted that that the shape of the ridge and trough are only valid in the northern hemisphere. For the southern hemisphere, the patterns must be flipped horizontally.*

minimum (F_{min}) and maximum (F_{max}) values specified by the user.

In addition to the four buttons mentioned above, the following buttons can be used to control the query field:

Value
sets all values of the query field to a user-specified value.

Minimum
sets all values of the query field to F_{min}.

Maximum
sets all values of the query field to F_{max}.

Median
sets all values of the query fields to $(F_{max} + F_{min})/2$.

Inverse
applies the transformation: $F_{i,j} \leftarrow (F_{max} + F_{min}) - F_{i,j}$ for each value $F_{i,j}$ in the query field.

6.2.4 Predefined regimes

Another possible way to query the system is to provide the user with a series of predefined queries (i.e. predefined weather regimes), which are described in a catalogue. These can be based on past events, such as the "Great Storm of October 1987", or common weather regimes, such as "heat wave", "cold spell" or "overcast".

Unlike the predefined patterns described in section 6.2.3, which simply modify the values of the currently selected meteorological variable, this input method consists of loading in the user interface a field from a predefined catalogue of field, which may represent a different variable.

Currently, the list of predefined regimes is as follows:

Heatwave:
a constant field of *surface air temperature*, around 40 °C.

Cold spell:
a constant field of *surface air temperature*, around −15 °C.

Great storm of 1987 (msl):
the *mean sea level pressure* field for 16 October 1987, 0 UTC.

Great storm of 1987 (wind speed):
the *10m wind speed* field for 16 October 1987, 0 UTC.

Clear sky:
a constant field of *total cloud cover*, set to 0 %.

Overcast:
a constant field of *total cloud cover*, set to 100 %.

Low pressure (msl):
a low pressure field of *mean sea level pressure*, between 950 hPa and 1000 hPa.

High pressure (msl):
a high pressure field of *mean sea level pressure*, between 1000 hPa and 1020 hPa.

Dry spell:
a constant field of *total precipitations* at 0 mm.

Deluge:
the climatological maximum of *total precipitations*.

Strong wind:
the climatological maximum of *10m wind speed*.

Anticyclone:
a constant field of *mean sea level pressure*, set to the variable's absolute maximum for the selected period and domain (around 1048 hPa).

Deep low:
a constant field of *mean sea level pressure*, set to the variable's absolute minimum for the selected period and domain (around 922 hPa).

Although regimes like "Great storm of October 1987" exist and are retrieved from the archive, "Heatwave" and "Overcast" are artificial fields. Nevertheless, selecting them will successfully return the appropriate matches from the system.

6.2.5 User-provided date

The user interface allows users to input a date and a time in a text field widget. Assuming the date is a valid one and is available in the archive, the corresponding field is downloaded and used as a query field.

This method of providing a query field is useful when the user knows a memorable date, such as a date of a specific storm, or heatwave, and wants to extract from the archive all available analogues.

6.3 Two fields queries

As part of this research, we have considered how to retrieve meteorological situations by providing two different query fields, for example, a field of precipitations and a field of temperatures.

There are two issues to consider:

- how to let the user specify the two fields using the web-based user interface;
- how to query the fingerprint database given two different fingerprints as input.

6.3.1 Selection of two query fields

To address the first point, the user interface offers a button to store the current query field. To select two query fields, the workflow is therefore as follows:

1. Select a meteorological variable to work on.
2. Use the tools and controls to define the query field for that variable.
3. Click on the *2nd field* button to store that query field.
4. Select a new meteorological variable to work on.
5. Use the tools and controls to define that second query field.

Clicking on the *2nd field* button again will revert to a single query field mode.

Figure 6.5 on the facing page shows, in its first row, the result of a query based on a single query field for the variable *mean sea level pressure*. The following rows show two fields queries by combining the *mean sea level pressure* query field with *surface air temperature*, with *total precipitations* and with *snow depth* respectively.

6.3.2 Matching of two query fields

To address the second point, the matching of a pair of fingerprints in the fingerprint database, we need to consider the meaning of such a query: we are looking for meteorological situations for which both queries field match *simultaneously*. A matching algorithm that would combine both distances between fingerprints without taking into account the dates would return matches for situations that did not occur, by considering the first field at a given date and the second field at a different date, and could lead to results such as a high snow cover on a hot summer day.

When performing the matching of two query fields, we are therefore looking for the dates at which both have matches:

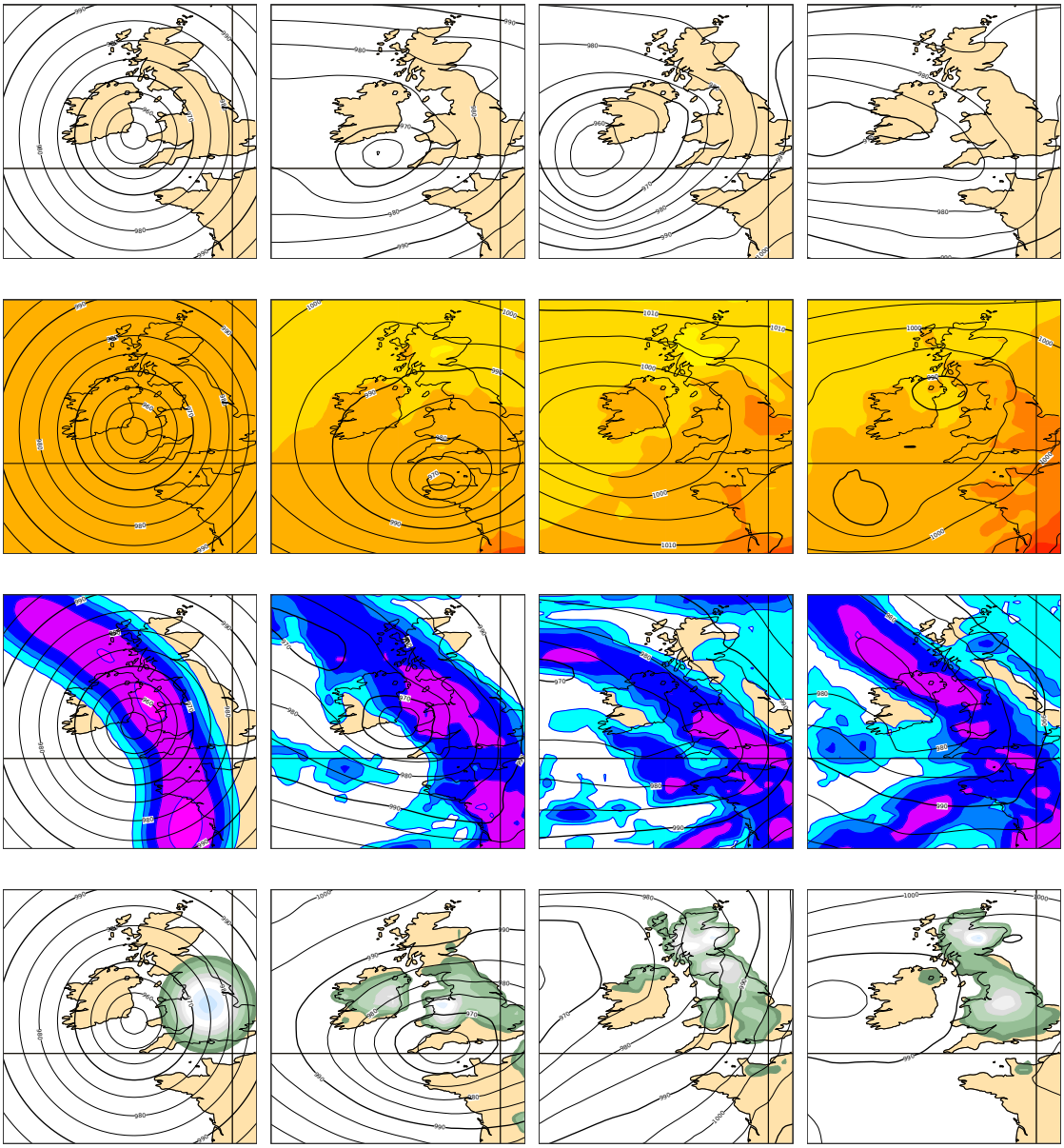
$$dates = dates_1 \cap dates_2 \tag{6.3}$$

where $date_1$ are the dates returned when using the first query field, and $dates_2$ are the dates returned when using the second query field.

The resulting set of dates is then sorted according to the position of the dates in both matching lists:

$$rank(d) = w\ rank_1(d) + (1 - w)\ rank_2(d) \tag{6.4}$$

where d is a date from the result set, $rank_1(d)$ is the position of d in $date_1$ and $rank_2(d)$ is the position of d in $date_2$. w is a coefficient allowing the user to give more or less weight to one query field over the other. The resulting set $dates$ is sorted in ascending values of $rank(d)$, with $d \in dates$.



(a) *Query.* **(b)** *Match 1.* **(c)** *Match 2.* **(d)** *Match 3.*

Figure 6.5: *The first row shows a query for* mean sea level pressure *and the first three matches returned. The same query is then combined with a constant field of* surface air temperature *at* 18 °C *(second row), a drawn field of* total precipitations *(third row) and a field of* snow depth *(fourth row).*

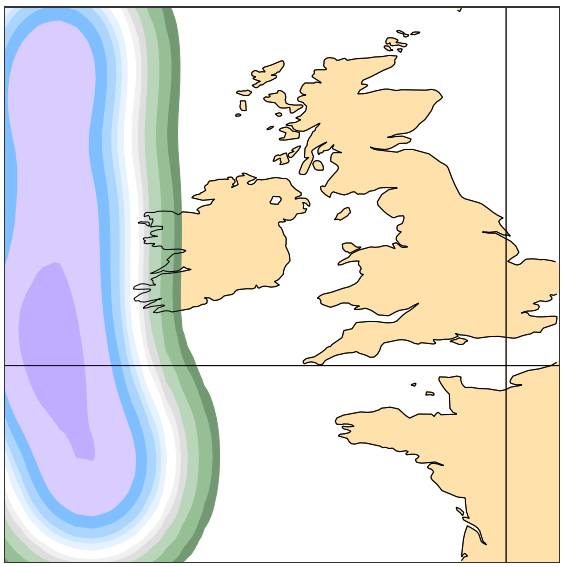


Figure 6.6: *Unrealistic user input for the variable* snow depth. *There cannot be any snow accumulating on the ocean.*

6.4 Realistic inputs

As users are free to paint any pattern on the input canvas, it is likely that they will provide an unrealistic field, i.e. that cannot be seen in nature. Figure 6.6 shows such an impossible pattern, with snow over the Atlantic ocean. Although it can snow over the ocean, the snow will not deposit on the surface of the water. Other unrealistic input could be extreme cold or heat, which are not seen at the latitudes considered.

6.4.1 Climatologies

The climatology of a meteorological variable for a given location describes how this variable varies with time (AMS (2012)). In this study, we will consider the absolute minimum and the absolute maximum of a given variable, at a given grid point, according to the *ERA5* dataset.

Figure 6.7 on page 104 shows the climatological maximum and minimum fields for *snow depth* and *surface air temperature*. The figure shows that the minimum and maximum values for grid points of the *snow depth* field are indeed zero over the seas.

It should be noted that these fields may not have existed, i.e. they may not have been a day in the past for which every grid points reach their maximum or

minimum value simultaneously.

6.4.2 Constraints

In section 6.2.1 we explain how the fields can be "painted" by the user, by adding (or subtracting) "paint" from an original field F to obtain a new field F'; the amount of "paint" is represented by ΔF. We then have the following relation:

$$F' = F + \Delta F \tag{6.5}$$

Assuming that F is contained by the climatology, we require that F' is also constrained by the climatology, so that it remains a realistic field. To achieve that, we will add to extra term δF to the equation above:

$$F' = F + \Delta F + \delta F \tag{6.6}$$

The role of δF is to compensate for values in ΔF that will lead to grid points values that will be outside the constraints. For example, assuming the climatologies for the field of temperature are (in °C below):

$$F_{min} = \begin{bmatrix} -2.1 & -2.2 & \dots \\ \vdots & \ddots & \\ -0.3 & & -10.1 \end{bmatrix} \qquad F_{max} = \begin{bmatrix} 23.4 & 23.8 & \dots \\ \vdots & \ddots & \\ 24.5 & & 28.8 \end{bmatrix}$$

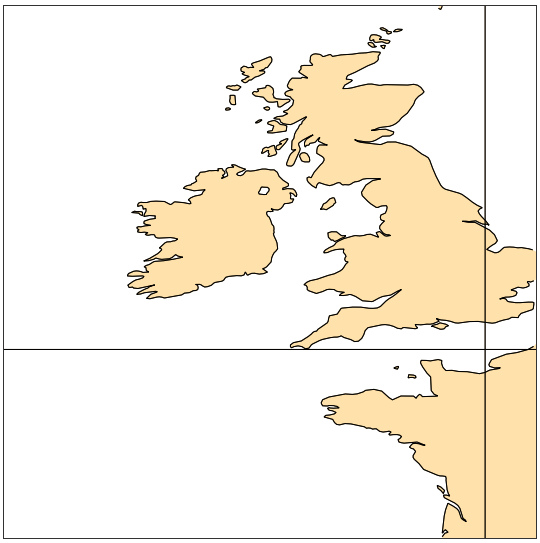
Given F and ΔF below, we can compute F':

$$F = \begin{bmatrix} 18.8 & 18.9 & \dots \\ \vdots & \ddots & \\ 16.1 & & 26.8 \end{bmatrix} \qquad \Delta F = \begin{bmatrix} 3.9 & 3.8 & \dots \\ \vdots & \ddots & \\ 4.0 & & 4.1 \end{bmatrix} \qquad F' = \begin{bmatrix} 22.7 & 22.7 & \dots \\ \vdots & \ddots & \\ 20.1 & & \mathbf{32.9} \end{bmatrix}$$

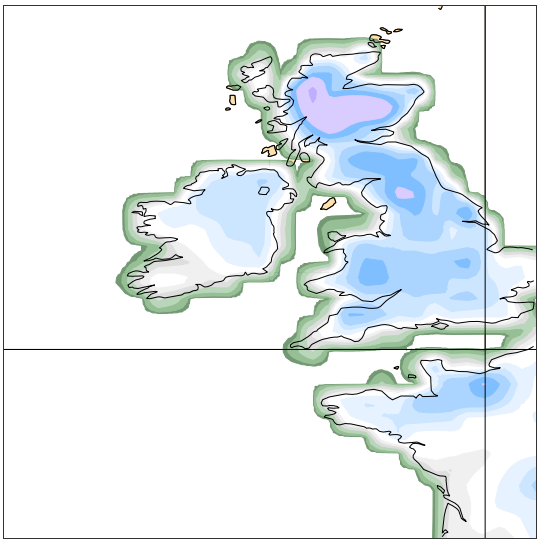
The highlighted value 32.9 in the lower right corner of F' is over the climatological maximum for that grid point, which is 28.8. In that case, F' can remain within the climatological constraints by adding the term:

$$\delta F = \begin{bmatrix} 0 & 0 & \dots \\ \vdots & \ddots & \\ 0 & & -4.1 \end{bmatrix}$$

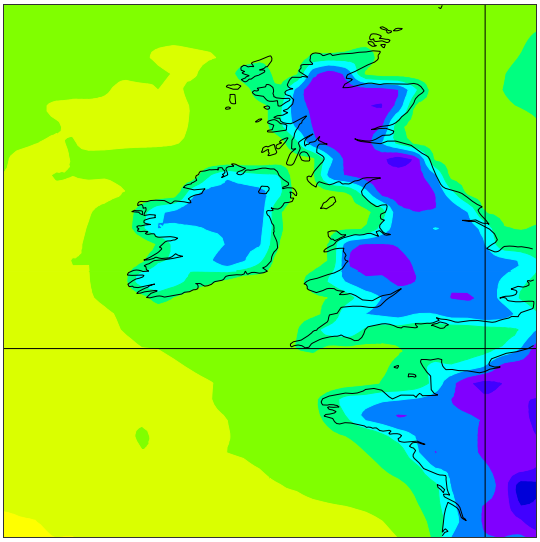
Our aim is that the adjustment δF is minimal so that we respect as much as possible the user's input. If F' was already within the constraints, δF would simply be zero. We will therefore try to minimise $(\delta F)^2$, using a *penalty method* (Smith et al. (2000)).



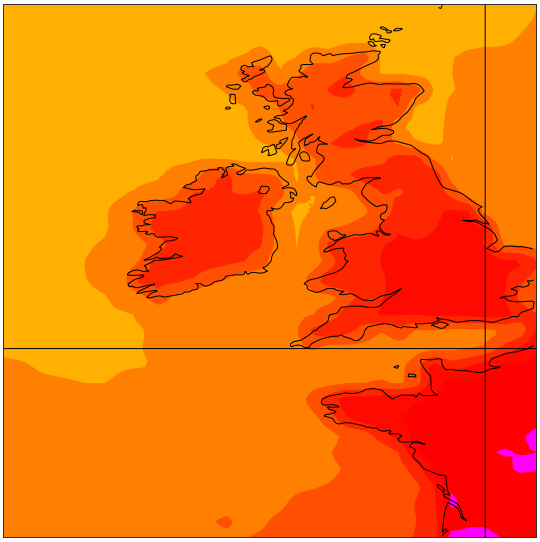
(a) *Climatological minimum for* snow depth.



(b) *Climatological maximum for* snow depth.



(c) *Climatological minimum for* surface air temperature.



(d) *Climatological maximum for* surface air temperature.

Figure 6.7: *Climatologies of selected meteorological variables over the British Isles, computed from* ERA5 *over the period 1979-2018.*

Penalty method

In order to minimise a function $f(x)$ given a set of constraints c_i such that $c_i(x) \leq 0$, one can minimise a family of functions ϕ_k:

$$\Phi_k(x) = f(x) + \sigma_k \sum_i max(0, c_i(x))^2 \tag{6.7}$$

where σ_k is a penalty coefficient. When x is within the constraint c_i, the term $max(0, c_i(x))^2$ evaluates to 0, when x is outside the constraints, the term evaluates to a large number. The algorithm consists of iterating through k, minimising ϕ_k using the result of the previous iteration as a seed, and increasing σ_k at each iteration, putting more and more weight to the constraints.

Procedure *minimise-with-constraints(f, constraints, x_0, iterations)*
  $\sigma \leftarrow 1$
  $x \leftarrow x_0$
  for $i \leftarrow 1$ **to** *iterations* **do**
    $\Phi \leftarrow f + \sigma * constraints$
    $x \leftarrow minimise(\Phi, x)$
    $\sigma \leftarrow increment(\sigma)$
  end
  return x
end

Algorithm 6.1: *Penalty method.*

As stated above, we are looking to minimise δF^2 so that $F' = F + \Delta F + \delta F$ is within a set of given constraints. Given then climatological minimum F_{min} and the climatological maximum F_{max}, we want to constrain F' between F_{min} and F_{max}. We can therefore write the constraints in the form expected by the penalty method:

$$F' \geq F_{min} \tag{6.8}$$

becomes

$$F_{min} - F' \leq 0 \tag{6.9}$$

and

$$F' \leq F_{max} \tag{6.10}$$

becomes

$$F' - F_{max} \leq 0 \tag{6.11}$$

Substituting variables, the function to minimise can be written as:

$$\Phi_k(\delta F) = (\delta F)^2 + \sigma_k[max(0, F_{min} - F')^2 + max(0, F' - F_{max})^2] \tag{6.12}$$

$$= (\delta F)^2 + \sigma_k[max(0, F_{min} - F - \Delta F - \delta F)^2 + max(0, F + \Delta F + \delta F - F_{max})^2] \tag{6.13}$$

It should be noted that the minimisation above will also work for a user-supplied field, or an artificial field from a catalogue (see section 6.2.4). In that case, only F is considered, and the minimisation is applied with $\Delta F = 0$. If the provided field respects the constraints the minimisation will return $\delta F = 0$. If the field provided does not respect the constraint, the minimisation with return a value of δF that contains the smallest adjustments to do in the input field to get a field that is within the constraints. This feature will also be important when applying constraints while providing a smooth user experience (see section 6.4.4).

6.4.3 Other constraints

It can be argued that, from a programming point of view, it will be simpler to adjust the user input to the climatology by simply computing:

$$F' = min(max(F + \Delta F, F_{min}), F_{max}) \tag{6.14}$$

with *min* and *max* being returning the element-wise minimum and maximum respectively.

The approach taken here is a framework in which additional constraints can be added. In particular, limiting the constraints to climatological minimum and maximum is not sufficient. Figure 6.8 on the next page shows that the climatological minimum of the *mean sea level pressure* is itself not realistic: for pressure values around 900 hPa exist in a deep low such as shown in 6.8b on the facing page and not a so-called barometric swamp as shown in 6.8a on the next page.

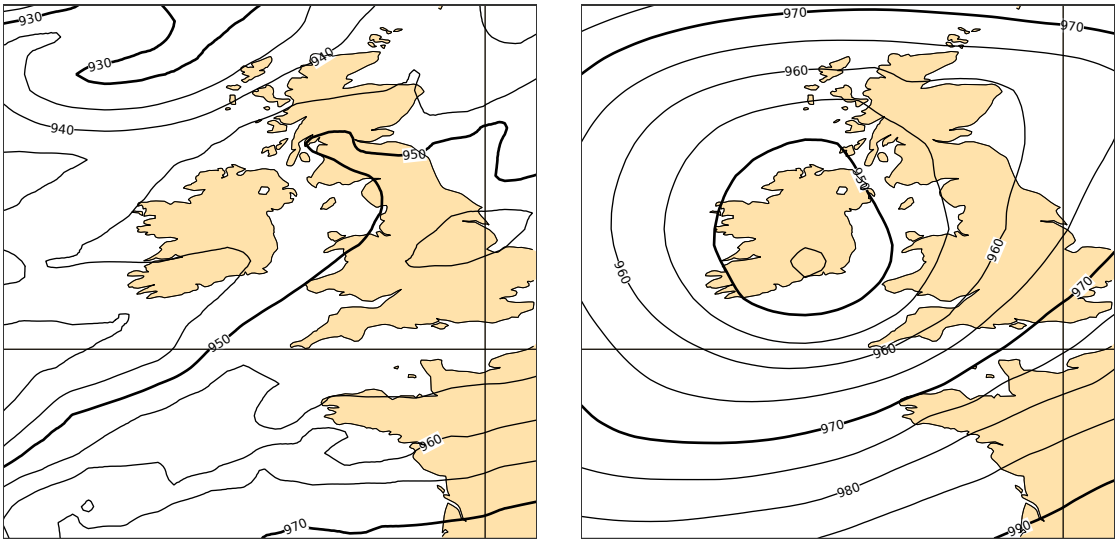
To ensure that user-provided fields are physically realistic, we should also constrain their gradients: for example, it is not possible, in the British Isles, to have a gradient of temperature of 25 °C between two adjacent grid points that are 55 km apart (the maximum gradient over the selected period is 22.1 °C, based on the ERA5 dataset).

Adding constraints on the gradients would simply mean adding an extra term in the definition of $\Phi_k(\delta F)$, with no change to the algorithm.

See section 8.2.3 for more information.

6.4.4 Minimisation in an interactive environment

As part of this work, the algorithm described above has been implemented in Javascript so that it can be run in the user's web browser. The minimisation is



(a) *Climatological minimum field for* mean sea level pressure.

(b) Mean sea level pressure *field, 17 December 1989 at 03UTC.*

Figure 6.8: *6.8a is the climatological minimum at each grid point, while 6.8b is one of the closest matches.*

based on Newton's method (Kelley (2003)).

Procedure *df(f, x, ε)*
  return $(f(x+\varepsilon) - f(x))/\varepsilon$
end

Procedure *minimise(f, x_0, ε)*
  $x \leftarrow x_0$
  $\delta \leftarrow \|f(x)\|$
  while $\delta > \varepsilon$ **do**
    $x \leftarrow x - f(x)/df(f, x, \varepsilon)$
    $\delta \leftarrow \|f(x)\|$
  end
  return x
end

Algorithm 6.2: *Newton's method.*

One issue with iterative methods is that they may converge slowly. In the case of an interactive system, it is important that the interface does not become frozen. According to Nielsen (2009), for a user interface to feel responsive, events should

be handled below 0.1 s, and certainly not over 1 s.

```
Procedure expired(start-time, max-elapsed)
    return (time() − start-time) > max-elapsed
end

Procedure df(f, x, ε)
    return (f(x + ε) − f(x))/ε
end

Procedure minimise(f, x_0, ε, start-time, max-elapsed)
    x ← x_0
    δ ← ‖f(x)‖
    while δ > ε do
        Φ ← f + σ ∗ constraints
        x ← x − f(x)/df(f, x, ε)
        δ ← ‖f(x)‖
        if expired(start-time, max-elapsed) then
            break
        end
    end
    return x
end

Procedure minimise-with-constraints(f, constraints, x_0, iterations,
                                    start-time, max-elapsed)
    σ ← 1
    x ← x_0
    for i ← 1 to iterations do
        Φ ← f + σ ∗ constraints
        x ← minimise(Φ, x)
        σ ← increment(σ)
        if expired(start-time, max-elapsed) then
            break
        end
    end
    return x
end
```

Algorithm 6.3: *Penalty method using Newton's minimisation algorithm. The iterations will stop once a result is found or if a maximum allowed elapsed time is reached.*

When the user "paints" on the canvas, to add an amount ΔF to the current field F, the F is updated by an amount $\Delta F + \delta F$, with δd being the result of the

minimisation described in algorithm 6.3, run for a maximum 0.1 s. This means that the user interface will be non-responsive for that amount of time, which according to Nielsen (2009) is perceived as instantaneous. If the algorithm has not converged in that amount of time, the field F is updated nevertheless, even if it does not respect the constraints, and a Javascript timeout procedure is posted to recheck the constraints after a delay of 0.1 s. In that case, the adjustment δF is computed from the current field F and $\Delta F = 0$. The process is repeated until F is within constraints, i.e. when $\delta F = 0$ (within an epsilon). This is illustrated in algorithm 6.4 and figure 6.9 on the next page.

Procedure *update-field(F, ΔF, constraints, iterations)*
  $\delta F \leftarrow$ *minimise-with-constraints*$(F +$
   $\Delta F, constraints, iterations, time(), 0.1)$
  if $\delta F \neq 0$ **then**
    $F \leftarrow F + \delta F$
    refresh-display(F)
    $setTimeout($*update-field*$(F, 0, constraints, iterations), 0.1)$
  end
end

Algorithm 6.4: *The penalty method is invoked by chunks of* 0.1 s *until the field is within constraints. In Javascript, the function* setTimeout(code, delay) *will run the code provided as the first parameter after a delay specified as the second parameter. In the meantime, the control is given back to the user interface, so the web page is updated to reflect any changes, and the user is able to interact with various controls and widgets.*

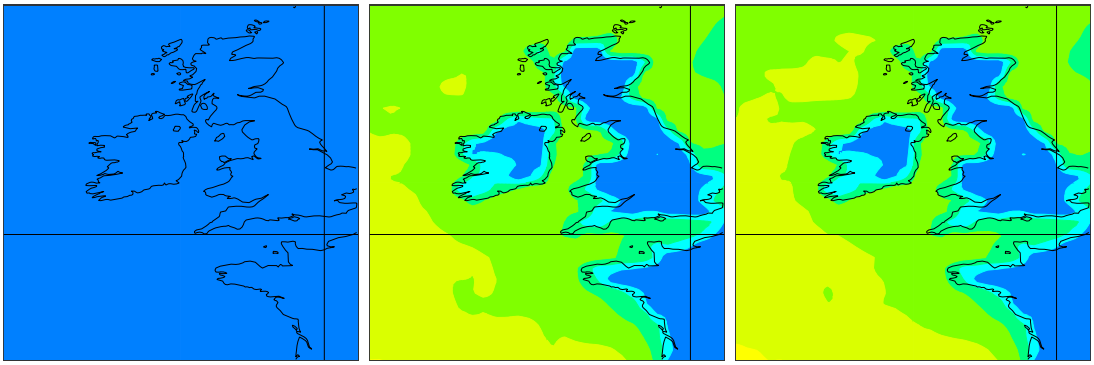
6.5 Presenting results

6.5.1 Presentation of results

Once the user has described a query field using one or more of the input methods described in the section above, the web-based user interface triggers a request to the web backend, by sending the matrix of values representing query field.

The backend will generate a fingerprint from that matrix and will perform a query against the fingerprint database, for a limited number of matches (12 at present). The results are sorted by their distance to the queried fingerprint and then the corresponding list of dates and times are returned to the user interface.

The user interface will create, for each resulting date, an HTML image and set its



(a) *Starting pattern.* **(b)** *Pattern after* 0.1 s. **(c)** *Pattern after* 0.2 s.

Figure 6.9: *Applying constraints on a constant field of* surface air temperature *at* −15 °C *(6.9a). After* 0.1 s *the minimisation is interrupted, and the user is presented with the intermediate result (6.9b), as well as the ability to interact with the user interface. Eventually, the minimisation converges to the climatological minimum (6.9c)*

.

source URL to an end-point in the backend that will return a PGN image for the current variable for that date.

Upon request of an image for a meteorological variable at a given date, the backend will retrieve the corresponding field from the archive, produce the plot from it, and return the resulting image. The output of the retrieval and plotting steps are cached, for faster future accesses.

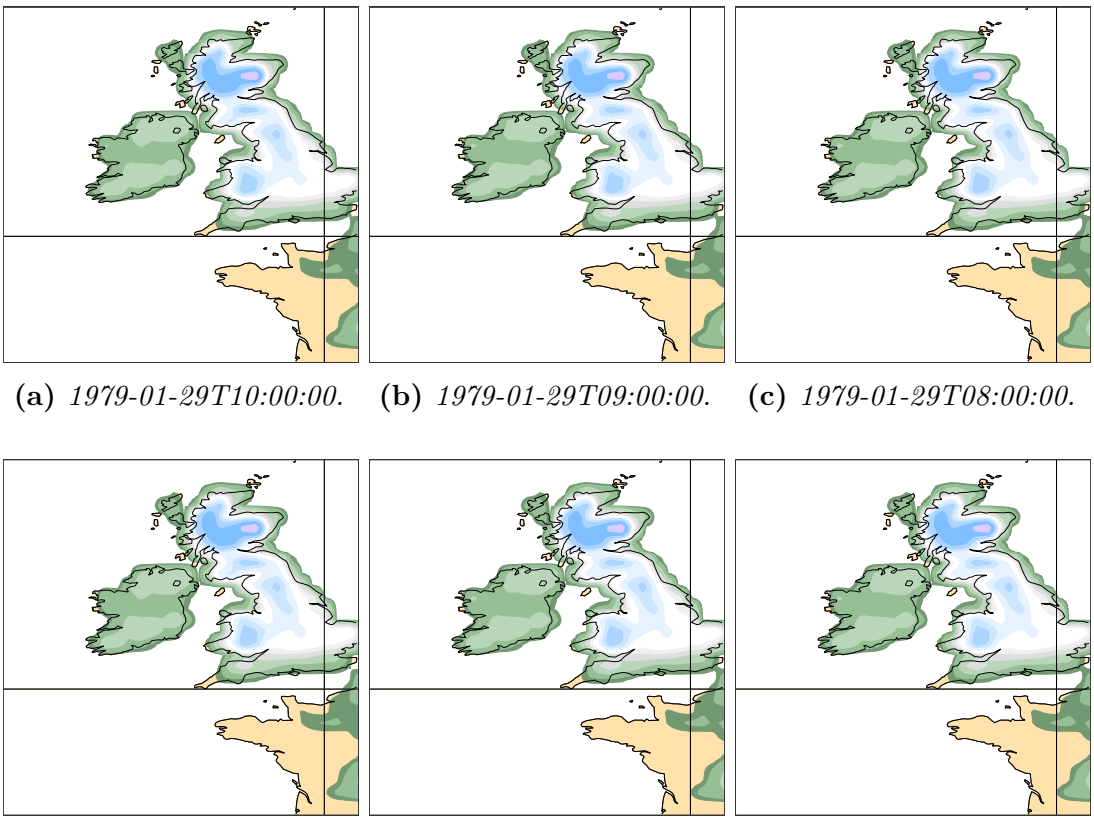
See figure 7.1 on page 121 and section 7.2 for more information on the architecture of the system and how the web frontend and backend communicate.

6.5.2 Issue with persistent weather

One of the issue with presenting the results as described in section 6.5.1 is that when a type of weather remains the same for a large period of time, it is likely that all the best matches for a given query relate to a single time interval.

Snow depth is the best parameter to illustrate that problem (see figure 6.10 on the next page), as the same snow coverage can persist for several days. Given a user query such as *snow cover in Scotland and nowhere else*, the 5 best matches, may all represent the same snow event, e.g. for the same dates. This is exacerbated with a dataset such as ERA5 where the fields are available hourly, and the matches may all represent fields from within a five hours interval.

From a user perspective, such results are not useful, as they may be several similar



(a) *1979-01-29T10:00:00.* **(b)** *1979-01-29T09:00:00.* **(c)** *1979-01-29T08:00:00.*

(d) *1979-01-29T11:00:00.* **(e)** *1979-01-29T07:00:00.* **(f)** *1979-01-29T06:00:00.*

Figure 6.10: *Six first matches when searching for analogues of the* snow depth *field from 29 January 1979 at 10 UTC. No result clustering is applied, there for all results are within hours of each other. Compare with figure 6.12 on page 113.*

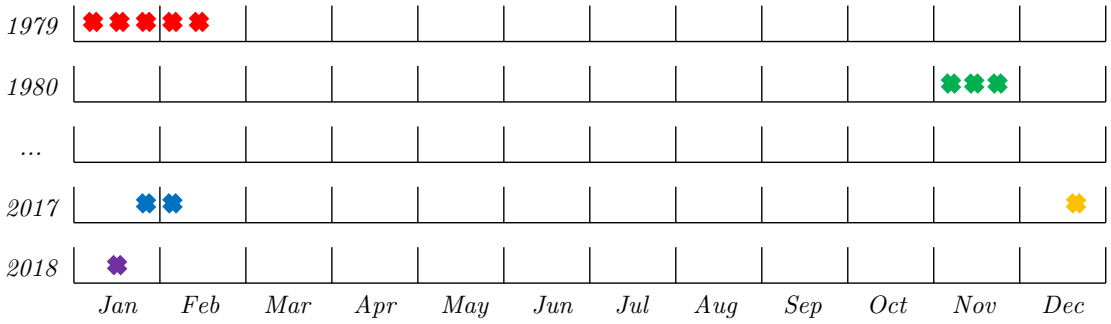


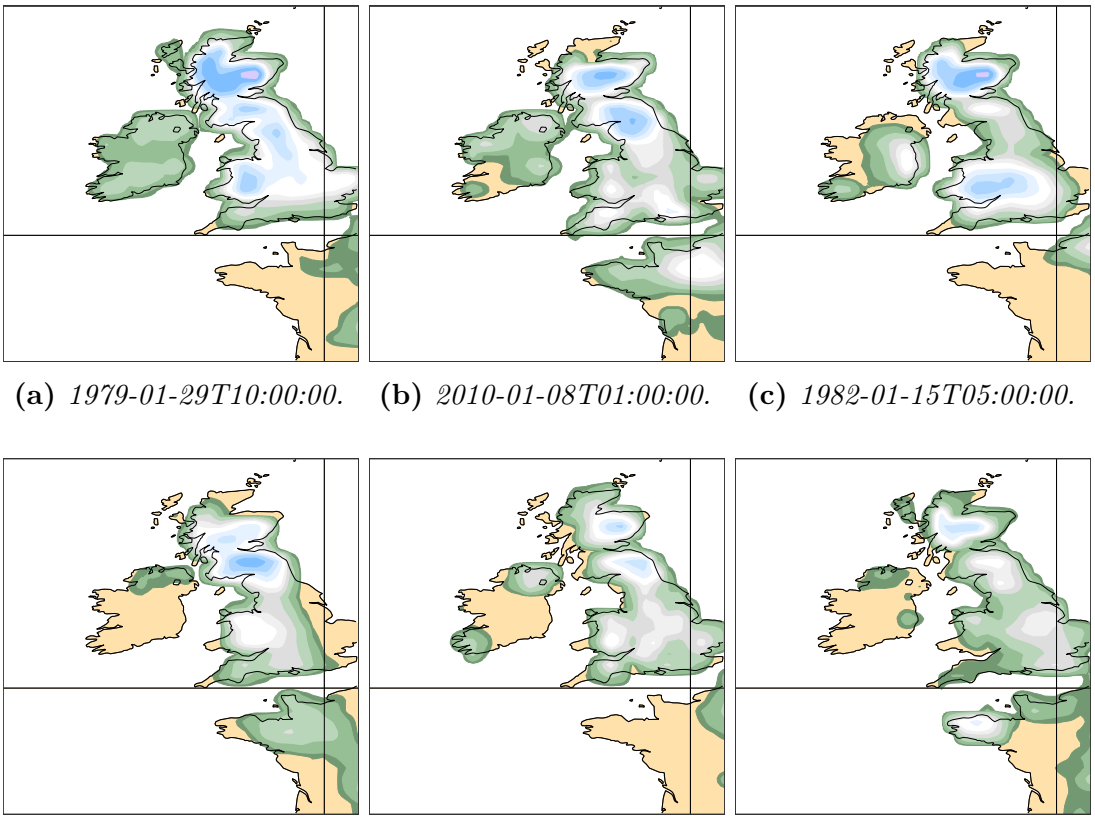
Figure 6.11: *This figure shows a hypothetical distribution of resulting dates for a given query. For persistent types of weather, the dates arise in clusters. From a user perspective, it is more informative to be returned a list of dates representing each cluster, that a list of dates from a single cluster.*

weather patterns which are years apart. Indeed, it will be better for the user to know that the event *snow cover in Scotland and nowhere else* happened so many times during the last forty years, at what dates this event occurred, by returning a significant date for each occurrence.

This is illustrated in figure 6.11. Starting from the user queries, several analogues are retrieved, shown in the figure as crosses on a calendar, showing the dates at which the analogues are found. Assuming that the analogues closest to the user's query are marked in red, and assuming that the system is set up to return the 5 best matches, then the dates in red will be returned. But these dates correspond to a single *event*, for example, some snow lingering on the ground for several days. From the figure, it is clear that the usefulness of the system would be greater if the user was made aware of the other events. The figure also shows that persistent weather events can be considered a cluster of dates in the time dimension.

A solution to this problem is to cluster the results by date and return to the user a list of representative dates. The clustering algorithm used is *Mean shift* (Comaniciu and Meer (2002)) as it allows fine control of the radius of each cluster, by controlling the width of the kernel it uses (see figure 6.12 on the next page).

The clustering radius depends on the meteorological parameter considered, and the location of the area considered. For the British Isles, precipitations will sweep over the domain in a matter of hours, snow may stay on the ground for over a week, and a heatwave or a cold spell may last for a few days. In this research, the slider is added to the web user interface so that the user can experiment with various settings.



(a) *1979-01-29T10:00:00.* **(b)** *2010-01-08T01:00:00.* **(c)** *1982-01-15T05:00:00.*

(d) *1996-02-07T00:00:00.* **(e)** *2013-01-23T00:00:00.* **(f)** *1983-02-11T12:00:00.*

Figure 6.12: *Six first matches when searching for analogues of the* snow depth *field from 29 January 1979 at 10 UTC. The results are clustered with a 365 days radius (1 year), a representative date of each cluster is returned. Compare with figure 6.10 on page 111.*

Chapter 7

Implementation

The software developed in the course of this research is simply called *Analogues*. It has two main components: a web frontend and a web backend. It is available at `https://github.com/b8raoult/analogues`.

In addition, some code has been developed to experiment with various parametrisation of the system, as well as produce maps and graphs for this report.

The prototype of the application can be accessed at `https://cds.climate.copernicus.eu/analogues`. As this is a development system, there is no guarantee that it is functional.

7.1 Software used

In the course of this work, some bespoke software has been developed, that makes use of many Open Source packages. They are listed below.

The following Javascript-based software packages are used for the frontend:

jQuery
(Bibeault and Kats (2008)) is a general-purpose Javascript framework that provides a unified, browser independent, interface to simplify DOM[1] manipulations and asynchronous programming.

[1]Document Object Model: the API to query and manipulate HTML documents.

jQueryUI
(Wellman (2009)) is a Javascript library that extends *jQuery* with a set of user interfaces elements (widgets) and themes.

The following Python-based software packages are used for the backend:

NumPy
(Oliphant (2007); Van Der Walt et al. (2011)) is a Python package that provides support for fast vector and matrix handling. It provides the foundations for most of the other Python scientific packages.

SciPy
(Oliphant (2007); Jones et al. (2014)) is a set of libraries for scientific computing. It offers functions for interpolations, statistics, clustering, linear algebra, optimisations, signal processing, image processing, fast Fourier transforms and much more.

Scikit-learn
(Pedregosa et al. (2011)) is a machine learning library for Python that interoperate with *Numpy* and *SciPy*. It provides, amongst others, a range of clustering algorithms.

PyWavelets
(Wasilewski (2010)) is a package that implements wavelet transform in Python, which supports n-dimensional wavelet transforms as well as over a hundred types of wavelets.

Flask
(Grinberg (2018)) is a web micro-framework in Python. It relies on other libraries such as *Werkzeug* for the routing of HTTP requests and *Jinja 2* for template rendering. It is used to implement the backend.

SQLAlchemy
(Copeland (2008)) is a Python package that implements an object-relational mapper (ORM) on top of popular relational databases. In this project, it is mostly used to abstract away the specifics of the underlying database engine used (*PostgreSQL* or *SQLite*).

Matplotlib
(Hunter (2007)) is a plotting library, supporting all type of scientific graphs. It is used to produce all the plots in this report, except for maps which are produced with *MAGICS* (see below).

Jupyter Notebook
(Pérez and Granger (2007); Kluyver et al. (2016)) previously known as *iPython Notebooks*. Notebooks are web-based interactive documents that embed

Python (or other scripted languages) codes and display their results inline, directly in the user's web browser. This is a powerful tool when used in conjunction with *Matplotlib*. All plots in this memoir have been developed using *Jupyter Notebook*.

ECMWF software is used to handle meteorological data:

MAGICS
(O'Sullivan (1993); Woods (2006)) is a large software library written in C++ that is specialised in plotting meteorological data. It also provides some Python bindings, so it can be used from within a Python program. In the course of this work, the Python bindings have been extended to allow for MAGICS maps to be displayed in *Jupyter Notebooks*.

MARS
(Raoult et al. (1995); Raoult (1997); Woods (2006)) is the tool used to retrieve data from the archive. It can also perform on-the-fly sub-area extraction and re-gridding using MIR (Maciel et al. (2017)), so that only the relevant information is returned to the user. MARS can also compute derived fields, such as computing wind speed from wind components.

ecCodes
(Fucile et al. (2016)) is a library used to encode and decode meteorological data formats (World Meteorological Organization (2009)), in particular, the *GRIB* format which is used to encode meteorological fields. In the course of this research, some Python bindings were developed, which have since been integrated into the main *ecCodes* package.

Other software tools and libraries are also used in the backend:

PostgreSQL
(Drake and Worsley (2002)) is a relational database, that is performant and feature-complete. It is used to implement the fingerprint database. It has been extended with a plugin that adds supports for Hamming distance (Rathna (2011)).

SQLite
(Owens (2006)) is a server-less SQL relational database system. It has been used during this project as a replacement to *PostgreSQL* when developing on a local laptop. During the course of this research, a plugin was implemented that extends *SQLite* with a function computing the Hamming distance between two integers.

uWSGI
(uWSGI (2019)) is a software that implements the *Web Server Gateway In-*

terface protocol that allows the running of Python code in response to HTTP requests. uWSGI also provides support for WebSockets, and relies on the `sendfile(2)` system call to efficiently send files back to the user.

nginx
(Reese (2008)) is a web server. It is used as a reverse proxy for HTTP requests and WebSockets. It communicates with the backend via the uWSGI protocol.

7.2 Software architecture

7.2.1 Frontend

The frontend is a single page web application that is written in *Javascript* and makes use of *jQuery* for most of the event processing and AJAX calls as well as *jQueryUI* for some of the most advanced widgets.

The interactive "painting" of the query field relies on the HTML5 *canvas* element. There are two canvases on the web page, one representing the field in its traditional portrayal (see section 6.2.1), the second is a greyscale rendering. The latter is managed entirely in the user's browser, updated by the Javascript code as the user interacts with either of the canvases. The former relies on *MAGICS* for plotting the fields.

As it is not possible to run MAGICS in the frontend, it needs to be run in the backend. The solution envisaged is to use a *WebSocket* (Wessels et al. (2011)) to connect both ends efficiently. Unlike traditional HTTP requests that are stateless, a WebSocket establishes a permanent connection between the user's browser and a process running in the backend. In this case, the process invokes the MAGICS package to plot the field which values have been received over the socket, and returns the bytes forming a base64 encoded PNG of the resulting plot. The encoded image is then displayed in the canvas.

With a good Internet connection, the elapsed time between a user updating the query field, the field values being sent over the socket, MAGICS plotting the resulting map, the map being sent back to the browser and eventually being displayed in the canvas is fast enough to be perceived as quasi-instantaneous. This step is marked as (1) in figure 7.1 on page 121.

The interactive tools introduced in section 6.2.2 are responsible for modifying the query field based on the user's interaction with one of the two canvases. Tools are implemented as Javascript objects (in the object-oriented sense), and implement

methods to react to the *mouseDown*, *mouseMove* and *mouseUp* and update the query field accordingly.

The code for penalty method minimisation (see section 6.4.2) that applies constraints of the query field is also implemented in Javascript. The speed at which the minimisation converges depends therefore on the performance of the user's browser and computer. As any long-running Javascript code would freeze the browser page and lead to poor user experience, the minimisation is run in chunks of 0.1 s, yielding the control back to the user interaction between each chunk. This is described in details in section 6.4.4.

Also implemented in Javascript is the code to generate the predefined patterns described in section 6.2.3. The predefined regimes (see section 6.2.4) are fetched from the backend using an AJAX call.

A short delay after the user has finished modifying the query field using one or more of the available input methods (see section 6.2), the frontend issues an AJAX call to the backend with the content of the query field and various other options. The backend returns the list of matching dates (step (2) in figure 7.1 on page 121), sorted according to their matching score, i.e. the distance between fingerprints, and filtered according to the clustering radius provided by the user (see section 6.5.2).

For each date returned, the source of one of the resulting HTML images is set to a URL referring to that date. The browser will then automatically resolve these URLs and fetch the corresponding maps from the backend (step (3) in figure 7.1 on page 121). The dates are also given as captions to the images.

7.2.2 Backend

The role of the backend is to respond to WebSocket events, AJAX calls and HTTP requests. The backend is written in Python, using the *Flask* web micro-framework that caters for the HTTP protocol. *Flask* itself is run within *uWSGI* that takes care of the management of threads and processes, ensuring that the service is up and running and uses the right amount of computing resources (CPU, memory). *uWSGI* also handles the communication with the *nginx* reverse proxy that is exposed on the open Internet.

The backend manages one WebSocket connection per user. As the user interacts with the web pages, changes to the query field are received over the socket. MAGICS is invoked to plot the field according to its default portrayal. The resulting plot is encoded and sent back to the user's browser via the same socket (step (4) in figure 7.1 on page 121). As the socket stays fully connected, there is an instance

```
SELECT   valid_date,
         distance(fingerprints.s,
                  fingerprints.r,
                  query_s,
                  query_r) AS distance
FROM     fingerprints
ORDER BY distance ASC
LIMIT    100
```

Listing 7.1: *Pseudo SQL code used to retrieve the dates of the nearest neighbours of the query fingerprint* $\langle query_s, query_r \rangle$.

of MAGICS loaded in memory for each user; this contributes to the performance and responsiveness of the system.

When the frontend requests the execution of a query, the backend is invoked via an AJAX call, describing the query fields and other options, such as the date clustering radius. Upon reception of the message, the backend computes the fingerprint of the query field. This fingerprint $\langle query_s, query_r \rangle$ is used to build an SQL statement similar to the one given in listing 7.1, where *distance*() is a function that implements the distance between fingerprints as described in section 5.1 (step (5) in figure 7.1 on the next page). The limit 100 here is greater than the desired number of results (e.g. 12) because the resulting list of dates is then clustered using the *MeanShift* clustering algorithm (see section 6.5.2). The first date of each cluster is then collected and the resulting list is sent back to the user's browser. If the clustering algorithm does not lead to at least 12 different clusters, the process is redone by rerunning the SQL query with a larger limit, when possible.

In order to display the maps corresponding to the dates resulting from a query, the frontend will issue, for each of the dates, a HTTP `GET` with an URL referencing that date. On receiving such a request, the backend will first check in its cache if the plot has already been done for a previous request. If so, the plot is sent back to the user's browser using the `sendfile(2)` system call, for performance. If the plot was not found in the cache, the backend checks if the field for that date is present in the cache. If this is the case, the field is plotted, and the plot is added to the cache (step (7) in figure 7.1 on the facing page) and sent back; otherwise, the field is first retrieved from the archive and added to the cache (step (6)) and then plotted, and finally sent to the user.

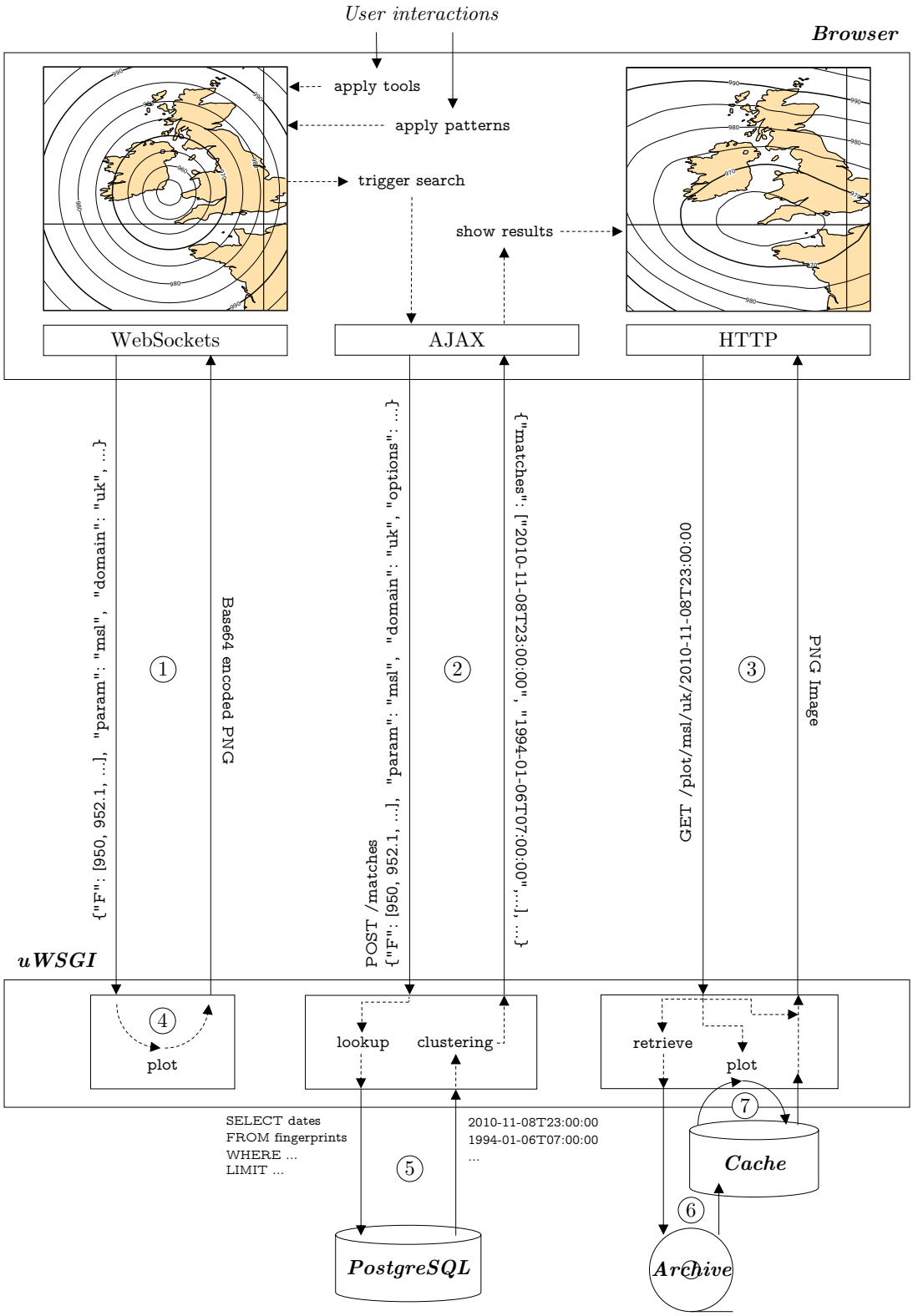


Figure 7.1: *Interactions between the user's web browser (frontend) and the backend. In the actual deployment, there is a* nginx *acting as a reverse proxy. It has been omitted here for clarity.*

7.3 Deployment

The *Analogue* software is deployed as part of the infrastructure of the *Copernicus Climate Data Store* (Raoult et al. (2017)). *Copernicus* is the European Union's Programme for Earth Observation.

One of the components of this programme is the *Copernicus Climate Change Service* (C3S) which aims at providing an authoritative source of knowledge to support adaptation and mitigation policies. The *Climate Data Store* (CDS) provides the technical infrastructure of the C3S.

The CDS is a distributed system, providing access to datasets via a service-oriented architecture. The datasets cover satellite measurements of Essential Climate Variables (Bojinski et al. (2014)), climate reanalyses, climate projections, as well as sectoral information, such as health, energy, transportation or tourism amongst others.

The CDS provides a unified REST (Fielding (2000)) API to download any data it serves, by dispatching users' requests to a broker which in turn will forward them to the relevant data source via a service implementing the *adaptor* design pattern (Gamma et al. (1995)).

The CDS is deployed in a private cloud infrastructure running *Open Stack*. As the ERA5 reanalysis is one of the most popular datasets available through the CDS. A large subset (around 1 Petabyte, 1.6 billion meteorological fields) has been copied online, indexed by a disk-only MARS server installed on the same cloud.

In order to get efficient access to ERA5, the *Analogue* software is deployed in its own virtual machine, running on the same cloud. The software also shares its web reverse proxy with the CDS, so that the result of this research may become an integral part of the service, and be open to the public.

The virtual machine is running Centos 7, with the Linux 64 bits kernel 3.10. It has 64 GiB of memory, and 40 single-core virtual CPUs running a 2.2 MHz, with a 32 KiB L1 cache, a 4 MiB L2 cache and a 16 MiB of L3 cache.

Two file systems are mounted on the virtual machine, one to host the fingerprint database, the other to serve as a cache for retrieved fields and generated maps. For performances, the underlying disks of both file systems rely on solid-state devices (SSD).

Figure 7.2 on the next page illustrates that deployment: the *Analogue* software is shown in orange. The web layer that is shared with the Climate Data Store is shown in green, while access to the CDS data is displayed in blue.

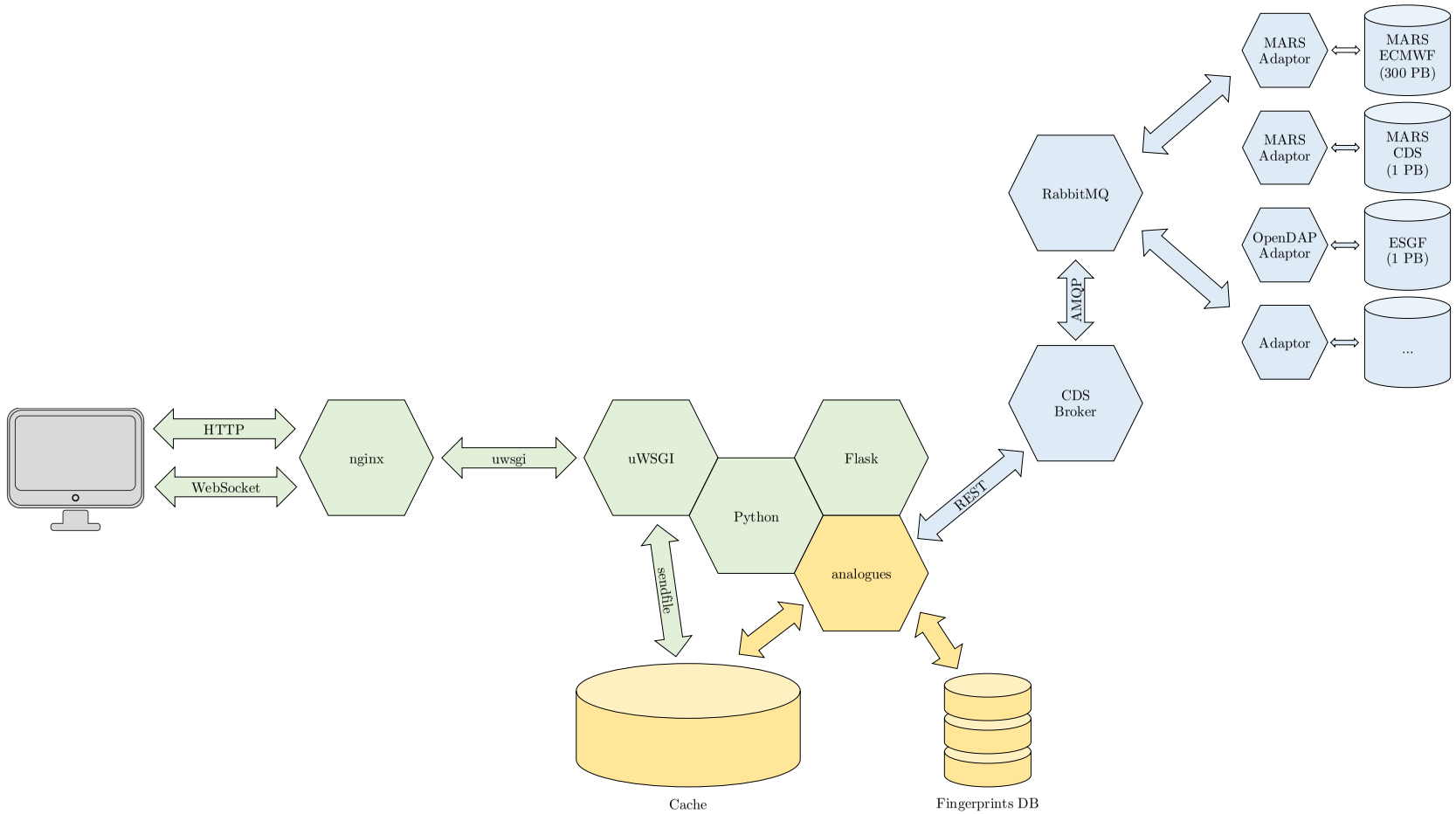


Figure 7.2: *Deployment of the* Analogue *software within the* Copernicus Climate Data Store

Chapter 8

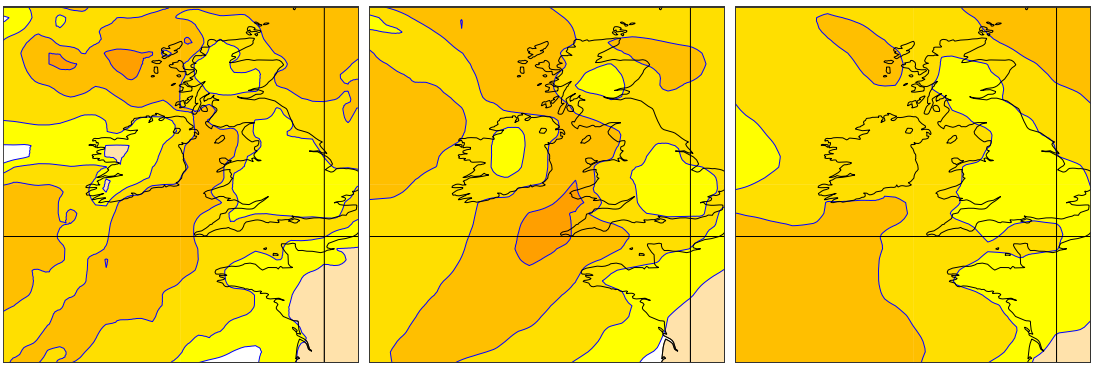
Conclusion

At the onset of this work, we set ourselves the following objectives:

- Define a scheme to extract a fingerprinting from meteorological fields. The computation of fingerprints should be fast; the resulting information should be small and distances between fingerprints should be representative of distances between fields. Searching for nearest neighbours using fingerprints should be fast enough to allow interactive queries. Furthermore, fingerprints can be computed once for each field, so they can be used without the need to retrieve data from the archive.
- Define an objective measure of the effectiveness of a fingerprinting method, so different methods can be compared, and their parametrisation tuned to minimise the error between nearest neighbours between fingerprints and the nearest neighbours between the fields they represent.
- Implement a web-based, interactive application allowing users to search for weather analogues by describing weather patterns of interests, and display the matching results. As for any user-facing interface, usability and responsiveness are key.

8.1 Results

During the course of this research, we have shown that we can use wavelet decomposition to generate fingerprints of fields from a selection of meteorological variables. The fingerprints retain enough information from the original field, and can therefore be used as a proxy for the fields, when comparing them. These fingerprints are several orders of magnitude smaller than the original field, and can



(a) *Query* **(b)** *Result (Fingerprint)* **(c)** *Closests (Euclidean distance)*

Figure 8.1: Query *is the field used to query the system (10m wind speed for 24 December 2015).* Result *is the best match according to the fingerprinting method (18 January 1999) and* Closest *is the closest field according to Euclidean distance (19 November 1997).*

easily fit in memory (see section 5.3).

We have proposed a way to measure the performance of a fingerprinting scheme and used it to tune various methods and select the most effective one. Furthermore, this will allow any future work to measure if a new proposed scheme improves the quality of query results.

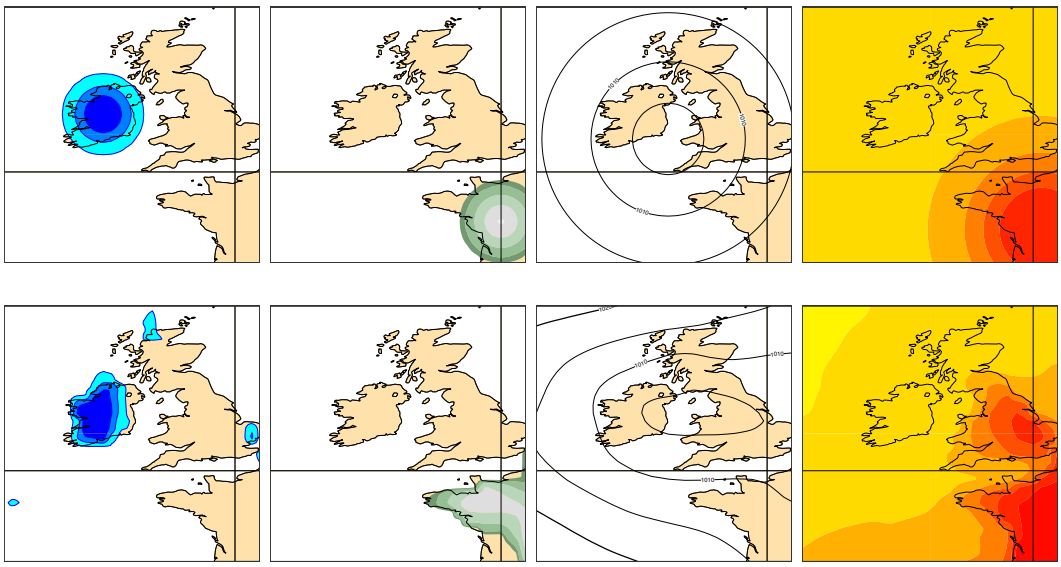
We have computed fingerprints for a selected number of meteorological variables from the *ERA5* dataset, which provides hourly data for a period of 40 years. The result has been stored an indexed in SQL database.

We have implemented a web-based application that allows users to perform interactive queries and be provided with matching results. We have considered several ways with which a user could express a query, from sketching patterns onto the screen to selecting weather regimes from a predefined list. This application will later be made available to expert users, such as forecasters and researcher, as part of the *Climate Data Store* (see section 7.3).

8.1.1 On user perception

We have successfully devised a system that allows users to search for weather analogues interactively. Nevertheless, the quality or the accuracy of the results is a matter of perception, and is often more subjective than objective:

- The Euclidean distance that we are comparing to has its own limitations,



(a) Total precipitations **(b)** Snow depth **(c)** Mean sea level pressure **(d)** Surface air temperature

Figure 8.2: *Using artificial fields as queries (first row), and the corresponding best matches (second row).*

in particular in very high dimensions, where it is subject to the curse of dimensionality. Furthermore, this distance is not a good measure of similarity between two fields that are slightly shifted in space, for example, two field showing the same low pressure pattern but positioned a few hundred kilometres from each other. From a user perspective, they should be the same. From the Euclidian distance perspective, they are not.

- The traditional portrayal of meteorological fields (section 2.3) may be hiding some details, such as very small amounts of precipitations, that are taken into account when computing distances, but are not when visually comparing two maps. This point, and the previous one, are both illustrated by figure 8.1 on the facing page. The figure shows a case where the best match according to the fingerprinting method is perceived as more "similar" than the best match according to Euclidean distance.

- Since we cannot expect a user to accurately "draw" a physically realistic weather situation, the system allows users to sketch a weather pattern. In that case, the matching field *will not* be similar to the query, as far as the Euclidean distance is concerned. So there is this duality in the user's expectation that the system should return very close matches when queried with "real" fields, but also should return satisfactory results when queried with "artificial" fields (see figure 8.2).

8.2 Future work

The results presented in this report open the door for many future research opportunities, which can be pursued independently: improvement of the fingerprints, work on the input of query fields or using fingerprint for other applications than looking for weather analogues.

8.2.1 Refine the fingerprinting method

In this research, we have been focussing on the Haar wavelet. Other wavelets have been successfully used in other disciplines, such as the Ricker wavelet (also known as the "Mexican hat wavelet") which is used in seismology (Ricker (1953)). JPEG2000 uses the CDF 5/3 wavelet for lossless, and CDF 9/7 wavelet for lossy compression (Usevitch (2001)), and is used by the WMO GRIB format to encode meteorological fields. These other wavelets may lead to better results.

To capture the intensity of the fields, we have selected the average value of the field. Adding other statistics such as the minimum, maximum and standard deviation may lead to better results, but will generate bigger fingerprints. These values will have to be factored in when computing distances between the fingerprints.

8.2.2 Implement an efficient data structure to represent the index

Currently, the index composed of all the fingerprints for the area and period considered is stored in a *PostgreSQL* database (see section 7.1). As the Hamming distance is computed using a bespoke extension, searching for the best match is done using a brute force linear scan of the whole index. As the index is small enough to fit in memory, queries can be performed in within a time short enough to be perceived as instantaneous by users.

Once the number of fingerprints to be managed become very large, efficient indexing methods must be used. We will be looking at existing indexing method and implement the most suitable to our problem, such as of Baluja and Covell (2008) who base their indexing on a mixture of *MinHash* (Broder (1998)) and Locally Sensitive Hashing (Slaney and Casey (2008)).

8.2.3 Consider climatological gradients in constraints

The concept of constraints was introduced in section 6.4.2, so that users can be guided to input query fields that are realistic. In this research, we have considered constraining fields to their climatological minimum and maximum.

In section 6.4.3 we alluded to additional constraints, such as respecting realistic gradients between grid point values. This can be as simple as constraining gradients to be lower than a given constant, for example, there cannot be a difference of more than 25 °C between two neighbouring grid point over a selected geographical area. Again, this constant value can be derived from the climatology based on the *ERA5* dataset.

Given $gradient(F)$ a function that returns the gradient of F:

$$G' = gradient(F') \tag{8.1}$$
$$= gradient(F + \Delta F) \tag{8.2}$$
$$\tag{8.3}$$

and G_{max} the maximum gradient possible.

$$\|G'\| - G_{max} \leq 0 \tag{8.4}$$

we just need to add to the definition of the penalty function $\Phi_k(\delta F)$ (see section 6.4.2) the following term:

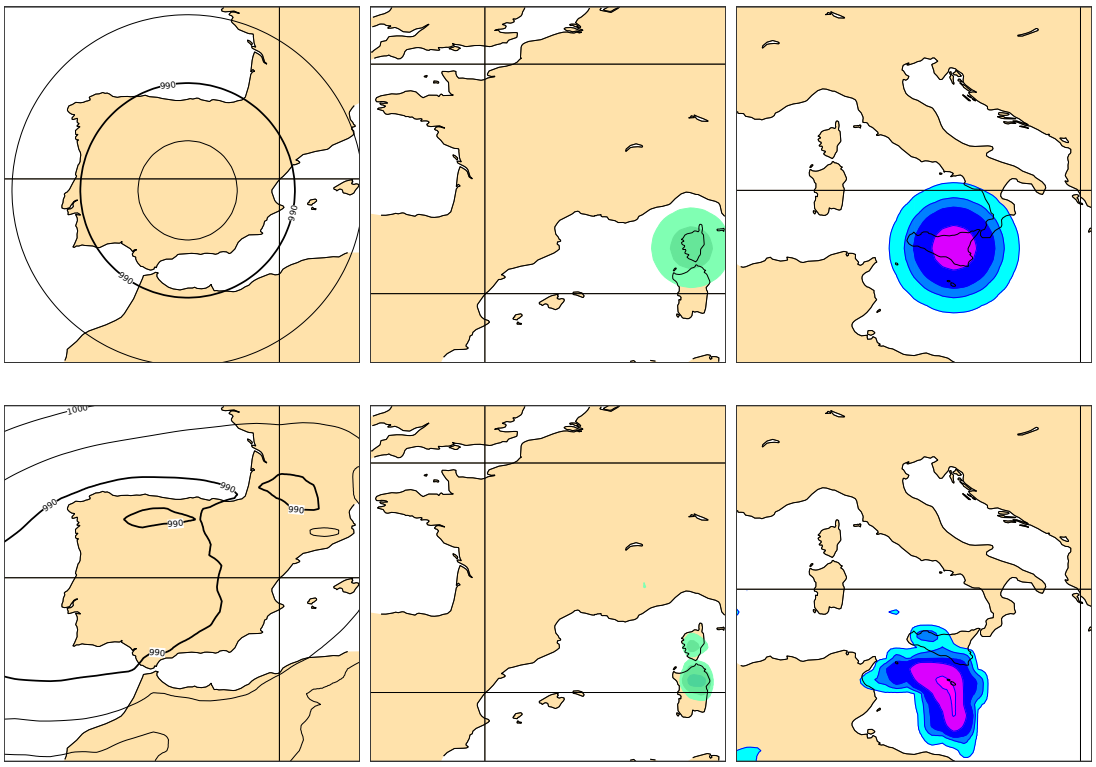
$$max(0, \|gradient(F + \Delta F)\| - G_{max})^2 \tag{8.5}$$

It is also possible to consider more specific gradient-based constraints, such as having different constants for north-south and west-east gradients, or even a per-grid point maximum gradient value.

The gradient represents the first spatial derivative. The concept can be extended to the second spatial derivative, the third spatial derivative, etc., thus having more and more realistic query fields.

8.2.4 Extend to the globe

During our initial research, we have been focussing on weather patterns over the British Isles. The system can trivially be extended to domains of identical size, and provide acceptable results even with the parametrisation of the fingerprint method that was selected to be most effective on the original area. Figure 8.3 on the following page shows queries and their results over Spain, France and Italy.



(1) *Low pressure in Spain, 2 February 2009.* **(2)** *Snow in Corsica, 17 December 2007.* **(3)** *Heavy rain in Sicily, 2 November 1982.*

Figure 8.3: *Artificial queries (top) and their first results (bottom), for other domains.*

Future research should study how changing the size of the domain would affect the fingerprint, as well as considering using user-provided areas of interests. One solution could be to partition the globe and collect fingerprints for each section.

8.2.5 Consider climate projections

The *Climate Data Store* provides an adaptor to the Earth System Grid Federation, or ESGF (Williams et al. (2011)). The ESGF is the repository of the climate projections of the Coupled Model Intercomparison Project, Phase 5 - CMIP5 (Taylor et al. (2012)). This dataset is the basis for the report of the Intergovernmental Panel on Climate Change (IPCC), which assesses the effect of climate change amongst other things.

The CMIP5 dataset is therefore available to the *Analogue* software, so it is possible to consider looking for weather analogues to the centuries to come. This will, of course, require careful consideration: the climate projections are run at a much lower resolution than ERA5 (1.5° instead of 0.5°)

8.2.6 Add a keyword or textual search

The concept of a catalogue of predefined regimes was introduced in section 6.2.4. The catalogue associates titles such as "Heatwave" or "Great storm of October 1987" with preset weather situations. This is an opportunity to add a new input method based in textual search: the user could simply type "overcast", the system will perform a free text lookup in the catalogue, and use the corresponding query field to trigger a search for analogues.

The catalogue could be extended with a set of keywords as well as an abstract for each entry to facilitate the text search. The system could make use of a popular search engine such as *Solr* (Grainger and Potter (2014)) or *Elasticsearch* (Gormley and Tong (2015)).

8.2.7 Analogues of well-known events

Extreme weather events often have very strong impacts on society and are studied for many years. For example, Europe suffered from a major heatwave in August 2003, which killed around 15 000 people in France (Fouillet et al. (2006)). Figure 8.4 on the next page shows that this well-known event is found by the system when querying with a field representing the climatological maximum for *surface*

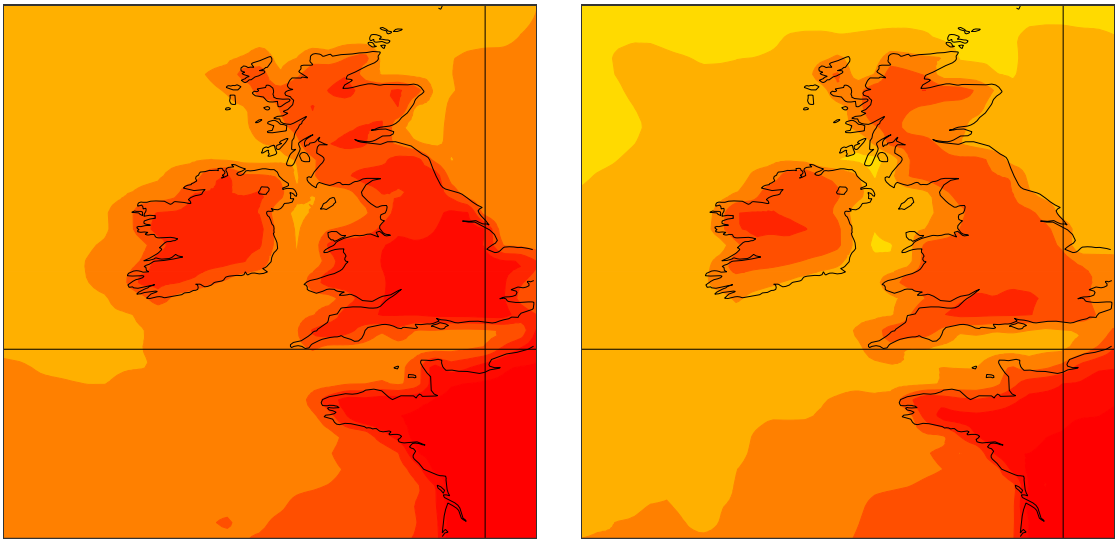


Figure 8.4: *Finding events: querying the system with the climatological maximum for* surface air temperature *(left) will return the date of 8 August 2003 16UTC (right).*

air temperature. The system will also return similar events from the archive. This will provide researchers with a list of similar events through-out the archive, which they could study.

Other examples of extreme events would be the great storm of October 1987, that caused casualties and made consequent damages in the south of England, or storms Lothar and Martin that crossed Europe in December 1999, which also killed many people and generated losses of several billion euros.

8.2.8 Consider time evolution

Meteorological situations are evolving in time as shown in figure 8.5 on the facing page. Further research could consider the use of 3D wavelets, with 2 dimensions of space as proposed for this work, and one dimension of time; alternatively, this work could be an extension of the two fields problem described in section 6.3.

Typical queries would be:

- a rapidly evolving wind storm;
- a band of rain crossing the country in 24 hours;
- a long-lasting high pressure situation;
- a heatwave lasting for several days.

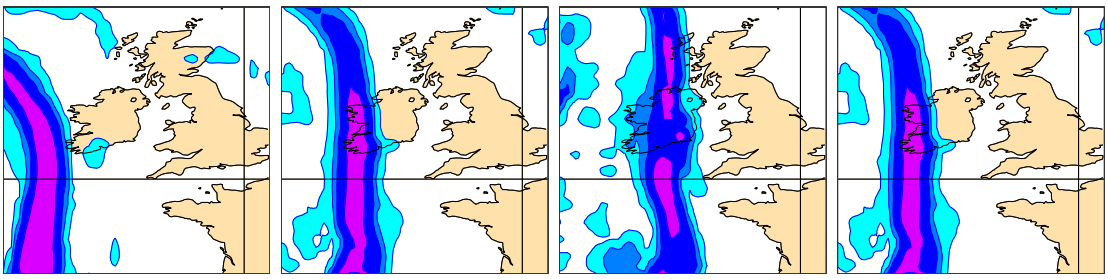


Figure 8.5: *Evolution of a field of* total precipitations*, every six hours, starting 9 February 1989 at 0 UTC.*

One of the challenges will be for users to describe such sequences, so that the system can be queried.

8.2.9 More localised search

Currently, when matching fields using their fingerprints, the whole geographical domain is considered. So a user cannot search for "heavy precipitations in the south-east, whatever happens in the rest of the country": if a user inputs a field with high values of precipitations over London, only fields with precipitations over that area are considered. If there are heavy precipitations over London, but also in Scotland, these fields will not match, although they could be a useful match to return to the user.

A more concrete example would be a user searching for days of strong Mistral, a wind that blows from the Rhone valley into the Mediterranean, in the south of France (assuming this domain is available in the user interface). For that, it is reasonable to think that the user will simply wish to "paint" strong values of *10m wind speed* around the French Mediterranean coast, ignoring the north of the map.

This could be implemented by only considering where the user has interacted with the canvas and mask out the rest of the domain. This solution presents two challenges: how to perform a partial fingerprint match, and how to preserve the usability of the web page.

8.2.10 Fingerprints as proxies for meteorological fields

We have shown in section 5.2.2 that performing a hierarchical clustering using the fingerprint was giving encouraging results. A possible research would be to study to what extent fingerprints can be used in the stead of their corresponding field, in any algorithm that considers distances between fields.

8.2.11 A possible application: finding cyclones

A possible application of the proposed fingerprinting scheme is to extend the search for matching patterns at different a geographical location.

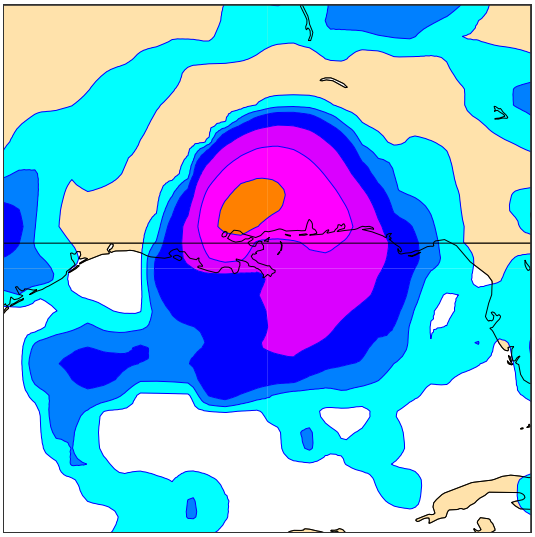
We have conducted the following simple experiment, using *total precipitations*:

- a query fingerprint is computed for the area 37°N 97°W 21.5°N 81.5°W, for the 29 August 2005, which is the location of hurricane Katrina for the corresponding date;
- the northern hemisphere for 24 September 2005 (date of hurricane Rita) is scanned using a 16°×16° window, and fingerprints are computed for these windows and compared to the query fingerprint.
- the best 10 matches are plotted of a global map, with their transparency proportional to their matching rank.

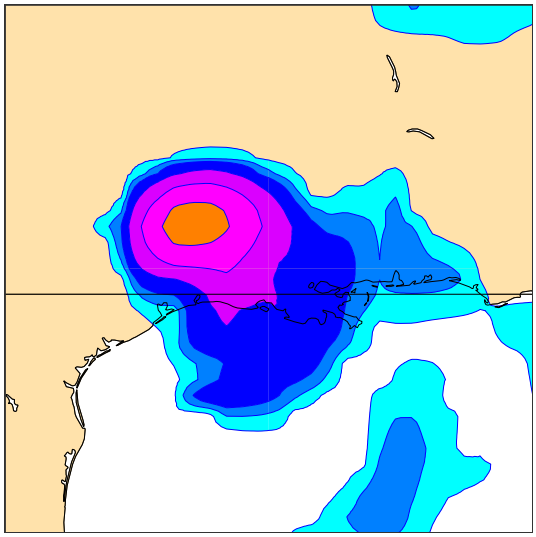
Figures 8.6 on the next page show the query and best matching area of the *total precipitations*. The best match corresponds to the location of Hurricane Rita, which although is also in the Gulf of Mexico, is different from the location of Hurricane Katrina. Running the same experiment with *10m wind speed* gives similar results.

It should be noted that the parametrisation of the fingerprinting scheme used as been tuned for the British Isles (see section 2.2), but is nevertheless effective over other parts of the globe.

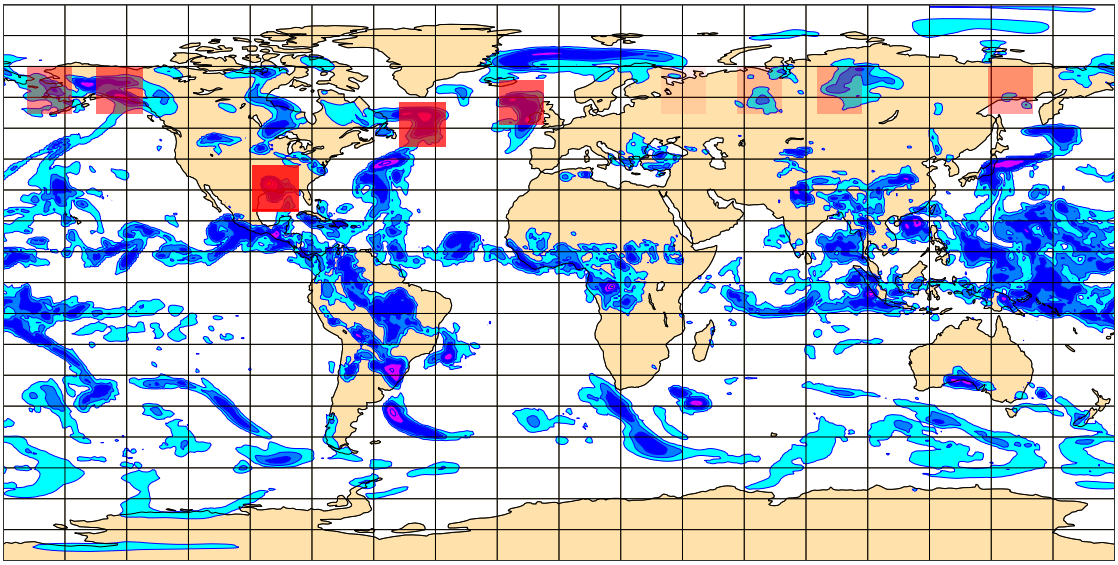
This simple result is very encouraging and should be investigated further.



(a) *Query pattern:* Total precipitations, *hurricane Katrina.*



(b) *Best result:* Total precipitations, *hurricane Rita.*



(c) *10 best matches are highlighted, the more opaque the better the match*

Figure 8.6: *Using the Hurricane Katrina* total precipitations *field as a query (2005-08-29 at 00 UTC) will return Hurricane Rita as a best match (2005-09-24 at 00UTC). This example is using the ERA Interim dataset, in which* total precipitations *are accumulated over 6 hours.*

Bibliography

AMS (2012). Climatology - Glossary of the America Meteorological Society. http://glossary.ametsoc.org/wiki/Climatology.

Aref, W. G. and Ilyas, I. F. (2001). Sp-gist: An extensible database index for supporting space partitioning trees. *Journal of Intelligent Information Systems*, 17(2-3):215–240.

Baluja, S. and Covell, M. (2008). Waveprint: Efficient wavelet-based audio fingerprinting. *Pattern recognition*, 41(11):3467–3480.

Bengtsson, L., Ghil, M., and Källén, E. (1981). *Dynamic Meteorology: Data Assimilation Methods*, volume 36. Springer.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517.

Berrisford, P., Dee, D. P., Poli, P., Brugge, R., Fielding, M., Fuentes, M., Kållberg, P. W., Kobayashi, S., Uppala, S., and Simmons, A. (2011). The ERA-Interim archive Version 2.0.

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When Is "Nearest Neighbor" Meaningful? In Goos, G., Hartmanis, J., van Leeuwen, J., Beeri, C., and Buneman, P., editors, *Database Theory — ICDT'99*, volume 1540, pages 217–235. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bibeault, B. and Kats, Y. (2008). *jQuery in Action*. Dreamtech Press.

Bojinski, S., Verstraete, M., Peterson, T. C., Richter, C., Simmons, A., and Zemp, M. (2014). The Concept of Essential Climate Variables in Support of Climate Research, Applications, and Policy. *Bulletin of the American Meteorological Society*, 95(9):1431–1443.

Broder, A. (1998). On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pages 21–29, Salerno, Italy. IEEE Comput. Soc.

Bruce, L. M., Koger, C. H., and Li, J. (2002). Dimensionality reduction of hyperspectral data using discrete wavelet transform feature extraction. *IEEE Transactions on geoscience and remote sensing*, 40(10):2331–2338.

Carroll, E. B. (1997). A technique for consistent alteration of NWP output fields. *Meteorological Applications*, 4(2):171–178.

Carroll, E. B. and Hewson, T. D. (2005). NWP Grid Editing at the Met Office. *Weather and Forecasting*, 20(6):1021–1033.

Castellanos, J., Gómez, S., and Guerra, V. (2002). The triangle method for finding the corner of the L-curve. *Applied Numerical Mathematics*, 43(4):359–373.

Chang, N.-S. and Fu, K.-S. (1980). Query-by-pictorial example. *IEEE Transactions on Software Engineering*, 6(6):519.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619.

Copeland, R. (2008). *Essential Sqlalchemy*. O'Reilly Media, Inc.

Daubechies, I. (1988). Orthonormal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 41(7):909–996.

Dee, D., Balmaseda, M., Balsamo, G., Engelen, R., Simmons, A., and Thépaut, J.-N. (2014). Toward a consistent reanalysis of the climate system. *Bulletin of the American Meteorological Society*, 95(8):1235–1248.

Dee, D., Uppala, S., Simmons, A., Berrisford, P., Poli, P., Kobayashi, S., Andrae, U., Balmaseda, M., Balsamo, G., and Bauer, P. (2011). The ERA-Interim reanalysis: Configuration and performance of the data assimilation system. *Quarterly Journal of the Royal Meteorological Society*, 137(656):553–597.

Delle Monache, L., Eckel, F. A., Rife, D. L., Nagarajan, B., and Searight, K. (2013). Probabilistic Weather Prediction with an Analog Ensemble. *Mon. Wea. Rev.*, 141(10):3498–3516.

DeMers, D. and Cottrell, G. W. (1993). Non-linear dimensionality reduction. In *Advances in Neural Information Processing Systems*, pages 580–587.

Do, M. and Vetterli, M. (Feb./2002). Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance. *IEEE Transactions on Image Processing*, 11(2):146–158.

Drake, J. D. and Worsley, J. C. (2002). *Practical PostgreSQL*. " O'Reilly Media, Inc.".

Eckley, I. A. (2001). *Wavelet Methods for Time Series and Spatial Data*. PhD thesis, University of Bristol.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press.

Evans, M. and Murphy, R. (2014). A Historical Analog-Based Severe Weather Checklist for Central New York and Northeastern Pennsylvania. *Journal of Operational Meteorology*, 2(18).

Fielding, R. (2000). Representational state transfer. *Architectural Styles and the Design of Network-based Software Architecture*, pages 76–85.

Fouillet, A., Rey, G., Laurent, F., Pavillon, G., Bellec, S., Guihenneuc-Jouyaux, C., Clavel, J., Jougla, E., and Hémon, D. (2006). Excess mortality related to the August 2003 heat wave in France. *International archives of occupational and environmental health*, 80(1):16–24.

Frauenfeld, O. W., Zhang, T., and Serreze, M. C. (2005). Climate change and variability using European Centre for Medium-Range Weather Forecasts reanalysis (ERA-40) temperatures on the Tibetan Plateau. *Journal of Geophysical Research: Atmospheres (1984–2012)*, 110(D2).

Fucile, E., Kertész, S., Lamy-Thépaut, S., and Najm, S. (2016). ECMWF's new data decoding software ecCodes. *ECMWF*.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software Addison-Wesley. *Reading, MA*, page 1995.

Ghil, M. and Malanotte-Rizzoli, P. (1991). Data assimilation in meteorology and oceanography. *Adv. Geophys*, 33:141–266.

Gibson, J. K., Kallberg, P., Uppala, S., Hernandez, A., Nomura, A., and Serrano, E. (1997). ERA description. ECMWF Re-Analysis Project Report Series 1, ECMWF. *Reading, UK*, page 77.

Golub, G. H. and Reinsch, C. (1971). Singular value decomposition and least squares solutions. In *Linear Algebra*, pages 134–151. Springer.

Gormley, C. and Tong, Z. (2015). *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine*. " O'Reilly Media, Inc.".

Grainger, T. and Potter, T. (2014). *Solr in Action*. Manning Publications Co.

Grenier, P., Parent, A.-C., Huard, D., Anctil, F., and Chaumont, D. (2013). An assessment of six dissimilarity metrics for climate analogs. *Journal of Applied Meteorology and Climatology*, 52(4):733–752.

Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media, Inc.

Gupta, M. R. and Jacobson, N. P. (2006). Wavelet principal component analysis and its application to hyperspectral images. In *2006 International Conference on Image Processing*, pages 1585–1588. IEEE.

Hamming, R. W. (1986). *Coding and Information Theory*. Prentice-Hall, Inc.

Hatcher, D. A. (1984). Simple formulae for Julian day numbers and calendar dates. *Quarterly Journal of the Royal Astronomical Society*, 25:53.

Hersbach, H., Bell, B., Berrisford, P., Horányi, A., Joaquí, M. S., Nicolas, J., Radu, R., Schepers, D., Simmons, A., and Soci, C. (2019). Global reanalysis: Goodbye ERA-Interim, hello ERA5. *ECMWF Newsl*, 159:17–24.

Hersbach, H. and Dee, D. (2016). ERA5 reanalysis is in production. *ECMWF newsletter*, 147(7):5–6.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Hungershöfer, J. and Wierum, J.-M. (2002). On the quality of partitions based on space-filling curves. In *International Conference on Computational Science*, pages 36–45. Springer.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in science and engineering*, 9(3):90–95.

Indyk, P. (2000). Dimensionality reduction techniques for proximity problems. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 371–378. Society for Industrial and Applied Mathematics.

Indyk, P. and Naor, A. (2007). Nearest-neighbor-preserving embeddings. *ACM Transactions on Algorithms (TALG)*, 3(3):31.

Jacobs, C. E., Finkelstein, A., and Salesin, D. H. (1995). Fast multiresolution image querying. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 277–286. ACM.

Jawerth, B. and Sweldens, W. (1994). An overview of wavelet based multiresolution analyses. *SIAM review*, 36(3):377–412.

Jones, E., Oliphant, T., and Peterson, P. (2014). SciPy: Open source scientific tools for Python.

Keinert, F. (2003). *Wavelets and Multiwavelets*. Chapman and Hall/CRC, 1st edition.

Kelley, C. T. (2003). *Solving Nonlinear Equations with Newton's Method*, volume 1. Siam.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., and Corlay, S. (2016). Jupyter Notebooks-a publishing format for reproducible computational workflows. In *ELPUB*, pages 87–90.

Lance, G. N. and Williams, W. T. (1967). A General Theory of Classificatory Sorting Strategies: 1. Hierarchical Systems. *The Computer Journal*, 9(4):373–380.

Lawder, J. K. and King, P. J. (2000). Using space-filling curves for multi-dimensional indexing. In *Advances in Databases*, pages 20–35. Springer.

Leskovec, J., Rajaraman, A., and Ullman, J. D. (2014). *Mining of Massive Datasets*. Cambridge university press.

Liao, S., Lopez, M. A., and Leutenegger, S. T. (2001). High dimensional similarity search with space filling curves. In *Proceedings 17th International Conference on Data Engineering*, pages 615–622. IEEE.

Lorenz, E. N. (1969). Atmospheric predictability as revealed by naturally occurring analogues. *Journal of the Atmospheric sciences*, 26(4):636–646.

Maciel, P., Quintino, T., Modigliani, U., Dando, P., Raoult, B., Deconinck, W., Rathgeber, F., and Simarro, C. (2017). The new ECMWF interpolation package MIR.

Mallat, S. G. (1989). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):674–693.

Marimont, R. and Shapiro, M. (1979). Nearest neighbour searches and the curse of dimensionality. *IMA Journal of Applied Mathematics*, 24(1):59–70.

Marsolo, K., Parthasarathy, S., and Ramamohanarao, K. (2006). Structure-based querying of proteins using wavelets. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 24–33. ACM.

McNames, J. (2001). A fast nearest-neighbor algorithm based on a principal axis search tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):964–976.

Mo, R., Ye, C., and Whitfield, P. H. (2014). Application Potential of Four Nontraditional Similarity Metrics in Hydrometeorology. *Journal of Hydrometeorology*, 15(5):1862–1880.

Molteni, F., Buizza, R., Palmer, T. N., and Petroliagis, T. (1996). The ECMWF ensemble prediction system: Methodology and validation. *Quarterly Journal of the Royal Meteorological Society*, 122(529):73–119.

Müllner, D. (2011). Modern hierarchical, agglomerative clustering algorithms. http://arxiv.org/abs/1109.2378.

Nielsen, J. (2009). Powers of 10: Time scales in user experience. https://www.nngroup.com/articles/powers-of-10-time-scales-in-ux/.

Oliphant, T. E. (2007). Python for Scientific Computing. *Computing in Science & Engineering*, 9(3):10–20.

O'Sullivan, P. (1993). MAGICS - The ECMWF graphics package. *ECMWF Newsletter*, (62).

Owens, M. (2006). *The Definitive Guide to SQLite*. Apress.

Papathomas, T. V., Schiavone, J. A., and Julesz, B. (1988). Applications of computer graphics to the visualization of meteorological data. In *ACM SIGGRAPH Computer Graphics*, volume 22, pages 327–334. ACM.

Patrikalakis, N. M., Engineering, M. I. o. T. D. o. M., and Engineering, M. I. o. T. D. o. M. (2006). *Wavelet Based Similarity Measurement Algorithm for Seafloor Morphology*. Massachusetts Institute of Technology.

Pauly, O., Padoy, N., Poppert, H., Esposito, L., and Navab, N. (2009). Wavelet energy map: A robust support for multi-modal registration of medical images. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference On*, pages 2184–2191. IEEE.

Pearson, K. (1901). LIII. 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(11):559–572.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Pérez, F. and Granger, B. E. (2007). IPython: A system for interactive scientific computing. *Computing in Science & Engineering*, 9(3):21–29.

Raoult, B. (1997). Architecture of the New MARS Server. In *Sixth Workshop on Meteorological Operational Systems, 17-21 November 1997*, pages 90–100, Shinfield Park, Reading. ECMWF.

Raoult, B. (2002). MARS on the Web. In *Ams.Confex.Com*.

Raoult, B., Bergeron, C., López Alós, A., Thépaut, J.-N., and Dee, D. (2017). Climate service develops user-friendly data store. *ECMWF*.

Raoult, B., Daabeck, J., Norris, B., and Câmara, G. (1995). Data manipulation and visualisation at ECMWF using Metview/ws macro language 11th International Conference on Interactive Information and Processing Systems for Meteorology. In *Oceanography and Hidrology (IIPS), AMS*.

Raoult, B., Di Fatta, G., Pappenberger, F., and Lawrence, B. (2018). Fast Retrieval of Weather Analogues in a Multi-petabytes Archive Using Wavelet-Based Fingerprints. In *International Conference on Computational Science*, pages 697–710. Springer.

Rathna, B. (2011). Hamming distance fn for postgresql. https://github.com/deepfryed/pghammer.

Reese, W. (2008). Nginx: The high-performance web server and reverse proxy. *Linux Journal*, 2008(173):2.

Regentova, E., Latifi, S., and Deng, S. (2000). A wavelet-based technique for image similarity estimation. In *ITCC-00*, pages 207–212. IEEE.

Ricker, N. (1953). The form and laws of propagation of seismic wavelets. *Geophysics*, 18(1):10–40.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Ruth, D. P. (1993). The interactive modification of gridded forecasts. In *Preprints Ninth International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography and Hydrology*, pages 327–332. Citeseer.

Santer, B. D., Wigley, T. M., Simmons, A. J., Kållberg, P. W., Kelly, G. A., Uppala, S. M., Ammann, C., Boyle, J. S., Brüggemann, W., and Doutriaux, C. (2004). Identification of anthropogenic climate change using a second-generation reanalysis. *Journal of Geophysical Research: Atmospheres (1984–2012)*, 109(D21).

Satopaa, V., Albrecht, J., Irwin, D., and Raghavan, B. (2011). Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior. In *2011 31st International Conference on Distributed Computing Systems Workshops*, pages 166–171, Minneapolis, MN, USA. IEEE.

Shapiro, J. M. (1993). Embedded image coding using zerotrees of wavelet coefficients. *Signal Processing, IEEE Transactions on*, 41(12):3445–3462.

Slaney, M. and Casey, M. (2008). Locality-Sensitive Hashing for Finding Nearest Neighbors [Lecture Notes]. *IEEE Signal Processing Magazine*, 25(2):128–131.

Smith, A. E., Coit, D. W., Baeck, T., Fogel, D., and Michalewicz, Z. (2000). Penalty functions. *Evolutionary computation*, 2:41–48.

Sorzano, C. O. S., Vargas, J., and Montano, A. P. (2014). A survey of dimensionality reduction techniques. *arXiv preprint arXiv:1403.2877*.

Stollnitz, E. J., DeRose, T., and Salesin, D. H. (1996). Wavelets for computer graphics - theory and applications. *Morgan Kaufmann 1996*.

Stollnitz, E. J., DeRose, T. D., and Salesin, D. H. (1995a). Wavelets for Computer Graphics: A Primer Part 1. pages 1–8.

Stollnitz, E. J., DeRose, T. D., and Salesin, D. H. (1995b). Wavelets for Computer Graphics: A Primer Part 2. pages 1–9.

Taylor, K. E., Stouffer, R. J., and Meehl, G. A. (2012). An overview of CMIP5 and the experiment design. *Bulletin of the American Meteorological Society*, 93(4):485–498.

Thirey, B. and Hickman, R. (2015). Distribution of Euclidean Distances Between Randomly Distributed Gaussian Points in n-Space. *arXiv:1508.02238 [math]*.

Thorndike, R. L. (1953). Who belongs in the family? *Psychometrika*, 18(4):267–276.

Traina, A. J. M., Castañón, C. A. B., Traina, C., and Jr (2003). *MultiWaveMed: A System for Medical Image Retrieval through Wavelets Transformations*. IEEE Computer Society.

Uppala, S. M., Kållberg, P. W., Simmons, A. J., Andrae, U., Bechtold, V. D. C., Fiorino, M., Gibson, J. K., Haseler, J., Hernandez, A., and Kelly, G. A. (2005). The ERA-40 re-analysis. *Quarterly Journal of the royal meteorological society*, 131(612):2961–3012.

Usevitch, B. (Sept./2001). A tutorial on modern lossy wavelet image compression: Foundations of JPEG 2000. *IEEE Signal Processing Magazine*, 18(5):22–35.

uWSGI (2019). uWSGI Documentation. https://uwsgi-docs.readthedocs.io/.

Van den Dool, H. M. (1989). A new look at weather forecasting through analogues. *Monthly weather review*, 117(10):2230–2247.

Van den Dool, H. M. (1994). Searching for analogues, how long must we wait? *Tellus A*, 46(3):314–324.

Van Der Maaten, L., Postma, E., and Van den Herik, J. (2009). Dimensionality reduction: A comparative. *J Mach Learn Res*, 10(66-71):13.

Van Der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.

Vaněček Jr, G. (1991). Brep-index: A multidimensional space partitioning tree. *International Journal of Computational Geometry & Applications*, 1(03):243–261.

Walker, J. S. (2005). A Primer on Wavelets and their Scientific Applications. pages 1–156.

Wang, A. (2003). An Industrial Strength Audio Search Algorithm. In *ISMIR 2003, 4th International Conference on Music Information Retrieval, Baltimore, Maryland, USA, October 27-30, 2003, Proceedings*.

Wang, J. and Shan, J. (2005). Space filling curve based point clouds index. In *Proceedings of the 8th International Conference on GeoComputation*, pages 551–562.

Wasilewski, F. (2010). Pywavelets: Discrete wavelet transform in python.

Wellman, D. (2009). *jQuery UI 1.6: The User Interface Library for jQuery*. Packt Publishing Ltd.

Wessels, A., Purvis, M., Jackson, J., and Rahman, S. (2011). Remote Data Visualization through WebSockets. In *2011 Eighth International Conference on Information Technology: New Generations*, pages 1050–1051, Las Vegas, NV. IEEE.

Williams, D. N., Taylor, K. E., Cinquini, L., Evans, B., Kawamiya, M., Lautenschlager, M., and Lawrence, B. N. (2011). The Earth System Grid Federation: Software framework supporting CMIP5 data analysis and dissemination. *CLIVAR Exchanges*, 16(2):40–42.

Woods, A. (2006). Archives and Graphics: Towards MARS, MAGICS and Metview. *Medium-Range Weather Prediction: The European Approach*, pages 183–193.

World Meteorological Organization (2009). *Manual on Codes. Volumes I.2 & 1.2 Volumes I.2 & 1.2*. Secretariat of the World Meteorological Organization, Geneva, Switzerland. OCLC: 607526193.

Zorita, E. and Von Storch, H. (1999). The analog method as a simple statistical downscaling technique: Comparison with more complicated methods. *Journal of climate*, 12(8):2474–2489.