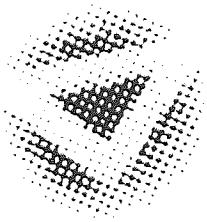
A pragmatic Bayesian approach to wind field retrieval

David J. Evans

Doctor Of Philosophy



The University of Aston in Birmingham

August 2001

THE UNIVERSITY OF ASTON IN BIRMINGHAM

A pragmatic Bayesian approach to wind field retrieval

DAVID J. EVANS

Doctor Of Philosophy, 2001

Thesis Summary

The ERS-1 Satellite was launched in July 1991 by the European Space Agency into a polar orbit at about 800 *km*, carrying a C-band scatterometer. A scatterometer measures the amount of backscatter microwave radiation reflected by small ripples on the ocean surface induced by sea–surface winds, and so provides instantaneous snap–shots of wind flow over large areas of the ocean surface, known as wind fields. Inherent in the physics of the observation process is an ambiguity in wind direction; the scatterometer cannot distinguish if the wind is blowing toward or away from the sensor device. This ambiguity implies that there is a one–to–many mapping between scatterometer data and wind direction. Current operational methods for wind field retrieval are based on the retrieval of wind vectors from satellite scatterometer data, followed by a disambiguation and filtering process that is reliant on numerical weather prediction models. The wind vectors are retrieved by the local inversion of a forward model, mapping scatterometer observations to wind vectors, and minimising a cost function in scatterometer measurement space.

This thesis applies a pragmatic Bayesian solution to the problem. The likelihood is a combination of conditional probability distributions for the local wind vectors given the scatterometer data. The prior distribution is a vector Gaussian process that provides the geophysical consistency for the wind field.

The wind vectors are retrieved directly from the scatterometer data by using mixture density networks, a principled method to model multi–modal conditional probability density functions. The complexity of the mapping and the structure of the conditional probability density function are investigated. A hybrid mixture density network, that incorporates the knowledge that the conditional probability distribution of the observation process is predominantly bi–modal, is developed. The optimal model, which generalises across a swathe of scatterometer readings, is better on key performance measures than the current operational model.

Wind field retrieval is approached from three perspectives. The first is a non-autonomous method that confirms the validity of the model by retrieving the correct wind field 99% of the time from a test set of 575 wind fields. The second technique takes the maximum *a posteriori* probability wind field retrieved from the posterior distribution as the prediction. For the third technique, Markov Chain Monte Carlo (MCMC) techniques were employed to estimate the mass associated with significant modes of the posterior distribution, and make predictions based on the mode with the greatest mass associated with it. General methods for sampling from multi–modal distributions were benchmarked against a specific MCMC transition kernel designed for this problem. It was shown that the general methods were unsuitable for this application due to computational expense. On a test set of 100 wind fields the MAP estimate correctly retrieved 72 wind fields, whilst the sampling method correctly retrieved 73 wind fields.

Keywords: Wind vector retrieval, Wind field retrieval, mixture density networks, pragmatic Baysian model

To Anna-Maria, Imogen,
Di, Bri,
and my parents.

Acknowledgements

I have been very fortunate to have two supervisors whilst studying for this PhD. Dr. Dan Cornford was my supervisor for the first year of this thesis. Dr. Ian Nabney was my supervisor for the second and third year whilst Dan took the role of my associate supervisor. I thank you both for enthusiastically welcoming and carefully listening to all of my ideas, offering sincere and profound advice, and teaching me the worthy qualities of a diligent and conscientious researcher.

I am indebted to Dr. Chris Williams, who along with Dr. Ian Nabney, gave me a chance to study at the Neural Computing Research Group (NCRG).

I thank the staff at NCRG, especially Prof. David Lowe, Prof. David Saad, Dr. David Barber and Dr. Chris Williams, who have created a wonderful learning environment which behaves just like osmosis - knowledge is always flowing from regions of high concentration to regions of low concentration.

I am grateful to my fellow students, Lars Hjorth, Guillaume Ramage, Earnest Fokoué, Huma Syed, Lehel Csató and Wei Lee Woon for their stimulating discussions during this project. I am further grateful to Lars for allowing me to use Figure 3.2 which he patiently generated using LaTeX.

My sincere thanks to Engineering and Physical Science Research Council for paying my tuition fees and awarding me a maintenance grant that has supported me whilst studying for this thesis.

I am very grateful to Vicky Stoddart who taught me on a weekly basis, over a two year period, the ins and outs of mind maps, short vowel sounds, some grammar and the function of the paragraph!

Sensei Mooney who has been a rock – I thank you for talking from your heart, telling it as it is and your resolute commitment to Aikido.

A big 'thank you' to Di and Bri who have given much support to Anna–Maria and myself, including superb hospitality and location at Chéz Landon (IoW) for our summer holidays and short breaks during this period and before.

Finally, I thank Anna-Maria, who has been my unwaivering support throughout my study and writing up; I have no words to express my gratitude – Annie you are a very special woman.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Reference to weather forecasts is part of our every day lives. For instance, they are used by individual members of the public, large corporations and governments to help plan activities and make decisions – it is going rain at the seaside today, we will stay indoors; there is a large storm in the Atlantic so we should re-route our shipping; there are extreme weather conditions approaching so alert the emergency services.

Many forecasts are based on the output of Numerical Weather Prediction (NWP) models which are a representation of the atmosphere by a computer model. The inputs to these models are observations taken from the earth's atmosphere, including those collected by weather satellites. Satellite observations include satellite imagery for television weather forecasts, infrared wavelengths for prediction of temperature at ground level and above and, active sensing for the prediction of wind speed and direction. Data collected from weather satellites provide opportunities to make fast, accurate weather predictions on a global scale. Improving the model and the assimilation of data means that more accurate forecasts can be produced.

This thesis is concerned with the processing of satellite data collected by active sensing devices, otherwise known as scatterometer data, for the inference of wind speed and direction close to the ocean surface on a global scale. The model used is centred on a pragmatic Bayesian philosophy.

In this chapter we introduce the main considerations of this thesis: the real world geophysical problem of wind vector inference from remotely sensed data; the application of a principled Bayesian framework to this real world problem; the development of methodologies to retrieve wind fields from such a Bayesian model.

1.1 The geophysical problem

Numerical Weather Prediction models are tools used by oceanographers, meteorologists, geophysicists and many other scientists to forecast the future state of the atmosphere. Initial conditions for the

NWP model are important, as they are essential for accurate predictions. The initial condition of the NWP model is the current state of the atmosphere described by the parameters of the model including wind speed and wind direction, otherwise known as wind vectors. The wind vectors are inferred, by an inverse model, from scatterometer data collected from space borne satellites. Such scatterometers provide fast and accurate global coverage of the world's oceans, providing an up to date 'picture' of the current state of the winds close to the ocean surface.

The wind vectors are dependent on the quality of the model used to infer the model parameters, which depends on the quality of the measurements collected by the space borne scatterometers. Therefore, new or improved methods for solving the inverse problem are of interest to a wide range of scientists and increase the quality of weather forecasts.

Depending on the context, wind vectors may be defined either by polar coordinates (s, χ) or by Cartesian coordinates (u, v). For the purpose of this thesis, a wind vector describes the estimated mean wind speed and direction of an area of ocean, called a *cell*, that is approximately 50 km by 50 km and equal to the size of the footprint that the satellite radar beam makes on the ocean surface; we call a wind vector in a cell the *local* wind vector. Each local wind vector is associated with a longitude and latitude co–ordinate and therefore has a fixed spatial location on the earth's surface. To study the flow of wind over the ocean surface we choose a grid of suitable longitude and latitude co–ordinates and plot the corresponding wind vectors at those co–ordinates. This grid of local wind vectors is called a *wind field* and the estimation of the wind field represents a *global* problem. Hence, in this thesis we investigate local solutions to wind vector retrieval on a cell by cell basis and global solutions to wind field retrieval where the local models are fused into a spatial context. An example wind field is shown in Figure 1.1, where one hundred local wind vectors organised on a ten by ten grid.

However local wind vector retrieval is not straight forward. Due to the physics of the the data collection system there is an inherent ambiguity in wind direction. This means that the mapping from satellite data to wind direction is one–to–many and it is difficult to autonomously resolve which of the solutions is correct. Generally there are two solutions, but sometimes three or four solutions may exist.

Because of the ambiguity in the local problem there also exists an ambiguity in the global problem; the global solution is generally multi–modal – there can be several geophysically plausible wind fields produced from a global model and the problem is to choose the wind field which describes the current state of wind flow. Retrieving the correct wind field is called disambiguation. Non-autonomous disambiguation involves reference to other data sources to indicate the 'correct' solution, whilst autonomous ambiguity removal uses only the satellite data to make wind field predictions. In Figure 1.2, four possible wind fields are shown which have been retrieved from a wind field model which had inputs of the satellite data corresponding to the forecast wind field in Figure 1.1. It is easy to choose the correct solution when given the reference wind field; but how is the correct wind field chosen without such a reference?

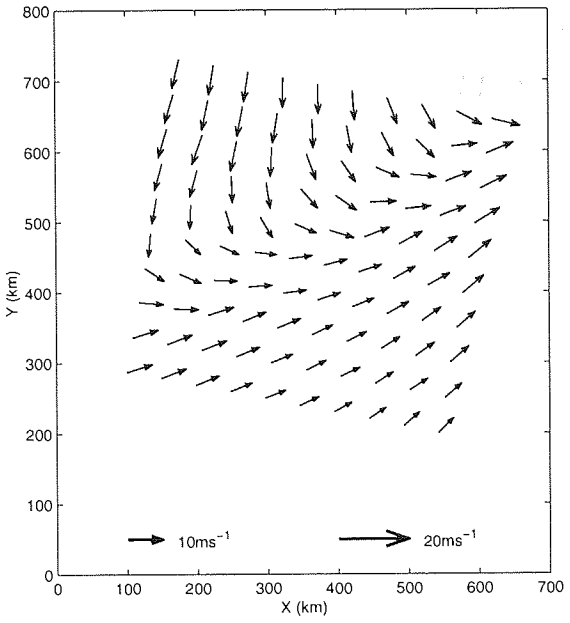


Figure 1.1: An example wind field. Each vector represents an estimated local wind vector inferred from scatterometer data. The local wind vectors are plotted on a longitude–latitude grid, giving a global perspective of the wind speed and direction over large areas of the ocean surface. In this example the grid is approximately 400 *km* by 600 *km*. This kind of coverage is only possible by assimilating data collected by satellite coverage.

Current best practice for wind field retrieval includes a complicated inversion of a forward model[1] to infer local wind vectors, followed by a disambiguation process which involves comparison of the inferred wind vectors with last estimated wind field from the NWP model; this process is non-autonomous. NWP wind fields are expensive to generate[2] as only a few institutes, such as the UK Meteorological Office, have adequate computing power to run models that generate the NWP winds. It is therefore in the interest of the community to have models which can autonomously assimilate remotely sensed data and run on desktop computers.

These problems have been recognised within the research community and a European project was established to investigate novel ways of solving such problems using neural network methodologies: the project was called 'NEUROSAT'. This project was shared between several European research centres including Aston University.

1.2 The NEUROSAT project at the Neural Computing Research Group (NCRG)

The NEUROSAT project, funded by the European Commission as part of the Fourth Framework, was concerned with applying neural network approaches to problems in satellite remote sensing to extract

[1] In geophysical terms a forward model maps wind vectors to the remotely sensed data; an inverse model maps the remotely sensed data to the model parameters.

[2] An NWP model will typically have 10^6 parameters of which the wind vectors are a subset.

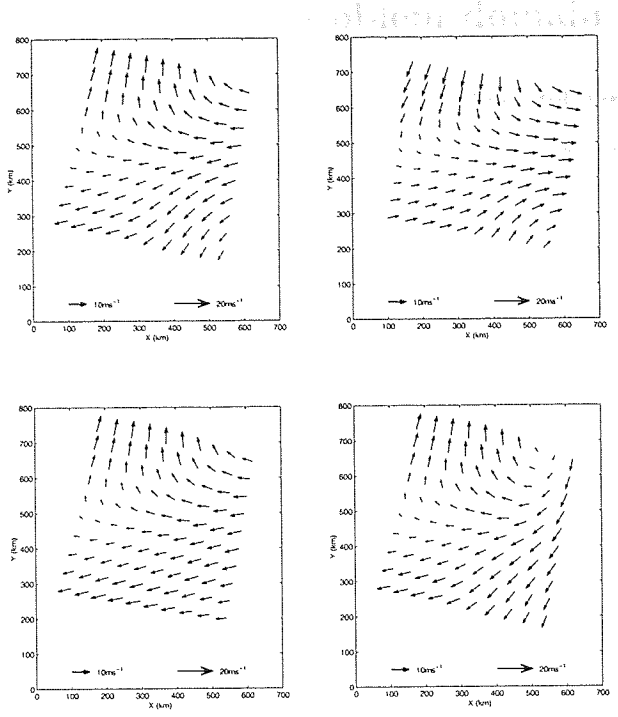


Figure 1.2: Due to the ambiguity in the local model there are many plausible modes in the global solution. The plots in this figure represent possible wind fields generated from the satellite data collected for the wind field Figure 1.1; the question is – which is the correct one? The disambiguation problem requires us to choose the appropriate wind field given the satellite data.

wind vectors, (u, v), from satellite scatterometer measurements, $\boldsymbol{\sigma}^o$, taken over the ocean. At Aston, the research was focused on a probabilistic interpretation of the problem. There were three distinct areas of research, although the boundaries are not distinct:

- Solving the forward model, the mapping of $(u, v) \rightarrow \boldsymbol{\sigma}^o$, by building the probabilistic model $p(\boldsymbol{\sigma}^o \mid u, v)$. As the mapping is single valued conventional neural networks were used.

- Solving the inverse model, the mapping of $\boldsymbol{\sigma}^o \rightarrow (u, v)$, by building the probabilistic model $p(u, v \mid \boldsymbol{\sigma}^o)$. Because the mapping is multi–valued conventional neural networks fail. Instead, Mixture Density Networks are employed to model the multi–valued conditional probability densities.

- Ambiguity removal by both autonomous and non-autonomous methods. In this project this implies inferring the correct wind field from the global model, given the scatterometer data. Both heuristic and Bayesian methods are applied to this problem.

The project was run by Ian Nabney. Within the scope of this project, Dan Cornford, who filled a postdoctoral position and supervised two MSc projects, jointly supervised this thesis with Ian Nabney. A technical report giving further information on the NEUROSAT project at Aston (Cornford and Nabney, 1998) is available from `http://www.ncrg.aston.ac.uk/Papers/`.

1.3 Motivation: defining the problem domain

In modelling the wind field retrieval problem a pragmatic Bayesian approach is taken. The global model solution is arrived at by combining local models with a global prior model,

$$p(U, V \mid \boldsymbol{\Sigma}^o) \propto p(\boldsymbol{\Sigma}^o | U, V) p(U, V). \tag{1.1}$$

In this context uppercase symbols describe vectors and matrices of a wind field; lower case symbols indicate parameters at a local level. The likelihood of the global wind field model is decomposed into a product of local forward models, where we assume that the local models are conditionally independent as the wind speed and direction of a given cell completely determine the observed $\boldsymbol{\sigma}^o$ values, and thus conditionally on the wind vectors in each cell the $\boldsymbol{\sigma}^o$ are independent (Nabney *et al.*, 2000),

$$p(\boldsymbol{\Sigma}^o \mid U, V) = \prod_i p(\boldsymbol{\sigma}_i^o \mid u_i, v_i). \tag{1.2}$$

Using Bayes' theorem the local forward models are expressed in terms of the local inverse model:

$$p(\boldsymbol{\sigma}_i^o \mid u_i, v_i) = \frac{p(u_i, v_i \mid \boldsymbol{\sigma}_i^o) p(\boldsymbol{\sigma}_i^o)}{p(u_i, v_i)}. \tag{1.3}$$

We are interested in the inference of (U, V), and once we have observed the data $\boldsymbol{\Sigma}^o$, the factor $p(\boldsymbol{\sigma}_i^o)$ in (1.3) is fixed and need not be further considered (Williams, 1997). We then consider the scaled likelihood for location i as:

$$L_i(\boldsymbol{\sigma}_i^o) = \frac{p(u_i, v_i \mid \boldsymbol{\sigma}_i^o)}{p(u_i, v_i)}. \tag{1.4}$$

This is why the word *pragmatic* appears in the title. In order for us to develop the model via a direct inversion method it is necessary to understand that once observed the observations are fixed and have no further influence on the inference of variables (U, V). Therefore we can write

$$p(\boldsymbol{\sigma}_i^o \mid u_i, v_i) \propto \frac{p(u_i, v_i \mid \boldsymbol{\sigma}_i^o)}{p(u_i, v_i)}, \tag{1.5}$$

and re-writing 1.1 using 1.5 we obtain:

$$p(U, V \mid \boldsymbol{\Sigma}^o) \propto \left(\prod_i \frac{p(u_i, v_i \mid \boldsymbol{\sigma}_i^o)}{p(u_i, v_i)} \right) p(U, V), \tag{1.6}$$

which is the global wind field model expressed in terms of the local inverse model (Nabney *et al.*, 2000).

Therefore wind fields are retrieved by the following process: train the local models; train the prior model; combine the local and prior models (Cornford, 1997; Cornford, 1998a) in the Bayesian framework and infer the wind fields.

The performance of the model is dependent on the quality of the global wind prior, and the inverse model $p(u_i, v_i \mid \boldsymbol{\sigma}_i^o)$ or the forward model $p(\boldsymbol{\sigma}_i^o \mid u_i, v_i)$. This thesis develops the local inverse model[3], $p(u_i, v_i \mid \boldsymbol{\sigma}_i^o)$, and the subsequent analysis of the global model, (1.6), for the retrieval of wind fields.

[3]The local forward model was developed by an MSc student called Guillaume Ramage (Ramage, 1998).

1.4 Thesis overview

This thesis contributes to the NEUROSAT project and reports several new areas of research: the investigation and implementation of a probabilistic direct inverse model which maps $\sigma^o \rightarrow (u, v)$ using general and hybrid Mixture Density Network frameworks, and, the investigation of methods for autonomous wind field retrieval from the global model, (1.6), by using Markov Chain Monte Carlo sampling techniques.

Within this thesis there is a balance between theory and application. It is necessary to present supporting evidence for model performance on a local and global scale, but at the same time these results must not interfere with the flow of text. It is for this reason that most tables of model performance results are included in the appendices where they can readily be viewed but do not interfere with the main thesis body.

Thesis content

This chapter provides the introduction to the work contained in this thesis, and puts the work of this thesis into context from an application perspective.

Chapter two is a short chapter to provide information on the data acquisition and pre-processing. Here the origins of the data sets used in the experiments of this thesis are described.

Chapter three provides the background to the problem of inferring the local wind vectors from remotely sensed scatterometer data; there is a literature survey of relevant work within the field which highlights where the work in this thesis fits into the current research landscape. The theory of the general Mixture Density Network is given along with a hybrid Mixture Density Network, which has been developed to encapsulate the known physics of the data acquisition system.

Chapter four describes the experimental details for the training of the local models. Exploratory experiments established that the Mixture Density Network provides a principled approach to modelling the inverse mapping. Further models are developed using a larger, second data set. Finally, a local model is chosen to be used in the next two chapters for inclusion in the global wind field model.

Chapter five explores wind field retrieval from the global model by integrating the local model trained in Chapter 4, with the global prior model developed by Dan Cornford. Non-autonomous retrieval techniques are explored to validate the model, followed by a first attempt at autonomous wind field retrieval. This method is implemented by retrieving the wind fields associated with the mode with that has the maximum *a posteriori* probability density of the posterior distribution.

Chapter six further develops autonomous wind field retrieval techniques. The theoretical and practical implications of sampling, using Markov Chain Monte Carlo techniques (MCMC), from a

probability distribution which is multi–modal and modes well separated are investigated. An algorithm is developed specifically for the wind field problem and applied.

Chapter seven concludes the thesis. There is a summary of significant results and suggestions for future research.

The content of this thesis represents original research. The work within has not previously appeared elsewhere with the exception of those research papers produced during the normal course of its preparation. Those research papers were:

- *Structured neural network modelling of multi-valued functions for wind vector retrieval from satellite scatterometer measurements*, Neurocomputing Letters (**30**) 23 – 30, 2000 (Evans, Cornford, and Nabney, 2000). Based on the work of Chapters 3 and 4.
- *Wind Field Retrieval From Scatterometer Data*, Eighth International Meeting on Statistical Climatology, University of Luneburg, 12–16 March 2001 (Cornford, Evans, and Nabney, 2001). Based on the work of Chapter 6.
- *Efficient sampling from high dimension, multi–modal probability distributions: a spatial application*, First European Conference on Spatial and Computational Statistics, Lancaster University, Ambleside, 17–21 September 2000 (Evans, Nabney, and Cornford, 2000). Based on the work of Chapter 6.
- *Bayesian Retrieval of Scatterometer Wind Fields*, 1999, technical report NCRG/99/015, Aston University, Neural Computing Research Group (Cornford, Nabney, and Evans, 1999). Based on the work of Chapters 4 and 5.
- *Mixture Density Network Training by Computation in Parameter Space*, 1998, technical report NCRG/98/016, Aston University, Neural Computing Research Group (Evans, 1998). Based on the work of Chapter 4.

A further paper is in preparation, based on the results of Chapter 4, titled: *A neural network approach to direct inversion of satellite scatterometer data for local wind vector retrieval*, for submission to International Journal of Remote Sensing with D. Cornford and I.T. Nabney.

1.5 Chapter review

This chapter has introduced the theme of this thesis. The specific application has been described along with how this thesis is tied in with the research at Aston and the NEUROSAT project. The research has been motivated by the challenge of developing algorithms which process remotely sensed scatterometer data on desk top computers, using principled Bayesian techniques.

In the next chapter, the data acquisition, the data pre-processing and the formulation of the data sets is described.

Chapter 2

Data acquisition and pre-processing

Key word summary

Satellite	**Acquisition**
Pre-filtering	**Pre-processing**
Wind	**Scatterometer**
ERS-1	**ERS-2**

2.1 Introduction

The data used in this thesis was obtained from real world satellites orbiting the earth. In this chapter the acquisition, filtering and pre-processing of the data for the training of the mixture density networks are described.

The data was sourced from two European Remote Sensing Satellites, ERS-1 and ERS-2. For the first year of this project only the ERS-1 data was available, while the ERS-2 satellite data was not available until after the beginning of the second year of this project.

2.2 Data acquisition

The ERS-1 Satellite was launched in July 1991 by the European Space Agency into a polar orbit at about 800 km, carrying a C-band scatterometer. The scatterometer has a microwave radar operating at 5.3 GHz, and measures the amount of backscatter generated by small ripples, of about 5 cm wavelength, on the ocean surface (Robinson, 1985). The small ripples are caused by the wind flowing over the ocean surface; the amplitude of the ripples is proportional to the wind speed.

The scatterometer has three independent antenna which point, on a horizontal plane, in three directions, 45°, 90°, 135°, with respect to the satellite propagation and are referred to as fore, mid and aft beam antennae respectively. The satellite samples a swathe of ocean surface approximately

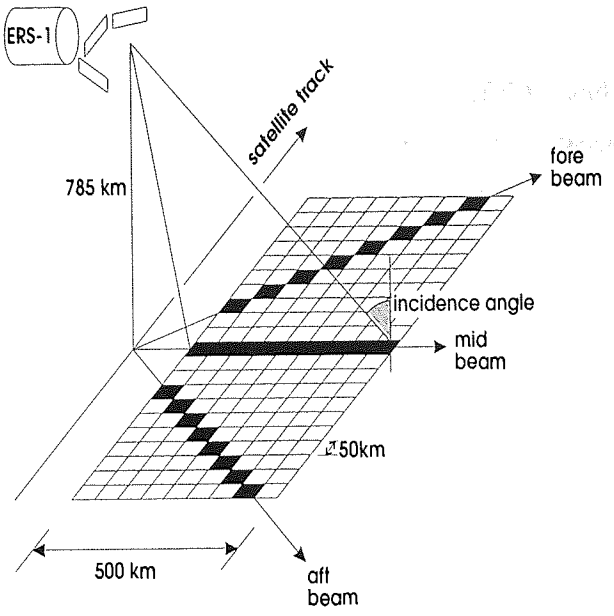


Figure 2.1: Relationship of the ERS-1 satellite antennas and the ocean surface. For simplicity, only nine non-overlapping satellite tracks are shown.

500 km by 500 km. This swathe is divided into nineteen tracks, where each track is approximately 25 km wide. Each measurement cell is approximately 50 km by 50 km and so there is some overlap between adjacent tracks.

Each track is identified by the incidence angle of the mid beam with respect to the local vertical, which varies from 18° to 45°, and is numbered from 1 to 19 respectively (see Figure 2.1). The odd numbered tracks are referred to as traces, which are identified as trace 0 to trace 9 where trace 0 is the innermost trace with respect to the satellite (has the smallest incidence angle). As the satellite passes over the ocean surface, each cell is illuminated by the footprint of each antenna, fore, mid and aft beam respectively and a measurement vector for each cell is collected, $(\sigma_{\mathrm{f}}^{o}, \sigma_{\mathrm{m}}^{o}, \sigma_{\mathrm{a}}^{o})$. This is referred to as the normalised radar cross section, denoted by $\boldsymbol{\sigma}^{o}$, and has units of decibels.

Due to the success of the ERS-1 satellite[1], a second satellite, ERS-2, was launched in April 1995 to replace the ERS-1 satellite. The ERS-1 satellite finally ceased operation in 1999. The ERS-2 carries similar measuring devices to that of ERS-1, and allowed for further scatterometer data to be collected and used in projects such as this.

Before investigating the use of neural networks for the direct inversion of the scatterometer, the data collected from the satellites was pre-processed to create training, validation and test data sets. A data set comprises of pairs of input and target patterns.

[1]For further information on the ERS-1 and ERS-2 satellites see the European Space Agency's web site at: `http://www.esa.int/export/esaSA/GGGWBR8RVDC_earth_0.html`

2.2.1 Data description

The input data, σ^o, is supplied by the European Space Agency (ESA), and is labelled by longitude, latitude and time. The data is then processed at IFREMER (Institut Français de la Recherche pour l'Exploitation de la Mer) where quality control is applied (including sea ice filter to remove observations taken over sea ice) to remove low quality observations. The target data is computed from a European Centre for Medium–Range Weather Forecasts (ECMWF) forecast by taking a 250 km by 250 km grid of $10m$ model winds and interpolating in space and time to the satellite observation position (IFREMER, 1996).

Philippe Richaume (working as part of the NEUROSAT team) selected the ERS-1 data from the North Atlantic regions (1995-96). It is not clear how strict the quality control process was to remove outliers or measurements that are spatially correlated.

Dan Cornford selected the observations from the ERS-2 Northern Hemisphere regions for 1997–1999. The data was selected to avoid any measurements that were near fronts and to ensure the observations were sufficiently spatially separate as to be uncorrelated. Furthermore, with the aid of a visualisation tool, inspection of the scatterometer data was also undertaken to remove scatterometer readings with excessive noise on the observations (Cornford *et al.*, 2001).

With respect to this project, for the first year the ERS-1 data sets were available, and then for the second and third year data was also available from the ERS-2 satellite. The ERS-1 data sets were used for the initial experiments; the results of these experiments established that direct inversion using mixture density networks was a plausible solution to the problem. The ERS-2 data sets, generated under the control of Dan Cornford, enabled further enhancement of the models from those developed using ERS-1 data, and showed how the quality of the training data sets can affect the quality of the trained model. Specifically, the data sets used were as follows:

- Year 1: Experiments using data generated from ERS-1. Initial experiments were carried out by training models for each of the ten traces in the satellite swathe. There were training, test and validation sets for each trace, where each training set contained 3,000 examples and each test and validation sets contained 1,000 examples. These data sets will be referred to as YR1TRACE.

- Year 1: A second data set generated from ERS-1, but additionally having incidence angle as an input variable. This data set was used for preliminary experiments for training a model across the satellite swathe. There were training, test and validation sets, where the training set contained 10,000 examples, the validation and test contained 5,000. These data sets will be referred to as YR1SWATHE.

- Year 2: Data set generated from ERS-2 data at Aston, containing incidence angle in the input data sets. Again, there were training, validation and test data sets. The training data set contained greater than 18,000 examples, the validation set greater than 9,000 examples and the test set 19,000 examples. These data sets will be referred to as YR2SWATHE.

- Year 2: An independent test set generated from the UK Meteorological Office's source was also generated to provide an independent test set. This test set contained 60,000 examples. These data sets will be referred to as UKMO.

The training and validation data sets have a wind speed distribution that is an equal combination of a uniform distribution over wind speed and a distribution close to that of naturally occurring wind speeds; the test data set is generated to have a distribution in wind speed as close to a uniform of distribution as possible.

The wind speed ranges from 4 ms^{-1} to 24 ms^{-1}. For each measurement in the data set there is a corresponding fore, mid and aft scatterometer measurement, incidence angle, azimuth angle, wind speed and meteo wind direction. It is this data that is referred to as 'raw' data in the scope of this thesis.

Before describing the pre-processing applied to the raw data, some new terminology is needed for the different methods of measuring wind direction (all in degrees):

- *Meteo* wind direction is the angle from which the wind is coming. Therefore a meteo wind direction of zero degrees describes a wind coming from the north toward the south. This wind direction will be referred to as $mdir$ (see Figure 2.2).

- *Vector* wind direction is the angle of the direction in which the wind is blowing. Therefore a vector wind direction of zero degrees describes a wind blowing toward the north. This wind direction will be referred to as $vdir$ (see Figure 2.2).

- *Relative* wind direction is the vector wind direction relative to the antenna azimuth angle. This wind direction will be referred to as $rdir$ (see Figure 2.2).

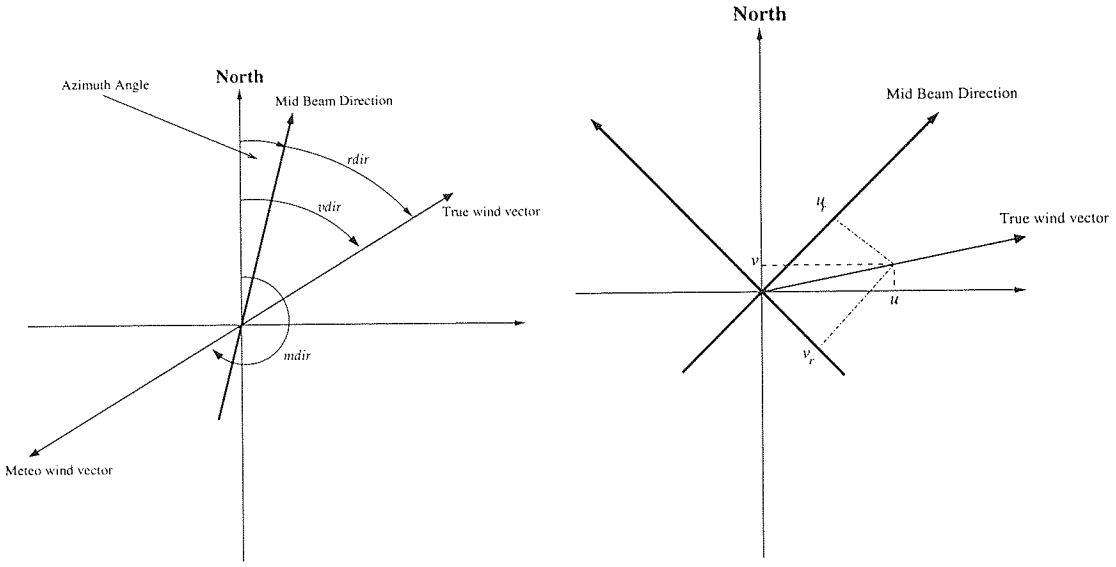
All angles given in this document are quoted in degrees and a clockwise direction from their reference point.

2.2.2 Pre-processing the wind data

The data set describes the wind in terms of wind speed, s, and wind direction, $mdir$. To model the wind vector directly, the data is transformed into relative wind vector components (u_r, v_r).

The wind direction is transformed from meteo direction, $mdir$, to vector direction , $vdir$, by adding 180°, and taking the modulus with respect to 360°. This maintains the convention of wind direction in the range $[0°, 360°)$. Because of the nature of the orbit of the satellite, observations are made in two flight directions (running from north to south, and from south to north). These directions are encoded in the satellite azimuth angle, φ.

The *relative* vector wind direction is the vector wind direction transformed relative to the azimuth angle. As the azimuth angle changes with respect to satellite flight direction it is convenient to reference all vectors relative to the azimuth angle. The transformation is made by subtracting the azimuth angle from vector wind direction, $vdir$, and taking the modulus with respect to 360°. The



(a) Angle definitions

(b) Vector definitions

Figure 2.2: The relationship between the wind direction angles, *mdir*, *vdir*, and *rdir* used in pre-processing and the wind vector components.

wind direction is now relative to the azimuth angle, and the relative wind vector components can be resolved. To ensure that a geophysical reference of zero degrees for north is maintained *rdir* is subtracted from 90° before either sin or cos are applied. The transformation is summarised as:

1. Compute *vdir*: $vdir = mdir + 180°$ (mod 360°).
2. Compute *rdir*: $rdir = vdir - \varphi$ (mod 360°).
3. Compute u_r: $u_r = s\cos(90° - rdir)$.
4. Compute v_r: $v_r = s\sin(90° - rdir)$.

2.2.3 Pre-processing the scatterometer data

The input data, σ^o, is pre-processed by a linear transformation to a zero mean, unit variance Gaussian distribution. The mean and variance for the rescaling are taken from the training set and then fixed. For input data that also contains the incidence angle, θ, we pre-process this input by taking the sine of the incidence angle. This insures that the input is in the range $[-1, 1]$.

2.3 Chapter review

The data used in this project was collected by two satellites orbiting the earth in a polar orbit. For the first year only the ERS-1 data was available, whilst from the second year, the ERS-2 data became available. Unlike the ERS-1 data sets, the ERS-2 data sets were generated at Aston.

Chapter 3

Wind vector retrieval from scatterometer data

Key word summary

Forward Model	Inverse Model
CMOD4	Neural Networks
Wind vector	Measurement triplet
Ambiguity	Mixture Density Network

3.1 Introduction

In this chapter we demonstrate why resolving wind vectors from satellite data is a difficult problem. We review the current published work on this problem; having assessed current models, we propose a new approach to solving this problem by direct inversion of the satellite data to the wind vector described in a Cartesian form. The theory of Mixture Density Networks (MDN) is given, and we show how these models maybe applied to this problem. Finally, a hybrid MDN is developed to model some of the physics of the problem, the inherent ambiguity in wind direction. The training and testing of these models is described in the next chapter.

3.2 Inherent ambiguity in wind direction

There is a set of wind vectors, called the *noisy ambiguity set*, which is identifiable from a single scatterometer measurement. This set shows that an inverse, multi-valued mapping exists between scatterometer data and the geophysical parameters of wind speed and direction (Long and Mendel, 1991).

Ambiguity in wind direction arises from noise on the observation; because of the noisy observations it becomes difficult to distinguish if winds are blowing toward or away from the antennae. This is illustrated in Figure 3.1, a sketch of a two dimensional slice through the manifold defined by the current operational model, CMOD4, at a roughly constant speed. If the observations were noiseless then they would fall on the surface of the manifold, indicated by the solid black line. The areas of ambiguity would only exist where the black lines intersect and hence only for a few wind directions.

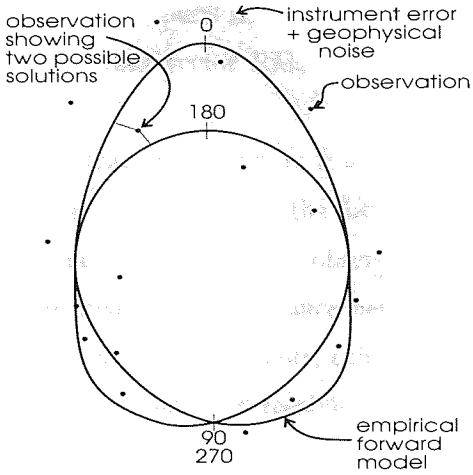


Figure 3.1: A two dimensional sketch of the scatterometer measurement space. The two dimensional slice is taken through the measurement manifold at constant wind speed. For a noisy observation there are at least two solutions in wind direction.

Now consider adding noise to the observation. The observation is placed somewhere near the surface of the manifold in the grey area. Theoretically a noisy observation can come from one of the two surfaces of the manifold which it is close to. Thus, there is no way to distinguish which is the correct solution, and it follows that there are at least two possible solutions for the wind direction for that observation. Any method of inversion of a single observation will have multiple solutions.

3.3 Current methods for local wind vector retrieval

Broadly speaking, two classes of model have been used for local wind vector retrieval. Firstly, there are models based on the physics of the observation system, which model the forward mapping from the local wind vector to scatterometer data; here, inversion requires a second stage. Secondly, there are neural networks which map the direct 'inverse' transfer function from scatterometer data to the local wind vector.

3.3.1 Physically derived forward models

Much effort has been applied to understanding the scatterometer measurement space. An empirical forward model has been developed that describes the mapping from wind speed and direction to the scatterometer space. The first model, CMOD2, was calibrated by the RENE-91 campaign of the coast

off Norway (Offiler, 1994). The forward model has been further developed to the now operational model CMOD4 (Stoffelen and Anderson, 1997b).

Stoffelen and Anderson (1997c), Stoffelen and Anderson (1997a)

Stoffelen and Anderson (1997c) show that scatterometer measurements lie close to a three dimensional manifold defined by CMOD4 in measurement space, and are largely dependent on two geophysical parameters, wind speed and direction. In general the measurements lie within 0.2 dB of the manifold (corresponding to an uncertainty in wind vector RMS error of 0.5 ms^{-1}), which is close to the instrumental measurement noise level.

Most inversion methods which extract wind vectors from scatterometer data are based on local inversions of a forward model. These methods invert the forward model by finding a triplet on the estimated measurement manifold that is closest to the observed triplet; the closest triplet is found by minimising a cost function that measures the distance between the observed and approximated measurements (Stoffelen and Anderson, 1997c). Finally, this paper suggests that a visualisation of the data surface shows that there is little chance of resolving the ambiguity in wind direction locally without some external information, such as reference to a Numerical Weather Forecast (NWP) model.

In Stoffelen and Anderson (1997a) a two stage process is applied to the retrieved wind vectors for removing ambiguous winds. In the first stage, the wind vectors generated during the inversion procedure are compared with a background wind field, where the background wind field is generated by a meteorological forecast. For each observation, the retrieved wind vector closest to the background wind vector is chosen. This is called non-autonomous ambiguity removal due to the reference to background wind fields; an autonomous method would retrieve winds solely on the scatterometer measurements. Following ambiguity removal, a heuristic filter is applied over the wind field to smooth and remove structures that are not geophysically plausible within the wind field.

These methods successfully invert the scatterometer measurements and show that these can provide higher resolution wind fields than those generated at the European Centre for Medium range Weather Forecasting (ECMWF), which are purely meteorologically based forecasts. These methods are not autonomous since they rely on NWP winds for selection of the initial wind fields. Furthermore, there is no probabilistic element to these models.

3.3.2 Neural network based direct inverse models

Thiria *et al.* (1993)

Thiria *et al.* (1993) used neural networks to model wind direction and speed directly from simulated scatterometer data. The model consisted of a total of twenty neural networks. The satellite swathe is divided into ten tracks representing different incidence angles; for each track there were two neural networks: one network modelled wind speed; the other, a classification network with thirty six bins representing ten degree intervals, modelled wind direction by interpreting the outputs of each bin as

a probability.

The inputs to the neural network took neighbourhood information from the eight surrounding cells of a nine by nine grid, where the centre cell was the measurement of interest. This was found to improve the performance by seventeen percent, showing that a spatial context may well be an important consideration in the inverse model. Another interpretation of this improvement is that this configuration may provide an additional wind direction disambiguation skill to the network.

The network for modelling wind direction has inputs of spatial information (the same as those used for the wind speed network) and the estimated wind speed. Wind speed as an input improves the position of the solution on the measurement manifold (the shape of Figure 3.1 is strongly dependent on wind speed). Using speed as an additional input was found to improve performance of direction estimation. Simulated data was used because ERS-1 was not fully operational at the time, and the results showed neural networks to be a promising avenue of investigation for a solution to this inverse problem.

Cornford *et al.* (1999)

Cornford *et al.* (1999) applied a feed forward neural network to model wind speed and a mixture density network (see section 3.5) with kernels of circular normal densities (Bishop and Nabney, 1996) to model the conditional probability density of the wind direction given the scatterometer measurements, σ^o. Two configurations of mixture density networks were considered for modelling wind direction.

In the first configuration the kernels were free to move (that is, input–dependent centres and variances), while in the second the centres and variances of the kernels were fixed. A committee of the direction networks (made up of members from the two configurations just mentioned) was also considered. Additional to the scatterometer triplet measurement $(\sigma^o_{\mathrm{f}}, \sigma^o_{\mathrm{m}}, \sigma^o_{\mathrm{a}})$, incidence angle was included as an input to the networks.

The wind speed model performed within the designed specification of the instrument of 2 ms^{-1}, the results being comparable to the inverted CMOD4 model (Stoffelen and Anderson, 1997c), although the model had some difficulty in learning the transfer function at high and low wind speeds. For wind direction, the models learnt the inherent ambiguity in the problem, but did not perform as well as the inverted CMOD4 model. However, these results suggest a sensible direction of research to follow. When considering the two most likely solutions for the committee of networks, the solution for direction, to within 20°, was correct roughly 75% of the time.

Ambiguity removal was not addressed in this work.

Richaume *et al.* (2000)

Following the methods of Thiria *et al.* (1993), Richaume *et al.* (2000) continued to address the inverse problem by training the networks using data collected from the ERS-1 satellite. There was a dedicated model for each of wind speed and wind direction for each trace.

The results reported indicate performance to be better than the methods proposed by wind retrieval systems based on the CMOD4 forward model. Ambiguity removal was achieved by using a NWP background wind to initialise the system. The application of a three by three cell spatial filter over the wind field minimised the global wind variance within the spatial filter. The results showed wind fields that were consistent, and which compared favourably with wind fields retrieved from the same measurements used by the ECMWF.

3.4 Motivating a probabilistic direct inverse local model.

There are two choices of co-ordinate system by which to model the local wind vectors; either the polar-coordinate system of wind speed and direction, (s, θ), or the Cartesian system of wind vectors (u, v). Therefore there are four ways in which the wind vector can be retrieved depending on the co-ordinate system and the method used (forward or inverse). Of these choices, three of the four systems have been implemented and documented (see Table 3.1). The research in this thesis fills the final area, the direct inversion of the scatterometer data to the Cartesian co-ordinate system, $\boldsymbol{\sigma}^o \rightarrow (u, v)$.

Model combinations		
	Forward model inversion	Direct inversion
(s, θ)	Stoffelen and Anderson (1997b)	Richaume *et al.* (2000), Cornford *et al.* (1999)
(u, v)	Stoffelen and Anderson (1997c)	**This thesis**

Table 3.1: Model types in publication.

Initial inspection of the problem might suggest that modelling the unconditional densities $p(u, v, \boldsymbol{\sigma}^o)$ and $p(\boldsymbol{\sigma}^o)$ may well be plausible and then derive the conditional density required such that

$$p(u, v \mid \boldsymbol{\sigma}^o) = \frac{p(u, v, \boldsymbol{\sigma}^o)}{p(\boldsymbol{\sigma}^o)}, \tag{3.1}$$

where unconditional densities may well be modelled in the Bayesian framework suggested by MacKay (1994). However, much effort has already been applied to modelling the distribution of $\boldsymbol{\sigma}^o$, (Stoffelen and Anderson, 1997c), and $p(\boldsymbol{\sigma}^o)$, (Ramage, 1998; Cornford *et al.*, 2001; Bullen, 2001), which turns out to be a complicated double skinned three dimensional cone. From these works, it is unclear, given the difficulty involved in modelling $\boldsymbol{\sigma}^o$, how straight forward modelling the full density would be. Hence, a more efficient method is to model the conditional density, $p(u, v \mid \boldsymbol{\sigma}^o)$, directly.

In this thesis the wind vectors are modelled directly in a probabilistic framework. Each observation is considered independently, and a model is developed that describes the probability density of a set of wind vector components, (u, v), given a satellite scatterometer observation, $\boldsymbol{\sigma}^o$, expressed $p(u, v \mid \boldsymbol{\sigma}^o)$.

For this purpose a Mixture Density Network (MDN), a principled method for modelling conditional probabilities (Bishop, 1994) which are multi-modal or non-Gaussian, is used. MDNs facilitate the

investigation of the complexity of the mapping from scatterometer data to wind vector component space ($\boldsymbol{\sigma}^o \rightarrow (u, v)$) *and* the complexity of the conditional joint probability distribution ($p(u, v \mid \boldsymbol{\sigma}^o)$) itself. This investigation attempts to answer the following questions:

- How difficult is it to model speed and direction simultaneously by mapping directly to the wind component space?
- Can the incidence angle be used as an input to the MDN? That is, do models trained over all tracks of the swathe perform as well as those models trained on each track within the swathe (and do not take incident angle as an input)?
- Is the conditional probability distribution of the Cartesian wind vectors, (u, v), bi-modal or more complex?
- Is the noise on the Cartesian wind vector components Gaussian?

3.5 Theory of mixture density networks

Mixture Density Networks (MDNs) provide a general framework for modelling conditional probability density functions, denoted $p(\boldsymbol{t}|\boldsymbol{x})$ (Bishop, 1994; Bishop, 1995). The distribution of the outputs, $\boldsymbol{t}$, is described by a model whose parameters are determined by the output of a neural network, which takes $\boldsymbol{x}$ as its inputs. The general model is described by the equations

$$p(\boldsymbol{t}|\boldsymbol{x}) = \sum_{j=1}^{M} \alpha_j(\boldsymbol{x}) \phi_j(\boldsymbol{t}|\boldsymbol{x}) \tag{3.2}$$

and

$$\sum_{j=1}^{M} \alpha_j(\boldsymbol{x}) = 1. \tag{3.3}$$

Here $\alpha_j(\boldsymbol{x})$ represent the mixing coefficients, which depend on $\boldsymbol{x}$; $\phi_j(\boldsymbol{t}|\boldsymbol{x})$ are the kernel distributions of the mixture model, whose parameters also depend on $\boldsymbol{x}$; M is the number of kernels in the mixture model. There are various choices available for the kernel functions, but for the purposes of this work the choice has been restricted to spherical Gaussians of the form:

$$\phi_j(\boldsymbol{t}|\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{c}{2}} \sigma_j^c(\boldsymbol{x})} \exp\left(-\frac{\|\boldsymbol{t} - \boldsymbol{\mu}_j(\boldsymbol{x})\|^2}{2\sigma_j^2(\boldsymbol{x})}\right), \tag{3.4}$$

where c is the dimensionality of the target space. This is a valid restriction: in principle a Gaussian Mixture Model (GMM) with sufficiently many kernels of the type given by (3.4) can approximate arbitrarily closely a density function of any complexity providing the parameters are chosen correctly (McLachlan and Bashford, 1988). It follows then that for any given value of $\boldsymbol{x}$, the mixture model (3.2) can model the conditional density function $p(\boldsymbol{t}|\boldsymbol{x})$. To achieve this the parameters of the mixture model[1] are taken to be general continuous functions of $\boldsymbol{x}$. These functions are modelled by

[1] Choosing a spherical Gaussian kernel implies that the parameter set consists of the mixing coefficients, the variances and the centres (or means) of the kernel functions.

the outputs of a conventional neural network that takes $\boldsymbol{x}$ as its input. It is this configuration of a GMM whose parameters are dependent on the output of a feed forward neural network that takes $\boldsymbol{x}$ as its inputs that is referred to as a *Mixture Density Network* (MDN). It is represented schematically in Figure 3.2.

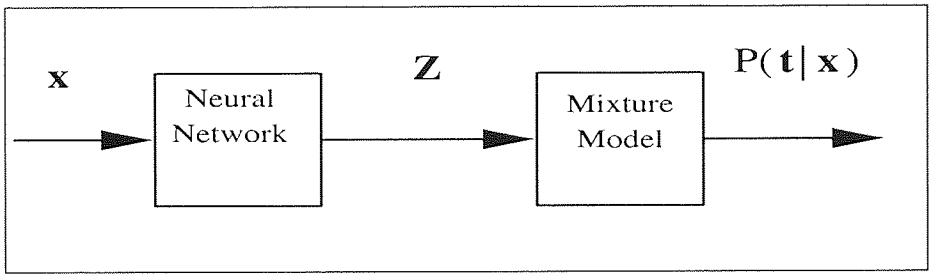


Figure 3.2: The structure of the Mixture Density Network. The inputs $\mathbf{x}$ are feed through a neural network. The outputs of the neural network, $\mathbf{Z}$, define the parameters of the GMM.

Hence, by choosing enough kernels in the mixture model and a neural network with sufficiently many hidden units the MDN can approximate, as closely as desired, any conditional density, $p(\boldsymbol{t}|\boldsymbol{x})$ (Bishop, 1994). The neural network element of the MDN is implemented with a standard Multi-Layer Perceptron (MLP) with single hidden layer of tanh units and an output layer of linear units. The output vector from the MLP, $\boldsymbol{Z}$, holds the parameters that the define the Gaussian mixture model. A single row of the parameter vector, for the n^{th} pattern, takes the following form:

$$\begin{aligned}[\underbrace{\alpha_{1,n}, \alpha_{2,n}, \cdots, \alpha_{j,n}, \cdots, \alpha_{M,n}}_{M \text{ mixing coefficients}},\\ \underbrace{\mu_{11,n}, \mu_{12,n}, \cdots, \mu_{1c,n}}_{1^{st} \text{ kernel centre}}, \cdots, \underbrace{\mu_{j1,n}, \mu_{j2,n}, \cdots, \mu_{jc,n}}_{j^{th} \text{ kernel centre}}, \cdots,\\ \underbrace{\mu_{M1,n}, \mu_{M2,n}, \cdots, \mu_{Mc,n}}_{M^{th} \text{ kernel centre}},\\ \underbrace{\sigma^2_{1,n}, \sigma^2_{2,n}, \cdots, \sigma^2_{j,n}, \cdots, \sigma^2_{M,n}}_{M \text{ widths}}], \end{aligned} \tag{3.5}$$

where the total number of outputs from the MLP is $(c+2) \times M$, as compared with the usual c outputs for an MLP network used in the conventional manner.

3.5.1 Output constraints of the MLP

Following Bishop (1994), the outputs of the MLP are denoted by z_i. These outputs undergo some transformations to satisfy the constraints of the mixture model. The first constraint is that

$$\sum_{j=1}^{M} \alpha(\boldsymbol{x}) = 1 \tag{3.6}$$

and

$$0 \leqslant \alpha(\boldsymbol{x}) \leqslant 1 \qquad \text{for} \quad j = 1, \ldots, M. \tag{3.7}$$

The outputs of the MLP which correspond to the mixing coefficients, z_j^α, are constrained using the 'softmax' function:

$$\alpha_j = \frac{\exp(z_j^\alpha)}{\sum_{i=1}^{M} \exp(z_i^\alpha)}. \tag{3.8}$$

This mapping ensures that the mixing coefficients always sum to unity.

The variance of the kernel represents a scale parameter and always takes a positive value. The variance parameters of the kernels are represented by exponentials of the corresponding outputs of the MLP, z_j^σ:

$$\sigma_i^2 = \exp(z_j^\sigma). \tag{3.9}$$

The centres of the Gaussians represent a location in the target space and can take any value within that space. They are therefore taken directly from the outputs from the MLP, z_{jk}^μ:

$$\mu_{jk} = z_{jk}^\mu \tag{3.10}$$

3.5.2 Deriving the likelihood function.

In order to optimise the parameters in a MDN, an error function is required that provides an indication of how well the model represents the underlying generating function of the training data. The error function of the MDN is motivated from the principle of maximum likelihood (Bishop, 1995). The likelihood of the training data set, $\{\boldsymbol{x}, \boldsymbol{t}\}$, may be written as:

$$\begin{aligned} \mathcal{L} &= \prod_n p(\boldsymbol{x}_n, \boldsymbol{t}_n) \\ &= \prod_n p(\boldsymbol{t}_n|\boldsymbol{x}_n) p(\boldsymbol{x}_n), \end{aligned} \tag{3.11}$$

where the assumption has been made that each data point has been drawn independently from the same distribution, and so the likelihood is a product of probabilities. Generally one wishes to maximise the likelihood function. However, in practice, it is usual to *minimise* the negative logarithm of the likelihood function (termed the negative log likelihood). These are equivalent procedures, since the negative log is a monotonic function. The error function E is defined as the negative log likelihood:

$$E = -\ln \mathcal{L} = -\sum_n \ln p(\boldsymbol{t}_n|\boldsymbol{x}_n) - \sum_n p(\boldsymbol{x}_n). \tag{3.12}$$

The second term in (3.12) is constant because it is independent of the network parameters and can be removed from the error function, which becomes:

$$E = -\sum_n \ln p(\boldsymbol{t}_n|\boldsymbol{x}_n). \tag{3.13}$$

Comparing (3.13) with (3.2), we substitute (3.2) into (3.13) and derive the negative log likelihood error function for the MDN:

$$E = -\sum_n \ln \left\{ \sum_{j=1}^{M} \alpha_j(\boldsymbol{x}_n) \phi_j(\boldsymbol{t}_n|\boldsymbol{x}_n) \right\}. \tag{3.14}$$

In order to minimise the error function the derivatives of the error E with respect to the network weights must be calculated. Providing that the derivatives can be computed with respect to the output of the neural network, the errors for the network weights may be calculated using the standard back-propagation procedure (Bishop, 1995). By first defining the posterior probability of the j^{th} kernel, using Bayes theorem:

$$\pi_j(\mathbf{x}, \mathbf{t}) = \frac{\alpha_j(\mathbf{x})\phi_j(\boldsymbol{t}|\boldsymbol{x})}{\sum_{l=1}^{m} \alpha_l(\boldsymbol{x})\phi_l(\boldsymbol{t}|\boldsymbol{x})} \tag{3.15}$$

the analysis of the error derivatives with respect to the network outputs is simplified. The computation of the error derivative is further simplified by considering the error derivative with respect to each training pattern, n. The total error, E, is defined as a summation of the error, E_n, for each training pattern:

$$E = \sum_{n=1}^{N} E_n, \tag{3.16}$$

where

$$E_n = -\ln\left\{\sum_{j=1}^{m} \alpha_j(\mathbf{x}_n)\phi_j(\mathbf{t}_n|\mathbf{x}_n)\right\}. \tag{3.17}$$

Each of the derivatives of E_n are considered with respect to the outputs of the networks and their respective labels for the mixing coefficients, z_j^α, variance parameters, z_j^σ and centres or position parameters z_{jk}^μ. The derivatives are as follows:

$$\frac{\partial E_n}{\partial z_j^\alpha} = \alpha_j - \pi_j, \tag{3.18}$$

$$\frac{\partial E_n}{\partial z_j^\sigma} = -\frac{\pi_j}{2}\left\{\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{\sigma_j^2} - c\right\}, \tag{3.19}$$

$$\frac{\partial E_n}{\partial z_{jk}^\mu} = \pi_j\left\{\frac{\mu_{jk} - t_k}{\sigma_j^2}\right\}, \tag{3.20}$$

where a full derivation is given in Appendix B.

3.6 Training mixture density networks for direct inversion of scatterometer data.

The aim of this section is to show how an MDN may be employed to model the inverse mapping from scatterometer data, σ^o, directly to the wind vector component space, (u, v). We also develop a hybrid MDN structure which encodes some of our prior geophysical knowledge: the inherent ambiguity in the wind direction.

The MDNs are implemented in NETLAB, a neural network toolbox for MATLAB. NETLAB is available freely on the Internet[2], an overview of the functions employed is given in Appendix C.

[2]Available from http://www.ncrg.aston.ac.uk/netlab/

3.6.1 Putting mixture density networks into a geophysical context.

In Section 3.5 the inputs and targets of the MDN are labelled as $\boldsymbol{x}$ and $\boldsymbol{t}$ respectively. In the context of this application, each input pattern for the MDN, $\boldsymbol{x}$, is scatterometer data (the triplet $(\sigma^o_{\rm f}, \sigma^o_{\rm m}, \sigma^o_{\rm a})$), or scatterometer data and incidence angle (the input vector $(\sigma^o_{\rm f}, \sigma^o_{\rm m}, \sigma^o_{\rm a}, \sin(\theta))$) if the MDN is being trained over all traces. Modelling the wind vector components directly implies that the targets of the MDN, $\boldsymbol{t}$, are the Cartesian wind vector components (u, v). The general description of the MDN, (3.2), is then re-expressed using geophysical parameters for a particular wind vector component as

$$p(u, v|\boldsymbol{\sigma}^o) = \sum_{j=1}^{M} \alpha_j(\boldsymbol{\sigma}^o)\phi_j(u, v \mid \boldsymbol{\sigma}^o). \tag{3.21}$$

This project uses data sets generated from real world processes, and so assumptions made in the modelling process need to be validated. There are three main assumptions made about the data set in our approach:

- The noise on the targets in (u, v) space is locally Gaussian and spherically symmetric.
- The theory in Section 3.5 assumes that the inputs, $\boldsymbol{x}$, are noiseless. It is therefore assumed that the inputs, $\boldsymbol{\sigma}^o$, are noiseless. This a reasonable assumption based on the quality of the $\boldsymbol{\sigma}^o$ measurements collected from the ERS-1 satellite. The scatterometer measurements are within $0.2db$ of the CMOD4 manifold, which corresponds to an uncertainty in wind vector RMS error of 0.5 ms^{-1}, when compared to the errors on NWP target winds which are typically 3.0 ms^{-1} (Stoffelen and Anderson, 1997c)
- For computation of the error function (3.14) it is assumed that all data is independently drawn from the same distribution. The selection of the data for the ERS–2 data sets ensures that this condition is met.

3.6.2 Training the conventional mixture density network

The networks are trained by minimising the negative log likelihood error function defined by (3.14):

$$E = -\sum_{i=1}^{N} \ln\left\{\sum_{j=1}^{M} \alpha_j(\boldsymbol{\sigma}^o_i)\phi_j(u_i, v_i \mid \boldsymbol{\sigma}^o_i)\right\} \tag{3.22}$$

where $\boldsymbol{\sigma}^o$ represents either the three or four dimensional input vectors, $(\sigma^o_{\rm f}, \sigma^o_{\rm m}, \sigma^o_{\rm a})$ or $(\sigma^o_{\rm f}, \sigma^o_{\rm m}, \sigma^o_{\rm a}, \sin(\theta))$, respectively. Network training was regularised by the 'early stopping' technique (Bishop, 1995) using the validation set.

3.6.3 Encoding geophysical knowledge in the hybrid mixture density network

In this section we describe a modification to the MDN structure in order to incorporate the knowledge that ambiguities in wind direction exist in the mapping from $\boldsymbol{\sigma}^o$ to (u, v) space. We call this the *hybrid* MDN model.

The following expression for an MDN with two kernels can be derived from (3.2):

$$p(\boldsymbol{t}|\boldsymbol{x}) = \alpha(\boldsymbol{x})\phi_1(\boldsymbol{t}|\boldsymbol{x}) + (1 - \alpha(\boldsymbol{x}))\phi_2(\boldsymbol{t}|\boldsymbol{x}). \tag{3.23}$$

As already established, the ambiguity in wind direction arises from the fact that there are alias solutions for the wind direction; it is not known for certain from the σ^o data in which of two directions the wind is blowing. This alias is at approximately 180°.

The ambiguity is encoded within the MDN framework by two spherical Gaussian kernels with diametrically opposed centres. One kernel is free to move (its parameters are determined during training), whilst the second mirrors the first by taking the negative mean (which is equivalent to an ambiguous direction of 180° in (u, v) space). The centres of the kernels (which correspond to wind vectors in (u, v) space) always represent the ambiguity within the mapping.

The noise model for each kernel is assumed to be the same; the variance of the mirroring Gaussian is the same as that of the free Gaussian.

The mixing coefficients determine the probability mass associated with each mode within the MDN. Expressed within the MDN framework the model becomes (for the n^{th} observation):

$$p(\boldsymbol{t}_n|\boldsymbol{x}_n) = \alpha(\boldsymbol{x}_n)\phi(\boldsymbol{t}_n|\boldsymbol{x}_n) + (1 - \alpha(\boldsymbol{x}_n))\psi(\boldsymbol{t}_n|\boldsymbol{x}_n), \tag{3.24}$$

where the kernels are defined by diametrically opposed spherical Gaussians:

$$\phi(\boldsymbol{t}_n|\boldsymbol{x}_n) = \frac{1}{2\pi\sigma^2(\boldsymbol{x}_n)} \exp\left(-\frac{\|\boldsymbol{t}_n - \boldsymbol{\mu}(\boldsymbol{x}_n)\|^2}{2\sigma^2(\boldsymbol{x}_n)}\right), \tag{3.25}$$

$$\psi(\boldsymbol{t}_n|\boldsymbol{x}_n) = \frac{1}{2\pi\sigma^2(\boldsymbol{x}_n)} \exp\left(-\frac{\|\boldsymbol{t}_n + \boldsymbol{\mu}(\boldsymbol{x}_n)\|^2}{2\sigma^2(\boldsymbol{x}_n)}\right). \tag{3.26}$$

The target data is two dimensional and therefore the dimension parameter c in the Gaussian model (3.23) is two. The error for a pattern n is defined as a negative log likelihood function and is derived from (3.14):

$$E_n = -\ln\left\{\alpha(\boldsymbol{x}_n)\phi(\boldsymbol{t}_n|\boldsymbol{x}_n) + (1 - \alpha(\boldsymbol{x}_n))\psi(\boldsymbol{t}_n|\boldsymbol{x}_n)\right\}. \tag{3.27}$$

The training of the network is identical in principle to the general NETLAB MDN framework. Some modifications are needed to compute the function which maps the outputs of the MLP to the parameters of the mixture model and the gradient of the error function with respect to the MLP outputs. The mixing coefficients are no longer constrained by the softmax rule, but by the simpler logistic function:

$$\alpha = \frac{1}{1 + \exp(-z^\alpha)}. \tag{3.28}$$

To train the hybrid architecture, the derivative of the gradient of the error function with respect to the outputs of the MLP is required. Two posterior probabilities are defined with respect to each kernel. For the free kernel:

$$\pi = \frac{\alpha\phi}{\alpha\phi + (1 - \alpha)\psi}, \tag{3.29}$$

and for the mirroring kernel:

$$\gamma = \frac{(1 - \alpha)\psi}{\alpha\phi + (1 - \alpha)\psi}. \tag{3.30}$$

Then the derivatives of the error function with respect to the network outputs are:

$$\frac{\partial E_n}{\partial z^{\alpha}} = \pi - \alpha, \tag{3.31}$$

$$\frac{\partial E_n}{\partial z^{\sigma}} = -\left[\frac{\pi}{2}\left\{\frac{\|\boldsymbol{t}_n - \boldsymbol{\mu}\|^2}{\sigma^2} - c\right\} + \frac{\gamma}{2}\left\{\frac{\|\boldsymbol{t}_n + \boldsymbol{\mu}\|^2}{\sigma^2} - c\right\}\right], \tag{3.32}$$

$$\frac{\partial E_n}{\partial z_k^{\mu}} = -\left[\pi\left(\frac{t_k - \mu_k}{\sigma^2}\right) - \gamma\left(\frac{t_k + \mu_k}{\sigma^2}\right)\right]. \tag{3.33}$$

$$\tag{3.34}$$

This configuration reduces the number of mixture model parameters by $M(\frac{c}{2} + 1)$. The full detail of the derivation is presented in Appendix D. Software for implementing this architecture is coded in MATLAB and designed to integrate with the NETLAB toolbox (this implementation inspired the general *fast mdn* implementation, see appendix C). The implementation was tested for accuracy using the methods detailed in Evans (1998).

3.7 Chapter review

Resolution of wind vectors is difficult from scatterometer data because of the inherent ambiguity in wind direction. We have reviewed current wind vector retrieval methods, describing approaches to resolving the ambiguity problem. We have established that there is not a model that directly maps scatterometer data to Cartesian wind vectors.

We have described the theory of the MDN, a general framework for modelling multi-modal conditional probabilistic functions. We showed that the MDN framework may be applied to the wind vector problem, and a hybrid MDN was developed that directly models the inherent ambiguity in the problem.

In the next chapter the MDN models are trained and the results assessed.

Chapter 4

The direct inversion of scatterometer data

Key word summary

Mapping: $\sigma^o \rightarrow (u, v)$	**Complexity:** $p(u, v \mid \sigma^o)$
MLP	**GMM**
# Hidden Units	**# Kernels**
Trace	**Swathe**

4.1 Introduction

In this chapter we show that MDNs successfully model the mapping $\sigma^o \rightarrow (u, v)$ and we train a definitive MDN for inclusion in the global wind field model.

Exploratory experiments investigate the complexity of the mapping from $\sigma^o \rightarrow (u, v)$ and the complexity of the $p(u, v \mid \sigma^o)$ space. We compare the results of MDNs trained over traces with other published work in the field and show that MDNs are capable of taking an additional input of incidence angle, and hence we train one model for the complete swathe.

With the YR2SWATHE data set we train several models over the swathe. We also consider committees of networks. Finally, we define the MDN model to be used in the global wind field model.

4.2 Exploratory Experiments

Two of the goals set out in Section 3.4 were to investigate the complexity of the mapping from $\sigma^o \rightarrow (u, v)$ and to understand the complexity of the $p(u, v \mid \sigma^o)$. By varying the configuration of the MDN we can investigate these areas of interest. The complexity of the mapping from $\sigma^o \rightarrow (u, v)$,

modelled by the MLP part of the MDN, is represented by the number of hidden units in the MLP. The complexity of $p(u, v \mid \boldsymbol{\sigma}^o)$ is represented by the number of kernels in the mixture model of the GMM. Hence, by varying the number of hidden units and the number of kernels in the MDN we can investigate both complexity of the mapping from $\boldsymbol{\sigma}^o \rightarrow (u, v)$ and the complexity of the $p(u, v \mid \boldsymbol{\sigma}^o)$.

Finally, we are also interested to know whether networks can be trained to take incidence angle as an input, and so have one model for a swathe of satellite data (instead of ten, if we do not have incidence angle as an input).

Training configurations for the standard MDN

The first experiments trained individual models for each trace (0 to 9) using the YR1TRACE data sets. The number of hidden units in the MLP was varied from ten to twenty five in steps of five. For each MLP configuration a model was trained with two and four kernels.

The models were trained over ten traces to ensure that there was no 'overlapping' of the scatterometer readings which would happen if the networks were trained with adjacent tracks (see Section 2.2.1). This helps maintain our assumption the the local models are conditionally independent. We have also established that the mapping from $\boldsymbol{\sigma}^o$ space to $p(u, v \mid \boldsymbol{\sigma}^o)$ is at least bi–modal; therefore at least two kernels are necessary in the GMM. Four kernels were also chosen to establish if: (i) $p(u, v \mid \boldsymbol{\sigma}^o)$ is more complex than a bi–modal distribution, (ii) the modes are more complex than the spherical Gaussian assumption made by an MDN with two kernels.

The initial experiments for modelling the complete swathe were carried out using the YR1SWATHE data set. There were two configurations for the number of hidden units in the MLP: thirty five and fifty. For each MLP configuration there were models trained with two, four, five, and twelve kernels.

Thirty five and fifty hidden units were chosen because the addition complexity of the incidence angle was unknown; the results from the training on individual traces also indicated that more than twenty five hidden units would be required. Further kernel configurations of five and twelve were chosen to investigate the effects of over-fitting the density $p(u, v \mid \boldsymbol{\sigma}^o)$.

Training configurations for the Hybrid MDN

Architectures with the same number of hidden units and input dimensions were trained to compare the hybrid MDN with the standard MDN. For the initial hybrid MDN experiments there were two kernels two (see Section 4.7.1 for further development of this model). Hence the following network configurations were trained.

- For models without incidence angle as an input:
  - a model for each trace (0 to 9) and with $10, 15, 20, 25$ hidden units in the MLP.
- For models with incidence angle as an input:
  - configure 35 and 50 hidden units in the MLP.

Model training

The networks are trained by minimising the negative log likelihood error function defined by (3.14):

$$E = -\sum_{i=1}^{N} \ln\left\{\sum_{j=1}^{M} \alpha_j(\boldsymbol{\sigma}_i^o)\phi_j(u_i, v_i \mid \boldsymbol{\sigma}_i^o)\right\}$$

where $\boldsymbol{\sigma}^o$ represents three or four dimensional input vectors, $(\sigma_{\mathrm{f}}^o, \sigma_{\mathrm{m}}^o, \sigma_{\mathrm{a}}^o)$ and $(\sigma_{\mathrm{f}}^o, \sigma_{\mathrm{m}}^o, \sigma_{\mathrm{a}}^o, \sin(\theta))$ respectively. The targets are the wind vector components $(u_{\mathrm{r}}, v_{\mathrm{r}})$. The error is minimised by applying the Scaled Conjugate Gradient algorithm implemented in the NETLAB toolbox. Training typically took approximately three thousand epochs when training per trace and eight thousand epochs when training over all traces. *Early stopping* regularisation was implemented to minimise over-fitting to training data; at every hundred epochs (two hundred when training over all incidence angles) the training was stopped and the error of the training set, the *training error*, and the error of the validation set, the *validation error* were then computed and recorded. By comparing the evolution of the two error measurements training was stopped at a point when the MDN best modelled the *underlying data generator* and not the noise on the target data, that is, where the validation error started to rise, whilst the training error continued to fall.

4.3 Results of exploratory experiments

In total twelve networks were trained for each trace (and so a total of one hundred and twenty networks) and ten networks were trained over all traces. Each network had the same seed in the random number generator when initialised.

The results are presented in tabular form, using summary statistics commonly used in the meteorological community; the *Figure of Merit*, an evaluation of how well the predictions compare to the instrument specifications of $\pm 2\ ms^{-1}$ for wind speed and $\pm 20°$ for wind direction; the vector Root Mean Square (RMS) error, a measure of how close the predictions are to the target; and *performance at* 20° (denoted *perf.* @ 20°), a measure of the percentage of predicted directions within 20° of the target wind directions (Cornford *et al.*, 1999; Richaume *et al.*, 2000). The technical details of these summary measures are given in Appendix E.

The reported results are computed using the *test set*, which was not used when choosing the model complexity, and so the results are unbiased with respect to the data set. The results are based on wind vector predictions derived from a simple ambiguity removal algorithm. The two most likely wind vectors were inferred from the MDN and then compared to the target wind vectors. For models with more than two kernels, the position of the two most probable modes, found by an SCG optimisation starting from the centres of each kernel, were inferred as the two most likely wind vectors. The predicted wind vector closest to the target wind vector was chosen as the disambiguated wind vector. The summary statistics were then computed.

Models trained by trace

Tables F.1, F.2, and F.3 present the summary results for networks trained per trace. Over all the tables is a general trend of increasing performance from trace 0 to trace 9, which is to be expected because the mapping is harder to model for the innermost traces of the swathe. Table F.1 presents model performance as measured by the FoM evaluation function. An FoM result greater than one indicates that the model is performing to the instrument specifications. Inspection of Table F.1 shows that the models are close to this threshold, and all the models except for one meet the specification for trace 9. The performance also exhibits trends over model complexity. For models with two kernels, including the hybrid configuration, the measure continually improves as complexity increases; for models with four kernels the maximum performance is achieved with twenty hidden units in the MLP.

The results in Table F.2 and Table F.3 are strongly correlated to the results in Table F.1. For vector RMS errors the current operational model CMOD4 returns wind vector RMS errors of 3 ms^{-1} when compared to ECMWF winds (Stoffelen and Anderson, 1997b). The results presented here show higher values than 3 ms^{-1}, but follow the same trends as those in Table F.1, the lowest vector RMS error being 3.11 ms^{-1}. The results in Table F.3 for *perf.* @ 20° show similar trends to those for FoM and vector RMS error. The results are comparable with the results of Cornford *et al.* (1999), where the correct solution, within 20°, is found more than 70% of the time from the two most probable aliases.

Models trained over all traces

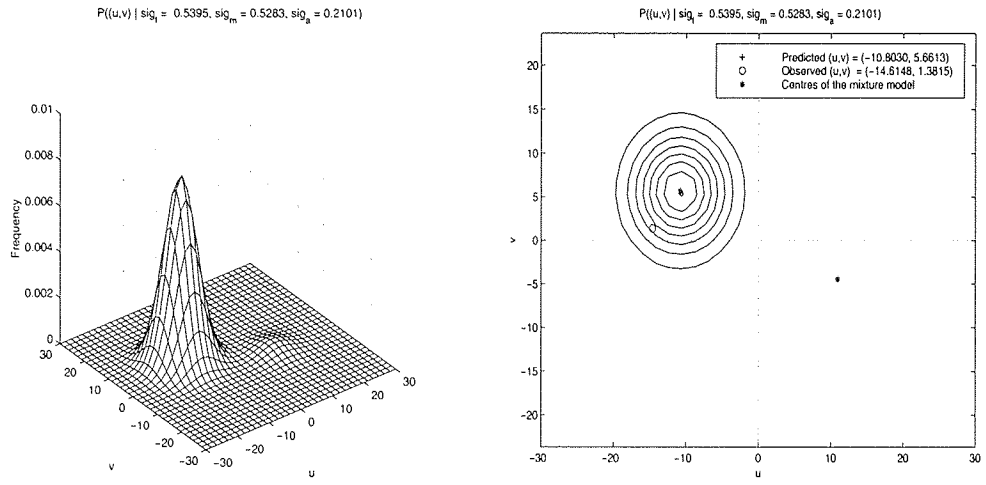
Table F.4 presents the results of networks trained over all traces. Model performance is similar to the MDNs with thirty five hidden units in the MLP. In addition, the model performance for MDNs with twelve kernels is worse than the other kernel configurations, reflecting that there is a limit to the complexity an MDN can have to model this problem.

For comparison between networks with and without incidence angle as an input, results are presented by trace for networks that take incidence angle as an input. The results are generated by using the test sets for models trained by trace and adding the respective incidence angles to the input patterns. Again, the summary results are presented in Tables F.5, F.6, F.7. These tables also show a general trend of improving performance, which is smoother than the networks trained individually per trace. However the best model trained with incidence angle is worse than those trained by trace.

Visualisation of $p(u, v \mid \sigma^o)$

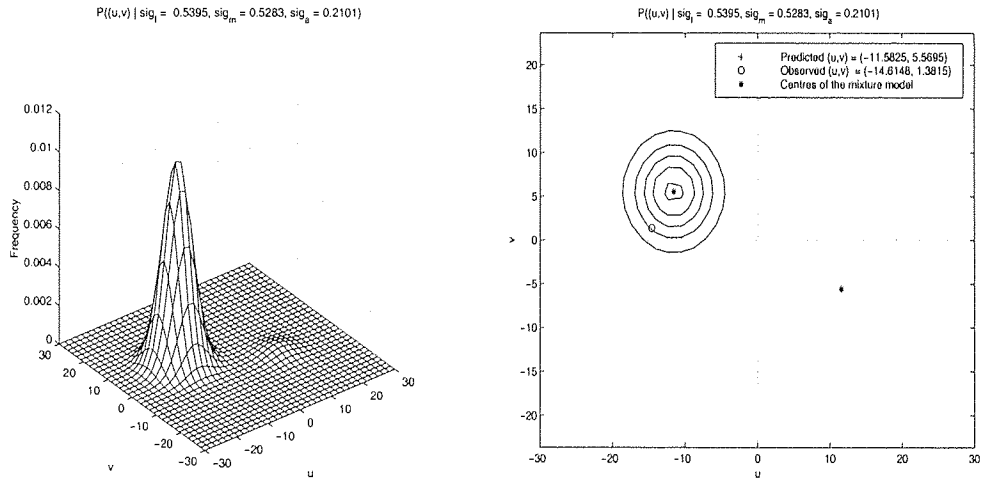
Visualisation of the conditional probability density function modelled by the MDN is of interest. Mesh and contour plots give a good visualisation of the probability density function for individual data points. Several model architectures have been selected to represent the range of models trained, and to show how the distribution varies for changing kernel configurations. An arbitrary wind vector (which gives a good graphical visualisation of the distribution) was chosen from trace 9 test data (for

information the target wind vector is $(-14.6, 1.4)$). Figures 4.1, 4.2 and 4.3 are for networks trained per trace and represent kernel configurations of two, two hybrid and four respectively.



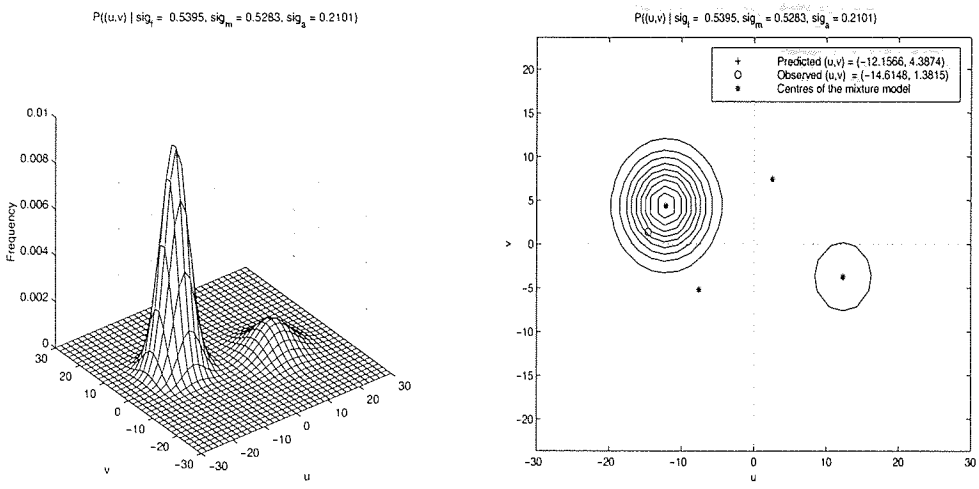
(a) (b)

Figure 4.1: Conditional probability distribution plots, for a model with 2 kernels and 25 hidden units. The circle denotes the true target.



(a) (b)

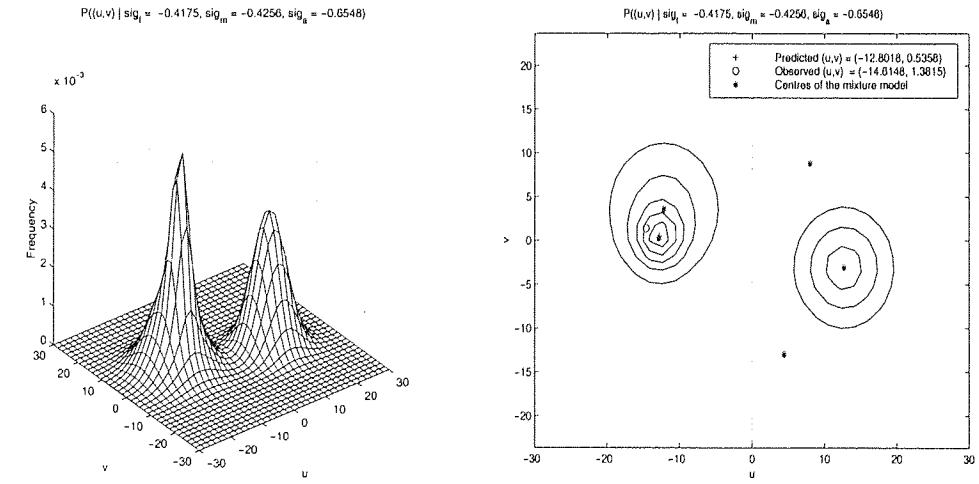
Figure 4.2: Conditional probability distribution plots, for a hybrid model with 2 kernels and 25 hidden units. The circle denotes the true target.



(a) (b)

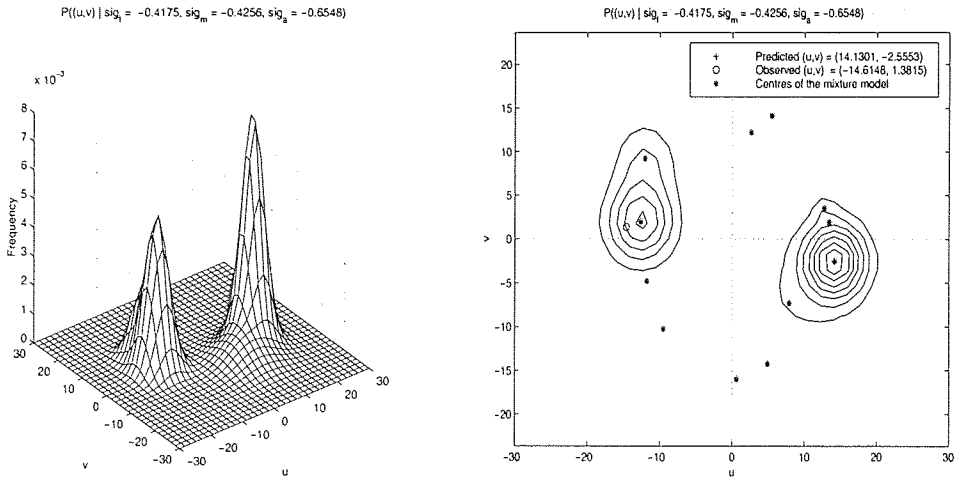
Figure 4.3: Conditional probability distribution plots, for a model with4 kernels and 25 hidden units. The circle denotes the true target.

Figures 4.4 and 4.5 are for networks trained with incidence angle as an input and are for MDNs with five and twelve kernels respectively. For this pattern, the networks trained with incidence angle as an input make a more accurate prediction. These plots show that the conditional probability distribution is generally bi–modal and closer inspection of Figures 4.4 and 4.5 also shows that the modes are not necessarily Gaussian.



(a) (b)

Figure 4.4: Conditional probability distribution plots with incidence angle, for a model with 5 kernels and 50 hidden units. The circle denotes the true target.



(a) (b)

Figure 4.5: Conditional probability distribution plots with incidence angle, for a model with 12 kernels and 50 hidden units. The circle denotes the true target.

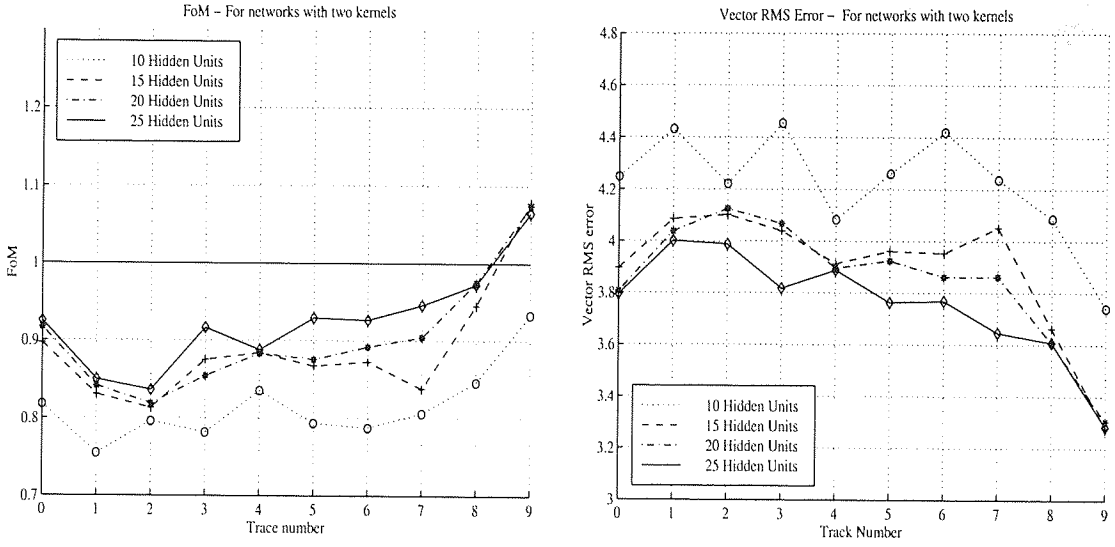
4.4 Analysis of the exploratory results

Both the complexity of the mapping $\boldsymbol{\sigma}^o \rightarrow (u, v)$ and of the probability density function $p(u, v \mid \boldsymbol{\sigma}^o)$ require analysis. These properties are represented by the MLP and GMM structures respectively within the MDN framework. The mapping $\boldsymbol{\sigma}^o \rightarrow (u, v)$ is modelled by the MLP, the complexity of which is controlled by the number of units in the hidden layer. The complexity of $p(u, v \mid \boldsymbol{\sigma}^o)$ is modelled by the GMM and the complexity of $p(u, v \mid \boldsymbol{\sigma}^o)$ is controlled by number of kernels in the GMM. There are two kinds of model to compare: those which have been trained without incidence angle as an input (trained for a specific trace) and those that take incidence angle as an input and are general models over the whole swathe. The results are presented by plotting the FoM and vector RMS results over all the traces. This format helps to highlight trends in the results.

4.4.1 The complexity of the mapping $\boldsymbol{\sigma}^o \rightarrow (u, v)$

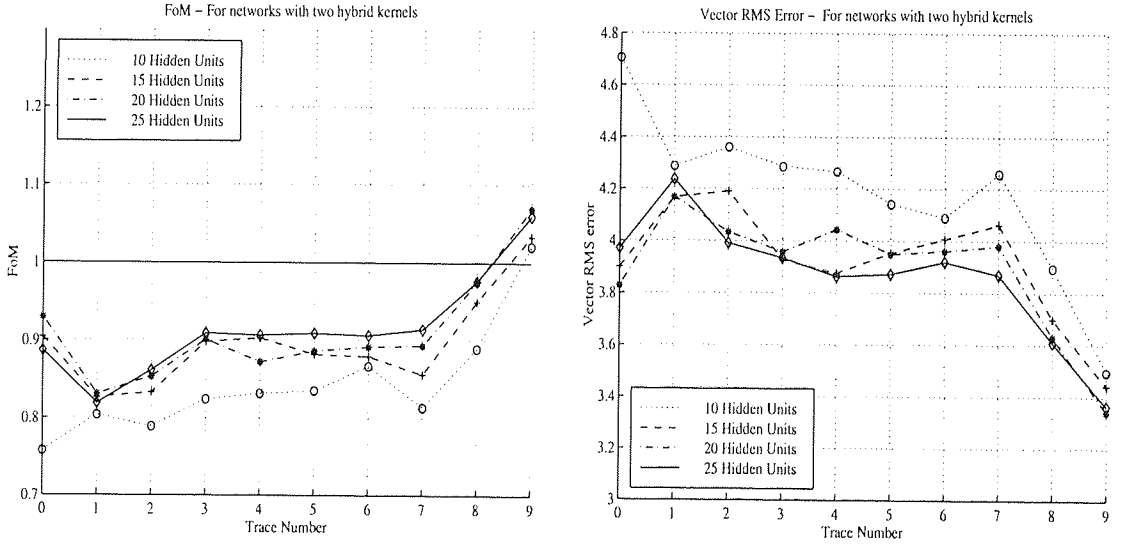
In order to investigate the complexity of the mapping $\boldsymbol{\sigma}^o \rightarrow (u, v)$ the MDN architectures were trained with ten, fifteen, twenty and twenty five units in the hidden layer of the MLP. Also networks were trained with different kernel configurations (to investigate the complexity of $p(u, v \mid \boldsymbol{\sigma}^o)$), which may also affect model performance, and so comparisons in this section are made by kernel configuration, over the number of units in the hidden layer of the MLP.

For two kernels, Figure 4.6 shows that increasing the number of hidden units improves the model performance. Figure 4.7, shows similar results, but for the hybrid MDN configuration. Again, increasing the number of units in the MLP improves the performance of the model, but in this case the improvement is not as distinct as for those configurations with two free kernels.



(a) *FoM* (b) Vector RMS error

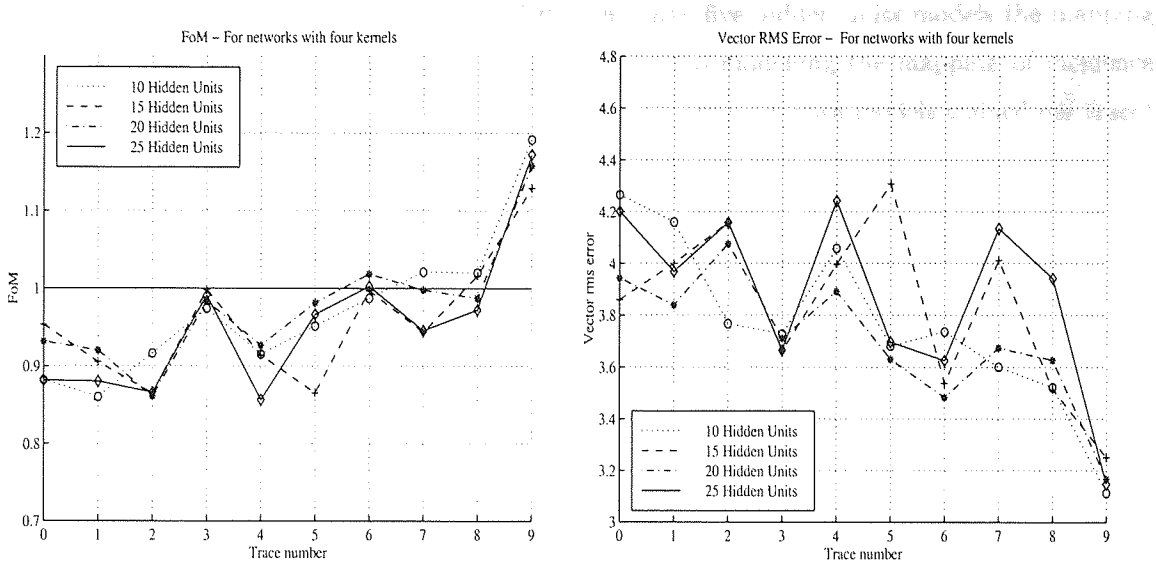
Figure 4.6: Model performance by the number of hidden units in the MLP, for models with two kernels and trained per trace.



(a) *FoM* (b) Vector RMS error

Figure 4.7: Model performance by the number of hidden units in the MLP, for models with hybrid kernels and trained per trace.

The results of MDNs with four kernels are plotted in Figure 4.8 and show that the model performance does not significantly improve by increasing the number of units in the hidden layer of the MLP over the range trained.



(a) FoM

(b) Vector RMS error

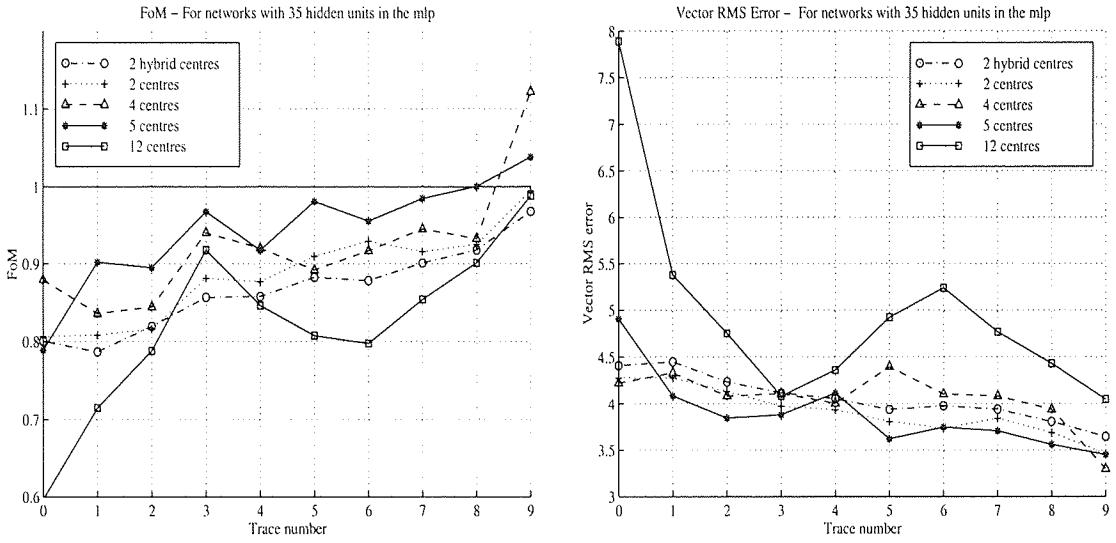
Figure 4.8: Model performance by the number of hidden units in the MLP, for models with four kernels and trained per trace.

For models with two kernels there is a correlation between increasing the number of hidden units in the MLP and improving model performance. The best models have twenty five units in the hidden layer of the MLP. For models with four kernels the results show that the best performance is achieved by an MLP with twenty hidden units (see Figure 4.8) and that model performance for an MLP with twenty five hidden units is worse. There are two possible explanations for this. Firstly the complexity of the MLP with twenty five hidden units is sufficient to over-fit the training data, decreasing the model's ability to generalise. Secondly, because of the complexity of the MLP, the MDN becomes stuck in a local minima in the error surface, and the network fails to find the weights that give optimum generalisation.

For MDNs with a hybrid kernel configuration, the results suggest that the MLP is reaching a limit in its ability to model the mapping $\boldsymbol{\sigma}^o \rightarrow (u, v)$, given the GMM configuration. The improvement in performance does not change significantly for hidden units of twenty and twenty five. Comparing these results with MDN configurations having two free kernels confirms that the assumption that the inherent ambiguity in the problem is generally 180°. The slightly lower performance for the hybrid MDN is attributed to fixing the ambiguity to 180° and having equal variances on each kernel. This implies that the model is less flexible than those with two free kernels. Generalising to data that violates these assumptions will be more difficult for the hybrid configuration (that is, if the ambiguity moves away from 180°, and/or the variance of each mode is significantly different).

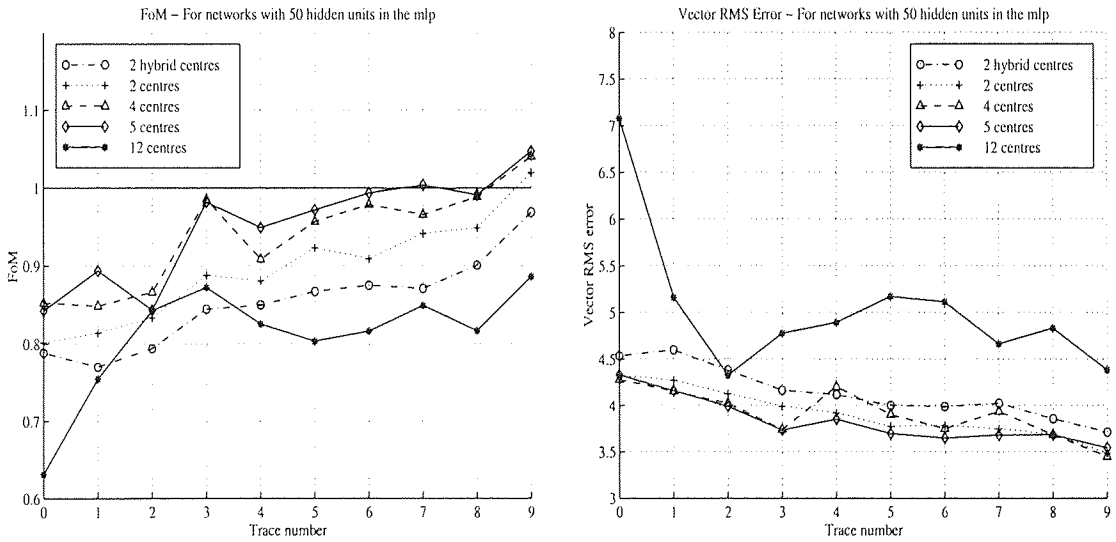
The MDNs that take incidence angle as an input were trained with two MLP configurations: thirty five, and fifty hidden units. Comparing the results of Table F.5 and the graphs in Figures 4.9 and 4.10, we see that the performance of the model is affected more by the number of kernels than

the number of hidden units in the MLP. An MLP with thirty five hidden units models the mapping as well as an MLP with fifty hidden units. The MLP is also modelling the mapping of incidence angle to $p(u, v \mid \sigma^o)$. An extra ten units in the hidden layer (compared with models trained per trace) adequately models this additional complexity.



(a) *FoM* (b) Vector RMS error

Figure 4.9: Model performance by the number of kernels. For a network with thirty five units in the hidden layer, and trained with incidence angle as an input.



(a) *FoM* (b) Vector RMS error

Figure 4.10: Model performance by the number of kernels. For a network with fifty units in the hidden layer, and trained with incidence angle as an input.

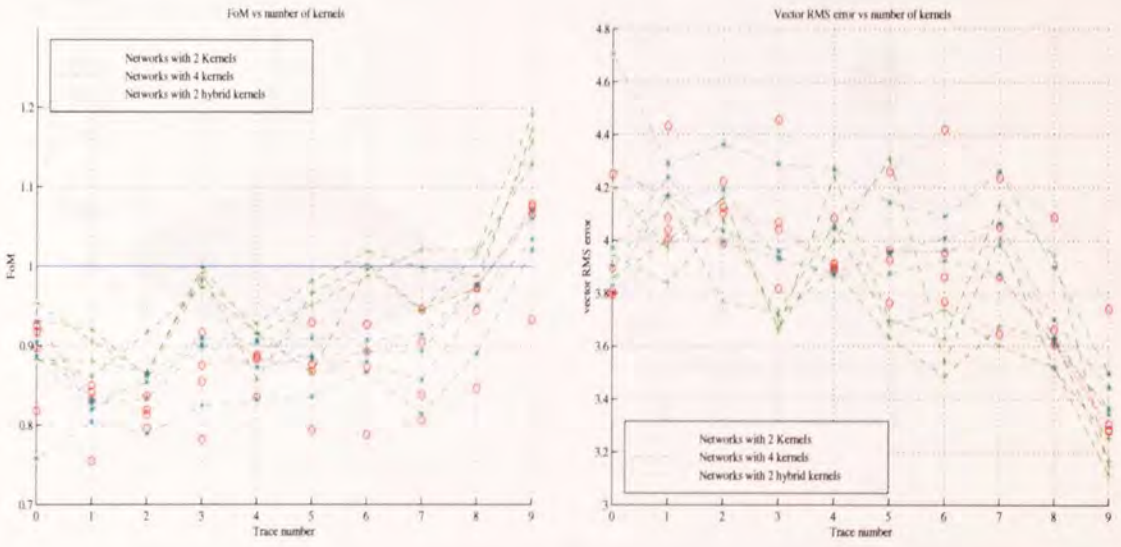
In summary, for MDNs trained per trace, with four kernels, the mapping of $\boldsymbol{\sigma^o} \rightarrow (u, v)$ is adequately approximated by an MLP with ten hidden units, although the optimal solution requires twenty hidden units. For MDNs trained per trace with two kernels (including hybrids), twenty five hidden units in the MLP are required. By comparing the model performance of the MDNs with hybrid and two free kernel configurations the assumption that the ambiguity in the problem is principally at 180° has been shown to be correct. Models with two kernels and twenty five hidden units in the MLP do not perform as well as models with four kernels and ten hidden units in the MLP. This is attributed to the structure of the probability density $p(u, v \mid \boldsymbol{\sigma^o})$, and is discussed in Section 4.4.2.

For networks trained with incidence angle as an input, an MLP with thirty five hidden units satisfactorily models the mapping $(\boldsymbol{\sigma^o}, \theta) \rightarrow (u, v)$. Again, the number of kernels in the MDN contributes more significantly to changes in model performance than changing the number of hidden units in the MLP.

4.4.2 The complexity of the conditional probability distribution $p(u, v \mid \boldsymbol{\sigma^o})$

The complexity of the probability distribution $p(u, v \mid \boldsymbol{\sigma^o})$ is reflected by the number of kernels in the MDN. If $p(u, v \mid \boldsymbol{\sigma^o})$ is a highly complex distribution then a larger number of kernels are required to model the distribution than if $p(u, v \mid \boldsymbol{\sigma^o})$ is relatively simple.

Considering MDNs trained by trace, the graphs in Figure 4.11 show a trend of increasing performance for an increasing number of kernels in the MDN.



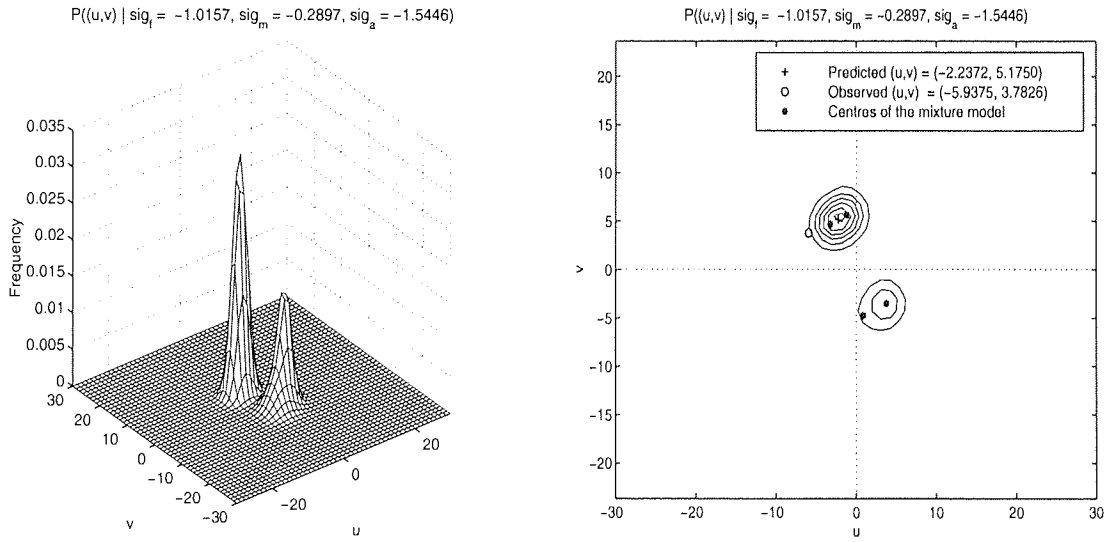
(a) (b)

Figure 4.11: Model performance, comparing networks which are trained by trace, by the number of kernels.

The plots for two kernels have a large variance which indicates a significant dependence on the

number of hidden units in the MLP. The plots for four kernels have less variance indicating that there is less dependence on the number of hidden units in the MLP. The inference from these plots is that an MDN with two kernels is insufficiently complex to model the conditional probability distribution, whereas an MDN with four kernels has increased flexibility and sufficiently models the conditional probability distribution.

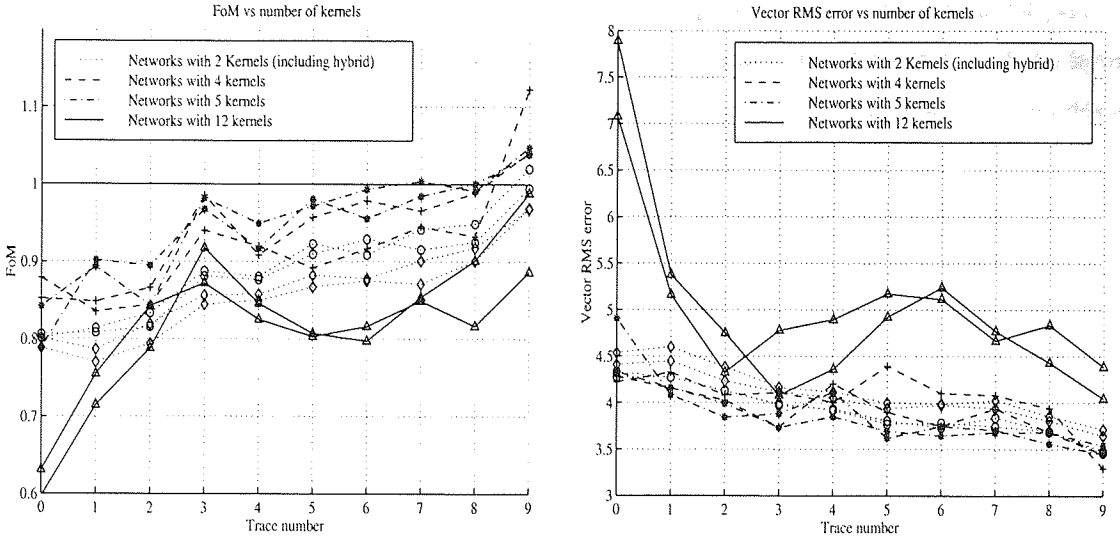
The plots in Figures 4.1, 4.2 and 4.3 show that the probability distribution is predominantly bi-modal. Our attention is drawn to the fact that the models with four kernels perform better than models with two kernels. Three possible explanations are offered. The first is that four centres provide more flexibility when modelling a probability distribution than two; this is shown in Figure 4.3 where the positions of the kernels are placed in roughly four quadrants of (u, v) space. Models with two kernels do not have that kind of flexibility. The second is that the noise on the targets is not Gaussian or spherically symmetrical. Figure 4.12 shows such a case where each mode is modelled by two superimposed kernels; again, the MDNs with two kernels do not have the flexibility to model non-Gaussian modes, only being able to approximate each mode by a symmetric Gaussian probability distribution function. The third is that at low wind speed, wind direction is uncertain and difficult to predict compared to wind speed. Hence, an MDN with four kernels has the ability to model this scenario, where as an MDN with two kernels does not have sufficient flexibility. This is illustrated by an example in Figure 4.15.



(a) (b)

Figure 4.12: An example of $p(u, v \mid \sigma^o)$ where the modes are non-Gaussian.

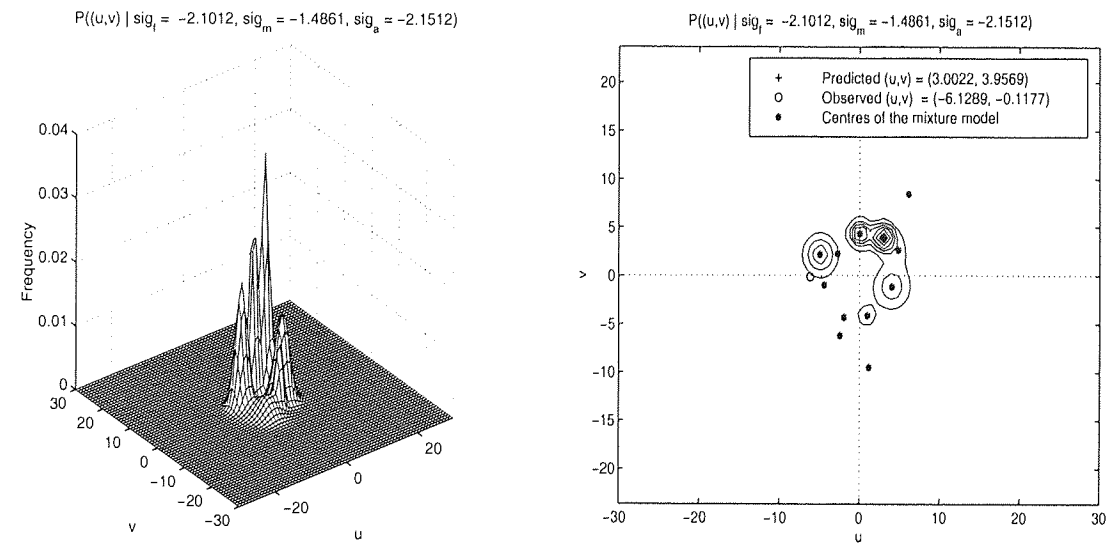
Switching attention to models that take incidence angle as an input, the results show an interesting relation with the number of kernels in the MDN. The MDNs with twelve kernels perform worse than MDNs with five or less kernels (see Figures 4.13, 4.10 and 4.9). Why is this so? The aim of any model is to describe the underlying data generator and not the noise on the targets. The MDNs with



(a) (b)

Figure 4.13: Model performance, comparing networks which take incidence angle as an input, by the number of kernels.

twelve kernels are complex enough to model some of the noise on the targets, and over-fit the data. The model then is not a general description of the underlying data generator describing the mapping $\sigma^o \rightarrow (u, v)$ but a specific description of the target data, including the noise. A good example is shown by the plots in Figure 4.14 where the probability distribution has several strong peaks, reflecting the distribution of the target data. Comparing the remaining configurations, performance is best for the MDN with five kernels for both MLP configurations of thirty five and fifty units in the hidden layer.



(a) (b)

Figure 4.14: An example of over-fitting in $p(u, v \mid \sigma^o)$ space.

Comparing the hybrid and two free kernel models, it is clear that the free kernel model performs better than the hybrid model. Again, this can be explained by the reduced flexibility of the hybrid model, but does suggest that the probability distribution is generally bi–modal though the modes are not exactly diametrically opposite one another or are non-Gaussian.

For the networks trained for each trace the MDN with four kernels and twenty hidden units in the MLP is the best model, and is denoted MDN_{trace}. For networks trained with incidence angle as an input, the best MDN has five kernels and fifty hidden units in the MLP, and is denoted $MDN_{incidence}$.

4.5 Comparison with other neural network methods

Cornford *et al.* (1999) give figures for their neural network approach for predicting wind vectors, based on ERS-1 satellite data. They solve the inverse problem by training distinct networks with incidence angle as an input to model wind speed and wind direction. Considering their models on an individual basis, they report that the correct solution within 20° is obtained more than 70% of the time for the first two aliases, which is similar to the results obtained with the MDNs thus far. They improved the performance by creating a committee of networks and then obtained the correct solution to within 20° roughly 75% of the time (averaged over the swathe). For wind speed they obtained an RMS error of 1.8 ms^{-1} which is within the design specifications of the instrument (2.0 ms^{-1}). In comparison, the networks trained in so far have a wind speed RMS error of 2.0 ms^{-1} when averaged over the swathe.

Richaume *et al.* (2000) give results for their neural network approach to wind vector retrieval based on ERS-1 data (following an initial study by Thiria *et al.* (1993) which was based on simulated scatterometer data). For each trace two networks are trained, one to model wind speed, denoted $S\text{-}NN_i$ (where i corresponds to the trace), and one to model wind direction, denoted $A\text{-}NN_i$. Results are quoted for predicted wind speed bias and RMS error as well as wind direction performance @ 20°. The wind direction results are quoted for the first, second, third and forth alias predictions. The second alias method is equivalent to the 'perfect' ambiguity removal method used in this project, and provides a means of comparison.

Table F.8 shows the results for *perf.* @ 20° of MDN_{trace}, $MDN_{incidence}$, $A\text{-}NN_i$. The $A\text{-}NN_i$ neural network performs better than both MDN networks when predicting direction to the second alias. However, we must note that the inputs to the $A\text{-}NN_i$ network also contain spatial information which gives additional disambiguation skill to the $A\text{-}NN_i$ networks. The MDN networks are purely local models, having no disambiguation skill whatsoever, inverting each scatterometer measurement on a per-cell basis.

Table F.9 presents the wind speed bias and RMS error results. Comparing the biases it is interesting to note that $A\text{-}NN_i$ has only negatively biased results, whereas the MDN models have both positively and negatively biased networks and so are less biased over the whole swathe. The RMS error results show that $A\text{-}NN_i$, performs within the instrument specification of 2 ms^{-1}, whereas $MDN_{incidence}$ and MDN_{trace} both fall outside the measurement specification for several of the middle

traces.

The superior performance of $S\text{-}NN_i$ and $A\text{-}NN_i$ may be attributed to:

- Larger data sets (the training set contains 24,000 pairs and the test set 5,000 pairs). Large data sets help to regularise the network during training, making it less susceptible to outliers in the data set.
- The spatial information presented on the inputs may provide extra disambiguation skill. The MDNs are trained with input from a single measurement.
- The range of wind speeds used by Richaume *et al.* (2000) is slightly greater, with a wind speed range of $[3.5\ ms^{-1}, 25.0\ ms^{-1}]$. There are likely to be more training patterns in the wind speed ranges that are more difficult to learn.
- The MDN networks may be getting stuck in local minima on the error surface during training.

4.6 Conclusions

It has been shown that MDNs model the mapping from scatterometer space directly into wind vector component space with a high degree of success. The probability distribution $p(u, v \mid \boldsymbol{\sigma}^o)$ is generally bi–modal, but the noise on the targets is more complex than the spherically symmetric Gaussian assumption that was first made. However, it is not clear whether this is a result of the training data or a true reflection of the function $p(u, v|\boldsymbol{\sigma}^o)$. It has also been shown that the mapping $(\boldsymbol{\sigma}^o, \theta) \rightarrow (u, v)$ can be modelled as accurately as models trained by trace. The best MDNs do not perform as well some other neural network methods, but they do have the advantage of modelling $p(u, v \mid \boldsymbol{\sigma}^o)$ directly. Further questions have been posed concerning the complexity of the mapping $\boldsymbol{\sigma}^o \rightarrow (u, v)$ and the complexity of $p(u, v \mid \boldsymbol{\sigma}^o)$.

- Investigating the mapping $\boldsymbol{\sigma}^o \rightarrow (u, v)$ it has been shown that an MDN that has an MLP with twenty units in the hidden layer, and a GMM with four kernels, successfully maps this relation. A more complicated model describing the mapping $(\boldsymbol{\sigma}^o, \theta) \rightarrow (u, v)$ was considered. This mapping is modelled using an MDN with an MLP with thirty five units in the hidden layer and four kernels in the GMM.
- Considering $p(u, v \mid \boldsymbol{\sigma}^o)$, the hybrid MDN configuration yields similar results to an MDN with two free centres, and shows that the conditional probability distribution is generally bi–modal with the two modes positioned diametrically opposite one another. The distribution of the noise on the wind vectors has been shown to be more complex than the spherically symmetrical Gaussian noise model originally assumed. This is shown by the MDNs which have the ability to model more complex (yet still predominantly bi–modal) distributions using four, five and twelve kernels. While the distribution is still generally bi–modal, it is heavier tailed than the Gaussian distribution assumed, and is not always spherical or symmetric.

- Other work in the field solves the inverse problem by directly modelling wind speed and wind direction with two separate models. The MDNs are similar in performance to other methods that model each cell individually (Cornford *et al.*, 1999). These methods however do not compare as favourably with those methods which take spatial information surrounding the cell of interest as part of their inputs (Richaume *et al.*, 2000).

The final conclusions are that mixture density networks provide a principled framework within which to model wind vectors directly from satellite scatterometer data, and the quality of the results provide an encouraging path of investigation for novel disambiguation techniques.

In the remaining sections of this chapter further models are trained using the YR2SWATHE data sets.

4.7 Further Experiments: Finding the definitive MDN for the global model

The ERS-1 data sets, YR1TRACE and YR1SWATHE, have been used to establish that MDNs can be used to model the inverse mapping $\sigma^o \rightarrow (u, v)$.

Using the ERS-2 data set, YR2SWATHE, further MDN experiments are discussed in the next section to build on the results of the exploratory experiments. The experiments investigate and establish a model over the whole swathe, which is used in the global wind field model (Section 1.3, equation (1.6)) and show how stringent pre-processing and outlier removal (see Section 2.2.1) on the training and validation data sets can improve model performance.

4.7.1 Extending the hybrid model

In order to investigate if a hybrid model with two pairs of diametrically opposed kernels will adequately model the mapping $\sigma^o \rightarrow (u, v)$ we extend the one kernel–pair hybrid model, (3.23), to the general case of l kernel–pairs. We let $m = 2l$, the total number of kernels in the GMM. We re-write the error function of the MDN, the negative log likelihood (3.13), as:

$$E(\boldsymbol{t}|\boldsymbol{x}) = \sum_n E(\boldsymbol{t}_n|\boldsymbol{x}_n) = -\sum_n \ln\left\{\sum_{j=0}^{(l-1)}\sum_{k=1}^{l} \alpha_{(2j+k)}(\boldsymbol{x}_n)\phi_{(2j+k)}(\boldsymbol{t}_n|\boldsymbol{x}_n)\right\}, \tag{4.1}$$

and

$$\phi_{(2j+k)}(\boldsymbol{t}_n|\boldsymbol{x}_n) = \frac{1}{(2\pi)^{\frac{c}{2}}\sigma_k^c(\boldsymbol{x}_n)} \exp\left(-\frac{\|\mathbf{t}_n - (-1)^j\boldsymbol{\mu}_k(\boldsymbol{x}_n)\|^2}{2\sigma_k^2(\boldsymbol{x}_n)}\right). \tag{4.2}$$

The $(-1)^j$ in the exponential of (4.2) gives the required switching for the centres, or means, of the kernels to be diametrically opposite in (u, v) space. Using the techniques described in Appendix B the gradient of the negative log likelihood, (4.1), was derived. For notational convenience, the posterior probability $\pi_{(2j+k)}$ is defined as:

$$\pi_{(2j+k)} = \frac{\alpha_{(2j+k)}\phi_{(2j+k)}}{\sum_{p=0}^{(l-1)}\sum_{q=1}^{l}\alpha_{(2p+q)}\phi_{(2p+q)}}. \tag{4.3}$$

Then the derivatives of the error function with respect to the network outputs are:

$$\frac{\partial E_n}{\partial z^{\alpha}_{(j,k)}} = \alpha_{(2j+k)} - \pi_{(2j+k)}, \tag{4.4}$$

$$\frac{\partial E_n}{\partial z^{\sigma^2}_{(j,k)}} = -\pi_{(2j+k)}\left\{\frac{\|\mathbf{t}_n - (-1)^j\boldsymbol{\mu}_k\|^2}{2\sigma_k^2} - c\right\}, \tag{4.5}$$

$$\frac{\partial E_n}{\partial z^{\mu}_{(j,k,l)}} = \pi_{(2j+k)}\left\{\frac{(-1)^j\mu_{k,l} - t_l}{\sigma_k^2}\right\}. \tag{4.6}$$

We also note that, because of the symmetry of the model, these equations may be derived by inspection from (3.18), (3.19) and (3.20).

4.7.2 Training Configurations

To find a definitive MDN configuration that takes incidence angle as an input, several configurations were trained and tested with the YR2SWATHE data sets. For the GMM the configurations were two and four centres, both hybrid and conventional architectures. For the MLP there were ten to forty, in steps of five, hidden units in the hidden layer of the network. The models were regularised using an early stopping technique; every two hundred epochs, training was paused and the training and validation errors were computed and recorded. For each configuration, the model with the minimum error on the validation set was chosen as the model with the best generalisation performance. The error statistics were computed on a test set, which was independent of the training and validation sets, and so the results are unbiased with respect to the training and validation data.

4.7.3 Committees of MDNs

So far many models have been generated and, using early stopping techniques, all but the 'best' model (by generalisation performance on the validation data set) are thrown away; the computational effort required to train the thrown away models essentially being discarded. Also, the 'best' model is chosen by its generalisation performance on the validation set, which has a random component due to the noise on the data, implying that the chosen model may not give the best performance on the test set (Bishop, 1995).

One way of using the discarded models and possibly improving generalisation performance is to combine all the 'best' trained models in a committee of models. Committees can sometimes give better generalisation performance than that of the best single network in the committee (Bishop, 1995). This has been demonstrated by Cornford *et al.* (1999) where committees where used to improve wind direction estimation performance. Hence, we are motivated in this work to apply committees in order to see if they can improve the directional estimates of the MDNs.

For notational convenience we consider networks with a single output, $y(\boldsymbol{x})$, and inputs $\boldsymbol{x}$. We then write $y(\boldsymbol{x})$ as the sum of the true regression we are approximating, $h(\boldsymbol{x})$, and an error, ϵ:

$$y(\boldsymbol{x}) = h(\boldsymbol{x}) + \epsilon. \tag{4.7}$$

Now suppose that L models that have been trained. They may be different models, the same model trained on different data sets or the same model with different architectures trained on the same data set. The models are indexed by $i = 1, 2, ..., L$. Then the average sum of squares error of each model, $y_i(\boldsymbol{x})$, is:

$$E_i = \mathrm{E}\left[\{y_i(\boldsymbol{x}) - h_i(\boldsymbol{x})\}^2\right] = \mathrm{E}[\epsilon_i^2], \tag{4.8}$$

where $\mathrm{E}[\cdot]$ is the expectation operator. From (4.8) the average error for the models acting individually is:

$$E_{av} = \frac{1}{L}\sum_{i=1}^{L} E_i = \frac{1}{L}\sum_{i=1}^{L} \mathrm{E}[\epsilon_i^2]. \tag{4.9}$$

We now define a simple committee of networks where each model is equally weighted within the committee, the output of the committee is therefore the average of the outputs of the L individual networks:

$$y_{com}(\boldsymbol{x}) = \frac{1}{L}\sum_{i=1}^{L} y_i(\boldsymbol{x}). \tag{4.10}$$

The error due to the committee is:

$$E_{com} = \mathrm{E}\left[\left(\frac{1}{L}\sum_{i=1}^{L}\epsilon_i\right)^2\right] = \frac{1}{L^2}\sum_{i=1}^{L}\mathrm{E}[\epsilon_i^2] \tag{4.11}$$

where we assume errors have zero mean, so that $\mathrm{E}[\epsilon_i] = 0$, and errors between models are uncorrelated, so that $\mathrm{E}[\epsilon_i\epsilon_j] = 0, \text{for } i \neq j$. By comparing with the error for individual models, (4.9), with that of the committee, (4.11):

$$E_{com} = \frac{1}{L^2}\sum_{i=1}^{L}\mathrm{E}[\epsilon_i^2] = \frac{1}{L}E_{av}, \tag{4.12}$$

we reveal that the error due to the committee is reduced by a factor L. Hence, for little extra computation effort we have a method for drastically improving the generalisation performance. However this is the best improvement we can expect; in practice, the models in the committee are rarely independent.

Given that we have many trained MDN models, we investigate possibility of improving generalisation performance by defining a committee of mixture density networks:

$$p(\boldsymbol{t}|\boldsymbol{x})_{com} = \frac{1}{L}\sum_{i=1}^{L} p_i(\boldsymbol{t}|\boldsymbol{x}), \tag{4.13}$$

expanded to,

$$p(\boldsymbol{t}|\boldsymbol{x})_{com} = \frac{1}{L}\sum_{i=1}^{L}\sum_{j=1}^{M}\alpha_{ij}\phi_{ij}(\boldsymbol{t}|\boldsymbol{x}). \tag{4.14}$$

We can rewrite (4.14) for the wind vector model in geophysical parameters:

$$p(u, v \mid \boldsymbol{\sigma}^o)_{com} = \frac{1}{L}\sum_{i=1}^{L}\sum_{j=1}^{M}\alpha_{ij}(\boldsymbol{\sigma}^o)\phi_{ij}((u,v)|\boldsymbol{\sigma}^o). \tag{4.15}$$

The configurations of the committees are decided after the individual models have been trained; these are discussed in the next section.

4.8 Results

There were four configurations of the GMM considered: conventional two kernel, hybrid two kernel, four kernel and hybrid four kernel. For each GMM configuration seven networks were trained, each with different numbers of hidden units in the MLP. Each network had the same seed in the random number generator when initialised. Additional networks were trained with the same configuration (four conventional kernels and twenty five hidden units) but with different seeds in the random number generator to ensure that the global minima was being found during training. The best network from each configuration was chosen by its performance on a validation data set. The reported results are

computed using a test set, which was not used when choosing the model complexity and so the results are unbiased with respect to the training and validation data sets.

The results for individual network and committee of networks are presented in tabular form using a combination of summary statistics commonly used in the meteorological community and some standard statistical measures: the average negative log–likelihood per pattern; the vector RMS error; the *Figure of Merit*; the *perf.* @ 20° for the most probable wind vector; the *perf.* @ 20° for the first and second most probable wind vector; wind speed bias; wind speed standard deviation; wind direction bias and wind direction standard deviation.

The results are based on wind vector predictions derived from a simple ambiguity removal algorithm. The two most likely wind vectors are inferred from the MDN and then compared to the target wind vectors; for each pattern, the position of the two most probable modes, found by a SCG optimisation starting from the positions of each kernel, are inferred as the two most likely wind vectors. The predicted wind vector closest to the target wind vector is chosen as the disambiguated wind vector, which essentially measures the best possible results that can be achieved after disambiguation. The summary methods are then applied.

Individual models

Tables G.1, G.2, G.3 and G.4 present the summary results for networks trained with conventional two kernels, hybrid two kernels, conventional four kernels and hybrid four kernels trained to four thousand epochs respectively. Generally, the performance of the hybrid models is comparable with their equivalent conventionally configured model. The models with two kernels do not perform as well as the models with four kernels.

Comparing the results of the conventional and hybrid models with two kernels (Tables G.1, G.2) we note that performance on all measures is similar, and that the hybrid model does better, by 1%, than the conventional model in the *perf.* @ 20° for the first most probable mode. However the standard deviation of wind direction for all the hybrid models is always greater than that for the conventional model. The best conventional model is generally better than the best hybrid model, both of which have forty hidden units in the MLP.

Similarly, the generalisation performance of the hybrid and conventional models with four kernels are comparable. It is difficult to choose the 'best' model from these results. The best conventional model has thirty units in the hidden layer of the MLP, whilst the best hybrid model has thirty five units in the hidden layer of the MLP. The best model, considering the FoM, is the conventional model with thirty hidden units in the MLP.

The performance of the models with four kernels is better than those with two. In all cases models with two kernels have worse results than an equivalent model with four kernels; decreased performance on the bias and standard deviation measures affects the FoM measure. The worst four kernel configuration is better than the best two kernel configuration for FoM.

To ensure that the model parameters were in fact converging to a global minimum, an MDN with

25 hidden units in the MLP and four kernels in the GMM was trained[1], starting from ten different random seeds. The results, presented in Table G.7, show two outliers, and that generally the models are converging to a global minimum.

We have established that the models with four kernels have better generalisation performance than those with two, and no further investigation was pursued for models with two kernels. However, for models with four kernels further trained was undertaken. This was motivated by the fact that after 4000 epochs the value of the validation error had not risen, although had significantly flattened out. Therefore the models with four kernels were trained for a further 4000 epochs, taking the total training duration to 8000 epochs. The generalisation performance of the models on the test set is shown in Table G.5 and Table G.6. Generally the all the models have similar performance, although it is fair to say that the better models have either 25, 30 or 35 hidden units in the MLP. The absolute best model though is the conventional model with 30 hidden units in the MLP, having the best *perf.* @ 20°, vector RMS, speed bias and speed standard deviation. In summary, training to 8000 epochs caused the model performance of each configuration to converge and did not significantly improve on the performance of the models after 4000 epochs.

Committees of MDNs

Four committee configurations were considered. The choice of committee members was based on the results of performance on the validation set (see Table G.9; there is very little to choose between the models in terms of performance). The committee members are shown in table Table 4.1. The performance of the committees on the test set is shown in Table G.8 and show that the performance is very similar for all committees. The performance of the committees is as good as, but not better than, the individual models that make up the committees.

Committee	Four kernels with hidden units:				
	15	20	25	30	40
a	√	√	√	√	
b	√	√		√	
c		√		√	
d		√		√	√

Table 4.1: Configurations of committees.

Performance on the YR1TRACE data set

Given that we are motivated to generate a model that is trained over all traces, results are presented for generalisation performance on the YR1TRACE test set for the best conventional and hybrid model

[1] This is the best model chosen from the validation data set.

with four kernels chosen from by their performance on the validation set. The results are presented by trace in Table G.11 and Table G.12.

Generally the *FoM* results are better for both the conventional and hybrid models trained over all traces when compared to the models trained on each trace (see Table F.1). The models trained on the YR2SWATHE data set have lower vector RMS errors, biases and standard deviations on wind speed and direction. The exception is for trace 9 where the models trained by trace have a marginally better *FoM* result. The vector RMS error is generally lower for the conventional model than the models trained by trace (see Table F.2), and similar for the hybrid model.

If we consider the *perf.* @ 20° results in Table F.3, which are equivalent to the *perf.* @ 20° for the first two modes in Table G.11 and Table G.12, column 5, we see that the results are comparable for traces 2–9, but the models trained per trace are better by about 5% for traces 0 and 1.

Performance on the independent UKMO data set

The final set of results compare the two best models with four kernels against the current operational model, CMOD4, on an independent test set generated from the UK meteorological office, UKMO data; the results are shown in Table 4.2. These results show that the models developed in this thesis out perform the operational model for obtained the *perf.* @ 20° on the first solution, and marginally out perform on the *perf.* @ 20° with two solutions. However, we note that the direction and speed standard deviation and the direction bias for CMOD4 were better than the MDN models, although they are not practically significant. Also, on average per pattern, the conventional MDN model uses at least three times less floating point operations to retrieve a wind vector estimate than CMOD4.

model	vector RMS error	*FoM*	first	second	speed bias	speed standard deviation	direction bias	direction standard deviation
standard	**2.93**	**1.03**	0.47	**0.70**	**0.16**	1.84	1.29	28.95
hybrid	2.99	1.01	**0.48**	**0.70**	0.17	1.87	1.19	28.91
CMOD4	3.34	0.97	0.37	0.67	−0.55	**1.79**	**0.46**	**28.87**

Table 4.2: Comparing the standard MDN with the hybrid MDN, with four kernels: Error measures computed against the UKMO test data set.

4.9 Analysis of results of further experiments

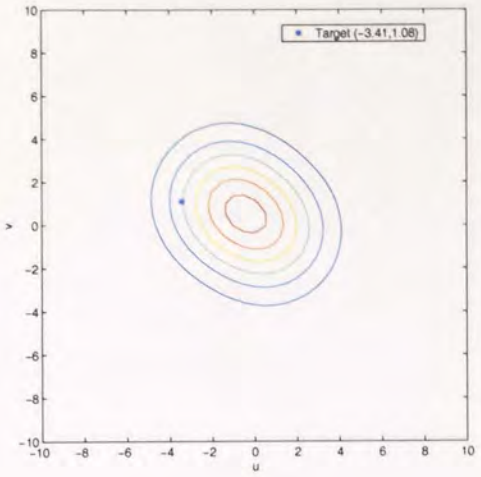
There are several results which are of interest and require explanation. The first is that the models with two kernels have a consistently lower performance than those with four kernels; the second is that the hybrid models perform as well as the equivalent conventional model; and finally, the third is the general performance of the individual models with four kernels when compared to the performance of

the committees of networks.

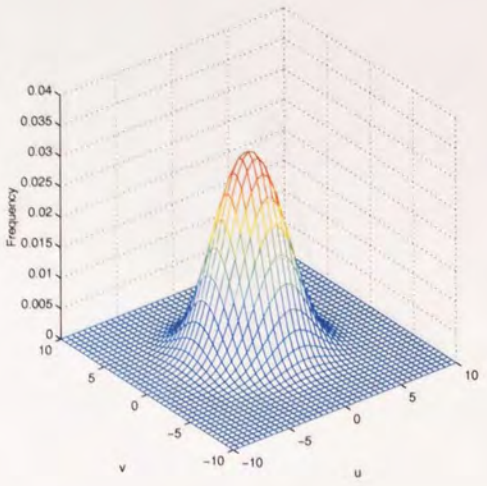
Four kernels vs two kernels

Previous work has shown that the inverse problem is predominantly bi–modal (Stoffelen and Anderson, 1997c). However we have seen from the results that a bi–modal model does not perform as well as a model with four kernels; why is this so? There are two reasons: the first is at low wind speed, less than 5 ms^{-1}, the returned scatterometer signal has a low signal to noise ratio making the resolution of wind *direction* very difficult (Offiler, 1994); the second is that in some places on the measurement manifold there are three or four possible wind vector solutions.

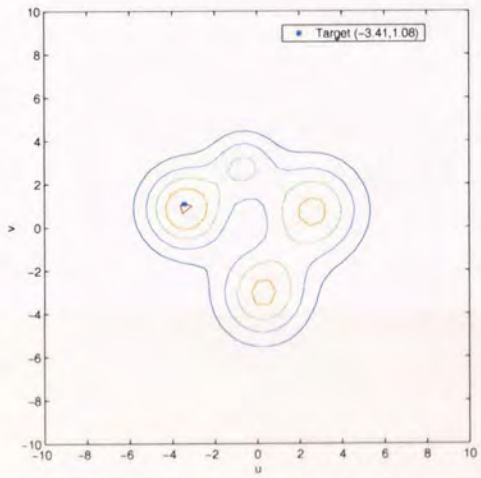
To illustrate how the models perform at low wind speed the density $p(u, v \mid \boldsymbol{\sigma}^o)$ is computed for a wind vector with a wind speed of 3.6 ms^{-1} with wind vector components $(-3.41, 1.08)$; the contour and mesh plots are given in Figure 4.15. Here we see that in plots (a) and (b) the model with two kernels is unable to model the general uncertainty in wind direction, and so chooses the best guess which is a unimodal density with sufficient variance to capture the target and with wind speed close to 0 ms^{-1}. The model with four kernels, plots (c) and (d), has greater flexibility and is able to model the uncertainty in wind direction, whilst accurately modelling the wind speed. Hence, the model with two kernels is too stiff and inflexible to model the low wind speed data. A further illustration is provided in Figure 4.16.



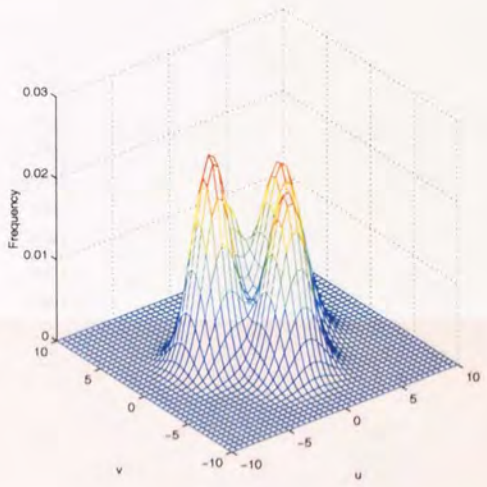
(a) Two kernels



(b) Two kernels

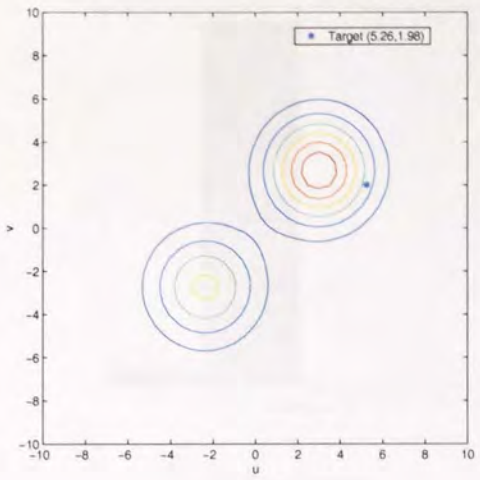


(c) Four kernels

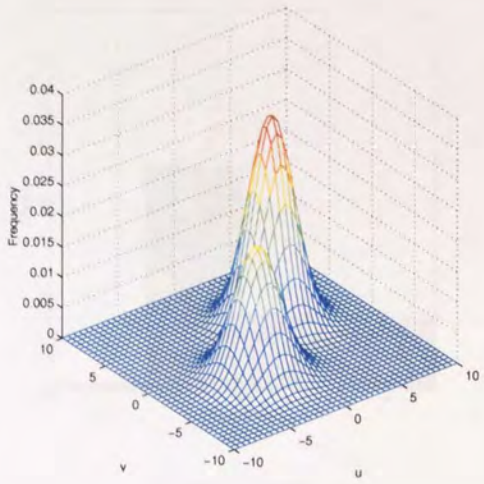


(d) Four kernels

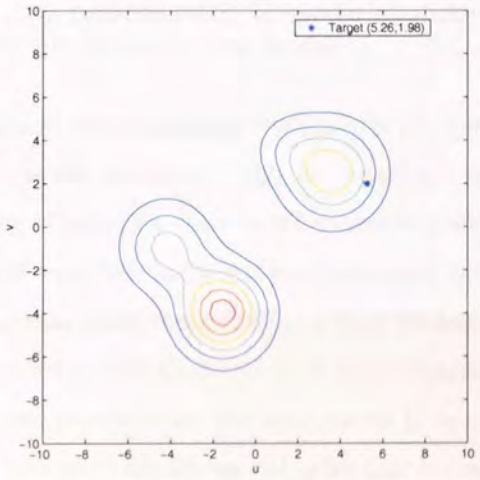
Figure 4.15: Comparing how models with two and four kernels cope with low wind speeds, in this case 3.6 ms^{-1}, the asterix denotes the target wind vector. In plots (a) and (b) the model with two kernels has insufficient flexibility to model the uncertainty in wind direction at low wind speed and puts a unimodal estimate near 0 ms^{-1}. The model with four kernels has sufficient flexibility to model the uncertainty in wind direction.



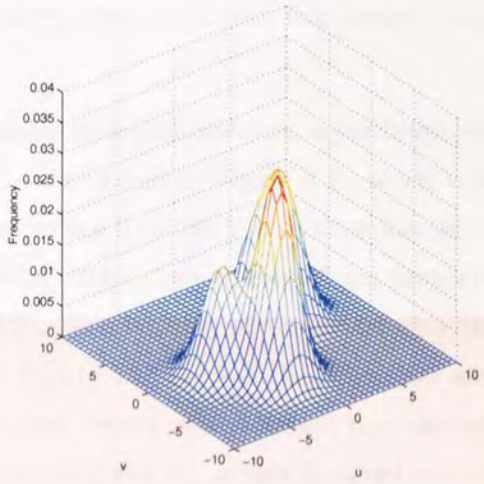
(a) Two kernels



(b) Two kernels

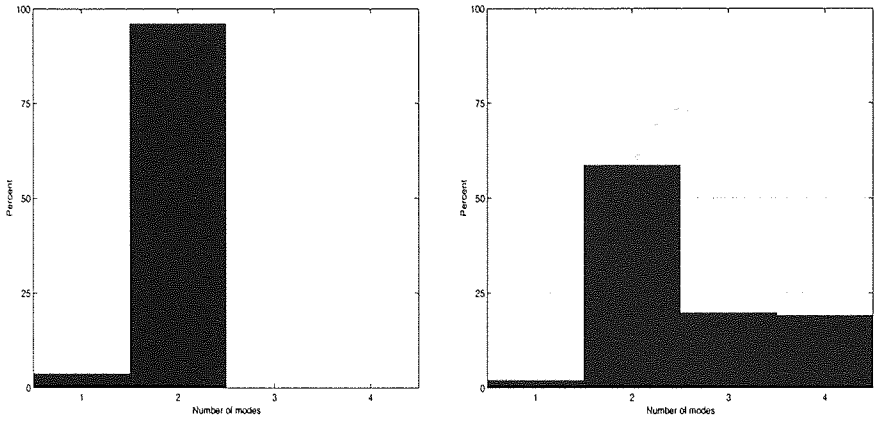


(c) Four kernels



(d) Four kernels

Figure 4.16: Comparing how models with two and four kernels cope with low wind speeds, in this case 3.9 ms^{-1}, the asterix denotes the target wind vector. In plots (a) and (b) the model with two kernels is able to model the wind direction, but requires kernels with larger variance to capture the target when compared to the model with four kernels. The model with four kernels has sufficient flexibility to model the uncertainty in wind direction, which allows kernels with lower variance and therefore better captures the data.

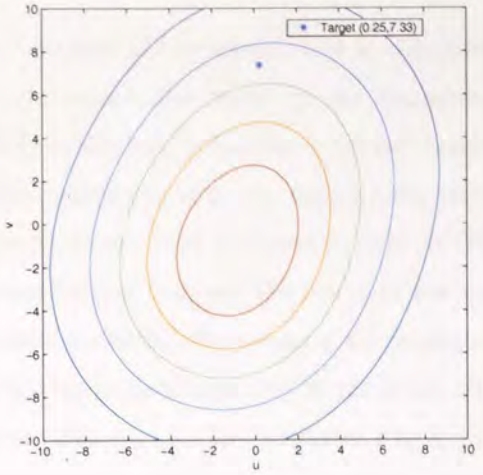


(a) Two kernels (b) Four kernels

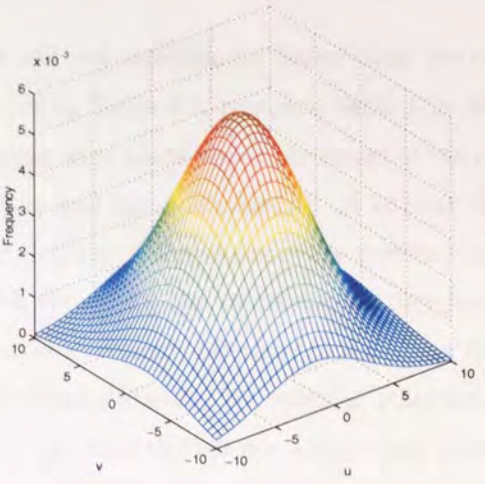
Figure 4.17: Histograms showing the frequency of one, two, three and four mode solutions for the test data set YR2SWATHE for models with two and four kernels. As expected the model with two kernels, plot (a), has predominantly bi–modal solutions, whereas the model with four kernels, plot (b), has solutions with three and four modes.

Although the mapping $\boldsymbol{\sigma}^o \rightarrow (u, v)$ is predominantly bi–modal there are, nevertheless, some three and four mode solutions. This is illustrated by the histograms in Figure 4.17 which compare the frequency of one, two, three and four mode solutions derived from the YR2SWATHE test set. Although the model with four kernels is predominantly bi–modal (59%), there is a significant proportion of the solutions that have three (20%) or four modes (19%). The inability of the model with two kernels to cope with a four mode solution is illustrated in Figure 4.18 (a) and (b), where we see that the two kernels approximate the four modes by a unimodal density with a spread that captures all the modes. The more flexible model, with four kernels, plots (c) and (d), is able to model each mode with a kernel.

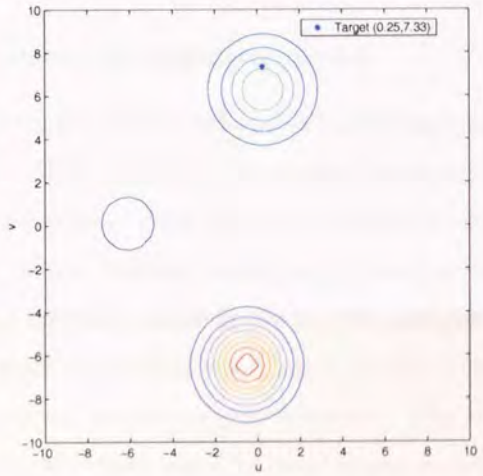
The combination of modelling low wind speed and direction and three and four mode solutions with a unimodal density centred close to the $(0, 0)$ point in (u, v) space will cause large vector RMS errors and larger standard deviations for wind direction for the models with two kernels. These in turn will contribute to a degraded performance when compared to models with four kernels, and is summarised in the FoM statistic. We can see this by comparing statistics of Table G.1 with Table G.3 and Table G.2 with Table G.4, which confirms that standard deviation measures for wind speed and direction are greater for models with two kernels than those with four and that the vector RMS error is higher for the models with two kernels.



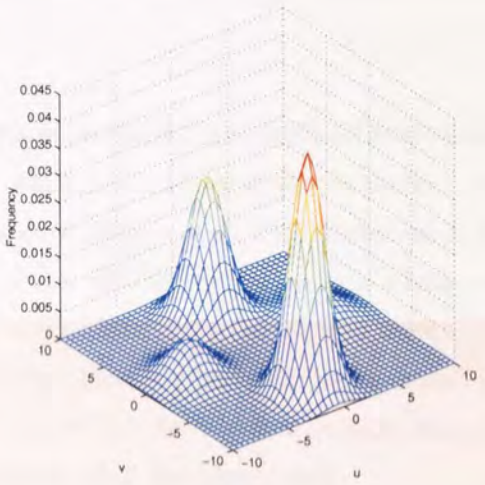
(a) Two kernels



(b) Two kernels



(c) Four kernels



(d) Four kernels

Figure 4.18: These plots demonstrate how the model with two kernels is insufficiently flexible to cope with four mode solutions. In plots (a) and (b) the best approximation is a unimodal probability density, with a mode close to $(0, 0)$ and a spread that captures the four modes. The plots (c) and (d) show that the model with four kernels is able to model each mode by assigning a kernel to each one.

Conventional vs hybrid configurations

The results presented in Tables G.1, G.3, G.2, G.4, G.5 and G.6 suggest that the hybrid models are a good model for the real world physical process. The hybrid case may be thought of constraining model flexibility or as a regularisation method, helping to decrease the chance of over–fitting. However this not the case for these models and this is primarily due to the size of the training data set from YR2SWATHE. This correlates with the experiments completed with the YR1TRACE and YR1SWATHE data sets.

Committees vs individual networks

The performance of the committees is comparable but not significantly better than the individual model components that make up the committees (using Table 4.1, compare Table G.8 with Table G.5). Why is this so? In Section 4.7.3 two assumptions were made about the errors of the individual committee members: that the errors have zero mean and are uncorrelated. It is clear that these assumptions do not hold for these models. In Table G.5, the performance of the models suggest that there is correlation between the errors of the models chosen for the committee as they have similar performance statistics. For instance, for committee a the maximum difference in the vector RMS error is 0.04; similar comparisons can be made for all the other statistics. By choosing statistics with the maximum variance, such as committee c or d, we still find that there is not a significant improvement in committee performance over individual performance. One point to note is that the performance of the committees is consistent; the variance between the statistics is less than the individual models, although these differences are generally in the second decimal place, which in practice is negligible.

Conclusions: the definitive model

We have continued the exploratory experiments set out in Section 4.2, and provided conclusive evidence that the MDN provides a principled framework in which to model the inversion problem across the complete swathe of data. This is in contrast to other work, (Thiria *et al.*, 1993; Richaume *et al.*, 2000), where a neural network model is required for each trace. MDNs trained over the whole swathe, by including incidence angle in the inputs, perform as well as MDNs trained on each trace.

We have shown that the solution to this inversion problem is predominantly bi–modal, but there are also three and four mode solutions. The MDN with four kernels models these mappings, and copes with low wind speed retrieval where uncertainty in wind direction is high.

The MDN model has been shown to be comparable with the current operational model, CMOD4, and better on key performance measures. The MDN model is also computationally more efficient than CMOD4 and hence may well provide an alternative method for wind vector retrieval in the initial stages of an NWP simulation.

And finally for the next part of this thesis, the definitive model for use in the global wind field model has four kernels in the GMM, twenty-five units in the hidden layer of the MLP, and has a conventional MDN configuration – this is the model with the best performance based on the YR2SWATHE validation data set.

4.10 Chapter Review

In this chapter we have established that MDNs model the inverse mapping from $\boldsymbol{\sigma}^o \rightarrow (u, v)$ and are comparable to the current operational model CMOD4. We have also established that the complexity of the $p(u, v \mid \boldsymbol{\sigma}^o)$ is predominantly bi–modal, although there are also three and four mode solutions

which an MDN with four kernels can adequately map. Finally, we have a direct inverse model to plug into the global wind field model.

Chapter 5

Ambiguity Removal

Key word summary

Ambiguity	**Autonomous**
Wind field retrieval	**Non-autonomous**
NWP	**Scatterometer**

5.1 Introduction

One aim of scatterometer data processing by some meteorologists is an autonomous method for wind field retrieval. However, in practice this is very difficult to achieve due to the directional ambiguity of the wind vectors; the current practice is a non-autonomous method which requires reference to a background wind field that has been generated by a physics based meteorological model of the atmosphere (Stoffelen and Anderson, 1997a). This is a two stage process, in the first stage, which is the disambiguation process, a 'rough' wind field is generated by choosing the retrieved wind vectors which are closest to the background wind field; in the second stage a filter is passed over the retrieved wind field to smooth non-meteorological structures and increase (meteorological) consistency.

In this chapter we take two approaches to retrieving wind fields using the Bayesian framework,

$$p(U,V|\Sigma) \propto \left(\prod_i \frac{p(u_i, v_i \mid \boldsymbol{\sigma}_i^o)}{p(u_i, v_i)} \right) p(U,V), \tag{5.1}$$

(1.6), introduced in Chapter 1. The first method is similar to the non-autonomous method of Stoffelen and Anderson (1997a) and uses the NWP target wind fields to aid disambiguation whilst the second method is an autonomous approach where predictions are based on the wind field with the maximum *a posteriori* probability under the distribution (5.1).

The prior model, $p(U,V)$, constrains the local solutions to produce consistent and meteorologically plausible winds. It is a vector Gaussian process based wind field model that is tuned using NWP

data. This ensures that the wind field model is representative of NWP wind fields, which are the best *a priori* estimates of the true wind fields, a more detailed explanation of the parameterisation of the prior model is given in Appendix A. Hence, the probability of a wind field, (U, V), under prior model is given by

$$p(U,V) = \frac{1}{(2\pi)^{\frac{n}{2}} \det(K_{uv})^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\begin{bmatrix} U \\ V \end{bmatrix}^T K_{uv}^{-1} \begin{bmatrix} U \\ V \end{bmatrix}\right), \tag{5.2}$$

where K is the joint covariance matrix for (U, V),

$$K = \begin{pmatrix} C_{uu} & C_{uv} \\ C_{vu} & C_{vv} \end{pmatrix}. \tag{5.3}$$

The C_{uu} are the variances and cross-correlations of the U components, C_{vv} are the variances and cross-correlations of the V components and, $C_{uv} = C_{vu}$ represent the cross-correlation between the U and V components.

The structure of the covariance matrix makes the calculation of the local unconditional probability, $p(u_i, v_i)$, straight forward. This is also a zero mean Gaussian,

$$p(u_i,v_i) = \frac{1}{(2\pi)^{\frac{n}{2}} \det(\Sigma_{u_i v_i})^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\begin{bmatrix} u_i \\ v_i \end{bmatrix}^T \Sigma_{u_i v_i}^{-1} \begin{bmatrix} u_i \\ v_i \end{bmatrix}\right), \tag{5.4}$$

where $\Sigma_{u_i v_i}$is the covariance matrix of the local wind vectors (u_i, v_i),

$$\Sigma_{u_i v_i} = \begin{pmatrix} C_{u_i u_i} & C_{u_i v_i} \\ C_{v_i u_i} & C_{v_i v_i} \end{pmatrix}, \tag{5.5}$$

which is a marginalisation of (5.2).

5.2 Non-autonomous wind field retrieval

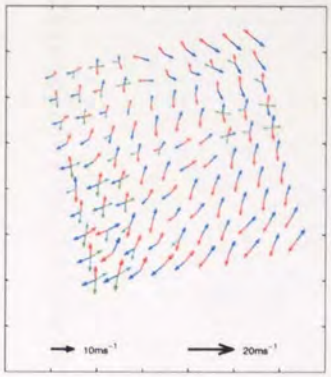
This method is applied to investigate the suitability of the model for wind field retrieval. We test the suitability of the MDNs as a local model and the method for combining the local models with the global prior under the scaled likelihood assumption (Nabney *et al.*, 2000).

5.2.1 Method

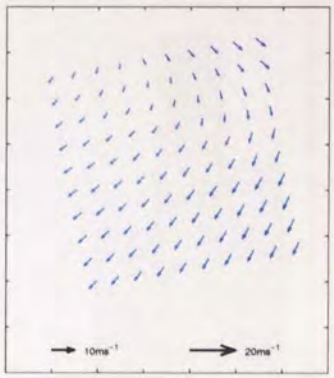
The non-autonomous retrieval of a wind field using (5.1) is a four stage process:

1. forward propagate the normalised, pre-processed scatterometer data through the mixture density network to generate the local conditional probability distribution of the wind vectors given the scatterometer data, $p(u, v \mid \boldsymbol{\sigma}^o)$, for each observation;

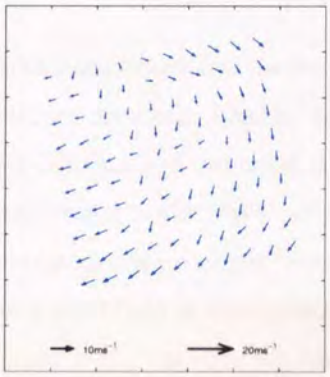
2. for each local cell, find all possible wind vector solutions from the mixture density network (generally there are 2 to 4 possible solutions in each cell);



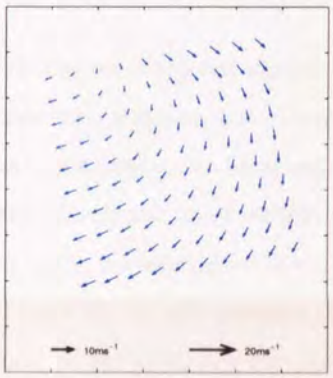
(a) Stage 2: The local wind vectors generated from the MDN.



(b) The reference NWP wind field.



(c) Stage 3: The disambiguated wind field.



(d) Stage 4: The smoothed, disambiguated wind field.

Figure 5.1: The non-autonomous wind field retrieval process.

3. generate an MDN first guess wind field, by choosing, on a cell by cell basis, the retrieved solution that is closest to the NWP reference – this stage is equivalent to the first stage of the method of Stoffelen and Anderson (1997a);

4. use the MDN first guess wind field as the starting point for a non–linear optimisation of the wind field model, (5.1), where the stopping point of the optimiser is the retrieved wind field of the process. This stage is equivalent to the smoothing stage of Stoffelen and Anderson (1997a), where for this model the Gaussian prior provides the global information for meteorological consistency.

An example of the process is given and illustrated in Figure 5.1. The wind vector solutions to each local cell are found by initialising a non-linear optimiser (in this case SCG from NETLAB) at the four modes of the GMM resulting from the forward propagation of the scatterometer data through the

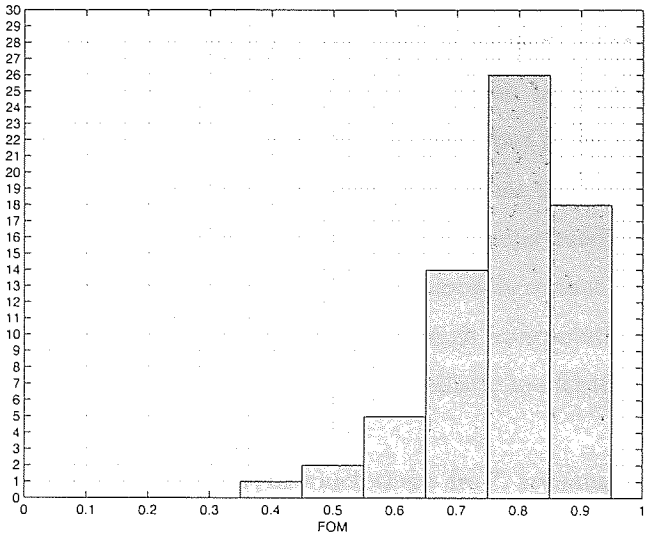


Figure 5.2: Histogram showing the distribution of the FOM of non-autonomously derived wind fields with an FOM < 1.

MDN. The four solutions are sorted and the unique modes recorded and stored. Each mode represents a local wind vector solution for a cell. Usually there are two solutions, but in some cases there are three or four, see Figure 5.1 (a). Each of the wind vectors identified in the local cell are then compared to the corresponding wind vector of the NWP wind field; the closest wind vector, by Euclidean distance, is chosen as the disambiguated wind vector, resulting in the un–smoothed first guess wind field, Figure 5.1 (c). The first guess wind field is then smoothed through an optimisation process, where the cost function of the optimiser is the negative log likelihood of the posterior distribution of the wind field model, (5.1),

$$E_{wf} = -\ln\left[\left(\prod_i \frac{p(u_i, v_i \mid \boldsymbol{\sigma}_i^o)}{p(u_i, v_i)}\right) p(U, V)\right] \tag{5.6}$$

$$= \sum_i \left\{\ln[p(u_i, v_i)] - \ln[p(u_i, v_i \mid \boldsymbol{\sigma}_i^o)]\right\} - \ln[p(U, V)], \tag{5.7}$$

the result of which is the retrieved wind field, Figure 5.1 (d).

5.2.2 Results

There were a total of 575 wind fields to analyse. Summary statistics were computed for the wind fields, comparing the retrieved wind field with the NWP target, these statistics were: vector RMS error, speed bias and standard deviation, direction bias and standard deviation, and the percentage of wind vectors within 20° of the target. These were combined into the FOM summary statistic.

Of the 575 wind fields, 509, representing 88.5%, were found to have an FOM value of greater the 1.0, indicating that the retrieved wind field was a good estimate of the target NWP wind field.

Of the 66 remaining wind fields the FOM values ranged between 0.478 and 0.998, and were distributed as shown in Figure 5.2.

The majority of wind fields have an FOM which is greater than 0.7 and this can be understood by comparing the visualisation of the NWP and retrieved wind fields. Typical examples are shown in Figure 5.3, where there is agreement about the wind direction between the target and retrieved wind field (that is the directional ambiguity has been resolved) but disagreement about the position of the meteorological features. The reason for this difference is that the NWP wind fields are derived by linear interpolation from a coarser model than the scatterometer readings. Therefore, the NWP model is less sensitive to features that are present in the atmosphere that are detectable by the finer resolution of the scatterometer measurements but not through interpolation. Hence, there are some features that are observed in the retrieval using the probabilistic model that are not observed in the NWP model.

The examples with FOM values below 0.7, of which there are 7, accounting for 1.2% of the test set, are shown in Figure 5.3, pair, (c) - (d) and Figure 5.4 (a) - (l). In Figure 5.4 we see that for NWP targets (a) and (g) the retrieved wind field reflects a move in the structure of the wind field which is attributed to the higher resolution of the scatterometer readings. For NWP targets (i) and (k), which represent low wind speed examples, the probabilistic model appears to over estimated the wind speed when compared to the targets. Finally, NWP targets (c) and (e) are examples of incorrectly retrieved wind fields due to some retrieved wind vectors being ambiguous in direction to their corresponding target wind vector.

For the first incorrect example, corresponding to Figure 5.4 (c), the retrieved wind vectors that do not correspond to the target are in a 7 by 4 grid referenced from the top right hand corner of the wind field. The initial MDN derived wind field, before smoothing, appears to be close to the NWP target, Figure 5.5 (a); it is during the optimisation process that the apparently incorrect wind field is retrieved. Figure 5.5 (b), which shows all the possible solutions from the MDNs, gives the explanation. The colour coded wind vectors in each cell represent the relative probability densities for each retrieved wind vector under the local GMM, where from the most to the least probable is coded as blue, red, cyan, and green. The vectors retrieved by the NWP comparison, for the 7 by 4 grid, are the least probable under the GMM (compare Figure 5.5 (a) and Figure 5.5 (b)); those returned from the optimisation are generally the most probable wind vectors (compare Figure 5.4 (d) and Figure 5.5 (b)). Therefore in this example the MDN has influenced the optimisation process away from the NWP initialised solution.

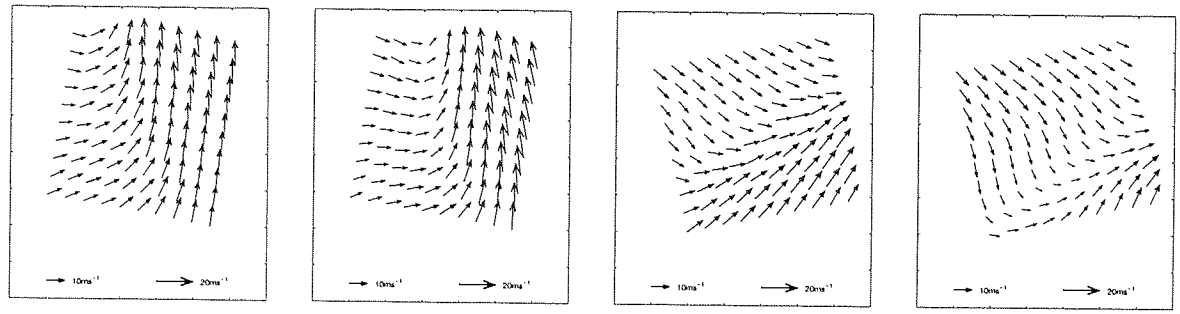
In the second incorrect example, Figure 5.4 (e), we see that the MDN retrieved wind vectors disagree with the NWP wind field, Figure 5.5. This is attributed to the finer resolution of the scatterometer, and suggests the NWP target, in this instance, is incorrect.

Finally to verify the retrieved wind fields, Dan Cornford, my associate supervisor and project expert meteorologist, has visually inspected each target and retrieved solution. He confirms that each wind field retrieved is meteorologically consistent, is of the correct direction and that 2 of the retrieved wind fields are 'incorrect' given the NWP target.

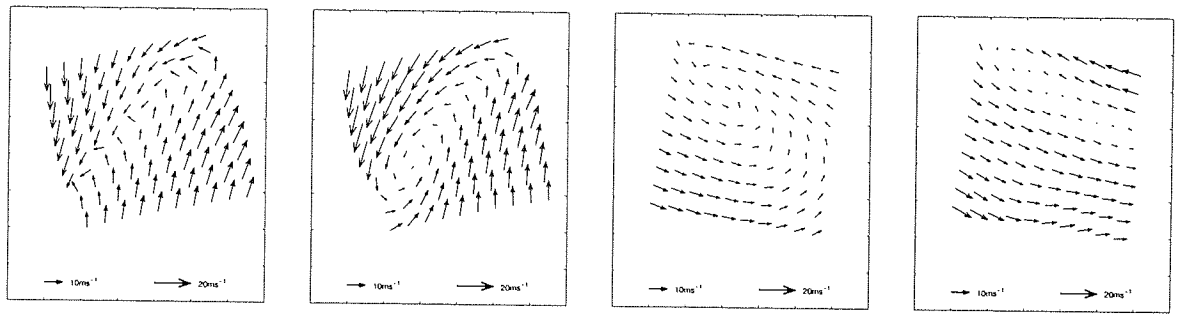
5.2.3 Conclusion

The results of the non-autonomous ambiguity removal shows two things: firstly, that the model itself, integrating the vector Gaussian process with the MDN likelihood model, works and produces valuable results and secondly, that, given an NWP initial wind field, this model can be used to retrieve wind field predictions from scatterometer data with 99.7% ($\frac{573}{575}$) accuracy.

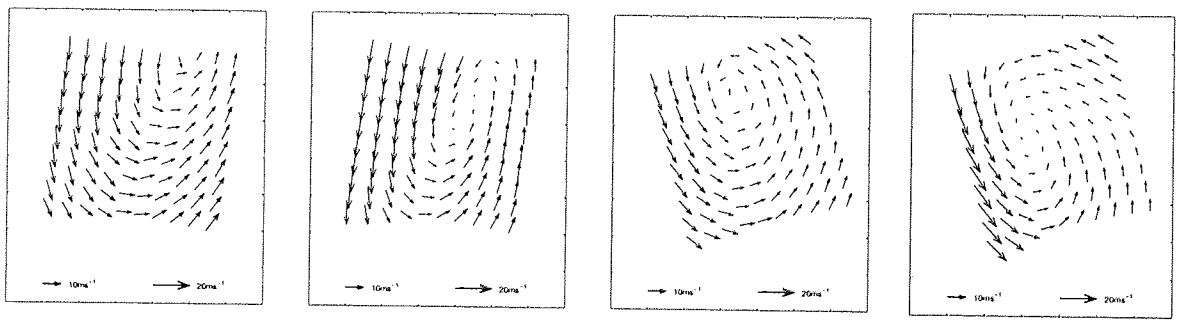
The next step is to try and retrieve wind fields autonomously, without reference to the NWP wind fields.



(a) NWP target. (b) FOM= 0.91 (c) NWP target. (d) FOM= 0.58

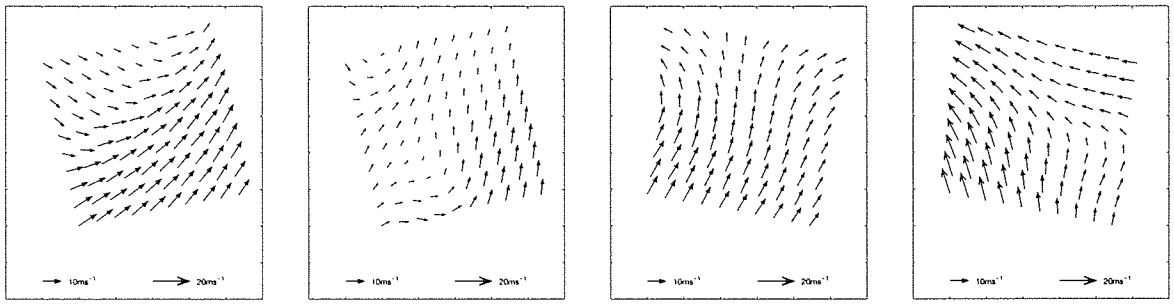


(e) NWP target. (f) FOM= 0.97 (g) NWP target. (h) FOM= 0.72

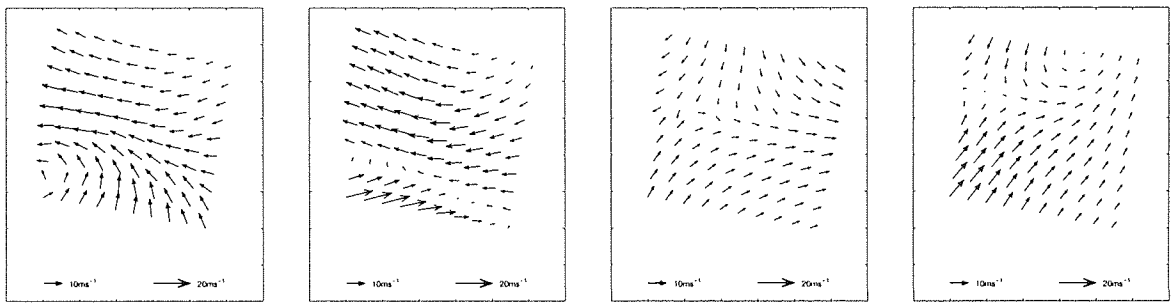


(i) NWP target. (j) FOM= 0.81 (k) NWP target. (l) FOM= 0.83

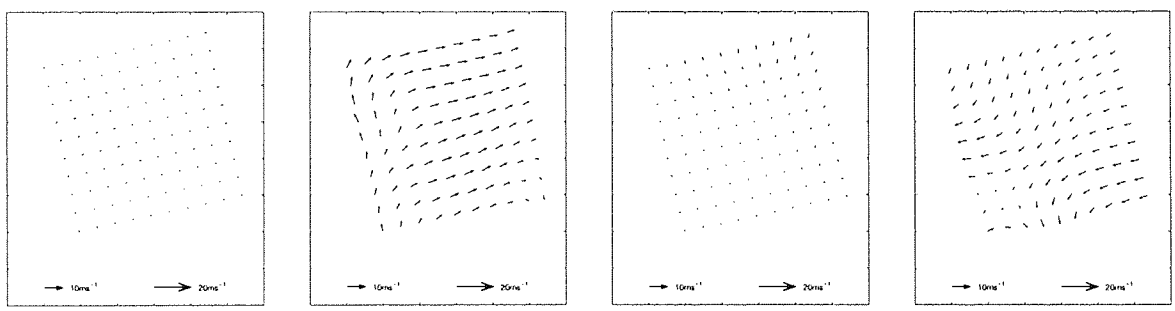
Figure 5.3: Examples of wind fields that have low statistics, but are good representations of the target NWP wind. The poor statistics are caused by greater accuracy of the probabilistic model in locating features of the wind field due the resolution of the data collected. The examples are paired: (a) and (b) represent a shift in the location of a front as does (c) and (d); (e) and (f) represent a difference in placement of a high pressure, and also greater smoothness in the retrieved wind field; (g) and (h) also show a high pressure front, but the retrieved wind field has sharper resolution due to the scatterometer readings; (i) and (j) are examples of low pressures as are (k) and (l).



(a) NWP target. (b) FOM= 0.48 (c) NWP target. (d) FOM= 0.50

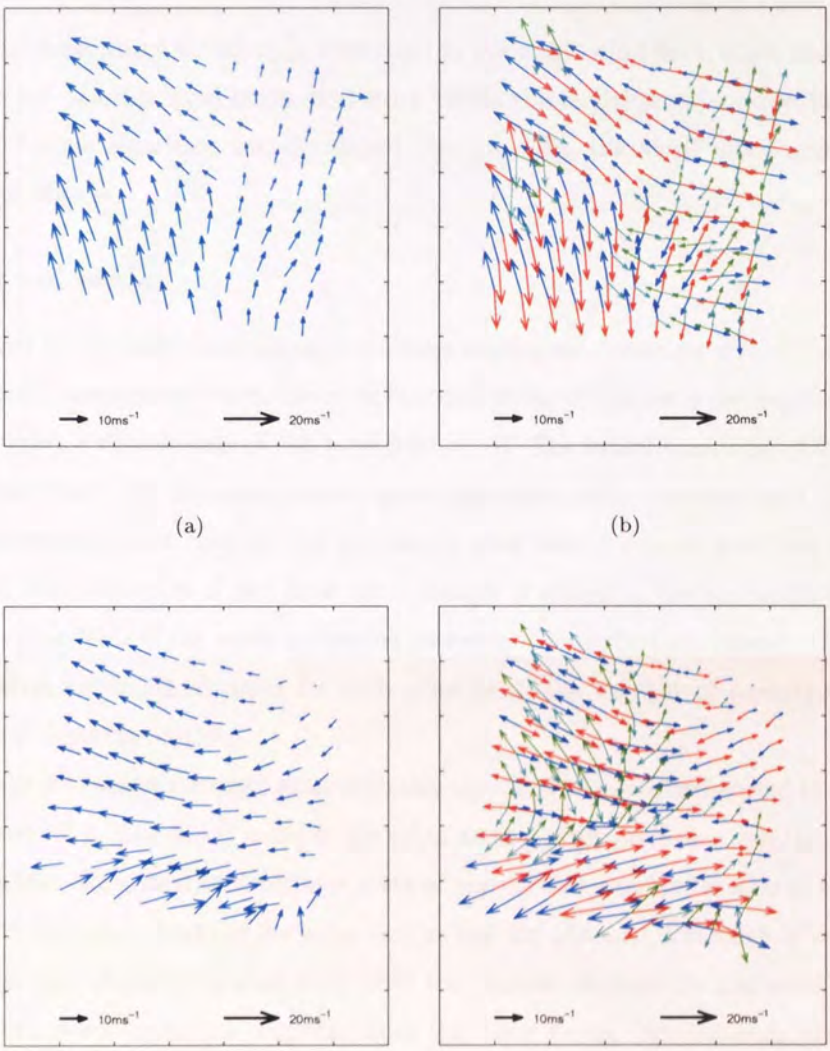


(e) NWP target. (f) FOM= 0.63 (g) NWP target. (h) FOM= 0.64



(i) NWP target. (j) FOM= 0.66 (k) NWP target. (l) FOM= 0.66

Figure 5.4: Examples of wind fields that have an FOM < 0.7. Retrieved wind fields for targets (a) and (g) show a move in structure; for targets (i) and (k) the model over estimates the wind speed; for targets (c) and (e) some of retrieved wind vectors are in the wrong direction.



(a) (b)

(c) (d)

Figure 5.5: Initial conditions of incorrect non-autonomously retrieved wind fields. (a) and (b) represent the first guess wind field and all possible MDN solutions respectively for example Figure 5.4 (c) of an incorrectly retrieved wind field. (c) and (d) represent the first guess wind field and all possible MDN solutions respectively for example Figure 5.4 (e) of an incorrectly retrieved wind field.

5.3 Autonomous ambiguity removal by MAP mode.

In this section we explore the idea of autonomous wind field retrieval by choosing wind field corresponding to the maximum *a posteriori* probability (MAP) point estimate of the posterior distribution, (5.1).

5.3.1 Method

Although for this experiment we are only interested in the MAP wind field, other unique modes, that correspond to other possible wind fields, that exist within the model are of interest later in our work. Hence, a mode finding algorithm was developed that identifies the MAP mode and the seven next most probable modes.

Mode finding and sorting

A mode is located by randomly initialising a non-linear optimiser (NETLAB's SCG) and then running the optimiser until convergence, where the error function of the optimiser is the negative log likelihood (NLL) of the posterior distribution of the wind field, (5.6). The initial search point for the optimiser is a sample drawn from the Gaussian process prior associated with the wind field. Using the prior to generate the starting point ensures that the initial wind field is smooth and that the variation in wind speed and wind direction of the local wind vectors is realistic. Once a mode is identified the wind vectors and the NLL of the mode under the posterior distribution are stored. The mode finding process is repeated five hundred times for each wind field (the number of repeats was determined through empirical experimentation).

The next stage is to select the eight most probable modes. The modes are sorted in ascending order of their associated NLL. The MAP mode is identified as that with the lowest negative log likelihood. A heuristic measure, H, is used to determine a set of seven further unique modes of the distribution. H takes into account the L_1 norm of the wind vectors and the absolute difference of the NLL between modes; we are required to separate wind fields that are opposite in direction and wind fields that have small variations between them but originate from the same mode. For instance, some modes may appear close in terms of the NLL value, but have slightly different structures in some part the wind field. In other cases the finite precision of the computer means that many of the modes identified will be very close in terms of their NLL values and originate from the same mode – here the mode with the minimum NLL is recorded. Hence we define H as:

$$H = d_{uv} + 10 * d_{nlp} \tag{5.8}$$

where

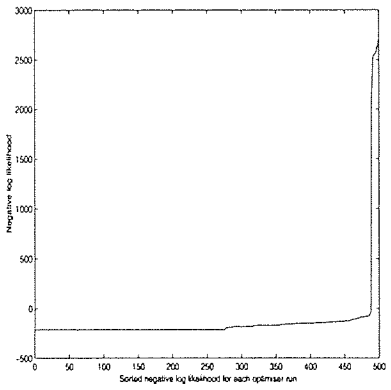
$$d_{uv} = \sum_{i=1}^{N} |u_i^{mode} - u_i^{test}| + |v_i^{mode} - v_i^{test}| \tag{5.9}$$

and

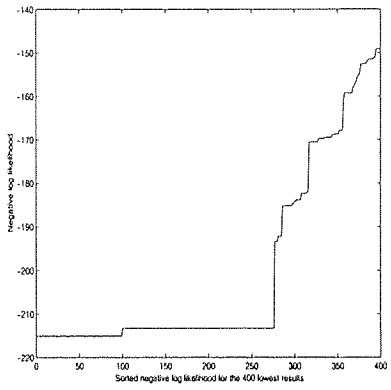
$$d_{nlp} = |E_{uv}^{mode} - E_{uv}^{test}| \tag{5.10}$$

H is computed on the modes sorted by their NLL values. The first H is computed for the MAP mode and the next mode in the list. A threshold value of 50 for H, which was derived empirically by testing several wind fields, ensures that modes corresponding to unique wind fields are identified. Hence, if the value of H is greater than 50 then the test mode is selected as a new unique mode, and added to the set of unique modes. The process is repeated sequentially through list of sorted modes, each time comparing the test mode with the current set of unique modes, until a predetermined number of unique modes have been identified.

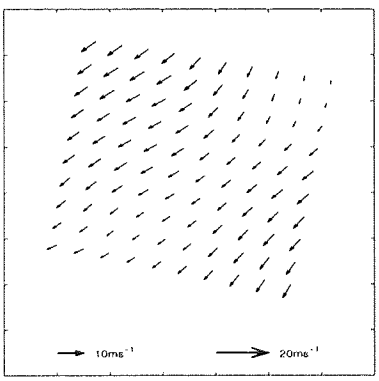
An example of the mode finding procedure is given. The NLL values are sorted in ascending order as shown in Figure 5.6 (a) and (b). The wind field associated with the minimum NLL is taken as the MAP wind field, Figure 5.6 (c). The unique modes are identified. The second most probable mode in this example is shown in Figure 5.6 (d). Note that the NLL of the two modes are close, but the wind vectors are opposite in direction to one another and the wind fields are meteorologically unique.



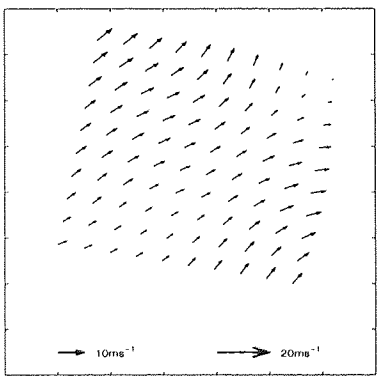
(a) All NLL values sorted



(b) First 400 sorted NLP values



(c) MAP mode, NLP = −215.09



(d) Second most probable mode, NLP = −213.29

Figure 5.6: An example of the mode finding process. (a) and (b) show how the negative log likelihood values increase after sorting – a stepped upward trend. (c) the MAP mode and (d) the second most probable mode – close in NLL value, but meteorologically unique.

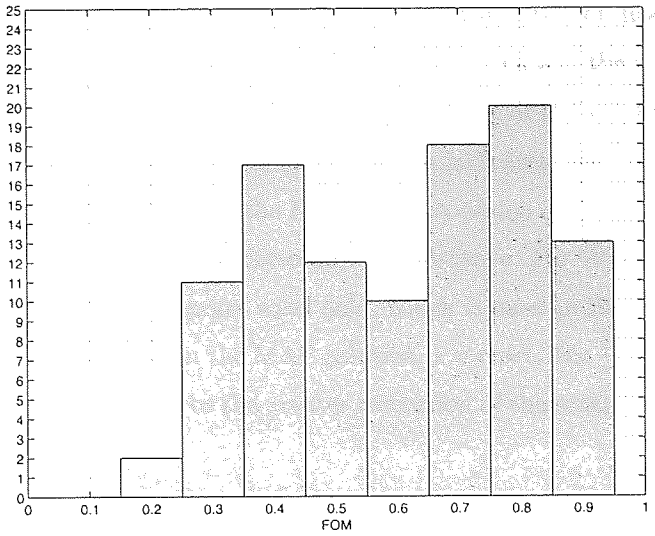


Figure 5.7: Histogram showing the distribution of the FOM values of MAP autonomously derived wind fields with an FOM < 1 which are not opposite in direction to their NWP target wind field.

Experiment

For each of the 575 target wind fields the MAP mode and the seven next most probable modes, which each correspond to unique wind field retrieval, were identified. Then the same summary statistics as those used in Section 5.2.1 were computed between the target NWP wind field and the retrieved wind fields. For the our predictions, we first consider the MAP retrieved wind field.

The analysis, however, is slightly different to that of Section 5.2.1 because of the possibility of retrieving wind fields that are opposite in direction to the target NWP wind field. Wind fields in the opposite direction to the target are first filtered out by checking the number of solutions to within twenty degrees of the target – if this value is zero then the wind field is labelled as opposite in direction to the target and recorded for visual inspection. The remaining wind fields (those that are not labelled opposite in direction) are then separated by their FOM value; those which have an FOM > 1 are considered correct and in agreement with the target; those that are < 1 are labelled for visual inspection.

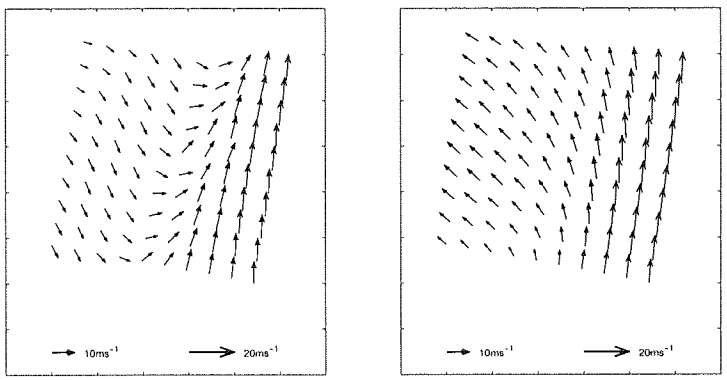
5.3.2 Results

There are 575 wind fields of which, when compared to their NWP target, 317 (55.1%) where not opposite in direction and had an FOM greater than one, 155 (27.0%) were identified as opposite in direction and 103 (17.9%) were not opposite in direction and had an FOM of less than one.

A histogram of the FOM values for those results that were not opposite in direction to their targets and had an FOM < 1 is shown in Figure 5.7. Compared with Figure 5.2 we notice there are two peaks in the distribution centred around an FOM value of 0.4 and 0.8. The two peaks are a result of two types of error in the retrieved wind field.

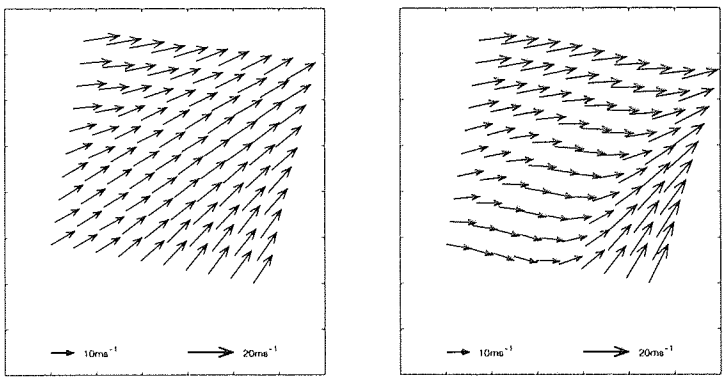
The first category, which accounts for the peak FOM at 0.8, is due to the same effect as seen in the

non-autonomous retrieval method where the solution is correct in terms of direction, but the features of the wind fields have shifted. Four examples, WF1–WF4, are given and the wind fields are plotted in Figure 5.8 and Figure 5.9. For WF1, Figure 5.8 (a) and (b), some of the wind vectors of the retrieved wind field are in the opposite direction to the target, which indicates that the retrieved wind field is incorrect. For WF2, Figure 5.8 (c) and (d), the MAP result contains more detail than the target wind field and the MAP wind field is considered correct in this instance. For WF3, Figure 5.9 (a) and (b), the MAP result is smoother than the target wind field, which is most probably due to the smoothing implied by the prior in the model, again the MAP solution is correct. For WF4 the MAP result shifts the weather system – this is attributed to the greater resolution of the scatterometer data – and hence this solution is considered correct.



(a) WF1: NWP target

(b) WF1: MAP mode, FOM = 0.50

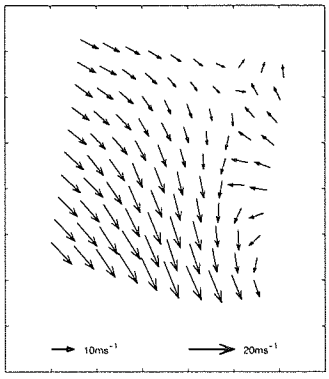


(c) WF2: NWP target

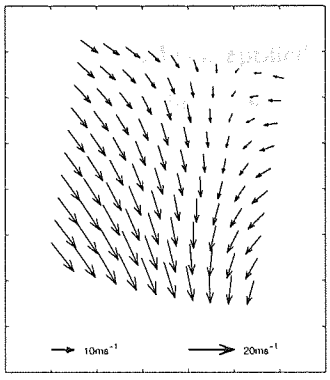
(d) WF2: MAP mode, FOM = 0.83

Figure 5.8: MAP retrieved wind fields where $0.5 \leqq \text{FOM} < 1$. For WF1 some of the wind vectors of the retrieved wind field are in the opposite direction to the target; for WF2 the MAP result contains more detail than the target.

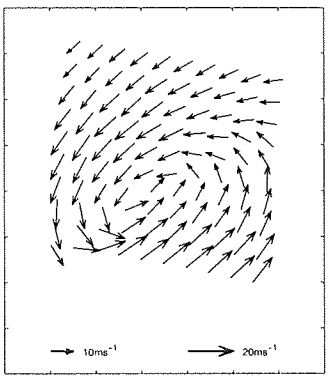
The second category, which accounts for the peak around FOM values of 0.4, is where the MAP solution has no correlation with the target wind field. Four examples, labelled WF5, WF6, WF7 and



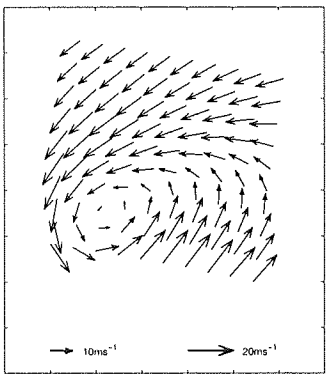
(a) NWP target, WF3



(b) MAP mode, WF3, FOM=0.95



(c) NWP target, WF4



(d) MAP mode, WF4, FOM=0.89

Figure 5.9: MAP retrieved wind fields where $0.5 \leqq \text{FOM} < 1$. For WF3 the MAP retrieved wind field is smoother than the target and for WF4 the position of the retrieved system is shifted when compared to the target.

WF8, of this kind of solution are given in Figure 5.10, where their respective FOM values are 0.39, 0.34, 0.36, 0.32. In these examples we see that the MAP wind field solutions are smooth, and are meteorologically plausible but bear no resemblance to the target. In these cases the prior is overriding the information provided by the MDNs, causing the over–smooth wind fields. Inspection of the 2^{nd}, 3^{rd} and 4^{th} most probable modes for examples WF5, WF6 and WF8 indeed shows that the correct solution exists within the model, but has a much lower probability density than the MAP mode. For example WF7 we see that none of the first three retrieved wind fields agree with the target.

Of the 103 solutions with an FOM < 1.0, a visual inspection with reference to the target NWP wind field classified 54 of the wind fields as correct and 49 solutions as incorrect.

5.3.3 Conclusions

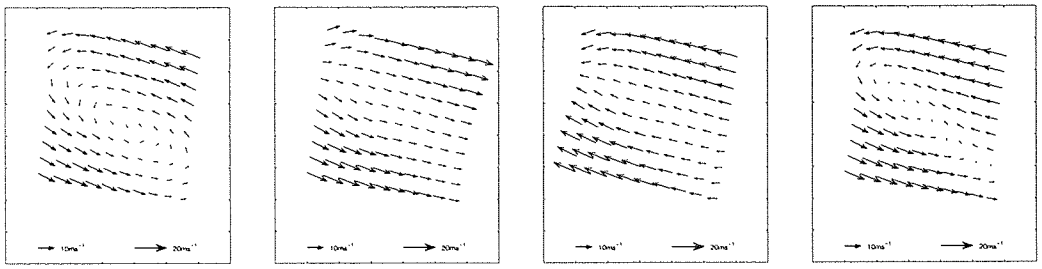
In this section an autonomous method for ambiguity removal has been applied by taking the MAP mode of the posterior distribution as the retrieved wind field from the model.

Although all the wind fields retrieved are meteorologically plausible, 371 (64.5%) predictions where in agreement with their target, 155 (27.0%) of predictions were in the opposite direction to their target, and 49 (8.5%) bear no resemblance to their target.

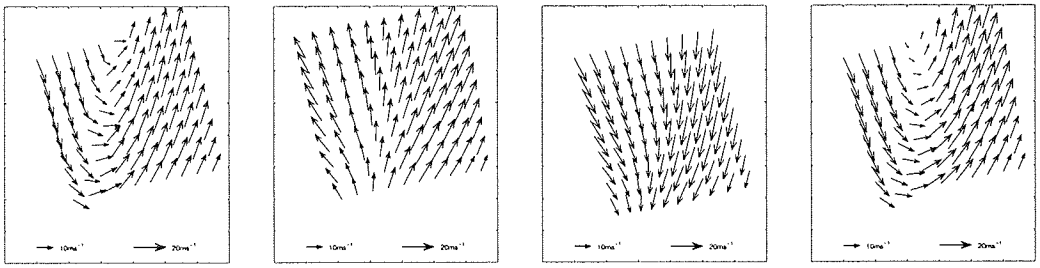
5.4 Chapter review

Using a non-autonomous ambiguity removal method we have shown that the wind field model, proposed in Chapter 1, is plausible, and that it is possible to retrieve wind fields by direct inversion of the scatterometer data. The MAP estimate for autonomous ambiguity removal has been tested and shown to be of mediocre success, where 64.5% of solutions were in agreement with their target.

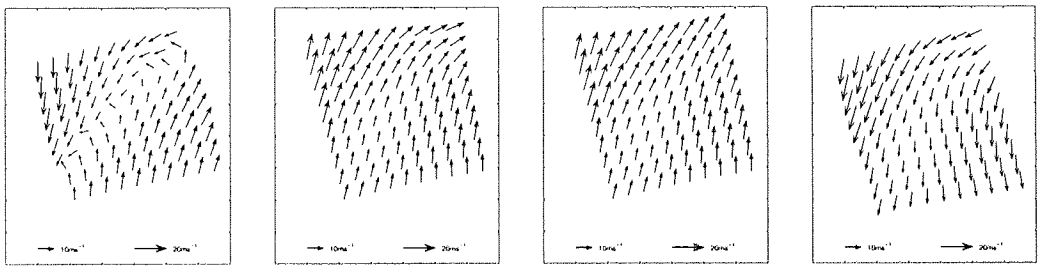
Using the MAP estimate for autonomous ambiguity removal ignores some of the information that is inherent in the model, such as the probability mass associated with each mode identified. In the next chapter we develop methods that sample from the multi-modal distributions in such a manner that they approximate the mass associated with each mode and we hope to improve our predictions by using this additional information. For instance, we may find that the mass associated with the correct mode is greater than that associated with other modes, even though the probability density of some of the other modes may be greater. This would alter our prediction, and in a principled and autonomous fashion, allow us to choose the correct mode.



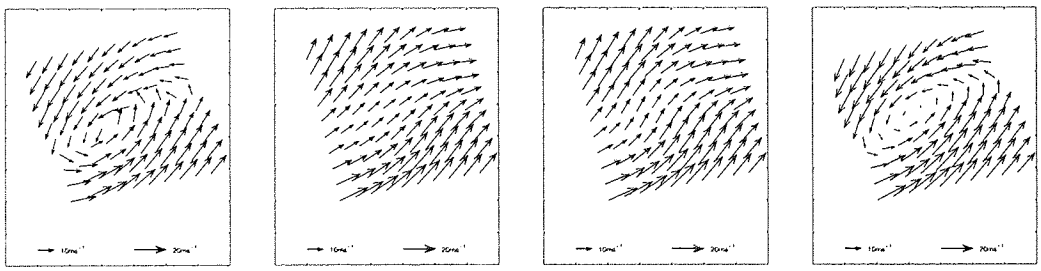
WF5: NWP | WF5: 1^{st} mode | WF5: 2^{nd} mode | WF5: 3^{rd} mode



WF6: NWP | WF6: 1^{st} mode | WF6: 2^{nd} mode | WF6: 3^{rd} mode



WF7: NWP | WF7: 1^{st} mode | WF7: 2^{nd} mode | WF7: 3^{rd} mode



WF8: NWP | WF8: 1^{st} mode | WF8: 2^{nd} mode | WF8: 3^{rd} mode

Figure 5.10: MAP retrieved wind fields with an FOM < 0.4. For these samples the MAP result bears no resemblance to the target wind field. This is attributed to the over smoothing the model. For examples WF5, WF6, and WF8, inspection of the second and third most probable wind field retrievals reveals that the solutions exist within the model, but with a lower NLL.

Chapter 6

Autonomous ambiguity removal through sampling.

Key word summary

Autonomous ambiguity removal	**Wind field retrieval**
MCMC	**Mode–jumping**
Probability mass estimation	**Computational efficiency**

6.1 Introduction

In the previous chapter it was shown how, using deterministic techniques, it was possible to autonomously retrieve wind fields from the posterior distribution by choosing the mode with the maximum *a posteriori* probability density. The results, however, were not totally reliable because an incorrect mode may have the maximum *a posteriori* (MAP) value. In this chapter we further develop autonomous ambiguity removal by taking into account the estimated probability mass associated with a mode to assist in choosing the retrieved wind field from the model.

The mass of each mode is approximated by using Markov Chain Monte Carlo (MCMC) techniques which are adapted to jump between modes of the posterior distribution at a rate that is proportional to probability mass associated with each mode; the amount of time the sampler spends in each mode directly relates to the mass associated with each mode. For instance, this would improve our prediction if the second most probable mode is associated with the correct wind field, and also has the most mass associated with it. If this is the case then we have a principled approach to choose this mode over the MAP mode, and we are a step closer to an autonomous ambiguity removal system.

In practice this turns out to be a difficult problem to solve. Although there are several algorithms available to solve this problem they all have a common feature: they mode search on–line. For

this problem this turns out to be a disadvantage because the algorithms become so computationally intensive that they become impractical to run in a realistic time frame as the dimensionality of the problem space increases. An alternative approach, developed in this chapter, is to identify the modes off-line and supply this information to an MCMC sampler adapted to choose the modes in a stochastic manner. Hence, a novel algorithm is developed for this problem and it is shown how the dichotomy of the on–line and off–line approaches is manifest in the computation efficiency: the off–line approach is significantly more computationally efficient whilst being as accurate as the on–line methods for mass approximation.

Two on-line algorithms and the off–line algorithm are compared for performance by applying them to a toy problem designed to emulate the wind field posterior distribution; for this exercise the dimension of the problem is relatively low due to the time taken to run the on–line algorithms. Once the credibility of the off–line method is established, including experiments on a high dimensional toy problem equivalent to the real problem, the algorithm is applied to five case study wind fields.

The wind fields selected for the case studies are interesting for their meteorological structure and from the application perspective. We see that for some cases the sampling approach helps to choose the correct wind field where the MAP method would choose an incorrect wind field. Finally, the case studies indicate a pragmatic approach to autonomous wind field retrieval which is tested on one hundred wind fields.

6.2 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is similar to sitting in seiza during Zen Buddhist meditation. The concept is simple and easy to understand, however the application can be one of the most demanding and difficult pastimes one may encounter! In this section MCMC theory is introduced along with the original and most simple implementation: the Metropolis-Hastings algorithm.

MCMC is used in a broad range of statistical applications including statistical physics and Bayesian analysis. Generally, MCMC techniques are used to approximate intractable integrals, where drawing samples directly from the distribution of interest is impossible. MCMC techniques make no assumptions about the form of the distribution of interest; it is because of this property that MCMC techniques are so broadly applicable and so powerful. In theory, state-spaces of any shape and size may be investigated and there is no constraint over approximation by a Gaussian or any other form of distribution. This property often turns out to be useful as sometimes the majority of mass in a distribution may well not be close to the modes.

One difficulty with applying MCMC techniques is assessing convergence. There are several techniques available, the simplest, but most unsophisticated, is to visually inspect the output of the sampler – a solution suitable for low dimensional problems where visualisation is a practical alternative. For higher dimensional problems more sophisticated techniques are employed to monitor the Markov chains as they develop. It is this kind of technique that is used in this work, and is introduced

after the mode sampling techniques have been discussed.

I cannot attempt to review all the MCMC literature available to the reader in this thesis, and neither should I – there are several excellent texts available[1] which cover the theory and application in fine detail; the reader is directed to: Neal (1993), Tierney (1994), Gilks *et al.* (1996), Brooks (1999).

6.2.1 MCMC theory and terminology

In the context of this thesis, we are interested in drawing samples from a posterior probability distribution, $\pi(\boldsymbol{x})$, generated from a Bayesian model. In our application $\boldsymbol{x}$ represents the geophysical parameters, the wind vectors, of the posterior distribution of the wind field model. MCMC is the union of two fundamental concepts, *Monte Carlo* integration and *Markov Chains*.

Monte Carlo integration is the approximation of an integration by a summation. Suppose we wish to find the expectation of $f(\boldsymbol{x})$ by the integral, given that $\boldsymbol{x}$ has the distribution $\pi(\boldsymbol{x})$,

$$E[f] \equiv \int f(\boldsymbol{x})\pi(\boldsymbol{x})\, d\boldsymbol{x}, \tag{6.1}$$

and this integral is analytically intractable, as in the case of the Bayesian posterior distribution of the wind field model. We may approximate (6.1) by the average:

$$E[f] \approx \frac{1}{N}\sum_{t=1}^{N} f(\boldsymbol{x}_t), \tag{6.2}$$

where $\boldsymbol{x}_t$ are the N samples drawn from $\pi(\boldsymbol{x})$. This is the Monte Carlo evaluation of (6.1). In order to compute the Monte Carlo evaluation it is necessary to generate a series $\boldsymbol{x}_1 \ldots \boldsymbol{x}_N$ such that its distribution is equal to $\pi(\boldsymbol{x})$. However, it is often infeasible to draw $\boldsymbol{x}$ directly from $\pi(\boldsymbol{x})$. Instead, we generate a series which has the following property:

$$T(\boldsymbol{x}_{t+1}|\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}, \boldsymbol{x}_t) = T(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t). \tag{6.3}$$

A series that fulfils property (6.3) is called a *Markov Chain*. A Markov Chain has the ability to 'forget' its past, and its future is conditional only on the previous state of the chain. Therefore, a Markov Chain is defined by its initial condition $\boldsymbol{x}_0$, and its transition probabilities $T(\cdot|\cdot)$.

A *stationary* (or *invariant*) Markov Chain is one that persists after time t, such that at t^*, where $t^* > t$, the distribution of $\boldsymbol{x}^*$ is equal to $\pi(\boldsymbol{x})$. We may write this as follows:

$$\pi(\boldsymbol{x}^*) = \int T(\boldsymbol{x}^*|\boldsymbol{x})\pi(\boldsymbol{x})\, d\boldsymbol{x}. \tag{6.4}$$

One way to ensure that stationarity with respect to $\pi(\boldsymbol{x})$ is maintained is to ensure that *detailed balance* holds. Detailed balance is a desirable, but not necessary, condition for any Markov Chain – if detailed balance holds then stationarity will be met. Formally, this is written as the probability:

$$\pi(\boldsymbol{x})T(\boldsymbol{x}^*|\boldsymbol{x}) = \pi(\boldsymbol{x}^*)T(\boldsymbol{x}|\boldsymbol{x}^*). \tag{6.5}$$

[1] The reader is also directed to the MCMC pre-print server located on the internet at: `http://www.statslab.cam.ac.uk/~mcmc/`

A Markov Chain satisfying detailed balance is also described as being *reversible*. When designing MCMC algorithms it is important to ensure that detailed balance holds on the transition kernels. Proving this guarantees that it is possible to reach a stationary distribution.

Once detailed balance is shown for a particular transition kernel, it is possible to combine any number of kernels together in order to create a single sampler that has enhanced convergence characteristics. There are two ways in which transition kernels may be combined: by *mixing* or by *cycles*. A cycle of kernels implies that the transition kernels are run in a pre-determined order and repeated until the sampler is stopped; for mixtures of kernels, each kernel is assigned a probability and at each step a kernel is chosen according to its probability (Brooks, 1999).

In practice a Markov Chain is generated by an iterative computer simulation. The transition kernels are implemented by a two stage process: firstly a candidate state is generated, by drawing samples from a *proposal distribution* for which direct sampling is tractable, for the next state of the chain which is conditional on the current state. Secondly, the candidate state is either rejected or accepted in a stochastic manner which depends on $\pi(\boldsymbol{x})$. Hence the new state of the chain either becomes the candidate state, or remains as the previous state. The art of MCMC analysis and implementation is the design of the proposal distribution(s) to generate sensible candidate states that will generally have a good acceptance rate, adequately explore the state space and satisfy detailed balance.

After the chain has been run for a certain length of time it will converge to a stationary distribution $\pi(\boldsymbol{x})$. The length of time taken to reach this stage is called the *burn-in* period. Assessing the time taken for burn-in is one of the major difficulties in using MCMC simulations and is the motivation behind much of the research into the assessment of convergence of Markov Chains. However, given that this time may be found, the Monte Carlo approximation is modified to account for the burn-in period such that

$$E[f] \approx \frac{1}{N-m} \sum_{t=m+1}^{N} f(\boldsymbol{x}_t), \tag{6.6}$$

where N is the total number of samples drawn from the distribution and m is the burn-in period.

6.2.2 Metropolis-Hastings Algorithm

Metropolis *et al.* (1953) and Hastings (1970) introduced a general algorithm for generating Markov Chains which was straight forward and easy to implement on a digital computer. The beauty of this algorithm is the elegant way in which the proposal distribution is applied, and the ease with which detailed balance is proved. The Metropolis-Hastings algorithm forms the basis for the mode jumping algorithms we encounter in the next section.

A new state $\boldsymbol{x}_{t+1}$ is generated from the previous state, $\boldsymbol{x}_t$, by generating a candidate state $\boldsymbol{x}^*$ specified by a *proposal distribution*, $q(\boldsymbol{x}^*|\boldsymbol{x}_t)$, and then deciding whether or not to accept the candidate state based on its probability density relative to that of the old state, with respect to the desired invariant distribution, π (Neal, 1995a).

In detail, the transition from state $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t+1}$ is defined as follows:

1. generate a candidate state, $\boldsymbol{x}^*$, by drawing a sample conditionally on the current state $\boldsymbol{x}_t$ from the proposal distribution $q(\boldsymbol{x}^*|\boldsymbol{x}_t)$.

2. if $\pi(\boldsymbol{x}^*) > \pi(\boldsymbol{x})$ accept the candidate state; if $\pi(\boldsymbol{x}^*) < \pi(\boldsymbol{x})$ then accept the candidate state with probability $\alpha = \frac{\pi(\boldsymbol{x}^*)}{\pi(\boldsymbol{x})}$.

3. If the candidate is accepted then $\boldsymbol{x}_{t+1} = \boldsymbol{x}^*$; if the candidate state is rejected let $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t$.

Often, when dealing with exponential probability distributions $\pi(\boldsymbol{x})$ is defined in terms of an 'energy' function where $E(\boldsymbol{x}) = -ln(\pi(\boldsymbol{x}))$ and hence $\pi(\boldsymbol{x}) \propto exp(-E(\boldsymbol{x}))$. In step 2 of the Metropolis–Hastings algorithm, when deciding whether to accept the candidate solution, the rule is modified such that a candidate state with a lower energy than the current state is always accepted and a candidate state with a higher energy is accepted with probability $\exp\{-(E(\boldsymbol{x}^*)) - E(\boldsymbol{x}_t)\}$.

Detailed balance is easily shown to hold if the proposal distribution is symmetric such that $q(\boldsymbol{x}^*|\boldsymbol{x}) = q(\boldsymbol{x}|\boldsymbol{x}^*)$. Here is the proof:

$$\pi(\boldsymbol{x})T(\boldsymbol{x}^*|\boldsymbol{x}) = \pi(\boldsymbol{x})q(\boldsymbol{x}^*|\boldsymbol{x})\, min\left(1, \frac{\pi(\boldsymbol{x}^*)}{\pi(\boldsymbol{x})}\right), \tag{6.7}$$

$$= q(\boldsymbol{x}^*|\boldsymbol{x})\, min\Big(\pi(\boldsymbol{x}), \pi(\boldsymbol{x}^*)\Big), \tag{6.8}$$

$$= \pi(\boldsymbol{x}^*)q(\boldsymbol{x}^*|\boldsymbol{x})\, min\left(\frac{\pi(\boldsymbol{x})}{\pi(\boldsymbol{x}^*)}, 1\right), \tag{6.9}$$

$$= \pi(\boldsymbol{x}^*)q(\boldsymbol{x}|\boldsymbol{x}^*),\, min\left(1, \frac{\pi(\boldsymbol{x})}{\pi(\boldsymbol{x}^*)}\right), \tag{6.10}$$

$$= \pi(\boldsymbol{x}^*)T(\boldsymbol{x}|\boldsymbol{x}^*). \tag{6.11}$$

One simple symmetric proposal distribution, which is often implemented, is a spherical Gaussian distribution centred at $\boldsymbol{x}_t$, which implies the generation of a candidate state is:

$$\boldsymbol{x}^* = \boldsymbol{x}_t + N(0, \sigma^2), \tag{6.12}$$

which implies drawing samples from a spherical Gaussian distribution.

6.3 Preview of algorithms for sampling from multi–modal distributions

The MCMC techniques discussed in the last section are very powerful if they are applied to problems which are uni–modal. However, they become unreliable if the distribution of interest, $\pi(\boldsymbol{x})$, has isolated modes separated by regions of near zero probability. The main reason for the unreliability is that the Markov Chain generated by these techniques will move between isolated modes very rarely, and therefore will not represent the probability mass of the distribution unless run for an excessively long periods of time. Hence, in order to improve the wind field predictions, by understanding the

probability mass associated with each mode, we need more sophisticated algorithms to draw samples from the multi–modal posterior distribution that represents the wind field model.

We therefore turn our attention to algorithms designed to generate samples from each mode in accordance with the relative probability mass associated with each mode. Four algorithms are reviewed. The first three, in chronological order of publication, are: Simulated Tempering (Marinari and Parisi, 1992), Tempered Transitions (Neal, 1995b) and Mode Jumping Kernels for MCMC (Tjelmeland and Hegstad, 1999; Tjelmeland and Hegstad, 2001). These algorithms are similar by way of their mode finding method as they rely on on–line methods to identify modes from which candidate states are proposed.

The fourth method, which is developed in this thesis, takes an alternative approach, by identifying candidate modes off–line, before the sampling commences, and then uses them during sampling.

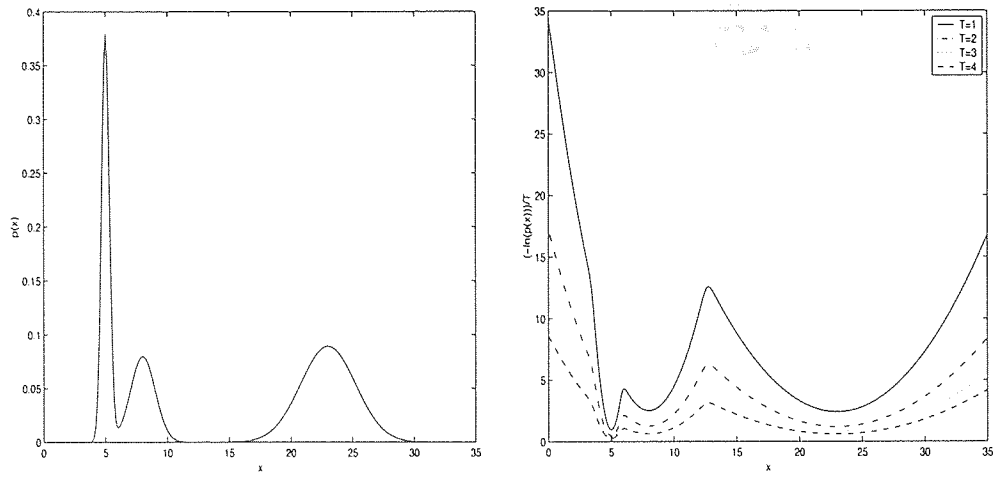
Simulated Tempering: Marinari and Parisi (1992)

This algorithm is an extension of the Simulated Annealing optimisation algorithm of Kirkpatrick *et al.* (1983). Kirkpatrick's algorithm is designed to identify the state associated with the global minimum energy within a state–space by sampling from a series of distributions, each one slightly different from the last. The starting distribution is easily sampled from, and allows macro exploration of the state–space. The final distribution is the distribution of interest and represents the micro exploration of the state–space (note that the distributions are often based on a series of canonical distributions which are obtained by varying a temperature parameter). The first distribution is defined by a high temperature and then cooling to the final distribution of interest at a low temperature. For optimisation, the temperature will decrease to zero, whilst for a sampling application the temperature will be non–zero, see Figure 6.1 and Figure 6.2. For mode finding and exploration the last state drawn from the algorithm is used to initialise a sampler for mode exploration. To explore multi–modal distributions this method may be repeated several times from different initial conditions in a hope to identify important modes.

Nevertheless, this method will not, in general, create a sample run in which all modes are fairly represented primarily due to the probability of sampling from a mode being dominated by its basin of attraction and not its associated probability mass.

Simulated Tempering solves this problem and is guaranteed to sample from the correct distribution. As in Simulated Annealing, a series distributions are defined which are systematically easier to sample from; often they are indexed by a temperature parameter. However, instead of systematically moving through the distributions, Simulated Tempering treats the distribution index as an extra variable to be updated stochastically. Therefore the different distributions are explored via a random walk. At the lowest temperature, the samples drawn are from the distribution of interest; at higher temperatures, where the distribution is easier to sample from, the macro elements of the state–space may be explored and facilitate movement between isolated modes.

Unfortunately, there is a disadvantage to using this method, which is that we are required to



(a) (b)

Figure 6.1: A 1-d multi–modal probability distribution is used as an example of how 'temperature' is applied to generate distributions that are easier to sample from. (a) shows the probability distribution; (b) shows the negative log–likelihood of the probability distributions, which is called the energy function $E(x) = -\ln(p(x))$, and how the energy landscape flattens as temperature is increased.

calculate the normalising constants of the intermediate distributions; generally this is time consuming and laborious and sometimes impractical. However, this disadvantage is the motivation for the design of the next algorithm Tempered Transitions.

Tempered Transitions: Neal (1995b)

The Tempered Transitions algorithm is a further development of the Simulated Tempering algorithm designed so as not to require the calculation of the normalising constants of the intermediate distributions. The random walk through the interpolating distributions in Simulated Tempering is replaced by an ordered movement from the desired distribution, to the easily sampled distribution and back to the desired distributions via the interpolating distributions. This method of movement through the interpolating distributions leads to a transition probability equation in which the normalising constants of the interpolating distributions cancel out, thus removing the need to calculate them – for practical implementations this offers a big advantage over the Simulated Tempering algorithm.

However, the advantage gained by removing the random walk through the interpolating distributions is lost by the extra number of interpolating distributions required for the method to have a reasonable acceptance probability for proposals that jump between modes. The series of interpolating probability distributions are generated at several temperatures where the intermediate distributions are systematically easier to sample from than the base distribution of interest.

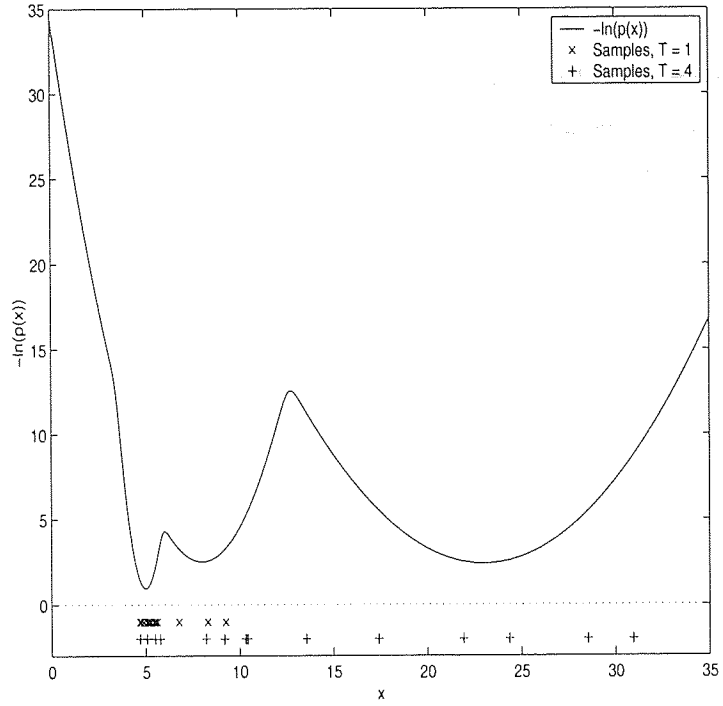


Figure 6.2: By increasing the temperature it is possible explore more of the state–space, with T=4 there are samples drawn from the region of the second isolated mode. Essentially, raising the temperature flattens the energy landscape, making it easier for proposals to be accepted that would otherwise be rejected when temperature is 1. Hence the distributions at higher temperatures are described as 'easier to sample from'.

Mode Jumping Proposals in MCMC: Tjelmeland and Hegstad (1999)

The algorithm of Tjelmeland and Hegstad (1999) introduces a mode jumping transition kernel which involves a stochastic jump followed by a deterministic optimisation step. The deterministic step in the algorithm implies that the proposal distribution is asymmetric. Therefore the conditional probabilities of the proposal distribution for the forward and reverse step in the detailed balance equation are computed for each mode jumping proposal; this contrasts with the previous two algorithms which relied on the Metropolis–Hasting algorithm for their proposal steps.

Hence, to generate a candidate state for a mode jumping proposal a large random step in state-space is taken followed by an optimisation step to arrive at a local mode. The curvature of the mode is calculated by inverting Hessian matrix of the log likelihood which is taken as the approximation of the covariance matrix of a Gaussian distribution with mean at the mode. A sample is drawn from the Gaussian distribution defined by the covariance matrix and mean at the mode.

This method is dependent of the availability of gradient information, which for large scale problems must be able to be computed directly (for computational efficiency), although for smaller scale problems we may use finite differences to approximate the gradient. Also, like Tempered Transitions, there is no need to compute the normalisation constants of any of the probability distributions employed in this algorithm.

Divide and Conquer

This method is motivated by a need for computational efficiency. All of the methods discussed so far in this section incorporate a mode searching algorithm within the mode jumping transition kernel – this turns out to be computationally inefficient for large scale problems such as the wind field retrieval application. Some of the inefficiencies arise as:

- there is a lot of effort invested in identifying a mode;
- the mode found could be a local mode with minimal probability mass and therefore almost always be rejected;
- there is no memory of modes visited implying that computational effort is wasted 're-finding' the same mode many times during a sample run;
- the basin of attraction of a local and insignificant mode may well dominate the probability distribution, hence causing low acceptance rates of candidate states.

For the wind field retrieval problem, there are a finite number of modes which contain the majority of the probability mass of the distribution, and it is the relation between these modes in which we are interested. The Divide and Conquer algorithm proposes to solve this problem by assuming the modes of interest are known before the sampling simulation is run. Once the modes are known, the mode jumping kernel is a simple modification of the Metropolis-Hasting algorithm in which the location of the proposed new mode is included along with the current state and a random element. Because the mode–finding is computed off-line the sampling algorithm is computationally efficient.

Summary

Of the methods reviewed, three appear suitable for the wind field application. These are Tempered Transitions (TT) , Mode Jumping Proposals in MCMC (MJMCMC) and the Divide and Conquer (DC) algorithm. Although Simulated Tempering is as efficient as Tempered Transitions, it requires a complicated and laborious initialisation which includes the approximation of the normalising constants of each intermediate probability distributions required in the algorithm. Given that Neal (1995b) shows that TT is equivalent to Simulated Tempering (ST), the ST algorithm shall no longer be considered in this thesis.

In the next section we take a closer look at the mathematics supporting the transitions kernels for our three chosen algorithms.

6.4 Technical detail of applied sampling methods

The three methods TT, MJMCMC and DC each have a unique approach for the generation of candidate proposal states in isolated modes which have a reasonable probability of being accepted.

For each algorithm we give the mathematical definition of the transition kernel and confirm that detailed balance is proved; this has been shown already for TT, but the proofs are provided for MJMCMC and DC.

6.4.1 Tempered Transitions

In Neal (1995b) the problem of sampling from a distribution, $\pi(\boldsymbol{x})$, which may well have isolated modes, is addressed. To assist sampling from what may be many isolated modes a series of other distributions are defined, $p_0(\boldsymbol{x}) = \pi(\boldsymbol{x})$, $p_1(\boldsymbol{x}), \cdots, p_n(\boldsymbol{x})$, with $p_n(\boldsymbol{x})$ being easier to sample from than $p_0(\boldsymbol{x})$ (see Figure 6.1 and Figure 6.2).

To apply the Tempered Transitions method a set of base transitions are required for each i, $\hat{T}_i$ and $\check{T}_i$, which both have $p_i(\boldsymbol{x})$ as their invariant distribution. They must also meet the following complementary reversibility condition for all $\boldsymbol{x}$ and $\boldsymbol{x}^*$:

$$p_i(\boldsymbol{x})\hat{T}_i(\boldsymbol{x}, \boldsymbol{x}^*) = \check{T}_i(\boldsymbol{x}^*, \boldsymbol{x})p_i(\boldsymbol{x}^*). \tag{6.13}$$

Here $\hat{T}_i$ and $\check{T}_i$ may be the same kernel, in which case it must satisfy detailed balance with respect to $p_i(\boldsymbol{x})$. Otherwise, it is possible to chain several transition kernels together as sub–transitions applied in sequence such that $\hat{T}_i = S_1 \cdots S_k$, and if all S_j satisfy detailed balance, then $\check{T}_i = S_k \cdots S_1$.

A state, $\boldsymbol{x}^*$, is proposed by applying the base transitions in the sequence $\hat{T}_1 \cdots \hat{T}_n \check{T}_n \cdots \check{T}_1$; the proposed state is accepted (or rejected) on the basis of the ratios of the probabilities of the intermediate states. As the generation of the proposal state involves the use of $\hat{T}_n$ and $\check{T}_n$, which are designed to move rapidly about the state space, we hope that the proposal state will have a wide distribution and not be restricted to the mode in which the initial state started. The main reason for the intermediate transitions is to keep the acceptance probabilities reasonably high (again, see Figure 6.1 and Figure 6.2 which demonstrate this effect).

Explicitly, a proposal state, $\check{\boldsymbol{x}}_0 = \boldsymbol{x}^*$, is generated from the current state $\hat{\boldsymbol{x}}_0$ (equivalent to $\boldsymbol{x}$) as follows:

$$\begin{gathered}
\text{generate } \hat{\boldsymbol{x}}_1 \text{ from } \hat{\boldsymbol{x}}_0 \text{ using } \hat{T}_1; \\
\text{generate } \hat{\boldsymbol{x}}_2 \text{ from } \hat{\boldsymbol{x}}_1 \text{ using } \hat{T}_2; \\
\vdots \\
\text{generate } \bar{\boldsymbol{x}}_n \text{ from } \hat{\boldsymbol{x}}_{n-1} \text{ using } \hat{T}_n; \\
\text{generate } \check{\boldsymbol{x}}_{n-1} \text{ from } \bar{\boldsymbol{x}}_n \text{ using } \check{T}_n; \\
\vdots \\
\text{generate } \check{\boldsymbol{x}}_1 \text{ from } \check{\boldsymbol{x}}_2 \text{ using } \check{T}_2; \\
\text{generate } \check{\boldsymbol{x}}_0 \text{ from } \check{\boldsymbol{x}}_1 \text{ using } \check{T}_1.
\end{gathered}$$

The proposal state is accepted with probability[2]

$$\min\left[1, \frac{p_1(\hat{\boldsymbol{x}}_0)}{p_0(\hat{\boldsymbol{x}}_0)} \cdots \frac{p_n(\hat{\boldsymbol{x}}_{n-1})}{p_{n-1}(\hat{\boldsymbol{x}}_{n-1})} \cdot \frac{p_{n-1}(\check{\boldsymbol{x}}_{n-1})}{p_n(\check{\boldsymbol{x}}_{n-1})} \cdots \frac{p_0(\check{\boldsymbol{x}}_0)}{p_1(\check{\boldsymbol{x}}_0)}\right]. \tag{6.14}$$

If the proposal state is not accepted then the new state of the Markov chain becomes the previous state of the Markov chain. The point to note is that each of the distributions, p_i, appear in both the enumerator and denominator of the acceptance probability calculation, which is therefore a product of ratios where the normalising constants of each p_i cancel out; this has the advantage that it is not necessary to compute the normalising constants of each p_i.

To ensure a reasonably high acceptance probability the intermediate distributions must be carefully chosen in order to provide adequate interpolation between $p_0(\boldsymbol{x})$ and $p_n(\boldsymbol{x})$. The number and spacing of these distributions will usually be determined by running pilot experiments before running the main sampling runs.

The approached used in this work to implement the Tempered Transition method is to index the distributions by their 'temperature':

$$p_0(\boldsymbol{x}) \propto \exp\left(-\frac{E(\boldsymbol{x})}{t_0}\right) \tag{6.15}$$

where E is the energy function and t_0 is the temperature. The intermediate distributions are defined by

$$p_i(\boldsymbol{x}) \propto \exp\left(-\frac{E(\boldsymbol{x})}{t_i}\right) \tag{6.16}$$

such that the temperatures are ordered $t_0 < t_1 < \cdots < t_n$.

Hence for the first half of the tempered transition the temperature rises; in the second half the temperature falls. The second half is similar to a simulated annealing run, except that in simulated annealing there is no principled way of moving between modes that are identified from multiple runs; tempered transitions permits the acceptance or rejection of a new state in a new mode in a valid manner. The acceptance probability is expressed as follows:

$$\min\left[1, \frac{p_1(\hat{\boldsymbol{x}}_0)}{p_0(\hat{\boldsymbol{x}}_0)} \cdots \frac{p_n(\hat{\boldsymbol{x}}_{n-1})}{p_{n-1}(\hat{\boldsymbol{x}}_{n-1})} \cdot \frac{p_{n-1}(\check{\boldsymbol{x}}_{n-1})}{p_n(\check{\boldsymbol{x}}_{n-1})} \cdots \frac{p_0(\check{\boldsymbol{x}}_0)}{p_1(\check{\boldsymbol{x}}_0)}\right] = \min\left[1, \exp(-(\hat{F} - \check{F}))\right] \tag{6.17}$$

where $\hat{F}$and $\check{F}$are defined to be:

$$\hat{F} = \log\left[\prod_{i=0}^{n-1} \frac{p_{i+1}(\hat{\boldsymbol{x}}_i)}{p_i(\hat{\boldsymbol{x}}_i)}\right] = \sum_{i=1}^{n-1} (\beta_i - \beta_{i+1}) E(\hat{\boldsymbol{x}}_i) \tag{6.18}$$

$$\check{F} = \log\left[\prod_{i=0}^{n-1} \frac{p_i(\check{\boldsymbol{x}}_i)}{p_{i+1}(\check{\boldsymbol{x}}_i)}\right] = \sum_{i=1}^{n-1} (\beta_i - \beta_{i+1}) E(\check{\boldsymbol{x}}_i), \tag{6.19}$$

and β_i is the inverse of temperature, t_i.

When implementing this algorithm there is a necessity to choose both n and t_n. Pragmatically, it is necessary to choose t_n first as this is the parameter that permits movement around the state–space to isolated modes. This is achieved by sampling $\pi(\boldsymbol{x})$ at several temperatures until a temperature is found that permits all modes to be reached. n is chosen so that the system is warmed and cooled at a rate that allows for good acceptance rates – again this is achieved through pilot runs.

[2]The proof of detailed balance is provided in Neal (1995b).

6.4.2 Mode jumping proposals in MCMC

In Tjelmeland and Hegstad (1999) a transition kernel is defined that proposes jumps to isolated regions (or modes) in state-space. The transition kernel is different from the majority of other transition kernels because it is asymmetric. This property requires careful handling to ensure that detailed balance is maintained. There is no detailed balance proof for this method presented in Tjelmeland and Hegstad (1999) and Tjelmeland and Hegstad (2001), but we can confirm that detailed balance holds; the proof is provided in Appendix H. The transition kernel is defined to be

$$T(A|\boldsymbol{x}) = \frac{1}{2}\int_A q_0(\boldsymbol{x}^*|\boldsymbol{x})\alpha_{0,1}(\boldsymbol{x}^*|\boldsymbol{x})\,d\boldsymbol{x}^* + \frac{1}{2}\int_A q_1(\boldsymbol{x}^*|\boldsymbol{x})\alpha_{1,0}(\boldsymbol{x}^*|\boldsymbol{x})\,d\boldsymbol{x}^* \tag{6.20}$$

where

$$\alpha_{0,1}(\boldsymbol{x}^*|\boldsymbol{x}) = \min\left\{1, \frac{\pi(\boldsymbol{x}^*)q_1(\boldsymbol{x}|\boldsymbol{x}^*)}{\pi(\boldsymbol{x})q_0(\boldsymbol{x}^*|\boldsymbol{x})}\right\}, \tag{6.21}$$

$$\alpha_{1,0}(\boldsymbol{x}^*|\boldsymbol{x}) = \min\left\{1, \frac{\pi(\boldsymbol{x}^*)q_0(\boldsymbol{x}|\boldsymbol{x}^*)}{\pi(\boldsymbol{x})q_1(\boldsymbol{x}^*|\boldsymbol{x})}\right\}, \tag{6.22}$$

and A is a measurable region[3] in state–space (Tjelmeland and Hegstad, 2001). Equations (6.21) and (6.22) are the acceptance probabilities for the candidate state in a new mode; the functions $q_0(\cdot|\cdot)$ and $q_1(\cdot|\cdot)$ represent the conditional probability density functions of the proposal distribution and $\pi(\cdot)$ represents the un-normalised probability density function of interest.

As the proposal distribution is asymmetric we are required to explicitly calculate the conditional probability densities $q_0(\cdot|\cdot)$ and $q_1(\cdot|\cdot)$; this is in contrast to the Metropolis-Hastings transition kernel algorithm where we do not explicitly compute the conditional probabilities, $q(\cdot|\cdot)$, of the proposal distribution.

To propose a candidate state, $\boldsymbol{x}^*$, we first jump to a random point $\boldsymbol{x}+\boldsymbol{\varphi}$ in state–space where $\boldsymbol{\varphi}$ is drawn from a Gaussian distribution with a large variance; then, taking $\boldsymbol{x}+\boldsymbol{\varphi}$ as a starting point, a non-linear optimiser is run until it converges. It is hoped that the finish point of the non-linear optimiser, $\mu(\boldsymbol{x},\boldsymbol{\varphi})$, will be a different mode of $\pi(\boldsymbol{x})$ to that in which $\boldsymbol{x}$ originates, see Figure 6.3. The curvature of the mode is approximated by the Hessian at the point $\mu(\boldsymbol{x},\boldsymbol{\varphi})$, which is used to approximate the inverse covariance matrix of a Gaussian approximation of the mode, $\Sigma^{-1}(\boldsymbol{x},\boldsymbol{\varphi})$. Hence, the mode is approximated by a Gaussian distribution $N(\mu(\boldsymbol{x},\boldsymbol{\varphi}),\Sigma^{-1}(\boldsymbol{x},\boldsymbol{\varphi}))$, from which a sample is drawn to generate the candidate state, $\boldsymbol{x}^*$. The probability density associated with $\boldsymbol{x}^*$ under the Gaussian, $\mathrm{N}(\mu(\boldsymbol{x},\boldsymbol{\varphi}),\Sigma^{-1}(\boldsymbol{x},\boldsymbol{\varphi}))(\boldsymbol{x}^*)$, represents the probability density of the forward step of the proposal distribution, hence $q(\boldsymbol{x}^*|\boldsymbol{x}) = \mathrm{N}(\mu(\boldsymbol{x},\boldsymbol{\varphi}),\Sigma^{-1}(\boldsymbol{x},\boldsymbol{\varphi}))(\boldsymbol{x}^*)$.

The next stage is to compute the probability density $q(\boldsymbol{x}|\boldsymbol{x}^*)$. Starting at the candidate state, $\boldsymbol{x}^*$, we repeat the procedure for generating the candidate state. From $\boldsymbol{x}^*$ a jump is made to $\boldsymbol{x}^*-\boldsymbol{\varphi}$; the non-linear optimiser is run to convergence, taking $\boldsymbol{x}^*-\boldsymbol{\varphi}$ as its initial conditions, where the finish point is $\mu(\boldsymbol{x}^*,\boldsymbol{\varphi})$. The curvature of this mode is also approximated by the Hessian matrix, and taken as an approximation of the inverse covariance matrix of a Gaussian approximation of the mode, $\Sigma^{-1}(\boldsymbol{x}^*,\boldsymbol{\varphi})$. Finally, $q(\boldsymbol{x}|\boldsymbol{x}^*)$ is computed as $\mathrm{N}(\mu(\boldsymbol{x}^*,\boldsymbol{\varphi}),\Sigma^{-1}(\boldsymbol{x}^*,\boldsymbol{\varphi}))(\boldsymbol{x})$. Returning to the mode $\boldsymbol{x}$

[3] Precisely, $\forall A \in \mathcal{F}$ where $\mathcal{F}$ is the Borel σ-field on $\Re^n$.

is not guaranteed with this method, but it is hoped that $x^* - \varphi$ will be in the basin of attraction of the mode in which x sits often enough to ensure efficient sampling.

One final technical detail, which is to ensure detailed balance is maintained, is that for half the proposals the first jump is $x + \varphi$ and the return jump is $x^* - \varphi$; for the other half, the first jump is $x - \varphi$ and the return jump is $x^* + \varphi$, for detail see Appendix H.

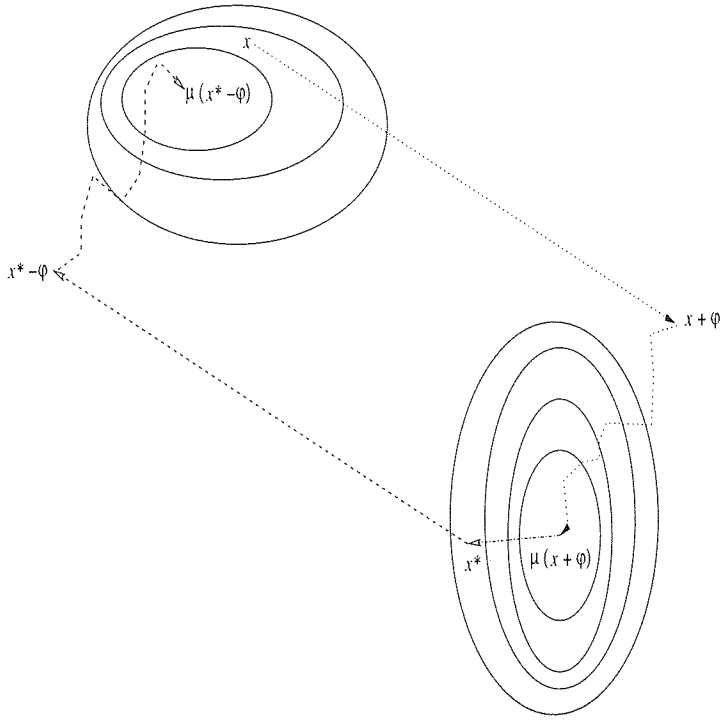


Figure 6.3: Generation of candidate states and proposal probability densities for MJMCMC. Starting at the current state, x, a candidate x^* is generated by jumping to $x + \varphi$, running a non-linear optimiser to $\mu(x, \varphi)$. The mode is approximated by a Gaussian distribution from which the candidate state, x^*, is drawn. The conditional density of the candidate state given the current state, $q(x^*|x)$ is approximated by a Gaussian distribution at $\mu(x, \varphi)$. The procedure is then applied in reverse to generate the conditional distribution of the current state given the candidate state, $q(x|x^*)$. A jump is made to $x - \varphi$, followed by a non-linear optimisation to a mode, $\mu(x^*, \varphi)$, where a Gaussian approximation of the mode is computed and represents the conditional probability density, $q(x|x^*)$.

The whole procedure is summarised as follows.

i. Choose either acceptance probabilities (6.21) or (6.22) with probability one half.

ii. If (6.21) compute the condition probabilities of the proposal distribution:

$$q_0(x^*|x) = N(\mu(T_0(x, \varphi)), \Sigma^{-1}(T_0(x, \varphi)))(x^*), \tag{6.23}$$

$$q_1(x|x^*) = N(\mu(T_1(x^*, \varphi)), \Sigma^{-1}(T_1(x^*, \varphi)))(x), \tag{6.24}$$

where $T_0(x, \varphi) = x + \varphi$ and $T_1(x^*, \varphi) = x^* - \varphi$. The parameters $\mu(T_0(x, \varphi))$ and $\mu(T_1(x^*, \varphi))$ are computed by a local optimisation starting at $T_0(x, \varphi)$ and $T_1(x^*, \varphi)$. It is hoped that some of the time $\mu(T_0(x, \varphi))$ will land in the basin of attraction of a different mode to that of x, whilst $\mu(T_1(x^*, \varphi))$ will remain in the basin of attraction of x.

iii. If (6.22) compute the conditional probabilities:

$$q_0(\boldsymbol{x}|\boldsymbol{x}^*) = N(\mu(T_0(\boldsymbol{x}^*, \boldsymbol{\varphi})), \Sigma^{-1}(T_0(\boldsymbol{x}^*, \boldsymbol{\varphi})))(\boldsymbol{x}^*), \tag{6.25}$$

$$q_1(\boldsymbol{x}^*|\boldsymbol{x}) = N(\mu(T_1(\boldsymbol{x}, \boldsymbol{\varphi})), \Sigma^{-1}(T_1(\boldsymbol{x}, \boldsymbol{\varphi})))(\boldsymbol{x}), \tag{6.26}$$

where $T_0(\boldsymbol{x}^*, \boldsymbol{\varphi}) = \boldsymbol{x}^* + \boldsymbol{\varphi}$ and $T_1(\boldsymbol{x}, \boldsymbol{\varphi}) = \boldsymbol{x} - \boldsymbol{\varphi}$. Choosing the two different transition kernels maintains detailed balance (see Appendix H).

6.4.3 Divide and Conquer

Suppose that some mode finding algorithm has identified two modes of $\pi(\boldsymbol{x})$, $\mathbf{m}_1$ and $\mathbf{m}_2$. The vector $\mathbf{j}_{12} = \mathbf{m}_2 - \mathbf{m}_1$ defines a jump in state–space from $\mathbf{m}_1$ to $\mathbf{m}_2$: $\mathbf{m}_2 = \mathbf{m}_1 + \mathbf{j}_{12}$; to move from $\mathbf{m}_2$ to $\mathbf{m}_1$ we take the reverse jump: $\mathbf{m}_1 = \mathbf{m}_2 - \mathbf{j}_{12}$. To propose a jump from $\mathbf{m}_1$ to $\mathbf{m}_2$, using the Metropolis-Hastings transition kernel, we slightly modify the proposal distribution of (6.12), $q(\cdot|\cdot)$, to incorporate the jump: $q(\boldsymbol{x}^*|\boldsymbol{x}_t + \mathbf{j}_{12})$. Provided that $q(\cdot|\cdot)$ is symmetric then detailed balance is proved in the same way as (6.7) to (6.12) as $q(\boldsymbol{x}^*|\boldsymbol{x}_t + \mathbf{j}_{12}) = q(\boldsymbol{x}_t + \mathbf{j}_{12}|\boldsymbol{x}^*)$. Hence a new sample is generated by the following algorithm.

i. Sample test probability p_{test} from uniform distribution $[0, 1]$.

ii. Propose a new state, $\boldsymbol{x}^*$, by drawing a sample conditionally on the current state $\boldsymbol{x}_t$ and the jump to the next mode $\mathbf{m}_2$:

$$\boldsymbol{x}^* = \boldsymbol{x}_t + \mathbf{j}_{12} + N(0, \sigma^2). \tag{6.27}$$

iii. Compute the acceptance probability:

$$\alpha = \min\left\{1, \frac{\pi(\boldsymbol{x}^*)q((\boldsymbol{x}_t + \mathbf{j}_{12})|\boldsymbol{x}^*)}{\pi(\boldsymbol{x}_t)q(\boldsymbol{x}^*|(\boldsymbol{x}_t + \mathbf{j}_{12}))}\right\}. \tag{6.28}$$

iv. If $\alpha > p_{test}$ then accept proposed new state $\boldsymbol{x}^*$ and set $\boldsymbol{x}_{t+1} = \boldsymbol{x}^*$; else set $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t$; the previous state.

To to ensure that it is possible to jump back from $\mathbf{m}_2$ (should we ever arrive there) we must also propose a jump $\mathbf{m}_1 = \mathbf{m}_2 - \mathbf{j}_{12}$ in our sampler scheme with equal probability to the jump from $\mathbf{m}_1$ to $\mathbf{m}_2$; this maintains detailed balance for the sampler. If there are multiple modes, then the mode to which the sampler jumps is chosen randomly, each with equal probability, and then the candidate state is generated as in (6.27). This method is equivalent to combining the transition kernels by the *mixing* method described in Section 6.2.1 and Brooks (1999).

The practical advantage of this method is that the mode finding is completed off-line, before the sampling commences. This means that specialist mode finding methods may be employed for each particular problem, and that computational effort required to find the modes is not repeated many times during a sampling run. Both TT and MJMCMC require a mode to be found during sampling; this is computationally very expensive and may lead to candidate states being generated from modes with very low probability.

6.5 Convergence

Gelman (1996) describe the dangers that exist if we are not careful about the MCMC simulation. We should be aware of:

i. inappropriate modelling: the assumed model may not be sufficiently complex to describe the data;

ii. errors in programming or computation/calculation: the stationary distribution of the simulation may not be the same as the target distribution; or, the algorithm may well be incorrectly implemented and therefore not converge to any proper distribution;

iii. slow convergence: the simulation remains heavily influenced by the starting state, and therefore inadequately explores the whole state–space, and can become stuck in regions influenced by the initial conditions;

iv. slow mixing: which leads to slow convergence, there are high correlations among consecutive states in the simulated sequence.

Items (i) and (ii) are common to any methods of modelling, but items (iii) and (iv) are more specific to MCMC simulations. Some form of convergence monitoring helps avoid the traps of (iii) and (iv). Therefore, if we are to use the sampling methods for wind field retrieval it is necessary to be able to assess the convergence of the sampling algorithms.

There are many methods for assessing convergence and these are reviewed in Brooks and Roberts (1997). Of those reviewed, the most pragmatic are those which are called *convergence diagnostics*; they generate a statistical analysis of the output of the sampler during the sampler simulation. Of these methods there is one in particular that has been applied and documented with success: it is this method that is used in this thesis. It first appeared in Gelman and Rubin (1992) and has subsequently been developed in various forms, as in Gelman (1996), Brooks and Giudici (1998), whilst being reviewed in Brooks and Roberts (1997).

The principle of this method is to generate two estimates of the target variance Gelman (1996). One estimate over–estimates the target variance, whilst the other under–estimates the target variance. Convergence is said to be met when the two estimates are roughly equal.

To apply the test m parallel simulations are required each of length $2n$. A burn–in period of n iterations is taken and so the monitoring statistics are computed on the $(n+1)^{\text{th}}$ to $2n^{\text{th}}$ iterations from each of the m chains. We monitor some scalar summary statistic of interest, such as the negative log likelihood of the model, or specific parameters of the model.

Two estimates of the variance, $var(\psi)$, of the target distribution are computed, namely W and $\widehat{var}(\psi)$. W is the within sequence variance estimate:

$$W = \frac{1}{m}\sum_{i=1}^{m} s_i^2, \tag{6.29}$$

where

$$s^2 = \frac{1}{n-1}\sum_{j=1}^{n}(\psi_{ij} - \bar{\psi}_{i.})^2 \tag{6.30}$$

and

$$\bar{\psi}_{i.} = \frac{1}{n}\sum_{j=1}^{n}\psi_{ij}. \tag{6.31}$$

Where ψ_{ij} is the j^{th} scalar summary statistic of sample run i. The second, $\widehat{var}(\psi)$, is a weighted sum of the between sequence variance, B, and the within sequence variance W

$$\widehat{var}(\psi) = \frac{n-1}{n}W + \frac{1}{n}B, \tag{6.32}$$

where

$$B = \frac{n}{m-1}\sum_{i=1}^{m}(\bar{\psi}_{i.} - \bar{\psi}_{..})^2 \tag{6.33}$$

and

$$\bar{\psi}_{..} = \frac{1}{m}\sum_{j=1}^{m}\bar{\psi}_{i.}\ . \tag{6.34}$$

As $\widehat{var}(\psi)$ is an *overestimate* of $var(\psi)$ and W is an *underestimate* of $var(\psi)$ then as $n \to \infty$, $\widehat{var}(\psi)$ and W both approach $var(\psi)$, (i.e. $\widehat{var}(\psi)$ is an upper bound on the variance estimate and W is a lower bound on the variance estimate), which is shown in Gelman and Rubin (1992). Convergence is monitored by computing the ratio of the upper and lower bounds:

$$\sqrt{\widehat{R}} = \sqrt{\frac{\widehat{var}(\psi)}{W}}. \tag{6.35}$$

This is called the 'Estimated Potential Scale Reduction', or EPSR. As the simulation converges so the EPSR declines to 1; in practice, an EPSR value of less then 1.1 or 1.2 is taken as indication that convergence has been reached (Gelman, 1996).

6.6 Algorithm selection

Before applying the sampling algorithms to the full wind field model they were assessed on a toy problem designed to represent a wind field. The toy problem is bi-modal probability distribution where the probability mass associated with each mode is known and adjustable. The sampling algorithms were applied to the toy model and assessed by comparing the estimated mass in each mode, the number of Floating Point OperationS (FLOPS) used during sampling and how well the chains have converged using the monitoring algorithm described in Section 6.5. The toy problem is a Gaussian mixture model,

$$\pi(\boldsymbol{x}) = \sum_{j=1}^{2}\alpha_j\phi_j(\boldsymbol{x}|\ \boldsymbol{\mu}_j, \Sigma_j) \tag{6.36}$$

with two kernels. The kernels are Gaussian distributions with full covariance matrices,

$$\phi_j = \frac{1}{(2\pi)^{\frac{c}{2}}|\Sigma_j|^{\frac{1}{2}}}\exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right\}, \tag{6.37}$$

where Σ_j is the covariance matrix, μ_j is the mean, c is the dimension of the Gaussian, and $\boldsymbol{x}$ is the variable. Each kernel in the mixture model represents a simplified wind field, where the the maximum *a posteriori* probability wind vectors are encoded as the mean, μ_j of the kernel; if there are n (and therefore $c = 2n$) wind vectors then μ_j is encoded as

$$[u_{1j}, u_{2j}, \cdots, u_{ij}, \cdots, u_{nj}, v_{1j}, v_{2j}, \cdots, v_{ij}, \cdots, v_{nj}]^T = [\mathbf{u}_j, \mathbf{v}_j]^T \tag{6.38}$$

to represent the local wind vectors in the wind field. The covariance matrix is decomposed in the same way as the prior model (Cornford, 1997; Cornford, 1998a),

$$\Sigma_j = \begin{pmatrix} C_{u_j u_j} & C_{u_j v_j} \\ C_{v_j u_j} & C_{v_j v_j} \end{pmatrix}, \tag{6.39}$$

where $C_{u_j u_j}$ are the variances and cross–correlations of the $\mathbf{u}_j$ components, $C_{v_j v_j}$ are the variances and cross–correlations of the $\mathbf{v}_j$ components and $C_{u_j v_j} = C_{v_j u_j}$ represent the cross–correlations between $\mathbf{u}_j$ and $\mathbf{v}_j$. Hence the local covariance structure for the ij^{th} wind vector, (u_{ij}, v_{ij}), is

$$\begin{pmatrix} \sigma_{C_{u_{ij}u_{ij}}} & \sigma_{C_{u_{ij}v_{ij}}} \\ \sigma_{C_{v_{ij}u_{ij}}} & \sigma_{C_{v_{ij}v_{ij}}} \end{pmatrix}. \tag{6.40}$$

Finally the simulated wind field is plotted on a grid to represent the relative spatial location between each local wind vector; in this case they are plotted on a grid which represents 50 km distance between each wind vector, see Figure 6.4.

Now that we have the structure of the wind field simulation model, we need to populate the parameters. All the local wind vectors, within a mode (or Gaussian), have the same mean and covariance structure. The two modes have wind vectors which are diametrically opposed to simulate the directional ambiguity which occurs in the local wind vector measurements. The local covariance structures are designed to represent the worst case that the samplers may encounter; they are symmetric and have orthogonal spreads in the u and v axes. Varying the mixing coefficients α_j changes the probability mass associated with each mode, thereby simulating modes of the wind field with different masses. The values are summarised in Table 6.1.

The simulation model is designed to represent wind fields on a square grid of any size; for the first benchmarking experiments a 2x2 grid is used; for the final experiment a 10x10 grid is used as a more realistic size and is designed to simulate a real wind field.

6.7 Experiments of wind field simulation

Two groups of experiments were carried out on the toy problem wind field simulation. The first group were benchmarking exercises which compared the performance of the different sampling algorithms and showed that the DC algorithm was the most efficient and best suited to the application; the second group were a simulations on 10x10 grid to assess how well the DC algorithm performs at the higher dimensions.

Label	Parameter	Mode $j = 1$	Mode $j = 2$
Local covariance	$\begin{pmatrix} \sigma_{C_{u_{ij}u_{ij}}} & \sigma_{C_{u_{ij}v_{ij}}} \\ \sigma_{C_{v_{ij}u_{ij}}} & \sigma_{C_{v_{ij}v_{ij}}} \end{pmatrix}$	$\begin{pmatrix} 6 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 6 \end{pmatrix}$
Local mean wind vector	(u_{ij}, v_{ij})	$(10, 0)$	$(-10, 0)$
Mode mass ratio, 1^{st} experiment	α_j	0.5	0.5
Mode mass ratio, 2^{nd} experiment	α_j	0.8	0.2

Table 6.1: Parameterisation of the toy problem. The two modes have wind vectors which are diametrically opposed to simulate the directional ambiguity in the wind field modes, and the local covariance structures are designed to represent the worst case that the samplers may encounter in which they are symmetric and have orthogonal spreads in the u and v axis. Varying the mixing coefficients α_j changes the probability mass associated with each mode, thereby simulating modes of the wind-field with different masses. Also, see Figure 6.4, where the variables are visualised for a four vector ($c = 8$) simulation.

6.7.1 First experiments – algorithm comparison

There are several parameters to adjust for each of the samplers; pilot runs were taken to establish ideal parameters before the full experiments were taken. The samplers were configured using the mixing method described in Brooks (1999) to combine the in–mode and mode jumping transition kernels. The ratio of switching between in–mode and mode jumping proposal for these experiments was 0.5. The in–mode transition kernel was the 'vanilla' Metropolis–Hastings algorithm with a standard deviation of 0.25. In the first experiment the toy problem had four wind vectors (that is a 2x2 grid) and an equal mass associated with each mode.

The TT algorithm was found to have the highest overall acceptance rates which are due to the high acceptance rates associated with the mode jumping transition kernel, see Table 6.2 columns 5 and 6. The sampler was configured to take $40,000$ samples of which every 10^{th} was stored. There were 200 temperature steps for each tempering, that is, 100 to 'warm' and 100 to 'cool'; tuning required several pilot runs of 'vanilla' MCMC at varying temperatures until the whole state space could be satisfactorily reached, and several TT runs to ensure that there were sufficient temperatures steps to ensure mode jump proposals were accepted.

Even after several tuning experiments the MJMCMC algorithm did not attain the acceptance rates of the TT and DC algorithms. Two values are given, as examples, for the variance of the jumping proposal distribution: 5.0 and 7.5. With the first value there were no mode jumps, and with the second value the sampler mode jumped but the acceptance rate was low in comparison to the other two methods. The sampler was run for $100,000$ iterations, again storing every 10^{th} sample.

The DC technique was straight forward to tune; the standard deviation of the mode jumping proposal was set to that of the in–mode proposal distribution at 0.25. The acceptance rates of the mode jumping proposals were lower than that of TT but higher than MJMCMC, and allowed practical sampling. The sampler was run for $250,000$ samples of which every 10^{th} was stored.

The convergence of these sample runs was assessed using the algorithm described in Section 6.5.

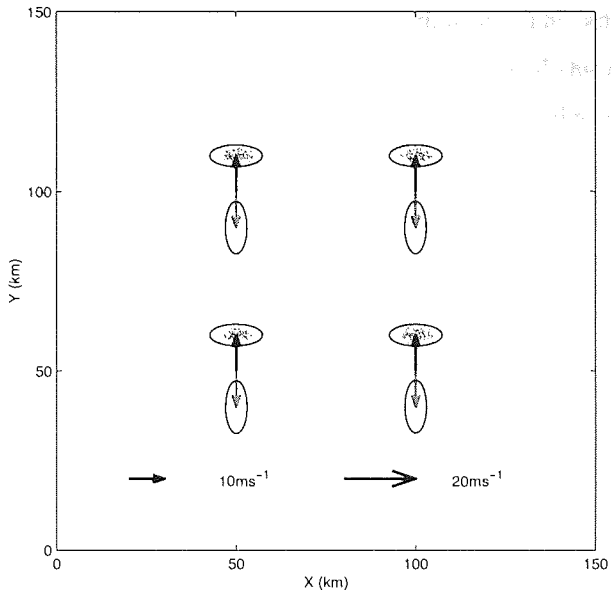


Figure 6.4: Toy problem to simulate the posterior distribution of the wind field model. The model is a mixture of two Gaussians. Each Gaussian represents a mode in the posterior distribution. The covariance structure of each mode is designed to be difficult to resolve and is represented by the ellipses in the figure which are plotted at three standard deviations from the mode. This represents the possible variance in wind speed and direction for each wind vector. The mean of each Gaussian represents the MAP wind vector in each local cell; for the mode coded in black the mean of each wind vector is $(u_{mean}, v_{mean}) = (10, 0)$, whilst for the mode coded in grey the mean of each wind vector is $(u_{mean}, v_{mean}) = (-10, 0)$. Hence the modes are diametrically opposed. The covariance structure associated with each local wind vector is designed to represent the worst possible case for the samplers (see text).

The final run, from which the results are calculated, was constructed by taking the last quarter of each sample run[4]. The convergence statistics were computed for each variable and for the negative log likelihood. These statistics are reported in Appendix I. Convergence was achieved for the TT and DC algorithms. For the MJMCMC algorithm convergence was not satisfactorily reached. For the first experiment the convergence statistics in Table I.3 suggest that convergence has been reached, but when the results are further analysed it is apparent that the algorithm has remained in one mode. For the second experiment mode jumping occurred, but the convergence statics show that convergence has not been reached for the the v chains, Table I.4, as the EPSR value is greater than the recommended 1.2.

The performance of these samplers is summarised in the first four rows of Table 6.2. The TT algorithm is the most efficient in terms of acceptance rates, followed by the DC and MJMCMC algorithms. The mass of each mode is estimated by the time that the sampler spends in each mode (Marinari and Parisi, 1992; Neal, 1995b). The results show that TT and DC are the best at approximating the mass, whilst for one experiment the MJMCMC remains stuck in one mode, and for the second experiment performs marginally worse than the other two algorithms; however we must remind ourselves that convergence of MJMCMC was not reached for the second experiment.

[4]Unless otherwise stated, from now on all the results from sampling are computed from a sample run that is constructed from the final quarter of each individual run used in the convergence testing.

Computationally there is a large difference in performance. The advantage gained by TT in the acceptance ratios is offset by the high computational demand of the algorithm; it is two orders of magnitude more computationally intensive than the MJMCMC algorithm. However, when the MJMCMC algorithm is compared to the DC algorithm the results show that the DC algorithm is far more efficient. Therefore the apparent loss of efficiency of the DC acceptance rates and higher numbers of samples required are more then offset by the computation efficiency of the algorithm.

Given the poor performance of the MJMCMC algorithm on the toy problem, this method is rejected as un-suitable for the wind field problem. The poor performance is attributed to the large random step taken before the optimisation. With this type of problem, where the modes are well separated, there is a large of area of state space with near zero probability. The large jump phase of the kernel spends a lot of time in this area, where the optimiser has insufficient information to find modes, and hence many proposals are rejected.

The TT and DC algorithms were further compared by running the samplers on the toy problem with the same number of wind vectors, but with the ratio of masses for the two modes set to 80 : 20. The convergence statistics are given in Table I.5 and Table I.6, which show that the samplers have converged. The performance statistics are presented as the last two rows in Table 6.2. This time the DC algorithm does better than the TT algorithm. Again, the TT algorithm is computationally very inefficient when compared to the DC algorithm. In real time, running on the same computer, the TT algorithm took several *days* to generate a single run, whilst the DC algorithm took about *two hours*. On these grounds alone the TT algorithm is rejected as unsuitable for the wind field experiment, where the dimension of the problem is at least 200, as opposed to 8 in the toy problem.

6.7.2 Second experiments: benchmarking the DC algorithm

Having shown that the DC algorithm is the most computationally efficient and comparable in accuracy when estimating the mass of separated modes with TT we choose the DC method as the best suited for application to the wind field retrieval problem.

Three further experiments were completed using this algorithm. The first two were computed on the toy problem with four wind vectors, but with the local covariance structures of the two modes being the same, (see Figure 6.5); in the case studies, Section 6.8, we shall see that this is a more realistic model for simulating the wind vector's local covariance structure. Two sampler runs were completed, one for equal masses on each mode, and one with an 80 : 20 split. The results are shown, after running the convergence diagnostics (Table I.7 and Table I.8), in Table 6.3. The results are accurate estimates for the masses of each mode, whilst being computationally efficient and having overall acceptance rates for proposals greater than 70%.

The third experiment is computed on the toy problem with a 10x10 grid designed to simulate the dimensionality of the wind fields found in the real problem. The local covariance structure of each mode was given a less extreme spread in the u and v (see Table 6.4). A visualisation of the model is shown in Figure 6.6. The DC sampler was run five times with different initial values, with each

Sampler type	Ratio of mode mass M1:M2	Mass estimate M1	Mass estimate M2	Acceptance rate for overall run	Acceptance rate for mode jumps	Mean FLOPS per run	Std FLOPS per run
DC	50:50	52.27	47.73	49.73	41.18	$5.6104e+02$	$1.2140e+01$
TT	50:50	53.40	46.60	69.40	63.19	$2.4895e+07$	$8.2675e+06$
MJMCMC	50:50	0.00	100.00	41.11	6.91	$5.9368e+05$	$1.8832e+05$
MJMCMC	50:50	44.21	55.79	40.61	5.80	$6.1658e+05$	$1.9520e+05$
DC	80:20	80.24	19.76	68.38	40.26	$5.3237e+02$	$1.1155e+01$
TT	80:20	77.16	22.84	67.61	59.61	$2.5133e+07$	$8.0356e+06$

Table 6.2: Results of running each of the sampling algorithms on the toy problem with four wind vectors. The first set of results, for modes with equal mass, disqualify the MJMCMC algorithm due to its low mode jumping acceptance rate. The second set of results are for the DC and TT algorithms where the mode ratio is split 80:20 and show the DC algorithm to be comparable in accuracy to the TT algorithm.

Sampler type	Ratio of mode mass M1:M2	Mass estimate M1	Mass estimate M2	Acceptance rate for overall run	Acceptance rate for mode jumps	Mean FLOPS per run	Std FLOPS per run
DC	50:50	49.92	50.08	75.24	75.19	$5.3262e+02$	$1.1538e+01$
DC	80:20	79.78	20.22	71.70	57.21	$5.3250e+02$	$1.1352e+01$

Table 6.3: Results of running DC algorithm on the toy problem with identical covariance structure in each mode.

Label	Parameter	Mode $j = 1$	Mode $j = 2$
Local covariance	$\begin{pmatrix} \sigma_{C_{u_{ij}u_{ij}}} & \sigma_{C_{u_{ij}v_{ij}}} \\ \sigma_{C_{v_{ij}u_{ij}}} & \sigma_{C_{v_{ij}v_{ij}}} \end{pmatrix}$	$\begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}$	$\begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix}$
Local mean wind vector	(u_{ij}, v_{ij})	$(10, 0)$	$(-10, 0)$
Mode mass ratio, 2^{nd} experiment	α_j	0.8	0.2

Table 6.4: Parameterisation of the toy problem for 10x10 grid, $c = 200$. The two modes have wind vectors which are diametrically opposed to simulate the directional ambiguity in the wind field modes, the local covariance structures are designed to represent modes with different degrees of certainty in wind speed and direction. Varying the mixing coefficients α_j changes the probability mass associated with each mode, thereby simulating modes of the wind-field with different masses. Also, see Figure 6.6, where the variables are visualised for on a 2x2 and 10x10 grid.

Mass estimate M1	Mass estimate M2	Acceptance rate for overall run	Acceptance rate for mode jumps	Mean FLOPS per run	Std FLOPS per run
80.05	19.95	53.83	45.53	$2.0726e+06$	$8.0936e+02$

Table 6.5: Results of running the DC algorithm on the toy problem with a 10x10 grid.

run consisting of two million samples, sub–sampled every twenty five to create a stored sample run of eighty thousand. The convergence monitoring algorithm was applied, and confirmed that convergence has been reached (see Figure 6.7); all the parameters and the negative log likelihood summary statistic are below the 1.2 threshold for the EPSR. These results are shown in Table 6.5 and show that the approximation of the mass in each mode is accurate. We note that the computational cost is greater per sample drawn due to the increased dimensionality of the problem. With these results we are now ready to apply the DC sampler to the real world problem. In the next section the DC algorithm is applied to five case study real world wind fields and a test set of 100 real world wind fields.

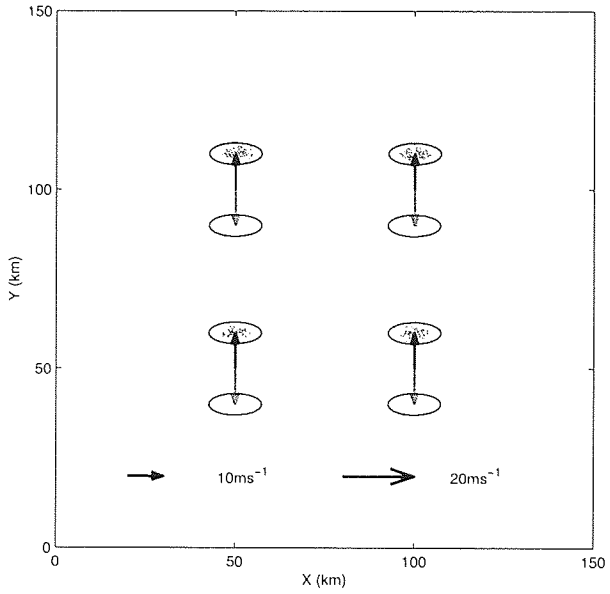
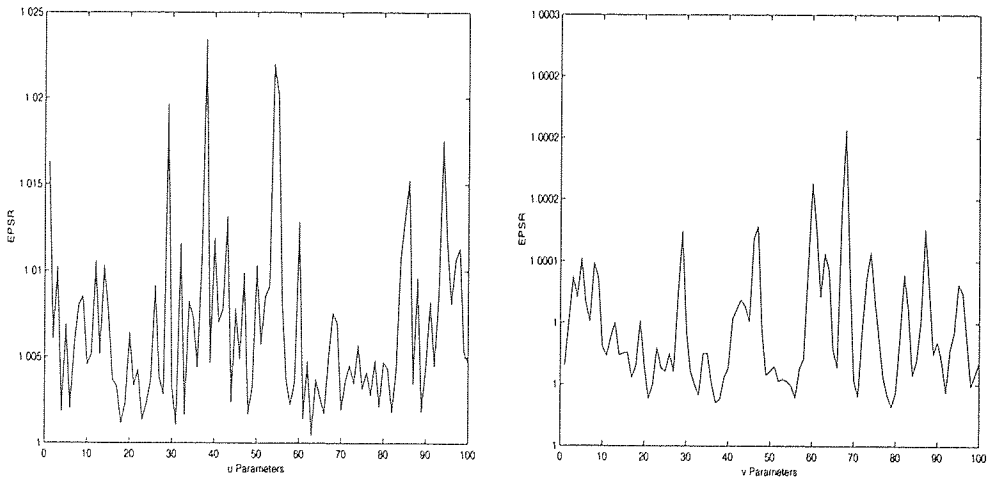


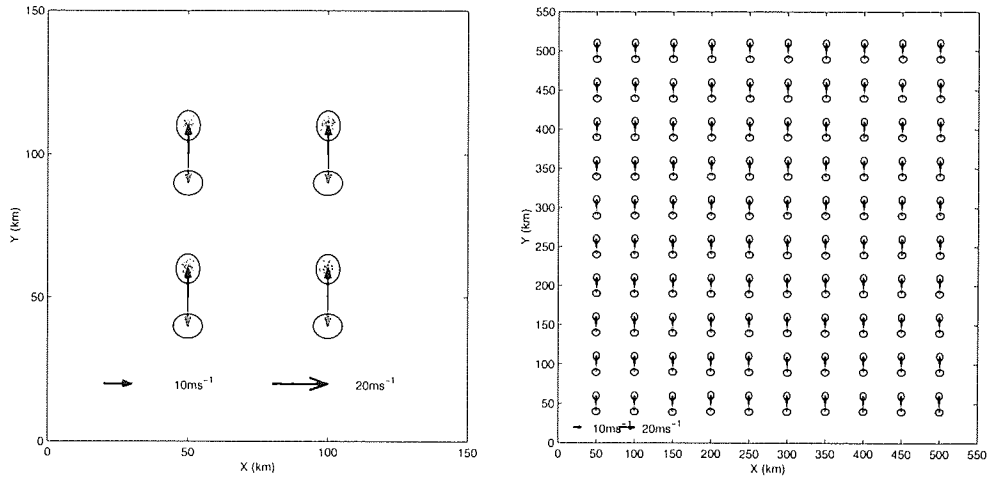
Figure 6.5: Toy problem to simulate the posterior distribution of the wind field model. The model is a mixture of two Gaussians. Each Gaussian represents a mode in the posterior distribution. The covariance structure of each mode is designed to represent greater uncertainty in wind direction than in wind speed, hence they are axis aligned, but not spherical. The ellipses in the figure which are plotted at three standard deviations from the mode. This represents the possible variance in wind speed and direction for each wind vector. The mean of each Gaussian represents the MAP wind vector in each local cell; for the mode coded in black the mean of each wind vector is $(u_{mean}, v_{mean}) = (10, 0)$, whilst for the mode coded in grey the mean of each wind vector is $(u_{mean}, v_{mean}) = (-10, 0)$. Hence the modes are diametrically opposed.



(a) Convergence statistics for u parameters.

(b) Convergence statistics for v parameters.

Figure 6.7: Convergence statistics for the 100 wind vector toy problem. Convergence statistics are computed for each parameter, where we see that they are all below the threshold of 1.1. Convergence statistics were also computed for the energy function, the value being 1.0004.



(a) Four wind vectors to show detail of local covariance structure.

(b) Visualisation of the full 100 wind vector simulation.

Figure 6.6: Toy problem to simulate the posterior distribution of the wind field model for higher dimensions. The model is a mixture of two Gaussians. Each Gaussian represents a mode in the posterior distribution. The covariance structure of each mode is designed to test the algorithm in high dimensions by making one mode more uncertain in wind direction than in wind speed, whilst the other is more uncertain in wind speed than wind direction, but not to the extreme of the first model. The ellipses in the figure are plotted at three standard deviations from the mode and represents the possible variance in wind speed and direction for each wind vector. The mean of each Gaussian represents the MAP wind vector in each local cell; for the mode coded in black the mean of each wind vector is $(u_{mean}, v_{mean}) = (10, 0)$, whilst for the mode coded in grey the mean of each wind vector is $(u_{mean}, v_{mean}) = (-10, 0)$. Hence the modes are diametrically opposed.

6.8 Case studies

In this section five wind field cases are studied which exemplify where the DC sampling technique may improve upon autonomous predictions made by choosing the MAP solution. In each case the retrieved wind field using non–autonomous methods was in agreement with the target NWP wind field. However, in some cases the MAP retrieved wind field was not correct due to the retrieved wind field being in the opposite direction to the target.

The first two case studies are examples of homogeneous wind fields which make up the majority of the targets in this application. For these case studies the MAP solution has wind vectors in the opposite direction to those in the target, and the second most probable wind field agrees with the target.

The third case study is of a low pressure system which means that the wind vectors are rotating in an anti–clockwise direction around a centre. For this case study the MAP solution is incorrect and it is the *third* most probable solution that agrees with the target.

The final two case studies have wind fields that contain frontal systems. A front is defined as a boundary between to air masses which are thermodynamically homogeneous; a front is identified by a sharp change in wind direction which is also cyclonic. In one of these cases the MAP derived wind field is the correct solution, whilst in the other it is the second most probable wind field that is correct.

The analysis is divided into two stages. In the first stage the four most probable modes (chosen using the methods described in Section 5.3) are analysed individually. Samples are drawn from each mode using MCMC. The local covariance structure, which represents uncertainty in wind speed and direction, of each wind field is estimated and visualised on a 2-D grid. In the second stage the DC algorithm used for wind field retrieval.

Sampling

For both the in–mode sampling and the DC algorithm a Metropolis-Hastings transition kernel transition kernel was employed. However, the usual proposal distribution within the 'vanilla' method is a spherical Gaussian probability distribution, as in (6.7). However, this may not be an optimal choice due to the non–spherical nature of the distribution being sampled. For the case studies the spherical proposal distribution was replaced by the prior distribution of the wind field. This distribution is a zero mean, vector Gaussian process which is described by a full covariance matrix. The Gaussian probability distribution is symmetric, and hence may be employed as a proposal distribution in the Metropolis–Hastings algorithm as it satisfies the detailed balance condition. Using the prior distribution as a proposal distribution will ensure that the proposed changes to the wind field are random, but smooth, and are thus more likely to be accepted.

Stage 1: MCMC of the modes

For each case study the four most probable modes were sampled independently to gain an understanding of their underlying structure. There where five chains run from different starting points. Each chain consisted of $41,000$ samples (where the chain is well converged) which are sub–sampled every twenty-five samples from a chain of $1,025,000$ samples. The samples were then visualised on a 2–d grid which represents a wind field. The mode located by the mode finding algorithm is plotted in blue and the mean of the samples is plotted in red. The local covariance for each wind vector is estimated from the samples, and an ellipse is plotted over the mean of the sample at 3 standard deviations from the sample mean. This indicated the *local* variability in each wind vector. This is of interest because if the local modes have similar structures to the first toy problem in Section 6.6 then the mode jumping sampling algorithm is unlikely to work. However, if the local modes are of similar structure and axis aligned then the sampler can be applied with confidence.

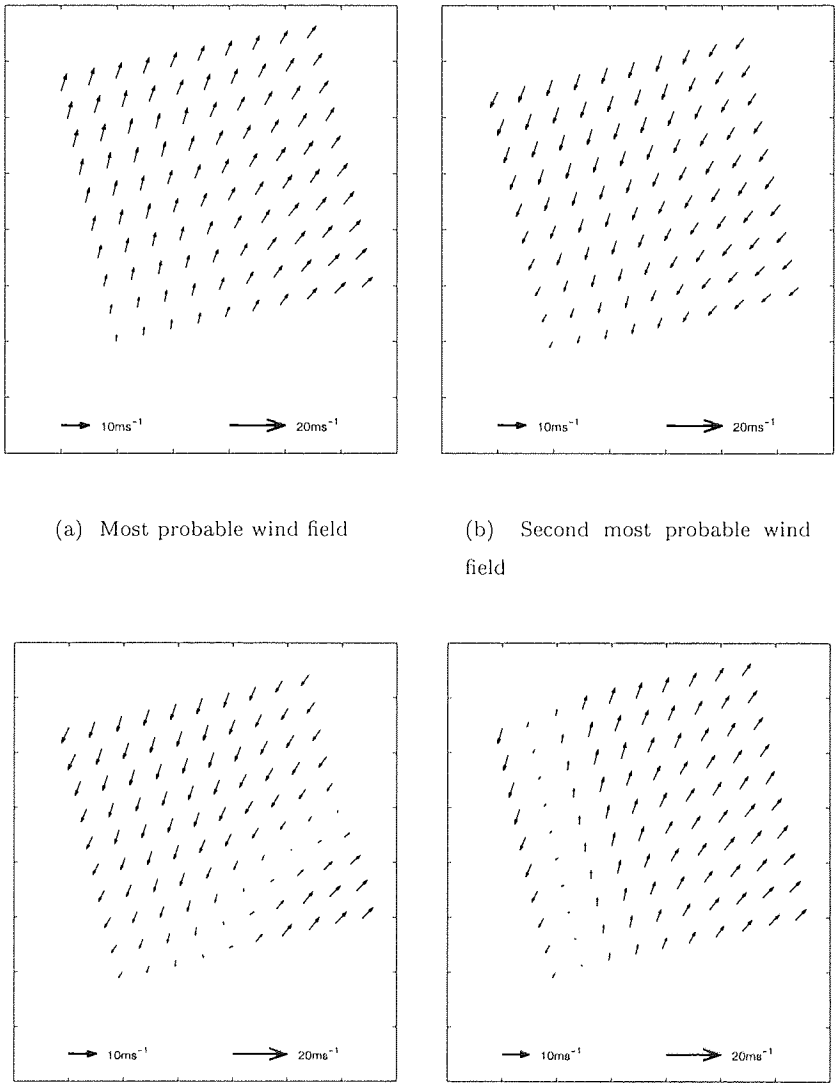
Stage 2: Mode mass estimation

In some cases the third and fourth most probable modes appear unlikely to have any significant probability mass associated with them. Hence, a pilot run (a single run without convergence testing) of the DC algorithm is made to establish if any significant probability mass resides in the third and fourth most probable modes. If the pilot run suggests a significant proportion of the mass is associated with the third and fourth most probable modes then they are included in the main experiment; otherwise the first and second most probable modes are used in the main experiment.

For the main experiment a total of two million samples are drawn from the posterior distribution, which are then sub–sampled every twenty five to create chains of $80,000$ samples. Five chains are generated, starting at different positions, and convergence is monitored using the methods described in Section 6.5. A mode jumping proposal step was generated with probability one half, where the in–mode and mode jumping proposals where chained together. The final chain is constructed and the mass of each mode is estimated by the amount of time that the sampler spends in each mode.

6.8.1 Case study 1: Homogeneous wind field I

This example was chosen because the MAP retrieved wind field is in the opposite direction to the target NWP wind field, and that the second most probable wind field is correct and is the same wind field as that derived from non-autonomous ambiguity removal. The first two most probable modes have similar negative log likelihoods whilst the third and forth modes have lower log likelihoods and appear to be unlikely solutions to the problem, these modes are shown in Figure 6.8.



(a) Most probable wind field

(b) Second most probable wind field

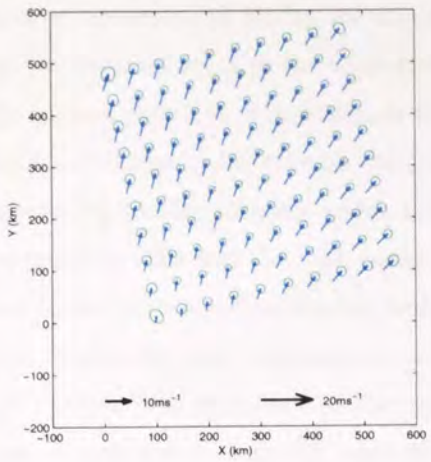
(c) Third most probable wind-field

(d) Fourth the most probable wind field

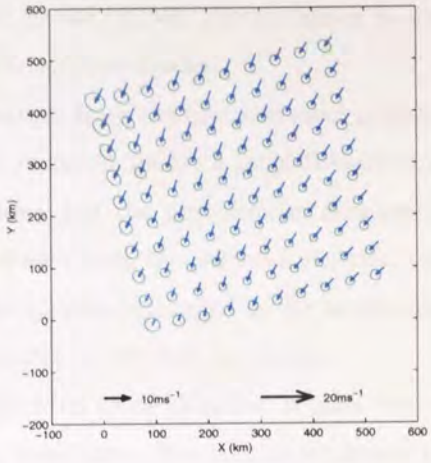
Figure 6.8: Case study 1: the four most probable modes.

The first stage of the analysis is to sample from the individual modes (see Figure J.1 (a) for convergence statistics). The results of the sampling are plotted in Figure 6.9. The first and second most probable modes show that the mean of the samples is a good estimate of the mode found through

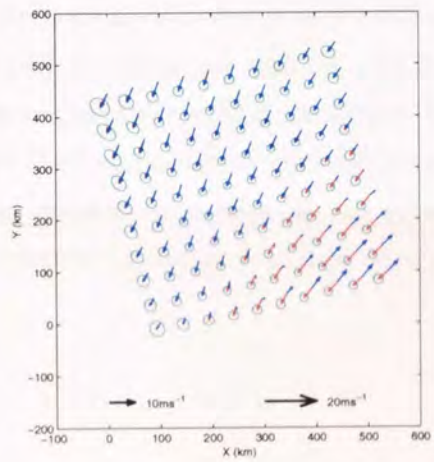
the mode finding exercise and that the local covariance structures are similar.



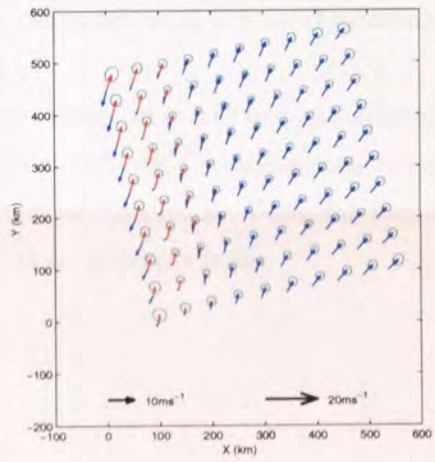
(a) Mode 1, mean sample energy: −264.0495



(b) Mode 2, mean sample energy: −257.6600



(c) Mode 3, mean sample energy: −257.6909



(d) Mode 4, mean sample energy: −264.0385

Figure 6.9: Case study 1: Visualisation of the structure of the four most probable modes after sampling. The blue wind vectors represent the modes identified in the mode finding exercise, the red wind vectors represent the sample mean. The cyan ellipses represent an estimate of the local covariance structure, plotted at three standard deviations from the sample mean. Modes 1 and 2 are opposite in direction to each with similar local structures. Modes 3 and 4 turn out to be sub–optimal modes of modes 2 and 1 respectively - the sample mean, shown in red, confirms this.

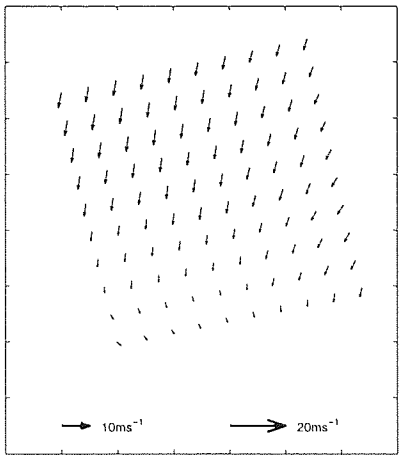
The results of sampling from the third and fourth most probable modes yield interesting results. Inspection of Figure 6.9 (c) and Figure 6.9 (d) reveals some red wind vectors which are distinct from the blue wind vectors. This indicates that the estimate of the mode through sampling is different from that through optimisation. Further computation shows that the mean energies of the modes found through sampling are close to the first and second most probable modes and that the mean of

the samples are also close to those of the second and first modes respectively. This suggests that the modes found by the optimisation routine are local modes, but meteorologically distinct, close to or within the basin of attraction of modes one and two; unlike MCMC, the optimiser is unable to take 'up–hill' steps and becomes stuck in the local and sub–optimal modes.

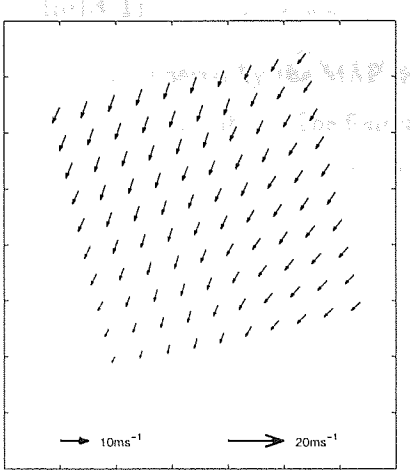
The results suggest that it is only necessary to sample from the first two most probable modes in this case. This is not unreasonable because the other modes found are a large distance from these two modes in negative log–likelihood space which indicates that the probabilities associated with them are tiny in comparison. The fact that the samples drawn from the posterior distribution, that were initialised close to the third and fourth most probable modes, converged to the second and first most probable modes respectively also indicates the dominance of the first two modes.

In order to confirm that only the first two modes need to be sampled, a pilot run, in which all four modes were incorporated in the DC algorithm, was taken. The results confirmed that samples were only drawn from the first two modes with the mass ratio of 13.16 : 86.94 for the 1^{st} to 2^{nd} most probable mode. The full experiment was run with the 1^{st} and 2^{nd} most probable modes used in the DC algorithm (convergence statistics are plotted in Figure J.1 (b)). Finally, the result we have been heading for, the mass associated with each mode was estimated as 10.65% for the first most probable mode and 89.35% for the second most probable mode.

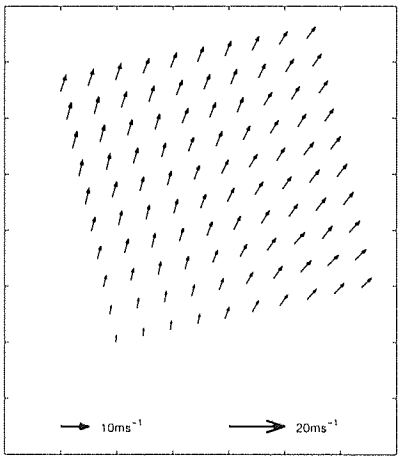
Hence the comparative results of the analysis for the target wind, the non-autonomously retrieved wind field, the MAP autonomously retrieved wind field and the MCMC autonomously retrieved wind field are shown respectively in plots (a), (b), (c) and (d) of Figure 6.10. Using the sampling method for this case selected the correct wind field when the MAP approach failed.



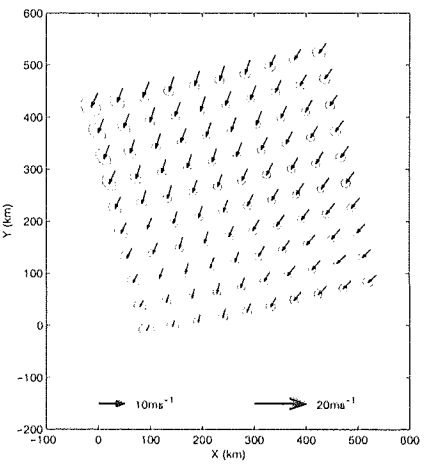
(a) NWP target wind



(b) Non-autonomous derived wind field



(c) Autonomous MAP wind field

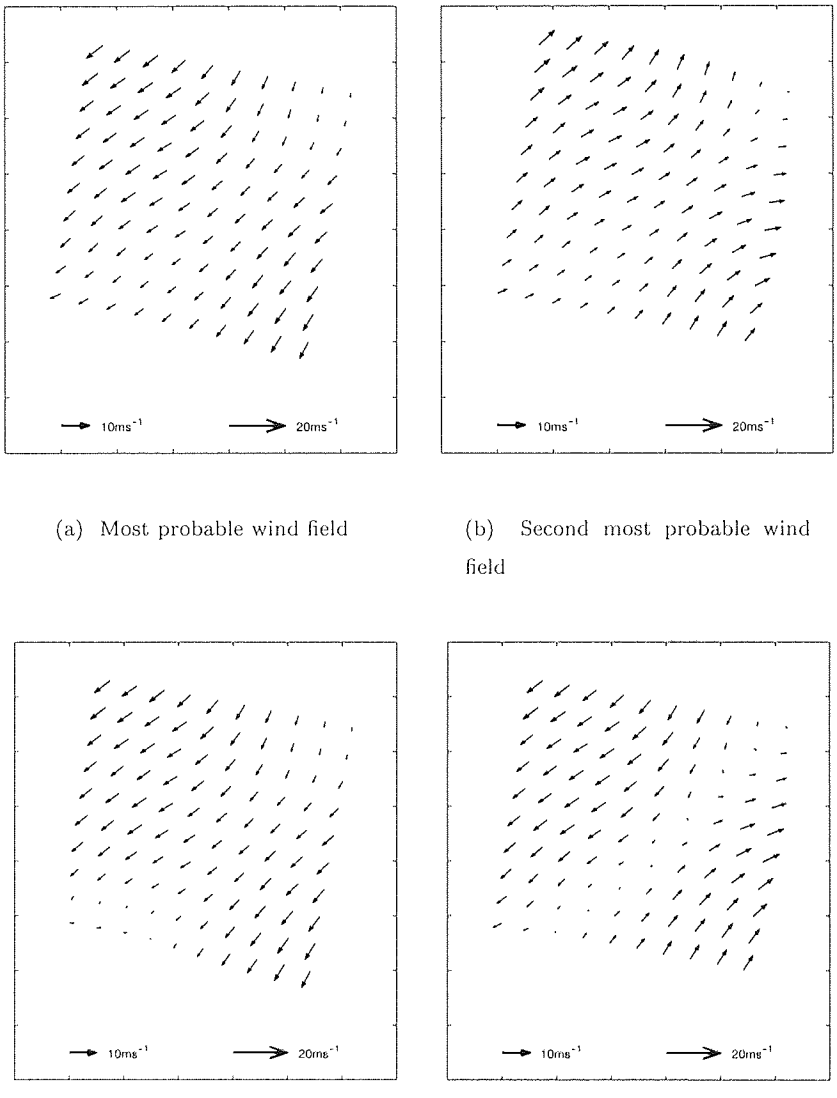


(d) Autonomous MCMC derived wind field

Figure 6.10: Case study 1: the final result.

6.8.2 Case study 2: Homogeneous wind field II

This is example also a homogeneous wind field. The wind field retrieved by the MAP method was in the opposite direction to the target and non–autonomously retrieved solution. The four most probable modes were identified (see Figure 6.11) and samples drawn from them (the convergence statistics are plotted Figure J.2 (a)).



(a) Most probable wind field

(b) Second most probable wind field

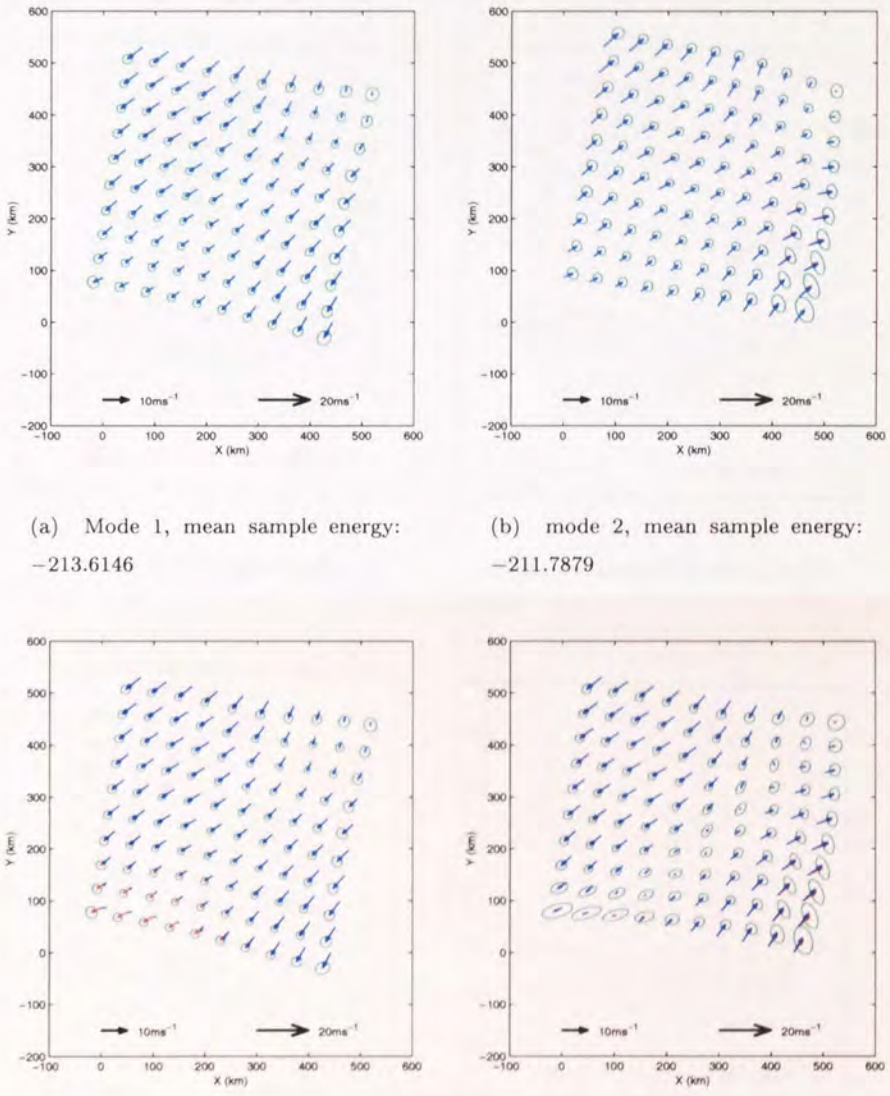
(c) Third most probable wind-field

(d) Fourth the most probable wind field

Figure 6.11: Case study 2: the four most probable modes.

The results from the sampling indicate that the first two most probable modes are the dominate ones in the posterior distribution. The results of the third mode show that this was a local mode within the basin of attraction of the first most probable mode, whilst the fourth mode shows much uncertainty in the edge wind vectors, and has a negative log likelihood that is far away from the first

and second mode – for this mode to contain a significant amount of probability mass we would expect to see a significantly larger variance in all wind vectors when compared with the other modes. This is not the case, and hence, the first and second most probable modes are proposed for the mode jumping algorithm, where the assumption is that these modes contain the significant mass of the posterior distribution. A pilot run of the DC sampler with four modes also confirmed that the estimated mass was split between first and second most probable modes in the ratio 0.28 : 99.72.



(a) Mode 1, mean sample energy: −213.6146

(b) mode 2, mean sample energy: −211.7879

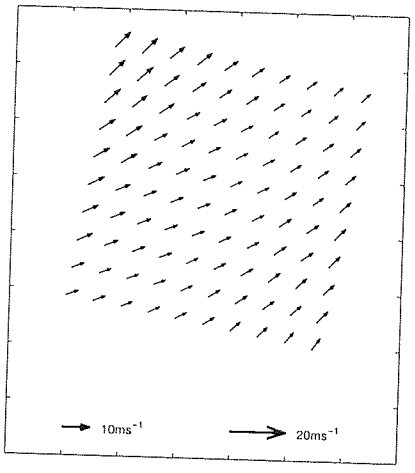
(c) Mode 3, mean sample energy: −213.6439

(d) Mode 4, mean sample energy: −182.1729

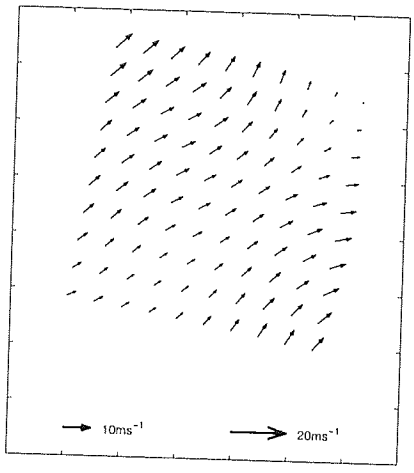
Figure 6.12: Case study 2: Visualisation of structure of the four most probable modes after sampling. The blue wind vectors represent the modes identified in the mode finding exercise, the red wind vectors represent the sample mean. The cyan ellipses represent an estimate of the local covariance structure, plotted at three standard deviations from the sample mean. Modes 1 and 2 are opposite in direction to each other with similar local structures. Mode 3 turns out to be a sub–optimal mode, finally settling in mode 1 as the sample mean (shown in red). Mode 4 is distinct from modes 1 and 2 and is some distance away in log probability space.

The DC algorithm was applied to the two most probable modes (convergence statistics are plotted in Figure J.2 (b)) and the results estimate the masses modes to be: mode 1; 0.25% and mode 2; 99.75%, hence predicting that the second most probable mode is the solution to the problem.

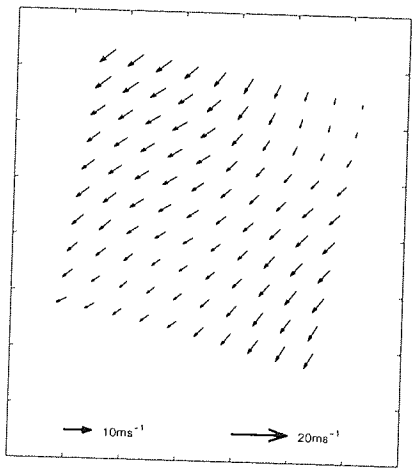
This result correlates with the target wind field and the wind field retrieved by non-autonomous methods; the MAP retrieved wind field is in the opposite direction to the target and sampling retrieved wind fields. The four wind fields are shown in Figure 6.13. Hence, the sampling method retrieved the correct wind field.



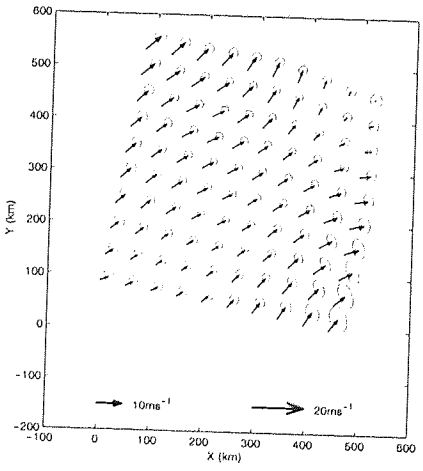
(a) NWP target wind



(b) Non-autonomous derived wind field



(c) Autonomous MAP wind field

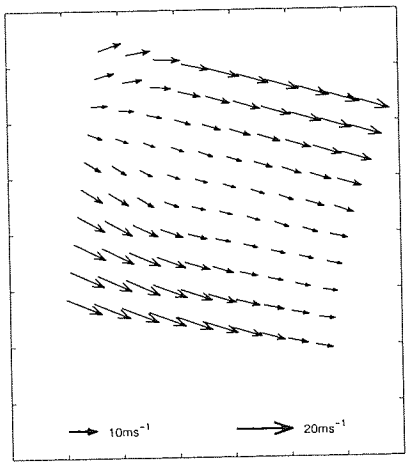


(d) Autonomous MCMC derived wind field

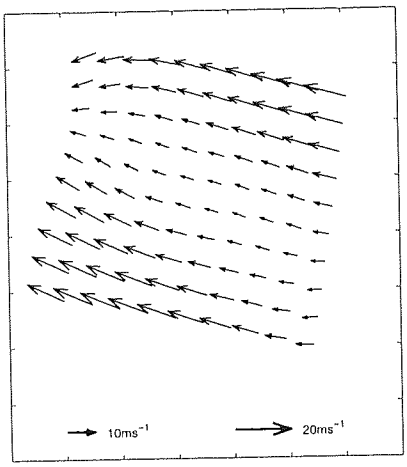
Figure 6.13: Case study 2: the final result.

6.8.3 Case study 3: A low pressure system.

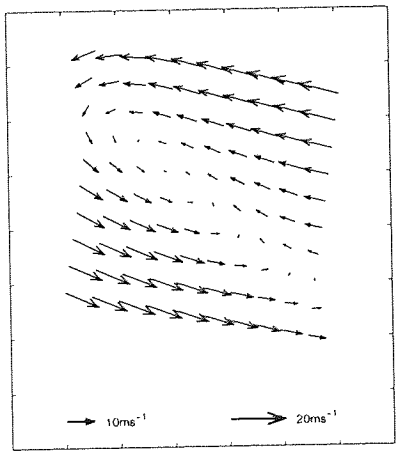
In this case study we encounter a low pressure system with a cyclonic circulation. The non–autonomously derived wind field correlates with the target wind field, but the modes derived from the mode finding exercise are somewhat different. The first two most probable modes, Figure 6.14 (a) and (b), are opposite in direction to one another and show no evidence of a low pressure system as they are homogeneous wind fields. The third and fourth most probable modes, Figure 6.14 (c) and (d), are cyclonic wind fields. The third most probable mode is the wind field retrieved through the non-autonomous method.



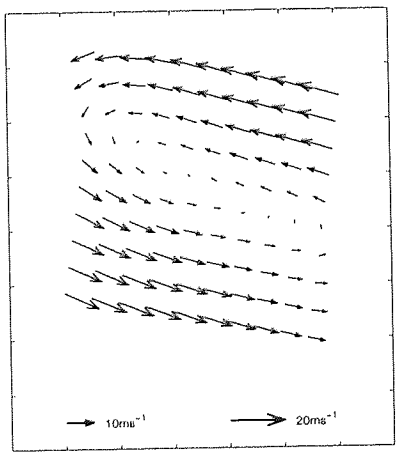
(a) First most probable wind field



(b) Second most probable wind field

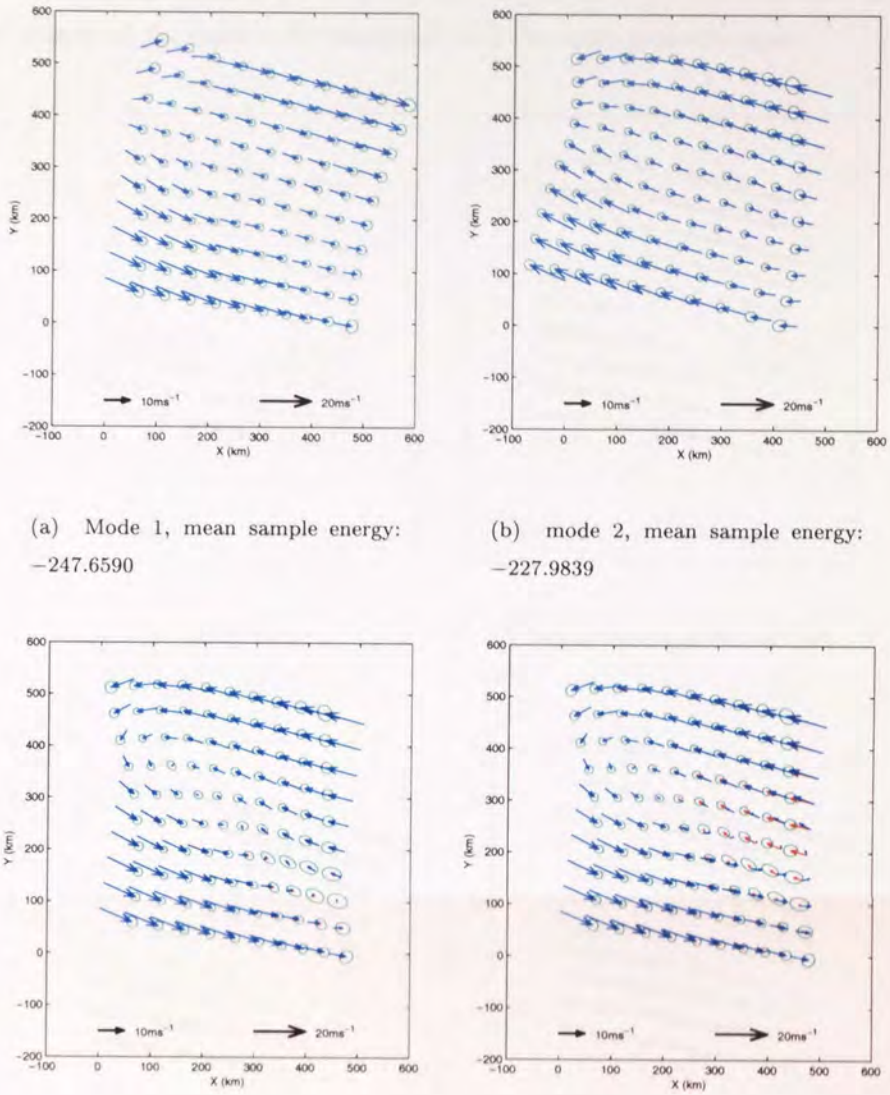


(c) Third most probable wind-field



(d) Fourth the most probable wind field

Figure 6.14: Case study 3: the four most probable modes.



(a) Mode 1, mean sample energy: −247.6590

(b) mode 2, mean sample energy: −227.9839

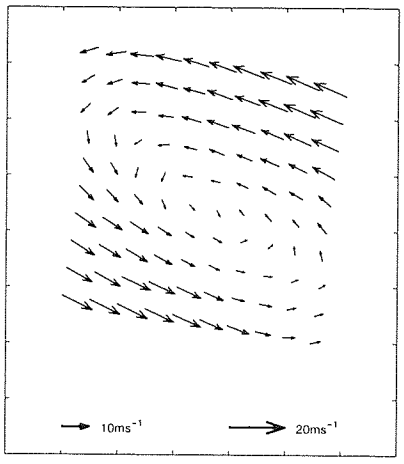
(c) Mode 3, mean sample energy: −201.7991

(d) Mode 4, mean sample energy: −202.0113

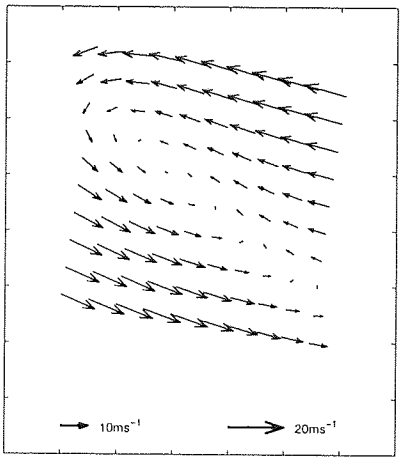
Figure 6.15: Case study 3: visualisation of structure of the four most probable modes after sampling. Modes 1 and 2 are opposite in direction to each other with similar local structures, although neither are representative of the target. Modes 3 and 4 originate from the same mode and represent the target distribution. However, they are a long way from the second most probable mode in terms of their energy, or, negative log–likelihood.

Sampling from the modes reveals that the first, second and third most probable modes are unique. The forth most probable mode appears to be a sub–optimal mode within the basin of attraction of the third most probable mode as the sample mean is close to that of the third most probable mode; the sampling results are visualised in Figure 6.15. The negative log–likelihood of the modes, taken from estimates of the samples, also shows that the first and second most probable modes dominate the posterior distribution, the difference between the third and second most probable modes is large in probability space and would indicate that the third most probable mode would require a large

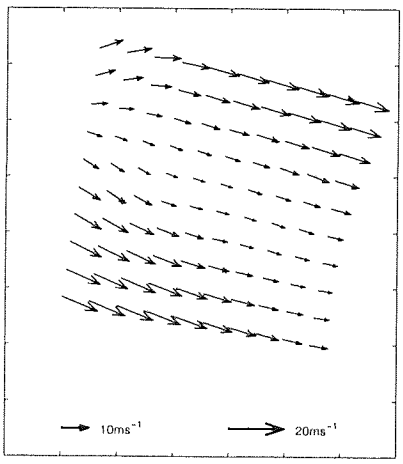
variance for it to have significant mass. A pilot DC run was taken for the first three most probable modes and estimated all the mass to be associated with the most probable mode.



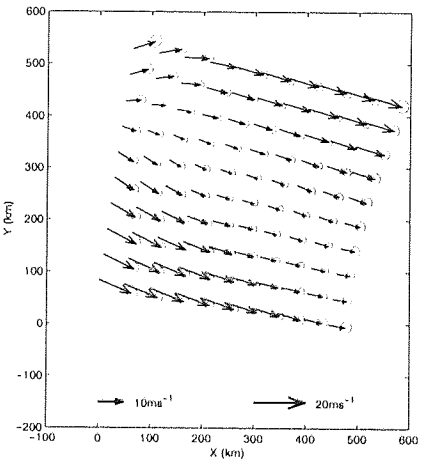
(a) NWP wind



(b) Non-autonomous derived wind field



(c) Autonomous MAP derived wind field



(d) Autonomous MCMC derived wind field

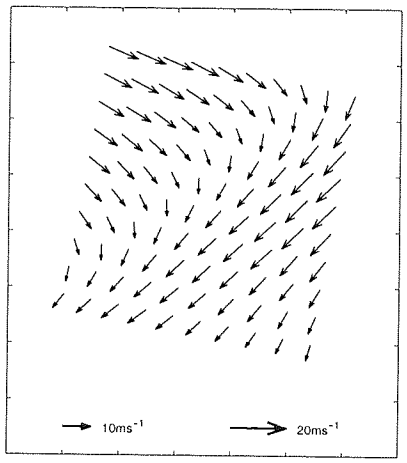
Figure 6.16: Case study 3: the final result.

Running the DC algorithm with the first and second most probable modes confirms this result (convergence statistics: Figure J.3 (b)). This is not a surprising result when the local covariance structures are considered in the visualisation along with the point estimates of the modes, which are some distance apart in negative log-likelihood space. However, there is still the question of why the first two modes are so inaccurate. This is attributed to the prior, which is too smooth. A smooth prior will enforce wind fields like the first two modes over the third mode (Cornford, 1998a). This has been observed for all the lows (and highs) that this model encounters - hence giving justification for further investigation into the prior model and the possibility of developing feature based prior models.

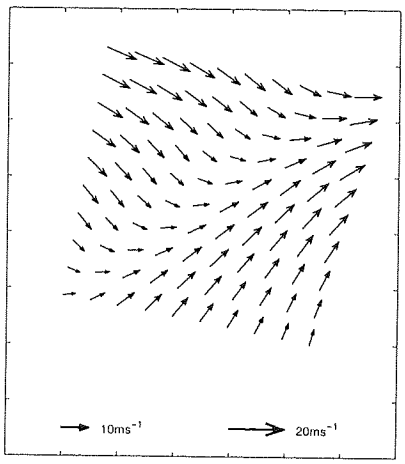
The resulting wind fields from the three methods of retrieval together with the NWP target wind field are shown in Figure 6.16. In this case study both methods of autonomous ambiguity removal are incorrect, whilst the non-autonomous method finds the correct mode.

6.8.4 Case study 4: Frontal wind field I

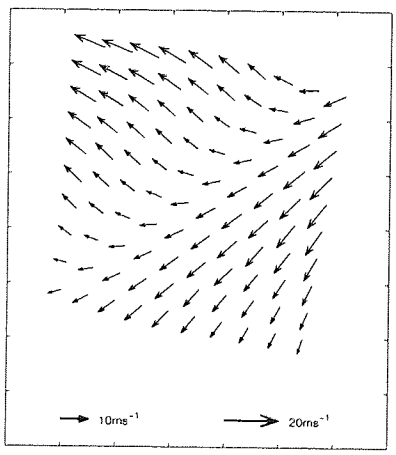
This case study is the first of two examples of a wind field containing a frontal system. The boundary of the front is identified by the distinct change in wind direction, for this example the boundary is approximately from the bottom left hand corner to the top right hand corner of the wind field. The four most probable modes are plotted in Figure 6.17. An interesting observation is that the four possible combinations of fronts with opposing wind directions occur in the first four modes; that is that each side of the front behaves independently of the other, creating four possible wind field permutations.



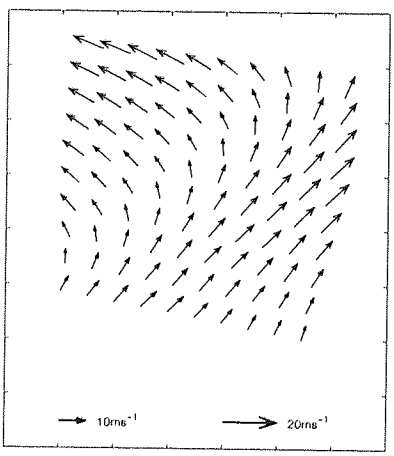
(a) MAP wind field



(b) Second most probable wind field



(c) Third most probable wind-field

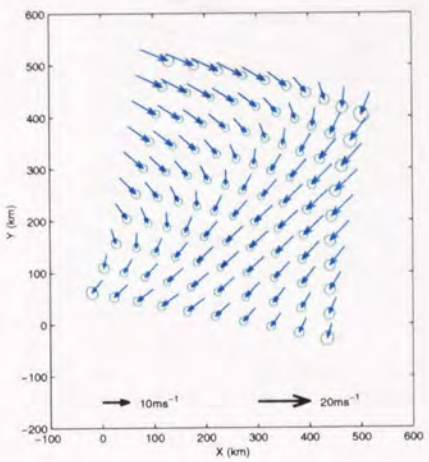


(d) Fourth the most probable wind field

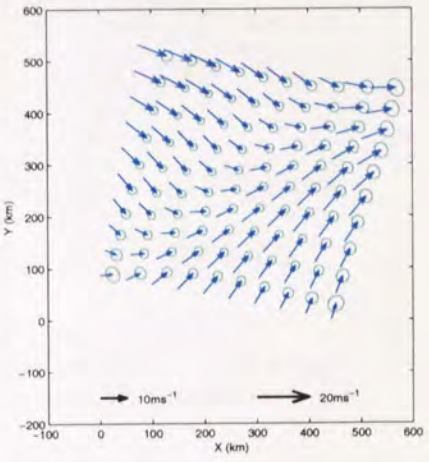
Figure 6.17: Case study 4: the four most probable modes.

Sampling from the modes was performed in the usual manner (convergence statistics are plotted in

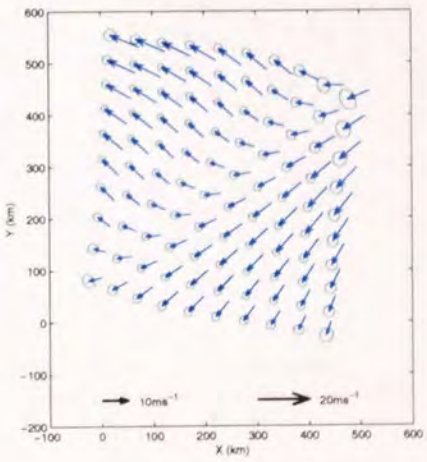
Figure J.4 (a)). Visualisation of the samples confirms that each of the modes are unique, and that they are similar in their local covariance structure. Furthermore, calculation of the negative log–likelihood of the sample mean of each mode indicates that, given the covariance structure, the MAP mode is the dominate mode of the posterior distribution; the second most probable mode is a large distance from the most probable mode in log–likelihood space, the third and forth modes being even further. A pilot run of the DC algorithm, with the four most probable modes, suggests that all of the mass of the posterior distribution is associated with the most probable mode.



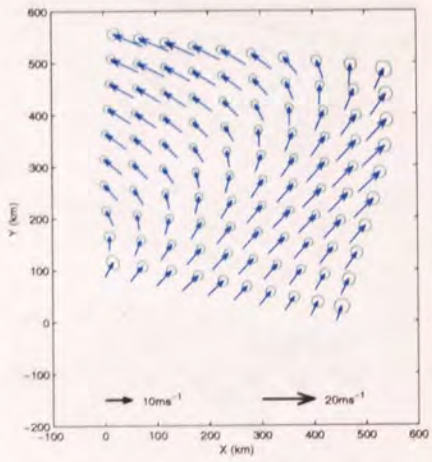
(a) Mode 1, mean sample energy: −206.9883



(b) Mode 2, mean sample energy: −165.1694



(c) Mode 3, mean sample energy: −149.9259

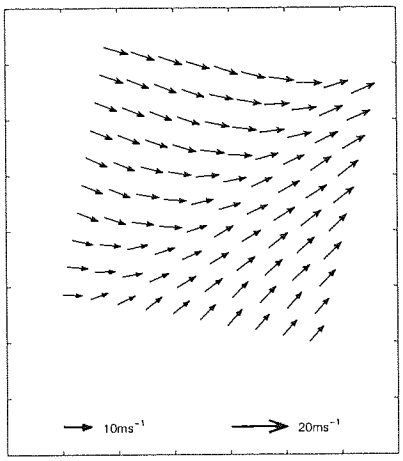


(d) Mode 4, mean sample energy: −148.0230

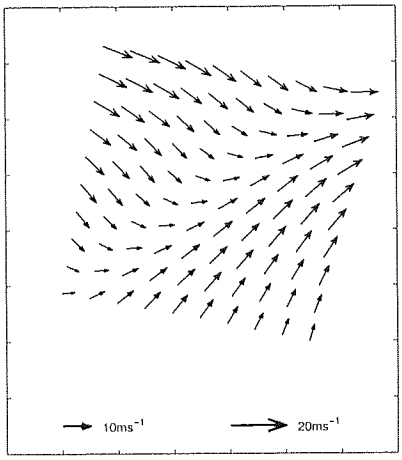
Figure 6.18: Case study 4: visualisation of four most probable modes after sampling. The blue wind vectors represent the modes identified in the mode finding exercise, the red wind vectors represent the sample mean. The cyan ellipses represent an estimate of the local covariance structure, plotted at three standard deviations from the sample mean. The four possible wind fields are represented in these four modes. The most probable mode, however, appears to dominate when considering the point estimate of the mode which is a large distance away form the second most probable mode.

The full experiment was run (see Figure J.4 (b) for convergence statistics) and the results estimate that 100% of the mass of the posterior distribution is associated with the most probable mode.

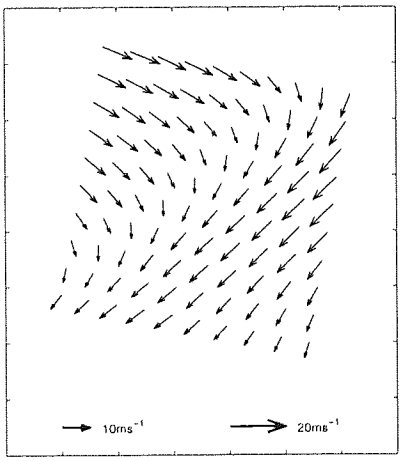
The results of the three wind field retrieval techniques and the NWP target wind field are shown in Figure 6.19. For this case study both autonomous retrieval methods predict the wind field in the opposite direction to the target, whilst the non-autonomous method retrieves the correct solution. We note that the second most probable wind field is the correct solution.



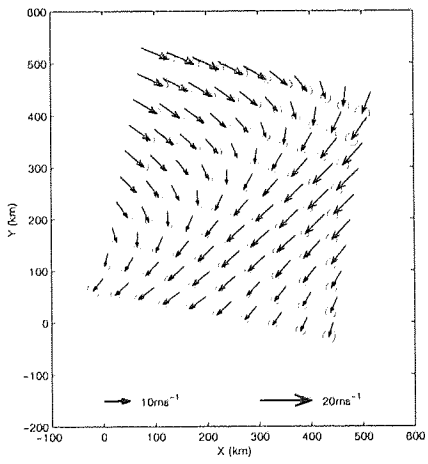
(a) NWP wind



(b) Non-autonomous derived wind field



(c) Autonomous MAP derived wind field

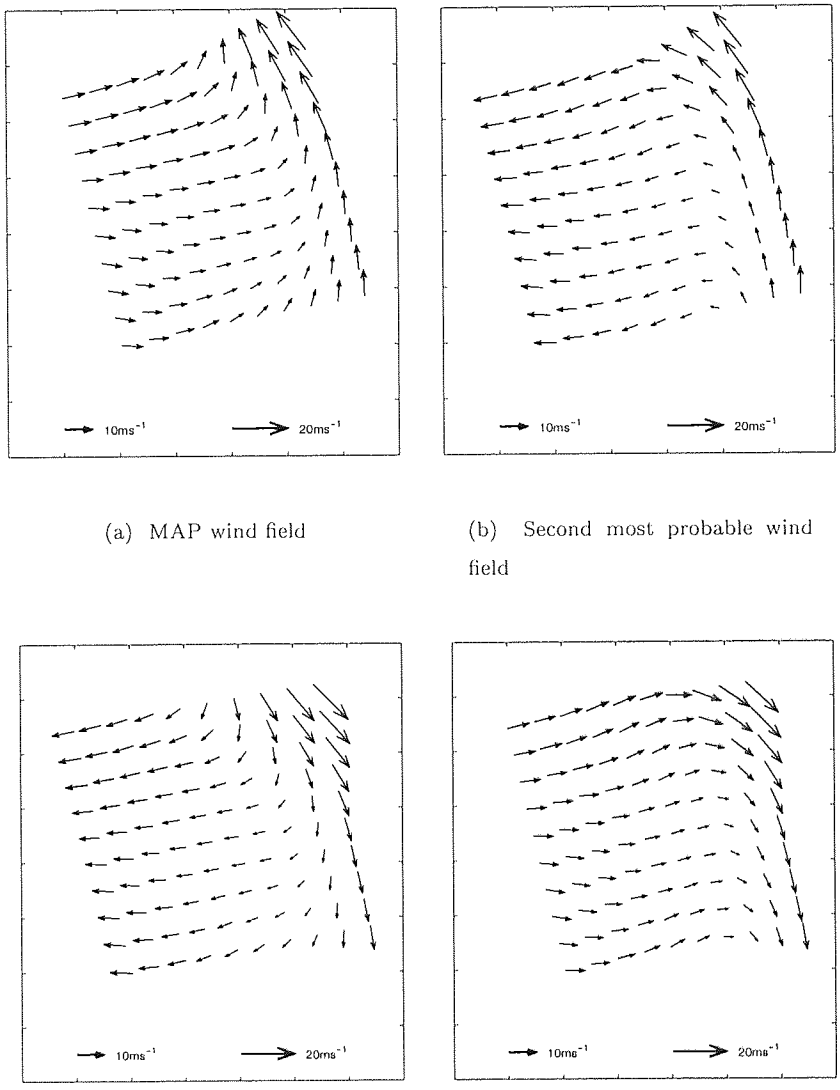


(d) Autonomous MCMC derived wind field

Figure 6.19: Case study 4: the final result.

6.8.5 Case study 5: Frontal wind field II

This is the final case study and is the second example of a frontal wind field. This example was chosen because it is a less extreme frontal system than case study 4; the change in wind direction is somewhat smoother. As in the previous case study the four most probable modes represent the four possible permutations of frontal wind fields: these are shown in Figure 6.20.



(a) MAP wind field

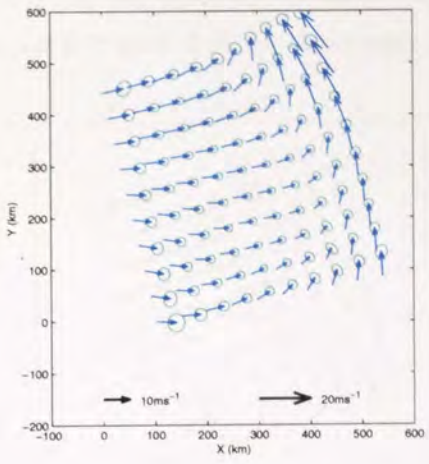
(b) Second most probable wind field

(c) Third most probable wind field

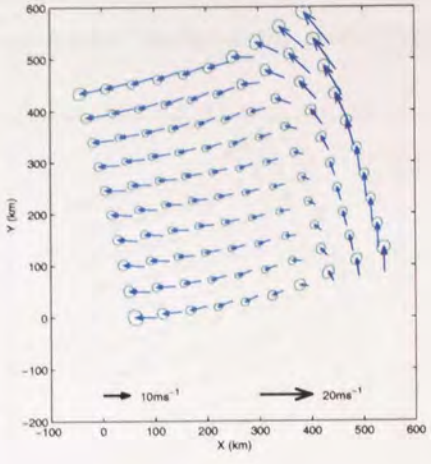
(d) Fourth the most probable wind field

Figure 6.20: Case study 5: the four most probable modes.

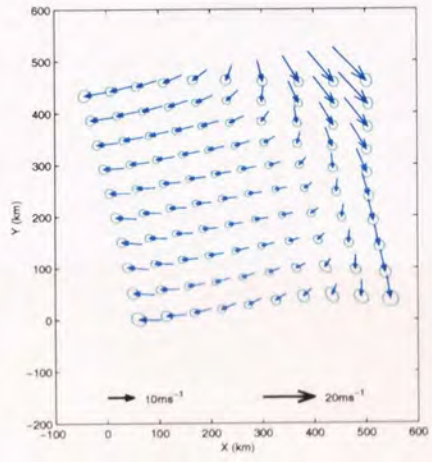
Sampling from the individual modes showed each of these modes to be distinct and unique (see Figure J.5 (a) for convergence statistics) and the results are visualised in Figure 6.21.



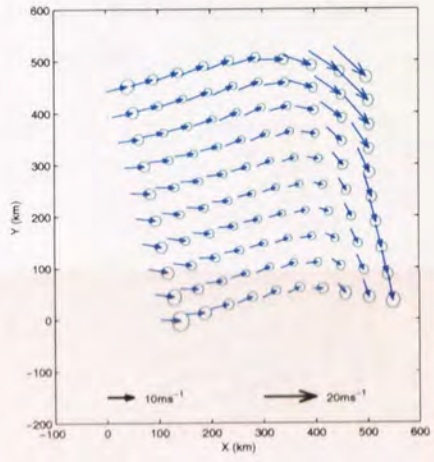
(a) Mode 1, mean sample energy: −171.3904



(b) mode 2, mean sample energy: −167.1620



(c) Mode 3, mean sample energy: −161.9980

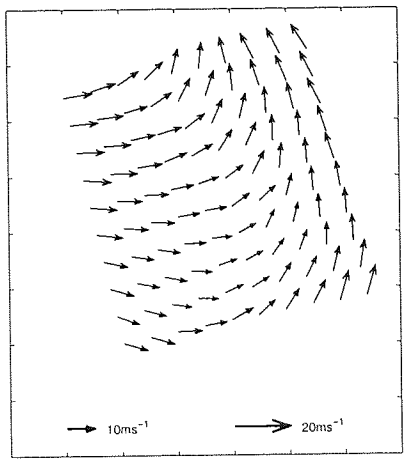


(d) Mode 4, mean sample energy: −152.1320

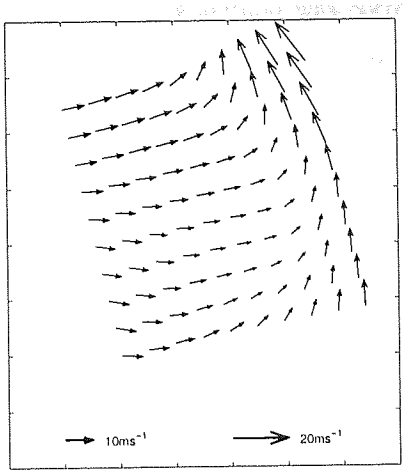
Figure 6.21: Case study 5: visualisation of the four most probable modes after sampling. The blue wind vectors represent the modes identified in the mode finding exercise, the red wind vectors represent the sample mean. The cyan ellipses represent an estimate of the local covariance structure, plotted at three standard deviations from the sample mean. Each mode is unique and distinct and the mean estimates from the samples are close to those found by mode finding. The first and second most probable modes (a) and (b) appear to dominate the posterior distribution.

A pilot DC sample run was executed incorporating the four modes and showed that the estimated mass associated with four modes to be 99.94%, 0.05%, 0.01%, 0.00% respectively. From these results the first and second most probable modes were applied in the full experiment (see Figure J.5 (b)). The results estimated 99.95% of the mass of the distribution associated with the most probable mode and, therefore, 0.05% associated with the second most probable mode, hence predicting the most probable mode for wind field retrieval.

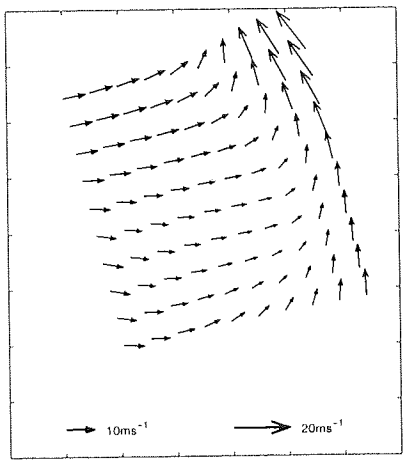
The results of the non-autonomous and two autonomous methods are shown in Figure 6.22 along with the target NWP wind field; for this example each method retrieved the correct wind field.



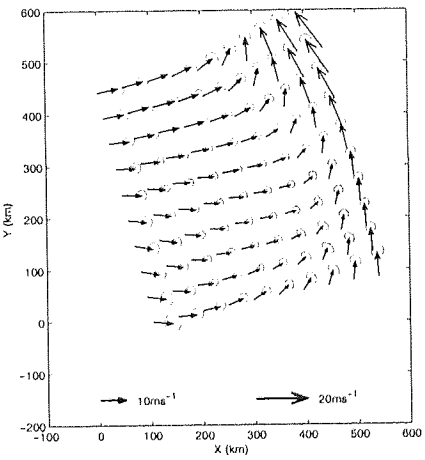
(a) NWP wind



(b) Non-autonomous retrieved wind field



(c) Autonomous MAP retrieved wind field



(d) Autonomous MCMC retrieved wind field

Figure 6.22: Case study 5: the final result.

6.8.6 Case study conclusions

In these five case studies the DC autonomous ambiguity removal method was correct three times whilst the MAP method was correct once and the non-autonomous method was correct on all five. Hence, for these examples, the DC method has improved our ability to perform autonomous wind field retrieval.

Of the three correctly selected wind fields, two were the second most probable wind field and one was the MAP. This is an encouraging result. The correct selection of the second most probable wind field over the most probable wind field suggests that the MAP solution is not always correct and gives a principled approach to making that decision.

The two incorrect examples showed how the model can fail. In case study 3, the low pressure system, the model failed by incorrectly predicting homogeneous wind fields in the first and second most probable modes – this was attributed to over smoothing by the prior. The second incorrect estimate was case study 4. In this example the point estimate of the most probable mode is over 30 units greater than the second most probable mode; for the second most probable mode to dominate we would expect to see a significantly larger covariance structure, but the visualisation shows this not to be so – the model is incorrect and very certain of it.

In each of the case studies the statistics returned from the pilot studies predicted the same wind field as those returned from the full experiments incorporating convergence statistics. It is therefore a reasonable assumption that these samples have converged after the $80,000$ sub–samples are collected. Also, these experiments have shown that the mass is associated with the first and second most probable modes. Therefore, a pragmatic approach to this problem is suggested: for a wind field of interest, run the DC algorithm once as described in the case studies sampling from the first and second most probable modes, estimating the mass associated with each mode. The predicted wind field will be the mode with the most mass associated with it. This approach will also reduce the running time and storage overheads. In the next section this approach is run on 100 test wind fields.

6.9 One hundred test cases

The DC algorithm was applied to 100 wind fields chosen randomly from 300 possible targets (so as not to include any of the case studies); each sample run took approximately 6 hours to run on a 333MHz Sun Sparc work station.

6.9.1 Results

Of the 100 wind fields 91 were predicted to be the most probable mode and 9 were predicted to be the second most probable mode. The predicted mode was then compared with NWP target wind field using the FOM measure and visual inspection

Of the 91 wind fields predicted to be the first most probable mode, 66 were correct, whilst 24 where incorrect. Four examples of correctly identified solutions, showing the target solution and the first and second most probable solutions are given in Figure 6.23. For the first two examples ((a)-(c) and (d)-(f)) the sampler estimates all of the mass to be in the most probable mode. For the second two examples ((g)-(i) and (k)-(l)) the sampler estimates that some of the mass is associated with the second most probable mode, although it is less than that associated with the first most probable mode.

Of the 24 incorrect predictions, 17 solutions had the correct solution as the second most probable mode; 1 was due to low wind speed where wind direction is difficult to predict; and 6 were due to the model being incorrect for the first and second modes, that is, a solution agreeing with the target wind field did not appear in the first or second modes. Examples of the first mode being incorrectly chosen are shown in Figure 6.24, in all cases the estimate of the mixing coefficients suggests that the model is certain that the incorrect mode is 'correct': hence, we are wrong about our prediction. Examples of the model not presenting correct solutions in the first and second modes are given in Figure 6.25. The first three examples show that there are two wind fields that are meteorologically distinct, but opposite in direction to the target, whilst the fourth shows a low pressure system being over smoothed by the prior.

Nine of the predictions were for the second most probable mode. Of those modes, 6 were correct (see Figure 6.26). Five showed subtle changes between the first and second most probable modes and hence the MAP mode solution would also be considered correct. Three were incorrect of which 1 was due to low wind speed prediction where there is more uncertainty in wind direction and the remaining 2 were due to the model being incorrect for the first two modes.

Overall the sampling method achieved a correct prediction rate of 73%. We can compare this result with that of the MAP method by comparing how the results would change for second most probable mode predictions. Of the 9 predictions, the MAP solution was correct for 5 predictions. Hence the overall MAP solution for this experiment would be 72%.

One final remark is that given the first and second most probable modes, this experiment was correct 95% of the time, the 5% incorrect solutions being due to the fact that the correct wind field

was not in the first or second most probable modes.

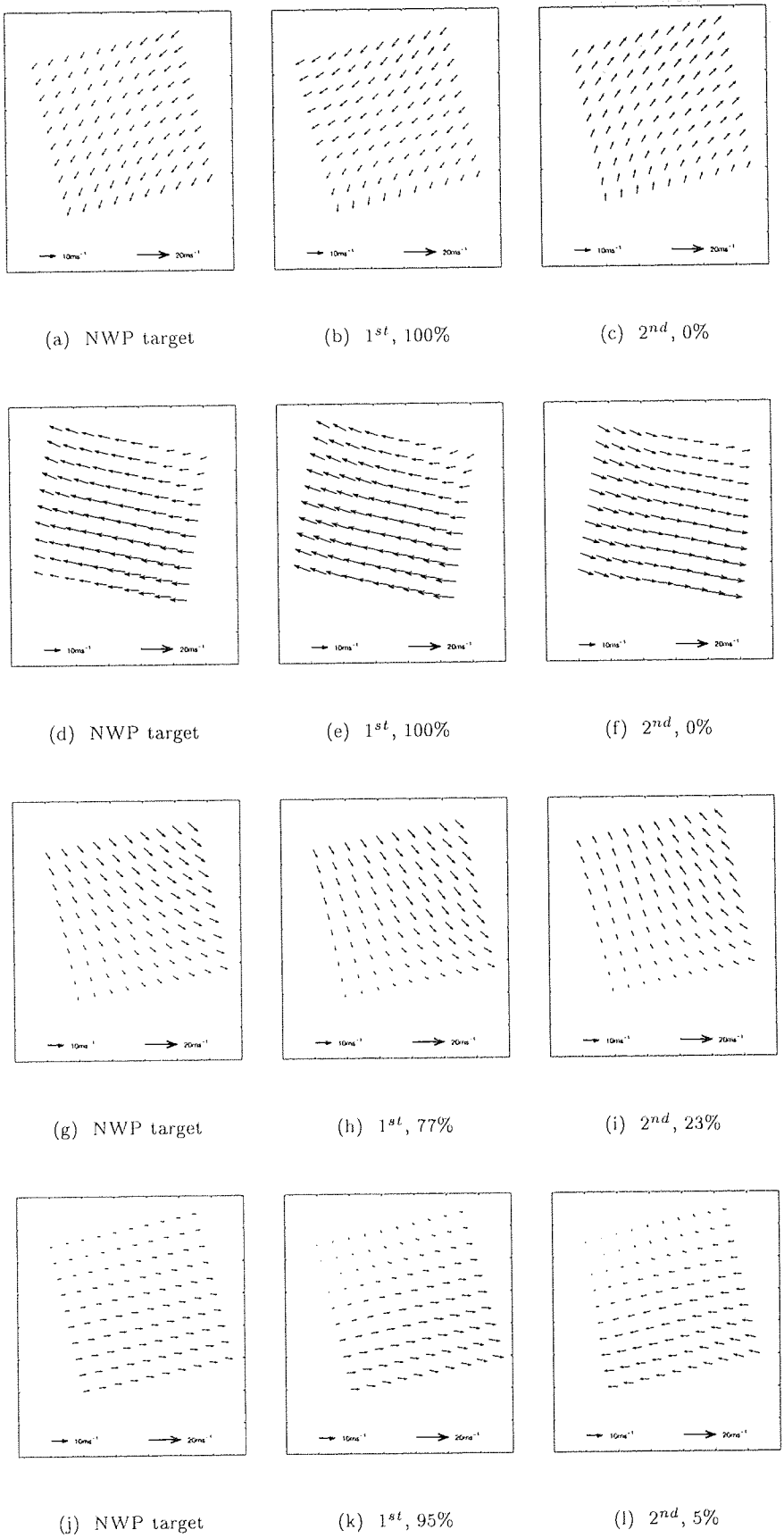
6.9.2 Conclusions

The results of this experiment are disappointing as we expected the DC algorithm to significantly improve on results of the MAP method, especially in cases where the correct solution is the second most probable mode. However, the model is capable of retrieving the correct wind field in either the first or second most probable mode 95% of the time. The problem of accurately and confidently autonomously identifying the correct solution is not solved.

6.10 Chapter review

In this chapter we set out to establish if sampling would improve autonomous ambiguity removal over the MAP estimate method. The problem was approached by benchmarking three MCMC algorithms that are designed to sample from multi-modal distributions, ensuring that the samples drawn from each mode reflect the relative mass associated with each mode. Of the three methods one was shown to be suitable for this application.

The initial results looked promising on the case studies, but when the algorithm was applied to a larger test set the sampling method only improved results by 1% over those of the MAP method, which is not significant.



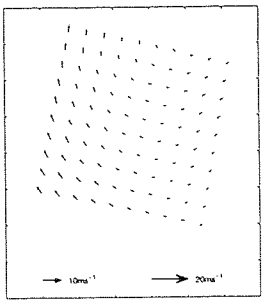
(a) NWP target (b) 1^{st}, 100% (c) 2^{nd}, 0%

(d) NWP target (e) 1^{st}, 100% (f) 2^{nd}, 0%

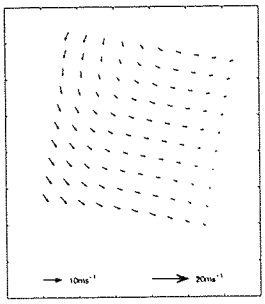
(g) NWP target (h) 1^{st}, 77% (i) 2^{nd}, 23%

(j) NWP target (k) 1^{st}, 95% (l) 2^{nd}, 5%

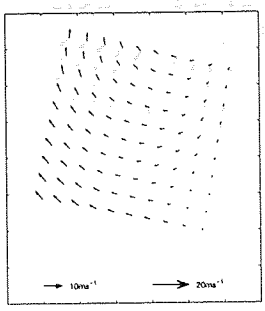
Figure 6.23: Examples of correctly chosen first most probable mode.



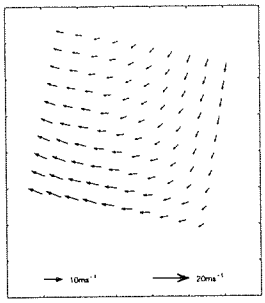
(a) NWP target



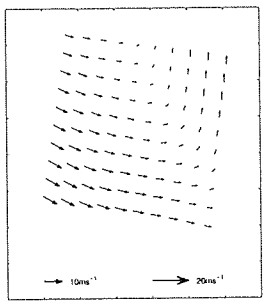
(b) 1^{st}, 100%



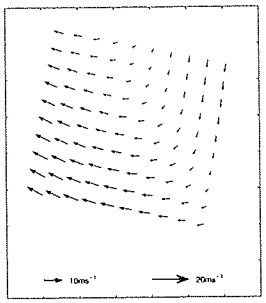
(c) 2^{nd}, 0%



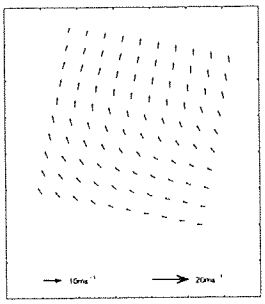
(d) NWP target



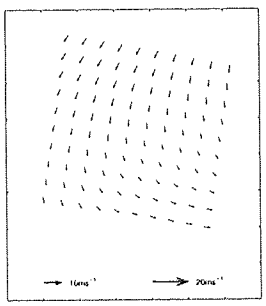
(e) 1^{st}, 89%



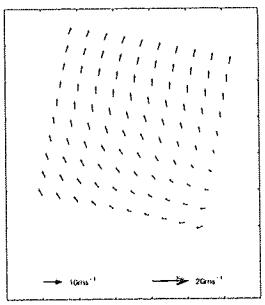
(f) 2^{nd}, 11%



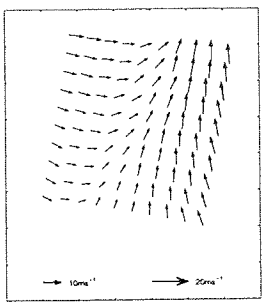
(g) NWP target



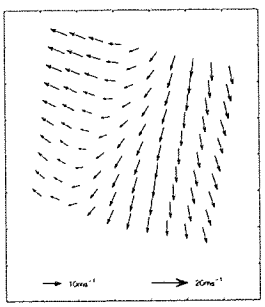
(h) 1^{st}, 88%



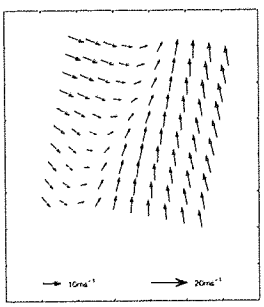
(i) 2^{nd}, 12%



(j) NWP target

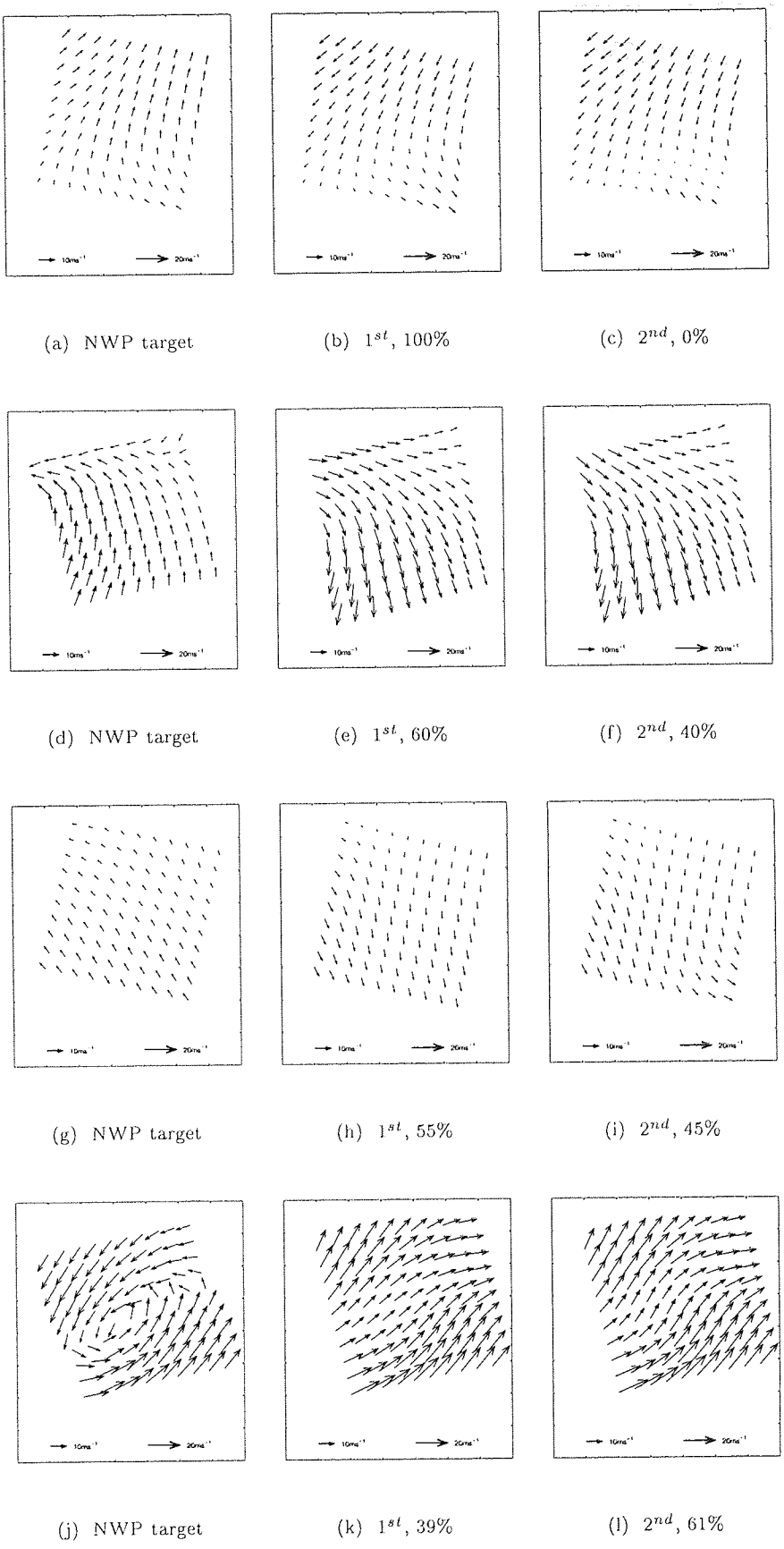


(k) 1^{st}, 100%



(l) 2^{nd}, 0%

Figure 6.24: Examples of incorrectly chosen first most probable mode.



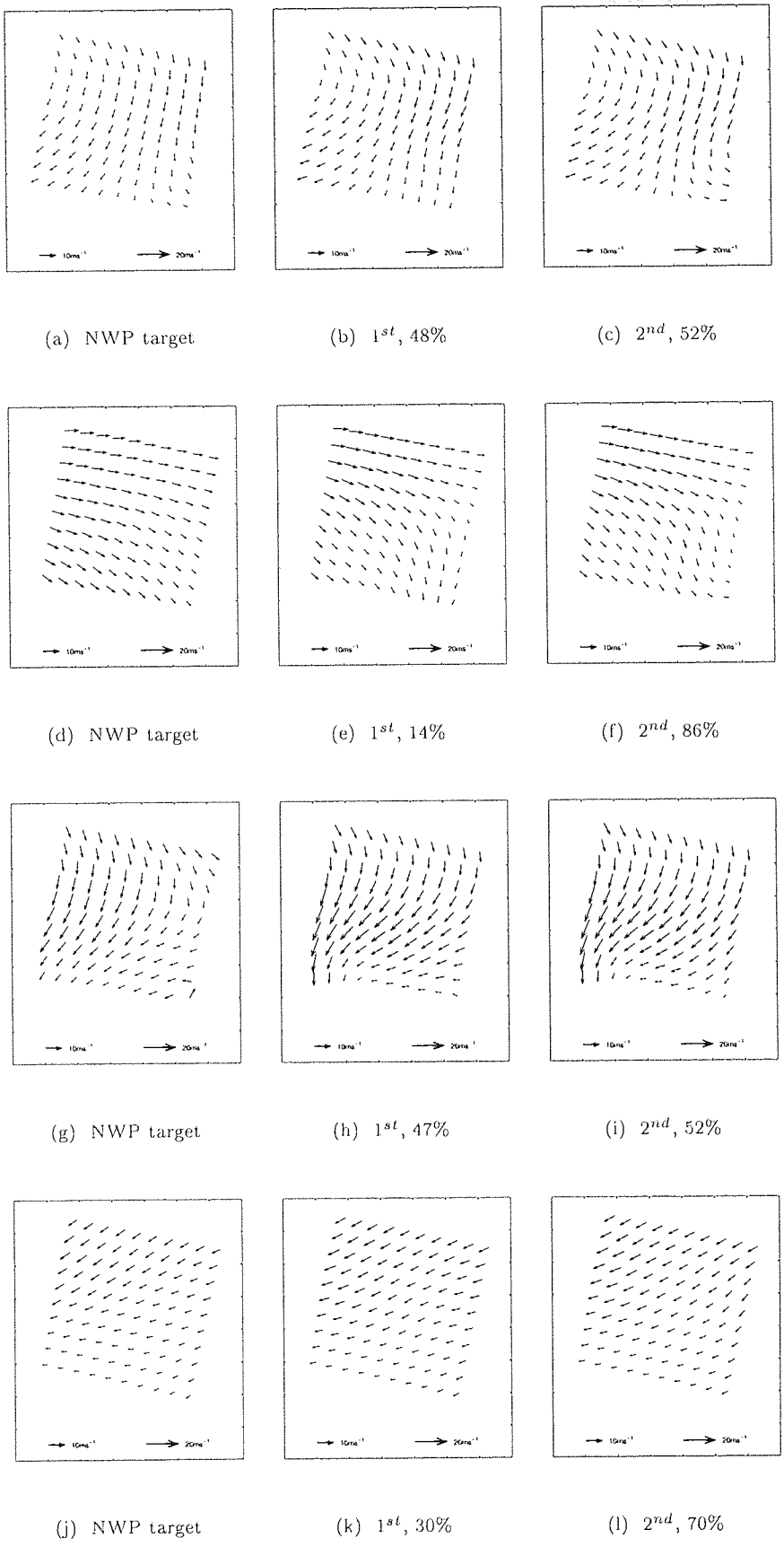
(a) NWP target (b) 1^{st}, 100% (c) 2^{nd}, 0%

(d) NWP target (e) 1^{st}, 60% (f) 2^{nd}, 40%

(g) NWP target (h) 1^{st}, 55% (i) 2^{nd}, 45%

(j) NWP target (k) 1^{st}, 39% (l) 2^{nd}, 61%

Figure 6.25: Examples of model being wrong in first and second most probable modes.



(a) NWP target (b) 1^{st}, 48% (c) 2^{nd}, 52%

(d) NWP target (e) 1^{st}, 14% (f) 2^{nd}, 86%

(g) NWP target (h) 1^{st}, 47% (i) 2^{nd}, 52%

(j) NWP target (k) 1^{st}, 30% (l) 2^{nd}, 70%

Figure 6.26: Examples of correctly chosen second most probable mode.

Chapter 7

Summary and Conclusions

Key word summary

Local model	**Global model**
MDN	**pragmatic Bayesian**
Non-autonomous	**Autonomous**

7.1 Introduction

In the first chapter a pragmatic Bayesian approach to retrieving wind fields directly from scatterometer data was introduced. The model was decomposed into a combination of a prior wind field distribution and a likelihood which was the product of local direct inversions of the scatterometer data:

$$p(U,V|\Sigma^o) \propto \left(\prod_i \frac{p(u_i,v_i \mid \sigma_i^o)}{p(u_i,v_i)} \right) p(U,V). \tag{7.1}$$

In the first half of this thesis the focus was on developing a robust local model, which directly inverted the scatterometer data into the wind vector components, (u,v), using a mixture density network. In the second half, the focus was on investigating methods for retrieving wind fields from (7.1). The model was implemented with the aid of the prior model developed by Dan Cornford. Three methods of wind field retrieval were applied: non-autonomous, autonomous by MAP estimate and autonomous by MCMC sampling estimate.

7.2 Summary

The results of the local modelling were very successful, whilst the results of the wind field retrieval from the global model showed that autonomous wind field retrieval still represents a challenge. However, the results of the non-autonomous ambiguity removal show that a probabilistic approach to wind

field retrieval was suitable for this application. The key results of this thesis, for the local and global models, are summarised in the next two sections.

7.2.1 Local Model

The overall aim was to assess the practicality of directly modelling the wind vectors (u, v) from scatterometer data σ^o. There are four key results from Chapter 4.

- We have shown that an MDN that has an MLP with twenty units in the hidden layer, and a GMM with four kernels successfully models the relation $\sigma^o \rightarrow (u, v)$ on a per trace basis. A more complicated model for all traces, which included the incidence angle θ in the inputs, describing the mapping $(\sigma^o, \theta) \rightarrow (u, v)$, was considered and was shown to be of comparable accuracy to the models which did not include incidence angle in the inputs. This mapping is modelled using an MDN with an MLP having twenty five units in the hidden layer, and four kernels in the GMM. Ultimately it was this model that was used in the wind field disambiguation work.

- The hybrid MDN configuration yields similar results to an MDN with two free centres, and shows that the conditional probability distribution, $p(u, v \mid \sigma^o)$, is generally bi–modal with the two modes positioned diametrically opposite one another. The experiments with hybrid models that had four kernels (two free centres and two mirrored centres) compared well with those of the conventional MDN with four free kernels. This confirms the belief that the solution is predominantly bi–modal, but further suggests that there are sometimes three modes or a second pair of diametrically opposed modes.

- Other work in the field solves the inverse problem by directly modelling wind speed and wind direction with two separate models. The models of this project are similar in performance to other methods that model each cell individually (Cornford *et al.*, 1999). The methods in this thesis, however, do not compare as favourably with those methods which take spatial information surrounding the cell of interest as part of their inputs (Richaume *et al.*, 2000).

- On the UKMO test set the MDN models perform better on key performance measures than the currently operational CMOD4 model.

7.2.2 Global Model

The main objectives of this work were to investigate the suitability of a Bayesian approach to the wind field retrieval problem.

- Non-autonomous ambiguity removal represents a realistic alternative to current methods(Stoffelen and Anderson, 1997a). The results presented in Chapter 5 show that for non–autonomous wind field retrieval over 99% of wind fields from a test set of 575 wind fields were correctly retrieved. With a probabilistic model it is also possible to add information to the point estimate of the wind field by estimating the variance of the local wind vectors, (u, v).

- Reliable autonomous ambiguity removal is still not completely solved, either by the MAP estimate or the sampling methods. The results of Chapters 5 and 6 show that autonomous ambiguity removal schemes correctly retrieved no more than 73 wind fields out of 100 test wind fields. Also, one further observation is that the correct wind field was either the first or second most probable mode for 95 of the 100 test cases.

- General MCMC sampling algorithms designed to cope with multi–modality were not suitable for this application. A specific algorithm was developed for the wind field application which greatly reduced computation cost by taking into account the pre–computed positions of the significant modes of the posterior distribution.

7.3 Further model development

For autonomous wind field retrieval there are two areas that require development. The first is to improve the disambiguation skill, as there is insufficient information in the scatterometer data alone for autonomous ambiguity removal. The second is to improve the skill in retrieving wind fields that contain features such as lows and highs. A suggested approach for each area is given.

Improving disambiguation skill. One possible method is to use information from observations of satellite cloud drift winds. Cloud drift winds are obtained from sequences of cloud images collected by satellite and give automatic, reasonably accurate, estimations of wind direction (wind speed is also estimated but it is not our concern here) (Daley, 1993). The estimates represent areas rather than points and are therefore not synoptic in the same way as a wind field. As the observations are autonomously calculated, without the need for an NWP model, they could be used to replace the directional information provided by NWP winds. Hence, in the non-autonomous ambiguity removal method described in this thesis, the optimisation step could be initialised by the local retrieved wind vectors that are closest to the direction estimated from cloud drift winds.

Improving retrieval of wind fields with features. The reason that wind fields with features such as highs and lows are poorly retrieved is due to the fact that the prior model has been trained to generalise all features which may exist in a wind field (Cornford, 1998a). It would be feasible to train prior models to cope with specific features, such as highs and lows, as suggested by Cornford (1997), but this would be time consuming and computationally expensive.

7.4 Conclusions

The pragmatic Bayesian approach applied to this problem has successfully modelled wind fields. There are difficulties in retrieving wind fields autonomously, but in a non-autonomous context this method retrieves the correct wind field nearly every time.

For the local models, mixture density networks provide a probabilistic framework within which to model wind vectors directly from satellite scatterometer data, where the results are comparable with the current operational model CMOD4.

Behind the work in this thesis are two fundamental ideas which we address here:

Autonomous ambiguity removal

Can autonomous ambiguity removal be achieved? Not yet – one of the underlying physical constraints of this problem is the inherent ambiguity in the observation process. It was hoped that within a wind field there would be enough information from wind vectors in the correct direction to assist the model in predicting the correct wind field. As we have seen this is not the case. Maybe around 70% is the best we can hope for autonomously.

The results suggest that there is sufficient information in the scatterometer readings to generate consistent wind fields, but there is not enough information to accurately predict which is the 'true' physical wind field. It would appear that further information is required to make accurate predictions such as buoy observations, ship observations or satellite cloud drift winds. The current disambiguation schemes rely on wind vector information from a previous estimate of the wind field and multiple observation sources (Stoffelen and Anderson, 1997a).

The merits of a probabilistic framework

The approaches used in this thesis have been termed as 'principled', in the probabilistic sense, when compared to other methods applied to this application. The methods are motivated by the Bayesian probabilistic framework in which the models have been constructed. But do these methods add any benefits?

One benefit that can be seen is the ease with which the posterior distribution is decomposed into the likelihood and prior probability distributions. It has been possible to build each part of the solution independently using specific tools to solve each part, and then fuse the models together for the global solution. This advantage is particularly visible in the local model, where an MDN is employed to cope with the multi-valued nature of the inversion problem.

On the other hand it could be argued that the Bayesian framework complicates the problem because of the difficulty experienced in autonomously retrieving wind fields. Other work (Thiria *et al.*, 1993; Richaume *et al.*, 2000) relies on a combination of engineering and scientific approaches to solving the disambiguation problem and report results which appear to be comparable (given that the test sets are different) with those reported in this thesis.

Hence, acceptance of the probabilistic methods within other communities is sometimes a problem, especially where other solutions already exist and are the result of investment and research over several years.

References

Abrahamsen, P. 1997. A Review of Gaussian Random Fields and Correlation Functions. Technical Report 917, Second Edition, Norwegian Computing Center, Box 114 Blindern, N-0314 Oslo, NORWAY.

Bishop, C. M. 1994. Mixture Density Networks. Technical Report NCRG/94/004, Neural Computing Research Group, Aston University, Birmingham. UK. Available from `http://www.ncrg.aston.ac.uk/Papers/index.html`.

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition.* Oxford University Press.

Bishop, C. M. and I. T. Nabney 1996. Modelling Conditional Probability Distributions for Periodic Variables. *Neural Computation* **8**, 1123–1133.

Brooks, S. P. 1999. Markov Chain Monte Carlo and its Application. *The Statistician* **47**, 69 – 100.

Brooks, S. P. and P. Giudici 1998. Convergence Assessment for Reversible Jump MCMC Simulations. *Bayesian Statistics* **6**, 733–742.

Brooks, S. P. and G. O. Roberts 1997. Assessing Convergence of Markov Chain Monte Carlo Algorithms. Technical report, School of Mathematics, University of Bristol and Statistical Laboratory, University of Cambridge. Avaliable from `http://www.statslab.cam.ac.uk/`.

Bullen, R. 2001, October. Noise Modelling and Outlier Detection in Satellite Scatterometer Data. Master's thesis, Aston University, Birmingham, UK.

Cornford, D. 1997. Random Field Models and Priors on Wind. Technical Report NCRG/97/023, Neural Computing Research Group, Aston University, Birmingham, UK. Available from `http://www.ncrg.aston.ac.uk/Papers/index.html`.

Cornford, D. 1998a. Flexible Gaussian Process Wind Field Models. Technical Report NCRG/98/017, Neural Computing Research Group, Aston University, Birmingham, UK. Available from `http://www.ncrg.aston.ac.uk/Papers/index.html`.

Cornford, D. 1998b. Non-Zero Mean Gaussian Process Prior Wind Field Models. Technical Report NCRG/98/020, Neural Computing Research Group, Aston University, Birmingham, UK. Available from `http://www.ncrg.aston.ac.uk/Papers/index.html`.

Cornford, D., D. J. Evans, and I. T. Nabney 12–16 March 2001. Wind Field Retrieval From Scatterometer Data. In 8^{th} *International Meeting on Statistical Climatology.* University of Luneburg.

Cornford, D. and I. T. Nabney 1998. NEUROSAT: An Overview. Technical Report NCRG/98/011, Neural Computing Research Group, Aston University, Birmingham, UK. Available from `http://www.ncrg.aston.ac.uk/Papers/index.html`.

Cornford, D., I. T. Nabney, and C. M. Bishop 1999. Neural Network–Based Wind Vector Retrieval from Satellite Scatterometer Data. *Neural Computing and Applications* **8**, 206–217.

Cornford, D., I. T. Nabney, and D. J. Evans 1999. Bayesian Retrieval of Scatterometer Wind Fields. Technical Report NCRG/99/015, Neural Computing Research Group, Aston University, Birmingham, UK. Available from `http://www.ncrg.aston.ac.uk/Papers/index.html`.

Cornford, D., I. T. Nabney, and G. Ramage 2001. Improved Neural Network Scatterometer Forward Models. *Journal of Geophysical Research - Oceans* **106**, 22331–22338.

Cressie, N. A. C. 1993. *Statistics for Spatial Data.* John Wiley and Sons, New York.

Daley, R. 1993. *Atmospheric Data Analysis.* Cambridge University Press.

Evans, D. J. 1998. Mixture Density Network Training by Computation in Parameter Space. Technical Report NCRG/98/016, Neural Computing Research Group, Aston University, Birmingham, UK. Available from `http://www.ncrg.aston.ac.uk/Papers/index.html`.

Evans, D. J., D. Cornford, and I. T. Nabney 2000. Structured Neural Network Modelling of Multi-Valued Functions for Wind Vector Retrieval from Satellite Scatterometer Measurements. *Neurocomputing Letters* **30**, 23 – 30.

Evans, D. J., I. T. Nabney, and D. Cornford 17–21 September 2000. Efficient Sampling from High Dimension, Multimodal Probability Distributions: A Spatial Application. In *First European Conference on Spatial and Computational Statistics.*

Gelman, A. 1996. *Markov Chain Monte Carlo in Practice*, Chapter Inference and Monitoring Convergence, pp. 131–140. Chapman & Hall.

Gelman, A. and D. B. Rubin 1992. Inference from Itterative Simulation using Multiple Sequences. *Statistical Science* **7** (4), 457–511.

Gilks, W. R., S. Richardson, and D. J. Spiegelhalter 1996. *Markov Chain Monte Carlo in Practice.* Chapman & Hall.

Hastings, W. K. 1970. Monte Carlo Sampling Methods using Markov Chains and Their Applications. *Biometrika* **57** (1), 97 — 109.

Homleid, M. and L.-A. Breivik 1995. Preparations for the Assimilation of ERS-1 Surface Wind Observations into Numerical Weather Prediction Models. *Tellus* **47A**, 62–79.

IFREMER 1996. *Off-line Wind Scatterometer ERS Products* (C2-MUT-01-IF, Version 2, 01/03/1996 ed.). IFREMER. Institut Français de la Recherche pour l'Exploitation de la Mer.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi 1983. Optimisation by Simulated Annealing. *Science* **220**, 671 — 680.

Long, D. and J. M. Mendel 1991. Identifiability in Wind Estimation From Scatterometer Measurements. *IEEE Transactions on Geoscience and Remote Sensing* **29**, 268–276.

Long, D. G. 1993. Wind Field Model-Based Estimation of SEASAT Scatterometer Winds. *Journal of Geophysical Research* **98**, 14651–14668.

MacKay, D. J. C. 1994, January. Bayesian Neural Networks and Density Networks. *Nuclear Instruments and Methods in Physics Research Section A-Accelerators Spectrometers Detectors and Associated Equipment* **354** (1), 73–80.

Marinari, E. and G. Parisi 1992. Simulated Tempering: a New Monte Carlo Scheme. *European Physical Letters.* **19**, 451 – 458.

McLachlan, G. J. and K. E. Bashford 1988. *Mixture Models: Inference and Applications to Clustering.* New York: Marcel Dekker.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, and A. H. Teller 1953. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics* **21** (6), 1087 — 1092.

Nabney, I. T., D. Cornford, and C. K. I. Williams 2000. Bayesian Inference for Wind Field Retrieval. *Neurocomputing Letters* **30**, 3–11.

Neal, R. M. 1993. Probabilistic Inference Using Markov Chain Monte Carlo Methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada.

Neal, R. M. 1995a. *Bayesian Learning for Neural Networks.* Ph.D. thesis, University of Toronto.

Neal, R. M. 1995b. Sampling from Multimodal Distributions using Tempered Transitions. *Statistics and Computing* **6**, 353 – 366.

Offiler, D. 1994. The Calibration of ERS-1 Satellite Scatterometer Winds. *Journal of Atmospheric and Oceanic Technology* **11**, 1002–1017.

Ramage, G. 1998. Neural Networks for Modelling Wind Vectors. Master's thesis, Aston University, Birmingham, UK.

Richaume, P., F. Badran, M. Crepon, C. Mejia, H. Roquet, and S. Thiria 2000. Neural Network Wind Retrieval from ERS-1 Scatterometer Data. *Journal of Geophysical Research, C4 Oceans* **105**, 8737–8751.

Robinson, I. S. 1985. *Satellite Oceanography, An Introduction for Oceanographers and Remote-sensing Scientists.* Ellis Horwood Limited.

Schyberg, H. and L.-A. Breivik 1998. Ambiguity Removal Algorithm Evaluation. Technical Report 64, Norwegian Meteorological Institute, Oslo, Norway.

Stoffelen, A. and D. Anderson 1997a. Ambiguity Removal and Assimilation of Scatterometer Data. *Quarterly Journal of the Royal Meteorological Society* **123**, 491–518.

Stoffelen, A. and D. Anderson 1997b. Scatterometer Data Interpretation: Estimation and Validation. *Journal of Geophysical Research* **102**, 5767–5780.

Stoffelen, A. and D. Anderson 1997c. Scatterometer Data Interpretation: Measurement Space and Inversion. *Journal of Atmospheric and Oceanic Technology* **14**, 1298–1313.

Thiria, S., C. Mejia, and F. Badran 1993. A Neural Network Approach for Modelling Nonlinear Transfer Functions: Application for Wind Retrieval From Spaceborne Scatterometer Data. *Journal of Geophysical Research* **98**, 22827–22841.

Tierney, L. 1994. Markov Chains for Exploring Posterior Distributions. *The Annals of Statistics* **22**, 1701 – 1762.

Tjelmeland, H. and B. K. Hegstad 1999. Mode Jumping Proposals in MCMC. Technical report, Norwegian University of Science and Technology., Department of Mathematical Sciences, N-7034 Trondheim, Norway. `http://www.stat.ntnu.no/preprint/statistics/1999/S1-1999.ps`.

Tjelmeland, H. and B. K. Hegstad 2001. Mode Jumping Proposals in MCMC. *Scandinavian Journal of Statistics* **28**, 205–223.

Williams, C. K. I. 1997. Combining Spatially Distributed Predictions From Neural Networks. Technical Report NCRG/97/026, Neural Computing Research Group, Aston University, Birmingham, UK. Available from `http://www.ncrg.aston.ac.uk/Papers/index.html`.

Williams, C. K. I. 1998. Prediction with Gaussian Processes: From Linear Regression to Linear Prediction and Beyond. In *Learning and Inference in Graphical Models*. Kluwer, London.

Appendix A

The prior model

The prior model, $p(U, V)$, constrains the local solutions to produce consistent and meteorologically plausible winds. It is a vector Gaussian process (GP) based wind field model that is tuned using NWP data. This ensures that the wind field model is representative of NWP wind fields, which are the best *a priori* estimates of the true wind fields (Nabney *et al.*, 2000).

GPs provide a flexible class of models (Williams, 1998) and are particularly appropriate when the variables are distributed in space (Cressie, 1993). The wind field data is assumed to come from a zero mean multi-variate Gaussian distribution, whose variables are the Cartesian wind vector components at each cell location and whose covariance matrix is a function of the spatial separation of the observations and some physically interpretable parameters (Daley, 1993). A particular form of GP was chosen to incorporate geophysical information in the prior model (Cornford, 1998b).

In common with (Schyberg and Breivik, 1998; Long, 1993) a decomposition of the near surface horizontal wind field, (U, V), in terms of a stream function, Ψ, and velocity potential, Φ (Daley, 1993) is assumed. The wind component covariance model, K_{uv}, is specified in terms of covariance functions applied independently to the scalars, Ψ and Φ. Initially a very general modified Bessel function based covariance function was applied, given by:

$$C(r; E^2, L, \nu, \eta^2) = E^2 \left(\frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\frac{r}{L}\right)^{\nu} K_{\nu}\left(\frac{r}{L}\right) \right) + \eta^2 , \tag{A.1}$$

where E^2 controls the variance (energy) in the scalar field, L controls the typical length scale and η^2 controls the noise (and sub-sampling scale) variance. ν controls the order of the modified Bessel function and thus the differentiability of the field (Abrahamsen, 1997). The Maximum *A posteriori* Probability (MAP) values for the parameters E^2, L and ν were determined for each month using three years of European Centre for Medium range Weather Forecasting (ECMWF) assimilated wind fields for the North Atlantic. Proper, independent, uninformative priors over the scalars E^2, L, ν and η^2

were assumed. It was found that $\nu \sim 2.5$ for all months so assuming $\nu \equiv 2.5$, Equation (A.1) becomes (see Cornford (1998a) for detail):

$$C(r;E^2, L, \eta^2) = \\ E^2 \left(1 + \frac{r}{L} + \frac{r^2}{3L^2}\right) \exp\left(-\frac{r}{L}\right) + \eta^2 \,. \tag{A.2}$$

This is often referred to as the Third Order AutoRegressive (TOAR) covariance function (Homleid and Breivik, 1995), and implies that the wind field is only once differentiable (the stream function and velocity potential are twice differentiable). This form of covariance function is computed in much faster time than that in Equation (A.1) and thus we use (A.2) in practice. Details of the computations involved in generating the prior model can be found in (Cornford, 1998a).

One significant difference from the typical use of this type of model is that the prior is placed directly over the wind field rather than the analysis increments (difference between the NWP forecast and observations). This is because we are concerned with scatterometer data retrieval rather than assimilation, and we want to allow the possibility that the scatterometer winds could be obtained independently of an NWP model; that is *autonomous* ambiguity removal.

The probability of a wind field, (U, V), under the prior model is given by:

$$p(U, V) = \frac{1}{(2\pi)^{\frac{n}{2}} |K_{uv}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} \begin{bmatrix} U \\ V \end{bmatrix}^T K_{uv}^{-1} \begin{bmatrix} U \\ V \end{bmatrix}\right) , \tag{A.3}$$

where K_{uv} is the joint covariance matrix for U, V. The covariance matrix is determined by the TOAR covariance functions applied to the scalars, Φ and Ψ, the ratio of divergence to vorticity in the wind fields being controlled by the ratio of E_Φ^2 to E_Ψ^2. Tuning the covariance function parameters to their maximum *a posteriori* probability values using NWP data produces an accurate prior wind-field model which requires no additional NWP data once the parameters are determined, see Figure A.1. Since there is also a temporal aspect to the problem, different parameter values are used for each month of the year. Because the GP model defines a vector field over a continuous space domain it can be used for prediction at unmeasured locations, and can easily cope with missing or removed data.

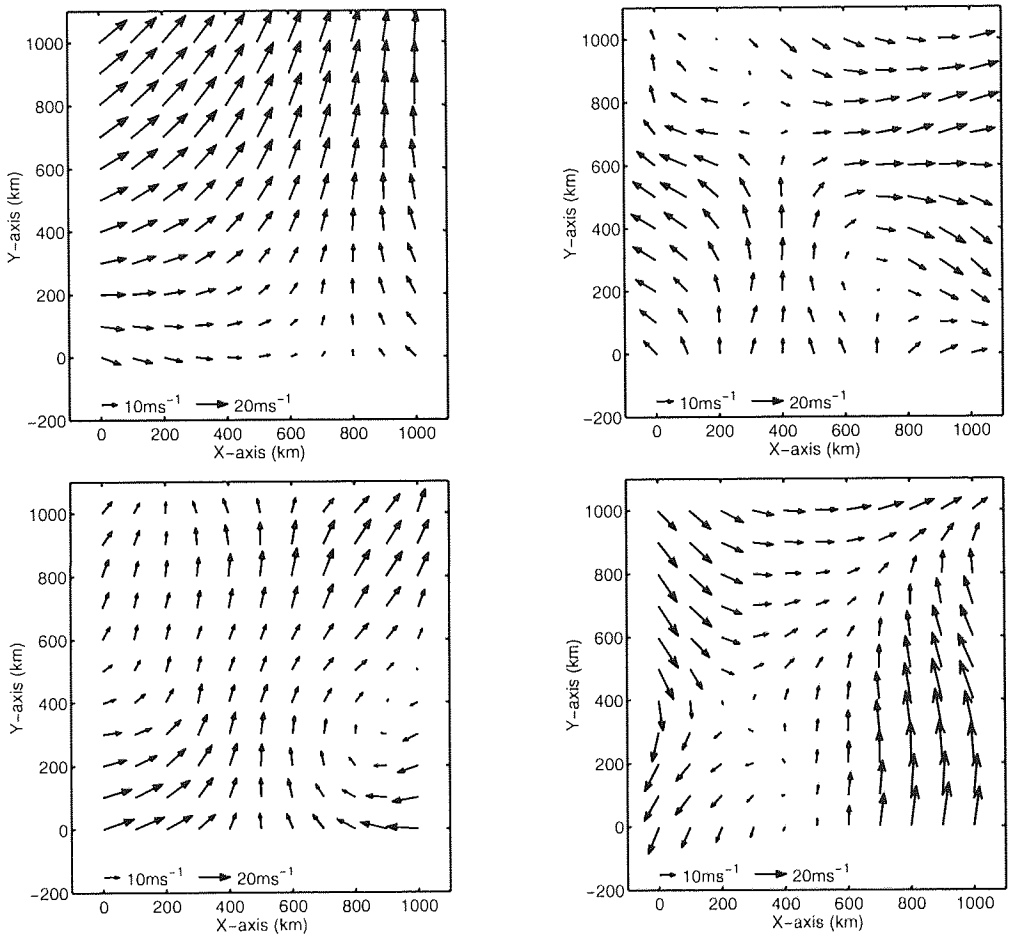


Figure A.1: Examples of realisations simulated from the modified Bessel covariance function based Gaussian process with various physically motivated parameter settings. The top left figure is based on physically realistic parameters, the top right has very short length scales (150 km), the bottom left is purely rotational flow and the bottom right is purely divergent flow.

Appendix B

The gradient of the error function of a MDN

This appendix is provided to show the reader the detail of the derivation of the gradient of the negative log likelihood of a mixture density network (the error function) with respect to the outputs of the feed forward network (which is usually a multi-layer perceptron).

B.1 The Error function and its partial derivatives

The negative log likelihood of MDN for the n^{th} training pattern, where $\boldsymbol{x}_n$ represents the n^{th} input pattern and $\boldsymbol{t}_n$ represents the n^{th} target pattern, is defined as:

$$E_n = -\ln\left\{\sum_{j=1}^{m} \alpha_j(\mathbf{x}_n)\phi_j(\mathbf{t}_n|\mathbf{x}_n)\right\}. \tag{B.1}$$

The j^{th} kernel, ϕ_j, for the n^{th} pattern, is defined as a spherical Gaussian of the form:

$$\phi_j(\boldsymbol{t}_n|\boldsymbol{x}_n) = \frac{1}{(2\pi)^{\frac{c}{2}}\sigma_j^c(\boldsymbol{x}_n)} \exp\left(-\frac{\|\boldsymbol{t}_n - \boldsymbol{\mu}_j(\boldsymbol{x}_n)\|^2}{2\sigma_j^2(\boldsymbol{x}_n)}\right). \tag{B.2}$$

The total error is the summation of the error of each pattern:

$$E = \sum_{n=1}^{N} E_n. \tag{B.3}$$

Because of (B.3) the following analysis is on a per-pattern basis. For typographical clarity references to the target and target data sets are removed where possible from (B.1) and (B.2) and are represented in the following form:

$$E_n = -\ln\left\{\sum_{j=1}^{m} \alpha_j\phi_j\right\}. \tag{B.4}$$

and

$$\phi_j = \frac{1}{(2\pi)^{\frac{c}{2}}\sigma_j^c} \exp\left(-\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right). \tag{B.5}$$

The objective is to compute the derivatives of E_n at the outputs of the MLP network. Back-propagation is used to compute the errors at the inputs of the MLP (Bishop, 1994; Bishop, 1995). The derivatives of interest (using the terminology of Bishop (1994)) are:

- The partial derivative with respect to the outputs corresponding to the mixing coefficients z^α:

$$\frac{\partial E_n}{\partial z_j^\alpha}. \tag{B.6}$$

- The partial derivative with respect to the outputs corresponding to the variances or widths z^σ:

$$\frac{\partial E_n}{\partial z_j^\sigma}. \tag{B.7}$$

- The partial derivative with respect to the outputs corresponding to the centres or positions in target space z_{jk}^μ:

$$\frac{\partial E_n}{\partial z_{jk}^\mu}. \tag{B.8}$$

In order to simplify the analysis and notation, the posterior probability of a pattern, π_j is defined. Using Bayes theorem:

$$\pi_j = \frac{\alpha_j \phi_j}{\sum_{l=1}^m \alpha_l \phi_l}, \tag{B.9}$$

where

$$0 \leqslant \pi_j \leqslant 1 \quad \forall j, \tag{B.10}$$

and

$$\sum_{j=1}^{M} \pi_j = 1. \tag{B.11}$$

B.1.1 Computing the derivatives of the mixing coefficients

The mapping constraints from the output of the MLP to the parameters of the GMM are considered when computing the partial derivatives. Using the chain rule:

$$\frac{\partial E_n}{\partial z_j^\alpha} = \sum_k \frac{\partial E_n}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial z_j^\alpha}, \tag{B.12}$$

then from (B.4):

$$\begin{aligned} \frac{\partial E_n}{\partial \alpha_k} &= -\left[\frac{1}{\sum_{j=1}^m \alpha_j \phi_j}.\phi_k\right]\frac{\alpha_k}{\alpha_k} \\ &= -\frac{\alpha_k \phi_k}{\sum_{j=1}^m \alpha_j \phi_j}\frac{1}{\alpha_k}, \end{aligned} \tag{B.13}$$

and substituting (B.9):

$$\frac{\partial E_n}{\partial \alpha_k} = -\frac{\pi_k}{\alpha_k}. \tag{B.14}$$

Some care is needed when deriving $\frac{\partial \alpha_k}{\partial z_j^\alpha}$. Each α_k represents a mixing coefficient for each Gaussian in the mixture model. To ensure that the mixing coefficients represent probabilities they must always sum to one, that is $\sum_{j=1}^{M} \alpha_j = 1$. This is achieved by using the 'softmax' function on the output of the network such that:

$$\alpha_j = \frac{\exp(z_j^\alpha)}{\sum_{l=1}^{m} \exp(z_l^\alpha)}. \tag{B.15}$$

Using the quotient rule for differentiation and considering the two cases for $j = k$ and $j \neq k$ we have:

- for the case when $j \neq k$:

$$\begin{aligned} \frac{\partial \alpha_k}{\partial z_j^\alpha} &= \frac{\sum_{l=1}^{m} \exp(z_l^\alpha).0 - \exp(z_j^\alpha)\exp(z_k^\alpha)}{(\sum_{l=1}^{m} \exp(z_l^\alpha))^2}, \\ &= -\alpha_j \alpha_k, \qquad j \neq k. \end{aligned} \tag{B.16}$$

- for the case when $j = k$,

$$\begin{aligned} \frac{\partial \alpha_k}{\partial z_j^\alpha} &= \frac{\sum_{l=1}^{m} \exp(z_l^\alpha).\exp(z_j^\alpha) - \exp(z_j^\alpha)\exp(z_k^\alpha)}{(\sum_{l=1}^{m} \exp(z_l^\alpha))^2} \\ &= \frac{\exp(z_j^\alpha)}{\sum_{l=1}^{m} \exp(z_l^\alpha)} - \left[\frac{\exp(z_j^\alpha)}{\sum_{l=1}^{m} \exp(z_l^\alpha)}\right]^2, \\ &= \alpha_k - \alpha_k^2, \qquad j = k. \end{aligned} \tag{B.17}$$

We can summarise (B.16) and (B.17) by

$$\frac{\partial \alpha_k}{\partial z_j^\alpha} = \delta_{jk}\alpha_k - \alpha_k \alpha_j \qquad \begin{cases} j = 1, 2, ..., m \\ k = 1, 2, ..., m \end{cases} \tag{B.18}$$

δ_{jk} is the Kronecker delta function. To compute the final derivative, substituting (B.14) and (B.18) into (B.12) yields

$$\begin{aligned} \frac{\partial \alpha_k}{\partial z_j^\alpha} &= \sum_k -\frac{\pi_k}{\alpha_k}\left(\delta_{jk} - \alpha_k \alpha_j\right) \\ &= \sum_k -\frac{\pi_k}{\alpha_k}\alpha_k \delta_{jk} + \sum_k \frac{\pi_k}{\alpha_k}\alpha_k \alpha_j \\ &= -\pi_j + \alpha_j. \end{aligned} \tag{B.19}$$

because $\sum_k \pi_k = 1$ and $\sum_k \pi_k \delta_{jk} = \pi_j$. Then the final result is

$$\frac{\partial E_n}{\partial z_j^\alpha} = \alpha_j - \pi_j. \tag{B.20}$$

B.1.2 Computing the derivatives of the variances

The term z_j^σ refers to the *variance* of the Gaussian. When differentiating, we must be aware that we are differentiating with respect to the variance, σ_j^2. Again, considering the mapping constraints between the outputs of the MLP and the model parameters, the chain rule is used to expand the partial derivative:

$$\frac{\partial E_n}{\partial z_j^\sigma} = \frac{\partial E_n}{\partial \sigma_j^2}\frac{\partial \sigma_j^2}{\partial z_j^\sigma}. \tag{B.21}$$

Differentiating (B.4) with respect to σ_j^2 yields:

$$\frac{\partial E_n}{\partial \sigma_j^2} = -\left[\frac{1}{\sum_{l=1}^{m} \alpha_j \phi_j} \frac{\partial(\alpha_j \phi_j)}{\partial \sigma_j^2}\right] \tag{B.22}$$

since the kernel, ϕ_j, is defined as a Spherical Gaussian (B.5), expanding (B.22) gives:

$$\alpha_j \phi_j = \alpha_j \frac{1}{(2\pi)^{\frac{c}{2}} \sigma_j^c} \exp\left(-\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right). \tag{B.23}$$

Completing the differentiation, we obtain:

$$\begin{aligned}
\frac{\partial(\alpha_j \phi_j)}{\partial \sigma_j^2} &= \alpha_j \left\{ -c \frac{1}{2\sigma_j^2} \frac{1}{(2\pi)^{\frac{c}{2}} \sigma_j^c} \exp\left(-\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) \right. \\
&\qquad \left. + \frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^4} \frac{1}{(2\pi)^{\frac{c}{2}} \sigma_j^c} \exp\left(-\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) \right\} \\
&= \underbrace{\alpha_j \frac{1}{(2\pi)^{\frac{c}{2}} \sigma_j^c} \exp\left(-\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right)}_{equation\ (B.23)} \left\{ -\frac{c}{2\sigma_j^2} + \frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^4} \right\} \\
&= \frac{\alpha_j \phi_j}{2} \left\{ -\frac{c}{\sigma_j^2} + \frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{\sigma_j^4}. \right\}
\end{aligned} \tag{B.24}$$

Combining (B.22) and (B.24):

$$\begin{aligned}
\frac{\partial E_n}{\partial \sigma_j^2} &= -\left[\frac{\alpha_j \phi_j}{2 \sum_{l=1}^{m} \alpha_l \phi_l} \left\{ -\frac{c}{\sigma_j^2} + \frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{\sigma_j^4} \right\}\right] \\
&= -\frac{\pi_j}{2} \left\{ \frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{\sigma_j^4} - \frac{c}{\sigma_j^2} \right\}.
\end{aligned} \tag{B.25}$$

The second term in expression (B.21) is easily computed:

$$\begin{aligned}
\frac{\partial \sigma_j^2}{\partial z_j^\sigma} &= \exp(z_j^\sigma) \\
&= \sigma_j^2.
\end{aligned} \tag{B.26}$$

Then substituting (B.25) and (B.26) into (B.21) the required derivative is

$$\frac{\partial E_n}{\partial z_j^\sigma} = -\frac{\pi_j}{2} \left\{ \frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{\sigma_j^2} - c \right\}. \tag{B.27}$$

B.1.3 Computing the partial derivative with respect to the kernel centres

For this derivative there is no constraint (that is to say $\mu_{jk} = z_{jk}^\mu$) applied on the output of the MLP as there is in the previous two cases. Therefore $\frac{\partial E_n}{\partial z_{jk}^\mu}$ is computed directly from (B.4):

$$\frac{\partial E_n}{\partial z_{jk}^\mu} = -\left[\frac{1}{\sum_{l=1}^{m} \alpha_l \phi_l} \frac{\partial(\alpha_j \phi_j)}{\partial z_{jk}^\mu}\right]. \tag{B.28}$$

Then differentiating ϕ_j (B.5) with respect to each z_{jk}^μ yields:

$$\begin{aligned}
\frac{\partial \phi_j}{\partial z_{jk}^\mu} &= \left(\frac{t_k - \mu_{jk}}{\sigma_j^2}\right) \frac{1}{(2\pi)^{\frac{c}{2}} \sigma_j^c} \exp\left(-\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) \\
&= \frac{t_k - \mu_{jk}}{\sigma_j^2} \phi_j.
\end{aligned} \tag{B.29}$$

Then substituting (B.29) into (B.28) yields the final result:

$$\begin{aligned}\frac{\partial E_n}{\partial z_{jk}^{\mu}} &= -\left[\frac{\alpha_j \phi_j}{\sum_{j'=1}^{m} \alpha_{j'} \phi_{j'}} \frac{t_k - \mu_{jk}}{\sigma_j^2}\right] \\ &= \pi_j \left\{\frac{\mu_{jk} - t_k}{\sigma_j^2}\right\}.\end{aligned} \tag{B.30}$$

B.2 Summary of results

The partial derivatives computed with respect to the feed forward network outputs are summarised below:

$$\begin{aligned}\frac{\partial E_n}{\partial z_j^{\alpha}} &= \alpha_j - \pi_j, \\ \frac{\partial E_n}{\partial z_j^{\sigma}} &= -\frac{\pi_j}{2}\left\{\frac{\|\mathbf{t}_n - \boldsymbol{\mu}_j\|^2}{\sigma_j^2} - c\right\}, \\ \frac{\partial E_n}{\partial z_{jk}^{\mu}} &= \pi_j \left\{\frac{\mu_{jk} - t_k}{\sigma_j^2}\right\}.\end{aligned}$$

Appendix C

MDN implementation in MATLAB

NETLAB tool box functions

This algorithm is conveniently implemented in the NETLAB toolbox for MATLAB which is available from `http://www.ncrg.aston.ac.uk/netlab/`. The following toolbox functions are relevant:

`mdn`	creates a data structure to model a MDN. This structure comprises of the feed forward network structure (an MLP) and a structure for the mixture model parameters.
`mdninit`	initialises the weights of the network. Uses the target data $\boldsymbol{t}$ to initialise the biases for the output units of the network after initialising the other weights randomly with a Gaussian prior. The biases are initialised by the parameters of a model of the unconditional density of $\boldsymbol{t}$. These parameters are computed using tool box function `gmminit`, which uses the k-means algorithm to compute the parameters of the unconditional model of $\boldsymbol{t}$.
`mdnfwd`	forward propagates the inputs through the model. The output is an array of structures containing a mixture model for each input pattern.
`mdnerr`	computes the error of the model for a set of inputs and targets.
`mdngrad`	computes the error gradient of the model using the results of (3.18), (3.19) and (3.20) and back propagating these results through the network using the tool box function `mlpbkp`
`mdnpak/unpak`	packs the weights of the network into a vector: this is required to use the optimisation routines.
`scg`	implementation of the scaled conjugate gradients algorithm, which is a general purpose optimisation algorithm.

All the MDNs trained in this project were optimised using the Scaled Conjugate Gradient (SCG) algorithm. A demonstration programme `demmdn1` is available from the web site which gives a worked example of training a MDN on the 'toy problem' described by Bishop (1994).

Training mixture density networks with 'fast mdn'

Initial experiments took considerable time to train (between a few days to one week depending on the computer) and so some time was spent during this project developing a *fast mdn* NETLAB implementation of MDNs. The focus of this work was re-engineering the computation of the error derivatives with respect to the neural network outputs, so that the computation was carried out in parameter space (3.5). The necessary expressions for the derivatives are contained in Appendix B. The details of the software re-engineering are given in Evans (1998)[1]. In summary the training time is decreased by a factor of sixty for the example provided in the technical report. The inverse model training time decreased from a few days to a few hours for networks without incidence angle as input. Decreased training time means that problems with larger data sets can be trained in a realistic time frame. For this project it was also possible to train networks that take incidence angle as an input and have a training data set size of nineteen thousand examples. This implementation is now part of the standard NETLAB release.

[1] Available from `http://www.ncrg.aston.ac.uk/Papers/`

Appendix D

The gradient of the error function of the hybrid MDN

In this appendix the derivation of the hybrid MDN framework is analysed in detail. The MDN framework is modified to encode the ambiguous directions that exist in the inverse mapping.

D.1 The Error function and its partial derivatives

The error function of a hybrid MDN is some what simpler than the full MDN:

$$E_n = -\ln\Big\{\alpha(\boldsymbol{x}_n)\phi(\boldsymbol{t}_n|\boldsymbol{x}_n) + (1-\alpha(\boldsymbol{x}_n))\psi(\boldsymbol{t}_n|\boldsymbol{x}_n)\Big\}, \tag{D.1}$$

$$\phi(\boldsymbol{t}_n|\boldsymbol{x}_n) = \frac{1}{2\pi\sigma^2(\boldsymbol{x}_n)}\exp\left(-\frac{\|\boldsymbol{t}_n - \boldsymbol{\mu}(\boldsymbol{x}_n)\|^2}{2\sigma^2(\boldsymbol{x}_n)}\right), \tag{D.2}$$

$$\psi(\boldsymbol{t}_n|\boldsymbol{x}_n) = \frac{1}{2\pi\sigma^2(\boldsymbol{x}_n)}\exp\left(-\frac{\|\boldsymbol{t}_n + \boldsymbol{\mu}(\boldsymbol{x}_n)\|^2}{2\sigma^2(\boldsymbol{x}_n)}\right), \tag{D.3}$$

simplified thus:

$$E_n = -\ln\Big\{\alpha\phi + (1-\alpha)\psi\Big\}. \tag{D.4}$$

Define the two posterior probabilities for each pattern:

The free centre:

$$\pi = \frac{\alpha\phi}{\alpha\phi + (1-\alpha)\psi}. \tag{D.5}$$

and the hybrid centre:

$$\gamma = \frac{(1-\alpha)\psi}{\alpha\phi + (1-\alpha)\psi}. \tag{D.6}$$

D.1.1 Computing the derivatives of the mixing coefficients

Using the chain rule:

$$\frac{\partial E_n}{\partial z^\alpha} = \frac{\partial E_n}{\partial \alpha}\frac{\partial \alpha}{\partial z^\alpha}, \tag{D.7}$$

we compute the derivative of (D.4):

$$\begin{aligned}\frac{\partial E_n}{\partial \alpha} &= -\left[\frac{1}{\alpha\phi + (1-\alpha)\psi}(\phi - \psi)\right] \\ &= -\left[\frac{\phi}{\alpha\phi + (1-\alpha)\psi} - \frac{\psi}{\alpha\phi + (1-\alpha)\psi}\right].\end{aligned} \tag{D.8}$$

Then, simplifying by using posterior distributions,

$$\begin{aligned}&= -\left[\frac{\phi}{\alpha\phi + (1-\alpha)\psi}\frac{\alpha}{\alpha} - \frac{\psi}{\alpha\phi + (1-\alpha)\psi}\frac{(1-\alpha)}{(1-\alpha)}\right] \\ &= -\left[\frac{\phi}{\alpha} - \frac{\gamma}{(1-\alpha)}\right] \\ &= -\left[\frac{\phi}{\alpha} - \frac{\gamma}{(1-\alpha)}\right] \\ &= -\left[\frac{(1-\alpha)\pi - \alpha\gamma}{\alpha(1-\alpha)}\right].\end{aligned} \tag{D.9}$$

But as $\gamma = 1 - \pi$, the required derivative is:

$$\frac{\partial E_n}{\partial \alpha} = \frac{\pi - \alpha}{\alpha(1-\alpha)}. \tag{D.10}$$

The mixing coefficient α is a probability, and therefore is constrained by $0 \leqslant \alpha \leqslant 1$. The logistic function on the output of the MLP achieves this:

$$\alpha = \frac{1}{1 + \exp(-z^\alpha)}. \tag{D.11}$$

Calculating the second term of (D.7),

$$\begin{aligned}\frac{\partial \alpha}{\partial z^\alpha} &= \frac{\exp(-z^\alpha)}{(1 + \exp(-z^\alpha))^2} \\ &= \frac{1}{(1 + \exp(-z^\alpha))}\left(\frac{\exp(-z^\alpha)}{1 + \exp(-z^\alpha)}\right) \\ &= \alpha\left(1 - \frac{1}{(1 + \exp(-z^\alpha))}\right) \\ &= \alpha(1-\alpha).\end{aligned} \tag{D.12}$$

Combining (D.10) and (D.12) the error derivative with respect to the mixing coefficients is given by,

$$\frac{\partial E_n}{\partial z^\alpha} = \pi - \alpha. \tag{D.13}$$

D.1.2 Computing the derivatives of the kernel variances (widths)

Using the chain rule:

$$\frac{\partial E_n}{\partial z^\sigma} = \frac{\partial E_n}{\partial \sigma^2}\frac{\partial \sigma^2}{\partial z^\sigma}, \tag{D.14}$$

and differentiating (D.4) with respect to σ^2 yields:

$$\begin{aligned}\frac{\partial E_n}{\partial \sigma^2} &= -\left[\frac{1}{\alpha\phi + (1-\alpha)\psi}\frac{\partial(\alpha\phi + (1-\alpha)\psi)}{\partial \sigma^2}\right] \\ &= -\left[\alpha\phi\left\{-c\frac{1}{2\sigma^2} + \frac{\|\boldsymbol{t}_n - \boldsymbol{\mu}\|^2}{2\sigma^4}\right\}\right. \\ &\quad \left. + (1-\alpha)\psi\left\{-c\frac{1}{2\sigma^2} + \frac{\|\boldsymbol{t}_n + \boldsymbol{\mu}\|^2}{2\sigma^4}\right\}\right]\frac{1}{\alpha\phi + (1-\alpha)\psi} \\ &= -\left[\frac{\pi}{2}\left\{-\frac{c}{\sigma^2} + \frac{\|\boldsymbol{t}_n - \boldsymbol{\mu}\|^2}{\sigma^4}\right\} + \frac{\gamma}{2}\left\{-\frac{c}{\sigma^2} + \frac{\|\boldsymbol{t}_n + \boldsymbol{\mu}\|^2}{\sigma^4}\right\}\right].\end{aligned} \tag{D.15}$$

The second term in expression (D.14) is easily computed:

$$\begin{aligned}\frac{\partial \sigma^2}{\partial z^\sigma} &= \exp(z^\sigma) \\ &= \sigma^2,\end{aligned} \tag{D.16}$$

and combining (D.15) and (D.16) equation (D.14) is complete:

$$\frac{\partial E_n}{\partial z^\sigma} = -\left[\frac{\pi}{2}\left\{\frac{\|\boldsymbol{t}_n - \boldsymbol{\mu}\|^2}{\sigma^2} - c\right\} + \frac{\gamma}{2}\left\{\frac{\|\boldsymbol{t}_n + \boldsymbol{\mu}\|^2}{\sigma^2} - c\right\}\right]. \tag{D.17}$$

D.1.3 Computing the derivatives of the kernel centres (means)

For this derivative there is no constraint (that is to say $\mu_{\mathbf{k}} = z_k^\mu$) applied on the output of the MLP as there is in the previous two cases. Therefore $\frac{\partial E_n}{\partial z_{jk}^\mu}$ is computed directly from (B.4):

$$\frac{\partial E_n}{\partial z_k^\mu} = -\left[\frac{1}{\alpha\phi + (1-\alpha)\psi}\frac{\partial(\alpha\phi + (1-\alpha)\psi)}{\partial z_k^\mu}\right], \tag{D.18}$$

and,

$$\frac{\partial(\alpha\phi + (1-\alpha)\psi)}{\partial z_k^\mu} = \alpha\phi\left(\frac{t_k - \mu_k}{\sigma^2}\right) - (1-\alpha)\psi\left(\frac{t_k + \mu_k}{\sigma^2}\right). \tag{D.19}$$

Combining (D.18) and (D.19) gives the required result:

$$\frac{\partial E_n}{\partial z_k^\mu} = -\left[\pi\left(\frac{t_k - \mu_k}{\sigma^2}\right) - \gamma\left(\frac{t_k + \mu_k}{\sigma^2}\right)\right]. \tag{D.20}$$

D.2 Summary of results

The partial derivatives computed with respect to the feed forward network outputs are summarised below:

$$\begin{aligned}\frac{\partial E_n}{\partial z^\alpha} &= \pi - \alpha, \\ \frac{\partial E_n}{\partial z^\sigma} &= -\left[\frac{\pi}{2}\left\{\frac{\|\boldsymbol{t}_n - \boldsymbol{\mu}\|^2}{\sigma^2} - c\right\} + \frac{\gamma}{2}\left\{\frac{\|\boldsymbol{t}_n + \boldsymbol{\mu}\|^2}{\sigma^2} - c\right\}\right], \\ \frac{\partial E_n}{\partial z_k^\mu} &= -\left[\pi\left(\frac{t_k - \mu_k}{\sigma^2}\right) - \gamma\left(\frac{t_k + \mu_k}{\sigma^2}\right)\right].\end{aligned}$$

Appendix E

Definitions of the summary measures used in the results

This appendix gives the details of the tools used to analyse the performance of the inverse model, FoM and vector root mean square error. These statistics are computed after applying a simple disambiguation procedure which is detailed first.

E.1 Disambiguation for model appraisal

The following method of disambiguation permits the comparison of inverse model performance in terms of the quality of retrieved wind vectors. The predicted direction, D_{pred}, and predicted wind speed, U_{pred}, are determined using a simple de-aliasing algorithm. The observed wind vector, V_{obs} (derived from the numerical weather prediction model, and gives the best estimate of a 'true' wind vector available), is compared with the two most probable wind vectors inferred from the model by measuring the Euclidean distance between each inferred wind vector and the observed wind vector. The predicted wind vector, V_{pred}, is chosen as the wind vector with a minimum Euclidean distance from the observed wind vector. The predicted wind vector is then resolved to compute the predicted direction D_{pred} and the predicted wind speed U_{pred}.

E.2 Figure of Merit

This measure was proposed by David Offiler of the UK Meteorological Office and is becoming a widely used statistic for comparing the performance of models within this field (Richaume *et al.*, 2000; Cornford *et al.*, 1999):

$$FoM = \frac{(F1 + F2 + F3)}{3}, \tag{E.1}$$

where:

$$F1 = \frac{40}{|U_{bias}| + 10U_{sd} + |D_{bias}| + D_{sd}}, \tag{E.2}$$

$$F2 = \frac{1}{2}\left(\frac{2}{U_{rms}} + \frac{20}{D_{rms}}\right), \tag{E.3}$$

$$F3 = \frac{4}{V_{rms}}. \tag{E.4}$$

where U represents wind speed, D the wind direction and V the wind vector (u, v). The parameters are defined as follows.

The wind speed bias

$$U_{bias} = \frac{1}{N}\sum_{i=1}^{N} U_{res(i)}, \tag{E.5}$$

where the residue between the target and predicted wind speed, $U_{res(i)}$, is

$$U_{res} = U_{pred} - U_{obs}. \tag{E.6}$$

The standard deviation of the estimates of the wind speed,

$$U_{sd} = \sqrt{\left(\frac{1}{N}\sum_{i=1}^{N}(U_{res(i)})^2\right) - (U_{bias})^2)}. \tag{E.7}$$

The root mean square error of the wind speed,

$$U_{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(U_{res(i)})^2}. \tag{E.8}$$

The vector root mean square error of the wind vector, (u, v),

$$V_{res} = \sqrt{U_{obs}^2 + U_{pred}^2 - 2U_{obs}U_{pred}\cos(D_{res})}, \tag{E.9}$$

where the residue wind direction is the difference between the predicted and observed directions,

$$D_{res} = D_{pred} - D_{obs}. \tag{E.10}$$

Assuming bias free estimators, then an $FoM = 1$ implies that a wind speed error of $\pm 2\ ms^{-1}$ and a wind direction of $\pm 20°$ are meet; which is the general specification to which the NWP are quoted. An FoM of greater than one implies better wind vector retrievals.

E.3 Predicted wind vector root-mean-square error

The predicted wind vector root-mean-square error is defined as

$$V_{rms} = \sqrt{\sum_{i}^{N}(u_{res(i)}^2 + v_{res(i)}^2)}, \tag{E.11}$$

where the residuals of u_i, v_i are:

$$u_{res(i)}^2 = (u_{pred(i)} - u_{obs(i)})^2 \tag{E.12}$$

and

$$v_{res(i)}^2 = (v_{pred(i)} - v_{obs(i)})^2, \tag{E.13}$$

where the predicted wind vectors are obtained using the method detailed in Subsection E.1

E.4 *Performance at* 20°

Performance at 20° (denoted *perf.* @ 20°) is a statistic that measures the percentage predicted wind directions that are within 20° of the target wind direction. This statistic is used in work by Thiria *et al.* (1993), Cornford *et al.* (1999) and Richaume *et al.* (2000).

Appendix F

Tabulated results for data sets YR1TRACE and YR1SWATHE

FoM- results for networks trained per trace											
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9	average over swathe
Two hybrid Kernels, 10 Hidden Units	0.76	0.80	0.79	0.82	0.83	0.83	0.87	0.81	0.89	1.02	0.84
Two Kernels, 10 Hidden Units	0.82	0.75	0.80	0.78	0.84	0.79	0.79	0.81	0.85	0.93	0.82
Four Kernels, 10 Hidden Units	0.88	0.86	**0.92**	0.97	0.92	0.95	0.99	1.02	**1.02**	**1.19**	0.97
Two hybrid Kernels, 15 Hidden Units	0.90	0.83	0.83	0.90	0.90	0.88	0.88	0.86	0.95	1.03	0.90
Two Kernels, 15 Hidden Units	0.90	0.83	0.81	0.88	0.88	0.87	0.87	0.84	0.95	1.08	0.89
Four Kernels, 15 Hidden Units	**0.95**	0.91	0.87	**1.00**	0.91	0.87	1.00	0.94	**1.02**	1.13	0.96
Two hybrid Kernels, 20 Hidden Units	0.93	0.83	0.85	0.90	0.87	0.89	0.89	0.89	0.97	1.07	0.91
Two Kernels, 20 Hidden Units	0.92	0.84	0.82	0.85	0.88	0.88	0.89	0.90	0.97	1.07	0.90
Four Kernels, 20 Hidden Units	0.93	**0.92**	0.86	0.98	**0.93**	**0.98**	**1.02**	**1.00**	0.99	1.16	**0.98**
Two hybrid Kernels, 25 Hidden Units	0.89	0.82	0.86	0.91	0.91	0.91	0.91	0.91	0.98	1.06	0.92
Two Kernels, 25 Hidden Units	0.93	0.85	0.84	0.92	0.89	0.93	0.93	0.95	0.97	1.07	0.93
Four Kernels, 25 Hidden Units	0.88	0.88	0.87	0.99	0.86	0.97	1.00	0.95	0.97	1.17	0.95

Table F.1: *FoM* results - for networks trained per trace. Results in **bold face** indicate best results per column.

Performance summary for networks trained with incidence angle as an input			
MDN architecture	*FoM*	RMS Errors	Perf @ 20°
Two Hybrid Kernels, 35 Hidden Units	0.85	4.18	73.54
Two Kernels, 35 Hidden Units	0.88	4.03	73.28
Four Kernels, 35 Hidden Units	0.96	3.83	76.96
Five Kernels, 35 Hidden Units	0.96	**3.84**	76.66
Twelve Kernels, 35 Hidden Units	0.80	5.13	74.82
Two Hybrid Kernels, 50 Hidden Units	0.82	4.33	71.10
Two Kernels, 50 Hidden Units	0.89	4.02	74.58
Four Kernels, 50 Hidden Units	0.96	3.86	77.08
Five Kernels, 50 Hidden Units	**0.97**	**3.84**	**77.14**
Twelve Kernels, 50 Hidden Units	0.82	4.87	75.32

Table F.4: Performance results over the whole swathe - for networks trained with incidence angle as an input. These results are generated with the test data set of 5000 examples. Results in **bold face** indicate best results per column.

Vector RMS error - results for networks trained per trace											
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9	average over swathe
Two hybrid Kernels, 10 Hidden Units	4.71	4.29	4.36	4.29	4.27	4.14	4.09	4.26	3.90	3.50	4.18
Two Centres, 10 Hidden Units	4.25	4.43	4.22	4.45	4.08	4.26	4.42	4.24	4.09	3.74	4.22
Four Centres, 10 Hidden Units	4.27	4.16	**3.77**	3.73	4.06	3.68	3.74	3.60	**3.52**	**3.11**	3.76
Two hybrid Kernels, 15 Hidden Units	3.90	4.17	4.19	3.93	3.88	3.95	4.01	4.06	3.70	3.44	3.92
Two Centres, 15 Hidden Units	3.90	4.09	4.10	4.04	3.91	3.96	3.95	4.05	3.66	3.28	3.90
Four Centres, 15 Hidden Units	3.86	4.00	4.16	**3.66**	4.00	4.31	3.54	4.01	**3.52**	3.25	3.83
Two hybrid Kernels, 20 Hidden Units	3.83	4.17	4.03	3.96	4.04	3.95	3.96	3.98	3.63	3.34	3.89
Two Centres, 20 Hidden Units	3.81	4.04	4.13	4.07	3.90	3.93	3.86	3.86	3.61	3.31	3.85
Four Centres, 20 Hidden Units	3.94	**3.84**	4.08	3.71	3.89	**3.63**	**3.48**	3.67	3.63	3.17	**3.70**
Two hybrid Kernels, 25 Hidden Units	3.97	4.24	3.99	3.94	**3.86**	3.87	3.92	3.87	3.61	3.36	3.86
Two Centres, 25 Hidden Units	**3.79**	4.00	3.99	3.82	3.89	3.76	3.77	**3.65**	3.61	3.28	3.76
Four Centres, 25 Hidden Units	4.20	3.97	4.16	3.67	4.24	3.70	3.63	4.14	3.94	3.15	3.88

Table F.2: Vector RMS error results - for networks trained per trace. Results in **bold face** indicate best results per column.

Performance @ 20° - results for networks trained per trace											
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9	average over swathe
Two hybrid Kernels, 10 Hidden Units	61.60	66.30	67.40	69.90	70.10	72.90	69.60	69.60	73.60	80.30	70.13
Two Centres, 10 Hidden Units	64.40	67.10	67.10	69.50	73.20	73.70	69.50	70.30	74.10	78.10	70.70
Four Centres, 10 Hidden Units	71.00	70.30	**72.60**	74.40	74.70	75.70	75.40	**75.00**	**77.60**	**81.90**	74.86
Two hybrid Kernels, 15 Hidden Units	66.10	68.80	67.50	72.50	72.90	74.20	71.00	72.00	74.90	79.00	71.89
Two Centres, 15 Hidden Units	69.30	68.50	70.70	72.80	74.40	75.10	73.10	72.90	75.80	79.40	73.20
Four Centres, 15 Hidden Units	**72.50**	71.00	72.30	**75.00**	**76.20**	75.10	**76.00**	74.80	77.40	81.10	75.14
Two hybrid Kernels, 20 Hidden Units	68.30	67.20	69.20	72.60	72.50	74.00	72.20	72.10	75.70	81.20	72.50
Two Centres, 20 Hidden Units	69.10	69.10	69.50	72.60	73.80	75.80	74.20	74.80	**77.60**	81.10	73.76
Four Centres, 20 Hidden Units	72.30	**71.40**	71.80	74.20	76.10	75.70	75.30	74.70	76.50	81.00	**74.90**
Two hybrid Kernels, 25 Hidden Units	67.20	67.70	69.80	72.30	73.00	75.00	72.00	71.40	75.80	80.60	72.48
Two Centres, 25 Hidden Units	69.20	69.50	70.70	73.60	73.70	75.10	73.60	74.10	75.90	79.50	73.49
Four Centres, 25 Hidden Units	71.60	69.10	72.50	74.40	75.40	**76.50**	75.70	74.80	76.90	81.70	74.86

Table F.3: *Performance* @ 20° - for networks trained per trace. The results are computed using the wind direction obtained by the 'perfect' ambiguity removal algorithm described in the text. Results in **bold face** indicate best results per column.

FoM- results by trace, for networks that take incidence angle as an input											
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9	average over swathe
Two hybrid Kernels, 35 Hidden Units	0.80	0.79	0.82	0.86	0.86	0.88	0.88	0.90	0.92	0.97	0.87
Two Kernels, 35 Hidden Units	0.81	0.81	0.82	0.88	0.88	0.91	0.93	0.92	0.93	0.99	0.89
Four Kernels, 35 Hidden Units	**0.88**	0.84	0.84	0.94	0.92	0.89	0.92	0.95	0.93	**1.12**	0.92
Five Kernels, 35 Hidden Units	0.79	**0.90**	**0.89**	0.97	0.92	**0.98**	0.96	0.98	**1.00**	1.04	0.94
Twelve Kernels, 35 Hidden Units	0.60	0.71	0.79	0.92	0.85	0.81	0.80	0.85	0.90	0.99	0.82
Two hybrid Kernels, 50 Hidden Units	0.79	0.77	0.79	0.84	0.85	0.87	0.88	0.87	0.90	0.97	0.85
Two Kernels, 50 Hidden Units	0.80	0.81	0.83	0.89	0.88	0.92	0.91	0.94	0.95	1.02	0.90
Four Kernels, 50 Hidden Units	0.85	0.85	0.87	**0.99**	0.91	0.96	0.98	0.97	0.99	1.04	0.94
Five Kernels, 50 Hidden Units	0.84	0.89	0.84	0.98	**0.95**	0.97	**0.99**	**1.00**	0.99	1.05	**0.95**
Twelve Kernels, 50 Hidden Units	0.63	0.75	0.84	0.87	0.83	0.80	0.82	0.85	0.82	0.89	0.81

Table F.5: *FoM* results - by trace, for networks trained with incidence as angle as an input. Results in **bold face** indicate best results per column.

Vector RMS error - results by trace, for networks that take incidence angle as an input											
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9	average over swathe
Two hybrid Kernels, 35 Hidden Units	4.41	4.45	4.23	4.12	4.05	3.94	3.98	3.94	3.81	3.65	4.06
Two Kernels, 35 Hidden Units	4.28	4.28	4.13	3.97	3.93	3.81	3.74	3.84	3.69	3.46	3.91
Four Kernels, 35 Hidden Units	**4.22**	4.33	4.08	4.11	4.01	4.40	4.10	4.08	3.94	**3.30**	4.06
Five Kernels, 35 Hidden Units	4.91	**4.09**	3.85	3.88	4.12	**3.62**	3.75	3.71	**3.56**	3.45	3.89
Twelve Kernels, 35 Hidden Units	7.89	5.38	4.75	4.08	4.36	4.93	5.24	4.77	4.43	4.05	4.99
Two hybrid Kernels, 50 Hidden Units	4.54	4.60	4.38	4.17	4.12	4.00	3.99	4.03	3.86	3.72	4.14
Two Kernels, 50 Hidden Units	4.33	4.27	4.13	3.99	3.92	3.77	3.79	3.75	3.69	3.49	3.91
Four Kernels, 50 Hidden Units	4.28	4.16	4.03	3.74	4.21	3.90	3.75	3.94	3.69	3.45	3.91
Five Kernels, 50 Hidden Units	4.33	4.16	**3.99**	**3.74**	**3.85**	3.70	**3.65**	**3.68**	3.69	3.54	**3.83**
Twelve Kernels, 50 Hidden Units	7.08	5.16	4.33	4.78	4.90	5.18	5.12	4.67	4.84	4.39	5.04

Table F.6: Vector RMS error - by trace, for networks trained with incidence as angle as an input. Results in **bold face** indicate best results per column.

Performance at 20° - results by trace, for networks that take incidence angle as an input											
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9	average over swathe
Two hybrid Kernels, 35 Hidden Units	61.60	63.60	69.50	71.70	71.80	73.90	71.00	72.00	74.50	79.10	70.87
Two Kernels, 35 Hidden Units	63.50	64.20	68.20	71.70	72.40	75.30	71.80	73.10	74.30	79.00	71.35
Four Kernels, 35 Hidden Units	**66.90**	66.30	71.30	73.50	77.00	74.90	74.30	75.60	75.70	**80.50**	73.60
Five Kernels, 35 Hidden Units	63.70	69.00	**71.40**	74.20	76.20	76.40	75.00	**75.80**	77.10	79.30	73.81
Twelve Kernels, 35 Hidden Units	63.10	**69.10**	71.20	73.40	75.10	74.40	74.20	74.80	77.50	79.90	73.27
Two hybrid Kernels, 50 Hidden Units	58.50	60.50	66.80	70.70	70.80	73.00	71.20	72.20	74.60	79.00	69.73
Two Kernels, 50 Hidden Units	62.80	66.10	69.20	72.30	73.30	76.20	73.60	74.40	76.30	79.60	72.38
Four Kernels, 50 Hidden Units	65.80	68.10	71.00	73.30	76.80	**76.80**	**75.90**	74.70	**77.80**	79.60	**73.98**
Five Kernels, 50 Hidden Units	65.00	68.10	71.20	74.00	76.90	76.20	75.00	75.40	76.60	79.20	73.76
Twelve Kernels, 50 Hidden Units	64.60	68.40	**71.40**	**74.10**	**77.10**	75.80	73.30	72.50	75.00	76.20	72.84

Table F.7: *Performance @* 20° - by trace, for networks trained with incidence as angle as an input The results are computed using the wind direction obtained by the 'perfect' ambiguity removal algorithm described in the text. Results in **bold face** indicate best results per column.

Predicted wind direction chosen from the first two most probable aliases, performance at 20° (%)										
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9	average
MDN_{trace}	72.3	71.4	71.8	74.2	76.1	75.7	75.3	74.7	76.5	81.0	74.9
$MDN_{incidence}$	63.7	69.0	71.4	74.2	76.2	76.4	75.0	75.8	77.1	79.3	73.8
$A\text{-}NN_i$	85.1	85.0	86.9	87.7	87.5	87.8	87.6	88.0	88.2	86.9	87.1

Table F.8: Comparing the direction performance of the best MDNs with published results.

Predicted wind speed bias ($m.s^{-1}$)									
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9
MDN_{trace}	0.0	-0.3	-0.1	0.0	-0.1	0.0	0.1	0.0	-0.1	0.1
$MDN_{incidence}$	0.0	-0.3	-0.1	0.0	-0.1	0.0	0.1	0.0	-0.1	0.1
$A\text{-}NN_i$	-0.2	0.0	-0.1	-0.1	-0.1	-0.1	-0.1	-0.1	0.0	0.0
Predicted wind speed RMS (ms^{-1})										
MDN architecture	trace 0	trace 1	trace 2	trace 3	trace 4	trace 5	trace 6	trace 7	trace 8	trace 9
MDN_{trace}	1.9	2.1	2.1	2.0	2.1	2.1	2.1	2.0	2.0	1.8
$MDN_{incidence}$	1.9	2.1	2.1	2.0	2.1	2.1	2.1	2.0	2.0	1.8
$A\text{-}NN_i$	1.6	1.6	1.5	1.6	1.6	1.6	1.6	1.6	1.6	1.7

Table F.9: Comparing the speed performance of the best MDNs with published results.

Appendix G

Tabulated results for data sets

YR2SWATHE

complexity	negative log likelihood	vector RMS error	*FoM*	first	second	speed bias	speed standard deviation	direction bias	direction standard deviation
10	4.84	3.66	0.80	0.46	0.79	−0.53	2.64	**−0.04**	35.18
15	4.69	3.33	0.90	0.48	0.81	−0.34	2.33	0.40	30.45
20	4.54	3.23	0.94	0.55	0.81	−0.28	2.07	−0.55	31.41
25	4.58	3.24	0.93	0.51	0.82	−0.29	2.25	−0.63	29.49
30	4.49	3.08	0.98	0.55	0.82	−0.23	2.07	−0.16	29.13
35	**4.43**	3.01	1.00	**0.57**	0.82	−0.21	**1.94**	−0.18	29.67
40	4.44	**2.90**	**1.05**	0.56	**0.83**	**−0.14**	1.96	0.11	**26.29**

Table G.1: YR2SWATHE test set results - MDNs with two kernels after 4000 training epochs. Results in **bold face** indicate best result(s) per column.

complexity	negative log likelihood	vector RMS error	*FoM*	first	second	speed bias	speed standard deviation	direction bias	direction standard deviation
10	4.86	3.75	0.78	0.40	0.77	−0.70	2.70	−0.33	37.10
15	4.68	3.31	0.91	0.48	0.80	−0.31	2.16	−0.24	31.98
20	4.60	3.26	0.92	0.52	0.80	−0.27	2.13	0.61	32.48
25	4.53	3.07	0.99	0.56	**0.82**	−0.21	1.95	**0.00**	**29.58**
30	4.51	3.10	0.97	0.56	0.81	−0.23	2.00	0.15	30.68
35	4.48	**3.03**	**1.00**	0.57	**0.82**	**−0.17**	1.95	−0.24	29.97
40	**4.47**	**3.03**	**1.00**	**0.58**	**0.82**	−0.19	**1.93**	−0.09	29.72

Table G.2: YR2SWATHE test set results - Hybrid MDNs with two kernels after 4000 training epochs. Results in **bold face** indicate best result(s) per column.

complexity	negative log likelihood	vector RMS error	*FoM*	first	second	speed bias	speed standard deviation	direction bias	direction standard deviation
10	4.36	2.38	**1.34**	0.49	0.85	**0.07**	1.51	−0.21	**19.74**
15	4.25	2.37	1.32	0.56	0.85	0.09	1.50	−0.10	21.28
20	4.28	2.42	1.30	0.57	0.85	0.09	1.51	−0.16	21.15
25	4.22	**2.34**	1.31	0.58	0.83	**0.07**	1.49	**−0.06**	22.14
30	**4.20**	**2.34**	1.32	**0.59**	**0.86**	0.08	**1.48**	−0.19	21.40
35	**4.20**	2.35	1.31	**0.59**	0.85	0.08	1.49	−0.48	21.52
40	4.33	2.86	1.08	0.57	0.84	−0.09	1.96	−0.62	24.25

Table G.3: YR2SWATHE test set results - MDNs with four kernels after 4000 training epochs. Results in **bold face** indicate best result(s) per column.

complexity	negative log likelihood	vector RMS error	*FoM*	first	second	speed bias	speed standard deviation	direction bias	direction standard deviation
10	4.40	2.43	1.30	0.49	0.84	0.07	1.53	−0.19	**20.94**
15	4.33	2.43	1.26	0.56	0.85	**0.06**	1.54	0.31	23.39
20	4.28	2.41	1.28	0.59	0.83	0.07	1.52	**−0.09**	22.64
25	4.28	2.41	1.28	**0.60**	0.85	0.07	1.51	−0.17	22.43
30	4.27	2.40	1.28	0.59	0.85	0.07	1.51	−0.22	22.65
35	**4.25**	**2.39**	**1.31**	**0.60**	**0.86**	0.08	**1.49**	−0.14	21.30
40	**4.25**	2.41	1.30	**0.60**	0.85	0.09	1.50	−0.20	21.57

Table G.4: YR2SWATHE test set results - Hybrid MDNs with four kernels after 4000 training epochs. Results in **bold face** indicate best result(s) per column.

complexity	negative log likelihood	vector RMS error	FoM	first	second	speed bias	speed standard deviation	direction bias	direction standard deviation
10	4.34	2.38	1.32	0.51	0.85	0.09	1.50	−0.28	**20.54**
15	4.24	2.37	1.31	0.56	0.85	0.08	1.49	−0.17	21.60
20	4.23	2.39	1.32	0.58	0.85	0.10	1.50	**−0.05**	21.01
25	4.20	**2.35**	**1.33**	0.59	0.85	0.08	1.49	−0.25	20.81
30	**4.19**	2.36	1.29	**0.60**	0.85	0.07	1.49	−0.39	22.96
35	**4.19**	**2.35**	1.32	**0.60**	**0.86**	0.07	**1.48**	−0.31	21.71
40	4.27	2.65	1.14	0.58	0.85	**−0.03**	1.80	−0.35	24.37

Table G.5: YR2SWATHE test set results - MDNs with four kernels after 8000 training epochs. Results in **bold face** indicate best result(s) per column.

complexity	negative log likelihood	vector RMS error	FoM	first	second	speed bias	speed standard deviation	direction bias	direction standard deviation
10	4.40	2.43	1.30	0.49	0.84	0.07	1.53	−0.19	**20.94**
15	4.33	2.43	1.26	0.56	0.85	**0.06**	1.54	0.31	23.39
20	4.28	2.41	1.28	0.59	0.83	0.07	1.52	**−0.09**	22.64
25	4.28	2.41	1.28	**0.60**	0.85	0.07	1.51	−0.17	22.43
30	4.27	2.40	1.28	0.59	0.85	0.07	1.51	−0.22	22.65
35	**4.25**	**2.39**	**1.31**	**0.60**	**0.86**	0.08	**1.49**	−0.14	21.30
40	**4.25**	2.41	1.30	**0.60**	0.85	0.09	1.50	−0.20	21.57

Table G.6: YR2SWATHE test set results - Hybrid MDNs with four kernels after 8000 training epochs. Results in **bold face** indicate best result(s) per column.

negative log likelihood	vector RMS error	*FoM*	first	second	speed bias	speed standard deviation	direction bias	direction standard deviation
4.35	2.56	1.23	0.57	0.84	0.03	1.60	−0.40	22.07
4.33	2.50	1.27	0.56	0.84	0.04	1.56	−0.11	21.46
4.50	**3.11**	**0.97**	**0.54**	**0.82**	−0.23	2.17	−0.46	28.31
4.34	2.50	1.28	0.56	0.84	0.07	1.54	−0.38	20.62
4.21	2.34	1.32	0.58	0.83	0.08	1.49	−0.15	21.30
4.22	2.39	1.29	0.59	0.85	0.09	1.50	−0.60	22.13
4.21	2.36	1.33	0.59	0.85	0.09	1.49	−0.10	20.94
4.45	**3.00**	**0.98**	**0.54**	**0.82**	−0.19	2.08	−0.35	30.20
4.21	2.36	1.31	0.59	0.85	0.07	1.49	−0.11	21.84

Table G.7: YR2SWATHE test set results - MDNs with the same configuration, four kernels and twenty five hidden units, but with different seeds, after 4000 epochs. Note the outliers, highlighted in bold, where the optimisation of the weight space has landed in a less than optimal local minima.

committee	vector RMS error	*FoM*	first	second	speed bias	speed standard deviation	direction bias	direction standard deviation
a	**2.35**	**1.33**	**0.60**	**0.86**	0.08	1.48	−0.34	**20.92**
b	**2.35**	1.32	0.59	0.84	0.08	1.48	−0.24	21.30
c	2.36	1.31	**0.60**	0.85	0.09	1.48	**−0.16**	21.68
d	2.37	1.31	**0.60**	**0.86**	0.08	1.49	−0.33	21.27

Table G.8: YR2SWATHE test set results for committee of MDNs with four kernels. See text, Section 4.8, for committee members.

complexity	negative log likelihood	vector RMS error	*FoM*	first	second	speed bias	speed standard deviation	direction bias	direction standard deviation
10	4.34	2.21	1.45	0.55	0.89	**0.04**	1.38	−0.32	18.11
15	4.24	**2.19**	1.42	0.61	**0.90**	**0.04**	1.37	−0.42	19.80
20	4.23	2.21	1.45	0.63	0.89	0.06	1.37	−0.05	18.22
25	4.20	**2.19**	**1.47**	0.64	**0.90**	**0.04**	**1.36**	**−0.01**	**17.91**
30	**4.19**	**2.19**	1.43	**0.65**	**0.90**	**0.04**	**1.36**	−0.55	19.48
35	**4.19**	**2.19**	1.45	**0.65**	**0.90**	0.05	**1.36**	−0.32	18.60
40	4.27	2.50	1.22	0.62	0.89	−0.07	1.70	−0.49	22.29

Table G.9: YR2SWATHE validation set results - MDNs with four kernels, after 8000 training epochs. Results in **bold face** indicate best result(s) per column.

complexity	negative log likelihood	vector RMS error	*FoM*	first	second	speed bias	speed standard deviation	direction bias	direction standard deviation
10	4.39	2.27	**1.47**	0.55	0.89	0.03	1.39	**−0.04**	**16.58**
15	4.32	2.26	1.38	0.61	0.89	**0.02**	1.40	−0.25	20.51
20	4.26	**2.23**	1.42	0.64	0.89	0.03	1.39	−0.13	19.17
25	4.27	2.25	1.42	0.64	0.89	0.03	**1.37**	−0.24	19.05
30	4.26	2.24	1.43	**0.65**	**0.90**	0.03	1.38	−0.08	18.87
35	4.25	2.24	1.44	**0.65**	0.89	0.05	**1.37**	−0.32	18.34
40	**4.24**	**2.23**	1.44	**0.65**	**0.90**	0.06	1.38	−0.07	18.46

Table G.10: YR2SWATHE validation set results - Hybrid MDNs with four kernels, after 8000 training epochs. Results in **bold face** indicate best result(s) per column.

Trace	vector RMS error	*FoM*	first	second	speed bias	speed standard deviation	direction bias	direction standard deviation
0	3.29	1.00	0.40	0.65	**−0.21**	**1.80**	−0.27	28.27
1	3.47	0.95	0.42	0.67	−0.43	2.05	0.52	27.10
2	3.47	0.96	0.48	0.73	−0.58	2.15	−0.14	24.36
3	3.38	1.00	0.48	0.75	−0.52	2.09	−0.63	22.49
4	3.46	0.96	0.50	0.77	−0.50	2.27	**−0.09**	23.27
5	3.45	0.97	0.56	0.76	−0.46	2.21	1.55	22.61
6	3.46	0.99	**0.57**	0.76	−0.44	2.19	−0.72	21.73
7	3.46	1.01	0.51	0.77	−0.56	2.15	−0.83	20.59
8	3.41	1.02	0.52	0.78	−0.35	2.16	0.49	20.85
9	**3.04**	**1.12**	0.50	**0.82**	0.36	1.86	−0.89	**19.92**

Table G.11: Swathe model performance across traces for the YR1TRACE test data sets for the best conventional model based on the performance of the YR2SWATHE validation data set: 25 hidden units and four kernels. Results in **bold face** indicate best result(s) per column.

Trace	vector RMS error	*FoM*	first	second	speed bias	speed standard deviation	direction bias	direction standard deviation
0	**3.25**	1.02	0.39	0.65	**−0.17**	**1.86**	−0.59	26.14
1	3.49	0.93	0.44	0.67	−0.34	2.10	0.84	28.31
2	3.57	0.92	0.51	0.72	−0.54	2.18	−0.30	27.00
3	3.45	0.99	0.48	0.74	−0.47	2.14	−0.70	22.72
4	3.54	0.95	0.53	0.76	−0.47	2.33	**−0.09**	23.52
5	3.53	0.96	0.56	0.76	−0.46	2.23	0.51	23.17
6	3.56	0.97	**0.57**	0.76	−0.46	2.21	−0.85	22.23
7	3.50	1.00	0.54	0.75	−0.55	2.20	−0.66	20.89
8	3.50	1.01	0.51	0.77	−0.32	2.21	−0.31	20.71
9	**3.25**	**1.12**	0.48	**0.81**	0.28	1.95	−0.53	**17.97**

Table G.12: Swathe model performance across traces for the YR1TRACE test data sets for the best hybrid model based on the performance of the YR2SWATHE validation data set: 40 hidden units and four kernels. Results in **bold face** indicate best result(s) per column.

Appendix H

Proof of detailed balance for Tjelmeland's Algorithm

The aim is to construct a stationary Markov chain, with transition kernel $T(\cdot|\cdot)$, from $\pi(\cdot)$. The transition kernel is chosen such that

$$\int_A \pi(\boldsymbol{x})d\boldsymbol{x} = \int_{\mathbb{R}^n} T(A|\boldsymbol{x})\pi(\boldsymbol{x})d\boldsymbol{x}. \tag{H.1}$$

For detailed balance we require the chain to be time reversible. This implies that

$$\int_A \pi(\boldsymbol{x})T(B|\boldsymbol{x})d\boldsymbol{x} = \int_B \pi(\boldsymbol{x})T(A|\boldsymbol{x})d\boldsymbol{x}. \tag{H.2}$$

For two proposal distributions, the transition kernel is defined:

$$T(A|\boldsymbol{x}) = \frac{1}{2}\int_A q_0(\boldsymbol{y}|\boldsymbol{x})\alpha_{0,1}(\boldsymbol{y}|\boldsymbol{x})d\boldsymbol{y} + \frac{1}{2}\int_B q_1(\boldsymbol{y}|\boldsymbol{x})\alpha_{1,0}(\boldsymbol{y}|\boldsymbol{x})d\boldsymbol{y} + I(\boldsymbol{x}\in A)r(\boldsymbol{x}) \tag{H.3}$$

where

$$\alpha_{1,0}(\boldsymbol{y}|\boldsymbol{x}) = \min\left\{1, \frac{\pi(\boldsymbol{y})q_1(\boldsymbol{x}|\boldsymbol{y})}{\pi(\boldsymbol{x})q_0(\boldsymbol{y}|\boldsymbol{x})}\right\} \tag{H.4}$$

and

$$\alpha_{0,1}(\boldsymbol{y}|\boldsymbol{x}) = \min\left\{1, \frac{\pi(\boldsymbol{y})q_0(\boldsymbol{x}|\boldsymbol{y})}{\pi(\boldsymbol{x})q_1(\boldsymbol{y}|\boldsymbol{x})}\right\}. \tag{H.5}$$

The probability of not accepting a transition, that is staying in the same state is defined as:

$$r(\boldsymbol{x}) = \frac{1}{2}\int_{\mathbb{R}^n} q_0(\boldsymbol{y}|\boldsymbol{x})(1-\alpha_{0,1}(\boldsymbol{y}|\boldsymbol{x}))d\boldsymbol{y} + \frac{1}{2}\int_{\mathbb{R}^n} q_1(\boldsymbol{y}|\boldsymbol{x})(1-\alpha_{0,1}(\boldsymbol{y}|\boldsymbol{x}))d\boldsymbol{y}. \tag{H.6}$$

Therefore the full transition kernel becomes

$$\begin{aligned} T(A|\boldsymbol{x}) = {} & \frac{1}{2}\int_A q_0(\boldsymbol{y}|\boldsymbol{x})\min\left\{1, \frac{\pi(\boldsymbol{y})q_1(\boldsymbol{x}|\boldsymbol{y})}{\pi(\boldsymbol{x})q_0(\boldsymbol{y}|\boldsymbol{x})}\right\}d\boldsymbol{y} \\ & + \frac{1}{2}\int_A q_1(\boldsymbol{y}|\boldsymbol{x})\min\left\{1, \frac{\pi(\boldsymbol{y})q_0(\boldsymbol{x}|\boldsymbol{y})}{\pi(\boldsymbol{x})q_1(\boldsymbol{y}|\boldsymbol{x})}\right\}d\boldsymbol{y} \\ & + I(\boldsymbol{x}\in A)\left[\frac{1}{2}\int_{\mathbb{R}^n} q_0(\boldsymbol{y}|\boldsymbol{x})(1-\alpha_{0,1}(\boldsymbol{y}|\boldsymbol{x}))d\boldsymbol{y}\right. \\ & \left. + \frac{1}{2}\int_{\mathbb{R}^n} q_1(\boldsymbol{y}|\boldsymbol{x})(1-\alpha_{1,0}(\boldsymbol{y}|\boldsymbol{x}))d\boldsymbol{y}\right]. \end{aligned} \tag{H.7}$$

Practically there is no need to compute the rejection probability; by definition we know the acceptance probability and therefore the rejection probability.

Proof

Show

$$\int_A \pi(\boldsymbol{x})T(B|\boldsymbol{x})d\boldsymbol{x} = \int_B \pi(\boldsymbol{x})T(A|\boldsymbol{x})d\boldsymbol{x}. \tag{H.8}$$

Proof:

Substituting (H.7) into the LHS of (H.8)

$$LHS = \int_B \pi(\boldsymbol{x})T(A|\boldsymbol{x})d\boldsymbol{x} \tag{H.9}$$

$$LHS = \int_B \pi(\boldsymbol{x})\left(\frac{1}{2}\int_A q_0(\boldsymbol{y}|\boldsymbol{x})\min\left\{1, \frac{\pi(\boldsymbol{y})q_1(\boldsymbol{x}|\boldsymbol{y})}{\pi(\boldsymbol{x})q_0(\boldsymbol{y}|\boldsymbol{x})}\right\}d\boldsymbol{y} + \frac{1}{2}\int_A q_1(\boldsymbol{y}|\boldsymbol{x})\min\left\{1, \frac{\pi(\boldsymbol{y})q_0(\boldsymbol{x}|\boldsymbol{y})}{\pi(\boldsymbol{x})q_1(\boldsymbol{y}|\boldsymbol{x})}\right\}d\boldsymbol{y} \tag{H.10}$$

we rearrange (H.10)

$$LHS = \frac{1}{2}\int_B\int_A \pi(\boldsymbol{x})q_0(\boldsymbol{y}|\boldsymbol{x})\min\left\{1, \frac{\pi(\boldsymbol{y})q_1(\boldsymbol{x}|\boldsymbol{y})}{\pi(\boldsymbol{x})q_0(\boldsymbol{y}|\boldsymbol{x})}\right\}d\boldsymbol{y}d\boldsymbol{x} + \frac{1}{2}\int_B\int_A \pi(\boldsymbol{x})q_1(\boldsymbol{y}|\boldsymbol{x})\min\left\{1, \frac{\pi(\boldsymbol{y})q_0(\boldsymbol{x}|\boldsymbol{y})}{\pi(\boldsymbol{x})q_1(\boldsymbol{y}|\boldsymbol{x})}\right\}d\boldsymbol{y}d\boldsymbol{x} \tag{H.11}$$

$$= \frac{1}{2}\int_B\int_A \min\{\pi(\boldsymbol{y})q_1(\boldsymbol{x}|\boldsymbol{y}), \pi(\boldsymbol{x})q_0(\boldsymbol{y}|\boldsymbol{x})\}d\boldsymbol{y}d\boldsymbol{x} + \frac{1}{2}\int_B\int_A \min\{\pi(\boldsymbol{y})q_0(\boldsymbol{x}|\boldsymbol{y}), \pi(\boldsymbol{x})q_1(\boldsymbol{y}|\boldsymbol{x})\}d\boldsymbol{y}d\boldsymbol{x} \tag{H.12}$$

$$= \frac{1}{2}\int_B\int_A \pi(\boldsymbol{y})q_1(\boldsymbol{x}|\boldsymbol{y})\min\left\{1, \frac{\pi(\boldsymbol{x})q_0(\boldsymbol{y}|\boldsymbol{x})}{\pi(\boldsymbol{y})q_1(\boldsymbol{x}|\boldsymbol{y})}\right\}d\boldsymbol{y}d\boldsymbol{x} + \frac{1}{2}\int_B\int_A \pi(\boldsymbol{y})q_0(\boldsymbol{x}|\boldsymbol{y})\min\left\{1, \frac{\pi(\boldsymbol{x})q_1(\boldsymbol{y}|\boldsymbol{x})}{\pi(\boldsymbol{y})q_0(\boldsymbol{x}|\boldsymbol{y})}\right\}d\boldsymbol{y}d\boldsymbol{x} \tag{H.13}$$

$$= \frac{1}{2}\int_A \pi(\boldsymbol{y})\int_B q_1(\boldsymbol{x}|\boldsymbol{y})\min\left\{1, \frac{\pi(\boldsymbol{x})q_0(\boldsymbol{y}|\boldsymbol{x})}{\pi(\boldsymbol{y})q_1(\boldsymbol{x}|\boldsymbol{y})}\right\}d\boldsymbol{x}d\boldsymbol{y} + \frac{1}{2}\int_A \pi(\boldsymbol{y})\int_B q_0(\boldsymbol{x}|\boldsymbol{y})\min\left\{1, \frac{\pi(\boldsymbol{x})q_1(\boldsymbol{y}|\boldsymbol{x})}{\pi(\boldsymbol{y})q_0(\boldsymbol{x}|\boldsymbol{y})}\right\}d\boldsymbol{x}d\boldsymbol{y} \tag{H.14}$$

$$= \int_A \pi(\boldsymbol{y})\left[\frac{1}{2}\int_B q_1(\boldsymbol{x}|\boldsymbol{y})\alpha_{1,0}(\boldsymbol{x}|\boldsymbol{y})d\boldsymbol{x}\right]d\boldsymbol{y} + \int_A \pi(\boldsymbol{y})\left[\frac{1}{2}\int_B q_0(\boldsymbol{x}|\boldsymbol{y})\alpha_{0,1}(\boldsymbol{x}|\boldsymbol{y})d\boldsymbol{x}\right]d\boldsymbol{y} \tag{H.15}$$

$$= \int_A \pi(\boldsymbol{y})T(B|\boldsymbol{y})d\boldsymbol{y} \tag{H.16}$$

which can be expressed in terms of $\boldsymbol{x}$ by replacing the indexing variable $\boldsymbol{y}$ by $\boldsymbol{x}$:

$$= \int_A \pi(\boldsymbol{x}) T(B|\boldsymbol{x}) d\boldsymbol{x} \tag{H.17}$$

$$= RHS. \tag{H.18}$$

Appendix I

Convergence statistics results

	Energy	u_1	u_2	u_3	u_4	v_1	v_2	v_3	v_4
EPSR	1.001	1.003	1.003	1.001	1.002	1.004	1.003	1.003	1.003
W	4.002	3.317	3.257	3.342	3.493	102.118	102.749	103.593	104.450
$\widehat{var}(\psi)$	4.013	3.334	3.278	3.350	3.509	102.928	103.326	104.153	105.165

Table I.1: Divide and conquer convergence statistics: 50:50.

	Energy	u_1	u_2	u_3	u_4	v_1	v_2	v_3	v_4
EPSR	1.084	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
W	7.987	3.499	3.298	3.305	3.459	101.026	100.667	100.505	100.155
$\widehat{var}(\psi)$	9.392	3.498	3.297	3.304	3.458	101.034	100.673	100.513	100.178

Table I.2: Tempered transitions statistics: 50:50.

	Energy	u_1	u_2	u_3	u_4	v_1	v_2	v_3	v_4
EPSR	1.009	1.001	1.004	1.004	1.002	1.016	1.014	1.013	1.010
W	3.900	0.972	0.954	0.934	0.976	5.819	5.591	5.915	5.958
$\widehat{var}(\psi)$	3.975	0.974	0.964	0.942	0.981	6.007	5.749	6.075	6.088

Table I.3: Mode jumping proposals in MCMC convergence statistics: 50:50.

	Energy	u_1	u_2	u_3	u_4	v_1	v_2	v_3	v_4
EPSR	1.001	1.009	1.004	1.004	1.001	1.857	1.887	1.876	1.879
W	3.830	3.038	3.298	3.386	3.412	32.891	31.504	31.113	31.847
$\widehat{var}(\psi)$	3.841	3.095	3.328	3.414	3.419	113.486	112.266	109.519	112.519

Table I.4: Mode jumping proposals in MCMC convergence statistics: 50:50.

	Energy	u_1	u_2	u_3	u_4	v_1	v_2	v_3	v_4
EPSR	1.001	1.004	1.001	1.001	1.002	1.002	1.001	1.002	1.002
W	4.333	5.009	4.745	4.725	5.072	69.946	68.842	68.971	69.660
$\widehat{var}(\psi)$	4.345	5.049	4.755	4.732	5.092	70.215	69.054	69.218	69.977

Table I.5: Divide and conquer convergence statistics: 80:20.

	Energy	u_1	u_2	u_3	u_4	v_1	v_2	v_3	v_4
EPSR	1.007	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
W	5.173	4.740	4.699	4.702	4.714	74.381	74.173	74.139	74.234
$\widehat{var}(\psi)$	5.250	4.739	4.700	4.704	4.713	74.393	74.168	74.142	74.232

Table I.6: Tempered transitions statistics: 80:20.

	Energy	u_1	u_2	u_3	u_4	v_1	v_2	v_3	v_4
EPSR	1.0001	1.013	1.022	1.008	1.015	1.000	1.000	1.000	1.000
W	3.848	5.991	5.952	5.848	5.522	101.195	100.919	100.862	100.708
$\widehat{var}(\psi)$	3.854	6.156	6.227	5.951	5.691	101.232	100.952	100.904	100.766

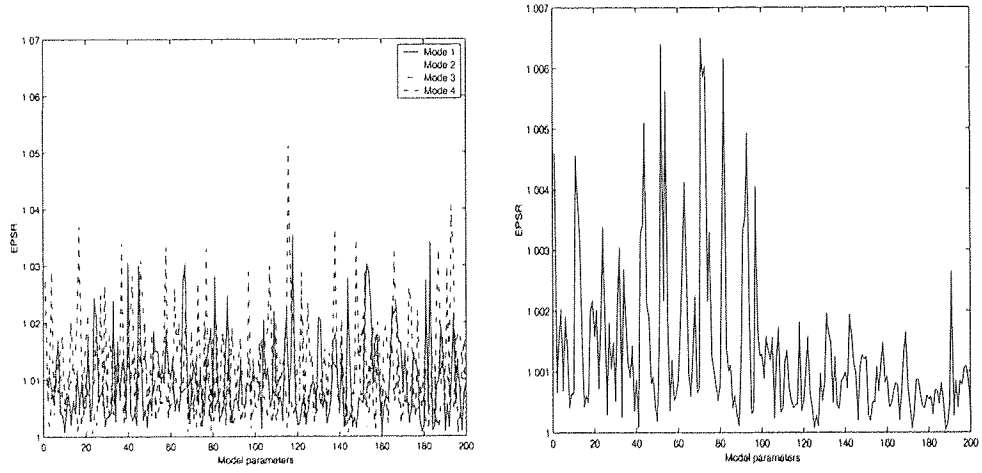
Table I.7: Divide and conquer convergence statistics, for equal covariance structures: 50:50.

	Energy	u_1	u_2	u_3	u_4	v_1	v_2	v_3	v_4
EPSR	1.002	1.007	1.004	1.003	1.009	1.000	1.000	1.000	1.000
W	4.189	6.296	5.725	5.964	5.751	65.332	65.420	65.487	65.362
$\widehat{var}(\psi)$	4.206	6.391	5.781	6.003	5.857	65.337	65.425	65.497	65.375

Table I.8: Divide and conquer convergence statistics, for equal covariance structures: 80:20.

Appendix J

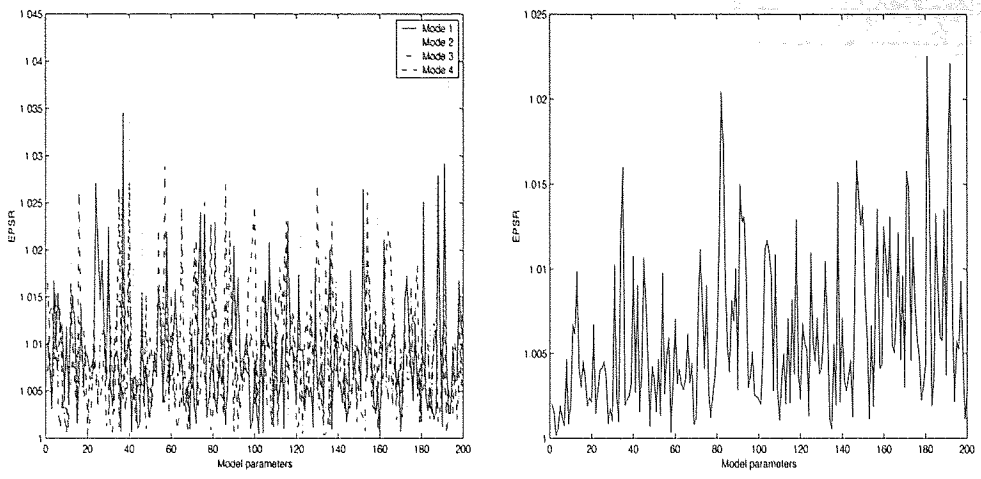
Convergence statistics for the case studies



(a) Convergence statistics for mode sampling

(b) Convergence statistics for mode jumping

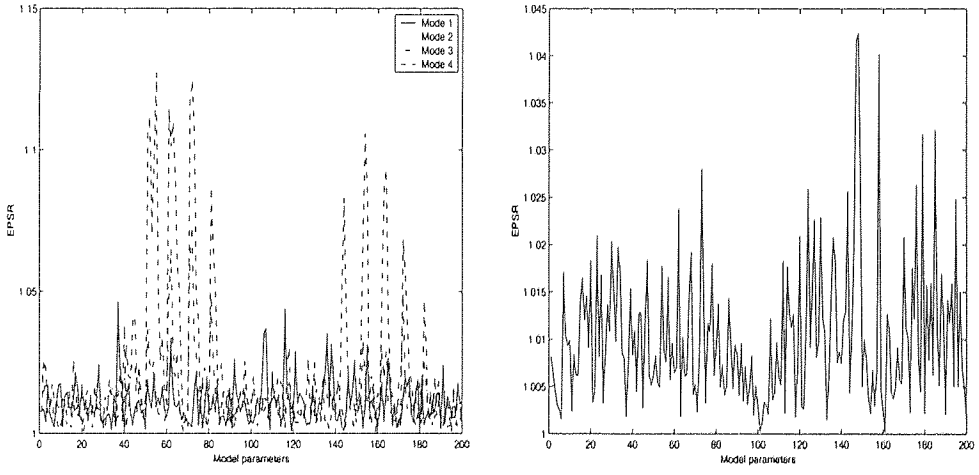
Figure J.1: Convergence statistics for case study 1. The EPSR value for the negative log-likelihood or free energy of the DC run (plot (b)) was 1.0003.



(a) Convergence statistics for mode sampling

(b) Convergence statistics for mode jumping

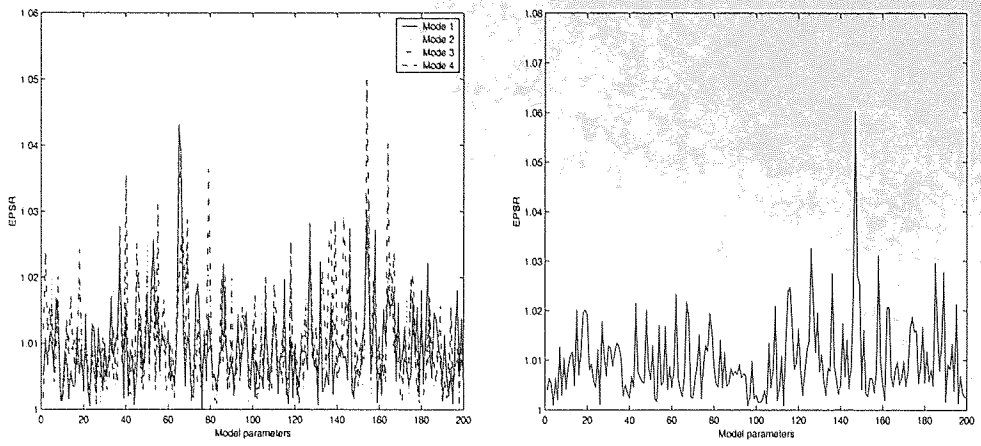
Figure J.2: Convergence statistics for case study 2. The EPSR value for the negative log–likelihood or free energy of the DC run (plot (b)) was 1.0012.



(a) Convergence statistics for mode sampling

(b) Convergence statistics for mode jumping

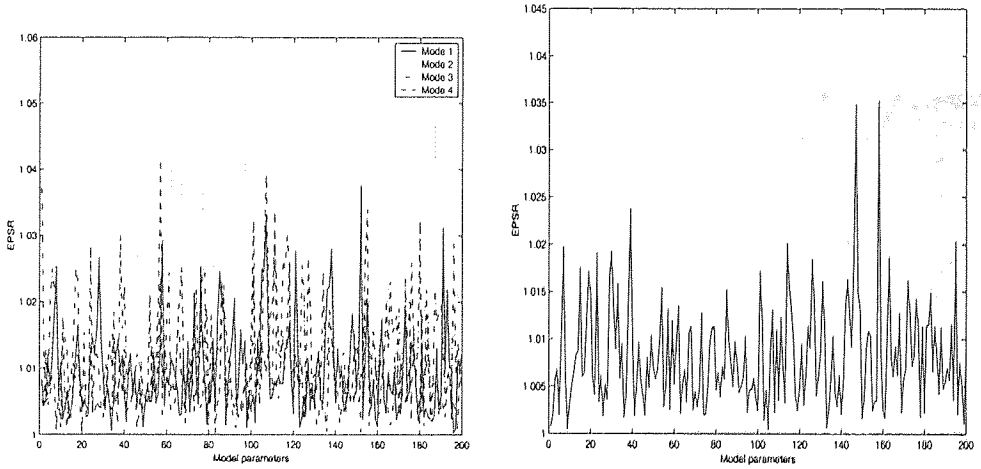
Figure J.3: Convergence statistics for case study 3. The EPSR value for the negative log–likelihood or free energy of the DC run (plot (b)) was 1.0006.



(a) Convergence statistics for mode sampling

(b) Convergence statistics for mode jumping

Figure J.4: Convergence statistics for case study 4. The EPSR value for the negative log-likelihood or free energy of the DC run (plot (b)) was 1.0045.



(a) Convergence statistics for mode sampling

(b) Convergence statistics for mode jumping

Figure J.5: Convergence statistics for case study 5. The EPSR value for the negative log-likelihood or free energy of the DC run (plot (b)) was 1.0020.