Aston University

Some parts of this thesis may have been removed for copyright restrictions.

If you have discovered material in AURA which is unlawful e.g. breaches copyright, (either yours or that of a third party) or any other law, including but not limited to those relating to patent, trademark, confidentiality, data protection, obscenity, defamation, libel, then please read our Takedown Policy and contact the service immediately

Pragmatic algorithms for implementing geostatistics with large datasets

BENJAMIN RANDALL INGRAM

Doctor of Philosophy

ASTON UNIVERSITY

March 2008

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without proper acknowledgement.

ASTON UNIVERSITY

Pragmatic algorithms for implementing geostatistics with large datasets

BENJAMIN RANDALL INGRAM

Doctor of Philosophy, 2008

Thesis Summary

With the ability to collect and store increasingly large datasets on modern computers comes the need to be able to process the data in a way that can be useful to a Geostatistician or application scientist. Although the storage requirements only scale linearly with the number of observations in the dataset, the computational complexity in terms of memory and speed, scale quadratically and cubically respectively for likelihood–based Geostatistics. Various methods have been proposed and are extensively used in an attempt to overcome these complexity issues. This thesis introduces a number of principled techniques for treating large datasets with an emphasis on three main areas: reduced complexity covariance matrices, sparsity in the covariance matrix and parallel algorithms for distributed computation. These techniques are presented individually, but it is also shown how they can be combined to produce techniques for further improving computational efficiency.

Keywords: Geostatistics, Kriging, Gaussian processes, Large datasets, Sparsity, Parallel computation, Sample design configuration

to me mam

Acknowledgements

Firstly, I would like to thank my supervisors Dr. David Evans and Dr. Dan Cornford for the inspiration that they both provided. Their interest, energy, encouragement and boundless patience through the completion of thesis was invaluable and much appreciated.

I give thanks also to those within the Neural Computing Research Group and Knowledge Engineering Group at Aston University who provide a stimulating research environment.

Also, I am grateful to my fellow PhD students for encouragement and long discussions.

Finally, I would like to express immense gratitude to my family for everything they have done to help, particularly during the challenging times.

Contents

List of Figures

List of Tables

Declaration

This thesis describes the work carried out between January 2004 and January 2008 in the Neural Computing Research Group at Aston University under the supervision of Dr. David Evans and Dr. Dan Cornford.

This thesis has been composed by myself and has not, nor any similar dissertation, been submitted in any previous application for a degree.

1 Thesis Outline

1.1 Introduction

The influence of computers in the world is growing rapidly. Computation speed and storage capacities are growing every year and at an ever increasing pace. Equally, user expectations are also rising. Vast quantities of information can be captured, processed and stored automatically. Many practitioners today work with large datasets which were not possible to analyse in an acceptable time, even relatively recently. Nevertheless, as computation speed increases practitioners try to analyse still larger datasets and encounter similar problems with unacceptable computation times. Such extensive datasets are frequently encountered by environmental scientists (Cressie et al. 1997).

In this thesis, the issues associated with the application of geostatistics to large datasets are addressed. A range of possibilities are explored to give a practitioner the flexibility to choose the techniques appropriate for addressing a specific problem. The aim of this thesis is to provide

accessible geostatistical techniques for use with large datasets.

In what follows, the main themes of the contributions that this thesis makes are introduced. Before any technical details are considered, the motivations for this work are discussed. Following the discussion of the background of the techniques that are used throughout this thesis, the contributions that this thesis makes are listed along with publications and presentations delivered during the completion of this work.

1.2 Large datasets

The problem of large datasets was once considered a solved issue (Schabenberger and Gotway 2005). By using method–of–moments variograms and moving window kriging, all but the very massive dataset are computationally tractable. In recent years the popularity of and interest in maximum likelihood–based algorithms has grown and such algorithms give rise to computational problems when more than a few thousand observations are encountered.

Traditional geostatistics can be divided into two main activities. The first step, usually referred to as variogram estimation, involves determining the covariance structure of the dataset being analysed. Secondly, once the covariance model and parameters have been selected, prediction is performed. Depending on the methods used, both stages can potentially be computationally intensive, hence both parameter estimation and prediction are addressed throughout this thesis. A number of techniques to treat large datasets have already been proposed and have been extensively used to avoid the large matrix inversion problems. However, as is discussed in Chapter 2, these techniques have a number of limitations. When likelihood–based parameter estimation techniques are used, the need to invert an $(n \times n)$ covariance matrix, $\mathbf{\Sigma}$, of all the observations (where n is the number of observations) cannot be avoided. Furthermore, Bayesian or model–based approaches (Diggle and Ribeiro Jr. 2007) that explicitly assume a probabilistic model also require an inversion of the covariance matrix, $\mathbf{\Sigma}$.

Many datasets that geostatisticians analyse are typically of the order of just a few hundred observations since it can be expensive to collect more observations, or an exhaustive dataset may not be available. Geostatistics can be performed efficiently with such small datasets. With the recent increase in the size of datasets, mainly due to the large number of satellite–based sensors, sampling potentially vast areas across the globe (eg. ERS (Offiler 1987; Andrews and Bell 1998)),

aerial photography, large monitoring networks (eg. EURDEP[1]) or large repositories of data accessible from online sources (eg. Atmospheric Radiation Measurement Program[2]), it is not uncommon that the number of observations can run into the millions. Performing geostatistics directly on large datasets of more than a few thousand observations becomes prohibitive.

Treating large datasets is not the only reason to research techniques that offer increased computational efficiency. For example, there is a need for efficient algorithms for use in a real–time mapping context (Williams et al. 2007; Ingram et al. 2008). In particular a *network monitoring* scenario could be considered whereby observations are collected and reported by a network of sensors and are then transformed into maps which decision makers can use for assessing a situation. As noted by Galmarini (2005), there still exist significant problems if the results are required in (near) real–time, as would be the case with many automatic monitoring networks when an emergency situation arises. Also, onboard systems found in some sensor devices (eg. those mounted on a satellite) tend to have limited processing power due to power consumption constraints. Hence the need to develop faster, more efficient algorithms.

There are a large number of solutions for overcoming the matrix inversion problem and as such it is rarely considered to be of serious consequence. As noted by Galmarini (2005), there still exist significant problems if the results are required in (near) real–time, as would be the case with many automatic monitoring networks. Also, the application of many of these solutions to the large matrix inversion problem often requires significant human intervention and would not be appropriate for use in an automatic mapping context. Hence there is a need to return to the question of how to treat large datasets in a likelihood–based framework in the automatic mapping context.

In this work a number of algorithms will be added to the existing techniques to give the geostatistician more options for their analysis of large datasets.

1.3 Machine Learning

Machine learning techniques have been researched for many years (Bishop 1995; Ripley 1996). The main focus is to generate models using computational and statistical techniques using a

[1] http://eurdep.jrt.it
[2] http://www.arm.gov

training dataset. Once a model for a specific dataset has been generated it can be used to make predictions, classifications or decisions depending on the purpose of the algorithm.

In the machine learning community, the Gaussian process framework has received increasing attention, particularly for the treatment of large datasets (Quiñonero-Candela and Rasmussen 2005). The basis of these types of techniques is to represent the full model by an approximation. By minimising any significant loss of accuracy, an approximation can be made. These types of approximations are generally referred to as *sparse* approximations, but this can be misleading since the assumption, often mistakenly made, is that the matrices involved are sparse. The reality is that the sparsity involved in these approximations induces low–rank matrices rather than sparse matrices. To avoid confusion and following the naming conventions of Rasmussen and Williams (2006), the term *projected process* will be used to refer to sparsity in models that use low–rank matrix approximations. To complicate matters somewhat, the concept of sparsity in matrices is discussed in Chapter 4 of this thesis and then combined with the *projected process* techniques to provide a very compact representation of the model.

These *projected process* techniques have gained modest popularity and are beginning to be used for many types of applications. As of yet, the wide spread application of *projected process* techniques by geostatisticians is extremely limited. This is largely a consequence of the presentation of the techniques. Chapter 2 presents these *projected process* methods in a context familiar to geostatisticians in the hope that the methods will be more accessible to geostatisticians. Many of these techniques are related to parallel developments in geostatistics which are highlighted.

1.4 Covariance models

Many of the datasets analysed by machine learning algorithms contain observations with high dimensionality in the inputs. It would not be uncommon for a dataset to have over one hundred input dimensions. The majority of the datasets analysed in geostatistics are commonly two or three dimensional in terms of their inputs. The covariance function is used to encode beliefs about the covariance structure of the dataset. It is fairly common practise to find the Gaussian or squared exponential covariance function used in machine learning algorithms since it is valid for use with high dimensional datasets, that is to say that it generates a positive definite covariance

matrix for all dimensions. A Gaussian covariance function is not always appropriate for use in geostatistics since it can be considered unrealistically smooth for many physical processes (Stein 1999). Some of the covariance functions used in geostatistics are not valid when applied to high dimensional problems. Hence there has been little research in the numerical and computational performance of alternative covariance functions in *projected process* techniques. Further to this, the advantages of space–limited covariance functions are considered and coupled with the *projected process* techniques to provide an universal method for representing a process in a compact form by identifying the redundancy in both large–scale and small–scale variation of the model.

1.5 Parallel computation

The early standardisation of a single machine computer model over sixty years ago (Neumann 1945), the von Neumann architecture, was one of the reasons for the rapid rise in the use of computers in business, science and education (Foster 1995). The von Neumann architecture comprises of a single processor that is connected to a data storage unit which is used to store a computer program and data. Computer program instructions that are stored in memory are fetched, decoded and executed sequentially. The standardisation of this computer model allowed programmers to focus on abstract algorithm design rather than having to design algorithms for specific computer architectures (Cobb 2000).

Multiple processors (or processor cores) in desktop computers are becoming commonplace. Software designed for von Neumann architectures is effectively limited to the speed of the fastest processor element in a parallel architecture. Changing the program language paradigms prevalent throughout the world is not a trivial task (Backus 1978). Programming parallel architectures is also notoriously difficult due to the complexities of concurrency, resource allocation performance and the diversity of differing parallel machine architectures (Darlington et al. 1993). Productivity in developing parallel software is a challenge being faced by many developers in the sense that the programming paradigms are more complex.

To utilise parallel architectures, the parallelisms in the underlying models have to be identified. To fully utilise modern commodity hardware, the development of parallel algorithms is essential otherwise a large proportion of the processing power of the computer will remain

under–used when performing computationally intensive calculations. The world of geostatistics would certainly benefit from the wide spread use of software that could exploit parallel architectures. The available literature for parallel geostatistics is scanty. Coupled with the other ideas presented in this thesis, algorithm parallelism is another technique that should be used to improve computational speed.

1.6 Sample design configuration

Having introduced some of the techniques presented in this thesis for analysis of data that has already been collected, attention is now turned to the topic of collecting data. The problem of where to locate sample points for data collection is often based on intuition or expert knowledge about the process being observed. Selecting a sampling scheme that is too coarse will result in loss of information. As the sampling scheme granuality increases, the cost of data collection increases. The techniques that will be presented later in this thesis for treating large datasets are related to techniques that can be used for sampling design optimisation. Here we look at the specific case of soil data where ancillary data is available.

In soil science in general, determining an optimal sampling configuration for a soil variable of interest can be informed by using ancillary data such as an aerial photograph. Selecting the most informative sampling locations can save both time and money for precision agriculture. Ancillary data has been shown to be effective for informing analysis because it tends to vary in similar ways to soil data (Kerry and Oliver 2003), although the relationships are complex and difficult to interpret. The ancillary data is analysed so that the covariance structure can be determined. The sampling configuration can then be determined by the process covariance parameters obtained from the ancillary data.

A drawback of using ancillary data is that it tends to be densely sampled which prohibits likelihood–based covariance parameter estimation. Another problem occurs even when the covariance structure of the soil data is estimated well from the aerial photograph. Selecting potential sampling locations based solely on the covariance structure can still miss important features in the data that would be captured had changes measured in the locally varying mean also been analysed.

The projected process framework can be utilised to enable the application of maximum

likelihood parameter estimation and also for sampling design configuration which takes into account the locally varying mean.

1.7 Contributions

The contributions that this thesis makes to scientific understanding are:

- Integration of methods for treating large datasets from machine learning for use in geostatistics.
- Application of space–limited covariance functions together with projected process approximations to provide an universal technique for treating large datasets.
- Parallel algorithms for treating large datasets.
- Sequential optimal sampling design configuration informed by ancillary data.

1.8 Thesis structure

This thesis is organised into the following chapters:

Chapter 1 is this introductory chapter.

Chapter 2 introduces the theory of the spatial interpolation methods known as kriging and Gaussian processes. Bayesian methodologies for spatial interpolation are also reviewed. Links to the techniques being developed in the machine learning community for treating large datasets are made and these techniques are presented in a geostatistical framework. The issues associated with commonly used geostatistical techniques for treating large datasets are reviewed. Alternative algorithms are presented which provide ways of treating large datasets.

Chapter 3 explains the datasets that will be used throughout the later chapters of this thesis. To provide consistency in the benchmarking of algorithms and gauging performance, three datasets will be used with each of the techniques introduced. Two of the datasets are synthetic and one is a real world example.

Chapter 4 extends the *projected process* methods already discussed by showing how space–limited covariance functions can be used to achieve computational advantages. Constructing valid covariance functions for spatial interpolation is reviewed and the computational advantages are demonstrated. The application of space–limited covariance functions is shown to provide further possibilities for increasing computational efficiency. Through the application of space–limited covariance functions coupled with the *projected process* methods discussed in Chapter 2, an universal approximation method created.

Chapter 5 familiarises the reader with parallel computing and current trends in parallel software. The basic principles of parallel computing are reviewed and current technologies are explained with particular relevance to mathematical problems. The main barriers to utilising such methods are discussed.

Chapter 6 develops on the contributions of the earlier chapters by showing how these models can exploit parallel architectures to achieve further speed up. A review of techniques currently utilised in a range of applications is presented. Parallel algorithms for treating large spatial datasets are introduced and benchmarked to show what range of speed ups can be obtained by their adoption.

Chapter 7 shows how sequential *projected process* techniques can be applied to optimal sampling design where ancillary data is available. Specific examples using data collected from two fields in the south of England are used to illustrate this method. By using a sequential algorithm, the efficiency of determining sampling locations is increased.

Chapter 8 concludes this thesis with a summary of the achievements of the techniques presented. Important considerations are discussed with a view to practical use in real world applications. Questions about future work are raised.

Some calculation details have been put in appendices to maintain the flow of the main ideas in this thesis.

1.9 Symbols and abbreviations

Bold lowercase letters are used to denote vectors and bold uppercase denote matrices. Scalar quantities will be typeset in normal print, for example as the particular elements of a vector or matrix, hence a vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ is a n–dimensional vector with the corresponding components.

Summary of the notation used in the thesis:

$\mathbf{x}$ – a spatial location or inputs from a d-dimensional space.

$\mathbf{y}$ – an observation or the output corresponding to a given input $\mathbf{x}$, it can be continuous or discrete.

$\mathcal{D}_N = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ – the dataset of observations.

n – the size of the dataset of locations and observations or *training* dataset.

m – the size of the active set.

t – the size of the prediction locations or *test* dataset.

$\mathbf{x}_N$ – the locations of the dataset.

$\mathbf{x}_M$ – the locations of the active points selected for the active set.

$\mathbf{x}_T$ – the prediction locations.

$Z(\cdot)$ – the observations given an location.

$Z(\mathbf{x}_N)$ – the observations at the locations in the dataset.

$Z(\mathbf{y}_M)$ – the observations of the dataset included in the active set.

$\mathbf{k}_{AB} = k(\mathbf{x}_A, \mathbf{x}_B) = \text{cov}(\mathbf{x}_A, \mathbf{x}_B)$ – the covariance between the locations spatial locations $\mathbf{x}_A$ and $\mathbf{x}_B$.

$\boldsymbol{\theta}$ – model parameters.

$P(\mathcal{D}|\boldsymbol{\theta})$ – is the likelihood of the dataset $\mathcal{D}$ given the model parameters $\boldsymbol{\theta}$.

$\mathcal{E}\{w\}$ – the expected value of w.

$\gamma(\|\mathbf{x}_A - \mathbf{x}_B\|)$ – the semivariogram between the locations $\mathbf{x}_A$ and $\mathbf{x}_B$.

$f(\mathbf{x})$ – the value of the random function at $\mathbf{x}$.

$\boldsymbol{\Sigma}$ – the covariance matrix.

$\boldsymbol{\Sigma}^{-1}$ – the inverse covariance matrix.

$\mathbf{diag}(\boldsymbol{W})$ – only preserve the diagonal elements of the matrix $\boldsymbol{W}$.

$\mathbf{blockdiag}(\boldsymbol{W})$ – only preserve the block diagonal elements of the matrix $\boldsymbol{W}$.

$\mathbf{nnz}(\boldsymbol{A})$ – the number of nonzero elements in $\boldsymbol{A}$.

$\langle \boldsymbol{A}, \boldsymbol{B} \rangle_F$ – the Frobenius norm between the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.

2

Algorithms for large scale datasets

2.1 Introduction

The machine learning community has seen an explosive interest in Gaussian process methods during the last decade (Rasmussen and Williams 2006). These methods have been known as kriging by geostatisticians for many years (Williams and Rasmussen 1996), and are used for addressing problems within the spatial prediction domain (Matheron 1963). In the 1960's, geostatistical techniques were largely developed independently of mainstream spatial statistics, having their own unique presentation. This led to connections between parallel developments within spatial statistics being unclear since the relationship between geostatistics and spatial statistics was unclear. An example of these parallel developments is that of kriging, which is equivalent to the minimum mean square error prediction under a linear Gaussian model (Diggle et al. 1998). It was not until the early 1980's that Ripley (1981) made this connection explicit. Over a decade later, Cressie (1993) presented geostatistics as one of the three main branches of

spatial statistics.

The purpose of this chapter is to present a unifying view of geostatistics and machine learning methods for treating large datasets. There have been many independent developments in both areas of study, and typically many of these developments are identical in principle. However, had a clear unifying framework been adopted, repeated research may not need to have been done.

In this chapter, the links between the methods from machine learning and geostatistics will be made. Subsequent developments in either field will be presented with terminology familiar to a geostatistician. Additionally, the problem of treating large datasets in a statistically principled manner will be discussed offering a number of new approaches.

2.1.1 Spatial Interpolation

Spatial interpolation encompasses a large number of techniques that are used for prediction of attributes at spatial locations where a variable has not been observed. For example, these techniques include: nearest neighbour, inverse distance weighting and kriging. The modelling process requires that a model is constructed of how a given process behaves at locations where a variable has not been observed. Selecting the underlying model is something that requires skill, judgement and experience and should be informed by prior knowledge about the nature of the process that is being predicted. Analysing the observations solely is often insufficient to determine how a process should be modelled. Additional knowledge about the variable is important and if it is available it should be used so that assumptions can be made to inform the model. In determining a suitable model, the physical processes that generated the observations are analysed. Ideally, understanding the process generating the variable being measured sufficiently well to create a deterministic model would be desirable. However, few processes that are observed in geosciences are understood sufficiently well to create accurate deterministic models of the complex interactions that take place with other processes across different scales. Therefore, probablistic or random field models are used and it is acknowledged from the outset that there will be uncertainty in understanding the properties of a process.

Probabilistic models often view the available observations as the result of a random process. Although this apparent randomness is evident, it is important to note that this does not mean

that the process is random, it means however, that the understanding of the process is limited. The question of which model is best is one that cannot be answered theoretically since this depends on the true spatial structure and this is unknown, although often prior evidence can be available to help inform more feasible models. The approach that is selected should correspond with what is known about the process that generated the spatial structure.

2.2 Kriging methodology

In the domain of spatial interpolation one family of techniques that is commonly used for prediction is called kriging (Cressie 1993). Kriging is a term generally used to describe a family of methods based on the theory of random processes for computing the minimum variance estimator. The estimation variance is optimal if the chosen covariance model is correct for the process that generated the observations. It is not the purpose of this thesis to give a thorough review of kriging, but rather to introduce the concepts that underpin kriging and then it will be shown how these can be generalised.

It is assumed that $Z(\mathbf{x})$ is a random spatial process with the covariance function $k(\cdot)$ where the process $Z(\mathbf{x})$ is known only at n spatial locations $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. The vector of available data is defined as:

$$\mathbf{Z} \equiv (Z(\mathbf{x}_1), \ldots, Z(\mathbf{x}_n)). \tag{2.1}$$

There are a number of kriging variants depending on what assumptions are made about what is known about the process being investigated. First, the kriging model will be defined as:

$$Z(\mathbf{x}) = m(\mathbf{x})^{\mathsf{T}} \boldsymbol{\beta} + \epsilon(\mathbf{x}), \tag{2.2}$$

where m is a deterministic trend component with parameter vector $\boldsymbol{\beta}$ for the trend and $\epsilon(\mathbf{x})$ is the random process given by:

$$\epsilon(\mathbf{x}) = \mathcal{N}(0, \boldsymbol{\Sigma}), \tag{2.3}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of the observations.

The most simple form of kriging is called *Simple kriging*. Simple kriging assumes that the mean or trend function takes the form

$$m(\mathbf{x}) = 0, \tag{2.4}$$

or that there is a known constant mean or trend.

If the mean of the trend is assumed to be constant, but the mean is unknown, then *Ordinary kriging* can be used to estimate the mean of the constant trend. Ordinary kriging assumes

$$m(\mathbf{x}) = 1, \tag{2.5}$$

and an unknown constant, β, is estimated to obtain mean parameter value.

If the constant trend model is too restrictive an assumption to describe the trend, a general linear trend can be assumed where the trend is modelled by a polynomial of a specified order. In this context, a more general trend function is used. The mean function parameters, $\boldsymbol{\beta}$, are estimated by a generalized least squares estimator (Stein 1999).

Simple kriging is an optimal spatial predictor in that it minimises the mean-squared prediction error. As discussed previously, to guarantee that the kriging predictor is optimal, the assumption has to be made that Z is a stationary process and that $Z(\mathbf{x})$ is zero mean, or that the mean function has already been removed from the dataset. Note that in this thesis, without loss of generality, the emphasis will be on simple kriging algorithms; for in depth discussion of other forms of kriging the reader is directed to Cressie (1993).

The best linear unbiased predictor at an unobserved location $\mathbf{x}_*$ is given by:

$$\hat{Z}(\mathbf{x}_*) = \sum_{i=1}^{n} \lambda_i Z(\mathbf{x}_i). \tag{2.6}$$

The predictor is simply a weighted sum of all the observations $Z(\mathbf{x}_i)$ in the dataset $\mathcal{D}_N$ (Webster and Oliver 2000). Each weight λ_i is the corresponding weight for the observation $Z(\mathbf{x}_i)$. The weights are calculated by some function of the distance between each observation location and the prediction location (Isaaks and Srivastava 1989). This scaled distance is based on choosing a covariance model which describes the variation in the process. The covariance function will be referred to as $k(\mathbf{x}_a, \mathbf{x}_b)$ where $\mathbf{x}_a$ and $\mathbf{x}_b$ are spatial locations and the covariance function returns the covariance between the two locations (or vectors of locations). A short hand notation will be used to refer to the covariance; the vector $\mathbf{k}_{ab}$ will denote the covariance $k(\mathbf{x}_a, \mathbf{x}_b)$.

To calculate the kriging weights at the prediction location $\mathbf{x}_*$, the covariance matrix of all the observations, $\boldsymbol{\Sigma}$, is inverted and multiplied by a vector of covariances between the prediction location and the data locations to give the weight vector $\boldsymbol{\lambda}$ as shown by:

$$\boldsymbol{\lambda} = \boldsymbol{\Sigma}^{-1}\mathbf{k}_{N*} \tag{2.7}$$

where Σ is the square $n \times n$ matrix of the covariance between each of the observations given by:

$$\Sigma = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \dots & k(x_n, x_n) \end{bmatrix} \tag{2.8}$$

and $\mathbf{k}_{N*}$ is the covariances between the observation locations $\mathbf{x}_N$ and the prediction location x_*:

$$\mathbf{k}_{N*} = \begin{bmatrix} k(x_1, x_*) \\ \vdots \\ k(x_n, x_*) \end{bmatrix}. \tag{2.9}$$

Substituting Equation (2.7) into Equation (2.6), the equation for predictive mean of the process at location $\mathbf{x}_*$ is:

$$\hat{Z}(x_*) = \mathbf{k}_{N*}^T \Sigma^{-1} Z(\mathbf{x}_N), \tag{2.10}$$

and the predictive variance is:

$$\sigma_*^2(x_*) = k_{**} - \mathbf{k}_{N*}^T \Sigma^{-1} \mathbf{k}_{N*}, \tag{2.11}$$

where

$$k_{**} = k(x_*, x_*) \tag{2.12}$$

is the total sill variance of the process or process variance.

The complexity of solving the linear system $\mathbf{b} = \Sigma^{-1}\mathbf{Z}$ directly in Equations (2.10) and (2.11) is $\mathcal{O}(n^3)$ in computation and $\mathcal{O}(n^2)$ in storage. This basically prohibits straightforward kriging for large datasets of more than a few thousand, and also raises issues for smaller datasets that need to be treated in (near) real-time. Furthermore, in an automatic mapping system the parameters of the process need to be estimated without human intervention. In a setting where maximum likelihood approaches are used to estimate the covariance parameters the need to invert the covariance matrix several times (often hundreds of times) cannot be avoided.

It can be seen that the main bottleneck in the kriging algorithm is the need to invert the covariance matrix, Σ, of all the observations. For large datasets, problems will arise and hence there has been a large amount of research into potential solutions to solve these computational problems. Many of the solutions are very simple to implement but at a cost; they are ad-hoc in nature or lack a sound statistical basis (Cornford et al. 2005). Many of the more statistically

principled approaches tend to be mathematically intense and this can cause practitioners to look elsewhere for solutions to these problems since the understanding of the principles of such methods is often non–trivial.

The basics of kriging have been explained in a manner requiring very little mathematics and in doing so it is hoped that the algorithms developed and described later in this chapter can more easily be understood and hence be more appealing to many practitioners.

2.2.1 Gaussian processes

A Gaussian process is a stochastic process where every joint density function is Gaussian and is therefore defined completely by its mean and covariance (Rasmussen and Williams 2006). Gaussian processes are equivalent to kriging under the assumption of a multigaussian distribution for the variable of interest. There are a number of ways to interpret Gaussian process regression models. Geostatistians will feel particularly comfortable with the *weight–space view* of Gaussian processes since this is the perspective in which kriging is usually presented. For some of the methods that will be explained later in this chapter, an understanding of the *function–space view* will give the reader an intuitive understanding since some of the methods explained are better posed in this view. These two perspectives are equivalent and yield identical results (Williams and Rasmussen 1996).

Bayesian methods

Although important, here a philosophical debate will be avoided of the merits of a Bayesian approach over a frequentist view of the world. For most applications, it is unrealistic to believe that one can understand the observed data and all the processes that generated it in sufficient detail to permit a deterministic approach to prediction. The observed data are generally the result of complex interactions of many processes of which it is unlikely that one has a clear understanding. Therefore, prior knowledge about the observed data can be used. This can be represented using a probability distribution where a model parameter is encoded, for example, by a mean and variance for a Gaussian distribution.

The second centred moment (or variance for a Gaussian) of the distribution encodes the precision or confidence about the accuracy of the observation as it relates to the process that is

to be modelled. Normal or Gaussian distributions are used more commonly, since if very little is known about the actual distribution shape a Gaussian distribution often seems like a best guess, and the maths is more tractable. The Central Limit Theorem states that if the sum of the variables has a finite variance, then it will have an approximately Gaussian distribution (Tijms 2004). Bayesian methods propagate the uncertainty through the model from which a posterior distribution can be obtained which describes uncertainty in the model parameters. Omre (1987) gives a thorough review of kriging in a Bayesian framework.

It is common practice within the field of geostatistics that explicit stochastic models are rarely declared and as a result little use is made of the likelihood-based methods of inference which are central to modern statistics (Diggle et al. 1998). Diggle et al. (1998) have used the phrase *model–based geostatistics* to describe an approach to geostatistical problems based on using formal statistical methods under an explicitly assumed stochastic model. For all the methods discussed in this thesis a model–based Bayesian approach is encouraged although each method is introduced without the additional mathematical complexity that a model–based framework can sometimes bring. Many statistically principled extensions to kriging models have been proposed but have yet to find common place in the geostatistical community due in part to the additional complexity that they require. For example, Heuvelink et al. (2006) state that the techniques presented by Diggle and Ribeiro Jr. (2007) would provide an elegant solution, but their application is not easy. Diggle and Ribeiro Jr. (2007) applies Monte Carlo based methods for numerically calculating non–tractable integrals. Appendix B reviews an alternative approach of Csató and Opper (2001b) for approximating non–Gaussian posterior distributions.

The Bayesian approach to modelling is an attempt to utilise all available information in order to create a realistic model. In doing so, prior knowledge such as experience, expert knowledge or previous datasets can all be taken into account. Bayes' theorem gives a mathematical rule to update existing prior beliefs given new observed evidence. The process of using this additional information to draw conclusions about situations not previously met is called inference. Inference is a common part of everyday life and many things are inferred perhaps without even realizing it. Bayes' theorem is given by:

$$\mathbf{p}(\mathbf{f}|\mathcal{D}) = \frac{\mathbf{p}(\mathcal{D}|\mathbf{f})\ \mathbf{p}(\mathbf{f})}{\mathbf{p}(\mathcal{D})} \tag{2.13}$$

which can be seen as a principled way of combining data-based information via the likelihood

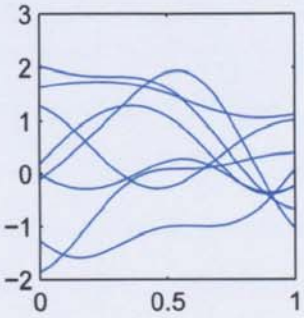


Figure 2.1: Samples from the Gaussian process prior with a Gaussian covariance function.

function with a prior distribution chosen by the practitioner.

The starting point for Gaussian processes is to define a finite set of functions:

$$\mathbf{f}_\chi = \{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\} \tag{2.14}$$

which are indexed by corresponding inputs:

$$\chi = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}. \tag{2.15}$$

The prior Gaussian process is defined as the joint distribution of the random variables, $\mathbf{f}_\chi$ and is given by:

$$p(\mathbf{f}_\chi) = \mathcal{N}(0, \boldsymbol{\Sigma}). \tag{2.16}$$

The prior distribution encodes prior beliefs about the function being modelled. Samples can be drawn from the Gaussian process prior distribution to show possible paths the functions can take. Figure 2.1 shows some randomly selected sample paths. The properties of the sample paths are similar in nature. It can be seen that the covariance function used results in smooth sample paths.

The likelihood is assumed to be factorised because it is assumed that the data is conditionally identically independently distributed on knowing the actual function value. Assuming the observation noise is given by σ^2, then the likelihood can be written as:

$$p(Z(\mathbf{x})|\mathbf{f}_\chi) = \mathcal{N}(\mathbf{f}_\chi, \sigma^2\mathbf{I}) \tag{2.17}$$

which for a Gaussian likelihood gives:

$$p(Z(\mathbf{x})|\mathbf{f}_\chi) = \prod_{i=1}^{N} p(Z(\mathbf{x})|\mathbf{f}_\chi) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{Z(\mathbf{x}_i) - \mathbf{f}_\chi(x_i)}{2\sigma^2}\right). \tag{2.18}$$

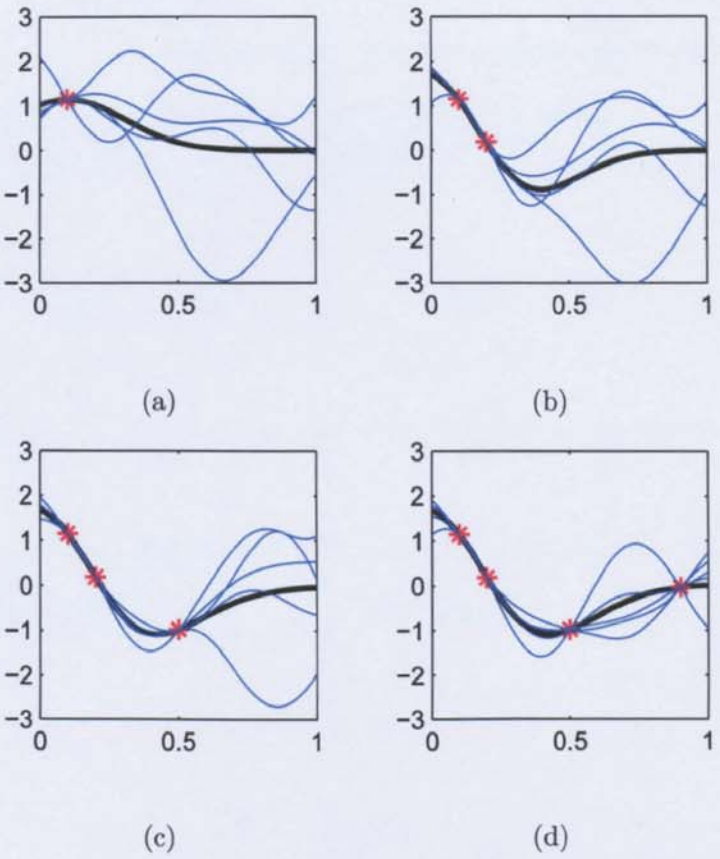


Figure 2.2: Samples from the posterior Gaussian process after (a) 1, (b) 2, (c) 3 and (d) 4 observations have been selected. The thick line in each plot represents the most plausible model for the data. The thin blue lines in each plot show possible models for the data given the parameters and observations. Each observation is indicated by a star.

By integrating over the unobserved function variances $\mathbf{f}$, the marginal likelihood can be obtained:

$$\mathrm{p}\left(Z(\mathbf{x})\right) = \int \mathrm{d}\mathbf{f}\mathrm{p}\left(Z(\mathbf{x})|\mathbf{f}_{\mathbf{x}}\right)\mathrm{p}\left(\mathbf{f}_{\mathbf{x}}\right) = \mathcal{N}\left(0, \boldsymbol{\Sigma} + \sigma^2\mathbf{I}\right) \tag{2.19}$$

Since both the prior distribution and the likelihood are Gaussian, Bayes' rule can be applied to obtain the posterior distribution by using the identity for the product of two Gaussians in Appendix A.6. The posterior distribution is given by:

$$\begin{aligned} \mathrm{p}\left(\mathbf{f}_{\mathbf{x}}|Z(\mathbf{x})\right) &= \frac{\mathrm{p}\left(Z(\mathbf{x})|\mathbf{f}_{\mathbf{x}}\right)\mathrm{p}\left(\mathbf{f}_{\mathbf{x}}\right)}{\mathrm{p}\left(Z(\mathbf{x})\right)} & (2.20) \\ &= \mathcal{N}\left(\boldsymbol{\Sigma}^{\mathrm{T}}\left(\boldsymbol{\Sigma}^{\mathrm{T}} + \sigma^2\mathbf{I}\right)^{-1} Z(\mathbf{x}), \boldsymbol{\Sigma} - \boldsymbol{\Sigma}^{\mathrm{T}}\left(\boldsymbol{\Sigma}^{\mathrm{T}} + \sigma^2\mathbf{I}\right)^{-1}\boldsymbol{\Sigma}\right) & (2.21) \end{aligned}$$

The posterior mean realisation does not necessarily need to pass through all the datapoints unless the noise variance $\sigma^2 = 0$. The posterior mean realisations can be seen in detail in

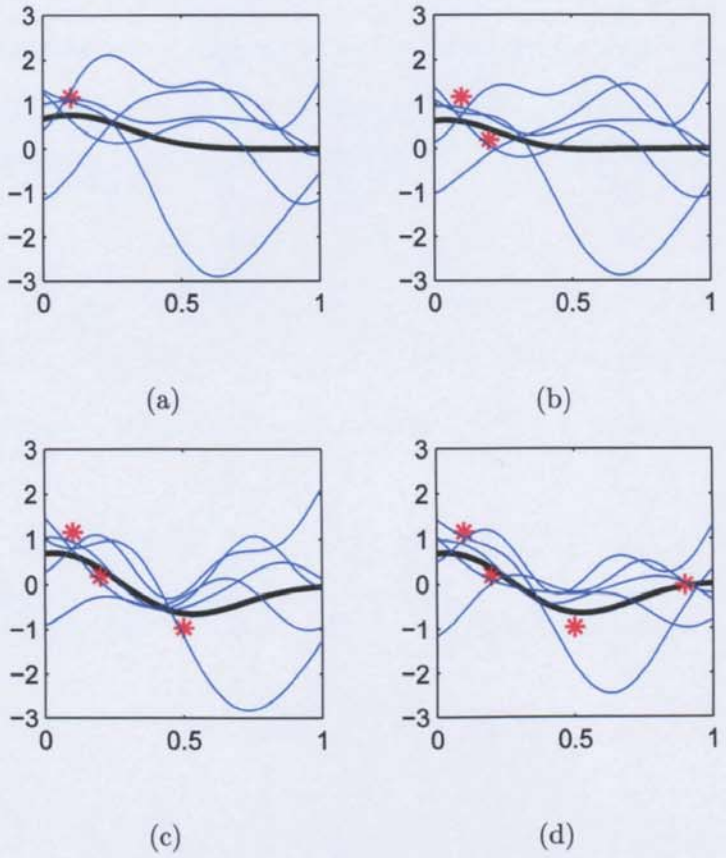


Figure 2.3: Samples from the posterior Gaussian process after (a) 1, (b) 2, (c) 3 and (d) 4 noisy observations have been selected. The thick line in each plot represents the most plausible (mean path) model for the data. The thin blue lines in each plot show possible models for the data given the parameters and observations. Each observation is indicated by a star.

Figure 2.2. Notice how as more observations are added to the Gaussian process, the function realisations are restricted to a smaller subset of realisations that are more plausible given the data. The number of possible realisations could be infinitely many, but for the purposes of this example only a few realisation are shown. Notice how the realisation passes exactly through the observations. Now compare these realisations to the process when noise has been added as shown in Figure 2.3. The realisations no longer pass through the observations exactly.

An intuitive way to view the *function–space view* of Gaussian processes is to consider an infinite number of functions that could be a plausible model for the data. Then upon the addition of each observation, the number of functions is restricted since some of the functions are less likely for the given observation. As further observations are added, the set of functions are restricted to those most likely given the data. Figure 2.2 gives a step by step example of this

process of restricting the possible functions. The thick line represents a mean for the functions. The thinner blue lines are samples from the posterior distribution which are likely interpolating functions. The plots show how these functions are successively restricted to pass through the given observations to give functions that are more likely.

Now assume that a new observation at location $\mathbf{x}_*$, is included. The model is updated with the new location:

$$p\left(\begin{bmatrix}\mathbf{f}_X\\ f_*\end{bmatrix}\middle|\begin{bmatrix}\mathbf{x}_X\\ \mathbf{x}_*\end{bmatrix}\right) \sim \mathcal{N}\left(0, \begin{bmatrix}\Sigma & \mathbf{k}_{N*}\\ \mathbf{k}_{N*}^T & k_{**}\end{bmatrix}\right) \tag{2.22}$$

giving the distribution of f_* conditioned on the function values $\mathbf{f}_X$ corresponding to the locations $\mathbf{x}_X$:

$$p(f_*|\mathbf{f}_X) \sim \mathcal{N}\left(\mathbf{k}_{N*}^T\Sigma^{-1}\mathbf{f}_X, k_{**} - \mathbf{k}_{N*}^T\Sigma^{-1}\mathbf{k}_{N*}\right) \tag{2.23}$$

To obtain the value of a function, $\mathbf{f}$, at a location, $\mathbf{x}$, the representation:

$$\mathbf{f}_x = \sum_i \alpha_i k_{x x_i} \tag{2.24}$$

can be used where $\boldsymbol{\alpha} = \{\alpha_1 \dots \alpha_N)\} = \Sigma^{-1}\mathbf{f}_X$.

2.2.2 Covariance functions and kernels

Performing prediction is only a part of the procedure that geostatistians apply when modelling spatial data. An important aspect of the whole procedure to obtain reliable results is in determining the form of the covariance matrix. The covariance matrix has already been mentioned with very few details about its purpose. The covariance matrix is typically computed from a covariance function. Covariance is a measure of similarity between observations.

There are two main methods that are used for covariance parameter estimation in geostatistics. Method–of–moments estimators are commonly used to estimate the semi–variogram. Method–of–moments or plug–in estimators are popular because the computational complexity scales $\mathcal{O}(n^2)$ and the empirical variogram is useful in determining the form of the covariance function model that would be appropriate given the data. The empirical variogram is defined as:

$$\hat{\gamma}(\mathbf{x}) = \frac{1}{2|N(\mathbf{h})|}\sum_{N(\mathbf{h})}\left\{Z(\mathbf{x}_i) - Z(\mathbf{x}_j)\right\}^2, \tag{2.25}$$

where:

$$N(\mathbf{h}) = \{(\mathbf{x}_i, \mathbf{x}_j) : \mathbf{x}_i - \mathbf{x}_j = \mathbf{h}; i, j = 1, \ldots, n\}. \tag{2.26}$$

and $|N(\mathbf{h})|$ is the number of pairs of data points separated by the particular lag vector $\mathbf{h}$. Once the semi–variances at each lag separation, $\gamma(\mathbf{x})$, have been determined, a variogram model is then selected and fitted to the calculated semi–variances. The variogram model is usually selected based on intuition about the process being observed. Selection can also often be aided by inspection of the empirical variogram (Ingram et al. 2005). Fitting a model to the semi–variances at each lag to determine the model parameters is usually performed using a least squares method (Webster and Oliver 2000). Once the model parameters have been determined, the covariance matrix can be constructed in the usual way.

Alternatively, maximum likelihood approaches have been used to estimate covariance parameters (Mardia and Marshall 1984; Kitanidis 1985). Maximum likelihood methods are more computationally intensive and scale $\mathcal{O}(n^3)$ making their direct application inappropriate for large datasets. Maximum likelihood was first used in the context of spatial statistics by Mardia (1980) although the idea has been around for many years. Despite the computational complexity issues and problems with multi–modality (Warnes and Ripley 1987; Mardia and Watkins 1989), these techniques have gained wide acceptance. For maximum likelihood estimation, the aim is to find the parameters, $\boldsymbol{\theta}$, that maximise the likelihood function (Pardo-Igúzquiza 1998). The parameters $\boldsymbol{\theta}$ are the parameters of the model. The maximum likelihood approach shown is given by

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{n}{2} - \frac{1}{2}|\boldsymbol{\Sigma}| - \frac{1}{2}Z(\mathbf{x}_N)^T\boldsymbol{\Sigma}^{-1}Z(\mathbf{x}_N). \tag{2.27}$$

and requires that the matrix $\boldsymbol{\Sigma}$, of size $(n \times n)$, has to be inverted.

Stationarity assumptions

It is important to make assumptions about the stationarity of the process. There are a number of assumptions that can be made, but here, *second order stationarity* is assumed. Second order stationarity can be assumed if the finite–dimensional distributions are invariant under shifts of the index, that is to say that the distribution of $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is identical to that of $(\mathbf{x}_1 + \mathbf{h}, \ldots, \mathbf{x}_n + \mathbf{h})$ where $\mathbf{h}$ is a vector.

Terminology for parameters

In the machine learning community the covariance function is often referred to as a kernel function. The terminology used to describe the parameters of the covariance functions also varies. In geostatistics the terms nugget, range and sill are commonly used to describe what in machine learning community are called noise, lengthscale (or sometimes relevance or roughness) and amplitude respectively.

Covariance function properties

Certain assumptions about the process that is being modelled help with the decision making process as to which covariance model is most appropriate. One property of a covariance function is the degree of smoothness. The degree of smoothness of a realisation of a process can be described mathematically by the degree of differentiability at the origin ($h \rightarrow 0$) of the covariance function.

In geostatistics, the choice of variogram model is often made based on empirical estimates given from the data. In method–of–moments estimation, the variogram model is often selected by eye and the chosen parametric variogram is fitted by ordinary least squares, weighted least squares or generalised least squares. As an alternative approach, it was proposed by Mardia (1980) that maximum likelihood methods could be used to fit the covariance function. This process involves the computationally intense maximisation of the log likelihood function with respect to the parameters of the covariance function. As Equation (2.27) shows, this is an $\mathcal{O}(n^3)$ operation and intractable for datasets of more than a few thousand observations.

The method–of–moments variogram estimator has been criticised since it can sometimes lead to misleading results (Stein 1999; Minasny and McBratney 2005). However, method–of–moments estimation remains a useful tool, particularly in helping to justify assumptions about variation in the data and in previous work it has been shown how this can aid the modeller in determining an appropriate covariance model (Ingram et al. 2005). A thorough comparison between method–of–moments and maximum likelihood techniques can be found in Lark (2000).

2.3 Treating large datasets

Notwithstanding the popularity of kriging, when applied to large datasets, non–trivial problems can arise due to numerical instabilities associated with solving large systems of equations (Davis and Grivet 1984; Dietrich and Newsam 1989).

In what follows, some of the common techniques that have been used to solve the above mentioned issues of computational speed and stability will be reviewed. Chapter 4 will also explore covariance selection and the relevance of the covariance function to increasing the stability of solving the kriging equations.

Methods that avoid directly inverting the covariance matrix, Σ, of the data are needed. Instead of inverting the covariance matrix directly, a number of researchers have proposed iterative conjugate gradient methods for solving the kriging equations (Gibbs 1997). These methods don't have the poor scaling associated with direct methods ($\mathcal{O}\left(n^2\right)$ per iteration), but solving the kriging equations exactly can only be guaranteed, machine precision allowing, if the algorithm is run for n iterations where n is the number of observations. Hence the number of iterations should be significantly less than n to achieve any speed–up using this method. Recently, approaches have been suggested to increase the speed of the conjugate gradient method, and these new approaches rely on approximate matrix–vector multiplications (Gray 2004; Yang et al. 2005).

2.3.1 Sequential kriging

Most kriging implementations are batch algorithms, in that all the observations (or sometimes a smaller subset) are processed in a single iteration. A sequential algorithm, whereby the model is updated as each observation is considered individually, will be presented in this section. Sequential algorithms have been used previously for a variety of applications (Vargas-Guzmán and Yeh 1999; Lawrence and Herbrich 2001; Csató 2002; Vargas-Guzmán and Yeh 2002; Sakata et al. 2004; Liu and Yeh 2004; Huang 2005).

What are the advantages of a sequential kriging algorithm? Suppose that a large covariance matrix inverse has been computed requiring a Herculean effort. Now a small change to the covariance matrix is required perhaps due to newly collected data. Ideally one would not want to have to compute the entire matrix inverse again. The matrix inverse can be updated with

significantly reduced computational complexity, ($\mathcal{O}(n^2)$) per update (Press et al. 1996).

It has also been noted that online or sequential algorithms can often be executed much faster, particularly when a dataset has a large amount of redundancy (Lawrence et al. 2003). These sequential algorithms can be particularly useful to solve the $\mathcal{O}(n^3)$ matrix inversion bottleneck that arises when dealing with large matrices. The idea of sequential kriging is not new; Sakata et al. (2004) proposed a sequential algorithm that has been used to improve numerical stability in large systems. Vargas-Guzmán and Yeh (1999) also present a similar sequential kriging algorithm which improves stability.

The first step in the sequential kriging algorithm is to define the partitioned covariance matrix:

$$\boldsymbol{\Sigma}_{s+1} = \begin{bmatrix} \boldsymbol{\Sigma}_N & \mathbf{k}_{N(s+1)} \\ \mathbf{k}_{N(s+1)}^T & \mathbf{k}_{**} \end{bmatrix} \tag{2.28}$$

and its inverse, which can be derived from the partitioned matrix inverse identity (see Appendix A.2):

$$\boldsymbol{\Sigma}_{s+1}^{-1} = \begin{bmatrix} \boldsymbol{\Sigma}_s^{-1} + \sigma_{s+1}^2 \mathbf{m}\mathbf{m}^T & \mathbf{m} \\ \mathbf{m}^T & (\sigma_{s+1}^2)^{-1} \end{bmatrix}. \tag{2.29}$$

Equation (2.28) shows how the covariance matrix $\boldsymbol{\Sigma}$ is partitioned and gives an intuition on the constituent parts of a covariance matrix. As this is an iterative algorithm, the subscript s will be used to denote the current state of the model and $(s+1)$ will be used to denote the model at the next iteration. The vector of covariances $\mathbf{k}_{N(s+1)} = k(\mathbf{x}_{1\ldots s}, \mathbf{x}_{s+1})$ gives the covariance evaluated between the new observation to be added ($\mathbf{x}_{s+1}$) and the observations already included in the model at that iteration. Equation (2.29) shows the partitioned inverse $\boldsymbol{\Sigma}^{-1}$. It can be seen that the matrix $\boldsymbol{\Sigma}_{s+1}$ does not need to be inverted directly, the matrix inverse can be expanded successively by extending with an extra row and column for each new observation. This uses calculations of $\mathbf{m} = -\sigma_{s+1}^{-2}\boldsymbol{\lambda}$ where $\boldsymbol{\lambda} = \boldsymbol{\Sigma}_s^{-1}\mathbf{k}_{N(s+1)}$ which the reader will recognise from Equation (2.7) and $\sigma_{s+1}^2 = \mathbf{k}_{**} - \mathbf{k}_{N(s+1)}^T\boldsymbol{\Sigma}_s^{-1}\mathbf{k}_{N(s+1)}$ which is also the predictive variance at the new location (MacKay 1998). Looking at the constituent parts individually, an intuition can be gained about the process that is taking place in this matrix update. The vector $\mathbf{m}$ is the vector of weights given by $\boldsymbol{\Sigma}_s^{-1}\mathbf{k}_{N(s+1)}$, of the existing system given the new observation that is to be added in the current iteration. This vector of weights is scaled by the inverse

predictive variance $\left(\sigma^2_{s+1}\right)^{-1}$ (or predictive precision (Menzefricke 1995)). The existing matrix inverse, $\mathbf{\Sigma}_s^{-1}$, is updated by the outer product $\mathbf{m}\mathbf{m}^T$ scaled by the predictive variance of the new observation.

Cholesky decomposition

It has been shown that the Sherman–Morrison–Woodbury formula (Golub and Van Loan 1989), described above, can be numerically unstable (Fine and Scheinberf 2002) particularly in cases where sufficient numerical accuracy is not retained or in the presence of round–off errors. Seeger (2003) argues that the Sherman–Morrison–Woodbury formula should only be used as a symbolic rewriting tool, and that for actual computations, the Cholesky decomposition should be used since this is well known to be numerical stable and efficient. Therefore, for stable implementations it is recommended that the Cholesky factor of the inverse is retained rather than calculating the inverse directly. By using the Cholesky factor, performance can be further increased. Further details of the Cholesky factor derivations can be found in Seeger (2003).

The Cholesky decomposition is a method for decomposing a positive–definite matrix into a lower triangular matrix and can be thought of as a matrix square root. Assuming that the inverse of the covariance matrix is to be sequentially updated then it is assumed that

$$\mathbf{\Sigma}^{-1} = \mathbf{L}^T\mathbf{L}. \tag{2.30}$$

Given the Cholesky factor $\mathbf{L}$, by substituting into (2.11), the predictive variance is given by:

$$\sigma_s^2 = \mathbf{k}_{**} - \mathbf{k}^T\mathbf{L}^T\mathbf{L}\mathbf{k} \tag{2.31}$$

which leads to the simplification:

$$\gamma_s = \mathbf{k}_{**} - \mathbf{v}^T\mathbf{v} \tag{2.32}$$

where:

$$\mathbf{v} = -\mathbf{L}\mathbf{k} \tag{2.33}$$

Updating the Cholesky factor representation of the inverse covariance matrix is given by:

$$\mathbf{L}_{s+1} = \begin{bmatrix} \mathbf{L}_s & 0 \\ \mathbf{L}_s^T\mathbf{v}\sqrt{\gamma^{-1}} & \sqrt{\gamma^{-1}} \end{bmatrix} \tag{2.34}$$

2.4 Subsetting the data

Geostatisticians have used many approaches to overcome the problems caused by the inversion of a large covariance matrix of all the observations in a kriging calculation. In this section the basics of some of these methods will be introduced and some alternative solutions that have been widely adopted in the machine learning community will be presented.

2.4.1 Subsampling

One simple approach is to subsample the data. For example, if a dataset is sampled at 10 m intervals, one could then subsample the dataset and use only observations at 20 m intervals. In doing so, up to 75% of the data are potentially discarded, depending on the sampling grid dimensions. Since collecting a dataset can often be very expensive in monetary terms, why discard this data? The designed sampling scheme could already be appropriate for the given process. Subsampling the data could lead to suboptimal results. It is not uncommon that there are many datasets that are densely sampled (where the sampling density is large with respect to the range parameter of the process) and hence such an approach would not result in a significant loss of information from the dataset. For sparsely sampled datasets (where the sampling density is small with respect to the range parameter of the process) a practitioner would need to use extreme caution in subsampling in this way.

A standard approach is to sample the dataset densely to see the effects of subsampling in terms of cross–validation error and generated maps. Comparing the subsamples with the original dense data and analysing the cross-validation error can give indications as to the appropriateness of the sampling scheme (Frogbrook 1999; Frogbrook and Oliver 2000; Frogbrook et al. 2002).

Computational speed is not the only reason one might want to subsample a dataset. In many geostatistical applications the collection of data can be extremely expensive and time consuming. Within soil science much work has been done using aerial photography to determine a suitable sampling scheme for a given area (Kerry and Oliver 2007). In Chapter 7 a technique for sample design configuration is presented which uses the sequential projected process kriging algorithm introduced in Section 2.5.2 of this Chapter.

Changing the granuality of the sampling grid is not the only way of subsampling. The

observations collected in some datasets do not lie on a defined grid. Some sampling schemes are designed with areas of differing sampling densities so that all the ranges of variation in the process can be observed. To learn a reliable variogram model, it is particularly important to sample at short range and long range lag separations, but by sampling on a grid it is possible to miss important properties of the variation at some lag separations (Webster and Oliver 2000).

Having discussed some specific approaches used to subsample dataset, a simple naïve approach could be considered. Randomly selecting a subset of the data is an option that could be considered. Where the data does not lie on a defined grid, it is common to take a random selection of the observations. In this thesis a discussion of the many different types of random sampling will be avoided and for experiments, simple random sampling will be used. De Gruijter et al. (2005) provide a thorough review of various random sampling methods. Random sampling could be suboptimal with regards to the accuracy of the prediction, but it is very fast. This will be used as a baseline measure in comparisons with other methods that will be discussed later in this chapter.

Ultimately, for the simple random subsampling method to be applied without a large loss of information, the sampling density of the dataset is the key factor. If the data is over subsampled, that is to say that too few observations are selected, this will lead to spurious results. Techniques using the variogram to determine the grid spacing of the sampling grid are commonly used in geostatistics (Webster and Oliver 1992).

Reduction in uncertainty

Lawrence and Herbrich (2001) present an approach inspired by information theory for automatically subsampling the dataset which attempts to minimise any loss of information. The essence of the algorithm is to select the observations which are the most informative or which cause the largest reduction of uncertainty in the model, that is to say that the observation which causes the predictive variance to shrink the most with their inclusion. Calculating the reduction in uncertainty has been used in geostatistics to select locations which are informative. Heuvelink et al. (2006) presents the Mean Universal Kriging Variance (MUKV) measure to determine optimal sample configuration locations, but not in a sequential framework.

The method of Lawrence and Herbrich (2001) requires that the number of observations to

be retained is selected *a–priori* since they state that there is no principled way of selecting an appropriate size for the number of observations to be retained under their method. One could think of using some carefully defined threshold selected for a particular dataset as an alternative but this would require significant insight into the problem. The algorithm used is a greedy forward selection algorithm. It is used within the sequential kriging framework discussed in Section 2.3.1. The reduction in uncertainty for the kriging process is calculated for each observation individually as if it were added to the kriging system. The observation which is most informative is then sequentially added to the kriging system. This process is repeated until the number of observations added to the kriging system is above some threshold selected *a-priori*. Later, Seeger and Williams (2003) proposed the information gain score, which not only measures the predictive variance but also the change in the predictive mean. It ensures that the data space is sufficiently well sampled (given the range of the process), and that observations are selected based on how much of a surprise or novelty they present to the current process covariance parameters. The higher the novelty of an observation, the higher the score for including the observation into the kriging system. After the observations have been selected, parameters are learned with this subset of the dataset. The algorithm is run for a predefined number of iterations, interleaving observation selection and covariance parameter estimation since the covariance parameters are related to the optimal subset.

2.4.2 Moving windows

One approach introduced by David (1976) uses a specified search radius from a selected centre to select a local neighbourhood of observations to use in the kriging system. This neighbourhood moves according to the location which is being predicted. An alternative approach for selecting the neighbourhood is to select a predetermined number of nearby observations for each prediction location. As noted by Davis and Culbane (1984), these methods produce spurious behaviour in some of the estimates and hence should be used with caution, this is particularly apparent as observations are added or removed from the moving window. Ad–hoc methods of subsetting the data were formalised by the moving-window approach of Haas (1990) and Haas (1995), although the local covariance functions fitted within the window may yield incompatible covariances at larger spatial lags. Cressie (1993) states that for datasets that are large, the

general feeling is that kriging is impossible and ad–hoc local kriging neighbourhoods are typically used. Isaaks and Srivastava (1989) devote a whole chapter to choosing an effective search strategy. Implementations of kriging such as the commonly used GStat[1] or Isatis[2] tend to use this approach for performing kriging efficiently.

2.4.3 Partitioning

Another commonly used approach is to partition the data into subsets based on natural spatial groupings (Meyer 2004). Kriging is then performed on these individual groups. Following this method, none of the data is discarded as with subsampling, however, there is a loss of information of the correlations between observations in different partitions. Clustering techniques are discussed briefly in Chapter 6. Stein (1999) argued in favour of partitioning the data into smaller subregions for reasons other than solely to reduce computational complexity. Partitioning the data can be an effective way for locating nonstationarities in the data. Often the loss of accuracy by partitioning the data is not a significant problem, however, for many problems this yields poor results. In particular so called edge effects can be apparent in the prediction stage at the boundaries of partitions.(Auñón and Gómez-Hernández 2000)

Later in this thesis (Chapter 6), the Bayesian committee machine (Tresp 2000a) framework will be presented which uses a similar partitioning scheme, but does not suffer from these edge effects. Choosing an effective partitioning scheme for the data is non–trivial since balancing the number of data in each partition is also an important consideration (Tresp 2000a).

2.4.4 Subset of data methods

Subset of data methods refer to methods for automating the selection of the subset and then performing prediction. For the methods that follow, some terminology will be introduced. First, the concept of an *active set* and that of an *inactive set* are defined. The term *active set* refers to the representative subset of the data used for prediction (Lawrence and Herbrich 2001). Depending on the algorithm used, the information in the *inactive set* can be retained by projecting the effect of the observations on to the *active set*. The *active set* contains *active points* which refer to the locations included in the active set.

[1] http://www.gstat.org/
[2] http://www.geovariances.com/

The point to make about these methods is that only a subset of the data is used in the prediction process, even though it is possible that all the data is used in the training stage. The subset of data used in prediction will be termed the *active set*, $\mathbf{x}_M$ and the data that does not belong to this subset, ie. the data that is not used in prediction, will be called the *inactive set*. In the machine learning literature, the *active set* is known by a number of terms such as basis vectors or $\mathcal{BV}$ set (Csató and Opper 2001b), pseudo–inputs (Snelson and Ghahramani 2006), support points or inducing points (Quiñonero-Candela and Rasmussen 2005). Although in each context these can often have a slightly different meaning, the terms *active set* or *active points* will be used universally throughout this thesis.

The essence of subset of data methods is to retain a small subset of the dataThis subset (*active set*) is then used for prediction. Good prediction accuracy can be achieved with subset of data methods when applied to datasets that are densely sampled. Selecting which points are included in the active set is an important part of this algorithm. Searching all the possible combinations of observations by doing an exhaustive search to obtain an optimal active set would result in prohibitively slow computation time. Alternatively, greedy algorithms can be used which use heuristics to score each observations according to some *informativeness* measure, observations that are most informative are selected and then added to the active set.

In Lawrence et al. (2003), it was initially suggested that each observation in the inactive set should be scored based on the reduction in entropy (or reduction in uncertainty of the kriging predictive variance distribution) that adding the observation to the Gaussian process would cause. The observation which causes the maximum reduction in entropy is considered to be the most informative and is added to the active set using the sequential kriging algorithm.

Often large amounts of data are discarded using the subset of data method. Discarding large amounts of data may not cause significant issues if there are sufficiently many densely sampled observations in the dataset. The reduction in the computational complexity is attractive for large datasets, but the ability to treat the whole dataset without discarding observations would be desirable.

The version of the subset of data algorithm introduced by Lawrence et al. (2003) uses a sequential kriging framework with a greedy selection procedure. An alternative would be to used a batch version of the subset of data method. Assuming that the active set has already

been determined, then the batch predictive mean equation for the subset of data method is given by:

$$\hat{Z}(\mathbf{x}_*) = \mathbf{k}_{*M}^{\mathsf{T}} \boldsymbol{\Sigma}_{SoD}^{-1} Z(\mathbf{x}_M) \tag{2.35}$$

and the predictive variance is given by

$$\sigma_*^2(\mathbf{x}_*) = \mathbf{k}_{**} - \mathbf{k}_{*M}^{\mathsf{T}} \boldsymbol{\Sigma}_{SoD}^{-1} \mathbf{k}_{*M} \tag{2.36}$$

where $\boldsymbol{\Sigma}_{SoD} = \mathbf{K}_{MM}$ and is the covariance matrix between the active points.

To calculate the likelihood, which is particularly useful for parameter estimation in the maximum likelihood framework, the following equation is used:

$$\mathrm{P}(\mathcal{D}_N|\boldsymbol{\theta}) = -\frac{n}{2} - \frac{1}{2}|\boldsymbol{\Sigma}_{SoD}| - \frac{1}{2} Z(\mathbf{x}_M)^{\mathsf{T}} \boldsymbol{\Sigma}_{SoD}^{-1} Z(\mathbf{x}_M). \tag{2.37}$$

Note how only the observations included in the active set, $Z(\mathbf{x}_M)$, are used in the likelihood calculation. In this context $\mathrm{P}(\mathcal{D}_N|\boldsymbol{\theta}) \equiv \mathrm{P}(\mathcal{D}_M|\boldsymbol{\theta})$

2.5 Projected process algorithms

Subset of data methods have their use, but prediction accuracy will only be retained for densely sampled datasets. While the algorithms will be fast for some data, features in the data will be lost due to discarding observations. Projected process methods help overcome this problem by using all of the data, but at the same time the computational complexity only scales cubically for active set size and linearly in terms of the number of observations (Snelson 2007). That is to say $\mathcal{O}(nm^2)$ where n is the total number of observations and m is the size of the active set.

In this section a family of algorithms will be discussed for treating large datasets that use the entire dataset. The algorithms that are discussed are divided into two categories. Section 2.5.1 discusses batch versions of the algorithms; algorithms where the calculations are performed on the entire dataset in one iteration. In contrast, Section 2.5.2 discusses iterative, online or sequential versions of these algorithms. To illustrate the difference, the subset of data method discussed in Section 2.4.4 can be viewed in both ways. The batch version of the subset of data method is equivalent to performing kriging with the observations that have been included in the active set, the observations in the inactive set are discarded as shown by Equations (2.35) & (2.36). Sequential algorithms process observations one at a time. In this context, the results

will be equivalent; however, in some of the methods discussed later, the ordering of the data in the sequential algorithms leads to different results (Csató and Opper 2002).

There are advantages to choosing sequential algorithms over batch algorithms in the context of these large scale geostatistical problems and these advantages will be discussed in more depth later. Sequential algorithms can be quite complicated and intuitions about the algorithms can be lost due to their sequential nature. Hence, the batch equivalents are presented first and links between the batch algorithms and the sequential algorithms will be made.

The key problem for batch algorithms is in the selection of the active set. Sequential algorithms provide a convenient framework for measuring the importance of each observation; no such method has yet been developed for batch versions. One alternative solution for determining the active set for batch versions is discussed in Section 2.6, but to summarise, selecting the active set in a batch framework is problematic and few solutions have been presented.

A naïve method that would offer reasonable performance in terms of computation speed and prediction accuracy would be selecting the active set randomly. Instead of the effort spent calculating *best* active points, the active set size should be set to be larger than would be expected for the particular dataset, in this way it increases the probability that the data space area is sufficiently well sampled.

2.5.1 Batch algorithms

Algorithms for treating large datasets have been a very popular area of research in the machine learning community recently. On one level, the approximations that will be discussed can be viewed as an approximation to the covariance function. In approximating the covariance function, the likelihood can also be approximated. Instead of assuming that the covariance matrix $\mathbf{\Sigma}$ is equal to the covariance matrix of all the observations $\text{cov}(\mathbf{x}_N, \mathbf{x}_N)$, different assumptions will be made regarding the structure of $\mathbf{\Sigma}$. A recent unifying view of these methods was presented in Quiñonero-Candela and Rasmussen (2005) which shows the connections that each method has to the others. In this thesis, the term *projected process* will be used to describe this family of algorithms although they are often known independently by the abbreviations DTC (Deterministic Training Conditional), FITC (Fully Independent Training Conditional) and PITC (Partial Independent Training Conditional). There are three different methods which approximate the

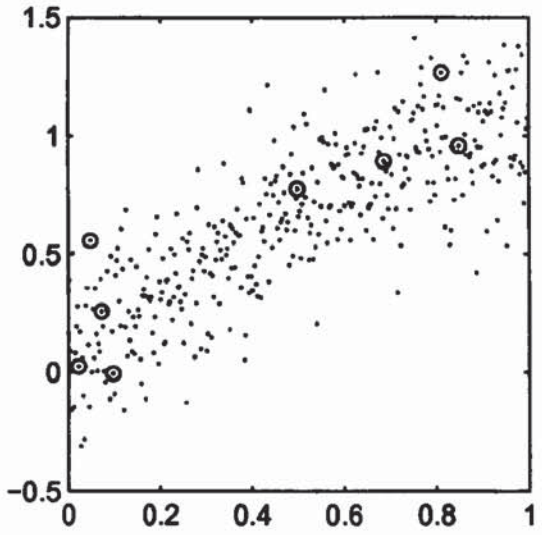


Figure 2.4: Observations from a simple 1D dataset. Points are observations, circled points are the selected active points. The x–axis is the observation location and the y–axis is the observed value.

covariance matrix to differing degrees of accuracy.

Low–rank approximation

The covariance matrix encodes the assumptions made about the covariance structure of the data. Throughout this section, the covariance matrix of a simple dataset will be approximated using 8 active points ($m = 8$). The simple dataset that will be used has 400 observations ($n = 400$). The data arrangement is shown in Figure 2.4 and the active points are circled. The covariance parameters have been fixed to predetermined values for all of the examples that will follow in this section. The values chosen are not optimal but were used to make the the illlustration clear.

Using traditional kriging algorithms, the covariance matrix of the observations is obtained by calculating the covariance between all the observations, $\mathbf{\Sigma} = \mathrm{cov}(\mathbf{x}_N, \mathbf{x}_N)$. Throughout this section the notation for the covariance matrix is given by:

$$\mathrm{cov}(\mathbf{a}, \mathbf{b}) = \mathbf{K}_{ab}, \tag{2.38}$$

since it helps with the presentation of the equations later in this section.

A pictorial representation of what the covariance matrix would look like for the dataset shown in Figure 2.4 can be seen in Figure 2.5. The covariance matrix is of size $n \times n$. In the

Figure 2.5: Pictorial representation of a covariance matrix. Dark regions represent areas of high covariance and light regions represent areas of low covariance.

covariance matrix, the observations are ordered in the same way that the data is ordered along the x–axis.

The techniques discussed in this section rely on approximating the covariance matrix structure by a low–rank matrix approximation. Instead of having to invert the $n \times n$ covariance matrix directly, a lower rank $m \times m$ matrix can be inverted.

The low–rank or Deterministic Training Conditional (Quiñonero-Candela and Rasmussen 2005) or Projected Latent Variables (Seeger 2003) approximation assumes that the covariance function takes the form:

$$\mathrm{cov}(\mathbf{a}, \mathbf{b}) = \mathbf{k}_{\mathbf{a}M} \mathbf{K}_{MM}^{-1} \mathbf{k}_{M\mathbf{b}}, \tag{2.39}$$

where M represents the active points that have been selected for the approximation. An important aspect of this method is the selection of the active points. Active point locations should be selected throughout the space of the data. For areas of high variability of the process, more active points should be chosen.

This low–rank covariance equation is assumed to be the default covariance function in the following equations for computing $\mathbf{k}_{*N}$ and $\boldsymbol{\Sigma}$. The approximation is derived by inserting this low–rank covariance function into the predictive mean and variance equations:

$$\hat{Z}(\mathbf{x}_*) = \mathbf{k}_{*N}^T \boldsymbol{\Sigma}_{DTC}^{-1} Z(\mathbf{x}_N), \tag{2.40}$$

$$\sigma_*^2(\mathbf{x}_*) = \mathbf{k}_{**} - \mathbf{k}_{*N}^T \boldsymbol{\Sigma}_{DTC}^{-1} \mathbf{k}_{*N}, \tag{2.41}$$

where the covariance is determined using the new, low–rank approximated covariance function described in Equation (2.39). The covariance matrix now has the form:

$$\boldsymbol{\Sigma}_{DTC} = \mathbf{k}_{NM} \mathbf{K}_{MM}^{-1} \mathbf{k}_{MN} + \boldsymbol{\beta}, \tag{2.42}$$

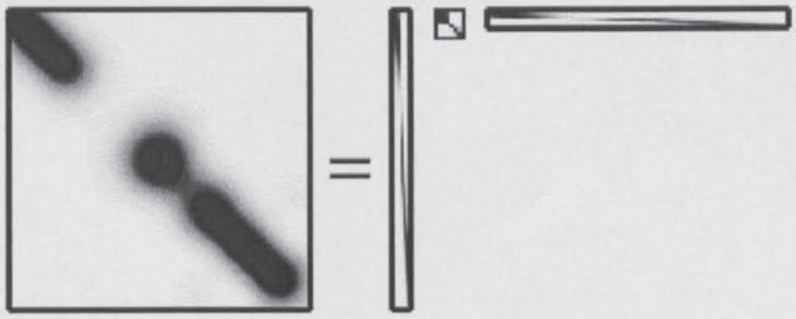


Figure 2.6: Pictorial representation of the construction of the noise free DTC covariance matrix approximation using a low–rank matrix.

where it is assumed that an active set has been selected *a–priori*. $\boldsymbol{\beta} = \sigma^2\mathbf{I}$ is a diagonal noise term. A pictorial representation of how the covariance matrix is constructed can be seen in Figure 2.6. Comparing the full covariance matrix shown in Figure 2.5, shows that the approximation is particularly good in regions where active points have been selected, but in other regions, where there are no active points, the covariance is severely underestimated.

It might seem that nothing has been gained, in fact, it seems that something has been lost since this approximate matrix, $\boldsymbol{\Sigma}_{\mathrm{DTC}}$, of size $n \times n$ still needs to be inverted and much of the data from the true covariance has been discarded. By writing the covariance in such a way, the Sherman–Morrison–Woodbury identity (see Appendix A.1) can be applied to derive an alternative representation:

$$\left[\mathbf{k}_{\mathrm{NM}}\mathbf{K}_{\mathrm{MM}}^{-1}\mathbf{k}_{\mathrm{MN}} + \boldsymbol{\beta}\right]^{-1} = \boldsymbol{\beta}^{-1} - \boldsymbol{\beta}^{-1}\mathbf{k}_{\mathrm{NM}}\mathbf{A}^{-1}\mathbf{k}_{\mathrm{MN}}\boldsymbol{\beta}^{-1}, \tag{2.43}$$

where $\mathbf{A} = \mathbf{K}_{\mathrm{MM}} + \mathbf{k}_{\mathrm{MN}}\boldsymbol{\beta}^{-1}\mathbf{k}_{\mathrm{NM}}$. Now, it is no longer necessary to invert a $n \times n$ but rather to invert a $m \times m$ matrix. The inverse of $\boldsymbol{\beta}$ is computationally trivial since it is a diagonal matrix. The predictive mean can now be written as:

$$\hat{Z}(\mathbf{x}_*) = \mathbf{k}_{*\mathrm{M}}\mathbf{A}^{-1}\mathbf{k}_{\mathrm{MN}}\boldsymbol{\beta}^{-1}Z(\mathbf{x}_{\mathrm{N}}), \tag{2.44}$$

and the predictive variance can be written as:

$$\sigma_*^2(\mathbf{x}_*) = \mathbf{k}_{**} - \mathbf{k}_{*\mathrm{N}}^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathrm{DTC}}^{-1}\mathbf{k}_{*\mathrm{N}}, \tag{2.45}$$

thus reducing the computational complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(nm^2)$. When n is much larger than m, significant speed ups will be achieved.

To compute the likelihood of the model, again the matrix inversion formula needs to be applied. Also the determinant identity, Appendix (A.2), should be used to calculate the $\mathcal{L}_1$

term in the likelihood equation.

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{n}{2} - \frac{1}{2}|\boldsymbol{\Sigma}_{\text{DTC}}| - \frac{1}{2}\mathbf{Z}(\mathbf{x}_N)^T\boldsymbol{\Sigma}_{\text{DTC}}^{-1}\mathbf{Z}(\mathbf{x}_N) \tag{2.46}$$

$$= -\frac{n}{2} - \frac{1}{2}\log\mathcal{L}_1 - \frac{1}{2}\mathcal{L}_2. \tag{2.47}$$

where:

$$\mathcal{L}_1 = |\boldsymbol{\beta}|\,|\mathbf{K}_{MM}|\,\left|\mathbf{K}_{MM}^{-1} + \mathbf{k}_{MN}(\mathbf{K}_{MM} + \mathbf{k}_{MN}\boldsymbol{\beta}^{-1}\mathbf{k}_{NM})^{-1}\mathbf{k}_{NM}\right|, \tag{2.48}$$

and:

$$\mathcal{L}_2 = \mathbf{Z}(\mathbf{x}_N)^T(\boldsymbol{\beta}^{-1} - \boldsymbol{\beta}^{-1}\mathbf{k}_{NM}\mathbf{A}^{-1}\mathbf{k}_{MN}\boldsymbol{\beta}^{-1})\mathbf{Z}(\mathbf{x}_N). \tag{2.49}$$

Low-rank approximation with exact diagonal

A further refinement was proposed in the work of Snelson and Ghahramani (2006) where they noted that the low–rank approximation in the previous section, the DTC approximation, tends to underestimate the predictive variance away from the active points. The Fully Independent Training Condition (FITC) was proposed to overcome this problem, where they propose to add a diagonal term to the covariance matrix which ensures the low–rank covariance matrix approximation is exact on the diagonal.

The equations for the predictive distribution for the FITC approximation are:

$$\hat{\mathbf{Z}}(\mathbf{x}_*) = \mathbf{k}_{*N}^T\boldsymbol{\Sigma}_{\text{FITC}}^{-1}\mathbf{Z}(\mathbf{x}_N), \tag{2.50}$$

$$\sigma_*^2(\mathbf{x}_*) = \mathbf{k}_{**} - \mathbf{k}_{*N}^T\boldsymbol{\Sigma}_{\text{FITC}}^{-1}\mathbf{k}_{*N}, \tag{2.51}$$

where the covariance matrix is given by:

$$\boldsymbol{\Sigma}_{\text{FITC}} = \mathbf{k}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{k}_{MN} + \boldsymbol{\Lambda} + \boldsymbol{\beta}, \tag{2.52}$$

with the diagonal matrix:

$$\boldsymbol{\Lambda} = \text{diag}(\mathbf{K}_{NN} - \mathbf{k}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{k}_{MN}), \tag{2.53}$$

ensures that the diagonal of the covariance matrix is exact. $\mathbf{K}_{NN}$ is the full covariance matrix shown in Figure 2.5. Figure 2.7 shows a pictorial representation of how the covariance matrix is constructed.

Figure 2.7: Pictorial representation of the construction of the noiseless FITC covariance matrix approximation which is identical to the noiseless DTC approximation except that the exact diagonal term has been retained in the matrix by adding the difference between the low–rank matrix approximation and the exact diagonal.

Using the matrix inversion identity from Appendix A.1, the FITC covariance approximation can be rewritten as:

$$\left[\mathbf{k}_{\mathrm{NM}}\mathbf{K}_{\mathrm{MM}}^{-1}\mathbf{k}_{\mathrm{MN}} + (\mathbf{\Lambda}+\boldsymbol{\beta})\right]^{-1} = (\mathbf{\Lambda}+\boldsymbol{\beta})^{-1} - (\mathbf{\Lambda}+\boldsymbol{\beta})^{-1}\mathbf{k}_{\mathrm{NM}}\mathbf{A}^{-1}\mathbf{k}_{\mathrm{MN}}(\mathbf{\Lambda}+\boldsymbol{\beta})^{-1}, \quad (2.54)$$

where:

$$\mathbf{A} = \mathbf{K}_{\mathrm{MM}} + \mathbf{k}_{\mathrm{MN}}(\mathbf{\Lambda}+\boldsymbol{\beta})^{-1}\mathbf{k}_{\mathrm{NM}}. \quad (2.55)$$

As with the DTC covariance approximation, the added accuracy does not result in a significant increase in computation since the matrix inversion $\mathbf{\Lambda}+\boldsymbol{\beta}$ is feasible, because it is again diagonal.

Following on from Equation (2.46) for the DTC likelihood approximation, the two parts of the FITC likelihood can be computed as:

$$\mathcal{L}_1 = |(\mathbf{\Lambda}+\boldsymbol{\beta})||\mathbf{K}_{\mathrm{MM}}|\left|\mathbf{K}_{\mathrm{MM}}^{-1} + \mathbf{k}_{\mathrm{MN}}(\mathbf{K}_{\mathrm{MM}} + \mathbf{k}_{\mathrm{MN}}(\mathbf{\Lambda}+\boldsymbol{\beta})^{-1}\mathbf{k}_{\mathrm{NM}})^{-1}\mathbf{k}_{\mathrm{NM}}\right| \quad (2.56)$$

and:

$$\mathcal{L}_2 = \mathrm{Z}(\mathbf{x}_{\mathrm{N}})^{\mathsf{T}}(\mathbf{\Lambda}+\boldsymbol{\beta})^{-1} - (\mathbf{\Lambda}+\boldsymbol{\beta})^{-1}\mathbf{k}_{\mathrm{NM}}\mathrm{A}^{-1}\mathbf{k}_{\mathrm{MN}}(\mathbf{\Lambda}+\boldsymbol{\beta})^{-1}\mathrm{Z}(\mathbf{x}_{\mathrm{N}}). \quad (2.57)$$

Low–rank approximation with exact block diagonal structure

The final approximation is similar to the FITC approximation, but instead of the covariance matrix being exact on the diagonal, an exact block diagonal structure is retained. This approach was initially proposed by Tresp (2000a) in a different framework and will be discussed in more detail in Section 6.4.3. The connection to the methods discussed here were made clear by Quiñonero-Candela and Rasmussen (2005). This approach is called the Partially Independent Training Condition.

The equations for the predictive distribution for the PITC approximation are:

$$\hat{Z}(\mathbf{x}_{**}) = \mathbf{k}_{*N}^{T}\boldsymbol{\Sigma}_{PITC}^{-1}Z(\mathbf{x}_N), \tag{2.58}$$

$$\sigma_*^2(\mathbf{x}_*) = \mathbf{k}_{**} - \mathbf{k}_{*N}^{T}\boldsymbol{\Sigma}_{PITC}^{-1}\mathbf{k}_{*N}, \tag{2.59}$$

where the covariance matrix is given by:

$$\boldsymbol{\Sigma}_{PITC} = \mathbf{k}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{k}_{MN} + \boldsymbol{\Lambda} + \boldsymbol{\beta}, \tag{2.60}$$

with the block diagonal matrix:

$$\boldsymbol{\Lambda} = \text{blockdiag}(\mathbf{K}_{NN} - \mathbf{k}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{k}_{MN}). \tag{2.61}$$

Rewriting the covariance using the matrix inversion identity (Appendix A.1), the PITC covariance approximation becomes:

$$\left[\mathbf{k}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{k}_{MN} + (\boldsymbol{\Lambda}+\boldsymbol{\beta})\right]^{-1} = (\boldsymbol{\Lambda}+\boldsymbol{\beta})^{-1} - (\boldsymbol{\Lambda}+\boldsymbol{\beta})^{-1}\mathbf{k}_{NM}\mathbf{A}^{-1}\mathbf{k}_{MN}(\boldsymbol{\Lambda}+\boldsymbol{\beta})^{-1}, \tag{2.62}$$

where:

$$\mathbf{A} = \mathbf{K}_{MM} + \mathbf{k}_{MN}(\boldsymbol{\Lambda}+\boldsymbol{\beta})^{-1}\mathbf{k}_{NM}. \tag{2.63}$$

The structure of the block diagonal matrix lambda depends on user requirements. When there is only one block this approximation is equivalent to the full predictive distribution equations or likelihood. When the block sizes are one, then this method is equivalent to the FITC approximation. It might seem that by introducing a block diagonal structure into $\boldsymbol{\Lambda}$ that the inverse can no longer be found easily. Using the block matrix inversion identity found in Appendix A.5, it shows that the inversion only requires matrix inversions of each diagonal block. The blocks are not constrained to be the same size, but it makes sense if they are since the algorithm complexity is related to the largest matrix block diagonal inverse.

Figure 2.8 shows how the covariance matrix is constructed from the low–rank matrix approximation with the exact block diagonal structure retained. Each block is of equal size. For the given covariance matrix it might be worth changing the block size for the areas where the covariance is poorly approximated. Figure 2.9 shows a variable block diagonal scheme which approximates the covariance better, although at the extra cost of having to invert the larger diagonal matrix block. It makes sense to keep the matrix blocks of the same size since the computational complexity will be dominated by the largest matrix block.

Figure 2.8: Pictorial representation of the construction of the PITC covariance matrix approximation which is identical to the DTC approximation except that an exact block diagonal structure is retained in the matrix. The block diagonal structure has fixed block sizes.

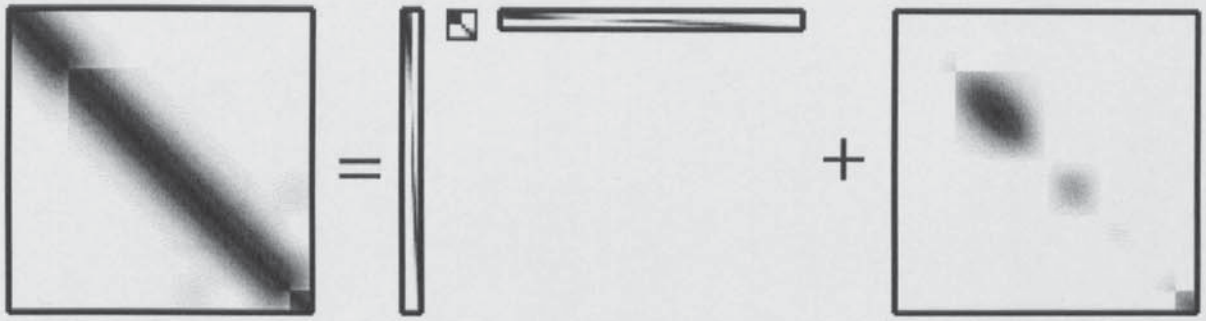


Figure 2.9: Pictorial representation of the construction of the PITC covariance matrix approximation which is identical to the DTC approximation except that an exact block diagonal structure is retained in the matrix. The block diagonal structure has variable block sizes.

The two terms of the likelihood are computed in the same way as the FITC approximation:

$$\mathcal{L}_1 = |(\mathbf{\Lambda}+\boldsymbol{\beta})|\,|\mathbf{K}_{MM}|\left|\mathbf{K}_{MM}^{-1}+\mathbf{k}_{MN}(\mathbf{K}_{MM}+\mathbf{k}_{MN}(\mathbf{\Lambda}+\boldsymbol{\beta})^{-1}\mathbf{k}_{NM})^{-1}\mathbf{k}_{NM}\right|, \tag{2.64}$$

and:

$$\mathcal{L}_2 = \mathbf{Z}(\mathbf{x}_N)^{\mathsf{T}}(\mathbf{\Lambda}+\boldsymbol{\beta})^{-1} - (\mathbf{\Lambda}+\boldsymbol{\beta})^{-1}\mathbf{k}_{NM}\mathbf{A}^{-1}\mathbf{k}_{MN}(\mathbf{\Lambda}+\boldsymbol{\beta})^{-1}\mathbf{Z}(\mathbf{x}_N), \tag{2.65}$$

where the main difference is that $\mathbf{\Lambda}$ has a block diagonal structure and requires a larger computational effort to invert.

Computational complexity

By exploiting the Sherman–Morrison–Woodbury identity, the computational complexity of inverting the covariance matrix is reduced when low–rank approximations are applied. The computational complexity of the DTC and FITC approximations is $\mathcal{O}\left(nm^2\right)$ where n is the number of observations and m is the active set size. The computational complexity in this case is

dominated by matrix multiplications. The block diagonal structure chosen determines the computational complexity for the PITC approximation. Tresp (2000b) recommends choosing equal block sizes for observation locations and prediction locations as this ensures the complexity remains $\mathcal{O}\left(nm^2\right)$ as with the DTC and FITC approximations.

Examples

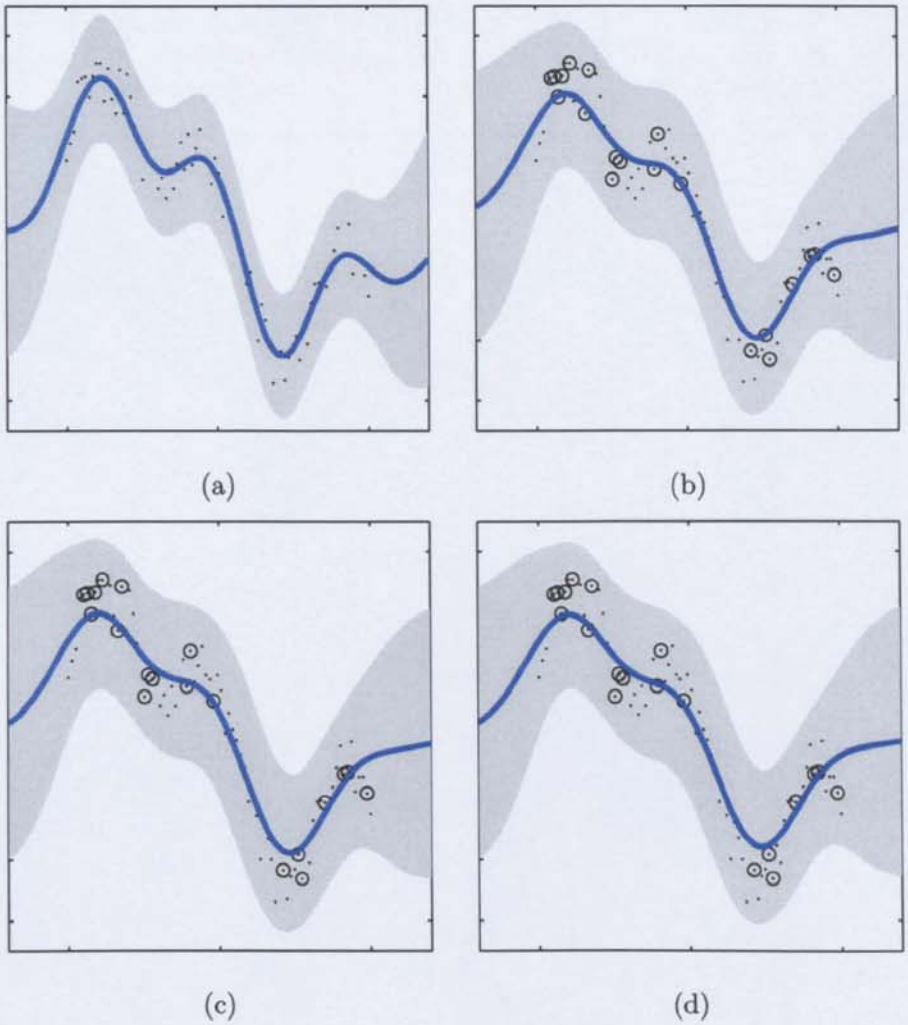
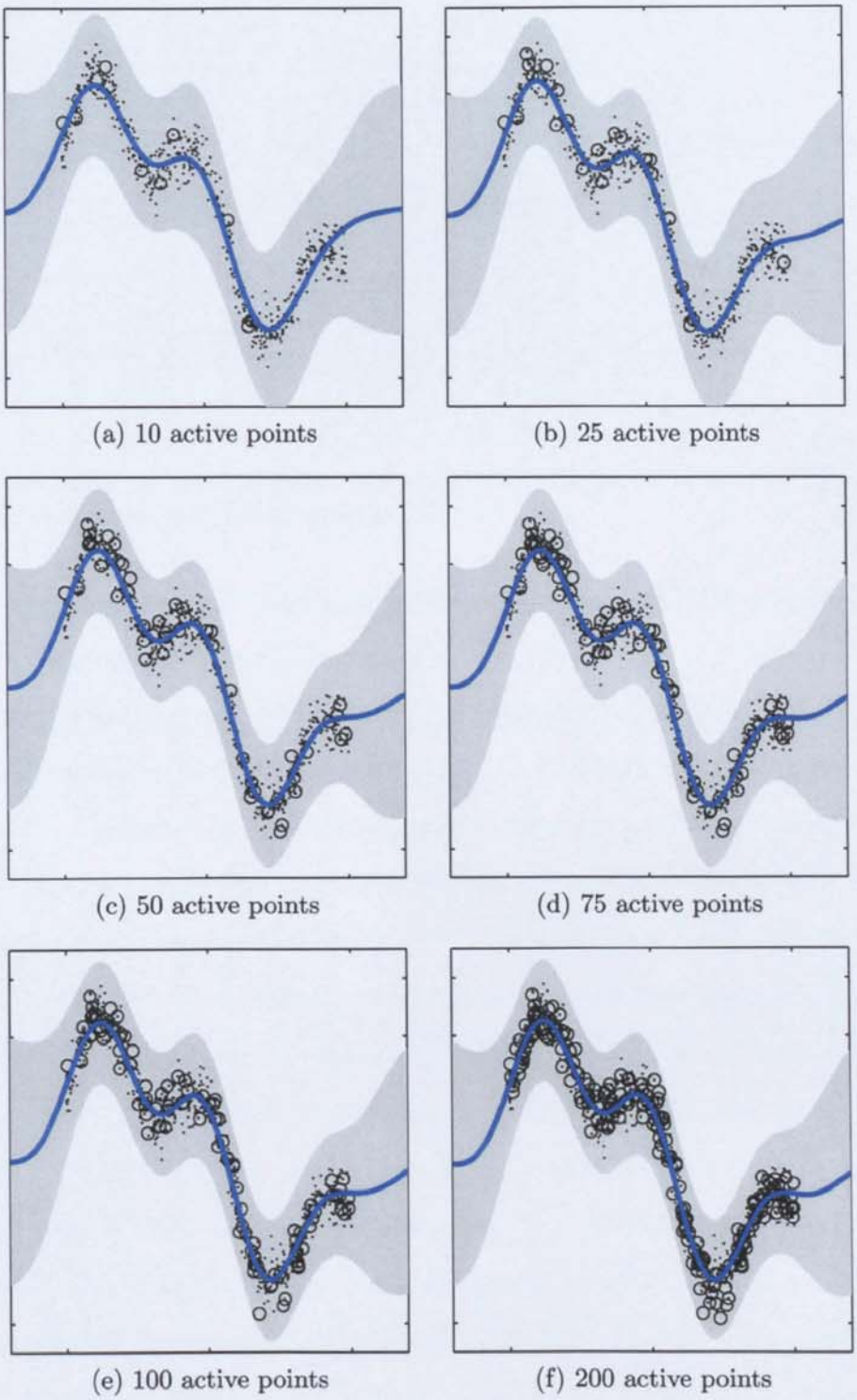


Figure 2.10: Comparison of different projected process algorithms. Mean (thick line) and variance (grey area) predictions of using (a) Simple kriging, (b) Low–rank covariance approximation, DTC or PLV, (c) Low–rank covariance approximation with exact diagonal or FITC and (d) Low–rank covariance approximation with exact block diagonal structure or PITC. Black dots are 80 observations generated from a noisy sinusoidal function. Circled data are the 20 observations that were added to the active set.

Figure 2.10 shows predictions using 20 randomly selected active points for a dataset with 80



(a) 10 active points

(b) 25 active points

(c) 50 active points

(d) 75 active points

(e) 100 active points

(f) 200 active points

Figure 2.11: Comparison of projected process algorithm (DTC) using varying active set sizes. 600 Observations are projected onto randomly selected subsets. Points are observations, circles are active points. The solid line is the mean prediction.

Method	MAE	MSDR	Time (s)
Full GP	0.04785	1.63212	0.00194
DTC	0.20571	1.60217	0.00039
FITC	0.20581	1.53972	0.00057
PITC	0.19215	1.52253	0.00091

Table 2.1: Table showing the prediction accuracy projected process algorithms.

Active Points	MAE	MSDR	Time (s)
10	0.199777	0.116101	0.013067
25	0.103454	0.045204	0.018920
50	0.075286	0.028637	0.029113
75	0.070107	0.026226	0.041495
100	0.067268	0.024959	0.050441
200	0.062648	0.022570	0.109058

Table 2.2: Table showing the prediction accuracy of the projected process algorithm with 600 observations represented by different sized active sets.

observations. The range, sill and noise were fixed with each method. The plots show reasonably consistent mean and variance predication results. Table 2.1 shows that there is little difference in the mean prediction accuracy of the DTC, FITC and PITC methods; it would seem that the low–rank matrix has a greater influence on the mean predictions than retaining exact segments of the covariance matrix. The low–rank approximations tend to smooth more than the full Gaussian process prediction. When considering the variance predictions, the MSDR (Mean Squared Deviation Ratio) shows that the low–rank methods that retain exact segments of the full covariance matrix tend to have a improved variance predictions. For such a simple example, some of the advantages of using the extra block diagonal of the PITC approach may not be evident for mean prediction. To obtain speed estimates for the different methods, computations were repeated thousands of times and the time taken for prediction was averaged. Some argue that the extra computation of the PITC approximation is not worth the extra computational effort and hence the FITC should be preferred (Snelson 2007). Likewise the DTC has been preferred over the FITC approximation due to the reduced computational complexity and little difference in accuracy of predictions (Seeger et al. 2007). To further illustrate the projected process algorithm, Figure 2.11 shows predictions given 600 observations with varying active set sizes using the DTC approximation. The observations have been selected from a simple noisy sinusoidal function. The active set has been selected randomly. By inspecting the plots, a barely noticeable difference can be seen between 10 active points and 25 points in terms of

prediction accuracy of the mean. But beyond that, 25 active points or more, the mean predictions are visually indistinguishable from each other. This shows the strength of the project process method. Table 2.2 shows the MAE steadily decreasing as the number of active points is increased. Likewise, the MSDR decreases with more active points. Once the effective complexity of the data has been sufficiently well represented by the active set, increasing the active set size adds little in terms of prediction accuracy. Whether the dataset shown in Figure 2.11 had 100 or 1000 observations, the projected process method requires the same number of active points to represent the process (within a tolerance level). Active points are only needed to represent the complexity of the model, or in other words, the number of active points needed is closely related to the range or the lengthscale of the process being modelled.

2.5.2 Sequential algorithms

Having discussed batch algorithms for projected process algorithms, in this section attention is turned to sequential algorithms. There are a number of reasons that a sequential algorithm is appealing in this context. For large datasets, it could become prohibitively expensive to store all the data on a computer system at one time. The sequential algorithm only needs to see one observation at a time. Secondly, by using a sequential algorithm, arbitrary likelihood functions can be used. Typically the assumption that the data is distributed in a manner that follows a Gaussian distribution does not always hold. Box–Cox transformations are often used in geostatistics (Cressie and Hawkins 1980) to ensure the observations have a log–normal distribution (Box and Cox 1964). Appendix B.1.1 examines non-Gaussian likelihoods in more detail and shows how sequential projected process algorithms can be used with arbitrary likelihoods.

Sequential projected process kriging

The sequential projected process kriging algorithm makes a refinement to the sequential kriging algorithm. Instead of increasing the size of the covariance matrix with each new observation, the SPPK algorithm calculates how informative each new observation is. This measure is then used to decide whether an observation is sufficiently important to warrant being added to the active set (or covariance matrix) as described by the sequential kriging algorithm. If the observation adds little or no new information then the observation effect can instead be projected onto the covariance matrix without having to increase the size of the covariance matrix. This process is repeated until all observations have been considered. The term *projected* is used in this context because the model representation uses a smaller number of observations (the active set) to represent all the observations in the dataset.

It will now be shown how to reduce the complexity of the algorithm, while retaining important features in the data. This is done in the framework of a sequential algorithm. At each iteration of the algorithm, s, an observation is added to the model, or active set, until a maximum number of observations (ie. the model complexity, or active set size m, that is desired) is reached. It is possible to update the model exactly with a new input, without increasing the size of the active

set if

$$k_{N(s+1)} = \sum_{i=1}^{s} \lambda_{s+1}(i) k_{Ni} \tag{2.66}$$

holds for all $\mathbf{x}$ (Csató and Opper 2001a). In such a case the model can be updated exactly without increasing the complexity. If Equation (2.66) does not hold then an approximation is made by minimising the Kullback–Leibler (KL) divergence measure (Kullback and Leibler 1951). This works by effectively finding a lower rank kriging predictive distribution that is most similar to the kriging predictive distribution at $s+1$. The details of this will be avoided in this thesis, a full derivation can be found in Csató and Opper (2001b).

Rewriting the predictive mean and variance equations for step $s+1$ in the algorithm as:

$$\hat{Z}(\mathbf{x}_{s+1}) = \vec{\boldsymbol{\alpha}} k_{N(s+1)}, \tag{2.67}$$

and:

$$\sigma^2_{s+1}(\mathbf{x}_{s+1}) = k_{**} - \mathbf{k}_{NM}^{\mathsf{T}} \vec{\boldsymbol{\Sigma}}^{-1} \mathbf{k}_{NM}, \tag{2.68}$$

gives an alternative parametrisation of the kriging equations. Notice that the projection operator notation, $\rightarrow$, has been used to refer to the parameters of the projected model to distinguish them from the parameters of the traditional kriging model. The difference between the two inverse covariance matrices in equations (2.11) and (2.68) is that $\boldsymbol{\Sigma}^{-1}$ is the inverse covariance matrix between the observations in the active set only. $\vec{\boldsymbol{\Sigma}}^{-1}$ is also the inverse covariance matrix between the observations in the active set, however, additionally the effect of the inactive observations has been projected on to this matrix. The notation $\boldsymbol{\alpha} = \vec{\boldsymbol{\Sigma}}^{-1} Z(\mathbf{x}_M)$ is introduced where $Z(\mathbf{x}_M)$ corresponds to the observations in the active set *prior to any projection*. However, after observation projections, $\vec{\boldsymbol{\alpha}} \neq \vec{\boldsymbol{\Sigma}}^{-1} Z(\mathbf{x}_M)$. Thus $\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\Sigma}}^{-1}$ parameterise the projected process kriging equations.

In the case where a residual error occurs in trying to satisfy Equation (2.66), the informitiveness of the new observation has to be calculated. The goal is to select the active set so that prediction error is minimised. A simple scoring heuristic is used to score the active points to determine which active point location is least informative with respect to the kriging predictive distribution. A number of scoring methods have been proposed which measure scores such as relative entropy gain (Lawrence and Herbrich 2001) and the information gain criterion (Seeger and Williams 2003). Upon deciding whether the new observation is sufficiently informative,

either it is added to the active set and the least informative point from the active set is removed, or alternatively if it is not sufficiently informative, the effect of this new observation is projected onto the active points in the active set.

The projection step that updates $\vec{\alpha}$ requires calculating a vector of weight innovations Γ for each new observation that is to be projected onto the existing active set:

$$\Gamma = \lambda - \left(\vec{\Sigma}^{-1}\mathbf{k}\right), \tag{2.69}$$

where $\lambda = \Sigma^{-1}\mathbf{k}$ which is the weight of the active set locations with respect to the new observation location. Γ essentially computes the difference in weights between the traditional kriging weights and the projected process kriging weights as each observation is incorporated. The update equations for the model are now:

$$\vec{\alpha}_{s+1} = \vec{\alpha}_s + q_{s+1}\Gamma, \tag{2.70}$$

$$\vec{\Sigma}^{-1}_{s+1} = \vec{\Sigma}^{-1}_s + r_{s+1}\Gamma\Gamma^T, \tag{2.71}$$

where for a Gaussian noise model on the observations, the projection parameters are:

$$q_{s+1} = \frac{\hat{Z}(\mathbf{x}_{s+1}) - Z(\mathbf{x}_{s+1})}{\sigma^2_{s+1}}, \tag{2.72}$$

which measures the scaled difference, at the currently processed location ($\mathbf{x}_{s+1}$), between the model prediction after s iterations and $\hat{Z}(\mathbf{x}_{s+1})$ (ie. the previous iteration and given an *active set* $\mathcal{D}_M$) and the observed value at the location $Z(\mathbf{x}_{s+1})$, and:

$$r_{s+1} = \frac{1}{\sigma^2_{s+1}} \tag{2.73}$$

which correctly scales the weights for the covariance updates. There are a number of things to note. First, the update parameters are mentioned. In this thesis it has been assumed that the likelihood function is Gaussian although the update equations can be derived for arbitrary noise/likelihood models (Csató and Opper 2002). These update equations were calculated by minimising the KL divergence between the true model, Σ_s, and the approximating model, $\vec{\Sigma}_s$ at each iteration s. Secondly, the reader should note that the updates for Equation (2.71) are similar to the updates shown by the partitioned matrix in Equation (2.28) but do not increase the size of the matrix.

The process of adding new observations to the model representation has been discussed. A new observation can be added to the model by increasing the model complexity or by projecting its effect onto the representative subset of the data. A further feature of this method is the removal of active points from the active set. This is basically the reverse process of adding active points. The reader is directed to Csató (2002) where full derivations can be found. To optimally apply this algorithm in practice, it has proven useful to discourage the use of active point deletion. A better approach is to select the active set *a priori* (selecting the most informative locations rather than randomly) and then projecting the inactive observations onto this set. Experiments have shown that a dynamic active set can lead to active point flip–flop behaviour whereby newly inserted active points are removed on the next iteration.

SPPK example

To understand the iterative projected process methods it will be helpful to refer back to the simple description of kriging given earlier in Section 2.2. Predictions with kriging are simply a weighted sum of all the observed data. Typically, weights are assigned to each observation based on the covariance function weighted distance from the prediction location (cross) as shown in Figure 2.12(a).

The next step is to see if the computational complexity of the kriging algorithm can be controlled but at the same time allowing important features in the dataset to be retained. This is done by calculating a score for each of the observations based on its informativeness to the kriging predictive process.

Next, the size of the dataset is reduced by removing the observation from the active set that is calculated to have the least impact on the predictive distribution, based on some heuristic, and then it is added to the inactive set. Figure 2.12(b) shows the scorings of each active point. The remaining weights are updated by projecting the influence of the removed observation (Figure 2.12(c)) so as to minimise the effect of removing an observation. It may be surprising, but this projection can be performed efficiently with minimal loss of information.

Figures 2.12(a-l) shows how a dataset of 8 observations is sequentially reduced in size and a contour map given the current state of the model. The first column shows the weights of each observation with respect to the prediction location (cross) in the centre. The second column

shows the scores of each of the observations with respect to some measure of informativeness that the observation provides the kriging predictive distribution. The circled observation has the lowest score and will be deleted from the active set shown in the third column. The third column shows how the weights are distributed (or projected) to the other observations or active points in the dataset.

2.6 Pseudo inputs

Learning the optimal active set is a major issue for improving prediction performance. Covariance parameter estimation and active set selection are usually interleaved a number of times. An alternative idea which requires that the size of the active set is chosen *a priori* is to optimise the active point locations jointly with the hyper–parameters. Unlike active set selection, which has already been discussed, these observation locations do not necessarily correspond to any subset of the data. Instead of adding and removing observations from the active set or updating just the weights on each active point, the algorithm searches for optimal positionings for the active points and optimal weights using a non–linear optimiser (Snelson and Ghahramani 2006). In this case, active points no longer need to be selected from a subset of the dataset.

A major drawback of this algorithm is the computational time; for a small number of sample locations, optimisation can take a significant period of time. Seeger et al. (2006) state that there are some additional problems such as the algorithm tending to find local minima and resulting in sub-optimal covariance parameters. Additionally, it is unclear what each active point represents or how it should be interpreted by a practitioner since the active points are not likely to be a subset of the observations and could effectively be located in a location disjoint from the dataset.

2.7 Other Reduced rank matrix methods

Some other methods for handling large datasets have been introduced. Each of these methods results in a reduction in the size of the covariance matrix Σ which is the cause of the major bottleneck in kriging type algorithms. The explanations of what Σ represents enables the practitioner to see the connection between the observations, the approximation and the predictions. Some alternative methods do not always retain this connection explicitly and hence it is unclear

how the model can be defined in a geostatistical context. Generally, these methods require that the covariance matrix $\boldsymbol{\Sigma}$ is approximated by a lower rank matrix $\tilde{\boldsymbol{\Sigma}}$ as with the techniques discussed in Section 2.5 and hence leads to a cheaper matrix inversion. Their inclusion here is for completeness.

2.7.1 Nyström method

The Nyström method replaces the covariance matrix $\boldsymbol{\Sigma}$ used in the predictive mean and variance Equations (2.10) and (2.11) by a lower rank matrix $\tilde{\boldsymbol{\Sigma}}$. This idea was first suggested in Williams and Seeger (2001) and they called this the Nyström method for approximate Gaussian process regression. This does not solve any of the complexity issues since an eigen–decomposition is required to make the approximation to the matrix $\tilde{\boldsymbol{\Sigma}}$ and this is an $\mathcal{O}(n^3)$ operation, although it should be noted that calculating an eigen–decomposition is only needed once with this method.

2.7.2 Fixed rank kriging

Recent work by Cressie (2006) has looked at using reduced rank matrix approximations with satellite dataset which tends to be massive. He calls this technique fixed rank kriging since it requires that only a matrix of fixed rank need to be inverted. First, a covariance matrix approximation is presented in the form:

$$\hat{\boldsymbol{\Sigma}} = \mathbf{k}_{NM}\mathbf{K}_{MM}\mathbf{k}_{MN} + \sigma^2\mathbf{V} \tag{2.74}$$

where $\mathbf{K}$ is a $m \times m$ matrix and $\mathbf{V}$ is a diagonal matrix with the measurement–error variances for each observation. The Frobenius norm (Appendix C.3) between the fixed rank covariance matrix $\hat{\boldsymbol{\Sigma}}$ and the covariance matrix of the data is minimised to give an approximate covariance. This is then used in prediction on a massive dataset.

Examining the FITC covariance matrix approximation discussed in Section 2.5.1, shows some connections to this method. Details of $\mathbf{V}$ are not clear, only that it represents the measurement–error variances for each observation. There is no indication whether the diagonal of $\hat{\boldsymbol{\Sigma}}$ has the exact measurement–error variances which would seem like a better approximation for this model, which would mean that $\mathbf{V} = \mathrm{diag}(\mathbf{k}_{NM}\mathbf{K}_{MM}\mathbf{k}_{MN} - \hat{\mathbf{K}}_{NN})$ rather than $\mathbf{V} = \mathrm{diag}(\mathbf{k}_{NN})$ which is what is hinted at in the paper. The full covariance matrix for all the data with the measurement–error variances is denoted by $\hat{\mathbf{K}}_{NN}$. Hence it is possible to choose $\mathbf{V}$ in such a

way so that this method is equivalent to the FITC approximation. Here, the diagonal $\mathbf{V}$ relates to the error characteristics of the sensor used to collect the data. It would be interesting to investigate clustering the dataset and using a block diagonal assumption (PITC approximation) for the covariance matrix.

There is also no indication of how the active set for constructing the low–rank matrix approximation was selected. The whole subject of active set selection is completely avoided by Cressie (2006).

2.8 Summary

This chapter has presented methods recently developed by the machine learning community in the context of geostatistics. Projected process kriging techniques were introduced in the context of approximating the covariance matrix by a low–rank matrix. By selecting a representative subset of the data as an active set, it was shown how a low–rank covariance matrix approximation is constructed using all of the data. The problematic selecting of the representative subset or active set was approached by using sequential algorithms rather than commonly used batch algorithms. Instead of distributing active points over a grid of locations, the sequential algorithm can determine regions of high spatial variability where more active points are required to more accurately represent the variability in the data being modelled. Projected process kriging has been shown to be effective when the dataset is densely sampled, or when the range of variation is large with respect to the area of study.

Geostatistics has traditionally relied on method–of–moment estimators for determining model parameters. Likelihood–based techniques are beginning to become more popular. It has been shown how low–rank matrix approximations for the likelihood can be exploited for efficiently determining the model parameters given a dataset. Again, active set selection is crucial to this obtaining accurate estimates with this procedure. As with prediction, the strengths of this method are for datasets which are densely sampled or where the range of variation is large with respect to the area of study.

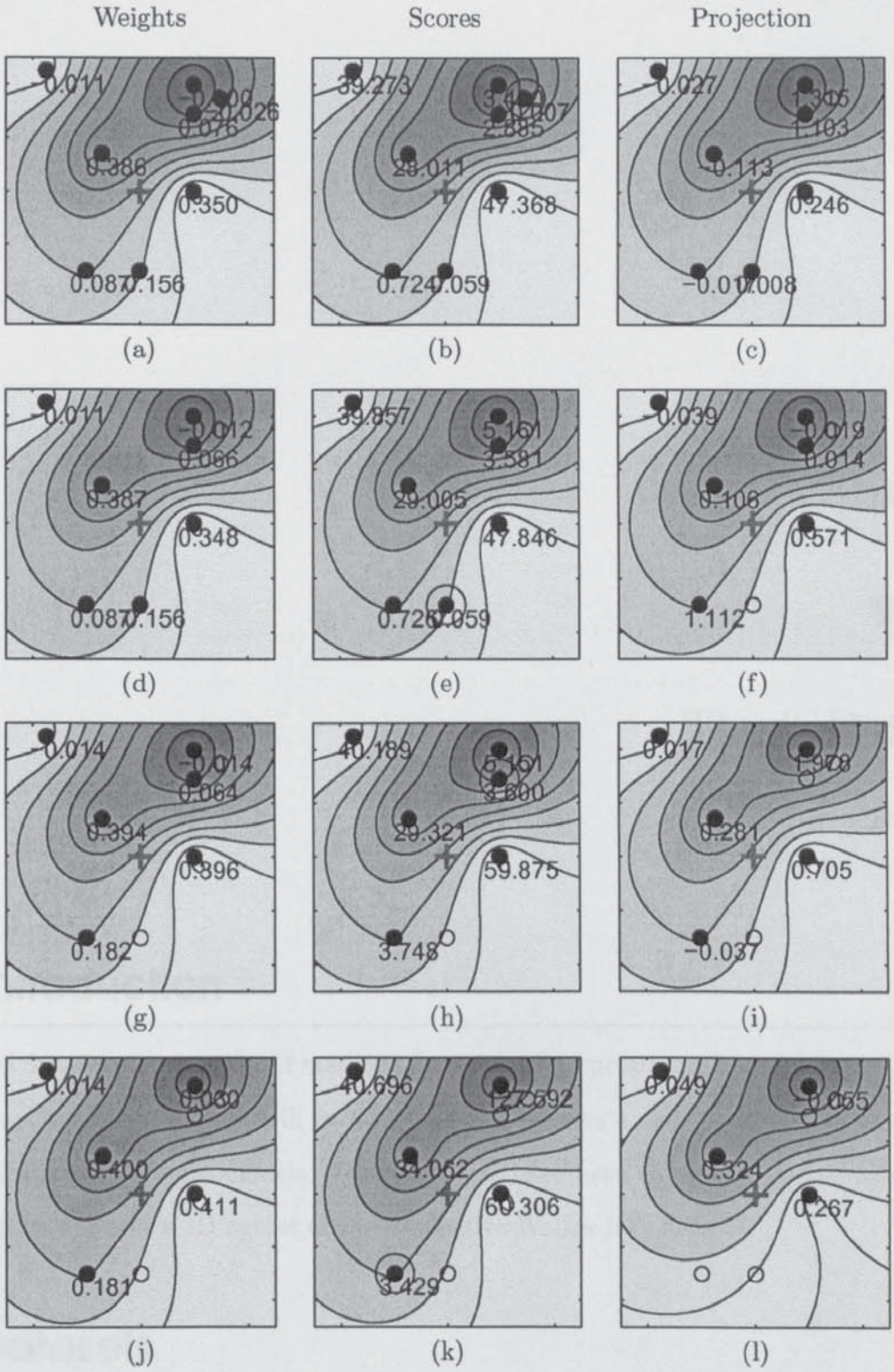


Figure 2.12: An example of iteratively projecting the projected process kriging weights for predicting at the cross, using 2D data. Each iteration (rows) shows firstly the kriging weights (1st column) given the data (filled dots). Each observation is scored (2nd column) based on its importance for representing the kriging model, the lowest scoring observation is circled. The 3rd column shows the effective projection of the weight removal (circled observation in the 2nd column) onto the remaining observations. (a-c) iteration 1, (d-f) iteration 2, (g-i) iteration 3, (j-k) iteration 4. The background contour maps show the prediction for a particular iteration

3

Thesis Datasets

3.1 Introduction

This thesis focuses on a number of methods for spatial interpolation. To demonstrate, compare and contrast the methods that will be described in Chapters 4 and 6, consistent datasets will be used across the different models. Three datasets have been chosen: a 1D synthetic data, a 2D synthetic data and a 2D subset of the exhaustive Walker lake dataset.

3.2 Datasets

The problems associated with model comparison studies based on a single dataset have been taken into account. It is common that the result of comparative studies is to give the practitioner a greater understanding of the data rather than the models being used. Additionally, the generalisation of these models when applied to other datasets, whether they be similar or

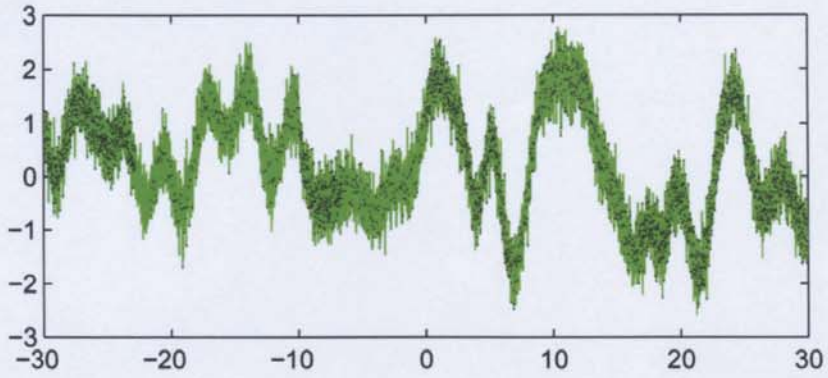


Figure 3.1: Graph showing the 1D Synthetic dataset and the observations sampled from the simulation. The light continuous line is the simulated function. The dots indicate where observations were sampled from.

significantly different, is often poor. The results across different datasets can often be difficult to interpret and can appear contradictory (Cornford 2005). The two synthetic datasets that have been created in this thesis have been chosen with the aim of demonstrating the properties of the many methods discussed. A 1D synthetic dataset was chosen to help the reader gain a greater intuition of the various algorithms developed in this thesis, through pictorial representation. The maximum dimensionality of the datasets that will be used in this thesis is 2D. Since a primary aim of this thesis is the application of interpolation algorithms to large datasets, a subset of a real–world large dataset is included.

One issue arising from attempting to describe the reasons for choosing a particular dataset is that often an explanation of the methods that are to be applied to the dataset is needed to justify the selection. A detailed explanation of the methods that will are applied is avoided in this chapter, instead the essential requirements of the data will be described. The basic properties of the datasets is to have regions of densely sampled data and regions of sparsely sampled data.

The synthetic datasets were generated using the Turning Bands method of simulation (Journel 1974). A Gaussian covariance function was chosen for the simulations and 1000 bands were used.

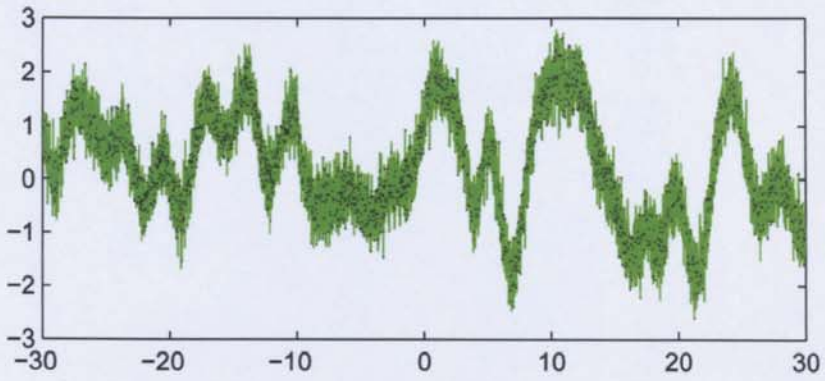


Figure 3.2: Graph showing the 1D Synthetic dataset and the prediction locations selected for cross–validation. The light continuous line is the simulated function. The dots indicate prediction locations.

3.2.1 1D Synthetic data

Figure 3.1 shows the data simulated using the Turning Bands method. The covariance parameters are fixed at a range of 1m, a sill of 1m and a nugget of 0.1m. This dataset has 3000 observations for learning the covariance structure and a further 2000 observations that will be used for cross–validation. The units used for the simulation are arbitrary, but here the measure of a metre was chosen. The observations that were sub–sampled from the simulated data were sub–sampled randomly with regions of differing densities. The regions with different densities were generated randomly. In a similar way, Figure 3.2 shows the underlying function and the prediction locations. The prediction locations were sampled across the whole dataset with the same density.

3.2.2 2D Synthetic data

For the 2D synthetic dataset, a thorough explanation of the reasons for its particular structure is not presented here. The main design feature is to have regions with observations sampled with different densities. Again the Turning Bands simulation method was used to generate the dataset. And again, the covariance parameters are fixed at a range of 1m, a sill of 1m and a nugget of 0.1m. 4000 observations were used for learning the covariance structure and then 2000 observations were used for cross–validation purposes. Figure 3.3 shows an image generated from this model that shows the variation in the dataset.

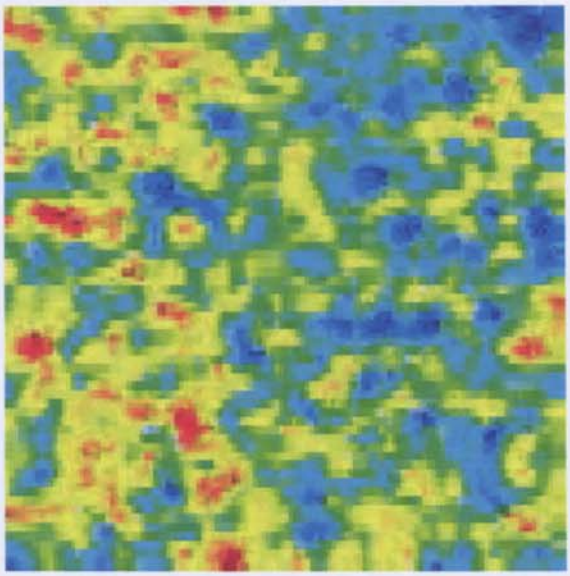


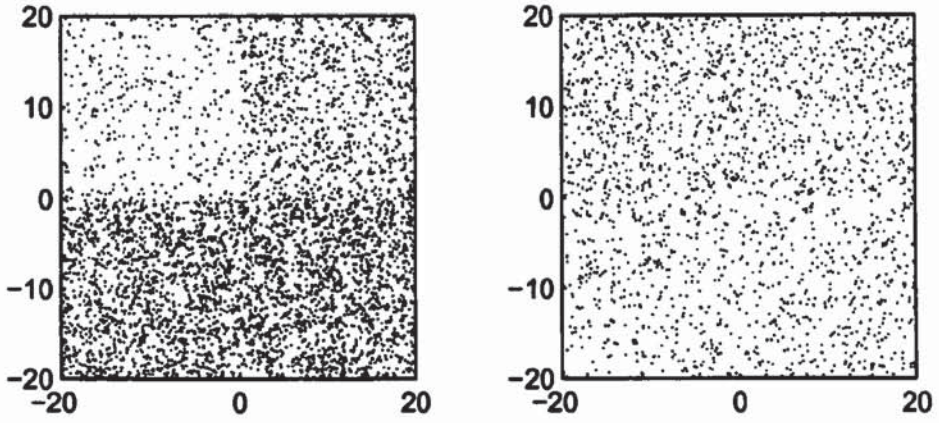
Figure 3.3: Map showing structure of 2D synthetic dataset.

Figure 3.4 shows the locations where observations are located. Differing densities of the observations are evident in the figure. Figure 3.4 shows the locations that will be used for cross–validation.

3.3 2D Walker lake subset

Demonstrating the methods developed in this thesis using synthetic datasets is suited to illustrating and understanding performance of the models on a synthetic dataset, however it is important that the developed models perform well with real–world data. Often with real–world datasets the underlying process to be modelled in the data is not fully understood and is almost certainly not a Gaussian process. It can also be unclear what assumptions should be made. There are many datasets that have been thoroughly analysed in a number of publications. Knowing the spatial properties of the dataset will be advantageous when comparing the models in this thesis with work that has already been undertaken. One thoroughly examined dataset with many years of investigation is the Walker lake dataset (Isaaks and Srivastava 1989). A further advantage is that it is a large exhaustive dataset with 78,000 observations.

These data were sampled from the Walker Lake area in the western United States, Nevada. This exhaustive dataset consists of three variables measured at each of 78,000 locations on a



(a) (b)

Figure 3.4: Locations of 2D synthetic dataset (a) observations, (b) prediction locations.

260×300 rectangular grid. A subset of this dataset was used in the book Applied Geostatistics (Isaaks and Srivastava 1989), however for the purposes of this thesis, the subset is too small. A major focus of this thesis is the use of large datasets, so rather than the 470 points as selected in Applied Geostatistics, a subset of 4571 observations will be used instead. The 4571 observations are randomly selected from the exhaustive Walker lake dataset as shown in Figure 3.6. The sampling density changes from region to region, clusters of different sampling densities are evident. Figure 3.6 shows the prediction locations that will be used for cross–validation.

As noted, the Walker lake dataset is densely sampled and hence this is a major advantage for the demonstration of the methods in this thesis. Three variables are recorded in the Walker lake dataset, for this thesis, only one of the variables will be considered.

The appendices in Isaaks and Srivastava (1989) give a complete description of the data and how it was collected. Here, in this thesis it is an aim to show that large datasets can be treated in a principled way in a reasonable amount of time ie. seconds and minutes rather than days and weeks. Hence an in-depth discussion of the dataset will not be given here, although in later chapters comparisons with the methods used in the available literature will made.

The data is assigned the units metres, although the real sampling units are much larger. For continuity, it will be assumed that the grid is also measured in metres for the experiments in this thesis.

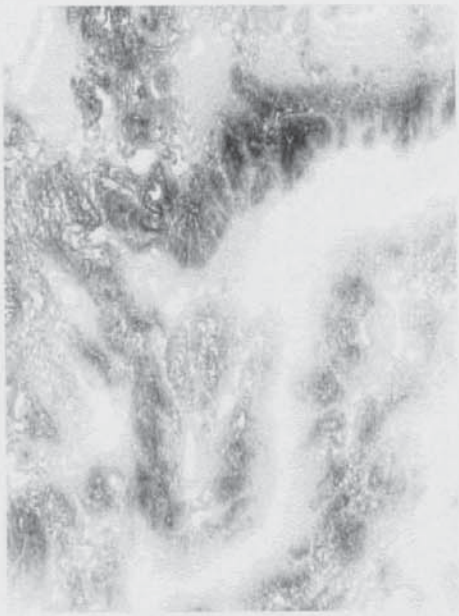


Figure 3.5: Map showing structure of 2D Walker lake dataset.

3.4 Computer hardware

In a thesis that compares algorithms across different computer architectures, maintaining equality for comparisons is challenging. All but the parallel experiments were carried out on a single core Pentium 4 2.4 Ghz. The parallel processing programs were carried out on an 8 node cluster, each node being an AMD Opteron running at 2 Ghz.

Octave was used for parallel processing algorithms and Matlab was used for all other algorithms. Octave[1] was used due to its unrestricted public license. One of the limitations of distributed Matlab[2] usage is the need for a separate license for each Matlab process spawned. The same version of ATLAS (Whaley et al. 2001) providing optimised BLAS (Lawson et al. 1979) and LAPACK (Anderson et al. 1990) routines was installed on each computer. The ATLAS extensions for dual–core processors available in recent versions were disabled.

[1] `http://www.gnu.org/software/octave`
[2] `http://www.mathworks.com`

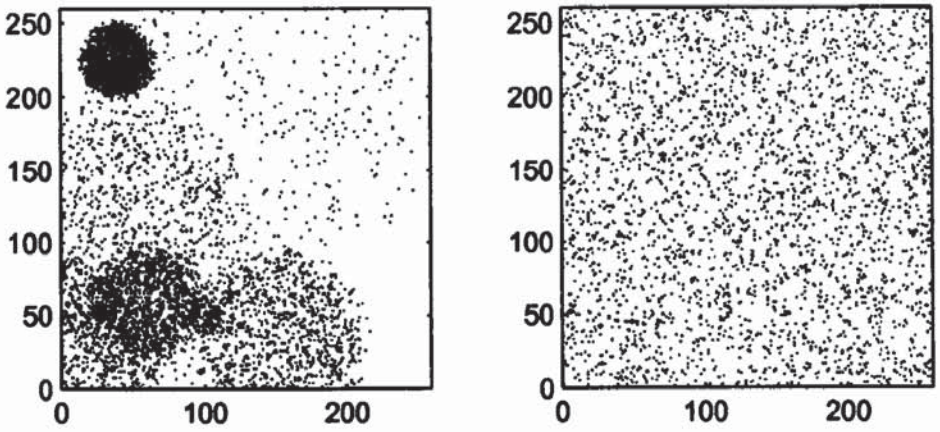


Figure 3.6: Locations of 2D Walker lake dataset (a) observations, (b) prediction locations.

4

Space–limited Covariance Functions

4.1 Introduction

Covariance functions have already been extensively mentioned. In geostatistics it is accepted that selecting an appropriate covariance model and associated parameters is a crucial step in achieving optimal results with kriging because assumptions about the spatial variation of the process are encoded in the covariance function (Wackernagel 2003). Variogram analysis is an important activity prior to prediction with kriging which has already been mentioned in Chapter 2. Once choices about the characteristics of the variation in a spatial process have been made, kriging can be performed. This being said, Isaaks and Srivastava (1989) list a small subset of variogram models that enable a satisfactory fit to all the sample variograms that a practitioner is likely to encounter, while also satisfying the positive definiteness condition.

In the machine learning community however, there is a tendency to approach covariance[1]

[1]Typically the machine learning community refer to covariance functions as kernels.

selection in a slightly different manner, such that selecting a covariance function is the process of using experience and intuition rather than using an analytical method to select the covariance function (Seeger 2004). Surveying the machine learning literature shows that only a very small family of covariance functions are used. The Gaussian, squared exponential or RBF[2] kernel is by far the most commonly used kernel which is unfortunate since it is too smooth for any realistic geostatistical process (Stein 1999; El-Shaarawi and Piegorsch 2002). Space–limited covariance functions were investigated by Hamers et al. (2002) in the context of solving machine learning problems but they arrived at the conclusion that they have little use for datasets of high dimensionality which are commonly found in datasets used in machine learning.

In this Chapter space–limited covariance functions will be exploited to provide sparse matrices. These sparse matrices will be exploited by the application of fast space matrix methods. These methods will then be combined with the projected process method discussed in Chapter 2.

4.2 Covariance functions

In this Section, general properties of covariance functions are discussed. Methods for the construction of space–limited covariance functions are introduced as these have important properties which will be exploited later in this chapter.

4.2.1 The Semi–variogram

Using an empirical (method–of–moments) method such as the empirical sample semi–variogram to determine the form of the covariance function is common in geostatistics. Sample semi–variograms provide a convenient way to visualise the covariance function. However, the widespread use has been limited to the geostatistics community. In an entry to SIC2004[3] a machine learning Gaussian process approach used the semi–variogram to estimate an appropriate covariance function (Ingram et al. 2005). It was shown that selecting an appropriate covariance for the data was linked to obtaining better prediction performance. In geostatistics, selecting an appropriate covariance function is accepted as a method for obtaining optimal prediction performance (Wackernagel 2003). The empirical semi–variogram can be helpful in identifying structures in spatial

[2]Radial basis function.
[3]Spatial Interpolation Comparison 2004 Exercise.

data. Care should be taken however, because it does not directly provide a technique for selecting a covariance function since it can be misleading for data that do not follow a Gaussian distribution (Stein 1999).

Throughout this Chapter, the notation s will be used to refer to the absolute separation distance, $s = ||x_1 - x_2||$, between two observations since only stationary covariance functions will be discussed. The range parameter will be denoted by l.

4.2.2 Trend or covariance?

As stated in Equation (2.2), the assumption will be made that the data is represented by decomposing the model into two components:

$$Z(\mathbf{x}) = m(\mathbf{x})^T \boldsymbol{\beta} + \epsilon(\mathbf{x}), \tag{4.1}$$

where $m(\mathbf{x})^T \boldsymbol{\beta}$ is the deterministic mean function that will be referred to as the large-scale variation or trend and where $\epsilon(\mathbf{x})$ is a correlated error process.

Without making specific assumptions a clear distinction between the mean function $m(\mathbf{x})^T \boldsymbol{\beta}$ and the stochastic process $\epsilon(\mathbf{x})$ is not possible. In modelling the mean function or trend, additional empirical information can be utilised to determine an appropriate model. It is a difficult problem to model the deterministic mean component without also modelling part of the underlying stochastic process. If more of the variability is described by the deterministic component the importance of correct covariance selection is reduced. As more and more spatial variability is modelled by the deterministic component the covariance tends to a pure nugget effect (Cornford 1996). The problems of modelling trend are well known and it has been stated that it is not often worth modelling anything more complex than a linear trend model (Diggle et al. 1998). The problem of fitting the trend is noted here as a continuing problem and it will not be discussed in detail. For the purposes of this thesis, the trend of the data is assumed to be known.

4.2.3 Differentiability

The degree of differentiability is a property of a spatial process. The more smooth the process, the higher the degree of differentiability of the covariance function at the origin (Abrahamsen 1997; Diggle et al. 1998). Understanding the differentiability of a process is crucial for selecting

an appropriate covariance function. It is common that covariance functions are selected based on the goodness of fit to the variogram, but it is often difficult to observe the properties of the variogram clearly near the origin. This is unfortunate since the behaviour of the covariance function at the origin dominates the resultant interpolation more than any other aspect of the covariance function (Abrahamsen 1997; Stein 1999) (assuming that prediction location is not further away than the smallest lag separation).

4.2.4 Nested covariance functions

Nested covariance functions consist of summing two or more covariance functions (Goovaerts et al. 1997; Wackernagel 2003; Journel and Huijbregts 1978). One example of this is advocated in Ingram et al. (2005) whereby a linear combination of a Gaussian covariance function and a exponential covariance function (with different lengthscale parameters) were used to model short range and long range variation respectively which were evident in background ambient gamma radiation data. The use of the nested model showed small, but not insignificant reductions in cross–validation error when compared with many commonly used covariance models, but at the expense of having to estimate an additional parameter and hence requiring more computation. Appendix C.1 lists other valid nested covariance models.

In the literature there are concerns about using nested covariance models; what matters most is fitting a covariance function which supports the empirical variogram near the origin (Adler 1981). Stein (1999) suggests that nested models should be avoided since there is little hope of estimating the parameters of such models with certainty for the size of datasets usually analysed. Additionally, with likelihood–based approaches there can be convergence problems due to identifiability of parameters. He does not discuss their application to large datasets where sufficient data could reduce parameter estimation uncertainty.

4.2.5 Space–limited covariance functions

Isaaks and Srivastava (1989) suggest that the process of creating new covariance functions is not worth the effort associated with verifying positive definitiveness since there are enough functions that provide a satisfactory fit to most sample variograms that are likely to be encountered. In what follows, a motivation for constructing additional covariance functions is discussed.

An often desirable property of covariance functions is that the covariance function decays to zero beyond a certain cut–off distance. Realistically in many spatial models the spatial correlations vanish beyond a given separation distance. This property of covariance functions is referred to by a number of terms: 'compactly supported', 'space–limited', 'transitive' or 'bounded'. That being the case why are functions, defined over an infinite range, such as the Gaussian, exponential and rational quadratic covariance functions commonly used? One answer might be that it seems to be that there are few flexible space–limited models other than the often used spherical function given by:

$$k(s) = \begin{cases} 1 - \frac{3}{2}\frac{s}{l} + \frac{1}{2}\left(\frac{s}{l}\right)^3 & 0 \leqslant s \leqslant l \\ 0 & s > l \end{cases} \quad \textit{positive definite in } \mathbf{R}^3, \tag{4.2}$$

and the circular function given by:

$$k(s) = \begin{cases} \frac{2}{\pi}\arccos\left(\frac{s}{l}\right) - \frac{2}{\pi}\frac{s}{l}\sqrt{1-\left(\frac{s}{l}\right)^2} & 0 \leqslant s \leqslant l \\ 0 & s > l \end{cases} \quad \textit{positive definite in } \mathbf{R}^2. \tag{4.3}$$

Throughout this chapter the advantages of using a space–limited covariance function will be presented and the associated effort required to define one are shown to be minimal.

4.2.6 Constructing space–limited covariance functions

Constructing a covariance function that is valid is not a trivial matter. One simply cannot replace entries in the covariance matrix that fall below a certain threshold value with a zero. Neither can a valid covariance function be constructed by simply "chopping off" the function beyond a certain threshold distance (Gaspari and Cohn 1999).

Due to the precision of the machine representation of small numbers, many covariance functions defined for infinite separation will yield covariance matrices with apparent sparsity; however this can lead to some numerical stability problems and is another disadvantage of covariance functions defined over infinite separation distances.

One simple way to construct a space–limited covariance function and at the same time maintain positive definitiveness is to simply multiply the covariance function $k(\cdot)$ by a space–limited covariance function $k_c(\cdot)$ (Wendland 1995; Genton 2001). The idea of calculating the product of a covariance function with a space–limited covariance function is also called covariance tapering where the space–limited covariance is called the tapering function (Furrer et al. 2006).

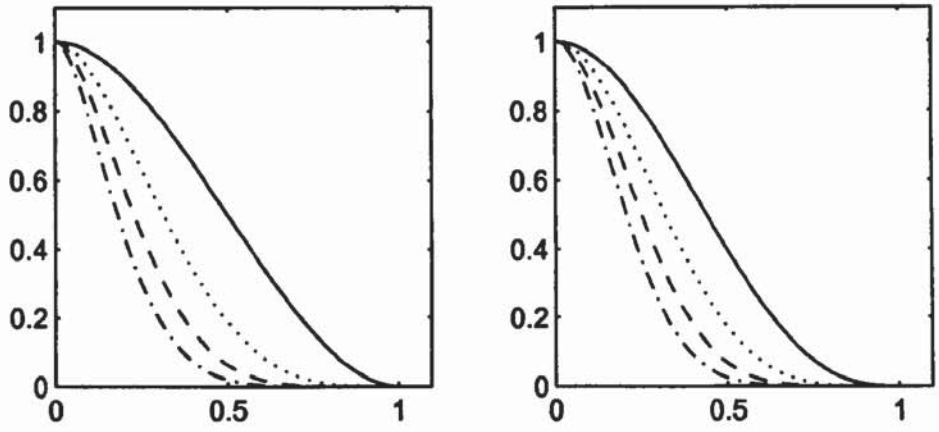
This product of two covariance matrices is called the Schur product and is explained briefly in Appendix C.2. The Schur product has be used frequently in atmospheric sciences to create space–limited covariance functions (Houtekamer and Mitchell 2001; Palmer and Hagedorn 2006). The taper function used by Furrer et al. (2006) is the popular spherical covariance function which is used to taper the flexible Matérn covariance.

Even if the assumptions made about a given covariance function do not indicate that a space–limited covariance is an appropriate model (eg. a large range parameter with respect to the overall region size), the space–limited model can still be exploited for computational purposes. Covariance tapering where long range correlations exist can still be justified by considering the *screening effect* in kriging (Stein 2002). Wackernagel (2003) also shows that even for processes where long range correlations exist, as the lag distance increases, the kriging weights rapidly decay to zero.

It has been mentioned earlier that in the majority of practical kriging implementations a slight modification has been made to the algorithm. Instead of solving a large matrix of all the observations in a dataset which scales cubicly, many smaller matrices are solved. These smaller matrices are typically constructed by selecting a search neighbourhood of the nearest n observations or by selecting all observations within a search radius r. There is a relationship between using a moving window for kriging and covariance tapering. By selecting a taper function,$k_c(\cdot)$, with a short range parameter, an equivalence with moving window kriging was noted by Furrer et al. (2006). Much depends on the taper used. If a step function style taper of the form:

$$k_c(s) = \begin{cases} 1 & \leqslant s \leqslant l \\ 0 & s > l \end{cases} \tag{4.4}$$

were used, that is to say that a hard cut off was induced in the covariance matrix, then the equivalence to moving window kriging, at a prediction location, would be exact. However, since the tapers discussed by Furrer et al. (2006) are more sophisticated, the relationship to moving window kriging is more complicated.



(a) (b)

Figure 4.1: Wendland construction covariance function using a polynomial of order 1 (a) and a polynomial of order 2 (b) with different smoothnesses, 1 (solid line), 3 (dotted line), 5 (dashed line), 7 (dotted–dashed line).

Wendland construction

The particular method proposed by Wendland (1995), Wendland (1998) uses the truncated power function:

$$k_c^1(s, L, \nu) = \begin{cases} \left(1 - \frac{s}{L}\right)^{\nu} & 0 \leqslant s \leqslant l \\ 0 & s > l \end{cases} \tag{4.5}$$

for constructing a valid space–limited covariance function where L is a truncation parameter and controls the support size of the covariance function or sparsity in the covariance matrix and ν is a smoothness parameter of the function which defines the differentiability of the covariance function. The truncated power function is not an interesting candidate covariance function because as the differentiability or smoothness increases, the covariance function decays more rapidly. Wendland (1995) suggests constructing a polynomial of a given order and creating a product covariance with the truncated power function to give a flexible space–limited covariance function for use in many dimensions. The differentiability or smoothness of the covariance function is controlled by the degree of the polynomial. An example of a function that is twice differentiable at the origin is given by:

$$k_c^1(t, L, \nu) = \begin{cases} \left(1 + (\nu+1)\frac{s}{L}\right)\left(1 - \frac{s}{L}\right)^{\nu+1} & 0 \leqslant s \leqslant l \\ 0 & s > l \end{cases} \tag{4.6}$$

and yields a positive definite covariance function if $\nu \geqslant \frac{(d+3)}{2}$ holds, where d is the dimensionality of the dataset. To increase the differentiability of the function, a 2nd order polynomial can be used instead. Wendland (1995) suggests:

$$k_c(t, L, \nu) = \begin{cases} \left(1 + (\nu+2)\frac{s}{L} + \frac{(\nu+2)^2-1}{3}\frac{s}{L}\right)\left(1 - \frac{s}{L}\right)^{\nu+2} & 0 \leqslant s \leqslant l \\ 0 & s > l \end{cases} \tag{4.7}$$

which is four times differentiable at the origin and yields a positive definite covariance function if $\nu \geqslant \frac{(d+5)}{2}$. Figure 4.1 shows the Wendland covariance functions constructed from order 1 and order 2 polynomials. The ability to control the differentiability at the origin is important and the Wendland construction covariance functions can be used as tapers for the Matérn covariance function because of this.

Gneiting's Gaussian approximation

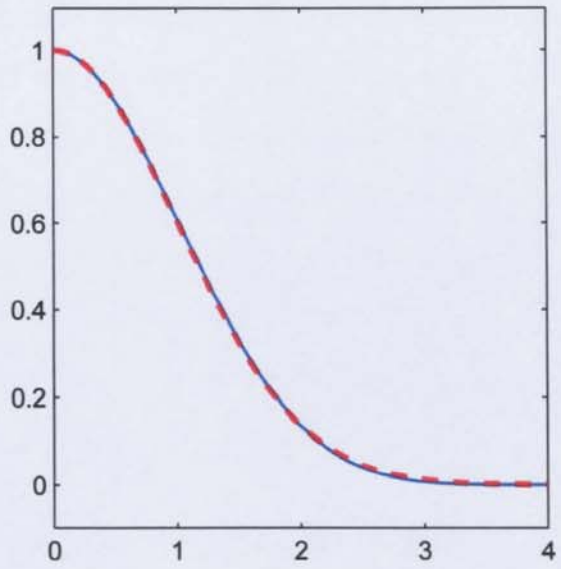


Figure 4.2: The Gneiting or space–limited Gaussian covariance function approximation (red dashed line) compared with the Gaussian covariance function (blue solid line). They are virtually identical in shape.

The Gneiting or space–limited Gaussian approximation covariance function was proposed by Gneiting (1999) as an alternative covariance function for atmospheric data analysis applications. The Gneiting Gaussian approximation is effectively a polynomial approximation (of the type discussed in Section 4.2.6) to a Gaussian covariance function. The construction of this

Gaussian covariance function approximation is given by:

$$k(t) = \begin{cases} \left(1 + \frac{80}{47}\frac{t}{\sigma^2} + \frac{2500}{2209}\frac{t^2}{(\sigma^2)^2} + \frac{32000}{103823}\frac{t^3}{(\sigma^2)^3}\right)\left(1 - \frac{10}{47}\frac{t^3}{\sigma^2}\right)^8 & 0 \leqslant \frac{10}{47}\frac{t^3}{\sigma^2} \leqslant 1 \\ 0 & \frac{10}{47}\frac{t^3}{\sigma^2} > 1 \end{cases} \tag{4.8}$$

and as Figure 4.2 shows, the curve of the Gaussian and Gneiting's Gaussian approximation are almost visually indistinguishable (Gneiting 2002). Gneiting's Gaussian covariance function is 6 times differentiable at the origin and remains positive definitive in 3 or fewer dimensions. This approximation also does not have the associated numerical disadvantages that are encountered with the Gaussian covariance function (Schlather 2006).

4.2.7 Determining the truncation parameter

One of the most neglected aspects of the application of space–limited covariance functions, particularly the functions that require a truncation parameter, is the process to be undertaken to determine the degree of sparsity that the covariance function yields. Since a prime motivation of using space–limited covariance functions is the additional performance increases that can be obtained it is not necessarily appropriate to perform some kind of cross–validation or maximum likelihood approach to determine values for the parameters that control the sparsity of the covariance matrix. Determining a suitable trade–off between prediction accuracy and sparsity in the covariance matrix is something that is open to interpretation depending on the problem domain and it will be difficult to give a definitive answer for a suitable "catch all" method. Furrer et al. (2006) adopts an approach similar to those of finding a suitable neighbourhood radius for moving window kriging.

Problems with likelihood–based parameter estimation

Likelihood–based parameter estimation methods raise a number of problematic issues for the application of space–limited covariance functions. To exploit the sparsity that space–limited covariance functions induce in the covariance matrix, sparse matrix storage methods should be used. The implementation details of sparse matrices can vary from one software package to the next, but one drawback likely to be encountered with all such software is that the sparsity in the covariance will vary during each iteration of the maximum likelihood algorithm. As the range and sill parameters are updated during each iteration, the storage requirements of the

sparse matrix will fluctuate. If the sparse matrix becomes too densely populated during an iteration then the system will run out of memory. One way to overcome this problem, to a certain degree, is to fix the truncation parameter. Unless the range or lengthscale parameter is less than the truncation parameter, then the memory requirement for the covariance matrix will remain constant. This is not necessarily true for the matrix inverse used to calculate the likelihood. Although not the aim of their research, Kaufman (2006) suggests using a tapering matrix twice which helps solve these issues. First the covariance matrix is tapered, then after the matrix inverse has been computed, the matrix inverse is tapered to ensure that the memory requirements do not exceed a defined limit.

The above assumed that a local parameter optimisation algorithm was used, such as quasi–Newton (Zeleznik 1968) or conjugate gradient methods (Hestenes 1980). Global parameter optimisation algorithms, such as simulated annealing (Kirkpatrick 1984), further increase the problem due to their exploration of the parameter space. Unless a fixed truncation parameter is specified, it is recommended that global optimisation algorithms are avoided when sparse matrices are being used.

The space–limited covariance functions such as the Spherical (Equation (4.2)), Circular (Equation (4.3)) or Gneiting's Gaussian approximation (Equation (4.8)) also suffer from the same truncation parameter problem because the range parameter determines the effective truncation of the covariance function directly. If a method–of–moments variogram estimation is the chosen method of determining covariance model parameters then the above listed issues are no longer problematic.

Prediction does not raise any specific issues when sparse matrix methods are used. A method–of–moments variogram estimator could be used without problem. Maximum likelihood methods can be used with care, but certain parameters may need to be constrained to within a certain range to avoid problems with computer memory.

Similarity and sparsity measures

One method that has been proposed for determining suitable truncation parameters for covariance functions is based on measuring heuristics about the sparsity and similarity between the space–limited covariance matrix, Σ_C, and the covariance matrix determined without a space–

limiting adjustment, Σ, (Zhang et al. 2004). Conveniently these heuristic measures can be determined before any processing of the data (ie. matrix inversion) has taken place. The similarity heuristic:

$$S(C) = \frac{\langle \Sigma, \Sigma_C \rangle_F}{\sqrt{\langle \Sigma, \Sigma \rangle_F \langle \Sigma_C, \Sigma_C \rangle_F}} \tag{4.9}$$

measures the alignment between the two covariance matrices Σ_C and Σ. The notation $\langle \mathbf{A}, \mathbf{B} \rangle_F$ denotes Frobenius norm between the matrices $\mathbf{A}$ and $\mathbf{B}$ and is defined in Appendix C.3. This idea was proposed by Cristianini et al. (2002) in a machine learning context to avoid the so called trial and error heuristics that are often used in determining covariance model parameters. Here they state a relationship to the Pearson correlation coefficient and show how this similarity heuristic can be used as a correlation measure between two covariance matrices. The measure has been shown to yield a value in the range of 0 to 1 where a result of 1 means that the covariance matrices are identical and hence there has been no information loss due to the sparsification process and 0 means entirely dissimilar.

Another heuristic, which measures the sparsity (or more precisely the density) of the covariance matrix, is a simple measure of the ratio of nonzero ($nnz(\cdot)$) elements in the covariance matrix divided by the overall number of elements, n^2, in the covariance matrix is given by:

$$T(C) = \frac{nnz(\Sigma_C)}{n^2}. \tag{4.10}$$

Zhang et al. (2004) suggest combining these two heuristic measures of similarity and sparsity to enable the truncation parameter to be tuned by maximising:

$$U(\Sigma_C) = S(\Sigma_C) + T(\Sigma_C), \tag{4.11}$$

where $U(\Sigma_C)$ is a combination of the two heuristics $S(\Sigma_C)$ and $T(\Sigma_C)$ that suit the requirements of the problem and for this thesis a linear combination will be used.

This heuristic procedure is especially useful for applications with large datasets since the storage requirements can be controlled depending on the storage capacity of a system

4.3 Experiments

Having discussed some space–limited covariance functions and ways in which they can be constructed, some experiments will be performed to show the performance with the datasets defined

in Chapter 3. Summary statistics are calculated for all of the experiments and the covariance function parameters for each dataset were fixed at optimal values.

For each dataset the same experiments were performed. As an exemplar of infinitely supported covariance functions, the Gaussian covariance function were chosen since this is a commonly used covariance function and there is a space–limited approximation which can be compared with truncated versions.

First the similarity and sparsity measures of Zhang et al. (2004) are explored. The truncation and smoothness parameters are determined for Wendland construction by analysing these measures of similarity and sparsity. Choosing the specific values depends on how much tolerance can be given to the prediction error and how much time can be afforded for the prediction algorithm. Inevitably, an increase in prediction accuracy leads to an increase in computation time.

The effects on prediction accuracy of varying the truncation parameter are investigated first. The smoothness parameter is then investigated.

4.3.1 Varying the truncation parameter

Figures 4.3, 4.4 and 4.5 each show how as the truncation parameter increases, the prediction accuracy increases.

The behaviour of the plots in Figure 4.3 shows that prediction is erratic when the truncation parameter is below a certain length (roughly 0.6). Since the truncation parameter determines how many observations are used to provide a prediction at a point, the reasons for these volatile results can be intuitively understood. As more observations are used (the truncation parameter is increased), the prediction results become more stable. Figure 4.4 shows that the optimal truncation parameter is around 1.4. The general trend of all the plots is that as the MAE is reduced the correlation between the observations and the true values increases which is what one would expect as accuracy and precision are improved.

By inspecting Figures 4.6, 4.7 and 4.8 clear optimums can be selected where the sum of the two lines (or Equation (4.11)) is maximised. The smooth behaviour seen in the first experiment is also seen here. As the similarity increases the sparsity decreases, but still as with the previous experiment, the similarity measure gives no additional information about prediction error. By

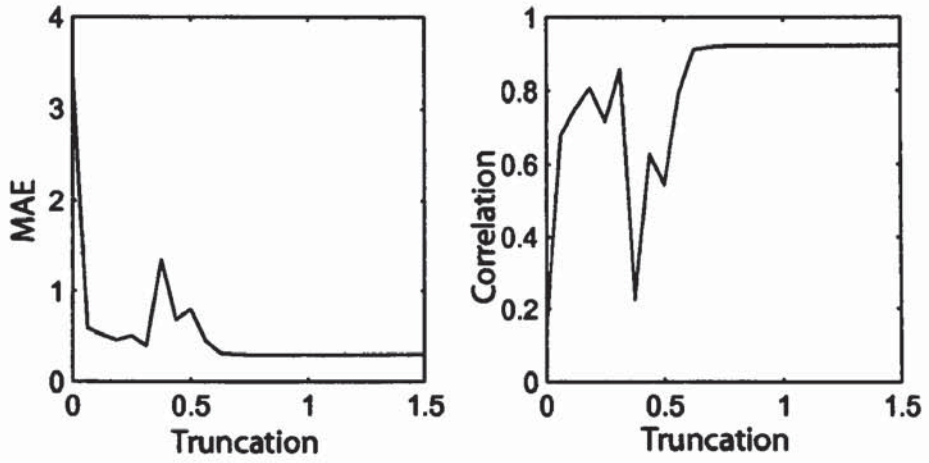


Figure 4.3: Mean absolute error (left) and correlation between predictions (right) with varying truncations for 1D dataset. The truncation parameter is measured on the horizontal axis.

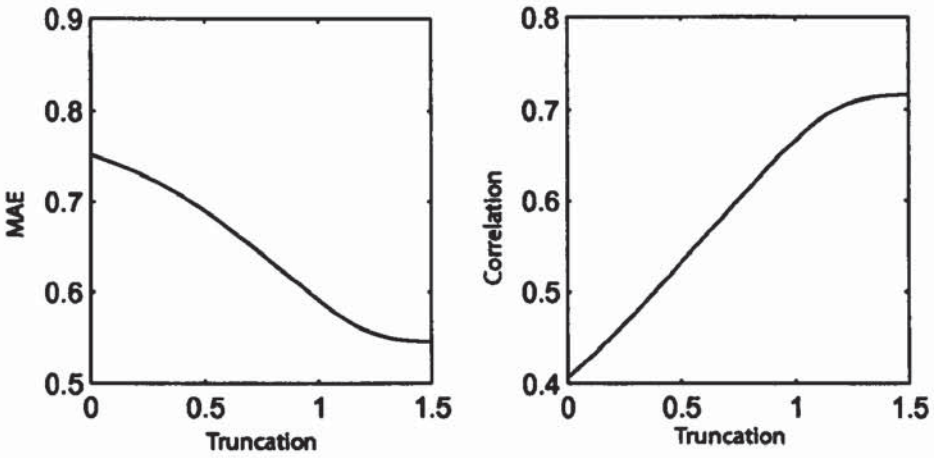


Figure 4.4: Mean absolute error (left) and correlation between predictions (right) with varying truncations for 2D dataset. The truncation parameter is measured on the horizontal axis.

comparing the Walker lake plots, the trough evident in Figure 4.3 cannot be seen at all in Figure 4.6.

4.3.2 Varying the smoothness parameter

Having looked at the truncation parameter it is now time to look at the smoothness parameter. In these experiments the truncation parameter has been fixed.

In these experiments where the smoothness of the truncating function is being changed, the net effect is to select or change the covariance function. So it should come as no surprise that by looking at the vertical axes of Figure 4.9 that the effects induced by varying the smooth-

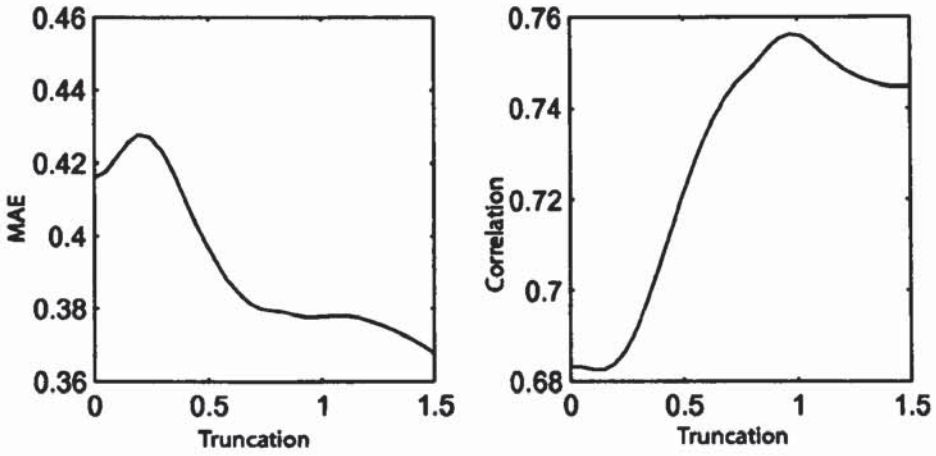


Figure 4.5: Mean absolute error (left) and Correlation between predictions (right) with varying truncations for Walker lake dataset. The truncation parameter is measured on the horizontal axis. The truncation parameter is measured on the horizontal axis.

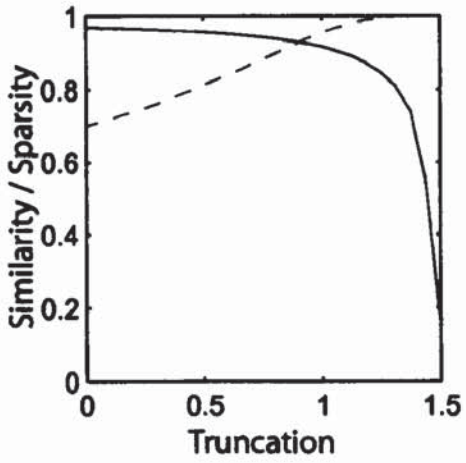


Figure 4.6: Similarity (dashed line) and Sparsity (solid line) with varying truncations for 1D dataset

ness parameter are small, but not neglible. The plots for the 1D dataset show how a plateau is reached gradually where one could say that extending the smoothness parameter beyond a certain threshold has little effect. Unlike the truncation parameter, the smoothness parameter does not dictate the sparsity in the covariance matrix. Hence the smoothness parameter should be chosen based on the properties of the process being interpolated since the Gaussian smoothness assumption is quite unrealistic for most applications. The smoothness parameter should be greater than $\frac{d+1}{2}$ where d is the dimensionality of the dataset.

Interestingly, the prediction performance plots for the 2D dataset (Figure 4.10) show how a smoothness process tends towards a good result. This makes sense since the data were sampled

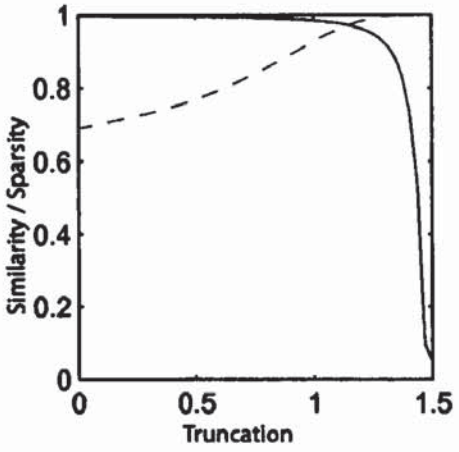


Figure 4.7: Similarity (dashed line) and Sparsity (solid line) with varying truncations for 2D dataset

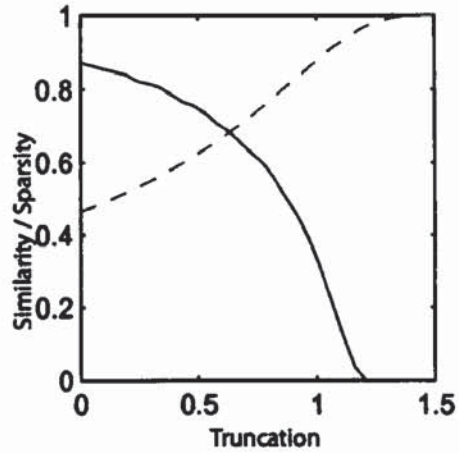


Figure 4.8: Similarity (dashed line) and Sparsity (solid line) with varying truncations for Walker lake dataset

from a simulation using a Gaussian covariance. The Walker Lake dataset (Figure 4.11) shows similar behaviour to the first dataset whereby a plateau is reached for an optimal value.

It should be expected that changing the smoothness parameter does not change the sparsity in the matrix at all. Confirming this Figures 4.12, 4.13 and 4.14 show a constant line for the sparsity heuristic measure. The similarity measure shows that as the smoothness is increased, the similarity decreases. This is to be expected since although the smoothness is increasing, the shape of the covariance function is becoming more and more peaked, although at the same time increasing the differentiability at the origin.

The results from these experiments have shown how space–limited covariance functions can be constructed easily and used to realise sparse covariance matrices. Various techniques have

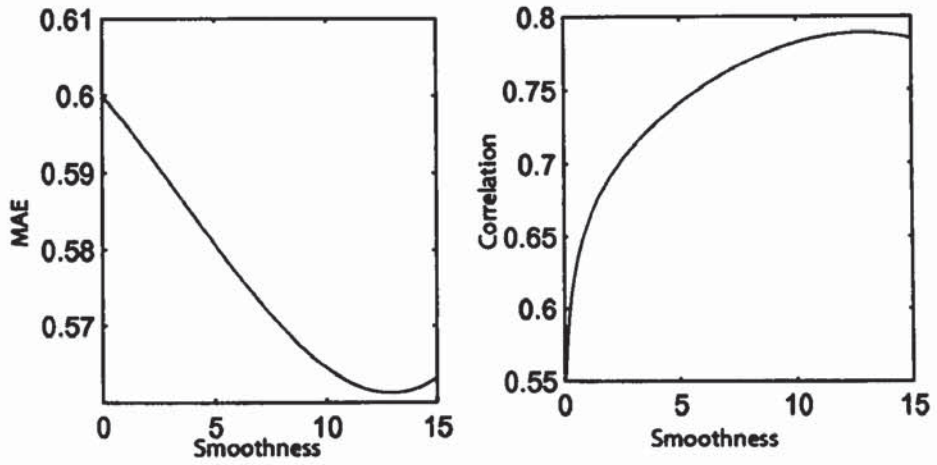


Figure 4.9: Mean absolute error (left) and Correlation between predictions (right) with varying smoothness for 1D dataset

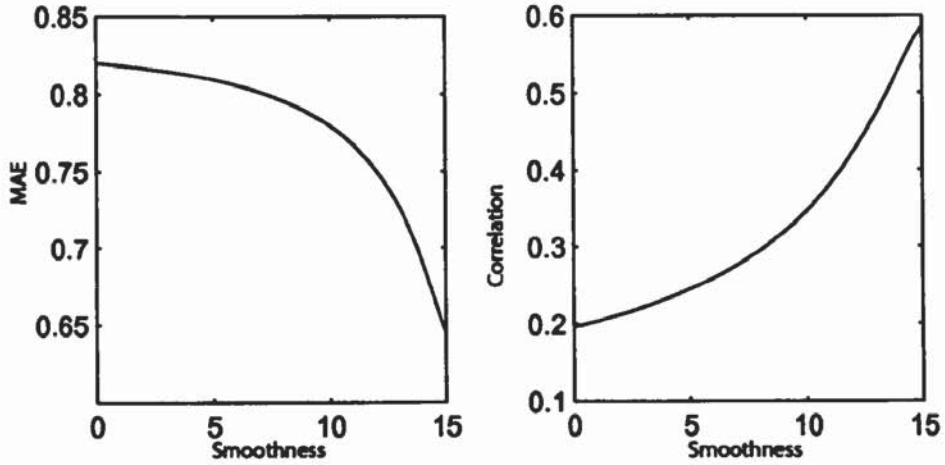


Figure 4.10: Mean absolute error (left) and Correlation between predictions (right) with varying smoothness for 2D dataset

already been proposed as ways to determine the parameters of the space–limited covariance function but there are still some short comings that make their application far from perfect. That being said, these techniques allow a number of advantages to be had with respect to computational speed.

Tables 4.1, 4.2 & 4.3 show summary statistics for the Gaussian covariance function with the Gneiting approximation and the space–limited Gaussian. The first thing to note is the performance of the Gaussian and the Gneiting covariance functions. The MAE and correlations are similar. The Wendland Gaussian performs poorly comparatively, but does have the flexibility of controlling matrix sparsity directly by the inclusion of a truncation parameter. Gneiting's covariance function still yields sparse matrices which would give significant performance im-

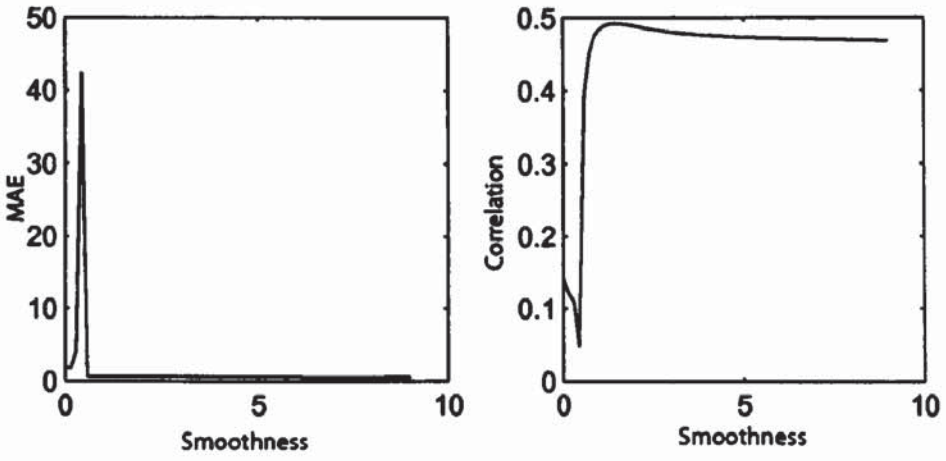


Figure 4.11: Mean absolute error (left) and Correlation between predictions (right) with varying smoothness for Walker lake dataset

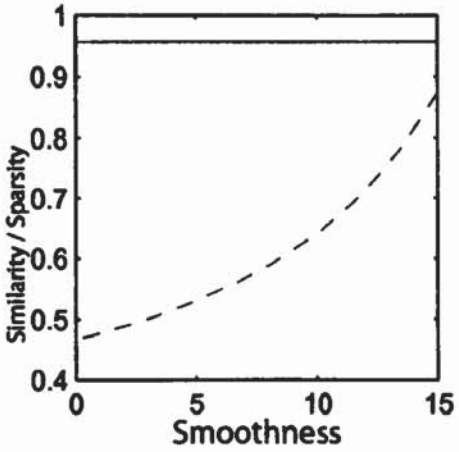


Figure 4.12: Similarity (dashed line) and Sparsity (solid line) with varying smoothness

provements.

To compare the speed of using Gneiting's approximation or the Wendland construction, timings of a simple matrix inversion were calculated for the 3 different datasets using the 3 different covariance functions. Except for the Gaussian covariance function where full matrix data structures were used, sparse matrix data structures were used to store the matrices. The Reverse Cuthill–McKee algorithm (discussed in more detail in Section 4.4) was used to reorder the matrix before the inversion operation was performed.

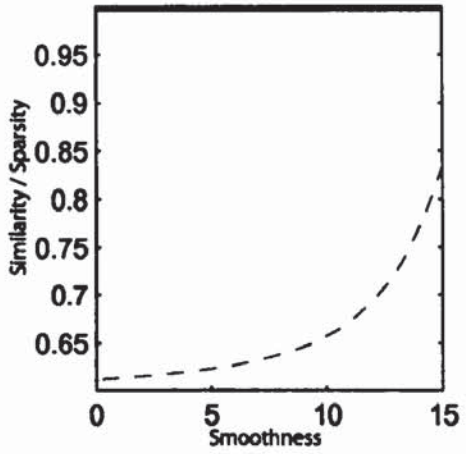


Figure 4.13: Similarity (dashed line) and Sparsity (solid line) with varying smoothness

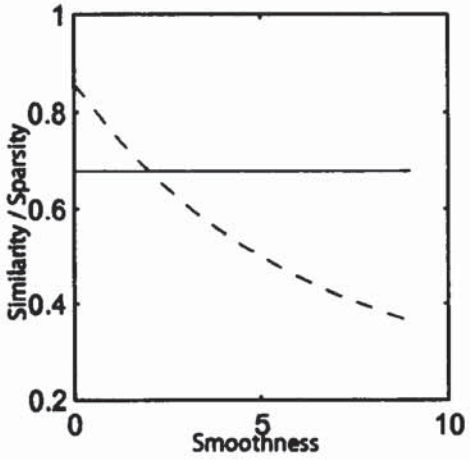


Figure 4.14: Similarity (dashed line) and Sparsity (solid line) with varying smoothness

4.4 Sparse Matrices

Having looked at some ways to represent the covariance matrix using various space–limited covariance functions, it is important that the algorithms used to take advantage of such theory are discussed. There are few advantages to be had by only using space–limited covariance functions without the use of optimised sparse matrix methods (Pissanetzky 1984; Saad 2003).

4.4.1 Sparse Matrix Representation

Typically (so called full) matrices are stored as two–dimensional arrays or arrays of arrays in memory depending on the programming language. Each row (or column depending on the

Covariance function	MAE	R	Matrix sparsity	Time (s)
Gaussian	0.300	0.922	0.0	82.9
Wendland Construction Gaussian	0.302	0.923	87.3	7.3 (0.4)
Gneiting	0.301	0.922	83.7	9.1 (0.5)

Table 4.1: Table showing summary statistics for the 1D synthetic dataset with each covariance functions used. Matrix sparsity and computation time (matrix ordering time).

Covariance function	MAE	R	Matrix sparsity	Time (s)
Gaussian	0.546	0.715	0.0	89.2
Wendland Construction Gaussian	0.552	0.709	96.5	6.8 (0.4)
Gneiting	0.546	0.714	94.6	7.2 (0.4)

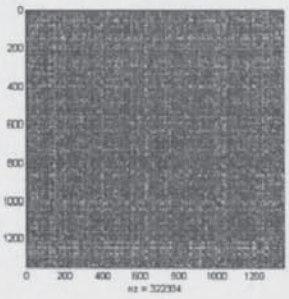
Table 4.2: Table showing summary statistics for the 2D synthetic dataset with each covariance functions used. Matrix sparsity and computation time (matrix ordering time).

implementation) is stored contiguously in memory whereby the last element of a row is stored next to the first element of the next row. One definition of a sparse matrix is *any matrix with enough zeros that it pays to take advantage of them* (Gilbert et al. 1992). The rationale for the use of sparse matrices is about avoiding performing arithmetic operation on zero elements which will yield zeros. This not only reduces computation time but also the storage requirements of a matrix. Many of the matrices due to space–limited covariances have a large proportion of zero elements and hence a more efficient storage scheme could be used to represent the matrix.

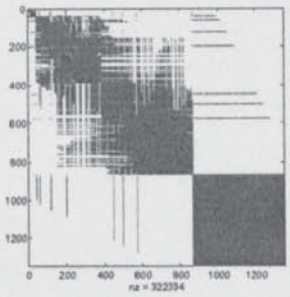
One of the most common forms and simplest ways of representing a sparse matrix is called coordinate format. The data structure needed to represent a matrix in this manner consists of three fields: matrix element, row index and column index. The assumption here is that only non–zero elements are stored and hence any element not listed is assumed to be zero. Specific details about the memory allocated to represent a row or column index are out of the scope of this thesis and hence their details will be avoided here. Variations on the coordinate format representation have been used which rely on the matrix elements being ordered in a specific way in the sparse matrix data structure which enables further savings in the memory requirement. For example,

Covariance function	MAE	R	Matrix sparsity	Time (s)
Gaussian	0.603	0.744	0.0	92.4
Wendland Construction Gaussian	0.610	0.743	86.3	14.9 (0.7)
Gneiting	0.600	0.747	79.2	17.0 (0.8)

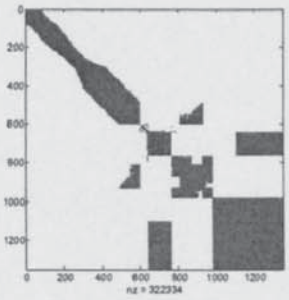
Table 4.3: Table showing summary statistics for the Walker lake dataset with each covariance functions used. Matrix sparsity and computation time (matrix ordering time).



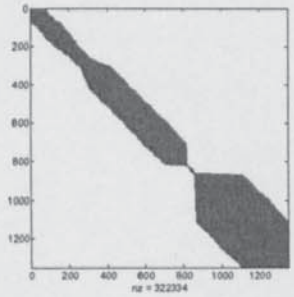
(a) Original matrix.



(b) Column count reordering algorithm.



(c) Approximate minimum degree algorithm.



(d) Reverse Cuthill-McKee algorithm.

Figure 4.15: Matrix structure of 1D dataset after using matrix reordering algorithms.

there is a sparse matrix data structure that assumes a diagonally structured matrix and takes advantage of this property by storing the diagonal elements of the matrix sequentially without using coordinates. The remaining off–diagonal elements are then stored using coordinate format. The most appropriate data structure is dependent on the structure of the matrices involved.

Matrix Reordering

Not all matrices are structured in a way whereby there is a diagonal structure to the non–zero elements. One reason for this is due to the order of the data from which the covariance matrix was constructed. It is possible that the matrix columns, rows or both can be reordered to achieve a more convenient structure to the covariance matrix. Furthermore it is possible (but not guaranteed) that any factorisations performed on the reordered covariance matrix can be sparser as a result.

Probably the simplest reordering scheme is to sort the columns by an increasing non–zero

Method	Computational Complexity
Column count	$\mathcal{O}(\|A\|)$
Minimum degree	$\mathcal{O}\left(\|V\|^2 \|E\|\right)$
R. Cuthill-McKee	$\mathcal{O}(\|E\| + \text{max\_degree}(V))$

Table 4.4: Complexity for matrix reordering algorithms.

count as shown by Figure 4.15b. The column count reordering algorithm moves columns (and rows) with larger nonzero counts towards the end of the matrix. This reordering method is particularly useful for some very irregular structures (Gilbert et al. 1992) and runtime can be relatively shorter than other algorithms in certain circumstances, but this cannot be guaranteed.

Another method which is popular in graph theory is the approximate minimum degree algorithm (Amestoy et al. 1996). This algorithm attempts to group together zero elements into blocks. Figure 4.15c shows how the matrix has a clear block structure.

Probably the most commonly used algorithm is the Cuthill–McKee algorithm. This algorithm aims to reorder the matrix so that the non–zero elements are as close to the diagonal as possible (Cuthill and McKee 1969). A variation on the Cuthill–McKee algorithm called the Reverse Cuthill–McKee algorithm which reverses the ordering of the index numbers generally gives a better solution. Figure 4.15 shows how the elements of this array are reordered on the diagonal.

The complexity of matrix reordering algorithms requires some further definitions. This is due to complexity being related to the structure of the sparse matrix. The computational time to reorder one matrix may be many orders of complexity greater than that of another matrix of the same dimensions. Describing the matrix in terms of number of vertices (V), number of edges (E) and number of columns (A) enables us to describe the complexities for each of the reordering algorithms. Table 4.4 shows the computational complexity for each of the matrix reordering algorithms. One interesting observation from reordering the matrices is the block diagonal structure of the covariance matrix. To illustrate the advantage of this structure a hypothetical covariance matrix is shown in Figure 4.16 which consists of two distinct blocks (70–by–70 and 30–by–30) which will be referred to as sub–matrices. These sub–matrices can be inverted individually and independently of the other blocks in the covariance matrix (Press et al. 1992) and hence instead of inverting a 100–by–100 matrix, a 70–by–70 and a 30–by–30 matrix can be inverted instead. This leads to a number of advantages, the details of which will

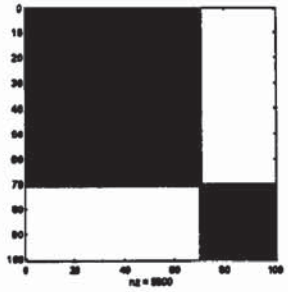


Figure 4.16: Hypothetical block diagonal covariance matrix structure

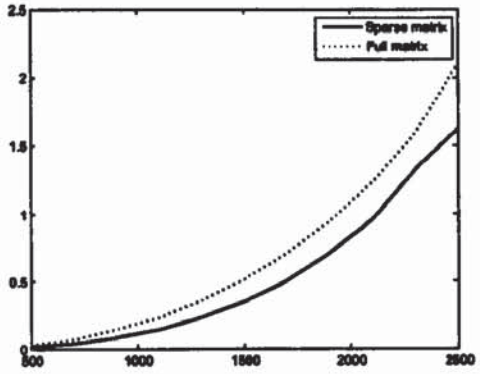


Figure 4.17: Comparison between sparse matrix operations and full matrix operations. Time is given in seconds and the size of the dataset used to construct the covariance matrix is the horizontal axis.

be discussed in Chapter 6. To summarise, firstly the inversion of the covariance matrix can be performed in parts on different processors or processor cores. Currently computing trends are moving towards multi–core processors where splitting a problem down into two independent parts will be executed more rapidly than one big program. Looking at the trends of computer processors in the last three years have shown that rather than increasing the clock speed of a processor, the preferred route of providing more power is via multi–core processors.

To gauge the order of speed up possible from using a sparse matrix method is a complex issue. Issues such as matrix structure play a large part in determining how fast an operation will be performed. As an illustrative example a direct matrix inversion algorithm will be compared. Figure 4.17 shows the speed and shows how consistently the sparse matrix method is faster. The

Method	MAE	R
Original	0.340	0.967
Block Inversion	0.339	0.967

Table 4.5: Results obtained by inverting block diagonal elements of the covariance matrix independently.

dataset used was constructed so that a roughly 10% occupancy was evident with each matrix size.

Again returning to Figure 4.15Ideally, it can be seen that there is an almost clear block structure to the matrix when the Reverse Cuthill–McKee algorithm is used. To verify the previous proposed hypothesis that each block section of a matrix can be inverted independently of the rest of the matrix, the matrix will be split into two parts. Block 1 will comprise of the first 820 matrix elements and Block 2 will comprise of the last 530 elements. A clear reason for this can be seen when looking closely at the matrix occupancy in Figure 4.15d since this is where the locations of low bandwidth occur in the matrix. Ideally, selecting two sub–matrices completely independent of each other is what is required, and by selecting two matrices where there is overlap correlations between observations in the dataset are being discarded. This is also being done in an unprincipled fashion, so that it isn't exactly clear what information is being lost. Accepting these limitations, the results of such an experiment are illustrative of some advantages that can be achieved using such methods.

Table 4.5 shows the results from such an experiment and shows very little deviation in the accuracy when using the block inversion method. By clustering the data or grouping similar data into smaller blocks there are many numerical advantages with respect to computation speed that are to be had. The computation of the above problem on a typical multi–core processor would only be limited by the largest of the two sub–matrices. Chapter 6 will cover more techniques of this type where speed advantages can be had due to dividing the problem into smaller parts and then solving the smaller parts individually.

4.5 Universal approximation

Applying sparse matrix methods leads to increases in computational efficiency when the range of the spatial process is long with respect to the spatial size of the dataset. In Chapter 2, projected

Figure 4.18: Pictorial representation of the construction of the noise free DTC covariance matrix approximation with a low–rank matrix with a space–limited covariance.

process methods were introduced as being a means of improving computational efficiency when the range of the spatial process is short with respect to the spatial size of the dataset.

An universal approximation is now proposed whereby sparse matrix methods are used in conjunction with projected process methods to exploit the advantages of both techniques. This is particularly useful in the case where a large spatial dataset is processed and has regions of densely sampled observations and other regions of sparsely sampled data. Also, in an automatic mapping context, the ability to treat large datasets irrespective of sampling density is very advantageous.

For the universal approximator, the predictive equations are given by:

$$\hat{Z}(\mathbf{x}_*) = \mathbf{k}_{*\mathrm{N}}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathrm{UNI}}^{-1} Z(\mathbf{x}_{\mathrm{N}}), \tag{4.12}$$

$$\sigma_*^2(\mathbf{x}_*) = \mathbf{k}_{**} - \mathbf{k}_{*\mathrm{N}}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathrm{UNI}}^{-1} \mathbf{k}_{*\mathrm{N}}, \tag{4.13}$$

where the covariance matrix is now given by:

$$\boldsymbol{\Sigma}_{\mathrm{UNI}} = \mathbf{k}_{\mathrm{NM}} \mathbf{K}_{\mathrm{MM}}^{-1} \mathbf{k}_{\mathrm{MN}} + \boldsymbol{\Lambda} + \boldsymbol{\beta}, \tag{4.14}$$

where $\boldsymbol{\Lambda}$ is now stored using a sparse matrix data structure. When computing the inverse of $\boldsymbol{\Sigma}_{\mathrm{UNI}}$ using the Sherman–Morrison–Woodbury identity (see Appendix A.1), the inversion $\boldsymbol{\Lambda}^{-1}$ can be computed efficiently using sparse matrix methods. Figure 4.18 shows this effect pictorially. Visually the covariance matrix approximation can now be seen to be a closer approximation to the full covariance matrix shown in Figure 2.5.

Tables 4.6, 4.7 and 4.8 show the performance of the universal approximation method that has been proposed. A clear pattern has emerged which shows the projected process method and the sparse matrix method perform with a similar accuracy. By combining the two using the

Prediction method	MAE	Time (s)
No approximation	0.901	11.22
Projected process	0.962	3.05
Sparse matrices	0.986	1.45
Universal method	0.919	5.58

Table 4.6: Prediction accuracy and time for 1D synthetic dataset.

Prediction method	MAE	Time (s)
No approximation	1.311	12.97
Projected process	1.398	3.44
Sparse matrices	1.427	2.45
Universal method	1.323	5.07

Table 4.7: Prediction accuracy and time for 2D synthetic dataset.

universal approximation framework, the prediction accuracy comes very close to that of the full covariance matrix method. This comes at a fraction of the computational cost as can be shown by the computation time required.

4.5.1 An approximation for very large datasets

Staying with the concept of exploiting sparsity, an alternative approximation is now considered which is particularly useful for very large datasets. Instead of introducing sparsity into the matrix $\mathbf{\Lambda}$ as proposed for the universal approximator, we instead propose introducing sparsity into the low–rank matrix component of the covariance function approximation.

Referring back to the projected process methods of Chapter 2 shows that the matrix inversion of the covariance function approximation:

$$\text{cov}(\mathbf{a}, \mathbf{b}) = \mathbf{k}_{\mathbf{a}\mathbf{M}} \mathbf{K}_{\mathbf{M}\mathbf{M}}^{-1} \mathbf{k}_{\mathbf{M}\mathbf{b}} \tag{4.15}$$

offers potential problems for large datasets. When the active set increases above a certain limit, the matrix inversion will become problematic. By introducing sparsity into $\mathbf{K}_{\mathbf{M}\mathbf{M}}$, the computation efficiency can be improved with the use of sparse matrix inverse and multiplication

Prediction method	MAE	Time (s)
No approximation	0.701	19.45
Projected process	0.761	5.72
Sparse matrices	0.769	4.62
Universal method	0.708	8.20

Table 4.8: Prediction accuracy and time for Walker Lake dataset.

Prediction method	MAE	Time (s)
Projected process	0.761	121
Sparse matrices	0.780	98
Combined	0.765	58

Table 4.9: Prediction accuracy and time for 10,000 observations from the Walker lake dataset.

methods. Furthermore, the size of the active set can be increased should it be needed for the sake of improved accuracy.

To test this alternative approximation for larger datasets, a random subset of 10,000 observations of the entire Walker lake dataset are used. Three main methods were compared: projected process, sparse matrix methods and then both combined. Ideally, a standard simple kriging algorithm would provide a good comparison, but the computation time would be prohibitive for a dataset of this size. An active set of 3,000 observations is used for the projected process methods. For the sparse matrix methods, a truncation parameter of half the range is specified and the matrix is reordered using the Reverse Cuthill–McKee algorithm.

Table 4.9 shows results for prediction and for computation speed. The projected process method gives the best predication accuracy, although it is the slowest. The sparse matrix methods give the worst accuracy, but prediction time is improved over that of the projected process methods. The combined method shows a significant improvement in the computation speed, the prediction accuracy is only slightly worse than using the full projected process method. One could argue that an inappropriate number of active points or truncation parameter were selected. Also, the Walker lake dataset is densely sampled hence it is particularly suited to projected process methods. Admittedly the active points were chosen arbitrarily. However, the main result is the combining of projected process methods with sparse matrix methods, yields a faster algorithm without significantly compromising prediction accuracy.

4.6 Conclusion

In this chapter the concept of space–limited or compactly supported covariance functions has been introduced. Space–limited covariance functions have been grouped into two classes: sparsity controlled by the lengthscale or range parameter and sparsity controlled by an additional truncation parameter. The advantage of the former is that it yields sparse covariance matrices

without any tuning of sparsity parameters whereas the latter gives the practitioner full control over the sparsity determined by the truncation parameter of the covariance function. It has been shown that applying these covariance functions yields sparse covariance matrices which have a number of advantages.

Sparse matrix representation can in many cases reduce the storage overhead for a matrix depending on the number of non–zero elements, matrix structure and the data structure used to represent the matrix. Computation can be more efficient using sparse matrices since operations on zeros need not be performed resulting in speed increases. The exact nature of the increase in computational efficiency is dependant on a large number of factors and cannot be fully described. Reordering the matrices using matrix reordering algorithms is a simple way that can be used to increase performance. A further advantage of matrix reordering algorithms is that a block structure is often evident in the covariance matrix. For block diagonal matrices, operations can be performed on these sub–matrices independently of the other parts of the matrix.

An universal approximator has been proposed that improves prediction accuracy without detrimentally decreasing the computational complexity. It better approximates the full covariance matrix. A further method was considered where a large number of active points are used. This showed how computational complexity can be reduced when very large datasets are used.

5

Parallel Computers

5.1 Introduction

The techniques for treating large datasets discussed in earlier chapters of this thesis have focused on models that exploit different kinds of redundancy in the representation of the model. By manipulating a less complex model or a model with a more sparse structure, computational speed–ups have been achieved coupled with a reduction in the storage requirement of the model. By applying these methods the treatment of many large datasets becomes tractable. However the speed–ups achieved are still not suitable for all applications such as automatic mapping systems where the calculations may need to be performed in (near) real–time (Dubois and Galmarini 2005). Small datasets do not pose significant issues for real–time mapping, but as the cost of deploying sensors over vast geographical regions decreases, the size of datasets obtained from sensor networks has increased (Deligiannakis et al. 2004; Loo et al. 2005). In addition, improvements in computer network organisation have facilitated the formation of computer

clusters and grids.

In this Chapter, some of the principles of parallel computation are introduced. However, specific applications to geostatistics will not be addressed until Chapter 6. Trends in microprocessors will be discussed and how this raises a number of issues for software developers. The different parallel architectures are introduced along with the different software that is appropriate for each architecture.

5.2 Microprocessor Trends

Over quarter of a century ago, one of Intel's co–founders Gordon Moore observed that the number of transistors possible on a given piece of silicon would double every couple of years. This became known as Moore's law throughout the world of computing [1]. Even though over the past years, chip manufacturers have encountered barriers and overcome them, it is estimated that Moore's law will eventually reach a final barrier due to the fact that a wire cannot be made thinner than the width of an atom using the understanding of today. Therefore with the prospect of this barrier in place, it may not always be possible to increase the speed of calculations simple by increasing microprocessor speed.

There has been much talk in recent years of quantum computers which overcome many of the problems posed by traditional silicon processors, but these are still largely theoretical (Hosten et al. 2005).

Parallel processing has long been a solution to increasing the speed of calculations. So called super–computers have been massively parallel for many years now and have been able to do billions of calculations a second. However, until relatively recently parallel computing has been out of reach of almost all except important government related institutions. In 1997, Intel's Deep Blue, a 256–processor massively parallel system, hit the news when it beat world chess champion Garry Kasparov. This showed the world the potential of parallel processor computers. There are a number of parallel processor architectures that are even within reach of the home user and are becoming increasingly popular such as the Beowulf cluster (Gropp et al. 2002).

[1] Intel Executive Biography – `http://www.intel.com/pressroom/kits/bios/moore.htm`

5.2.1 Parallel architectures

In this Section a few of the basic common architectures that can be commonly found will be listed. Intricate details about each architecture will be avoided but rather an outline of each architecture will be preferred.

Multi–core architectures

In this year, 2008, it is becoming commonplace that many new microprocessors have what are termed as multi–core architectures. A multi–core microprocessor is a single chip containing essentially a number of microprocessors or cores. Currently dual–core processors are very common although in the high end server market, quad–core systems are also being used. The term *uniprocessor* has now been defined to describe those processors that just have a single core to distinguish them from multi–core architectures. Multi–core architectures provide a symmetric multi–processing architecture where multiple processor cores access a single shared memory resource. Multi–core architectures are being found in many systems, not just modern desktop computers; the latest games consoles are also utilising this technology. For example, the Playstation 3 console has a *Cell* processor, designed by a collaboration of IBM, Toshiba and Sony and has 9 processing elements. Since the processors are linked by the memory bus, dual–core parallel architectures can be described as tightly coupled.

Parallelism has been evident in computer processors for many years. Pipelining of instructions is the parallel process of fetching, decoding and executing an instruction. While one instruction is being fetched from the stored program, the previously fetched instruction is being decoded and the instruction previously decoded is being executed. Often the pipeline can be longer than three stages and some modern computers have used as many as twenty stages (Quinn 2003).

Parallelism in modern computers is something that has been needed for some time since it is natural with the presence of parallelism in today's operating systems. Most computers have a multi–tasking operating system which explicitly tries to process many programs in parallel. A regular home computer can have more than fifty tasks running concurrently which the operating system manages.

Multi–processor architectures

Another symmetric multi–processing architecture with a single shared memory resource are multi–processor systems. These systems are simply multiple processors sharing the same memory, but the processors are separate and have to share the same memory bus and hence one processor can be left waiting while a large memory transfer takes place from another processor in the system. Multi–core architectures are often described as multi–processor architectures. The main different being that multi–core processors are connected to the computer motherboard by a single processor socket and any inter–processor communication takes place on a communication bus wired on the chip instead of using the system bus. Likewise this system architecture can be described as tightly coupled.

Clusters

Cluster computing has become very popular in recent years thanks to the popularization of the Beowulf cluster. A Beowulf cluster is a cluster of inexpensive *off the shelf* computer hardware that is usually connected by an ethernet link for communication between each node of the cluster. Systems such as this are described as loosely coupled.

Hybrids

Some parallel systems are a hybrid of a number architectures. For example, it would not be uncommon to see a dual processor system where each processor would be dual core giving effectively a quad–processor system. Likewise a Beowulf cluster could utilise many dual–core processor systems.

5.2.2 Popularity of Parallel Processing

Despite the promise of parallel computing it is still not utilised to its full potential.

Predictions from the past

Writing in 1995, the authors of the well respected series of books titled Numerical Recipes, (Press et al. 1996), stated that they were convinced that a parallel processing revolution was imminently about to take place whereby parallel processing was to become the mainstream methodology

for all computing, home and industrial. Additionally, the authors make a prediction that the first years of the new century will bring 4 to 8 user–accessible microprocessors to the standard desktop computer. Furthermore they add that a decade later the number will be between 16 and 512 microprocessors in each desktop computer.

Having the advantage of hindsight, it can be seen that their predictions were ambitious, although their hypothesis that parallel processing would become common place in the home may about to be validated when one looks at the marketshare of dual–core processors. However this does not mean that software is being written to utilise this extra power. Thread–level parallelism, whereby an application runs several threads at once, has been common in many server based applications but is still rare in much software, even with the advent of dual–core processors. Concurrent computing, as thread–level parallelism is often called, focuses on how threads communicate with each other and have become used more commonly since popular programming languages such as Java or C# include constructs for handling concurrency.

Reasons for Slow Uptake

There are a number of potential reasons for the slow uptake of parallel software design. Traditional computer programs may need a redesign to exploit a parallel architecture which may require a significant redesign effort. A program designed for a parallel architecture will also (more than likely) run more slowly when executed on a uniprocessor system. With the legacy of millions of uniprocessor systems still in use, it can be seen why a complete shift to parallel processing has not yet been made in the home computing market.

The problem of many legacy uniprocessor systems is not the only potential reason for a slow uptake in applications being designed to exploit parallelism. There are a number of reasons why the development stage of parallel programming can be considered more time consuming.

Firstly the issue of load balancing needs to be addressed. It is an undesirable situation that processors in a multi–processor system are available, but at the same time idle due to an application being designed in a way that only one process is being used for the majority of the work. During the computation on this one processor, the other processors in the system are dormant. As a result these other processors are not being used and are unavailable to be scheduled to other tasks. It should be a goal of parallel programs that task idle time be

minimised. Another issue linked with load balancing is synchronisation. If the tasks reach a synchronisation barrier point the performance of the system will only be as fast as the slowest task. Normally the issue of load balancing should be addressed in the design of the parallel algorithm, although it may be required that dynamic load balancing is implemented in the algorithm itself which can be in the form of when a task is finished, the processor queues to be allocated another task. Dynamic load balancing can add greatly to the complexity of the implementation of an algorithm but can improve performance.

There are many thousands of different parallel set ups currently in use. Programming architecture-independent parallel algorithms is also an important issue since it depends largely on the architecture in use what design the algorithm should take. For loosely coupled systems, interprocess communication can be very expensive and cause long delays in the execution of an algorithm, but at the same time the memory available for processing in a cluster is often many times that of a tightly coupled system. This leads to a further advantage of parallel algorithms in that it may not only be speed that is the issue but rather the memory storage requirement. By splitting the data into smaller chunks and distributing across a cluster of computers massive datasets can be handled without a system running out of memory.

Amdahl's law

Utilising parallel processing computers does not guarantee increases in slgorithm execution time. Many algorithms or fractions of algorithms are sequential in nature and limit significantly increased parallel performance. There is a law associated with this called Amdahl's law, named after computer architect Gene Amdahl. It is an example of the law of diminishing returns whereby a large amount of extra processing power could be made available to a particular system and the overall speed–up could be a tiny fraction of the extra processing power. Amdahl's law (Amdahl 1967) applied to parallelisation is given by (5.1). Where F is the fraction of a calculation that is sequential and hence 1 - F being the fraction that can be parallelised, then the maximum speed–up achievable by using N processors is given by:

$$\text{MaxSpeedup} = \frac{1}{F + (\frac{1-F}{N})}. \tag{5.1}$$

Amdahl's law is a best case. It does not take into account factors such as coordinating processors or the overhead of inter–process communication.

It is often the case that the structure of the sequential algorithm drives the structure of the parallel algorithm and the architecture that an algorithm is better suited to. It could be the case that an algorithm cannot avoid interprocess communication. This is an unfortunate problem with parallel processing.

The design process is crucial so that an algorithm can best utilise a given system. On the other hand it can often be the case that no extra design or programming effort is needed to split a problem into a group of large parallel tasks where no communication between the parallel tasks is needed. Parallel processing problems can be regarded as being on a spectrum going from *embarrassingly parallel* problems, problems that require no communication between large parallel tasks, and then on the opposite end of the scale there are problems that require lots of interprocess communication. The effective speed–up from parallelising an algorithm can be a complicated issue dependent on the platform used. For example, an algorithm that requires interprocess communication is limited by the bandwidth of the interprocess connection speeds. As mentioned earlier these can range from an on–board memory bus to an ethernet network connection.

5.2.3 Software for Parallel Processing

Communication between processes is vital for parallel processing. Unfortunately, the exact hardware details for the communication medium between processes in a parallel environment can vary. To reduce the programming effort for the program various APIs (Application Programming Interface) and language extensions have been developed which are useful in a number of ways.

Message Passing Software

A specification called MPI[2] (Message Passing Interface) has become the *de facto* standard for communication between processes in distributed memory systems and allows software creators to avoid dealing with the low level details of communication between processors. Implementations of the MPI standard provide an API callable from many popular programming languages.

There are currently two main versions of the MPI specification in use; versions 1.2 and 2.0. There are a number of reasons for the slow adoption of version 2.0 of the MPI specification. Version 1.2 supports a static runtime environment which means the number of processors has

[2] `http://www.mpi-forum.org/`

to be specified before the algorithm is executed. There are some very good and obvious reasons for this. Take for example a typical super computer set up in a research environment where batch jobs are submitted, scheduling software will schedule submitted jobs based on availability of resources and the resource requirements of a particular job submission.

A major change to the MPI specification came in version 2.0 when the availability of a dynamic process model was included. This means that an application can spawn a new thread when it is needed during runtime. This makes sense for many applications since it is often unknown *a-priori* details about the algorithm being run or the data used. Realistically, in all but a limited number of applications such as the automatic mapping and real–time mapping examples listed earlier, it should be possible to know what resources should be allocated to an algorithm. Scheduling dynamic process model systems effectively is basically impossible if an application is free to spawn processes whenever it desires.

MPI has become a standard way for writing portable code for parallel computing. MPI is available for tightly coupled multi–processor machines as well as loosely coupled clusters.

OpenMP

Open Multi–Processing or OpenMP is an API that supports multi–processing programming in C/C++ or Fortran and uses a shared memory architecture. The essence of the API is to use a number of compiler directives that during the compilation stage are used to generate an executable that can run on a multi–processor system. Essentially, OpenMP provides an interface to multi–threading whereby a main or master thread of the compiled program forks a number of child or slave threads and the necessary task is divided amongst them. The compiler directives provide enough flexibility so that both task parallelism and data parallelism can be achieved.

Data parallelism describes the splitting of the data into smaller segments. Each small segment is distributed to a different processor in the system. The program being exectuted on each processor then processes the assigned data. For example, simple *for* loops, where the calculation in each iteration of the loop does not depend on results from previous iterations can be easily parallelised by including a simple compiled directive such as *#pragma omp for*.

Task parallelism on the other hand describes splitting the algorithm into smaller tasks. Each subtask is executed on a different processor in the system. For example, suppose that the aim of

a simple program was to calculate a variety of summary statistics for a given dataset. The task of calculating each summary statistic can be performed on a different processor. By using the compiler directive *#pragma omp section*, independent sections of the algorithm can be indicated to the compiler. When the generated code is executed, the different summary statistics (as indicated by the OpenMP compiler section directive) would be executed on different processors.

One of the main advantages of OpenMP system is the simple interface for enabling programmers to generate parallel applications without a significant reprogramming effort which also reduces the chances of bugs being introduced in the process.

As well as dealing with parallelisation, OpenMP handles the issues to do with the scalability of the compiled application. Returning to the discussion on parallel architectures, OpenMP is often used in hybrid parallel development whereby a combination of message passing such a MPI and shared memory programming are used simultaneously.

This all sounds very good, but ultimately it is still the responsibility of the programmer to identify parallelism by the use of compiler directives. The compiler directives instruct the compiler with regards to the sharing of work in different parts of the program (Quinn 2003).

BLAS and PBLAS

In 1979 a library of subroutines for performing linear algebra operations called Basic Linear Algebra Subprograms[3] was released. These subroutines have become the standard for linear algebra and are used frequently in parallel processing. The popularity of these subroutines and their importance for producing efficient programs has led to chip hardware vendors releasing specially optimised versions of this library to suit a particular hardware platform. Also an open source version of BLAS has been developed, which is called ATLAS[4], that provides a self–optimizing BLAS implementation given the platform it is compiled on. ATLAS has been developed so that it is portable and is used extensively across many platforms. Recent versions of ATLAS come with the option of exploiting multiple processor cores, although this multi–threaded option should be used with care because for some mathematical operations performance decreases due to the overhead of spawning new processes. BLAS is split into three levels depending on the complexity of the subroutine being called. Level 1 deals with scalar operations and vector

[3] http://www.netlib.org/blas/
[4] http://math-atlas.sourceforge.net/

operations. Level 2 has all the functionality for the operations between matrices and vectors. Level 3 contains all the matrix solvers and operations between matrices. The function names that BLAS uses can often be considered quite cryptic although they have a sensible structure and if the abbreviations are known it can make coding productive without having to refer to the reference manual frequently.

A parallel version of BLAS is available and it called PBLAS[5] and is part of the ScaLAPACK project which will be discussed later. PBLAS is comprised of BLAS and BLACS (Basic Linear Algebra Communication Subprograms). BLACS[6] provides a linear algebra oriented message passing interface that can be implemented across a large range of distributed memory platforms. Since it would be extremely impractical to rewrite efficient distributed memory algorithms for every different parallel machine, BLACS provides an interface for linear algebra applications to be more easily developed and ported to other parallel machines.

LAPACK and ScaLAPACK

Linear Algebra PACKage or LAPACK is a software library for numerical computing. It provides routines for solving linear equations, eigenvalue problems, decompositions and other matrix operations. It relies on BLAS. A parallel version of LAPACK has been released and is called ScaLAPACK. It is designed for use with MPI.

5.3 Conclusion

There are many options for software for exploiting parallel architectures. Each software package is aimed at a particular architecture and currently there is no one parallel programming paradigm that suites all parallel architectures.

MPI will be used in this thesis since it is a common, well recognised interface. It is available on all architectures and offers sufficient flexibility to the programmer.

[5] http://www.netlib.org/scalapack/pblas_qref.html
[6] http://www.netlib.org/blacs/

6

Parallel algorithms for geostatistics

6.1 Introduction

In the previous chapter a review of parallel computing showed the potential that parallel algorithms show for increasing computational power. Also, the review showed a shift, or emerging trend in the commodity hardware market, that it is now becoming common that a computer will have more than one processor core. Unfortunately this does not necessarily translate to immediate speed–ups to existing computational algorithms. Algorithms have to have been designed in a manner that can take advantage of a parallel architecture. It isn't always possible to redesign an algorithm to exploit a parallel architecture so sometimes approximations to a particular algorithm need to be used. Continuing the theme of this thesis, a number of algorithms for treating large–scale geostatistical datasets will be presented that can provide speed–ups, particularly where parallel system architectures are available.

The ever increasing computational power of the personal computer has enabled their applica-

tion to datasets of modest size. Also it has been shown that maximum likelihood–based methods can give more reliable results when compared with method–of–moment estimators (Stein 1999; Lark 2000). Throughout this chapter the treatment of parameter estimation and prediction are treated separately even though for many of the techniques, the core challenge and solution to the challenge is based on the same techniques for inverting a large matrix of the same size as the number of observations in the dataset.

6.2 Review of Existing Parallel Algorithms

Little progress has been made in parallel kriging algorithms. The same basic techniques have been used since they were first proposed. This is could be due to the difficulty of considering such algorithms in a parallel context. It should not be considered uncommon that difficulties arise from trying to create a parallel implementation of an algorithm.

Another observation is that some of the researchers developing parallel algorithms tend to be doing so in a vacuum, in that contributions are developed independently of other contributions without recognising or building on the work of other authors. The same parallel approaches are discussed in Gajraj et al. (1997) and Pedelty et al. (2003) without any citation to the work of the other author. Likewise Gebhardt (2003) and Kerry and Hawick (1998), who present alternative parallel kriging techniques, seem unaware of other developments in the parallel kriging research area.

Where large amounts of data are available, some geostatistians would ask the question "Why would we want to interpolate? Surely we already have the data sampled at a sufficiently fine resolution?" Firstly, one should think about the issue of scale. Although a large dataset obtained by sampling across the globe might seem exhaustive, each grid cell might account for large areas of interest where more precise interpolation is required. Furthermore, the properties of a dataset may not be apparent until analysis has taken place. By using geostatistics many of the properties of the dataset such as smoothness, noisiness and ranges of variation can be estimated and interpreted to give a better understanding of the dataset.

An alternative direction for parallel algorithms have been developed by Wilkinson (2005) and Crouchley et al. (2005). They demonstrate how Monte Carlo sampling techniques can be used in a parallel framework. Sampling techniques are not considered in this thesis.

In what follows, existing parallel approaches to geostatistics are summarily reviewed. The review is split based on the two major activities undertaken in geostatistics: prediction and model parameter estimation.

6.2.1 Prediction

Parallel implementations of prediction algorithms can be split into two main approaches depending on the architecture of the parallel system being used (Lu and Goddard 2004). Lu and Goddard (2004) go on to describe how loosely coupled architectures suit domain decomposition algorithms better whereas tightly coupled architectures suit techniques that rely on a library call to some parallel matrix library such as ScaLAPACK (Blackford et al. 1996) to solve the kriging equations. There are a number of ways of implementing kriging algorithms using the entire dataset or using moving–windows to select a subset of the data on which to perform kriging. Each algorithm design decision dictates what parallelism can be achieved in the algorithm.

Domain Decomposition

One example of using domain decomposition techniques is given by Pedelty et al. (2003). By assuming the moving–window kriging algorithm, near linear speed–up is achieved by assigning segments of the prediction grid and all the included observations to each processor. Given a segment of the prediction grid, each processor computes the predictions using a moving–window kriging approach. One can see how this results in a near linear speed–up since the moving–window kriging approach allows predictions to made independently of each other by using small covariance matrices of a local neighbourhood. The domain decomposition technique provides a standard implementation of the kriging algorithm, but that said, the issues associated with moving–window kriging previously discussed in Chapter 2 such as discontinuities that arise in the prediction as observations are included and removed in the moving window remain. Also worth noting is that Pedelty et al. (2003) do not discuss variogram parameter selection and it is assumed that these parameters are already known. Either the variogram is estimated from a subset of the data or a method–of–moments algorithm needs to be developed.

Loosely coupled architectures could be used as a platform for the domain decomposition algorithm since there is no inter–process communication during the computation. Additionally,

if the area for prediction is large, it may be advantageous to have the prediction area spread over a number of systems since the memory requirement could be prohibitively large for a single kriging system on a single node.

A further optimisation not discussed in the work on this technique could be considered. Broadcasting the entire dataset to each node, which for architectures with a slow interconnection could become prohibitive, is not recommended. The observations relevant to a particular partition, that is to say the observations that are within a certain distance, **d**, of the partition where **d** is related to the range parameter, should be sent to each node rather than the entire dataset.

In Chapter 4 the relationship between moving–window kriging and the use of space–limited covariance functions was explored. Continuing on from this, by assuming the range parameter to be short with respect to the size of the overall prediction area, the full kriging method could be used in a parallel fashion. As with the moving–window approach just discussed, only a subset of the observations need to be sent to each node. This subset of observations would include all the observations within the spatial support of the prediction area given the variogram parameters. The number of observations that would be needed for prediction using space–limited covariance functions is likely to be larger than the number of observations needed for the moving–window approach, except now that space–limited covariance functions are being used the drawbacks of moving–window kriging are no longer apparent.

A discussion of and proposal for estimating appropriate variogram parameters can be found later in this chapter. In all the literature surveyed, there is no clear indication of any parallel algorithms created for estimating the variogram directly in the context of geostatistics.

Another variation of the domain decomposition method was implemented by Gebhardt (2003) whereby instead of the whole dataset being sent to each processor, the data is divided into tiles and each tile is sent to a processor. To overcome the border effect apparent with algorithms that rely on tiles, surrounding observations are included so that each tile area now overlaps. This overlap will reduce the border effect, but this will not remove the effect completely should the nugget be non–zero. A complete implementation using PVM[1] and the R statistical package was made available.

[1] `http://www.csm.ornl.gov/pvm/`

Parallel Linear Algebra Routines for Prediction

Another way to implement a parallel kriging algorithm would be to utilise a parallel linear algebra library such as ScaLAPACK[2] or PLAPACK[3] to perform large matrix operations such as calculating the matrix inverse which is crucial in kriging algorithms. This is exactly the route taken by Kerry and Hawick (1998) for the prediction of rainfall across Australia. The authors argue that moving–window kriging is commonly used for two reasons.

First, observations at long lag separations from the prediction location are unlikely to be of benefit to the accuracy of the prediction at a desired location and secondly, it is less expensive computationally. Kerry and Hawick (1998) then state that that improved prediction accuracy are generally obtained when the largest possible number of known points are used (Border 1993) and since this is an expensive option it would be desirable to develop a parallel implementation. The algorithm essentially uses a ScaLAPACK function for LU factorisation to obtain the full matrix solution (Kerry and Hawick 1997). Kerry and Hawick (1997) consider an alternative method, which is a system called NetSolve.[4] The application acts as a server that accepts requests from client applications with large complex matrix operations. Instead of the application needing to be run on a system with a parallel architecture, only a server running NetSolve is needed which will accept function calls from a uniprocessor client application.

The use of parallel linear algebra libraries is limited to tightly coupled architectures since there are currently no such libraries designed specifically for loosely coupled architectures. The linear algebra algorithms require a large amount of inter–process communication and hence are not generally suited to loosely coupled architectures.

To facilitate the use of parallel algebra libraries, a compiler was developed that selects linear algebra function calls in an algorithm and automatically creates parallel code for that function call (Ramaswamy et al. 1996). In doing so, this reduces the amount of effort that the programmer has to make to develop parallel friendly applications. Hiding much of the parallel programming from the user will be advantageous to development time because their will be no need to learn new complicated language features or interfaces.

One of the key functionalities of many geostatistical algorithms is the ability to solve sys-

[2] http://www.netlib.org/scalapack/
[3] http://www.cs.utexas.edu/~plapack/
[4] http://icl.cs.utk.edu/netsolve/

tems of equations (Cressie 1993). As shown in Chapter 4, when using space–limited covariance functions which yield sparse covariance matrices, solving these equations by using a Cholesky decomposition is both stable and fast. There are many algorithms for exploiting sparsity when calculating the Cholesky factorisation. Likewise, much research has been undertaken into parallel versions of these algorithms (Gilbert et al. 1992; Manne and Hafsteinsson 1995; Irony et al. 2002). Just as with the serial case, parallel algorithms tend to be developed to solve a specific structure in the Cholesky factorisation although there are a number of general purpose, scalable algorithms available also. For a comprehensive review of parallel Cholesky decomposition algorithms the reader is referred to Gupta et al. (1997) and for more general information on parallel algorithms to solving linear systems a good review paper is that of Duff and van der Vorst (1999).

6.2.2 Parameter estimation

A topic not addressed in parallel kriging algorithms is how the parameters of the model are estimated. Traditionally two approaches are used: method–of–moments or maximum likelihood estimators with computational complexities of $\mathcal{O}\left(n^2\right)$ and $\mathcal{O}\left(n^3\right)$ respectively. Computing a method–of–moments variogram at $\mathcal{O}\left(n^2\right)$ complexity may seem relatively insignificant in comparison to prediction with $\mathcal{O}\left(n^3\right)$ complexity and perhaps this is why there is no prior research detailing this process. With respect to parameter estimation, the research effort has been entirely limited to maximum likelihood methods.

Typically the likelihood, or negative log likelihood is minimised using numerical optimisation algorithms. For some optimisation algorithms gradient information is needed and can be provided cheaply in terms of computation. Since the log likelihood and the gradient of the log likelihood with respect to the variogram parameters can be calculated independently it can be seen how the log likelihood can be calculated on one processor and the gradients could be calculated on other processors in a parallel fashion (Neumaier and Groeneveld 1998). This is a particularly attractive solution for dual–core processors found in common desktop computers and if the current developers of software wrote their applications in such a way to exploit this, speed–ups could be achieved without much extra programming effort.

One observation, already noted in a previous chapter of this thesis, is that one of the main

barriers to increasing computational performance is not necessarily the machine architectures, but rather increasing the performance of software developers (Press et al. 1996).

An alternative idea discussed by Malard (2002) is to assume that the covariance matrix is block diagonal. In doing so, the block diagonal elements of the covariance matrix can be inverted independently of the other block diagonal elements. The assumptions about where to impose the block diagonal structure are crucial decisions in obtaining a reliable result with this method. It is important to realise that this technique is an approximation to the actual likelihood. Information about correlations between matrix blocks is being discarded and hence the approximation does not yield optimal results, unless of course the matrix has a sparse structure and the block diagonal elements in the matrix reflect the actual structure of the covariance matrix. Chapter 4 discusses the application of space–limited covariance functions which realise sparse covariance matrices and this idea of manipulating block diagonal covariance matrices has already been discussed.

A further idea based on this approach is to assume that only the log likelihood gradient calculations depend on block diagonal matrices, but the full log likelihood still assumes a full matrix structure. Hence the algorithm complexity can be reduced to increase performance when using optimisers that require gradient information (Malard 2002) although potential convergence problems may arise if the gradient is not consistent with the objective function.

Parallel Linear Algebra Routines for Parameter Estimation

An alternative approach based on the ideas already discussed is to use a parallel linear algebra library to perform the operations for computing the likelihood (Misztal 1990). Again these approaches use parallel matrix techniques that are largely limited to architectures with expensive tightly coupled hardware.

Further speed–ups can be achieved by further manipulating the likelihood equations and have particular application to dense matrices. Suppose an equation has the form $\mathbf{z} = \mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d}$ then $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ can be computed separately on different processors (Malard 2002).

There are a number of barriers preventing the realistic application of linear algebra matrix library approach. Firstly a tightly coupled system is needed. Apart from current trend with towards dual–core processors in low–cost personal computers, tightly coupled architectures are

expensive and generally are only available to the few. Tightly coupled dual–cored personal computers benefit from such algorithms, in fact, current implementations of the ATLAS[5] software include options for specifying the number of cores that a target platform has. Assuming that the system has the correct libraries available, parallel linear algebra operations can be performed seamlessly without any intervention from the computer operator.

A second barrier to using these techniques to solve large systems of equations is that there may not be enough computer memory to store the entire covariance matrix. A Cholesky factorisation is often used as a preconditioner so that the size of the representation of the matrix can be reduced to almost half of its original size. If the latency of remote memory access is not too large, then it may be possible to store the covariance matrix in parts across the parallel system. This may not seem like a realistic solution, but when it is considered that a covariance matrix, represented using double precision arithmetic, of a dataset with 50,000 observations would require nearly 150 Gigabytes of storage, it is clear that the memory of a single processor is inadequate. In 2007, 150 Gigabytes of RAM would be rare in a uniprocessor system, however, only a decade ago, 1 Gigabyte RAM in a computer would have been considered rare. It is acknowledged that computer hardware is constantly progressing and that barriers of *yesterday* are constantly being surpassed.

6.3 Algorithm implemention using MPI

For quick prototyping of algorithms, Matlab[6] is the preferred environment in this thesis due to the speed of development and the natural way that mathematical algorithms can be created. Matlab is a commercial software package which requires that a user has a license per process that is active in memory. In the context of parallel programming this constrains users to the number of licenses that can be afforded rather than hardware constraints. A cheaper solution is to use a freely available open source Matlab clone called GNU Octave[7]. Although GNU Octave does not provide 100% Matlab compatibility, it is possible to write programs using the langauge elements common to both environments.

MPI has been chosen for handling the communication between processes since it can be used

[5] http://math-atlas.sourceforge.net/
[6] http://www.mathworks.com
[7] http://www.gnu.org/software/octave

on all types of parallel architectures. The communication between processes in a parallel system can be a complicated issue since there are a number of communication scenarios which the MPI manages automatically giving further speed–up advantages. For example, the *MPI_Bcast* function involves sending data from a master process to all the other processes in the communicator group. If the inter–process communication medium were an ethernet network and the number of processes was set at 256, then sending 256 identical messages to each process would be time consuming and it may take some time to finally broadcast the data to the 256th process. Since synchronisation in a parallel algorithm is very important because the computational speed is only as fast as the slowest process, it can be seen why this is an issue. Instead, various strategies are implemented behind the MPI interface which depend on the particular MPI implementation. One strategy is to have a tree–like structure of the processes where one process sends the data to two other processes, which in turn send the data onto two other processes. This is one example of ensuring a balanced mechanism is used to distribute the data.

Some incomplete MPI implementations do not always provide many of the features that the full MPI specification details. Users are constrained by the software that is available and at the moment for programming languages like Matlab there is still a lot of development that needs to be done. As the MPI specification is continuously evolving, MPI implementations are constantly trying to keep up. To maximise compatibility and simplicity some implementations just have a core number of MPI functions. For example, an MPI implementation available for Matlab called MatlabMPI was created with a small subset of commands (Kepner 2001). As far as inter–process communication is concerned, there is only simple blocking Send, Recv and Bcast functionality. The underlying communication medium in reality, is the NFS file system shared by the nodes on the network so there is a large latency for inter–process communication. One side effect of this implementation is efficient program design. Instead of using the rich function set available in MPI implementations to distribute and gather data such as *MPI_Reduce*, *MPI_Gather*, *MPI_Scatter* and *MPI_Alltoall*, programs have to be designed around the basic Send, Recv and Bcast subroutines.

A more complete implementation of the MPI standard and the implementation that will be used throughout this chapter can be found in the MPITB toolbox[8]. MPITB can be used with Matlab and GNU Octave. The implementation is very fast, the software is written in C

[8] http://atc.ugr.es/javier-bin/mpitb

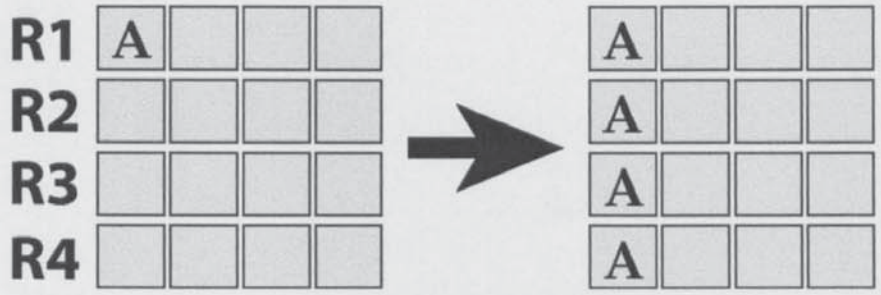


Figure 6.1: Illustration of the MPI Bcast function. Process R1 has data A initially. This data is broadcast to the other processes.

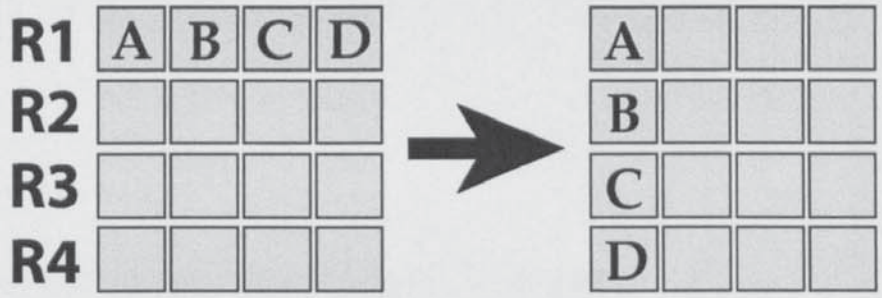


Figure 6.2: Illustration of the MPI Scatter function. Process R1 has data A, B, C and D initially. This data is then scattered amongst the other processes until each processor has a part of the data.

and can use shared memory or ethernet for communication between different nodes (Fernández et al. 2004). Currently the implementation is lagging somewhat since it relies on the somewhat dated LAM/MPI[9] MPI v1.2 implementation although work on compatibility with Open–MPI[10] is ongoing.

6.3.1 Visual explanation of common MPI subroutines

Figures 6.1, 6.2 and 6.3 visually explain how the data is distributed across the parallel system and show the typical message passing functions useful for the algorithms discussed in this Chapter.

MPI_Bcast causes a single item of data to be sent to each of the processes in the parallel system. *MPI_Scatter* causes data to be split into smaller chunks and then distributed across the processes in the system. This is particularly useful for dividing datasets across processes. *MPI_Reduce* collects data from each of the processes and then performs an arithmetic operation on the data such as add or multiply. There is a large range of MPI functionality, but the above

[9] http://www.lam-mpi.org/
[10] http://www.open-mpi.org/

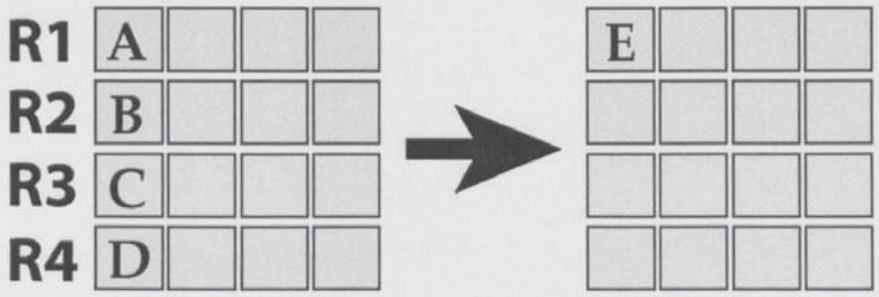


Figure 6.3: Illustration of the MPI Reduce function. Each process has a different item of data A, B, C and D initially. Each process then sends its data to a single process whereby a simple arithmetic operation is performed such as add, subtract, multiply, etc.. and the result is available in E. Effectively the data scattered across the processes is collected together in one process.

listed three functions are of most use for the methods that will be discussed here.

6.4 Parallel Ensemble Methods

Section 6.2 surveyed some of the options that are available for treating large datasets. In this section, instead of representing the whole dataset by one model, which can be seen to be problematic for large datasets, attention is turned to representing the dataset by ensembles of smaller models. In this context, the word *ensemble* is used to mean that a group of sub–models are combined in some way to obtain predictions for a model of the whole.

One of the issues with throwing more computational power at existing algorithms has to do with the algorithm complexity. Trying to increase performance of linear algebra subroutines by doubling the processing power can only at the very best half the speed of the calculation. Since most of the linear algebra algorithms that will be used have a complexity that scales cubically with the size of the dataset, it can be seen that doubling the computational power will have little impact as the size of the dataset is doubled. As worldwide storage capacity doubles every 9 months (Fayyad and Uthurusamy 2002) it can be seen that the necessity to handle increasingly large datasets is crucial and hence alternative approaches must be considered for large datasets.

Many ensemble approaches have been developed in the machine learning community and the reader is directed to Bishop (2006) for a more thorough treatment of the commonly used methods. A number of popular ensemble methods such as boosting and bagging are more appropriate for classification problems although they have been used in a regression context (Avnimelech

and Intrator 1999). These methods are easily parallised since they involve creating independent models (Lazarevic and Obradovic 2002); however, for regression, in comparison to other methods that will be discussed, their generalisation performance is poor.

An essential ingredient of ensemble models is that of splitting the problem into a number of smaller submodels. This generally reduces the complexity of the problem from $O(n^3)$ to $O(qm^3)$ where $m << n$ and q is the number of sub models. The task is then to decide how to perform predictions given these submodels. A simple approach is to average the predictions of each submodel at a given location (Ormoneit and Tresp 1998). Submodels can be assigned partitions of data and when a prediction is made, each model can provide a prediction which will then be averaged (Poulet 2003).

Issues with selecting appropriate parameters for the covariance model have yield little research. A method–of–moments variogram could be computed and parameters could be fixed before any further processing of the data although in the machine learning community, likelihood–based approaches are more extensively used. Using likelihood–based approaches, some authors have allowed submodels to learn individual parameters but research is still lacking on how to combine these models in a statistically principled fashion other than a simple weighted combination. Some models require that common parameters are estimated for each submodel. There are reasons for this since it enables a consistent model to be created. Learning independent submodel parameters is advantageous for parallel algorithms since no inter–process communication need take place. For algorithms where common parameters are required across the submodels, the process for estimating these parameters in the implementation is no more advanced than summing the log likelihood of the individual submodels, although no mathematical description of learning the model parameters is given in the associated literature. The author is aware of some research work completed using *Laplace propagation* techniques to determine model parameters in a consistent framework (Latouche 2006)

Generally there are three basic types of ensemble models:

- **Mixtures** : A different submodel is trained in each spatial location, and the submodel assigned to a particular spatial location will be used in prediction.
- **Committees** : The submodels are combined in some form of weighted average whereby a weight is assigned to each submodel based on the confidence of its predictions.

- **Mixtures of Experts** : A gating network is trained with the submodels to determine mixing coefficients. A linear combination of submodels and mixing coefficients are used for prediction.

In this chapter, Committees and Mixtures of Experts will be considered in more detail.

6.4.1 Clustering the Data

Clustering the dataset into subsets that are spatially separate blocks is recommended to improve predictions for ensemble learning methods (Schwaighofer and Tresp 2003). One useful clustering algorithm that is commonly used is that of k-means (Lloyd 1982). One method termed GeoClust has been proposed which attempts to create spatially balanced clusters (Choudhury et al. 2002) and is based on the k-means algorithm. Another approach is to use some kind of hierarchical model to divide the data into subsets. One example is that of Shen et al. (2006) whereby a *kd–tree* is used to divide the data into subsets at different levels. Along the same lines, a full Bayesian approach is considered by Gramacy and Lee (2006) which they call *treed models*. The input space is iteratively partitioned into subregions creating a tree structure and then each partition or branch of the tree is considered individually. A further advantage to the Bayesian treed model is that it is effective for dealing with nonstationarities. Although this method relies on MCMC sampling methods, the computational time is reasonable since the sampling is over each branch of the tree rather than the whole model and therefore convergence is quicker.

6.4.2 Mixture of Experts

The Mixture of Experts approach was proposed by (Jacobs et al. 1991) as a way to overcome some of the issues associated with large datasets such as the need to invert a large matrix based on all of the observations. Submodels are trained on subsets of the data which are weighted by coefficients learnt from a gating network.

The Mixture of Experts can be expressed as:

$$p\left(\hat{Z}(\mathbf{x})\,|\mathbf{x}\right) = \sum_{c=1}^{C} \pi_c\left(\mathbf{x}\right) p_c\left(Z(\mathbf{x})\,|\mathbf{x}\right), \tag{6.1}$$

where $\boldsymbol{\pi}(\mathbf{x})$ are mixing coefficients calculated by the gating network and c denotes the submodel

1. Master to scatter training data to each process (MPI_Scatter)
2. Each process to perform kriging on given training data
3. MLP used to learn weights for each model across prediction space
4. Predictions made by summing each expert's weighted contribution (MPI_Reduce)

Figure 6.4: Pseudocode for parallel Mixture of Experts.

and

$$\sum_{c=1}^{C} \pi_c(\mathbf{x}) = 1, \tag{6.2}$$

where the total number of submodels is given by C.

Mixtures of Experts have already been extensively used for parallel applications using Support Vector Machines (Collobert et al. 2002; Gibbs 2003). They have also been used in a Gaussian Process framework but were not applied in a parallel context (Tresp 2000c). The idea is that different submodels (or experts) can model the different spatial regions more accurately. The gating network or function determines which components are dominant in which spatial region.

There are some links between the Mixture of Experts approach and moving–window kriging. In moving–window kriging a subset of the dataset is used for prediction at a particular location. Here, many subsets are created and instead of using just one subset for prediction at a particular location, all the subsets are used. However, each subset is assigned a weight for a particular location in the prediction space.

In this parallel kriging implementation, the methodology for hard Mixtures of Experts will be followed (Collobert et al. 2002). In this methodology, each expert is trained individually on a subset of the dataset. Then a multi–layer perceptron (MLP) Bishop (2006) is used to learn the weights of each expert over the prediction space given some prediction locations. The basic algorithm is given in Figure 6.4.

6.4.3 Bayesian Committee Machine

The Bayesian Committee Machine (BCM) was proposed by Tresp (2000a) as an alternative method for reducing the computational complexity of an algorithm. Here the submodels are weighted by the inverse variance or precision at the prediction location. The BCM can also been used for non–Gaussian likelihood models (Tresp 2000b).

One important feature to note about the BCM is that it is a *transductive* method rather than an *inductive* method. The term *transductive* means that the method computes a model dependent on a user–specified set of prediction locations (Schwaighofer and Tresp 2003). In this way knowledge about the prediction locations is exploited in the approximation.

Quiñonero-Candela and Rasmussen (2005) and Schwaighofer and Tresp (2003) show how the BCM method is a projected process method and is equivalent to the PITC approximation discussed in Chapter 2. As stated it follows that a covariance matrix of the prediction locations needs to be constructed:

$$\Sigma_{\text{pred}} = \begin{bmatrix} k(x_1^*, x_1^*) & \dots & k(x_1^*, x_n^*) \\ \vdots & \ddots & \vdots \\ k(x_n^*, x_1^*) & \dots & k(x_n^*, x_n^*) \end{bmatrix} \tag{6.3}$$

where x^* refers to the prediction locations. The covariance function used should be noise free.

The apparent limitation of having to compute the covariance matrix (and the inverse) of the prediction locations is not too restrictive since smaller prediction covariance matrices can be created and the BCM equations can be repeatedly calculated without a growth of the algorithm complexity overhead. Another further drawback is the need to have sufficient prediction locations. Rasmussen and Williams (2006) state that the prediction accuracy will be increased with larger prediction sets and suggest that it may be necessary to hallucinate (create additional prediction locations) some prediction locations to ensure reliable results with this method. The predictive distribution equations are calculated as:

$$\hat{Z}_{\text{bcm}} = \Sigma_{\text{bcm}} \sum_{c=1}^{C} \tilde{\Sigma}_c^{-1} \hat{Z}_c \tag{6.4}$$

and the inverse BCM predictive covariance is:

$$\Sigma_{\text{bcm}}^{-1} = -(C-1)\,\Sigma_{\text{pred}}^{-1} \sum_{c=1}^{C}, \tilde{\Sigma}_c^{-1} \tag{6.5}$$

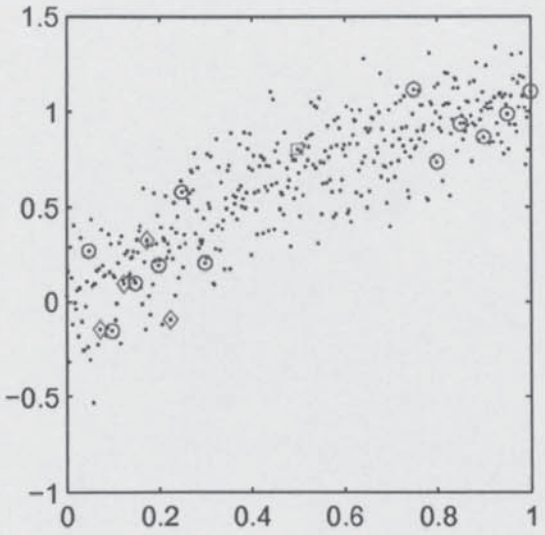


Figure 6.5: Active point locations for 1 active point (square), 4 active point (diamonds) and 12 active points (circles)

Figure 6.6: (left) BCM low rank covariance with 1 active point, (middle) BCM low rank covariance with 4 active points, (right) BCM low rank covariance with 12 active points.

where C is the number of model submodels used and Σ_{pred} is the noiseless covariance matrix between the prediction locations (Tresp 2001). An interesting observation is that the BCM predictive mean is constructed from a weighted sum of the individual submodel predictive means:

$$\hat{Z}_{\mathrm{c}} = \mathbf{k}_{\mathrm{c}}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathrm{c}}^{-1} \mathbf{Z}_{\mathrm{c}}, \tag{6.6}$$

where the matrix $\boldsymbol{\Sigma}_{\mathrm{c}}$ is the covariance matrix of the observations assigned to a submodel. The prediction locations are conditioned on the observed data assigned to a submodel c by:

$$\tilde{\Sigma} = \boldsymbol{\Sigma}_{\mathrm{pred}} - \mathbf{k}_{\mathrm{c}}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathrm{c}}^{-1} \mathbf{k}_{\mathrm{c}}. \tag{6.7}$$

Another observation is that the weights are obtained by the inverse predictive covariance (or predictive precision) at the prediction location. Effectively the BCM scales the contribution of each

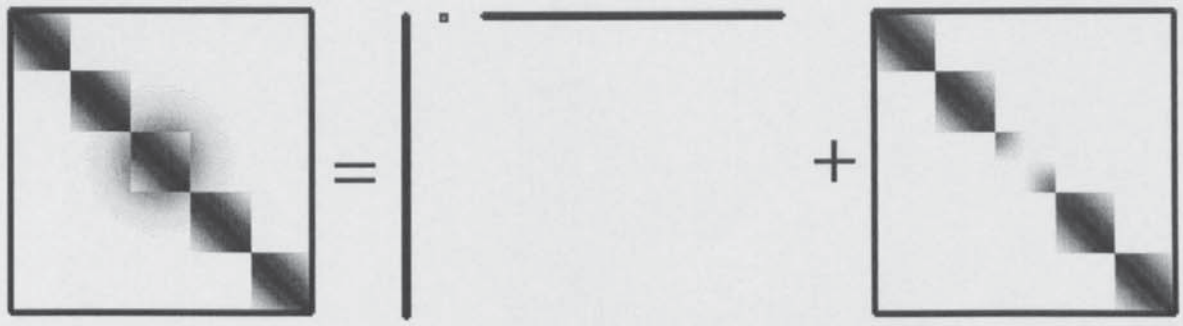


Figure 6.7: Pictorial representation of the BCM covariance matrix approximation with 1 active point.

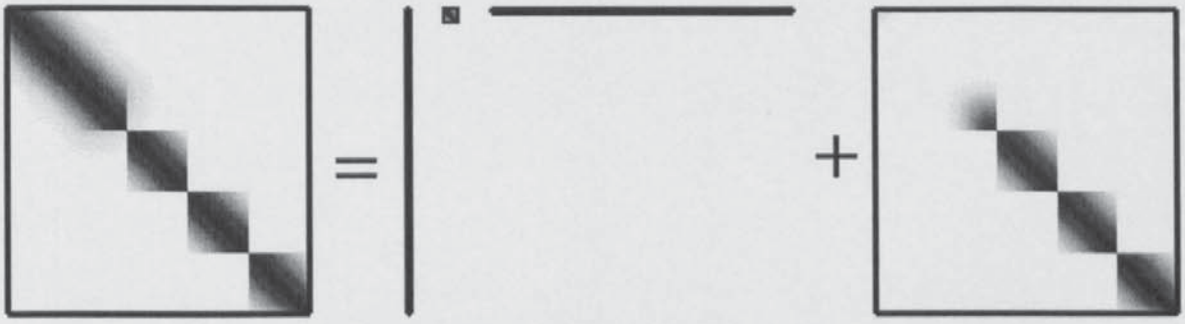


Figure 6.8: Pictorial representation of the BCM covariance matrix approximation with 4 active points.

submodel based on how confident it is about the prediction from each submodel. Substituting the individual committee members' predictive means and variances gives complete expressions for the full predictive mean:

$$\hat{Z}_{\text{bcm}} = \Sigma_{\text{bcm}} \sum_{c=1}^{C} \left(\Sigma_{\text{pred}} - \mathbf{k}_c^T \Sigma_c^{-1} \mathbf{k}_c \right)^{-1} \mathbf{k}_c^T \Sigma_c^{-1} Z_c \tag{6.8}$$

and predictive variance:

$$\Sigma_{\text{bcm}} = \left(-(C-1)\Sigma_{\text{pred}}^{-1} \sum_{c=1}^{C} \left(\Sigma_{\text{pred}} - \mathbf{k}_c^T \Sigma_c^{-1} \mathbf{k}_c \right)^{-1} \right)^{-1} . \tag{6.9}$$

Equations (6.8) and (6.9) indicate that there are a number of matrix inversions needed for this calculation. Some of these matrix inversions can be performed independently of other calculations and hence in parallel. The iterations in the sum calculation are completely independent of each other. By assigning these iterations to other processors in a parallel system it is proposed that speed–ups can be achieved since the main bottleneck in this algorithm (and many other algorithms) is the matrix inversion.

An interesting observation about the BCM is that if the committee size is fixed at 1 observation, then this becomes a transductive version of projected process approximation with exact

Figure 6.9: Pictorial representation of the BCM covariance matrix approximation with 12 active points.

diagonal (Quiñonero-Candela and Rasmussen 2005).

A pictorial example of BCM may be helpful to provide a clearer intuition of how it works and how it is related to the PITC approximation previously discussed. Suppose a dataset such as the one shown in Figure 6.5 is used and differing numbers of active points are used in representing the process: 1, 4 and 12 as indicated. These active points are selected from the prediction locations in this instance. Figures 6.7, 6.8 and 6.9 show how the effective BCM covariance matrix is constructed. As mentioned previously, there is a need to predict at more than one prediction location. The reasons for this can be seen particularly in Figure 6.7 as the BCM covariance matrix essentially tends towards a block diagonal covariance matrix. If prediction locations are selected in a particular way, the covariance matrix approximation accuracy can be increased. For a particular prediction location (left hand side of dataset in Figure 6.5), Figure 6.8 shows a good covariance approximation (top left-hand region). However for prediction at other locations in the dataset the prediction accuracy is reduced. This is not too problematic because a covariance matrix of the prediction locations in the right hand side of Figure 6.5 can then be inverted and used without having to invert the block diagonal blocks again.

To improve clarity, Figure 6.6 shows the resulting low rank covariance matrix approximations (without the exact matrix diagonal blocks) for each of the active point configurations. Choosing an appropriate size for this again depends on the complexity of the dataset.

$$\hat{Z}(\mathbf{x}_{**}) = \mathbf{k}_{*N}^{T} \boldsymbol{\Sigma}_{bcm}^{-1} Z(\mathbf{x}_N), \tag{6.10}$$

$$\sigma_*^2(\mathbf{x}_*) = \mathbf{k}_{**} - \mathbf{k}_{*N}^{T} \boldsymbol{\Sigma}_{bcm}^{-1}, \mathbf{k}_{*N}, \tag{6.11}$$

where the covariance matrix is given by:

$$\boldsymbol{\Sigma}_{BCM} = \mathbf{k}_{NT} \mathbf{K}_{TT}^{-1} \mathbf{k}_{TN} + \boldsymbol{\Lambda} + \boldsymbol{\beta}, \tag{6.12}$$

1. Master to broadcast committee parameters to each process (MPI_Bcast)
2. Master to broadcast test data locations to each process (MPI_Bcast)
3. Master to scatter training data to each process (MPI_Scatter)
4. Each node to calculate the contribution of assigned committee
5. Master to collect the mean and variance at the test locations from each process and sum results (MPI_Reduce)

Figure 6.10: Pseudocode for parallel Bayesian Committee Machine.

with the block diagonal matrix:

$$\mathbf{\Lambda} = \text{blockdiag}(\mathbf{K}_{NN} - \mathbf{k}_{NT}\mathbf{K}_{TT}^{-1}\mathbf{k}_{TN}). \tag{6.13}$$

where the subscript T has been used to denote the prediction locations. In this case $\mathbf{K}_{TT} = \mathbf{\Sigma}_{\text{pred}}$.

The two terms of the likelihood are computed in the same way as the PITC approximation in Chapter 2:

$$\mathcal{L}_1 = |(\mathbf{\Lambda} + \boldsymbol{\beta})|\,|\mathbf{K}_{TT}|\left|\mathbf{K}_{TT}^{-1} + \mathbf{k}_{TN}(\mathbf{K}_{TT} + \mathbf{k}_{TN}(\mathbf{\Lambda} + \boldsymbol{\beta})^{-1}\mathbf{k}_{NT})^{-1}\mathbf{k}_{NT}\right|, \tag{6.14}$$

and:

$$\mathcal{L}_2 = \mathbf{Z}(\mathbf{x}_N)^{\mathsf{T}}(\mathbf{\Lambda} + \boldsymbol{\beta})^{-1} - (\mathbf{\Lambda} + \boldsymbol{\beta})^{-1}\mathbf{k}_{NT}\mathbf{A}^{-1}\mathbf{k}_{TN}(\mathbf{\Lambda} + \boldsymbol{\beta})^{-1}\mathbf{Z}(\mathbf{x}_N), \tag{6.15}$$

where the only change is that the active points are also the prediction locations.

For the BCM parallel implementation, the individual committee predictive mean and predictive variance will be performed on separate processors. The calculations of the predictive mean and predictive variance require the inverse of a matrix of the same size as the number of observed data assigned to each committee. A further inversion is needed to calculate the inverse of the predictive variance which is a matrix of the same size as the number of prediction locations.

The basic algorithm for a parallel BCM is given in Figure 6.10.

Method	MAE	Time (s)
BCM (100)	0.672	1.98
BCM (200)	0.665	2.21
BCM (400)	0.661	2.44
BCM (1000)	0.655	3.64
MoE (100)	0.680	1.70
MoE (200)	0.675	2.10
MoE (400)	0.670	2.40
MoE (1000)	0.665	3.58
PP (500)	0.659	7.41

Table 6.1: Prediction accuracy and time for 1D synthetic dataset with varying subset/active set sizes.

Method	MAE	Time (s)
BCM (100)	0.838	2.91
BCM (200)	0.827	3.49
BCM (400)	0.813	4.12
BCM (1000)	0.806	5.74
MoE (100)	0.841	2.70
MoE (200)	0.833	3.12
MoE (400)	0.820	3.86
MoE (1000)	0.812	5.44
PP (500)	0.822	7.88

Table 6.2: Prediction accuracy and time for 2D synthetic dataset with varying subset/active set sizes.

6.4.4 Ensemble method results

Now that parallel algorithms are being used, the computation time is significantly reduced. Each algorithm was executed 100 times, and the average time was recorded. The MAE remained constant across each run since the same data and parameters were used.

Prediction error and subset size

The results for the three different datasets are tabulated in Tables 6.1, 6.2 and 6.3. In the first column the name of the method is followed by the size of each data subset. A projected process algorithm running on a single processor with 500 active points was used to act as a benchmark for comparison. A setup to mimic a dual–core processor was used. A familiar pattern is emerging, the faster algorithm has a reduced prediction accuracy. The algorithm increased prediction accuracy is slower. The differences in speed are not significant for such small datasets.

Method	MAE	Time (s)
BCM (100)	0.812	2.95
BCM (200)	0.805	3.61
BCM (400)	0.802	4.24
BCM (1000)	0.798	5.69
MoE (100)	0.837	2.72
MoE (200)	0.830	3.21
MoE (400)	0.822	3.76
MoE (1000)	0.815	5.51
PP (500)	0.827	8.02

Table 6.3: Prediction accuracy and time for Walker lake dataset with varying subset/active set sizes.

Processors	100	200	400	1000	2000
1	1.06	3.74	12.45	57.75	212.92
2	2.01	3.84	11.23	29.23	108.71
4	2.12	4.01	8.48	15.64	55.23
8	2.34	4.22	5.56	8.01	27.59

Table 6.4: Table showing how the computation speed changes as the number of processors changes for BCM.

Number of processors and computation speed

Since the effects of changing the subset size can be understood from Tables 6.1, 6.2 and 6.3, attention is turned to the timings associated with differing numbers of processors and subset size. The key information here is time taken, since the prediction accuracy will be the same whether computed with 1 processor or with 1000 processors. A large dataset has been used. Again, a random subset of 30,000 observations from the Walker dataset is selected. The BCM algorithm is used here, since the scaling to more processors will be similar for the MoE algorithm. Table 6.4 shows significant results for computational performance increases. By looking in the 2000 observations per subset column, processing on 8 processors reduces computation by nearly a factor of 8. Curiously for smaller subset sizes, no improvements are found, indeed the opposite seems to be the case. This is due to the overhead of using a parallel architecture. The communication time to send the data to each of the nodes is significant when large datasets are being used.

Summary

Near linear (in terms of number of processors) speed-ups can be achieved by applying the algorithms discussed in this section. Treating large datasets becomes possible in a principled way. One further improvement could be made. The sparse matrix methods from Chapter 4 could be introduced. Load balancing issues could ensue because each subset could have a different sparsity structure. Since the BCM is effectively a projected process method, the same prediction accuracy can be expected that is obtained with sparse matrix methods and projected process methods.

6.5 Variogram Parameter Estimation

Estimating parameters for a particular model is important. Parameters not only affect prediction but also uncertainty measures calculated by the model. In some cases the parameters may be assumed to be known, but in a large range of situations, this is not the case.

6.5.1 Method-of-Moments

Being able to split the computation of the variogram into smaller parts that could be performed on other processors would be convenient. The computation of a method-of-moments variogram is not considered a complex procedure when compared to the matrix inversion needed to calculate the kriging weights. Computing the empirical variogram is an $\mathcal{O}\left(n^2\right)$ operation and hence for most reasonable datasets not an issue. If the dataset were of large proportions then it may be necessary to look at a parallel solution, not only in terms of computation speed but also in terms of the memory requirement. The empirical variogram equation,

$$\hat{\gamma}\left(\mathbf{x}\right) = \frac{1}{2\mathrm{N}\left(\mathbf{h}\right)} \sum_{\mathrm{N}(\mathbf{h})} \left\{\mathrm{Z}\left(\mathbf{x}_i\right) - \mathrm{Z}\left(\mathbf{x}_j\right)\right\}^2, \tag{6.16}$$

where:

$$\mathrm{N}\left(\mathbf{h}\right) = \left\{\left(\mathbf{x}_i, \mathbf{x}_j\right) : \mathbf{x}_i - \mathbf{x}_j = \mathbf{h}; i, j = 1, \ldots, n\right\} \tag{6.17}$$

shows it is trivial to produce a parallel version. The simple sum can be split into independent parts since the calculation of the current iteration does not depend on the result of the previous iteration in the summing process. The main issue here is to split the data into sensible blocks

```
1. Master to broadcast number of lags and lag separation to each
   process (MPI_Bcast)
2. Master to scatter training data to each process (MPI_Scatter)
3. Each node to bin assigned two blocks of data into lags
4. Master to collect data at each lag (MPI_Reduce)
```

Figure 6.11: Pseudocode for Method–of–Moments variogram.

that each processor can compute. For large datasets, it would be time consuming to send each processor all the data; it would be better if only the necessary data were sent.

The distance matrix for the observations is symmetric, so further speed–ups could be obtained by recognising this fact when assigning data to different processors. Calculating the empirical variogram is an example of a method which can be parallelised without significant programming effort. Each process should be assigned two blocks of observations between which to compute the semi–variance.

Assuming the number of lags is not of large proportions, fitting the variogram to the data can then be performed on a single processor. A basic algorithm for this can be computed as shown in Figure 6.11.

6.5.2 Maximum Likelihood

To use maximum likelihood methods, it is convenient to calculate the log likelihood:

$$\mathcal{L}(\theta) = -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma(\theta)| - \frac{1}{2}Z(x)^T\Sigma(\theta)^{-1}Z(x) \tag{6.18}$$

where $\Sigma(\theta)^{-1}$ is the covariance of the data $Z(x)$ as a function of the parameters θ. Using an optimisation algorithm, the parameters θ are iteratively updated to give better estimates for the parameters of the variogram. This process is computationally intensive particularly due to the need to invert the covariance matrix $\Sigma(\theta)^{-1}$ of all the observations at each iteration.

Block Diagonal Covariance Matrix

Earlier in this chapter, some techniques for approximating the likelihood were reviewed. One such technique relied on making the covariance matrix block diagonal. Figure 6.12 shows how

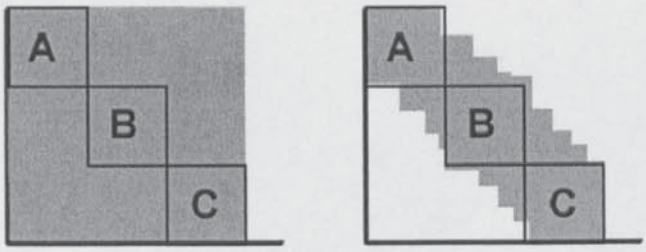


Figure 6.12: Shows how a block diagonal structure can be imposed on the covariance matrix. Three block diagonal elements are imposed. (left) typical covariance matrix, (right) covariance matrix calculated using space–limited covariance function.

the block diagonal technique can be improved by applying space–limited covariance functions to increase the sparsity in the covariance matrix. The amount of information discarded is a lot smaller when a space–limited covariance function is used. Imposing the block diagonal condition on the covariance matrix is equivalent to summing the log likelihood of each block diagonal element individually (see Appendix A.5). This can be considered as equivalent to dividing the dataset into subregions and calculating the likelihood for each subregion separately. Stein (1986) recommended using subregions that contained at least 100 observations. In doing so, little information about the parameters governing the local behaviour of the process will be lost. In addition to the reduction in computational complexity, it is noted that this technique has other added advantages. It could be considered desirable that subregions are explored in this way so as to identify possible nonstationarities in the data (Stein 1999). For example, in moving–window kriging, only observations near the prediction location are used, which allows for nonstationarities in the data (Haas 1995).

This particular method would work particularly well with the parallel Bayesian Committee Machine implementation mentioned earlier in this Chapter. Since each computer process has a subregion assigned to it, calculating the log likelihood of the whole model is just the sum of the individual log likelihoods of each committee. This can be implemented with *MPI_Reduce* subroutine to automatically collect and sum the log likelihoods. An outline algorithm is given in Figure 6.13.

1. Master to broadcast covariance parameters to each process (MPI_Bcast)
2. Master to scatter training data to each process (MPI_Scatter)
3. Each node to calculate likelihood of assigned subregion
4. Master to collect log likelihoods and sum(MPI_Reduce)

Figure 6.13: Pseudocode for parallel independent likelihoods.

Approximate Likelihood

One approach proposed by Vecchia (1988) suggests approximating the likelihood by an alternative method. The approximation is based on the multiplicative theorem which states for any number of N events: $z_1, z_2, \ldots, z_N$ the following relationship holds:

$$p(z_1 \cap z_2 \cap \ldots \cap z_N) = p(z_1) \cdot p(z_2|z_1) \cdot \ldots \cdot p(z_N|z_1, z_2, \ldots, z_{N-1}), \tag{6.19}$$

where $p(z_a|z_b)$ is the conditional probability of z_a given z_b (Pardo-Igúzquiza and Dowd 1997).

In the case of a multivariate probability density function, the following relationship is obtained:

$$p(Z(\mathbf{x})) = \prod_{i=1}^{N} p(Z(\mathbf{x}_i)|Z(\mathbf{x}_1), \ldots, Z(\mathbf{x}_{i-1})). \tag{6.20}$$

One then assumes that some of the information in the dataset is redundant and hence instead of conditioning on the whole dataset the observations are conditioned on smaller subsets of size $m < i-1$ where i is the current observation of the dataset. This gives the following relationship:

$$p(Z(\mathbf{x}_i)|Z(\mathbf{x}_1), \ldots, Z(\mathbf{x}_{i-1})) \cong p(Z(\mathbf{x}_i)|Z(\mathbf{x}_1), \ldots, Z(\mathbf{x}_m)). \tag{6.21}$$

where the approximation becomes almost exact as m approaches the number of observations in the dataset.

Assuming that the data is a zero mean multivariate Gaussian, the conditional probability $p(Z(\mathbf{x}_i)|Z(\mathbf{x}_j))$, where $j = 1, \ldots, m$ is also Gaussian for any observation $\mathbf{x}_i$ and any conditioning subset size m and is given by :

$$\mathcal{N}\left(-\Sigma_{ij}\Sigma_{jj}^{-1}t_j, \Sigma_{ii} - \Sigma_{ij}\Sigma_{jj}^{-1}\Sigma_{ji}\right). \tag{6.22}$$

The following give the mean:

$$\mu_{i|j} = -\Sigma_{ij}\Sigma_{jj}^{-1}Z(\mathbf{x}_j) \tag{6.23}$$

and covariance

$$\Sigma_{i|j} = \Sigma_{ii} - \Sigma_{ij}\Sigma_{jj}^{-1}\Sigma_{ji} \tag{6.24}$$

conditioned on a subset of j observations where Σ_{jj} is a $j \times j$ covariance matrix between the points of vector y_j, Σ_{ij} is a vector of covariances between the ith observation and m points of the vector $Z(\mathbf{x}_j)$ and $Z(\mathbf{x}_j)$ are m observations at locations chosen for each subset.

This leads to the following log likelihood approximation:

$$\mathcal{L}(\theta) = -\frac{N}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{n}\log\left|\mathbf{\Sigma}(\boldsymbol{\theta})_{i|j}\right| - \frac{1}{2}\sum_{i=1}^{n} Z(\mathbf{x})^{T}\mathbf{\Sigma}(\boldsymbol{\theta})_{i|j}^{-1} Z(\mathbf{x}), \tag{6.25}$$

which instead of depending the inverse of a covariance matrix $\mathbf{\Sigma}(\boldsymbol{\theta})^{-1}$ of size N, depends on i covariance matrices of maximum size m. Hence the smaller size m the more computationally the algorithm is but at the expensive of yielding a poorer approximation to the *true* probability density function.

Although Vecchia (1988) notes that the order of the data makes a difference to the approximation, this is not considered a significant issue and it is not dealt with. A number of years after this approximation method was proposed, Stein et al. (2004) suggested a number of improvements to the algorithm. Firstly it is suggested that the approximation gives better results when the observations are ordered so as to give clustered data. Secondly, by not only conditioning on observations near, but also on some observations far away, the approximation is further improved.

Since the approximate maximum likelihood approach has reduced the calculation to a sum of a number of independent calculations, a parallel implementation follows trivially. A further desirable feature is that all the data need not be sent to each process in the parallel system. How much data sent to each process depends on m, the size of the conditioning subset. Particularly accurate approximations to the likelihood can be achieved with large m. The basic algorithm for a parallel architecture is shown in Figure 6.14. To test the two parallel methods of prediction: BCM and Mixture of Experts, timings were run with 2, 4 and 8 processor versions. The dataset used for the experiments was the Walker lake data as described in Chapter 3. As in previous chapters, a section of the data was set aside so that cross–validation could be performed and the MAE could be calculated.

The prediction accuracy results for the parallel methods were obtained by varying the number of observations in each submodel in the committee or the number of observations that were in

1. Master to broadcast covariance parameters to each process (MPI_Bcast)
2. Master to scatter training data to each process (MPI_Scatter)
3. Each node to calculate likelihood of each supplied observations conditioned on a subset of the data
4. Master to collect log likelihoods and sum (MPI_Reduce)

Figure 6.14: Pseudocode for Vecchia approximation

assigned each expert. The plot shows that by doubling the number of processes, the computation time seems halve to obtain the same prediction accuracy. It is intuitive to believe that it will not exactly halve because of the overhead of interprocess communication. Comparatively, it seems that the communication overhead is negligible which is to be expected from the system that was used for the experiments since the message passing software used shared memory for interprocess communication. It would be interesting to see the scaling of the algorithm on a loosely coupled system or web–based grid architecture.

Figure 6.15 shows how the different methods compared. To provide a baseline, a projected process kriging approximation (DTC) was used to compare the results of the two parallel methods. The projected process kriging approximation was executed on a simple processor computer. The number of active points used was varied to obtain results for different prediction accuracies and time. Figure 6.15 shows prediction time plotted against predication accuracy (MAE). The plot gives an idea of which algorithm effectively offers the greater accuracy given available time. The projected process algorithm is at a severe disadvantage for this comparison because the number of active points used was too low for modelling the data, since obtaining increased prediction accuracy would be at the expense of increased computation time.

The ability to increase the number of processors and at the same time maintain a prediction accuracy of the same magnitude is evident looking at the plot. For large–scale problems, using parallel BCM can facilitate their practical application. The MoE showed less accurate cross–validation performance for the predictions. However, the one advantage here is that each cluster learned separate parameters, so in a sense non–stationary phenomenon can be modelled.

6.6 Conclusion

In this Chapter a number of parallel algorithms suitable for application to Geostatistics have been presented. The results show that the BCM approach seems to generally give the better prediction accuracy and computation speed when compared to the MoE method. The potential drawbacks of having to know the prediction locations beforehand is not problematic for many situations. However, being able to perform maximum likelihood parameter estimation given specific prediction locations may raise issues for predicting at large numbers of locations. Alternative parallel methods for determining the model parameters were discussed also.

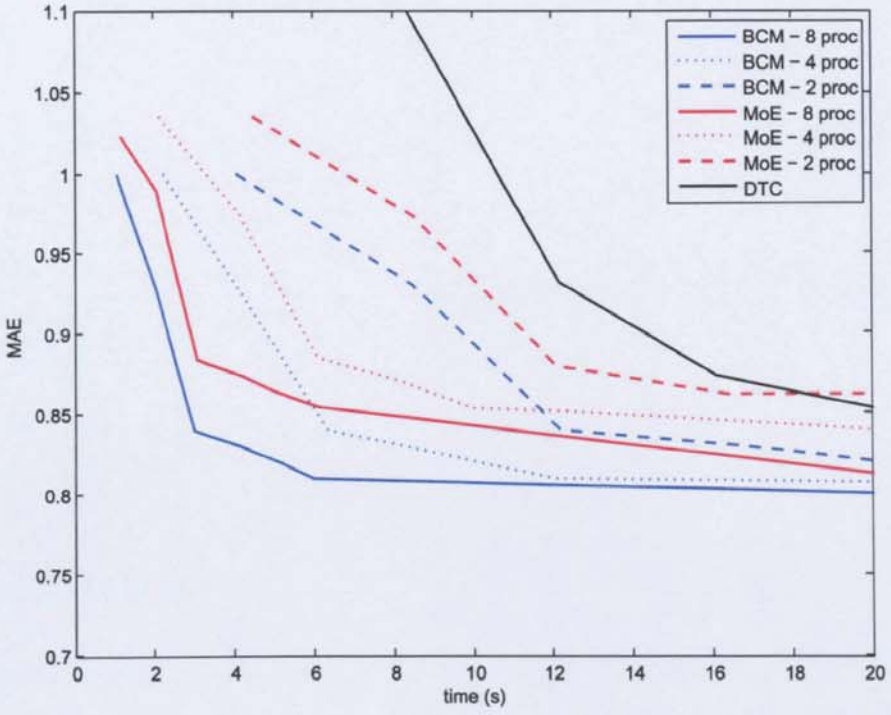


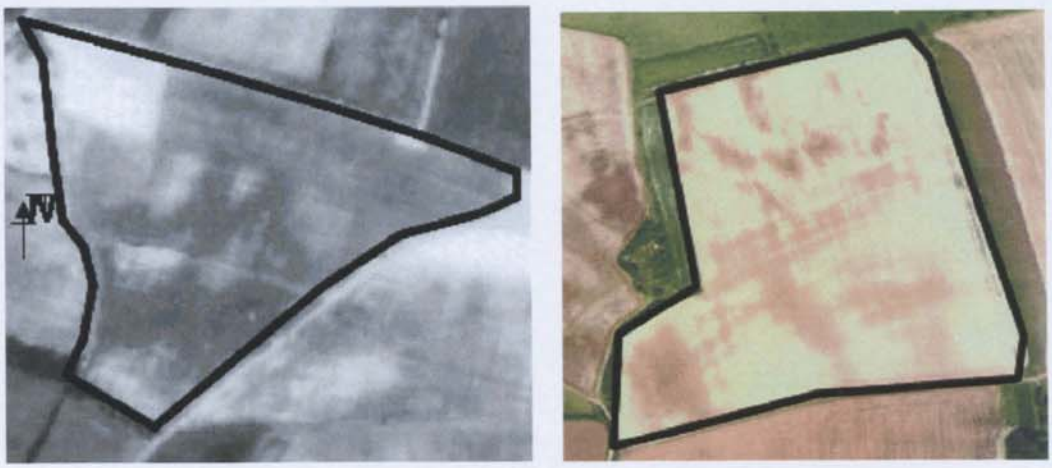
Figure 6.15: Plot showing comparing the performance of the BCM and MoE algorithms with a baseline projected process kriging using various numbers of active points. The horizontal axis is time measured in seconds and the vertical axis is MAE. The red lines are the MoE results, the blue lines are the BCM results and the solid black line is the projected process algorithm.

7

Sample Design Configuration

7.1 Introduction

The sequential projected process kriging algorithm presented in Chapter 2 includes a framework for automatically selecting observations that are informative or representative of the dataset. In this Chapter a further application of sequential projected process kriging is presented. Projected process kriging has been shown to be efficient for maximum likelihood estimation and prediction for large spatial datasets. Soil scientists rarely have large densely sampled datasets unless ancillary data are being used to make inferences about the soil properties. It will be shown how dense ancillary data, such as an aerial photograph, can be used to identify optimal sampling locations that best capture important features for computing soil maps.



(a) (b)

Figure 7.1: Aerial photo data from standard surveys scanned at 75 dpi to give a ground pixel size of 3.4 m and geo–corrected to UK ordnance survey coordinates. a) Wallingford, Oxfordshire. b) Yattendon, Berkshire.

7.2 Datasets

Soil data available at two sites in southern England: Wallingford, Oxfordshire and Yattendon, Berkshire were used. Aerial photos (Figure 7.1 of the fields were obtained and geocorrected to UK ordnance survey coordinates. Greyscale values were extracted from the photos (Kerry 2004).

7.2.1 Wallingford

At the Wallingford site, data were collected at 296 locations on a 30 m grid. Properties such as soil depth, clay content, stoniness, soil pH, loss of ignition (LOI), etc... were measured. For the examples in this Chapter, the soil depth data will be used. Figure 7.2 shows the traditional grid–based subsampling schemes that have been used.

7.2.2 Yattendon

At the Yattendon site, data were collected at 118 locations on a 30 m grid. As with the Wallingford site, many soil properties were measured. For the examples with the Yattendon dataset in this Chapter, the clay content data will be used. Figure 7.3 shows the traditional grid–based subsampling schemes that were used at this site.

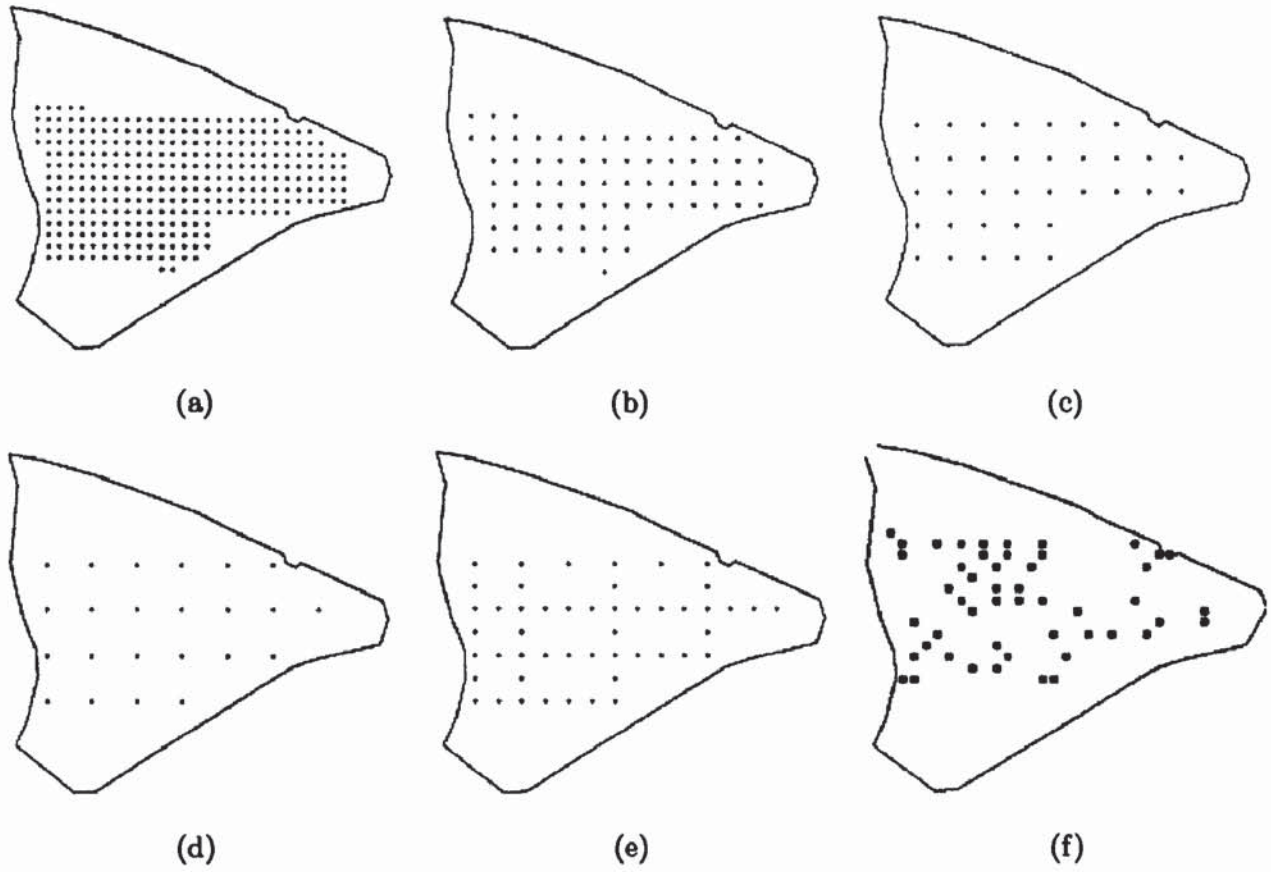


Figure 7.2: Sampling schemes for Wallingford field: (a) 30 m grid (296 observations), (b) 60 m grid (70 observations), (c) 90 m grid (36 observations), (d) 120 m grid (23 observations), (e) 120 m + 60 m grid (50 observations), (f) Subset selected by PPK method.

7.3 Methodology

Ancillary data often vary in similar ways to soil data but can have complex relationships between them and can be difficult to interpret in terms of actual values of soil properties. Soil samples should always be collected and analysed, but ancillary data can give insight into how this should be done. Selecting how such soil samples are collected for geostatistical analysis has been the topic of much research.

Samples are usually collected on a grid or a nested grid system and several studies have investigated what constitutes a suitable sampling interval (McBratney and Webster 1981). When sampling intervals are large, some patterns of variation in the soil can be missed due to the configuration of sampling locations and, therefore there may not be sufficient samples to estimate a reliable variogram (Frogbrook and Oliver 2000). Webster and Oliver (1992) showed that a

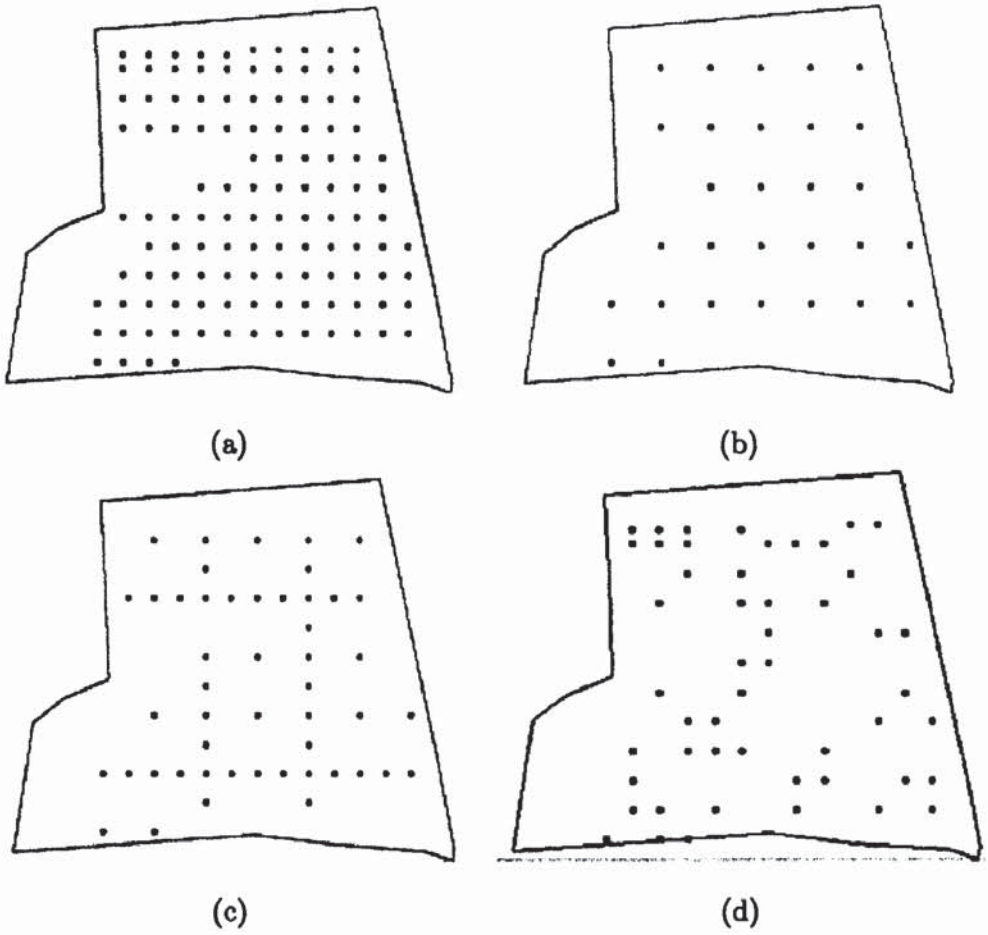


Figure 7.3: Sampling schemes for Yattendon field: (a) 30 m grid (118 observations), (b) 60 m grid (29 observations), (c) 60 m + 30 m grid (50 observations), (d) Subset selected by PPK method.

reliable method–of–moments variogram needs about 100 samples. Later it was shown by Lark (2000) that if a maximum likelihood variogram was required then fewer samples were needed. Kerry and Oliver (2007) suggests that this be about 50 samples. Figures 7.4–7.8 show maps computed using method–of–moments variograms. Comparing the method–of–moments maps to those generated using maximum likelihood (Figures 7.9–7.12) shows vast improvements in the quality of the maps in the sense that features of the dataset are retained and visible in the generated map. When maximum likelihood was applied to the 120 m gridded data which has only 23 samples, the algorithm failed to converge to parameters. The inability to determine parameters is not problematic as Kerry and Oliver (2007) have shown that at leave 50 samples are needed to obtain reliable parameter estimates.

Two questions need to be answered: What is the optimal sampling configuration for prediction at unobserved locations? What is the optimal sampling configuration for estimating

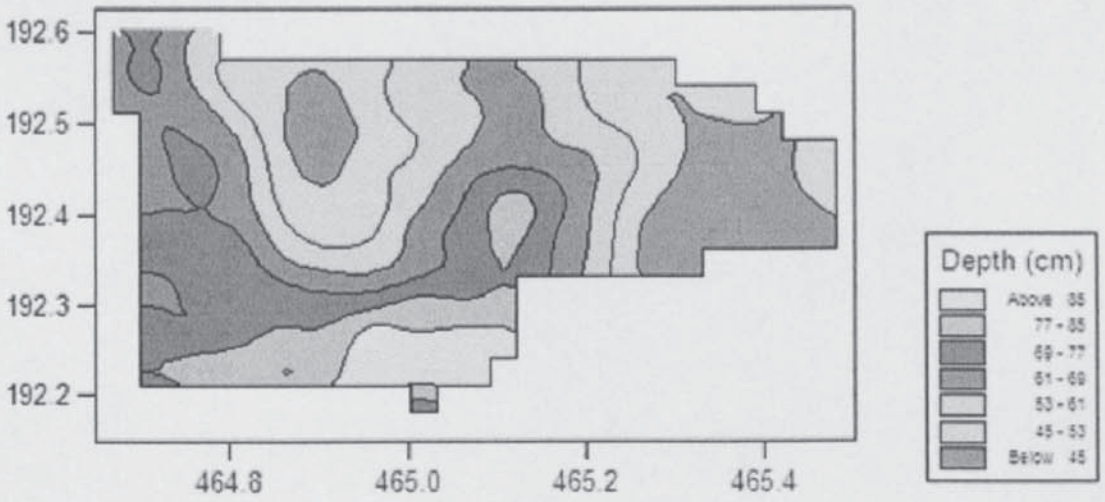


Figure 7.4: Maps for Wallingford using 30 m gridded data and MoM parameter estimation

covariance function parameters? The two questions cannot often be mutually satisfied. The method presented here avoids relying only on minimising uncertainty in the model such as suggested by (Zidek et al. 2000).

The best sampling configuration for computing a maximum likelihood variogram requires further investigation. In the situation where ancillary data is not available, Brus and Heuvelink (2007) suggest a method that attempts to minimise the global kriging variance. The methodology presented here is related to this idea. Aerial photography may not be of much use, due to each of the images having over 10,000 pixels representing the reflectance of the soil which would normally restrict parameter estimation to method–of–moment based estimators.

The projected process kriging algorithm was initially developed for the interpolation of spatial data. Within the framework of the algorithm are a number of interesting features. As has already been mentioned, selecting the *best* subset of the data to include in the active set is not a trivial activity. Ideally, the active set should contain those observations that are maximally informative about the underlying process.

The active set can be selected based on which observations reduce the predictive variation most in the model. Likewise, sample locations in sample configuration problems can be selected using the same measure. Alternative measures have been suggested which include the predictive mean also. Instead of following the global optimisation techniques used by Heuvelink et al. (2006) which can be extremely time consuming (running into days), a sequential kriging algo-

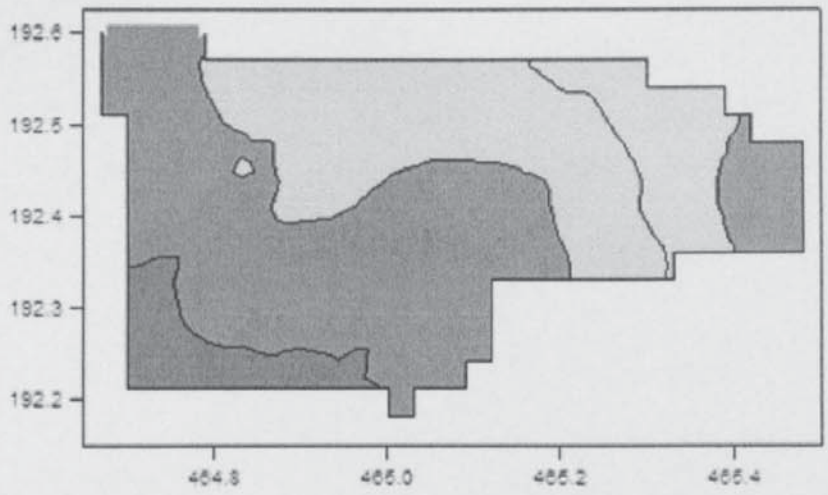


Figure 7.5: Maps for Wallingford using 60 m gridded data and MoM parameter estimation

rithm could be used which, although cannot guarantee a globally optimal solution, can provide a good estimate quickly (minutes or hours). It should be noted that simulated annealing techniques cannot guarantee a globally optimal solution either. Heuvelink et al. (2006) does not consider the situation where ancillary data are available.

Often large datasets of ancillary data are used to inform a geostatistian about decisions in the sampling process. Where ancillary data exists, such as in the form of aerial photography, optimal sampling locations can be inferred. This inference assumes some relationship between the processes evident in the ancillary data and the processes in the property being observed. These relationships are likely to be complex and poorly understood. Because of the earlier assumption that there exists some relationship between the ancillary data and the true process, these active points can be used as the sample configuration.

Furthermore, the ancillary data facilitates the alignment of the sample configuration to locations where interesting features occur in the data. This alignment would not be possible if ancillary data were not used.

For the available data that are used within this Chapter, there are only soil data available at a small subset of the locations in the aerial photograph, hence the subsample locations need to be constrained to the locations where soil data are available. This can be done with the projected process algorithm by fixing the active set to be at the locations where soil data are available. Then the data from the aerial photograph are iteratively projected onto these active points.

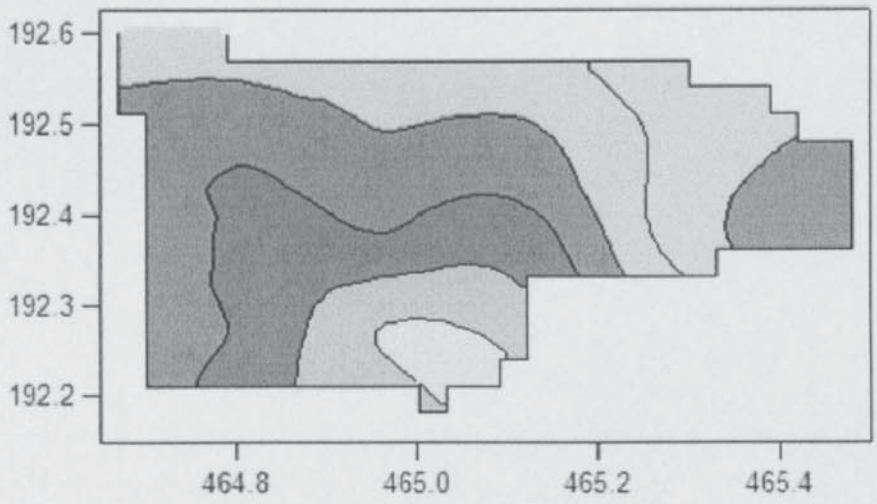


Figure 7.6: Maps for Wallingford using 90 m gridded data and MoM parameter estimation

After the entire aerial photography data has been projected, active points can be removed from the active set projecting the process on to the remaining active points. This can be repeated until the desired active set size is obtained. The remaining locations in the active set determine the optimal sampling configuration.

To test this methodology, soil data from the determined sampling locations was used to krige at a subset of the locations remaining where soil data were available.

7.3.1 Measuring informativeness

The process of iteratively removing locations from the active set is quick. As was discussed in Chapter 1, the criterion for selecting a specific location can be based on a number of measures. One simple measure would be to calculate which location reduced the predictive variance by the smallest amount. This is equivalent to the measure suggested by Heuvelink et al. (2006). Calculating solely the variance relies only on the spatial locations of the observations, not on the value of the observation. One key motivation of this work is that the chosen sample configuration aligns with important features in the dataset hence the value of the observation will also be used. By modelling local variations in the mean function, important features can be retained. Appendix B gives a thorough review of the Bayesian projected process framework. Equation (B.11) is the measure used to determine which observations are the most informative.

Seeger and Williams (2003) suggests that "all things equal, select the observation which

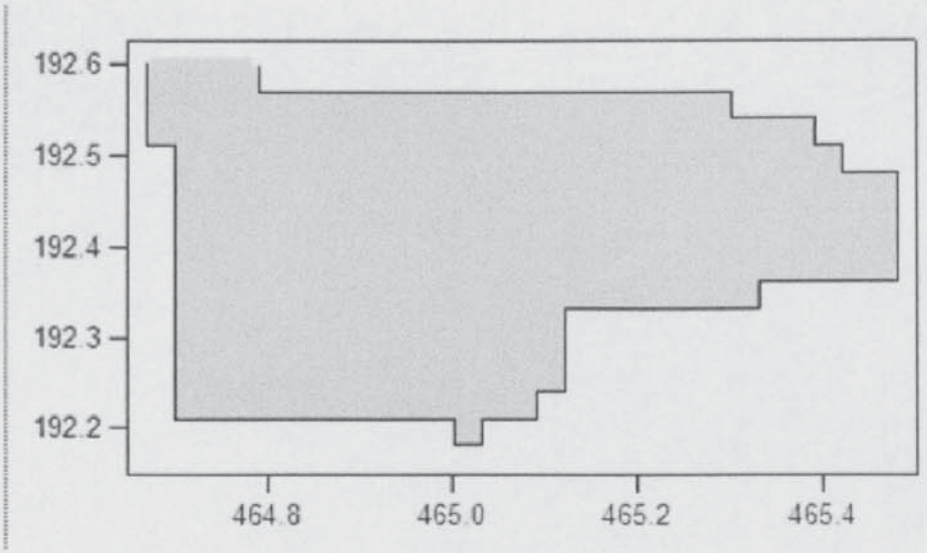


Figure 7.7: Maps for Wallingford using 120 m gridded data and MoM parameter estimation

deviates furthest from the predictive mean." In doing so, observations which provide a surprise or novelty to the model are given priority over those that do not. This is the heuristic measure that is used for these experiments. Csató and Opper (2002), Seeger (2003) derive this measure of information with more mathematical rigour. They show how it is equivalent to approximating a distribution by minimising the KL–divergence to the distribution of interest. The mathematical details of this scoring measure are given by Equation (B.11).

7.3.2 Sampling density analysis

Lark (2000), Kerry and Oliver (2007) suggest that 50–60 samples is sufficient for calculating a maximum likelihood variogram. An interesting feature of the sequential projected process kriging framework is the ability to iteratively reduce the size of the active set. The active set has the effect of the entire dataset projected onto it. Figures 7.18 and 7.19 show how the prediction error increases as the size of the active set is decreased. An intuition can be gained, by looking at the curves, about what a good subsample size would be. Such a curve can be computed quickly and is invaluable when deciding how many sample locations should be obtained.

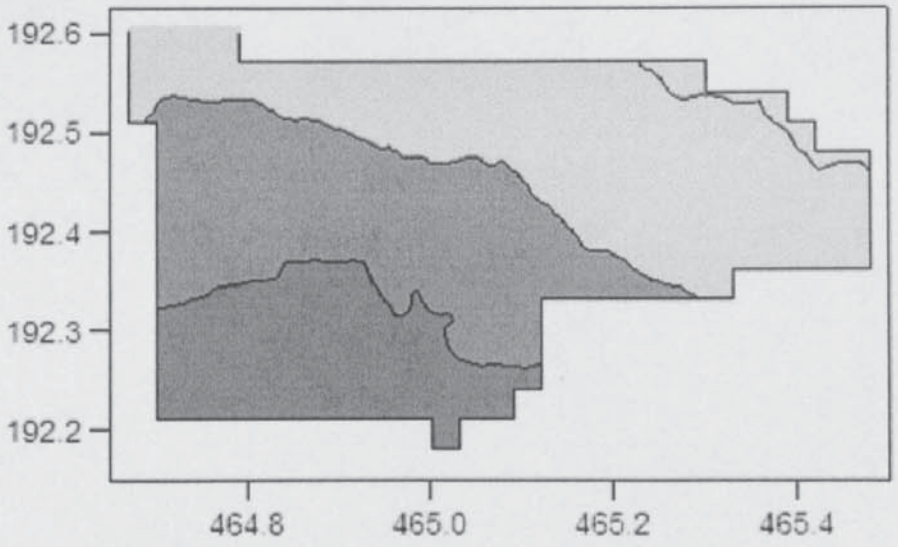


Figure 7.8: Maps for Wallingford using 120 m + 60 m gridded data and MoM parameter estimation

Sampling scheme	MAE	MSDR
60 m grid	19.05	1.21
90 m grid	24.79	4.83
120 m + 60 m grid	23.68	4.12
PPK Best 50	18.97	1.19
Random Best 50	18.27	1.20
Random 50 mean (stdev)	22.84 (1.91)	1.8443 (0.67)

Table 7.1: Table showing the prediction results using different sampling schemes at Wallingford site.

7.3.3 Benchmark comparison

Due to the datasets available, there is no currently available benchmark technique for selecting a subset of the data informed using ancillary data. Hence, to provide a comparative study of this technique, random subsets of the available data will be used. As previously discussed, 50 samples are considered to be sufficient for computing a maximum likelihood variogram. A subset of 50 samples from the available soil data will be selected to compute a variogram. Using this model, prediction will then be performed at the remaining locations in the dataset. The process of selecting a random subset of 50 samples will be repeated 100 times to give an indication of the benchmark performance.

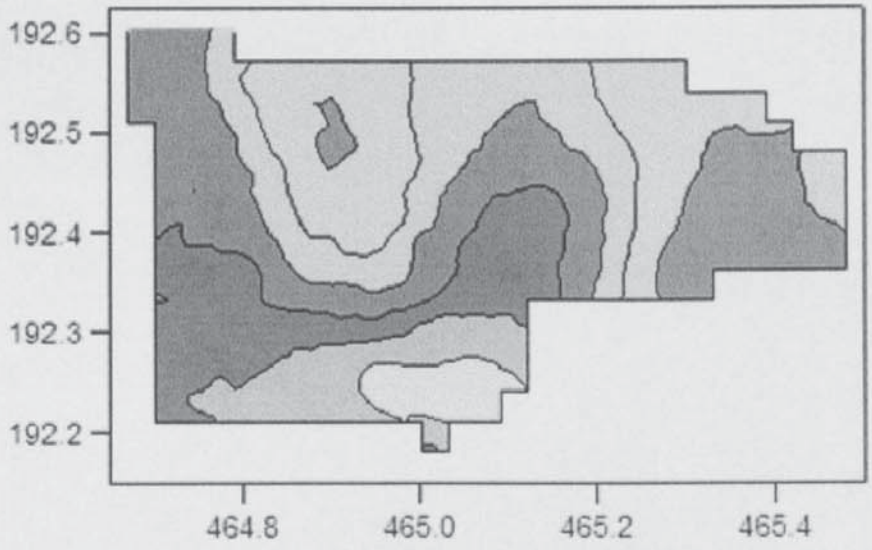


Figure 7.9: Maps for Wallingford using 30 m gridded data and ML parameter estimation

Sampling scheme	MAE	MSDR
60 m grid	27.57	4.70
60 m + 30 m grid	21.09	2.08
PPK Best 50	16.05	1.68
Random Best 50	16.17	1.82
Random 50 mean (stdev)	25.79 (3.02)	2.55 (0.96)

Table 7.2: Table showing the prediction results using different sampling schemes at Yattendon site.

7.4 Results

Figure 7.2f and Figure 7.3d show the sampling schemes obtained by using the projected process method. Both sampling schemes show areas where the data have been more densely sampled so that the important features in the data are captured.

Looking at Table 7.1, which records the MAE (Mean Absolute Error) and the MSDR (Mean Squared Deviation Ratio) from cross–validation, shows how the 50 locations selected by the projected process method outperform all sampling configurations, even the Wallingford 60 m grid size which uses 70 observations on a regular grid. The best random subset selected from the 100 random subsets does give a lower MAE than the PPK selected subset. Looking at the mean MAE and MSDR for the 100 random subsets shows that the PPK method performs better than random.

Visually inspecting the Wallingford maps seems to confirm the tabular results. Assuming

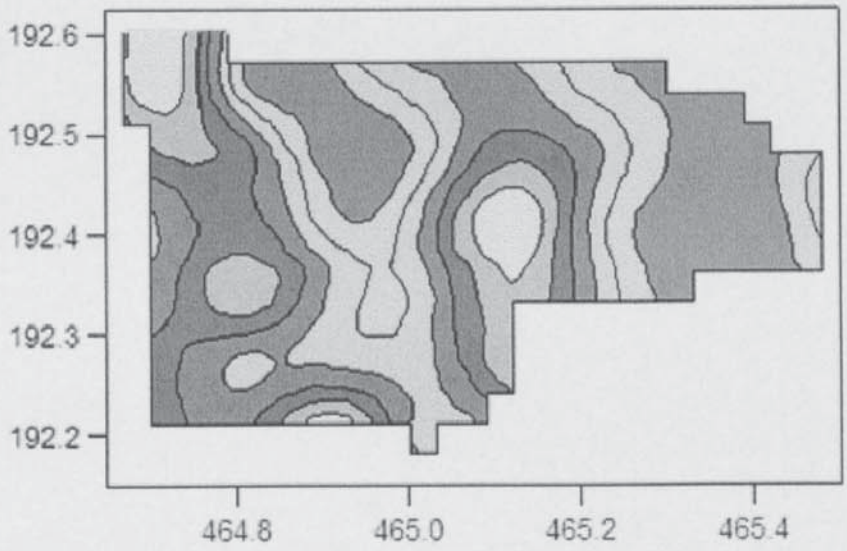


Figure 7.10: Maps for Wallingford using 60 m gridded data and ML parameter estimation

that the 297 observations on the 30 m grid are used to created a map (Figure 7.9), then comparing the map with the projected process method sampling configuration (Figure 7.13) shows many similar features. The 120 m + 60 m grid configuration with 50 observations (Figure 7.12), has the same number of datapoints as the projected process method, but does not have all the features that are present in the 30 m grid map. The projected process method in Figure 7.13 seems to have smoothed the map compared to the 30 m grid sampled map (Figure 7.9). Figure 7.20 shows a plot of observations against predicted values. This plot does not show any clear bias in the predictions obtained.

The same seems to hold for the Yattendon maps too. Figure 7.17, the projected process configuration, shares the most features with Figure 7.14, which was generated from the 30 m grid with more than 3 times the data.

The Yattendon tabulated results recorded in Table 7.2 show similar results to those from the Wallingford site. Except this time the PPK selected best 50 give lower MAE and better MSDR when compared to the best random subset selected. Figure 7.21 shows the observations against the predications for the selected subset locations. The plot indicates that there is not any serious bias in the predictions.

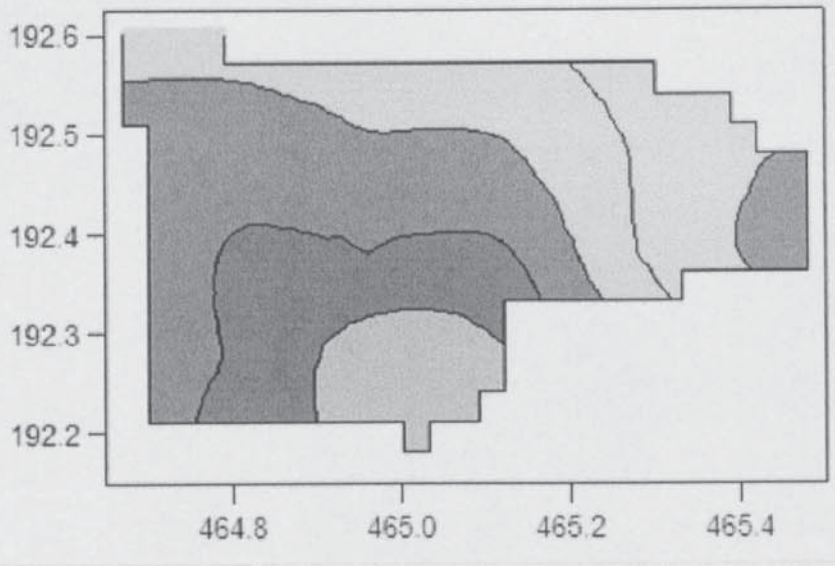


Figure 7.11: Maps for Wallingford using 90 m gridded data and ML parameter estimation

7.5 Conclusion

In this Chapter it has been shown that the sequential projected process framework introduced in Chapter 2 can be applied to the problem of sample configuration design with ancillary data. Instead of using a slow global optimisation algorithm, which is not even guaranteed to converge to global minima, a flexible but fast iterative algorithm is used. Although the sequential projected process method cannot guarantee globally optimal sample configurations, it has been shown that by its application, improvements over randomly selecting subsets or imposing regular grids can be made in determining sampling locations when ancillary data are available in a significantly reduced time. Examining the contours of the maps show that the projected process method captures many of the features visible in the most dense grids

In this example, the sampling locations were limited to where soil data was already available so that an analysis of the method could take place. Normally, the sampling locations would not be constrained since the object of the study is to determine where to collect soil samples. This added flexibility would further improve generalisation performance.

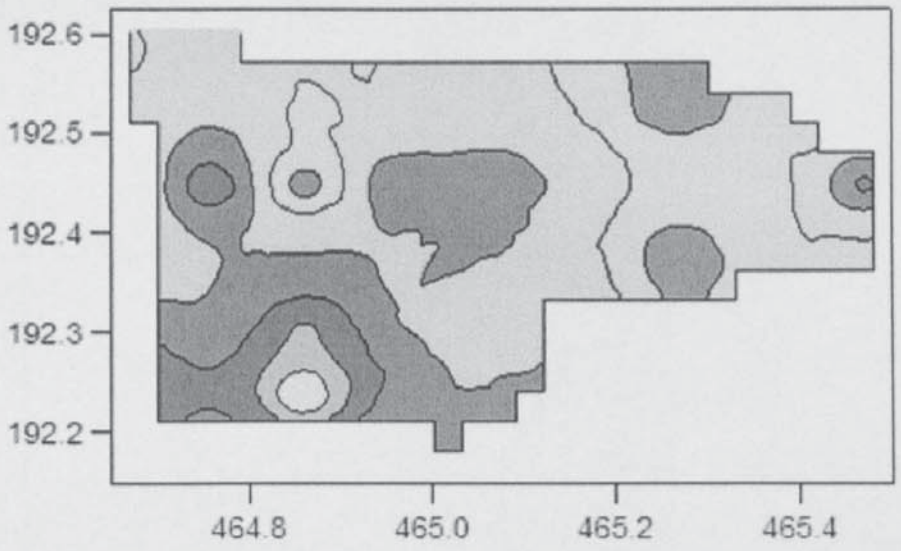


Figure 7.12: Maps for Wallingford using 120 m + 60 m gridded data and ML parameter estimation

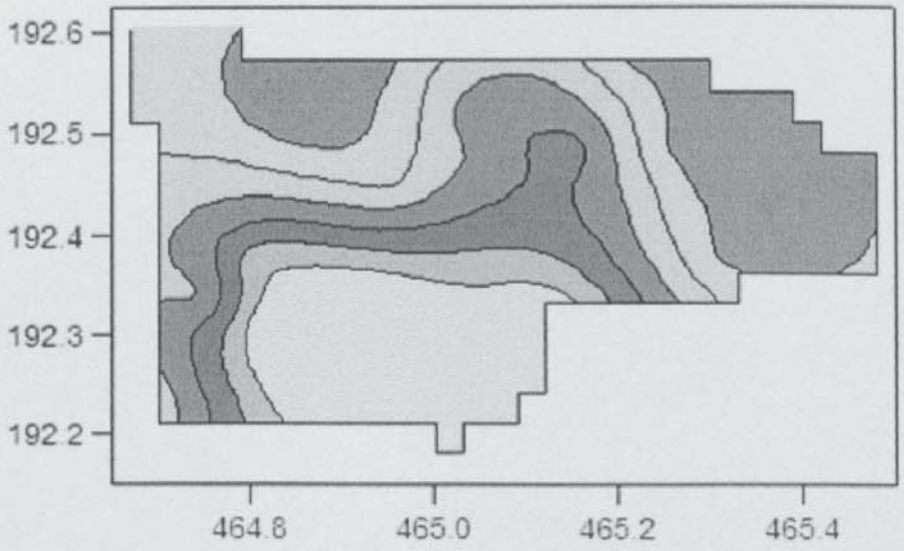


Figure 7.13: Maps for Wallingford using PPK selected data and ML parameter estimation

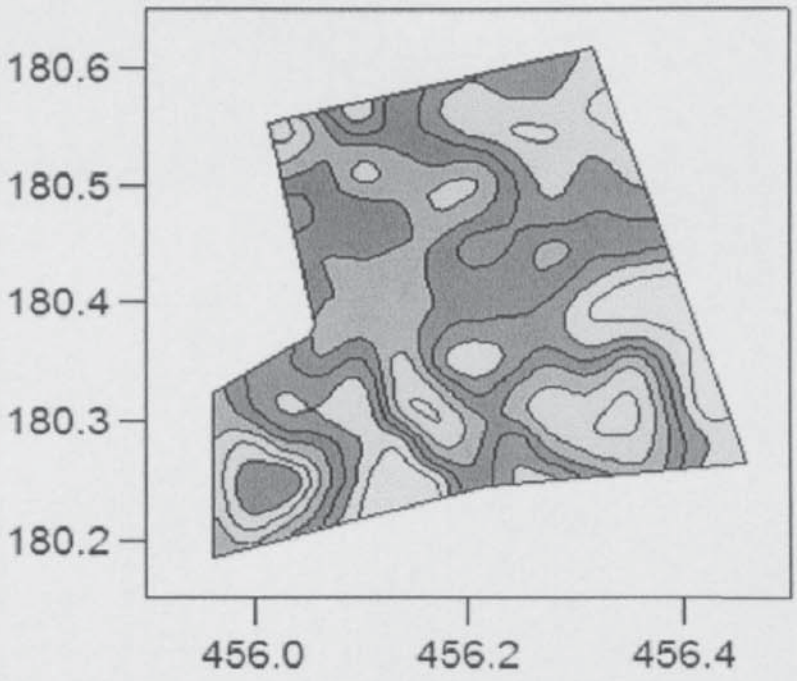


Figure 7.14: Maps for Yattendon using 30 m gridded data and ML parameter estimation

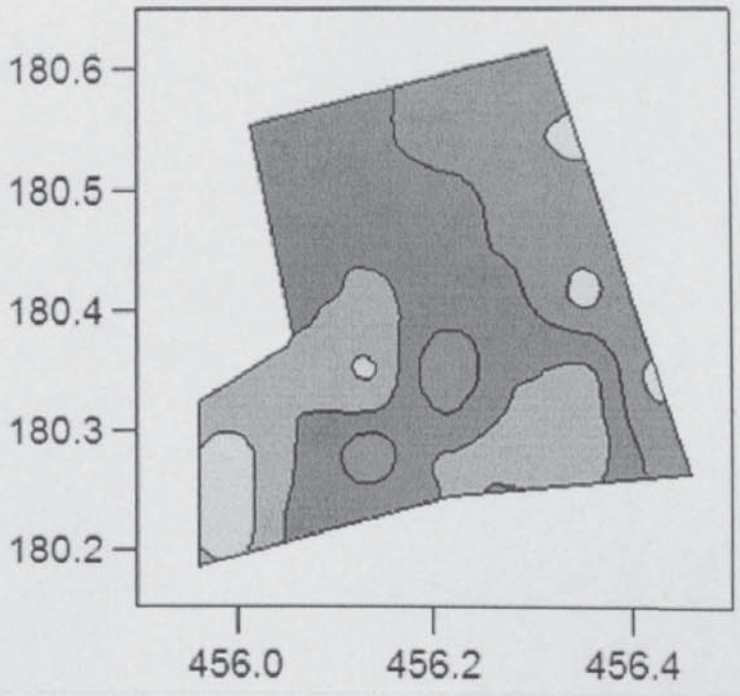


Figure 7.15: Maps for Yattendon using 60 m gridded data and ML parameter estimation

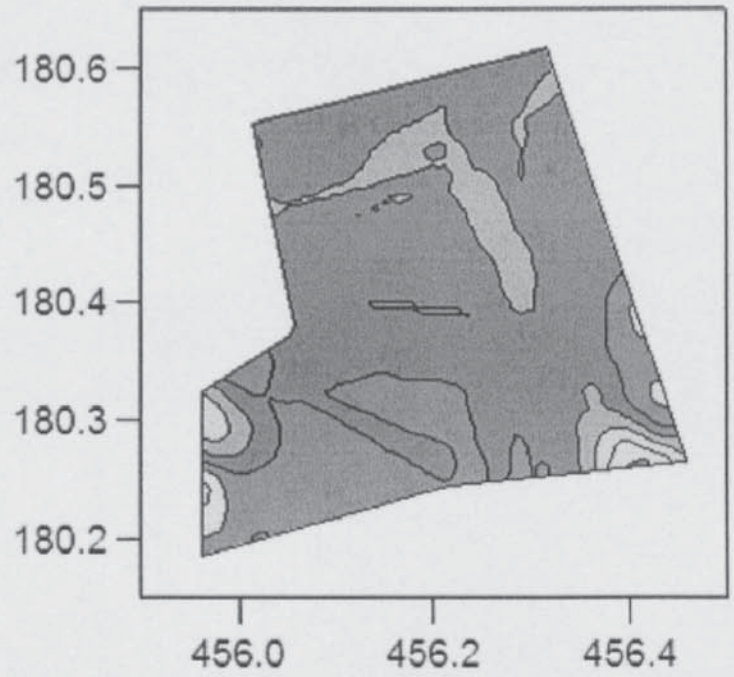


Figure 7.16: Maps for Yattendon using 60 m + 30 m gridded data and ML parameter estimation

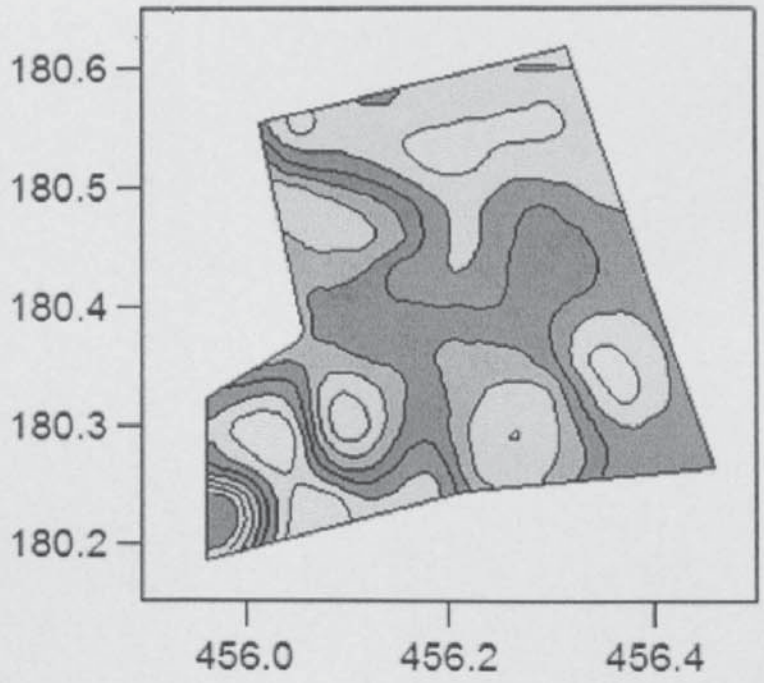


Figure 7.17: Maps for Yattendon using PPK selected data and ML parameter estimation

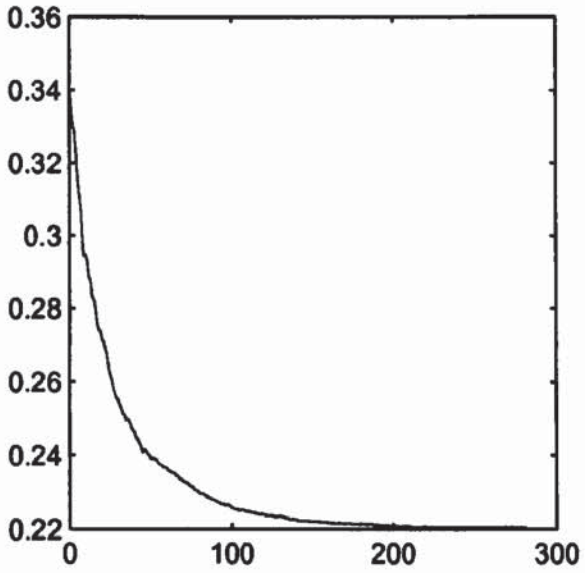


Figure 7.18: Wallingford prediction error as a function of the active set size. The vertical axis is the prediction mean absolute error and the horizontal axis is the number of sampling locations used for prediction.

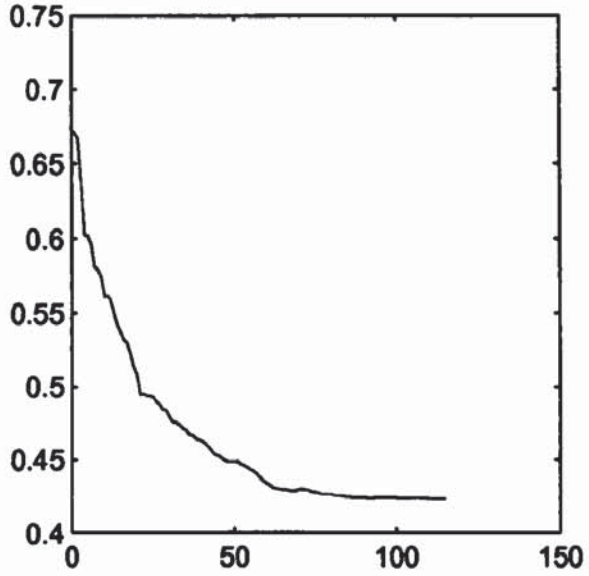


Figure 7.19: Yattendon prediction error as a function of the active set size. The vertical axis is the prediction mean absolute error and the horizontal axis is the number of sampling locations used for prediction.

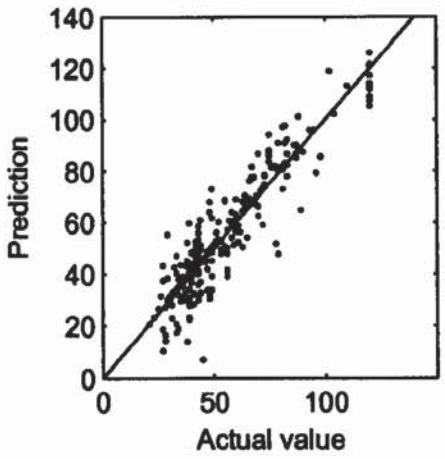


Figure 7.20: Plot showing Wallingford observations against predictions.

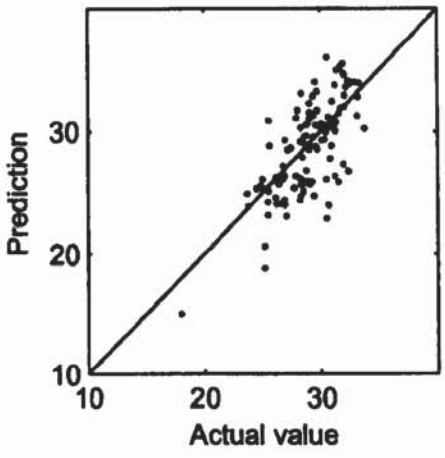


Figure 7.21: Plot showing Yattendon observations against predictions.

8

Conclusions

In this thesis, a variety of techniques for addressing the efficient application of geostatistics have been examined and developed. A wide spectrum of ideas has been presented which addressed the computation complexity issues in different ways.

This thesis focuses on three main areas for increasing the efficiency in which geostatistics are applied and reducing the redundancy in the representation of the model. Chapter 2 reviewed techniques which are appropriate for datasets where the sampling interval is short with respect to the process lengthscale. Effectively, densely sampled datasets can be projected onto a reduced complexity model. In contrast to this, Chapter 4 explores the use of space–limited covariance functions which are more appropriate for datasets where the sampling interval is long with respect to the process lengthscale. Fast and efficient matrix methods can be used by making the assumption that beyond a certain distance threshold, correlations effectively vanish and can be considered redundant information. By applying space–limited covariance functions in the projected process framework, a compact representation can be obtained in both the large–scale

and small–scale regimes. Both types of redundancy are exploited giving an universal model with the best of both worlds for treating large datasets. Performance comparisons showed an increase in prediction accuracy and showed their applicability to large datasets. A further application of space–limited covariance methods and sparse matrix methods showed how computational performance can also be improved.

The third type of redundancy which was discussed in Chapter 6 is the prevailing trend that modern computers effectively have at least two processors. The full potential of this increased processing power is generally under–used in many applications. Chapter 6 specifically addressed parallel algorithms which could be executed on a range of hardware from personal computers up to the very fastest super–computers. In the modern desktop computer paradigm where a computer has a number of processors, existing geostatistical techniques are generally limited by the speed of the fastest processor. Recent years have seen a significant increase in awareness about energy usage (MacKay 2007). This is driving manufacturers to be more conscious about the energy usage of their products. This trend is emerging in the computing world also. For many years, chip manufacturers would tout the performance of their processors by the clock frequency, however, recently there has been a shift away from measuring processor speed solely in gigahertz. One of the main performance measures is now MIPS[1]–per–watt. Two lower frequency processors can consume less power than a higher frequency processor (Low 2005) and at the same time deliver greater performance.

The definition of performance is difficult to define since performance will be related to the application being tested. Modern operating systems are multi–tasking in the sense that multiple applications are in memory at any one time and these can be executed simultaneously on a parallel system so performance will be increased.

Splitting the datasets into subsets was a key activity for the techniques discussed in Chapter 6. The subset selection is important as this impacts prediction performance so clustering algorithms are needed in automatic geostatistical frameworks.

Chapter 7 presented a technique for the optimisation of sampling design given ancillary data. One of the advantages of this method is the speed at which the sample configuration can be determined and the ability to select sampling locations which retain features in the data. Some previous work relies on minimising the global predictive variance to optimise the sampling

[1]Millions of Instructions Per Second.

locations (Brus and Heuvelink 2007), whereas the method discussed in this thesis also includes the predictive mean of the ancillary data in the sample design optimisation process to identify and retain interesting features in the data.

8.1 Future work

Selecting the observations which best represent the process by their inclusion in the active set still raises a number of issues. Whether using the heuristic based scoring methods such as those of Csató and Opper (2001b) or whether optimising the locations as in Snelson and Ghahramani (2006) it is still difficult to obtain a globally optimal set of active points. The methods of Snelson and Ghahramani (2006) can be prone to finding bad local minima in the optimisation process (Seeger et al. 2007). The iterative procedures of Csató and Opper (2001b) require interleaving parameter estimation with active set selection over a number of cycles. Extensions to the existing algorithms for selecting the active set could be designed that would exploit parallel architectures.

The covariance function used to construct the covariance matrix will impact the size of the active set determined if the active set has not been specified in advance. It would be interesting to see how covariance function properties such as lengthscale, noise and smoothness would relate to the active set size when using projected–process algorithms. To date there is no discussion of the covariance function properties and the relationship to active set size, although some issues relating to the covariance function properties are alluded to in a number of publications (Rasmussen and Williams 2006; Snelson 2007).

The parallel algorithms discussed in Chapter 6 required the data to be spatially clustered. Techniques for clustering the data should be investigated. One proposal would be to create a model dependent on a number of sub–models. Sequentially, the observations could be presented to each sub–model and the likelihood of the overall model could be calculated based on the inclusion to the particular sub–model. The inclusion into the sub–model that increases the likelihood by the greater margin is the cluster it should be assigned to. Having included it into a particular cluster, the next observation is considered.

The sample design configuration technique discussed in Chapter 7 could be used as a general (without ancillary data) purpose algorithm for sample location optimisation particularly in

situations where a fast result is needed. Iterative algorithms for sample location optimisation could provide a fast alternative to the simulated annealing techniques frequently used (Lark 2002; Heuvelink et al. 2006; Brus and Heuvelink 2007).

A

Matrix identities and useful algebra

A.1 Sherman–Morrison–Woodbury formula

The Sherman–Morrison–Woodbury formula or Woodbury matrix identity, as it is frequently referred to, states that the inverse of a rank–k matrix can be updated by doing a rank–k correction to the inverse of the original matrix (Golub and Van Loan 1989).

Explicitly, the Sherman–Morrison–Woodbury matrix inversion formula is:

$$\left(\mathbf{A}+\mathbf{X}\mathbf{B}\mathbf{X}^{\mathsf{T}}\right)^{-1}=\mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{X}(\mathbf{B}^{-1}+\mathbf{X}^{\mathsf{T}}\mathbf{A}^{-1}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{A}^{-1} \tag{A.1}$$

where $\mathbf{A}$ and $\mathbf{B}$ are square and invertible matrices and are $\mathbf{n}\times\mathbf{n}$ and $\mathbf{k}\times\mathbf{k}$, respectively. $\mathbf{X}$ is $\mathbf{n}\times\mathbf{k}$.

If one wishes to compute $\left(\mathbf{A}+\mathbf{X}\mathbf{B}\mathbf{X}^{\mathsf{T}}\right)^{-1}$ and if $\mathbf{B}$ is of much lower rank than that of $\mathbf{A}$ and if $\mathbf{A}^{-1}$ is already known then it is only necessary to find $\left(\mathbf{B}^{-1}+\mathbf{X}^{\mathsf{T}}\mathbf{A}^{-1}\mathbf{X}\right)^{-1}$

This allows hard inverses to be converted into easy inverses when $\mathbf{A}$ is large and diagonal

and when $\mathbf{X}$ has many rows but few columns.

Additionally, the calculation of the determinant can be computed efficiently using this identity:

$$\left|\mathbf{A}+\mathbf{X}\mathbf{B}\mathbf{X}^{\mathsf{T}}\right| = |\mathbf{A}|\,|\mathbf{B}|\left|\mathbf{B}^{-1}+\mathbf{X}^{\mathsf{T}}\mathbf{A}^{-1}\mathbf{X}\right| \tag{A.2}$$

A.2 Partitioned matrix inverse identity

The partitioned matrix inverse identity for symmetric matrices expresses the relationship between block sub–matrices of a matrix and the inverse. Given a partitioned matrix:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^{\mathsf{T}} & \mathbf{C} \end{bmatrix} \tag{A.3}$$

the inverse is given by:

$$\mathbf{Z}^{-1} = \begin{bmatrix} \mathbf{D}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{E}^{-1} \\ -\mathbf{E}^{-1}\mathbf{B}^{\mathsf{T}}\mathbf{A}^{-1} & \mathbf{E}^{-1} \end{bmatrix} \tag{A.4}$$

$$= \begin{bmatrix} \mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{B}\mathbf{E}^{-1}\mathbf{B}^{\mathsf{T}}\mathbf{A}^{-1} & -\mathbf{E}^{-1}\mathbf{B}\mathbf{C}^{-1} \\ -\mathbf{C}^{-1}\mathbf{B}^{\mathsf{T}}\mathbf{E}^{-1} & \mathbf{C}^{-1}+\mathbf{C}^{-1}\mathbf{B}^{\mathsf{T}}\mathbf{D}^{-1}\mathbf{B}\mathbf{C}^{-1} \end{bmatrix} \tag{A.5}$$

with $\mathbf{D} = \mathbf{A} - \mathbf{B}\mathbf{C}^{-1}\mathbf{B}^{\mathsf{T}}$ and $\mathbf{E} = \mathbf{C} - \mathbf{B}^{\mathsf{T}}\mathbf{A}^{-1}\mathbf{B}$.

A.3 Cholesky factorisation

Many of the algorithms used throughout this thesis require the inversion of symmetric positive definite matrices. For numerical stability reasons, the Cholesky factorisation should be used since the matrix inverse is rarely required directly (Seeger 2004). The partitioned matrix inversion identity given by equation (A.4) can be used with the Cholesky factorisation. A symmetric positive definite matrix can be factorised:

$$\mathbf{Z}^{-1} = \mathbf{R}^{\mathsf{T}}\mathbf{R} \tag{A.6}$$

where $\mathbf{R}$ is the lower left triangle matrix. The partitioned matrix can then be updated sequentially:

$$\mathbf{R}_{t+1} = \begin{bmatrix} \mathbf{R}_t & 0 \\ -\mathbf{E}^{-1}\mathbf{B}^{\mathsf{T}}\mathbf{A}^{-1} & \mathbf{E}^{-1/2} \end{bmatrix} \tag{A.7}$$

where $\mathbf{E} = \mathbf{C} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}$ and t relates to the current time step or iteration in the algorithm.

The Cholesky decomposition can also be used for calculating the determinant:

$$|\mathbf{D}| = \prod_{i=1}^{n} R_{ii}^2 \tag{A.8}$$

A.4 Block matrix inverse identity

A matrix, $\mathbf{Z}$, can be partitioned as:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \tag{A.9}$$

where $\mathbf{A}$ and $\mathbf{D}$ are square matrices of sizes $a \times a$ and $d \times d$ respectively. The matrices $\mathbf{B}$ and $\mathbf{C}$ may not be square and have the sizes $a \times d$ and $d \times a$.

Assuming the inverse is partitioned as:

$$\mathbf{Z}^{-1} = \begin{bmatrix} \tilde{\mathbf{A}} & \tilde{\mathbf{B}} \\ \tilde{\mathbf{C}} & \tilde{\mathbf{D}} \end{bmatrix} \tag{A.10}$$

then $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, $\tilde{\mathbf{C}}$ and $\tilde{\mathbf{D}}$, which are of the same size as $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ respectively, can be calculated by:

$$\tilde{\mathbf{A}} = \left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} \tag{A.11}$$

$$\tilde{\mathbf{B}} = -\left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1}\left(\mathbf{B}\mathbf{D}^{-1}\right) \tag{A.12}$$

$$\tilde{\mathbf{C}} = -\left(\mathbf{D}^{-1}\mathbf{C}\right)\left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} \tag{A.13}$$

$$\tilde{\mathbf{D}} = \mathbf{D}^{-1} + \left(\mathbf{D}^{-1}\mathbf{C}\right)\left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1}\left(\mathbf{B}\mathbf{D}^{-1}\right) \tag{A.14}$$

A.5 Block diagonal matrix inverse identity

The block matrix identity in Appendix A.4 can be exploited when it has a block diagonal structure. Given a matrix:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{A} & 0 \\ 0 & \mathbf{D} \end{bmatrix} \tag{A.15}$$

then the complexity of inverting this partitioned block diagonal matrix can be reduced to:

$$\mathbf{Z}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} & 0 \\ 0 & \mathbf{D}^{-1} \end{bmatrix} \tag{A.16}$$

. where only the diagonal block elements need to be inverted.

A.6 Product of two Gaussians

The Bayesian Gaussian processes discussed in this thesis consist of likelihoods and priors that are both Gaussian distributions. The posterior distribution is a product of two Gaussians normalised by a constant. The following identity can be used:

$$\mathcal{N}(\mathbf{x}|\mathbf{a},\mathbf{A})\,\mathcal{N}(\mathbf{x}|\mathbf{b},\mathbf{B}) = \mathbf{Z}^{-1}\mathcal{N}(\mathbf{x}|\mathbf{c},\mathbf{C}) \tag{A.17}$$

where:

$$\mathbf{C} = \left(\mathbf{A}^{-1} + \mathbf{B}^{-1}\right)^{-1} \tag{A.18}$$

and:

$$\mathbf{c} = \mathbf{C}\left(\mathbf{A}^{-1}\mathbf{a} + \mathbf{B}^{-1}\mathbf{b}\right). \tag{A.19}$$

The normalising constant is given by:

$$\mathbf{Z}^{-1} = (2\pi)^{-\frac{D}{2}} |\mathbf{A}+\mathbf{B}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{a}-\mathbf{b})(\mathbf{A}+\mathbf{B})^{-1}(\mathbf{a}-\mathbf{b})\right). \tag{A.20}$$

B

Sequential Sparse Gaussian Processes

B.1 A Bayesian derivation

A projected process model based algorithm was first proposed by Csató (2002) and is formulated in a Bayesian framework with the name Sequential, Sparse Gaussian processes (SSGP). There are a number of aspects of this approach that are appealing to the geostatistician. Principled algorithms for large datasets where all the observations are used are scarce. Additionally, the SSGP method facilitates the use of non–Gaussian noise models.

The elegance of this method is that it is a model–based, principled application of the extension to kriging algorithm that has been discussed earlier (Sections 2.5.1 and 2.5.2), employed within a Bayesian Gaussian process framework.

Instead of the conventional kriging algorithm whereby a single large matrix inversion takes place, a sequential estimation scheme will be employed. In this sequential scheme the algorithm considers a single observation of the dataset during each iteration of the algorithm and builds

a sequence of intermediate posterior distributions until a global posterior distribution is arrived at on the last iteration. The speed improvement is achieved by a subsequent modification to the above mentioned sequential inclusions. At the end of each iteration the possibility of removing certain input locations from the representation of the posterior process is considered. The input location selection and removal is done in such a way that it guarantees that the posterior still includes the maximum amount of *information* about the data item. Using this iterative scheme leads to a modified sparse posterior distribution which relies solely on a subset of the training inputs. Hence, the computational complexity scales cubically only with the size of this subset.

The Gaussian process is represented by a parameterisation of the first two moments of the posterior process. The parameterisation is given by:

$$\begin{aligned} \langle f_{\mathbf{x}} \rangle_{\text{post}} &= \langle f_{\mathbf{x}} \rangle_0 \sum_{i=1}^{N} k_0(\mathbf{x}, \mathbf{x}_i) q(i) \\ k_{\text{post}}(\mathbf{x}, \mathbf{x}^T) &= k_0(\mathbf{x}, \mathbf{x}^T) + \sum_{i,j=1}^{N} k_0(\mathbf{x}, \mathbf{x}_i) R(ij) k_0(\mathbf{x}_j, \mathbf{x}^T). \end{aligned} \tag{B.1}$$

where q_t and R_t are update coefficients of the posterior process. The GP prior mean function, $\langle f_{\mathbf{x}} \rangle_0$, is updated by adding a new covariance function scaled by the update coefficient q_t to the representation. The GP prior in the 2nd iteration is the posterior from the previous iteration. Effectively there is a recursive relationship. Likewise for the GP prior covariance function. This is updated with the covariance update coefficient R_t.

The update parameters $q(i)$ and $R(ij)$ are given by:

$$\begin{aligned} q(i) &= \frac{1}{Z} \int df p_0(f) \, \frac{\partial p(\mathcal{D}|f)}{\partial f(\mathbf{x}_i)} \\ R(ij) &= \frac{1}{Z} \int df p_0(f) \, \frac{\partial^2 p(\mathcal{D}|f)}{\partial f(\mathbf{x}_i) \partial f(\mathbf{x}_j)} - q(i) q(j) \end{aligned} \tag{B.2}$$

where $Z = \int df p_0(f) \, p(\mathcal{D}|f)$ is a normalising constant. The parameters $q(i)$ and $R(ij)$ only have to be computed once during the training of the model. One of the problems with this algorithm is in this step since the posterior distribution is not usually a Gaussian distribution and hence the integrals are not analytically tractable (Csató et al. 2000). One common method is to approximate this intractable posterior distribution using a variational framework to giving the *nearest* Gaussian distribution. Nearest in this sense could mean, for this algorithm, the moments of the mean and variance are matched when approximating the Gaussian distribution.

As previously stated the observations are considered sequentially in an iterative fashion. Bayes' rule is used

$$p_{post}(f) = \frac{p(Z(x_t)|f)\hat{p}_{t+1}(f)}{\langle p(x_{t+1}|f_X)\rangle_t} \tag{B.3}$$

where each prior distribution is the posterior distribution from the previous iteration in a recursive fashion. Since the posterior distribution is likely to be no longer Gaussian, it is projected to be the nearest Gaussian distribution in the Kullback–Leibler (Kullback and Leibler 1951) sense which matches the first two moments of the distribution (Saad 1998). This is possible, since the likelihood term is for only one observation and hence is one dimensional and can be solved efficiently. Figure B.1 shows this iterative process of treating each observation individually.

To compute the sequential approximations of the mean and covariance k_{t+1}, Equation (B.1) is applied sequentially with only one likelihood term $p(Z(x_{t+1})|x_{t+1})$ at an iteration step. Proceeding recursively,

$$\begin{aligned} \langle f_x\rangle_{t+1} &= \langle f_x\rangle_t + q^{(t+1)}\, k_t(x, x_{t+1}) \\ k_{t+1}(x, x^T) &= k_t(x, x^T) + r^{(t+1)}\, k_t(x, x_{t+1}) k_t(x_{t+1}, x^T) \end{aligned} \tag{B.4}$$

is arrived at, where the scalars $q^{(t+1)}$ and $r^{(t+1)}$ follow from Equation (B.2):

$$\begin{aligned} q^{(t+1)} &= \frac{\partial}{\partial\langle f_{t+1}\rangle_t} \ln\langle p(Z(x_{t+1})|f_{t+1})\rangle_t \\ r^{(t+1)} &= \frac{\partial^2}{\partial\langle f_{t+1}\rangle_t} \ln\langle p(x_{t+1}|f_{t+1})\rangle_t. \end{aligned} \tag{B.5}$$

and where the averages in (B.5) are with respect to the Gaussian process at time t and the derivatives taken with respect to $\langle f_{t+1}\rangle_t = \langle f(x_{t+1})\rangle_t$. It should be noted that these averages only require a one dimensional integral to be solved.

To arrive at a parameterisation for the Gaussian process it is necessary to unfold the recursive steps in the update given by Equation (B.4) which leads to:

$$\begin{aligned} \langle f_x\rangle_{t+1} &= \sum_{i=1}^{t} k_0(x, x_i)\alpha_t(i) = \alpha_{t+1}^T k_x \\ k_{t+1}(x, x^T) &= k_0(x, x^T) + \sum_{i,j=1}^{t} k_0(x, x^T)\Sigma_{t+1}(ij)k_0(x_j, x^T) = k_0(x, x^T) + k_x^T \Sigma_{t+1} k_{x^T} \end{aligned} \tag{B.6}$$

It should be noted that the coefficients $\alpha_{t+1}(i)$ and $\Sigma_{t+1}(ij)$ are not dependent on x. To

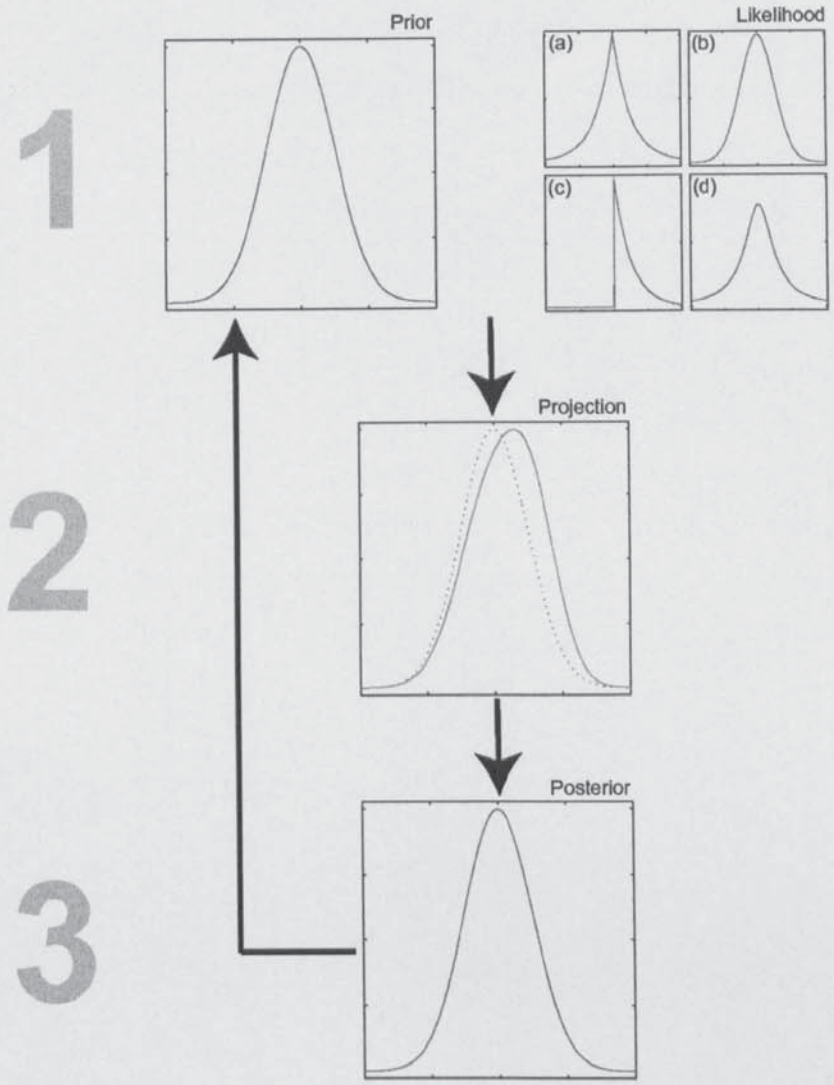


Figure B.1: Step 1 is the standard Bayesian way of combining a prior distribution with a likelihood. Four potential likelihoods are shown: (a) Laplace, (b) Gaussian, (c) One–sided exponential, (d) Student–t. Except in the case of a Gaussian likelihood, the posterior distribution will be non–Gaussian as shown by the solid line in Step 2. The posterior distribution is projected to the *closest* Gaussian distribution (based on some distance metric) shown in Step 3. The prior distribution for the next iteration of Step 1 of the algorithm is given by the Gaussian posterior in Step 3 from the previous step.

increase the size of the representation, the following equation is used:

$$\begin{aligned}\alpha_{t+1} &= T_{t+1}(\alpha_t) + q^{(t+1)}s_{t+1},\\ \Sigma_{t+1} &= U_{t+1}(\Sigma_t) + r^{(t+1)}s_{t+1}s_{t+1}^T,\\ s_{t+1} &= T_{t+1}(\Sigma_t k_{t+1}) + e_{t+1}.\end{aligned} \tag{B.7}$$

The operators T_{t+1} and U_{t+1} have been introduced for increasing by one the dimensionality of vectors or matrices respectively.

The next step is to ask how can sparsity be achieved in this model? The goal is to reduce the number of parameters but with a minimal loss of information. This can be achieved if the following equation holds for all $\mathbf{x}$:

$$k_0(\mathbf{x}, \mathbf{x}_{t+1}) = \sum_{i=1}^{t} \hat{e}_{t+1}(i) k_0(\mathbf{x}, \mathbf{x}_i). \tag{B.8}$$

Although it is improbable that this equation will hold exactly, it is still possible that there will only be a small error induced. The only required change to the model would be the change of s_{t+1} by:

$$\hat{s}_{t+1} = \Sigma_t k_{t+1} + \hat{e}_{t+1}. \tag{B.9}$$

$\hat{e}_{t+1}$ is used in the approximate update of the model parameters where:

$$\hat{e}_{t+1} = \Sigma_{t+1}^{-1} k_{t+1}. \tag{B.10}$$

Now that sparsity is defined, it is important to know the influence that each active point in the active set has on the posterior process. The model uses a simple heuristic to determine how important each active point is:

$$e_i = \frac{\alpha_i}{\mathrm{diag}\left(\Sigma_i^{-1}\right)}. \tag{B.11}$$

Given the index of the least informative active point in the set, the point can be deleted. It is also necessary to maintain the inverse covariance between the active point locations, this is represented by $\mathbf{Q}$. There are three equations which are used to update the parameters given a particular deletion; the updated parameters are given by $\hat{\alpha}$, $\hat{\Sigma}$ and $\hat{\mathbf{Q}}$. The columns to be deleted from the covariance matrix and the inverse covariance matrix of the active set are represented by $\mathbf{Q}^*$ and Σ^{-1*} and the diagonal element of the column to be deleted from the two matrices is

given by q^*, and σ^*:

$$\begin{aligned}\hat{\boldsymbol{\alpha}} &= \boldsymbol{\alpha}^{(t-1)} - \alpha^* \frac{\mathbf{Q}^*}{q^*}, \\ \hat{\boldsymbol{\Sigma}} &= \boldsymbol{\Sigma}^{(t-1)} + \sigma^* \frac{\mathbf{Q}^* \mathbf{Q}^{*\mathsf{T}}}{q^{*2}} - \frac{1}{q^*}\left[\mathbf{Q}^* \boldsymbol{\Sigma}^{*\mathsf{T}} + \boldsymbol{\Sigma}^* \mathbf{Q}^{*\mathsf{T}}\right], \\ \hat{\mathbf{Q}} &= \mathbf{Q}^{(t-1)} - \frac{\mathbf{Q}^* \mathbf{Q}^{*\mathsf{T}}}{q*}.\end{aligned} \tag{B.12}$$

The real strength of this method is that arbitrary likelihoods which may be zero in certain regions, can be treated by this method. Such likelihoods may cause problems for other Gaussian approximations based on the averaging of the log-likelihood (variational Gaussian approximation). This method merely requires the explicit computation of a Gaussian smoothed likelihood and is thus well suited for cases where (local) likelihood functions can be modelled empirically as mixtures of Gaussian distributions. If such expressions are available, the necessary one-dimensional integrals can be done analytically and the online updates require just matrix multiplications and function evaluations.

The sequential, sparse Gaussian process framework is equivalent to the sequential projected process kriging algorithm discussed in Section 2.5. The ordering of the observations can make a slight difference to predictions. This version of the algorithm is equivalent to the DTC approximation. By viewing this SSGP method in a kriging framework, it is hoped that the advantages of this method become more accessible. Batch equivalents to these methods have also been discussed, but the issue of selecting the active set is still the major obstacle to their widespread usage.

An extension to the SSGP algorithm was discussed in Csató (2002) which uses the Expectation Propagation (EP) framework to ensure that the ordering of the data is not a problematic issue. The EP extension has been shown to be equivalent to the FITC approximation (Snelson 2007). The data is processed numerous times with differing orderings during each iteration. The method relies on the theory underpinning cavity–fields from statistical physics. The EP method is a generalisation of cavity–fields to arbitrary probability distributions (Minka 2001).

B.1.1 Non–Gaussian likelihoods

The sequential methods have been discussed in the previous sections and made mention of non–Gaussian likelihoods. The assumption that the likelihood (or noise model) is Gaussian

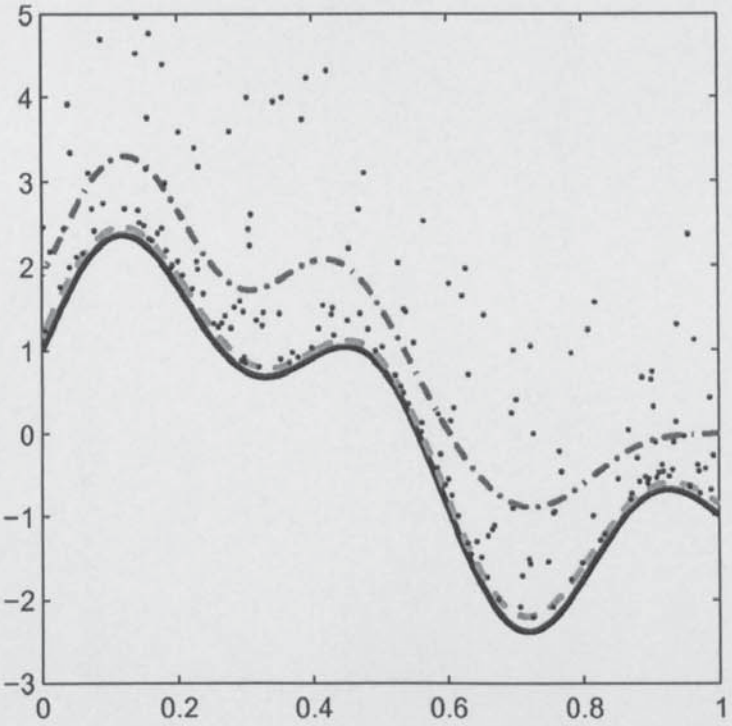


Figure B.2: Data generated from a sinusoidal function with additive non–Gaussian noise (positive exponential noise). The noise distribution has a one–sided exponential distribution. The dot–dashed line represents the prediction using a Gaussian noise model assumption. The dashed line shows the prediction using the correct noise distribution used to generate the data, positive exponential noise.

is common, although it is unlikely that any dataset will be exactly Gaussianly distributed. Although not a significant issue for many datasets, should the geostatistician wish to deal with this, the Box–Cox transformation is often used (Box and Cox 1964). The Box–Cox transform is a family of transformations and a computational technique to select a transformation that will best resolve the problems of non-normality.

The sequential methods previously discussed in this chapter can be updated to provide a solution to the problem of non-Gaussianity. The basis of this approach is to approximate non-Gaussian integrals. This variational approach facilitates the use of other likelihood models.

C

Covariance identities and notation

C.1 Combining covariance functions

Shawe-Taylor and Cristianini (2004) extensively discuss methods for creating new covariance models. In what follows, it is shown how a valid covariance function can be created in a number of forms from one or more valid covariance functions. Spatial locations are denoted by $\mathbf{a}$ and $\mathbf{b}$. $\mathbf{c}$ is a constant that satisfies $\mathbf{c} > 0$ and $\mathbf{f}(\cdot)$ is any function.

$$\mathbf{k}(\mathbf{a}, \mathbf{b}) = \mathbf{c}\mathbf{k}_1(\mathbf{a}, \mathbf{b}) \tag{C.1}$$

$$\mathbf{k}(\mathbf{a}, \mathbf{b}) = \mathbf{f}(\mathbf{a})\,\mathbf{k}_1(\mathbf{a}, \mathbf{b})\,\mathbf{f}(\mathbf{b}) \tag{C.2}$$

$$\mathbf{k}(\mathbf{a}, \mathbf{b}) = \mathbf{f}(\mathbf{k}_1(\mathbf{a}, \mathbf{b})) \tag{C.3}$$

$$\mathbf{k}(\mathbf{a}, \mathbf{b}) = \mathbf{k}_1(\mathbf{a}, \mathbf{b}) + \mathbf{k}_2(\mathbf{a}, \mathbf{b}) \tag{C.4}$$

$$\mathbf{k}(\mathbf{a}, \mathbf{b}) = \mathbf{k}_1(\mathbf{a}, \mathbf{b})\,\mathbf{k}_2(\mathbf{a}, \mathbf{b}) \tag{C.5}$$

(C.6)

C.2 Schur Product

The Schur or Hadamard product refers to the element–wise multiplication of a matrix. The notation $\mathbf{A} \circ \mathbf{B}$ is used where $\mathbf{A}$ and $\mathbf{B}$ are matrices of equal size and $\circ$ denotes the Schur product. The Schur product of two positive definite matrices yields a positive definite matrix (Horn and Johnson 1994).

C.3 Frobenius norm

The Frobenius norm between two square matrices $\mathbf{A}$ and $\mathbf{B}$ is given by:

$$\langle \mathbf{A}, \mathbf{B} \rangle_F = \text{trace}\left(\mathbf{A}\mathbf{B}^T\right) = \sum_{i,j=1}^{n} A_{ij} B_{ij} \tag{C.7}$$

and is also know as the Euclidean norm.

Bibliography

Abrahamsen, P. (1997). *A review of Gaussian random fields and correlation functions.* Norsk Regnesentral/Norwegian Computing Center.
(pages 76 and 77)

Adler, R. (1981). *The Geometry of Random Fields.* Chichester, U.K.; New York, U.S.A.: Wiley.
(page 77)

Amdahl, G. (1967). Validity of the single processor approach to achieving large scale computing capabilities. *AFIPS Conference Proceedings 30*(8), 483–485.
(page 106)

Amestoy, P. R., T. A. Davis, and I. S. Duff (1996). An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications 17*(4), 886–905.
(page 94)

Anderson, E., D. Sorensen, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammerling, J. Demmel, and C. Bischof (1990). LAPACK: A portable linear algebra library for high-performance computers. *Proceedings of the 1990 ACM/IEEE conference on Supercomputing*, 2–11.
(page 72)

Andrews, P. and R. Bell (1998). Optimizing the United Kingdom meteorological office data assimilation for ERS-1 scatterometer wind. *Monthly Weather Review 26*, 736–746.
(page 16)

Auñón, J. and J. Gómez-Hernández (2000). Dual kriging with local neighborhoods: Application to the representation of surfaces. *Mathematical Geology 32*(1), 69–85.
(page 44)

Avnimelech, R. and N. Intrator (1999). Boosting regression estimators. *Neural Computation 11*(2), 499–520.

(page 122)

Backus, J. (1978, August). Can programming be liberated from the von Neumann style? a functional style and its algebra of programs. *Commun. ACM 21*(8), 613–641.
(page 19)

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford: Clarendon Press.
(page 17)

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* New York, NY, USA: Springer.
(pages 121 and 124)

Blackford, L. S., J. Choi, A. Cleary, A. Petitet, R. C. Whaley, J. Demmel, I. Dhillon, K. Stanley, J. Dongarra, S. Hammarling, G. Henry, and D. Walker (1996). Scalapack: A portable linear algebra library for distributed memory computers - design issues and performance. In *Supercomputing '96: Proceedings of the 1996 ACM/IEEE conference on Supercomputing (CDROM)*, Washington, DC, USA, pp. 5. IEEE Computer Society.
(page 113)

Border, S. (1993). The use of indicator kriging as a biased estimator to discriminate between ore and waste. In *Applications of Computers in the Mineral Industry*, University of Wollongong, New South Wales.
(page 115)

Box, G. E. P. and D. R. Cox (1964). An analysis of transformations. *Journal of the Royal Statistical Society 26*(2), 211–252.
(pages 59 and 172)

Brus, D. J. and G. B. Heuvelink (2007, February). Optimization of sample patterns for universal kriging of environmental variables. *Geoderma 138*(1–2), 86–95.
(pages 144, 159, and 160)

Choudhury, A., P. B. Nair, and A. J. Keane (2002). A data parallel approach for large-scale Gaussian process modeling. In *Proceedings of the Second SIAM International Conference on Data Mining, Arlington, VA, April.*
(page 123)

Cobb, S. (2000). *The Dictionary of Computer Science, Engineering, and Technology.* CRC Press Inc.
(page 19)

Collobert, R., S. Bengio, and Y. Bengio (2002). A parallel mixture of SVMs for very large scale problems. In *International Conference on Processors, Architectures and Compilation Techniques*, Cambridge, MA. MIT Press.

(page 124)

Cornford, D. (1996). *The development and application of techniques for mapping daily minimum air temperatures.* Ph. D. thesis, School of Geography, `http://www.ncrg.aston.ac.uk/~cornfosd/phd.php`.

(page 76)

Cornford, D. (2005). Are comparative studies a waste of time? SIC2004 examined. In *Automatic Mapping Algorithms for Routine and Emergency Monitoring Data.*

(page 68)

Cornford, D., L. Csato, and M. Opper (2005). Sequential, Bayesian geostatistics: A principled method for large data sets. *Geographical Analysis 37*(2), 183–199.

(page 29)

Cressie, N. and D. Hawkins (1980). Robust estimation of the variogram: I. *Mathematical Geology 12*(2), 115–125.

(page 59)

Cressie, N., A. Olsen, and D. Cook (1997). Massive data sets: Problems and possibilities, with application to environmental monitoring. In *Massive Data Sets: Proceedings of a Workshop*, pp. 115–119. The National Academies Press.

(page 15)

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

(pages 25, 27, 28, 43, and 116)

Cressie, N. A. (2006). Spatial prediction for massive datasets. In *Mastering the Data Explosion in the Earth and Environmental Sciences: Proceedings of the Australian Academy of Science Elizabeth and Frederick White Conference.*

(pages 64 and 65)

Cristianini, N., J. Shawe-Taylor, B. Technologies, A. Elisseeff, and J. Kandola (2002). On kernel-target alignment. *International Conference on Processors, Architectures and Compilation Techniques 14.*

(page 84)

Crouchley, R., T. van Ark, J. Pritchard, J. Kewley, R. Allan, M. Hayes, and L. Morris (2005). Putting social science applications on the grid. *First International Conference on e-Social Science, June.*

(page 112)

Csató, L. (2002). *Gaussian Processes – Iterative Sparse Approximation.* Ph. D. thesis, Neural Computing Research Group, `http://www.ncrg.aston.ac.uk/Papers`.
(pages 38, 62, 165, and 170)

Csató, L., E. Fokoue, M. Opper, B. Schottky, and O. Winther (2000). Efficient approaches to Gaussian process classification. *International Conference on Processors, Architectures and Compilation Techniques 12*, 251–257.
(page 166)

Csató, L. and M. Opper (2001a). Greedy sparse approximation to Gaussian processes by relative entropy projection. Technical report, Neural Computing Research Group.
(page 60)

Csató, L. and M. Opper (2001b). Sparse representation for Gaussian process models. In *International Conference on Processors, Architectures and Compilation Techniques*, Volume 13, Cambridge, MA, pp. 444–450. MIT Press.
(pages 31, 45, 60, and 159)

Csató, L. and M. Opper (2002). Sparse online Gaussian processes. *Neural Computation 14*(3), 641–669.
(pages 47, 61, and 147)

Cuthill, E. and J. McKee (1969). Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, New York, NY, USA, pp. 157–172. ACM Press.
(page 94)

Darlington, J., A. J. Field, P. G. Harrison, P. H. J. Kelly, D. W. N. Sharp, Q. Wu, and R. L. While (1993). Parallel programming using skeleton functions. In A. Bode, M. Reeve, and G. Wolf (Eds.), *PARLE '93: Parallel Architectures and Languages Europe*, pp. 146–160. Berlin, DE: Springer-Verlag.
(page 19)

David, M. (1976). The practice of kriging. *Advanced Geostatistics in the Mining Industry 31*, 461.
(page 43)

Davis, M. W. and P. G. Culbane (1984). Contouring very large data sets using kriging. *Geostatistics for Natural Resources Characterization 2*, 599–619.

(page 43)

Davis, M. W. and C. Grivet (1984). Kriging in a global neighborhood. *Mathematical Geology 16*, 249–265.

(page 38)

De Gruijter, J., D. Brus, M. Bierkens, and M. Knotters (2005, February). *Sampling for Natural Resource Monitoring: Statistics and Methodology of Sampling and Data Analysis.* Springer-Verlag Berlin and Heidelberg GmbH and Co. K.

(page 42)

Deligiannakis, A., Y. Kotidis, and N. Roussopoulos (2004). Compressing historical information in sensor networks. *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 527–538.

(page 101)

Dietrich, M. W. and G. N. Newsam (1989). A stability analysis of the geostatistical approach to aquifer transmissivity identification. *Stochastic Environmental Research and Risk Assessment 3*, 293–316.

(page 38)

Diggle, P. J. and P. J. Ribeiro Jr. (2007). *Model-based Geostatistics.* New York, NY, U.S.A.: Springer Science.

(pages 16 and 31)

Diggle, P. J., J. A. Tawn, and R. A. Moyeed (1998). Model-based geostatistics. *Applied Statistics 47*, 299–350.

(pages 25, 31, and 76)

Dubois, G. and S. Galmarini (2005). Spatial interpolation comparison (SIC) 2004: Introduction to the exercise and overview of results. In *Automatic Mapping Algorithms for Routine and Emergency Monitoring Data.*

(page 101)

Duff, I. S. and H. A. van der Vorst (1999). Developments and trends in the parallel solution of linear systems. *Parallel Computing 25*(13–14), 1931–1970.

(page 116)

El-Shaarawi, A. and W. Piegorsch (2002). *Encyclopedia of Environmetrics.* Wiley.

(page 75)

Fayyad, U. and R. Uthurusamy (2002). Evolving data into mining solutions for insights. *Commun. ACM 45*(8), 28–31.

(page 121)

Fernández, J., M. Anguita, S. Mota, A. Cañas, E. Ortigosa, and F. J. Rojas (2004). MPI toolbox for Octave. In *Proc. VECPAR*, Valencia, Spain.

(page 120)

Fine, S. and K. Scheinberf (2002). Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research 2*, 243–264.

(page 40)

Foster, I. (1995). *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

(page 19)

Frogbrook, Z. L. (1999). The effect of sampling intensity on the reliability of predictions and maps of soil properties. In *Proceedings of the 2nd European Conference on Precision Agriculture.*

(page 41)

Frogbrook, Z. L. and M. A. Oliver (2000). The effects of sampling on the accuracy of predictions of soils properties for precision agriculture. In *Proceedings of the 4th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences.*

(pages 41 and 142)

Frogbrook, Z. L., M. A. Oliver, M. Salahi, and R. Ellis (2002). Exploring the spatial relations between cereal yield and soil chemical properties and the implications for sampling. *Soil Use and Management 18*(1), 1–9.

(page 41)

Furrer, R., M. G. Genton, and D. Nychka (2006). Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics 15*(3), 502–523.

(pages 78, 79, and 82)

Gajraj, A., W. Joubert, and J. Jones (1997). A parallel implementation of kriging with a trend. Technical report, U.S. Department of Energy, Office of Scientific and Technical Information.

(page 112)

Galmarini, S. (2005). Real-time geostatistics for atmospheric dispersion forecasting and vice verse. In *Automatic Mapping Algorithms for Routine and Emergency Monitoring Data.*

(page 17)

Gaspari, G. and S. E. Cohn (1999). Construction of correlation functions in two and three dimensions. *Quarterly Journal of the Royal meteorological Society 125*, 723–757.
(page 78)

Gebhardt, A. (2003). PVM kriging with R. In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing, Vienna.*
(pages 112 and 114)

Genton, M. G. (2001). Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research 2*, 299–312.
(page 78)

Gibbs, M. N. (1997). *Bayesian Gaussian processes for Regression and Classification.* Ph. D. thesis, Cambridge University. `http://www.inference.phy.cam.ac.uk/mng10/`.
(page 38)

Gibbs, S. (2003). Data parallelism and the support vector machine. *Online Proceedings of the Information Processing Systems (IPS) Laboratory Research Forum, Department of Electrical Engineering, Ohio State University, Colombus, Ohio, USA.*
(page 124)

Gilbert, J. R., C. Moler, and R. Schreiber (1992). Sparse matrices in MATLAB: Design and implementation. *SIAM Journal on Matrix Analysis and Applications 13*(1), 333–356.
(pages 92, 94, and 116)

Gneiting, T. (1999). Correlation functions for atmospheric data analysis. *Quarterly Journal of the Royal meteorological Society 125*, 2449–2464.
(page 81)

Gneiting, T. (2002). Compactly supported correlation functions. *Journal of Multivariate Analysis 83*, 493–508.
(page 82)

Golub, G. H. and C. F. Van Loan (1989). *Matrix Computations* (Second ed.). Baltimore: Johns Hopkins University Press.
(pages 40 and 161)

Goovaerts, P. et al. (1997). *Geostatistics for Natural Resources Evaluation.* Oxford University Press New York.
(page 77)

Gramacy, R. B. and H. K. H. Lee (2006). Bayesian treed Gaussian process models. Technical report, Dept. of Applied Math & Statistics, University of California, Santa Cruz.
(page 123)

Gray, A. (2004). Fast kernel matrix-vector multiplication with application to Gaussian process learning. Technical report, School of Computer Science, Carnegie Mellon University.
(page 38)

Gropp, W., E. Lusk, and T. Sterling (2002). *Beowulf Cluster Computing in Linux*. Cambridge, MA: MIT Press.
(page 102)

Gupta, A., G. Karypis, and V. Kumar (1997). Highly scalable parallel algorithms for sparse matrix factorization. *IEEE Transactions on Parallel and Distributed Systems 8*(5), 502–520.
(page 116)

Haas, T. C. (1990). Kriging and automated variogram modeling within a moving window. *Atmospheric Environment 24A*(7), 1759–1769.
(page 43)

Haas, T. C. (1995). Local prediction of a spatio-temporal process with an application to wet sulfate deposition. *Journal of the American Statistical Association 90*(432), 1189–199.
(pages 43 and 134)

Hamers, B., J. A. K. Suykens, and B. De Moor (2002). Compactly supported rbf kernels for sparsifying the gram matrix in LS-SVM regression models. In *Proceedings of the International Conference on Artificial Neural Networks, ICANN 2002*, Madrid Spain.
(page 75)

Hestenes, M. (1980). *Conjugate direction methods in optimization*. Springer.
(page 83)

Heuvelink, G. B. M., D. J. Brus, and J. J. de Gruijter (2006). Optimization of sample configurations for digital soil mapping with universal kriging. *Digital Soil Mapping: An Introductory Perspective 31*, 137–151.
(pages 31, 42, 144, 145, 146, and 160)

Horn, R. and C. Johnson (1994). *Topics in Matrix Analysis*. Cambridge University Press.
(page 174)

Hosten, O., M. T. Rakher, J. T. Barreiro, N. A. Peters, and P. G. Kwiat (2005). Counterfactual quantum computation through quantum interrogation. *Nature 439*(7079), 891–1030.

(page 102)

Houtekamer, P. and H. Mitchell (2001). A sequential ensemble Kalman filter for atmospheric data assimilation. *Monthly Weather Review 129*(1), 123–137.

(page 79)

Huang, D. (2005). *Experimental planning and sequential kriging optimization using variable fidelity data.* Ph. D. thesis, Ohio State University.

(page 38)

Ingram, B., D. Cornford, and D. Evans (2008). Fast algorithms for automatic mapping with space–limited covariance functions. *Stochastic Environmental Research and Risk Assessment 22*(5), 661–670.

(page 17)

Ingram, B., L. Csató, and D. Evans (2005). Fast spatial interpolation using sparse Gaussian processes. *Applied GIS 1*(2), 15:1–17.

(pages 36, 37, 75, and 77)

Irony, D., G. Shklarski, and S. Toledo (2002). Parallel and fully recursive multifrontal supernodal sparse cholesky. In *International Conference on Computational Science (2)*, pp. 335–344.

(page 116)

Isaaks, E. H. and R. M. Srivastava (1989). *An Introduction to Applied Geostatistics.* Oxford University Press.

(pages 28, 44, 70, 71, 74, and 77)

Jacobs, R. A., M. I. Jordan, S. J. Nowlan, and G. E. Hinton (1991). Adaptive mixture of local experts. *Neural Computation 3*, 79–87.

(page 123)

Journel, A. (1974). Geostatistics for conditional simulation of ore bodies. *Economic Geology 69*(5), 673–687.

(page 68)

Journel, A. G. and C. J. Huijbregts (1978). *Mining Geostatistics.* Academic Press.

(page 77)

Kaufman, C. (2006). *Covariance Tapering for Likelihood Based Estimation in Large Spatial Datasets.* Ph. D. thesis, PhD thesis, Carnegie Mellon University.

(page 83)

Kepner, J. (2001, Sep). Parallel programming with MatlabMPI. In *Proceedings of the High Performance Embedded Computing (HPEC 2001) workshop*, MIT Lincoln Laboratory, Lexington, MA.

(page 119)

Kerry, K. and K. Hawick (1998). Kriging interpolation on high-performance computers. *Proceedings of the International Conference and Exhibition on High-Performance Computing and Networking*, 429–438.

(pages 112 and 115)

Kerry, K. E. and K. A. Hawick (1997). Spatial interpolation on distributed, high-performance computers. Technical Report DHPC-015, University of Adelaide, Australia.

(page 115)

Kerry, R. (2004). *Determining the effect of parent material and topography on the spatial structure of variation in soil properties for precision agriculture*. Phd, University of Reading.

(page 141)

Kerry, R. and M. A. Oliver (2003). Variograms of ancillary data to aid sampling for soil surveys. *Precision Agriculture 4*(3), 261–278.

(page 20)

Kerry, R. and M. A. Oliver (2007). Comparing sampling needs for variograms of soil properties computed by the method of moments and residual maximum likelihood. *Geoderma 140*(4), 383–397.

(pages 41, 143, and 147)

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics 34*(5), 975–986.

(page 83)

Kitanidis, P. (1985). Minimum-variance unbiased quadratic estimation of covariances of regionalized variables. *Mathematical Geology 17*(2), 195–208.

(page 36)

Kullback, S. and R. Leibler (1951). On information and sufficiency. *The Annals of Mathematical Statistics 22*(1), 79–86.

(pages 60 and 167)

Lark, R. (2002). Optimized spatial sampling of soil for estimation of the variogram by maximum likelihood. *Geoderma 105*, 49–80(32).

(page 160)

Lark, R. M. (2000). Estimating variograms of soil properties by the method-of-moments and maximum likelihood. *European Journal of Soil Science 51*(4), 717–728.
(pages 37, 112, 143, and 147)

Latouche, P. (2006). Distributed machine learning. Master's thesis, Neural Computing Research Group. Aston University.
(page 122)

Lawrence, N., M. Seeger, and R. Herbrich (2003). Fast sparse Gaussian process methods: The informative vector machine. *International Conference on Processors, Architectures and Compilation Techniques 15*, 609–616.
(pages 39 and 45)

Lawrence, N. D. and R. Herbrich (2001). A sparse Bayesian compression scheme — the Informative Vector Machine. In *International Conference on Processors, Architectures and Compilation Techniques.*
(pages 38, 42, 44, and 60)

Lawson, C., R. Hanson, D. Kincaid, and F. Krogh (1979). Basic linear algebra subprograms for Fortran usage. *ACM Transactions on Mathematical Software (TOMS) 5*(3), 308–323.
(page 72)

Lazarevic, A. and Z. Obradovic (2002). Boosting algorithms for parallel and distributed learning. *Distributed Parallel Databases 11*(2), 203–229.
(page 122)

Liu, S. and T. Yeh (2004). An integrative approach for monitoring water movement in the vadose zone. *Vadose Zone Journal 3*(2), 681–692.
(page 38)

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory 28*, 129–137.
(page 123)

Loo, K. K., I. Tong, B. Kao, and D. Cheung (2005). Online algorithms for mining inter-stream associations from large sensor networks. *Proceedings of the Ninth Pacific-Asia Conference of Advances in Knowledge Discovery and Data Mining, Hanoi, Vietnam: Springer 149.*
(page 101)

Low, R. (2005, September). Microprocessor trends: Multicore, memory and power developments. *Embedded Computing Design 1*(9), 17–19.

(page 158)

Lu, K. and S. Goddard (2004). Grass-based high performance spatial interpolation component for spatial decision support systems. In *Proceedings of the FOSS/GRASS Users Conference.*

(page 113)

MacKay, D. (1998). Introduction to Gaussian processes. *Neural Networks and Machine Learning 168*, 133–165.

(page 39)

MacKay, D. J. C. (2007). Sustainable energy – without the hot air. `http://www.withouthotair.com/`.

(page 158)

Malard, J. M. (2002). Parallel restricted maximum likelihood estimation for linear models with a dense exogenous matrix. *Parallel Computing 28*(2), 343–353.

(page 117)

Manne, F. and H. Hafsteinsson (1995). Efficient sparse Cholesky factorization on a massively parallel SIMD computer. *SIAM Journal on Scientific Computing 16*(4), 934–950.

(page 116)

Mardia, K. (1980). Some statistical inference problems in kriging. II: Theory. *Proceedings of the 26th International Geological Congress, Paris. Sciences de la Terre, Serie Informatique Geologique*, 113–131.

(pages 36 and 37)

Mardia, K. and A. Watkins (1989). On multimodality of the likelihood in the spatial linear model. *Biometrika 76*(2), 289–295.

(page 36)

Mardia, K. V. and R. J. Marshall (1984). Maximum likelihood estimation of models for residual covariance in spatial regression. *Biometrika 71*(1), 135–146.

(page 36)

Matheron, G. (1963). Principles of geostatistics. *Economic Geology 58*(8), 1246–1266.

(page 25)

McBratney, A. B. and R. Webster (1981). The design of optimal sampling schemes for local estimation and mapping of regionalized variables II. program and examples. *Computers and Geosciences 7*(4), 335–365.

(page 142)

Menzefricke, U. (1995). On the performance of the Gibbs sampler for the multivariate Normal distribution. *Communications in Statistics - Theory and Methods 24*, 191–213.
(page 40)

Meyer, T. (2004). The discontinuous nature of kriging interpolation for digital terrain modeling. *Cartography and Geographic Information Science 31*(4), 209–217.
(page 44)

Minasny, B. and A. B. McBratney (2005, October). The matérn function as a general model for soil variograms. *Geoderma 128*, 192–207.
(page 37)

Minka, T. (2001). Expectation propagation for approximate Bayesian inference. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, San Francisco, CA, pp. 362–36. Morgan Kaufmann.
(page 170)

Misztal, I. (1990). Restricted maximum likelihood estimation of variance components in animal model using sparse matrix inversion and a supercomputer. *J. Dairy Sci. 73*(1), 163–172.
(page 117)

Neumaier, A. and E. Groeneveld (1998). Restricted maximum likelihood estimation of covariances in sparse linear models. *Genetics Selection Evolution 30*(1), 3–26.
(page 116)

Neumann, J. (1945). First draft of a report on the EDVAC. *University of Pennsylvania*.
(page 19)

Offiler, D. (1987). Wind measurements from the earth remote-sensing satellite (ERS-1). *Meteorological Magazine 116*, 279–285.
(page 16)

Omre, H. (1987). Bayesian kriging - merging observations and qualified guesses in kriging. *Mathematical Geology 19*(1), 25–39.
(page 31)

Ormoneit, D. and V. Tresp (1998). Averaging, maximum penalized likelihood and Bayesian estimation for improving Gaussian mixture probability density estimates. *IEEE Transactions on Neural Networks 49*(4), 639–650.

(page 122)

Palmer, T. and R. Hagedorn (2006). *Predictability of Weather and Climate.* Cambridge University Press.

(page 79)

Pardo-Igúzquiza, E. (1998). Maximum likelihood estimation of spatial covariance parameters. *Mathematical Geology 30*(1), 95–108.

(page 36)

Pardo-Igúzquiza, E. and P. Dowd (1997). AMLE3D: A computer program for the inference of spatial covariance parameters by approximate maximum likelihood estimation. *Computers and Geosciences 23*(7), 793–805(13).

(page 135)

Pedelty, J. A., J. L. Schnase, and J. A. Smith (2003). High performance geostatistical modeling of biospheric resources in the Cerro Grande Wildfire Site, Los Alamos, New Mexico and Rocky Mountain National Park, Colorado. NASA Goddard Space Flight Center, Code 930.

(pages 112 and 113)

Pissanetzky, S. (1984). *Sparse Matrix Technology.* London: Academic Press.

(page 91)

Poulet, F. (2003, September). Multi-way distributed SVM algorithms. In *Parallel and Distributed computing for Machine Learning. In conjunction with the 14th European Conference on Machine Learning (ECML'03) and 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'03)*, Cavtat-Dubrovnik, Croatia.

(page 122)

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling (1992). *Numerical Recipes: The Art of Scientific Computing* (2nd ed.). Cambridge (UK) and New York: Cambridge University Press.

(page 94)

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (1996). *Numerical Recipes in Fortran 90 : The Art of Parallel Scientific Computing* (2nd ed.). New York, NY, USA: Cambridge University Press.

(pages 39, 104, and 117)

Quinn, M. J. (2003, September). *Parallel Programming in C with MPI and OpenMP.* McGraw-Hill Education (ISE Editions).

(pages 103 and 109)

Quiñonero-Candela, J. and C. E. Rasmussen (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research 6*, 1935–1959.
(pages 18, 45, 47, 49, 52, 125, and 128)

Ramaswamy, S., I. Eugene W. Hodges, and P. Banerjee (1996). Compiling MATLAB programs to ScaLAPACK: Exploiting task and data parallelism. In *IPPS '96: Proceedings of the 10th International Parallel Processing Symposium*, Washington, DC, USA, pp. 613–619. IEEE Computer Society.
(page 115)

Rasmussen, C. E. and C. K. I. Williams (2006, 01). *Gaussian processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press.
(pages 18, 25, 30, 125, and 159)

Ripley, B. D. (1981). *Spatial Statistics*. New York, N.Y.: John Wiley Sons.
(page 25)

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge: Cambridge University Press.
(page 17)

Saad, D. (1998). *On-Line Learning in Neural Networks*. Cambridge Univ. Press.
(page 167)

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems, 2nd edition*. Philadelpha, PA: SIAM.
(page 91)

Sakata, S., F. Ashida, and M. Zako (2004). An efficient algorithm for kriging approximation and optimization with large-scale sampling data. *Computer Methods in Applied Mechanics and Engineering 193*, 385–404.
(pages 38 and 39)

Schabenberger, O. and C. Gotway (2005). *Statistical Methods For Spatial Data Analysis*. CRC Press.
(page 16)

Schlather, M. (2006). Simulation and analysis of random fields. Technical report, Georg–August–Univ. Göttingen.
(page 82)

Schwaighofer, A. and V. Tresp (2003). Transductive and inductive methods for approximate Gaussian process regression. *International Conference on Processors, Architectures and Compilation Techniques 15*, 953–960.
(pages 123 and 125)

Seeger, M. (2003, July). *Bayesian Gaussian process models: PAC-Bayesian generalisation error bounds and sparse approximations*. Ph. D. thesis, University of Edinburgh.
(pages 40, 49, and 147)

Seeger, M. (2004). Gaussian processes for machine learning. *International Journal of Neural Systems 14*(2), 69–106.
(pages 75 and 162)

Seeger, M., N. Lawrence, and R. Herbrich (2007). Efficient nonparametric Bayesian modelling with sparse Gaussian process approximations. Technical report, Max Planck Institute for Biological Cybernetics.
(pages 57 and 159)

Seeger, M., N. D. Lawrence, and R. Herbrich (2006). Efficient nonparametric Bayesian modelling with sparse Gaussian process approximations.
(page 63)

Seeger, M. and C. Williams (2003). Fast forward selection to speed up sparse Gaussian process regression.
(pages 43, 60, and 146)

Shawe-Taylor, J. and N. Cristianini (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.
(page 173)

Shen, Y., A. Y. Ng, and M. Seeger (2006). Fast Gaussian process regression using KD-trees. *International Conference on Processors, Architectures and Compilation Techniques 15*, 1225–1232.
(page 123)

Snelson, E. and Z. Ghahramani (2006). Sparse Gaussian processes using pseudo-inputs.
(pages 45, 51, 63, and 159)

Snelson, E. L. (2007). *Flexible and efficient Gaussian process models for machine learning*. Ph. D. thesis, Gatsby Computational Neuroscience Unit, `http://www.gatsby.ucl.ac.uk/~snelson/pub.html`.
(pages 46, 57, 159, and 170)

Stein, M. (1986). A modification of minimum norm quadratic estimation of a generalized covariance function for use with large data sets. *Mathematical Geology 18*(7), 625–633.
(page 134)

Stein, M. (2002). The screening effect in kriging. *The Annals of Statistics 30*(1), 298–323.
(page 79)

Stein, M., Z. Chi, and L. Welty (2004). Approximating likelihoods for large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 66*(2), 275–296.

(page 136)

Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging.* N.Y.: Springer-Verlag.
(pages 19, 28, 37, 44, 75, 76, 77, 112, and 134)

Tijms, H. (2004). *Understanding Probability: Chance Rules in Everyday Life.* Cambridge: Cambridge University Press.
(page 31)

Tresp, V. (2000a). A Bayesian committee machine. *Neural Computation 12*(11), 2719–2741.
(pages 44, 52, and 125)

Tresp, V. (2000b). The generalized Bayesian Committee Machine. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD-2000 130*, 130–139.
(pages 55 and 125)

Tresp, V. (2000c). Mixtures of Gaussian processes. *International Conference on Processors, Architectures and Compilation Techniques 13*, 654–660.
(page 124)

Tresp, V. (2001). *Committee machines, in Handbook for Neural Network Signal Processing*, Chapter 5, pp. 1–18. CRC Press.
(page 126)

Vargas-Guzmán, J. and T. Yeh (2002). The successive linear estimator: A revisit. *Advances in Water Resources 25*(7), 773–781.
(page 38)

Vargas-Guzmán, J. A. and T.-C. Yeh (1999). Sequential kriging and cokriging: Two powerful geostatistical approaches. *Stochastic Environmental Research and Risk Assessment 13*, 416–435.

(pages 38 and 39)

Vecchia, A. V. (1988). Estimation and model identification for continuous spatial processes. *Journal of the Royal Statistical Society. Series B. Methodological 50*(2), 297–312.
(pages 135 and 136)

Wackernagel, H. (2003). *Multivariate Geostatistics: An Introduction with Applications.* Springer.
(pages 74, 75, 77, and 79)

Warnes, J. and B. Ripley (1987). Problems with likelihood estimation of covariance functions of spatial Gaussian processes. *Biometrika 74*(3), 640–642.
(page 36)

Webster, R. and M. A. Oliver (1992). Sample adequately to estimate variograms of soil properties. *European Journal of Soil Science 43*(1), 177–192.
(pages 42 and 142)

Webster, R. and M. A. Oliver (2000). *Geostatistics for Environmental Scientists.* Wiley.
(pages 28, 36, and 42)

Wendland, H. (1995). Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics 4*, 389–396.
(pages 78, 80, and 81)

Wendland, H. (1998). Error estimates for interpolation by compactly supported radial basis functions of minimal degree. *Journal of Approximation Theory 93*(2), 258–272.
(page 80)

Whaley, R. C., A. Petitet, and J. J. Dongarra (2001). Automated empirical optimization of software and the ATLAS project. *Parallel Computing 27*(1–2), 3–35. Also available as University of Tennessee LAPACK Working Note #147, UT-CS-00-448, 2000 (`www.netlib.org/lapack/lawns/lawn147.ps`).
(page 72)

Wilkinson, D. (2005). Parallel Bayesian computation. *Handbook of Parallel Computing and Statistics*, 481–512.
(page 112)

Williams, C. and C. Rasmussen (1996). Gaussian processes for regression. *International Conference on Processors, Architectures and Compilation Techniques 8*, 514–520.
(pages 25 and 30)

Williams, C. K. I. and M. Seeger (2001). Using the Nyström method to speed up kernel machines. *International Conference on Processors, Architectures and Compilation Techniques 13*, 682–688.

(page 64)

Williams, M., D. Cornford, B. Ingram, L. Bastin, T. Beaumont, E. Pebesma, and G. Dubois (2007, May). Supporting interoperable interpolation: The INTAMAP approach. In *International Symposium on Environmental Software Systems*, Prague.

(page 17)

Yang, C., R. Duraiswami, and L. Davis (2005). Efficient kernel machines using the improved fast Gauss transform. *International Conference on Processors, Architectures and Compilation Techniques 17*, 1561–1568.

(page 38)

Zeleznik, F. J. (1968). Quasi-Newton methods for nonlinear equations. *J. ACM 15*(2), 265–271.

(page 83)

Zhang, H. H., G. M. Genton, and P. Liu (2004). Compactly supported radial basis function kernels. In *Institute of Statistics Mimeo Series 2570*. North Carolina State University.

(pages 84 and 85)

Zidek, J., W. Sun, and N. Le (2000). Designing and integrating composite networks for monitoring multivariate Gaussian pollution fields. *Applied Statistics 49*, 63–79.

(page 144)