Gaussian Process Emulators for Uncertainty Analysis in Groundwater Flow

Nicola Stone.

Thesis submitted to The University of Nottingham
for the degree of Doctor of Philosophy

April 2011

Abstract

In the field of underground radioactive waste disposal, complex computer models are used to describe the flow of groundwater through rocks. An important property in this context is transmissivity, the ability of the groundwater to pass through rocks, and the transmissivity field can be represented by a stochastic model. The stochastic model is included in complex computer models which determine the travel time for radionuclides released at one point to reach another. As well as the uncertainty due to the stochastic model, there may also be uncertainties in the inputs of these models. In order to quantify the uncertainties, Monte Carlo analyses are often used; the computer code will be run many times and a sample of outputs will be obtained, from which sample statistics can be calculated. However, for computationally expensive models, it is not always possible to obtain a large enough sample to provide accurate enough uncertainty analyses. In this thesis, we present the use of Bayesian emulation methodology as an alternative to Monte Carlo in the analysis of stochastic models. The idea behind Bayesian emulation methodology is that information can be obtained from a small number of runs of the model using an small sample from the input distribution. This information can then be used to make inferences about the output of the model given any other input.

The current Bayesian emulation methodology is extended to emulate two statistics of a stochastic computer model; the mean and the distribution function of the output. The mean is a simple output statistic to emulate and provides some information about how the output changes due to changes in each input. The distribution function is more complex to emulate, however it is an important statistic since it contains information about the entire distribution of the outputs. Distribution functions of radionuclide travel times have been used as part of risk analyses for underground radioactive waste disposal. The extended methodology is presented using a case study of a site currently

used for underground disposal of radioactive waste. In this example, three models for the mean of the transmissivity field are investigated, so that any uncertainty due to the choice of model can be observed. Available measured transmissivity data is used to provide distributions for the uncertain inputs of each model for the transmissivity. The computer code is then run using samples from these distributions to provide a sample of output data. Emulators are built using this information, and then used to approximate the mean and distribution function of the output for each of the three models. The emulators provided estimates comparable to the Monte Carlo estimates, but in a shorter time. The complexity of the emulation increased as the number of input parameters increased, but the output of the computer model was not changed very much by using different models for the mean transmissivity field.

Acknowledgements

I thank my supervisors; Andrew, Ian and Jeremy for supporting and encouraging me through my research. I am grateful for the help and knowledge they have given me to enable me to complete this thesis. My family and friends, in particular Eddie, also deserve thanks for keeping me smiling through the last 4 years.

I also acknowledge the EPSRC for funding this research under grant number EP/E018084/1 as part of the project: "Coupled Models: Expert Judgement, Emulators and Model Uncertainty". Being involved in this project has helped me to develop an understanding of how my research relates to that from other disciplines and I have enjoyed the many discussions that we have had. I would like to thank all involved in this project for providing me with new insights into my research through sharing their own work.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Earth's crust is made up of rocks. These rocks are porous allowing water and air to fill them. Close to the surface, the rocks are unsaturated and contain mostly air. Further down, the rocks are saturated; the pores are completely filled with water. In this saturated zone, the pores of the rocks are usually connected and so water can move through the rocks. This is groundwater flow, and plays an important role in the movement of water through our environment.

Groundwater is a valuable source of fresh water, providing drinking water and irrigation for crops. Groundwater is therefore an important component of a good water supply and needs to be managed carefully. It is vital that groundwater is kept free from pollution. An important example of this regards the safe disposal of radioactive waste. One option is to bury the waste underground. Groundwater is the most likely route for radionuclides from this waste to reach our environment. Mathematical models of groundwater flow are therefore built to carry out risk assessments for the burial of radioactive waste.

Another important use of mathematical models of groundwater flow is to ensure that the extraction of groundwater does not affect the environment, and that lakes, ponds and wetlands are preserved. In order to manage groundwater effectively, groundwater flow models are needed so that experiments can be carried out to explore the flow in certain regions.

One of the major problems in modelling groundwater flow is a lack of knowledge about the properties of the rock and of the flow of groundwater through the rock. Two important quantities in this context are; the head which is the ability of the groundwater

to rise above a datum level, and the transmissivity which describes how fast the water is travelling through the rock. Boreholes can be drilled to measure various quantities at a small number of locations. Whilst this information offers us a glimpse of what is happening underground, we can never know for sure the exact physical structure of the rock. Therefore, we build models which represent our uncertainty about the physical quantities of the rock, whilst honouring the measured data.

In this thesis, we are concerned with analysing stochastic groundwater flow models related to the disposal of nuclear waste underground. We develop models describing the transmissivity field of the rock. The transmissivity field across an entire region of rock is uncertain, as it is only known at a small number of locations. We represent this uncertainty using stochastic models with uncertain hyperparameters. These hyperparameters are used as inputs to the groundwater flow computer models. The uncertainty in the hyperparameters arises as they relate to physical properties of the rock that the groundwater is flowing through. We want to know how the uncertainty in the inputs affects the uncertainty in the output of the model.

In order to carry out this analysis, we will use Gaussian process emulation to provide us with an approximation to our computer models. The emulation allows us to determine the output of a computer model at any input, given that we know the output at a small sample of design inputs. These emulators have been used in many applications (Higdon et al. (2004); Bayarri et al. (2007); Kennedy and O'Hagan (2001); Rojnik and Naveršnik (2008)) and have been shown to provide good estimates to the model output for deterministic computer models. These GP emulators are derived using Bayesian inference. In the Bayesian interpretation of probability, probabilities are assigned according to the belief that a given hypothesis is true. This is in contrast to the frequentist approach, where probabilities are assigned to random events according to their relative frequencies of occurance. Bayesian inference is based on updating prior beliefs using new information or data to produce a refined set of beliefs. Therefore, under the Bayesian framework, the basic idea is to represent the computer model output as a function of the inputs. Then, using a small number of runs of the computer model, an emulator is built to approximate this function. As well as approximating the output of the computer model, emulators have built in bounds on their estimate. These bounds enable us to quantify how much confidence we have in the approximation to the computer model.

The emulation methodology will be discussed further in the next chapter.

The groundwater flow models we will investigate are stochastic, and so we will extend the basic emulation methodology to take this into account. For stochastic models, we will no longer be able to estimate the output of the computer model since there is more than one possible output for each input configuration. However, we can estimate statistics of the output, such as the mean and the distribution function. Once the emulators have been built, we will then be able to use them to analyse the statistics of interest faster than traditional methods such as Monte Carlo.

In the next section we will introduce the case study of the Waste Isolation Pilot Plant (WIPP) that we will investigate in this thesis. The Bayesian emulation methodology will be applied to this case study and the uncertainty in the output of the WIPP computer model will be analysed.

1.1 Waste Isolation Pilot Plant

The WIPP is a US Department of Energy (D.O.E) repository for disposing of radioactive waste (U.S. D.O.E. (2010)). In the past it has been used for research and development into the safe underground disposal of radioactive waste (LaVenue et al. (1990)). Since 1999, the WIPP repository has been fully operational. Due to the research carried out by the US government, there is a large amount of data available for the WIPP site in comparison with other regions. This makes it ideal as a case study, and many models have been built to determine the flow of groundwater in the region (Gotway (1994); Kröhn and Schelkes (1996); Corbet (2000)). Much of this research has considered how a radioactive particle would move through the groundwater in surrounding rock if it were to escape from the repository.

Groundwater moves through different types of rocks at different rates; for a given head gradient, the more transmissive the rock, the faster the groundwater will move through it. WIPP is situated in the Delaware Basin in New Mexico. The WIPP repository lies approximately 650m below the surface in the lower part of the Salado formation. This formation contains mainly salts, indicating an absence of moving water (which could move radionuclides to the surface). If there was water in this formation, the salts would be dissolved and washed away. Above the Salado formation lies a layer of Culebra

Dolomite, the most transmissive rock in the region.

Computer models have been built to represent different processes in the region (such as determining the flow of groundwater) in order to carry out performance assessments to meet the compliance requirements of the US Environmental Protection Agency (U.S. D.O.E. (2004)). The main part of these requirements were the following (U.S. E.P.A. (2010)):

- isolation of radionuclides sufficient to meet the containment requirements of the disposal system,
- protection of individuals from radiation exposures for a period of 10,000 years,
- protection of groundwater from radioactive contamination for 10,000 years.

As part of the risk assessment the probabilities of a set of scenarios occurring are calculated using uncertainty analysis of the computer models. One such scenario is the possibility of radionuclides escaping the repository and entering the accessible environment. This would only occur if humans were to drill into the repository when mining. In this scenario, the radionuclides would escape the repository in the centre of the region, flow upwards through the Salado formation, and reach the most transmissive Culebra Dolomite lying above it. The groundwater would then transport the radionuclides through the rocks where it may affect the surrounding environment. The probability of the accessible environment being contaminated by radionuclides within 10,000 years is very important for the WIPP site to comply with the EPA regulations. It is through modelling the region that this probability can be estimated.

We do not have access to the computer models that have been used previously, and so we will develop our own model for this thesis. The model we build will not be as complex as those developed by the US D.O.E, as we have less expert knowledge about the geology of the site. We will, however, develop a stochastic transmissivity model which honours the transmissivity data collected through many studies of the region (summarised in: Cauffman et al. (1990)). In contrast to the D.O.E model, we will treat the hyperparameters of our transmissivity model as uncertain rather than fixed. By doing so, we will incorporate more uncertainty into our model, but we believe that this important source of uncertainty can not be ignored when analysing the uncertainty we have about groundwater flow at the WIPP site.

The transmissivity field is one part of the model. The other part is to use this field in a partial differential equation to determine a head field and then use this to calculate the travel time of a particle travelling from the centre of the region to get to the site boundary. Our model differs again from the US DOE model in that they use an inverse model. This inverse modelling uses an iterative procedure to incorporate the head data as well as the transmissivity data. If a generated transmissivity field does not provide a head field that is true to the head data, the transmissivity field hyperparameters are altered and a new transmissivity field is generated and a new head field calculated from the groundwater equations. This calibration process is repeated until the head field generated represents the available head data in that the difference between the calculated head field and the measured head data at each measurement point is below a given small value. A brief overview of some of the methods used for carrying out this procedure is given in Section 3.5. Whilst incorporating the head data would reduce the uncertainty in the output, the extra complexity that it involves is beyond the scope of this thesis. Here we are most interested in developing a model, which includes the major sources of uncertainty, that we can analyse the behaviour of using Gaussian process emulation.

Building our own model of the flow through the Culebra Dolomite will involve developing a stochastic model for the transmissivity field. We will not investigate model uncertainty per se, although we will investigate several models to find out if the model itself is adding to the uncertainty in the problem. The hyperparameters of these models will be used in a computer model which determines the travel time of a particle released in the centre of the region to reach the boundary of the WIPP site. This site boundary is a square region extending approximately 2 miles in each direction from the centre of the site, and lies in the centre of the modelling domain. Choosing only one output to study will simplify the emulation of the computer model. Since we have a stochastic model for the transmissivity field, we will obtain a different output every time the computer code is run with the same inputs. Therefore, we will investigate the mean and distribution function of the computer model output.

Emulating the mean of the output will provide information about which of the hyperparameters have the largest effect on the output of the computer model. When we emulate the distribution function of the output, we integrate out the uncertainty in the hyperparameters. This will help us to understand if any of our transmissivity field models have

a larger effect on the output of the computer model than the others. The distribution function is very useful to emulate as it contains all of the information about the output of the computer model. In previous studies of modelling the groundwater flow through the Culebra dolomite (Gotway (1994); U.S. D.O.E. (2004)), the distribution function of the output has been used to provide information about the distribution travel times for the purposes of performing a risk analysis of the WIPP site. Therefore, it is important to estimate this quantity using the Bayesian emulation methodology.

1.2 Outline of the thesis

We review methods for analysing uncertainty in computer models in Chapter 2. Both uncertainty analysis and sensitivity analysis are discussed. We start by introducing traditional Monte Carlo methods for carrying out analysis of computer models, and then move on to the emulation approach. Gaussian Process emulation is described, along with some simple examples of the method. The approach is then extended to emulating stochastic models. Simple examples of emulating the mean of a stochastic function are provided.

The groundwater flow equations are described in Chapter 3. These equations link the head to the transmissivity using Darcy's law. If the transmissivity is known, then the head can be determined. However, the transmissivity cannot possibly be known everywhere in the region, only at a few measured points. Therefore, we represent our uncertainty in this quantity using a Gaussian random field model for the transmissivity. In this chapter we review some of the methods used for generating Gaussian random transmissivity fields, and for conditioning these fields on available data.

In Chapter 4 we will analyse the data for our models of the transmissivity through the Culebra dolomite at the Waste Isolation Pilot Plant (WIPP) site. The stochastic model for the transmissivity field will be described using three different forms for the mean and one covariance function. These will allow us to investigate how much the choice of stochastic model affects the output of the computer model. As well as the uncertainty about the transmissivity field, which we have represented with a stochastic model, we will also be uncertain about the hyperparameters of this model. We therefore carry out Bayesian inference to obtain distributions for the hyperparameters for each of our chosen

models. These Bayesian distributions reflect the beliefs, as well as the uncertainty, we have about the values of the hyperparameters. These hyperparameters will then be used as inputs to our computer model of the WIPP region.

After deriving distributions for our inputs, we then develop a mixed finite element model for solving the groundwater flow equations in Chapter 5. The discretisation of the equations is described. We then discuss how we will generate the transmissivity field, concentrating on two methods discussed in Chapter 2. We investigate the errors in the computer model when using different methods to generate the transmissivity field. We also investigate the error in estimating the mean and distribution function from a sample of outputs from the computer model. The two errors we consider here are the standard error of the mean and the standard error of proportion. These errors are frequentist notions of uncertainty estimating the standard deviation of the sampling distributions of the sample mean and sample proportion respectively. They are defined as:

Standard error of the sample mean

Consider a random sample $X = (X_1, X_2, \ldots, X_n)$ of size n taken from a population with mean, μ and variance σ^2, then the sample mean $\bar{X}$ has mean μ and variance σ^2/n.

The standard error of the sample mean, $\hat{\sigma}/\sqrt{n}$, is an estimate of the standard deviation of the sampling distribution of the mean, where $\hat{\sigma}$ is the estimator of σ.

Standard error of the sample proportion

Consider a random sample $X = (X_1, X_2, \ldots, X_n)$ of size n taken from a population in which the proportion of successes is p, then $X \sim \text{Bin}(n, p)$. For large n, $X \sim \text{N}(np, pq/n)$, where $q = (1 - p)$. Now if we let P_s be the proportion of successes in the sample, then $P_s = X/n$ and $P_s \sim \text{N}(p, pq/n)$.

The standard error of the sample proportion, $\sqrt{\hat{p}\hat{q}/n}$, is the estimate of the standard deviation of the sampling distribution of proportions, where $\hat{p}$ and $\hat{q}$ are estimates for p and q respectively.

We use these standard errors in Chapter 5 when discussing how many runs of the model we will need to calculate statistics of the output, such as the mean and percentiles of the distribution function, as we want to keep the standard error of the mean and the standard error of the proportion small.

In Chapter 6 we use Gaussian Process emulation to approximate the mean and the distribution function of the computer model output. The emulation of the mean is used to determine which of the hyperparameters has the greatest effect on the mean of the computer model output using the results for each of the three stochastic models for the transmissivity. We also introduce a method to approximate the distribution function of the output. This method is illustrated using a reduced model, where all but one of the hyperparameters are fixed. Then the distribution function of the output is approximated for all three stochastic models of the transmissivity field.

Finally, in Chapter 7 we draw conclusions on the use of Gaussian Process emulation in the analysis of groundwater flow models and further areas of research are identified.

Chapter 2

Analysis of computer models

Computer models are used to predict the outcome of physical processes when it is too expensive or impractical to carry out a physical experiment. The process is represented in terms of a mathematical model, which is then implemented in a computer code. The output of the computer code is then a prediction of the outcome of the process, and numerical experiments can be carried out by varying different parameters of the code.

For simplicity, the computer models we are considering for the majority of this chapter are deterministic, with an output t from an input θ. We will extend these ideas to stochastic computer models at the end of this chapter, as we wish to analyse stochastic groundwater flow models later in the thesis. The deterministic computer model input will typically be a vector $\boldsymbol{\theta}$, and the output will be a scalar or vector deterministic function of the inputs, $t = \eta(\boldsymbol{\theta})$. Here, we will consider the case when t is a scalar. One run of the model involves choosing one value of $\boldsymbol{\theta}$ and running the code at this input value to get the corresponding output. Often computer codes may be highly complex, and may take many days or weeks to run.

There may be many uncertainties when using a computer model to determine the outcome of physical processes. Kennedy and O'Hagan (2001) give a detailed list of these uncertainties. An overview of their list is given below.

1. There may be uncertainty about the values of the inputs of the computer code. These inputs can be thought of as unknown parameters of the model, and the uncertainty about them is therefore called parameter uncertainty.

2. As the mathematical model may be a simplification of the process, the model will inadequately predict the value of the true process, even if the inputs are known. This is known as structural uncertainty, and is the difference between the true process and the code output at the best values of the input.

3. The model predicts the process under conditions specified by the inputs. However, the process itself may not give the same value under repeated conditions. This residual variability is due to conditions that are not recognised in the mathematical model on which the computer model is based, i.e. structural uncertainty.

4. If any observations of the process are used to calibrate the code, these may include errors. The observation errors add to the uncertainty in the model.

5. The output of the code can also be uncertain. Even though it is a mathematical function of the inputs, it may not be practical to know the output of the code for any set of inputs if the code is complex and takes a long time to run. However if it is only required to know the output for a small number of inputs code uncertainty would not be a problem.

The following four methods have been used to address some of these uncertainties. Uncertainty analysis is used to predict the output of the code and try to quantify the uncertainties in the output due to uncertainties in the inputs of the computer model. Sensitivity analysis examines how the code output varies in response to changes in inputs, particularly finding out which inputs have the most impact on the output. Calibration relates to changing the parameters of the model, represented by a computer code, so that the code output fits the observed data, in the sense that the difference between the observed outcome and the model output is small. Validation assesses how well the code predicts reality, to an acceptable level of accuracy.

In this thesis we are interested in analysing the output of groundwater flow models due to the uncertainties in the inputs. Therefore, we will concentrate on uncertainty and sensitivity analysis in this chapter. Firstly, we will discuss how simple computer codes are analysed using Monte Carlo methods. Then we will investigate how the analysis of more complex codes is carried out using an emulator to approximate the computer model. We will give an overview of some emulation methods that have been used in the analysis of computer models, before giving a more detailed explanation of Gaussian

Process emulation. Finally, we will present some illustrative examples of emulating simple functions, of 1 and 4 inputs. The 1 dimensional input example will then be extended to emulating a simple stochastic function.

2.1 Analysis of simple computer models - Monte Carlo approach

For simple computer models that can be run for a large number of different input configurations with little computational effort, a Monte Carlo approach can be used to carry out uncertainty and sensitivity analysis. These methods started in the late 1940s and early 1950s as a way of carrying out computer experiments in the nuclear industry (Metropolis and Ulam (1949); Donsker and Kac (1950)). Since then, they have been widely used to solve problems in areas such as radiation transport (Spanier and Gelbard (2008)), financial modelling (Glasserman (2003)) and statistical physics (Landau (1999)).

2.1.1 Monte Carlo uncertainty analysis

When carrying out uncertainty analysis, we suppose that one or more of the inputs are uncertain; we may not know which values to use when running the model. This may be due to an inability to measure the variable accurately. We would then assign a probability distribution to the input. The input of the model is then a random vector $\boldsymbol{\Theta}$. The output $T = \eta(\boldsymbol{\Theta})$ is also a random variable. Given $G(\theta)$, the probability distribution function of $\boldsymbol{\Theta}$, we want to find mean and variance of T. We can the calculate the mean of T using

$$\mu_T = \int \eta(\boldsymbol{\theta}) dG(\theta), \tag{2.1.1}$$

and the variance of T is given by

$$\sigma_T^2 = \int (\eta(\boldsymbol{\theta}) - \mu_T)^2 dG(\theta). \tag{2.1.2}$$

When we are able to sample Θ as in the case of the distribution of the output of a computer model, Monte Carlo methods can be used to carry out the integration in (2.1.1) and (2.1.2).

The basic idea of Monte Carlo uncertainty analysis is as follows: A large sample of inputs, $\theta_1, \ldots, \theta_n$, are iid samples from the distribution of $\mathbf{\Theta}$. The output of the computer model is evaluated at each of these inputs to get a random sample of outputs, $t_1 = \eta(\theta_1), \ldots, t_n = \eta(\theta_n)$. Then from the Central Limit Theorem, the sample average $\hat{\mu}_n$ can be used as an estimator for (2.1.1) and the sample variance $\hat{\sigma}_n^2$ is an unbiased estimator for the variance of T, (2.1.2). The estimators for the mean and variance of T are given by

$$\mu_T \approx \hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} t_i,$$

and

$$\sigma_T^2 \approx \hat{\sigma}_n^2 = \frac{1}{(n-1)}\sum_{i=1}^{n}(t_i - \hat{\mu}_n)^2.$$

Also by the Central Limit Theorem, a 95% confidence interval for μ_t can be calculated using

$$\hat{\mu}_n \pm 1.96\frac{\hat{\sigma}_n}{\sqrt{n}}.$$

The Monte Carlo approach is one of the most popular methods, as it is simple to apply, the method is sequential, in that more evaluations can be added without having to restart the analysis, and it is relatively easy to calculate statistics and errors on these statistics. However, it does take considerable computational time when using computationally expensive computer models. For these computer models it is also not very practical since a new set of runs of the model would be required if the distribution of the uncertain inputs were to change (Cox (1977)).

2.1.2 Latin Hypercube sampling

McKay (1992) describes Latin Hypercube sampling (LHS) as an improvement on simple random sampling of the inputs for carrying out Monte Carlo analysis. The idea is an extension of a Latin Square, where a square grid is sampled once from each row and each column. Latin Hypercube sampling is a generalisation of this concept over a larger number of dimensions, where each sample is taken from a single hyperplane. In this way, we get a sample of input variables that represents the whole of the sample space, where simple random sampling may miss sections. This improved coverage of the sample space may lead to reductions in variability in the estimates of the expectation as the largest distance between two sample points may be smaller than when using simple random

sampling. The size of a Latin Hypercube sample may be smaller than that of a simple random sample, whilst still ensuring that the entire input region is taken into account.

LHS is carried out by dividing the sample space into areas of equal probability. For multivariate LHS of size n this would involve dividing the range of each input into n intervals of equal probability. This ensures that each interval is represented in the sample. Each input is randomly sampled once from each of its possible intervals and given a number from 1 to n determined by the order in which these intervals lie. These numbers are then randomly permutated for each input to give the numbers a new order. Each input is then matched with other inputs in the same (new) order.

As a simple bivariate example, consider two inputs both having a uniform distribution ranging between 0 and 1. If we wanted a sample of size 5, we would split each input range into five intervals. Samples from each of the five intervals would be taken, and numbered 1 to 5. The same would be done for the second variable. Then the order of the numbers would be randomly permutated. If the numbers for the first input were $\{3, 5, 2, 1, 4\}$, and the second $\{1, 3, 4, 2, 5\}$, then the samples would be $\{3, 1\}, \{5, 3\}, \{2, 4\}, \{1, 2\}$ and $\{4, 5\}$.

This gives a sample that is spread over the entire range of each input, which may give a more accurate probability distribution of the output by reducing the variability in the estimates. More than one LH design may be constructed, and the design with the largest distance between the closest points chosen as the sample of inputs to use for running the model. This would be fairly computationally expensive, but would ensure that the design points were spread over the region as much as possible. McKay et al. (1979) compares latin hypercube sampling with simple random sampling and finds that LHS provides better estimates of the output statistics than simple random sampling.

2.1.3 Monte Carlo sensitivity analysis

Monte Carlo sensitivity analysis is one step on from Monte Carlo uncertainty analysis. Rather than concentrating on the statistics of the output, sensitivity analysis uses the output data to determine which of the inputs has the greatest effect on the output. For sensitivity analysis (SA), we are interested in the output $T = \eta(\boldsymbol{\Theta})$ where $\boldsymbol{\Theta}$ is uncertain. After carrying out SA on a computer model, we have more information

about how the model output responds to changes in the inputs. Once we know which inputs have the greatest impact on the computer model output, we can investigate ways of reducing our uncertainty on these inputs, which will in turn reduce the uncertainty in our model. On the other hand, inputs which have little effect on the output of the computer model could be removed from the model to simplify the calculations. There are many sensitivity analysis methods available, but we will concentrate on just a few methods here (See Saltelli et al. (2000) and Hamby (1994) for wider reviews of the literature).

Saltelli et al. (2000) groups SA into three classes. Firstly screening methods, such as presented in Morris (1991), which identify the parameters with the most effect on the variance of the output. Screening methods are computationally inexpensive, but only give qualitative information. The second class is local SA, which looks at the impact of input factors when they are varied by a small amount about a nominal value, usually the mean of the input factor. The sensitivities are usually found using derivatives and so these methods are useful when the output of the code is a linear function of the inputs, where derivatives of the output can be computed easily. The last class is global SA, which weights the uncertainty in the output to the uncertainty in the inputs, combining the influence of the entire range of uncertainty and distribution of each input.

Local and global SA are quantitative methods which provide more information than qualitative methods about how much more important one factor is than another. Global sensitivity analysis is preferred when there are a large number of inputs and the relationship between the inputs and outputs are non-linear (Cukier et al. (1973)). Here we will concentrate on global Monte Carlo based methods, as we do not expect our groundwater flow models to be linear. As in uncertainty analysis, the model is run many times using randomly selected inputs. The results of these runs are then used to split up the variation in the output to the sources of variation in the input factors. Many techniques can be used to achieve this, those discussed here involve correlation and regression analysis.

Scatterplots This is one of the simplest methods for sensitivity analysis. Scatterplots of the output variable against the sample for each input factor may reveal relationships between the model inputs and outputs. They only give qualitative measures of these relationships, and if there is a large number of inputs, there will be a large number of

plots to be examined.

Correlation coefficients Other simple measures are given by the Pearson product moment correlation coefficient and the Spearman correlation coefficient. These coefficients are recommended in Gardner et al. (1981) as a way of ranking model parameters in terms of their contribution to overall uncertainty in the output. Both coefficients give measures of the correlation between an input variable Θ_i and the output T.

The Pearson product moment correlation coefficient is found by dividing the covariance of the variables by the product of their standard deviation. For the correlation between Θ_i and T it is defined as:

$$r = \frac{\sum_{j=1}^{n} \left(\Theta_{ij} - \bar{\Theta}_i\right)\left(T_j - \bar{T}\right)}{\left[\sum_{j=1}^{n} \left(\Theta_{ij} - \bar{\Theta}_i\right)^2 \sum_{j=1}^{n} \left(T_j - \bar{T}\right)^2\right]^{\frac{1}{2}}}, \tag{2.1.3}$$

where $\bar{\Theta}_i$ and $\bar{T}$ are the respective means of Θ_i and T, and Θ_{ij} and T_j are samples from the distribution of Θ_i and T. The major drawback of this coefficient is that it assumes that the relationship between the inputs and outputs is linear. For non-linear models, the data can be ranked and the Spearman correlation coefficient is calculated as follows. The differences between each rank of corresponding values Θ and T, d_i, are calculated to give the coefficient:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)},$$

where n is the number of pairs of values. Spearman's correlation coefficient can also be calculated using equation (2.1.3) with the rank transformed data. The ranking of the data transforms the relationship between the inputs and outputs of the model from non-linear to linear by assuming a monotonic relationship between the input and output data. This may limit the use of the coefficient for more complex relationships, such as non-monotonic, between the inputs and outputs (Hamby (1994)).

Regression analysis This gives a more quantitative measure of sensitivity and is the basis for many other Monte Carlo SA methods and the emulation approach discussed in the next section. For and input θ_j, the computer model $t_j = f(\theta_j)$ is represented by a regression model of the form

$$t_j = b_0 + \sum_j b_j \theta_{ij} + \epsilon_i,$$

where b_j are unknown regression coefficients, $\theta_{ij}, j = 1, 2, \ldots$ are samples from the distribution of θ_j, and ϵ_i is the residual error due to the approximation of the computer model by the regression model. The b_j are determined, by a least-squares analysis or otherwise, and are then used as an indicator of how important each input value θ_j is with respect to the uncertainty in t. The regression model is standardised, so that all variables are placed on a common scale, and then rewritten as

$$\frac{t - \bar{t}}{\hat{s}} = \sum_j \frac{b_j \hat{s}_j}{\hat{s}} \frac{\theta_j - \bar{\theta}_j}{\hat{s}_j},$$

where

$$\bar{t} = \sum_i \frac{t_i}{N}, \bar{\theta}_j = \sum_i \frac{\theta_{ij}}{N}, \hat{s} = \left[\sum_i \frac{(t_i - \bar{t})^2}{N-1} \right]^{\frac{1}{2}}, \hat{s}_j = \left[\sum_i \frac{(\theta_{ij} - \bar{\theta}_j)^2}{N-1} \right]^{\frac{1}{2}}.$$

The standardised regression coefficients (SRCs), $\frac{b_j \hat{s}_j}{\hat{s}}$, can be used for sensitivity analysis as they quantify the effect of varying each input variable a small amount away from its mean, while keeping all other input variables constant.

Saltelli et al. (2000) discuss the importance of the coefficient of determination,

$$R_t^2 = \frac{\sum_{i=1}^N (\hat{t}_i - \bar{t})^2}{\sum_{i=1}^N (t_i - \bar{t})^2},$$

where $\hat{t}_i$ denotes the estimate of t_i given by the regression model, and the Predicted Error Sum of Squares (PRESS) when carrying out regression analysis. The value of R_t^2 determines the performance of the regression model. The closer R_t^2 is to 1, the better the model fits the data, and the more valid the SRCs are. The value of the PRESS determines the adequacy of the regression model. Several regression models are constructed using $N - 1$ of the N observations, the value of the omitted observation in each model is then estimated using the model, and the PRESS statistic calculated for each of the models. The best model is the one with the smallest PRESS value.

2.2 Analysis of complex computer models - Emulation approach

Whilst Monte Carlo is simple to carry out, its use may be limited by the time that the computer model takes to run. Monte Carlo analysis may require many thousands of

runs to converge and, even for a computer model that has a relatively short running time of a few seconds, this may take a long time to provide results.

For complex computer models, we can build an emulator to statistically approximate the computer model output. The emulator can then be used as a cheap approximation to the model when carrying out Monte Carlo analysis. These emulators are sometimes known by the alternative term meta-model. There are many different techniques for building approximations to computer codes, some of which are briefly outlined in Barton (1998).

Here we will give a brief overview of response surface methodology, kriging and then concentrate on the Gaussian process approach which we will use later in the thesis. All three methods use data from a number of runs of the code to create an emulator to approximate the computer model. Only a small number of runs of the expensive computer model is needed to build an emulator making it computationally cheaper to carry out the Monte Carlo analysis. It is important that the design of these runs is chosen to give a good representation of the input space, as the choice of experimental design for these runs affects the accuracies of these methods. A discussion of this is given by Allen et al. (2003), although they investigate only response surface methods and kriging. We will discuss choice of design points further in Section 2.2.3.

An emulator is a statistical representation of the function $\eta(.)$, with mean $\hat{\eta}(.)$. As the code may be very expensive, we are limited to running the code at a set of design points (inputs), $(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$. Data $(t = \eta(\boldsymbol{\theta}_1), \ldots, \eta(\boldsymbol{\theta}_n))$ are obtained from these runs. Using the data, we want to be able to make inferences about $\eta(\boldsymbol{\theta})$ for any $\boldsymbol{\theta} \in \times$, where $\times$ is the sample space of Θ.

The emulator can be used to provide a point estimate and variance for $\eta(\boldsymbol{\theta})$. A number of input configurations are sampled as before, but are evaluated using samples from the posterior distribution of the emulator instead of using the computer model. These emulator runs will take much less computational time than using the computer model. The emulator outputs are

$$\hat{t}_1 = \hat{\eta}(\boldsymbol{\theta}_1), ..., \hat{t}_n = \hat{\eta}(\boldsymbol{\theta}_n).$$

The sample mean $\bar{\hat{t}}$ will then be the estimate of $\mathrm{E}(T)$, and the sample variance will be an estimate of $\mathrm{Var}(T)$.

2.2.1 Response surface methodology

The response surface method described in Box and Draper (1987) seeks to relate a response (output) to a number of predictors (inputs). The output t is related to inputs $\boldsymbol{\theta}$ by a functional relationship $t = \eta(\boldsymbol{\theta})$. There may be little known about the relationship between the inputs and output, as in the case where $\eta(.)$ is uncertain until the model is run. If the relationship is assumed to be smooth, then $\eta(.)$ can be approximated by a regression function subject to error ϵ

$$\eta(\boldsymbol{\theta}) = \sum_{i=1}^{n} \beta_i \mathbf{h}_i(\boldsymbol{\theta}) + \epsilon, \tag{2.2.1}$$

where β are unknown regression coefficients, $\mathbf{h}(.)$ are specified regression functions. These commonly take the form $(1, \theta)^T$, so that a linear prior mean is provided. The computer code is evaluated at a set of design points. The β values are then usually found by least squares. We regard $\eta(\boldsymbol{\theta})$ as the mean response of the computer model at inputs $\boldsymbol{\theta}$. This can then be used as an estimate for the computer code output, subject to error. In this way it is emulating the computer code output.

This approach is recommended over Monte Carlo methods by Cox (1977) since it provides an inexpensive framework for evaluating the effects of the model inputs. Downing et al. (1985) also compare the use of response-surface methodology with the Monte Carlo approach for uncertainty analysis of a complex model. For their model, it is found that latin hypercube sampling of the original model produces a distribution for T which is closer to the Monte Carlo distribution than that obtained when using Monte Carlo analysis of a response surface with reduced input space. Downing et al. (1985) admit that the response surface model that they created may not be a good fit for their original model as it may only be a good enough approximation over a limited range of the input space. Response surface approximations may not give good approximations to highly nonlinear models or those with a large input space (Simpson et al. (2001) describes this as more than 10 inputs).

2.2.2 Kriging

Kriging has also been used to create a statistical approximation to the computer code output (Sacks et al. (1989); Currin et al. (1991); Martin and Simpson (2004)). This

method started as a way of predicting spatial data in geostatistics (Matheron (1971)) and we will discuss more details of the simple kriging predictor in this context in Chapter 3.

Sacks et al. (1989) use the simple kriging predictor due to its simplicity. This approach assumes that the mean and covariance structure are known. Therefore, in the same way as the response surface method, Sacks et al. (1989) assume that the mean of the output is a linear function of the inputs (2.2.1) with coefficients to be determined. They also assume the covariance function to be the exponential function

$$c(\boldsymbol{\theta}, \boldsymbol{\theta}') = \exp[-B(\boldsymbol{\theta} - \boldsymbol{\theta}')^2].$$

Using data

$$\mathbf{t} = (t_1 = \eta(\boldsymbol{\theta}_1), t_2 = \eta(\boldsymbol{\theta}_2), ..., t_n = \eta(\boldsymbol{\theta}_n))^T,$$

Sacks et al. (1989) consider $\hat{\eta}(\boldsymbol{\theta}) = c(\boldsymbol{\theta}, \boldsymbol{\theta})^T \mathbf{t}$ to be a linear predictor of $\eta(\boldsymbol{\theta})$ at an untried $\boldsymbol{\theta}$. They use a frequentist approach, treating this predictor as random, and replacing the data with a random vector. The mean squared error of the predictor is computed. Then the best linear unbiased predictor is obtained by choosing the B parameter in $c(\boldsymbol{\theta}, \boldsymbol{\theta})$ to minimise

$$MSE(\hat{\eta}(\boldsymbol{\theta})) = \mathrm{E}[c(\boldsymbol{\theta}, \boldsymbol{\theta})^T \mathbf{t} - \eta(\boldsymbol{\theta})]^2,$$

subject to the unbiasedness constraint

$$\mathrm{E}[c(\boldsymbol{\theta}, \boldsymbol{\theta})^T \mathbf{t}] = \mathrm{E}[\eta(\boldsymbol{\theta})].$$

Sacks et al. (1989) recognise that the frequentist approach will give the same result as the Bayesian approach when considering a Gaussian process for $Z(.)$ and improper uniform prior distributions for the βs. Kriging is more complex than response surface methods, but can deal with a larger number of inputs (Simpson et al. (2001) states that kriging can handle applications with up to 50 inputs, whereas response surfaces can only handle up to 10 inputs).

2.2.3 Choosing design points

As a run of the computer model may take a long time, the choice of which input values to use when running the computer model is important. We may only have a limited

number of runs from which to gain as much information as possible about the output of the computer model. Therefore we need to choose a design which covers the full range of uncertainty about the input values. We need a design which is spread across the sample space. If two input values are very close together, there may be issues when building the emulator as the variance-covariance matrix A may become singular and therefore cannot be inverted. There may also be problems if a large number of inputs are used to create an emulator as again the input values may be very close together.

The simplest way of choosing input values from which to run the computer code is to randomly sample from the input distribution $G(\boldsymbol{\theta})$. However, this may not provide a sample which best represents our uncertainty about the inputs. A larger sample size may be needed for the inputs to be spread across their sample space. We may also choose the inputs to be uniformly spread across the sample space of the inputs. However, with many inputs, this sampling design will generate a large number of inputs with which the computer model will need to be run.

Another way to sample the input values is to use stratified sampling. The sample space $\times$ is split into a number of strata, and a random sample taken from each strata. The random samples from all the strata are then put together as a sample from the input distribution. Latin Hypercube (LH) sampling described by McKay et al. (1979) is a form of stratified sampling, where the sample space is split into strata of equal probability. If we want a sample of size N, the sample space is split into strata with probability $\frac{1}{N}$. Then we sample once from each strata.

Using Latin Hypercube sampling to choose an input design may give better estimates of the posterior mean and variance of the output than simple random sampling (McKay et al. (1979)). Different Latin Hypercube samples can be generated when the sample space is split into the same strata. We can decide which of many LH samples to choose for our design points by using the maximin criterion as described by Morris and Mitchell (1995). The maximin criterion is to choose a design which gives the maximum minimum distance between two points, then the maximum second minimum distance and so on.

Busby (2009) suggests a hierarchical approach to choosing design points as an improvement on maximin Latin Hypercube design. The idea behind the procedure is to reduce the uncertainty in the emulator prediction by adding more design points in the areas of highest uncertainty. A maximin LH design is generated with $N_d = (d+2)(d+1)/2+10$

points, where d is the number of dimensions. This corresponds to the minimum number of points to build a quadratic response surface, and the computer model is run to obtain data with which to build the emulator. The input space is split into a number of smaller subspaces based on the correlation lengths of the observed data in each direction. Then an emulator is built and the accuracy in each subspace, α_i, is estimated using cross validation. These accuracies are compared with a preselected accuracy level, α, and where $\alpha_i > \alpha$ a new point is added to that subspace of the input design using a maximin design. The outputs at these new inputs are found by running the computer model, and a new emulator built using all of the data. The process is repeated, with more subspaces of smaller size in each iteration (due to the reduction in correlation length), until all $\alpha_i < \alpha$.

2.3 Gaussian Process emulators

The final method, which we will discuss in more detail than the previous methods, is to build a Gaussian process emulator to statistically approximate the output of the computer model. A non-technical tutorial discussing the main points of this approach is given in O'Hagan (2006). The method can be thought of as an extension of the kriging approach in a Bayesian setting.

The Bayesian approach to emulating computer models has wide applications and is popular due to its flexible framework capable of adapting to complex relationships between inputs and outputs (Liu and West (2004)). However, the method is not always computationally feasible for problems with high-dimensional outputs (Higdon et al. (2008)). An illustration of how Bayesian emulators can be used to deal with various problems with computer models, including prediction, uncertainty and sensitivity analyses and verification is given in Kennedy et al. (2006a) using three case studies in carbon dynamics. Gaussian process models have been widely used in many applications: a charged particle accelerator computer model (Higdon et al. (2004)), models of spot welding (Bayarri et al. (2007)), oil reservoir models (Kennedy and O'Hagan (2001)), nuclear release models (Kennedy and O'Hagan (2000)) and a health economic model (Rojnik and Naveršnik (2008)). Here we will concentrate on emulating deterministic computer models with a scalar output, but the approach has been extended to multidimensional

outputs (Rougier (2008)) and dynamic models (Conti et al. (2009)).

For the emulator, we treat $\eta(.)$ as a stochastic process, where $\eta(.)$ denotes the approximation to the output function over the entire sample space $\times$. The output can be considered as uncertain as its value is not known until the model is run, and it may be too expensive to run the code for all input distributions.

We start with the following model for $\eta(.)$

$$\eta(\boldsymbol{\theta}) = \sum_{i=1}^{q} \beta_i h_i(\boldsymbol{\theta}) + Z(\boldsymbol{\theta}), \tag{2.3.1}$$

where for each i, $h_i(\boldsymbol{\theta})$ is a specified regression function, β_i is an unknown regression coefficient and $Z(.)$ is a stochastic process with mean zero and covariance between $Z(\boldsymbol{\theta})$ and $Z(\boldsymbol{\theta'})$ given by some function $c(\boldsymbol{\theta}, \boldsymbol{\theta'})$, where $c(\boldsymbol{\theta}, \boldsymbol{\theta'})$ is a positive semi-definite function. A simple regression model on its own may not be a good enough approximation to the computer code. Therefore this model includes a stochastic process term to allow for any deviations from the regression term. Using the Bayesian approach, $Z(.)$ is treated as a Gaussian process and a posterior distribution for $\eta(\boldsymbol{\theta})$ is derived by updating the prior information using data, $\mathbf{t}$, obtained from runs of the computer code (Haylock and O'Hagan (1996)).

O'Hagan (2006) gives two criteria which the emulator must satisfy. First, at a design point, the emulator must have the same value as the computer model output. Secondly, the distribution for $\eta(\boldsymbol{\theta})$ must give a realistic mean and probability distribution about this mean, given the design data.

2.3.1 Prior assumptions

We can also think of $\eta(.)$ in equation (2.3.1) as our prior assumption of the output of the computer model, T,with the β's either specified or given a prior distribution. Therefore the general form for the prior mean is

$$\mathrm{E}[\eta(\boldsymbol{\theta}) \mid \boldsymbol{\beta}] = \mathbf{h}(\boldsymbol{\theta})^T \boldsymbol{\beta},$$

where $\mathbf{h}(.)$ is a vector of q known regression functions of $\boldsymbol{\theta}$, chosen to incorporate prior beliefs about $\eta(.)$, and $\boldsymbol{\beta}$ is a vector of q unknown coefficients. The most basic regression functions to use for $\mathbf{h}(.)$ is $(1, \boldsymbol{\theta})$, which provides a linear regression. This has been used

in previous emulators (Oakley and O'Hagan (2002); Kennedy et al. (2006b)) for its simplicity, and is the function we will use for the rest of this chapter. Other forms of the prior mean could be considered if they would provide a more accurate estimation of the output we are emulating. In Chapter 6 we consider the use of a distribution function as the prior mean for emulating the distribution function of the output.

The covariance between $\eta(\boldsymbol{\theta})$ and $\eta(\boldsymbol{\theta}')$,

$$\text{Cov}(\eta(\boldsymbol{\theta}), \eta(\boldsymbol{\theta}') \mid \beta, \sigma^2) = \sigma^2 c(\boldsymbol{\theta}, \boldsymbol{\theta}'), \tag{2.3.2}$$

must decrease as the distance between $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ increases, and must satisfy $c(\boldsymbol{\theta}, \boldsymbol{\theta}) = 1$ for all $\boldsymbol{\theta}$. We must also choose $c(.,.)$ so that the covariance matrix is positive semidefinite. A typical choice of function is

$$c(\boldsymbol{\theta}, \boldsymbol{\theta}') = \exp[-(\boldsymbol{\theta} - \boldsymbol{\theta}')^T B(\boldsymbol{\theta} - \boldsymbol{\theta}')], \tag{2.3.3}$$

where B is a diagonal matrix of positive smoothing parameters. For the following derivation of the emulator, we assume that B is known. We will discuss how we will approximate this matrix in Section 2.3.4. This choice of function implies that the output is a smooth function of the inputs, which is one of the main assumptions that we make when building the emulator. It also has the advantage that $\eta()$ has derivatives of all orders. The use of other covariance functions has been discussed (Sacks et al. (1989), Rougier et al. (2009)). For a general discussion on the properties of covariance functions see Stein (1999) and Cressie (1995). For the purposes of this thesis we will use equation (2.3.3).

For convenience, the conjugate prior for $\boldsymbol{\beta}$ and σ^2 is assumed by Oakley and O'Hagan (2002) to have a normal inverse gamma distribution:

$$p(\boldsymbol{\beta}, \sigma^2) \propto \sigma^{\frac{1}{2}(r+q+2)} \exp\left[-\frac{(\boldsymbol{\beta} - z)^T V^{-1}(\boldsymbol{\beta} - z) + a}{2\sigma^2}\right] \tag{2.3.4}$$

The weak form of this:

$$p(\boldsymbol{\beta}, \sigma^2) \propto \sigma^{-2}, \tag{2.3.5}$$

is generally used which implies infinite prior variance of $\eta(\boldsymbol{\theta})$ suggesting that there is little knowledge about output of the computer model. There may be cases when the developer of the computer model can provide some proper prior knowledge about $\eta(\boldsymbol{\theta})$,

and the process for dealing with this, and finding r, z, V and a for equation (2.3.4), is discussed in Oakley (2002).

To build the emulator, the model is run with a number of different design points $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$,...,$\boldsymbol{\theta}_n$. The data

$$\mathbf{t} = (t_1 = \eta(\boldsymbol{\theta}_1), t_2 = \eta(\boldsymbol{\theta}_2), ..., t_n = \eta(\boldsymbol{\theta}_n))^T$$

are then observed. This observations vector is assumed to have the distribution

$$\mathbf{t}|\boldsymbol{\beta}, \sigma^2 \sim N(H\boldsymbol{\beta}, \sigma^2 A), \tag{2.3.6}$$

where

$$\begin{aligned} H^T &= (\mathbf{h}(\boldsymbol{\theta}_1), \ldots, \mathbf{h}(\boldsymbol{\theta}_n)), \\ A &= \begin{pmatrix} 1 & c(\theta_1, \theta_2) & \cdots & c(\theta_1, \theta_n) \\ c(\theta_2, \theta_1) & 1 & & \vdots \\ \vdots & & \ddots & \\ c(\theta_n, \theta_1) & \cdots & & 1 \end{pmatrix}. \end{aligned}$$

2.3.2 Updating the prior

Let $\mathbf{z}$ be the joint distribution of $\eta(.)$ and $\mathbf{t}$ given $\boldsymbol{\beta}$ and σ^2. This distribution is multivariate normal of the form

$$\mathbf{z} \sim N(\boldsymbol{\mu}, \Sigma).$$

Now $\mathbf{z}$ can be split up into two vectors $\mathbf{z}_1$ and $\mathbf{z}_2$, with $\boldsymbol{\mu}$ and Σ split into

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix} \text{ and } \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}.$$

Then $\mathbf{z}_1|\mathbf{z}_2$ is also multivariate normal with mean

$$\boldsymbol{\mu}_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{z_2} - \boldsymbol{\mu}_2)$$

and variance-covariance matrix

$$\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}.$$

Using this property of multivariate distributions, the prior distribution of η is updated to

$$\eta(.)|\mathbf{t}, \boldsymbol{\beta}, \sigma^2 \sim N\left(m^*(.), \sigma^2 c^*(.,.)\right), \tag{2.3.7}$$

where

$$\begin{aligned} m^*(\boldsymbol{\theta}) &= \mathbf{h}(\boldsymbol{\theta})^T\beta + \mathbf{t}(\boldsymbol{\theta})^T A^{-1}(\mathbf{t} - H\boldsymbol{\beta}), \\ c^*(\boldsymbol{\theta}, \boldsymbol{\theta}') &= c(\boldsymbol{\theta}, \boldsymbol{\theta}') - \mathbf{t}(\boldsymbol{\theta})^T A^{-1}\mathbf{t}(\boldsymbol{\theta}') \\ \mathbf{t}(\boldsymbol{\theta})^T &= (c(\boldsymbol{\theta}, \boldsymbol{\theta}_1), \ldots, c(\boldsymbol{\theta}, \boldsymbol{\theta}_n)). \end{aligned}$$

2.3.3 Removing the conditioning on β and σ^2

It will usually be unrealistic to specify $\boldsymbol{\beta}$ and σ^2, so the posterior distribution on $\eta(.)|\mathbf{t}$ will need to be obtained. First we find

$$f(\eta(.), \boldsymbol{\beta}, \sigma^2|\mathbf{t}) = f(\eta(.)|\mathbf{t}, \boldsymbol{\beta}, \sigma^2) f(\boldsymbol{\beta}, \sigma^2|\mathbf{t}). \tag{2.3.8}$$

We already know the first term on the right hand side of equation (2.3.8), and so only need to find $f(\boldsymbol{\beta}, \sigma^2|\mathbf{t})$. Using Bayes' Theorem, we can obtain this up to proportionality with

$$f(\boldsymbol{\beta}, \sigma^2|\mathbf{t}) \propto f(\boldsymbol{\beta}, \sigma^2) f(\mathbf{t}|\boldsymbol{\beta}, \sigma^2). \tag{2.3.9}$$

From (2.3.6) the likelihood function of $\mathbf{t}$ is given by

$$f(\mathbf{t}|\boldsymbol{\beta}, \sigma^2) = \left(2\pi\sigma^2\right)^{\left(-\frac{n}{2}\right)} \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{t} - H\boldsymbol{\beta})^T A^{-1}(\mathbf{t} - H\boldsymbol{\beta})\right\}. \tag{2.3.10}$$

We note that (Oakley (1999))

$$(\mathbf{t} - H\boldsymbol{\beta})^T A^{-1}(\mathbf{t} - H\boldsymbol{\beta}) = (\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})^T H^T A^{-1} H(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}) + (n - q - 2)\hat{\sigma^2}$$

where

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= (H^T A^{-1} H)^{-1} H^T A^{-1}\mathbf{t}, \\ \hat{\sigma^2} &= \frac{\mathbf{t}^T(A^{-1} - A^{-1}H(H^T A^{-1}H)^{-1}H^T A^{-1})\mathbf{t}}{n - q - 2}. \end{aligned}$$

Combining the prior (2.3.5) with the likelihood (2.3.10) as in (2.3.9) we get the normal inverse gamma distribution:

$$f(\boldsymbol{\beta}, \sigma^2|\mathbf{t}) \propto \left(\sigma^2\right)^{-\frac{n}{2}-1} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})^T H^T A^{-1} H(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}) + (n - q - 2)\hat{\sigma^2}\right\}. \tag{2.3.11}$$

Then, treating σ^2 as a constant, we have

$$\boldsymbol{\beta} \sim N\left(\hat{\boldsymbol{\beta}}, \sigma^2(H^T A^{-1} H)^{-1}\right).$$

This can then be combined with (2.3.7), as in equation (2.3.8), and $\boldsymbol{\beta}$ integrated out to give

$$\eta(.)|\mathbf{t}, \sigma^2 \sim N\left(m^{**}(.), \sigma^2 c^{**}(.,.)\right), \tag{2.3.12}$$

where

$$\begin{aligned} m^{**}(\boldsymbol{\theta}) &= \mathbf{h}(\boldsymbol{\theta})^T \hat{\beta} + \mathbf{t}(\boldsymbol{\theta})^T A^{-1}(\mathbf{t} - H\hat{\boldsymbol{\beta}}), \\ c^{**}(\boldsymbol{\theta}, \boldsymbol{\theta}') &= c(\boldsymbol{\theta}, \boldsymbol{\theta}') - \mathbf{t}(\boldsymbol{\theta})^T A^{-1} \mathbf{t}(\boldsymbol{\theta}') \\ &\quad + (\mathbf{h}(\boldsymbol{\theta})^T - \mathbf{t}(\boldsymbol{\theta})^T A^{-1} H)(H^T A^{-1} H)^{-1} \\ &\quad \times (\mathbf{h}(\boldsymbol{\theta}')^T - \mathbf{t}(\boldsymbol{\theta}')^T A^{-1} H)^T. \end{aligned}$$

To remove the condition on σ we use

$$f(\eta(.), \sigma^2|\mathbf{t}) = f(\eta(.)|\mathbf{t}, \sigma^2) f(\sigma^2|\mathbf{t}),$$

and then integrate out σ^2 (Haylock and O'Hagan (1996)). The second term on the right hand side is obtained by integrating out $\boldsymbol{\beta}$ from (2.3.11) to give

$$\sigma^2|\mathbf{t} \sim (n - q - 2)\hat{\sigma^2}\chi^{-2}_{n-2}.$$

Combining this with (2.3.12), we obtain

$$\frac{\eta(\boldsymbol{\theta}) - m^{**}(\boldsymbol{\theta})}{\hat{\sigma}\sqrt{c^{**}(\boldsymbol{\theta}, \boldsymbol{\theta})}} \sim t_{n-q}. \tag{2.3.13}$$

The estimate of the expectation of $\eta(.)$ is given by $m^{**}(.)$. The first term of m^{**} is similar to the prior mean of $\eta(.)$, but with the β values updated taking into account the data. The second term adjusts the posterior mean so that the emulator has the same value as the computer model at the design points, so meeting the first criteria set out in O'Hagan (2006). The posterior covariance of $\eta(\boldsymbol{\theta})$ and $\eta(\boldsymbol{\theta}')$ is $\hat{\sigma}^2 c^{**}(\boldsymbol{\theta}, \boldsymbol{\theta}')$. At the design points where $\eta(\boldsymbol{\theta}_i) = t_i$ is known, $c^{**}(\boldsymbol{\theta}_i, \boldsymbol{\theta}) = 0$ for all $\boldsymbol{\theta}$. O'Hagan (2006) gives an example that shows by using the Bayesian Gaussian process approach, fewer runs of the computer code are needed to produce a similar estimate of the mean and standard deviation to that obtained when using the Monte Carlo approach with a larger number of runs.

2.3.4 Estimating smoothness parameters

The matrix of smoothing parameters B contained in the covariance function $c(.,.)$ is also uncertain as our uncertainty about $\eta(.)$ means we do not know how smooth $\eta(.)$ is. This uncertainty cannot be dealt with in the same way as $\boldsymbol{\beta}$ and σ^2. Instead, for simplicity, B is given a fixed value. This fixed value of B can be estimated by allowing B to vary and finding the value for which the emulator provides the best estimate of the computer model. The 'best' estimate is determined by the method used to find it. Using the data, we can find the estimate for B in two ways. For the first method the best estimate is that which maximises the posterior mode. In the second method, using cross validation, the best estimate for B is one which minimises a sum of squared distances. These methods are discussed below.

Estimating B using the posterior mode

Following Oakley (1999), we go through a similar process as before, but this time including the unknown variable B in the calculations. The likelihood function of $B, \boldsymbol{\beta}$ and σ^2 is

$$f\left(\mathbf{t}|\boldsymbol{\beta},\sigma^2,B\right) = \frac{|A|^{-\frac{1}{2}}}{(\sigma^2)^{\frac{n}{2}}(2\pi)^{\frac{n}{2}}}\exp\left\{-(\mathbf{t}-H\boldsymbol{\beta})^T\frac{A^{-1}}{2\sigma^2}(\mathbf{t}-H\boldsymbol{\beta})\right\}.$$

This likelihood function can be combined with the prior distributions for $B, \boldsymbol{\beta}$ and σ^2 using Bayes' theorem. If we consider non-informative priors for $\boldsymbol{\beta}$ and σ^2, and an improper uniform priors for the elements of B, the posterior density of $B, \boldsymbol{\beta}$ and σ^2 is

$$f\left(\boldsymbol{\beta},\sigma^2,B|\mathbf{t}\right) = \frac{|A|^{-\frac{1}{2}}}{(\sigma^2)^{\frac{1}{2}(n+2)}(2\pi)^{\frac{q}{2}}}\exp\left\{-(\mathbf{t}-H\boldsymbol{\beta})^T\frac{A^{-1}}{2\sigma^2}(\mathbf{t}-H\boldsymbol{\beta})\right\}. \tag{2.3.14}$$

To obtain the distribution of B conditional only on the data $\mathbf{t}$ we now need to marginalise (2.3.14) with respect to $\boldsymbol{\beta}$ and σ^2. Integrating out $\boldsymbol{\beta}$ from (2.3.14) we get

$$f\left(\sigma^2,B|\mathbf{t}\right) \propto \frac{|A|^{-\frac{1}{2}}|H^TA^{-1}H|^{-\frac{1}{2}}}{(\sigma^2)^{\frac{1}{2}(n+2-q)}}\exp\left\{-(\mathbf{t}-H\hat{\boldsymbol{\beta}})^T\frac{A^{-1}}{2\sigma^2}(\mathbf{t}-H\hat{\boldsymbol{\beta}})\right\}, \tag{2.3.15}$$

which is proportional to an inverse gamma function. Then integrating out σ^2 from (2.3.15) gives us

$$f\left(B|\mathbf{t}\right) \propto \left(\hat{\sigma^2}\right)^{-\frac{(n-q)}{2}}|A|^{-\frac{1}{2}}|H^TA^{-1}H|^{-\frac{1}{2}}. \tag{2.3.16}$$

To obtain our estimate of B, we find the value of B which maximises (2.3.16). For more than one input this maximisation can be achieved by using the Nelder Mead algorithm (Nelder and Mead (1965)).

Estimating B using cross validation

A cross validation method can also be used to estimate B. One observation $t_i = \eta(\boldsymbol{\theta}_i)$ is removed from the data $\mathbf{t}$ to give t_{-i}. Then for a chosen value of B, the posterior distribution of $\eta(.)$ is derived. The distance d_i between the posterior mean of $\eta(\boldsymbol{\theta}_i)$ and the observed value $t_i = \eta(\boldsymbol{\theta}_i)$ is then calculated. This process is repeated for $i = 1, \ldots, n$. The best choice for B is the one which minimises $\sum_{i=1}^{n} d_i^2$. This method of obtaining a value for B will only give a suitable value if $\eta(.)$ deviates smoothly from the regression function. This approach also works better than estimating B from the posterior mode when considering higher dimensional problems. However, through fixing B at a posterior estimate, uncertainty about B is not fully taken into account.

2.3.5 Gaussian Process emulators for uncertainty analysis

Rather than simply estimating the mean and variance of T from a sample of outputs using the emulator, Haylock and O'Hagan (1996) use the posterior distribution for $\eta(.)$ (equation 2.3.13) to estimate the expectation and the variance of the computer model output $T = \eta(\boldsymbol{\Theta})$. The expectation of the output, conditional on $\eta(.)$, over the sample space $\times$ is given by

$$K = \mathrm{E}[T|\eta(.)] = \int_{\times} \eta(\boldsymbol{\theta}) dG(\boldsymbol{\theta}). \tag{2.3.17}$$

As the posterior distribution of $\eta(.)$ is available at each point in the sample space, it is possible to derive the posterior distribution of K. Haylock and O'Hagan obtain the following posterior distribution for K.

$$\frac{K - \hat{K}}{\hat{\sigma}\sqrt{W}} \sim t_{n-q}, \tag{2.3.18}$$

where

$$\begin{aligned}
\hat{K} &= R\hat{\beta} + TA^{-1}(\mathbf{t} - H\hat{\beta}), \\
W &= U - TA^{-1}T^T + (R - TA^{-1}H)(H^TA^{-1}H)^{-1}(R - TA^{-1}H)^T, \\
R &= \int_{\times} \mathbf{h}(\boldsymbol{\theta})^T dG(\boldsymbol{\theta}), \\
T &= \int_{\times} \mathbf{t}(\boldsymbol{\theta})^T dG(\boldsymbol{\theta}), \\
W &= \int_{\times}\int_{\times} \mathbf{c}(\boldsymbol{\theta}, \boldsymbol{\theta'})^T dG(\boldsymbol{\theta}) dG(\boldsymbol{\theta'}).
\end{aligned}$$

A point estimate for K and variance of K can be found by calculating the mean and variance of the distribution (2.3.18).

The estimation of the variance of the output over the sample space $\times$ is given by

$$L = \mathrm{Var}[T|\eta(.)] = K_2 - \hat{K}^2,$$

where

$$K_2 = \int_{\times} \eta^2(\boldsymbol{\theta}) dG(\boldsymbol{\theta}).$$

It would be very difficult to derive this distribution. Therefore, Haylock and O'Hagan derive the first two posterior moments of L conditional on the data $\mathbf{t}$. This gives a posterior mean and variance, conditional on $\mathbf{t}$, of L, the variance of η over $\times$.

Oakley and O'Hagan (2002) extend this approach to make inferences about the distribution and density functions of the output T. They acknowledge that the analytical approach is not always practical and so use simulation to obtain summaries about T. The computational method used to generate draws from the distribution of $\eta(.)$ is as follows.

Step 1 Choose n' simulation points $\theta'_1, \ldots, \theta'_{n'}$.

Step 2 Generate random data $d_{(i)} = \{\eta_{(i)}(\theta'_1), \ldots, \eta_{(i)}(\theta'_n)\}$ where $\eta_{(i)}(.)$ denotes the function we wish to generate from the distribution of functions that the emulator describes.

Step 3 Approximate $\eta_{(i)}(.)$ by $m^{**}_{(i)}(.)$, the posterior mean of $\eta_{(i)}(.)$ given d and $d_{(i)}$.

To get a new realisation $\eta_{(j)}(.)$, the process is repeated. These realisations can then be used to find the distribution and density functions of T

Oakley and O'Hagan find the distribution function of T,

$$F(s) = \text{pr}(T \leq s) = \int_{\times} I\{\eta(\theta) \leq s\} dG(\theta),$$

where I is an indicator function, using the simulation approach. Draws of $F_{(i)}(.)$ from the posterior distribution of $F(.)$ are simulated as follows.

1. Obtain a realisation of $\eta_{(i)}(.)$ as above.
2. Draw a random sample of inputs $\theta_1^*, \ldots, \theta_N^*$ from $G(.)$ (N large).
3. Approximate $F_{(i)}(.)$ using

$$F_{(i)}(s) = \frac{1}{N} \sum_{j=1}^{N} I\{m_{(i)}^*(\theta_j^*) \leq s\}$$

4. Use M realisations, $F_{(i)}(.), \ldots, F_{(M)}(.)$ (M large), of the distribution functions to obtain any required inference about $F(.)$

The density function of T is also found using a straightforward method:

1. Randomly sample a number of inputs $\theta_1^*, \ldots, \theta_k^*$ from $G(.)$.
2. Estimate $\eta_{(i)}(\theta_j^*)$ by $m_{(i)}^{**}(\theta_j^*), j = 1, \ldots, k$.
3. Use kernel density estimation to estimate density function $f_{\eta_{(i)}(\Theta)}(.)$.

This process is repeated a number of times to obtain a sample of density functions $f_{T(1)}, \ldots, f_{T(M)}$ (M large). The median of this sample gives an estimate of $f_T(t)$.

Oakley and O'Hagan give examples of using these methods to obtain distribution and density functions. They found that their estimates of the distribution function used fewer model runs than Monte Carlo methods and that the estimates had narrower posterior intervals. They found that there was more uncertainty when estimating the density functions.

2.3.6 Gaussian Process emulators for sensitivity analysis

When considering complex computer models, the MC methods for SA in Section 2.1.3 will be very computationally expensive. Oakley and O'Hagan (2004) develop Bayesian

tools for SA when the model is very expensive. These can only be used when the model output can be represented by a smooth function of the inputs. The methods are based on the idea of building an emulator to statistically approximate the computer model as in section 2.1.2. Inferences about the main effects and interactions of the input variables can be made from the posterior distribution of $\eta(.)$ (equation (2.3.13)).

The posterior means of the main effect of input θ_i and the interaction between inputs θ_i and θ_j are given as

$$\mathrm{E}[\eta_i(\theta_i)] = R_i(\theta_i) - R\hat{\beta} + T_i(\theta_i) - T\mathbf{e},$$

and

$$\begin{aligned}\mathrm{E}[\eta_{ij}(\theta_{ij})] &= R_{ij}(\theta_{ij}) - R_i(\theta_i) - R_j(\theta_j) - R\hat{\beta} \\ &\quad + T_{ij}(\theta_{ij}) - T_i(\theta_i) - T_j(\theta_j) - T\mathbf{e},\end{aligned}$$

where, if p is a set of indices and $-p$ the set containing all indices except p,

$$\begin{aligned}R_p(\boldsymbol{\theta}) &= \int_{\times_{-p}} \mathbf{h}(\boldsymbol{\theta})^T dG_{-p|p}(\boldsymbol{\theta}_{-p}|\boldsymbol{\theta}_p), \\ T_p(\boldsymbol{\theta}) &= \int_{\times_{-p}} \mathbf{t}(\boldsymbol{\theta})^T dG_{-p|p}(\boldsymbol{\theta}_{-p}|\boldsymbol{\theta}_p), \\ \mathbf{e} &= A^{-1}(\mathbf{t} - H\hat{\beta}).\end{aligned}$$

As these are linear functionals of $\eta(.)$, they can be calculated and plotted against the relevant input variable. If the inputs are standardised, all variables can be placed on the same plot to give a graphical summary of how each input variable affects the output. However, posterior variances of the main effect or interaction term must be taken into account as the inputs that show most variation may not have the largest effect on the output of the computer model.

Oakley and O'Hagan (2004) use a regression model which approximately fits the regression coefficients to $T = \eta(\boldsymbol{\theta})$. The fit is judged by the expected square error. Regression fits are usually computed using a sampling based approach (such as Monte Carlo described above). However, Oakley and O'Hagan (2004) define the regression fits as functions of η, so there is no need to estimate these measures using a sample. Inferences about variances V_i can also be made from the posterior distribution of the model. The methods are very computationally efficient as they require fewer model runs than Monte Carlo methods, and one set of runs of the model provides enough information to calculate a complete range of sensitivity measures.

2.4 Using a GP emulator to statistically approximate deterministic models

We now consider two simple examples of using emulators to approximate a given function. This will allow us to compare the true solution to the estimated value given by the emulator.

2.4.1 1-D example

First we emulate the function

$$t(\theta) = 3\cos\theta + 4. \tag{2.4.1}$$

We evaluate this function at a number of θ values (our inputs) and then use the value of the function at those points as our observed outputs. We then build an emulator using the formulation described in Section 2.3, and the design input and output values, to give an estimated mean and variance for the function. In this example the smoothing parameter was set to unity, and we have used an exponential covariance function. The emulator mean and 95% bounds are plotted against the true function in Figure 2.1 for different numbers of design points.

Increasing the number of design points gives a better approximation to the true function. The mean of the emulator is a good approximation to the function, and the 95% bounds of the emulator encapsulate the true function. We can also see that there is no uncertainty at the data points. Outside the range of the design points, $\theta \in [0, 4]$, the variance of the emulator rapidly increases. This is because there are no data available outside this range to train the emulator. If we wanted to predict a value outside the input range, there would be large uncertainty in the predicted value. This could be remedied by including a new design point and building a new emulator.

2.4.2 4-D example

We also look at emulating the following function with four inputs:

$$3\cos\theta_1 + \sin\theta_2 + \theta_3^2 + 4\theta_4.$$

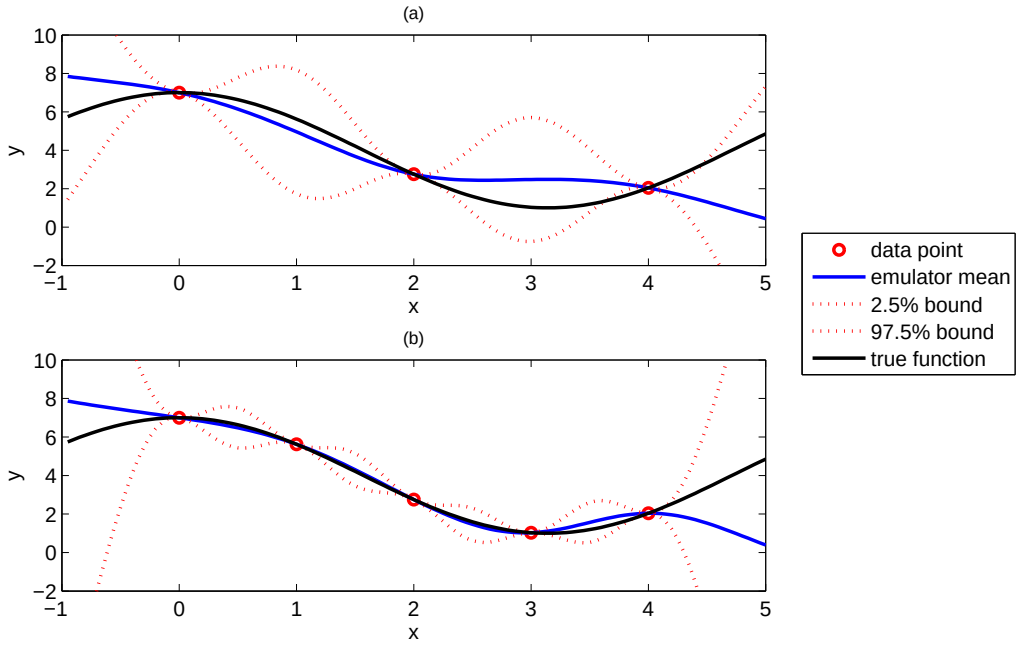

Figure 2.1: Emulator approximation to $t = 3\cos\theta + 4$ with (a) 3 design points and (b) 5 design points.

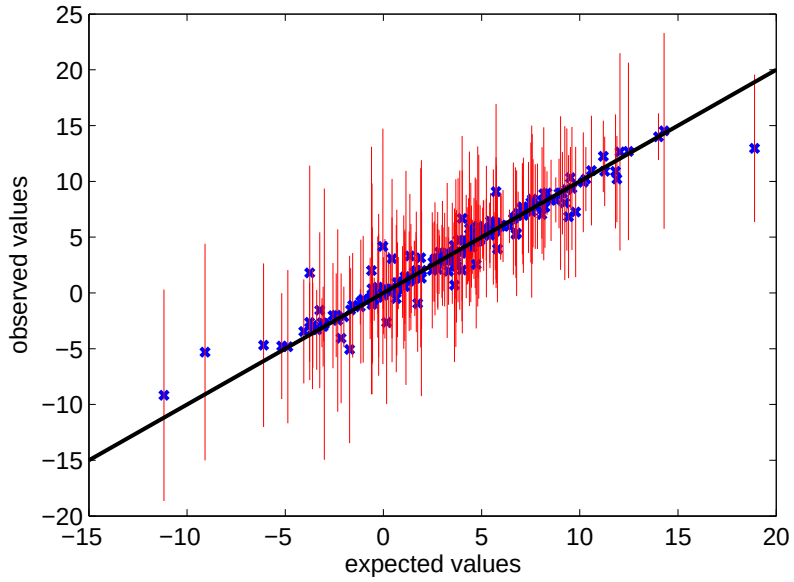

Figure 2.2: Estimated values (crosses) and 95% bounds given by the emulator against observed values for 4 inputs. Solid line shows observed = expected.

Again, we evaluate the function at a number of inputs (40 in this case) and then build an emulator using these inputs and the observed outputs. The smoothing parameter matrix was found from the posterior mode as in Section 2.3.4. Figure 2.2 shows the expected values, given by the emulator, against the true observed (function) values at a further 100 points evenly spread across the input space. We see that the 95% bounds include the observed values for each of the further 100 points in the input space.

2.5 Using a GP Emulator to statistically approximate stochastic models

So far, we have discussed how to create an emulator for deterministic computer models. Many computer models, including those we wish to develop later in this thesis, are not deterministic but stochastic; different outputs may be obtained when using the same input value. In this case, we can use an emulator to approximate the mean of the stochastic output of the computer model.

Whilst there is a large literature on the emulation of deterministic computer models, the emulation of stochastic models has not been as widely studied. Kleijnen (2009) and van Beers and Kleijnen (2008) have studied the use of kriging to approximate stochastic queuing models and are mainly concerned with emulating the mean of the model output. They take a sample of outputs at each design input and then emulate the mean output at each input. Bates et al. (2006) include the random "noise factor" inputs along with the usual "design factor" inputs in their emulation approach to engineering problems. They build an emulator for this deterministic problem. This emulator is then run with a new set of design inputs which covers the "design factors" only, with replications taken by sampling from the "noise factor" inputs. They then have a sample of outputs for each of the design inputs from which they can calculate statistics such as the mean or variance of the output. They can then build a second emulator to approximate the statistic of interest, given a "design factor" input.

2.5.1 Changes to the emulator equations

We think of the outputs as being the mean value of the output plus an error term. This is the approach used by van Beers and Kleijnen (2008). For input θ_i, we observe a number of outputs:

$$t_{ij} = \mathrm{E}[\eta(\theta_i)] + \epsilon_j(\theta_i),\ j = 1, \ldots, M.$$

The estimate of $\bar{t}_i$ to use as the observed output of the computer model is given by the mean of these observations

$$\hat{\bar{t}}_i = \frac{1}{M}\sum_{j=1}^{M} t_{ij}.$$

The estimated mean outputs, $\hat{\bar{t}}_i$, will then have variance

$$\begin{aligned} \mathrm{Var}[\hat{t}_i] &= \frac{\hat{\sigma}_i^2}{M} \\ &= \frac{\sum_{j=1}^{M}\left(t_{ij} - \hat{\bar{t}}_i\right)^2}{M(M-1)}, \end{aligned}$$

where $\hat{\sigma}_i^2$ is an estimate of the variance of the observed outputs for input θ_i.

The variance of each output $\bar{t}_i$ will need to be incorporated into the assumed distribution for the mean output vector t. Thus the assumed distribution (2.3.6) becomes

$$\mathbf{t}|\boldsymbol{\beta}, \sigma^2, \hat{\boldsymbol{\sigma}}^2 \sim N\left(H\boldsymbol{\beta}, D\right),$$

where

$$\begin{aligned} D &= \sigma^2 A + diag\left(\frac{\hat{\boldsymbol{\sigma}}^2}{M}\right), \\ \frac{\hat{\boldsymbol{\sigma}}^2}{M} &= \left(\frac{\hat{\sigma}_1^2}{M}, \frac{\hat{\sigma}_2^2}{M}, \ldots, \frac{\hat{\sigma}_n^2}{M}\right)^T. \end{aligned}$$

This 'nugget' allows the emulator to deviate away from the estimated means, $\bar{t}_i$, at the design points. If the variances are large, the resulting mean emulator function will be very smooth and may not pass through any of the estimated mean values, $\bar{t}_i$. The emulator variance will increase to allow for the increase in uncertainty of its approximation to the computer code.

The equations follow through as before, until removing the conditioning on σ^2 as σ^2 cannot be removed from the covariance of $\eta(.)$. Therefore we get the following distribution for $\eta(.)$:

$$\eta(.)|\mathbf{t}, \sigma^2, \hat{\boldsymbol{\sigma}}^2 \sim N\left(m^{**}(.), \hat{c}^{**}\right),$$

where

$$\begin{aligned} m^{**}(\boldsymbol{\theta}) &= \mathbf{h}(\boldsymbol{\theta})^T\hat{\beta} + \mathbf{t}(\boldsymbol{\theta})^T D^{-1}(\mathbf{t} - H\hat{\boldsymbol{\beta}}), \\ \hat{c}^{**}(\boldsymbol{\theta}, \boldsymbol{\theta}') &= c(\boldsymbol{\theta}, \boldsymbol{\theta}') - \mathbf{t}(\boldsymbol{\theta})^T D^{-1}\mathbf{t}(\boldsymbol{\theta}') \\ &\quad +(\mathbf{h}(\boldsymbol{\theta})^T - \mathbf{t}(\boldsymbol{\theta})^T D^{-1}H)(H^T A^{-1}H)^{-1} \\ &\quad \times(\mathbf{h}(\boldsymbol{\theta}')^T - \mathbf{t}(\boldsymbol{\theta}')^T D^{-1}H)^T. \end{aligned}$$

We can estimate σ^2 using a cross validation method in the same way as we discussed estimating B at the end of Section 2.3.4. One observation is removed from the data $\mathbf{t}$, and for a chosen value of σ^2, the posterior distribution of $\eta(.)$ is derived. The distance d_i between the posterior mean of $\eta(\boldsymbol{\theta})$ and the observed value t_i is calculated. The process is repeated by removing each of the other input data points in turn, and calculating the distances, $d_i, i = 1, \ldots, n$. The best value for σ is that which minimises $\sum_{i=1}^{n} d_i^2$.

2.5.2 1-D Example for emulating a stochastic equation

We consider a simple example of using an emulator to approximate the mean of the random function

$$t(\theta) = a\cos\theta + 4, \tag{2.5.1}$$

where a is a normally distributed random variable with mean 3 and variance 4. We choose this example as the mean of this function is the same as equation (2.4.1). Therefore, if M is a large enough sample, our emulator output should be similar to that shown in Figure 2.1. When emulating this function, we assume that this function is a black box, and that we do not know the relationship between the inputs and outputs, or the distribution of the random variable a.

When evaluated at the same input value θ, equation (2.5.1) will produce different output values t due to the variability in a. Therefore, when evaluating this function for a design point θ_i, we need to evaluate a large number of times to give a sample of the mean value t_i at θ_i. We can then calculate the sample mean and variance of this sample of outputs at the design point θ_i. The data point t_i can then be estimated by the sample mean value, and the sample variance is included in the covariance matrix as described above.

Figure 2.3 shows the emulator mean and 95% bounds. The equation $t = 3\cos\theta + 4$ is also plotted, as this is the mean of equation (2.5.1). This curve passes near the data

points, but not through them, as the data points are only estimates of the mean. Since the equation we are emulating is stochastic, the emulator mean is close to, but does not pass through, the data points. This smoothing is caused by including the variance of the sample outputs at each input.

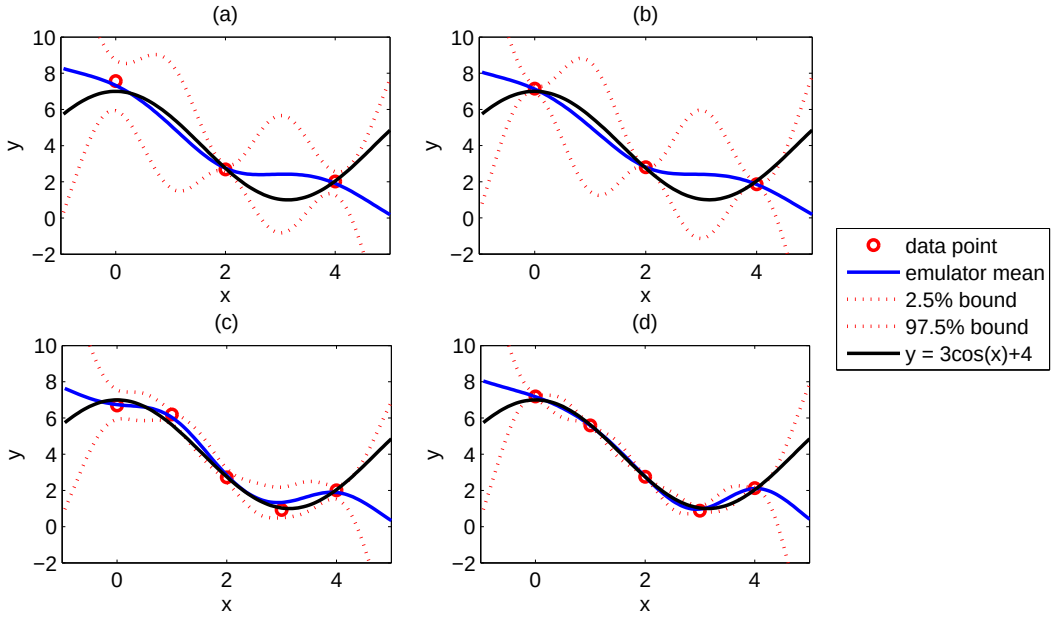

Figure 2.3: Emulator approximation to $t = a\cos\theta + 4$, where $a \sim N(3, 4)$, with (a) 3 design points and 10 evaluations of t, (b) 3 design points and 100 evaluations of t, (c) 5 design points and 10 evaluations of t, and (d) 5 design points and 100 evaluations of t.

The plots (a) and (c) on the left hand side use only 10 evaluations of the stochastic function (2.5.1), yet the emulator mean still gives a reasonable estimation of the mean of t. We also notice that the 95% bounds about the emulator mean are larger, since we have only a few evaluations of t and so we have more uncertainty about the value of the t. In the right hand plots (b) and (d), 100 evaluations of t have been carried out, and we can see that the emulator mean is closer to to true mean of t, and the variance of the emulator decreases. The right hand plots are similar to those in Figure 2.1, which is as we expected.

Including more design points and carrying out more evaluations at each design point improves the accuracy of the emulator, but it can be computationally expensive. For

complex stochastic models, and where the number of dimensions is much larger, the time taken to evaluate the model a large number of times for each design point could be immense. Therefore, for stochastic models, we need to weigh up the value of obtaining accurate means and variances of output data for fewer design points, where the emulator accuracy may be lower between these points, or of including output data from more design points, which may improve the accuracy of the emulator, but with less accuracy on the means and variances of the data points.

2.5.3 Emulating stochastic models when the stochastic variable has a known distribution

In the previous example, we assume that the distribution of the random variable a in equation (2.5.1) was unknown. Therefore the function was evaluated many times at each input point to give a distribution of the output at that point. In this case it did not take a long time to evaluate the function. However, if we considered a more computationally expensive stochastic computer code instead of a simple function it would take a long time to run the model a large number of times for each input. This leads us to consider another way of approximating the stochastic code by including the random variables responsible for the stochastic nature of the code as inputs, which is similar to the approach of Bates et al. (2006).

This method can only be used if the distribution of any random variables in the model is known, and that these random variables can be used as inputs to the computer code. The code then becomes a deterministic function of inputs $\boldsymbol{\theta}$ and random inputs $\mathbf{a}$, since for any one combination of inputs the same output will be obtained for every run of the model. This deterministic function $t(\boldsymbol{\theta}, \mathbf{a})$ can then be emulated using the methods in Section 2.3 to give an emulator $\eta(\theta, a)$. We then run this emulator with a sample of θ-values, to build a second emulator $\eta(\theta)$ for the mean of the stochastic function $t(\theta)$ as follows:

1. For each input $\theta_i, i = 1, \ldots, n$, the emulator is run with a number of samples of $a_j, j = 1, \ldots, M$ from $f(a)$ to give a sample of outputs $\eta(\theta_i, a_j), j = 1, \ldots, M$.

2. The mean of the samples, and the variance of the mean for each input θ_i are

calculated using:

$$\begin{aligned} \bar{t}_i &= \frac{1}{M}\sum_{j=1}^{M}\eta(\theta_i, a_j). \\ \widehat{\mathrm{Var}}[t_i] &= \frac{\sum_{j=1}^{M}\left(\eta(\theta_i, a_j) - \bar{t}_i\right)^2}{M-1}. \end{aligned}$$

3. The calculated mean and variance of the mean for each θ_i are then used as in the first method to emulate the mean of the stochastic function $t(\theta)$.

This second approach is more complicated than the first, but it means that the stochastic computer code needs to be evaluated fewer times. For computer codes with a higher computational cost this is an advantage, but this second method can only be used when the random variables which describe the stochastic model can be used as inputs to the model and their distributions are known. If the random variable a has been incorrectly specified then the process of choosing design points for a, and obtaining data from the computer code will need to be repeated for the correct distribution.

2.5.4 1-D example of emulating a stochastic function when the distribution of the stochastic variables is known

We emulate the same stochastic function as before, but this time consider that we know the distribution of the random variable a. Therefore, we can write equation (2.5.1) as:

$$t(\theta, a) = a\cos\theta + 4, \tag{2.5.2}$$

where a is again a normally distributed random variable with mean 3 and variance 4. We emulate equation (2.5.2) by sampling the input points over the two dimensional (θ, a)-space using a latin hypercube sample. This gives us an emulator $\eta(\theta, a)$. This emulator is repeatedly evaluated at a small sample of θ using samples from the distribution of a to provide an output mean and variance of the mean for each of the design points. These are used to build a second emulator for the stochastic function (2.5.1).

Figure 2.4 shows a comparison of the two methods for approximating the mean of the stochastic function (2.5.1). Plot (a) shows the emulator mean and 95% bounds using the first method. For this plot, 100 evaluations of $t(\theta)$ were carried out at each of the 5 input points, giving a total of 500 runs of the function. Plot (b) shows the emulator mean and

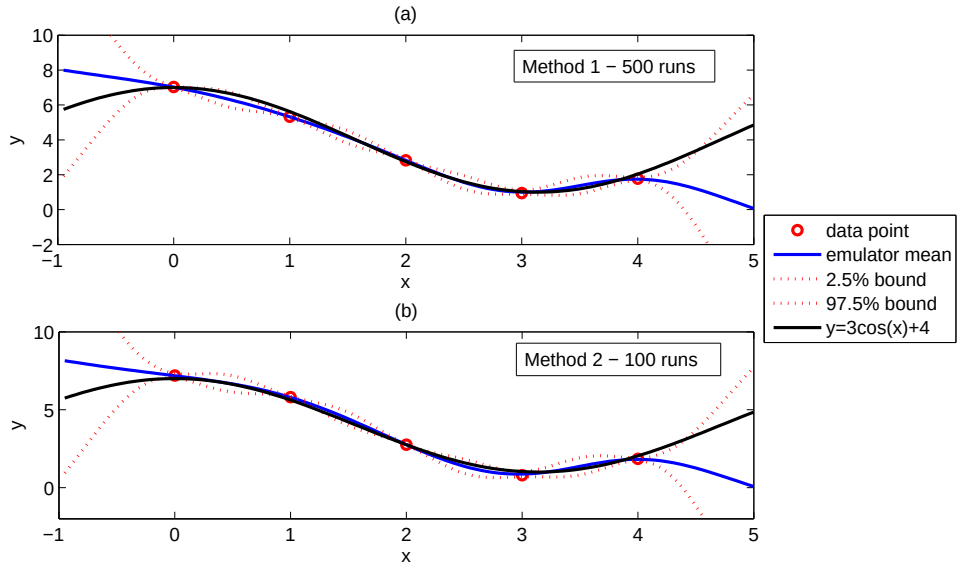

Figure 2.4: Emulator approximation to $t = a\cos\theta + 4$, where $a \sim N(3, 4)$, using (a) first method with 100 evaluations of $t(\theta)$ at each data point θ to obtain output data at each point (b) second method with 100 evaluations of $t(\theta, a)$ across the (θ, a)-space to build an initial emulator, then 100 runs of the emulator at each data point θ to obtain output data at each point.

variance using the second method. For this plot, 100 training runs were taken across the two-dimensional (θ, a)-space for the initial emulator. This emulator was then evaluated 100 times at each of the 5 input points using 100 values of a. A second emulator was then built using the mean and variance of the samples at each input point.

As in the previous example, we see that outside the range of the design points, the emulator does not perform very well. The results of the second method are comparable to the first method, but only 100 runs of the initial function were required for the second method instead of 500 used in the first method. In this example it is not a problem to evaluate the function 500 times, but for more complex computer codes where running the code a large number of times is very expensive, it is desirable to run the code as few times as possible. Therefore, if the distributions of the random variables in the model are known, and the random variables can be used as inputs to the model, it is preferable to use the second method to approximate the mean of the stochastic function.

Chapter 3

Groundwater flow modelling

This chapter will explore the main ideas behind groundwater modelling. First we will review the equations of groundwater flow. Then we will discuss how these equations have been used to model groundwater flow. We will also consider the sources of uncertainty that arise when modelling groundwater flow, and ways of dealing with this uncertainty.

3.1 Equations of groundwater flow

In this section we describe the basic concepts and equations of groundwater flow. Groundwater flow equations describe how groundwater flows through its environment. The main equation of flow is based on Darcy's law.

Before we look at the equations, we need to outline a few important concepts of groundwater flow. We have already mentioned that the rocks which make up the Earth's crust are porous. Porosity, ϕ (dimensionless), is the ratio of void volume in a rock to total volume, giving us an idea of how much fluid a rock can hold. We are interested in how this fluid will move through the rock. The permeability, k (m^2), of a rock measures the ability of the rock to transmit fluid through its pores . The driving force behind this movement of groundwater is head gradient. Head, h (m), is the height, above an arbitrary given level, that a fluid in a rock can reach due to the fluid pressure, p ($\mathrm{kgm}^{-1}\mathrm{s}^{-2}$), the density of the fluid, ρ (kgm^{-3}), and acceleration due to gravity, g ($\approx 9.81\mathrm{ms}^{-2}$). Head gradient is the difference in head between two points in a region over the distance between those two points. Fluid will always flow from areas of high head to areas of low

head.

3.1.1 Darcy's law

Darcy's law (Darcy (1856)) is an empirical law which describes how a fluid flows through a porous medium. In three dimensions, Darcy's law can be expressed in the generalised form

$$\mathbf{q} = -\frac{\mathbf{k}}{\mu}(\nabla p - \rho g \hat{\mathbf{e}}_z),$$

where $\mathbf{q}$ is the Darcy flux (m s^{-1}), $\mathbf{k}$ is a tensor of permeability, μ is the fluid viscosity (kg $m^{-1}s^{-1}$), and $\hat{\mathbf{e}}_z$ is the unit vector in the vertical (z)-direction (Bear (1972)). All other terms are defined as above. Whilst the flux $\mathbf{q}$ has the units of velocity, it is not the velocity which the water travelling through the porous rock is experiencing. To obtain the velocity, we need to divide the flux by the porosity of the rock:

$$\mathbf{u} = \frac{\mathbf{q}}{\phi}. \tag{3.1.1}$$

This accounts for the fact that groundwater can only flow through the voids in the rock. If we consider the hydraulic conductivity

$$K = \frac{\mathbf{k}\rho g}{\mu},$$

and the head defined as

$$h = z + \frac{p}{\rho g},$$

where z is a datum level from which the head is measured, then we can write Darcy's law as

$$\begin{aligned} \mathbf{q} &= -\frac{\mathbf{k}\rho g}{\mu}\nabla\left(\frac{p}{\rho g} - z\right), \\ &= -K\nabla h. \end{aligned} \tag{3.1.2}$$

3.1.2 Continuity equation

Conservation of mass is given by the equation

$$\nabla \cdot \mathbf{u} = 0. \tag{3.1.3}$$

Combining Darcy's law (3.1.2) with the continuity equation (3.1.3) and (3.1.1) gives (de Marsily (1986))

$$\nabla \cdot K\nabla h = 0. \tag{3.1.4}$$

In some applications when modelling groundwater flow, the region of interest is much greater in magnitude in the horizontal direction than in the vertical direction. This would be the case if we were modelling the flow of groundwater through a thin layer of rock in a large geographical region. For this case, we assume that the flow is essentially horizontal and the governing equations can be simplified to two dimensional equations. If a vertical flow does exist, it is small relative to the horizontal component and so can be ignored.

In two-dimensions, the hydraulic conductivity, K (ms^{-1}), can be replaced by transmissivity T ($\mathrm{m}^2\mathrm{s}^{-1}$); the ability of the rock to transmit water. These two terms are related with the equation $T = Kb$, where b is the thickness of the thin layer of rock. If we think of hydraulic conductivity to represent how much groundwater can flow through a unit square in the region, for a given time and hydraulic gradient, then transmissivity represents the flow through a rectangle of unit width and height equal to the thickness of the layer of rock.

Therefore in two dimensions equation (3.1.4) becomes

$$\nabla \cdot T\nabla h = 0. \tag{3.1.5}$$

If the transmissivity is considered to be isotropic and constant throughout the region, then (3.1.5) reduces down to Laplace's equation

$$\nabla^2 h = 0. \tag{3.1.6}$$

Equations (3.1.5) and (3.1.6) are those most often used to model the flow of water through a region of rock. The next section describes groundwater flow modelling and the issues involved in generating solutions to these equations.

3.2 Groundwater flow modelling

Groundwater flow models have traditionally been deterministic models. Deterministic models rely on a high degree of understanding of a process within a system. The system

response can then be defined through the understanding of the governing processes. The accuracy of deterministic models is therefore partly dependent on how close the concepts of the governing process reflect the true process. Even if the model accurately represents the true process, there are a number of factors that are also needed to determine the response of the system. These are (Konikow and Mercer (1988))

- definition of the properties and boundaries of the domain in which the process is acting,
- the state of the system at some point in time,
- an estimate of what future processes will be.

In groundwater modelling there is often insufficient or inadequate data to enable these factors to be fully represented. Therefore the ability of traditional deterministic models to make predictions about the system response is limited. More recently, research has been carried out to try to incorporate the data inadequacies in a stochastic manner into the modelling approach. We will discuss the ways in which this has been carried out in this section.

3.2.1 Uncertainty in groundwater flow modelling

Much of the uncertainty in modelling groundwater flow comes from limited knowledge about the values of parameters in the model. Measurements of transmissivity, head and other variables can only be made at a few locations in the region. To solve the mathematical equations in section 3.1, we need to know the value of transmissivity everywhere in the region. Therefore, we need to estimate the transmissivity values everywhere in the region using the limited data collected at a small number of locations. This estimation introduces uncertainty into the model, which must be analysed. The uncertainty in the model parameters will then be propagated through the model to provide the uncertainty in the model output. This can be carried out using any of the uncertainty analysis methods, such as Monte Carlo methods (Lahkim and Garcia (1999)), described in Chapter 2.

Uncertainty can be reduced by including other information such as head data into the model. This can be used to validate the model, by comparing the computed head values

to measured values. If the model reproduces the head values at the measurement points, we can be more sure that our model gives a good representation of the true process. We discuss the inclusion of head data later in this Chapter.

If the values of transmissivity were to be measured across the entire region, the true transmissivity field could be known, and the groundwater flow equations solved exactly. However, it is neither practical or economically viable to measure the transmissivity values at more than a few locations. Therefore an estimate must be made from the limited data available. There are different ways of including the uncertainty of the transmissivity into the data. In the next two sections, we discuss the following methods; representing the transmissivity with a random field then using realisations of this field to solve the equations, and using perturbation methods to express the transmissivity as an expansion of terms then solving the equations for the higher order terms in the expansion.

3.3 Random field representation of transmissivity fields

The transmissivity of rocks in a region is inherently heterogeneous. To determine the level of heterogeneity over the region, measurements would need to be made at all points in the region. This would be impractical, requiring much time and expense. Therefore, a small number of measurements can be used to estimate the variability in the transmissivity field within a statistical framework. That is, the spatial variation of transmissivity can be characterised by its probability distribution estimated from a small sample of measurements. Transmissivity measurements have been shown to have a lognormal distribution (Hoeksema and Kitanidis (1985); de Marsily (1986)). Freeze (1975) used this statistical approach to represent transmissivity as a lognormal random variable in the analysis of uncertainty in groundwater flow modelling. It has been shown, however, that although the transmissivity values show large spatial variations, these variations are not entirely random but spatially correlated (Byers and Stephens (1983); Hoeksema and Kitanidis (1985); Russo and Bouton (1992)). Therefore, transmissivity is better represented as a stochastic process, instead of a single random variable.

Stochastic or random spatial process are also known as random fields or random functions. In this thesis, we will refer to the transmissivity to be determined everywhere in

the region as a random field. A random field $\mathbf{Z}(\mathbf{x})$ can be thought of as a set of random variables, each of which is associated with a point in space, $\mathbf{x}$. The statistical structure of the random field is determined by how all the of the random variables relate to each other. To solve equations using a computer model, the domain of interest is discretised into N points or nodes, $\mathbf{x}_1, \ldots, \mathbf{x}_N$ where $\mathbf{x}_i$ is the position of node $i, i = 1, \ldots, N$. To construct a discrete transmissivity field on this domain, we can think of the joint cdf of $\mathbf{Z}(\mathbf{x}) = \{Z(\mathbf{x}_1), \ldots, Z(\mathbf{x}_N)\}$ as a multidimensional random variable.

Taking one sample or realisation of each of the random variables $\mathbf{Z}(\mathbf{x})$, we get a discretised function of $\mathbf{x}$. This realisation of the estimated field then gives a possible representation of the true field for use in a computer model. Due to the uncertainty we have about the true field, we consider many equally likely representations of the transmissivity field by sampling the random variables which make up the random field. The model can then be evaluated for each representation to give a sample of outputs. This Monte Carlo method of conditional simulations (Delhomme (1979)) then provides a distribution for the output, which can then be used in a risk analysis of a nuclear waste site, for example.

In groundwater modelling, it is often assumed that the log transmissivity field is a Gaussian random field defined by its mean and covariance function (Delhomme (1979)). That is, it is constructed from a set of random variables, each with a Gaussian distribution, to give a multidimensional Gaussian random variable. The mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ are usually estimated from the available data, and then representations of the random field need to be generated to solve the groundwater flow equations.

Methods of generating realisations of random fields can be conditional or unconditional. Unconditional realisations use the mean and covariance to generate spatial data with a known spatial distribution. Conditional realisations generate spatial data which preserve the values at the measurement points as well as having a known distribution. Unconditional realisation methods are usually the simplest, but do not take any measured data into account when generating the field. Therefore the generated field may not be as good an estimation of the true field as a conditioned field.

There are a number of ways of generating conditional realisations of Gaussian random fields which preserve the values at the measurement points and keep the statistical structure defined by the mean and covariance function. Both Schabenberger and Gotway

(2005) and Chiles and Delfiner (1999) have a chapter exploring ways of generating conditional realisations. In this review, we will concentrate on methods which generate realisations with a Gaussian spatial distribution.

3.3.1 Unconditional realisation of Gaussian random fields

The simplest method of generating an unconditioned field is based on being able to decompose a positive definite covariance matrix $\mathbf{\Sigma}$ as

$$\mathbf{\Sigma} = LL^T.$$

We also use the following reproductive property of the Gaussian distribution. Let $\mathbf{Y} \sim N(\mu, \mathbf{\Sigma})$, and let $\mathbf{a}, \mathbf{B}$ be constants. Then the following holds:

$$\mathbf{a} + \mathbf{BY} \sim N\left(\mathbf{a} + \mathbf{B}\boldsymbol{\mu}, \mathbf{B\Sigma B}^T\right).$$

Therefore, if $\mathbf{X} \sim N(\mathbf{0}, \mathbf{I})$, and we let $\mathbf{a} = \mu$, $\mathbf{B} = \mathbf{L}$, then

$$\boldsymbol{\mu} + \mathbf{LX} \sim N(\boldsymbol{\mu}, \mathbf{\Sigma})$$

(Schabenberger and Gotway (2005)). This means that if we can obtain the matrix $\mathbf{L}$, then along with a vector of independent Gaussian random variables $\mathbf{x} \sim N(\mathbf{0}, \mathbf{I})$, we can generate a realisation $\mathbf{z} = \boldsymbol{\mu} + \mathbf{Lx}$ from $N(\boldsymbol{\mu}, \mathbf{\Sigma})$.

Cholesky decomposition One way of obtaining the matrix $\mathbf{L}$ is to use Cholesky decomposition. The positive definite matrix $\mathbf{\Sigma}$ is decomposed into an upper triangular matrix $\mathbf{U}$ and a lower triangular matrix $\mathbf{L} = \mathbf{U}^T$, where $\mathbf{\Sigma} = \mathbf{LU}$. Realisations of the Gaussian random field from a $G_n(\boldsymbol{\mu}, \mathbf{\Sigma})$ distribution can then be generated simply by generating a vector, $\mathbf{x}$, of n independent Gaussian random variables with zero mean and unit covariance and calculating the Cholesky root $\mathbf{L}$ of the covariance matrix $\mathbf{\Sigma}$. An $n \times 1$ vector of means, $\boldsymbol{\mu}$, is then used to generate a realisation,

$$\mathbf{z} = \boldsymbol{\mu} + \mathbf{Lx},$$

from a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\mathbf{\Sigma}$. This method is good for small problems, but as the size, n, of the covariance matrix increases, it becomes more computationally expensive to decompose the covariance matrix.

3.3.2 Karhunen-Loéve expansion

Another way of generating realisations of the log transmissivity field comes from the property that every centred Gaussian process with a continuous covariance function has an expansion of the form of an orthogonal expansion (Adler and Taylor (2007)):

$$Z(\mathbf{x}) = \sum_{k=1}^{\infty} \xi_k \phi_k(\mathbf{x}), \tag{3.3.1}$$

where ξ_k are independent Gaussian random variables with zero mean and unit variance, and ϕ_k are functions on Ω, the space on which $Z(\mathbf{x})$ is defined, determined by the correlation function, $C(Z(\mathbf{x}), Z(\mathbf{x}'))$, between two points in the log transmissivity field. Here we will denote this by $C(Z(\mathbf{x}), Z(\mathbf{x}')) = C(\mathbf{x}, \mathbf{x}')$. By ordering the eigenvalues from largest to smallest, and truncating (3.3.1) at an appropriate point, we can generate an approximation to the Gaussian process by determining the $\phi_k, k \geq 1$. These can be found by solving an eigenfunction problem involving C. When Ω is a compact subset of $\Re^N$, then the eigenfunction problem takes the form of an integral equation,

$$\int_{\Omega} C(\mathbf{x}, \mathbf{x}')\psi_k(\mathbf{x}')d\mathbf{x}' = e_k\psi_k(\mathbf{x}), \tag{3.3.2}$$

where C is the correlation function between two points $\mathbf{x}$ and $\mathbf{x}'$ in Ω. The solutions e_k and ψ_k of (3.3.2) are the sets of eigenvalues and eigenfunctions respectively, of C.

Mercer's theorem (Mercer (1909)) states that the expansion

$$C(\mathbf{x}, \mathbf{x}') = \sum_{k=1}^{\infty} e_k\psi_k(\mathbf{x})\psi_k(\mathbf{x}')$$

converges absolutely and uniformly on Ω, with the result that $\{\sqrt{e_k}\psi_k\}$ is a complete orthonormal system. This leads to the Karhunen-Loéve expansion as described by Karhunen (1946) and Loéve (1955). We let $\phi_k = \sqrt{e_k}\psi_k$ in the expansion (3.3.1). Then, given the mean, μ, variance, ω^2, and correlation structure of the Gaussian process, we can represent the transmissivity field as the infinite sum of random functions

$$\mathbf{Z}(\mathbf{x}) = \boldsymbol{\mu} + \omega \sum_{k=1}^{\infty} \xi_k \sqrt{e_k}\psi_k(\mathbf{x}). \tag{3.3.3}$$

When approximating the transmissivity field, we consider a truncated Karhunen-Loéve expansion,

$$\mathbf{Z}(\mathbf{x}) = \boldsymbol{\mu} + \omega \sum_{k=1}^{N} \xi_k \sqrt{e_k}\psi_k(\mathbf{x}), \tag{3.3.4}$$

to reduce the number of degrees of freedom, and therefore the computational cost. For the purposes of computing the expansion using numerical methods, we must discretise the equations. Therefore, the covariance function C becomes a covariance matrix $\mathbf{\Sigma}$, and so e_k and ψ_k are now eigenvalues and eigenvectors of $\mathbf{\Sigma}$. We will describe the discretisation of these equations in more detail in Chapter 5.

The use of the truncated Karhunen-Loéve expansion relies on the majority of the variability in the transmissivity field being captured in the first few eigenvalues and eigenvectors (or eigenmodes), with the remaining eigenmodes providing the smaller scale variability. Each pair of eigenvalues and eigenvectors needs to be computed separately and so there is a large computational cost. Therefore, the truncation point N is chosen in order to capture most of the uncertainty in the transmissivity field in the smallest number of eigenmodes. We will see in Section 5.3.1 that the number of eigenmodes needed to capture most of the uncertainty is dependent on the correlation function chosen to represent the uncertainty in the transmissivity field. If a large number of eigenmodes is needed, other methods of generating random fields may be preferable.

3.3.3 Conditional realisation of Gaussian random fields

Conditional realisations $S(\mathbf{s})$ of a random field $Z(\mathbf{s})$ honour the observed values of $Z(\mathbf{s})$ at the data points $\mathbf{d} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_m\}$. A conditional realisation therefore consists of $n = m + k$ values:

$$\mathbf{S}(\mathbf{s}) = [Z(\mathbf{s}_1), Z(\mathbf{s}_2), \ldots, Z(\mathbf{s}_m), S(\mathbf{s}_{m+1}), \ldots, S(\mathbf{s}_{m+k})]^T.$$

Methods for generating Gaussian random fields either start with an unconditioned realisation which is then conditioned, or condition on the data directly.

Conditioning the Cholesky decomposition realisations The Cholesky decomposition method described above generates unconditioned realisations of the Gaussian random field at spatial locations $s_1, \ldots, s_n$. We want to condition this field on data values at locations $s_{01}, \ldots, s_{0m}$ to generate realisations such that at the data points the realisations have the same value as the data points.

The covariance matrix using all $n+m$ locations can be partitioned into

$$\mathbf{\Sigma} = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix},$$

where

$$\begin{aligned} & \Sigma_{11} && \text{has entries} && C(s_{0i} - s_{0j}), i, j = 1, \ldots, m, \\ & \Sigma_{12} && = \Sigma_{2,1}^T \quad \text{has entries} && C(s_{0i} - s_k), i = 1, \ldots, m, k = 1, \ldots, n, \\ \text{and} \quad & \Sigma_{22} && \text{has entries} && C(s_k - s_l), k, l = 1, \ldots, n. \end{aligned}$$

The Cholesky root can then be partitioned in the same way as Σ:

$$\mathbf{L} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix}.$$

The vector $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)^T$ now has $n+m$ entries, but only n of these are independent Gaussian random variables. To generate a realisation, we need to calculate

$$\mathbf{Lx} = \begin{pmatrix} L_{11}\mathbf{x}_1 \\ L_{21}\mathbf{x}_1 + L_{22}\mathbf{x}_2 \end{pmatrix}.$$

If $Z_1 = (Z(s_{0,1}), \ldots, Z(s_{0,m}))^T$ is the vector of data values, then to condition the realisation we set $L_{11}\mathbf{x}_1 = Z_1$. A realisation can then be calculated from

$$\mathbf{z} = \mu + \mathbf{Lx} = \mu + \begin{pmatrix} Z_1 \\ L_{21}L_{11}^{-1}Z_1 + L_{22}\mathbf{x}_2 \end{pmatrix}.$$

This method conditions the realisation on the data, but makes the Cholesky root slightly more computationally expensive to calculate since now Σ is an $(n+m)\times(n+m)$ matrix.

3.3.4 Conditioning by kriging

One of the most popular methods is to use kriging to condition the transmissivity field to the available data (Delhomme (1979); Gotway (1994)). The procedure is described in more detail below, but the main idea is to first generate an unconditional simulation using the mean and covariance structure inferred from the data, then condition this on the data using kriging. Because of the use of kriging in the conditioning step, the values of transmissivity are preserved at the measurement points, and the covariance structure

is kept. However as Delhomme (1979) discusses, there may be problems in determining the true correlation structure of the log transmissivity field as there may be only a small number of measurements with which to determine the properties of this field.

Before describing how kriging is used to condition a realisation of the transmissivity field, we first need to introduce the kriging predictor (Matheron (1971)). This predictor can then be used to condition an unconditioned field, to generate a realisation of the transmissivity field that honours the observed data.

Simple kriging Since kriging allows us to generate an estimate of the transmissivity field with a given mean and covariance structure, we can assume that the mean and covariance are known. Therefore, we describe the use of a simple kriging predictor to optimally estimate the transmissivity field from the measured data. We assume that

$$\mathbf{Z}(\mathbf{s}) = \boldsymbol{\mu}(\mathbf{s}) + \mathbf{e}(\mathbf{s}),$$

where $\mathbf{e}(\mathbf{s}) \sim N(\mathbf{0}, \boldsymbol{\Sigma})$, so that $\mathrm{E}[\mathbf{Z}(\mathbf{s})] = \boldsymbol{\mu}(\mathbf{s})$ and $\mathrm{Var}[\mathbf{Z}(\mathbf{s})] = \boldsymbol{\Sigma}$.

The aim is to find a predictor of $Z(\mathbf{s}_0)$, $p(\mathbf{Z}; \mathbf{s}_0)$, that minimises $\mathrm{E}\left[(p(\mathbf{Z}; \mathbf{s}_0) - Z(\mathbf{s}_0))^2\right]$, where $\mathbf{s}_0$ is a known spatial location where we want to predict the value of Z (Cressie (1995)). We can consider a linear predictor

$$p(\mathbf{Z}; \mathbf{s}_0) = \lambda_0 + \boldsymbol{\lambda}^T \mathbf{Z}(\mathbf{s}_0), \tag{3.3.5}$$

where $\lambda_0, \boldsymbol{\lambda}^T = [\lambda_1, \ldots, \lambda_n]^T$ are unknown coefficients to be determined. We can then write

$$\mathrm{E}\left[(p(\mathbf{Z}; \mathbf{s}_0) - Z(\mathbf{s}_0))^2\right] = \mathrm{E}\left[(\lambda_0 + \boldsymbol{\lambda}^T \mathbf{Z}(\mathbf{s}_0) - Z(\mathbf{s}_0))^2\right].$$

If we add and subtract $(\boldsymbol{\lambda}^T \boldsymbol{\mu}(\mathbf{s}) - \mu(\mathbf{s}_0))^2$ from the right hand side of this equation, where $\mu(\mathbf{s}_0) = \mathrm{E}[Z(\mathbf{s}_0)]$, it can be shown that

$$\mathrm{E}\left[(p(\mathbf{Z}; \mathbf{s}_0) - Z(\mathbf{s}_0))^2\right] = \mathrm{Var}\left[\boldsymbol{\lambda}^T \mathbf{Z}(\mathbf{s}) - Z(\mathbf{s}_0)\right] + \left(\lambda_0 + \boldsymbol{\lambda}^T \boldsymbol{\mu}(\mathbf{s}) - \mu(\mathbf{s}_0)\right)^2. \tag{3.3.6}$$

Since both terms on the right-hand side of equation (3.3.6) are non-negative, we can minimise $\mathrm{E}\left[(p(\mathbf{Z}; \mathbf{s}_0) - Z(\mathbf{s}_0))^2\right]$ by choosing $\lambda_0, \boldsymbol{\lambda}^T$ such that both terms are minimised.

The second term is minimised when

$$\lambda_0 = \mu(\mathbf{s}_0) - \boldsymbol{\lambda}^T \boldsymbol{\mu}(\mathbf{s}). \tag{3.3.7}$$

If we let $\text{Var}[Z(\mathbf{s}_0)] = \omega^2$ and $\text{Cov}[\mathbf{Z}(\mathbf{s}), Z(\mathbf{s}_0)] = \boldsymbol{\omega}$, then we can rewrite the first term as

$$\text{Var}\left[\boldsymbol{\lambda}^T\mathbf{Z}(\mathbf{s}) - Z(\mathbf{s}_0)\right] = \omega^2 + \boldsymbol{\lambda}^T\boldsymbol{\Sigma}\boldsymbol{\lambda} - 2\boldsymbol{\omega}^T\boldsymbol{\lambda}.$$

If we then differentiate the right hand side of this equation with respect to $\boldsymbol{\lambda}$ and equate it to zero, we get $\boldsymbol{\lambda}^T\boldsymbol{\Sigma} = \boldsymbol{\omega}^T$. Therefore, if $\boldsymbol{\Sigma}$ is non-singular, the first term on the right hand side of equation (3.3.6) is minimised when

$$\boldsymbol{\lambda} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\omega}. \tag{3.3.8}$$

Equations (3.3.7) and (3.3.8) give us values for the coefficients of the linear predictor (3.3.5). Therefore, the optimal linear predictor is given by

$$p(\mathbf{Z}; \mathbf{s}_0) = \mu(\mathbf{s}_0) + \boldsymbol{\omega}^T\boldsymbol{\Sigma}^{-1}\left(\mathbf{Z}(\mathbf{s}) - \boldsymbol{\mu}(\mathbf{s})\right). \tag{3.3.9}$$

This linear predictor uses the mean of the random field, and then adds a term which adjusts the mean so that it passes through the data points. We could use this optimal linear predictor as our estimate of the transmissivity field, since it has the same mean and variance structure as our data, and it honours the data at the measurement points. However, the resulting transmissivity field would be too smooth for the purpose of generating a realisation of a heterogenous field. Below we describe how this kriging predictor can be used to condition random realisations of the transmissivity field. The resulting realisations of the transmissivity field would be appropriately rough.

Conditioning a realisation using kriging Now that we have derived the kriging predictor, $p(\mathbf{Z}; \mathbf{s}_0)$, we can use this to condition a realisation of the transmissivity field to the measured data, $\mathbf{Z}(\mathbf{s}) = [Z(\mathbf{s}_1), \ldots, Z(\mathbf{s}_m)]^T$. We want to generate a random field with the same mean and covariance structure as $Z(s)$, that passes through these observed values. We can do this by considering an unconditioned realisation of the field $S(\mathbf{s})$ with the same covariance function as $Z(\mathbf{s})$.

We can write

$$Z(\mathbf{s}) = p(\mathbf{Z}; \mathbf{s}) + Z(\mathbf{s}) - p(\mathbf{Z}; \mathbf{s}),$$

where $p(\mathbf{Z}; \mathbf{s})$ is given by (3.3.9). The residual, $Z(\mathbf{s}) - p(\mathbf{Z}; \mathbf{s})$, cannot be obtained, and so we substitute it with the residual, $S(\mathbf{s}) - p(\mathbf{S}_m; \mathbf{s})$. Here, $\mathbf{S}_m = [S(\mathbf{s}_1), \ldots, S(\mathbf{s}_m)]^T$, and the simple kriging predictor, $p(\mathbf{S}_m; \mathbf{s})$, is based on the unconditional simulation at

locations, $\mathbf{s}_1, \ldots, \mathbf{s}_m$, where Z was measured. The conditional realisation can then be written

$$\begin{aligned} Z_c(\mathbf{s}) &= p(\mathbf{Z}; \mathbf{s}) + S(\mathbf{s}) - p(\mathbf{S}_m; \mathbf{s}) \\ &= S(\mathbf{s}) + \mathbf{c}^T \boldsymbol{\Sigma} \left(\mathbf{Z}(\mathbf{s}) - \mathbf{S}_m(\mathbf{s}) \right). \end{aligned} \tag{3.3.10}$$

The conditional realisation (3.3.10) corrects the unconditioned realisation $S(\mathbf{s})$, by adding the residual between values of the unconditioned field $\mathbf{S}_m(\mathbf{s})$ and the observed data $\mathbf{Z}(\mathbf{s})$ at the measurement points $\mathbf{s}_1, \ldots, \mathbf{s}_m$. Therefore, we obtain a realisation of the log transmissivity field that is not as smooth as one generated from the simple kriging predictor (3.3.9). We can also generate a sample of realisations to carry out a Monte Carlo analysis of the computer code, by conditioning a large number of randomly generated fields using this method.

3.4 Perturbation expansion

Dagan (1982) suggests perturbation methods as an alternative scheme to the conditional simulations presented by Delhomme (1979). The log transmissivity has been shown to be normally distributed (Freeze (1975); Hoeksema and Kitanidis (1985)) and so we consider this instead of transmissivity since it is simpler to use the multivariate normal distribution. The main idea of perturbation methods is to expand the log transmissivity about its mean value, where the mean and covariance structure have been inferred from the data, and then substitute the expansion into the groundwater flow equations. Terms of the same order of perturbation are then collected to give a set of equations. If we consider small perturbations, each resulting equation in the set can then be solved analytically. The solutions up to a desired order are then combined to provide an approximate solution to the groundwater flow equations.

Lu and Zhang (2004) compare the use of Monte Carlo simulations with a conventional perturbation approach, and a perturbation approach based on a Karhunen-Loéve expansion approach . They find that the computational cost is significantly lower when using the Karhunen-Loéve approach than when using the other two approaches. They also discuss that, unlike the conventional approach, the computational cost for the KL approach does not depend on the number of grid nodes in the discretised domain. Therefore, they

describe this approach as most suitable for applying to large-scale problems. Below, we will describe the perturbation expansion method algebraically using a Karhunen-Loéve approach as in Roy and Grilli (1997).

First we want to consider log transmissivity instead of transmissivity, so we substitute in $Z = \log T$ into Darcy's Law (3.1.5) to get

$$\nabla^2 h + \nabla Z \cdot \nabla h = 0. \tag{3.4.1}$$

Next, we expand the log transmissivity about its mean m_Z using its standard deviation ω_Z to get

$$Z = m_Z + \omega_Z \tilde{Z}, \tag{3.4.2}$$

where

$$\begin{aligned} \tilde{Z}(\mathbf{x}) &= \sum_{k=1}^{N} \xi_k g_k(\mathbf{x}), \\ g_k(\mathbf{x}) &= \sqrt{\lambda_k}\phi_k, \end{aligned}$$

ξ_k are independent normal random variables with zero mean and unit variance, and λ_k and ϕ_k are the eigenvalues and eigenvectors of the covariance matrix of log transmissivity.

Replacing Z by (3.4.2) in equation (3.4.1), we obtain

$$\nabla^2 h + \nabla m_Z \cdot \nabla h = -\omega_Z \nabla \tilde{Z}. \tag{3.4.3}$$

Then, assuming ω_Z is small, we can expand the solution for h in the following form:

$$h(\mathbf{x}) = h_0(\mathbf{x}) + \omega_Z h_1(\mathbf{x}) + \omega_Z^2 h_2(\mathbf{x}) + \ldots. \tag{3.4.4}$$

Putting this into (3.4.3) and identifying terms of the same order of ω_Z we get the following sequence of equations

$$\begin{aligned} \nabla^2 h_0 + \nabla m_Z \cdot \nabla h_0 &= 0, & (3.4.5)\\ \nabla^2 h_1 + \nabla m_Z \cdot \nabla h_1 &= -\nabla \tilde{Z} \cdot \nabla h_0, & (3.4.6)\\ \nabla^2 h_2 + \nabla m_Z \cdot \nabla h_2 &= -\nabla \tilde{Z} \cdot \nabla h_1, & (3.4.7)\\ &\vdots \quad . \end{aligned}$$

The first equation (3.4.5) is deterministic and can be solved analytically for h_0 given a specified mean m_Z and suitable boundary conditions. Equations (3.4.6) and (3.4.7)

are stochastic, but can be transformed into a sequence of deterministic functions by expanding h_1 on the same set of random variables, ξ_k, used to represent $\tilde{Z}$:

$$h_1(\mathbf{x}) = \sum_{k=1}^{\infty} \xi_k h_{1,k}(\mathbf{x}). \tag{3.4.8}$$

Substituting (3.4.8) into (3.4.6), and using the orthonormal property of the random variables ξ_k, we get

$$\nabla^2 h_{1,k} + \nabla m_Z \cdot \nabla h_{1,k} = -\nabla g_k \cdot \nabla h_0, \quad k = 1, 2, \ldots. \tag{3.4.9}$$

Then, given h_0, we can solve this for each $h_{1,k}$, $k = 1, \ldots, N$ where N is the truncation point of the infinite sum (3.4.8), given suitable boundary conditions.

We can approximate the head at this order of ω_Z as

$$h(\mathbf{x}) = h_0(\mathbf{x}) + \omega_Z \sum_{k=1}^{N} \xi_k h_{1,k}(\mathbf{x}), \tag{3.4.10}$$

where h_0 represents the mean field m_h and $\omega_Z h_1$ represents the random fluctuation about $m_h, \tilde{h} = h - m_h$. The next order of ω_Z could then be determined to provide corrections to m_h and $\tilde{h}$.

This method is simple, as it is easy to generate values from a multivariate normal distribution, and has the advantage of being able to solve the equations analytically. However, for large variances in the transmissivity values, the equations will need to be solved numerically and this advantage is lost (Dagan (1982)). Another issue with perturbation methods is discussed in Roy and Grilli (1997). They use perturbation methods to solve three test problems and find that while perturbation methods are more efficient than a Monte Carlo approach, the results are overestimated by 15-20%. They believe this may be due to the using a large value of $\omega_Z = 1$, which stretches the validity of the perturbation scheme. For moderately heterogeneous porous media where $\omega_Z^2 = 2$, Lu and Zhang (2004) find that the results of the perturbation approach deviates from the Monte Carlo approach, and so higher order corrections are needed. They also find that for highly heterogeneous porous media, the inclusion of higher order terms does improve the results, but there are still discrepancies with the Monte Carlo methods. Therefore, if the variability of the transmissivity field we wish to model is high, perturbation methods may not provide the best method for carrying out uncertainty analysis.

3.5 Use of head data to reduce uncertainty

As well as transmissivity data, which can be used to help determine the transmissivity field, there are often measurements of head in a region. We will not go into too much detail about this here as the extra complexity of incorporating this head data is beyond the scope of this thesis. Here we will briefly overview some of the methods used. For more comprehensive reviews of inverse modelling of groundwater models see Sun (1999) and McLaughlin and Townley. The head values can be used to validate the solution of the groundwater flow equations, by comparing the solutions of the equations to the measured values. If there is a large amount of difference between the two values, then the representation of the transmissivity field is not adequate. A traditional calibration process can be used to improve the model by varying the original model parameters until there is a best fit between the measured and predicted head values. This ad-hoc approach means that the model parameters will need to be changed an unknown number of times before a best fit is achieved.

Inverse modelling is an extension to this calibration procedure, where the adjustment of model parameters or other model aspects is automated within the model. Poeter and Hill (1997) demonstrate the benefits of inverse modelling, as opposed to non-automated calibration procedures, on a simple groundwater flow problem. Their method is to use least-squares regression to calculate residuals between the measured and calculated values and use these to obtain a weighted measure (objective function) of how well the model values match the measured values. The advantages of including the calibration in the model are that the calibration of parameters can be carried out much faster than the traditional ad-hoc approach, and the method is more thorough as sensitivities and correlations between parameters can be included to help with the parameter estimation. However, there is no guarantee that the model will be a more accurate representation of the true process.

A similar method is shown in more detail by Gómez-Hernánez et al. (1997). Rather than finding one optimal transmissivity field which, when substituted into the groundwater flow equations, reproduces the head values, they generate a large sample of transmissivity fields which honour both the transmissivity and head data. They stress that any solution which reproduces the spatial variability of the transmissivity and honours the transmissivity and head data is an acceptable solution. Their approach therefore lacks

the identifiability and non-uniqueness of other inverse problems.

Stuart (2010) presents a Bayesian approach to the inverse problem. He argues that solving a least squares optimisation problem may be difficult to solve as inverse problems are often ill-posed. This problem is overcome by finding a "probability measure" on the input space, which is the space of possible transmissivity fields in the case of groundwater flow modelling. The measure contains information about the relative probabilities of different inputs given the observed data, in this case the head measurements, subject to noise. The inverse problem is related to the more stable forward problem using this measure. The approach outlined in Stuart (2010) is applicable to a range of inverse problems for functions when formulated in a Bayesian fashion.

Chapter 4

WIPP model and data analysis

This chapter discusses the analysis of the available data for our case study based on the Waste Isolation Pilot Plant (WIPP). The equations describing groundwater flow in the region are introduced. The uncertainty in the transmissivity field of the WIPP region is discussed, along with the approach of using a stochastic model to represent the transmissivity field. We then discuss how the uncertainty in the model can be quantified using uncertainty analysis. Distributions for the uncertain hyperparameters of the stochastic model are then derived using Bayesian methodology. These distributions will be used to provide a sample of inputs for the groundwater flow model when the uncertainty analysis is carried out in Chapter 6.

4.1 WIPP data

The region we wish to model is a rectangular region Ω of Culebra Dolomite 21500m by 30500m. The WIPP site lies in the centre of Ω. Within this region, boreholes have been drilled at locations $\mathbf{x}_i, i = 1, \ldots, 39$, and the 39 measurements of transmissivity (Cauffman et al. (1990)) and their $\log_{10}$ values are shown in Table A.1 and in Figure 4.1. The WIPP site boundary, $\partial\Gamma$, is given by the inner rectangular region, where most of the values lie. From this data, we wish to find the transmissivity field $T(\mathbf{x}), \mathbf{x} \in \Omega$ of the entire region Ω for use in a computer model which estimates the time, t, taken for a particle to exit the site Γ. The next sections describe the mathematical equations for groundwater flow, and the stochastic model for the transmissivity field. These form the basis for the groundwater flow model of the WIPP site.

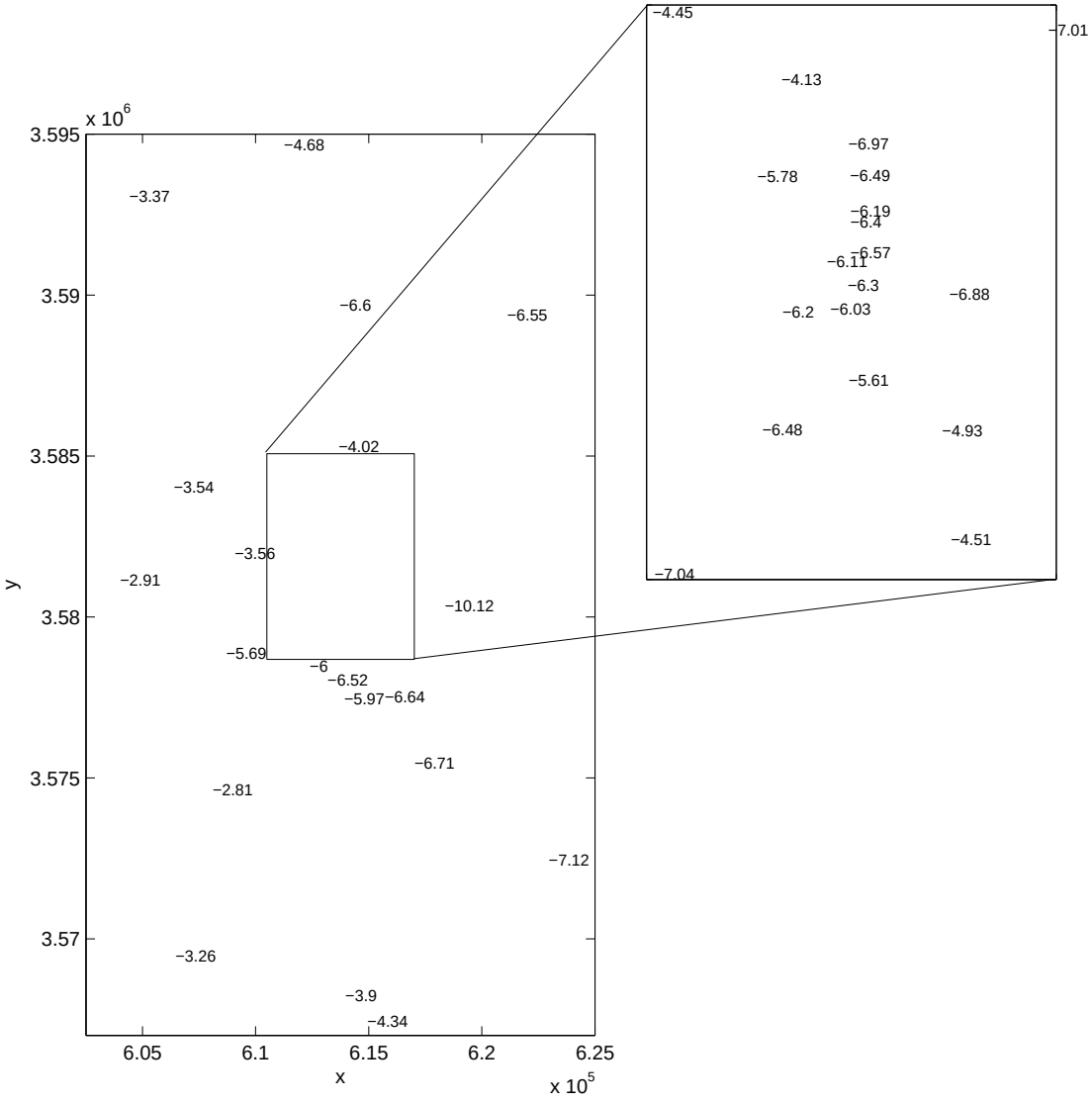

Figure 4.1: Locations and values of the $\log_{10}$ transmissivity data.

4.1.1 Groundwater flow equations

The Culebra dolomite is a thin layer of rock approximately 8m thick. This is very small in comparison to the size of the region which is about 655 km^2. Therefore, we consider the flow to be two dimensional. The groundwater flow equations are derived from Darcy's law

$$\mathbf{q} = -\frac{T(\mathbf{x})}{b}\nabla h(\mathbf{x}), \tag{3.1.2}$$

where $\mathbf{q}$ is the flux of groundwater across the region ($m^2 s^{-1}$), T is the transmissivity and h is the hydraulic head, and conservation of mass

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega, . \tag{4.1.1}$$

From these two equations we obtain the standard equation for representing steady-state groundwater flow in a two-dimensional region Ω (de Marsily (1986))

$$\frac{1}{b\phi}\nabla \cdot T(\mathbf{x})\nabla h(\mathbf{x}) = 0 \quad \text{in } \Omega. \tag{4.1.2}$$

We also have the boundary condition

$$h(\mathbf{x}) = h_0(\mathbf{x}) \quad \text{on } \partial\Omega, \tag{4.1.3}$$

where $\partial\Omega$ is the boundary of the region Ω. The boundary condition is obtained by extrapolating the head data at the boreholes to the boundary of the region. This is the approach used in the original analysis of the WIPP site. This does introduce another source of uncertainty into the problem, which could also be incorporated into our emulator. However, we do not investigate the uncertainty in the head boundary data as deriving a distribution for the boundary condition would overcomplicate the problem of investigating how the uncertainty in the transmissivity field affects the uncertainty in the travel time.

Given the boundary condition (4.1.3) and the transmissivity field, T, we solve equation (4.1.2) for h. However, we only know the values of transmissivity at a limited number of points across the region, and cannot possibly measure the values everywhere. Therefore we represent the transmissivity $T(\mathbf{x})$ as a random field. We will discuss how we are to model this field in the next section.

An important quantity in modelling the flow in the WIPP region is the time, t, at which a particle released in the centre of the region reaches the site boundary $\partial\Gamma$. This is

calculated using the transport equation

$$\dot{\boldsymbol{\zeta}}(t) = \frac{\mathbf{q}(\boldsymbol{\zeta})}{b\phi} = -\frac{T(\boldsymbol{\zeta})}{b\phi}\nabla h(\boldsymbol{\zeta}), \tag{4.1.4}$$

where $\boldsymbol{\zeta}(t)$ is the position of the particle at time t, b is the thickness of the rock and ϕ is the porosity of the rock. The thickness of the rock is considered constant, and here we use $b = 8$m as this is the average thickness of the Culebra in the region. We also consider the porosity of the rock to be constant, and use $\phi = 0.16$ as this is the value used in the original WIPP analysis. These two quantities could also be considered uncertain and included as inputs for our emulator. However, again we want to keep the problem simple and only consider investigating the uncertainty in the hyperparameters of the transmissivity field, which we believe have the greatest effect on the travel time, t.

Equation (4.1.4) is solved using the approximation for T, the calculated value for h and the initial condition

$$\boldsymbol{\zeta}(0) = \boldsymbol{\zeta}_0. \tag{4.1.5}$$

Other interesting quantities that can be obtained from calculating the travel time are the position along the boundary $\partial\Gamma$ at which the particle leaves the region, $\boldsymbol{\zeta}(t)$, and the velocity at the release point, $\dot{\boldsymbol{\zeta}}(0)$.

4.1.2 Stochastic model for the log transmissivity field

The values of transmissivity T have been measured at a limited number of locations across the region Ω. To solve the mathematical model describing groundwater flow in Ω, we need to know the values for all $\mathbf{x} \in \Omega$. At each point in the rock, transmissivity has one deterministic value. However, these values cannot possibly be measured everywhere, and so a physical interpretation of the rock is not feasible. Therefore, we represent the transmissivity $T(\mathbf{x})$ as a random field. Even though the true transmissivity is not a random quantity, we can represent it by a random field to represent our uncertainty about this field (refer to Section 3.3 for more discussion about this point). The observed data can be used to help us generate a number of realisations of the transmissivity field.

Before we see the data, we can express what we believe the properties of the rock to be in the prior information. We know that transmissivity values must be greater than zero,

therefore we consider log transmissivity ($\log_{10}(T) = Z(\mathbf{x})$). The accepted distribution for the transmissivity is a lognormal distribution. As discussed in Chapter 3, this arises from analysis of field data by Freeze (1975), and Hoeksema and Kitanidis (1985), who found that the transmissivity data displayed an approximately lognormal distribution. This leads us to investigate a Gaussian random field for the log transmissivity. Another prior assumption is that all points and directions are the same after removing any spatial trend; the Gaussian random field should be second order stationary and isotropic. Therefore the prior representation of the $Z(\mathbf{x})$ field is as a second order stationary, isotropic Gaussian random field.

The form of the mean of the Gaussian random field is unknown a priori, so we consider three different means for the log transmissivity field. The first mean we consider is a constant mean.

$$\mathrm{E}[Z(\mathbf{x})] = \beta. \tag{4.1.6}$$

This assumption that the mean of the log transmissivity field is the same everywhere in the region is the most basic model used in groundwater flow modelling. Another form for the mean is to consider a linear trend. In this case, we have

$$\mathrm{E}[Z(\mathbf{x}))] = \beta + \beta_x x + \beta_y y, \tag{4.1.7}$$

where x and y are the coordinates of $\mathbf{x}$. This mean allows for changes in the transmissivity field due to the position in the region, and any east-west or north-south trends in the data can be accounted for. The third mean we consider is dependent on the depth of the overlying rock

$$\mathrm{E}[Z(\mathbf{x}))] = \beta + \beta_d \mathrm{depth}(\mathbf{x}), \tag{4.1.8}$$

where depth($\mathbf{x}$) is the depth of the Culebra dolomite at position $\mathbf{x}$. The depth of the overlying rock has two effects on the transmissivity field. Firstly, the thicker the rock above the layer of Culebra, the more compressed in the rock will be leading to smaller fractures and so the transmissivity will be smaller. Secondly, where the overlying rock has been eroded away and is thinner the Culebra is subject to stress-relief fractures and so the transmissivity will be higher as the number of fractures in the rock will be greater.

We only have 39 transmissivity and 35 depth data values (Appendix A) with which to approximate the parameters of the transmissivity field, and so we want to keep the number of unknowns as small as possible. Therefore, will start by considering the mean

to be constant as in (4.1.6). We will investigate the linear and depth trends in the mean later.

We also assume that the variance is finite everywhere. The Gaussian random field representing the $Z(\mathbf{x})$ field has covariance

$$\mathrm{Cov}[Z(\mathbf{x}), Z(\mathbf{x}^*)] = \omega^2 \mathrm{Corr}(\mathbf{x}, \mathbf{x}^*), \tag{4.1.9}$$

where $\mathrm{Corr}(\mathbf{x}, \mathbf{x}^*)$ is the correlation function between points $\mathbf{x}$ and $\mathbf{x}^*$, and ω^2 is the variance of the $Z(\mathbf{x})$ field. As we are assuming that $Z(\mathbf{x})$ is second order stationary and isotropic, we can write

$$\begin{aligned} \mathrm{Corr}(\mathbf{x}, \mathbf{x}^*) &= c(\mathbf{x} - \mathbf{x}^*) \\ &= c(r), \end{aligned}$$

where r is the distance $\| \mathbf{x} - \mathbf{x}^* \|$ between two points in the region Ω.

There are a number of forms that the correlation function can take, and we need to make sure that the results are not sensitive to the particular form chosen. The main prior assumptions we have are that the transmissivities at points very far apart are uncorrelated, and that any realisation of the transmissivity field should be heterogeneous on small scales with appropriate "roughness". That is the field should not be too smooth. The most popular correlation function in the groundwater flow modelling literature is the exponential correlation function (Hoeksema and Kitanidis (1985); Gotway (1994)). The exponential correlation function is given by:

$$c(r) = \exp\left\{ -\left(\frac{r}{\lambda} \right)^{\kappa} \right\}, \tag{4.1.10}$$

where the correlation length, λ, is the length at which correlations are nearly zero, and κ controls the amount by which spatial variations in the data are smoothed. The usual value for groundwater flow modelling purposes is to set $\kappa = 1$ (Hoeksema and Kitanidis (1985); Gotway (1994); Dagan (1989)), since it results in a field which is not overly smooth.

The exponential correlation function with $\kappa = 1$ belongs to a larger family of correlation functions, the Matérn family (Stein (1999))

$$c(r) = \frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\frac{r}{\lambda} \right)^{\nu} K_{\nu}\left(\frac{r}{\lambda} \right), \tag{4.1.11}$$

where λ is the same as in (4.1.10), $\nu > 0$ determines the smoothness of the random field Z, and K_ν is the modified Bessel function of order ν. This parameterisation of the Matérn family is recommended by Handcock and Wallis (1994) as it does not depend on the dimension of the problem, and λ is largely independent of ν. The exponential correlation function with $\kappa = 1$ is a special case of the Matérn correlation function when $\nu = 0.5$. As $\nu \to \infty$, the Matérn function becomes similar to the Gaussian correlation function

$$c(r) = \exp\left\{-\left(\frac{r}{\lambda}\right)^2\right\}. \tag{4.1.12}$$

This correlation function leads to a very smooth random field $Z(\mathbf{x})$. For the purposes of using the correlation function to describe the correlation structure of the log transmissivity field, we do not wish the field to be too smooth. Therefore, if we were to use the Matérn family of correlation functions (4.1.11) to describe the correlation in $Z(\mathbf{x})$, we would need to consider smaller values of ν. Diggle et al. (2003) state that, rather than estimating ν from sparse data, it is sensible to choose ν from a small set of values which reflect the knowledge about the smoothness of Z. They suggest choosing ν from a discrete set $\{0.5, 1, 1.5, \ldots, \frac{K}{2}\}$, where K is a small integer.

Another function which is sometimes used is the spherical correlation function, which in two dimensions is given by

$$c(r) = \begin{cases} \frac{2}{\pi}\left(\arcsin\frac{r}{\lambda} - \frac{r}{\lambda}\sqrt{1-\left(\frac{r}{\lambda}\right)^2}\right) & \text{if } r \leq \lambda \\ 0 & \text{if } r \geq \lambda \end{cases} \tag{4.1.13}$$

This function gives no correlation between two points which are further apart than λ.

For simplicity, we choose the exponential correlation function (4.1.10). This representation of the log transmissivity field allows us to fit a model to the observed data, $\mathbf{d} = (Z(\mathbf{x}_1), \ldots, Z(\mathbf{x}_n))$, which allows for nonlinearity. The model also allows for a spatial correlation such that two values of transmissivity close together will be similar.

In order to generate a realisation of the transmissivity field from our stochastic model, we need to know the hyperparameters of the stochastic model, $\boldsymbol{\theta}$, given the data $\mathbf{d}$ in Table A.1. The details of how we will generate these fields will be given in Chapter 5. For the constant mean case $\boldsymbol{\theta}_c = (\beta, \omega^2, \lambda)$, for the linear mean case $\boldsymbol{\theta}_l = (\beta, \beta_x, \beta_y, \omega^2, \lambda)$ and for the depth mean case $\boldsymbol{\theta}_d = (\beta, \beta_d, \omega^2, \lambda)$. We wish to find distributions for the values of $\boldsymbol{\theta}$, rather than point estimations. We can then run the computer model with

a sample of different input values of $\boldsymbol{\theta}$ from the derived distributions. For each input, we will generate a large number of realisations of the log transmissivity field. For each realisation, the groundwater flow equations will be solved to give a travel time, t. This way we obtain a distribution of output travel times, t. This will allow us to find out whether any uncertainty in our computer model output is due to the uncertainty we have on the log transmissivity field $Z(\mathbf{x})$, or on the uncertainty in the hyperparameters, $\boldsymbol{\theta}$, which are used to generate realisations of $Z(\mathbf{x})$.

4.1.3 Uncertainty analysis of WIPP computer model

We propose the following method to find the source of uncertainty in the travel time, t. We derive a distribution for $\boldsymbol{\theta}$ given the measured transmissivity data $\mathbf{d}$. A sample is taken of N values of $\boldsymbol{\theta}$. Each of these is used to generate M realisations of $Z(.)$, where $Z(.)$ denotes the log transmissivity field over the entire region Ω, with which to evaluate the model. For each realisation the travel time, t, is calculated. Statistics for the travel time can then be calculated from the sample of travel times.

This method allows us find out where the uncertainty in the computer code is coming from, $\boldsymbol{\theta}$ or $Z(.)$. We require a large number of evaluations ($N \times M$) of the computer code using this method and so it is likely to be very computationally expensive to evaluate $\widehat{\mathrm{E}[t|\boldsymbol{\theta}]}$ or $\widehat{\mathrm{Var}[t|\boldsymbol{\theta}]}$, for example. Therefore we would need to create an emulator if we wanted to estimate $\mathrm{E}[t|\boldsymbol{\theta}]$ for many different values of $\boldsymbol{\theta}$. We will emulate $\widehat{\mathrm{E}[\log t|\boldsymbol{\theta}]}$ as we want to ensure the emulator only generates positive values of t. Since our WIPP groundwater flow model is stochastic, the emulator will be built using $\boldsymbol{\theta}_i$ as our n input design points, and $\widehat{\mathrm{E}[\log t|\boldsymbol{\theta}_i]}$ as our output at those design points. The emulator would be cheaper as it will only require $n \times M$ evaluations of the computer code.

Next we present the basic idea for how the emulator will be built, as a motivation to why we need to derive distributions for $\boldsymbol{\theta}$. A fuller description of the method will be given in Chapter 6.

1. Derive a distribution for $\boldsymbol{\theta}$ given data $\mathbf{d}$ using WinBUGs to generate output from a Markov chain Monte Carlo algorithm. After the chain has converged, a sample from the posterior distribution can be considered as a sample of the distribution for $\boldsymbol{\theta}$ (see Section 2.1.1).

2. Use the derived distribution as a guide to take a latin hypercube sample of θ, $\boldsymbol{\theta}_i, i = 1 \ldots, n..$ The method for doing this will be described in Chapter 6. This will ensure that the samples are distributed across the range of possible values of $\boldsymbol{\theta}_i$. For each sample $\boldsymbol{\theta}_i$:

   (a) Find the eigenvalues and eigenvectors of the truncated Karhunen-Loéve (KL) expansion.

   (b) Sample $\boldsymbol{\xi}_j, j = 1, \ldots, M$ from $\mathcal{N}_M(\mathbf{0}, \mathrm{I})$. For each sample ξ_j:

      i. Generate a realisation of the log transmissivity field $Z_{i,j}(.)|d$, where d is the transmissivity data, using $\boldsymbol{\xi}_j, \boldsymbol{\theta}_i$. The eigenvectors and eigenvalues calculated in step 2a are also required to generate this field.

      ii. Evaluate $\log t_{i,j} = \eta\left(Z_{i,j}(.)\right)$, $j = 1, \ldots, M$, $i = 1, \ldots, N$ (where η is the computer model).

   (c) Calculate sample mean and variance for $\log t|\boldsymbol{\theta}_i$, $i = 1, \ldots, n$, using

$$\begin{aligned} \widehat{\mathrm{E}[\log t|\boldsymbol{\theta}_i]} &= \frac{1}{M}\sum_{j=1}^{M} \log t_{i,j}, \quad i = 1, \ldots, n, \\ \widehat{\mathrm{Var}[\log t|\boldsymbol{\theta}_i]} &= \frac{1}{M-1}\sum_{j=1}^{M} \left\{\log t_{i,j} - \widehat{\mathrm{E}[\log t|\boldsymbol{\theta}_i]}\right\}^2, \quad i = 1, \ldots, n. \end{aligned}$$

3. Build an emulator using $\boldsymbol{\theta}_i$ as the input points, and the sample mean $\widehat{\mathrm{E}[\log t|\boldsymbol{\theta}_i]}$, and variance of the sample mean, $\widehat{\mathrm{Var}[\mathrm{E}[\log t|\boldsymbol{\theta}_i]]} = \frac{\widehat{\mathrm{Var}[\log t|\boldsymbol{\theta}_i]}}{M}$, as the outputs.

A flow chart to illustrate this method will be given in Chapter 6, when the mean of the computer model output will be emulated. The emulator can then be used for further analyses. We generate a sample of $\boldsymbol{\theta}$ from the derived distribution with which to evaluate the emulator. The emulator will then provide outputs $\log t$. These outputs are then used to calculate statistics for the uncertainty in the WIPP model.

The following section introduces the Bayesian methodology we will use to derive posterior distributions for the hyperparameters $\boldsymbol{\theta}$. The difficulty in doing this is to choose prior distributions for the hyperparameters when there is limited, if any, prior information about each hyperparameter. We use the BUGS (Bayesian inference Using Gibbs Sampling) software to perform Bayesian analysis of our stochastic model using MCMC methods. We will derive distributions for $\boldsymbol{\theta}$ for each of our three stochastic models for the log transmissivity field; constant mean $\boldsymbol{\theta}_c$, linear mean $\boldsymbol{\theta}_l$ and depth mean $\boldsymbol{\theta}_d$.

4.2 Bayesian inference using Monte Carlo methods

A Bayesian approach will be used to find distributions for $\boldsymbol{\theta}$. For density functions, Bayes theorem can be written in the form

$$f(\boldsymbol{\theta}|\mathbf{d}) = \frac{f(\boldsymbol{\theta})f(\mathbf{d}|\boldsymbol{\theta})}{f(\mathbf{d})}, \tag{4.2.1}$$

where $f(\boldsymbol{\theta}|\mathbf{d})$ is the posterior distribution of $\boldsymbol{\theta}$, conditional on the observed data, $f(\boldsymbol{\theta})$ is the prior distribution of $\boldsymbol{\theta}$, $f(\mathbf{d}|\boldsymbol{\theta})$ is the likelihood function and $f(\mathbf{d})$ is a normalising constant. If we have observed data $\mathbf{d}$, we can use this equation to derive a density function for an unknown $\boldsymbol{\theta}$.

However, if we cannot obtain the terms in equation (4.2.1) then we cannot find a posterior distribution in this way. In this case we can use MCMC to produce a sample $\boldsymbol{\theta}_j,\ j = 1, \ldots, N$ from the posterior distribution $f(\boldsymbol{\theta}|\mathbf{d})$. If the sample is large enough, we can use this sample to give a posterior summary of $\boldsymbol{\theta}$. MCMC sampling requires us to give initial values of $\boldsymbol{\theta}$ for the chain, and to choose prior distributions for the hyperparameters. After the Markov chain has converged, any sample of $\boldsymbol{\theta}$ will be from the posterior distribution $f(\boldsymbol{\theta}|\mathbf{d})$ regardless of the chosen prior distribution or initial values.

WinBUGS (Lunn et al. (2000)) is used to obtain a MCMC sample of the posterior distribution. WinBUGS contains a powered exponential spatial correlation function, which fits a Gaussian kriging model to the data. This model has mean β and a covariance function of the form:

$$\mathrm{Cov}[Z(\mathbf{x}), Z(\mathbf{x}')] = \frac{1}{\tau}\exp\left[-\phi \parallel \mathbf{x} - \mathbf{x}' \parallel^{\kappa}\right], \tag{4.2.2}$$

where $\tau = \frac{1}{\omega^2}$ controls the overall precision, and $\phi = \frac{1}{\lambda}$ controls the rate of decline of correlation over distance. Note that equation (4.2.2) is the same as equation (4.1.9) with correlation function (4.1.10).

4.2.1 Convergence properties of the chain

To obtain a sample from the posterior distribution for $\boldsymbol{\theta}$, we need to know when the chain has converged. A sample from the distribution taken after this point is from the approximate posterior distribution for $\boldsymbol{\theta}$. By running several chains with widely dispersed starting values we can check that all the chains convergence to the same

distribution. By running just one chain, it may seem that it has converged, but it may be stuck in a local mode.

We can also check the convergence of the chain using the Bayesian output analysis (boa) package (Smith (2005)) in R. This package allows us to carry out convergence diagnostics on the output from the MCMC chain. The boa package contains four commonly used convergence diagnostics for MCMC output: Brooks, Gelman and Rubin; Geweke; Heidelberger and Welch; and Raftery and Lewis (see Smith (2007) for more information on all of these diagnostics). These diagnostic tests do not provide proof of convergence, but provide evidence of non-convergence to a stationary distribution. Therefore it is sensible to use more than one diagnostic for the MCMC output. If there is no evidence of non-convergence in more than one of the diagnostics, then we can assume that the MCMC chain has converged to a stationary distribution. This stationary distribution can be used to provide samples from the joint posterior distribution.

For the WIPP data, we use the Geweke, and Heidelberger and Welch diagnostics as these can be used to analyse single chains of more than one parameter.

4.2.2 Geweke convergence diagnostic

The Geweke diagnostic (Geweke et al. (1992)) is a measure of how well the Markov chain has converged to its stationary distribution. The basic idea is to compare the sample mean $\bar{x}_1$ of n_1 samples from an early part of the chain to the sample mean $\bar{x}_2$ of n_2 samples in a later part of the chain. The two parts of the chain that are sampled from must be considered independent for the diagnostic to be valid. Geweke suggests that the comparison of means should be between the first $n_1 = 0.1n$ and last $n_2 = 0.5n$ samples in the chain, after it is thought convergence has occurred, and this choice of samples is taken here. The diagnostic is based on the statistic

$$z = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{\hat{S}(0)}{n_1} + \frac{\hat{S}(0)}{n_2}}},$$

where $\hat{S}(0)$ is a variance estimate calculated as the spectral density at frequency zero. As the number of iterations in the chain approaches infinity, the z statistic approaches a Normal distribution with zero mean and unit variance if the chain has converged. If the z-value falls in the extreme tails of the $\mathcal{N}(0, 1)$ distribution, then this suggests that

the earlier part of the chain had not fully converged. Otherwise it can be said that there is no evidence that the chain has not converged.

4.2.3 Heidelberger and Welch convergence diagnostic

The Heidelberger and Welch diagnostic (Heidelberger and Welch (1983)) tests that the chain has reached a stationary distribution. As well as a test for non-convergence, the diagnostic gives an estimate of the number of samples which should be discarded as a burn in sequence. The diagnostic is based on Brownian bridge theory and uses the Cramér-von-Mises test statistic

$$\int_0^1 B_n(t)^2 dt,$$

where

$$\begin{aligned} B_n(t) &= \frac{T_{\lfloor nt \rfloor} - \lfloor nt \rfloor \bar{x}}{\sqrt{nS(0)}} \\ T_k &= \begin{cases} 0, & k = 0 \\ \sum_{j=1}^{k} x_j, & k \geq 1 \end{cases}, \end{aligned} \tag{4.2.3}$$

and $S(0)$ is the spectral density, estimated from the second half of the chain, evaluated at frequency zero. If there is evidence of non-stationarity, the first 10% of iterations are discarded and the test repeated with the remainder of the chain. This process is continued until the rest of the chain passes the test, or until more than 50% of the iterations have been discarded. If the chain fails the test, this indicates that a longer run of the chain is needed to achieve convergence.

For the part of the chain which has passed the test, a halfwidth test is performed to determine the accuracy of the posterior mean. The mean of the partial chain and its associated confidence interval are calculated. The partial chain passes the test, and therefore the posterior mean is estimated with acceptable accuracy, if the halfwidth of the confidence interval for the mean is below a specified accuracy level. For this work, we have chosen the accuracy to be 0.1. If the halfwidth test is not passed, it suggests that a longer run of the MCMC sampler is needed to increase the accuracy of the posterior mean.

4.3 Bayesian inference for the hyperparameters $\boldsymbol{\theta}_c$ of the log transmissivity field with constant mean

Using the Bayesian approach outlined above, we now derive distributions for the hyperparameters, $\boldsymbol{\theta}_c = (\beta, \omega^2, \lambda)$, of the log transmissivity field with a constant mean.

Various prior distributions are investigated for each hyperparameter to give posterior distributions for $\boldsymbol{\theta}_c$. The convergence properties of each MCMC chain are also considered. Then cross validation of the log transmissivity data given the posterior distributions is carried out. We also consider how $\boldsymbol{\theta}_c$ affects the log transmissivity field $Z(\mathbf{x})$. Finally we investigate whether changing κ has an effect on the posterior distribution of $\boldsymbol{\theta}_c$.

4.3.1 Initial values of $\boldsymbol{\theta}_c$

Initial values of each hyperparameter in $\boldsymbol{\theta}_c$ are needed to start the MCMC chain in WinBUGS. We investigate three chains for the hyperparameters. The first chain uses initial values according to the data, and the other two chains start at values either side of these values for β and λ, and larger values for ω^2. If the chains converge to the same posterior distribution, despite having very different initial values, then the posterior distribution after convergence is independent of the initial values of the chain. The initial values for each of the three chains are as follows:

Chain 1: $\beta = -5.5997$, the sample mean of the WIPP data; $\omega^2 = 2.298$, the sample variance of the WIPP data; $\lambda = 10000$, approximately half of the total modelled distance in the x-direction and a third of the total modelled distance in the y-direction.

Chain 2: $\beta = -100$; $\omega^2 = 100$; $\lambda = 100$, approximately the minimum distance between two boreholes (data points).

Chain 3: $\beta = 100$; $\omega^2 = 1000$; $\lambda = 30000$, approximately the maximum distance between two boreholes (data points).

4.3.2 Choice of prior distributions for $\boldsymbol{\theta}_c$

Suitable choices for prior distributions for the hyperparameters $\boldsymbol{\theta}_c$ need to be made before starting the MCMC sampling in WinBUGS. These priors are updated using the data in Table A.1 to give posterior distributions for $\boldsymbol{\theta}_c$. Various choices of prior distribution were investigated and the effects of these choices on the posterior distribution after convergence of the MCMC sample are discussed below. To investigate the effect of each prior distribution on the posterior distribution for each hyperparameter, we fix the prior distributions for the other two hyperparameters. We use the following relatively uninformative prior distributions: Improper prior for β, Uniform prior $U[0, 100]$ for ω^2 and Uniform prior $U[0, 40000]$ for λ. Therefore, whilst investigating what effect changing the prior distribution for β has, we fix the prior distributions for ω^2 and λ at the prior distributions mentioned above.

The distributions given for $\boldsymbol{\theta}_c$ in the next few sections are the distributions after convergence as described in Section 4.2.1.

4.3.3 Prior distribution for β

As there is no prior information about the value of β we start by trying an improper prior distribution for β. This represents a prior distribution where any value is equally likely. This may not seem sensible to experts who have knowledge about the range of values that the mean of a log transmissivity field has. However, in the absence of expert information, this prior is a sensible choice. Whilst a large variance, of say 10^{100}, does seem improbable, this prior distribution is then updated using the available data. Therefore the posterior distribution, which is the quantity we are interested in, may not have as large a variance.

Another possible prior distribution for β is a normal distribution with zero mean and large variance. This large variance reflects our uncertainty about β. The results from using improper and normal priors are shown in Table 4.1. As the variance increases in the normal prior, the posterior distribution tends to that obtained when using an improper prior distribution for β. This is what we would expect as the prior distributions become similar. The improper distribution is a sensible prior to use for β as a normal prior with large variance did not change the posterior distribution much. The results from this

Prior distribution for β	Posterior distribution for β				
	mean	s.d.	2.5 %	median	97.5 %
Improper	-4.934	1.645	-8.318	-5.012	-1.27
$\mathcal{N}(0, 10^3)$	-4.927	1.645	-8.242	-5.009	-1.281
$\mathcal{N}(0, 10^6)$	-4.934	1.645	-8.318	-5.012	-1.27
$\mathcal{N}(0, 10^{12})$	-4.934	1.645	-8.318	-5.012	-1.27

Table 4.1: Effects of prior distributions for β on the posterior distribution of β.

prior will be used for further analysis. The first 50 MCMC iterations for the posterior distribution given an improper prior are shown in the trace in Figure 4.2. We see that the three chains converge very quickly to the same distribution.

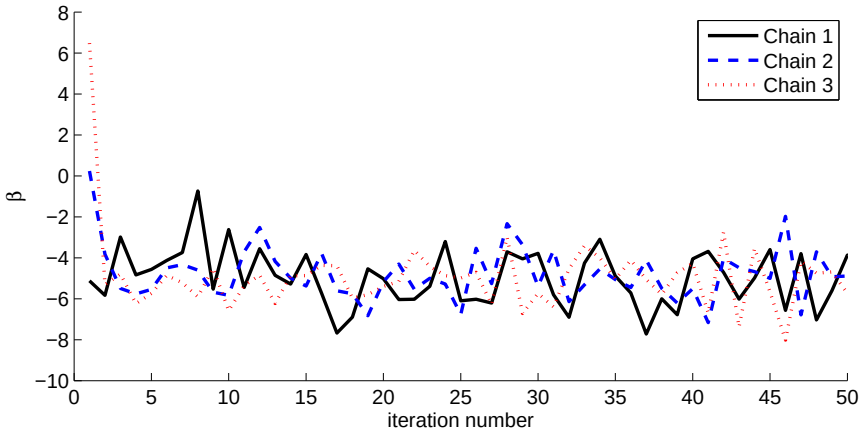

Figure 4.2: MCMC traces for all three β chains.

The results of the Geweke convergence diagnostic do not provide any evidence against convergence of these chains. The chains also pass the Heidelberger and Welch stationarity and halfwidth tests. Figure 4.3 shows the posterior density of β given an improper prior using samples from the posterior distribution.

4.3.4 Prior distribution for ω^2

As ω^2 represents the variance of $Z(\mathbf{x})$, the only prior information we have about ω^2 is that it is non-negative. Therefore, we will investigate a uniform prior with lower bound 0 for ω, and inverse gamma and lognormal priors for ω^2. The uniform prior

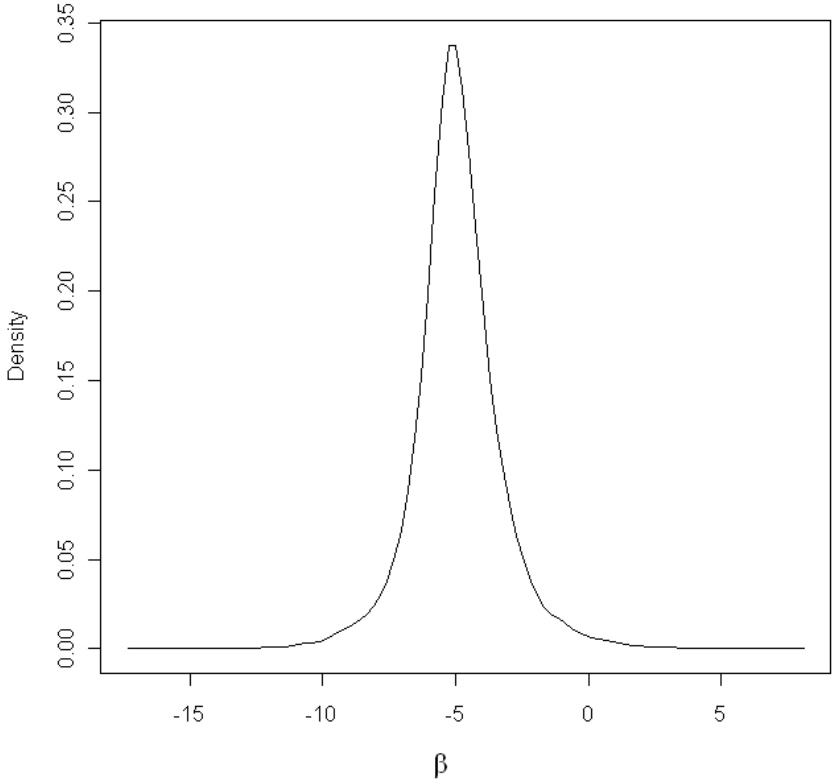

Figure 4.3: Posterior density obtained when using improper prior for β.

provides a fairly uninformative prior given the lack of prior information about ω. The inverse gamma and lognormal priors allow us to investigate the effect of the shape of the prior distribution on the posterior distribution. As discussed before, we fix the prior distributions of the other two hyperparameters when investigating the effect of changing the prior distribution for ω^2.

We first investigate a uniform distribution for ω. From Table 4.2, we can see that as the uniform prior distribution for ω widens, the posterior distribution for ω^2 becomes less sensitive to changes in the prior. The posterior distribution will not allow values outside of the uniform prior distribution. Therefore the range of the first two prior distributions may not be wide enough to include all possible values of ω. The mean and standard deviation of the posterior distribution increases as the prior range increases.

Prior distribution for ω	Posterior distribution for ω^2				
	mean	s.d.	2.5 %	median	97.5 %
$\mathcal{U}(0,2)$	3.123	0.5666	1.928	3.19	3.953
$\mathcal{U}(0,5)$	7.329	4.442	2.311	5.984	18.95
$\mathcal{U}(0,10)$	7.568	4.875	2.322	5.927	20.27
$\mathcal{U}(0,100)$	7.568	4.875	2.322	5.927	20.27

Table 4.2: Effects of uniform prior distributions for ω on the posterior distribution of ω^2.

The posterior distributions obtained when using various inverse gamma prior distributions for ω^2 are shown in Table 4.3. A large scale second parameter has been used for all inverse gamma distributions to reflect our uncertainty about ω^2. We can see that as the inverse gamma prior distribution becomes flatter as the shape first parameter decreases the posterior distribution mean and standard deviation increases. This is expected as the mean and variance of the prior distributions also increase as the shape parameter decreases. The sensitivity of the posterior distribution to the prior distribution reduces as the prior becomes flatter.

Prior distribution for ω^2	Posterior distribution for ω^2				
	mean	s.d.	2.5 %	median	97.5 %
$\text{inv}\mathcal{G}(0.0001, 10^3)$	6.531	4.311	2.124	5.126	18.43
$\text{inv}\mathcal{G}(0.001, 10^3)$	6.479	4.284	2.111	5.076	18.23
$\text{inv}\mathcal{G}(0.5, 10^3)$	5.363	3.395	1.966	4.315	15.15
$\text{inv}\mathcal{G}(1.0, 10^3)$	4.736	2.984	1.875	3.837	13.5
$\text{inv}\mathcal{G}(5.0, 10^3)$	2.483	0.8516	1.414	2.305	4.579
$\text{inv}\mathcal{G}(10.0, 10^3)$	1.747	0.409	1.123	1.687	2.702

Table 4.3: Effects of inverse gamma prior distributions for ω^2 on the posterior distribution of ω^2.

In the case of a lognormal prior distribution, as the variance of the prior distribution increases, the less sensitive the posterior distribution becomes to these changes (Table 4.4). Using a lognormal prior distribution with large variance gives similar results to using an inverse gamma prior with small shape parameter, as they both give very flat priors for ω^2.

The flatter inverse gamma prior distribution for ω^2, $\text{inv}\mathcal{G}(0.001, 10^3)$, gives a posterior

Prior distribution for ω^2	Posterior distribution for ω^2				
	mean	s.d.	2.5 %	median	97.5 %
$\log\mathcal{N}(0, 10^2)$	6.19	3.903	2.13	4.976	17.1
$\log\mathcal{N}(0, 10^3)$	6.199	4.087	2.121	4.883	17.48
$\log\mathcal{N}(0, 10^6)$	6.162	4.004	2.142	4.812	17.23
$\log\mathcal{N}(0, 10^{12})$	6.162	4.004	2.142	4.812	17.23

Table 4.4: Effects of lognormal prior distributions for ω^2 on the posterior distribution of ω^2.

distribution for ω^2 that is close to the posterior distributions given by a lognormal prior with large variance for ω^2 and a uniform prior with small range for ω. However using a small range limits the posterior distribution of ω^2. Figure 4.4 compares the posterior densities obtained when using each of the following prior distributions: $\text{inv}\mathcal{G}(0.001, 10^3)$ and $\log\mathcal{N}(0, 10^6)$ for ω^2 and $\omega \sim \mathcal{U}(0, 100)$.

We can see that the posterior densities for inverse gamma and lognormal priors are very similar. The uniform prior gives a slightly flatter posterior with a slightly larger variance. However, most of this posterior distribution is positioned in the same area as the other two posterior distributions. Therefore, we will use the posterior distribution obtained from the prior $\text{inv}\mathcal{G}(0.001, 10^3)$ for ω^2 for further analysis. The MCMC traces for this distribution are shown in Figure 4.5. Again we can see that the three chains converge to the same distribution. The results of the Geweke convergence diagnostic do not provide any evidence against convergence of these chains and the Heidelberger and Welch stationarity and halfwidth tests are passed. The posterior density for ω^2 is shown in Figure 4.6.

4.3.5 Prior distribution for λ

The value of the correlation length λ determines the distance at which the correlations of the random field $Z(\mathbf{x})$ are nearly zero. There is often little information in the data about the value of λ. Therefore, a reasonably informative prior should be used for λ, or the value fixed a priori based on expert knowledge or variogram analysis.

We want to find a distribution for λ rather than a fixed value. Therefore we choose a prior distribution for λ which gives a wide, but sensible range of correlations between the

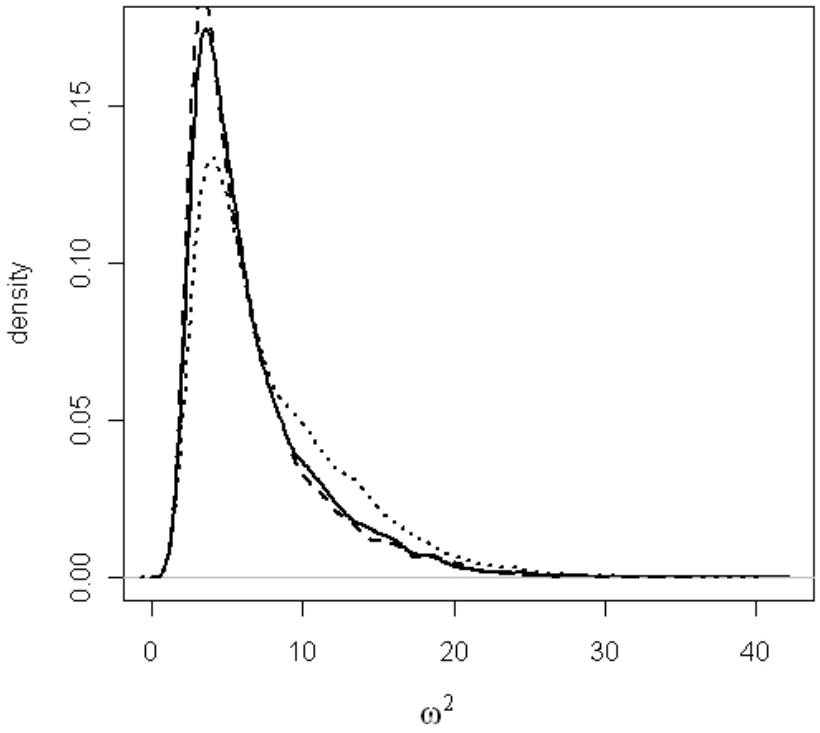

Figure 4.4: Posterior densities obtained when using inverse gamma: $\text{inv}\mathcal{G}(0.001, 10^3)$ (solid), lognormal: $\log\mathcal{N}(0, 10^6)$ (dashed) priors for ω^2, and uniform: $\mathcal{U}(0, 100)$ (dotted) prior for ω.

maximum and minimum distances between two boreholes in the region. The correlation between any two points in the region must lie in the interval [0,1], where 0 is the minimum correlation and 1 is the maximum correlation. These correlations are absolute values and so we do not consider negative correlations here as we are not concerned with the direction of the correlation.

The minimum distance between two boreholes in the region is 129m, between boreholes WIPP-19 and WIPP-22, and the maximum distance is 27731m between boreholes H-10 and WIPP-27. We consider that boreholes closer together will be more highly correlated than those further apart. The correlation between two points further apart may also be high since we are considering points lying in the same type of rock. However as discussed

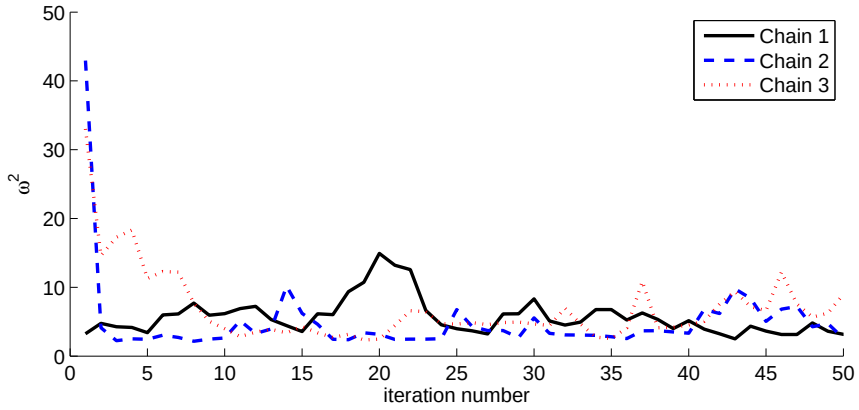

Figure 4.5: MCMC trace for all three ω^2 chains.

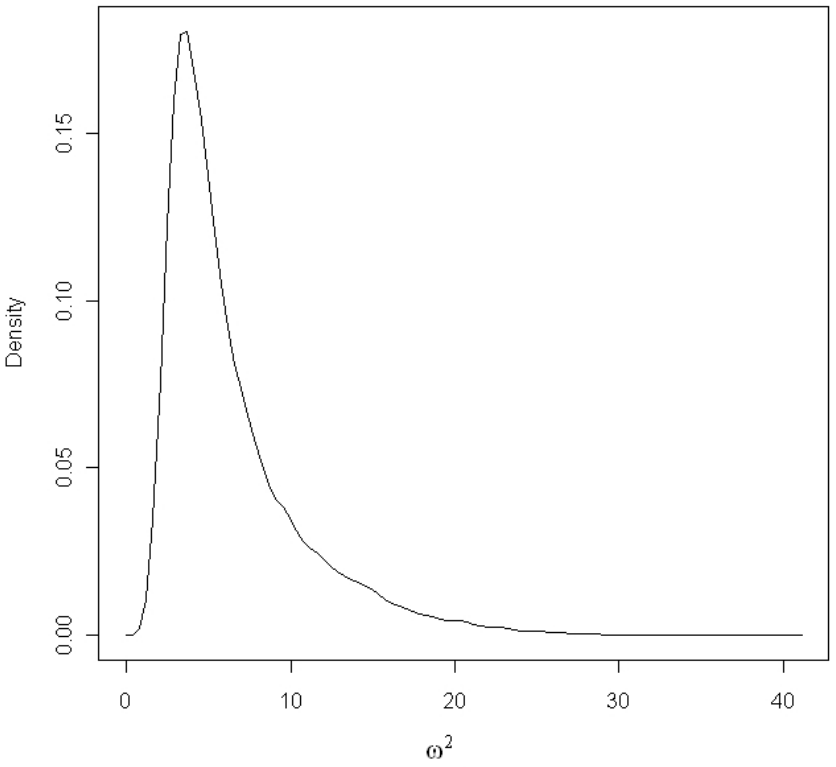

Figure 4.6: Posterior density for ω^2 given an inverse Gamma prior.

in Thomas et al. (2004), for modelling purposes, the correlation at the maximum distance in the region must not be too high as this may lead to identifiability problems between the overall mean of the spatial random variable $Z(\mathbf{x})$ and the correlation parameter $\phi = \frac{1}{\lambda}$.

We investigate a uniform prior with various minimum and maximum values for λ. We first choose $\lambda_{min} = 50$ and $\lambda_{max} = 20000$. At the minimum distance of 129m between boreholes, the correlations are between 0.076 and 0.994, and at the maximum distance of 27731m between boreholes, the correlations are between 0 and 0.250. We then consider extending the range of λ to increase the range of correlations at the maximum distance. The results are shown in Table 4.5.

Prior distribution	Posterior distribution for λ				
for λ	mean	s.d.	2.5 %	median	97.5 %
$\mathcal{U}(50, 20000)$	9280	4367	2972	8397	18870
$\mathcal{U}(50, 30000)$	10920	6216	3248	9231	27050
$\mathcal{U}(50, 40000)$	12390	8380	3098	9635	35340
$\mathcal{U}(30, 40000)$	12470	8380	3126	9736	35330
$\mathcal{U}(10, 40000)$	12630	8287	3193	10030	34900

Table 4.5: Effects of prior distributions for λ on the posterior distribution of λ.

The posterior distributions of λ are all skewed towards the lower end of values. We can also see that the posterior distribution of λ becomes less sensitive to changes in the prior distribution as the range of the uniform distribution increases. As the lower limit for λ is decreased, there is only a small amount of change in the posterior distribution of λ. A correlation length of 10 gives a very small correlation of 2.5×10^{-6} at the minimum distance of 129m between two boreholes. This seems too small given our assumption that boreholes closer together will have higher correlations.

Figure 4.7 shows that the three MCMC chains convergence to the same posterior distribution. The results of the Geweke convergence diagnostic do not provide any evidence against convergence of the chains. The chains also pass the Heidelberger and Welch stationarity and halfwidth tests.

From Figure 4.8 we can see that because of the upper limit for λ in the prior, the posterior density is cut off at 40000. The values of the posterior distribution do not tail off to zero at this upper limit, suggesting that the values of λ could be larger than

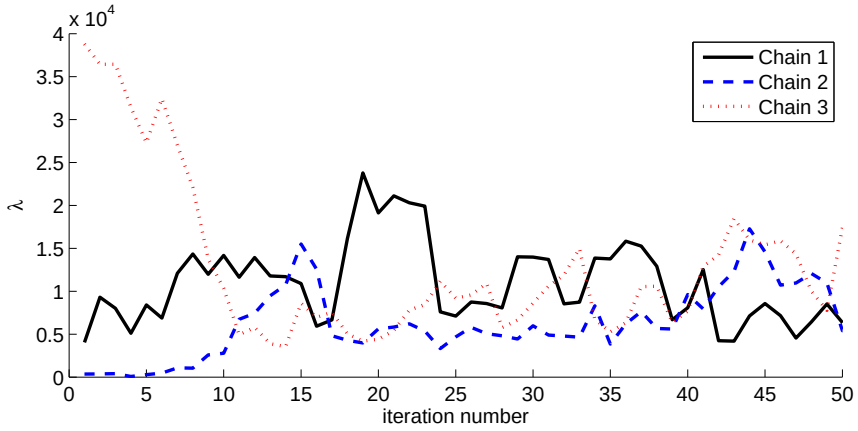

Figure 4.7: MCMC trace for all three λ chains.

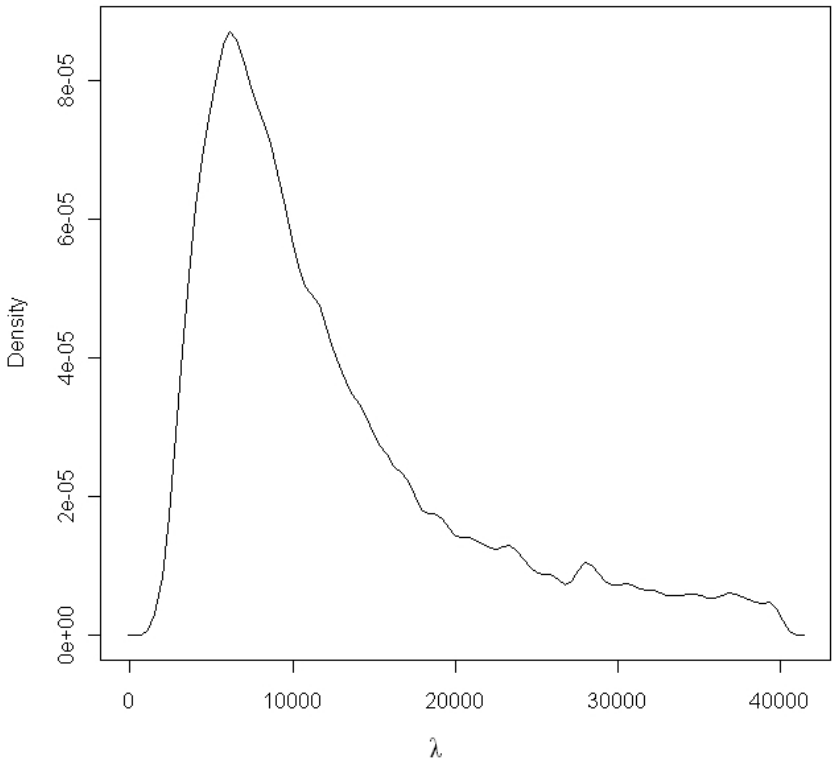

Figure 4.8: Posterior density for λ given a uniform prior.

40000. However, we do not want the upper limit for λ to be much larger than the spatial region of interest due to the identifiability problem mentioned earlier. At the lower limit for λ the posterior distribution tails off to zero fairly quickly. Therefore we choose the uniform prior with range between 50 and 40000 for further analysis.

4.3.6 Correlation between ω^2 and λ

The MCMC output for ω^2 and λ suggest that there may be a correlation between ω^2 and λ. Figure 4.9 plots the values of ω^2 against λ for 10000 iterations after convergence of the MCMC output. We see that there is a slight correlation between ω^2 and λ.

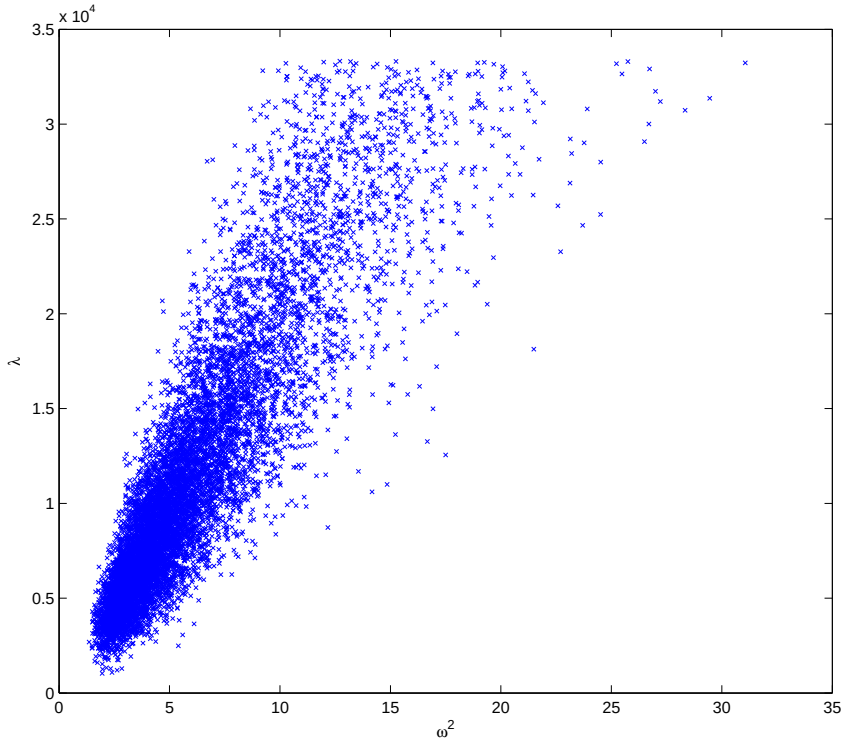

Figure 4.9: Correlation between ω^2 and λ.

Larger values of ω^2 correspond to larger values of λ, and smaller values of ω^2 to smaller values of λ. This correlation could be used when sampling from the sample space of $\boldsymbol{\theta}_c$ to reduce the space sampled from. The region of the sample space of ω^2 and λ which

contains no values from the posterior distribution, for example where ω^2 is large and λ is small, could be excluded from the sample space.

The correlation between ω^2 and λ has been documented in a previous study of transmissivity fields by Hoeksema and Kitanidis (1985). They found that, as well as the parameter estimates for ω^2 and λ being highly correlated, their algorithm for estimating the parameters often did not converge. They ascribe this non-convergence to large sampling error associated with estimating λ combined with nonlinear dependence of the covariance on λ. The lack of identifiability between covariances of the log transmissivity field is described in Hoeksema and Kitandis and by Stein (1999) and can be explained, as follows, by exploring the variogram. The variogram is a function that describes how spatially dependent a spatial random field, or spatial random process, is.

The equation for the variogram for our exponential model is given by

$$2\gamma(r_m) = 2\omega^2 \left[1 - \exp\left(-\frac{r_m}{\lambda}\right)\right].$$

Expanding this equation using Taylor series for the exponential, we can see that if $\lambda \gg r_m$, where r_m is the distance between two measurement points, then

$$2\gamma(r_m) \simeq 2\omega^2 \frac{r_m}{\lambda}.$$

The variogram is approximately linear in $\frac{\omega^2}{\lambda}$, and so estimating these parameters individually becomes very difficult. It is also impossible to differentiate between two transmissivity fields when $\frac{\omega^2}{\lambda} = \text{const}$, $\lambda \gg r_m$. For example, $\omega^2 = 5$ and $\lambda = 20000$ will provide similar transmissivity fields as $\omega^2 = 10$ and $\lambda = 40000$. Even though the two correlation functions are different, their variograms are approximately the same when $\lambda \gg r_m$ as we can see in Figure 4.10.

Hoeksema and Kitandis found that this problem could be greatly reduced by placing very strict bounds on the allowable value of λ. They discovered in practice, that the parameters are individually identifiable only if λ is between narrow bounds. They suggest that the upper bound on λ should be equal to the maximum separation distance between two measurement points.

Stein investigates the identifiability problem by comparing measures based on the mean squared error (mse) of the best linear predictor (BLP). He provides the following measure of how well predictions based on a covariance function C_1 do when the correct covariance

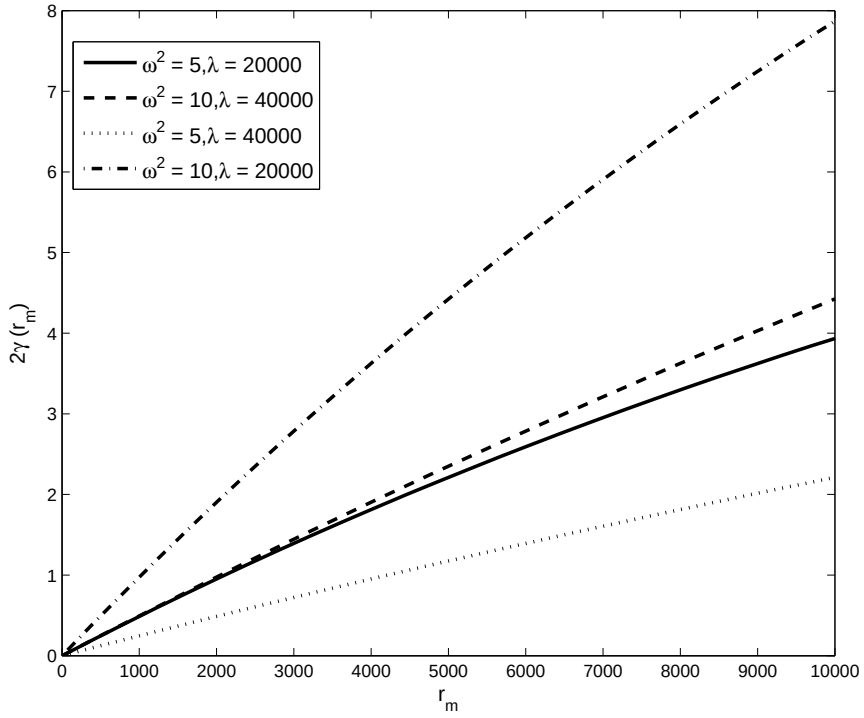

Figure 4.10: Variograms for different values of λ and ω^2. r_m is the distance between two points in the region.

function is C_0:

$$\frac{E_0 e_1^2}{E_0 e_0^2} = \frac{\text{mse of suboptimal pseudo BLP}}{\text{mse of optimal BLP}}, \tag{4.3.1}$$

where e_j is the error of the best linear predictor if the correct mean is m_j and the correct covariance is C_j, and E_j is the expected value under the correct second order structure (m_j, C_j). He also provides a measure of the accuracy of assessing the mse of the psuedo BLP:

$$\frac{E_1 e_1^2}{E_0 e_1^2} = \frac{\text{presumed mse of pseudo BLP}}{\text{actual mse of psuedo BLP}}. \tag{4.3.2}$$

Stein states that if both of these measures are close to 1, then as far as predicting $Z(x)$, little is lost by using the suboptimal covariance C_1 instead of the correct C_0. Therefore we can use a covariance function, even if it is not optimal, to predict the log transmissivity field $Z(\mathbf{x})$.

4.3.7 Cross validation

We now check that the posterior distributions for $\boldsymbol{\theta}_c$ give sensible realisations of the log transmissivity field. The posterior distributions chosen for further analysis are summarised in Table 4.6. The posterior distributions for $\boldsymbol{\theta}_c$ are used to estimate each ob-

Hyperparameter	Posterior distribution				
	mean	s.d.	2.5 %	median	97.5 %
β	-4.934	1.645	-8.318	-5.012	-1.27
ω^2	6.479	4.284	2.111	5.076	18.23
λ	12390	8380	3098	9635	35340

Table 4.6: Posterior distributions chosen for further analysis.

served value $Z(\mathbf{x}_i)$, $i = 1, \ldots, 39$ using $\mathbf{d}_{-i}$, the data $\mathbf{d}$ with one observation, $d_i = Z(\mathbf{x}_i)$, omitted.

The prior representation of $Z(\mathbf{x})$ as a Gaussian random field (equations (4.1.6) and (4.1.9)) is updated using the property of multivariate distributions given in Section 2.3.2 to give

$$Z(.)|\boldsymbol{\theta}_c, \mathbf{d} \sim \mathcal{N}\left(m^*(.), \omega^2 c^*(.,.)\right), \tag{4.3.3}$$

where

$$m^*(\mathbf{x}) = \beta + \mathbf{t}(\mathbf{x})^T A^{-1}(\mathbf{d}_{-i} - \boldsymbol{\beta}^{(n-1)}), \tag{4.3.4}$$

$$c^*(\mathbf{x}, \mathbf{x}') = c(\mathbf{x}, \mathbf{x}') - \mathbf{t}(\mathbf{x})^T A^{-1} \mathbf{t}(\mathbf{x}), \tag{4.3.5}$$

$$\mathbf{t}(\mathbf{x}) = (c(\mathbf{x}_1, \mathbf{x}), \ldots, c(\mathbf{x}_{n-1}, \mathbf{x})),$$

$$A = \begin{pmatrix} 1 & c(\mathbf{x}_1, \mathbf{x}_2) & \cdots & c(\mathbf{x}_1, \mathbf{x}_{n-1}) \\ c(\mathbf{x}_2, \mathbf{x}_1) & 1 & & \vdots \\ \vdots & & \ddots & \\ c(\mathbf{x}_{n-1}, \mathbf{x}_1) & \cdots & & 1 \end{pmatrix},$$

and $\boldsymbol{\beta}^{(n-1)}$ is a vector of length $n-1$ containing β.

We need to estimate the values of $Z(\mathbf{x})$ conditional on the data $\mathbf{d}_{-i}$ only. We can use the equation:

$$f\left(Z(\mathbf{x})|\mathbf{d}\right) = \int f\left(Z(\mathbf{x})|\mathbf{d}, \boldsymbol{\theta}_c\right) f\left(\boldsymbol{\theta}_c|\mathbf{d}\right) d\boldsymbol{\theta}_c. \tag{4.3.6}$$

This integral cannot be done analytically due to the prior distributions chosen for $\boldsymbol{\theta}_c$. Therefore to estimate $Z(.)$ conditional only on the data, we use the following method to approximate the integral (4.3.6).

1. Sample $\boldsymbol{\theta}_{c_j} = (\beta_j, \omega_j^2, \lambda_j), j = 1, \ldots, N$, from $f\left(\beta, \omega^2, \lambda | \mathbf{d}\right)$.
   (When $\mathbf{d}$ is a large enough sample, $f\left(\beta, \omega^2, \lambda | \mathbf{d}_{-i}\right) \approx f\left(\beta, \omega^2, \lambda | \mathbf{d}\right)$).

2. For each sample, $\boldsymbol{\theta}_{c_j}$ obtain a sample $Z_j(\mathbf{x}_i), i = 1, \ldots, M$ from
   $f\left(Z(\mathbf{x}_i) | \boldsymbol{\theta}_c = \boldsymbol{\theta}_{c_j}, \mathbf{d}\right)$.

3. Calculate the total sample mean and variance using

$$\widehat{\mathrm{E}\left[Z(\mathbf{x}_i)\right]} = \frac{1}{N}\sum_{j=1}^{N} Z_j, \quad \widehat{\mathrm{Var}\left[Z(\mathbf{x}_i)\right]} = \frac{1}{N-1}\sum_{j=1}^{N}\left(Z_j - \widehat{\mathrm{E}\left[Z(\mathbf{x}_i)\right]}\right)^2.$$

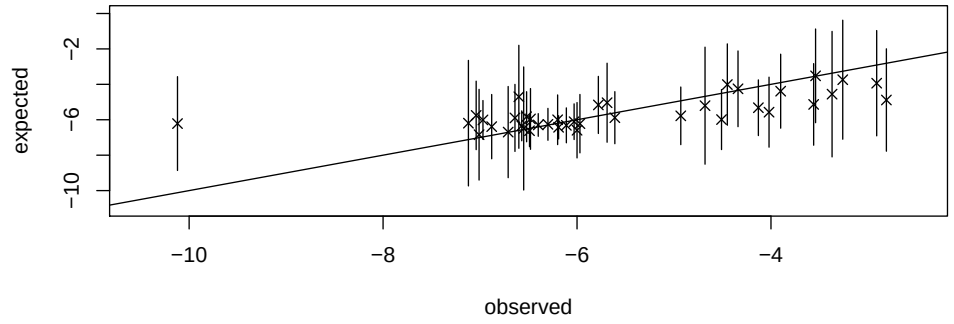

Figure 4.11: Cross validation results for all data points with 95% bounds. Solid line shows expected = observed.

Figure 4.11 shows the results from the cross validation. We can see that all observed values of $Z(\mathbf{x})$, except for at borehole P-18, lie within 95% of the approximated values. If the cross validation is carried out without this point, so that we are only estimating $Z(\mathbf{x})$ at the remaining 38 data points, we get results as shown in Figure 4.12. There is little difference in the expected values between the two graphs. The estimations of the observed variables below -6 when using 38 points were slightly larger than when estimated using all 39 data points, but the bounds still included the observed value.

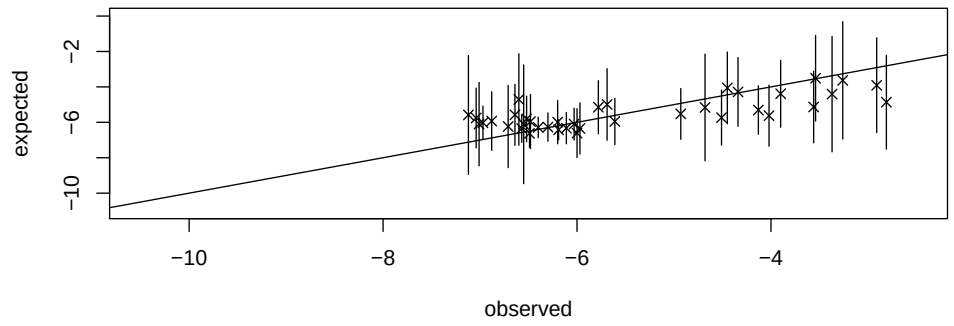

Figure 4.12: Cross validation results for all data points, except borehole P-18, with 95% bounds. Solid line shows expected = observed.

Therefore we will leave the value obtained at borehole P-18 in the data, as does not affect the expected values at other points.

The cross validation shows that the distributions for $\boldsymbol{\theta}_c$ give a good approximation to the log transmissivity field, and can be used to estimate how the uncertainty in $Z(.)$ and $\boldsymbol{\theta}_c$ are propagated through our computer model.

4.4 Effects of $\boldsymbol{\theta}_c$ on the log transmissivity field $Z(\mathbf{x})$

Before using the distribution for $\boldsymbol{\theta}_c$ to approximate the log transmissivity field, we can explore how the field changes in response to changes in $\boldsymbol{\theta}_c$. Choosing the value of the hyperparameter we are interested in, we sample N values of $\boldsymbol{\theta}_c$, $\boldsymbol{\theta}_{c_j}, j = 1, \ldots, N$, from $f(\boldsymbol{\theta}_c|\mathbf{d})$. Then we sample $Z(.)$ from $f(Z(.)|\boldsymbol{\theta}_c = \boldsymbol{\theta}_{c_j}, \mathbf{d})$. Evaluating Z at a point in the region, we are able to see how the distribution of Z changes with changes in the hyperparameter of interest. For this section, boreholes WIPP-28 and H-15 have been arbitrarily chosen as points to evaluate Z. The measured values of log transmissivity at these two boreholes is $z = -4.68$ at WIPP-28 and $z = -6.88$ at H-15.

4.4.1 Changing β

From equation (4.3.4) we can see that β will affect the posterior mean of $Z(.)$. We would expect that a larger β would give a larger mean value, and a smaller β would give a smaller mean value. The posterior distribution for β gained from the MCMC sample gives 2.5% and 97.5% quantiles as -8.3 and -1.3 respectively. Looking at Figure 4.13, setting β to these values results in a change in the mean of $Z(.)$ at both boreholes. A larger value of β gives a larger mean of $Z(.)$. This effect is more pronounced at borehole WIPP-28 than borehole H-15. At both boreholes the posterior distribution for Z(.) contains the observed value.

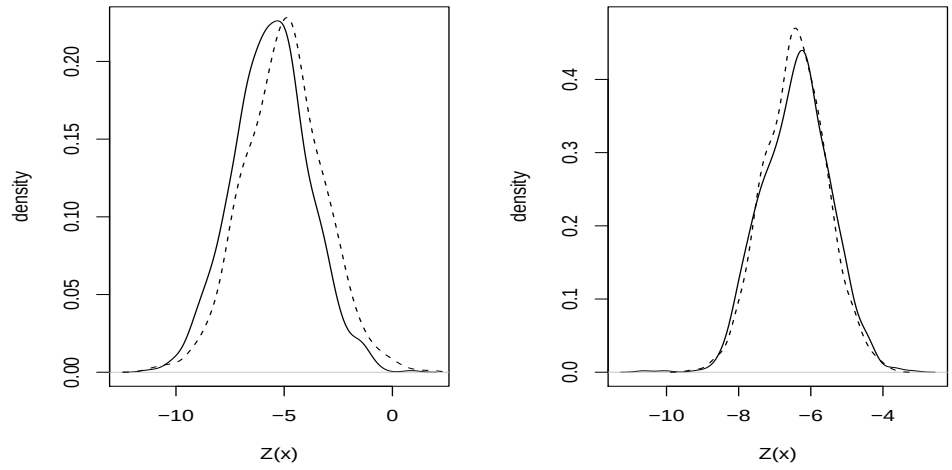

Figure 4.13: Effects on the posterior distribution of Z of changing β: $\beta = -8.3$ (solid line), $\beta = -1.3$ (dashed line). Density plot on the left is for borehole WIPP-28, and on the right for H-15.

4.4.2 Changing ω^2

Changing the value of ω^2 will change the posterior variance of $Z(.)$ (equation (4.3.3)). A larger ω^2 will lead to a larger variance of $Z(.)$. Values of ω^2 were set as 2.1 and 18.2, the 2.5% and 97.5% values respectively from the posterior distribution of ω^2. Figure 4.14 shows that for both boreholes, a larger value of ω^2 gives a larger variance.

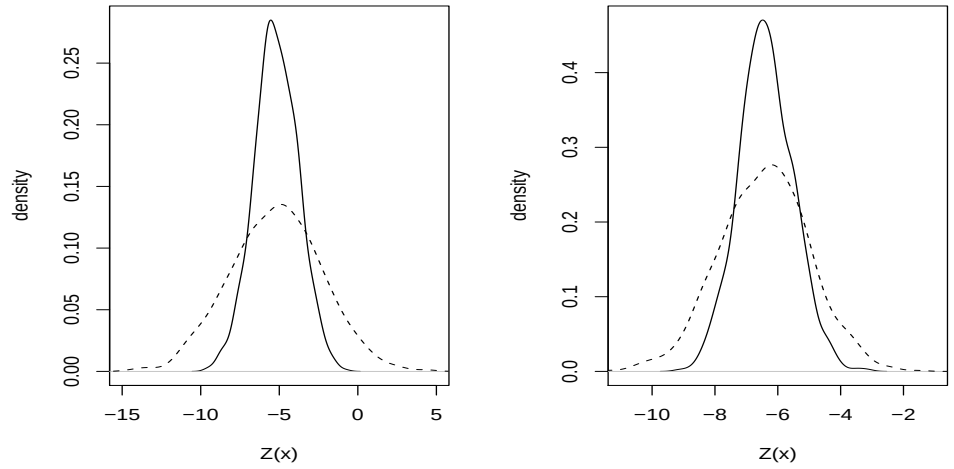

Figure 4.14: Effects on the posterior distribution of Z of changing ω^2: $\omega^2 = 2.1$ (solid line), $\omega^2 = 18.2$ (dashed line). Density plot on the left is for borehole WIPP-28, and on the right for H-15.

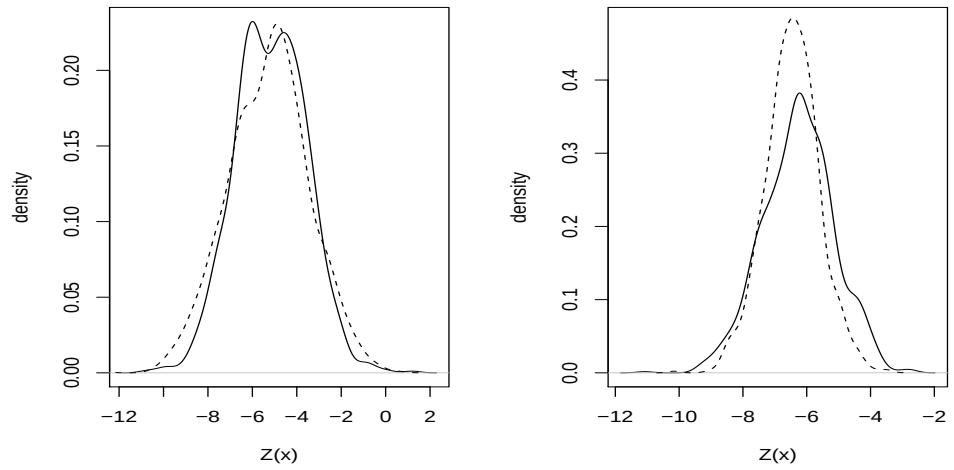

Figure 4.15: Effects on the posterior distribution of Z of changing λ: $\lambda = 3000$ (solid line), $\lambda = 35000$ (dashed line). Density plot on the left is for borehole WIPP-28, and on the right for H-15.

4.4.3 Changing λ

The mean and variance of the posterior distribution of $Z(.)$ will be affected by changing the value of λ. Small values of λ will give estimates of the posterior mean of $Z(.)$ that do not deviate very far from β. This is because if λ is smaller, the second term on the right hand side of equation (4.3.4) will be smaller. Changing λ will also affect the posterior variance of $Z(.)$. Smaller values of λ correspond to lower correlations between two points and therefore the variance of the field will be larger. However for borehole H-15 in Figure 4.15 this is not the case. The larger value of λ gives a slightly narrower distribution for Z.

4.4.4 Effect of varying κ

So far we have been investigating the case where $\kappa = 1$. We now explore how allowing κ to vary affects the log transmissivity field $Z(.)$. Assigning κ a prior uniform distribution between 1 and 2, and using the prior distributions decided upon above for $\boldsymbol{\theta}_c$, we use WinBUGs to obtain posterior distributions for each of the hyperparameters shown in Table 4.7. Comparing these distributions to those in Table 4.6, allowing κ to vary

Hyperparameter	Posterior distributions for $\boldsymbol{\theta}$ and κ				
	mean	s.d.	2.5 %	median	97.5 %
β	-5.08	1.405	-7.551	-5.162	-1.896
ω^2	6.505	7.465	2.08	4.399	25.16
λ	7321	6370	1942	5076	27400
κ	1.3	0.2002	1.016	1.269	1.734

Table 4.7: Effects of varying κ on the posterior distribution of ω^2.

changes the posterior distributions of β, ω^2 and λ slightly. The standard deviation of β has reduced, decreasing the range of the posterior distribution. The distribution for ω^2 is similar to that obtained when setting κ to 1. The introduction of κ as a variable seems to have affected the distribution of λ the most, with the values approximately halved. We also note that the posterior distribution obtained for κ is closer to 1 than 2.

Figure 4.16 shows the effect of varying κ on the posterior distribution of $Z(\mathbf{x})$ at the boreholes WIPP-28 and H-15. We can see the reduced variance of the distribution of

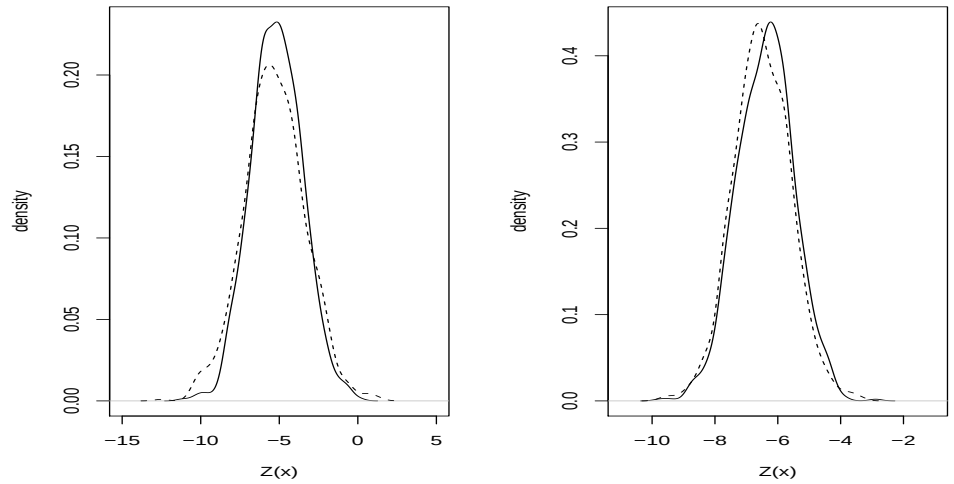

Figure 4.16: Effects on the posterior distribution of Z of allowing κ to vary: $\kappa = 1$ (solid line), $\kappa \sim \mathcal{U}(1, 2)$ (dashed line). Density plot on the left is for borehole WIPP-28, and on the right for H-15.

$Z(.)$ at WIPP-28. This effect is due to the reduction in the distribution of λ. As we noted before, decreasing the value of λ increases the variance of the distribution of $Z(.)$. At H-15, there is not a great difference between the variances of the two distributions, but the mean of $Z(.)$ is smaller when κ is allowed to vary.

Since allowing κ to vary does not change the posterior distribution of $Z(.)$ very much, and the posterior distribution for κ is close to 1 which is the generally accepted value for groundwater flow models (Dagan (1989)), we keep the constant $\kappa = 1$.

4.5 Bayesian inference for the hyperparameters $\boldsymbol{\theta}_l$ of the log transmissivity field with linear mean

So far, we have taken the simple assumption that the log transmissivity field has a constant mean. We would like our stochastic model for the log transmissivity field to be as good as possible, so we now consider using the linear mean,

$$\mathrm{E}[Z(\mathbf{x}))] = \beta + \beta_x x + \beta_y y, \tag{4.1.7}$$

as the mean of the log transmissivity field. We will need to carry out the inference for the distributions again for all hyperparameters $\boldsymbol{\theta}_l = (\beta, \beta_x, \beta_y, \omega^2, \lambda)$. We will see if the use of a linear mean changes the distributions of the original three hyperparameters. We expect that the linear mean will provide a better fit to the data and so the variance, ω^2, will be smaller for the linear model. If we see the same relationship between variance and correlation length as for the constant model, then we would also expect λ to be smaller in this case. The distributions obtained for all five hyperparameters will then be checked using cross validation.

4.5.1 Prior assumptions for the linear trend parameters

To simplify the analysis, the prior distributions for the original three hyperparameters, β, ω^2 and λ will be fixed to those chosen previously; improper prior for β, inverse gamma distribution with shape parameter 0.001 and scale parameter 1000 for ω^2, and uniform distribution between 50 and 40,000 for λ. The initial values for the chains of the original hyperparameters will also be the same as those chosen before. The initial values for β_x and β_y will both be set to zero for chain 1, -0.01 for chain 2 and 0.01 for chain 3. We now consider various prior distributions for β_x and β_y. As when investigating the effect of priors for the constant model, we fix the prior distributions for the other hyperparameters, and so the prior for β_y will be fixed at an improper prior whilst investigating β_x and vice versa.

4.5.2 Prior distribution for β_x

There is no prior information for β_x. Therefore we start by trying an improper prior distribution for β_x, as we did for β when considering a constant mean. We also investigate the use of a normal distribution with zero mean and large variance. The results from using these priors are shown in Table 4.8.

We can see that the change in prior distribution has no effect on the posterior distribution of β_x. The first 50 MCMC iterations for the posterior distribution given an improper prior is shown in the trace in Figure 4.17. The chains converge to the same posterior distribution.

Prior distribution for β_x	Posterior distribution for β_x				
	mean	s.d.	2.5 %	median	97.5 %
Improper	-2.69×10^{-4}	6.78×10^{-5}	-3.97×10^{-4}	-2.71×10^{-4}	-1.27×10^{-4}
$\mathcal{N}(0, 10^3)$	-2.69×10^{-4}	6.78×10^{-5}	-3.97×10^{-4}	-2.71×10^{-4}	-1.27×10^{-4}
$\mathcal{N}(0, 10^6)$	-2.69×10^{-4}	6.78×10^{-5}	-3.97×10^{-4}	-2.71×10^{-4}	-1.27×10^{-4}

Table 4.8: Effects of prior distributions for β_x on the posterior distribution of β_x.

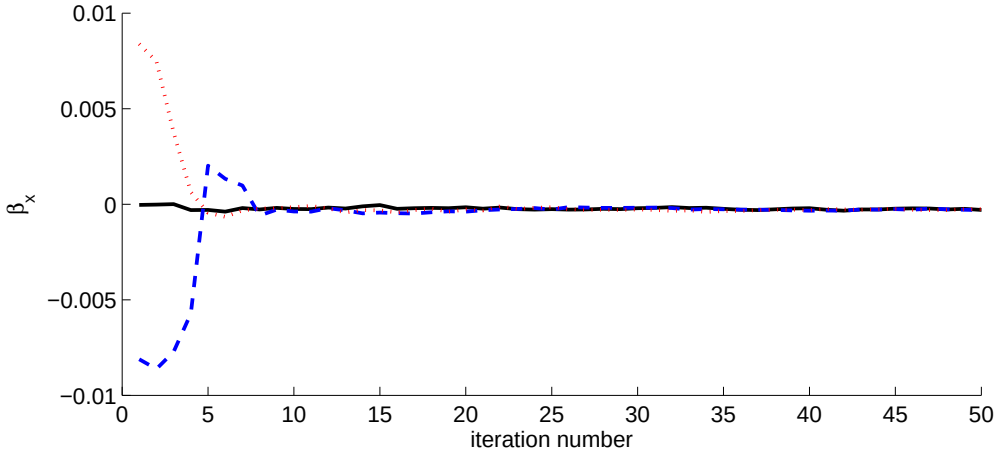

Figure 4.17: MCMC traces for all three β_x chains.

The same convergence diagnostics as before were used to test whether the MCMC chains had converged. The Geweke convergence diagnostic showed no evidence that the chains had not converged. The Heidelberger and Welch stationarity test showed that there was no evidence of non-stationarity in β_x. The halfwidth test was passed, suggesting that the mean of each sample has been estimated with acceptable accuracy. Figure 4.18 shows the posterior density of β_x given an improper prior using samples from the posterior distribution. We will use the distribution resulting from the use of the improper prior for further analysis.

4.5.3 Prior distribution for β_y

We have no prior information for β_y, so again we investigate the use of an improper prior and a normal prior with large variance. The results are shown in Table 4.9. Again the change in prior distribution has no effect on the posterior distribution of β_y. The first 50

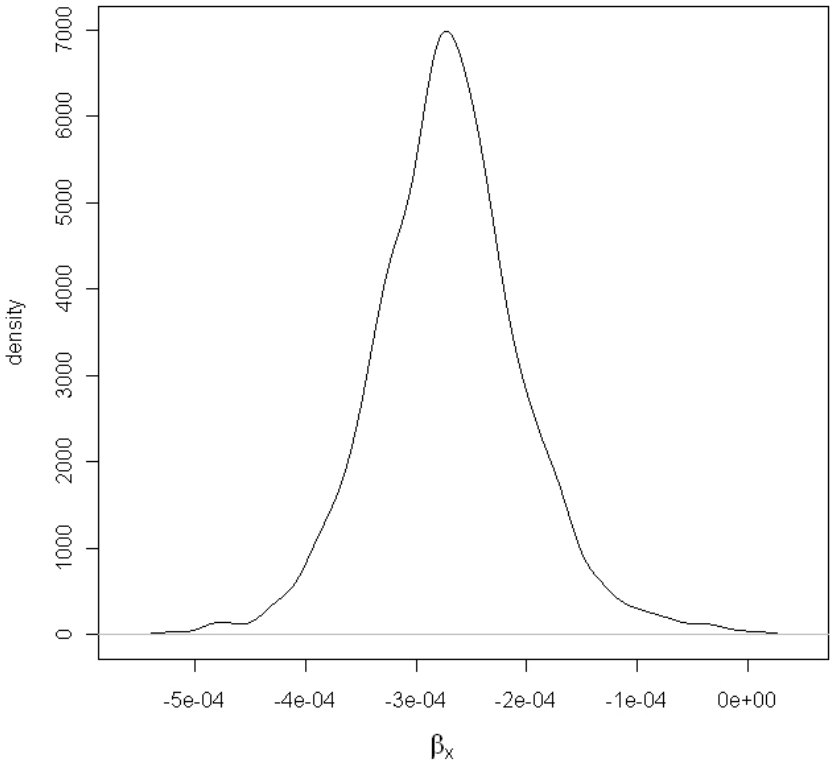

Figure 4.18: Posterior density obtained when using improper prior for β_x.

MCMC iterations for the posterior distribution given an improper prior is shown in the trace in Figure 4.19. We can see that the three chains converge to the same posterior distribution.

The results of the Geweke convergence diagnostic do not provide any evidence against convergence of this chain. The chain also passes the Heidelberger and Welch stationarity and halfwidth tests, which suggests that all iterations be retained for posterior inference. Figure 4.20 shows the posterior density of β_y given an improper prior using samples from the posterior distribution.

We will use the distribution resulting from the use the improper prior for further analysis.

Prior distribution for β_y	Posterior distribution for β_y				
	mean	s.d.	2.5 %	median	97.5 %
Improper	-3.56×10^{-5}	4.99×10^{-5}	-1.33×10^{-4}	-3.59×10^{-5}	6.43×10^{-5}
$\mathcal{N}(0, 10^3)$	-3.56×10^{-5}	4.99×10^{-5}	-1.33×10^{-4}	-3.59×10^{-5}	6.43×10^{-5}
$\mathcal{N}(0, 10^6)$	-3.56×10^{-5}	4.99×10^{-5}	-1.33×10^{-4}	-3.59×10^{-5}	6.43×10^{-5}

Table 4.9: Effects of prior distributions for β_y on the posterior distribution of β_y.

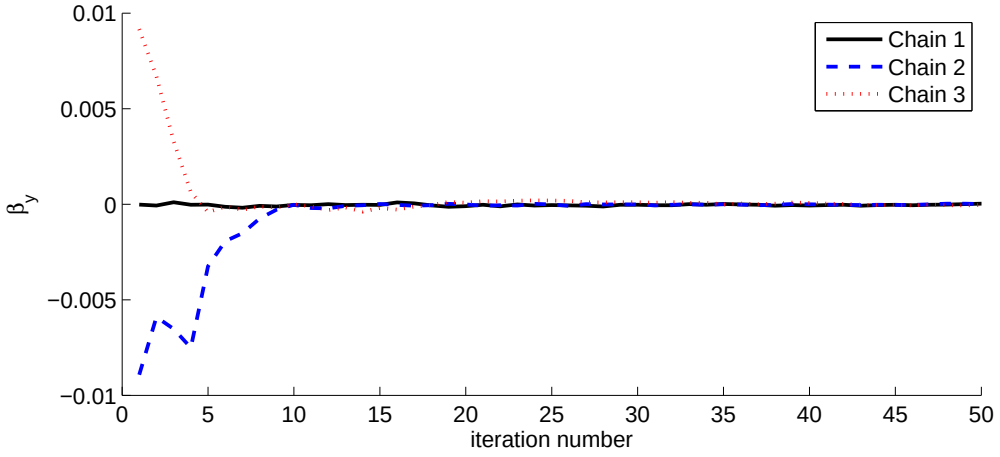

Figure 4.19: MCMC trace for all three β_y chains.

4.5.4 Changes to the posterior distributions of the original stochastic model parameters

By including a linear trend in the data analysis, we expect the posterior distributions of the original hyperparameters β, ω^2 and λ to change. If the linear model fits the data better than the constant model, the variance, ω^2, and the correlation length, λ, of the log transmissivity field should be expected to be smaller. This reduction in the variability of the log transmissivity field should lead to less variability in the travel time. Table 4.10 shows the distributions of all the linear model hyperparameters. From the table we see that the distributions of ω^2 and λ have smaller means and spread than when we were investigating a constant mean (c.f. Table 4.6). The mean and spread for β has also reduced, although this is not comparable to before since the mean now has three parameters instead of one. Using the means of the three mean parameters the trend in the x direction amounts to a decrease in log transmissivity of about 6 from east to west

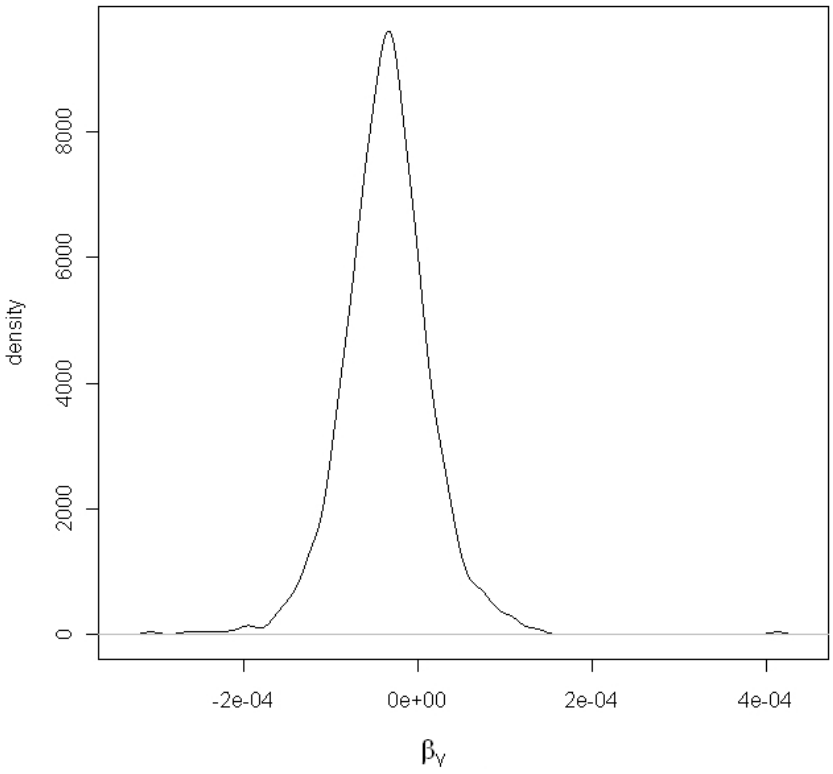

Figure 4.20: Posterior density obtained when using improper prior for β_y.

across the region. The trend in the y direction is smaller, with a reduction of about 1 from north to south across the region. This suggests that the south of the region is more transmissive than the north, and there may also be a slight increase in transmissivity from east to west. Therefore if radionuclides were released in the centre of the region, we might expect them to be carried via the most transmissive route from north to south.

4.5.5 Cross validation

We now check that the new posterior distributions derived for $\boldsymbol{\theta}_l$ give sensible realisations of the log transmissivity field. The posterior distributions for $\boldsymbol{\theta}_l$ are used to estimate each observed value $Z(\mathbf{x}_i)$, $i = 1, \ldots, 39$ using $\mathbf{d}_{-i}$, the data $\mathbf{d}$ with one observation, $d_i = Z(\mathbf{x}_i)$, omitted. We update the prior representation of the log transmissivity field

Hyperparameter	Posterior distribution				
	mean	s.d.	2.5 %	median	97.5 %
β	-2.034	1.186	-4.319	-2.039	0.2919
β_x	-2.69×10^{-4}	6.78×10^{-5}	-3.97×10^{-4}	-2.71×10^{-4}	-1.27×10^{-4}
β_y	-3.56×10^{-5}	4.99×10^{-5}	-1.33×10^{-4}	-3.59×10^{-5}	6.43×10^{-5}
λ	3001	3604	13.88	2163	12600
ω^2	1.956	1.676	0.8628	1.521	6.362

Table 4.10: Derived posterior distributions of $\boldsymbol{\theta}_l$.

$Z(\mathbf{x})$ given in (4.1.7) and (4.1.10) using the log transmissivity data to give

$$Z(.)|\boldsymbol{\theta}_l, \mathbf{d} \sim \mathcal{N}\left(m^*(.), \omega^2 c^*(.,.)\right), \tag{4.5.1}$$

where

$$m^*(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta} + \mathbf{t}(\mathbf{x})^T A^{-1}(\mathbf{d}_{-i} - H_{(n-1)}\boldsymbol{\beta}), \tag{4.5.2}$$

$$c^*(\mathbf{x}, \mathbf{x}') = c(\mathbf{x}, \mathbf{x}') - \mathbf{t}(\mathbf{x})^T A^{-1}\mathbf{t}(\mathbf{x}'), \tag{4.5.3}$$

$$\begin{aligned}
\mathbf{h}(\mathbf{x})^T &= (1, x, y) \\
\boldsymbol{\beta}^T &= (\beta, \beta_x, \beta_y) \\
H_{(n-1)}^T &= (\mathbf{h}(\mathbf{x}_1), \ldots, \mathbf{h}(\mathbf{x}_{n-1})), \\
\mathbf{t}(\mathbf{x}) &= (c(\mathbf{x}_1, \mathbf{x}), \ldots, c(\mathbf{x}_{n-1}, \mathbf{x})),
\end{aligned}$$

and

$$A = \begin{pmatrix} 1 & c(\mathbf{x}_1, \mathbf{x}_2) & \cdots & c(\mathbf{x}_1, \mathbf{x}_{n-1}) \\ c(\mathbf{x}_2, \mathbf{x}_1) & 1 & & \vdots \\ \vdots & & \ddots & \\ c(\mathbf{x}_{n-1}, \mathbf{x}_1) & \cdots & & 1 \end{pmatrix}.$$

Again, we need to estimate the values of $Z(.)$ conditional on the data $\mathbf{d}_{-i}$ only. The integral (4.3.6) with $\boldsymbol{\theta}_l$ instead of $\boldsymbol{\theta}_c$ can be used for this, but it cannot be calculated analytically due to the choice of prior distributions. We therefore use the same method as in Section 4.3.7 to approximate this integral.

Figure 4.21 shows the results from this cross validation. All observed values, except that at borehole P-18, lie within the 95% of the approximated values of log transmissivity.

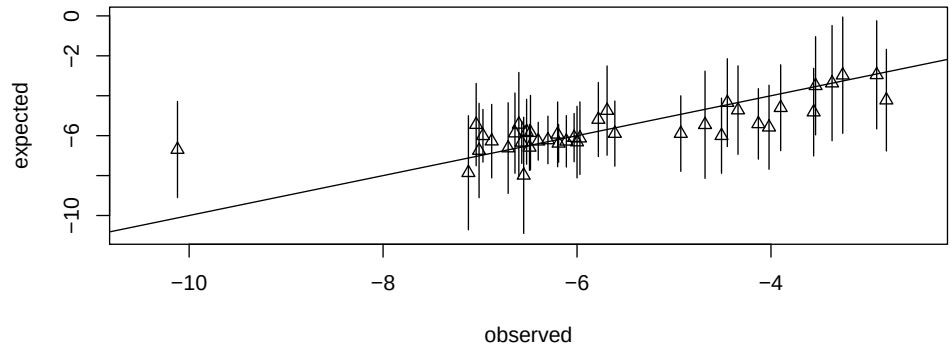

Figure 4.21: Cross validation results for all data points with 95% bounds. Solid line shows expected = observed.

Therefore the distributions of the hyperparameters $\boldsymbol{\theta}_l$ give a good approximation of the log transmissivity field. We can now use these distributions to estimate how the uncertainty in $Z(.)$ and $\boldsymbol{\theta}_l$ are propagated through our computer model.

4.6 Bayesian inference for the hyperparameters $\boldsymbol{\theta}_d$ of the log transmissivity field with depth dependent mean

We now consider a third form for the mean of the log transmissivity field which takes the depth of the overlying rock into account,

$$\mathrm{E}[Z(\mathbf{x}))] = \beta + \beta_d \mathrm{depth}(\mathbf{x}). \tag{4.1.8}$$

The depth term depth($\mathbf{x}$) allows the transmissivity field to vary when the depth of the Culebra dolomite changes. The effect of the depth is that as the thickness of the overlying rock increases, the transmissivity decreases and vice-versa. We need to carry out inference for the distributions of the hyperparameters of this field $\boldsymbol{\theta}_d = (\beta, \beta_d, \lambda, \omega^2)$. We will see how including the depth term changes the distributions of the original three hyperparameters, $\boldsymbol{\theta}_c$. We can also compare the hyperparameter distributions to those obtained when using a linear mean. As with including a linear trend, we expect

that a log transmissivity mean dependent on depth will provide a better fit to the data, and so the variance, ω, and correlation length, λ, hyperparameters will be smaller than those obtained using a constant mean. The distributions obtained for the four hyperparameters, $\boldsymbol{\theta}_d$, will be checked using cross validation.

4.6.1 Dealing with missing depth data

The measured depth data for the boreholes are given in Table A.2. The depth values at four of the measurement points (CB-1, ENGLE, USGS-1, D-268) are missing from the data set. Therefore, we need to approximate the depth values at these points when using WinBUGs to approximate the transmissivity field parameters. We could use an interpolation method such as kriging to estimate the missing depth values. However, WinBUGs allows us to incorporate missing data into a model by including a `NA` entry into the data file. This specifies the missing values as parameters of the model. We can then use multiple imputation (Kenward and Carpenter (2007)) to determine the conditional distribution for the missing data given the observed depth values.

To carry out multiple imputation, the analysis is based on two separate models. The first model is the usual model of interest or target model. The second model is an imputation model which defines the conditional distribution of the missing data given the observed data. The imputation model provides the model of interest with distributions for the missing data values, or parameters of the target model as they have now become. The target model then uses these distributions when providing distributions for the parameters of interest.

In our case, the target model describes the log transmissivity field with a depth dependent mean and the usual exponential correlation function. The imputation model describes the depth field with mean given by

$$\mathrm{E}[d(\mathbf{x}))] = \beta^{\mathrm{imp}} + \beta_t^{\mathrm{imp}} Z(\mathbf{x}), \tag{4.6.1}$$

where β^{imp} and β_t^{imp} are regression coefficients. We use the Gaussian exponential correlation function for the depth field,

$$c(r) = \exp\left\{-\left(\frac{r}{\lambda^{\mathrm{imp}}}\right)^2\right\}, \tag{4.6.2}$$

where λ^{imp} is the correlation length. The Gaussian correlation function is chosen for the depth as we expect this field to be smooth. Even if it is not very smooth, variations

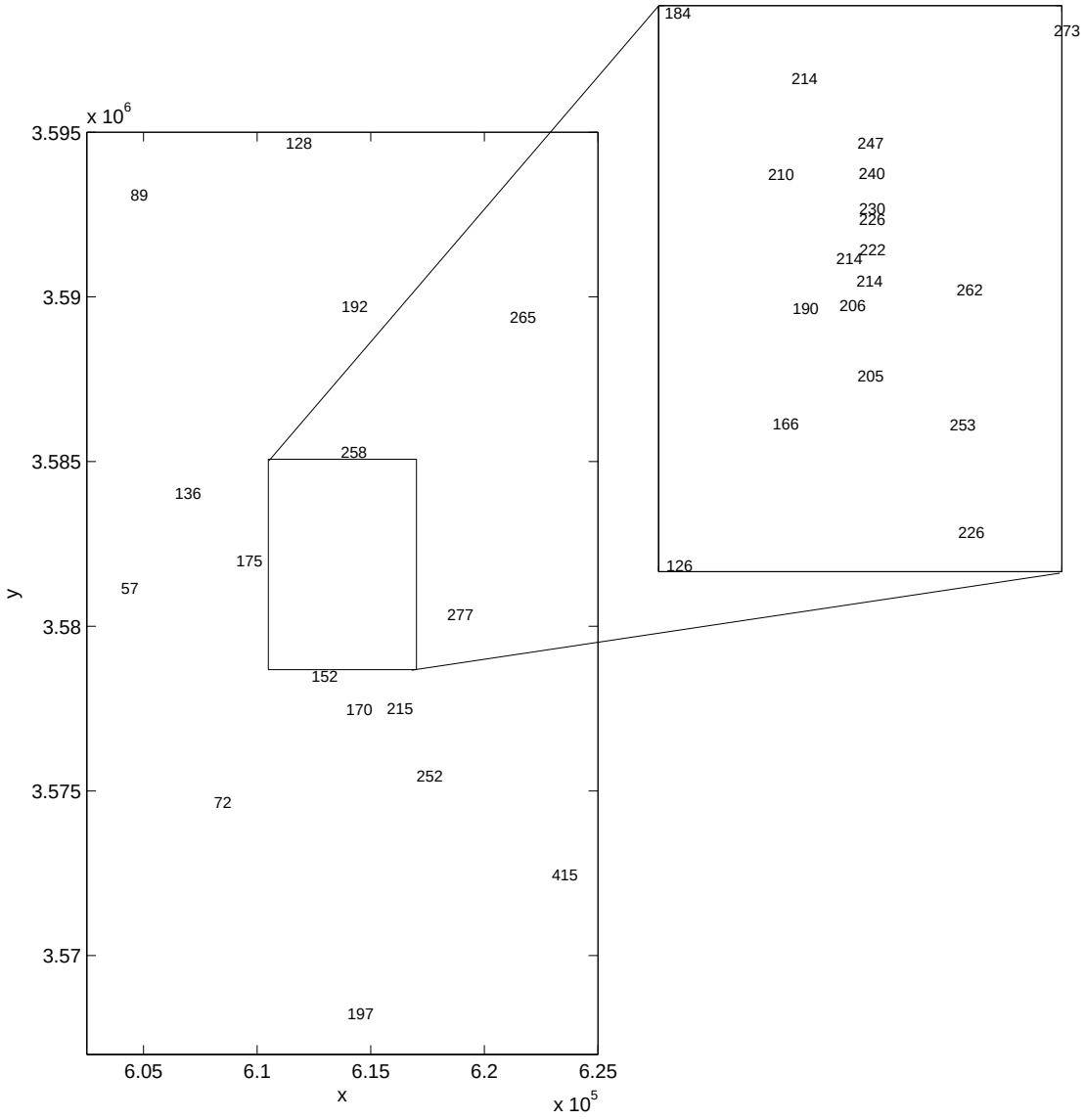

Figure 4.22: Locations and values of the depth data.

in the depth field will be provided for in the stochastic model for the log transmissivity. The variance of this field is given by $\omega^{2^{\text{imp}}}$.

This imputation model is purely used to allow us to estimate distributions for the missing depth parameters. This in turn allows WinBUGs to estimate posterior distributions for the hyperparameters $\boldsymbol{\theta}_d$. Samples from these distributions are then used to generate log transmissivity fields for use in the groundwater flow code. The additional model means that there are now eight unknown parameters that need to be assigned prior distributions and initial values for in WinBUGs: four from the stochastic model for the log transmissivity field $(\beta, \beta_d, \lambda, \omega^2)$, and four from the imputation model for the depth field $(\beta^{\text{imp}}, \beta_t^{\text{imp}}, \lambda^{\text{imp}}, \omega^{2^{\text{imp}}})$. The distributions of the missing depth data are estimated and used within the WinBUGs code and so do not require initial values and prior distributions. We also use the `cut` function in WinBUGs so that information flows only from the imputation model to the target model. Therefore the target model will only use the distributions for the missing depth data, and will ignore all other parameters of the imputation model.

4.6.2 Prior assumptions for the depth trend parameters

For the main model of the log transmissivity field, we again choose the prior distributions for the original three hyperparameters $\boldsymbol{\theta}_c$ to be the same as before. This will simplify the additional analysis that needs to be carried out. The initial values for the chains of β, λ and ω^2 will also be the same as previously used.

We also need to assign prior distributions for the parameters of the imputation model of the depth field. We have no prior information about these parameters so we choose the following uninformative prior distributions: Gaussian priors with zero mean and large variance for β^{imp} and β_t^{imp}, and inverse Gamma distributions; inv$\mathcal{G}(1, 100)$ for λ^{imp} and inv$\mathcal{G}(0.001, 1000)$ for $\omega^{2^{\text{imp}}}$.

Since the log transmissivity field depends only on the parameters of the target model, we will only look at varying the prior distribution for β_d in this section.

4.6.3 Prior distribution for β_d

We have no prior information about the hyperparameter β_d, so we investigate an improper prior distribution, and zero-mean normal distributions with large variances. The results of this analysis is shown in Table 4.11. We can see from the table that changing

Prior distribution for β_d	Posterior distribution for β_d				
	mean	s.d.	2.5 %	median	97.5 %
Improper	-4.98×10^{-4}	3.91×10^{-3}	-9.11×10^{-3}	-1.45×10^{-4}	6.47×10^{-3}
$\mathcal{N}(0, 10^3)$	-4.98×10^{-4}	3.91×10^{-3}	-9.11×10^{-3}	-1.45×10^{-4}	6.47×10^{-3}
$\mathcal{N}(0, 10^6)$	-4.98×10^{-4}	3.91×10^{-3}	-9.11×10^{-3}	-1.45×10^{-4}	6.47×10^{-3}

Table 4.11: Effects of prior distributions for β_d on the posterior distribution of β_d.

the prior distribution has no effect on the posterior distribution for β_d. Therefore we will use an improper prior for the resulting analysis.

We can also output the distributions for the missing depth values to check that sensible values of depth are being generated when the posterior distribution for β_d is being derived. The depth field used in the original WIPP analysis (U.S. D.O.E. (2004)) was generated with the help of expert geologists, but we do not have the values at these missing data points to compare these estimates with. We can use kriging to estimate of the mean of these values using the method described in Section 3.3.4. These kriged estimates are displayed in Table 4.12 along with the estimates generated in WinBUGS. The kriged estimates are similar to the posterior distributions. We also note that that 95% of the distributions for the depth at each borehole all lie with the minimum depth of 57m and maximum depth of 415m measured at the other 35 boreholes.

Borehole	Kriged estimate	Posterior distribution for depth at borehole				
		mean	s.d.	2.5 %	median	97.5 %
CB-1	147.7	142.8	13	116.2	143.2	167.5
ENGLE	205.6	210.7	16.17	177.1	211.3	241.3
USGS-1	163.8	184.9	14.09	155.7	185.4	211.5
D-268	108.7	109.2	14.51	80.36	109.5	137.4

Table 4.12: Distributions for the missing depth data along with the mean kriged estimates.

4.6.4 Changes to the posterior distributions of the original stochastic model parameters

By including the depth dependant term in the mean of the log transmissivity field we expect the posterior distributions of the hyperparameters to change. If the depth model is a better fit to the data than the constant model, we would expect the covariance hyperparameters, λ and ω^2 to reduce. This reduction in variability should then reduce the variability in the log transmissivity field. The posterior distributions of the hyperparameters $\boldsymbol{\theta}_d$ are shown in Table 4.13. When the distributions of λ and ω^2 are compared

Hyperparameter	Posterior distribution				
	mean	s.d.	2.5 %	median	97.5 %
β	-4.872	1.68	-8.509	-4.776	-1.468
β_d	-4.98×10^{-4}	3.91×10^{-3}	-9.11×10^{-3}	-1.45×10^{-4}	6.47×10^{-3}
λ	10600	7393	815	8714	27790
ω^2	5.987	3.846	1.611	4.885	15.85

Table 4.13: Derived posterior distributions of $\boldsymbol{\theta}_d$.

to those derived when using a constant mean (c.f. Table 4.6), we see that they are both slightly smaller. However, they are not as small as when we considered a linear mean (c.f. Table 4.10). This may be because we have added more variability by including our uncertainty about the depth field as well as the log transmissivity field. The distribution for β cannot be directly compared to that derived when using a constant mean model.

We observe in Figure 4.22 that the Culebra dolomite is deeper in the east of the region than in the west. If we take the mean values for β and β_d, along with this trend in the depth data, the mean log transmissivity field will have slightly larger values (between 0.1 and 0.2 larger) in the west than in the east, suggesting that the groundwater will flow slightly faster in the west of the region. This is a similar trend to that observed in the linear model, although the difference between the east and west transmissivities described by the linear model were around ten times as large. We also considered a model incorporating both the linear and depth trends. However the MCMC did not converge suggesting that the information given by both models together does not add anything new to the analysis.

Chapter 5

Waste Isolation Pilot Plant computer model

This chapter describes the computer model for the WIPP case study that we wish to emulate. The groundwater flow equations need to be approximated using a numerical model. The finite element method we use is described, along with the discretisation of the equations. A simple test example, where the analytical solution is known, is used to check the computer model is giving the correct answers. Then we discuss the generation of the log transmissivity fields, starting with how the eigenvalues and eigenvectors of the correlation matrix of the log transmissivity field are found using the finite element method approximation to the eigenvalue equation. We then use the available measured log transmissivity data to condition the generated log transmissivity fields. The realisations of the log transmissivity field can be generated using either the Cholesky decomposition of the covariance matrix, which captures the full uncertainty of the problem, or the truncated K-L expansion which reduces the number of degrees of freedom and therefore the computational cost. We discuss how many K-L modes we need in order to capture the most uncertainty in the smallest number of modes. We then compare the realisations of transmissivity and head fields generated when using a constant, a linear and a depth dependent mean for the log transmissivity field. Finally, we consider the errors in our model that arise from truncating the K-L expansion and from estimating the mean travel time for a given set of hyperparameters.

5.1 Finite element approximation to WIPP model equations

Recall from Section 4.1.1, the groundwater flow equation

$$-\nabla \cdot T(\mathbf{x})\nabla h(\mathbf{x}) = 0 \quad \text{in } \Omega, \tag{5.1.1}$$

and the boundary condition

$$h = h_0(\mathbf{x}) \quad \text{on } \partial\Omega. \tag{5.1.2}$$

We now want to formulate equation (5.1.1) so it can be approximated numerically using the finite element method. We also want to solve the transport equation

$$\dot{\boldsymbol{\zeta}} = \mathbf{u}(\boldsymbol{\zeta}) = -\frac{T(\boldsymbol{\zeta})}{b\phi}\nabla h(\boldsymbol{\zeta}), \tag{5.1.3}$$

with initial condition

$$\boldsymbol{\zeta}(0) = \boldsymbol{\zeta}_0. \tag{5.1.4}$$

Equation (5.1.3) can be solved using integration once we know the head everywhere in the region from solving (5.1.1).

We choose a mixed finite element method, which will allow us to solve for the head, $h(\mathbf{x})$, and velocity $\mathbf{u}$ simultaneously and to the same order of accuracy. This is preferable to solving (5.1.1) for the head and then differentiating to obtain the velocity, where accuracy in the velocity approximation would be lost (Russell and Wheeler (1983)). The velocity can then be used directly in the transport equation (5.1.3) and differentiated to obtain the travel time.

To obtain a mixed formulation of (5.1.1), we separate it into the two equations that we obtained from; Darcy's law

$$\mathbf{u} = -T\nabla h, \tag{5.1.5}$$

and the mass conservation equation

$$\nabla \cdot \mathbf{u} = 0. \tag{5.1.6}$$

These two equations can now be formulated using finite element methods.

5.1.1 Weak formulation

We find the weak formulation of (5.1.5) by dividing through by T, multiplying by a weight function **w** and then integrating over the domain:

$$\int_\Omega \frac{1}{T}\mathbf{u}\cdot\mathbf{w}d\Omega = -\int_\Omega \nabla h\cdot\mathbf{w}d\Omega.$$

Integration by parts on the right hand side gives

$$\int_\Omega \frac{1}{T}\mathbf{u}\cdot\mathbf{w}d\Omega - \int_\Omega h\nabla\cdot\mathbf{w}d\Omega = -\int_{\partial\Omega} h\mathbf{w}\cdot\mathbf{n}d\Gamma. \tag{5.1.7}$$

For the mass conservation equation (5.1.6), we integrate over the domain and note that, by the divergence theorem,

$$\int_\Omega \nabla\cdot\mathbf{u}d\Omega = \int_{\partial\Omega}\mathbf{u}\cdot\mathbf{n}d\Gamma = 0. \tag{5.1.8}$$

Before approximating equations (5.1.8) and (5.1.7) using finite elements, we will describe how the domain is to be discretized.

5.1.2 Discretization of the domain and equations

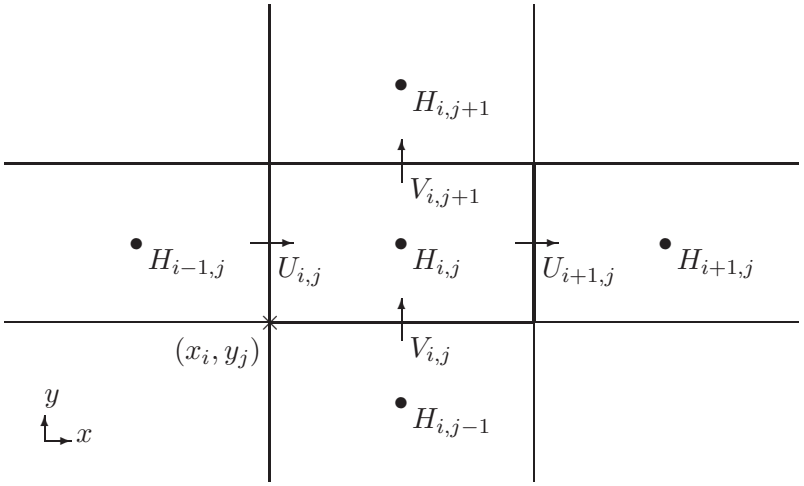

Figure 5.1: One element of the discretised domain.

The domain Ω is divided up into an N_X by N_Y grid of rectangular elements of size $\Delta x \times \Delta y$. The weak forms (5.1.7) and (5.1.8) can be expressed in finite element form by

allowing the continuous functions $h, \mathbf{u}$ and $\mathbf{w}$ to be represented by piecewise functions $H, \mathbf{U}$ and $\mathbf{W}$ which are described on the rectangular elements as follows. The head values H (and the transmissivity values T) are found on the centre of each element ($\bullet$ nodes), while the velocity values $\mathbf{U} = (U, V)$ are constant over the faces of each element ($\rightarrow$ nodes). Element $i, j, (i = 1, \ldots, N_X, j = 1, \ldots, N_Y)$, and the nodes on and around this element are shown in Figure 5.1. Each element is referenced by its bottom left hand corner.

The head H is represented by a piecewise constant function where $H_{i,j}$ is the value of H on element i, j. The velocity $\mathbf{U}$ is represented by a piecewise linear function. On element i, j this discretised form for U is

$$\mathbf{U} = U_{i,j}W^U_{i,j}(x) + U_{i+1,j}W^U_{i+1,j}(x) + V_{i,j}W^V_{i,j}(y) + V_{i,j+1}W^V_{i,j+1}(y).$$

where

$$W^U_{i,j}(\mathbf{x}) = \frac{x_i + \Delta x - x}{\Delta x}, \qquad W^U_{i+1,j}(\mathbf{x}) = \frac{x - x_i}{\Delta x},$$
$$W^V_{i,j}(\mathbf{x}) = \frac{y_i + \Delta y - y}{\Delta y}, \qquad W^V_{i,j+1}(\mathbf{x}) = \frac{y - y_i}{\Delta y}.$$

5.1.3 Flow equation

Under the above discretisation, equation (5.1.7) becomes

$$\underbrace{\int_\Omega \frac{1}{T}\mathbf{U}\cdot\mathbf{W}d\Omega}_{I} - \underbrace{\int_\Omega H_0 \nabla\cdot\mathbf{W}d\Omega}_{II} = -\underbrace{\int_{\partial\Omega} H\mathbf{W}\cdot\mathbf{n}d\Gamma}_{III}. \tag{5.1.9}$$

We will consider each term of equation (5.1.9) separately.

Term I First we look at I. The integral can be split up elementwise and evaluated at each of the four nodes on the boundary of each element.

We consider the contribution to the west node $U_{i,j}$ from element i, j:

$$\int_{\Omega_{i,j}} \frac{1}{T}\mathbf{U}\cdot\mathbf{W}^U_{i,j}d\Omega_{i,j},$$

where $\mathbf{W}^U_{i,j} = 1$ on node $U_{i,j}$ and 0 on all other nodes. For the west node $U_{i,j}$, we only consider the terms in the x direction. We also have that

$$W^U_{i,j}(x) = 1 - \frac{(x - x_i)}{\Delta x} \quad \text{and} \quad W^U_{i+1,j}(x) = \frac{(x - x_i)}{\Delta x}.$$

Therefore,

$$\begin{aligned}\int_{\Omega_{i,j}} \frac{1}{T}\mathbf{U}\cdot\mathbf{W}_{i,j}d\Omega_{i,j} &= \frac{\Delta y}{T_{i,j}}\int_{x_i}^{x_i+\Delta x}\left\{U_{i,j}\left(1-\frac{(x-x_i)}{\Delta x}\right)^2\right.\\ &\quad \left.+U_{i+1,j}\left(1-\frac{(x-x_i)}{\Delta x}\right)\left(\frac{(x-x_i)}{\Delta x}\right)\right\}dx\\ &= \frac{\Delta y}{T_{i,j}}\left[-\frac{U_{i,j}}{3}\left(1-\frac{(x-x_i)}{\Delta x}\right)^3\Delta x\right.\\ &\quad \left.+U_{i+1,j}\left\{\frac{(x-x_i)^2}{2\Delta x}-\frac{(x-x_i)^3}{3\Delta x^2}\right\}\right]_{x_i}^{x_i+\Delta x}\\ &= \frac{\Delta x\Delta y}{T_{i,j}}\left\{\frac{U_{i,j}}{3}+\frac{U_{i+1,j}}{6}\right\}.\end{aligned}$$

The west node also has a contribution from element $i-1,j$. On this element, in the x direction,

$$\mathbf{U} = U_{i-1,j}W^U_{i-1,j}(x) + U_{i,j}W^U_{i,j}(x),$$

where

$$W^U_{i,j}(x) = 1-\frac{(x_i-x)}{\Delta x} \quad \text{and} \quad W^U_{i-1,j}(x) = \frac{(x_i-x)}{\Delta x}.$$

Therefore,

$$\begin{aligned}\int_{\Omega_{i-1,j}} \frac{1}{T}\mathbf{U}\cdot\mathbf{W}_{i,j}d\Omega_{i-1,j} &= \frac{\Delta y}{T_{i-1,j}}\int_{x_i-\Delta x}^{x_i}\left\{U_{i,j}\left(\frac{(x_i-x)}{\Delta x}\right)^2\right.\\ &\quad \left.+U_{i-1,j}\left(1-\frac{(x_i-x)}{\Delta x}\right)\left(\frac{(x_i-x)}{\Delta x}\right)\right\}dx\\ &= \frac{\Delta x\Delta y}{T_{i-1,j}}\left\{\frac{U_{i-1,j}}{6}+\frac{U_{i,j}}{3}\right\}.\end{aligned}$$

Adding the contributions from elements i,j and $i-1,j$ together we have the contribution to term I from the west node of element i,j is

$$\frac{\Delta x\Delta y}{6}\left\{\frac{1}{T_{i-1,j}}U_{i-1,j}+2\left(\frac{1}{T_{i-1,j}}+\frac{1}{T_{i,j}}\right)U_{i,j}+\frac{1}{T_{i,j}}U_{i+1,j}\right\}. \tag{5.1.10}$$

Similarly, we have the contribution to term I from the east node $U_{i+1,j}$ of element i,j;

$$\frac{\Delta x\Delta y}{6}\left\{\frac{1}{T_{i,j}}U_{i-1,j}+2\left(\frac{1}{T_{i,j}}+\frac{1}{T_{i+1,j}}\right)U_{i,j}+\frac{1}{T_{i+1,j}}U_{i+1,j}\right\}. \tag{5.1.11}$$

The contribution to term I from the north node $V_{i,j+1}$ of element i,j is

$$\frac{\Delta x\Delta y}{6}\left\{\frac{1}{T_{i,j}}V_{i,j-1}+2\left(\frac{1}{T_{i,j}}+\frac{1}{T_{i,j+1}}\right)V_{i,j}+\frac{1}{T_{i,j+1}}V_{i,j+1}\right\}, \tag{5.1.12}$$

and from the south node $V_{i,j}$ of element i,j the contribution is

$$\frac{\Delta x \Delta y}{6}\left\{\frac{1}{T_{i,j-1}}V_{i,j-1}+2\left(\frac{1}{T_{i,j-1}}+\frac{1}{T_{i,j}}\right)V_{i,j}+\frac{1}{T_{i,j}}V_{i,j+1}\right\}. \tag{5.1.13}$$

Over the entire domain, we obtain a linear system of equations. In matrix form, this can be written as

$$M(\mathbf{T})\mathbf{U},$$

where $\mathbf{T}$ is a vector containing the values of T for each element and $\mathbf{U}$ is a vector containing the values of U at each node.

Term II Next we evaluate term II. The contribution to the west node $U_{i,j}$ from element i,j is

$$-\int_{\Omega_{i,j}} H\nabla\cdot\mathbf{W}^U_{i,j}d\Omega_{i,j}.$$

We use the same function for U and basis function for $W^U_{i,j}$ as when considering the west node of element i,j for term I;

$$\begin{aligned}\mathbf{U} &= U_{i,j}W^U_{i,j}(x)+U_{i+1,j}W^U_{i+1,j}(x),\\ \text{and}\quad W^U_{i,j}(x) &= 1-\frac{(x-x_i)}{\Delta x}.\end{aligned}$$

Therefore $\nabla\cdot\mathbf{W}_k=-\frac{1}{\Delta x}$, and so

$$\begin{aligned}-\int_{\Omega_{i,j}} H\nabla\cdot\mathbf{W}^U_{i,j}d\Omega_{i,j} &= \frac{H_{i,j}}{\Delta x}\int_{\Omega_{i,j}}d\Omega_{i,j}\\ &= \frac{H_{i,j}}{\Delta x}\Delta x\Delta y = H_{i,j}\Delta y.\end{aligned}$$

The west node also has a contribution from element $i-1,j$. On this element,

$$\begin{aligned}\mathbf{U} &= U_{i-1,j}W^U_{i-1,j}(x)+U_{i,j}W^U_{i,j}(x),\\ \text{and}\quad W^U_{i,j}(x) &= 1-\frac{(x_i-x)}{\Delta x}.\end{aligned}$$

Then the contribution to node $U_{i,j}$ from element $i-1,j$ is

$$\begin{aligned}-\int_{\Omega_{i-1,j}} H\nabla\cdot\mathbf{W}^U_{i,j}d\Omega_{i-1,j} &= -\frac{H_{i-1,j}}{\Delta x}\int_{\Omega_{i-1,j}}d\Omega_{i-1,j}\\ &= H_{i-1,j}\Delta y.\end{aligned}$$

Therefore, the contribution to term II from the west node of element i, j is $\Delta y(H_{i,j} - H_{i-1,j})$. Similarly, the contribution to term II from the east node $U_{i+1,j}$ of element i, j is $\Delta y(H_{i+1,j} - H_{i,j})$. The contribution to term II from the north node $V_{i,j+1}$ of element i, j is $\Delta x(H_{i,j+1} - H_{i,j})$, and from the south node $V_{i,j}$ of element i, j, the contribution is $\Delta x(H_{i,j} - H_{i,j-1})$.

Over the entire domain, the contribution from each of the nodes gives a linear system which can be represented in matrix from as

$$C\mathbf{H},$$

where H is a vector containing the values of H for each element.

Term III Finally the boundary term III. From the boundary condition (5.1.2), we know the values of H_0 at some points along the boundary. We use interpolation to obtain the boundary values at the boundary nodes.

For each element i, j that lies on the boundary of the region Ω, we can evaluate term III at the node which lies on the boundary $\partial\Omega$. If we consider an element i, j, with its west node $U_{i,j}$ lying on $\partial\Omega$, we have

$$-\int_{\partial\Omega} H^0_{i,j}\mathbf{W}^U_{i,j} \cdot \mathbf{n} d\Gamma.$$

where $H^0_{i,j}$ is the interpolated value of H_0 on the boundary node of element i, j. On the west boundary, $\mathbf{W}^U_{i,j} \cdot \mathbf{n} = -1$ and so the contribution to term III from the west node of boundary element i, j is

$$-H^0_{i,j}\int_{\partial\Omega} d\Gamma = H^0_{i,j}\Delta y.$$

For the east node of element i, j on the east boundary of Ω, the contribution to term III is $-H^0_{i,j}\Delta y$. Similarly, the contributions to term III from the north and south boundary nodes are, respectively, $-H^0_{i,j}\Delta x$ and $H^0_{i,j}\Delta x$.

Over the entire boundary, the contribution from each of the boundary nodes can be written as a vector $\mathbf{b}$.

5.1.4 Mass conservation equation

Equation (5.1.8) becomes

$$\int_{\partial\Omega} \mathbf{U} \cdot \mathbf{n} d\Gamma = 0. \tag{5.1.14}$$

If we consider this equation over one element of the domain, we have that

$$\begin{aligned}\int_{\partial\Omega_e} \mathbf{U}\cdot\mathbf{n}d\Gamma_e &= \int_E U_{\mathrm{E}}d\Gamma_E + \int_S U_{\mathrm{S}}d\Gamma_S + \int_N V_{\mathrm{N}}d\Gamma_N + \int_S V_{\mathrm{S}}d\Gamma_S,\\ &= -U_{\mathrm{E}}\Delta y + U_{\mathrm{W}}\Delta y - V_{\mathrm{N}}\Delta x + V_{\mathrm{S}}\Delta x. \end{aligned} \tag{5.1.15}$$

Over the entire domain, we can represent equation (5.1.14) as

$$C^T\mathbf{U} = 0,$$

where $\mathbf{U}$ is a vector containing the values of U at each node.

5.1.5 Solving the groundwater flow equations

We therefore have the following system of matrix equations to solve

$$\begin{aligned} M(\mathbf{T})\mathbf{U} + C\mathbf{H} &= \mathbf{b}, \\ C^T\mathbf{U} &= 0. \end{aligned} \tag{5.1.16}$$

To simplify the calculations, we replace the matrix M by the mass lumped matrix $\tilde{M}$. Multiplying the equation (5.1.16) through by $C^T\tilde{M}^{-1}$, we get

$$\underbrace{C^T\mathbf{U}}_{=0} + C^T\tilde{M}^{-1}(\mathbf{T})C\mathbf{H} = C^T\tilde{M}^{-1}(\mathbf{T})\mathbf{b}.$$

So we solve

$$A\mathbf{H} = \mathbf{B}, \tag{5.1.17}$$

to obtain $\mathbf{H}$, where $A = C^T\tilde{M}^{-1}(\mathbf{T})C$ and $\mathbf{B} = C^T\tilde{M}^{-1}(\mathbf{T})\mathbf{b}$.

The elements of matrix A and vector $\mathbf{B}$ can be found by considering the contribution to each matrix from the west, east, north and south nodes separately, and then combining the entries. For matrix A, the contribution from the node on the west face of an interior element i, j is

$$\begin{aligned} A_W = C^T\tilde{M}^{-1}(\mathbf{T})C &= \Delta y \times \frac{2}{\Delta x\Delta y}\left(\frac{1}{T_{i-1,j}} + \frac{1}{T_{i,j}}\right)^{-1} \times \Delta y \\ &= \frac{2\Delta y}{\Delta x}\left(\frac{1}{T_{i,j}} + \frac{1}{T_{i-1,j}}\right)^{-1}. \end{aligned}$$

The contribution to $\mathbf{B}$ from the node on the west boundary of a boundary element i, j is

$$\begin{aligned}\mathbf{B}_W = C^T \tilde{M}^{-1}(\mathbf{T})\mathbf{b} &= \Delta y \times \frac{2}{\Delta x \Delta y} T_{i,j} \times H^0_{i,j} \Delta y \\ &= \frac{2\Delta y}{\Delta x} T_{i,j} H^0_{i,j}.\end{aligned}$$

Similarly we have,

$$\begin{aligned}A_E &= \frac{2\Delta y}{\Delta x}\left(\frac{1}{T_{i,j}} + \frac{1}{T_{i+1,j}}\right)^{-1}, \qquad \mathbf{B}_E = \frac{2\Delta y}{\Delta x} T_{i,j} H^0_{i,j}, \\ A_N &= \frac{2\Delta x}{\Delta y}\left(\frac{1}{T_{i,j}} + \frac{1}{T_{i,j+1}}\right)^{-1}, \qquad \mathbf{B}_N = \frac{2\Delta x}{\Delta y} T_{i,j} H^0_{i,j}, \\ A_S &= \frac{2\Delta x}{\Delta y}\left(\frac{1}{T_{i,j-1}} + \frac{1}{T_{i,j}}\right)^{-1}, \qquad \mathbf{B}_S = \frac{2\Delta x}{\Delta y} T_{i,j} H^0_{i,j}.\end{aligned}$$

The contribution from each of the four boundaries of each element is used to generate 4 matrices in MATLAB, which are then added together to obtain matrix A. The vector B is obtained in a similar manner. Then we can use MATLAB to solve equation (5.1.17) for H. To solve for $\mathbf{U}$, we rearrange equation (5.1.16) and again replace matrix M by the mass lumped matrix $\tilde{M}$, to give

$$\mathbf{U} = \tilde{M}^{-1}(\mathbf{b} - CH). \tag{5.1.18}$$

Since $\mathbf{U}$ is constant on the faces of the elements, the contributions to $\mathbf{U}$ from the west face of element i, j, U, and south face of element i, j, V, are given as

$$\begin{aligned}U_{i,j} &= -\frac{2}{\Delta x}\left(H_{i,j} - H_{i-1,j}\right)\left(\frac{1}{T_{i-1,j}} + \frac{1}{T_{i,j}}\right)^{-1}, \\ V_{i,j} &= -\frac{2}{\Delta y}\left(H_{i,j} - H_{i,j-1}\right)\left(\frac{1}{T_{i,j-1}} + \frac{1}{T_{i,j}}\right)^{-1}.\end{aligned}$$

On the boundary these become

$$U_{i,j} = -\frac{2}{\Delta x}\left(H_{i,j} - H^0_{i,j}\right) T_{i,j}, \quad V_{i,j} = -\frac{2}{\Delta y}\left(H_{i,j} - H^0_{i,j}\right) T_{i,j}.$$

We use these equations in MATLAB to generate a matrix of $N_X \times N_Y$ velocities, one for each element. This velocity matrix is then used when solving the transport equation.

5.1.6 Solving the transport equation

Once we have approximated the transmissivity field and calculated the head values for each element, we can solve the transport equation (5.1.3) to calculate the time at which a particle released in the centre of the region reaches the WIPP site boundary. We solve equation (5.1.3) for $\boldsymbol{\zeta}$ using one of the `ode` solvers in MATLAB. Using the initial condition (5.1.4), we can then find the time t for which $\boldsymbol{\zeta}(t)$ lies on the boundary of the site $\partial\Gamma$.

5.1.7 Testing the computer model

We check the computer model is correct by running the model using a simple example where we know the solution. We consider the region Ω to be a unit square and set the boundary conditions such that the solution h is linear in x. We also set T to be unity everywhere in this region. So we solve

$$\begin{aligned} \nabla^2 h(x,y) &= 0, \quad (x,y) \in [0,1] \times [0,1], \\ h(x,0) = h(x,1) &= 10x, \\ h(0,y) &= 0, \\ h(1,y) &= 10. \end{aligned} \tag{5.1.19}$$

Equation (5.1.19) has particular solution $h(x,y) = Ax + By + C$, and using the boundary conditions the solution is $h = 10x$. We run our MATLAB model using the same region size as this test example, and with a 40×60 grid. We obtain the correct solution for h which is linear in x (Figure 5.2).

To check the travel time, we can consider a particle released at point $(0.5, 0.5)$ in the centre of the test region. We solve

$$\begin{aligned} \dot{\boldsymbol{\zeta}}(t) = -T\nabla h, \quad (x,y) \in [0,1] \times [0,1] \\ \boldsymbol{\zeta}(0) = (0.5, 0.5), \end{aligned} \tag{5.1.20}$$

and then use this to find the time at which the particle leaves the region; when $\boldsymbol{\zeta} \in \partial\Omega$. For our test problem, we have

$$\dot{\boldsymbol{\zeta}}(t) = -\nabla 10x = (-10, 0),$$

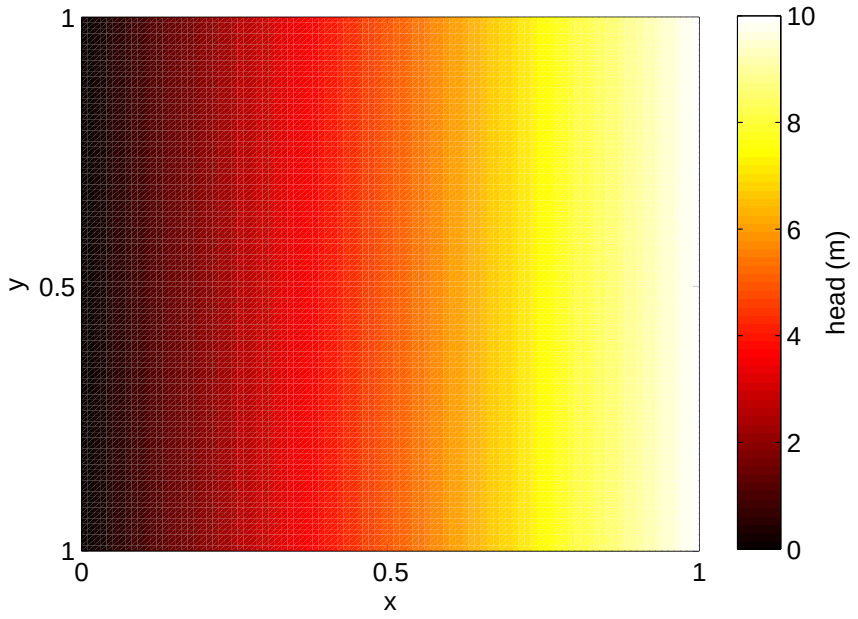

Figure 5.2: MATLAB solution of the head field for the test example.

and so, using the initial data,

$$\boldsymbol{\zeta}(t) = (-10t + 0.5, 0.5).$$

The velocity is in the negative x direction, so the particle will leave the region by the west boundary. The particle will reach the boundary at point $(0, 0.5)$ at time $t = 0.05$. The MATLAB model gives the correct solution for this simple problem.

5.2 Generating realisations of the log transmissivity field

For the computer model we need to generate realisations of the transmissivity field with which to solve the groundwater flow equations. These realisations must all have the same mean and covariance structure. This can be achieved by generating realisations of the log transmissivity field using the Cholesky decomposition, which provides the full variability of the log transmissivity field, or the truncated Karhunen-Loéve expansion, which reduces the number of degrees of freedom and therefore the computational cost. If the K-L expansion is not truncated, then all $N_X \times N_Y$ modes are used and the full

variability in the problem is restored.

In the following section we describe how both methods are used to generate piecewise constant log transmissivity fields which can then be used in the finite element equation (5.1.17) to find the head field. We also consider how the available log transmissivity data can be used to condition the covariance matrix so that the generated log transmissivity fields are true to the measured data.

5.2.1 Calculating the Cholesky decomposition

Due to the discretisation of the equations, the covariance $\mathrm{Cov}(Z(\mathbf{x}), Z(\mathbf{x}')) = \omega^2 \exp \frac{|\mathbf{x}-\mathbf{x}'|}{\lambda}$ is represented by a piecewise constant function $C^p(\mathbf{x}_k, \mathbf{x}_l)$. The covariance can then be represented in matrix form

$$\mathbf{C}_{kl} = C^p(\mathbf{x}_k, \mathbf{x}_l), \quad k, l = 1, \ldots, N_X \times N_Y.$$

This matrix can then be decomposed into lower and upper triangular matrices $C = LU$, where $U = L^T$, using the `chol` function in MATLAB. This function becomes more computationally expensive to calculate as $N_X \times N_Y$ increases.

A realisation of the log transmissivity field is then generated from

$$Z = \mathrm{E}[\log T] + \mathbf{L}\boldsymbol{\xi},$$

where $\mathrm{E}[\log T]$ is the constant or linear mean function of $\log T$, and ξ is a vector of $N_X \times N_Y$ i.i.d. normal random variables with zero mean and unit variance. As discussed in Chapter 3, this generates a log transmissivity field which keeps the desired covariance structure. The exponential of this realisation is then taken to give a realisation of the transmissivity field, T, which can be used in the WIPP computer model. The transmissivity field is therefore represented as a piecewise constant function.

5.2.2 Calculating the eigenvectors and eigenvalues of the K-L expansion

The truncated K-L expansion,

$$Z(\mathbf{x}) = \mathrm{E}[\log T] + \omega \sum_{k=1}^{N} \xi_k \sqrt{e_k} \psi_k(\mathbf{x}), \tag{5.2.1}$$

requires the eigenvalues and eigenfunctions of the correlation function, $C(\mathbf{x}, \mathbf{x}')$. To find these, we formulate

$$\int_\Omega C(\mathbf{x}, \mathbf{x}')\psi_k(\mathbf{x}')d\mathbf{x}' = e_k\psi_k(\mathbf{x}) \tag{5.2.2}$$

using the finite element method.

The solution to equation (5.2.2) can be found numerically using a Galerkin method as follows. First, we multiply equation (5.2.2) through by a weight function $w(\mathbf{x})$, and then integrate over $\mathbf{x}$

$$\int_\Omega \int_\Omega C(\mathbf{x}, \mathbf{x}')\phi_i(\mathbf{x}')w(\mathbf{x})d\mathbf{x}'d\mathbf{x} = e_i \int_\Omega \phi_i(\mathbf{x})w(\mathbf{x})d\mathbf{x}.$$

We discretise the domain Ω into rectangular elements so that now the problem is to find the eigenvalues and eigenvectors of matrix C. We choose a set of piecewise constant basis functions $V = \{\psi_1, \psi_2, \ldots, \psi_M\}$, where $M = N_X \times N_Y$ is the number of elements, and let $w(\mathbf{x}) \in V$. Representing C and ϕ_i by piecewise constant functions C^p and φ_i, we have

$$\int_\Omega \int_\Omega C^p(\mathbf{x}_k, \mathbf{x}'_l)\varphi_i(\mathbf{x}'_l)\psi_j(\mathbf{x}_k)d\mathbf{x}'d\mathbf{x} = e_i \int_\Omega \varphi_i(\mathbf{x}_k)\psi_j(\mathbf{x}_k)d\mathbf{x}.$$

Splitting this equation up element by element gives

$$\begin{aligned}
\sum_{k=1}^{M}\sum_{l=1}^{M} C^p(\mathbf{x}_k, \mathbf{x}'_l)\varphi_i(\mathbf{x}'_l)\psi_j(\mathbf{x}_k) \int_{\Omega_k}\int_{\Omega_l} d\mathbf{x}'d\mathbf{x} &= \sum_{k=1}^{M} e_i\varphi_i(\mathbf{x}_k)\psi_j(\mathbf{x}_k) \int_{\Omega_k} d\mathbf{x} \\
\sum_{k=1}^{M}\sum_{l=1}^{M} C^p(\mathbf{x}_k, \mathbf{x}'_l)\varphi_i(\mathbf{x}'_l)\psi_j(\mathbf{x}_k)(\Delta x\Delta y)^2 &= \sum_{k=1}^{M} e_i\varphi_i(\mathbf{x}_k)\psi_j(\mathbf{x}_k)\Delta x\Delta y.
\end{aligned}$$

If $j = k$ then $\psi_j(\mathbf{x}_k) = 1$, otherwise it is zero. Therefore for each element k,

$$\sum_{l=1}^{M} C^p(\mathbf{x}_k, \mathbf{x}'_l)\varphi_i(\mathbf{x}'_l)\Delta x\Delta y = e_i\varphi_i(\mathbf{x}_k).$$

We can express this equation in matrix form

$$C\Phi = \mathbf{\Lambda}\Phi, \tag{5.2.3}$$

where

$$\begin{aligned}
C_{k,l} &= C^p(\mathbf{x}_k, \mathbf{x}'_l)\Delta x\Delta y, \\
\Phi_{i,k} &= \varphi_i(\mathbf{x}_k), \\
\mathbf{\Lambda}_i &= e_i.
\end{aligned}$$

The solution of the eigenvalue problem (5.2.3) can be found by finding the eigenvalues and eigenvectors of the matrix C. The N largest of these can be found using the `eigs` function in MATLAB. We note here that MATLAB may not use the best technique to calculate the eigenvalues and eigenvectors of a matrix and that better methods are available (Ernst (2009)).

The eigenfunctions of the positive definite symmetric matrix C are orthonormal functions:

$$\int_\Omega \phi_j(\mathbf{x})\phi_k(\mathbf{x})d\mathbf{x} = \delta_{j,k}.$$

Using this orthonormal property of the eigenvectors,

$$\int_\Omega \phi_k(\mathbf{x})^2 = 1.$$

Discretising this we have

$$\begin{aligned} \sum_{i=1}^{M} \varphi_k(\mathbf{x}_i)^2 \int_{\Omega_i} d\mathbf{x} &= 1 \\ \sum_{i=1}^{M} \varphi_k(\mathbf{x}_i)^2 &= \frac{1}{\Delta x \Delta y}. \end{aligned}$$

For each i, we have

$$\varphi_k(\mathbf{x}) = \frac{1}{\sqrt{\Delta x \Delta y}}.$$

Therefore, after calculating the eigenvectors, we must divide them by $\sqrt{\Delta x \Delta y}$ in order to satisfy the orthonormal property.

The calculated eigenvalues and eigenfunctions can then be put into equation (5.2.1), along with N random ξ_i's, to obtain a realisation of the log transmissivity, $Z(\mathbf{x})$. We then obtain a realisation of the transmissivity field, T, by taking the exponential of this field.

5.2.3 Conditioning on measured data

The realisations of the log transmissivity field generated by the above method are based on an estimation of the covariance structure of the field. We can use measured values of transmissivity to condition this field. We assume that there is no measurement error and so the values of transmissivity at the measurement points are the exact values of the

field at these points. There will be zero variance at the measurement points. The easiest way of doing this is to set the values of the piecewise transmissivity function to be equal to the measured value on the elements which include the measurement points. The methods used condition the transmissivity fields generated by the K-L expansion and Cholesky methods are described in Chapter 3. The correlation matrix is conditioned so that the measured values are reproduced in the correct elements, and then the eigenvalue problem (5.2.3) is solved in the same way as before.

We could also consider that the measurements are independently normally distributed around their measured value. At the measurement points there will be a small amount of variance. We could allow for this in the computer model by adding a 'nugget' effect. This may result in a smoother approximation to the transmissivity field, as we allow the field to move away from the measured values at the measurement points.

5.2.4 Realisations of the log transmissivity field

We can use the derived distributions for $\boldsymbol{\theta}_c$, $\boldsymbol{\theta}_l$ and $\boldsymbol{\theta}_d$ to generate samples of $\boldsymbol{\theta}_c$, $\boldsymbol{\theta}_l$ and $\boldsymbol{\theta}_d$ which represent the distribution structure of $\boldsymbol{\theta}_c$, $\boldsymbol{\theta}_l$ and $\boldsymbol{\theta}_d$. From these samples we can generate a number of realisations for the log transmissivity field using the Cholesky decomposition. A realisation generated for each model, using the same value of $\boldsymbol{\xi}$ for each field, are shown in Figure 5.3. We see from plots (a) and (c) that the constant and depth mean models generate almost identical fields when using the same random vector $\boldsymbol{\xi}$, and the linear mean field (b) is similar, although not as much, to both of these. If we look at the central box, where the boundary of the WIPP site lies, we see that all three fields are almost identical in this region. This could be due to the comparatively larger amount of points inside this region than in the surrounding area.

If we examine Figure 4.1, the data seem to suggest larger values of log transmissivity in the East of the region and smaller values in the West. Comparing the plots in Figure 5.3, we see that the realisations generated from the constant mean and depth mean models, plots (a) and (c), do not reflect this East-West trend as much as when using the linear mean model, plot (b). This leads us to consider that a linear mean as a better fit to the data, and perhaps to reduce the uncertainty in the hyperparameters and therefore in the log transmissivity field.

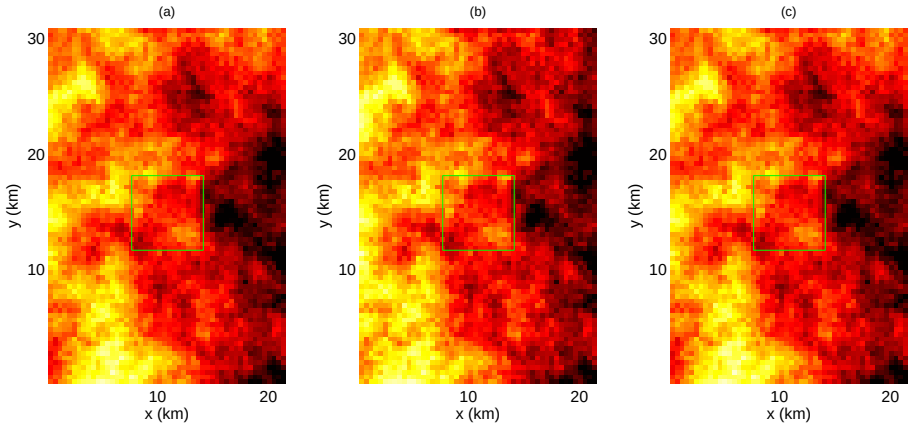

Figure 5.3: Realisations of the conditioned log transmissivity field (a) constant mean (b) linear mean (c) depth mean using the same ξ. White indicates high log transmissivity values and black indicates low log transmissivity values. Inner box represents the WIPP site boundary.

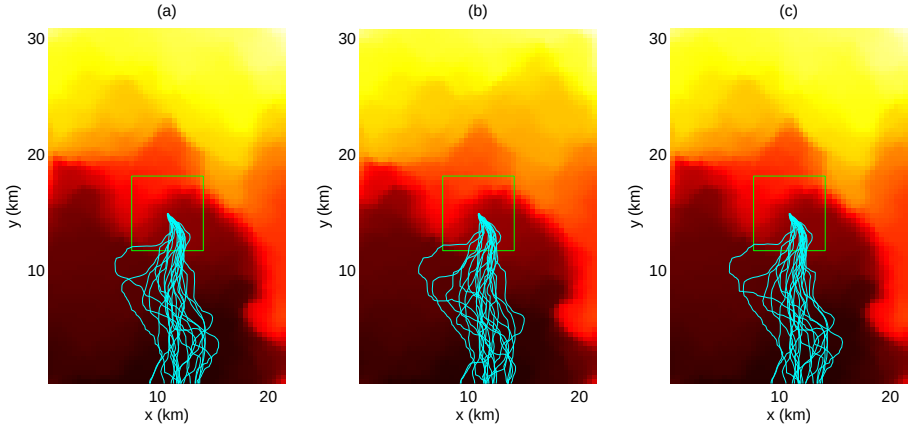

Figure 5.4: Head fields corresponding to the transmissivity fields in Figure 5.3 generated using (a) constant mean (b) linear mean (c) depth mean, with 20 pathlines. White indicates high head values and black indicates low head values. Inner box represents the WIPP site boundary.

5.2.5 Realisations of the head field and pathlines

We can also compare how each model for the transmissivity field affects the head field and pathlines. Realisations of the head field and pathlines generated using the three log transmissivity fields from Figure 5.3 are shown in Figure 5.4. We see again that all three models produce similar fields and pathlines. In plot (b), the linear mean produces a slightly different head field in the northeast of the plot, due to the more pronounced east-west trend in the corresponding transmissivity plot (Figure 5.3(b)). However, since the main direction of travel is south, this does not cause a marked effect on the pathlines, which are also similar for all three models. From this small sample of pathlines, none of the models appears to show more variability than any of the others. However, there may be a difference in the time taken to travel along these paths, and this is what we will investigate by emulating the travel time generated by the computer model in Chapter 6. Before doing so, we want to estimate the errors in the computer model.

5.3 Errors in computer model

The computer model contains errors from three main sources. Firstly there is discretisation error ϵ_d resulting from approximating the continuous equations with a discretised domain. There is also an error introduced by truncating the K-L expansion ϵ_t. The final source of error is statistical error ϵ_s arising from solving the equations using a Monte Carlo method. Each of these errors can be quantified. In this section we discuss the truncation error and statistical error, and attempt to quantify them.

5.3.1 Truncation error for K-L expansion

As mentioned in Section 3.3.2, the truncated K-L expansion (5.2.1) is used to reduce the number of degrees of freedom and so the computational cost of calculating the eigenvalues and eigenvectors. We want to determine the value of N for which we can capture the most variability in the smallest number of modes. For a 40 by 60 grid, we have a total of 2400 eigenvalues and eigenvectors. If we were to use all of these, we would obtain the full variance of the problem and would obtain the same field as that captured by the Cholesky decomposition. Figure 5.5 plots the variance captured against

the number of modes for the conditioned covariance matrices using the mean e values of the constant and linear cases derived in the last chapter.

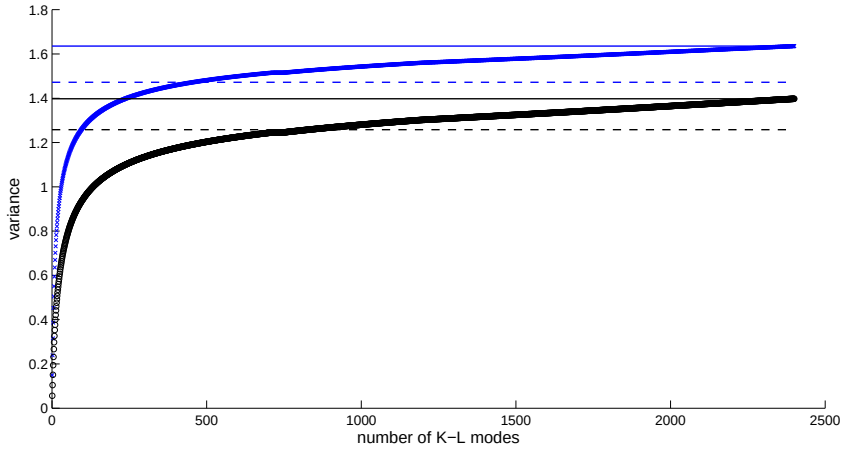

Figure 5.5: Number of modes against the amount of variance captured for the constant mean model (blue), and the linear mean model (black). Total variance for both the linear and constant models are shown by a solid line, and 90% of the total variance is shown by a dashed line.

We see that the constant mean model captures more variance than the linear mean model. This is due to the constant mean model having a larger variance in comparison to the linear mean model; the mean value of $\omega^2 = 6.5$ for constant mean and 2 for linear mean. Therefore, when using the full 2400 modes, the linear mean is capturing approximately $1.4/2 \times 100\% = 70\%$ of the variability in its model. This is much more than the constant model which only captures approximately $1.64/6 \times 100\% = 25\%$ of the variability in its model. This suggests that we should use the linear mean model for the log transmissivity field.

The number of K-L modes to capture 90% of the variance for the constant case is around 500 modes, and for the linear it is around 800 modes. This is large, and so the K-L expansion may not provide as large a reduction in degrees of freedom as anticipated. The calculation of the eigenvalues and eigenvectors of the correlation matrix is one of the most expensive parts of the computer code, and becomes more expensive the more modes we include. However, as mentioned before, there are faster methods than the one which MATLAB uses to calculate the eigenvalues and eigenvectors (Ernst (2009)).

For the purposes of this thesis, we will use a different method to generate realisations of the log transmissivity field such as the Cholesky decomposition method. This method generates a field using all 2400 degrees of freedom, and so there is no truncation error.

5.3.2 Statistical error

Due to the uncertainties in the transmissivity field, discussed in Chapter 4, the computer model needs to be run a large number of times to calculate statistics for the log travel time given the uncertainties in the transmissivity field and in the hyperparameters of this field. It will be impossible to run the computer model over the infinite number of combinations of hyperparameters and transmissivity fields. Therefore we use Monte Carlo (MC) methods to analyse the computer model. There is a statistical error in carrying out this analysis, which reduces as the number of samples increases. We want to quantify and reduce the statistical error as much as possible, while still being able to carry out the analysis within a sensible computational time.

5.3.3 Computational expense of Monte Carlo methods to calculate statistics of the WIPP computer model

When using the K-L expansion the eigenvalue problem is solved once for each set of hyperparameters. To calculate 800 modes on a 40 by 60 grid of the domain, takes approximately 140 seconds using MATLAB, and is the most expensive part of the computer model. This is not too long, but the calculation needs to be repeated for each sample of hyperparameters we wish to evaluate the model with. After the eigenvalues and eigenvectors have been calculated, the computer model takes approximately 0.05 seconds to generate a random log transmissivity field and provide one possible value of s. It will therefore take approximately $140 + 0.05 \times M$ evaluations in order to carry out a Monte Carlo analysis of the computer model over the space of random variables of the K-L expansion. This becomes $(140 + 0.05 \times M) \times N$ when the MC analysis is extended over the space of hyperparameters as well.

For the Cholesky decomposition method, the Cholesky decomposition is solved once for each set of hyperparameters. This takes approximately 4 seconds for a 40 by 60 grid using MATLAB, and again, is the most expensive part of the computer model. After

the decomposition has been calculated, the computer model takes approximately 0.75 seconds to generate a random log transmissivity field and provide one possible value of s. For this method, it will take $4 + 0.75 \times M$ evaluations in order to carry out a Monte Carlo analysis of the computer model over the space of random variables of the Cholesky decomposition. This becomes $(4 + 0.75 \times M) \times N$ when the MC analysis is extended over the space of hyperparameters as well. When $M > 200$, this method is more computationally expensive than the K-L expansion method using 800 modes, but it does include all 2400 modes.

We note that these times have been calculated for a 40 by 60 grid. If we consider a 80 by 120 grid, the Cholesky decomposition takes around 76.8 seconds, and then generating the transmissivity field and calculating the travel time a further 1.04 seconds. Therefore the MC analysis of the computer code would take longer. For even finer meshes, the times taken will be longer again.

5.3.4 Use of emulator to approximate WIPP computer model statistics

Since M and N are both large numbers, this makes the computer model very expensive to analyse for either method. Therefore, we are using a Gaussian process emulator to make the analysis less computationally expensive. The emulator will be used as a cheap substitute for the computer model when performing MC analysis. The use of an emulator also introduces an error, but we can also quantify this error.

To build the emulator we need to calculate statistics (such as the mean, median, cdf etc.) for $s = \log t$ given the hyperparameters $\boldsymbol{\theta}$ for a small set of hyperparameters. We choose to use a log transform on the time so that the emulator provides positive travel times. We are also assuming that, after taking logs, the Gaussian marginal distribution is appropriate and the relationship between $\boldsymbol{\theta}$ and log travel time is linear.

Calculating statistics for each set of hyperparameters involves using a Monte Carlo analysis to estimate the statistics of interest. We can quantify the statistical error in estimating the mean travel time given one set of hyperparameters for different values of M. We also want to estimate the cumulative distribution function $F(s)$ using the emulator. Again, we need to carry out a MC analysis to find an estimate $\widehat{F(s)}$ of the distribution function at a small sample of values of s for each hyperparameter. Then

we can use the emulator to approximate $F(s)$ at those values of s, and then interpolate between them to provide an estimate of the cumulative distribution function. We can quantify the statistical error in calculating $\widehat{F(s)}$ for each set of hyperparameters for different values of M.

5.3.5 Accuracy of the mean log travel time

For one set of hyperparameters, $\boldsymbol{\theta}_i$, we use the following method to carry out the MC analysis to find an estimate of the mean log travel time $\hat{\mathrm{E}}[s|\boldsymbol{\theta}_i]$.

1. Choose a number of evaluations of the computer model M.
2. Generate M independent realisations of $\log T$.
3. For each realisation of $\log T$ find the log travel time s to obtain
$$\boldsymbol{s}|\boldsymbol{\theta}_i = (s_1, s_2, \ldots, s_M).$$
4. Approximate the mean log travel time $\bar{s}|\boldsymbol{\theta}_i$ by
$$\hat{\mathrm{E}}[s|\boldsymbol{\theta}_i]_M = \frac{1}{M}\sum_{j=1}^{M} s_j, \tag{5.3.1}$$

The larger M is, the more accurate our estimate $\hat{\mathrm{E}}[s|\boldsymbol{\theta}_i]$. However, it could become expensive to provide an estimate of the travel time for each set of hyperparameters this way. Therefore, we want to calculate the error in our estimate so we can obtain an accurate estimate within a given tolerance and using a reasonable computational time.

We consider the estimate of the mean travel time given M samples of t, $\widehat{\mathrm{E}[s|\boldsymbol{\theta}_i]}_M$ to be a random variable. The statistical error ϵ_s is defined by

$$\epsilon_s = \hat{\mathrm{E}}[s|\boldsymbol{\theta}_i]_M - \bar{s}|\boldsymbol{\theta}_i. \tag{5.3.2}$$

We define a random variable Y_M by

$$Y_M = \frac{\sqrt{M}}{\sigma}\left(\hat{\mathrm{E}}[s|\boldsymbol{\theta}_i]_M - \bar{s}|\boldsymbol{\theta}_i\right), \tag{5.3.3}$$

where σ is the standard deviation of $\bar{s}|\boldsymbol{\theta}_i$, and $F_M(s) = P(Y_M \leq s)$ is the cdf of Y_M. Given that $\hat{\mathrm{E}}[s|\boldsymbol{\theta}_i]_j, j = 1, \ldots, M$ are independent random variables each with

zero mean, positive variance and finite third absolute moment, then the Berry-Esseen theorem (Feller (1971)) implies that

$$\sup_{s\in\Re}|F_M(s)-\Phi(s)| \leq 0.7655\frac{\mathrm{E}\left[|(s|\boldsymbol{\theta}_i)-(\bar{s}|\boldsymbol{\theta}_i)|^3\right]}{\sigma^3\sqrt{M}}, \tag{5.3.4}$$

where $\Phi(s)$ is the cdf of the standard normal distribution.

Therefore if $M \gg 0.586\frac{\mathrm{E}\left[|(s|\boldsymbol{\theta}_i)-(\log\bar{t}|\boldsymbol{\theta}_i)|^3\right]^2}{\sigma^6}$ then Y_M has the normal distribution and

$$P\left(|\epsilon_s| \leq c_0\frac{\sigma}{\sqrt{M}}\right) = 2\Phi(c_0)-1 \quad \text{for } c_0>0.$$

If $c_0 = 1.96$ then the event $|\epsilon_s| \leq c_0\frac{\sigma}{\sqrt{M}}$ has probability 0.95. We can not calculate σ directly, but instead can estimate it from the sample standard deviation, $\hat{\sigma}$

$$\sigma \approx \hat{\sigma} = \left(\frac{1}{M}\sum_{j=1}^{M}(s_j|\boldsymbol{\theta}_i)^2 - \widehat{\mathrm{E}[s|\boldsymbol{\theta}_i]}_M^2\right)^{\frac{1}{2}}.$$

We now have an upper bound on the statistical error in the estimate of mean log travel time given one set of hyperparameters. The rate of convergence is of order $M^{-\frac{1}{2}}$. Table 5.1 gives the statistical error for increasing values of M using the mean values for $\boldsymbol{\theta}_i$ for the linear case using the Cholesky method. We see that the error decreases as M

Mean estimate, $\widehat{\mathrm{E}[s\|\boldsymbol{\theta}_i]}_M$	Statistical error, $\|\epsilon_s\| \approx c_0\frac{\hat{\sigma}}{\sqrt{M}}$	Error in estimate, %	Number of runs, M	Time, secs
4.6264	0.008	0.1729	500	597
4.6099	0.0054	0.1171	1000	1117
4.6059	0.0033	0.0716	2500	2677
4.6025	0.0024	0.0521	5000	5277
4.6053	0.002	0.0434	7500	7877
4.6063	0.0017	0.0369	10000	10477
4.6062	0.0014	0.0304	15000	15677

Table 5.1: Statistical errors of the estimated mean $\log_{10}$ travel time estimate for increasing number of runs of the computer model and the computational time taken to calculate the estimate.

increases, which is as expected.

We can also consider the running mean values for the mean estimate when deciding what value M should take. Figure 5.6 shows the running mean of the log travel time along with

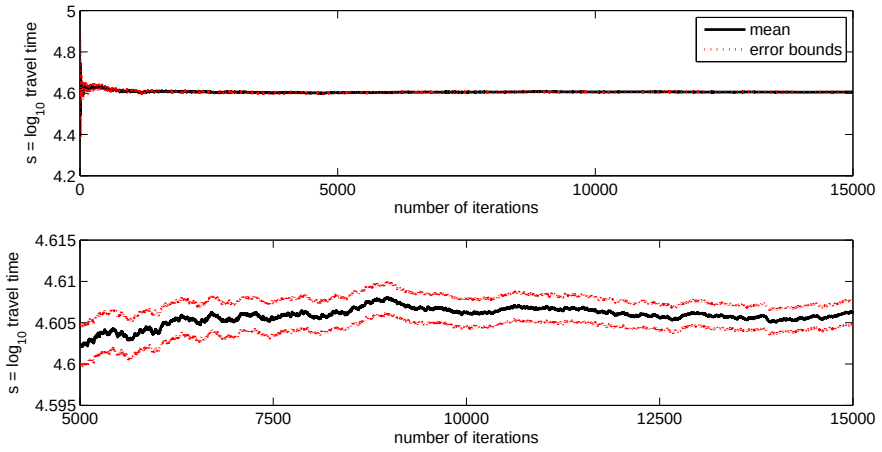

Figure 5.6: Running means of the estimated mean with statistical error bounds.

statistical error bounds on the mean. We see that the mean levels off after around 7500 iterations. This suggests that we can use the value of the estimated mean log travel time after 7500 evaluations of the code. This takes approximately 131 minutes and estimates the mean value with 0.04% error on the mean. The analysis can be repeated to estimate the mean for a small set of hyperparameters. The results are then used to build an emulator to estimate the mean travel time given any set of hyperparameters. The building of this emulator is discussed in the next chapter.

5.3.6 Accuracy of the cumulative distribution of the log travel time

We also need to calculate the accuracy of estimating the cumulative distribution of the log travel time from MC runs of the code. The sample cumulative distribution function $F_M(s)$ of a variable s can be used to estimate the cumulative distribution function $F(s)$. It is given by

$$\widehat{F(s)} \approx F_M(s) = \frac{\sum_{i=1}^{M} I(S_i \leq s)}{M}.$$

This is equivalent to random repetitions of a Bernoulli trial. Therefore, as M becomes large,$\widehat{F(s)} \to F(s)$. For finite M, the standard error of $\widehat{F(s)}$ is $\sqrt{p(1-p)/M}$.

For our computer model, we want to calculate $\widehat{F(s|\boldsymbol{\theta})}$ and the standard deviation of this estimate for a small number of s values. For ease of notation we will use $F(s) = F(s|\boldsymbol{\theta})$

and $\hat{F(s)} = \widehat{F(s|\boldsymbol{\theta})}$ from this point. Table 5.2 shows the mean estimate for $\widehat{F(4.5)}$, along with the standard deviation of that estimate for increasing number of runs of the code.

estimate of mean	s.d of estimate	number of runs	time, secs
0.36	4.80×10^{-2}	100	181
0.38	2.17×10^{-2}	500	597
0.366	1.52×10^{-2}	1000	1117
0.3802	6.87×10^{-3}	5000	5277
0.3872	4.87×10^{-3}	10000	10477

Table 5.2: Estimate of the mean value of $F(s)$, and standard deviations of these estimates for $s = 4.5$, given one set of hyperparameters and increasing number of runs of the code.

We see that the number of runs to calculate the estimated mean of $\widehat{F(s)}$ with small standard deviation is much smaller than the number of runs we needed to estimate the mean of log travel time given one set of hyperparameters. Figure 5.7 shows the running means of the estimated mean with error bounds for three values of s. The estimates for the larger and smaller values of s level out after around 1500 iterations, but for the middle value of $s = 4.6$, the running estimate levels out after around 2500 iterations of the code. This suggests that we need about 2000 runs of the code to provide a good

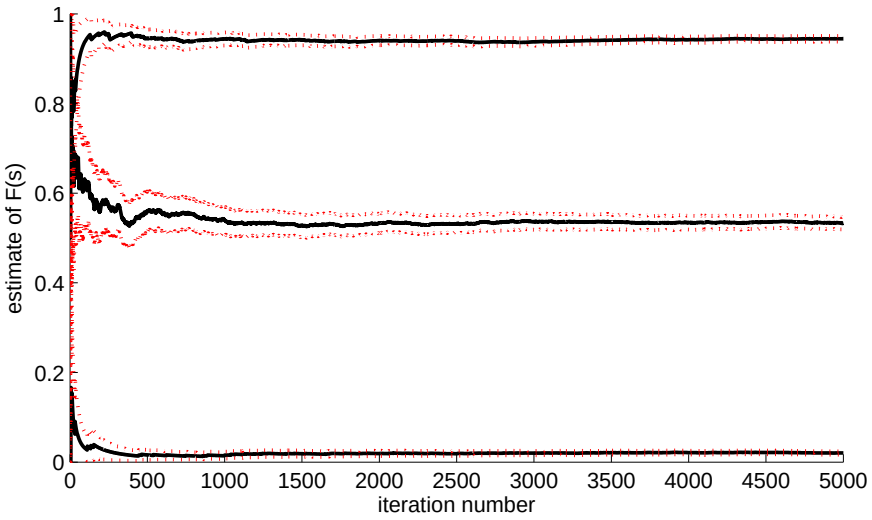

Figure 5.7: Running means of the estimated mean with error bounds for $s = 4.1, 4.6$ and 5.1.

estimate of $F(s)$ for each s, which is faster to converge than the mean of the computer model output. This could be due to the distribution function lying in $[0, 1]$ and the mean of s lying in $(0, -\infty)$. To run the code 2000 times for one set of hyperparameters takes approximately 38 mins, and provides the data to estimate $F(s)$ for any value of s. Table 5.3 shows the mean estimate of $\widehat{F(s)}$, and its standard deviation for 11 values of s, given one set of hyperparameters, and 2000 runs of the code.

s	estimate of mean	s.d. of estimate
3.5	0	0
4	0.005	2.23×10^{-3}
4.2	0.043	6.41×10^{-3}
4.4	0.216	1.30×10^{-2}
4.5	0.366	1.52×10^{-2}
4.6	0.509	1.58×10^{-2}
4.8	0.769	1.33×10^{-2}
5	0.905	9.27×10^{-3}
5.25	0.967	5.65×10^{-3}
5.5	0.992	2.82×10^{-3}
6	1	0

Table 5.3: Estimates of the mean values of $F(s)$, and standard deviations of these estimates for various values of s, given one set of hyperparameters and 1000 runs of the code.

We can repeat this for a small sample of hyperparmeters to obtain estimates of the mean $\widehat{F(s)}$, and standard deviation of the mean, for different values of s. The results can then be used to build an emulator to approximate the cumulative distribution function of the log travel times. This emulation will be described in the next chapter.

Chapter 6

Emulation of WIPP groundwater model

We now want to perform uncertainty analysis on the computer model created in Chapter 5. Whilst this computer model does not take a large amount of time to run once, carrying out uncertainty analysis using a Monte Carlo approach will be very computationally expensive. If we want to find out how much uncertainty there is in the output of the computer model, we could run the code for 50000 runs with a different set of hyperparameters for each run. Running the code 50000 times with a 80 by 120 mesh would take approximately 45 days, which is a long time for carrying out uncertainty analysis. We note that our model is not the most complicated model, and that to carry out the analysis on more complex models of groundwater flow may take longer. As well as taking a long time, carrying out the Monte Carlo analysis this way does not tell us about the sources of uncertainty in the model.

In order to find out whether the uncertainty in the output is due to the uncertainty in the hyperparameters of the log transmissivity field, or to the uncertainty in this field, we need to run the model a large number of times with each set of hyperparameters. This will give a sample of outputs for each set of hyperparameters, so we can calculate the uncertainty in the output when the hyperparameters are fixed, and when they are allowed to vary. To carry out this analysis for 1000 realisations of the log transmissivity field for each of 1000 hyperparameters (a total of 1000000 runs) would take approximately 12 days for a 80 by 120 mesh. For a larger sample of 5000 hyperparameters (or 5000000 runs),

this would take over 60 days. This is a much longer time for carrying out the analysis. Therefore we choose to build an emulator to statistically approximate the groundwater flow code. This much less computationally expensive emulator will then be used in place of the groundwater flow code when carrying out uncertainty analysis.

In order to build an emulator, we need to run the computer code with a small sample of inputs to provide data with which to build the emulator. The inputs to the groundwater flow code are the hyperparameters of the log transmissivity field. To investigate how the choice of mean function for the log transmissivity field affects the output of the groundwater flow model, we will emulate the groundwater flow code for the constant, linear and depth mean hyperparameters and compare the results. A sample of hyperparameters to be used to train the emulator can be generated using the distributions derived in Chapter 4. The code can then be run a number of times for each set of hyperparameters to give a sample of outputs for each of the hyperparameters. Once we have run the code and obtained our data we can then use this data to build an emulator.

The following section discusses the sampling plan for the hyperparameters, and how the data is obtained. Then we discuss the emulation of the groundwater flow model. We wish to emulate several different statistics for the computer model for our uncertainty analysis. First we will investigate the approximation of the mean log travel time, as this is the simplest emulator we can build. Once built, the emulator will provide us with the mean log travel time given any set of hyperparameters. We can use this emulator to provide us with information on how the mean log travel time varies as the hyperparameters vary. The next emulator we build will approximate the cumulative distribution function of the log travel times. This emulator is more complex, but will provide information about the entire distribution of the log travel time.

6.1 Running the WIPP groundwater flow model

To provide data for training an emulator to approximate the groundwater flow code, we need to run the code for a small sample of inputs. The general rule is 10 training points for each dimension of the input space (Loeppky et al. (2009)). This is due to approximation of the emulator to the true output becoming better as we include more points. However, as we increase the number of training points the correlation matrix

in the emulator becomes larger and more expensive to invert. Therefore, the use of 10 values for each input is used. We start by obtaining a latin hypercube sample of $\boldsymbol{\theta}$, from a sample space consistent with the distributions derived in Section 4.3. The method for this is described in more detail in the next Section. We then need to decide how many runs of the code need to be carried out for each set of hyperparameters in the sample, and run the code accordingly. The output from these runs will be used to create an emulator to statistically approximate the output from the WIPP groundwater flow model.

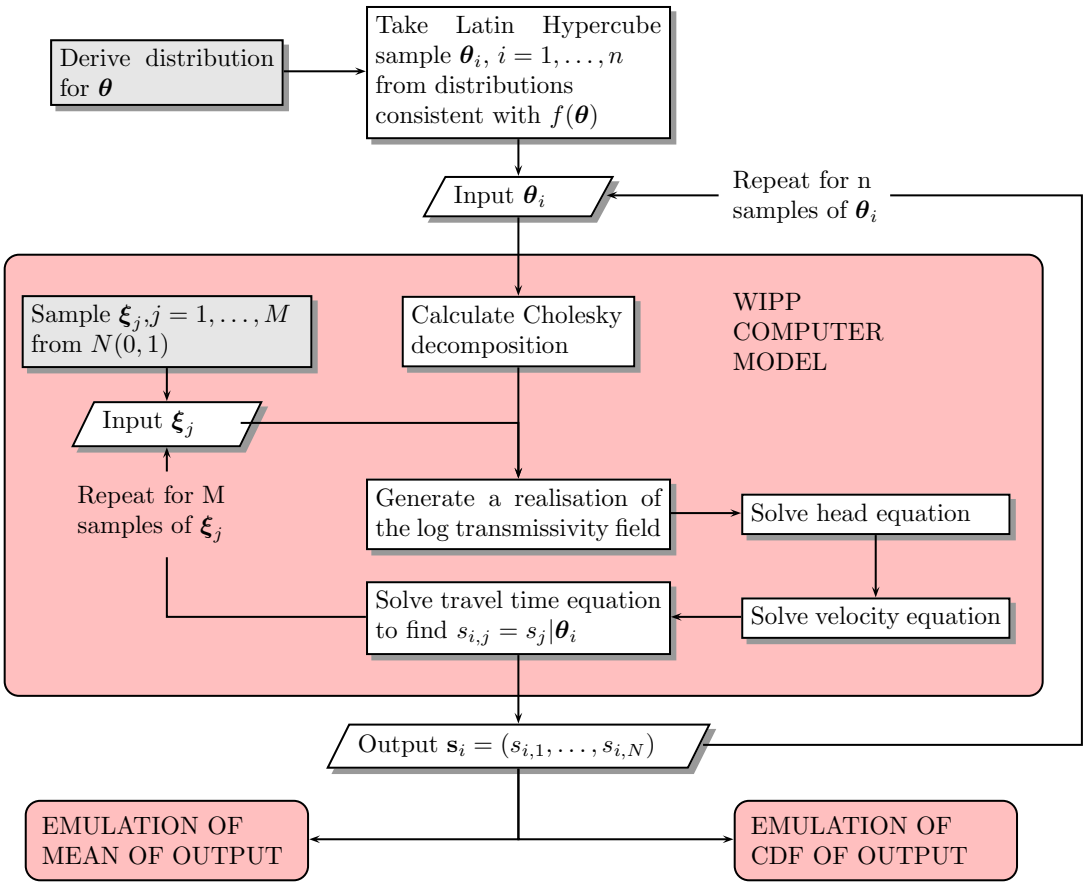

Figure 6.1: Flowchart showing the steps for running the WIPP computer model to obtain data for emulation. The steps in the emulation boxes are shown in Figures 6.2 (mean emulation) and 6.14 (cdf emulation).

Figure 6.1 shows a flow chart detailing the steps included in running the computer model to obtain the data for building the emulators for the mean and distribution function. The steps in the emulation boxes will be presented later.

6.1.1 Sampling from posterior distribution of the hyperparameters

For building the emulator, we want to have 10 values for each input dimension. In the constant mean model we have 3 inputs and so require a sample of 30 hyperparameters $\boldsymbol{\theta}_c$ from the input space $\mathcal{X}_c$. For the linear mean we have two more hyperparameters and so will need a sample of 50 hyperparameters $\boldsymbol{\theta}_l$ from the input space $\mathcal{X}_l$. The depth mean will require a sample of 40 hyperparameters $\boldsymbol{\theta}_d$ from the input space $\mathcal{X}_d$.

The sample from the derived distributions for $\boldsymbol{\theta}_c$, $\boldsymbol{\theta}_l$ and $\boldsymbol{\theta}_d$ must cover the entire range of values for each input. This is to make sure that the emulator can provide a good estimate of the output of the WIPP code. To represent the sample space, we take a Latin hypercube sample from the input space of $\boldsymbol{\theta}_c$, $\boldsymbol{\theta}_l$ and $\boldsymbol{\theta}_d$ as described in Subsection 2.1.2. The input space can be split into sections of equal probability to ensure that the areas of the input space with highest density of input values are represented with more samples than areas with lower density of input values. However, this may cause problems when building an emulator, because if the input values are too close together, the correlation matrix may become nearly singular and therefore nearly uninvertable.

Another way of generating the sample is to split the sample space uniformly into sections of equal size. This deals with the problem of the input values being too close together, but may not represent the sample by having too many values away from the areas of interest in the input space. A compromise between these two is to split the sample space of each input into quartiles given by their distribution, and then split each quartile up uniformly. This would make sure that the areas of interest are well represented, whilst data points will not be too close together. A Latin hypercube sample can then be taken using the sectioned sample space.

6.1.2 Sample of hyperparameters for the constant mean model

Table 6.2 shows the quartile ranges of the posterior distribution of all hyperparameters $\boldsymbol{\theta}_c = (\beta, \omega^2, \lambda)$ for the constant mean model. Using the distributions in Table 6.2

we use the maximin criterion to find a constrained Latin Hypercube sample with the maximum minimum distance between points in the sample space. We constrain the sample space so that the sampling design takes into account the correlation between ω^2 and λ discussed in Section 4.3.6. The constraints are needed as it does not make sense to run the computer model with values outside of the range of ω^2 and λ. This is because the computer model will produce incorrect or no results for some values outside this range. The constraints were found by looking at Figure 4.9 and finding two straight lines either side of the correlated sample. The constraint is therefore that the sample of ω^2 and λ must lie in the area between these two lines.

The sampling then takes the form of a rejection sampler, where any samples that include values of ω^2 and λ that do not meet the constraint are rejected and the next sample is taken. A sampling design for 30 samples of $\boldsymbol{\theta}_c$ is shown in Table B.1. This sampling scheme achieves the required result that larger values of ω^2 are paired with larger values of λ and smaller values of ω^2 are paired with smaller values of λ.

6.1.3 Sample of hyperparameters for the linear mean model

Table 6.1 shows the quartile ranges of the posterior distribution of all hyperparameters $\boldsymbol{\theta}_l = (\beta, \beta_x, \beta_y, \omega^2, \lambda)$ for the linear mean model. Using the distributions in Table 6.1 we again use the maximin criterion to find a Latin Hypercube sample of $\boldsymbol{\theta}_l$. A sampling design for 50 samples of $\boldsymbol{\theta}_l$ is shown in Table B.2. This sampling design also takes into account the correlation between ω^2 and λ in the same way as the sample design from the constant model.

6.1.4 Sample of hyperparameters for the depth mean model

Table 6.1 shows the quartile ranges of the posterior distribution of all hyperparameters $\boldsymbol{\theta}_d = (\beta, \beta_d, \omega^2, \lambda)$ for the depth mean model. Using the distributions in Table 6.3 we use the maximin criterion to give a sampling design for 40 samples of $\boldsymbol{\theta}_d$ shown in Table B.3. Again, this sampling design also takes into account the correlation between ω^2 and λ in the same way as the sample design from the constant model.

Hyper-parameter	Posterior distribution						
	mean	s.d.	2.5 %	25 %	median	75 %	97.5 %
β	-2.034	1.186	-4.319	-2.738	-2.039	-1.384	0.291
β_x	-2.69×10^{-4}	6.78×10^{-5}	-3.97×10^{-4}	-3.11×10^{-4}	-2.71×10^{-4}	-2.28×10^{-4}	-1.27×10^{-4}
β_y	-3.56×10^{-5}	4.99×10^{-5}	-1.33×10^{-4}	-6.28×10^{-5}	-3.59×10^{-5}	-5.90×10^{-5}	6.43×10^{-5}
λ	3001	3604	13.88	1445	2163	3842	12600
ω^2	1.956	1.676	0.862	1.230	1.521	2.104	6.366

Table 6.1: Posterior Distributions for the hyperparameters $\boldsymbol{\theta}_l$ including quartile ranges.

Hyperparameter	Posterior distribution						
	mean	s.d.	2.5 %	25 %	median	75 %	97.5 %
β	-4.934	1.645	-8.318	-5.76	-5.012	-4.151	-1.27
ω^2	6.479	4.284	2.111	3.537	5.076	8.007	18.23
λ	12390	8380	3098	6340	9635	15820	35340

Table 6.2: Posterior Distributions for the hyperparameters $\boldsymbol{\theta}_c$ including quartile ranges.

Hyper-parameter	Posterior distribution						
	mean	s.d.	2.5 %	25 %	median	75 %	97.5 %
β	-4.872	1.68	-8.509	-5.738	-4.776	-3.955	-1.468
β_d	-4.98×10^{-4}	3.91×10^{-3}	-9.11×10^{-3}	-2.88×10^{-3}	-1.45×10^{-4}	2.18×10^{-3}	6.47×10^{-3}
λ	10600	7393	815	4816	8714	15030	27790
ω^2	5.987	3.846	1.611	3.112	4.885	7.86275	15.85

Table 6.3: Posterior Distributions for the hyperparameters $\boldsymbol{\theta}_d$ including quartile ranges.

6.1.5 Obtaining data from the groundwater flow code

We can now evaluate the code using the input designs chosen in the previous section. The number of evaluations for each set of input hyperparameters depends on what we wish to emulate from the code. In the last chapter we analysed how many runs we would need for different statistics of interest. To give a good approximation of the mean log travel time, it was found that we would need approximately 5000 evaluations of the code for each set of hyperparameters. To approximate the cumulative distribution function at given values of the log travel time, it was found that we needed approximately 1000 evaluations of the code for each set of hyperparameters.

Therefore for each model, we run the groundwater flow code 1000 times for each corresponding sample set of the hyperparameters. We have $10n$ sets of hyperparameters, where n is the input dimension, and we evaluate each of these $10n$ sets, 1000 times giving $1000 \times 10n$ evaluations. Therefore, for the constant model, this involves evaluating the code 30000 times taking approximately 12.5 hours, whilst for the linear model, the code is evaluated 50000 times taking approximately 20.8 hours. The depth model will need 40000 evaluations of the code taking around 16.7 hours. It is much faster to evaluate the code for these small number of runs than to carry out the full Monte Carlo analysis which, as discussed at the start of this chapter, would take many days.

The runs of the code provide all of the data needed to estimate the statistics such as mean and cumulative distribution of our outputs. Using the estimated values, we can now go on to emulate the WIPP groundwater flow model, and use these emulators to perform uncertainty analysis of the code.

6.2 Emulating the mean log travel time

The simplest statistic to emulate for a stochastic computer model is the mean of the output. Therefore we start by emulating the mean of the log travel time, $s = \log t$. This assumes that the relationship between s and θ is approximately linear, and also ensures that the travel time will always be positive. We obtain a sample from the distribution for $\boldsymbol{\theta}$ to use to run the WIPP computer code to obtain data to build the emulator. The samples used for the three models are shown in Tables B.1, B.2 and B.3.

Since we have a stochastic model, we run the code many times for each design point. This will provide us with a sample of outputs from each input. The mean and variance of the mean for each sample can then be calculated as described in Chapter 5, and an emulator for the mean output can then be built with this information.

6.2.1 Mean emulator method

The method for emulating the mean of the stochastic computer model is as follows:

1. Run the computer model a number of times for each of the sets of hyperparameters of the input design to obtain a sample of travel times for each emulator input point.

   Mean emulator $\hat{\bar{S}} = \eta^{\boldsymbol{\theta}}$

   (a) Using the samples collected, calculate an estimate for the mean log travel time and the variance of this mean for each set of hyperparameters:

   $$\begin{aligned} \widehat{\mathrm{E}[s|\boldsymbol{\theta}_i]} &= \frac{1}{M}\sum_{j=1}^{M} \log t_{i,j}, \quad i = 1, \ldots, n, \\ \widehat{\mathrm{Var}[s|\boldsymbol{\theta}_i]} &= \frac{1}{M-1}\sum_{j=1}^{M} \left\{s_{i,j} - \mathrm{E}[s|\boldsymbol{\theta}_i]\right\}^2, \quad i = 1, \ldots, n. \end{aligned}$$

   (b) Build an emulator for $\hat{\bar{s}}$ using the estimated mean and variance of this mean calculated in step 1a and the formulation for emulating stochastic computer models, described in Chapter 2.3.

2. The mean emulator can then be used to estimate the mean log travel time for any set of hyperparameters.

A flowchart showing how this method fits in to the whole analysis is shown in Figure 6.2. We can use the mean emulator to investigate the main effects of each of the hyperparameters, to see which has the most impact on the mean of the output of the computer model. This can be done by fixing all hyperparameters except one and seeing how varying this hyperparameter affects the mean of the computer model output. We can also integrate the emulator over $\boldsymbol{\theta}$ to give a mean log travel time with all uncertainty in the model accounted for. However, this will give us only one value and so may not be very useful in giving us information about the model.

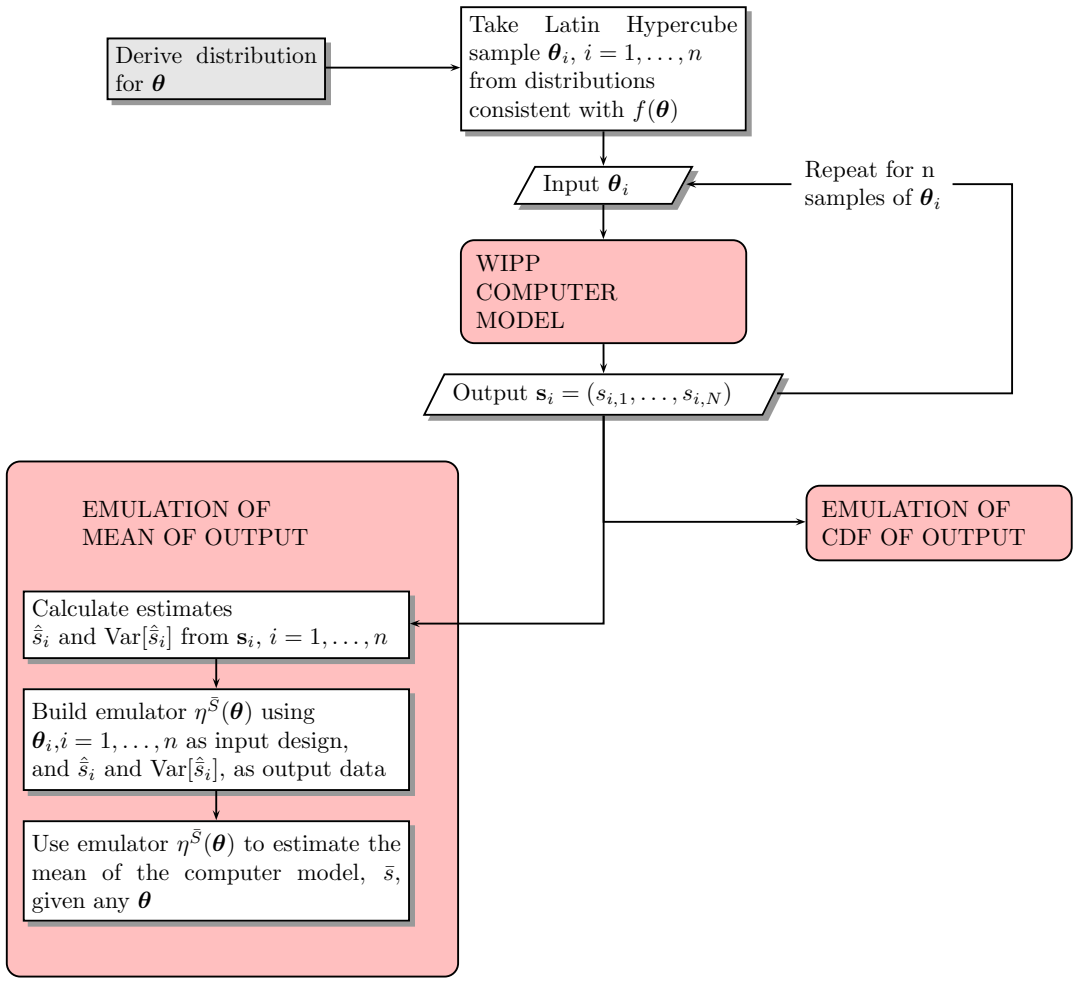

Figure 6.2: Flowchart showing the steps for emulating the mean of the WIPP computer model output. The steps in the WIPP computer model and cdf emulation boxes are shown in Figures 6.1 (WIPP computer model) and 6.14 (cdf emulation).

Before we emulate the mean log travel time, we have to make a transformation to the hyperparameters, which are the inputs to our emulator. Due to the hyperparameters varying by different orders of magnitude from each other, we scale the hyperparameters so that they all lie between 0 and 1. This stops the correlation matrix becoming singular. The hyperparameters can then be transformed back after emulation for analysis of the emulator output.

6.2.2 Emulated mean log travel time for constant mean model

We emulate the mean log travel time, $\bar{s}$ for the constant mean model using 30 sets of hyperparameters to train the emulator. From this emulator, we can investigate the effect that each hyperparameter has on the mean log travel time. We do this by fixing all hyperparameters except one to their mean values. The hyperparameter of interest is then allowed to vary within the range of its derived distribution, and the change in the mean log travel time can be plotted against it.

We start by looking at the mean hyperparameter, β of the constant model. Based on samples of the distribution gained using WinBUGs, the extreme values of this hyperparameter lie at around -13 and 1. The mean and 95% bounds of the derived posterior distribution for $\hat{\bar{S}} = \eta^{\boldsymbol{\theta}}$ are plotted against β in Figure 6.3.

We see that as β increases, the mean log travel time decreases. We would expect this effect since β is the mean of the log transmissivity field. As the log transmissivity increases, so does the speed of the groundwater flow, and therefore the travel time decreases. For the constant model, this decrease in the log travel time is of around 0.2 from the smallest value of β to the largest value. From Table 6.2, we see that 95% of the distribution for β lies between -8.3 and -1.3. In this range, the values of $\hat{\bar{s}}$ mostly stay the same, at around 4.62, and only decreases by around 0.02 between around $\beta = -3$ and $\beta = -1.3$. Therefore changing β has an effect on the mean log travel time, but this is not a large effect for the majority of values of β. We also note that the bounds on the emulator are very small in the middle region of the plot. This could be due to the training data for the emulator having very small variances.

Next we investigate the effects of varying the covariance hyperparameters λ and ω^2. Looking at Figures 6.4 and 6.5, it appears that these two hyperparameters both have

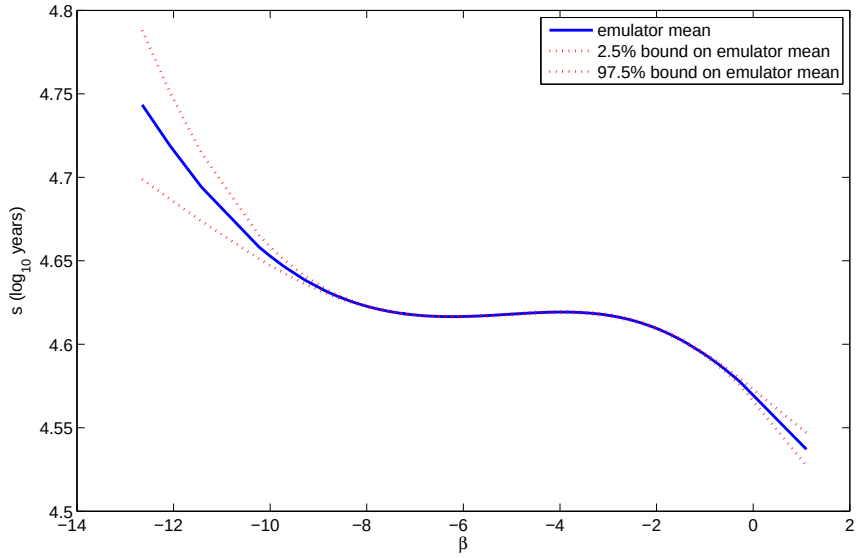

Figure 6.3: Effects of the constant transmissivity field mean hyperparameter, β, on log travel time.

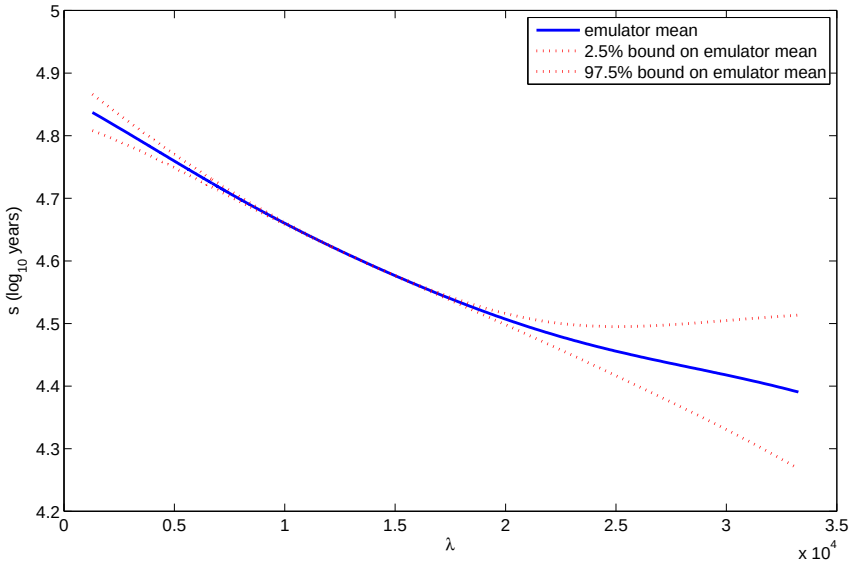

Figure 6.4: Effects of the constant transmissivity field correlation length hyperparameter, λ, on log travel time.

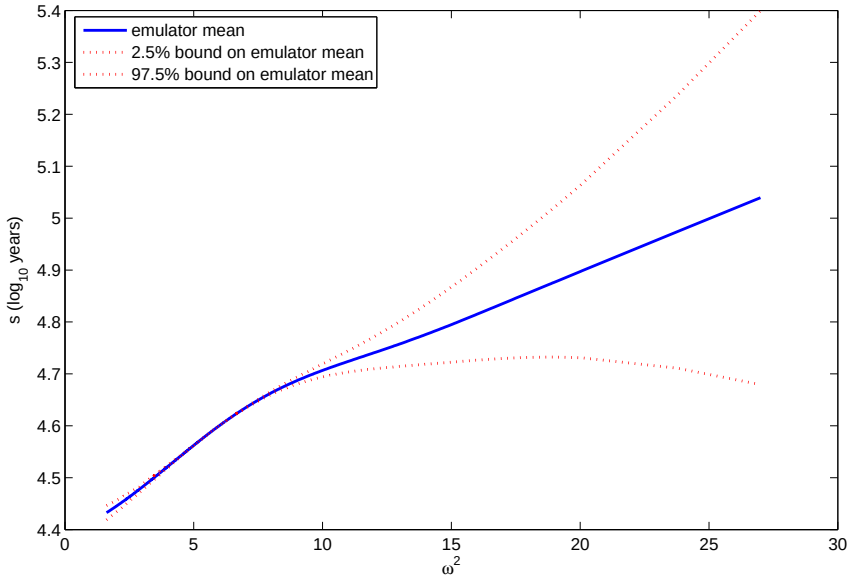

Figure 6.5: Effects of the constant transmissivity field mean variance hyperparameter, ω^2, on log travel time.

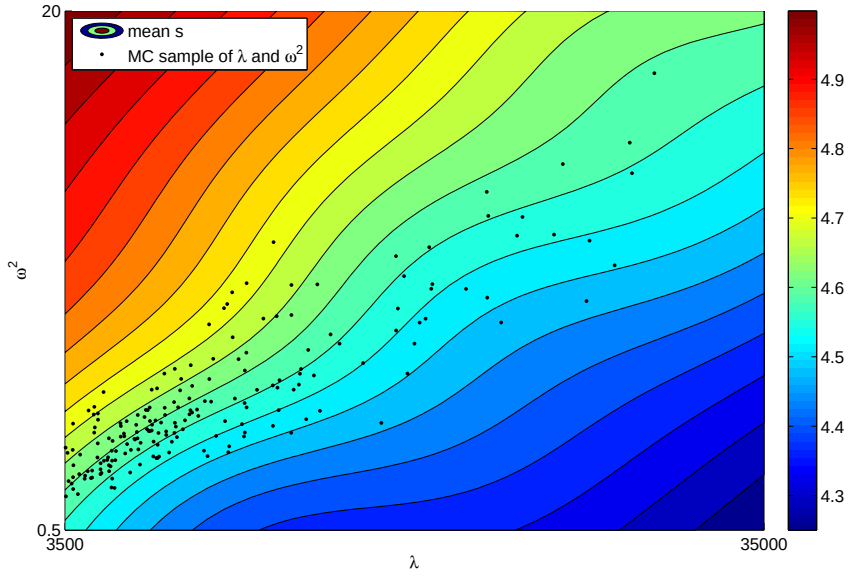

Figure 6.6: Effects of the constant transmissivity field covariance hyperparameters, λ and ω^2, on log travel time.

a larger effect on the mean log travel time than the mean hyperparameter β. For the full range of samples of λ the mean log travel time varies by about 0.45, and for ω^2 the values vary by about 1. This is more of an effect than β has. We also notice that both of these figures have very narrow emulator bounds for a section of the values of that hyperparameter. Narrow bounds like this are usually due to being close to training points. In this case, it could be due to the correlation between λ and ω^2, and the values at which one was fixed and the other allowed to vary in the figures.

Another problem with fixing λ and ω^2 separately at their mean values and then varying the other is that it does not provide us with a true representation of how the mean log travel time is affected. This is due to the correlation between λ and ω^2 as discussed in Section 4.3.6. We therefore plot the surface of $\hat{\bar{s}}$ as in Figure 6.6. From this figure we can see that the mean log travel time varies almost linearly in λ and ω. A small sample from the MC output of WinBUGs for the distribution of λ and ω^2 has also been plotted. The values of log travel time that this sample lies in is between around 4.45 and 4.7, with most of the values between 4.5 and 4.65. The effect of λ and ω^2 together is smaller than the effect on the mean values of log travel time given by λ and ω^2 separately, and is now similar to the effect of β on the mean. Therefore, for the constant mean model, the mean and covariance hyperparameters have a similar effect on the mean log travel time across their whole range. When we consider 95% of the distributions of the hyperparameters, the covariance hyperparameters have a larger effect.

6.2.3 Emulated mean log travel time for linear mean model

Again, we use the emulator to show the main effects of the hyperparameters on $\hat{\bar{s}}$. We fix all hyperparameters except one, then run the emulator for a sample of values of the unfixed value. We then see how the mean log travel time varies with each of the input hyperparameters. We start with the hyperparameters of the mean in the linear mean model, β, β_x and β_y.

The effect of β on the log travel time is shown in Figure 6.7. We see that, as for the constant model, the mean log travel time decreases as β increases. However, in this case, the effect that β has is much greater. The range of values for $\bar{s}$ is around 2.6 for the full range of β in our MC sample. From Table 6.1, the 95% range of values for β is between -4.3 and 0.3. For this range of β values, the values of $\hat{\bar{s}}$ vary between 5 and 4.5.

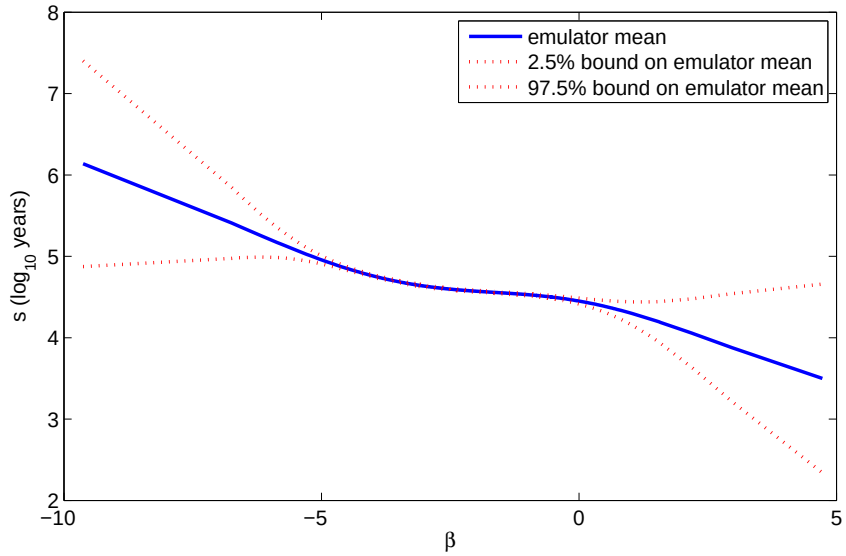

Figure 6.7: Effects of the linear transmissivity field mean hyperparameter, β, on log travel time.

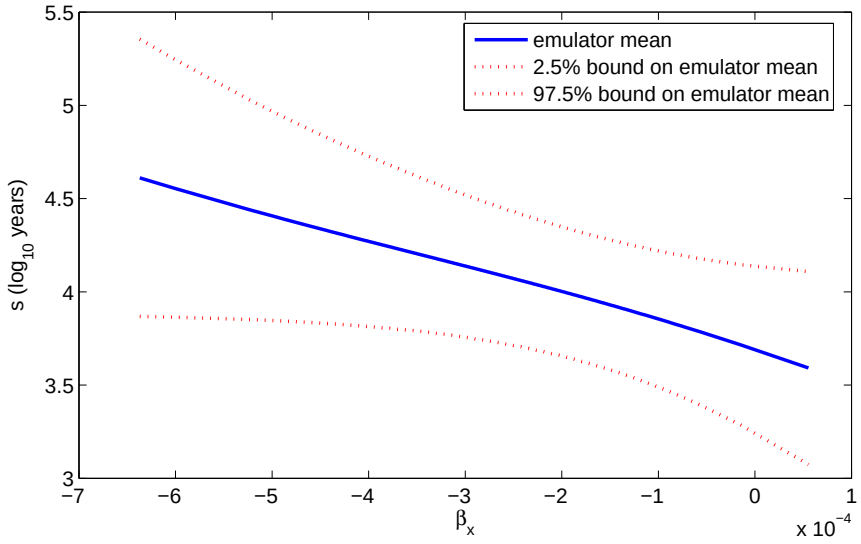

Figure 6.8: Effects of the linear transmissivity field mean hyperparameter, β_x, on log travel time.

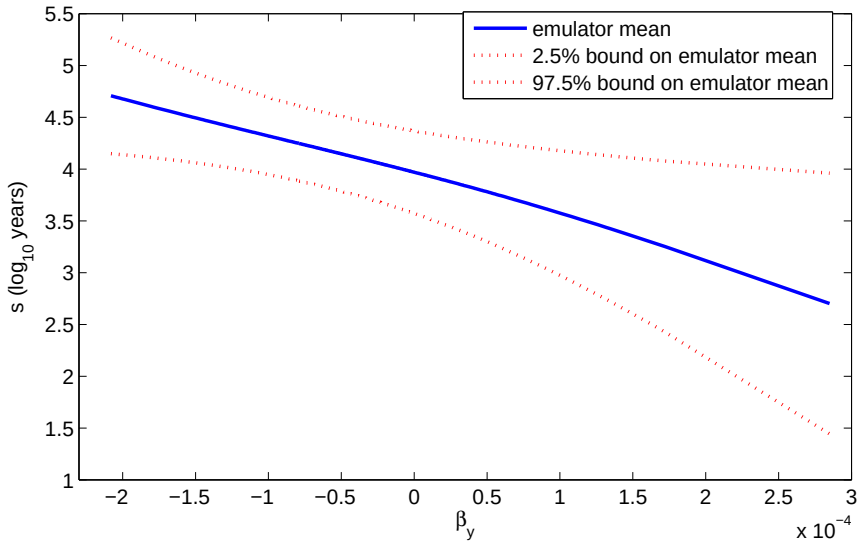

Figure 6.9: Effects of the linear transmissivity field mean hyperparameter, β_y, on log travel time.

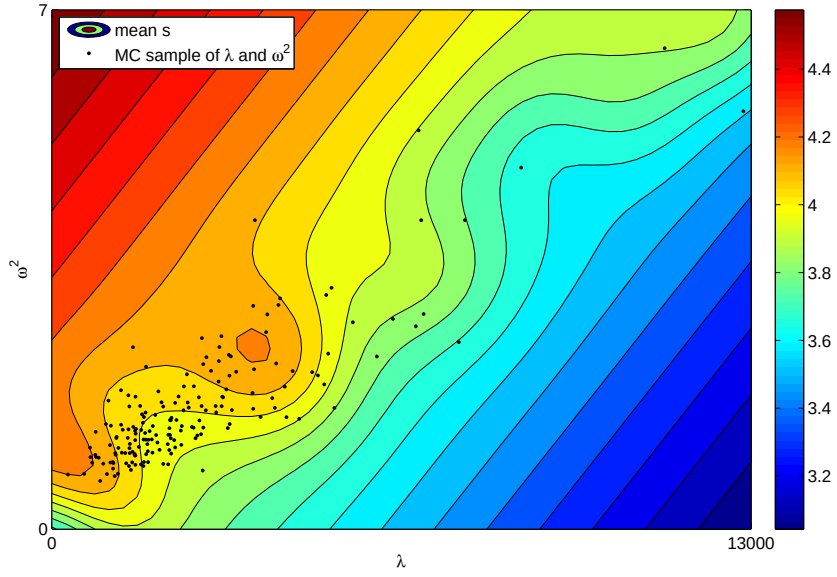

Figure 6.10: Effects of the linear transmissivity field covariance hyperparameters, λ and ω^2, on log travel time.

This decrease of 0.5 is larger than we found for the 95% values of beta in the constant model. Therefore the mean hyperparameter β has more of an effect on the mean log travel time in the linear model than in the constant model.

In the linear model, we have two more hyperparameters relating to the mean of the log transmissivity field. The effects of these are shown in Figures 6.8 and 6.9. Both of these hyperparameters have less of an effect across their range of values than β, but they still have an effect. We notice the same trend as for β: as β_x and β_y increase, the mean log travel time decreases. For β_x, the values of $\hat{\bar{s}}$ decrease by around 1, and for β_y the values of $\hat{\bar{s}}$ decrease by around 2. When we look at the 95% intervals of β_x and β_y, we find that the values of $\hat{\bar{s}}$ both change by around 0.5. This means that, for the 95% interval, β_x and β_y have a similar effect on the mean log travel time as β.

When we investigate the correlation parameters, we know that we cannot learn much by looking at these separately, so we consider them together. Figure 6.10 shows a contour plot of the mean log travel time for λ and ω^2. This time, we not not have as much of a linear trend in log travel time as we observed in the constant model. The range of log travel times relating to the MC sample of λ and ω^2 is between around 3.6 and 4.1. This is a larger range than for the constant model. It is similar to the effect of the linear mean hyperparameters, suggesting that for the linear model, all of the hyperparameters have a similar effect on the mean log travel time.

We notice that for the linear model, there is more variability in the mean log travel times than for the constant model. This may be because we are considering more hyperparameters in the linear model, and therefore have introduced more uncertainty into the model. However, for both models we see that the most important source of uncertainty is in the covariance hyperparameters. By reducing our uncertainty in these hyperparameters, we could reduce the effect that this uncertainty has on the mean output of the computer model.

6.2.4 Emulated mean log travel time for depth mean model

Finally, we emulate the mean log travel time for the depth mean model. We again want to investigate the effect that each hyperparameter has on the mean log travel time. We may expect that as in the other two models, the covariance hyperparameters have a

greater effect on the mean log travel time than the mean hyperparameters.

Figure 6.11 shows the effect of the mean hyperparameter β on $\hat{\bar{s}}$. We see the same effect of mean log travel time decreases as β increase as when investigating the constant and linear models. The effect on the mean log travel time is very similar to that produced when considering a constant model, with a small range of around 0.2 in values of $\hat{\bar{s}}$, and a 95% range of around 0.03.

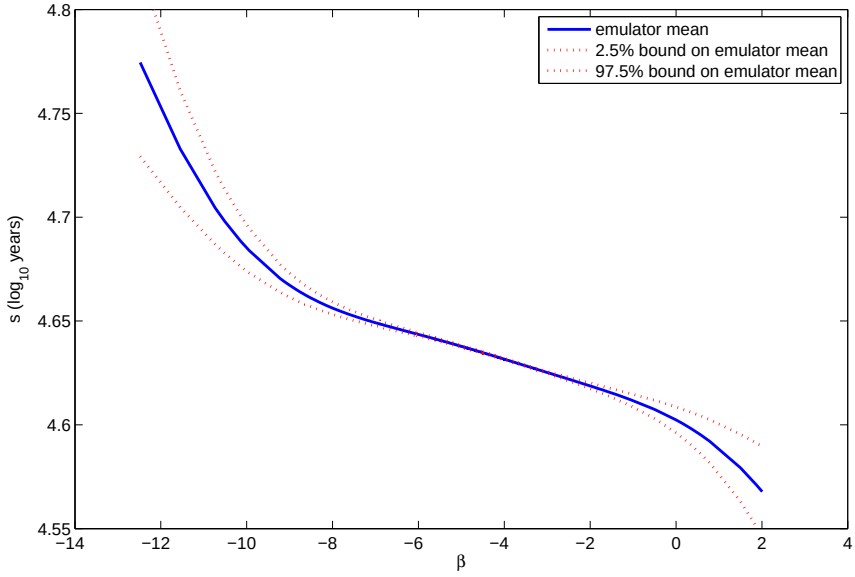

Figure 6.11: Effects of the depth transmissivity field mean hyperparameter, β, on log travel time.

The hyperparameter β_d indicates how much the depth affects the log transmissivity field, and therefore the log travel time. Investigating its effect on the mean log travel time may give us information on how much of an effect including the depth data has on the output of the computer code. We see the effect of β_d on the mean log travel time in Figure 6.12.

When $\beta_d > 0$, this leads to higher mean log transmissivity fields and so shorter travel times, which we see in the figure. When $\beta_d < 0$, the mean log transmissivity fields are lower, and so the travel times will be longer. This effect is shown on the graph for $-0.005 < \beta_d < 0$. When $\beta_d < -0.005$, the mean travel times reduce again, but not as quickly as for when $\beta_d > 0$. However, the general trend is as we would expect with

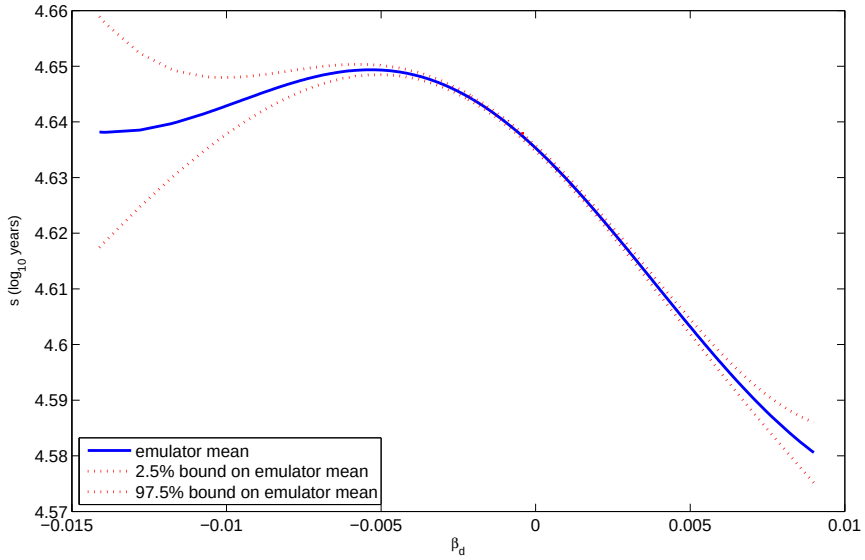

Figure 6.12: Effects of the depth transmissivity field mean hyperparameter, β_d, on log travel time.

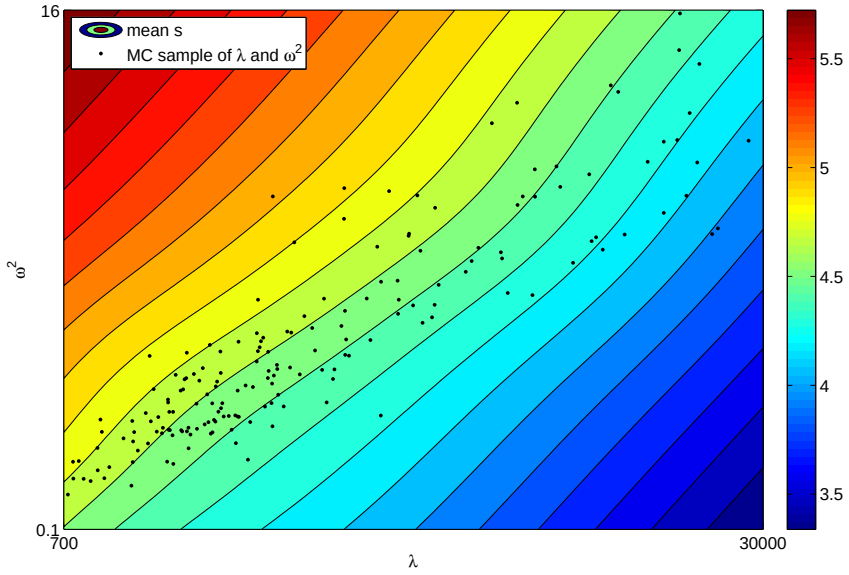

Figure 6.13: Effects of the depth transmissivity field covariance hyperparameters, λ and ω^2, on log travel time.

shorter travel times when $\beta_d > 0$ and longer travel times when $\beta_d < 0$. Varying the value of β_d does affect the mean log travel time, but not as much as β. Across the whole range of values for β_d given by the MC output, we see a change of around 0.13 in $\hat{\bar{s}}$. For the 95% range of β_d, there is a only a small range of about 0.05 in $\hat{\bar{s}}$. This suggests that including this hyperparameter into our model may not have much of an effect on the mean output.

We now investigate the effect of the covariance hyperparameters on the mean log travel time. Again, we consider the two hyperparameters together due to their correlation. Figure 6.13 shows the effect of varying λ and ω^2 on the mean log travel time. We see the same linear trend in the mean log travel time as when considering a constant or a linear model. The range of $\hat{\bar{s}}$ values covered by the MC sample is between around 4 and 5, with most of the values lying between about 4.3 and 4.8. If we compare this to the range of travel times estimated for β, the covariance hyperparameters have a greater effect on the travel times than β for the depth model. The range of travel times for the covariance hyperparameters of the depth model is larger than for the constant model, but similar to the linear model. This may be because of the addition of the depth term in the model for the log transmissivity field which introduces more uncertainty than is in the constant model.

6.3 Calculating the unconditional mean

The emulator can be used to calculate the unconditional mean for each of the three models. We can run the emulator with 10000 samples of $\boldsymbol{\theta}$ from the derived posterior distributions $f(\boldsymbol{\theta})$, to get a sample of log travel times s. We then integrate out $\boldsymbol{\theta}$ using Monte Carlo integration as described in Section 2.1.1. The results of this integration are shown in Table 6.4.

Model	Mean estimate for $\log_1 0$ travel time, $\hat{\mu}_s$	Variance of mean estimate, $\hat{\sigma}_s$
Constant mean	4.6128	0.0027
Linear mean	4.6141	0.7928
Depth dependent mean	4.6313	0.0315

Table 6.4: Unconditional estimates for the mean log travel time.

As seen in the previous few sections, the variance in the emulated estimates increases as the number of inputs are increased. The small variances are likely to be due to the large number of runs of the computer model used to approximate the mean log travel time for each sample of $\boldsymbol{\theta}$ in the inputs. We also notice that the three estimates are very similar in size, suggesting that there is not much difference between each of the three models. Therefore the simplest model may be the best as it reduces the complexity and the computational time, but does not lose as much accuracy when estimating the mean log travel time.

6.4 Emulating the cumulative distribution of the log travel time

Using an emulator to provide estimates of the mean log travel time gives limited information about the log travel time. To find out more about the distribution of travel times, we could estimate percentiles of the distribution. This could be carried out in the same way as the mean. In terms of radioactive waste disposal, useful percentiles are the 5th percentile or the 10th percentile of travel times. These shorter travel times are of interest, since regulations normally state a minimum number of years that the radioactive waste would reach the accessible environment.

It may also be useful to find out about the whole distribution of travel times, and not just the mean or percentiles. Therefore we consider emulating the cumulative distribution function (cdf) of the log travel time $F_S(s)$. The method we propose is more complicated than estimating the mean or percentiles, but we can use the samples of s obtained when emulating the mean, so there is no need to run the WIPP computer model again for this purpose. The distribution function captures all of the uncertainty in the model and so it provides information about the entire range of model outputs. We note that model uncertainty is not captured by the distribution function, but by comparing different models we can see if the choice of models affects the distribution function of the output.

Knowing about the whole range of outcomes, rather than just the mean, is important in the area of radioactive waste disposal as models are built to help with risk assessments. For the travel times, the shorter travel times are of interest as we wish to know the fastest time that radionuclides could enter the environment. Distribution functions of

the travel times were part of the evidence used by the US D.O.E. to satisfy the US Enviromental Protection Agency that the WIPP disposal site would comply with the guidance and standards for the management and disposal of radioactive waste (U.S. D.O.E. (2004)). Therefore, emulating the distribution function is useful in the field of radioactive waste disposal.

For this second emulation method, we are no longer approximating a computer model with an emulator, but instead using the emulation methodology to approximate a function $F(s)$. We can consider this function unknown in the same way that we can consider the output of a computer model unknown. Therefore we can treat it in a similar way, with a sample from the distribution of inputs and uncertain outputs arising from these inputs. In this case, s is no longer a control variable, but an input to the function. We use the sample of s from our earlier runs of the model, along with the fact that s is conditional on θ to estimate $F(s|\boldsymbol{\theta})$ for any $(s, \boldsymbol{\theta})$. Therefore, it is possible to build a conditional emulator to estimate $F(s|\boldsymbol{\theta})$ which can then be integrated over $\boldsymbol{\theta}$ to give an estimate for $F(s)$.

Using the output data, we can find an approximation to the conditional cdf of the travel time and the hyperparameters:

$$\hat{F}_{S|\Theta}(s|\boldsymbol{\theta}) = \sum_{i=1}^{N_{\text{runs}}} \frac{I(s_i \leq s)}{N_{\text{runs}}} = p, \tag{6.4.1}$$

where I is an indicator function equal to 1 when $s_i \leq s$ and 0 otherwise. This cdf is an estimate of $F_{S|\Theta}(s|\boldsymbol{\theta})$. We note that (6.4.1) is equivalent to random repetitions of a Bernoulli trial and therefore it has variance

$$\text{Var}[\hat{F}_{S|\Theta}(s|\boldsymbol{\theta})] = \frac{p(1-p)}{N_{\text{runs}}}. \tag{6.4.2}$$

As $N_{\text{runs}} \to \infty, \hat{F}_{S|\Theta}(s|\boldsymbol{\theta}) \to F_{S|\Theta}(s|\boldsymbol{\theta})$.

We would like to find the marginal distribution $F_S(s)$ using the integral

$$F_S(s) = \int F_{S|\Theta}(s|\boldsymbol{\theta}) f_{\boldsymbol{\theta}}(\boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{6.4.3}$$

However we only have an approximation to $F_{S|\Theta}(s|\boldsymbol{\theta})$, and a Monte Carlo sample from the distribution $f_{\boldsymbol{\theta}}(\boldsymbol{\theta})$; $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{N_{\text{sample}}}$. Therefore, we estimate the integral using Monte Carlo integration:

$$\hat{F}_S(s) = \frac{1}{N_{\text{sample}}} \sum_{i=1}^{N_{\text{sample}}} \hat{F}_{S|\Theta}(s|\boldsymbol{\theta}_i). \tag{6.4.4}$$

The sum (6.4.4) simplifies the problem of finding the marginal distribution for $F_S(s)$. However, we still do not have all of the information needed to be able to calculate it. From the output data, we can only calculate $\hat{F}_{S|\Theta}(s|\boldsymbol{\theta})$ for the small sample of $\boldsymbol{\theta}$ that the sample was obtained from. In order to calculate (6.4.4), we need to be able to find $\hat{F}_{S|\Theta}(s|\boldsymbol{\theta})$ for any value of $\boldsymbol{\theta}$.

Therefore we use another emulator, $\eta^{S|\Theta}(s|\boldsymbol{\theta})$, to approximate $\hat{F}_{S|\Theta}(s|\boldsymbol{\theta})$. This conditional cdf emulator $\eta(s|\boldsymbol{\theta})$ will be built using the same Latin hypercube sampling design for $\boldsymbol{\theta}$ as when we emulated the mean of the log travel times, with a sample of s values included. We do not know what values of s the cdf of $F_S(s)$ will include, so the s values are chosen to be evenly spread between 0 and 10 to give a wide range of values. The values of s are then simply shuffled and then added to the Latin hypercube design. Whilst this may not give us the best maximin design that we desire, it means that the computer code does not need to be run again for this problem. The output data for building this emulator will be the estimates of $\hat{F}_{S|\Theta}(s|\boldsymbol{\theta})$, given each $(s, \boldsymbol{\theta})$, and the variance of these estimates (6.4.2).

For any given $(s, \boldsymbol{\theta})$, the emulator will provide a distribution for $\eta^{S|\Theta}(s|\boldsymbol{\theta})$ with a mean $m^*(s|\boldsymbol{\theta})$ and variance $\hat{\sigma}^2 c^*(s, s|\boldsymbol{\theta})$. The emulator can be used as an approximation to $\hat{F}_{S|\Theta}(s|\boldsymbol{\theta})$, and the variance of the emulator gives the variance of this approximation. Therefore we can estimate (6.4.4) as

$$\begin{aligned} \hat{F}_S(s) &= \frac{1}{N_{\text{sample}}} \sum_{i=1}^{N_{\text{sample}}} \mathrm{E}\left[\eta^{S|\Theta}(s|\boldsymbol{\theta}_i)\right] \\ &= \frac{1}{N_{\text{sample}}} \sum_{i=1}^{N_{\text{sample}}} m^*(s|\boldsymbol{\theta}_i). \end{aligned} \tag{6.4.5}$$

To calculate (6.4.5) for a large number of s values may also be very computationally expensive. We choose to build a second cdf emulator $\eta^S(s)$ to approximate $\hat{F}_S(s)$. This marginal cdf emulator will allow us to approximate $\hat{F}_S(s)$ given any value of s. The sampling design to build this emulator will be a small sample of s values between 0 and 10 . The output data from this sample of inputs can be found using (6.4.5). We also need to know the variance of this output in order to build the marginal cdf emulator.

This variance is given by

$$\begin{aligned}\mathrm{Var}\left[\hat{F}_S(s)\right] &= \frac{1}{N_{\text{sample}}}\sum_{i=1}^{N_{\text{sample}}}\mathrm{Var}\left[\eta_j^{S|\Theta}(s|\boldsymbol{\theta}_i)\right] \\ &\quad + \frac{1}{N_{\text{sample}}}\sum_{i=1}^{N_{\text{sample}}}\{\mathrm{E}\left[\eta^{S|\Theta}(s|\boldsymbol{\theta}_i)\right] - \eta^{S|\Theta}(s|\boldsymbol{\theta}_i)\}^2 \\ &= \frac{1}{N_{\text{sample}}}\sum_{i=1}^{N_{\text{sample}}}\{\hat{\sigma}^2 c^*(s,s|\boldsymbol{\theta}_i)\} \\ &\quad + \frac{1}{N_{\text{sample}}}\sum_{i=1}^{N_{\text{sample}}}\{m^*(s|\boldsymbol{\theta}_i) - \eta^{S|\Theta}(s|\boldsymbol{\theta}_i)\}^2. \qquad (6.4.6)\end{aligned}$$

The emulator $\eta^S(s)$ can then be used to approximate $F_S(s)$ for any value of s, and therefore we can construct an approximate cumulative distribution function using this emulator. The cumulative distribution function provides us with information about the distribution of travel times, which we did not know when we considered just the mean of the log travel times.

6.4.1 Summary of method for emulating the cdf of the log travel time

The full method of approximating the cumulative distribution function of the log travel time can be summarised as follows:

1. From the data obtained from our runs of the code, we can build an emulator $\eta^{S|\Theta}(s|\boldsymbol{\theta})$ to approximate the cumulative distribution function $F_{S|\Theta}(s|\boldsymbol{\theta})$.

   Conditional CDF emulator $\eta^{S|\Theta}$

   (a) Using the output data from the runs of the computer code, calculate $\hat{F}_{S|\Theta}(s|\boldsymbol{\theta})$ using (6.4.1) for each $(s|\boldsymbol{\theta})$.

   (b) Calculate the variances of each $\hat{F}_{S|\Theta}(s|\boldsymbol{\theta})$ using (6.4.2).

   (c) Use the input design, estimates of $\hat{F}_{S|\Theta}(s|\boldsymbol{\theta})$ calculated in step 1a, and variances of these estimates calculated in step 1b to build a conditional cdf emulator $\eta^{S|\Theta}(s|\boldsymbol{\theta})$ to approximate $\hat{F}_{S|\Theta}(s|\boldsymbol{\theta})$ given any $(s|\boldsymbol{\theta})$

2. We then use this conditional cdf emulator to approximate the marginal distribution $\hat{F}_S(s)$ using (6.4.5). To carry out this sum for a large number of s values may also

be computationally expensive so we will evaluate (6.4.5) at a small sample of s values. Then we interpolate between these values using a second marginal cdf emulator $\eta^S(s)$ which will approximate the marginal distribution $\hat{F}_S(s)$ given any s.

Marginal CDF emulator η^S

(a) Using the conditional cdf emulator $\eta^{S|\Theta}$, calculate $\hat{F}_S(s)$ at the small sample of s using (6.4.5), and the Monte Carlo sample from the distribution $f_{\boldsymbol{\theta}}(\boldsymbol{\theta})$.

(b) Calculate the variances of each $\hat{F}_S(s)$ using (6.4.6).

(c) Use the input design, estimates of $\hat{F}_S(s)$ calculated in step 2a, and variances of these estimates calculated in step 2b to build a marginal cdf emulator $\eta^S(s)$ to approximate $\hat{F}_S(s)$ given any s

3. The marginal cdf emulator η^S can then be used to construct an approximate cumulative distribution function, and can be plotted against s.

A flowchart showing this method and how it fits into the whole analysis is shown in Figure 6.14. As when we emulated the mean, we scale all of the input variables of the emulator $\eta^{S|\Theta}$ to lie between 0 and 1 so that the correlation matrix does not become singular due to the different scales of the hyperparameters. We do not need to make this transformation when building the emulator η^S since the values of s are only in one dimension and evenly spread and so we do not have the same problems as when emulating a function of $\boldsymbol{\theta}$.

6.5 Emulation issues

Emulating the cumulative distribution function for the log travel times is more complicated than emulating the mean. Firstly, we have the problem that the emulator output is constrained to lie between 0 and 1. We can solve this problem by transforming the data from $[0, 1]$ to $\mathbb{R}$, then emulating and then transforming the emulator outputs back to $[0, 1]$. We need to find an appropriate function and its inverse to carry this out. This transform may have its own problems if the transformed data is not smooth. The emulator would then perform badly, since its main assumption is a smooth relationship

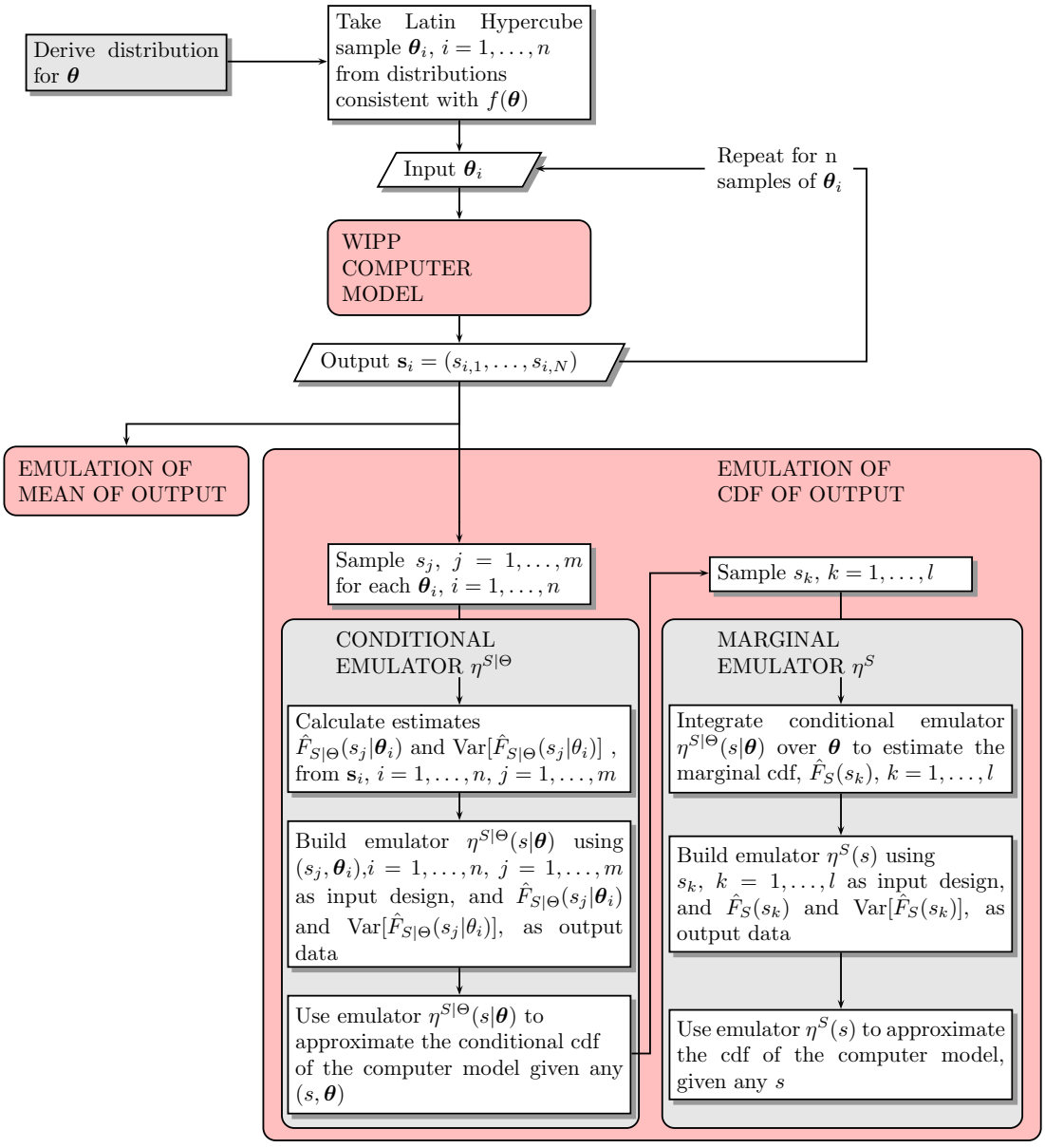

Figure 6.14: Flowchart showing the steps for emulating the distribution function of the WIPP computer model output. The steps in the WIPP computer model and mean emulation boxes are shown in Figures 6.1 (WIPP computer model) and 6.2 (mean emulation).

between the data. In this case, we try a different approach of using more design points. This will reduce the amount of the emulated surface that lies outside $[0, 1]$, but we will still have some values outside this space.

The second issue we need to consider is the scale of each of the hyperparameters when calculating the smoothing parameters. The algorithm to calculate the smoothing parameters by maximising the posterior mode depends very much on the starting values that it is given. Therefore, an appropriate scaling is needed when each hyperparameter varies from the others by many orders of magnitude. We also have the additional problem of the convergence time of the algorithm when the number of design points is large. In this case we can choose the smoothing parameters manually and use a cross validation procedure as described in Section 2.3.4 to improve our choice.

In the next few sections we will discuss how to deal with the problems of constraining the output, using more design points and estimating the smoothing parameters. We will illustrate these issues by considering the reduced problem of emulating the cdf with only one hyperparameter. In this example, we arbitrarily choose to fix all the hyperparameters except λ. This reduced example means that we are only emulating a two dimensional function of s and λ and so the resulting surface can be visually compared with that obtained from the Monte Carlo runs of the code at each training point λ_i. This surface can then be integrated over the distribution of λ at a small number of points to provide us with data for the second emulator. A second emulator is then built from the data collected from the first emulator. This second emulator provides us with an approximation for the cumulative distribution function of log travel times, with all uncertainty in λ and in the log transmissivity field integrated out.

6.5.1 Constraining the emulator output to lie in [0,1]

The main issue of emulating a cumulative distribution function is that when the emulators are evaluated with new values away from the design points, the emulator outputs may lie outside [0,1], which we do not to happen. To solve this problem, we can transform the data from [0,1] to $\mathbb{R}$, then emulate. The emulator is then run with a sample of inputs, then the output is transferred back onto [0,1]. This is achieved by using a scaled error function and its inverse in the following way.

For the first emulator $\eta^{S|\boldsymbol{\theta}}$, we let the data $\hat{F}(s|\boldsymbol{\theta}) = p$ where $p \in [0,1]$. Then to transform this data to $\mathbb{R}$, we can use the standard normal quantile function:

$$y = \Phi^{-1}(p) = \sqrt{2}\text{erf}^{-1}(2p-1), \quad p \in [0,1],$$

where

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt.$$

Once the function has been emulated in $\mathbb{R}$, the emulator outputs $y = \eta^{S|\boldsymbol{\theta}}$ can then be transformed back into [0,1] using the standard normal cumulative distribution function:

$$p = \Phi(y) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{y}{\sqrt{2}}\right)\right], \quad y \in \mathbb{R}.$$

To evaluate the estimate $\hat{F}_S(s)$ and its variance, the equations (6.4.5) and (6.4.6) need to be changed to reflect the transformation of the data. Noting that after the transformation, the emulator $\eta_j^{S|\Theta}(s|\boldsymbol{\theta}_i)$ is not an approximation to $F_{S|\boldsymbol{\theta}}(s|\boldsymbol{\theta})$, but to $G_{S|\boldsymbol{\theta}}(s|\boldsymbol{\theta}) = \sqrt{2}\text{erf}^{-1}(2F_{S|\boldsymbol{\theta}}(s|\boldsymbol{\theta}) - 1)$, the estimate (6.4.5) becomes

$$\hat{F}_S(s) = \frac{1}{N_{\text{sample}}} \sum_{i=1}^{N_{\text{sample}}} \left\{\frac{1}{2} + \frac{1}{2}\text{erf}\left(\frac{m^*(s|\boldsymbol{\theta}_i)}{\sqrt{2}}\right)\right\}. \tag{6.5.1}$$

The variance (6.4.6) then becomes

$$\text{Var}[\hat{F}_S(s)] = \frac{1}{N_{\text{sample}}} \sum_{i=1}^{N_{\text{sample}}} \left\{\frac{1}{2} + \frac{1}{2}\text{erf}\left(\frac{1}{\sqrt{2}}\left\{\sigma^2 c^*(s,s|\boldsymbol{\theta}_i) + \left[m^*(s|\boldsymbol{\theta}_i) - \eta^{(S|\Theta)}(s|\boldsymbol{\theta}_i)\right]^2\right\}\right)\right\}. \tag{6.5.2}$$

We can carry out the same transformation when building the second emulator η^S, but with data $\hat{F}(s) = p$. In this case we do not need to evaluate an integral, so the emulator outputs can be transformed back directly. These transformations of the data ensure that our approximation to the cumulative distribution function of the log travel time do not lie outside the interval [0,1].

One problem that can occur when emulating a transformed data set is that the transformed data may not be as smooth as the original data. We see this in our reduced example of emulating the surface $F(s|\lambda)$. In our example, we run the computer code with a sample of 40 values of λ from $f(\lambda)$ with all other hyperparameters fixed. From these runs we obtain a sample of 1000 log travel times, $\mathbf{s}|\lambda_\mathbf{i} = (s_1|\lambda_i, \ldots, s_{1000}|\lambda_i), i = 1, \ldots, 40$. We can construct approximate cdfs for each λ in our sample set from these samples.

These cdfs may not be smooth since they are estimated from a MC run of the code. This problem is solved by estimating the cdf using density estimation methods, and the MATLAB statistics toolbox contains a function `ksdensity` to enable us to do this. We also expect the cdfs to be smooth between values of λ_i as values of λ closer together should produce similar cdfs. This smoothness in each direction enables us to build an emulator $\eta^{S|\Lambda}(s, \lambda)$ to approximate $F(s|\lambda)$.

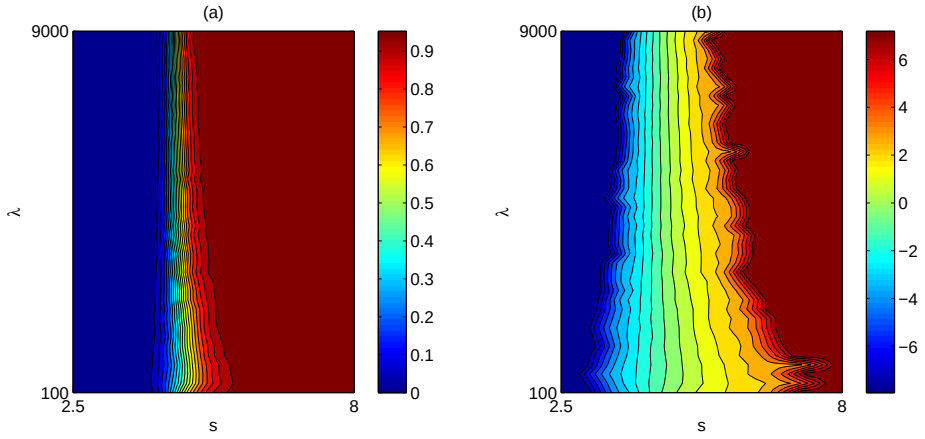

Figure 6.15: Contour plot $\hat{F}(s|\lambda)$ obtained from (a) MC runs of the code and (b) transformed contour plot $\hat{G}(s|\lambda)$.

Problems occur when we transform the surface from $[0, 1]$ to $\mathbb{R}$. The first difficulty is that we cannot deal with infinities when coding in MATLAB. Therefore the mapping of 0 to $-\infty$ and 1 to ∞ needs to be changed slightly. We get around this numerical issue by adding a small number to the data when $\hat{F}(s|\lambda) = 0$, and subtracting a small number to the data when $\hat{F}(s|\lambda) = 1$. The second issue is of the surface becoming rougher when it is transformed. Figure 6.15 shows the MC surface $\hat{F}(s|\lambda)$ and the transformed surface $\hat{G}(s|\lambda)$. We see that the transformed surface is much rougher than the original surface along the right-hand side of the transformed cdf in the λ direction. An emulator may have a problem approximating this rough surface. The original surface is much smoother in the λ direction. This roughness is only a problem when we are considering any of $\boldsymbol{\theta}$ as the inputs to the emulator. The second emulator that we build for s only will not have the same problems, since the cdfs in the s direction are much smoother.. Therefore, for the first emulator in this example, we need to find a way of constraining the surface to lie between 0 and 1 without transforming the data, if possible.

6.5.2 Using more design points

One way to improve the approximation of the emulator to the true function is to increase the number of design points. This can be restricted in the $\boldsymbol{\theta}$ direction, as it may be too computationally expensive to run the computer code many times for the added data points. However, since we have run the computer model a number of times for each hyperparameter, we are able to have as many design points in the s direction, for each design hyperparameter $\boldsymbol{\theta}$, as we wish, without having to obtain any more information from the computer model. We can therefore build the emulator with more data, and so the approximation to the function we are trying to emulate will be better.

For each set of hyperparameters, we have used the runs of the code to construct an approximate cumulative distribution function. Therefore, for any $(s, \boldsymbol{\theta}_i)$, we can find an approximation to $F(s|\boldsymbol{\theta}_i)$, along with an associated variance of $F(s|\boldsymbol{\theta}_i)$. For the example we have been looking at, we can increase the number of design points in the s direction and then estimate $F_{S|\Lambda}(s|\lambda_i)$, and the variance of this estimate $\mathrm{Var}[F_{S|\Lambda}(s|\lambda_i)]$, at each of these design points. This data can then be used to build the emulator. As we increase the number of points in the s direction, the approximation to the Monte Carlo surface will improve, and a smaller amount of the surface will lie outside of $[0, 1]$. We can see the effect of increasing the number of design points in the s direction in Figure 6.16. When only a few values of s are used in the input design, the emulator is not as good an approximation to the true surface as when more points are included. The emulator overshoots and some of the surface lies outside the interval $[0, 1]$. As the number of points is increased, the emulator is tied down in more places, and so we see a better approximation to the true surface and less of the surface outside the required region. Therefore, the more points we use, the better our approximation. However, we do not want to use too many points as the correlation matrix will become close to singular if two points are too close together, which is more likely to happen as we introduce more points.

In our example, we choose to sample 20 values of s for each λ_i. We therefore have 800 training points with which to build our emulator $\eta^{S|\Lambda}(s|\lambda)$. We build the emulator using the formulation for emulating stochastic models. The resulting mean surface is shown in Figure 6.17 along with the surface obtained using the MC data from the runs of the code.

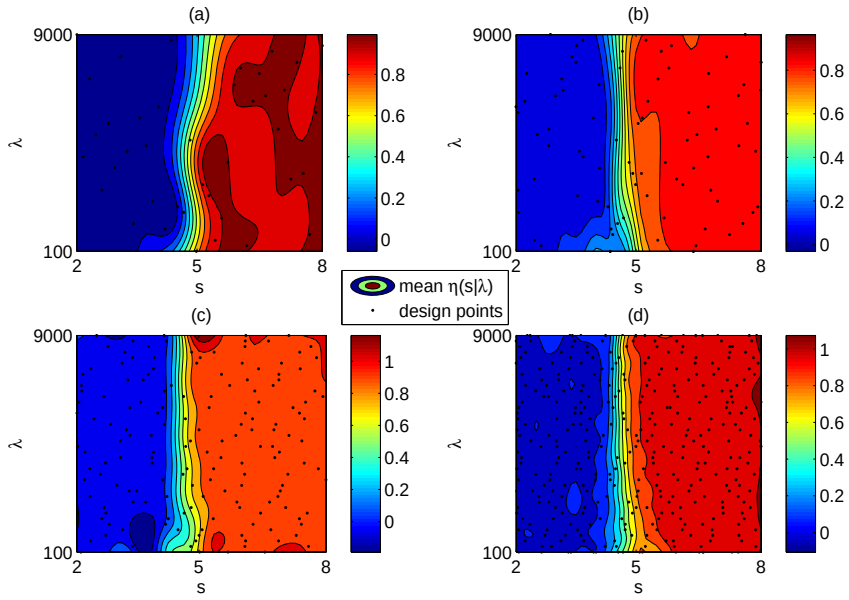

Figure 6.16: Mean emulated contour plots $\eta^{S|\Lambda}(s|\lambda)$ with (a) 1 sample of s for each λ_i (Latin Hypercube), (b) 2 samples of s for each λ_i, (c) 4 samples of s for each λ_i and (d) 8 samples of s for each λ_i.

We can see that the emulated conditional cdf is a good approximation to the cdf obtained from MC runs of the computer code. This is due to the large number of design points used to build the emulator. We can plot the residual error between the emulated and MC surfaces (Figure 6.18) to see how large the error between these values is.

We note that most of the errors between the mean of the emulator and the Monte Carlo output we are trying to emulate are close to zero. The larger errors are along the areas where there is a large amount of change in $F(s|\lambda)$. Where the surface changes by a large amount over a small spatial region, we may expect the emulator to not perform as well since the emulator is based on an assumption of smoothness. As the emulator smoothes through the data points, the emulator may slightly over, or under, shoot the function it is trying to approximate. We notice that the undershooting is mostly in the region where $F(s|\lambda) = 0$, and the overshooting is in the region where $F(s|\lambda) = 1$. Using more design points in our emulator reduces this effect, but does not stop it entirely. We need to be careful and check the errors are small. When we integrate over λ to build the second emulator, the errors in the first emulator average out to be very small. These

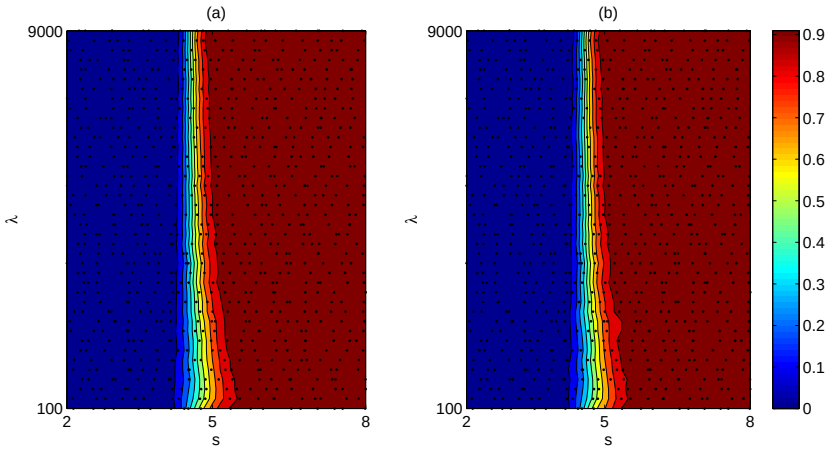

Figure 6.17: Contour plot $\hat{F}(s|\lambda)$ obtained from MC runs of the code (a) and mean emulated contour plot $\eta^{S|\Lambda}(s|\lambda)$ (b).

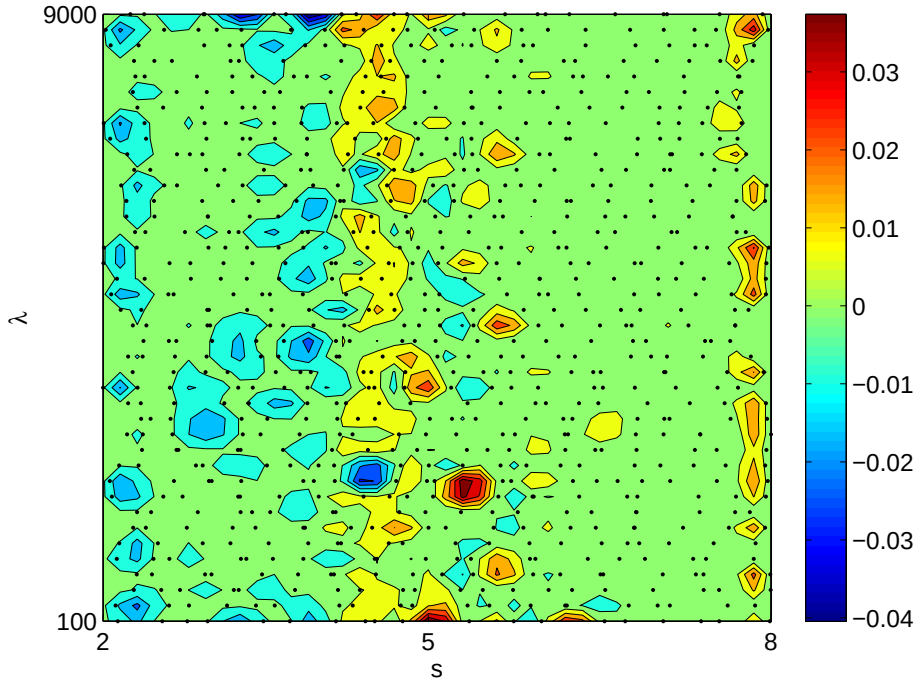

Figure 6.18: Residual error contour plot between $\hat{F}(s|\lambda)$ obtained from MC runs of the code and mean emulated surface $\eta^{S|\Lambda}(s|\lambda)$.

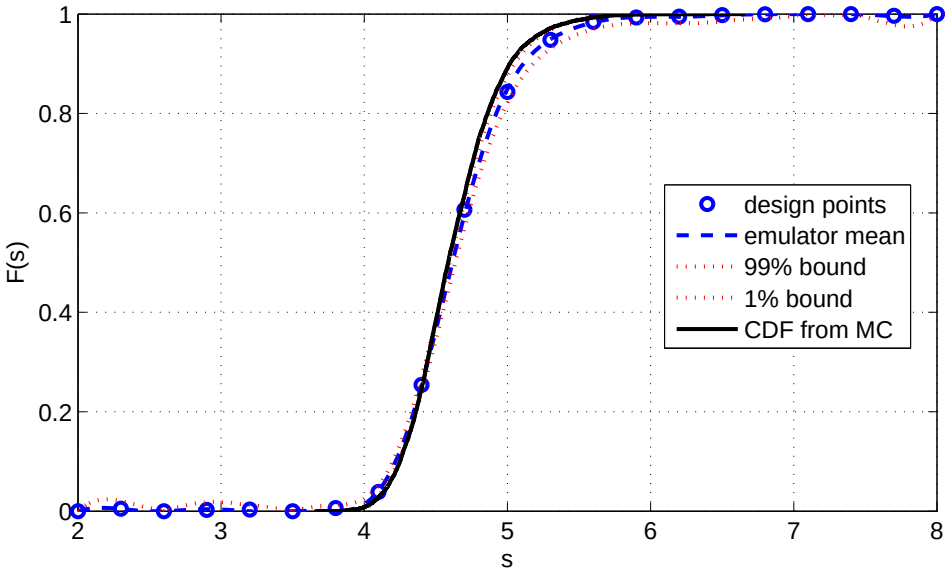

Figure 6.19: CDF for log travel times with all uncertainty in λ and $Z(\mathbf{x})$ integrated out (all other hyperparameters fixed).

slight errors can be fixed by adjusting the data $\hat{F}_S(s)$ to lie within the interval $[0, 1]$.

To complete our example of emulating the cdf of s with all hyperparameters fixed, we integrate the emulator over $f(\lambda)$. To do this for all values of s would be very computationally expensive. Therefore, we choose a small sample of s and then calculate an estimate $\hat{F}(s)$, and the variance of $\hat{F}(s)$, by integrating the emulator over λ at each of these s values. We then use this information to build a second emulator which approximates $F(s)$ for any s. This emulation of the cdf is shown in Figure 6.19, along with a cdf estimated using MC methods on the output of the computer model. We see that the emulator is a good approximation to the MC output. The time taken to calculate the cdf using the emulator was much faster than when using Monte Carlo methods. Including running the code to obtain data, the emulator took around 18 hours to provide an approximation to the distribution function, whereas the MC sample took around 10 days to obtain.

6.5.3 Using a different prior mean for the emulator

Whilst increasing the number of design points in the s direction provides a better approximation to the distribution function, it also increases the computational cost of building the emulator. As we increase the dimensions of the problem to include all of the hyperparameters $\boldsymbol{\theta}$, we may still have problems with the emulator generating values outside the range $[0, 1]$, and tending towards the linear prior,

$$\mathrm{E}[\eta(\boldsymbol{\theta}, s) \mid \boldsymbol{\beta}] = \mathbf{h}(\boldsymbol{\theta}, \mathbf{s})^T \boldsymbol{\beta}, \tag{6.5.3}$$

away from the design points. This leads us to consider the use of an alternative prior mean for the emulator.

A more suitable prior mean would be a monotonic, non-linear function with the same properties as a distribution function. The simplest prior mean to use would therefore be the Gaussian distribution function:

$$\mathrm{E}[\eta(s) \mid \mu, \tau] = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{s-\mu}{\sqrt{2}\tau}\right)\right]. \tag{6.5.4}$$

We no longer have a regression problem, where we had to estimate the regression coefficients, $\boldsymbol{\beta}$. Instead, we need to estimate μ and τ of this function in order to determine the prior mean. Since these two parameters relate to the mean and standard deviation of a distribution function, we propose that they be estimated from the samples of log travel times collected from the computer model at each of the design points. We see the estimation of these parameters from these output samples as no different to obtaining estimates for $\boldsymbol{\beta}$ for the previous mean function. We also note that, for each design point $\boldsymbol{\theta}$, the estimates $\hat{\mu}$ and $\hat{\tau}$ may vary and so we need to determine $\hat{\mu}(\boldsymbol{\theta})$ and $\hat{\tau}(\boldsymbol{\theta})$. The prior mean, (6.5.4), will therefore become

$$\begin{aligned} \mathrm{E}[\eta(s|\boldsymbol{\theta}) \mid \mu = \hat{\mu}, \tau = \hat{\tau}] &= \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{s-\hat{\mu}(\boldsymbol{\theta})}{\sqrt{2}\hat{\tau}(\boldsymbol{\theta})}\right)\right] \\ &= h(s|\boldsymbol{\theta}). \end{aligned} \tag{6.5.5}$$

To demonstrate the effect of each of the two prior means on the emulator output, we can use the values of s obtained from one λ and compare the use of the two prior mean functions in estimating the distribution function(Figure 6.20). We see that away from the design points, the functions revert back to the prior mean. Whilst this is not

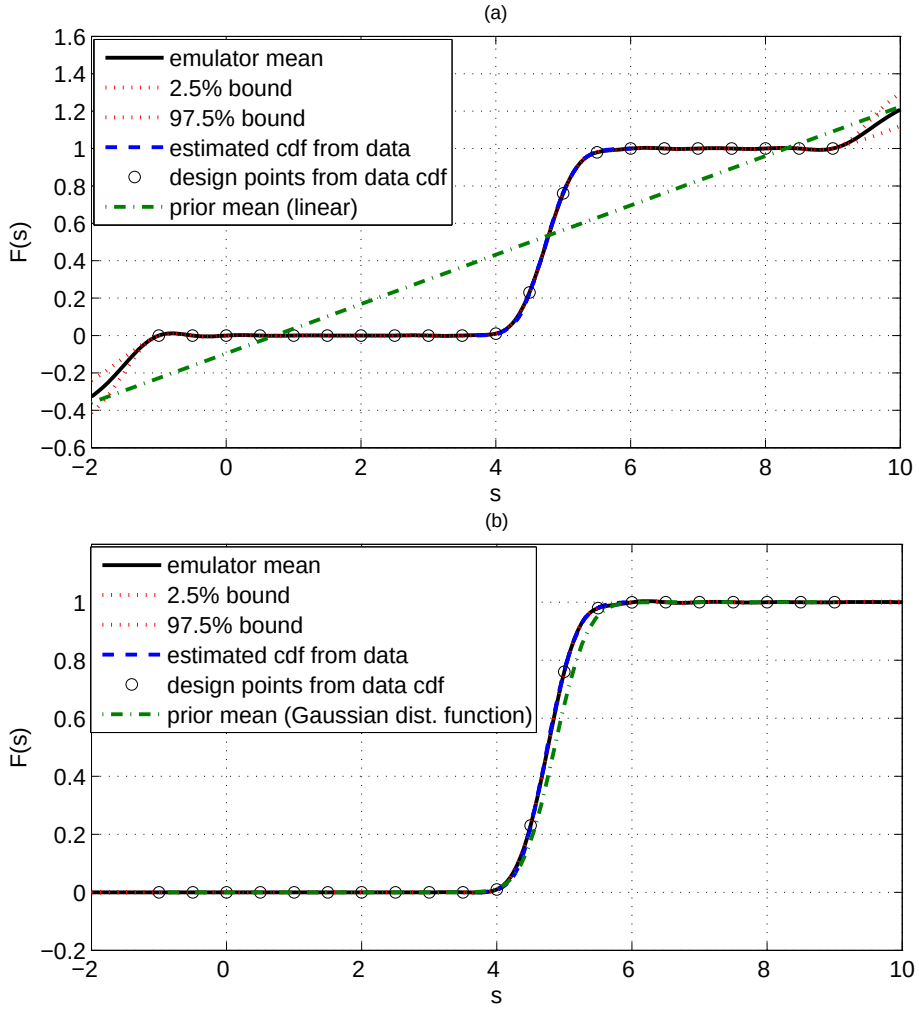

Figure 6.20: Effect of prior means away from data points. Plot (a) shows a linear prior mean and plot (b) shows a Gaussian distribution function as the prior mean.

too much of a problem in one dimension, when fitting the surface in more than one dimension, it is more important that the prior gives a good fit to the data as there is more space between data points. For the linear prior mean, this means that much of the surface may not be close to the true surface, unless a large number of design points are used. For the Gaussian distribution function prior, with mean and variance estimated from the data as described above, much more of the surface will be closer to the true surface, and fewer data points will be required. The emulator should then provide a better approximation to the true surface between the data points.

The estimates $\hat{\mu}(\boldsymbol{\theta})$ and $\hat{\tau}(\boldsymbol{\theta})$ are obtained from mean and standard deviation of the sample at each $\boldsymbol{\theta_i}, i = 1, \ldots, M$:

$$\begin{aligned} \hat{\mu}(\boldsymbol{\theta}_i) &= \frac{1}{M}\sum_{j=1}^{M} s_j|\boldsymbol{\theta}_i, \\ \hat{\tau}(\boldsymbol{\theta}_i) &= \sqrt{\frac{1}{M-1}\sum_{j=1}^{M}(s_j|\boldsymbol{\theta}_i - \hat{\mu}(\boldsymbol{\theta}_i))^2}. \end{aligned} \tag{6.5.6}$$

The standard errors of $\hat{\mu}(\boldsymbol{\theta}_i)$ and $\hat{\tau}(\boldsymbol{\theta}_i)$, ϵ_M and ϵ_S respectively, can be found using (Lehmann and Casella (1998))

$$\begin{aligned} \epsilon_M(\boldsymbol{\theta}_i) &= \frac{\hat{\tau}(\boldsymbol{\theta}_i)}{\sqrt{M}}, \\ \epsilon_S(\boldsymbol{\theta}_i) &= \frac{\hat{\tau}(\boldsymbol{\theta}_i)}{\sqrt{2(M-1)}}. \end{aligned} \tag{6.5.7}$$

In order to use $h(\mathbf{s}|\boldsymbol{\theta})$ in the emulator equations, we will need to know $\hat{\mu}(\boldsymbol{\theta})$ and $\hat{\tau}(\boldsymbol{\theta})$ at any point $\boldsymbol{\theta}$, not just at the design points $\boldsymbol{\theta}_i$. Since these are estimates for use in our prior function, it is not too important to know the exact value at any point, so a smooth interpolation can be used to obtain the general trend of these parameters throughout the $\boldsymbol{\theta}$ space. Therefore, we can use a spline or kriging interpolator to approximate these values.

Since we have a different mean, we need to reformulate the emulator equations we derived in Section 2.3. Here we have $(s, \boldsymbol{\theta})$, instead of t. To simplify the notation in the following equations, we will use $\mathbf{s} = (s|\boldsymbol{\theta})$. We start by providing a new prior for the data vector $\mathbf{y}$. Instead of the distribution (2.3.6), we now have

$$\mathbf{y}|\sigma^2, \mu, \tau \sim N(\mathbf{h}, \sigma^2 A), \tag{6.5.8}$$

where $\mathbf{h}^T = (h(\mathbf{s}_1), \ldots, h(\mathbf{s}_n))$, and σ^2 and A are the same as before. This data is then used to update the prior distribution of η in the same way as in Section 2.3.2 to give

$$\eta(.)|\mathbf{y}, \sigma^2, \mu, \tau \sim N\left(m^*(.), \sigma^2 c^*(.,.)\right), \tag{6.5.9}$$

where

$$\begin{aligned} m^*(\mathbf{x}) &= h(\mathbf{x}) + \mathbf{t}(\mathbf{x})^T A^{-1}(\mathbf{y} - \mathbf{h}), \\ c^*(\mathbf{x}, \mathbf{x}') &= c(\mathbf{x}, \mathbf{x}') - \mathbf{t}(\mathbf{x})^T A^{-1} \mathbf{t}(\mathbf{x}') \\ \mathbf{t}(\mathbf{x})^T &= (c(\mathbf{x}, \mathbf{x}_1), \ldots, c(\mathbf{x}, \mathbf{x}_n)). \end{aligned}$$

The terms $\mu = \hat{\mu}, \tau = \hat{\tau}$ will be ignored from this point on, as we are estimating these from the data $\mathbf{y}$. We will now concentrate on removing the conditioning on σ^2, which cannot be specified beforehand. As in Section 2.3.3, we want to obtain the posterior distribution $\eta(.)|\mathbf{y}$ by integrating out σ^2 from

$$f(\eta(.), \sigma^2|\mathbf{y}) = f(\eta(.)|\mathbf{y}, \sigma^2) f(\sigma^2|\mathbf{y}).$$

The first term on the right hand side is given by (6.5.9). We can find the second term up to proportionality using

$$f(\sigma^2|\mathbf{y}) \propto f(\mathbf{y}|\sigma^2) f(\sigma^2).$$

We use a similar weak prior mean for σ^2 as before; $f(\sigma^2) \propto \sigma^{-2}$. Combining this with the likelihood function for $\mathbf{y}$ obtained from (6.5.8), we have the inverse gamma distribution:

$$f(\sigma^2|\mathbf{y}) \propto \left(\sigma^2\right)^{-\frac{n}{2}-1} \exp\left\{\frac{1}{2\sigma^2}(n - q - 2)\hat{\sigma}^2\right\}, \tag{6.5.10}$$

where

$$\hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{h})^T A^{-1}(\mathbf{y} - \mathbf{h})}{n - q - 2}. \tag{6.5.11}$$

Combining the prior (6.5.9) with (6.5.10), we get the normal inverse gamma distribution:

$$f(\eta(.), \sigma^2|\mathbf{y}) \propto \left(\sigma^2\right)^{-\frac{(n+2)}{2}-1} \exp\left\{-\frac{1}{2\sigma^2}\left[c^*(.,.)^{-1}\left(\eta(.) - m^*(.)\right)^2 + (n - q - 2)\hat{\sigma}^2\right]\right\}.$$

Integrating out σ^2, we obtain

$$\frac{\eta(\mathbf{s}) - m^*(\mathbf{s})}{\hat{\sigma}\sqrt{c^*(\mathbf{s}, \mathbf{s})}} \sim t_{n-q}. \tag{6.5.12}$$

We can use this new formulation to provide us with a distribution for the output of the computer code with a prior mean given by (6.5.5).

To illustrate the use of the Gaussian distribution function as a prior, we now go back to the two dimensional example of emulating the surface $\hat{F}(s|\lambda)$. We will estimate $\mu(\lambda)$ and $\tau(\lambda)$ and their standard errors from the output data. For each $\lambda_i, i = 1, \ldots, 40$, we have a sample of 1000 log travel times. For each of these samples, we can estimate values of μ and τ. Then, we use spline interpolation to predict the values of $\hat{\mu}$ and $\hat{\tau}$ between the data points. The results of this are shown in Figure 6.21. In plot (a) the

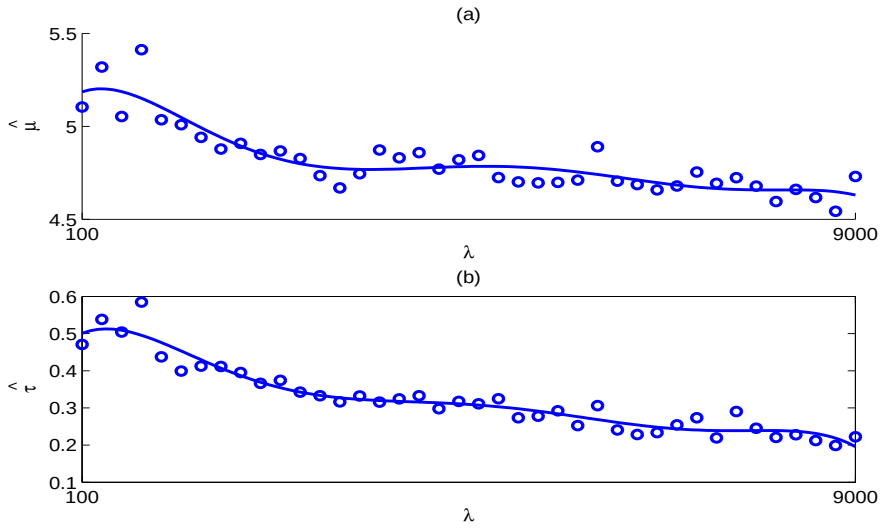

Figure 6.21: Interpolation of (a) $\hat{\mu}$ and (b) $\hat{\tau}$. Circles indicate the estimated values at the design points λ.

circles are the estimates of $\hat{\mu}$ given from the each of the computer code runs using λ. The interpolation through these estimates is given by the solid line $\hat{\mu}$. Since these values are estimated from data, and have an associated standard error, the interpolation does not pass through the estimated values, but provides the general trend of $\hat{\mu}$ to inform our prior mean. We see that the interpolated line is similar in shape to the shape in the contour plot in Figure 6.17(a) obtained from MC runs of the code. Plot (b) in Figure 6.21 shows the estimated and interpolated values for $\hat{\tau}$. Again, the interpolation does not pass through the data points, but provides a smooth approximation to $\hat{\tau}$ given the estimated values and their standard errors. These approximations can then be used in the emulator equations for the prior mean function.

Given the estimated values of $\hat{\mu}$ and $\hat{\tau}$, we can repeat the emulation of $F(s|\lambda)$ using our new prior mean. Figure 6.22 shows the mean of the emulator $\eta^{S|\Lambda}(s|\lambda)$ using increasing numbers of design points. We can compare these plots with those in Figure 6.16 where

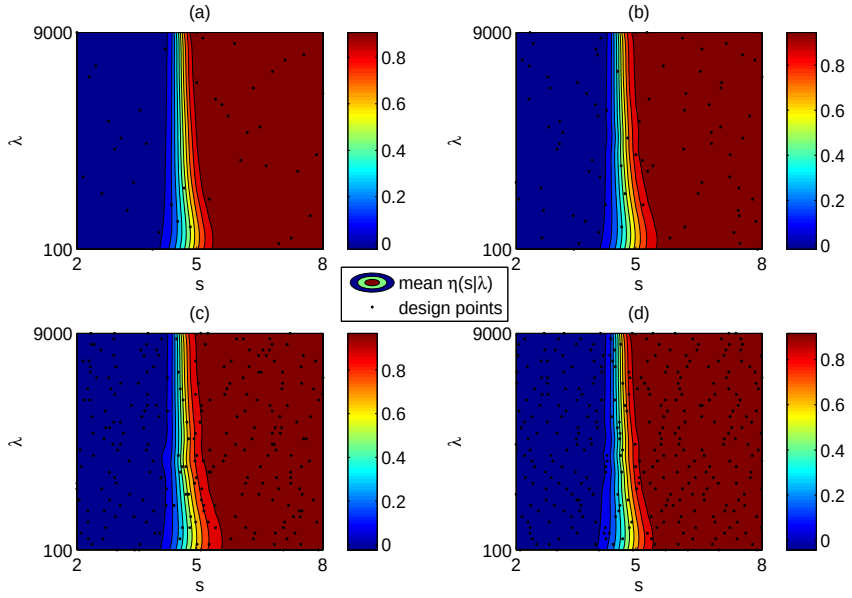

Figure 6.22: Mean emulated contour plots $\eta^{S|\Lambda}(s|\lambda)$, using the Gaussian distribution function as a prior, with (a) 1 sample of s for each λ_i (Latin Hypercube), (b) 2 samples of s for each λ_i, (c) 4 samples of s for each λ_i and (d) 8 samples of s for each λ_i.

a linear prior mean was used. We see that all of the approximations are better than when using the linear prior mean, even in plot (a) where we have a latin hypercube sample. We also notice that, as before, the approximation improves when more samples are taken in the s direction for each λ_i. The values to the left and right edges of each plot in Figure 6.22 are 0 and 1 respectively, due to the prior mean being 0 and 1 at these edges. This is an improvement to the linear prior where the emulator strayed back to the prior mean outside of the data points. We can use this improvement to sample most of the values of s for each λ_i in the area where there is the most change in the surface, and only have a few points outside of this area. In this way, we can reduce the number of design points and so reduce the computational cost of building the emulator.

We finish this example building an emulator built using 16 samples of s for each λ. The

resulting distribution function after this emulator has been integrated over λ is shown in Figure 6.23. If we compare this plot with Figure 6.19, we see that by using the Gaussian distribution function as our prior mean for the emulator, the approximation to the MC estimate has been improved.

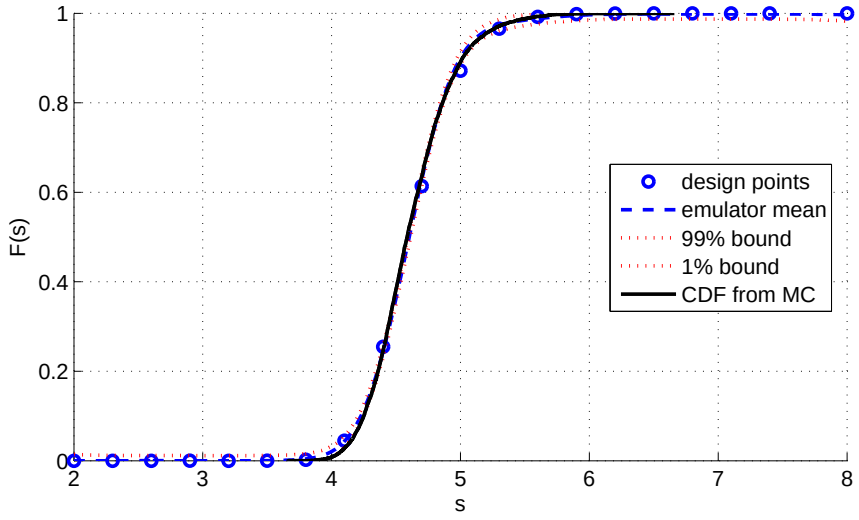

Figure 6.23: CDF for log travel times using a gaussian distribution function prior with all uncertainty in λ and $Z(\mathbf{x})$ integrated out (all other hyperparameters fixed).

6.5.4 Estimating the smoothing parameters

The final issue is how to estimate the smoothing parameters. This becomes more difficult as we increase the dimension of the input sample space. In Section 2.3.4, two methods were described to estimate the smoothing parameters; maximising the posterior mode, and using cross validation. Maximising the posterior mode is simple when using an optimisation algorithm. However, the results that the algorithm provides depend very much on the starting values that are given to the algorithm. Since the hyperparameters vary by a number of orders of magnitude, we need to make sure that the smoothing parameters are scaled appropriately so that the correlation matrix does not become singular. Again we use $\mathbf{s} = (s|\boldsymbol{\theta})$ to simplify the notation. In the following equations, the simplification, $\mathbf{s} = (s_1, s_2, \ldots, s_k)$ where $s_1 = s$ and $s_2, \ldots, s_k$ are the hyperparameters

$\boldsymbol{\theta}$. For example, in the constant mean case, $\mathbf{s} = (s, \beta, \omega^2, \lambda)$

The correlation matrix has terms

$$c(\mathbf{s}, \mathbf{s}') = \exp\{-(\mathbf{s} - \mathbf{s}')^T B(\mathbf{s} - \mathbf{s}')\}$$

where $\mathbf{s}$ is a design point. Since B is a diagonal matrix, the term inside the exponential is given as

$$-\sum_{k=1}^{n}(s_k - s_k')^2 B_{kk}, \tag{6.5.13}$$

where B_{kk} is a different smoothing parameter for each of the parameters that the emulator is built from. Therefore for the constant mean case, 6.5.13 becomes

$$-(s - s')B_{11} - (\beta - \beta')B_{22} - (\omega^2 - \omega^{2'})B_{33} - (\lambda - \lambda')B_{44}, \tag{6.5.14}$$

and we want to determine the smoothing parameters $B_{kk}, k = 1, \ldots, 4$.

For the correlation matrix to be non-singular, we need $B_{kk} \approx \mathrm{O}\left(\frac{1}{(s_k - s_k')^2}\right), k = 1, \ldots, n$. However, we may still run into problems as the range of distances $(s_k - s_k')$ may also run over several orders of magnitude, and so we may find it difficult to choose a value to scale B_{kk} with. We can simplify this by scaling the hyperparameters so that they all lie within $[0, 1]$. This is done using

$$\frac{s_k - s_{k_{\min}}}{s_{k_{\max}} - s_{k_{\min}}}, k = 1, \ldots, n.$$

Therefore if we apply this scaling to (6.5.13), we get

$$-\sum_{k=1}^{n}\frac{1}{(s_{k\max} - s_{k\min})^2}(s_k - s_k')^2 B_{kk}.$$

We now require $B_{kk} \approx \mathrm{O}\left((s_{k\max} - s_{k\min})^2\right), k = 1, \ldots, n$ for the correlation matrix to be non-singular. Since we know $s_{k\max}$ and $s_{k\min}, k = 1, \ldots, n$ from the design points, the scaling of the smoothing parameters is easier to apply.

For the example we have been looking at, the correlation matrix with λ and s scaled to [0,1] has terms

$$-\left[\frac{1}{(\lambda_{\max} - \lambda_{\min})^2}(\lambda_i - \lambda_j)^2 B_{11} + \frac{1}{(s_{\max} - s_{\min})^2}(s_i - s_j)^2 B_{22}\right].$$

We know that $\lambda_{\min} = 100$, $\lambda_{\max} = 9000$, and we have set $s_{\min} = 2$, $s_{\max} = 8$. Therefore, we can make sure that the smoothing parameters B_{11} and B_{22} are of the correct order.

We can then choose suitable starting values for our algorithm which maximises the posterior mode. When the number of design inputs of the emulator increase, the algorithm takes longer to run and find the optimal values for the smoothing parameters. In this case, we choose smoothing parameters using a cross validation procedure as described in Section 2.3.4.

In the next sections, we will extend the previous simple example to include the uncertainty in all of the hyperparameters for each of the three mean models of log transmissivity. By estimating the cumulative distribution function for s, with all other uncertainty in the model integrated out, we can compare how the different assumptions on the prior mean of the log transmissivity field affect the log travel time.

6.6 Emulated cdf for the constant mean model

We now approximate the cumulative distribution for s for the constant model. If we use the same data collected from the computer model as we used to emulate the mean log travel time, then we do not have to run the computer model any more to approximate the distribution function. Therefore, we have 30 design points in $\mathcal{X}_c$, the sample space of $\boldsymbol{\theta}_c$, and 1000 sample of s for each of these points. For each of the design points, we include 20 values of s, to give us a total of 600 design points across the four dimensional space $(s, \boldsymbol{\theta}_c)$. This four dimensional space means that we cannot plot the first emulator surface $\eta^{s|\boldsymbol{\theta}_c}$. We can however, plot slices through the surface in each of the three directions, β, λ and ω^2. We have scaled the hyperparameters and s so that they all lie in $[0, 1]$. The scaled parameters are given the notation $\beta', \lambda', \omega^{2\prime}, s'$. Slices taken by fixing two of the scaled hyperparameters at 0.5 and allowing the other scaled hyperparameter and s' to vary are shown in the first three plots in Figure 6.24. The fourth plot takes into account the correlation between λ' and $\omega^{2\prime}$, by fixing β' only and letting $\lambda' = \omega^{2\prime}$.

In the slice through β', s in plot (a), we see the values on the left are zero, and on the right are 1 with a monotonic increase from left to right, as we would expect from a distribution function. From the sampling design in Table B.1, we have that $\beta' = [0, 1]$ relates to $\beta = [-8.625, -0.939]$. The shape of the plot is similar to the shape in Figure 6.3 between these values of β. Plot (b) gives a slice in the λ', s' direction. We see that the surface is not as smooth as plot (a). In plot (c) we have a slice through the $\omega^{2\prime}, s'$

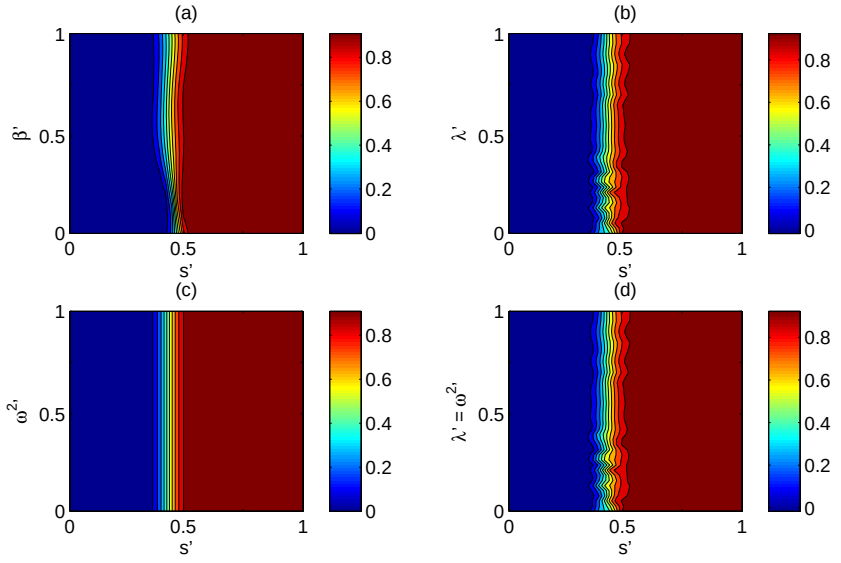

Figure 6.24: Slices through mean of emulator $\eta^{s|\boldsymbol{\theta}_c}$ using a constant model (a) $\lambda', \omega^{2\prime} = 0.5$, (b) $\beta', \omega^{2\prime} = 0.5$, (c) $\beta', \lambda' = 0.5$, (d) $\beta' = 0.5, \lambda' = \omega^{2\prime}$.

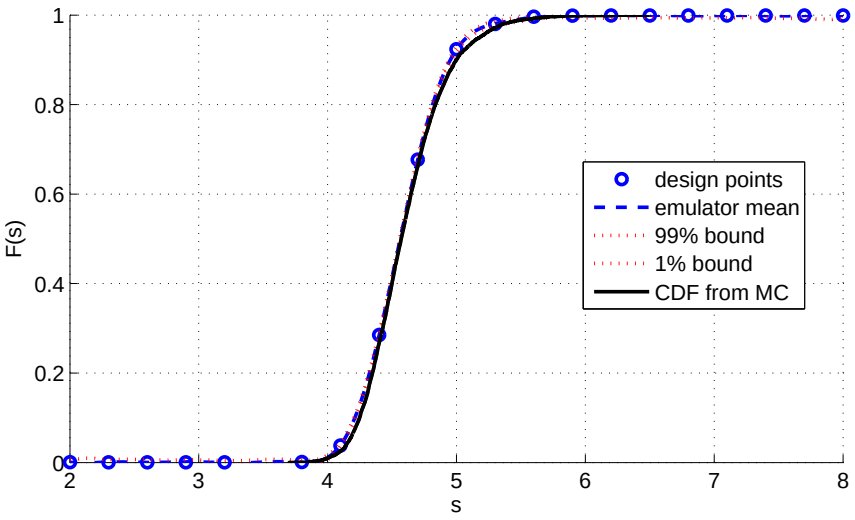

Figure 6.25: Estimated cumulative distribution function for log travel time using a constant model.

directions. This plot suggests that there is no difference in the distribution function as we change ω. We also note that the shape of the surface is similar to that across $\beta' = 0.5$ in plot (a). This leads us to the conclusion that ω does not affect the distribution function of the output as much as λ and β. Due to the correlation between λ and ω^2, we would like to consider these hyperparameters together and so we show a slice through $\lambda' = \omega^{2'}, s'$ in plot (d). There is not much difference in this plot to plot (b). Again, the values lie in [0,1] and we have a monotonically increasing function from left to right.

The first emulator is then integrated over $\boldsymbol{\theta}_c$ at a small sample of s. Figure 6.25 shows the cumulative distribution function for s constructed using this method, and that obtained from a MC sample of 10000 runs of the original computer code. We see that the MC estimate lies within the 98% bounds of the emulator from $F(s) = 0.2$ to 0.8. At the lower and upper ends of the distribution, the emulator overestimates the distribution function by up to around 0.05. The approximation could be improved by including more design points from $\mathcal{X}_c$ in the first emulator, but this would increase the computational cost of approximating this function.

To obtain the MC estimate takes approximately 153 hours. In contrast the emulated estimate takes approximately 13.5 hours, which is less than a tenth of the time. Most of this time is taken up with running the computer model to obtain 1000 samples for each of the design points. After the data has been collected, it only takes around an hour to emulate the conditional cdf, integrate over θ_c and then to emulate the marginal cdf.

6.7 Emulated cdf for the linear mean model

Next we emulate the distribution function for the linear model. In this case, we have 50 design points in $\mathcal{X}_l$ and we choose 20 more design points in the s direction for each of these. We therefore have 1000 design points to cover the six dimensional space of $(s, \boldsymbol{\theta}_l)$. As before, we build an emulator $\eta^{s|\boldsymbol{\theta}_l}$. Two-dimensional slices through the emulator are plotted in Figure 6.26. Plot (a) shows a slice in the β', s direction with λ' and $\omega^{2'}$ fixed at 0.5. As in the constant model, the shape of the plot from $\beta' = 0$ to $\beta' = 1$ is similar to the shape in Figure 6.7. The next three plots, (b), (c) and (d) are all very similar. Again, the shape of the plots is similar to when $\beta = 0.5$ in plot (a). These plots suggest that the hyperparameters, $\beta_x, \beta_y, \lambda$ and ω^2 do not affect the distribution function of the output

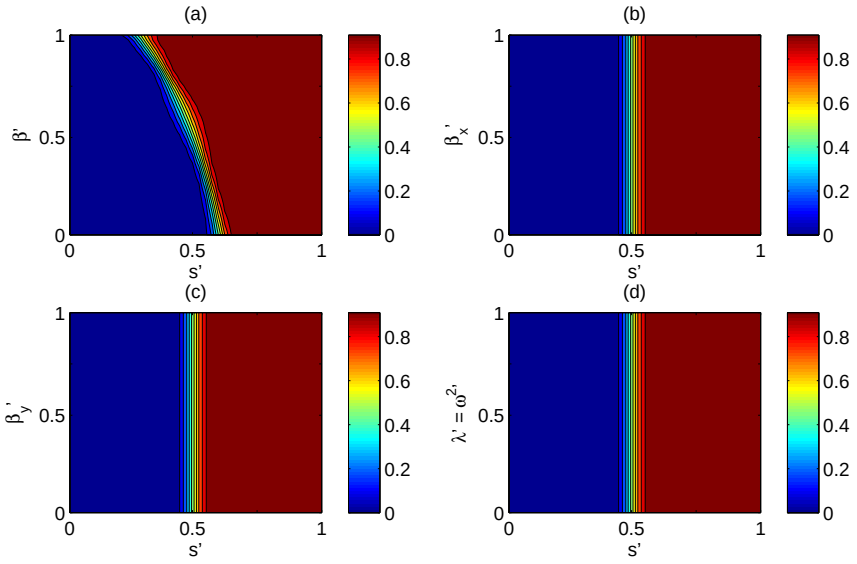

Figure 6.26: Slices through mean of emulator $\eta^{s|\boldsymbol{\theta}_l}$ using a linear model (a) $\beta'_x, \beta'_y, \lambda', \omega^{2\prime} = 0.5$, (b) $\beta', \beta'_y, \lambda', \omega^{2\prime} = 0.5$, (c) $\beta', \beta'_x \lambda', \omega^{2\prime} = 0.5$, (d) $\beta', \beta'_x, \beta'_y = 0.5, \lambda' = \omega^{2\prime}$.

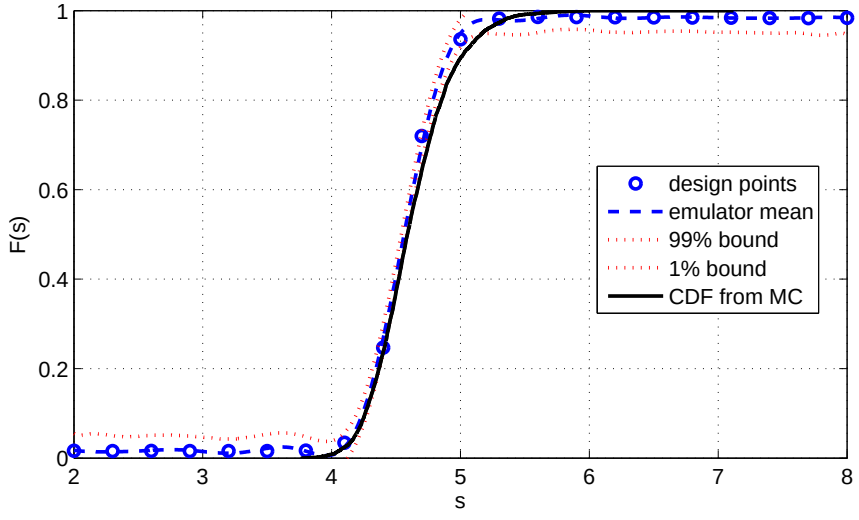

Figure 6.27: Estimated cumulative distribution function for log travel time using a linear model.

as they change, and that β is the most important hyperparameter for determining the distribution function. This is in contrast with the results we obtained for the mean of the computer code output, where we found that all of the hyperparameters have a similar effect on the mean output.

Integrating over $\boldsymbol{\theta}_l$ gives the approximation to the marginal cumulative distribution function for s shown in Figure 6.27, along with an estimate obtained from 10000 MC runs of the computer code. We see that the MC estimation of the cdf lies within the 98% bounds of the emulator for $F(s)$ between 0 and 0.75 and 0.95 and 1. The mean of the emulator overestimates the MC estimate for most of the plot, and does not approximate the MC estimate very well for the values at $F(s) = 0$ and $F(s) = 1$, although the bounds in these regions include the MC estimate. For $s = 4$ to $s = 4.6$, the emulator and the MC estimate are very similar. For the linear model, the emulated approximation is worse than for the constant model. This could be due to the increased number of dimensions that the conditional emulator contained, increasing the uncertainty in the approximation, which is shown by the larger bounds of the emulator. The MC estimate is similar to that obtained for the constant model, so the increase in hyperparameters has not affected the variability of the travel times very much.

For the linear model, the emulated distribution takes approximately 21.8 hours, and the MC estimate takes the same time as before. The emulation time is due to the larger number of design points used to evaluate the model. It takes approximately 25 mins to run the computer model 1000 times with the same input to approximate a distribution function with. After this, the time taken to emulate the conditional cdf, integrate over θ_l and then to emulate the marginal cdf is just over an hour, slightly longer than for the constant model due to the larger number of design points.

6.8 Emulated cdf for the depth mean model

Finally we emulate the distribution function for the depth model. We have 40 design points in $\mathcal{X}_d$ and we choose 20 more design points in the s direction for each of these. The five dimensional space of $(s, \boldsymbol{\theta}_d)$ is covered with 800 design points. Again we build an emulator $\eta^{s|\boldsymbol{\theta}_d}$, which we will then integrate over. Two-dimensional slices through the first emulator are plotted in Figure 6.28. In plot (a), the shape of the surface as β'

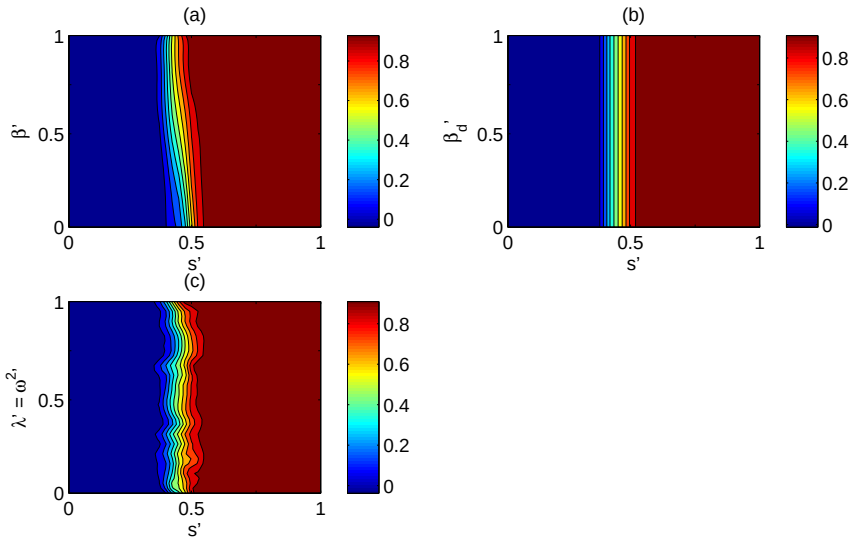

Figure 6.28: Slices through mean of emulator $\eta^{s|\boldsymbol{\theta}_d}$ using a depth model (a) $\beta_d', \lambda', \omega^{2\prime} = 0.5$, (b) $\beta', \lambda', \omega^{2\prime} = 0.5$, (c) $\beta', \beta_d' = 0.5, \lambda' = \omega^{2\prime}$.

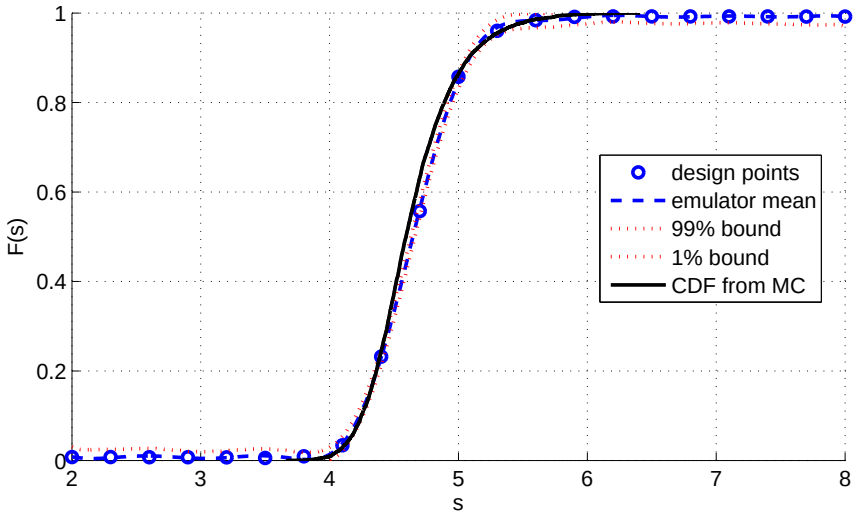

Figure 6.29: Estimated cumulative distribution function for log travel time using a depth model.

goes from 0 to 1 is similar to that seen in Figure 6.11. Plots (b) is the slice through β_d', s. This plot suggests that varying β_d does not affect the distribution function of the output. This is similar to the result we found when emulating the mean of the distribution function; that including β_d in the model does not affect the analysis by very much. We notice that the surface is similar in shape to plot (a) when $\beta' = 0.5$. Finally in plot (c) we have a slice through $\lambda' = \omega^{2'}, s$. We see that the surface is less smooth than the previous plots for β' and β_d', and again we have a good approximation surface of distribution functions.

We now integrate over the first emulator over $\boldsymbol{\theta}_d$ at a small sample of s, and then emulate the distribution function for s only. Figure 6.29 shows the cumulative distribution function for s constructed in this way, along with an estimate obtained from 10000 MC runs of the computer code. We see that the MC estimate lies within the emulator bounds from $F(s) = 0$ to around 0.4 and then again from around $F(s) = 0.8$ to 1. In the middle section of the distribution function, the emulator underestimated the distribution function by up to about 0.05. We also notice that the MC estimation is very similar to that obtained from the constant model.

For the linear model, the emulated distribution takes approximately 17.7 hours, and the MC estimate takes the same time as before. Again, it takes approximately 25 mins to run the computer model 1000 times with the same input to approximate a distribution function with. After this, the time taken to emulate the conditional cdf, integrate over θ_d and then to emulate the marginal cdf is around an hour, slightly longer than for the constant model, and slightly shorter than the linear model due to the difference in the number of design points.

Chapter 7

Conclusions

We have investigated the application of Gaussian process emulators to carry out uncertainty analysis of stochastic models of groundwater flow. In particular, we applied the Bayesian emulation methods to a case study of the WIPP nuclear waste disposal facility. For this site, it is important to be able to quantify the our uncertainty in groundwater flow models of the region, as these types of models are used as part of risk assessments. In this thesis we set out to discover how the uncertainty in the transmissivity field would propagate through the groundwater flow equations to the travel time. This involved developing three stochastic models for the mean of the transmissivity field, and using the available data to derive distributions for the parameters of these fields. A computer model was built to solve the groundwater flow equations numerically, and then the output of this model was analysed for each of the three mean transmissivity field models.

7.1 Development of transmissivity field models

The WIPP case study was used in this thesis as there is a comparatively large amount of data available with which to characterise the transmissivity field. However, even with this data, we found that there is still a lot of uncertainty about the transmissivity field. We chose to represent the transmissivity field using a Gaussian random field model. This allowed us to describe the field using mean and covariance functions. We investigated three different functions for the mean of the log transmissivity field. These were a constant mean, a linear trend in the mean and a depth dependent mean.

The three models investigated in this thesis were simpler than those used in the most recent report on the WIPP site by the US D.O.E.: the 2009 WIPP Compliance Recertification Application (U.S. D.O.E. (2009)). Here the modelling region is split into three zones according to the measured transmissivity values and known geology of the region. The following model for the mean transmissivity is described:

$$Y(x) = \beta_1 + \beta_2 d(x) + \beta_3 I_f(x) + \beta_4 I_D(x) + \beta_5 I_H(x), \tag{7.1.1}$$

where the β's are regression coefficients to be determined, $d(x)$ is the depth to the Culebra dolomite. In addition, there are three indicator functions;

- $I_f(x)$ is the fracture-interconnectivity indicator equal to 1 if fracturing and high transmissivity values have been observed at point x and 0 otherwise.
- $I_D(x)$ is a dissolution indicator function that equals 1 if dissolution of the Salado formation has occurred at point x and 0 otherwise. Dissolution leads to larger fractures and therefore high transmissivity.
- $I_H(x)$ is a halite indicator function equal to 1 in locations where halite occurs beneath the Culebra dolomite and 0 otherwise. If halite is present it acts like cement, blocking the pores of the Culebra and so reducing transmissivity.

Equation (7.1.1) is the same depth dependent model as we have investigated in this thesis, but with additional terms that allow the transmissivity to be determined according to where it lies in the region. In the west of the region is the fracture zone with high transmissivity, and in the west is the halite zone where transmissivity is low. The WIPP site and boundary lies within the central zone, where the Culebra is not affected by either fractures or halite, and areas of high transmissivity are said to occur stochastically (U.S. D.O.E. (2009)). Therefore, for calculating the travel times within this site, these extra indicators may add an extra level of complexity than is required, and our simpler depth model for the mean transmissivity field may be adequate for this purpose.

For the covariance function, we chose an exponential function, which is the same type as used in the 2009 WIPP CRA. The stochastic models we investigated for the transmissivity field provided us with uncertain hyperparameters. In previous work on the WIPP site, the hyperparameters of the transmissivity fields had been set at fixed values using standard regression analysis on equation (7.1.1) using measured data from the site and

expert knowledge about where the geological zones lay. However, we did not have this expert knowledge, and we did not want to ignore the uncertainty in these hyperparameters. Therefore we decided to provide distributions for these hyperparameters rather than fix them at set values. In Chapter 4 we used Bayesian methods to provide us with distributions for the hyperparameters given the available data. We assumed little or no prior information about the hyperparameters when deriving the distributions, and chose posterior distributions which were not influenced by the prior distributions. We used cross validation to check that the distributions obtained gave sensible values of the transmissivity field. Some of the distributions, especially for the correlation length λ, contained wide ranging values. The derivation of these distributions could have been improved by obtaining expert knowledge about these hyperparameters.

We introduced and discussed methods of generating realisations of transmissivity fields using a Gaussian random field model in Chapter 3. Two of the methods, the K-L expansion and the Cholesky decomposition method were investigated further in Chapter 5. The K-L expansion is used to reduce the dimensionality of a problem to make it easier to solve. However, we found that the number of nodes required to give an adequate level of accuracy was greater than expected. In terms of computational time, it was faster to carry out a Cholesky decomposition of the correlation matrix than to calculate the eigenvalues and eigenvectors required for enough accuracy with the K-L expansion. However, the eigenvalue solver `eig` in MATLAB may not be the fastest. The Cholesky decomposition method also has the benefit of including all of the uncertainty in the problem, and so this was used in the computer model for the rest of the analysis.

The methods of generating transmissivity fields presented in this thesis are different from that used in the WIPP certification. In the CRA analysis, the mean transmissivity field, or base field, is generated from equation (7.1.1) using normal random variables in the central zone in the region. Then a residual field is generated through conditional simulations, as discussed in Chapter 3, and combined with the base field. This stochastic simulated field was then used as the initial field for an inverse calibration procedure using the head data. We discussed the idea behind inverse modelling briefly at the end of Chapter 3. This procedure generates a field which is conditional on both head and transmissivity data, and for the CRA, pilot points were used to improve the accuracy of the generated field. The pilot point method involves estimating the transmissivity

values at a set of arbitrary points so that the head field fits the measured head values as accurately as possible. For the CRA, this procedure was repeated for 150 base fields and the groundwater flow equations were solved for each of these fields. The cumulative distribution function of this small sample of calculated travel times was then plotted and used as an indicator of how long a particle released in the centre of the region would take to reach the site boundary.

7.2 Analysis of the WIPP computer model using Bayesian emulation methodology

Our analysis of the travel times was different to that carried out in the WIPP CRA. We considered a larger number of possible transmissivity fields and calculated travel times for each field. This would have been very computationally expensive using MC methods and so we chose to use Gaussian process emulation to reduce the computational cost of the analysis. These methods were introduced in Chapter 2 and then extended to emulate the mean of the output of a stochastic model. In Chapter 6, we then discussed the use of Bayesian emulation methodology to approximate the distribution function of the output of a stochastic model.

In order to emulate and analyse the computer model, we needed to obtain data with which to build an emulator. Therefore, at the end of Chapter 5, we investigated how many runs of the computer model would be required to obtain estimates of the mean and points of the distribution function with enough accuracy to emulate. We found that around 500 runs were needed to estimate the distribution function and around 7500 runs were needed to estimate the mean to a required accuracy. However, given the time that it took to run the code for each hyperparameter, we decided to that the code should be run 1000 times for each design point, and then the same sample data could be used for emulating both the mean and distribution function of the output. After obtaining data from runs of the computer model, we then emulated the mean and distribution function of the output in Chapter 6. The emulators were used to reduce the time taken to carry out the analysis and provide statistical approximations to the output of the computer model.

Firstly, we emulated the mean output, since this is the simplest thing to do for a stochas-

tic model. We estimated the mean, and variance of the mean relating to each of our design points, and emulated the mean of the computer model output using this information. The resulting emulators gave us information about the main effects of each hyperparameter. In all three cases the results showed that the hyperparameters which affected the mean log travel time the most were those of the covariance function. As we increased the number of hyperparameters in our model, the variability in the mean log travel time also increased. This could have been due to the increase in the uncertainty in the problem introduced as more hyperparameters were introduced. Another explanation could have been that the number of input dimensions increased and so the accuracy of the emulator was reduced, although this should have been taken into account by increasing the number of design points.

Emulating the mean did not give us information about the whole distribution of travel times so we emulated the cumulative distribution function of the model. We used a simple example with only one varying hyperparameter to illustrate the method used. We found that the application of emulators to emulating the distribution function was not simple. We came across several problems when building the emulator. The largest problem was of constraining the emulator output to lie within [0,1]. This was solved by specifying a more suitable prior mean for the emulator. Using a Gaussian distribution function as a prior mean meant that the resulting emulator mean would have similar properties, and so would also lie in [0,1]. This meant that the parameters of the Gaussian distribution function prior had to be approximated from the output samples of the code. We saw this as no different to estimating the regression coefficients from the data in the usual prior formulation for the emulator.

The second problem was to estimate the smoothing parameters. We had to make sure that the smoothing parameters were appropriately scaled, since we had scaled the hyperparameters before building the emulator. This gave us a starting point to choose initial values for the procedure to maximise the posterior mode of the emulator. As we had a large number of design points and data, the maximisation algorithm was too expensive to use, and could not guarantee suitable smoothing parameters, as these depended largely on the initial values. Therefore, we had to estimate the smoothing parameters using a cross validation procedure across some of the design points. We could not use them all, or this would also have been very computationally expensive.

Once these problems had been overcome, we could then emulate the distribution function for each of the three models and compare them to the MC estimate. As for the mean, we found that as the number of hyperparameters (inputs) increased, the emulator became less accurate. Therefore, the emulation of the depth model, with 4 hyperparameters, provided a better approximation to the MC cdf than the linear model, with 5 hyperparameters. The emulation of the constant model, with 3 hyperparameters, was more accurate again than the depth model. We found that the distribution functions for each of the three models were very similar, and so the choice of mean transmissivity model did not affect the log travel times. In terms of emulating the distribution function of the log travel time, the constant model was the simplest and fastest as it contained the fewest input hyperparameters.

The emulation of the distribution function presented in Chapter 6, could be used to estimate the distribution function of other stochastic models with uncertain inputs. However, we note that in this case we are fortunate that we can run the computer code enough times for each hyperparameter that an approximation to the conditional cdf of the inputs and outputs to a fairly high degree of accuracy is possible. Since we then integrate over this approximation to estimate the marginal cdf, it is important that the estimation to the conditional cdf does not have too much variability. For more expensive codes, it may only be possible to run the code for a few runs for each input. In this case the cdf may not be able to be calculated with enough accuracy for the results to be useful. We would also have this problem when emulating the mean. However for either case, emulator bounds are larger, giving a quantitative value for the error in the emulator.

There were some limitations to our approach. The main limitation is that the emulators that we built for this thesis were very specific to this problem, and the input space is dependent on the distributions that we derived for the hyperparameters. Therefore if more information was gathered that moved the range of the hyperparameters outside of the current sample space, the emulator would need to be rebuilt. However, we wanted to build the best emulator we could with the current information and so chose this limited approach to give the best results. We also have the problem that the emulated distribution functions do not respect monotonicity. In Chapter 6 we tried a probit transformation to try to solve the problem, but other transformations from $[0, 1]$ to $\mathbb{R}$

such as logit or complementary log-log could be used.

Since we performed a larger MC analysis over the hyperparameters as well as the stochastic model and have not used inverse modelling, our distribution functions may not be comparable to those found in the WIPP CRA. They also investigated several scenarios based on whether full, partial or no mining would occur in the region. For their no mining scenario, they obtained a median travel time of 18,289 years. This is under half the time we found in all three of our models which estimated the median travel time to be around 38,000 years. The minimum and maximum travel times they estimated to be 3,111 years and 101,205 years respectively. All three of our models overestimated the minimum travel time with a minimum of around 5,000 years. The maximum travel times were also over estimated by our model with the constant and depth means having a maximum of around 3 million years and the linear mean having a maximum of around 13 million years. These overestimations of our model to the WIPP results could be due to our model having more uncertainty due to including the uncertainty in the hyperparameters. We also did not use the head data in our analysis, which may lead to a reduction in uncertainty in the code output.

7.3 Further work

Whilst researching this thesis, a number of interesting ideas have arisen that we have not had the time to investigate. A few of these are discussed below.

7.3.1 Investigating different covariance functions for the log transmissivity field

By investigating three stochastic models for the mean log transmissivity field, we have been able to see that the choice of mean transmissivity affects the log travel time very little. From this analysis, we were able to determine that the covariance hyperparameters were the most important parameters in terms of the mean of the log travel time. Therefore, another area of investigation could be to check the sensitivity of the travel times to the correlation function used for the log transmissivity field. In Chapter 4, we introduced the Matérn covariance function. This function includes an additional parameter to the exponential function which controls the smoothness of the correlation.

It may therefore provide a better fit to the log transmissivity data than the exponential correlation function used in this thesis. However, as discussed in Stein (1999), it would be difficult to derive a distribution for the smoothness parameter and so it may need to be set a priori. To provide a transmissivity field with appropriate roughness, a small value of the smoothing parameter, between around 0.5 and 2.5 would be required. Since the exponential function is equivalent to the Matérn with smoothing parameter 0.5, this may mean that the log travel time is only affected by a small amount.

7.3.2 Emulating other outputs of the code

As well as the log travel time, there are other properties of the groundwater flow which may be of interest. Remember that the code represents the groundwater flow in the WIPP region, considering how a particle released in the centre of the region will travel to the site boundary. In this thesis, we have been interested in the time taken for the particle to reach the site boundary, but our code can be easily adapted to output more information about this scenario.

One of the outputs of interest is the position of the particle when it exits the region. This is easily output since the ode to calculate the travel time also provides this information. From preliminary runs, the exit position is usually along the south of the site, although it occasionally exits from the west edge of the site. Therefore, to simplify the emulation, the exit position x_f could be considered as a one dimensional position along the southern boundary of the site. When the exit position exits through the west boundary of the site, the exit position could be considered as $x_f = 0$. The other output of interest that is easily output from our code is the velocity of the groundwater flow at the release point. In order to solve the ode for the travel time, the code calculates the velocities everywhere in the region and so the velocity at the release point is easily obtainable from this. Future work could be carried out to investigate whether these additional outputs are dependent on the hyperparameters, and whether each output is correlated with the other outputs.

This extra information we can obtain from the code could be analysed in the same way as we have done with the log travel time, by emulating the mean and distribution functions of these output. However, a more interesting analysis could look into how these outputs interact with each other; do faster travel times leave the site at a similar point?; or does

the initial velocity lead affect the travel time? Emulators for multivariate outputs could be adapted to deal with emulating stochastic models in order to carry out this analysis of the correlations between the outputs.

7.3.3 Including other parameters into the uncertainty analysis

As mentioned in Section 4.1.1, there are other uncertain parameters which we could have included in $\boldsymbol{\theta}$ as inputs to our emulator. These are

- $h_0(\mathbf{x})$ the head values on the boundary of the WIPP region.
- $b(\mathbf{x})$ the thickness of the Culebra Dolomite across the region.
- $\phi(\mathbf{x})$ the porosity of the Culebra Dolomite across the region.

In this thesis we used a head boundary condition $h_0(\mathbf{x})$ used in the WIPP CRA report. This came from extrapolating the head data to a number of points on the boundary of the region. The values of h_0 between these values were then interpolated along the boundary. We did not consider the uncertainty in the approximation of this condition from the head measurements or in the head measurements themselves. We also used estimates $b(\mathbf{x}) \approx \hat{b} = 8\text{m}$ and $\phi(\mathbf{x}) \approx \hat{\phi} = 0.16$ which were given in the WIPP CRA report. These quantities can also be found from measurements taken in the WIPP region. We could extend our emulator to include these parameters if we wanted to analyse how they affect the log travel time.

Since all three parameters are functions of $\mathbf{x}$, we would need to approximate each of these parameters using suitable functions, in the same way that we approximated $Z(\mathbf{x})$. The hyperparameters of these approximations could then be used as inputs to the computer model as we have shown in this thesis. A result of increasing the number of inputs in the emulator, is that the uncertainty in approximation the log travel time would increase.

7.3.4 Reducing the uncertainty in the output of the computer model

Our computer model contained two main sources of uncertainty. The uncertainty arising from the hyperparameters could be reduced by using expert opinions to help to determine distributions for the hyperparameters. The improved distributions may be narrower

and so the resulting log transmissivity fields and travel times may also be less variable. Another way of reducing uncertainty, which was briefly discussed at the end of Chapter 3, would be to include head data to improve the accuracy of the transmissivity field realisations. Again, this would reduce the variability in the log travel times.

Appendix A

WIPP data

Borehole	Location		Transmissivity T (m^2s^{-1})	$\log_{10} T$
	m East	m North		
H-1	613423	3581684	9.3×10^{-07}	-6.03
H-2	612660	3581652	6.3×10^{-07}	-6.20
H-3	613714	3580892	2.5×10^{-06}	-5.61
H-4	612398	3578484	1.0×10^{-06}	-6.00
H-5	616888	3584793	9.8×10^{-08}	-7.01
H-6	610595	3584991	3.5×10^{-05}	-4.45
H-7	608106	3574644	1.5×10^{-03}	-2.81
H-9	613974	3568252	1.3×10^{-04}	-3.90
H-10	622967	3572458	7.6×10^{-08}	-7.12
H-11	615341	3579124	3.1×10^{-05}	-4.51
H-12	617023	3575452	1.9×10^{-07}	-6.71
H-14	612341	3580354	3.3×10^{-07}	-6.48
H-15	615315	3581859	1.3×10^{-07}	-6.88
H-16	613369	3582212	7.8×10^{-07}	-6.11
H-17	615718	3577513	2.3×10^{-07}	-6.64
H-18	612264	3583166	1.7×10^{-06}	-5.78
DOE-1	615203	3580333	1.2×10^{-05}	-4.93
DOE-2	613683	3585294	9.5×10^{-05}	-4.02
P-14	609084	3581976	2.8×10^{-04}	-3.56
P-15	610624	3578747	9.1×10^{-08}	-7.04
P-17	613926	3577466	1.1×10^{-06}	-5.97
P-18	618367	3580350	7.6×10^{-11}	-10.12
WIPP-12	613710	3583524	1.1×10^{-07}	-6.97
WIPP-13	612644	3584247	7.4×10^{-05}	-4.13
WIPP-18	613735	3583179	3.2×10^{-07}	-6.49
WIPP-19	613739	3582782	6.5×10^{-07}	-6.19
WIPP-21	613743	3582319	2.7×10^{-07}	-6.57
WIPP-22	613739	3582653	4.0×10^{-07}	-6.40
WIPP-25	606385	3584028	2.9×10^{-04}	-3.54
WIPP-26	604014	3581162	1.2×10^{-03}	-2.91
WIPP-27	604426	3593079	4.3×10^{-04}	-3.37
WIPP-28	611266	3594680	2.1×10^{-05}	-4.68
WIPP-30	613721	3589701	2.5×10^{-07}	-6.60
ERDA-9	613696	3581958	5.0×10^{-07}	-6.30
CB-1	613191	3578049	3.0×10^{-07}	-6.52
ENGLE	614953	3567454	4.6×10^{-05}	-4.34
USGS-1	606462	3569459	5.5×10^{-04}	-3.26
D-268	608702	3578877	2.0×10^{-06}	-5.69
AEC-7	621126	3589381	2.8×10^{-07}	-6.55

Table A.1: Measured transmissivity values at WIPP.

Borehole	Location		Depth D	
	m East	m North	ft	m
H-1	613423	3581684	676	206.0
H-2	612660	3581649	623	189.9
H-3	613720	3580892	672	204.8
H-4	612400	3578484	498	151.8
H-5	616888	3584793	897	273.4
H-6	610590	3584992	604	184.1
H-7	608110	3574644	237	72.2
H-9	613974	3568252	647	197.2
H-10	622970	3572458	1360	414.5
H-11	615350	3579124	740	225.6
H-12	617023	3575452	825	251.5
H-14	612341	3580354	545	166.1
H-15	615315	3581859	859	261.8
H-16	613369	3582212	703	214.3
H-17	615718	3577513	706	215.2
H-18	612264	3583166	689	210.0
DOE-1	615203	3580333	829	252.7
DOE-2	613683	3585294	846	257.9
P-14	609084	3581976	573	174.7
P-15	610624	3578747	413	125.9
P-17	613926	3577466	558	170.1
P-18	618367	3580350	909	277.1
WIPP-12	613710	3583524	810	246.9
WIPP-13	612644	3584247	701	213.7
WIPP-18	613735	3583179	786	239.6
WIPP-19	613739	3582782	756	230.4
WIPP-21	613743	3582319	729	222.2
WIPP-22	613739	3582653	742	226.2
WIPP-25	606385	3584028	447	136.2
WIPP-26	604014	3581162	186	56.7
WIPP-27	604426	3593079	292	89.0
WIPP-28	611266	3594680	420	128.0
WIPP-30	613721	3589701	631	192.3
ERDA-9	613696	3581958	704	214.6
CB-1	613191	3578049	no data	no data
ENGLE	614953	3567454	no data	no data
USGS-1	606462	3569459	no data	no data
D-268	608702	3578877	no data	no data
AEC-7	621126	3589381	870	265.2

Table A.2: Measured depth values to top of Culebra at WIPP.

Appendix B

WIPP model input sampling designs

Sample	β	ω^2	λ
1	-7.757	0.414	3488
2	-5.531	0.833	3952
3	-8.625	1.270	4302
4	-3.160	1.882	4628
5	-2.342	2.085	5007
6	-4.643	2.663	5875
7	-5.047	3.432	6180
8	-0.939	3.547	6652
9	-5.687	3.869	7040
10	-7.109	4.102	7804
11	-3.992	4.161	8357
12	-4.199	4.379	8957
13	-5.106	4.771	9320
14	-8.495	4.791	9880
15	-2.975	5.029	10375
16	-4.999	5.112	10869
17	-5.701	5.533	11180
18	-7.521	6.196	12136
19	-4.550	6.390	13619
20	-1.675	6.808	14057
21	-6.618	7.318	15911
22	-5.456	7.760	16474
23	-5.323	8.654	17279
24	-2.193	10.789	20824
25	-1.166	12.442	24439
26	-4.976	14.429	25840
27	-4.299	15.198	29205
28	-4.727	17.944	30579
29	-5.284	18.603	34046
30	-4.113	20.136	35591

Table B.1: Sampling design for $\boldsymbol{\theta}_c$.

Sample	β	β_x	β_y	ω^2	λ
1	-2.667	-2.97×10^{-4}	2.16×10^{-5}	0.094	91
2	-4.210	-2.88×10^{-4}	-4.34×10^{-5}	0.178	175
3	-1.033	-1.50×10^{-4}	-4.53×10^{-5}	0.227	288
4	-0.297	-2.66×10^{-4}	-4.40×10^{-5}	0.371	409
5	-2.821	-2.43×10^{-4}	-5.22×10^{-5}	0.472	565
6	-0.933	-1.78×10^{-4}	-1.24×10^{-4}	0.524	623
7	-1.761	-2.60×10^{-4}	-1.12×10^{-4}	0.658	794
8	0.105	-1.69×10^{-4}	-5.82×10^{-5}	0.714	910
9	-1.873	-3.52×10^{-4}	-5.65×10^{-5}	0.836	1045
10	-2.612	-2.63×10^{-4}	-6.19×10^{-5}	0.916	1146
11	-4.608	-1.30×10^{-4}	2.33×10^{-6}	1.048	1192
12	-3.793	-2.34×10^{-4}	4.04×10^{-5}	1.158	1385
13	-1.738	-3.14×10^{-4}	-8.81×10^{-5}	1.223	1437
14	-2.467	-3.62×10^{-4}	-5.62×10^{-5}	1.237	1486
15	-1.514	-1.39×10^{-4}	4.39×10^{-5}	1.263	1531
16	-2.322	-3.26×10^{-4}	-9.59×10^{-5}	1.281	1568
17	-3.187	-2.44×10^{-4}	-3.61×10^{-5}	1.301	1622
18	-1.921	-3.67×10^{-4}	5.83×10^{-5}	1.333	1655
19	-1.584	-3.46×10^{-4}	-1.17×10^{-4}	1.369	1711
20	-3.403	-1.61×10^{-4}	2.68×10^{-5}	1.380	1758
21	-0.418	-3.04×10^{-4}	-4.29×10^{-5}	1.423	1818
22	-0.767	-3.25×10^{-4}	-4.15×10^{-5}	1.437	1885
23	-2.019	-2.85×10^{-4}	-5.01×10^{-5}	1.470	1921
24	-1.162	-2.74×10^{-4}	-5.39×10^{-5}	1.494	1958
25	-1.994	-2.70×10^{-4}	-4.81×10^{-5}	1.524	2022
26	-0.584	-2.31×10^{-4}	-7.42×10^{-5}	1.503	2091
27	-2.122	-2.47×10^{-4}	7.03×10^{-5}	1.575	2163
28	-2.452	-3.03×10^{-4}	-3.58×10^{-5}	1.616	2269
29	-2.544	-2.79×10^{-4}	-5.22×10^{-5}	1.696	2390
30	-3.547	-2.13×10^{-4}	-1.91×10^{-5}	1.729	2521
31	-1.422	-3.81×10^{-4}	-5.88×10^{-5}	1.761	2674
32	-1.467	-2.55×10^{-4}	-4.02×10^{-5}	1.803	2820
33	-4.258	-3.08×10^{-4}	-4.04×10^{-5}	1.888	3000
34	-0.580	-2.01×10^{-4}	-5.83×10^{-5}	1.925	3066
35	-1.614	-2.92×10^{-4}	-7.22×10^{-5}	1.981	3249
36	-1.960	-3.38×10^{-4}	-1.05×10^{-4}	2.001	3344
37	-2.266	-2.73×10^{-4}	-5.17×10^{-5}	2.090	3472
38	-0.004	-1.95×10^{-4}	-3.95×10^{-5}	2.245	4022
39	-4.009	-3.75×10^{-4}	-3.06×10^{-5}	2.601	4366
40	-4.977	-1.85×10^{-4}	-4.64×10^{-5}	3.287	5760
41	-4.765	-2.50×10^{-4}	-1.66×10^{-5}	3.448	6546
42	-1.840	-1.30×10^{-4}	-3.73×10^{-5}	3.761	7309
43	-2.190	-3.91×10^{-4}	-6.58×10^{-5}	4.283	7703
44	-2.076	-2.69×10^{-4}	-9.76×10^{-5}	4.601	8521
45	-1.373	-2.93×10^{-4}	-1.30×10^{-4}	5.233	9369
46	-2.405	-2.82×10^{-4}	-4.90×10^{-5}	5.511	10178
47	-1.177	-3.93×10^{-4}	6.85×10^{-6}	5.990	10807
48	-2.986	-2.15×10^{-4}	-5.44×10^{-5}	6.503	12128
49	-0.170	-2.28×10^{-4}	-8.49×10^{-5}	6.983	12630
50	-1.654	-2.39×10^{-4}	-4.77×10^{-5}	7.111	13254

Table B.2: Sampling design for $\boldsymbol{\theta}_l$.

Sample	β	β_d	ω^2	λ
1	-5.22273	-0.00271263	0.199002	784.541
2	-5.30486	3.73×10^{-4}	0.457315	1230.03
3	-2.54291	0.00188727	0.637836	1438.12
4	-6.82796	0.00452542	0.848427	1765.31
5	-6.74737	0.00322399	1.29201	2551.5
6	-4.8878	0.00120288	1.73691	2974.45
7	-6.31389	-0.00256472	1.84879	3389.76
8	-4.43893	-7.10×10^{-4}	2.17548	3555.74
9	-6.13563	-0.00108936	2.63435	4348.3
10	-7.70815	-5.15×10^{-4}	2.97228	4595.53
11	-2.28088	-0.00341948	3.15356	5022.63
12	-4.30286	0.00216171	3.30646	5289.41
13	-8.16685	-1.01×10^{-4}	3.62606	5811.38
14	-5.11627	-0.00717561	3.83974	6259.15
15	-4.25843	-0.00332578	3.97749	6606.63
16	-1.77656	0.00681048	4.1009	6840.57
17	-4.1133	-1.19×10^{-4}	4.41611	7434.64
18	-5.00063	-0.00191072	4.47568	7961.06
19	-4.39833	-0.00169054	4.64361	8263.79
20	-5.42449	-2.81×10^{-4}	4.84179	8478.95
21	-5.6513	0.0036968	4.90637	9081.65
22	-4.02663	-8.90×10^{-4}	5.37741	9592.71
23	-5.04689	6.64×10^{-4}	5.45111	10240.8
24	-3.48393	-0.00629184	5.67778	10558.9
25	-1.23698	-0.00769209	6.16567	11520.5
26	-8.62361	-0.00496951	6.42068	11994.1
27	-3.38971	-0.00217988	6.79992	12657.6
28	-2.75056	0.00571024	7.12265	12984.5
29	-1.41006	-0.00482498	7.40537	14151.5
30	-2.14605	0.00337802	7.76636	14951.5
31	-4.74129	6.08×10^{-4}	8.45565	16237.4
32	-4.60015	-0.00682908	8.69012	17085.1
33	-4.85439	8.98×10^{-4}	9.98312	19115.3
34	-8.89602	0.00129674	11.1534	21056
35	-4.76976	-0.00421502	11.3811	22394.4
36	-3.06746	-0.0014829	11.985	23032.8
37	-5.62239	0.00492735	13.37	26121.1
38	-3.97255	0.00164543	14.0754	26770.3
39	-4.19161	1.17×10^{-4}	14.8326	28569.8
40	-8.36755	-0.00848732	15.2508	29272.8

Table B.3: Sampling design for $\boldsymbol{\theta}_d$.

References

Adler, R.J. and Taylor, J.E. *Random fields and Geometry*. Springer Science + Business Media LLC, 2007.

Allen, T.T., Bernshteyn, M.A., and Kabiri-Bamoradian, K. Constructing meta-models for computer experiments. *Journal of Quality Technology*, 35(3):264–274, 2003.

Barton, R.R. Simulation metamodels. In *Proceedings of the 1998 Winter Simulation Conference*, 1998.

Bates, R.A., Kenett, R.S., Steinberg, D.M., and Wynn, H.P. Achieving robust design from computer simulations. *Quality Technology and Quantitative Management*, 3(2): 161–177, 2006.

Bayarri, M.J., Berger, J.O., Paulo, R., and Sacks, J.et al. A framework for validation of computer models. *Technometrics*, 49(2):138–154, 2007.

Bear, J. *Dynamics of fluids in porous media*. New York ; London : American Elsevier, 1972.

Box, G.E.P. and Draper, N.R. *Empirical Model-Building and Response Surfaces*. Wiley, 1987.

Busby, D. Hierarchical adaptive experimental design for gaussian process emulators. *Reliability Engineering and System Safety*, 94:1183–1193, 2009.

Byers, E. and Stephens, D.B. Statistical and stochastic analyses of hydraulic conductivity and particle-size in a fluvial sand. *Soil Science Society of America Journal*, 47: 1072–1081, 1983.

Cauffman, T.L., LaVenue, A.M., and McCord, J.P. Ground-water flow modeling of the culebra dolomite. volume II: Data base. Technical Report SAND89-7068/2., Sandia National Laboratories, 1990.

Chiles, J-P. and Delfiner, P. *Geostatistics: Modelling Spatial Uncertainty*. Wiley, 1999.

Conti, S., Gosling, J.P., Oakley, J.E., and O'Hagan, A. Gaussian process emulation of dynamic computer codes. *Biometrika*, 96(3):663–676, 2009.

Corbet, T.F. A groundwater-basin approach to conceptualize and simulate post-pleistocene and subsurface flow in a semi-arid region, southeastern New Mexico and western Texas, USA. *Hydrogeology Journal*, 8:310–327, 2000.

Cox, N.D. Comparison of two uncertainty analysis methods. *Nucl. Sci. and Eng.*, 64 (1), 1977.

Cressie, N. *Statistics for Spatial Data*. Wiley, 1995.

Cukier, R.I., Fortuin, C.M., and Shuler, K.E. Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. i theory. *The Journal of Chemical Physics*, 59(8):3873–3878, 1973.

Currin, C., Mitchell, T.J., Morris, M.D., and Ylvisaker, D. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, 8(416):953–963, 1991.

Dagan, G. Stochastic modelling of groundwater flow by unconditional and conditional probabilities. 1. Conditional simulation and the direct problem. *Water Resources Research*, 18(4):813–833, 1982.

Dagan, G. *Flow and transport in porous media*. Springer-Verlag, New York, 1989.

Darcy, H. *Les Fontaines Publiques de la Ville de Dijon ("The Public Fountains in the city of Dijon")*. Dalmont, Paris, 1856.

de Marsily, G. *Quantitative Hydrology*. Academic Press, 1986.

Delhomme, J.P. Spatial variability and uncertainty in groundwater flow parameters: A geostatistical approach. *Water Resources Research*, 15(2):269–280, 1979.

Diggle, P.J., Ribeiro Jr, P.J., and Christensen, O.F. *Spatial statistics and computational methods*, chapter 2. An Introduction to Model-based Geostatistics. Springer Verlag, 2003.

Donsker, M.C. and Kac, M. The monte carlo method and its applications. In *Proceedings, Seminar on Scientific Computation*, pages 74–81. International Business Machines Corporation, New York, 1950.

Downing, D.J., Gardner, R.H., and Hoffman, F.O. An examination of response-surface methodologies for uncertainty analysis in assessment models. *Technometrics*, 27(2): 151–163, 1985.

Ernst, O.G. (personal communication), 2009.

Feller, W. *An introduction to probability theory and its applications. Vol 2.* New York: Wiley, 1971.

Freeze, R.A. A stochastic-conceptual analysis of one-dimensional groundwater flow in nonuniform homogeneous media. *Water Resources Research*, 11(5):725–741, October 1975.

Gardner, R.H, O'Neill, R.V., Mankin, J.B., and Carney, J.H. A comparison of sensitivity analysis and error analysis based on a stream ecosystem model. *Ecol. Modelling*, 12: 173–190, 1981.

Geweke, J., Berger, J.O., Dawid, A.P., and Smith, A.F.M. (eds). Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In *In Bayesian Statistics 4*, pages 169–193. University Press, 1992.

Glasserman, P. *Monte Carlo methods in financial engineering.* New York: Springer, 2003.

Gómez-Hernánez, J.J., Sahuquillo, A., and Capilla, J.E. Stochastic simulation of transmissivity fields conditional to both transmissivity and piezometric data - I. Theory. *Journal of Hydrology*, 203:162–174, 1997.

Gotway, C.A. The use of conditional simulation in nuclear-waste-site performance assessment. *Technometrics*, 36(2):129–141, 1994.

Hamby, D.M. A review of techniques for parameter sensitivity analysis of environmental models. *Environmental Modelling and Assessment*, 32:135–154, 1994.

Handcock, M.S. and Wallis, J.R. An approach to statistical spatial-temporal modeling of meteorological fields (with discussion). *Journal of the American Statistical Association*, 89:368–390, 1994.

Haylock, R.G. and O'Hagan, A. On inference for outputs of computationally expensive algorithms with uncertainty on the inputs. *Bayesian Statistics*, 5:629–637, 1996.

Heidelberger, P. and Welch, P.D. Simulation run length control in the presence of an initial transient. *Operations Research*, 31:1109–1144, 1983.

Higdon, D., Kennedy, M., Cavendish, J.C., Cafeo, J.A., and Ryne, R.A. Combining field data and computer simulations for calibration and prediction. *SIAM Journal of Scientific Computing*, 26(2):448–466, 2004.

Higdon, D., Gattiker, J., Williams, B., and Rightley, M. Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103 (482):570–584, June 2008.

Hoeksema, R.J. and Kitanidis, P.K. Analysis of the spatial structure of properties in selected aquifers. *Water Resources Research*, 21(4):563–572, April 1985.

Karhunen, K. Zur spektraltheorie stochastischer prozesse. *Ann. Acad. Sci. Fennicae*, 37(2), 1946.

Kennedy, M.C. and O'Hagan, A. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.

Kennedy, M.C. and O'Hagan, A. Bayesian calibration of computer models. *J. R. Statistical Society Part B*, 63(Part 3):425–464, 2001.

Kennedy, M.C., Anderson, C.W., Conti, S., and OHagan, A. Case studies in gaussian process modelling of computer codes. *Reliability Engineering and System Safety*, 91: 1301–1309, 2006a.

Kennedy, M.C., O'Hagan, A., Anderson, C.W., Lomas, M., Woodward, F.I., Heinemeyer, A., and Gosling, J.P. Quantifying uncertainty in the biospheric carbon flux

for england and wales. *Journal of the Royal Statistical Society, Series A*, 171(1): 109–135, 2006b.

Kenward, M.G. and Carpenter, J. Multiple imputation: current perspectives. *Statistical Methods in Medical Research*, 16:199–218, 2007.

Kleijnen, J.P.C. Kriging metamodeling in simulation: a review. *European Journal of Operational Research*, 192(3):707–716, 2009.

Konikow, L.F. and Mercer, J.W. Groundwater flow and transport modelling. *Journal of Hydrology*, 100:379–409, 1988.

Kröhn, K-P. and Schelkes, K. Modelling of regional variable density groundwater flow in an area in New Mexico: importance of influencing parameters and processes. In *Calibration and Reliability in Groundwater Modelling (Proceedings of the ModelCARE96 Conference)*, pages 353–361, Golden, Colorado, September 1996.

Lahkim, B.M. and Garcia, L.A. Stochastic modeling of exposure and risk in a contaminated heterogeneous aquifer. 1: Monte carlo uncertainty analysis. *Environmental Engineering Science*, 16(5):315–328, 1999.

Landau, D.P. *A guide to Monte Carlo simulations in statistical physics, 3rd Edition.* Cambridge University Press, 1999.

LaVenue, A.M., Cauffman, T.L., and Pickens, J.F. Ground-water flow modeling of the culebra dolomite. volume I: Model calibration. Technical Report SAND89-7068/1., Sandia National Laboratories, 1990.

Lehmann, E.L. and Casella, G. *Theory of point estimation.* Springer-Verlag, New York, 1998.

Liu, F. and West, M. A dynamic modelling strategy for bayesian computer model emulation. *Bayesian Analysis*, 4(2):393–412, 2004.

Loeppky, J.L., Sacks, J., and Welch, W.J. Choosing the sample size of a computer experiment: a practical guide. *Technometrics*, 51(4):366–376, 2009.

Loéve, M.M. *Probability Theory.* Princeton, N.J.: VanNostrand, 1955.

Lu, Z. and Zhang, D. A comparative study on uncertainty quantification for flow in randomly heterogeneous media using monte carlo simulations and conventional and k-l based moment-equation approaches. *SIAM Journal of Scientific Computing*, 26 (2):558–577, 2004.

Lunn, D.J., Thomas, A., Best, N., and Spiegelhalter, D. Winbugs – a Bayesian modelling framework: concepts, structure, and extensibility. *Statistics and Computing*, 10:325–337, 2000.

Martin, J.D. and Simpson, T.W. On the use of kriging models to approximate deterministic computer models. In *Proceedings of DETC '04: ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, September 2004.

Matheron, G. *The Theory of regionalised variables and its applications.* Ecole des Mines, Fontainebleu, 1971.

McKay, M.D. Latin hypercube sampling as a tool in uncertainty analysis of computer models. In *WSC '92: Proceedings of the 24th conference on Winter simulation*, pages 557–564, New York, NY, USA, 1992. ACM Press.

McKay, M.D., Beckman, R.J., and Conover, W.J. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

McLaughlin, D. and Townley, L.R. A reassessment of the groundwater inverse problem. *Water Resources Research*, 32(5).

Mercer, J. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society A*, 209:415446, 1909.

Metropolis, N. and Ulam, S. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

Morris, M.D. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33:161–174, 1991.

Morris, M.D. and Mitchell, T.J. Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference*, 43(3):381–402, 1995.

Nelder, J.A. and Mead, R. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

Oakley, J. *Bayesian uncertainty analysis for complex computer codes*. PhD thesis, University of Sheffield, 1999.

Oakley, J. Eliciting gaussian process priors for complex computer codes. *Statistician*, 51:81–97, 2002.

Oakley, J. and O'Hagan, A. Bayesian inference for the uncertainty distribution of computer model outputs. *Biometrika*, 89(4):769–784, 2002.

Oakley, J. and O'Hagan, A. Probabilistic sensitivity analysis of complex models: a bayesian approach. *J. R. Statistical Society Part B*, 66(Part 3):751–769, 2004.

O'Hagan, A. Bayesian analysis of computer code outputs: A tutorial. *Reliability Engineering and System Safety*, 91:1290–1300, 2006.

Poeter, E.P. and Hill, M.C. Inverse models: A necessary step in ground-water modeling. *Ground Water*, 35(2):250–260, 1997.

Rojnik, K. and Naveršnik, K. Gaussian process metamodeling in bayesian value of information analysis:a case of the complex health economic model for breast cancer screening. *Value in Health*, 11(2):240–250, 2008.

Rougier, J. Efficient emulators for multivariate deterministic functions. *Journal of Computational and Graphical Statistics*, 17(4):827–843, 2008.

Rougier, J., Guillas, S., Maute, A., and Richmond, A.D. Expert knowledge and multivariate emulation: The thermosphereionosphere electrodynamics general circulation model (tie-gcm). *Technometrics*, 51(4):414–424, November 2009.

Roy, R.V. and Grilli, S.T. Probabilistic analysis of flow in random porous media by stochastic boundary elements. *Engineering Analysis with Boundary Elements*, 19: 239–255, 1997.

Russell, T.F. and Wheeler, M.F. *The Mathematics of Reservoir Simulation*, chapter II. Finite Element and Finite Difference Methods for Continuous Flows in Porous Media. SIAM, 1983.

Russo, D. and Bouton, M. Statistical analysis of spatial variability in unsaturated flow parameters. *Water Resources Research*, 28(7):1911–1925, 1992.

Sacks, J., Welch, W.J., Mitchell, J., and Wynn, H.P. Design and analysis of computer experiments. *Statistical Science*, 4:409–435, 1989.

Saltelli, A., Chan, K., and Scott, E.M. *Sensitivity Analysis*. Wiley, 2000.

Schabenberger, O. and Gotway, C.A. *Statistical Methods for Spatial Data Analysis*. Chapman and Hall/CRC Press, 2005.

Simpson, T.W., Peplinski, J.D., Koch, P.N., and Allen, J.K. Metamodels for computer-based engineering design: Survey and recommendations. *Engineering with Computers*, 17:129–150, 2001.

Smith, B.J. Bayesian Output Analysis program (BOA), version 1.1.5. `http://www.public-health.uiowa.edu/boa`, March 23 2005.

Smith, B.J. boa: An R package for MCMC output convergence assessment and posterior inference. *Journal of Statistical Software*, 21(11):1–37, 2007.

Spanier, J. and Gelbard, E. *Monte Carlo principles and neutron transport problems*. New York: Dover, 2008.

Stein, M.L. *Interpolation of Spatial Data: Some theory for kriging*. Springer-Verlag, New York, 1999.

Stuart, A.M. Inverse problems: a bayesian perspective. *Acta Numerica*, 19:451–559, 2010.

Sun, N-Z. *Inverse Problems in Groundwater Modeling*. Kluwer Academic Publishers, 1999.

Thomas, A., Best, N., Lunn, D.J., Arnold, R., and Spiegelhalter, D. GeoBUGS User Manual (version 1.2).

http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/geobugs12manual.pdf, September 2004.

U.S. D.O.E. 2009 WIPP Compliance Recertification Application. http://www.wipp.energy.gov/Documents_EPA.htm, March 2009.

U.S. D.O.E. 2004 WIPP Compliance Recertification Application. http://www.wipp.energy.gov/library/CRA/index.htm, March 2004.

U.S. D.O.E. The U.S. Department of Energy Waste Isolation Pilot Plant website. http://www.wipp.energy.gov, 2010.

U.S. E.P.A. Environmental radiation protection standards for management and disposal of spent nuclear fuel, high-level and transuranic radioactive wastes. http://www.epa.gov/enviro/html/rad/rad_cfr191_194.html, 2010.

van Beers, W. and Kleijnen, J.P.C. Customized sequential designs for random simulation experiments: Kriging metamodeling and bootstrapping. *European Journal of Operational Research*, 186:1099–1113, 2008.