UNIVERSITY OF SOUTHAMPTON

Faculty of Social Sciences
School of Mathematical Sciences

Hierarchical experiments and likelihood approximations

by

Theodora Nearchou

A thesis for the degree of
Doctor of Philosophy

March 2024

University of Southampton

Abstract

Faculty of Social Sciences
School of Mathematical Sciences

Doctor of Philosophy

Hierarchical experiments and likelihood approximations

by Theodora Nearchou

The objective of this thesis is the development of new methods for exploiting multi-level approximations in statistical modelling and design using hierarchical Gaussian processes.

An important application is in multi-level approximations of intractable likelihood problems, where a hierarchy of likelihood approximations is available, each having different accuracy and cost. For example, in generalised linear mixed models, quadrature methods may be used to approximate the likelihood, and by varying the number of quadrature points used we may trade off accuracy against computational cost. Multi-level Gaussian processes methods have been established and implemented to emulate the highest level of accuracy available, using data from all stages of the hierarchy. Moreover, Gaussian process emulated likelihoods are used to conduct inference about the model parameters and to address uncertainty.

There exists a general design problem for hierarchical Gaussian process: to decide how many times to evaluate each level of approximation, and at which parameter values, in order to find an accurate approximation to the point maximising that function, given a limited computational budget. A decision-theoretic approach, called expected gain in utility, based on Bayesian optimisation has been developed and demonstrated through examples.

This thesis is focused on likelihood approximations and the aim is to compute an accurate approximation to the likelihood and to the maximum likelihood estimate (the maximiser of the high-level likelihood approximation) at minimal cost. The methodology is demonstrated on generalised linear mixed model and Ising model examples throughout the thesis. While this thesis in mainly focused on likelihood approximation, the expected gain in utility methodology for choosing the experimental design could also be used on other applications of multi-level Gaussian processes.

Contents

List of Figures

List of Tables

List of Algorithms

Declaration of Authorship

I declare that this thesis and the work presented in it is my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. None of this work has been published before submission

Signed:..... Date:.....

Acknowledgements

Words cannot express my appreciation to the people who were beside me and supported me with their own unique way during this journey.

I acknowledge funding for my PhD studies from EPSRC.

I would like to express my deepest gratitude to my supervisors, Dr Helen Ogden and Professor Dave Woods, for their invaluable guidance, knowledge, dedication and feedback throughout my PhD studies. I am grateful for these 4 years we have shared together, and I will cherish all these moments.

Getting through my PhD required more than academic support, and I am grateful to the people at the office for their advice and ideas, especially Damianos, for being a great influence and for the valuable company from the very first day of our PhDs until the end.

I am extremely thankful to all of the friends that I was fortunate to meet in Southampton and Glasgow during my undergraduate and postgraduate studies, mainly my close friends Maria and Panayiotis. I am grateful to my friend and flatmate Sevi, for accompanied me for the first two years of my PhD time, for the friendship, the laughs, and experiences we shared together. This endeavour would not have been possible without my dearest friend Geoff who has supported me and had to keep up with my stresses and breakdowns for the past four years. I will be forever thankful for our friendship, for all the long conversations we had and the unforgettable memories we have created together.

Most importantly, none of this could have happened without the efforts and unconditional love from my family, my brothers and sister and especially my parents, Kyriaki and Stelios. Thank you for everything you have given me, for believing in me and guiding me even while I was miles away from home.

Last, but not least, I could not have undertaken this PhD without a very special person in my life, Charalambos, who has been next to me and encouraged me throughout this roller-coaster journey with love, patience and support in every step of the way.

List of Symbols

$nAGQ$	number of quadrature points in the AGQ approximation method
σ^2	variance of the random effects of GLMM
k	tuning parameter of the RDA method
β	scalar parameter of the Ising model indicating the inverse temperature
$\boldsymbol{y}$	vector of responses
$\boldsymbol{x}$	vector of predictors
α_0	intercept in the SLR model
α_1	slope in the SLR model
ϵ_i	errors
σ_r^2	variance of residuals
$\boldsymbol{X}$	design matrix of the predictor variables
$\boldsymbol{\beta}$	vector of fixed-effect regression coefficients
$\boldsymbol{Z}$	design matrix of the random effects
$\boldsymbol{u}$	vector of the random effects
$\boldsymbol{G}$	variance-covariance matrix of the random effects
$\boldsymbol{\eta}$	vector of linear predictors
$g(.)$	link function
μ	mean
$\boldsymbol{\phi}$	vector of the variance components in a GLMM
$\phi_q(.)$	normal density function
p_{ij}	parameter of the Bernoulli distribution
β_0	intercept parameter of the GLMM
β_1	slope parameter of the GLMM
$\boldsymbol{\theta}$	vector of the model parameters

$\boldsymbol{H}_e$	Hessian matrix
w_i	quadrature weights
n_q	number of quadrature points
$\phi(.)$	standard normal density function
x_l	quadrature nodes
$O(.)$	Big-O symbol determining the rate of growth-order of the function
$H_n^{(.)}$	Hermite polynomial of degree n
n	number of rows in the Ising model grid
m	number of columns in the Ising model grid
α	threshold parameter of the Ising model
$Z_{n,m}(.)$	function of the normalising constant of the Ising model
σ_ϵ^2	variance of the normally distributed measurement error/nugget effect
$\boldsymbol{\mu}$	mean vector of the multivariate normal distribution
$\boldsymbol{\Sigma}$	covariance matrix of the multivariate normal distribution
Cov	covariance
$\mu(.)$	mean function of the GP
$K(.,.)$	covariance function of the GP
$\boldsymbol{h}$	vector of known regression of known regression functions giving the prior mean structure
$\boldsymbol{b}$	vector of unknown trend parameters
$R(.,.)$	correlation function of the GP
$\boldsymbol{H}$	model matrix of the GP
$\boldsymbol{R}$	correlation matrix of the GP
$\boldsymbol{K}$	covariance matrix of the GP
$\boldsymbol{d}$	separation vector
$\|\|.\|\|$	Euclidean distance
τ^2	signal-variance parameter of the covariance function of the GP
$\boldsymbol{\zeta}$	vector of the hyperparameters of the covariance function of the GP
l	length-scale parameter of the covariance function of the GP
N_p	multivariate normal distribution for a p-dimensional random vector

$\boldsymbol{f}_*$	vector of function evaluations at the test points
$\boldsymbol{x}^*$	vector of the test points
n_*	number of the test points
n	number of the training points
$\boldsymbol{m}_*$	mean vector of the evaluations at the test points
$\boldsymbol{m}$	mean vector of the evaluations at the training points
$\tilde{\boldsymbol{m}}$	predictive mean vector
$\tilde{\boldsymbol{K}}$	predictive covariance matrix
t	number of simulator levels
D_t	experimental design for the t level of simulator
n_t	design points for the t level of simulator
$\boldsymbol{f}_t$	vector of outputs for the t level of simulator
ρ	autoregressive parameter
$h(.)$	prior mean structure of the GP
τ_t^2	variance parameter for the t level of simulator
l_t	length-scale parameter for the t level of simulator
$V_f(.,.)$	function giving the block covariance matrix
$\boldsymbol{V}_f$	covariance matrix of the multi-level GP
β_c	critical value of the parameter β in the Ising model
d	dimension of the parameter space
D	design
$u(.)$	utility function
$\tilde{x}$	candidate point
$\tilde{D}$	design with additional candidate point
$I(.)$	improvement function
$EI(.)$	expected improvement function
$EGU(.)$	expected gain in utility function
μ_f^*	maximum of the current posterior mean
μ_y	posterior predictive mean
σ_y	posterior predictive standard deviation
z	Gauss-Hermite quadrature point
w	Gauss-Hermite quadrature weight

List of Abbreviations

GLMM	**G**eneralised **L**inear **M**ixed **M**odel
LA	**L**aplace **A**pproximation
AGQ	**A**daptive **G**aussian **Q**uadrature
GP	**G**aussian **P**rocess
BO	**B**ayesian **O**ptimisation
EGU	**E**xprected **G**ain in **U**tility
SLR	**S**imple **L**inear **R**egression
MCMC	**M**arkov **C**hain **M**onte **C**arlo
RDA	**R**educed-**D**ependance **A**pproximation
MLE	**M**aximum **L**ikelihood **E**stimate
PDF	**P**robability **D**istribution **F**unction
LHD	**L**atin **H**ypercube **D**esign
MC	**M**onte **C**arlo
SE-ARD	**S**quared **E**xponential-**A**utomatic **R**elevance **D**etermination
MaxPro	**Max**imum **Pro**jection
GALE	**G**iven **A**pproximate **L**ikelihood **E**valuations
EI	**E**xpected **I**mprovement
MF-MI-Greedy	**M**ulti-**F**idelity **M**utual **I**nformation-Greedy
GP-UCB	**G**aussian **P**rocess-**U**pper **C**onfidence **B**ound

Chapter 1

Introduction

1.1 Overview

Understanding and exploiting hierarchical differences in the accuracy and cost of systems and approximations across different (physical or computational) scales is an important topic in many areas of statistical research. In general, accuracy and cost will be inversely related, with economically or computationally expensive "evaluations", physical observations or computational approximations, giving highly accurate results.

The example of approximations across different computational scales, which will focus on this thesis, is the approximation of the likelihood function for models where the likelihood is intractable or computationally expensive by combining existing likelihood approximations, each with different cost and accuracy, to obtain statistically valid and efficient inferences for the model parameters.

Generalised linear mixed model (GLMM) is an example model with intractable likelihood. Figure 1.1 shows approximations of the log-likelihood of a two-level random intercept GLMM. The approximations are obtained using the Laplace approximation (LA) method and the adaptive Gaussian quadrature (AGQ) approximation method with nAGQ quadrature points for multiple quadrature points, where nAGQ is the number of quadrature points.

Figure 1.1 also includes a highly accurate approximation of the log-likelihood which is available for this example calculates using AGQ with 10 quadrature points. As can be seen, the shape of the log-likelihood approximated using different methods is similar, therefore the different approximations are related which is important for the hierarchical structure of the multi-level Gaussian process method we implement. However, the less accurate approximations to the log-likelihood, such as the Laplace approximation, which is a special case of the AGQ with nAGQ $= 1$, and the AGQ with nAGQ $= 2$,

considerably underestimate the log-likelihood for large values of σ, where σ is the standard deviation of the random effects. The location of the maximum differs for each approximation and the maximum value of the log-likelihood decreases for smaller nAGQ numbers. For cases where we are not able to compute an accurate approximation to the likelihood, this problem would have been overlooked. More details on the examples are given in the following chapters.

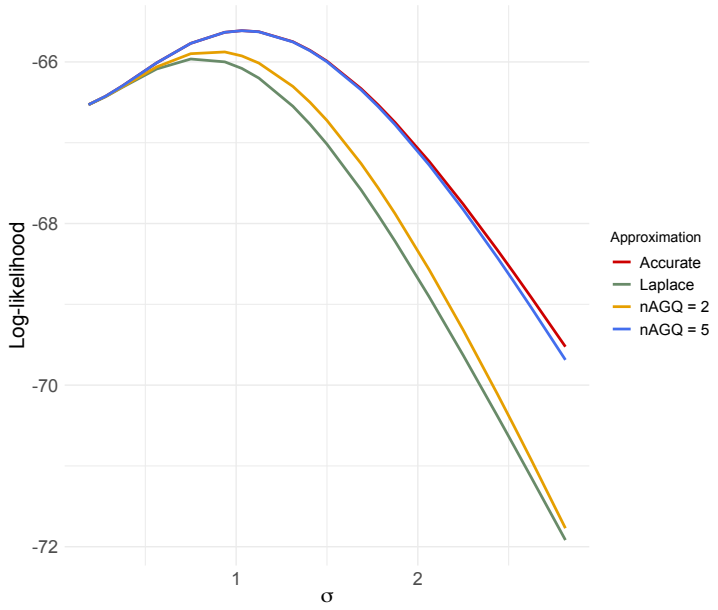


FIGURE 1.1: Log-likelihood approximations for the GLMM example using different approximation methods for varying values of the standard deviation parameter along with an accurate approximation.

Complex approximations can generally provide more accurate approximations to real systems when compared with simpler approximations. However, the complexity of the approximation can turn into a problem when lots of evaluations are needed to obtain the required accuracy. Complex computer approximations are usually computationally expensive meaning that obtaining a single evaluation requires a significant amount of time and cost (Le Gratiet, 2012). Most of the time, budgets will be insufficient to allow statistical modelling and inference to rely solely on results from these evaluations. However, using only lower level computer approximations and evaluations with lower cost, which are available in much greater quantities, will result to lower accuracy and, potentially, misleading conclusions about quantities such as the location of the maximum likelihood estimate.

In this thesis, we are interested in combining evaluations of likelihood approximations from different levels of complexity and accuracy to achieve higher accuracy with lower cost. The methodology can also be adapted and used for problems which do not involve likelihood approximations.

In most statistical situations, inference methodology based on the likelihood function is widely used. However, such methods are not always easy to apply. For instance, for models where the likelihood is **intractable**, and cannot be computed analytically, computational expensive approximations may be available. Within a statistical framework, there is a broad class of methods to handle an intractable likelihood including methods that rely on numerical approximations to the likelihood.

The aim of this thesis is to develop new statistical methods for statistical design, modelling and inference where there is a hierarchy of approximations available, each having different cost and accuracy. The methodology is applied to compute likelihood approximations for models where the likelihood is intractable or computationally expensive to compute. We work with cases where various approximation methods are already available, each having different computational cost and accuracy. We use Gaussian process (GP) models and we combine evaluations from multiple hierarchical scales of approximation to obtain an accurate and efficient approximation to the intractable or too expensive to compute quantity.

Gaussian processes are commonly used for non-parametric modelling of unknown functions (O'Hagan, 1978; Schulz et al., 2018) and have a long history in various fields such as geostatistics, meteorology, computer experiments and machine learning. Gaussian processes can be used in a Bayesian setting where the Gaussian process is a prior on the function. Gaussian processes are commonly used to compute a cheap approximation, or emulator, of an expensive to evaluate computer simulator. They are particularly useful when the evaluation of a function is expensive. GPs are a useful tool, for this thesis, since the type of function we are interested in is an intractable likelihood, and the whole likelihood or log-likelihood function is required to be able to conduct inference for the model parameter.

A computer model, or simulator, can be regarded as a function which takes inputs and returns outputs. Gaussian process models can be used as a surrogate for the simulator output (Perdikaris et al., 2017). Multi-level Gaussian process model-based approximations are applied where multiple levels of approximations are available, some more complex and accurate than other.

In this thesis, we treat the likelihood approximation as a multi-level simulator and we use GP approximation as the surrogate. We have multiple-level approximations and we want to approximate the most complex approximation using evaluations from lower level approximations as well. Using a Bayesian approach, we compute the Gaussian process posterior distribution for a multi-level approximation. The resulting posterior mean function of the GP posterior can be used as an approximation for the output of the complex approximation as well as a prediction for new inputs (Kennedy and O'Hagan, 2000). Therefore, a good approximation of the complex approximation can be achieved

with a sufficient number of evaluations of simpler (lower level) approximations and a limited number of evaluations of the complex approximation (higher level).

Assuming that we have managed to compute a good approximation of the likelihood, using multi-level GP approximation, we aim to estimate the unknown model parameters from the observed data and quantify the uncertainty in these estimates. One approach is to use Bayesian inference and compute the posterior distribution of the model parameters. The posterior distribution of the model parameter can be deduced from the likelihood and a prior distribution describing the prior knowledge of the model parameters.

In most cases, the posterior cannot be expressed analytically due to the complexity of the likelihood or the computation of the normalising constant which involves high dimensional integrals. Samples from the posterior distribution can be used to obtain an approximated posterior distribution for the model parameters. Therefore, using the mean of the Gaussian process posterior density as an approximation to the likelihood we can obtain likelihood samples. Each of these likelihood samples can be then used to compute a posterior sample as an approximation to the posterior. To achieve that we sample from the multi-level GP approximation of the likelihood or log-likelihood and we multiply it with a predefined prior of the model parameter. Doing this we obtain a sample from the posterior. Repeating this procedure for a large number of samples we manage to compute an approximated posterior given approximated likelihood evaluations from the multi-level GP. Hence, we can conduct inference for the model parameters and combine the underlying uncertainty about the model parameters given the data with the uncertainty from using the multi-level Gaussian process model as an approximation to the likelihood rather than the true likelihood itself, which is not always available.

Bayesian optimisation (BO) is a particularly well-suited tool to global optimisation problems where the function we want to optimise is a computationally expensive function (Jones et al., 1998). The goal is to choose an experimental design (set of training points) for the Gaussian Process emulator of the likelihood, the function of interest, with the aim of finding the point maximising the likelihood. To do this, we first define a utility function which is the maximum of the emulated high-level function given the current data.

In this thesis, we work with multi-level approximations each with different cost. Given some initial design and corresponding approximate function evaluations for each level used, we need to choose a new point to add to the existing design without further evaluations of the function approximations and decide at which level of approximation we will add the new point based on the relative cost of each level. We have developed a method for multi-level BO using expected gain in utility (EGU). We choose the point

to add to the design and the level which gives the largest expected gain in utility relative to the cost of evaluating each level of function approximation. Low values of the EGU show that there is no further benefit for adding any new points to the design. The search for new points stops when the maximum value of the EGU falls below a predefined threshold.

1.2 Examples

Throughout the thesis, we illustrate the methodology using three running examples. We work with a simple linear regression model (SLR), a GLMM and an Ising model. Refer to Section 3.4 where the models and the data used will be presented in more detail.

1.2.1 Simple linear regression

The first example is a simple linear regression model which is used to illustrate how the Gaussian process can be used as an approximation to the likelihood. This is a toy example where the analytic form of the likelihood and the posterior distribution is given in closed form so that we can compare and validate the results obtained using GP approximation. For this example, we work with the simple case of a single-level Gaussian process to emulate the likelihood surface given a small number of evaluations of the true likelihood.

1.2.2 Generalised linear mixed models

The second example is a generalised linear mixed model which works as an example of model with intractable likelihood. GLMMs are a flexible class of models for non-normally distributed response data which include additional random effects in their linear predictor (McCulloch and Searle, 2004). The main problem arising when using GLMMs is that the likelihood has no analytic expression for most cases (Pinheiro and Chao, 2006). The integral defining the likelihood function is intractable and its dimension depends on the structure of the random effects. This problem motivated the application of numerical approximations to the likelihood such as Laplace approximation, adaptive Gaussian quadrature, or Bayesian approaches implementing Markov Chain Monte Carlo (MCMC) techniques.

We work with multiple levels of approximation of the log-likelihood for the GLMM example, the Laplace approximation as the lower level of our hierarchy and adaptive Gaussian quadrature, with different choices of the number of quadrature points nAGQ,

as the higher levels. Figure 1.1 shows the different levels of log-likelihood approximation as a function of the standard deviation of the random effects for the GLMM. For each level of approximation used we get a different value of the maximum each at a different location. Explicitly, we have that the maximum for the Laplace approximation is at $\sigma = 0.75$, for nAGQ $= 2$ at $\sigma = 0.94$, for nAGQ $= 5$ at $\sigma = 1.03$ and lastly for the most accurate approximation is at $\sigma = 1.03$. As can be seen, as we increase the number of quadrature points used for computing the AGQ approximation method we get a more accurate approximation which is closer to the accurate log-likelihood function.

1.2.3 Ising models

For our third and last example we work with Ising models (Ising, 1925). The Ising model is a mathematical model of ferromagnetism in statistical mechanics. It can be used as a theoretical model of empirical phenomena and as a data analytic model that provides a statistical likelihood model for dependencies between binary variables (Finnemann et al., 2021). It consists of variables that represent spins that can take the values $+1$ or -1 and the spins are arranged in a lattice (grid) giving the ability to each spin to interact with its neighbors.

The likelihood of an Ising model given the model parameters is usually **computationally expensive** depending on the size of the grid because of the computation of the normalising constant. To obtain likelihood approximations for the multi-level GP approximation we aim to use the reduced-dependence approximation (RDA) method (Friel et al., 2009), which can be computed in different levels of accuracy depending on a tuning parameter k. We replace the expensive to compute normalising constant with an approximation obtained using the hierarchical experiments approach combining evaluations from multi-level approximations.

Figure 1.2 shows the log-likelihood approximations of the Ising model for a simulated example with a 10×10 grid. As can be seen, the shape of the approximated log-likelihood is similar the different methods which is required for the hierarchical approximation method. For this example, the exact log-likelihood is available in closed form and is shown in red in the figure. We have chosen to work with this case so that we will be able to use the exact log-likelihood as a measure of comparison for the calculations. However, for larger grids computing the closed exact form of the normalising constant and consequently the log-likelihood is computationally expensive.

The log-likelihood approximations are given as functions of the parameter of interest β using the reduced-dependence approximation method for various values of k, where $k = \{2, 3, 4, 6\}$. The parameter β of the Ising model is a scalar indicating the inverse temperature. As shown, as we increase the value of k the approximation of

the log-likelihood is closer to the exact log-likelihood of the model. Clearly, the maximum value of the log-likelihood and its location are different for each approximation as presented in the Figure 1.2. More explicitly, we have that the maximum of the approximation obtained using $k = 2$ is at $\beta = 0.27$, for $k = 3$ at $\beta = 0.29$, for $k = 4$ at $\beta = 0.28$, for $k = 6$ at $\beta = 0.26$ and lastly for the exact log-likelihood at $\beta = 0.25$. It is obvious that as we increase the value of k in the RDA method we get a more accurate approximation of the log-likelihood which is closer to the exact log-likelihood function.

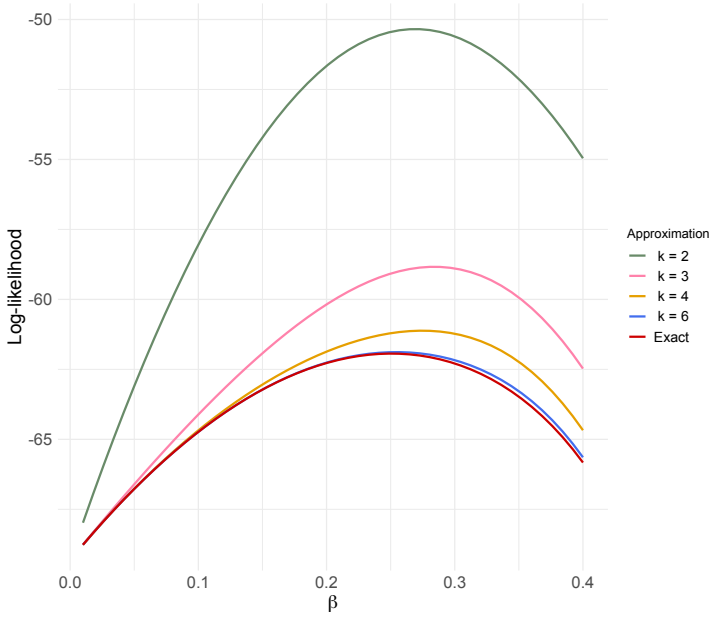


FIGURE 1.2: Log-likelihood approximations for the Ising model example using different values of the tuning parameter k of the RDA method for varying values of the parameter β along with the exact log-likelihood.

1.3 Thesis outline

The structure of the remainder of the thesis is as follows.

Chapter 2 introduces some preliminary material on the thesis running examples: simple linear regression, generalised linear mixed models and Ising models. It also gives an overview of approximation methods for GLMM and Ising models.

Chapter 3 focuses on Gaussian process regression and hierarchical computer experiments. It demonstrates through examples how Gaussian processes can be used to approximate the whole likelihood or log-likelihood surface based on evaluations of the likelihood from multiple approximation levels at a certain set of parameter values. Also, the extension of the GP regression for multi-dimensions is described and an example illustrates how it can be incorporated with multi-level GP approximations.

After obtaining a good approximation of the likelihood surface we conduct inference for the model parameters and combine the uncertainty arising from the Gaussian process approximation to the likelihood and the underlying uncertainty of the model parameters given the data. This is achieved using Bayesian inference and obtaining posterior samples by computing an approximated posterior distribution of the model parameters. The methodology is explained and implemented in Chapter 4.

Chapter 5 introduces the concept of expected gain in utility where we use Bayesian optimisation to choose the experimental design of multi-level GP approximation relative to the cost of each level.

For the purpose of this thesis, we have developed an `R` package called `hela`, which stands for hierarchical experiments and likelihood approximations, to apply the methodology developed for computing hierarchical likelihood approximations. The `R` functions included in the package are presented and described through an example in Chapter 6.

We conclude with some discussion in Chapter 7 of the main ideas and methods presented in this thesis. We also give suggestions for potential directions and future work to expand the methodology by assessing the accuracy of the GP approximation using cross validation, extend the EGU for more dimensions and investigate the choice of the experimental area of the parameters and how hierarchical experiments can be integrated into optimal designs.

For better understanding, a glossary of notation is available in the list of symbols and abbreviations are given in the list of abbreviations. The items are ordered according to context. Figures, tables and algorithms are also summarised in the list of figures, the list of tables and the list of algorithms respectively. Any supplementary work is available as appendices at the end of the thesis.

Chapter 2

Preliminaries

2.1 Overview

This chapter introduces the preliminary materials which will be used in the thesis. Throughout the thesis the ideas and methodology are demonstrated using three example models: simple linear regression model, generalised linear mixed model and Ising model. For the simple linear regression example the likelihood can be given in analytical form. Hence, we use this as an example to validate our methods. The GLMM and the Ising model are examples of models with intractable or computationally expensive likelihoods due to the computation of an intractable or computationally costly quantity.

Generalised linear mixed models are described in detail, explaining the likelihood approximation problem along with some existing methods of likelihood approximation such as the Laplace and the adaptive Gaussian quadrature methods. We also introduce the Ising model, describing the issue arising when computing the normalising constant of the likelihood. As an approximation method of the normalising constant of the Ising model, we present the reduced-dependence approximation method which is a class of approximations which can be computed in various levels each with different cost and accuracy based on a tuning parameter.

2.2 Simple linear regression model

Simple linear regression is a statistical method that allows us to establish a relationship between two continuous variables. The simple linear model can be expressed as

$$y_i = \alpha_0 + \alpha_1 x_i + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, \sigma_r^2), \tag{2.1}$$

where y_i is the response value for each observation i for $i = 1, \ldots, n$, α_0 is the intercept, α_1 is the slope, x_i is the predictor value for each observation i and ϵ_i is the error term which is normally distributed with zero mean and variance σ_r^2 with ϵ_i and ϵ_j assumed to be independent for $i \neq j$.

The likelihood function for the SLR model is available in closed form and is given by

$$L(\alpha_0, \alpha_1, \sigma_r^2; x_1, \ldots, x_n) = (2\pi\sigma_r^2)^{n/2} \exp\left(\frac{1}{2\sigma_r^2} \sum_{j=1}^{n} (y_j - (\alpha_0 + \alpha_1 x_j))^2\right). \tag{2.2}$$

2.3 Generalised linear mixed model

2.3.1 Introduction

Generalised linear mixed models are an important and commonly used model class for statistical analysis. They are appropriate for the analysis of grouped data when the responses depend on a set of covariates and are correlated due to the presence of clusters or groups. In practice, the implementation of GLMMs is not easy due to the complexity of the likelihood function (McCulloch and Searle, 2004). GLMMs are an example of models with intractable likelihood, the type of models we are interested in this thesis.

Multilevel modelling using GLMMs enables researchers to investigate the nature of between-group variability, and the effects of group-level characteristics on individual outcomes. For example, when individuals form groups or clusters it might be expected that two randomly selected individuals from the same group will tend to be more alike than two individuals selected from different groups. Measurements taken on the same individual at different occasions will tend to be more highly correlated than two measurements from different individuals. When the clustering is ignored the standard errors of the regression coefficients will generally be underestimated (Browne and Rasbash, 2004). Consequently, the confidence intervals will not be accurate leading to wrong interpretations, such as claiming that a predictor has a significant effect on the outcome when in fact the effect could be due to chance.

Generalised linear mixed models are an extension of linear mixed models to non-normal responses. Therefore, let us introduce linear mixed models first.

2.3.2 Linear mixed models

Linear mixed models are considered as an extension of linear models allowing for both fixed and random effects (McCulloch and Searle, 2004). A linear model in vector form

can be written as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\boldsymbol{y}$ is a $n \times 1$ vector of the response; $\boldsymbol{X}$ is a $n \times p$ known design matrix of the p predictor variables; $\boldsymbol{\beta}$ is a $p \times 1$ column vector of the fixed-effects regression coefficients and $\boldsymbol{\epsilon}$ is a $n \times 1$ column vector of the errors which gives the part of $\boldsymbol{y}$ that is not explained by the model, with $\epsilon_i \sim N(0, \sigma_r^2)$, and σ_r is an unknown parameter. This model can be extended to include random effects resulting to the **linear mixed model** written in vector form as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\epsilon},$$

where $\boldsymbol{Z}$ is the $n \times q$ known design matrix for the q random effects and $\boldsymbol{u}$ is a $q \times 1$ vector of the random effects, where $\boldsymbol{u}$ follows a normal distribution, $\boldsymbol{u} \sim \mathcal{N}(0, \boldsymbol{G})$ where $\boldsymbol{G}$ is the variance-covariance matrix of the random effects.

Adding a random effect to the linear model is applicable to any class of regression models. Therefore, generalised linear mixed models can be regarded as an extension of the linear mixed models. However, the normality assumption is no longer needed for the errors; $\boldsymbol{u}$ is still assumed normal, and the mean does not need to be a linear combination of the parameters.

2.3.3 Generalised linear mixed models

In a generalised linear model the response variable can follow a non-Gaussian distribution. Let the linear predictor $\boldsymbol{\eta}$ given by

$$\boldsymbol{\eta} = \boldsymbol{X}\boldsymbol{\beta},$$

for a known design matrix $\boldsymbol{X}$. A generic **link function** $g(.)$ is usually applied for modelling the responses. The link function defines the relation of the mean response $\boldsymbol{y}$ with the linear predictor $\boldsymbol{\eta}$. We assume that the distribution of the response has an exponential family form with mean

$$\mu = E[\boldsymbol{y}|\boldsymbol{\eta}] = g^{-1}(\boldsymbol{\eta}).$$

The linear predictor $\boldsymbol{\eta}$ of the generalised linear mixed model is given by the combination of the fixed and random effects excluding the residuals. That is,

$$\boldsymbol{\eta} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u}, \tag{2.3}$$

where $\boldsymbol{X}$ and $\boldsymbol{Z}$ are known design matrices and the random effects component $\boldsymbol{u} = (u_1, \ldots, u_q)^{\mathrm{T}}$ is a sample from a distribution known up to a parameter vector $\boldsymbol{\phi}$ (McCulloch and Searle, 2004). Usually, it is assumed that $\boldsymbol{u}$ follows a multivariate normal distribution with zero mean and covariance matrix $\boldsymbol{G}$ defined via $\boldsymbol{\phi}$, where $\boldsymbol{\phi}$ is a known vector of length q containing the variance components.

2.3.4 Likelihood for generalised linear mixed models

The likelihood is one of the key ingredients in inference and is generally used to generate estimators such as the maximum likelihood estimator and to perform hypothesis testing. The likelihood function is defined on the parameter space while the data is considered fixed.

The likelihood for the generalised linear mixed model described in Section 2.3.3 is given by

$$L(\boldsymbol{\beta}, \phi) = \int \prod_{i=1}^{n} f(y_i|\eta_i = x_i^{\mathrm{T}}\beta + z_i^{\mathrm{T}}u)\phi_q(u, 0, G(\phi))d\boldsymbol{u}, \tag{2.4}$$

where $f(.|\eta)$ is the probability function or probability density function for $\boldsymbol{y}$, for a given value of the linear predictor $\boldsymbol{\eta}$, and $\phi_d(., \mu, \Sigma)$ is the $\mathcal{N}_d(\mu, \Sigma)$ density function, where μ is the mean and Σ is the covariance matrix, and the integration is over the q-dimensional distribution of $\boldsymbol{u}$.

Therefore, is not easy to compute the likelihood in the general case since the integral is usually at least as high dimensional as the number of random effects of the model and it is intractable. Hence, approximation methods are needed to solve the integral (2.4) to compute the likelihood of the GLMM and are described in Section 2.3.5.

For this thesis, we use the example of a two-level hierarchical generalised linear mixed model. As a general framework, the lowest level of observation in the hierarchy is denoted by j, and the group or cluster of the level two unit with i. There are $j = 1, \ldots, m_i$ level one units within the i^{th} level two unit, and $i = 1, \ldots, m$ level two units and let the total sample size be $m_t = \sum_{i=1}^{m} m_i$.

Let us introduce the **two-level random intercept model**, which is one of the simplest multi-level models. Assume that the data is clustered, so that we have y_{ij} observations for $i = 1, \ldots, m$ and $j = 1, \ldots, m_i$, where the j^{th} observation from the i^{th} cluster is given by y_{ij} and let x_{ij} be the corresponding explanatory variable. For this two-level random intercept model, the distribution of y_{ij} is controlled by the linear predictor η_{ij}, defined as

$$\eta_{ij} = x_{ij}^{\mathrm{T}}\boldsymbol{\beta} + u_i,$$

where $u_i \sim N(0, \sigma^2)$ and σ^2 is the variance of the random effects.

The likelihood for a two-level structure GLMM can be written as

$$L(\boldsymbol{\beta}, \sigma) = \int_{\mathbb{R}^n} \prod_{i=1}^{m} \prod_{j=1}^{m_i} f(y_{ij}|\eta_{ij} = x_{ij}^{\mathrm{T}}\beta + u_i)\phi(u_i, 0, \sigma^2)du_i. \tag{2.5}$$

The two-level GLMM is a special case due to its nested structure. Each observation is contained within one cluster and the likelihood can be simplified by swapping the product and the integration of the likelihood in (2.5)

$$L(\boldsymbol{\beta}, \sigma) = \prod_{i=1}^{m} \int_{\mathbb{R}} \prod_{j=1}^{m_i} f(y_{ij}|\eta_{ij} = x_{ij}^{\mathrm{T}}\beta + u_i)\phi(u_i, 0, \sigma^2)du_i. \tag{2.6}$$

We use this simplification so that we will be able to use the AGQ approximation as one of the approximation methods of the likelihood as described in section 2.3.5. For a GLMM in a general form this simplification is not applicable.

In our demonstrations, we will use a two-level random intercept model with binary response. Suppose that we have binary responses, thus each response y_{ij} follows a Bernoulli distribution with parameter p_{ij}, that is $y_{ij} \sim \text{Bernoulli}(p_{ij})$. The two-level random intercept logistic model is given by

$$\log\left(\frac{p_{ij}}{1 - p_{ij}}\right) = \beta_0 + \beta_1 x_{ij} + u_i, \tag{2.7}$$

where $u_i \sim \mathcal{N}(0, \sigma^2)$. The model parameters are given by $\boldsymbol{\theta} = (\beta_0, \beta_1, \sigma)^{\mathrm{T}}$ where β_0 is the intercept, β_1 is the slope which is the same for each group and σ is the standard deviation of the random effects. We would like to make inference about the model parameters $\boldsymbol{\theta}$.

2.3.5 Likelihood approximation methods

A common approach of calculating the likelihood for a GLMM is by using an approximation method. However, the choice of the approximation method may influence the resulting inference. Some of the main likelihood approximation methods based on the literature are the Laplace approximation, Gauss-Hermite quadrature, adaptive Gaussian quadrature and penalised quasi-likelihood. Each method has a different degree of accuracy and computational complexity. These methods are widely used and can be implemented in most software like the `lme4` package (Bates et al., 2015) or the `glmmPQL` function (Venables and Ripley, 2002) in `R`. However, with the `lme4` package the AGQ method can only be used for simple two-level models. We mainly focus on the AGQ approximation and LA methods, where the LA method can be considered as a special case of the AGQ method.

Laplace approximation

The Laplace approximation method (Tierney and Kadane, 1986) approximates the integrand with a function which is proportional to a Gaussian density. In general, the LA method uses Taylor series expansion of the log-likelihood function to give a numerical approximation to the log-likelihood.

Suppose we want to evaluate the intractable integral

$$\int_U g(\boldsymbol{u}) d\boldsymbol{u},$$

where g is some function and $\boldsymbol{u}$ has dimension κ.

The second-order Taylor series expansion of $\log g(\boldsymbol{u})$ around the value that maximises $g(\boldsymbol{u})$, given by $\hat{\boldsymbol{u}}$, is

$$\begin{aligned}\log g(\boldsymbol{u}) &\approx \log g(\hat{\boldsymbol{u}}) + (\boldsymbol{u}-\hat{\boldsymbol{u}})^{\mathrm{T}}(\log g)'(\hat{\boldsymbol{u}}) + \frac{1}{2}(\boldsymbol{u}-\hat{\boldsymbol{u}})^{\mathrm{T}}(\log g)''(\hat{\boldsymbol{u}})(\boldsymbol{u}-\hat{\boldsymbol{u}}) \\ &= \log g(\hat{\boldsymbol{u}}) + \frac{1}{2}(\boldsymbol{u}-\hat{\boldsymbol{u}})^{\mathrm{T}}(\log g)''(\hat{\boldsymbol{u}})(\boldsymbol{u}-\hat{\boldsymbol{u}}),\end{aligned}$$

where $^{\mathrm{T}}$ is the transpose,

$$(\log g)'(\hat{\boldsymbol{u}}) = \left.\frac{\partial \log g(\boldsymbol{u})}{\partial \boldsymbol{u}}\right|_{u=\hat{u}} = 0,$$

and

$$(\log g)''(\hat{\boldsymbol{u}}) = \left.\frac{\partial^2 \log g(\boldsymbol{u})}{\partial \boldsymbol{u} \partial \boldsymbol{u}^{\mathrm{T}}}\right|_{u=\hat{u}}$$

is the Hessian evaluated at $\hat{u}$.

Writing $\boldsymbol{H}_e = (\log g)''(\hat{\boldsymbol{u}})$ we have

$$g(\boldsymbol{u}) \approx g(\hat{\boldsymbol{u}}) \exp\left(-\frac{1}{2}(\boldsymbol{u}-\hat{\boldsymbol{u}})^{\mathrm{T}}\boldsymbol{H}_e(\boldsymbol{u}-\hat{\boldsymbol{u}})\right).$$

Therefore, the Laplace approximation of the intractable integral is given by

$$\int_U g(\boldsymbol{u}) d\boldsymbol{u} \approx g(\hat{\boldsymbol{u}})(2\pi)^{\frac{\kappa}{2}}|\boldsymbol{H}_e|^{\frac{1}{2}}.$$

Using the LA method, the maximum likelihood estimate (MLE) of the fixed effects and the standard deviation of the random effects can be calculated by maximising numerically the approximate likelihood. The accuracy of the LA method can be increased using a higher order of Taylor expansion. However, when the variance of the random effect is large, the LA method is less accurate (Handayani et al., 2017). It can be shown

that the LA method is a special case of the adaptive Gaussian quadrature approximation method with one iteration.

Adaptive Gaussian Quadrature

Quadrature is another method for numerical integration which uses weighted sums, that is

$$I = \int_a^b f(x)dx \approx \sum_{i=1}^{n_q} w_i f(x_i),$$

where n_q denotes the number of quadrature points, w_i are the quadrature weights and x_i are the evaluation or quadrature points. For notation purposes, let nAGQ denote the number of quadrature point of the AGQ method.

The complexity of AGQ increases with the dimension of random effect. Handayani et al. (2017) noted that the AGQ method becomes computationally more expensive when the dimension of random effects is greater than two. The complexity depends on the structure of the integral since the method is limited in the factorisation of high-dimensional integrals into some integrals with low-dimension, which is one of the main reasons why we have chosen to work with the two-level random intercept model where the likelihood can be simplified. The AGQ method can provide an accurate approximation if the number of quadrature points is large enough, but this may increase computational cost. Note that for one quadrature point the AGQ reduces to the Laplace approximation method.

For the purposes of this thesis we are only going to present the AGQ mathematically in one dimension, but extensions to higher dimensions are possible, but their application is expensive. In general, suppose we are interested in approximating in an integral

$$\int f(x)\phi(x)dx, \tag{2.8}$$

where $\phi(.)$ is the standard normal density function. Gaussian quadrature approximates this integral by a weighted sum:

$$\int f(x)\phi(x)dx \approx \sum_{l=1}^{L} w_l f(x_l) \tag{2.9}$$

where L is the number of quadrature points x_l, w_l is a weight constant and x_l is an evaluation points that is solution for the Lth order Hermite polynomial. They are designed to provide an accurate approximation when the function $f(.)$ is a polynomial up to degree $(2L-1)$ or less. Nodes x_l and weight w_l can be found in Gauss-Hermite

quadrature table from Abramowitz and Stegun (1965) for different values of L. Alternatively, they can be calculated as:

$$x_l = i\text{th zero of } H_n(x)$$
$$w_l = \frac{2^{n-1}n!\sqrt{\pi}}{n^2[H_{n-1}(x_i)]^2},$$

where $H_n(x)$ is the Hermite polynomial of degree n (Liu and Pierce, 1994).

In Gaussian quadrature, the nodes x_l and weight w_l are fixed and independent of the integrand. However, for the AGQ x_l and w_l are adapted to the support of the integrand. The AGQ is called adaptive since it refers to the integrand which is scaling by using Hessian at optimum point similar to the Laplace method.

In general, suppose we are interested in approximating the interval $\int g(u)du$. By the Laplace approximation given above, we may approximate the integrand as proportional to a normal distribution density, $\mathcal{N}(\hat{u}, -H^{-1})$. Writing $\mu = \hat{u}$ and $\sigma^2 = -H^{-1}$, we have

$$\begin{aligned}\int g(u)du &= \int \frac{g(u)}{\phi(u,\mu,\sigma^2)}\phi(u,\mu,\sigma^2)du \\ &= \int h(u)\phi(u,\mu,\sigma^2)du,\end{aligned}$$

where $h(u) = g(u)/\phi(u,\mu,\sigma^2)$ and $\phi(.,\mu,\sigma^2)$ is the $\mathcal{N}(\mu,\sigma^2)$ probability density function.

By a change of variable $x = \frac{u-\mu}{\sigma}$, we have

$$\int g(u)du = \int h(\mu+\sigma x)\phi(x)dx,$$

which is of the form (2.8), for $f(x) = h(\mu+\sigma x)$. We can then approximate the integral using the Gaussian quadrature weighted sum (2.9).

The AGQ method increases efficiency by selecting the nodes in a more suitable way (Handayani et al., 2017). AGQ centers the nodes with respect to the mode of the function being integrated and scales them according to the estimated curvature at the mode. This results in a reduction of quadrature points needed to approximate the integrals effectively. Despite the need of additional computing time for the computation of the mode and curvature for each unique cluster, many fewer quadrature points are needed to obtain the same degree of accuracy (Hartzel et al., 2001).

The AGQ method is more time consuming compared to the LA method and it can only be used with a single scalar random effect in most software. The AGQ method gives accurate approximations for models with only one random effect or for two nested random effects. In a situation where a model has a nested structure, like the two-level

random intercept model in (2.6) the AGQ method can be used for likelihood approximation (Handayani et al., 2017).

2.4 Ising models

2.4.1 Introduction

The Ising model (or Lenz-Ising model, named after Ernest Ising (Ising, 1925) and Wilhelm Lenz (Lenz, 1920)) has attracted scientific attention for multiple purposes. It can be used as a theoretical model of empirical phenomena and as a data analytic model that provides a statistical likelihood model for dependencies between binary variables (Finnemann et al., 2021). It is an example of a thermodynamic system and it can be used to model systems for understanding phase transitions.

The model consists of discrete variables that can be in one of two states $(+1$ or $-1)$. In the original problem from physics, these can be magnetic dipole moments of atomic spins which can be arranged in a graph, usually a grid giving the ability to each spin to interact with its neighbours but there are other possible uses as well.

The likelihood of an Ising model given the model parameters is usually computationally expensive because it involves a computationally expensive normalising constant. There are various approaches in the literature tackling the approximation of the likelihood of Ising models. We aim to use reduced-dependence approximation (RDA) (Friel et al., 2009) to obtain likelihood approximations of the Ising model with multiple levels of accuracy.

2.4.2 Likelihood of the Ising model

We consider a simple Ising model for $v = nm$ variables $y_i \in \{-1, 1\}$, for $i = 1, \dots v$, arranged on an $n \times m$ lattice, with parameters $\boldsymbol{\theta} = (\alpha, \beta)$, so that

$$\text{pr}(Y = y; \boldsymbol{\theta}) = Z_{n,m}(\boldsymbol{\theta})^{-1} \exp(\alpha V_0(y) + \beta V_1(y)), \tag{2.10}$$

where $V_0(y) = \sum_i y_i$ and $V_1(y) = \sum_{i \sim j} y_i y_j$. The notation $i \sim j$ indicates that there is an edge between i and j in the lattice, and

$$Z_{n,m}(\boldsymbol{\theta}) = \sum_{y \in \{-1,1\}^v} \exp\{\alpha V_0(y) + \beta V_1(y)\} \tag{2.11}$$

is the normalising constant. The likelihood function $L(\boldsymbol{\theta}; y) = \text{pr}(Y = y; \boldsymbol{\theta})$ depends on $Z_{n,m}(\boldsymbol{\theta})$. The computation of the normalising constant makes the evaluation of the likelihood function challenging and very expensive for large values of v. This is another

kind of problem we want to deal with using the multi-level GP approximation of the likelihood. There are different methods for approximating the normalising constant in the literature such as using Monte Carlo approach (Geyer and Thompson, 1992) or using stochastic approximation expectation algorithm (Zhu et al., 2007). We will use the RDA method described in Section 2.4.4.

The model parameter α of the Ising model controls how likely the state of the variable will be $+1$ or -1. More specifically, for positive values of α it is more likely that the variable will be $+1$, a negative α indicates that it is more likely that the variable will be -1 and $\alpha = 0$ means that there is equal probability of being positive or negative. Considering the second model parameter, β, it indicates how similar the neighbour vertices could be. For positive β the neighbours are likely to be similar, for negative β it is more likely to be different and for $\beta = 0$ the neighbour vertices are independent.

We will focus on a simple special case of an Ising model for which it is possible to compute the normalising constant analytically, to enable comparison of approximate likelihood methods with the true likelihood. The lattice for this special case is shown in Figure 2.1 for $n = m = 3$. We set $\alpha = 0$ in (2.10) and the top row of variables is joined to the bottom row and the left is joined to the right, so that the lattice has periodic boundaries. We have an $m \times m$ grid where the vertices are connected as described.

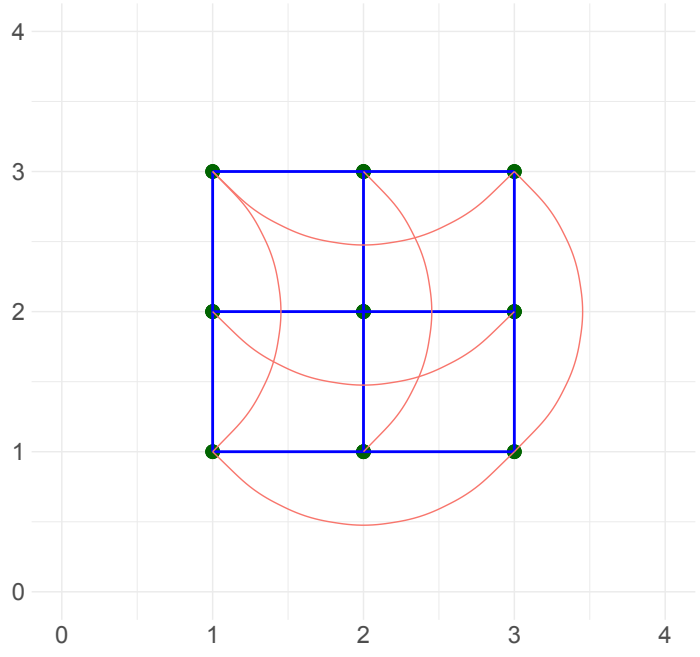


FIGURE 2.1: Lattice with periodic boundary for the special case of Ising model for $n = m = 3$ and $\alpha = 0$.

2.4.3 Normalising constant and log-likelihood

Based on Kaufman (1949), for the special case of the Ising model where $\alpha = 0$ we have a periodic boundary and we can compute the normalising constant $Z_{n,m}(0, \beta)$ for the Ising model exactly. The likelihood can be computed exactly even for larger lattices however it is computationally expensive as the lattice gets bigger. The exact formulas for computing the normalising constant given by Kaufman (1949) are:

$$Z_{n\times m}(0, \beta) = \{2\sinh(2\beta)\}^{nm/2}\bar{A}_{n,m}(\beta)/2,$$

where:

$$\bar{A}_{n,m}(\beta) = A^{(1)}_{n,m}(\beta) + A^{(2)}_{n,m}(\beta) + A^{(3)}_{n,m}(\beta) + A^{(4)}_{n,m}(\beta),$$

with

$$A^{(1)}_{n,m}(\beta) = \prod_{q=0}^{n} 2\cosh\{ma_{2q+1,n}(\beta)/2\}, \qquad A^{(2)}_{n,m}(\beta) = \prod_{q=0}^{n} 2\sinh\{ma_{2q+1,n}(\beta)/2\},$$

$$A^{(3)}_{n,m}(\beta) = \prod_{q=0}^{n} 2\cosh\{ma_{2q,n}(\beta)/2\}, \qquad A^{(4)}_{n,m}(\beta) = \prod_{q=0}^{n} 2\sinh\{ma_{2q,n}(\beta)/2\}.$$

Also,

$$a_{l,n}(\beta) = \cosh^{-1}\{\cosh(2\beta)^2/\sinh(2\beta) - \cos(\pi l/n)\},$$

for $l \geq 1$ and $a_{0,n}(\beta) = a_0(\beta) = 2\beta + \log\{\tanh(\beta)\}$.

We use a special case of the Ising models where the calculation of the normalising constant is given in closed form. The exact computation is not particularly time consuming for the lattice we use for the example. However, in other cases where we do not have a periodic boundary and $\alpha \neq 0$, the calculation would be very time consuming or even impossible especially for large lattices. Therefore, there is a need for an approximation method. We choose this case so that the likelihood can be computed exactly to compare it with the approximation obtained using multi-level GP approximations. We assume that the computationally expensive cases (no periodic boundary and $\alpha \neq 0$) would behave similar with the special case we work with, so that we could use the same methods to deal with the computation of the likelihood.

2.4.4 Reduced-dependence approximation method

The reduced-dependence approximation method was developed by Friel et al. (2009) and we will use it to compute approximations of the normalising constant of the Ising

model. Friel et al. (2009) extended a recursion method by Reeves and Pettitt (2004) which is an exact method for calculating the normalising constant for an un-normalised distribution expressible as a product of factors. The idea of the RDA method is that the likelihood is approximated as a product of factors each of which is defined on sub-lattices whose normalising constant can be calculated using the recursion method.

The reduced-dependence approximation method is a family of approximations controlled by a tuning parameter k which is a positive integer. Larger values of the parameter k provide a more accurate approximation at a higher cost. The parameter k represents the number of rows used in the smaller grid when we compute the approximation. The number of columns, m, remains the same. The tuning parameter k can take any integer value greater or equal to 2. The approximation of the normalising constant of the Ising model for fixed k using the RDA method is given by

$$\tilde{Z}^{(k)}_{n,m}(\boldsymbol{\theta}) = Z^{n-k+1}_{k,m}(\boldsymbol{\theta}) / Z^{n-k}_{k-1,m}(\boldsymbol{\theta}).$$

In general, for any case of Ising model, using the RDA method at level k, with $n = m = c$, the likelihood can be written as

$$\tilde{L}^{(k)}_c(\beta) = \tilde{Z}^{(k)}_{c,c}(\boldsymbol{\theta})^{-1} \exp\{\alpha V_0(y) + \beta V_1(y)\}.$$

The cost of computing the true likelihood is $O(c^2 2^c)$ and the reduced-dependence approximation at level k is $O(c^2 + kc2^k)$ (Ogden, 2017), where the Big-O, $O(.)$, determines the rate of growth-order of the function. It is helpful to know the computational cost of each level of approximation since we will use it when we would like to choose the experimental design of the multi-level GP approximation in Chapter 5.

2.5 Summary

This chapter was an introduction to the main preliminary materials used in this thesis. We introduced the background of the example models that will be used throughout this thesis as an illustration of the ideas and methodology. We reviewed some basic aspects of generalised linear mixed models and compared numerical approximate methods of the likelihood of the GLMM. We presented the Ising model and highlighted the issue arising with the computation of the normalising constant of the likelihood and a method to approximate it.

Chapter 3

Gaussian Processes and Hierarchical Experiments

3.1 Overview

In this chapter we present an overview of Gaussian process and Gaussian process regression. We introduce the idea of hierarchical computer experiments, where our goal is to obtain information about the highest level, which is more "expensive" but the most accurate one, by combining evaluations from lower levels, which are "cheap" and less accurate, with the highest level.

Using a hierarchical computer experiment we aim to reduce the evaluations of high-level approximation required so that we can decrease the computational cost. We also expand the GP regression for multi-dimensional model parameter space of the log-likelihood and how the covariance function can be altered to adjust for multi-dimensions. The running examples of the thesis are presented and demonstrate how the methodology can be used to compute approximation of likelihood functions using multiple levels of approximation.

3.2 Gaussian processes

3.2.1 Introduction

Gaussian processes are commonly used to model unknown functions (O'Hagan, 1978; Schulz et al., 2018). They have being applied in areas such as the design and analysis of computer experiments (Sacks et al., 1989) and in machine learning. Further details and explanation of Gaussian process models can be found in Rasmussen and Williams (2005).

Gaussian processes are particularly useful when it is expensive to evaluate a function $f(.)$ and they can be used to optimise expensive functions. We will focus on situations where $f(.)$ is an intractable likelihood and we need access to the whole likelihood or log-likelihood function to be able to conduct inference. We will study this example in depth in this thesis.

Assuming that the response of a point $\boldsymbol{x}_i$ is $y(\boldsymbol{x}_i) = y_i$, where $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^{\mathrm{T}} \in \mathcal{X}_p$ are the vectors of p input points, then, the relation between the response and the inputs can be described by

$$y_i = f(\boldsymbol{x}_i) + \epsilon_i, \tag{3.1}$$

where ϵ_i is the measurement error with $\epsilon_i \sim N(0, \sigma_\epsilon^2)$ being independent and σ_ϵ being the standard deviation of the measurement error. The function $f(.)$ describes an unspecified function which gives an approximation of the mean relation between the point and the response. Our aim is to estimate the unknown function $f(.)$ using Gaussian processes.

Definition 3.1 (Gaussian process)**.** Suppose that $\mathcal{X}_p$ is a fixed subset of $\mathbb{R}^d$ having positive d-dimensional volume; $f(\boldsymbol{x}_i)$ with $\boldsymbol{x}_i \in \mathcal{X}_p, i = 1, \ldots, n$, is a Gaussian process provided that for any finite integer $n \geq 1$ and any choice of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ in $\mathcal{X}_p$, the vector $\boldsymbol{F} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))^{\mathrm{T}}$ has a multivariate Gaussian distribution given by

$$\boldsymbol{F} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

where $\boldsymbol{\mu}$ is the $n \times 1$ mean vector and $\boldsymbol{\Sigma}$ is the $n \times n$ covariance matrix (Santner et al., 2018).

In general, Gaussian processes extend multivariate Gaussian distributions to infinite dimensionality (Rasmussen and Williams, 2005). They can be used in order to express distributions over functions.

A Gaussian process is a **stochastic process** which is completely determined by its **mean function**, $\mu(.)$ and by its **covariance function**, $K(.,.)$. The mean and the covariance functions encode the assumptions and prior beliefs of the form of the function which we want to learn about. The mean function $\mu(\boldsymbol{x})$ of a real process $f(\boldsymbol{x})$ indicates the expected function value and is given by

$$\mu(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})],$$

and the covariance function $K(\boldsymbol{x}, \boldsymbol{x}')$ of a real process $f(\boldsymbol{x})$ is defined by

$$K(\boldsymbol{x}, \boldsymbol{x}') \equiv \mathrm{cov}(f(\boldsymbol{x}), f(\boldsymbol{x}')) = \mathbb{E}[(f(\boldsymbol{x}) - \mu(\boldsymbol{x}))(f(\boldsymbol{x}') - \mu(\boldsymbol{x}'))],$$

for any position $\boldsymbol{x}$ and $\boldsymbol{x}'$ in an input domain $\mathcal{X}_p$. Hence, the Gaussian process can be written as

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), K(\boldsymbol{x}, \boldsymbol{x}')).$$

The Gaussian process is a non-parametric model which has an infinite dimensional parameter space. Non-parametric methods are useful when we cannot assume the explicit form of a function (Orbanz and Teh, 2010).

For modelling the behaviour of an unknown mathematical function $f(.)$ we use Gaussian processes and we adopt a non-parametric Bayesian approach. Assume the Gaussian process prior

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\boldsymbol{h}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{b}, K(\boldsymbol{x}, \boldsymbol{x}')), \tag{3.2}$$

where $\boldsymbol{h}(\boldsymbol{x}) = (h_0(\boldsymbol{x}), h_1(\boldsymbol{x}), \ldots, h_{k-1}(\boldsymbol{x}))^{\mathrm{T}}$ is a k-vector of known regression functions giving the prior mean structure and $\boldsymbol{b} = (b_0, b_1, \ldots, b_{k-1})^{\mathrm{T}} \in \mathcal{B}$ is a k-vector of unknown trend parameters. $K(\boldsymbol{x}, \boldsymbol{x}') = \tau^2 R(\boldsymbol{x}, \boldsymbol{x}')$, where $R(\boldsymbol{x}, \boldsymbol{x}')$ represents the correlation function for any input $\boldsymbol{x}$ and $\boldsymbol{x}'$ in the domain and τ^2 is the constant variance. Based on Definition 3.1, the Gaussian process prior (3.2) implies a normal prior on function values at any given inputs points $x_1, \ldots, x_n$, of

$$\boldsymbol{F} \sim \mathcal{N}(\boldsymbol{H}\boldsymbol{b}, \boldsymbol{K}), \tag{3.3}$$

where $\boldsymbol{H} = (\boldsymbol{h}(\boldsymbol{x}_1), \boldsymbol{h}(\boldsymbol{x}_2), \ldots, \boldsymbol{h}(\boldsymbol{x}_n))^{\mathrm{T}}$ is the $n \times k$ model matrix and $\boldsymbol{K}$ is the $n \times n$ covariance matrix (Rasmussen and Williams, 2005).

3.2.2 Covariance function

The covariance function of the Gaussian process determines how the model generalises or extrapolates to new data. The covariance function can determine the stationarity and smoothness of the function. Therefore, it must be chosen carefully to achieve a good fit to the training data and to produce more reliable posterior distributions and predictions.

Properties of the Gaussian process associated with the covariance function

Stationarity

A general stochastic process $f(.)$ is **strongly stationary** when for any *separation vector* $\boldsymbol{d} \in \mathbb{R}^d$, any given $n \geq 1$ and any set of points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ in $\mathcal{X}_p$ having $\boldsymbol{x}_1 + \boldsymbol{d}, \ldots, \boldsymbol{x}_n + \boldsymbol{d}$ in $\mathcal{X}_p$, the joint distributions of $f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n)$ and $f(\boldsymbol{x}_1 + \boldsymbol{d}), \ldots, f(\boldsymbol{x}_n + \boldsymbol{d})$ are the same. This establishes that the relation between $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ is entirely determined

by the difference between $\boldsymbol{x}$ and $\boldsymbol{x}'$. A stochastic process is **second-order stationary**, if the covariance function depends only on the separation vector $\boldsymbol{d} = \boldsymbol{x} - \boldsymbol{x}'$, i.e. it satisfies $\text{cov}[f(\boldsymbol{x}), f(\boldsymbol{x}')] = K(\boldsymbol{d})$ (Santner et al., 2018). For a stationary process the distribution is invariant with time. Therefore, parameters such as mean and variance are invariant under any translation in the input space (Le Gratiet, 2013). When the requirements described above are not valid the Gaussian process can be characterised as **non-stationary**.

For a Gaussian process, strong stationarity and second order stationarity are equivalent. This is a result of the fact that the mean and covariance characterise completely the finite-dimensional distributions of a Gaussian process.

Choice of the covariance function

Most of the times is useful to express covariance functions as

$$K(\boldsymbol{x}, \boldsymbol{x}') = \tau^2 R(\boldsymbol{x}, \boldsymbol{x}'),$$

where τ^2 is the variance parameter and $R(\boldsymbol{x}, \boldsymbol{x}')$ is the correlation function for $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}_p$. The correlation function is also a function of the separation vector, $R(\boldsymbol{x}, \boldsymbol{x}') = R(\boldsymbol{d})$.

The covariance and correlation functions should be chosen carefully since they must have certain properties in order to be valid. They must be positive definite to make sure that the resulting covariance matrix is also positive semi-definite and non-singular.

Moreover, the covariance and correlation functions must be symmetric, that is $K(\boldsymbol{d}) = K(-\boldsymbol{d})$ and $R(\boldsymbol{d}) = R(-\boldsymbol{d})$. The correlation function must also satisfy the condition $R(\boldsymbol{0}) = 1$ (Santner et al., 2018). Another assumption that needs to be taken into consideration is the smoothness of the function being modelled. It is required that input values that are similar, close to each other, must correspond to output values that are close as well.

There are many different families of covariance functions that can be used when modelling a Gaussian process such as the Gaussian correlation function, the power exponential family and the Matérn correlation family (Santner et al., 2018). However, we do not examine them further here since it is beyond the scope of the current thesis. For the examples of this thesis, we choose to use the squared exponential covariance function because it is a smooth stationary covariance function.

The **squared exponential** covariance function is a commonly used choice of the covariance function. It provides smooth sample functions which are infinitely differentiable

which are applicable for modelling smooth functions (Damianou, 2015). It is given by

$$K(\boldsymbol{x}, \boldsymbol{x}') = \tau^2 \exp\left[-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2l^2}\right], \tag{3.4}$$

where τ^2 is the signal variance parameter and l is the characteristic length-scale parameter. In this thesis, we are focusing on log-likelihood approximations and we expect them to be smooth and well behaved. Hence, the squared exponential covariance function is a reasonable choice. When applying the methods presented in the thesis for other examples, the choice of the covariance function could be different and should be based on the specific application.

The parameters $\boldsymbol{\zeta} = (\tau^2, l)^{\mathrm{T}}$ are the two **hyperparameters** of the covariance function of the Gaussian process. The hyperparameter τ^2 controls the amplitude of the function and the variation of function values from their mean and l controls the length-scale of variation. The values of the hyperparameters have to be determined.

More details about the hyperparameters are given in Section 3.2.7, which illustrates how the variation of the values of the hyperparameters affects the covariance function and subsequently the posterior distribution. Also, Section 3.2.8 deals with the case where the hyperparameters are not fixed and provides a hyperparameter estimation method.

The Gaussian process for values at some particular set of inputs or test points using the equations (3.1), (3.3) and conditioning on the hyperparameters can be written

$$\boldsymbol{y}|\boldsymbol{b}, \tau^2, l \sim \mathcal{N}(\boldsymbol{H}\boldsymbol{b}, \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}')). \tag{3.5}$$

Figure 3.1 illustrates random samples from the Gaussian process prior with zero mean given by the green line. The covariance function used is the squared exponential covariance function. It can be seen that the samples are very smooth which vary around the zero mean. This is due to the smoothness of the squared exponential covariance function. Looking at the covariance function (3.4), the points which are close to each other are anticipated to operate more alike, since their correlation is approximately unity, compared with points which are further away from each other, where their covariance gets smaller as the distance between them increases. Hence, training points which are close to points we want to make predictions at, the test points, are going to be more informative in terms of the predictions.

3.2.3 Training points

The choice of the number and location of the training points used to estimate the function is important. The **experimental design** consists of the choice of the $\boldsymbol{x}$ values, the

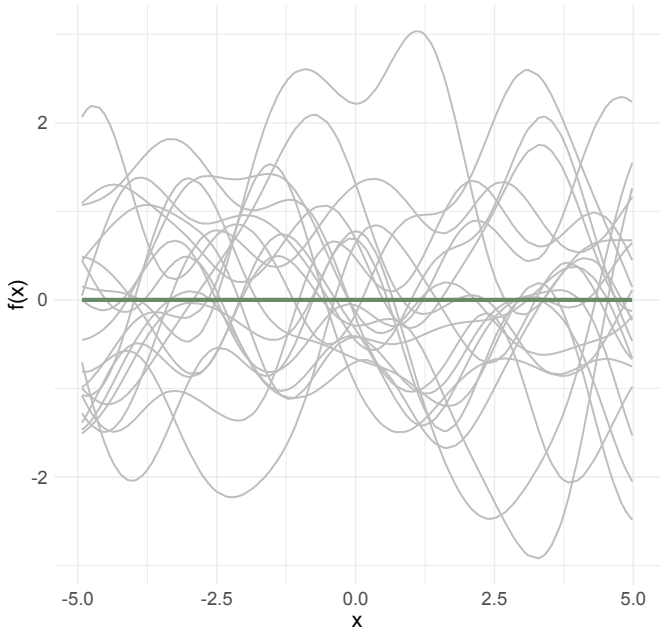


FIGURE 3.1: Random sampling from the Gaussian process prior with zero mean given by the green line and squared exponential covariance function.

inputs, in the **experimental region** at which we compute the output of a computer experiment and collect the data. We would like to have a design which is **space-filling**, meaning that the inputs are evenly spread in the experimental region. Designs which are not space-filling may result to predictions performing quite poorly in some areas of the experimental region that are sparsely observed (Santner et al., 2018).

There are various methods and statistical strategies used to choose the experimental design and subsequently the training points that can be found in the literature of the design of computer experiments (Santner et al., 2018). For the examples presented in this thesis we use random Latin hypercube sampling (LHS) (McKay et al., 1979) to generate the initial design, training points, of the GP. Latin hypercube designs (LHD) are space-filling designs which are often used to generate the training points for computer experiments or for Monte Carlo (MC) integration. In principle, the methodology can be used with any space-filling design. In Chapter 5 we investigate how we can choose the experimental design including points that will benefit the approximation based on Bayesian optimisation and the expected gain in utility method.

3.2.4 Multivariate normal distribution

The multivariate normal distribution and its conditional distribution will be used to compute the posterior predictive distribution in the Gaussian process regression in Section 3.2.5. Therefore, in this section we introduce the equations and the notation of the

multivariate normal distribution.

The multivariate normal distribution or joint normal distribution can be considered as a generalisation of the univariate normal distribution to higher dimensions. The multivariate normal distribution for a p-dimensional random vector $\boldsymbol{y} = (y_1, \ldots, y_p)^{\mathrm{T}}$ is given by

$$\boldsymbol{y} \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

with components the p-dimensional mean vector

$$\boldsymbol{\mu} = \mathrm{E}[\boldsymbol{y}] = (\mathrm{E}[\boldsymbol{y}_1], \mathrm{E}[\boldsymbol{y}_2], \ldots, \mathrm{E}[\boldsymbol{y}_p])^{\mathrm{T}},$$

and the $p \times p$ covariance matrix which has ij entries

$$\Sigma_{ij} = \mathrm{E}[(y_i - \mu_i)(y_j - \mu_j)] = \mathrm{cov}[y_i, y_j],$$

where $1 \leq i \leq p$ and $1 \leq j \leq p$.

Conditional distribution

We can partition the p-dimensional vector $\boldsymbol{y}$ into two subsets $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ such that

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix} \text{ with sizes } \begin{bmatrix} q \times 1 \\ (p-q) \times 1 \end{bmatrix},$$

and respectively $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ can be partitioned as

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix} \text{ with sizes } \begin{bmatrix} q \times 1 \\ (p-q) \times 1 \end{bmatrix},$$

and

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix} \text{ with sizes } \begin{bmatrix} q \times 1 & q \times (p-q) \\ (p-q) \times q & (p-q) \times (p-q) \end{bmatrix}.$$

The conditional distribution of $\boldsymbol{y}_2$ given $\boldsymbol{y}_1 = y_1$, $f(\boldsymbol{y}_2 | \boldsymbol{y}_1 = y_1)$ is a multivariate normal distribution with

$$f(\boldsymbol{y}_2 | \boldsymbol{y}_1 = y_1) \sim \mathcal{N}(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}),$$

where the mean vector is given by

$$\bar{\boldsymbol{\mu}} = \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21} \boldsymbol{\Sigma}_{11}^{-1} (\boldsymbol{y}_1 - \boldsymbol{\mu}_1), \tag{3.6}$$

and the covariance matrix by

$$\bar{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}. \tag{3.7}$$

3.2.5 Gaussian process regression

Introduction

This section will demonstrate how the Gaussian process framework can be used to conduct inferences for functions given some training data. We will derive two types of Gaussian process regression based on the multivariate normal distribution and more specifically its conditional distribution given in Section 3.2.4. For the first type, we consider that the observations are noise free and for the second type we assume that the observations include some noise.

Often, for simplicity a zero mean function for the GP process prior is assumed. However, we choose a non zero mean since including a simple mean function makes the model more interpretable and it is more convenient to express prior information. When using a zero mean function the computations associated with the covariance function are easier, however, predictions far away from the training data may be less accurate. Therefore, it is useful to specify some fixed basis functions $\boldsymbol{h}(.)$ whose coefficients $\boldsymbol{b}$ will be inferred from the data as described in equation (3.9).

We are interested in combining the information from the training data with the prior to compute the posterior distribution. In order to do that and to be able to make predictions for a new point one has to calculate first the joint distribution of the training observations $\boldsymbol{y}$ and the function evaluations, test outputs, $\boldsymbol{f}_*$ where $\boldsymbol{f}_* = (f(\boldsymbol{x}_1^*), \ldots, f(\boldsymbol{x}_{n_*}^*))^{\mathrm{T}}$ and $\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_{n_*}^*$ are the test inputs. Using the results for the conditional distribution derived in Section 3.2.4, we can compute the conditional distribution of $\boldsymbol{f}_*|\boldsymbol{y}$.

Noise free observations

Considering the no noise case, the observations $y(\boldsymbol{x}_i)$ are expressed by the Gaussian process model given by

$$y(\boldsymbol{x}_i) = f(\boldsymbol{x}_i), \quad \text{for } i = 1, \ldots, n.$$

The joint distribution of $\boldsymbol{y}$ and $\boldsymbol{f}_*$ for noise free observations under the Gaussian process prior with non zero mean function is given by

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{m}_* \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) & \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}^*) \\ \boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{x}) & \boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{x}^*) \end{bmatrix} \right), \tag{3.8}$$

where $\boldsymbol{x}$ represents the n training points and $\boldsymbol{x}^*$ the n_* test points. Also, $\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x})$ is the $n \times n$ covariance matrix between the training points, $\boldsymbol{K}(\boldsymbol{x}^*,\boldsymbol{x}^*)$ is the $n_* \times n_*$ covariance matrix between the test points, $\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x}^*)$ is the $n \times n_*$ covariance matrix between the training points and test points and $\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x}^*)^{\mathrm{T}} = \boldsymbol{K}(\boldsymbol{x}^*,\boldsymbol{x})$, where $^{\mathrm{T}}$ is the transpose. Consider the mean vectors $\boldsymbol{m} = \boldsymbol{H}^{\mathrm{T}}\boldsymbol{b}$ and $\boldsymbol{m}_* = \boldsymbol{H}_*^{\mathrm{T}}\boldsymbol{b}$ where the matrix $\boldsymbol{H}$ collects the $\boldsymbol{h}(\boldsymbol{x}_i)$ vectors for all training points and $\boldsymbol{H}_*$ for the test points.

The parameter $\boldsymbol{b}$ can be characterised as a compromise between the data term and the prior and describes the mean of the global linear model parameters and is estimated using (3.9). We estimate $\boldsymbol{b}$ based on the training data. The parameter $\boldsymbol{b}$ will be estimated when we fit the model along with the hyperparameters of the covariance function using the relation

$$\hat{\boldsymbol{b}} = (\boldsymbol{H}\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x})^{-1}\boldsymbol{H}^{\mathrm{T}})^{-1}(\boldsymbol{H}\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x})^{-1}\boldsymbol{y}). \tag{3.9}$$

After estimating $\boldsymbol{b}$ we consider it fixed.

The posterior provides the updated beliefs in the light of the observations. It can be calculated by conditioning the joint Gaussian prior (3.8) on the observations using standard results of the normal distribution given in Section 3.2.4. Therefore, the posterior-predictive distribution for noise free observations is a Gaussian process with predictive mean vector

$$\bar{\boldsymbol{m}} = \boldsymbol{m}_* + \boldsymbol{K}(\boldsymbol{x}^*,\boldsymbol{x})\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x})^{-1}(\boldsymbol{y} - \boldsymbol{m}),$$

and covariance matrix

$$\bar{\boldsymbol{K}} = \mathrm{cov}(\boldsymbol{f}_*) + \boldsymbol{Q}^{\mathrm{T}}(\boldsymbol{H}\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x})^{-1}\boldsymbol{H}^{\mathrm{T}})^{-1}\boldsymbol{Q},$$

where

$$\boldsymbol{Q} = \boldsymbol{H}_* - \boldsymbol{H}\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x})^{-1}\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x}_*)$$

and

$$\mathrm{cov}(\boldsymbol{f}_*) = \boldsymbol{K}(\boldsymbol{x}^*,\boldsymbol{x}^*) - \boldsymbol{K}(\boldsymbol{x}^*,\boldsymbol{x})\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x})^{-1}\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x}^*).$$

Hence, the posterior distribution is given by

$$\boldsymbol{f}_*|\boldsymbol{x},\boldsymbol{y},\boldsymbol{x}^* \sim \mathcal{N}(\bar{\boldsymbol{m}},\bar{\boldsymbol{K}}).$$

Looking at the major components of the Gaussian process, the predictive mean $\bar{\boldsymbol{m}}$ is given by the sum of the mean linear output and the prediction of the Gaussian process from the residuals. The covariance is described by the covariance for the zero mean case plus a non negative term due to the prior mean structure.

Nevertheless, the case where there is no noise in the observation is not always applicable.

Observations with noise

In some cases, observations are often tainted by measurement noise. Hence, the observations $y(\boldsymbol{x}_i)$ are expressed by the Gaussian process model given by

$$y(\boldsymbol{x}_i) = f(\boldsymbol{x}_i) + \epsilon(\boldsymbol{x}_i), \quad \text{for } i = 1, \ldots, n, \tag{3.10}$$

where $\epsilon(\boldsymbol{x}_i)$ denotes the Gaussian noise related to the observations. We assume that $\epsilon(\boldsymbol{x}_i)$ is additive and independent identically distributed with zero mean and variance σ_ϵ^2, that is $\epsilon(\boldsymbol{x}_i) \sim \mathcal{N}(0, \sigma_\epsilon^2)$ for $i = 1, \ldots, n$ (Rasmussen and Williams, 2005).

Therefore, using the Gaussian process model (3.10) for "noisy" observed values the joint distribution of the observed values $\boldsymbol{y}$ and the function values $\boldsymbol{f}_*$ under the Gaussian process prior with non zero mean is now given by

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{m}_* \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\epsilon^2 \boldsymbol{I} & \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}^*) \\ \boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{x}) & \boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{x}^*) \end{bmatrix} \right),$$

where σ_ϵ^2 is the noise variance and $\boldsymbol{I}$ is the $n \times n$ identity matrix.

Similarly to the noise free observations, the Gaussian process posterior over functions for observations with noise has mean vector

$$\bar{\boldsymbol{m}} = \boldsymbol{m}_* + \boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{x})(\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\epsilon^2 \boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{m}), \tag{3.11}$$

and covariance function given by

$$\bar{\boldsymbol{K}} = \operatorname{cov}(\boldsymbol{f}_*) + \boldsymbol{Q}^{\mathrm{T}}(\boldsymbol{H}(\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{H}^{\mathrm{T}})^{-1}\boldsymbol{Q}, \tag{3.12}$$

where

$$\boldsymbol{Q} = \boldsymbol{H}_* - \boldsymbol{H}(\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}_*)$$

and

$$\operatorname{cov}(\boldsymbol{f}_*) = \boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{x})(\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}^*).$$

From (3.12) it can be seen that the variance of the Gaussian process posterior, given by the diagonal entries of the covariance matrix, is equal to the variance of the Gaussian process prior minus a positive term depending on the observations. Thus, the variance

of the Gaussian process posterior is always less than the variance of the prior due to the fact that the training data provide additional information.

3.2.6 Nugget effect

A simulator is deterministic indicating that repeating inputs produce the same outputs and therefore the data are observed with no error; that is if $\boldsymbol{x} = \boldsymbol{x}'$ then $f_t(\boldsymbol{x}) = f_t(\boldsymbol{x}')$. Hence, in a deterministic simulator, the **nugget** effect σ_ϵ^2 of the Gaussian process model is set equal to zero.

However, Ababou et al. (1994) recommended the addition of the nugget term in the Gaussian process model, since it improves the **condition number** of the covariance matrix by reducing it. The condition number can be defined as the ratio of the largest eigenvalue of the covariance matrix to its smaller eigenvalue (Press et al., 2007). The small eigenvalues of the covariance matrix make the matrix close to a singular matrix and numerical problems are arising when the matrix needs to be inverted, which is a common calculation involved in fitting a Gaussian process to data. The problem of the condition number comes from a lack of linear independence in the training set (Andrianakis and Challenor, 2012). Radford (1997) characterised desirable the addition of a small amount of error to the covariance function and mentioned that it is not going to result a significant alteration of the statistical properties of the Gaussian process model. Therefore, we include a small nugget effect due to the computational issues arising when inverting the covariance matrix.

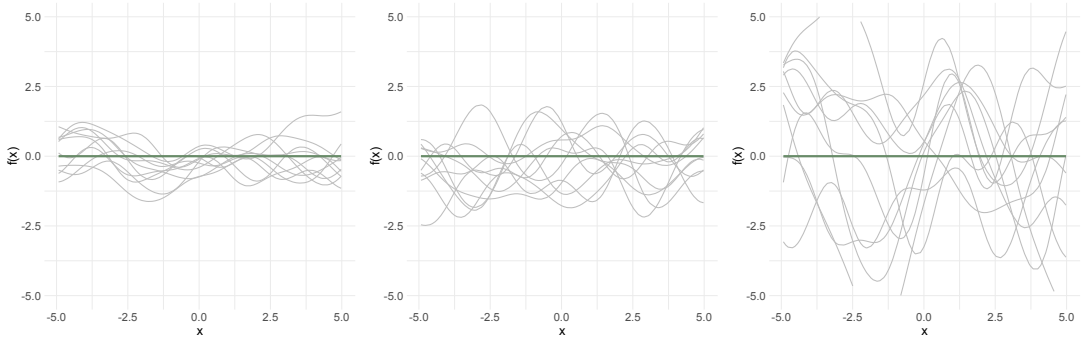
A nugget term could be specified, and does not necessarily need estimating. But in the noisy case, we would need to estimate the noise variance. For this thesis we do not work with noisy observations.

3.2.7 Hyperparameters

The squared exponential covariance function (3.4) has two hyperparameters given by the signal variance parameter τ^2 and the characteristic length-scale parameter l, that is $\boldsymbol{\zeta} = (\tau^2, l)^{\mathrm{T}}$. The length-scale parameter indicates the fall-off rate of the correlation and the signal variance parameter establishes the variation of the distance between the function and the mean. We examine how the variation of the hyperparameters affects different elements of the related prior distribution and subsequently the posterior distribution. Figures 3.2 and 3.3 illustrate random samples from the prior Gaussian process with zero mean, given by the green line, for different values of hyperparameters showing the effect of each hyperparameter on the prior.

Signal variance parameter

Figure 3.2 shows the effect of the signal variance parameter τ^2 on the prior keeping the length-scale parameter l constant. As τ^2 increases the amplitude of the intervals increases as well. Small values of τ^2 characterise functions that stay close to their mean value compared to larger values that allow more variation. This indicates that τ^2 is related with the amount of variation of the Gaussian process (Le Gratiet, 2013).



(A) $\tau^2 = 0.5, l = 1$ (B) $\tau^2 = 1, l = 1$ (C) $\tau^2 = 5, l = 1$

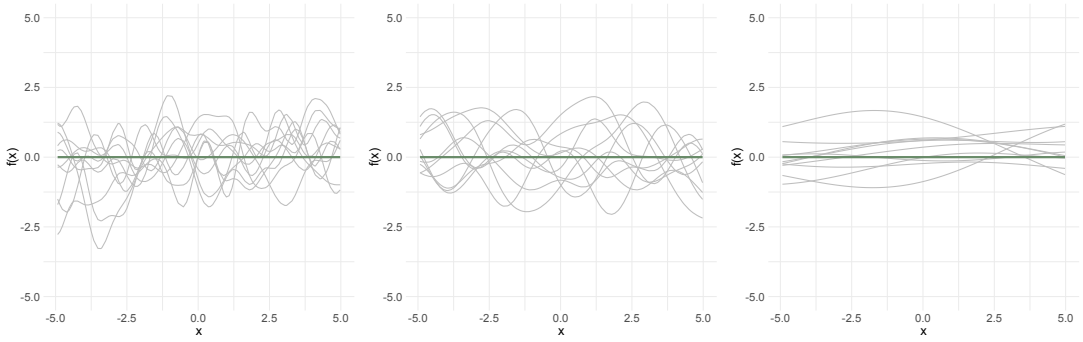
FIGURE 3.2: Random samples from the Gaussian process prior with zero mean, given by the green line, and squared exponential covariance function. The signal variance parameter τ^2 varies and the length-scale parameter remains constant, $l = 1$.

Length-scale parameter

Similarly, Figure 3.3 shows the effect of the length-scale parameter l on the prior keeping τ^2 constant. As can be seen, when the length-scale parameter decreases the correlation falls off more quickly. That is, for small values of the length-scale parameter the points which are further away are less correlated (Rasmussen and Williams, 2005) demonstrating that l affects the oscillation frequencies. Specifically, for small values of l the function oscillates rapidly and for large values of l the function oscillates slowly.

3.2.8 Hyperparameter estimation

The hyperparameters of the covariance function can be estimated directly from the training data. In many cases, we are not able to determine all the elements that specify the covariance function such as the hyperparameters $\boldsymbol{\zeta} = (\tau^2, l)^{\mathrm{T}}$ and we have to estimate them. The choice of the hyperparameters is important since the covariance function has a serious impact on the effectiveness of the Gaussian process regression model. Unsuitable choices of hyperparameters result in over-fitting the approximation as well as in significant effect on credible intervals.



(A) $\tau^2 = 1, l = 0.5$ (B) $\tau^2 = 1, l = 1$ (C) $\tau^2 = 1, l = 5$

FIGURE 3.3: Random samples from the Gaussian process prior with zero mean, given by the green line, and squared exponential covariance function. The length scale parameter l varies and the signal variance parameter remains constant, $\tau^2 = 1$.

The hyperparameters provide flexible customisation of the Gaussian process in multiple practical applications. The optimisation procedure used is called the *training* of the Gaussian process (Blum and Riedmiller, 2013). There are several estimation criteria in the literature that can be followed for the hyperparameter estimation.

The approach we take for hyperparameter estimation is based on the idea that we would like to make predictions for new data using the Gaussian process regression model. Hence, we require to know the coefficient vector $\mathbf{b}$ of the fixed basis functions given in (3.9) and to be able to evaluate the covariance function for any input points given the hyperparameters $\boldsymbol{\zeta}$. Therefore, we need to estimate $\mathbf{b}$ and $\boldsymbol{\zeta}$ from the data.

We approach the hyperparameter estimation problem using maximum likelihood techniques. Another method that could be used is the Bayes approach which is useful when we want to take into consideration prior information about the parameters.

A method of estimating these parameters it to maximise the likelihood given in (3.5) as a function of $\mathbf{b}$ and $\boldsymbol{\zeta}$, that is

$$\hat{\mathbf{b}}, \hat{l}, \hat{\tau^2} = \arg\max_{\boldsymbol{b}, l, \tau^2} \log P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{b}, l, \tau^2),$$

where $\hat{\mathbf{b}}, \hat{l}, \hat{\tau^2}$ are estimates of $\mathbf{b}, l$ and τ^2 respectively. The marginal log-likelihood function is obtained from a multivariate normal distribution

$$\begin{aligned}\log P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{b}, l, \tau^2) = &-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{H}\boldsymbol{b})^{\mathrm{T}}[\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\epsilon^2 \boldsymbol{I}_n]^{-1}(\boldsymbol{y} - \boldsymbol{H}\boldsymbol{b}) \\ &- \frac{n}{2}\log 2\pi - \frac{1}{2}\log|\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\epsilon^2 \boldsymbol{I}_n|.\end{aligned}$$

To estimate the parameters we compute first $\hat{\boldsymbol{b}}(l, \tau^2)$ which maximises the log-likelihood function with respect to $\boldsymbol{b}$ for given l and τ^2. Then we estimate the b-profile

log-likelihood over l and τ^2:

$$\log P(\boldsymbol{y}|\boldsymbol{x}, \hat{\boldsymbol{b}}(l,\tau^2), l, \tau^2).$$

The estimate of $\boldsymbol{b}$ for given l and τ^2 is

$$\hat{\boldsymbol{b}}(l,\tau^2) = (\boldsymbol{H}[\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x}) + \sigma_\epsilon^2 \boldsymbol{I}_n]^{-1}\boldsymbol{H}^{\mathrm{T}})^{-1}\boldsymbol{H}[\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x}) + \sigma_\epsilon^2 \boldsymbol{I}_n]^{-1}\boldsymbol{y}.$$

Then, the b-profile log-likelihood is given by

$$\begin{aligned}\log P(\boldsymbol{y}|\boldsymbol{x}, \hat{\boldsymbol{b}}(l,\tau^2), l, \tau^2) = &-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{b}}(l,\tau^2))^{\mathrm{T}}[\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x}) + \sigma_\epsilon^2 \boldsymbol{I}_n]^{-1}(\boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{b}}(l,\tau^2)) \\ &- \frac{n}{2}\log 2\pi - \frac{1}{2}\log|\boldsymbol{K}(\boldsymbol{x},\boldsymbol{x}) + \sigma_\epsilon^2 \boldsymbol{I}_n|.\end{aligned}$$

We maximise the b-profile log-likelihood over l and τ^2 to find their estimates. Bayesian inference can also be used to estimate the hyperparameters analytically using appropriate prior distributions (Oakley and O'Hagan, 2002).

3.3 Hierarchical computer experiments

3.3.1 Introduction

Complex approximations can provide a better approximation to reality when comparing with a simpler approximation. However, complex approximations are often computationally expensive due to the large number of equations that need to be solved, or the high resolution of the approximation. Occasionally, an evaluation of a complex approximation could take a few hours or even days to run.

A simulator can be usually run at various levels of complexity each with different accuracy. Kennedy and O'Hagan (2000) explored numerous ways in which evaluations from different levels of approximations could be used for predicting the output of the most complex approximation. The objective of the current section is to illustrate how we can obtain information about the highest level approximation, which is the most "expensive" and accurate one by combining evaluations of lower level approximations, which are "cheaper" but less accurate. The aim is to reduce the evaluations of the high level approximation and decrease the computational cost.

3.3.2 Regression for multilevel simulators

The main object of interest here is the calculation of the posterior distribution of the expensive approximation given the observations from evaluations of the expensive and

cheaper approximations. The posterior distribution can be then used for predictions. In this section, we present a Bayesian approach of constructing an approximation which is based on the hierarchy of the levels of the simulator in line with Kennedy and O'Hagan (2000). Our aim is to apply this approach to approximate intractable or computationally expensive likelihoods using various existing levels of likelihood approximation.

Firstly, we have to look at some assumptions. Smoothness of the function being modelled is a crucial assumption that needs to be taken into consideration when building an approximation for a simulator. Kennedy et al. (2006) mentioned that the output of a simulator has to be a smooth function of its inputs. That is, for similar input values the corresponding output values have to be close. We assume that we can assign a Gaussian process prior to each level of the approximation and that each simulator output is scalar. It is also assumed that there is some correlation between the different levels of the same approximation and they contain information about one another.

For modeling the relation between the levels of the approximation Kennedy and O'Hagan (2000) suggested that an **autoregressive model** could be used. Autoregressive models assume that observations obtained from previous steps in approximation hierarchy are useful for the prediction of the value at the next step. When working with autoregressive models observations from previous steps are used in a regression equation as inputs to predict the output of the next step.

Let us introduce some mathematical notation that we will use for the multi-level regression. Consider that we have s levels of simulators given by $f_1(\boldsymbol{x}), \ldots, f_s(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathbb{R}^d$, where s is a positive integer. Let $f_t(.)$ be a simulator indexed by a p-dimensional input vector $\boldsymbol{x}$ for the levels $t = 1, \ldots, s$.

For each level of simulator, $t = 1, \ldots, s$, let $D_t = \{\boldsymbol{x}_1^{(t)}, \ldots, \boldsymbol{x}_{n_t}^{(t)}\}$ be the experimental design at level t consisting of n_t design points. Let $\boldsymbol{f}^{\mathrm{T}}$ be the column vector of responses given by $\boldsymbol{f}^{\mathrm{T}} = (\boldsymbol{f}_1^{\mathrm{T}}, \ldots, \boldsymbol{f}_s^{\mathrm{T}})$, where $\boldsymbol{f}_t^{\mathrm{T}}$ is the vector of outputs for the level t simulator and can be written as $\boldsymbol{f}_t^{\mathrm{T}} = (f_t(x_1^{(t)}), \ldots, f_t(x_{n_t}^{(t)}))$, where $^{\mathrm{T}}$ is the transpose. Let $f_s(.)$ be the highest level. We aim to compute the posterior distribution of $f_s(.)$ given the outputs at training points at all levels $1, \ldots s$. Kennedy and O'Hagan (2000) assume that the design points at different levels are nested. We generalise their results so that this assumption is not required.

The **Markov property** indicates that if the nearest point $f_{t-1}(\boldsymbol{x})$ is given, there is nothing else one can learn concerning $f_t(\boldsymbol{x})$ from any other run of $f_{t-1}(\boldsymbol{x}')$ for $\boldsymbol{x} \neq \boldsymbol{x}'$, where $\boldsymbol{x}$ and $\boldsymbol{x}'$ are input vectors. For all $\boldsymbol{x} \neq \boldsymbol{x}'$ the Markov property can be written as

$$\mathrm{cov}\{f_t(\boldsymbol{x}), f_{t-1}(\boldsymbol{x}') | f_{t-1}(\boldsymbol{x})\} = 0. \tag{3.13}$$

Using an autoregressive model we have a hierarchy of the s levels; from the least complicated and least accurate to the most complex one (Le Gratiet, 2011). Using the

Markov property from (3.13) along with the stationarity of $f_t(\boldsymbol{x})$ over the input space for each level t of the simulator the autoregressive model is given by

$$\boldsymbol{f}_t(\boldsymbol{x}) = \rho_{t-1}\boldsymbol{f}_{t-1}(\boldsymbol{x}) + \boldsymbol{\delta}_t(\boldsymbol{x}) \quad \text{for} \quad t = 2, \ldots, s, \tag{3.14}$$

where ρ_{t-1} gives a kind of regression parameter between $\boldsymbol{f}_t(.)$ and $\boldsymbol{f}_{t-1}(.)$, called the **autoregressive parameter**, and $\boldsymbol{\delta}_t(.)$ is a Gaussian process independent of $\boldsymbol{f}_{t-1}(.), \ldots, \boldsymbol{f}_t(.)$ representing the difference between the levels of the model. Conditioning on the parameters l_t and τ_t^2, $\boldsymbol{\delta}_t(.)$ can be modelled as a stationary Gaussian process with mean function $h(.)^{\mathrm{T}}b_t$ and covariance function given by $K_t(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{cov}\{\boldsymbol{\delta}_t(\boldsymbol{x}), \boldsymbol{\delta}_t(\boldsymbol{x}')\}$ for each t. The prior mean structure $h(.)$ is a vector of q regression functions.

Each level in hierarchy can be statistically modelled as a stationary Gaussian process with a particular mean function and covariance function. For example, it is assumed that the simplest model $\boldsymbol{f}_1(.)$ has a stationary Gaussian process when conditioning on b_1 and σ_1^2 which is independent of the $\boldsymbol{\delta}_t(.)$ (Kennedy and O'Hagan, 2000).

The choice of the covariance function is crucial since it controls the smoothness of the model as described in Section 3.2.2. There are various forms of covariance functions that one can choose from. For the purpose of this thesis, for each level of computer model we assume that the covariance function will have a squared exponential form and is given by

$$K_t(\boldsymbol{x}, \boldsymbol{x}') = \tau_t^2 \exp\left[-\frac{(\boldsymbol{x} - \boldsymbol{x}')^2}{2l_t^2}\right] = \tau_t^2 \exp\left[-\frac{(\boldsymbol{x} - \boldsymbol{x}')^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}')}{2l_t^2}\right], \tag{3.15}$$

where τ_t^2 is the variance parameter and l_t is the length-scale parameter for level t. Recall that the squared exponential form of the covariance function in (3.15) is infinitely differentiable indicating that the model has a high degree of smoothness. Considering the calculations related to the covariance, we denote $\boldsymbol{K}_t(D_k, D_l)$ the covariance matrix between points in D_k and D_l. The covariance matrix $\boldsymbol{K}_t(D_k, D_l)$ is a $n_k \times n_l$ matrix and for all $x_i^{(k)} \in D_k$ and $x_j^{(l)} \in D_l$ the ij^{th} element is given by

$$[\boldsymbol{K}_t(D_k, D_l)]_{i,j} = \tau_t^2 \exp\left[-\frac{(x_i^{(k)} - x_j^{(l)})^{\mathrm{T}}(x_i^{(k)} - x_j^{(l)})}{2l_t^2}\right].$$

We also introduce the notation $\boldsymbol{K}_t(D_k) = \boldsymbol{K}_t(D_k, D_k)$.

3.3.3 Two-level simulator

In this section, the case of a two-level simulator, $s = 2$, is described. The aim is to apply the GP emulation for multi-level simulators presented in Section 3.3.2 and compute the posterior distribution for a simulator which has two levels. Assume that there are

two levels: $f_1(.)$ representing the less accurate and cheaper model and $f_2(.)$ representing the more accurate and expensive model. In other words, we want to compute the conditional distribution of the output $f_2(.)$ given the hyperparameters and the observations from $f_1(.)$ which is the output from the lower level of the simulator.

From the autoregressive model (3.14) we want to approximate the expensive approximation by multiplying the cheap model with a scalar factor ρ_1 and adding a Gaussian process $\delta_2(.)$, where $\delta_2(.)$ gives the difference between $\rho_1 f_1(.)$ and $f_2(.)$ (Alexander et al., 2007). Hence, we have that

$$f_2(\boldsymbol{x}) = \rho_1 f_1(\boldsymbol{x}) + \delta_2(\boldsymbol{x}).$$

For simplicity, we can assume that the regression parameter ρ_1 represents a constant factor between $f_1(.)$ and $f_2(.)$. The hyperparameters of the Gaussian process are given by $\boldsymbol{\zeta} = (\tau_1^2, \tau_2^2, l_1, l_2, \rho_1)$ and $\boldsymbol{b} = (\boldsymbol{b}_1, \boldsymbol{b}_2)^{\mathrm{T}}$.

In order to calculate the required posterior distribution we need two sets of training points, one for the cheap model, D_1, and a smaller one for the expensive model, D_2. Let the number of points of the cheap and expensive approximations be n_1 and n_2 respectively.

Firstly, we have to compute $\boldsymbol{f}$ which is the joint distribution of f_1 and f_2, where $\boldsymbol{f}^{\mathrm{T}} = (\boldsymbol{f}_1^{\mathrm{T}}, \boldsymbol{f}_2^{\mathrm{T}})$. That is

$$\boldsymbol{f} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \text{ with sizes } \begin{bmatrix} n_1 \times 1 \\ n_2 \times 1 \end{bmatrix}.$$

We have that $f_1(.)|b_1, \tau_1^2, l_1 \sim \mathcal{GP}(h(.)^{\mathrm{T}} b_1, K_1)$ with design points from D_1 and $\delta_2(.)|b_2, \tau_2^2, l_2 \sim \mathcal{GP}(h(.)^{\mathrm{T}} b_2, K_2)$ with design points from D_2. Note that both of the Gaussian processes are stationary and $f_1(.)$ is independent of $\delta_2(.)$. Therefore, the joint distribution of $\boldsymbol{f}$ given $\boldsymbol{\zeta}$ and $\boldsymbol{b}$ is a multivariate normal distribution and can be written as

$$\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{m}_f, \boldsymbol{V}_f),$$

where the mean is given by

$$\boldsymbol{m}_f = \begin{bmatrix} h(.)^{\mathrm{T}} \boldsymbol{b} \\ \rho_1 h(.)^{\mathrm{T}} \boldsymbol{b} \end{bmatrix}, \tag{3.16}$$

and the covariance matrix can be obtained in block form using the function $V_f(.,.)$ given by

$$V_f(D_1, D_2) = \begin{bmatrix} \boldsymbol{K}_1(D_1) & \rho_1 \boldsymbol{K}_1(D_1, D_2) \\ \rho_1 \boldsymbol{K}_1(D_1, D_2) & \rho_1^2 \boldsymbol{K}_1(D_2) + \boldsymbol{K}_2(D_2) \end{bmatrix}. \tag{3.17}$$

The covariance matrix $\boldsymbol{V}_f$ obtained from (3.17) includes the covariance matrix between the data points of the expensive model $\boldsymbol{K}_2(D_2)$, the cheap model $\boldsymbol{K}_1(D_1)$ and the cross-covariance between the points for the cheap and expensive approximations $\boldsymbol{K}_1(D_1, D_2)$. The entries of the covariance matrix $\boldsymbol{V}_f$ are derived by

$$\text{cov}\{\boldsymbol{f}_1(D_1), \boldsymbol{f}_1(D_1)\} = \boldsymbol{K}_1(D_1, D_1),$$

$$\begin{aligned}\text{cov}\{\boldsymbol{f}_1(D_1), \boldsymbol{f}_2(D_2)\} &= \text{cov}\{\boldsymbol{f}_1(D_1), \rho_1\boldsymbol{f}_1(D_2) + \boldsymbol{\delta}_2(D_2)\} \\ &= \rho_1\boldsymbol{K}_1(D_1, D_2),\end{aligned}$$

$$\begin{aligned}\text{cov}\{\boldsymbol{f}_2(D_2), \boldsymbol{f}_2(D_2)\} &= \text{cov}\{\rho_1\boldsymbol{f}_1(D_2) + \boldsymbol{\delta}_2(D_2), \rho_1\boldsymbol{f}_1(D_2) + \boldsymbol{\delta}_2(D_2)\} \\ &= \rho_1^2\text{cov}\{\boldsymbol{f}_1(D_2), \boldsymbol{f}_1(D_2)\} + \text{cov}\{\boldsymbol{\delta}_2(D_2), \boldsymbol{\delta}_2(D_2)\} \\ &= \rho_1^2\boldsymbol{K}_1(D_2) + \boldsymbol{K}_2(D_2).\end{aligned}$$

For the examples presented in this thesis, we are using a variety of prior mean structures, $h(.)$, based on each example such as constant, linear or quadratic prior mean functions, but any prior mean structure could be used as well.

To compute the posterior distribution for $\boldsymbol{f}_2$ for a simulator with two levels we use the results from Section 3.2.4. Firstly, we find the joint distribution of training and test data. To achieve that, we extend the training points of the high-level D_2 to include the test points $\boldsymbol{x}^*$, that is $D_2 \cup \boldsymbol{x}^*$. The function (3.17) which is used to compute the covariance matrix of the joint distribution can be written as

$$V_f(D_1, D_2 \cup \boldsymbol{x}^*) = \begin{bmatrix} \boldsymbol{K}_1(D_1) & \rho_1\boldsymbol{K}_1(D_1, D_2 \cup \boldsymbol{x}^*) \\ \rho_1\boldsymbol{K}_1(D_2 \cup \boldsymbol{x}^*, D_1) & \rho_1^2\boldsymbol{K}_1(D_2 \cup \boldsymbol{x}^*) + \boldsymbol{K}_2(D_2 \cup \boldsymbol{x}^*) \end{bmatrix}. \tag{3.18}$$

Following standard results of the conditional distribution of the multivariate normal distribution given in Section 3.2.4 we partition $V_f(D_1, D_2 \cup \boldsymbol{x}^*)$ into blocks given by

$$V_f(D_1, D_2 \cup \boldsymbol{x}^*) = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \text{ with sizes } \begin{bmatrix} n_{\text{train}} \times n_{\text{train}} & n_{\text{train}} \times n_{\text{test}} \\ n_{\text{test}} \times n_{\text{train}} & n_{\text{test}} \times n_{\text{test}} \end{bmatrix}, \tag{3.19}$$

where subscript $_1$ denotes the training data of the low-level, subscript $_2$ the training data of the high-level and the test data. n_{train} and n_{test} are the number of training and test data respectively. The partition splits $V_f(D_1, D_2 \cup \boldsymbol{x}^*)$ into blocks according to training and test data, so that V_{22} contains the last n_{test} rows and columns of the covariance matrix. The main purpose of this approach is that this can be generalised to $s = k$ levels of approximation and the posterior distribution of the high-level approximation can be considered as a conditional distribution using the idea of adding the test data as an extension of the high-level data.

Following the conditional equations for multivariate normal distribution from Section 3.2.4, we want to compute the conditional distribution

$$Y_{\text{test}}|Y_{\text{train}} = \boldsymbol{y}_{\text{train}} \sim \mathcal{N}(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{V}}), \tag{3.20}$$

where

$$\bar{\boldsymbol{\mu}} = \boldsymbol{\mu}_2 + \boldsymbol{V}_{21}\boldsymbol{V}_{11}^{-1}(\boldsymbol{y}_{\text{train}} - \boldsymbol{\mu}_1), \tag{3.21}$$

$$\boldsymbol{\mu}_1 = \boldsymbol{H}\hat{\boldsymbol{b}} \quad \text{and} \quad \boldsymbol{\mu}_2 = h'(\boldsymbol{x}^*)^{\mathrm{T}}\hat{\boldsymbol{b}},$$

with

$$h'(\boldsymbol{x})^{\mathrm{T}} = (\rho_1 h(\boldsymbol{x})^{\mathrm{T}}, h(\boldsymbol{x})^{\mathrm{T}}),$$

$$\boldsymbol{H} = \begin{pmatrix} h(x_1^{(1)})^{\mathrm{T}} & 0 \\ \vdots & \vdots \\ h(x_{n_1}^{(1)})^{\mathrm{T}} & 0 \\ \rho_1 h(x_1^{(2)})^{\mathrm{T}} & h(x_1^{(2)})^{\mathrm{T}} \\ \vdots & \vdots \\ \rho_1 h(x_{n_2}^{(2)})^{\mathrm{T}} & h(x_{n_2}^{(2)})^{\mathrm{T}} \end{pmatrix},$$

$$\hat{\boldsymbol{b}} = (\hat{\boldsymbol{b}_1}, \hat{\boldsymbol{b}_2})^{\mathrm{T}} = (\boldsymbol{H}^{\mathrm{T}}\boldsymbol{V}_{11}^{-1}\boldsymbol{H})^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{V}_{11}^{-1}\boldsymbol{y}_{\text{train}}, \tag{3.22}$$

where $\boldsymbol{h}_{\text{train}}$ are the prior evaluations of each approximation level at the training points, $\boldsymbol{y}_{\text{train}}$ are the evaluations of the approximation at the training points and $\hat{\boldsymbol{b}}$ is the posterior mean of $\boldsymbol{b}$.

The covariance matrix is given by

$$\bar{\boldsymbol{V}} = \boldsymbol{V}_{22} - \boldsymbol{V}_{21}\boldsymbol{V}_{11}^{-1}\boldsymbol{V}_{12}. \tag{3.23}$$

Therefore, (3.20) can be considered as the posterior distribution of the high-level approximation for a simulator with two levels and it is a Gaussian process. The posterior mean function can be used as an approximation for the highest level of approximation and to predict the output of the simulator at untried inputs. In addition, the posterior mean function can be more accurate when an adequate number of evaluations of the cheap approximation are provided compared with using only evaluations of the expensive approximation (Kennedy and O'Hagan, 2000). This will be demonstrated with examples in later sections. The methodology presented in this section can be also extended for more than two levels of approximation.

3.3.4 Extending for more levels

For situations were there are more than two levels of approximation, where $s > 2$, the normality of each level of simulator, given by $\boldsymbol{f}_s(.)$, conditional on all the hyperparameters and the observed runs is still valid. The resulting Gaussian process from using more than two levels of approximation will have the same format for the mean (3.21) and the covariance matrix (3.23) as in the two-level case following the conditional distribution results form Section 3.2.4. However, the forms for the function V_f, the vector $h'(x)$ and the matrix $\boldsymbol{H}$ are different, and are given in equations (3.24), (3.25), (3.26) and (3.27) respectively.

The covariance function V_f will have s blocks, where s is the number of levels used. The $(1,1)$ block of V_f will be $V_f^{(1,1)} = \boldsymbol{K}_1(D_1)$ and for $k > 1$, where k is an integer, the (k,k) block will be

$$\begin{aligned} V_f^{(k,k)} &= \boldsymbol{K}_k(D_k) + \rho_{k-1}^2 \boldsymbol{K}_{k-1}(D_k) \\ &\quad + \rho_{k-1}^2 \rho_{k-2}^2 \boldsymbol{K}_{k-2}(D_k) + \cdots + \left(P_i^{k-1}\right)^2 \boldsymbol{K}_1(D_k), \end{aligned} \tag{3.24}$$

where $P_i^j = \prod_{n=i}^{j} \rho_n$.

The off-diagonal (k,l) blocks for $k < l$ are given by

$$\begin{aligned} V_f^{(k,l)} &= P_k^{l-1} \boldsymbol{K}_k(D_k, D_l) + \rho_{k-1} P_{k-1}^{l-1} \boldsymbol{K}_{k-1}(D_k, D_l) \\ &\quad + P_{k-2}^{k-1} P_{k-2}^{l-1} \boldsymbol{K}_{k-2}(D_k, D_l) \\ &\quad + \cdots + P_2^{k-1} P_2^{l-1} \boldsymbol{K}_2(D_k, D_l) + P_1^{k-1} P_1^{l-1} \boldsymbol{K}_1(D_k, D_l). \end{aligned} \tag{3.25}$$

Moreover, the vector $h'(x)$ is given by

$$h'(x)^{\mathrm{T}} = (P_i^{s-1}, P_2^{s-1}, \ldots, \rho_{s-1}, 1), \tag{3.26}$$

with $h(x) = 1$ and the related matrix $\boldsymbol{H}$ is a lower diagonal matrix given by

$$\boldsymbol{H} = \begin{pmatrix} 1_{n_1} & & & & & \\ \rho_1 1_{n_2} & 1_{n_2} & & & 0 & \\ \rho_1\rho_2 1_{n_3} & \rho_2 1_{n_3} & 1_{n_3} & & & \\ P_1^3 1_{n_4} & \rho_2\rho_3 1_{n_4} & \rho_3 1_{n_4} & 1_{n_4} & & \\ & \vdots & & & & \vdots \\ P_1^{s-1}\rho_{n_s} 1_{n_s} & & \cdots & & \rho_{s-2}\rho_{s-1} 1_{n_s} & \rho_{s-1} 1_{n_s} & 1_{n_s} \end{pmatrix}. \tag{3.27}$$

The hyperparameter estimation for the model with more than two levels follows the same methods with the hyperparameter estimation approach presented in Section 3.2.8.

For the purposes of this thesis we demonstrate examples using up to three-levels of approximation, where $s = 3$. The same methodology of multi-level GP approximations can be used for larger values of s.

3.4 Example models

3.4.1 Overview of the examples

In this Section, we present the data used for the three example models, introduced in Chapter 2, which are used as running examples throughout the thesis. We have a simple linear regression model where the closed form of the likelihood can be computed analytically, a generalised linear mixed model which illustrates the intractable likelihood problem and an Ising model where the computation of the normalising constant of the likelihood is extremely expensive for large grids.

3.4.2 Simple linear regression model

Recall the simple linear regression model introduced in Section 2.2 given by

$$y_i = \alpha_0 + \alpha_1 x_i + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, \sigma_r^2), \tag{3.28}$$

where y_i is the response value for each observation i for $i = 1, \ldots, n$, α_0 is the intercept, α_1 is the slope, x_i is the predictor value for each observation i and ϵ_i is the error term which is normally distributed with zero mean and variance σ_r^2.

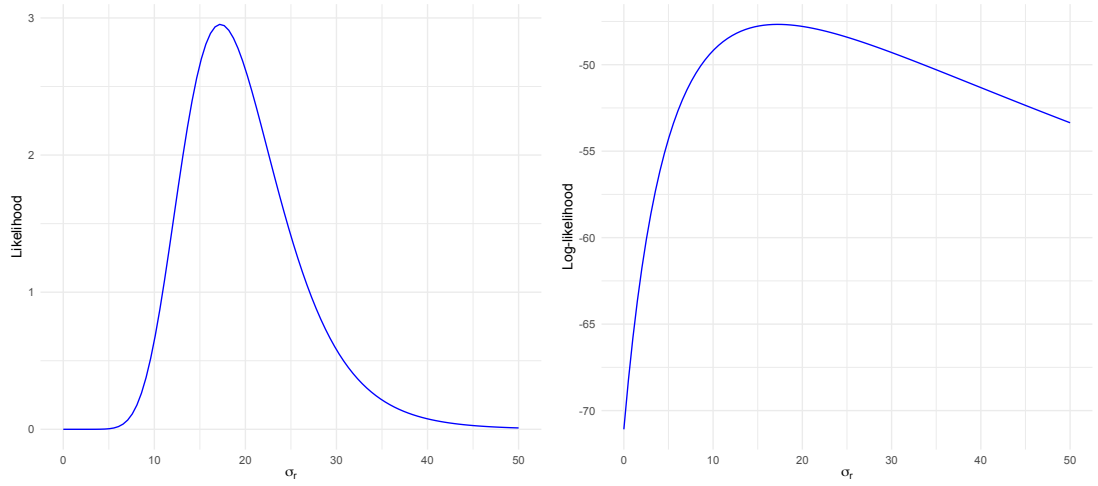
We start by generating some artificial data. Assume that we have a continuous explanatory variable x, where $x = 1, 2, \ldots, 16$, $n = 16$, and a measured response y for each x. We set the intercept α_0 equal to 40 and a unit increase in x is associated with a 1.5 decrease in y, that is $\alpha_1 = -1.5$. The data were drawn from a normal distribution with zero mean and variance of 25, that is $\epsilon_i \sim \mathcal{N}(0, 5^2)$.

The likelihood function is given by

$$L(\alpha_0, \alpha_1, \sigma_r^2; x_1, \ldots, x_n) = (2\pi\sigma_r^2)^{n/2} \exp\left(\frac{1}{2\sigma_r^2} \sum_{j=1}^{n} (y_j - (\alpha_0 + \alpha_1 x_j))^2\right). \tag{3.29}$$

For simplicity, after we fit the model to the data we set the parameters α_0 and α_1 equal to the model coefficient estimated value and we are interested in the relation between the likelihood and the standard deviation of the residuals σ_r. Figure 3.4 shows the standardise likelihood and log-likelihood for the specific data set as functions of the parameter of interest, σ_r. The values of the likelihood for this particular example are

really small, hence we compute the standardise likelihood by using a transformation such that the standard deviation of the likelihood evaluations at the training points is one.



(A) Likelihood (B) Log-likelihood

FIGURE 3.4: Likelihood and log-likelihood functions of the simple linear regression model as functions of σ_r at a fine grid of points.

For the SLR example we do not use any likelihood approximation. We treat this example as a toy example to show how GPs can be used as approximations of the likelihood for a single-level case. Therefore, we use a single-level GP regression to approximate the likelihood using only a small number of likelihood evaluations.

3.4.3 Generalised linear mixed model

Recall the two-level random intercept logistic model presented in Section 2.3.4 which is given by

$$\log\left(\frac{p_{ij}}{1-p_{ij}}\right) = \beta_0 + \beta_1 x_{ij} + \sigma u_i, \quad u_i \sim \mathcal{N}(0,1). \tag{3.30}$$

The data set that we will use results from a simulation of a binary two-level model and taken from the `glmmsr` package (Ogden, 2019). We have two observations for each cluster, $m_i = 2$, and 50 clusters, $m = 50$, so 100 observations in total. The number of observations for each group m_i is small and the model is a sparse model.

The likelihood for this special case can be simplified to

$$L(\boldsymbol{\beta}, \sigma) = \prod_{i=1}^{m} \int_{\mathbb{R}} \prod_{j=1}^{m_i} f(y_{ij}|\eta_{ij} = x_{ij}^{\mathrm{T}}\beta + u_i)\phi(u_i, 0, \sigma^2)du_i. \tag{3.31}$$

As previously mentioned, the likelihood for a GLMM is not analytically tractable for this type of models and needs to be approximated. In order to fit and analyse generalised linear mixed models we can use the package `glmmsr` in `R` which fits GLMMs with various approximation methods. The log-likelihood function is calculated using LA and AGQ methods for different nAGQ in the quadrature formula.

Using the Laplace approximation the fixed parameter estimates were $\hat{\beta}_0 = 0.6525, \hat{\beta}_1 = -1.1583$ and the random effect parameter estimate $\hat{\sigma} = 0.7484$. However, for the specific data set we have that the data are sparse and the Laplace approximation might not provide a good approximation to the likelihood. Consequently, a more accurate approximation method is required.

The likelihood can be simplified to the form of equation (3.31) for a two-level model. In this situation AGQ could be used for different number of nAGQ in the quadrature formula to approximate each of the one dimensional integrals. Using the AGQ approximation with 10 quadrature points (nAGQ = 10) the fixed parameter estimates were $\hat{\beta}_0 = 0.7168, \hat{\beta}_1 = -1.2734$ and the random effect parameter estimate was $\hat{\sigma} = 1.041$.

The estimate of the standard deviation of the random effect σ is considerably higher as nAGQ increases compared with the one obtained from the Laplace approximation to the likelihood, where nAGQ = 1. Therefore, the inference for the specific parameter will be different based on the method and accuracy of the approximation method used.

In order to check why this occurs, we investigate in Figure 3.5 a cut across the various approximate log-likelihood surfaces for σ, setting the parameters (β_0, β_1) equal to their estimated value obtained from model fitting, $(\hat{\beta}_0, \hat{\beta}_1)$, for various values of σ, that is $l(\sigma, (\hat{\beta}_0, \hat{\beta}_1))$. An accurate approximation, which is available for this case is shown with the red curve. As can be seen, the maximum of each approximation is different. For larger values of σ the approximations obtained using small nAGQ considerably underestimate the likelihood and each approximation has different maximum value. The aim is to obtain an accurate approximation to the high-level of approximation of the likelihood.

3.4.4 Ising model

Recall that for the Ising model example we consider a special case of Ising models where we set $\alpha = 0$, and that the lattice has periodic boundaries as shown in Figure 2.1. We also have $n = m = c$. Hence, the model we are looking at is

$$\text{pr}(Y = y; \theta) = Z_{c,c}(\theta)^{-1} \exp(\beta V_1(y)), \tag{3.32}$$

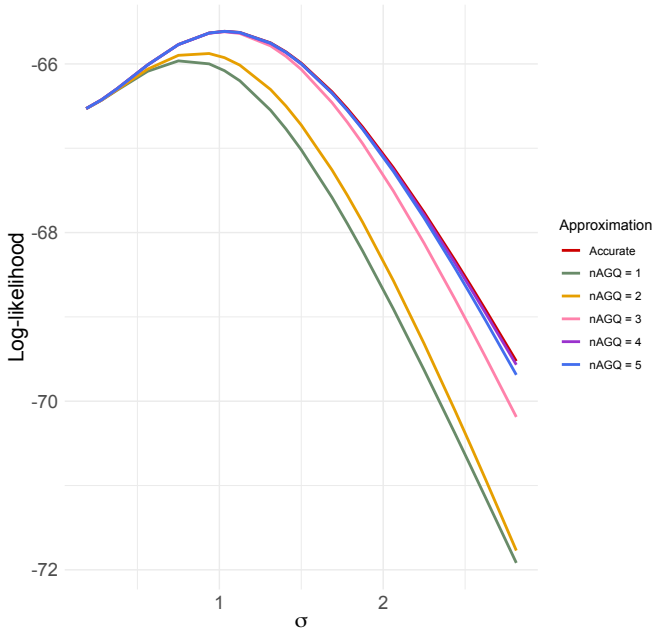


FIGURE 3.5: Cut across the approximate log-likelihood surface of σ of the two-level random intercept model approximated using AGQ method for different nAGQs for varying values of σ and keeping the remaining model parameters β_0 and β_1 constant.

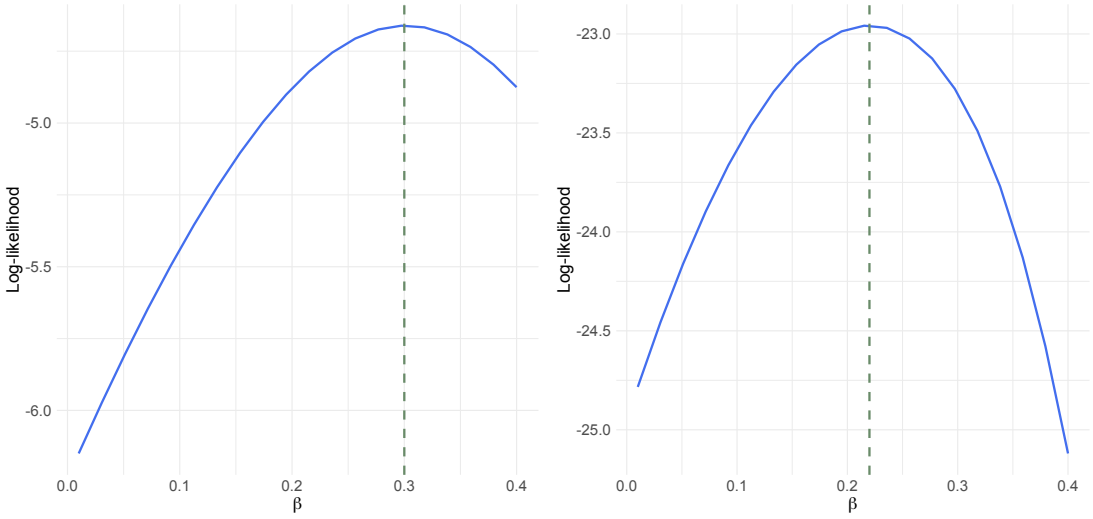
where $V_1(y) = \sum_{i\sim j} y_i Y_j$ and β is a scalar indicating the inverse temperature. The normalising constant is given by

$$Z_{c,c}(\theta) = \sum_{y\in\{-1,1\}^v} \exp\{\beta V_1(y)\}, \tag{3.33}$$

and is computationally expensive for larger grids.

As mentioned above, we are looking at a special case of the parameter space setting $\alpha = 0$ and we assume that $\beta \in [0, 0.43]$. This is essential since for the critical value β_c, where $\beta_c = \log(1+\sqrt{2})/2 \approx 0.44$, the behaviour of the Ising model changes suddenly. For values of $\beta > \beta_c$ we observe large areas of all plus ones or all minus ones. Note that for $\beta = \beta_c$ the maximum likelihood estimator may not have a normal limiting distribution (Ogden, 2017).

Figure 3.6 shows the log-likelihood as a function of β for simulated data from the model using the `IsingSampler` function of the `IsingSampler` package (Epskamp and Boot, 2023) in `R`, with the specified value of β for the given parameter space, for various grids calculated using the formulas for the exact normalising constant given in Section 2.4.3. The value of β is different for each case and the data is a single grid of size 3×3 and 6×6 respectively. As can be seen, the location of the maximum value of the log-likelihood for each case is close to the value of β used to generate the data given by the green dashed line.



(A) 3×3, $\beta = 0.3$ (B) 6×6, $\beta = 0.22$

FIGURE 3.6: Log-likelihood for data simulated from the Ising model with a specific value of β for different grids using the exact normalising constant for a single grid.

We use the RDA method presented in Section 2.4.4 as an approximation method for the normalising constant using different levels of approximation. As in the examples above, we use simulated data, and for this example we have simulated data for a 10×10 grid. The log-likelihood computed using the RDA method for two different levels of approximation for the normalising constant is shown in Figure 3.7. We have $k = 2$ for the low-level approximation (green curve), $k = 4$ for the high-level (blue curve) and the exact calculation (red curve). The true value of β is 0.22 and is given by the orange dashed line. The maximum log-likelihood value for $k = 2$ is at $\beta = 0.277$, for the $k = 4$ at $\beta = 0.277$ and the exact value at $\beta = 0.256$. As can be seen, the maximum value of the log-likelihood obtained using the high-level approximation is closer to the exact value compared to the one obtained from the low-level approximation. This shows that it is sufficient to use approximations instead of exact calculations in cases where they are computationally expensive. Our approach and methodology aims to illustrate how an approximation of the normalising constant of the Ising model and therefore the log-likelihood can be formulated based on hierarchical models and multi-level GP approximations.

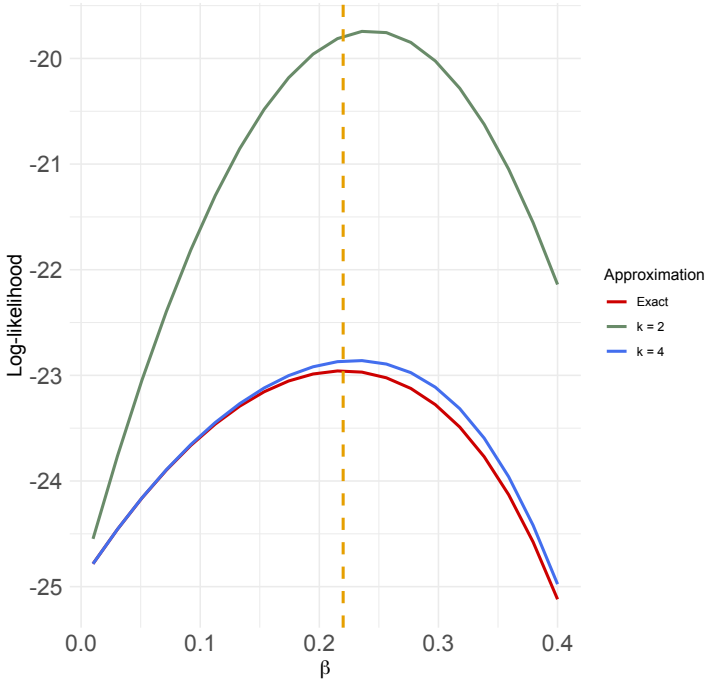


FIGURE 3.7: Log-likelihood using two approximation levels for the normalising constant, along with the exact log-likelihood, of the Ising model for a simulated example for a 10×10 grid. Parameter β used for the simulation is at 0.22 (orange dashed line).

3.5 Applications

3.5.1 Overview

The main goal of this section is to illustrate how one can use the methodology presented in this chapter to compute approximation of likelihood functions, by combining evaluations of approximations available in multiple levels of accuracy. We use Gaussian processes as a simulator to multi-level computer experiments to model the likelihood surface, based on evaluations of the likelihood or likelihood approximations at a small number of parameter values. We work with the SLR model for the single-level GP approximation and with the GLMM and Ising model examples for the multi-level GP approximation.

3.5.2 Single-level case

Simple linear regression model example

Recall the simple linear regression model in Section 3.4.2. The normal distribution of the likelihood of the model has two parameters, the mean μ and the standard deviation σ_r. For simplicity we assume that the mean is constant and we are interested in the

likelihood as a function of σ_r. The likelihood function for the simple linear regression model can be computed analytically as shown in (3.29).

We use this example since the closed form of the likelihood is available in analytic form and we want to demonstrate how the methodology works and compare the GP approximation with the true likelihood. We assume that the likelihood could not be calculated analytically, which is the case for models with intractable likelihood. Therefore, we use Gaussian process regression to compute the Gaussian process posterior. This could be used as an approximation to the likelihood.

Figure 3.8 shows the posterior mean as approximation to the likelihood using 7 training points of the parameter of interest σ_r in the interval $(2, 10)$ for the single-level case. As can be seen, the posterior mean of the GP, given by the red curve, is close to the true value of the likelihood, given by the dashed orange curve, for the majority of values in the given interval. We can achieve a better approximation by increasing the training points in the whole interval (green dots) or by focusing on the neighbourhood of the maximum, since this is the area of the design we are most interested in. A method for choosing the experimental design is presented in Chapter 5. The 95% interval estimation is shown by the blue shaded area. The hyperparameters of the Gaussian process were estimated as described in Section 3.2.8 and we have used a quadratic prior mean.

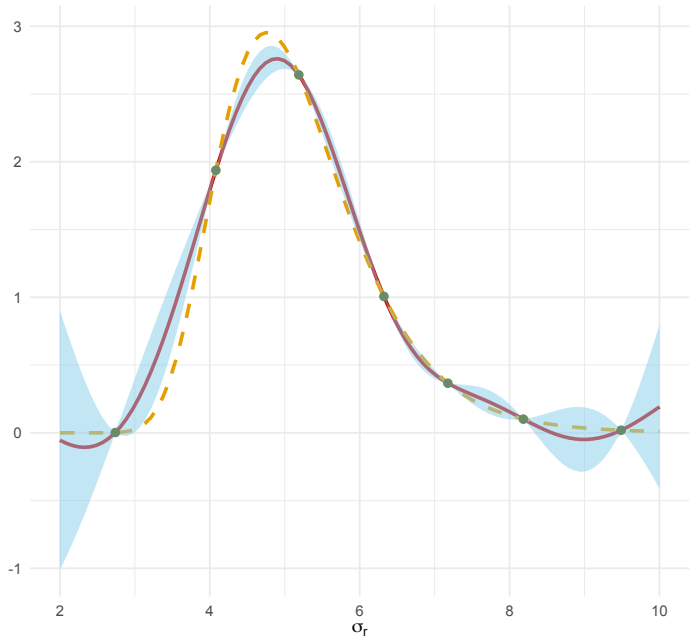


FIGURE 3.8: Posterior mean of the Gaussian process as an approximation to the likelihood for the simple linear regression model given by the red curve. The true likelihood is given by the orange dashed curve, 95% credible intervals by the blue shaded area and the training points are in green.

Looking at the plot in Figure 3.8, there is a problem arising from the use of likelihood evaluations for the GP approximation. The posterior mean is negative for some values of σ_r. To solve this problem we choose to use log-likelihood evaluations for the GP approximation as shown in Figure 3.9 and then for computing the emulated likelihood we can take the exponential of the posterior mean. The orange curve in Figure 3.9 is the exact log-likelihood and the red curve the posterior mean of the GP approximation using log-likelihood evaluations. As in the likelihood approach, we have used a quadratic prior mean structure and the same training points. There is a sudden drop in the shape of the log-likelihood at the left boundary and the GP finds it difficult to estimate the log-likelihood function in that area because it generally assumes that the rates of change are similar for the whole function. As a solution to this issue, as part of the future work we could look to re-parameterise the parameter of interest to get a more symmetric function.

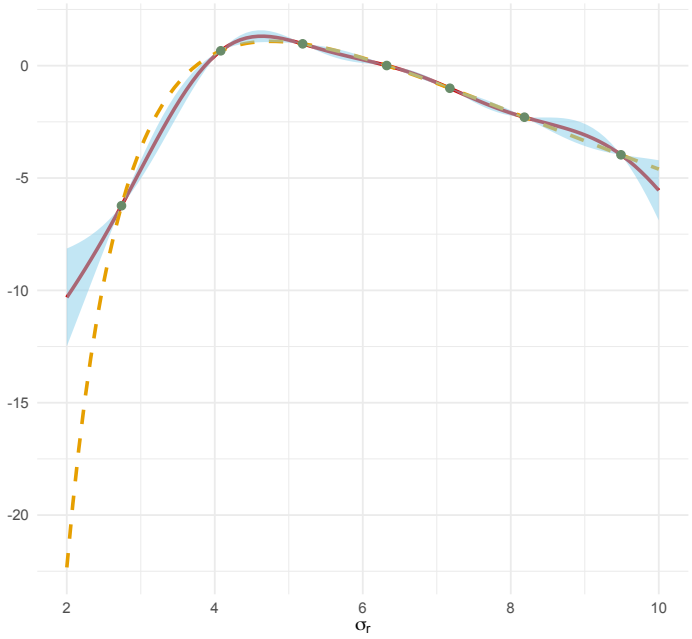


FIGURE 3.9: Posterior mean of the Gaussian process as an approximation to the log-likelihood for the simple linear regression model given by the red curve. The true likelihood is given by the orange dashed curve, 95% credible intervals by the blue shaded area and the training points are in green.

Figure 3.10 shows a comparison between the posterior mean resulted from the GP approximation using evaluations of the likelihood (green curve), the exponential of the posterior mean using evaluations of the log-likelihood (blue curve) and the likelihood obtained using the analytic form (orange). In this example, the log-likelihood approach overestimates the likelihood close to its maximum, but is much more accurate than directly emulating the likelihood in the tails. As can be seen, the likelihood approximation obtained by the exponential of the posterior mean using the log-likelihood

approach is always non-negative which does not hold when using the likelihood approach. We need to be more careful with the choice of approximation approach especially when we would like to approximate the whole surface or if the focus is to approximate the associated uncertainty and not only the area of the maximum. For the rest of the thesis, we prefer to use the log-likelihood approach for GP approximations because the emulated likelihood resulting from that approach is always non-negative.

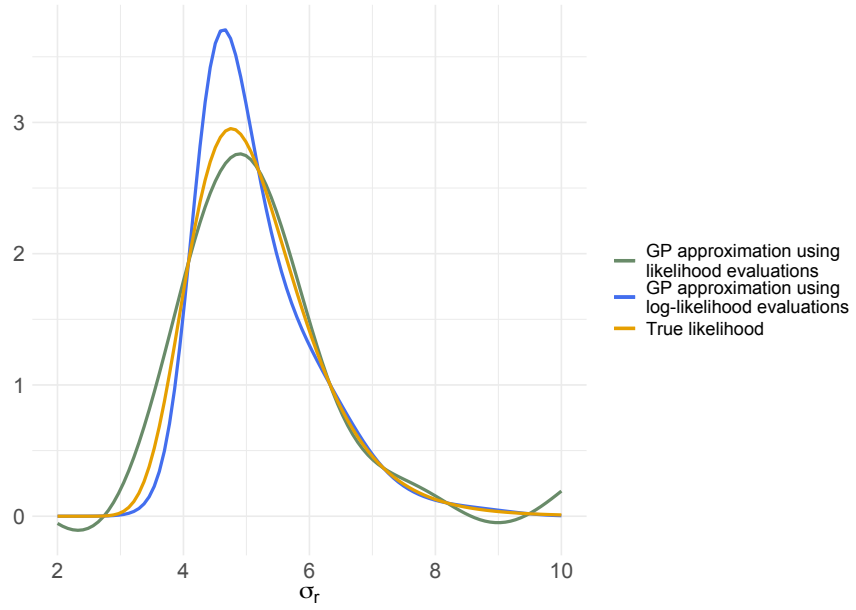


FIGURE 3.10: Comparison of emulated likelihoods using two approaches for GP approximation along with the true likelihood.

3.5.3 Two-level likelihood approximation

Overview

Suppose that we have a simulator with two levels of approximation: the low-level $f_1(.)$, with n_1 training points, and the high-level $f_2(.)$ with n_2 training points. The main idea is to have evaluations from both levels of approximation simultaneously to obtain information from the evaluations of $f_1(.)$ and $f_2(.)$ to predict the output of $f_2(.)$ using the posterior distribution given in Section 3.3.3. The hyperparameters of the GP are estimated using the calculations presented in Section 3.2.8.

We also want to examine the behavior of the posterior distribution as the number of training points for each level varies. We aim to have a good approximation for the output of the expensive model with as few evaluations of the expensive model as possible. As examples of a model with intractable or computationally expensive likelihood functions we will use the generalised linear mixed model introduced in Section 2.3 and the

Ising model introduced in Section 2.4. The training points used for the examples of this chapter have been chosen randomly using a space-filling design. A method for choosing the training points used for each level taking into account the cost of each level is given in Chapter 5.

Generalised linear mixed model example

Recall the generalised linear mixed model from Section 2.3. Considering the two hierarchical levels of the simulator we will use the Laplace approximation method and the adaptive Gaussian quadrature approximation method, with 10 quadrature points, of the likelihood as the lower and the higher level respectively.

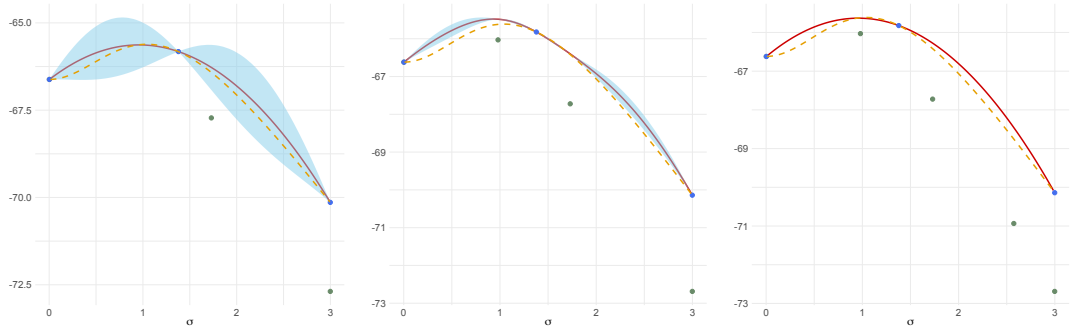
The parameter of interest is the standard deviation of the random effects in (2.7). As before, we fixed the remaining parameters of the model (2.7), β_0 and β_1 and we set them equal to their estimated value after fitting the GLMM.

Figures 3.11 and 3.12 show how the posterior mean of the Gaussian process varies for different number of training points of each of the two levels. In Figure 3.11 we keep the number of training points of the high-level constant and we increase the number of points of the low-level. Similarly, in Figure 3.12 we increase the high-level training points keeping the low-level points constant. Each time we add one additional point to the design and we keep the remaining the same. In both Figures the red curve is the posterior mean, the blue shaded area represents the 95% credible intervals and the blue and green dots are the training points of the high and low-levels respectively. The orange dashed line is the accurate approximation of the log-likelihood. The prior mean structure used for this example was quadratic since a log-likelihood surface is close to a quadratic surface for large samples.

We investigate how the posterior mean is affected as the number of training points of the two levels of approximation varies. Figure 3.11 shows that we manage to obtain a good approximation of the log-likelihood, as the number of training points of the cheap approximation increases, since the more accurate log-likelihood approximated by the AGQ method at the test points is close to the posterior mean of the Gaussian process given by the red curve. However, the GP is overconfident with the increase of training points in Figure 3.11c resulting to overestimation of the log-likelihood and minimal uncertainty.

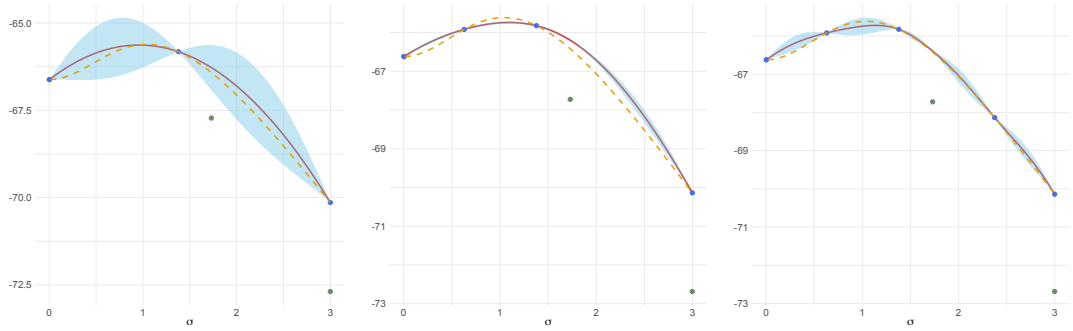
We observe a similar situation as we increase the number of training points of the higher level of approximation as can be seen in Figure 3.12 where the approximation is overestimating the log-likelihood for some values of σ.

Nevertheless, in most situations evaluations of the expensive approximation are not available in great quantities. Therefore, using multi-level GP approximations we could



(A) $D_1 = 3, D_2 = 3$ (B) $D_1 = 4, D_2 = 3$ (C) $D_1 = 5, D_2 = 3$

FIGURE 3.11: Posterior mean of the Gaussian process as an approximation to the log-likelihood for the GLMM (red curve) for various combinations of training points from the two-level GP approximation, with increasing number of low-level training points (green dots) keeping the high-level constant (blue dots), along with the accurate AGQ approximation of the log-likelihood (orange dashed curve).



(A) $D_1 = 3, D_2 = 3$ (B) $D_1 = 3, D_2 = 4$ (C) $D_1 = 3, D_2 = 5$

FIGURE 3.12: Posterior mean of the Gaussian process as an approximation to the log-likelihood for the GLMM (red curve) for various combinations of training points from the two-level GP approximation, with increasing number of high-level training points (blue dots) keeping the low-level constant (green dots), along with the accurate AGQ approximation of the log-likelihood (orange dashed curve).

use more evaluations of the cheap approximation and less of the expensive approximation and still get a good approximation for the log-likelihood.

Ising model example

Recall the special case of the Ising model which was introduced in Section 2.4. Considering the two hierarchical levels of the simulator we will use the reduced-dependence approximation method for approximating the normalising constant (2.11).

We use the simulated Ising model example described in Section 3.4.4 with a 10×10 grid. Figure 3.13 shows the log-likelihood of the Ising model using three different levels of approximation for the normalising constant. The approximations were obtained

using $k = 3, 4$ and 6 in the reduced-dependence approximation method for estimating the normalising constant of the likelihood of Ising models. Increasing k increases the accuracy but also increases the computational cost. As can be seen, the low-level approximation does not give an adequate approximation to the log-likelihood. As we increase the level of complexity the accurate approximation is relatively close to the high-level approximation. However, we would like to avoid using multiple evaluations of the high-level since it is more computationally expensive compared to lower level approximations.

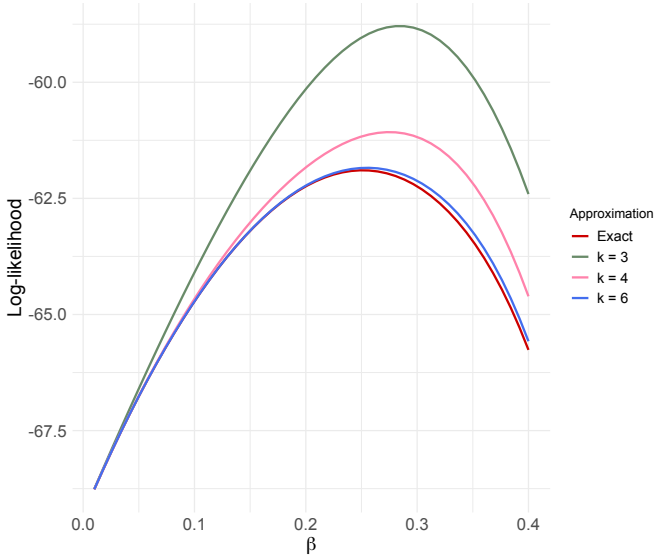
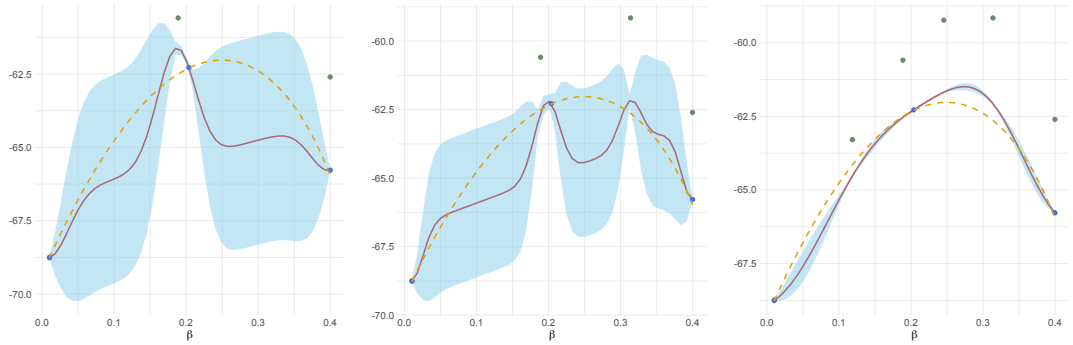


FIGURE 3.13: Log-likelihood using three levels of approximation of the RDA method for the normalising constant of the Ising model for a simulated example with a 10×10 grid along with the exact log-likelihood.

As in the GLMM case, we demonstrate the methodology for the log-likelihood approximation using two-level Gaussian processes approximation. For the example, the tuning parameters in the RDA method for the cheap approximation of the normalising constant is $k = 3$ and the expensive approximation is $k = 6$.

Figure 3.14, shows the approximation to the log-likelihood for the Ising model where we keep the number of high-level training points constant and we increase the number of the low-level points. In Figure 3.14a the posterior mean (red curve) is not close to the exact log-likelihood (orange curve) probably due to the choice of the prior mean as explained below. With the increase of training points of the low-level the approximation improves.

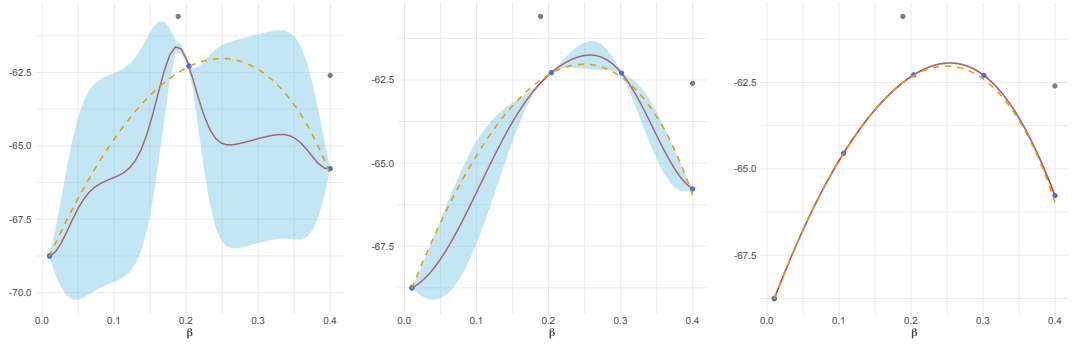
Figure 3.15, shows the approximation to the log-likelihood (red curve) for the Ising model where we keep the number of low-level training points constant and we increase the number of the high-level points. With the increase of the number of high-level



(A) $D_1 = 3, D_2 = 3$ (B) $D_1 = 4, D_2 = 3$ (C) $D_1 = 6, D_2 = 3$

FIGURE 3.14: Posterior mean of the Gaussian process as an approximation to the log-likelihood (red curve) for the Ising model for various combinations of training points of the two-level GP approximation, increasing number of the low-level training points (green dots) keeping the number of a high-level training points (blue dots) constant along with the exact log-likelihood (orange dashed curve).

points the two-level GP approximation is closer to the exact log-likelihood, however for larger number of points it becomes overconfident with minimal uncertainty.



(A) $D_1 = 3, D_2 = 3$ (B) $D_1 = 3, D_2 = 4$ (C) $D_1 = 3, D_2 = 5$

FIGURE 3.15: Posterior mean of the Gaussian process as an approximation to the log-likelihood for the Ising model for various combinations of training points of the two-level GP approximation, with keeping the number of a low-level training points (green dots) constant and increasing the number of a high-level training points (blue dots) along with the exact log-likelihood (orange dashed curve).

The prior mean structure used for the example was linear. A quadratic prior mean structure could be used for this example as well. However, the GP approximation for the initial design was really close to the accurate approximation of the log-likelihood with minimal uncertainty overestimating the log-likelihood. Therefore, we work with a linear prior to demonstrate how the additional points at each level can affect the GP approximation even for a case where the choice of the prior mean structure does not result to the best initial fit.

It will be interesting to examine how introducing more than two levels will affect and probably benefit the approximation.

3.5.4 Three-level likelihood approximation

Overview

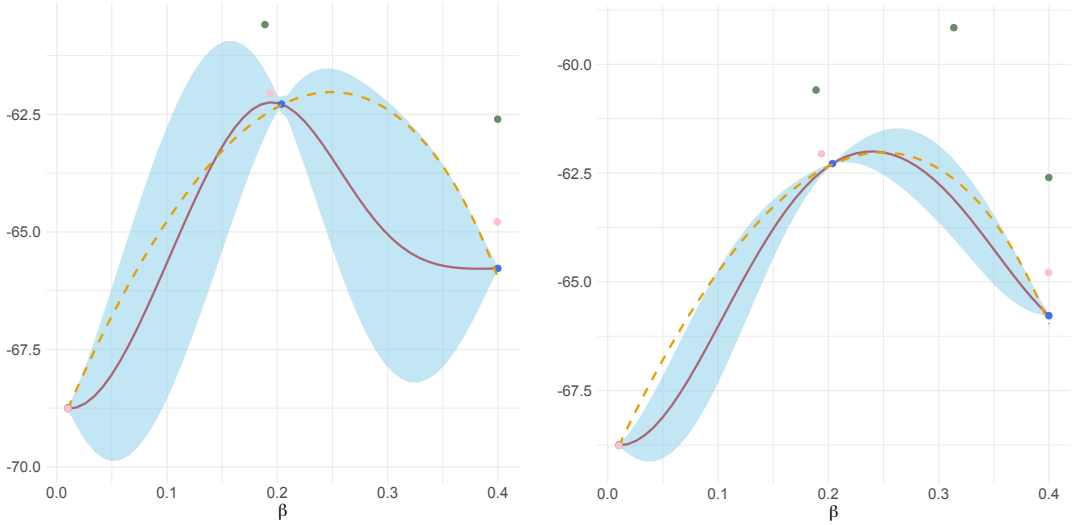
For this section we suppose that multiple levels of approximation are available. Approximating the likelihood under this scenario can be easily generalised. For the purposes of this thesis, we only present examples up to three levels each with different cost and accuracy. Our aim is to see how the log-likelihood approximation is affected by using multiple-levels and if high-level evaluations can be reduced.

Assume that we have a simulator with three levels: the low-level $f_1(.)$, with n_1 training points and design D_1, the middle-level $f_2(.)$, with n_2 training points and design D_2, and finally the high-level $f_3(.)$ with n_3 training points and design D_3. The main idea is to have evaluations from all levels of model simultaneously trying to obtain information from the output of $f_1(.)$, $f_2(.)$ and $f_3(.)$ to predict the output of $f_3(.)$ using the posterior distribution given in Section 3.3.4 for $s = 3$ combining information from all three levels. The posterior mean can be then used as the approximation of the complex approximation of the likelihood similarly to the two-level case. The hyperparameters used are estimated using the calculations presented in Section 3.2.8.

Ising model example

As a continuation from the two-level example on the Ising model we introduce a third level of approximation which lies in between the previous two levels based on accuracy and cost and we want to examine how the additional level impacts the approximation.

Following the examples in Figures 3.14a and 3.14b where we have two levels of approximating the normalising constant where the tuning parameter of the RDA method is $k = 3$ and $k = 6$, we add another level with $k = 4$. The RDA approximation of the log-likelihood for each level is shown in Figure 3.13. We keep the training points of the two already existing levels the same. The results are shown in Figure 3.16. With the additional level of approximation the three-level GP approximation given by the posterior mean (red curve) significantly improves in Figures 3.16a and 3.16b compared to Figures 3.14a and 3.14b respectively and it is closer to the exact log-likelihood (orange curve). Therefore, using multiple levels for the GP approximation is substantially beneficial.



(A) $D_1 = 3, D_2 = 3, D_3 = 3$ (B) $D_1 = 4, D_2 = 3, D_3 = 3$

FIGURE 3.16: Posterior mean of the Gaussian process as an approximation to the log-likelihood for the Ising model for various combinations of training points for the three-level GP approximation: low-level (green dots), middle-level (pink dots) and high-level (blue dots), along with the exact log-likelihood (orange dashed curve).

3.6 Increase of the dimensions

3.6.1 Overview of Gaussian process regression for multi-level simulators

Up to this stage, the log-likelihood was used only as a function of one parameter and the remaining were considered fixed. In this section, we increase the dimension of the parameter space and treat the likelihood as a function of the other model parameters as well. Therefore, we will be able to conduct inference for all the model parameters.

The Gaussian process regression presented in Section 3.2.5 focused on the case where the dimension of the parameters space, d, is one. The aim of this section is to extend Gaussian process regression to compute the posterior predictive distribution for test points in the case where the dimension is $d = 2, 3, \ldots$.

In general, there are no major changes and the formulas for computing the posterior mean (3.11) and the posterior covariance (3.12) remain the same. The posterior mean will now be a function from $\mathbb{R}^d$ to $\mathbb{R}$ and posterior covariance a function from $\mathbb{R}^d \times \mathbb{R}^d$ to $\mathbb{R}$. Considering the multi-level GP approximation and the hyperparameter estimation of the GP the calculations stay the same and we can follow the same methodology as for the one-dimensional cases.

3.6.2 Covariance function

The covariance function needs to be adapted for multi-dimensions. The covariance function can be non-isotropic by setting $d^2(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} - \boldsymbol{x}')^{\mathrm{T}} \boldsymbol{M} (\boldsymbol{x} - \boldsymbol{x}')$ for some positive semi-definite $\boldsymbol{M}$ where $d = \boldsymbol{x} - \boldsymbol{x}'$ is the separation vector. Various forms of $\boldsymbol{M}$'s have been considered in the literature. In general, we can assume that M has the form

$$\boldsymbol{M} = \boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\mathrm{T}} + \boldsymbol{\Psi}, \tag{3.34}$$

where $\boldsymbol{\Lambda}$ is a $D \times k$ matrix whose columns define k directions of high relevance, and $\boldsymbol{\Psi}$ is a diagonal matrix with positive entries, that captures the axis-aligned relevances. Therefore, $\boldsymbol{M}$ has a factor analysis form. To choose k we have to consider a trade-off between flexibility and required number of parameters (Rasmussen and Williams, 2005). For the special case where $\boldsymbol{M}$ is diagonal we can use different length-scales on different dimensions.

The squared exponential covariance function can be used for more than one dimensions and is given by

$$K(\boldsymbol{x}, \boldsymbol{x}') = \tau^2 \exp\left(-\frac{1}{2l^2} \sum_{j=1}^{d} (x_{i,j} - x_{k,j})^2 \right). \tag{3.35}$$

The length-scale hyperparameter could be different for each input dimension given by l_j. To model functions with more than one input we could multiply kernels defined on each individual input. We can use the product of squared exponential kernels for different dimensions each with different length-scale parameter to get the squared exponential automatic relevance determination (SE-ARD) kernel given by

$$K(\boldsymbol{x}, \boldsymbol{x}') = \tau^2 \exp\left(-\sum_{j=1}^{d} \frac{(x_{i,j} - x_{k,j})^2}{2l_j^2} \right), \tag{3.36}$$

where the term $-\frac{1}{2l_j^2}$ will move inside the summation and each of the length-scale parameters determines the input relevance (Duvenaud, 2014).

For this stage of the thesis, we use a single length-scale parameter for all of the dimensions. As part of the future work we would like to introduce and estimate different length-scale hyperparameter for each dimension. However, this can be more complicated with the increase of the levels of approximation in the multi-level GP regression since we will have to estimate multiple length-scale parameters for each level of approximation used.

3.6.3 Training points

The training points for each dimension are generated using a space-filling design. We work with multiple parameters and we would like to be able to have comparable scales for easier interpretation of the results. Therefore, we standardise the training points such that all of the dimensions are on the same scale. We will use normalisation which re-scales the values into a range of $(0, 1)$ by subtracting the minimum value and dividing with the difference between the maximum and minimum value.

Choosing the experimental design of each model parameter could be challenging. For the example used in the thesis, we choose an interval of the model parameters which contains the location of the maximum of the log-likelihood. However, that will not be always applicable and easy to compute. In the future, we would like to automate this procedure by expanding the expected gain in utility method for choosing the experimental design presented in Chapter 5 to include multi-dimensional problems and expand the area of experimentation by probably using a response surface methodology by finding the optimal response (Box and Wilson, 1951).

3.6.4 Applications

As an illustration of extending the multi-level Gaussian process to multi-dimensions we present the GLMM example using the two-level GP approximation for $d = 2$. The log-likelihood becomes a function of the standard deviation of the random effects, σ and the intercept parameter β_0 from (3.30). We consider the third model parameter β_1 fixed. Looking at the figures of this section, the x-axis represents the standard deviation of the random effects and the y-axis the intercept parameter. We use a two-dimensional parameter space example due to demonstration limitations of the results.

For this example, we choose a space-filling design obtained using the locally optimal maximum projection (MaxPro) design from the `MaxPro` package in `R` (Ba and Joseph, 2015) in the neighborhood of a given initial design for continuous factors, where the initial design was a Latin hypercube design. We then re-scaled the design points so that they lie in the range $(0, 1)$. For each parameter we use 20 points for the low-level and 10 points for the high-level of approximation.

We have used the same methodology as in the one-dimensional case for estimating the hyperparameters and computing the GP posterior predictive distribution. The low-level of approximation is the LA method and the high-level is the AGQ method. The surface of the LA method at the test points is given in Figure 3.17. For the calculations we used a linear prior mean structure.

Figure 3.18a shows the surface of the of the Gaussian process posterior mean as an approximation to the log-likelihood and Figure 3.18b shows the accurate approximation

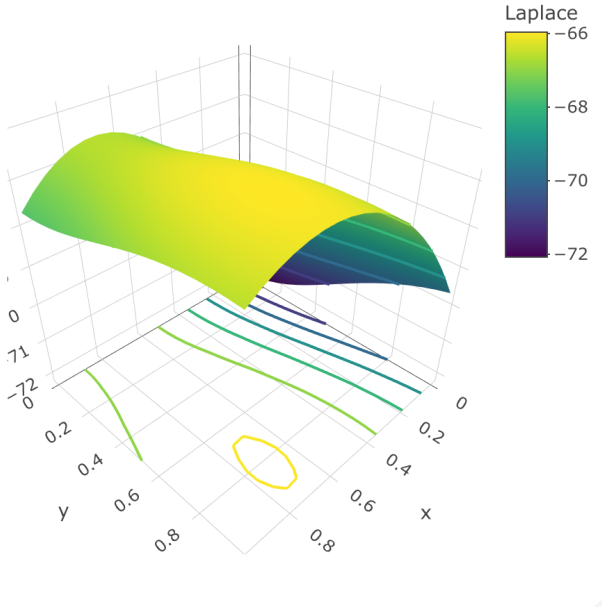
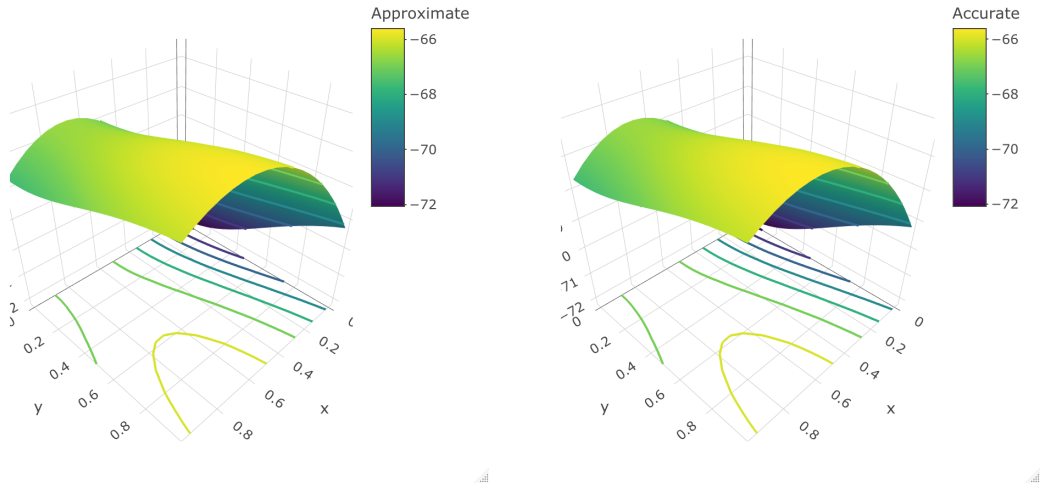


FIGURE 3.17: Surface plot of the Laplace approximation at the test points as the low-level approximation of the log-likelihood of the GLMM for a two dimensional parameter space.

of the log-likelihood. As can be seen, both surfaces look similar. The similarity of the surfaces can also be assessed with computing their difference.



(A) Approximate log-likelihood (B) Accurate log-likelihood

FIGURE 3.18: Surface plots of the posterior mean of the two-level GP approximation and the accurate approximation of the log-likelihood of the GLMM for a two dimensional parameter space.

We compute the difference between the approximated log-likelihood and the accurate log-likelihood of the GLMM and the result is given in Figure 3.19. As shown in the legend of the figure, the maximum difference varies from 0 to just below 0.02 which indicates that the two surfaces are similar. This can also be seen from the contour plot

of the differences given in Figure 3.20. Therefore, the methodology of the multi-level GP approximation can be used in a multi-dimensional parameter space.

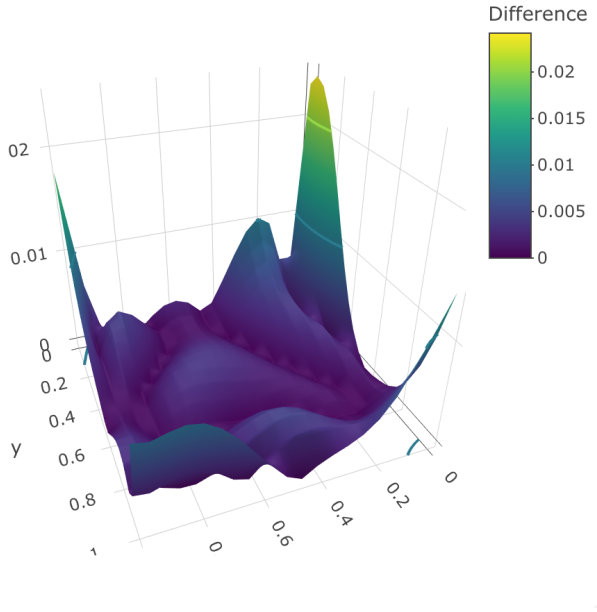


FIGURE 3.19: Difference between the accurate approximation and the two-level GP approximation of the log-likelihood of the GLMM for a two dimensional parameter space.

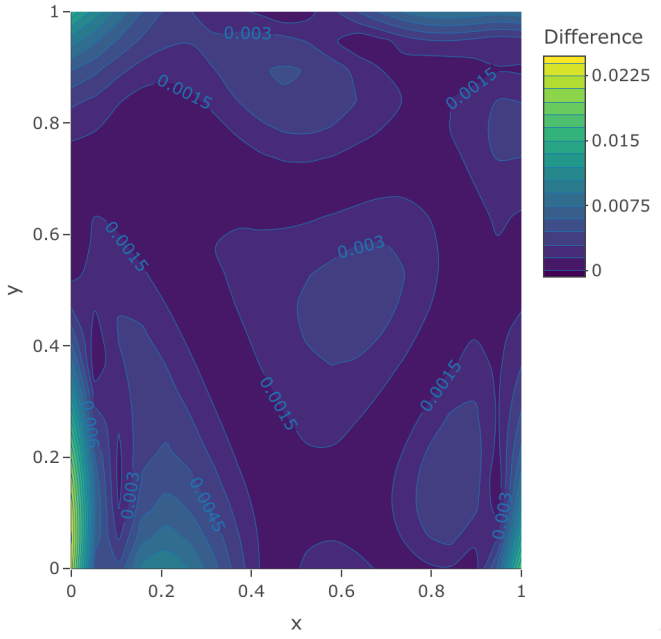


FIGURE 3.20: Contour plots of the difference between the accurate approximation and the two-level GP approximation of the log-likelihood of the GLMM for a two dimensional parameter space.

3.7 Summary

This chapter introduced two of the main components of the thesis: Gaussian processes and hierarchical experiments and explained how these two can be combined to obtain an approximation to the likelihood by using evaluations from various levels of accuracy. We introduced the example models and the data that are used throughout this thesis as an illustration of the ideas and methodology. Moreover, the posterior distribution of a Gaussian process was defined using results form the multivariate normal distribution and its conditional distribution and we discussed a method for estimating the hyperparameters of the Gaussian process.

We focused on the prediction of the output of complex approximation when fast approximations or lower level approximations are available. The Bayesian approach of the approximation of complex approximation using lower level approximations, and especially the use of Gaussian processes, can be implemented for a broad field of multi-level simulators. In a situation where the most accurate approximation is complex and expensive to run, being able to use information obtained from evaluations of the simulator at different levels is exceptionally advantageous. If only a few evaluations of the expensive approximation are available, the approximation of the expensive approximation can been considerably improved using observations from cheaper approximations as well.

We applied the methodology to approximate intractable likelihood functions using multiple levels of likelihood approximations. We used a simple linear regression model, where we know the closed form of the likelihood, to illustrate and validate how we can approximate the likelihood using Gaussian process regression for a single-level case. Moreover, we demonstrated how we can approximate a likelihood function using observations from two or three scales of approximation. Generalised linear mixed model and Ising model were used as examples of models with intractable or computationally expensive likelihood functions. Using the GLMM example we demonstrated the two-level case with the Laplace approximation method as the lower level of the approximation and the AGQ approximation method as the higher level. Considering the Ising model example, the RDA method was used for multi-levels for approximating the normalising constant and therefore the likelihood.

Furthermore, we extended the GP regression for a multi-dimensional model parameter space. We presented how the covariance function can be adapted to count for more than one dimensions and some assumptions for the training points. We applied the methodology for a two-dimensional model parameter space for the GLMM example. We compared the accurate log-likelihood with the approximated obtained using the two-level GP approximation and we found that there is not a big difference between the two surfaces.

In general, it was shown that the posterior mean of the Gaussian process posterior distribution can be used as a likelihood approximation and analyses at multiple levels of a simulator, which are related through an autoregressive model, can improve the accuracy of the highest level of analysis. From the example models, we can see that the posterior mean function for multi-level simulator, using few of the expensive evaluations, has been considerably improved when we are combining observations from multiple levels.

Chapter 4

Inference for the model parameters and uncertainty

4.1 Overview

In the previous chapters, we have demonstrated how to use a use a Gaussian process to obtain an approximation of the likelihood surface. As shown, the resulting posterior mean of the GP can be used as an approximation to the likelihood. We aim to use the approximation to the likelihood to conduct inference for the model parameters. That inference could also include finding uncertainty intervals for the parameters. However, those uncertainty intervals might not be very accurate, because they don't include the uncertainty about the likelihood approximation itself. Hence, there is a concern that we will underestimate the uncertainty by ignoring the uncertainty in the likelihood approximation.

The main aim of this chapter is to conduct inference for the model parameters and to combine the underlying uncertainty about the parameters given the likelihood with the uncertainty about the likelihood itself. For simplification we name the posterior given approximate likelihood evaluations, the posterior GALE.

To deal with the uncertainty problem we will use a Bayesian framework to enable us to combine those different types of uncertainty. We will use a sampling-based method to approximate the posterior GALE. We do this in two steps, first we sample from the multi-level GP approximation for the likelihood, we multiply with the prior of the model parameter of interest to compute the posterior and then we sample from the posterior of the model parameter resulting from that sampled likelihood. Following that procedure we can obtain an approximated posterior distribution of the model parameter.

To approach this problem, sampling methods such as MC could be used but that could be used but it is unnecessarily complicated for the examples we present. In the literature, there exist methods that use single-level GPs with more complex sampling algorithms such as Rasmussen (2003), Gramacy and Lee (2010) and Overstall and Woods (2013).

We compare the resulting approximated posterior distribution of the model parameter with the true posterior, if this is available, or with a more accurate approximation of the posterior, which in most cases will be more costly to compute. The methodology will be applied to two examples using multi-level GP approximation for the log-likelihood.

4.2 Sampling from the Gaussian process posterior distribution

Obtaining samples from the Gaussian process posterior distribution will enable us to base inferences on posterior summaries of the parameters calculated from the samples. To sample from the Gaussian process posterior distribution we draw random samples from the corresponding multivariate Gaussian distribution with the posterior mean and posterior covariance of the Gaussian process posterior at the test points as the mean vector and the covariance matrix respectively.

We would like to describe or at least approximate the approximate log-likelihood function at every point in the given interval, not just at the test points used. Therefore, we use **spline interpolation** to find a function underlying the value between the given points. Spline interpolation is useful for this case to overcome the situation where the value of the sample of the parameter of interest at the maximum of the log-likelihood does not usually match exactly with a value of the test points used. Using the computed samples, we obtain the interpolation function for each sample at the test points.

The samples of the posterior distribution with the corresponding interpolation function could be used as a tool for various aspects such as finding the estimator of the parameter of interest by maximising the approximation to the log-likelihood generated from each draw of the posterior distribution of the Gaussian process. The value of these estimates will give us more information about how certain or uncertain we should be regarding the location of the true maximum likelihood estimate. We obtain samples for each example in Section 4.4

We would like to compute an estimate of the parameter of interest by maximising the approximation to the log-likelihood for each of the samples from the posterior Gaussian process distribution as described in Section 4.2, to get the maximum likelihood estimate. Even when it is not possible to compute the MLE analytically, it can be calculated numerically (Myung, 2003). Hence, optimisation algorithms could be used to find the location of the maximum of the function in the required interval. For each sample

from the Gaussian process we get a different estimate of the parameter of interest with its own uncertainty and therefore a different interval estimation.

The problem we would like to emphasise is how to quantify and combine the uncertainty generated from two sources: the underlying uncertainty about the model parameters given the data with the uncertainty from using the Gaussian process model as an approximation for the likelihood in order to be able to conduct inference for the model parameters. One way of dealing with this problem is using Bayesian inference and computing an approximated posterior distribution of the model parameters.

4.3 Approximated posterior distribution of model parameters

In order to combine the uncertainty obtained from the posterior given approximate likelihood evaluations, the posterior GALE uncertainty, and the underlying uncertainty about the model parameters given the data we compute an approximated posterior distribution for the model parameter of interest. The methodology is described in Algorithm 1.

We start by computing the Gaussian process posterior distribution using Gaussian process regression either for a single-level approximation as described in Section 3.2.5 or for the multi-level approximation case as described in Section 3.3.3 given by $\mathcal{N}(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{V}})$. The posterior mean $\bar{\boldsymbol{\mu}}$ can then be used as an approximation to the likelihood or the log-likelihood and the posterior covariance $\bar{\boldsymbol{V}}$ as a measure of uncertainty. This will remain fixed for the following steps.

We then obtain a sample $l_i(y_1, \ldots, y_n|\boldsymbol{\theta})$ from the Gaussian process emulator of the likelihood or log-likelihood evaluated at fixed test points $\boldsymbol{\theta}_{\text{test}}$ for the model parameter from the corresponding multivariate Gaussian distribution where the mean will be given by the posterior mean of the GP, $\bar{\boldsymbol{\mu}}$, and the covariance by the posterior covariance of the GP, $\bar{\boldsymbol{V}}$. We use a fine grid of test points in the given interval. For future work for a high dimensional parameter space it would become difficult to evaluate the whole likelihood/posterior surface at a dense grid of test points, so in that case we would need an alternative posterior sampling-based method.

Following, we multiply the likelihood or the exponential of the log-likelihood sample with the predefined prior of the parameter of interest at the test points, $\pi(\boldsymbol{\theta}_{\text{test}})$, to compute a sample from the posterior, unnormalised sample, under Bayesian inference

$$f_i(\boldsymbol{\theta}_{\text{test}}|y_1, \ldots, y_n) = l_i(y_1, \ldots, y_n|\boldsymbol{\theta}_{\text{test}}) \times \pi(\boldsymbol{\theta}_{\text{test}}).$$

In general, Bayesian statistics considers the parameters of a model to be random variables in their own right rather than fixed unknowns. In order to implement a Bayesian

approach, prior beliefs about the model parameters $\boldsymbol{\theta}$ are required and are represented by the **prior**, a probability density (or mass) function. The **posterior** density (mass) function, represents the density of the parameters giving a modified belief about $\boldsymbol{\theta}$ in the light of the observed data and provides inference about the unknown parameters combining information from the data through the likelihood with the information from the prior using Bayes' theorem (Gelman et al., 2013).

Posterior inference can be used in a formal way to make predictions using the posterior predictive distribution. The specification of parameter priors is a primary element of Bayesian inference. The prior distribution represents the knowledge about $\boldsymbol{\theta}$ prior to observing the data. The prior of the model parameter for each of the examples of this chapter is tailored to the specific example and the characteristics of the model parameter.

For the next step, we compute the average of the resulting samples from the posterior to calculate the normalising constant

$$Z_i = \sum_{\boldsymbol{\theta} \in \boldsymbol{\theta}_{\text{test}}} f_i(\boldsymbol{\theta}|y_1, \ldots, y_n),$$

and we divide each unnormalised sample with Z_i so that the posterior will sum up to one, that is

$$F_i(\boldsymbol{\theta}_{\text{test}}|y_1, \ldots, y_n) = f_i(\boldsymbol{\theta}_{\text{test}}|y_1, \ldots, y_n)/Z_i,$$

We store each normalised posterior sample, $f_i(\boldsymbol{\theta}_{\text{test}}|y_1, \ldots, y_n)$, and we repeat this procedure for a large enough number of likelihood or log-likelihood samples, n_{samples}.

Finally we compute the average of the samples of the posterior to get the approximated normalised posterior GALE of the model parameter

$$G(\boldsymbol{\theta}|y_1, \ldots, y_n)) = \sum_{i}^{n_s} F_i(\boldsymbol{\theta}|y_1, \ldots, y_n)/n_s.$$

We compare the approximated posterior GALE distribution of the model parameter of interest with the true posterior, if that is available, or with an accurate approximation of the posterior visually through density plots. When interpreting these comparisons we have to take into consideration that we do not expect to have a perfect match. The reason is that we expect extra uncertainty in the posterior sampling from the approximated posterior because we conduct inference based on a few likelihood evaluations, rather than based on the whole likelihood surface. We also anticipate the posterior GALE to get closer to the posterior as the number of likelihood evaluations increases.

Algorithm 1 Approximated posterior GALE distribution of model parameter

1. **Compute** the Gaussian process posterior distribution as an approximation to the likelihood or log-likelihood $\mathcal{N}(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{V}})$;

2. **Draw a sample** from the Gaussian process emulator of the likelihood or log-likelihood, $l_i(y_1, \ldots, y_n|\boldsymbol{\theta}_{\text{test}})$, evaluated at fixed grid of test points, $\boldsymbol{\theta}_{\text{test}}$, for the model parameter using the corresponding multivariate Gaussian distribution;

3. **Multiply** the likelihood or the exponential of the log-likelihood sample from Step 2 with a predefined prior of the model parameter at the test points to compute the posterior and consider this as an unnormalised sample of the posterior given by

$$f_i(\boldsymbol{\theta}_{\text{test}}|y_1, \ldots, y_n) = l_i(y_1, \ldots, y_n|\boldsymbol{\theta}_{\text{test}}) \times \pi(\boldsymbol{\theta}_{\text{test}});$$

4. **Calculate** the normalising constant of each sample of the posterior to compute the normalised sample of the posterior so that the area under the density curve will sum up to 1;

$$F_i(\boldsymbol{\theta}_{\text{test}}|y_1, \ldots, y_n) = f_i(\boldsymbol{\theta}_{\text{test}}|y_1, \ldots, y_n)/Z_i,$$

   where

$$Z_i = \sum_{\boldsymbol{\theta} \in \boldsymbol{\theta}_{\text{test}}} f_i(\boldsymbol{\theta}|y_1, \ldots, y_n)$$

   is the normalising constant;

5. **Store** each sample of the posterior;

6. **Repeat** steps 2 - 5 for a reasonable number, n_s, of likelihood or log-likelihood samples;

7. **Compute** the average of the samples of the posterior to get the approximated normalised posterior GALE

$$G(\boldsymbol{\theta}|y_1, \ldots, y_n)) = \sum_{i}^{n_s} F_i(\boldsymbol{\theta}|y_1, \ldots, y_n)/n_s.$$

The resulting normalised posterior GALE, $G(\boldsymbol{\theta}|y_1, \ldots, y_n)$, will form an approximated posterior distribution for the model parameter of interest.

4.4 Applications

4.4.1 Introduction

For demonstrating the ideas presented in this chapter we work with examples where the approximation was obtained using the multi-level GP approximation. More specifically the GLMM example for a two-level GP approximation and the Ising example

using a three-level GP approximation. The same methods can be applied for any number of levels used for approximation since the only requirement is the resulting GP posterior distribution as the approximation of the likelihood.

The goal of the examples is to demonstrate how we can obtain an approximated posterior distribution of the parameter of interest when the likelihood is intractable following the steps of Algorithm 1. We will also demonstrate how the validity of the resulting approximated posterior distribution of the model parameter can be checked.

For the examples, the likelihood is either intractable or computationally expensive. We work in one dimension and we aim to approximate the posterior distribution of the model parameter of interest. We will investigate how the number of training points used at each level of approximation influence the approximated posterior distribution of the parameter of interest by increasing the number of training points of the low-level as has been done in previous examples.

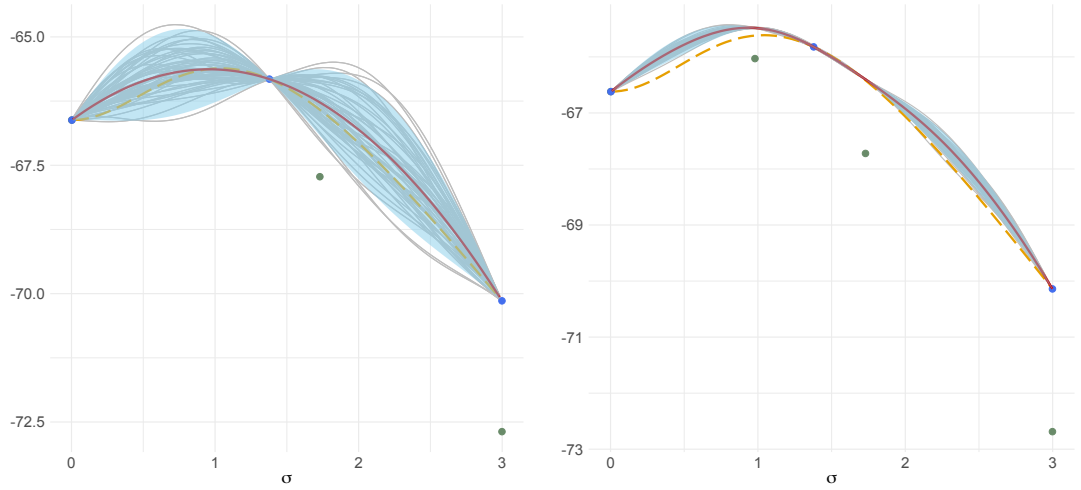
Recall the GLMM example given in Section 3.5.3. For the GLMM example the parameter of interest is the standard deviation of the random effects, and we assume that the remaining model parameters are known and fixed. The examples we refer to are demonstrated in Figure 3.11. For the Ising model we work with the three-level GP approximation given in Section 3.5.4 and the parameter of interest is the scalar parameter β indicating the inverse temperature.

4.4.2 Sampling from the multi-level GP approximation

Generalised linear mixed model

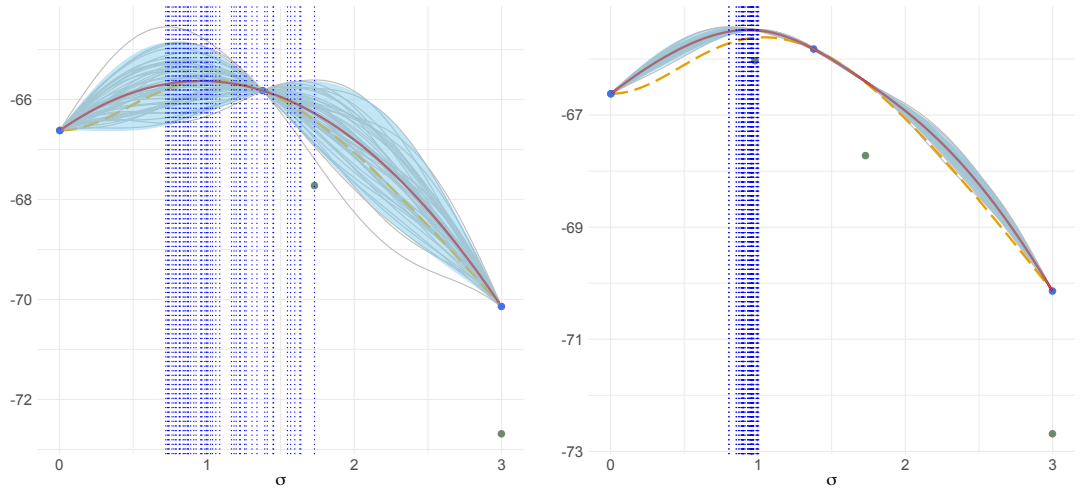
Figure 4.1 demonstrates log-likelihood samples (grey curves) from the Gaussian process posterior distribution computed using the two-level GP approximation for various designs where the number of training points of the low-level increases. As shown, the maximum value of each sample varies. Increasing the low-level training points results to a better approximation (red curve) of the log-likelihood, with lower uncertainty, however overconfident in some areas.

We aim to compute the maximum likelihood estimate through optimisation for each sample. Figure 4.2 demonstrates the random samples of the approximation of the log-likelihood from the Gaussian process posterior given in Figure 4.1 and the vertical blue dashed lines illustrate the approximate location of the maximum for each sample using optimisation. As can be seen, the location of the maximum is different for each sample. With the increase of training points the locations of the maximum for each sample gets closer.



(A) $D_1 = 3, D_2 = 3$ (B) $D_1 = 4, D_2 = 3$

FIGURE 4.1: Log-likelihood samples (grey curves) along with the posterior mean of the two-level GP approximation (red curve), the accurate log-likelihood (orange dashed curve) and the training points of low-level (green dots) and high-level (blue dots) for two designs of the GLMM example.



(A) $D_1 = 3, D_2 = 3$ (B) $D_1 = 4, D_2 = 3$

FIGURE 4.2: Log-likelihood samples (grey curves) along with the posterior mean of the two-level GP approximation (red curve), the accurate log-likelihood (orange dashed curve), the training points of low-level (green dots) and high-level (blue dots) for two designs and the location of the maximum for each sample (vertical blue dashed lines) for two designs of the GLMM example.

Ising model

Similarly with the GLMM example we have in Figure 4.3 log-likelihood samples (grey curves) from the Gaussian process posterior distribution computed using the three-level GP approximation for a design where each level has three points. The posterior

mean (red curve) of the GP approximation is close to the accurate likelihood (orange curve) and most of the samples are within the interval estimation given by the blue shaded area. For instance, for $\beta = 0.105, 92\%$ of the samples are within the interval estimation. However, the maximum value varies for each sample.

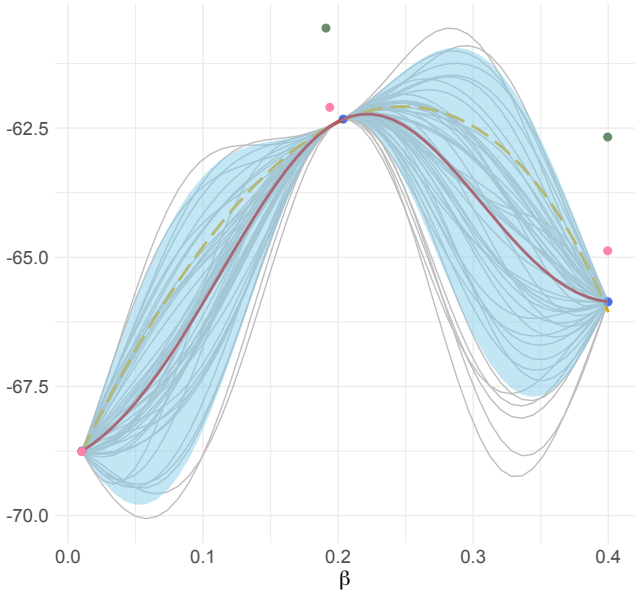


FIGURE 4.3: Log-likelihood samples (grey curves) along with the posterior mean of the three-level GP approximation (red curve), the accurate log-likelihood (orange dashed curve) and the training points of low-level (green dots), middle level (pink dots) and high-level (blue dots) for the Ising model example.

In Figure 4.4 we have the approximate location of the maximum for each sample using optimisation. As can be seen, the location of the maximum is different for each sample. With the increase of the training points we expect the locations of the maximum to concentrate to one area like the GLMM example.

4.4.3 Posterior distribution of the model parameter

To compute the posterior distribution of the model parameter we follow Algorithm 1. For Step 1 we compute the GP posterior distribution. After fitting the GP process posterior distribution, we sample from a corresponding multivariate normal distribution. We obtain samples of the unormalised posterior distribution of the model parameter by multiplying the samples of the approximate likelihood from Gaussian process regression with a prior distribution. We normalise the accurate posterior density numerically so that the area under the curve is one and get the posterior GALE of the model parameter.

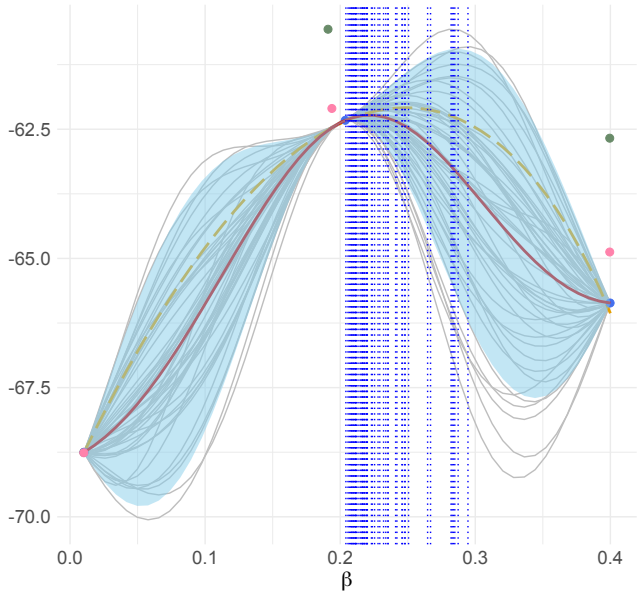


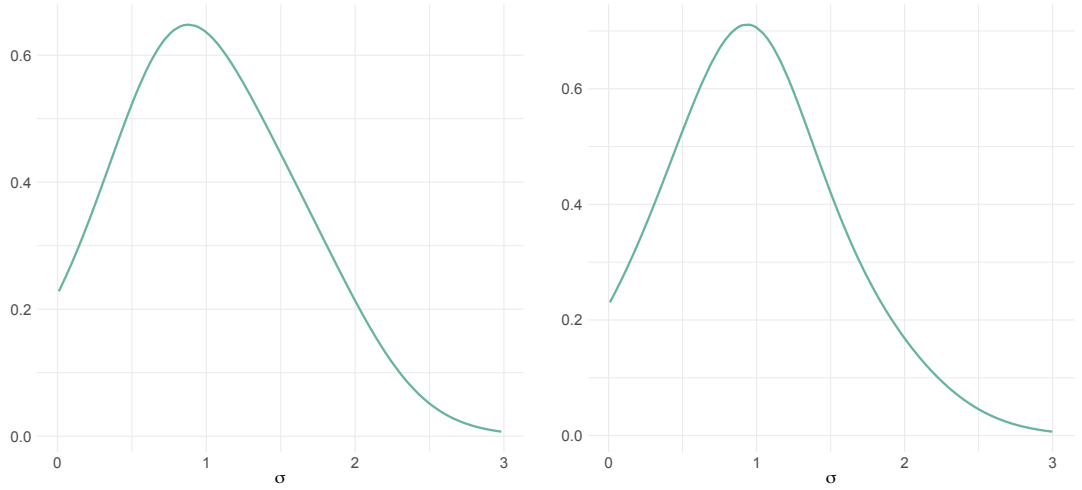
FIGURE 4.4: Log-likelihood samples (grey curves) along with the posterior mean of the three-level GP approximation (red curve), the accurate log-likelihood (orange dashed curve) and the training points of low-level (green dots), middle level (pink dots) and high-level (blue dots) and the location of the maximum for each sample (vertical blue dashed lines) for the Ising model example.

Generalised linear mixed model

Working with the GLMM example, as a prior distribution of the standard deviation of the random effects we use a half-Cauchy prior distribution with scale equal to 5. The half-Cauchy is a convenient weakly informative distribution which has a broad peak at zero and a scale parameter (Gelman et al., 2013). Moreover, the half-Cauchy prior performs well near the origin and does not lead to drastic compromises in other parts of the parameter space (Polson and Scott, 2012). Most importantly, we have chosen a half-Cauchy prior since the variance parameter has to be positive. The posterior GALE of the model parameter for the GLMM example is shown in Figure 4.5 for two designs.

Ising model

For the Ising example, we choose as a prior distribution of the parameter β a Cauchy distribution with location 0.2 and scale 25. The posterior GALE for the GLMM example is shown in Figure 4.6.



(A) $D_1 = 3, D_2 = 3$ (B) $D_1 = 4, D_2 = 3$

FIGURE 4.5: Posterior GALE for the two-level approximation of the GLMM example for two designs.

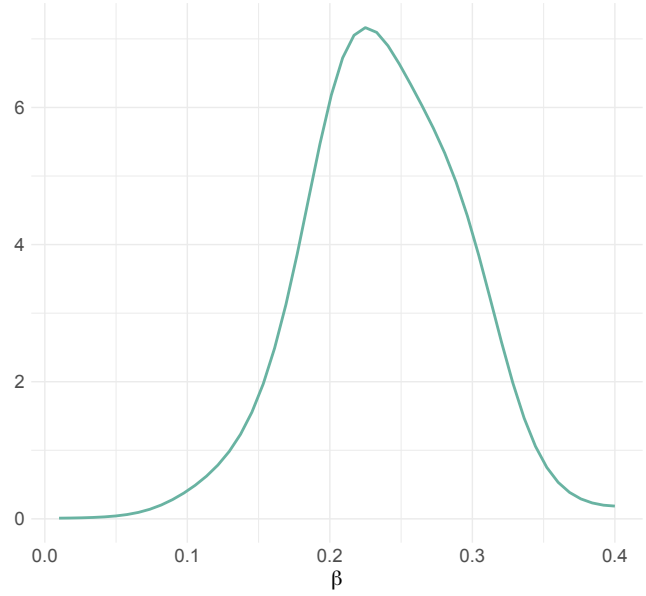


FIGURE 4.6: Posterior GALE for the three-level approximation of the Ising model example.

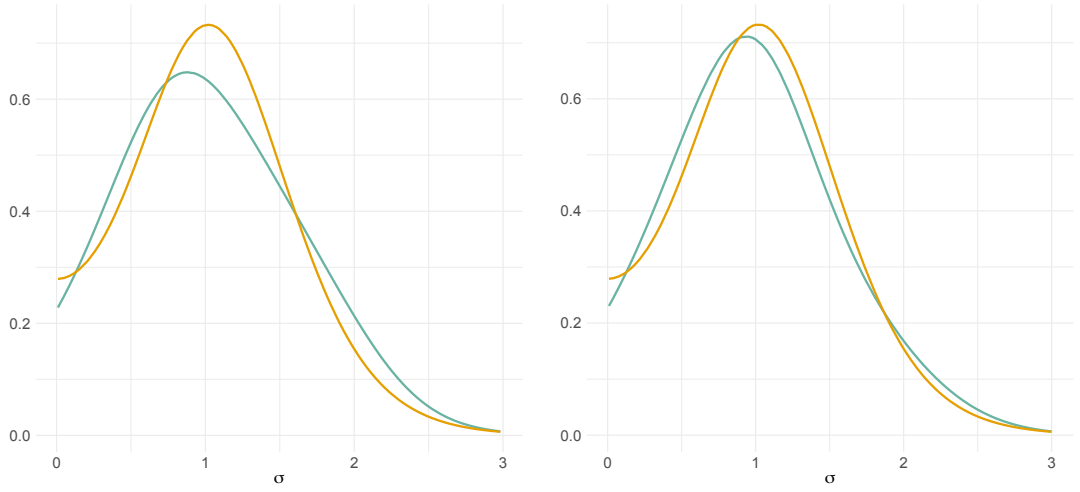
4.4.4 Comparisons

We would like to check if the approximated posterior distribution of the model parameter is valid. For examples where we cannot compute the posterior distribution analytically, we have an accurate approximation of the likelihood and hence we can multiply that with the chosen prior to get an accurate approximation to the posterior. As another form of comparison, we have a plot of three posterior densities of the model

parameter obtained using the posterior GALE of the model parameter, the accurate approximation of the posterior and the posterior calculated using the GP posterior mean as an approximation of the log-likelihood, which we call the single approximated posterior. We apply these to both of the examples.

Generalised linear mixed model

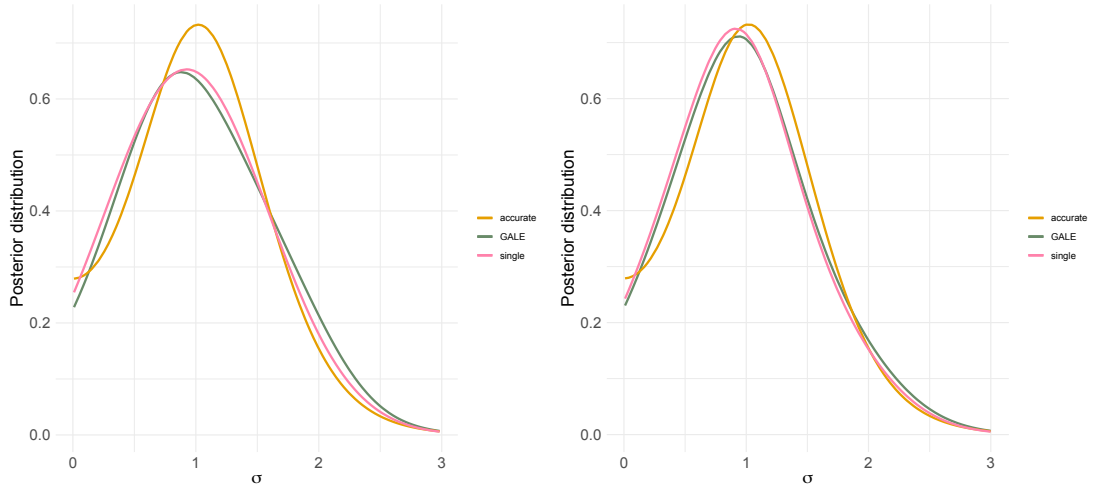
Figure 4.7 shows the posterior GALE (green curve) for each combination of the training points and the accurate approximation of the posterior (orange curve). With the increase of the training points used for each level of approximation used in the GP regression model we obtain a better approximation since the two curves look similar and overlap for some values of the parameter.



(A) $D_1 = 3, D_2 = 3$ (B) $D_1 = 4, D_2 = 3$

FIGURE 4.7: Normalised posterior GALE (green) for the GLMM example for the model parameter with the accurate posterior of the model parameter (orange) for two designs.

Figure 4.8 compares the posterior densities of the model parameter σ obtained using the posterior GALE (green), the accurate approximation of the posterior (orange) and the single approximated posterior (pink). The three curves giving the posterior of the model parameter have a similar shape. The posterior GALE and single approximated posterior distributions are a bit wider than the accurate posterior in Figure 4.8a where there are less training points of the low-level due to the extra uncertainty of the log-likelihood approximation.



(A) $D_1 = 3, D_2 = 3$ (B) $D_1 = 4, D_2 = 3$

FIGURE 4.8: Comparing posterior densities of the model parameter of the GLMM example obtained using three different ways for two designs.

Ising model

Figure 4.9 shows a comparison between the posterior GALE of the model parameter (green curve) and the exact posterior (orange curve) for the Ising model example.

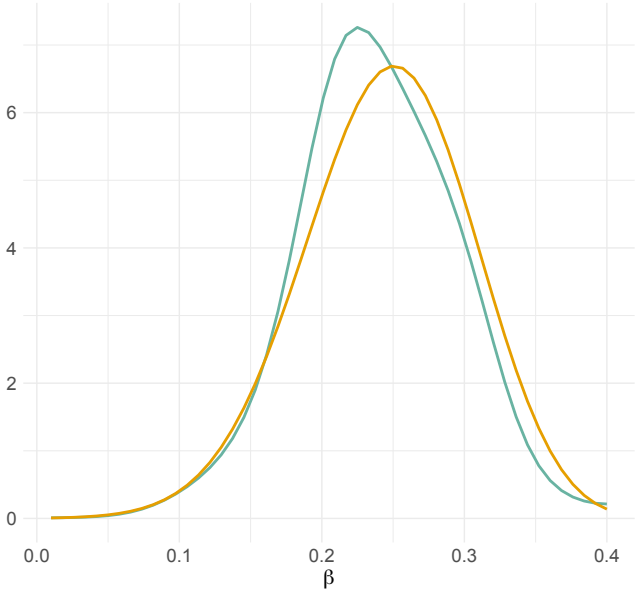


FIGURE 4.9: Normalised posterior GALE (green) for the Ising example for the model parameter with the exact posterior of the model parameter (orange).

Figure 4.10 compares the posterior densities of the model parameter β obtained using

the posterior GALE (green), the accurate/exact posterior (orange) and the single approximated posterior (pink). The posterior GALE is closer to the exact posterior compared to the single approximated posterior indicating the need of sampling from the multi-level GP approximation of the log-likelihood and not relying solely in the single approximated posterior which overestimates the posterior around the maximum.

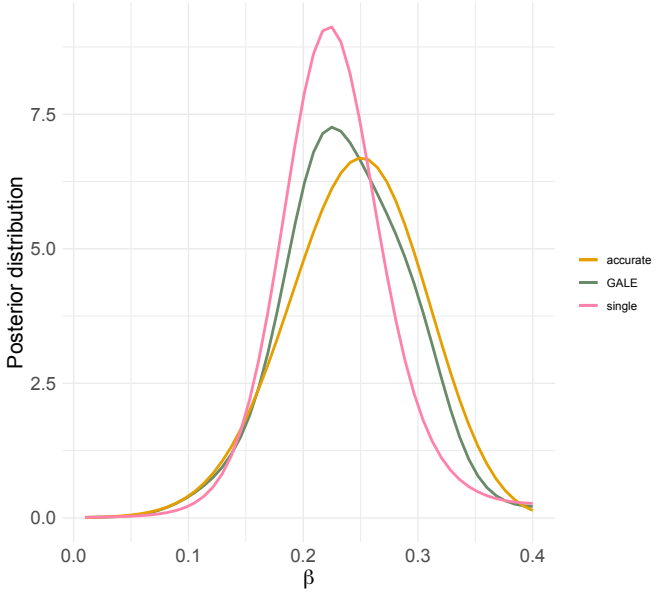


FIGURE 4.10: Comparing posterior densities of the model parameter of the Ising model example obtained using three different ways.

4.5 Summary

In this chapter, the main goal was the approximation of the posterior distribution of model parameter in order to conduct inference of the model parameter and to assessed the uncertainty generated from the Gaussian process regression when approximating the likelihood and the underlying uncertainty from the likelihood itself. We presented how sampling form the GP posterior distribution can be applied to compute an approximated posterior distribution of the model parameters.

We demonstrated how we can approximate the posterior distribution of the model parameters using the methodology presented through the generalised linear mixed model where the likelihood is intractable and the posterior distribution cannot be calculated analytically. We used the Gaussian process regression with multi-level approximation case to compute an approximation of the log-likelihood and then we sampled to obtained the posterior given approximation likelihood evaluations. As shown in this

chapter, with the increase of the low-level training points, for the multi-level approximation case, we managed to compute a good approximated posterior distribution of the model parameter.

Concluding, the approximated posterior distribution will help us to conduct inference for the model parameters and to combine the uncertainty from the likelihood approximations using Gaussian process regression and hierarchical modelling and the underlying uncertainty of the model parameters given the data.

Chapter 5

Experimental Design and Expected Gain in Utility

5.1 Overview

Bayesian optimisation (BO) is a particularly well-suited sequential design method to global optimisation problems where the function we want to optimise is an expensive to evaluate function, f, or it might require running an expensive simulation. It was first proposed by Kushner (1964) and improved by Močkus (1975) and Jones et al. (1998).

Likelihood approximations are one case where general Bayesian optimisation methods for a function f could be used, where f is a likelihood. Considering the hierarchical experiments and the likelihood approximations discussed in previous chapters, we need to decide at how many points to evaluate each level of likelihood approximation based on their cost, and at which values of the parameters, so that we compute an accurate approximation to the maximum likelihood estimate at minimal cost.

We use BO to choose the experimental design of the computer experiment by optimising the likelihood. Given a design and the corresponding approximate function evaluations, we define a utility as the maximum of the emulated high-level function given the data. Given some initial design and corresponding approximate function evaluations we need to choose a new point to add to the design without further evaluations of the function approximations. We have the posterior distribution of each approximation at any candidate point and we use this to compute the expected utility of adding any candidate point to the existing design. We add the point which gives the largest **expected gain in utility (EGU)** relative to the cost of evaluating that level of function approximation. When the value of the EGU falls below a predefined threshold then we do not gain anymore information from adding any new points to the design and so we stop.

5.2 Bayesian optimisation

5.2.1 Introduction

Bayesian optimisation is a class of methods used for optimising objective functions that are computationally expensive to evaluate. In general, Bayesian optimisation creates a surrogate for the objective function and computes the uncertainty in that surrogate using Gaussian process regression. Then it uses an acquisition function to choose where to sample next.

The Bayesian optimisation is outlined in Algorithm 2. More specifically, when we work with Bayesian optimisation, we assume a prior distribution for the unknown or expensive function f, usually a Gaussian process prior. We then start by evaluating the function at an initial design of a small number of randomly selected space-filling design points, obtaining their function values and fitting a Gaussian process (GP) regression model to the results. The aim is to choose a new point to add to the design. We achieve this via an **acquisition function** to select the location of the next observation and update the prior to the posterior for each new point added to the design (Frazier, 2018). The acquisition function calculates the value that would have been generated by evaluating the objective function at a new point based on the current posterior distribution over f.

The posterior probability of f is updated using the prior and all the available function evaluations. The GP posterior estimates the function value for any point we consider in the design space, as well as the uncertainty of the estimation. Therefore, we are able to make inference about the optimum and we repeat this process until some stopping criterion is met.

Algorithm 2 Bayesian optimisation

1. **Assume** a prior for the expensive function f;
2. **Evaluate** f at an initial design;

   Repeat sequentially steps 3-5 until some stopping criterion is met:
3. **Choose** a new design point via an acquisition function;
4. **Update** the posterior probability distribution of f using the prior and all the available data (function evaluations at the existing points and at the new point) using Bayes' theorem;
5. **Conduct** inference about the optimum.

To choose the new design point we need to make sure that there is a balance between exploration and exploitation behaviour. Exploration means to evaluate the f at a point

x in a highly uncertainty region and examine the case where $f(x)$ might be greater that $f(x^*)$ where x^* is the point that maximise f. Exploitation is gained from function evaluations, best estimated function value, meaning to include locations where there might be a high probability that $f(x)$ could be larger than $f(x^*)$. Both exploitation and exploration are required, that means that a candidate point x that is both more exploitative and more exploratory than an alternative point x is to be chosen over x^*.

Acquisition functions guide how to explore the surface during Bayesian optimisation (Wilson et al., 2018). They encode the value of potential points at which to evaluate the function in the optimisation and define the balance between exploration and exploitation. There are a large number of acquisition functions available. Some common choices for the acquisition functions are expected improvement (EI), entropy search and knowledge gradient. We will give more emphasis on the EI acquisition function, which measures the expected value of the improvement at each point over the best observed point.

5.2.2 Expected improvement

The EI acquisition function was first introduced by Močkus (1975) and developed afterwards by Jones et al. (1998). It is used for efficient optimisation of computationally expensive black-box functions. The improvement can be considered as an increase in utility when a new point is been added to the design.

Consider the case where there is no noise in the observations and let

$$y_i = f(x_i).$$

Let $D = (x_1, \ldots, x_n)$ be a design, and define the utility of that design to be

$$u(D) = \max_{x \in D} f(x). \tag{5.1}$$

This utility corresponds to a particular method for choosing which point x to report as the **best** point, once we have chosen a design and fitted a Gaussian process. With this utility, we say that we would choose the point in our training set which has the largest value of $f(.)$. Once the utility is fixed, we can define an improvement for a candidate point $\tilde{x}$ as the difference in the utility between a new design $\tilde{D} = (x_1, \ldots x_n, \tilde{x})$ and the current design D. The improvement $I(.)$ is given by

$$I(\tilde{x}) = u(\tilde{D}) - u(D). \tag{5.2}$$

We want to calculate how much the utility can be expected to improve over the current optimum for every possible input.

By the time we consider adding the new candidate point, we know the values of the function at the design points $D, f(D) = (f(x_1), \ldots, f(x_n))$. Let $f^*(D) = \max_{x \in D} f(x)$ be the maximum point in the design D. Conditional on $f(D)$,

$$u(D)|f(D) = f^*(D) \tag{5.3}$$

is a known quantity, whereas

$$u(\tilde{D})|f(D) = \begin{cases} f(\tilde{x}) & \text{if } f(\tilde{x}) > f^*(D) \\ f^*(D) & \text{otherwise} \end{cases} = \max\{f(\tilde{x}), f^*(D)\} \tag{5.4}$$

is random, depending on the unknown value of $f(\tilde{x})$. Conditional on $f(D)$, the improvement is

$$I(\tilde{x})|f(D) = \max\{f(\tilde{x}) - f^*(D), 0\}. \tag{5.5}$$

Therefore, the expected improvement is

$$EI(\tilde{x}) = E[I(\tilde{x})|f(D)], \tag{5.6}$$

where the expectation is taken over $f(\tilde{x})$, which is normally distributed with mean and variance which can be calculated from the GP fit to $(D, f(D))$. In a single-level case of the multi-level GP approximation with no noise the expected improvement is available in closed form, which makes it easy to use. Some useful properties of the EI are that EI is zero at points that are already in the design and the distance from a sampled point increases then EI increases as well.

The expected improvement process works following Algorithm 3. We start by choosing an initial set of sampled points spread over the entire design space. Preferably a design with space-filling properties so that we uniformly cover the domain to explore the function globally. We then evaluate the function at the initial design and use the EI formula from (5.6) to choose the point that gives the greatest EI from a set of candidate points and we add it to the existing design. This process stops when the EI is smaller than a predefined tolerance value.

Note that the main focus of the examples used in the thesis is the location of the maximum and we assume MLE for the underlying inference. In a different situation, where we work in Bayesian inference we would need to emulate the whole likelihood surface rather than just the maximum.

Generalised linear mixed model example

As a demonstration of the EI we have a single-level approximation of the log-likelihood of the GLMM model. Figure 5.1 shows the posterior mean of the GP calculated in

Algorithm 3 Computing expected improvement

1. **Choose** a small initial set of sampled points spread over the entire design space;
2. **Evaluate** the true function at these points;
3. **Use** EI formula

$$EI(\tilde{x}) = E[I(\tilde{x})|f(D)]$$

   to pick the location of the next evaluation among some candidate points - choose the point that gives the greatest expected improvement;
4. After observing that point **add** it to the training set and repeat;
5. **Stop** when the maximum of the EI is smaller than a tolerance value.

red calculated using the initial experimental design consisting of three points given in green. The credible intervals are shown with the blue shaded area and the accurate approximation with the dashed orange line.

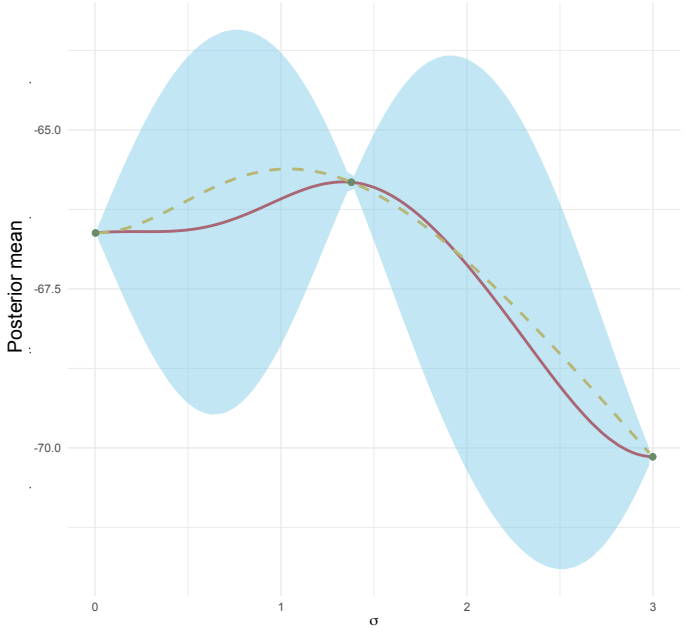
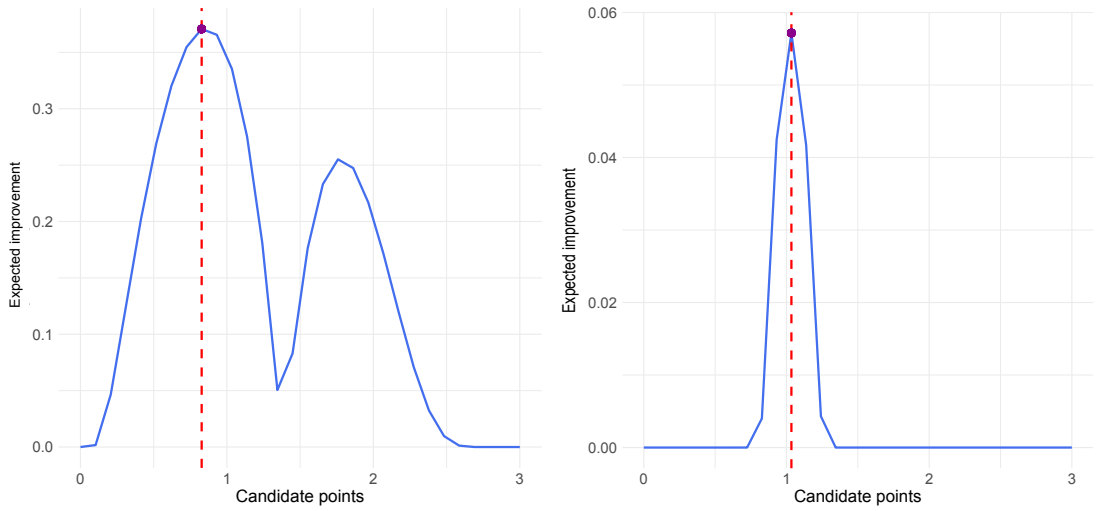


FIGURE 5.1: Posterior mean (red curve) as a single-level approximation to the log-likelihood for the GLMM example with three training points as the initial design (green dots) and the accurate log-likelihood (orange dashed curve).

We add new points to the design using Algorithm 3. The results of the EI for all the candidate points and the resulting GP posterior mean after each addition of a new point are shown in Figure 5.2 with the chosen point to add to the design indicated in purple. For the first iteration, shown in Figure 5.2a, the EI curve has one peak at the candidate point $\tilde{x} = 0.83$ indicating the we would sample there next. For the next iteration, as

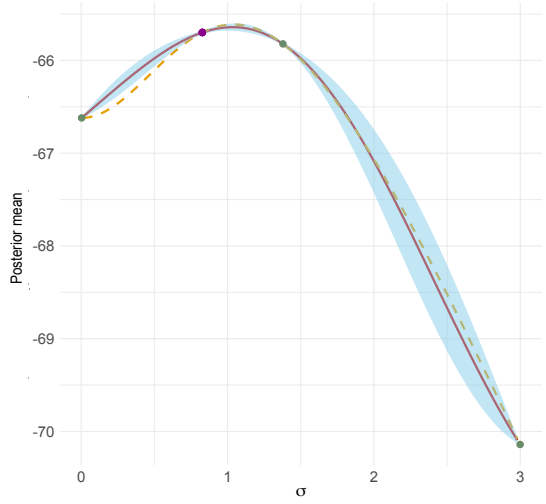
shown in Figure 5.2b, we add the new point at $\tilde{x} = 1.03$. The final EI iteration and the resulting posterior mean are given in Figure 5.3 where the new point is at $\tilde{x} = 1.03$ again.

The prior mean structure for this example was constant so that we could be able to demonstrate how the EI works since using a quadratic prior mean structure resulted to the GP approximation being overconfident and overestimating the log-likelihood. This shows that even with not the appropriate prior mean structure the GP approximation can benefit from using the EI to choose the experimental design.

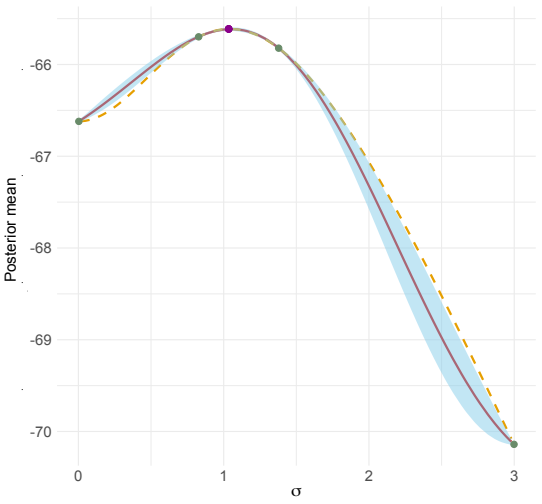


(A) Expected improvement I

(B) Expected improvement II



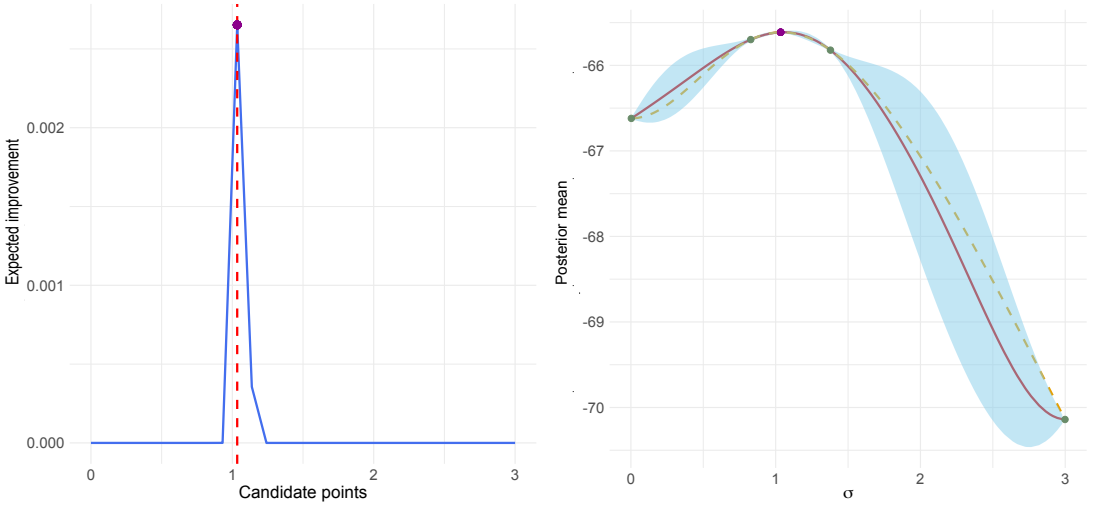
(C) GP approximation I



(D) GP approximation II

FIGURE 5.2: Expected improvement for each candidate point for two iterations and resulting GP posterior mean after the addition of a new point (purple dot) for the single-level approximation of the GLMM example.

The hyperparameters of the GP were re-estimated after the addition of each new point to the design. The estimated hyperparameters of the GP for each iteration are shown



(A) Expected improvement (B) GP approximation

FIGURE 5.3: Final expected improvement iteration and GP approximation using the posterior mean (red curve) as a single-level approximation to the log-likelihood for the GLMM example. New point added is shown with the purple dot.

in Figure 5.4. The blue dots give the estimated signal variance hyperparameter and the red dots give the estimated length-scale hyperparameter values. As can be seen, the hyperparameters tends to decrease after the first few iterations of the EI. The addition of uncertainty bounds in Figure 5.4 and subsequent figures of hyperparameter estimation will be beneficial and we will consider this as part of the future work.

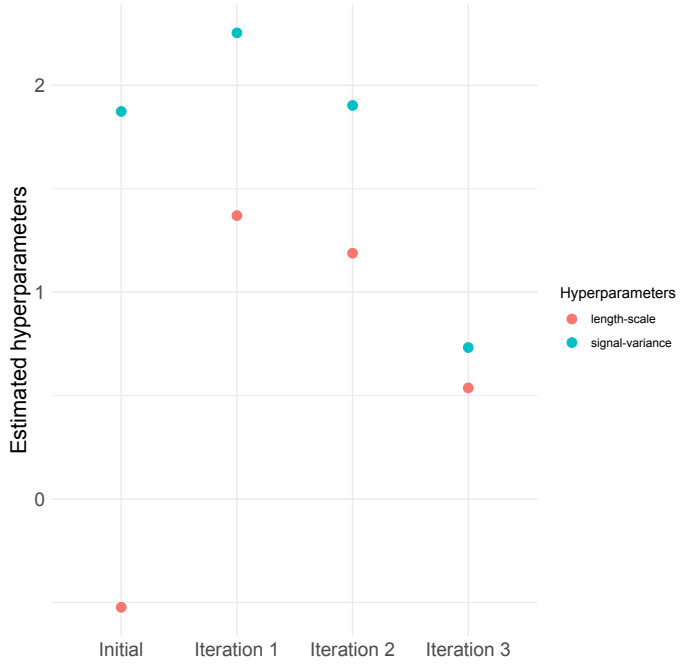


FIGURE 5.4: Hyperparameter estimation after each iteration of the expected improvement and the addition of new design points for the GLMM example.

5.2.3 Expected improvement for noisy observations

Suppose that we do not observe $f(x_i)$ but we observe

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2).$$

Given noisy observations the GP regression proceeds similarly to the noiseless case and we obtain the GP posterior as explained in Section 3.2.5. Computing EI for noisy observations is challenging because we no longer know the function value of the current best point. The utility of a design D can still be defined as in (5.1) but we no longer observe $f(D)$, and we need to condition on the observed values of y_i at the design points $D = (x_1, \ldots, x_n)$ which are $y(D) = (y_1, \ldots, y_n)$.

The expected improvement is

$$EI(\tilde{x}) = E[I(\tilde{x})|y(D)], \tag{5.7}$$

which is not available in closed form for noisy observations. Calculating the EI in the noisy case is more challenging that in the noise-free case.

It is questionable whether the utility (5.1) is sensible for the noisy case. In the noise-free setting, this utility could be justified as taking a cautious approach, only choosing to report a point as optimal if we have evaluated the function $f(.)$ there. This guard against the possibility that the true function values outside of the design points may differ substantially from the values predicted under the model.

However, in the noisy case, we do not observe the function values even at the design points, so this justification for the utility (5.1) no longer holds. A consequence of using this utility is that we will never choose the same point twice in the design as the expected improvement for a point already in the design is zero. This is a weakness of this approach. Intuitively it could be helpful to evaluate the function several times at a single point, in order to improve our knowledge of the value of the underlying function there. When evaluations have noise, the final solution reported will necessarily include uncertainty since we can hardly evaluate it an infinite number of times (Frazier, 2018).

For the likelihood approximations we are interested in the rest of the thesis, the underlying function is deterministic, so in the absence of error we will always get the same point, which the GP will exactly interpolate. The methodology of this chapter can be applied to underlying functions which are stochastic.

Numerous methods in the literature approach Bayesian optimisation for noisy observations. The knowledge gradient, entropy search, and parallel entropy search acquisition functions apply directly in the setting where noise is included, and they retain their one-step optimality properties.

Knowledge gradient can outperform expected improvement substantially in problems with substantial noise (Frazier et al., 2009). The knowledge gradient accounts for the inclusion of noise and does not restrict the final recommendation to a previously sampled point. It explores a class of solutions wider than just the ones that have been previously evaluated when recommending the final solution. A simple way to compute the knowledge gradient acquisition function is by using simulation (Frazier, 2018).

The entropy search (Hennig and Schuler, 2011) acquisition function values the information there is about the location of the global maximum according to its differential entropy. It explores the point to evaluate that causes the largest decrease in differential entropy. Predictive entropy search (Hernández-Lobato et al., 2014) looks for the same point, but uses a reformulation of the entropy reduction objective based on mutual information. Entropy search can be computed and optimised approximately (Hennig and Schuler, 2011), however, it is still challenging because the entropy of the maximiser of a Gaussian process is not available in closed form and need a large number of evaluations (Frazier, 2018).

Unlike EI, which only considers the posterior at the point sampled, knowledge gradient, entropy search and predictive entropy search are influenced by how the measurement alters the posterior over the whole domain, not only whether it has an improvement over an incumbent solution at the sampled point which is beneficial when working with noisy observations. Therefore, they can provide significant value to the Bayesian optimisation problem compared to EI.

Knowledge gradient, entropy search and predictive entropy search are valid approaches for dealing with noisy observations. However, for this thesis we do not consider knowledge gradient or entropy search further since we are interested in a method that can be expanded to consider multi-level Bayesian optimisation as well as noisy observations. Therefore, we introduce in the next section the expected gain in utility.

5.3 Expected gain in utility

5.3.1 Introduction

The utility (5.1) corresponds to a certain **reporting method** which is a method for choosing which point x to report as the **best** point, which maximises the function $f(.)$, once we have chosen a design and fitted a GP. By considering different possible reporting methods, we can derive corresponding utility, improvement and expected improvement functions.

In general, let $x^*(D)$ be the reported maximiser of the function $f(.)$, given design points D and corresponding response values $y(D)$. Note that the response values could be of any kind of function evaluations such as noise-free function evaluations, noisy function evaluations, or function evaluations at different accuracy levels, which is what we work with. Given any such $x^*(D)$, the utility for the design D can be defined as

$$u^*(D) = f(x^*(D)). \tag{5.8}$$

The utility (5.1) for the EI can be considered as a special case of (5.8) by choosing $x^*(D) = \arg\max_{x \in D} f(x)$. Note that the reported maximiser $x^*(D)$ behaves in the same way as in expected improvement so that we choose the point in the design D that gives the largest value of $f(.)$.

The function we want to maximise is the posterior mean of $f(.)$. Hence, we have as our reporting method x^* that maximises the posterior mean of $f(.)$, given the data. That is

$$x^*(D) = \arg\max_{x \in \mathcal{X}} \mu_f(x|D, y(D)), \tag{5.9}$$

where $\mu_f(.|D, y(D))$ is the posterior mean and $\mathcal{X}$ is the set of all possible values of x.

In general, the improvement can be given by

$$I(\tilde{x}) = u(\tilde{D}) - u(D) = f(x^*(\tilde{D})) - f(x^*(D)). \tag{5.10}$$

The expected improvement can be considered as the difference in expected utilities between the current design and the design with the candidate point $\tilde{x}$ added. To distinguish between this extension of expected improvement and the original definition of expected improvement, we follow the work of the unpublished manuscript of Waite and Woods (2017) and call this quantity the Expected Gain in Utility (EGU) which is given by

$$\text{EGU}(\tilde{x}) = E[I(\tilde{x})] = E[u(\tilde{D})] - E[u(D)] = E[f(x^*(\tilde{D}))] - E[f(x^*(D))]. \tag{5.11}$$

Hence, we need to compute the expected utility for the current design, $E[u(D)]$, and the expected utility of the design with the added candidate point $\tilde{x}$, $E[u(\tilde{D})]$. Some useful properties of the EGU is that the EGU is non-negative and it can be shown that the EI is a special case of the EGU (Waite and Woods, 2017).

Recall that for the cases we work on in this thesis, to find $x^*(D)$, we fit a Gaussian process to the data $(D, y(D))$. Depending on the application, this might be a simple GP without noise, a GP with noise, or a multi-level GP. We then compute the posterior mean for the function $f(.)$, and find the value of x which maximises this posterior mean. At the point which we consider adding $\tilde{x}$, the responses $y(D)$ are known, so $x^*(D)$ is fixed.

It can be shown that the expected utility of the current design is the maximum of the posterior mean

$$E[u(D)] = E[f(x^*(D))] = \mu_f(x^*(D)|D, y(D)) = \max_{x \in \mathcal{X}} \mu_f(x|D, y(D)). \tag{5.12}$$

For notation purposes, let $\mu_f^*(x|D, y(D))$ be the maximum of the current posterior mean, hence, $E[u(D)] = \mu_f^*(x|D, y(D))$.

Computing the expected utility of the design with added $\tilde{x}$ it is a bit more challenging since the expectation is with respect to the posterior distribution of the function $f(.)$ and the response at the candidate point $y(\tilde{x})$, given the current training data $(D, y(D))$. The expected utility of the design with added $\tilde{x}$ can be decomposed as

$$E[u(\tilde{D})] = E_f\{E_{y(\tilde{x})}[u(\tilde{D})]\}.$$

We can use a Monte Carlo approximation to $E[u(\tilde{D})]$ if in the inner expectation we fix the function $f(.)$ and calculate the expected utility conditional on this and in the outer expectation, we take the expectation with respect to the current posterior distribution for $f(.)$. Another idea to approximate $E[u(\tilde{D})]$ based on Waite and Woods (2017) is to switch the ordering of the conditioning, resulting to

$$E[u(\tilde{D})] = E_{y(\tilde{x})}\{E_{f|y(\tilde{x})}[u(\tilde{D})]\}. \tag{5.13}$$

In the inner expectation, we fix the value of the response at the candidate point $y(\tilde{x})$ and calculate the expected utility conditional on this. In the outer expectation, we take the expectation with respect to the current posterior distribution for $y(\tilde{x})$. The inner expectation from (5.13) is given by the maximum of the new posterior mean of $f(.)$ given the additional point $(\tilde{x}, y(\tilde{x}))$. That is,

$$\begin{aligned} E_{f|y(\tilde{x})}[u(\tilde{D})] &= E_{f|y(\tilde{x})}[f(x^*(\tilde{D}))] \\ &= \mu_f(x^*(D)|\tilde{D}, y(\tilde{D})) \\ &= \max_{x \in \mathcal{X}} \mu_f(x|D + \tilde{x}, y(\tilde{D})) \\ &= \mu_f^*(\tilde{D}, y(\tilde{D})). \end{aligned} \tag{5.14}$$

The posterior predictive density for $y(\tilde{x})$ given the current training data $(D, y(D))$ is a normal density derived from the Gaussian process posterior at candidate point $\tilde{x}$. For notation, let $\mu_y(\tilde{x})$ be the posterior predictive mean and $\sigma_y(\tilde{x})$ be the standard deviation of $y(\tilde{x})$. We have that $E[u(\tilde{D})]$ can be written as an integral

$$E[u(\tilde{D})] = \int_{-\infty}^{\infty} \mu_f^*(\tilde{D}, y(\tilde{D}))\phi(y, \mu_y(\tilde{x}), \sigma_y(\tilde{x}))dy, \tag{5.15}$$

where $y(\tilde{D}) = y(D) + y$ and $\phi(y, \mu, \sigma)$ is the $N(\mu, \sigma^2)$ density. We could approximate this one-dimensional integral by using quadrature. Suppose that $z_1, \ldots, z_{n_q}$ and

$w_1, \ldots, w_{n_q}$ are standard Gauss-Hermite quadrature (Liu and Pierce, 1994) points and weights such that

$$\int_{-\infty}^{\infty} g(z) \exp(-z^2) dz \approx \sum_{i=1}^{n_q} w_i g(z_i). \tag{5.16}$$

Let $y_i = \mu + \sqrt{2}\sigma z_i$, $v_i = w_i/\sqrt{p}$, then (5.16) becomes

$$\int_{-\infty}^{\infty} g(y)\phi(y, \mu, \sigma) dz \approx \sum_{i=1}^{n_q} y_i g(y_i). \tag{5.17}$$

For a multi-level case we are concern about which is the best level to add the new point. Therefore, we set $y(\tilde{x}) = f_i(\tilde{x})$ if $\tilde{x}$ is a candidate point to be added to level i.

Recall that our aim is to approximate $E[u(\tilde{D})]$ at each candidate point $\tilde{x}$ to compute the EGU from (5.11). The procedure for approximating $E[u(\tilde{D})]$ is described in Algorithm 4. We start by fitting a GP approximation using the initial training data to compute the posterior mean and posterior covariance. We compute quadrature points based on the mean and covariance and we compute the maximum of the posterior mean for each quadrature point after fitting a new GP where the new point has been added to the design. We continue by finding the maximum of the posterior mean and we compute an approximation to the expected utility at the new design given by

$$E[u(\tilde{D})] \approx \sum_{i=1}^{n_q} v_i \mu_i^*.$$

Algorithm 4 Approximating $E[u(\tilde{D})]$ at each candidate point $\tilde{x}$

1. **Find** the mean $\mu_y(x)$ and standard deviation $\sigma_y(\tilde{x})$ of posterior predictive distribution of $y(\tilde{x})$ given the current Gaussian process fitted to the initial training data $(D, y(D))$;
2. **Find** quadrature points $y_i = \mu_y(\tilde{x}) + \sqrt{2}\sigma_y(\tilde{x})z_i$, $i = 1, \ldots, n_q$;
3. **Compute** $\mu_i^* = \mu_f^*(D + \tilde{x}, y(D) + y_i)$ for each quadrature point $y_i, i = 1, \ldots, n_q$, by fitting a new Gaussian process with the new point $(\tilde{x}, y_i)$ added to the training set;
4. **Find** the maximum of the new posterior mean of $f(.)$;
5. **Calculate** $\sum_{i=1}^{n_q} v_i \mu_i^*$ as an approximation to $E[u(\tilde{D})]$.

Therefore, we can combine the approximation of $E[u(\tilde{D})]$ from Algorithm 4 with the expected utility of the current design from (5.12) to compute the expected gain in utility from the formula

$$\text{EGU}(\tilde{x}) = E[u(\tilde{D})] - E[u(D)].$$

The quality of the approximation of $E[u(\tilde{D})]$ should improve as the number of quadrature points n_q increases. We have also used a Monte Carlo approach for approximating $E[u(\tilde{D})]$ but it was more time-consuming, compared to the approximation presented in Algorithm 4. This happened because the MC approach required fitting a new GP at each of the simulation runs for each candidate point at which we would like to evaluate the expected gain in utility. For sufficiently large n_q the approximation in Algorithm 4 was within the uncertainty interval for the analogous Monte Carlo approach.

5.3.2 Applications

Introduction

We consider some simple examples to demonstrate the implementation of the EGU. For each example we start with an initial design D, set of training points, and each time we add one more point to the design based on the results of EGU. The posterior mean of the Gaussian process is been plotted each time, and we can see how the uncertainty reduces with the addition of new training points. The estimated values of the hyperparameters after each iteration are presented as well.

The EGU for all the candidate points and the location and the value of the maximum, which is where we add the new point to the design are shown in the relevant plots. The EGU plots are useful as a visual confirmation of convergence. With the addition of new points in the design, based on points that maximise the EGU, the value of EGU will eventually settle down. As stopping criteria of the EGU we define a tolerance value of EGU that we are satisfied with.

Generalised linear mixed model

Recall the GLMM example in Section 3.4.3 where we want to compute an approximation the log-likelihood. We work with a single-level GP approximation. We start with the initial design and we fit a GP. In Figure 5.5 we have the initial posterior mean, given by the red curve, which can be used as the approximation to the log-likelihood. Our aim is to maximise the posterior mean using the EGU to choose the experimental design. The green dots are the training points and the blue area gives the 95% credible interval estimate. We use this simple example as a demonstration for the EGU algorithm for the single-level approximation case. For this example we do have an accurate approximation of the log-likelihood given by the orange dashed curve.

Figure 5.6 shows the EGU function for the GLMM example for the first few iterations. The EGU of the first iteration has one peak at the candidate point $\tilde{x} = 1.26$ indicating that we would sample there. For the following iteration, as shown in Figure 5.6b the

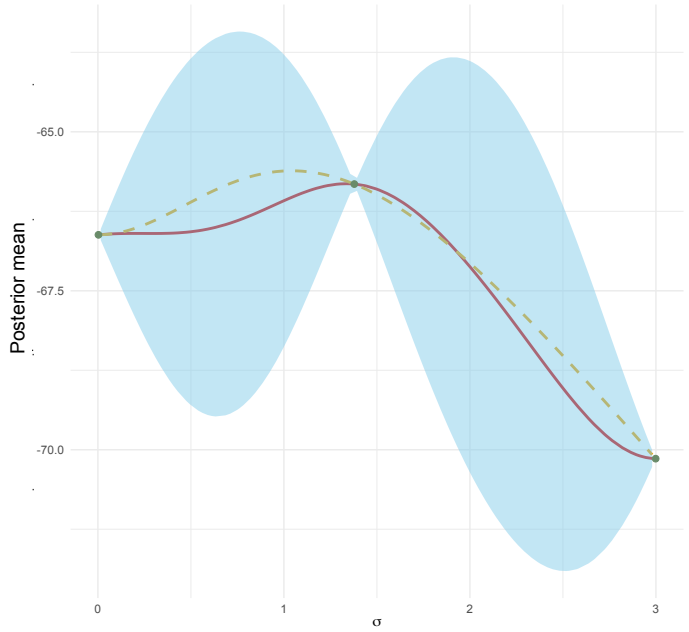


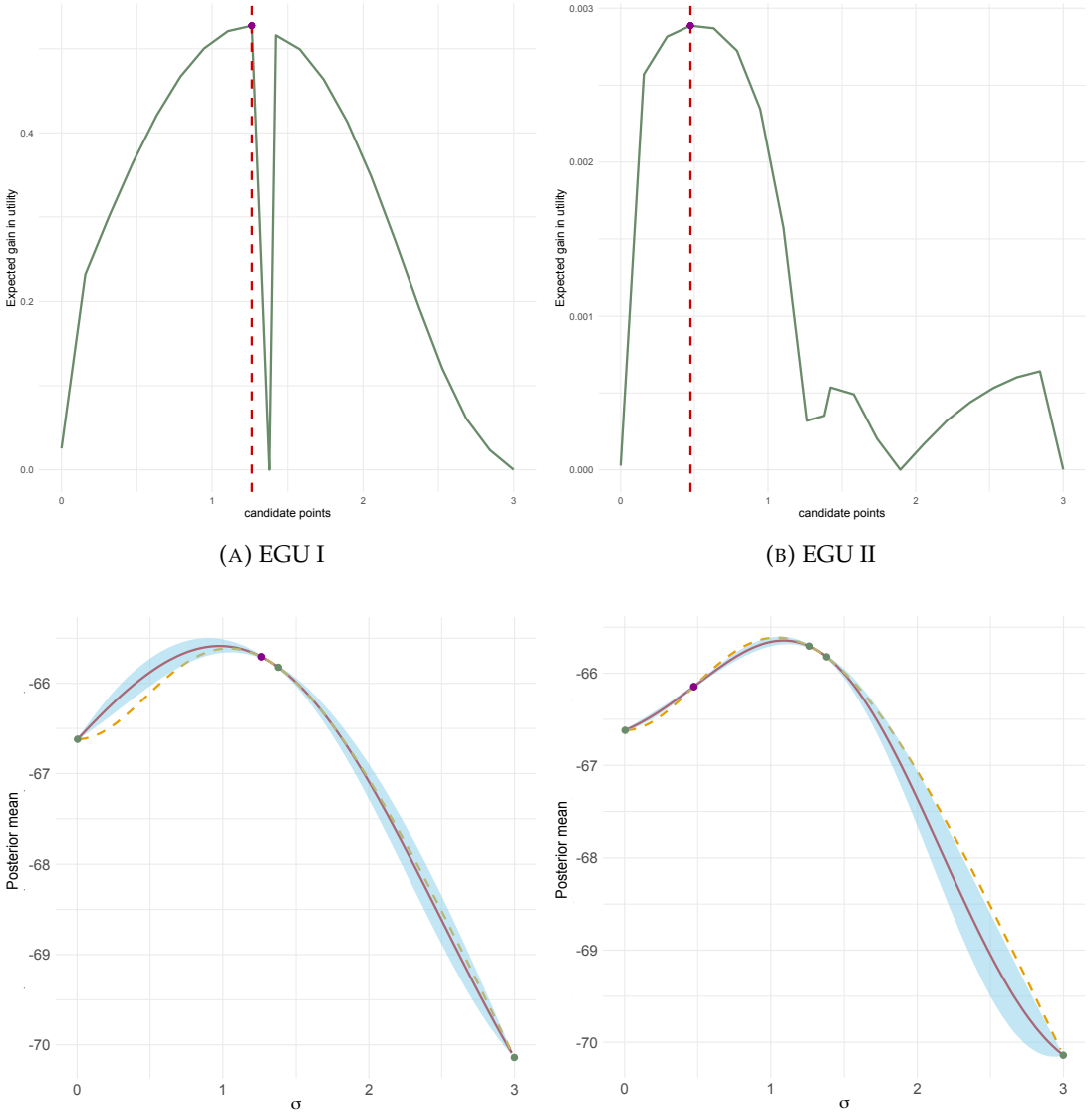
FIGURE 5.5: Posterior mean (red curve) as a single-level approximation to the log-likelihood for the GLMM example with two training points as the initial design (green dots) and the accurate log-likelihood (orange dashed curve).

EGU is maximised at $\tilde{x} = 0.47$. The EGU meets its stopping criteria, hence there is no need to add any new points to the design.

The resulting Gaussian process posterior mean after each iteration of the EGU along with the new candidate points added (purple dots) each time are shown in Figures 5.6c and 5.6d. We can clearly observe an improvement on the posterior mean which is now closer to the accurate approximation of the log-likelihood (orange curve) and a decrease in uncertainty. At some areas, where there are limited design points the GP approximation underestimates the log-likelihood. However, the main focus of the EGU is the location of the maximum and not the whole likelihood surface. To deal with the case where we want to emulate the whole likelihood surface, a Bayesian inference approach would be more suitable.

Following this procedure we achieve a good approximation of our log-likelihood by avoiding multiple unnecessary random evaluations which might be beneficial for the approximation but costly at the same time.

The EGU function is multi-modal as shown in the EGU plots. Similarly to the EI approach, it is easy to demonstrate that EGU is zero at the sampled points. For this example, we have included a design point at $x = 1.73$ in the set of candidate points to demonstrate that the EGU is zero at points that have already been in the design. The EGU is positive in between the candidate points, though perhaps very small, as can be



(A) EGU I

(B) EGU II

(C) GP approximation I

(D) GP approximation II

FIGURE 5.6: EGU for each candidate point for the GLMM example for two iterations. Gaussian process posterior mean after each iteration of the EGU for the GLMM example with the new point added in purple.

seen in Figure 5.6. It is also common to have large areas where the EGU is essentially zero and so appears fairly flat.

We have the estimated hyperparameters of the GP for each iteration after adding the new candidate point to the design in Figure 5.7. The blue dots give the estimated signal variance hyperparameter and the red give the length-scale hyperparameter values.

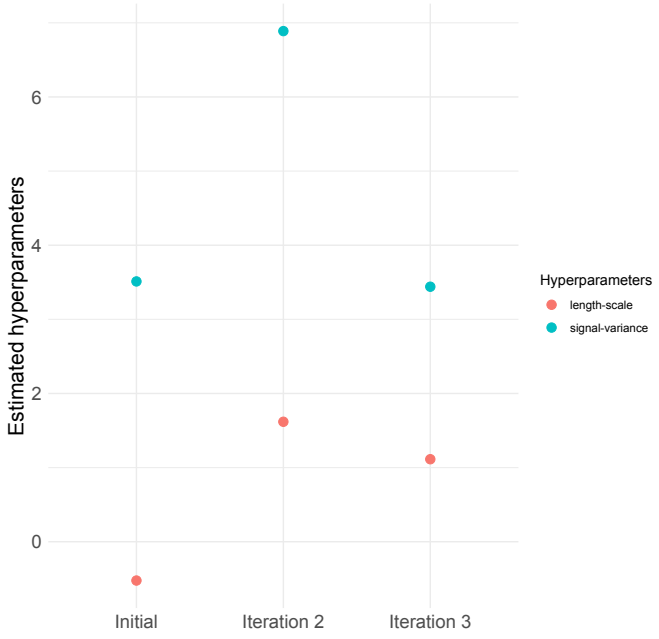


FIGURE 5.7: Hyperparameter estimation after each iteration of the EGU and the addition of new design points for the GLMM example.

Ising model

Recall the Ising model introduced in Section 2.4. We use the EGU algorithm to choose the design points for this example. Similarly to the GLMM example we have in Figure 5.8 the initial GP posterior mean as an approximation to the log-likelihood of the Ising model. As can be seen, we use only three space-filling training points for our initial design (green dots) and the approximation is not particularly good. The red curve giving the approximation is not close to the accurate approximation given by the orange dashed curve. That could be also due to the choice of the prior mean structure which was linear prior. However, we use this example which does not have a great initial fit, to demonstrate how the EGU works.

Figure 5.9 shows the first two iterations of the EGU for the Ising model example. To start with, EGU has one peak at the candidate point $\tilde{x} = 0.22$ indicating the we would sample there. For the next iteration, the expected EGU is maximised at $\tilde{x} = 0.28$ and thus we add this point in our design. The resulting Gaussian process posterior mean after each iteration of the EGU along with the new candidate points added (purple dots) each time is also shown in Figure 5.9.

The final EGU iteration and the resulting posterior mean are given in Figure 5.10 where the new point is at $\tilde{x} = 0.17$. We managed to obtain a really good approximation of the log-likelihood using the EGU. The estimated hyperparameters of the GP for each iteration after adding the new candidate point to the design in Figure 5.11 where there is not a big change in the hyperparameters and they follow a similar trend.

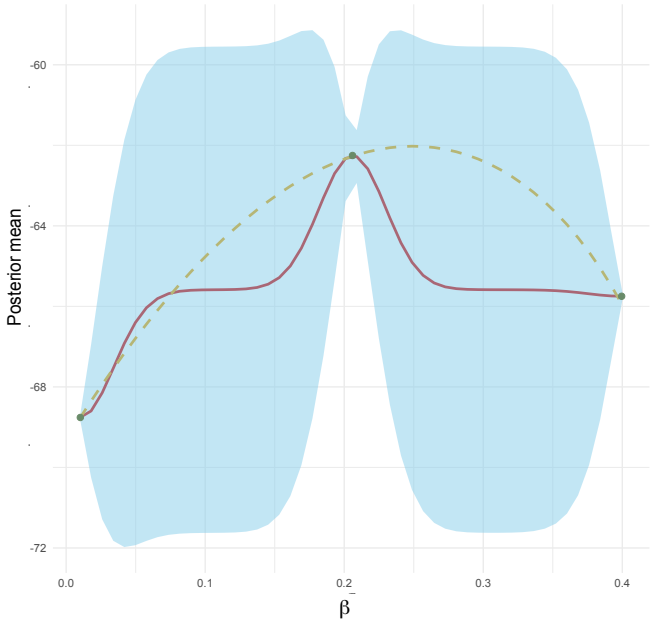


FIGURE 5.8: Posterior mean (red curve) as a single-level approximation to the log-likelihood for the Ising model example with the accurate log-likelihood approximation (orange dashed curve) and initial design points (green dots).

5.4 Bayesian optimisation for multi-level Gaussian process

5.4.1 Introduction

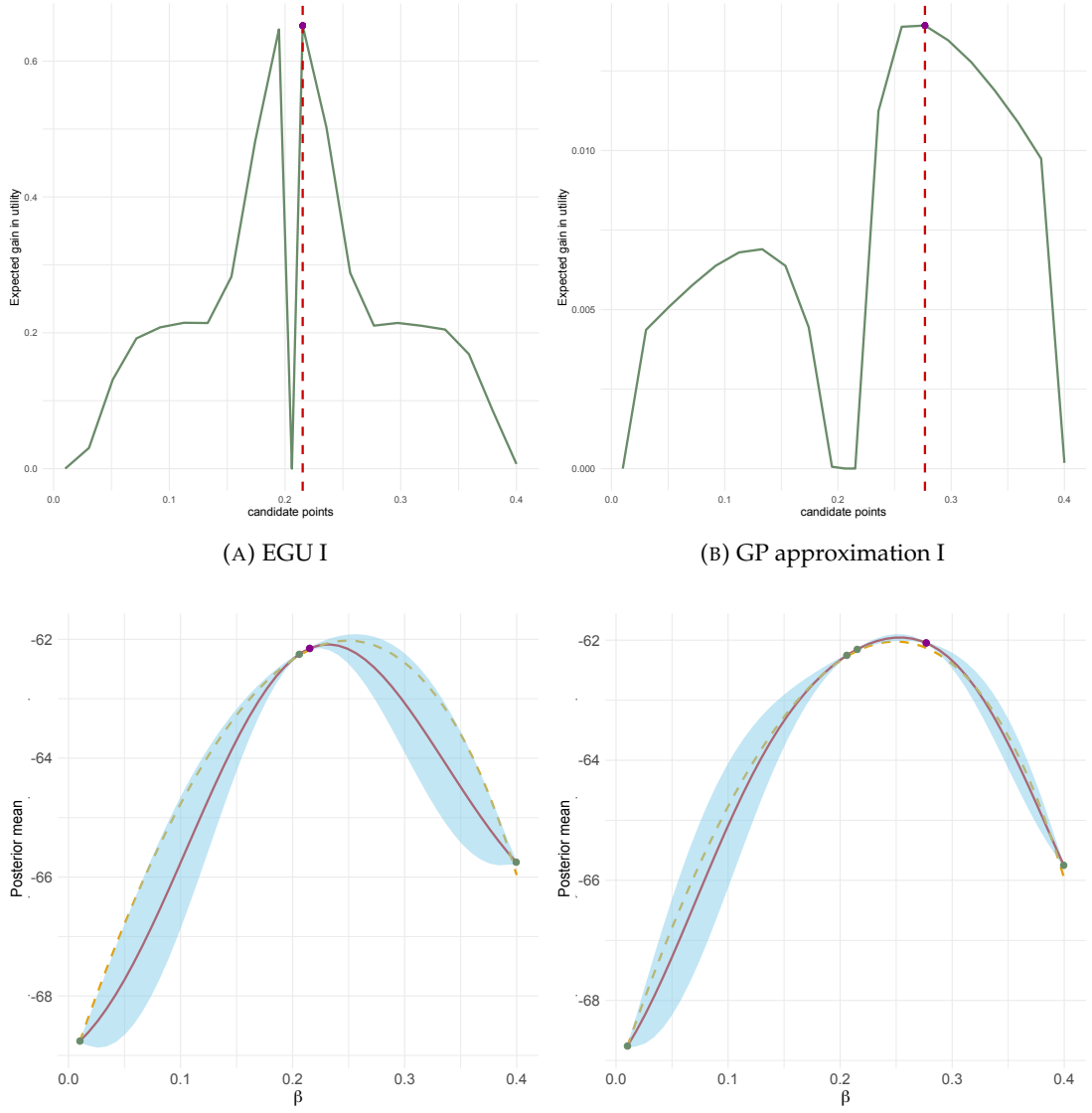
One of the main focuses of our work are systems that can be approximated with models of varying degrees of accuracy. It would be preferable to work with the most accurate model. However, highly accurate models may need more time to compute making optimisation not efficient for most algorithms. On the other hand, working with a fast, low-accuracy model may result to a lower accuracy. Instead of searching for the optimal intermediate degree of accuracy, a sensible approach is to use all levels of models and construct a utility based on multiple levels.

Considering the utility, suppose we have d levels, where level-1 is the least accurate and level-d is the most accurate, with design $D_i = (x_1^{(i)}, \ldots, x_{n_i}^{(i)})$ at level i, and noise-free evaluations of the function $f_i(.)$ at the level-i design points D_i, where $f_1(.) = f(.)$ is the function we wish to emulate.

The utility of a design $D = (D_1, \ldots, D_d)$ could be

$$u(D) = \max_{x \in D_1} f(x)$$

to match the initial utility (5.1) as closely as possible, since we only know the values of the function of interest $f(.)$ for the level-1 design points D_1. However, with this choice,



(A) EGU I

(B) GP approximation I

(C) EGU II

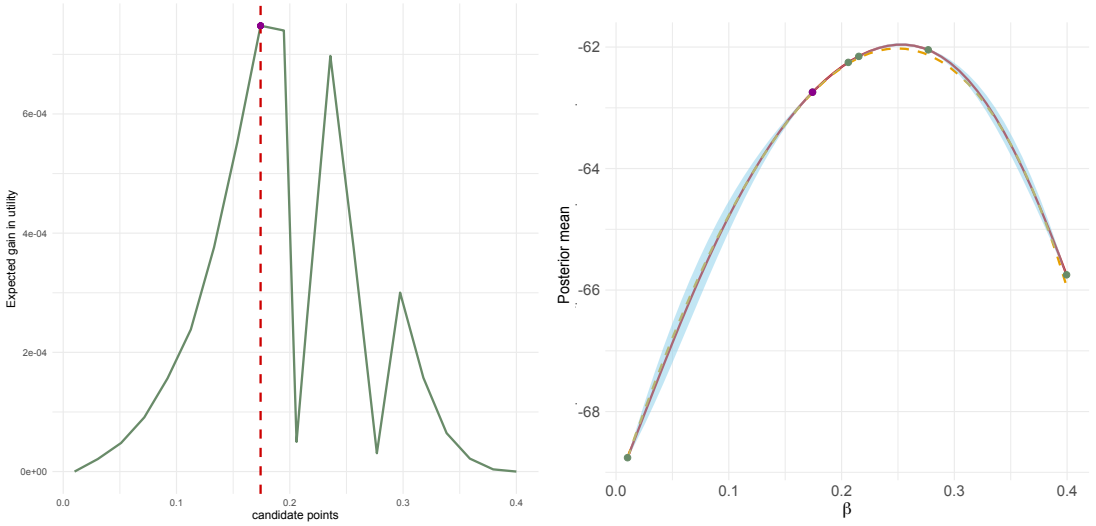
(D) GP approximation II

FIGURE 5.9: EGU for each candidate point for two iterations and Gaussian process posterior mean after each iteration of the EGU for the Ising model example. New point added in purple.

adding a point to the design at level-2 or above would leave the utility unchanged, so the improvement and expected improvement for such points are zero. With this choice of utility we would only ever choose to add points to the level-1 design.

5.4.2 Literature on multi-level Bayesian optimisation

Other researchers examine ways to do optimisation using both low and high-level models. Two of the approaches found in the literature for multi-level BO are the **two-stage learning**, where low-level approximations are trained for a certain amount of



(A) EGU III (B) GP approximation III

FIGURE 5.10: Final EGU iteration and resulting Gaussian process posterior mean for the Ising model example. New point added in purple.

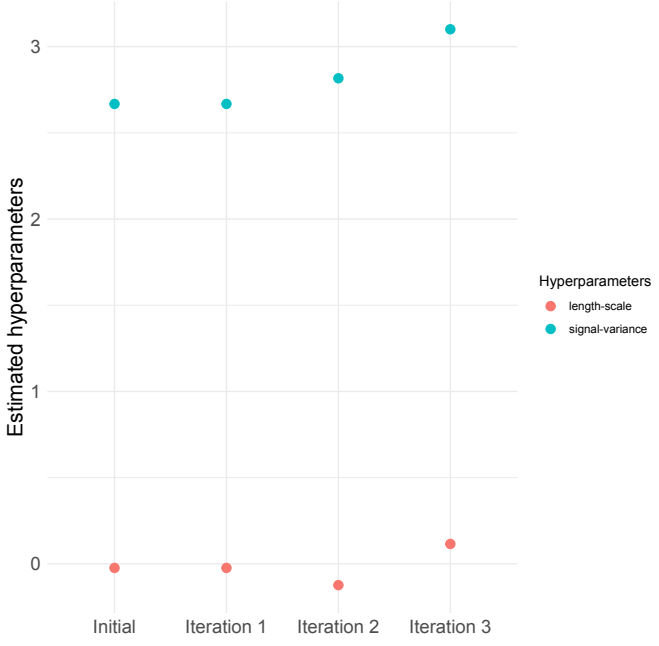


FIGURE 5.11: Hyperparameter estimation after each iteration of the EGU and the addition of new design points for the Ising model example.

time in order to warm-up - start the learning on the expensive more accurate approximation, and the **adding points to each level simultaneously** approach.

Considering the two-stage learning method, Alexander et al. (2007) use an **exchange algorithm** to choose which points of the search space to sample within each level of analysis. Their main idea is to explore the space at the low-level of accuracy and exploit at the high-level. They start by choosing an initial design for the low-level model, they

fit a GP approximation making sure that the cheap model achieves the desired model accuracy. Then, they select a subset of the design points of the lower level using their exchange algorithm and they build an initial multi-level model using both designs. Following an iterative process of updating the multi-level model with new data using expected improvement (probably any acquisition function could work) they evaluate next where the expected improvement is maximum.

Song et al. (2018) propose the use of a Multi-fidelity Mutual Information Greedy (MF-MI-Greedy) which is a general and principled multi-fidelity BO framework that prioritises maximising the amount of mutual information gathered across different levels. Consider the fidelity as a level of accuracy of a model. The cost of each lower fidelity is determined according to the maximal approximation error in function value when compared with the target fidelity. MF-MI-Greedy is divided in two phases: the exploration phase where the algorithm focuses on exploring the low fidelity actions and the optimisation phase where the algorithm tries to optimise the payoff function by performing an action at the target fidelity.

Another approach presented in Kandasamy et al. (2017) is to use the cheap approximations to guide search for the optimum of the function and reduce the overall cost of optimisation. The key idea is to choose a fidelity only if we have sufficiently reduced the uncertainty at all lower fidelities. They are implementing Gaussian process upper confidence bound algorithm (GP-UCB) which models the function as a Gaussian process and uses upper confidence bounds techniques to determine the next point of evaluation.

We propose the multi-level EGU which is an extension of the EGU presented in Section 5.3. The multi-level EGU work with all of the models simultaneously and takes into consideration the cost of each level rather than working with each level separately. It will be interesting to compare the results from the EGU for multi-level BO with methods from the literature based on efficiency, accuracy and cost as part of the future work.

5.5 EGU for multi-level Gaussian processes

5.5.1 EGU algorithm for multi-level Gaussian processes

Choosing the experimental design of each level of approximation in a multi-level approximation case is challenging and there are several factors that need to be taken into consideration. To tackle this issue, we introduce the EGU for multi-level Gaussian processes in order to choose the experimental design for each level of approximation based on its cost.

The EGU for a multi-level GP approximation is described in Algorithm 5. We start by estimating the hyperparameters of the multi-level GP and we fit the GP using the initial set of the design points of each level. Furthermore, we compute the EGU for each of the candidate points for each level separately based on Algorithm 4. With this step we compute what will be the expected gain in utility if we add the candidate point to each level.

As part of the process, it will be necessary to compute a factor that determines the cost of each level of approximation and calculate the **EGU per unit cost**. the The EGU per unit cost is calculated by dividing the maximum EGU of each level of the set of candidate points with the predefined cost of each level. We will examine if it will be better to add the next point at the higher level of approximation, which we already know that will be beneficial but could me costly, or to add a point at the lower levels of approximation by comparing the EGU per unit cost of each level.

Algorithm 5 EGU for multi-level GP approximation

1. **Estimate** the hyperparameters and fit the Gaussian process using the multi-level approximation using the initial set of training points;
2. **Compute** the EGU for each candidate point for each level separately based on the Algorithm 4;
3. **Calculate** the EGU per unit cost of each level, where

$$\text{EGU per unit cost} = \max(\text{EGU}/\text{cost});$$

4. **Compare** the EGU per unit cost for each level to choose where is best to add the point;
5. **Add** the new point to the level with the highest EGU per unit cost.

5.5.2 Applications

Overview

We demonstrate the use of EGU for multi-level Gaussian process approximation using two examples. For each example we present the EGU for each level of approximation, the decision made, which includes the new point added to the design and at which level. We also plot the resulting posterior mean of the GP, which is used as the approximation to the high-level approximation of the log-likelihood function and the estimated values of the hyperparameters after the addition of the new point to the design. We have the GLMM example demonstrating the two-level approximation and the Ising model for the three-level approximation case.

Generalised linear mixed model

For the two-level approximation of the likelihood of the GLMM we have the initial design and the posterior mean in Figure 5.12. We use Algorithm 5 to decide where and at which level is more beneficial to add the next training point.

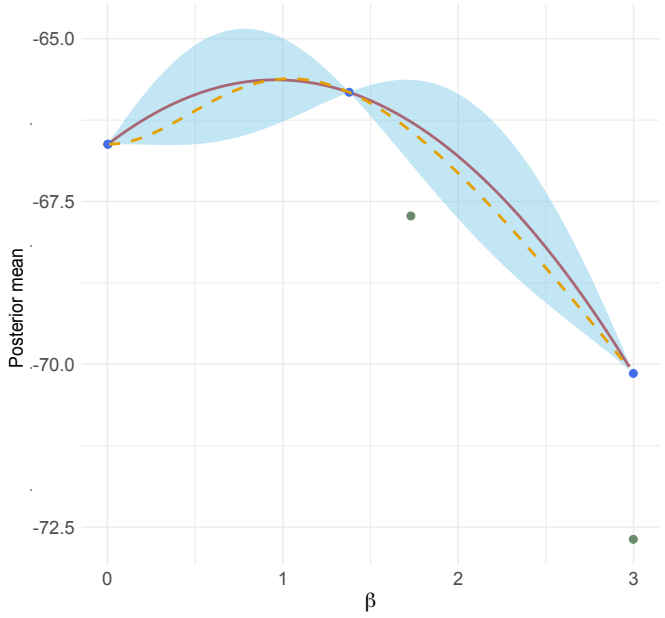
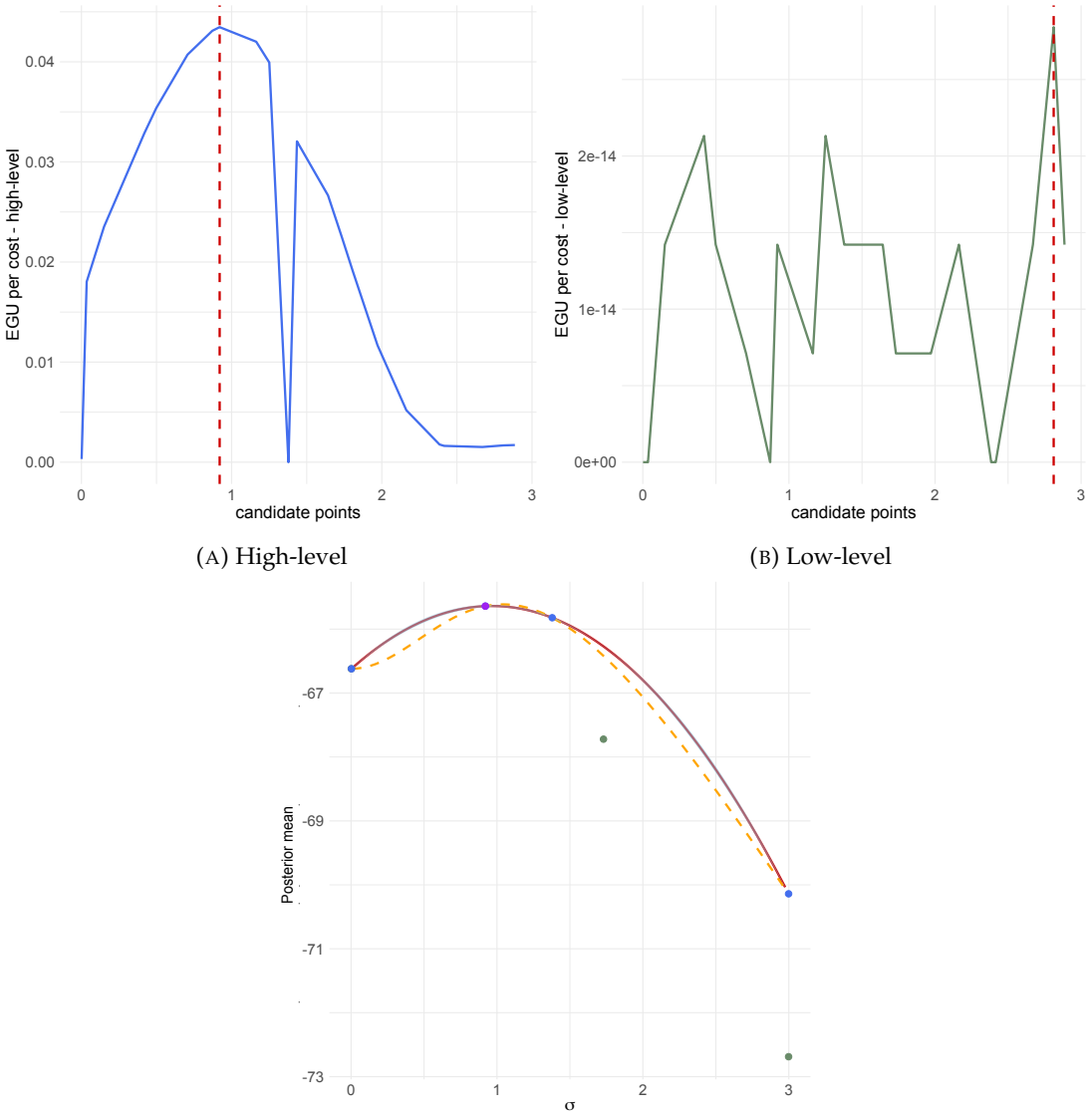


FIGURE 5.12: Posterior mean (red curve) as a two-level approximation to the log-likelihood for the GLMM example for the initial design with three points in each level.

For the GLMM example we use LA as the low-level and AGQ as the high-level. There is not a big difference between the cost of the LA and the AGQ methods. Therefore for illustration purposes of this example we choose arbitrary values of the cost of each level with the AGQ method being 10 times more costly.

Figure 5.13 shows the EGU per cost function for the GLMM example for the one iteration for the low-level and the high-level and the resulting GP posterior after the iteration of the EGU along with the new candidate points added in the design given in purple. We choose to add the new point at the level with the higher EGU per cost. Hence, we add the point to the high-level at $\tilde{x} = 0.92$. The stopping conditions for the EGU is met because there is no more gain from adding more points to the design. As can be seen, there is an improvement on the approximation and we manage to decrease uncertainty. However, for some values in the interval, the approximation is overconfident.

The estimated hyperparameters for each approximation level after each iteration are shown in Figure 5.14. For this example, we only needed one iteration of the EGU to choose the experimental design, Hence, we can not have a clear understanding about the behavior of the hyperparameters.



(A) High-level

(B) Low-level

(C) Gaussian process

FIGURE 5.13: EGU per cost of the candidate points for each level and resulting GP posterior after the addition of the new point (purple dot).

Ising models

In section 3.5.4 we introduced a third level of log-likelihood approximation for the Ising model. We would like to choose the experimental design of each level for a three-level GP approximation, which can easily be generalised for multi-levels. We follow the same procedure as for the two-level approximation EGU. The levels of approximation are based on the tuning parameter of the RDA method.

For the Ising model example we have a cost function for each level of the RDA method for each k is given by

$$O(c^2 + kc2^k), \tag{5.18}$$

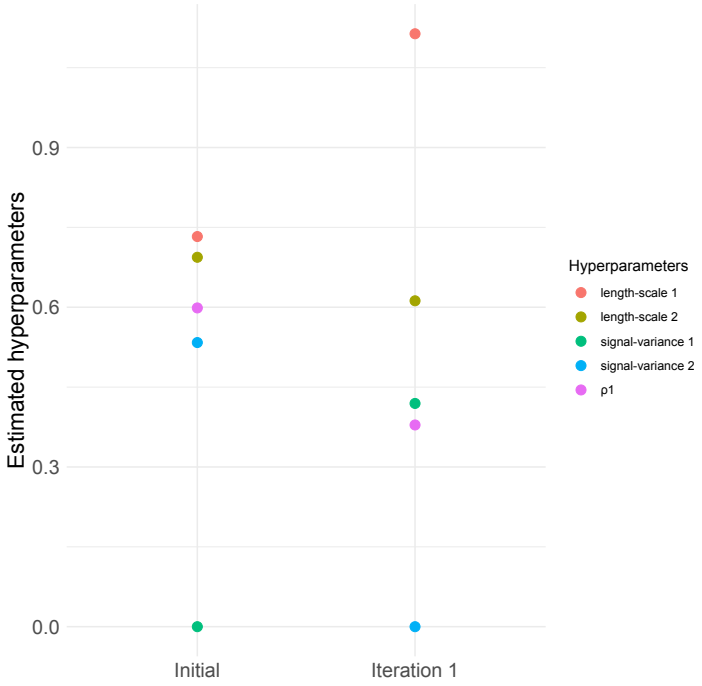


FIGURE 5.14: Hyperparameter estimation after the EGU iteration and the addition of the new design point for the GLMM example.

where $n = m$ for a $m \times m$ grid as described in Section 2.4. For this example $m = 10$. This cost can be used when computing the EGU per cost to decide at which level we could add the candidate point $\tilde{x}$.

The initial design for the example consists of four points in the low-level (green dots), 3 points of the middle level (pink dots) and 3 points in the high-level (blue dots) as shown in Figure 5.15. The tuning parameter of the RDA method is $k = 3, 4$ and 6 for each level respectively.

We add points to the design until there is no more benefit on the approximation of adding anymore points. This is based on a stopping criterion where the maximum value of EGU of all levels hits a minimum predefined threshold value.

Figure 5.16 shows the EGU per cost of each candidate point for each level of approximation. As can be seen, the EGU per cost of the low-level is higher than the EGU per cost of the high-level or the middle-level suggesting we add the next point to the low-level design. The approximation of the log-likelihood (red curve) considerably improves and gets closer to the exact log-likelihood (orange dashed line).

The EGU per cost for the next iteration for each level is given in Figure 5.17. The resulting posterior mean of the GP is also shown and the new point added to the design is given with the purple dot. Based on the EGU per cost of each level we add a new point at the low-level at $\tilde{x} = 0.16$. For the second iteration of the EGU we can see that the EGU of each level is negative, however it is really close to zero. The final GP

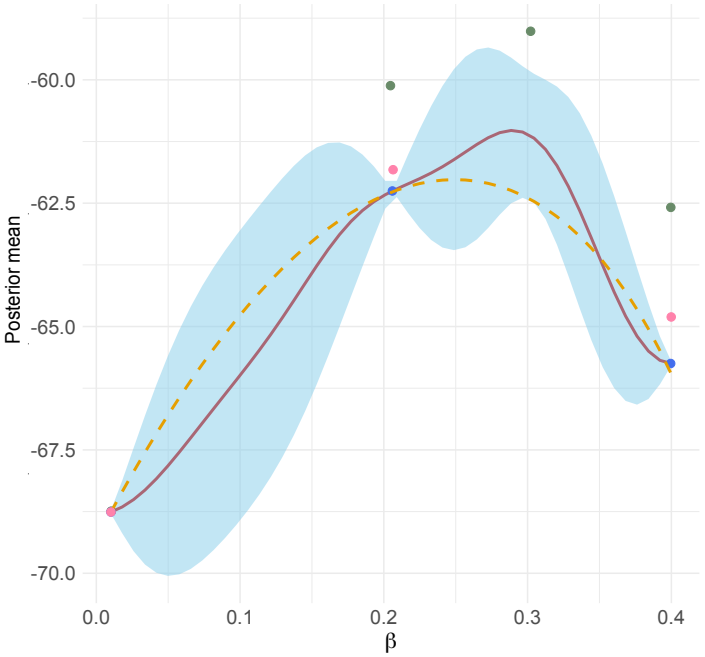


FIGURE 5.15: Posterior mean (red curve) as a three-level GP approximation to the log-likelihood for the Ising model example with the initial design: 4 training points of the low-level (green dots), 3 points of the middle-level (pink dots) and the 3 points of the high-level (blue dots).

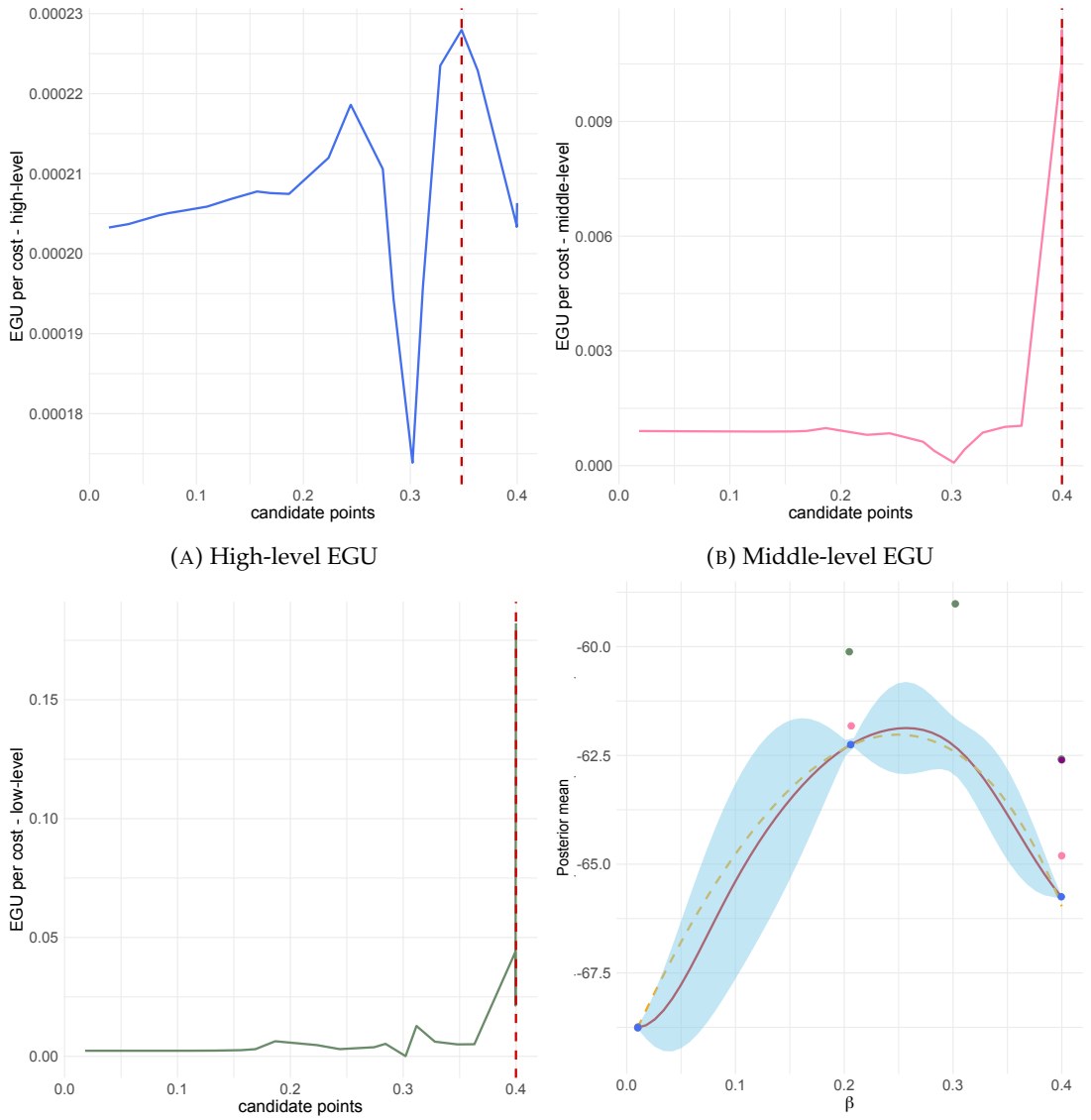
approximation has really small uncertainty and underestimates the log-likelihood for some parts of the interval.

The estimated hyperparameters for each approximation level after each iteration are shown in Figure 5.18. As can be seen, the hyperparameters converge to a single value after few iterations of the EGU.

5.6 Summary

In this chapter we presented a Bayesian optimisation approach to choose the experimental design (set of training points) for the Gaussian process emulator of the function of interest, which in our case is the log-likelihood function with the aim to find the point maximising the log-likelihood. The reporting method, utility, we use it such that we choose the new point that maximises the posterior mean of our function of interest.

We presented the multi-level EGU for BO which can be used when we aim to choose a new point to add to the existing design without further evaluations of the function approximations when multiple levels of approximation are available each with different



(A) High-level EGU

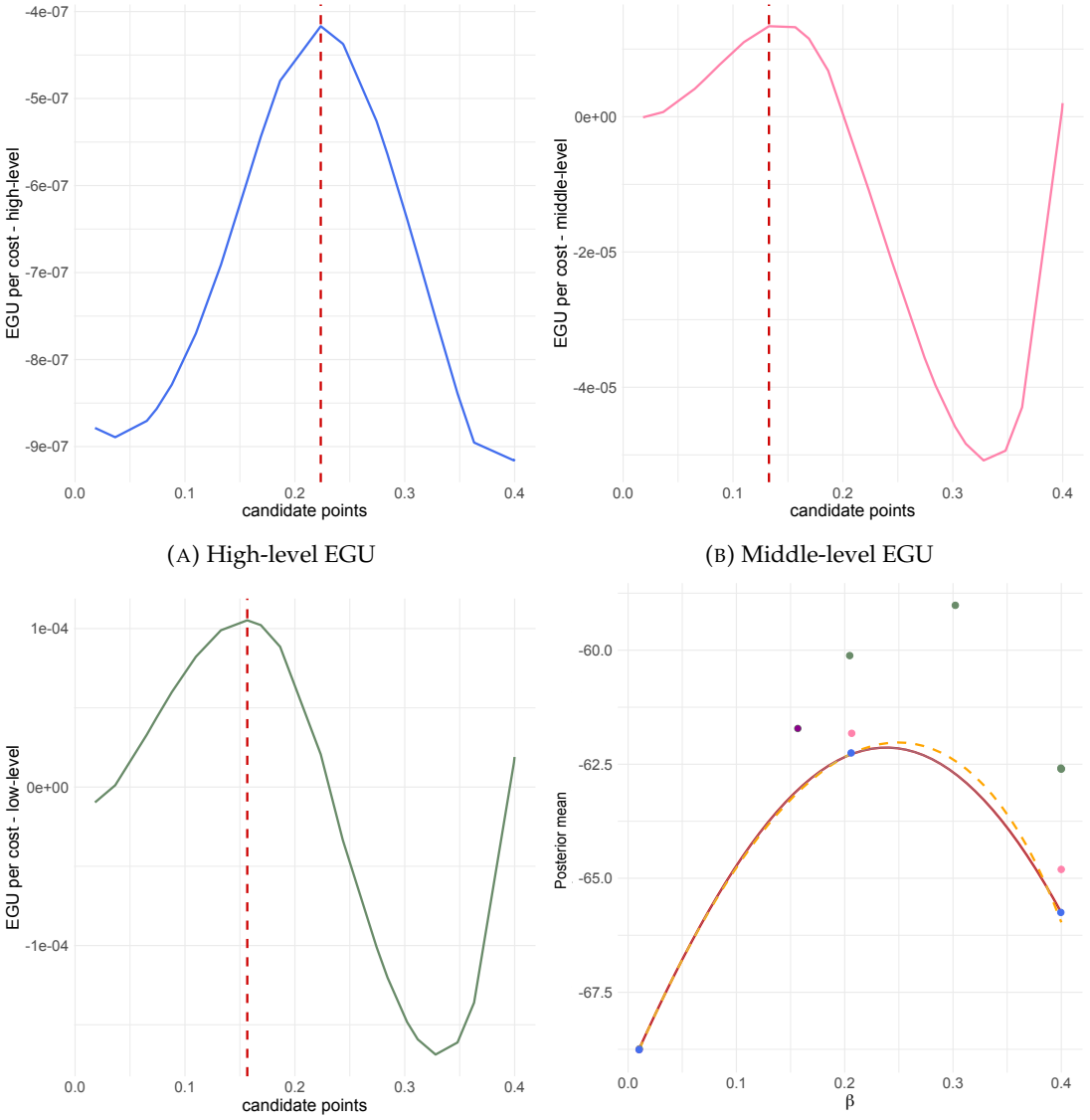
(B) Middle-level EGU

(C) Low-level EGU

(D) GP posterior mean

FIGURE 5.16: EGU per cost of the candidate points for each of the three levels and the resulting GP posterior for the Ising model example. New point added given by the purple dot - Iteration 1.

computational cost. The EGU can be used to develop high quality design for hierarchical experiments across multiple areas beyond the likelihood approximation examples we have worked with in this thesis.



(A) High-level EGU

(B) Middle-level EGU

(C) Low-level EGU

(D) GP posterior mean

FIGURE 5.17: EGU per cost of the candidate points for each of the three levels and the resulting GP posterior for the Ising model example. New point added given by the purple dot - Iteration 2.

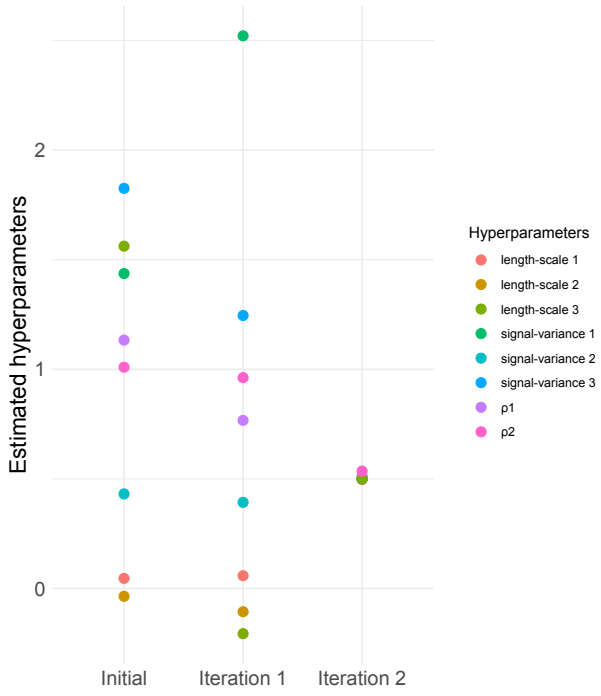


FIGURE 5.18: Hyperparameter estimation after each iteration of the EGU and the addition of new design points for the Ising model example for a three-levels GP approximation.

Chapter 6

R package hela: hierarchical experiments and likelihood approximations

6.1 Overview

The aim of this chapter is to present the R package hela which stands for hierarchical experiments and likelihood approximations. The package hela offers a statistical method for statistical design, modelling and inference using systems and approximations available on at least two hierarchical scales. Moreover, it provides a method for choosing the experimental design for experiments with multiple levels.

An example of the usage of hela is the approximation of intractable likelihood function using hierarchical experiments by combining approximations each having a different level of accuracy and cost. The user can input any model to approximate the likelihood they want. The package works for up to three levels of approximation of the intractable likelihood. For future versions of hela we would like to give the option to the user to use as many levels of approximation as they want. The package has three main functions, the gp_post(), hyper_est() and egu(). The package also has a plotting function gp_posterior_plot() to produce plots of the results.

The GLMM example that will be used in this chapter to demonstrate how the package works has already been presented in this thesis. Hence, for more detailed analysis of the model structure and the dataset used refer to Section 3.4.3.

The function gp_post() stands for Gaussian process posterior and it is used to compute a multi-level GP posterior distribution using the method presented in Section 3.3. The implementation of gp_post(), the arguments and the output are described in Section

6.3. The functions hyper_est() and hyperest_three() stand for hyperparameter estimation for two and three-level cases respectively, and they are used to estimate the hyperparameters of the GP using optimisation methods described in Section 3.2.8. The implementation of these two functions, the arguments and the output are described in Section 6.2. The egu() and egu_three() functions stand for expected gain in utility and they are used to compute the experimental design of two and three-level GP approximations respectively. The implementation of these functions, the arguments and the output are described in Section 6.4.

The hela package can compute an approximation using up to three levels of approximation. The extension to generalise hela for multi-level approximations, with no restriction on the number of levels used, will be included in future versions of hela.

The package is available on GitHub and can be installed using the R code

```
devtools::install_github("thdrnrch/hela", subdir="hela")
library("hela")
```

6.2 Hyperparameter estimation in hela

Usage of hyper_optim() and hyper_optim_three()

The hyperparameters of the Gaussian process can be estimated using the functions hyper_optim() and hyper_optim_three() for two and three-level approximations accordingly. The approach we use to estimate the hyperparameters $\boldsymbol{\zeta} = (\tau^2, l)$ for each level of the GP and the coefficient vector $\boldsymbol{b}$ is to maximise the likelihood as a function of the hyperparameters and $\boldsymbol{b}$. We do that by maximising first the $\boldsymbol{b}$-profile likelihood over the hyperparameter. The numerical optimisation method used for the estimation is the Nelder-Mead method (Nelder and Mead, 1965). More details for the method used to compute the hyperparameters are given in Section 3.2.8.

The functions have necessary and optional arguments. For both functions, the user needs to specify the initial values of the hyperparameters, that will be used for the optimisation, the training points of each level, the function evaluations of each level at the training points used and the prior mean structure. The usage of the hyper_optim() is given by

```
hyper_optim(hypers, train_points_low, train_points_high,
            l_train_low, l_train_high, prior_mean_str)
```

In a similar way, the usage of hyper_optim_three() is given by

```
hyper_optim_three(hypers, train_points_low, train_points_mid,
                  train_points_high, l_train_low, l_train_mid,
                  l_train_high, prior_mean_str)
```

The explanation of the arguments of the functions can be found below and in the documentation file of the functions by typing in the R console the code

```
?hyper_optim
?hyper_optim_three
```

Arguments of hyper_optim()

The hyper_optim() function has as its first argument the initial values of the hyperparameters, hypers, that the user wants to use in the optimisation to estimate the hyperparameters. hypers will have to be a vector containing the initial values of the signal variance parameter and the length scale parameter of each level and the initial value of the autoregressive parameter.

The user needs to specify the training points used for each level of approximation through the vectors train_points_low and train_points_high for low and high-level respectively. Similarly, the function has as arguments the evaluations of each approximation level at the specified training points through the vectors l_train_low and l_train_high for low-level and high-level respectively.

The prior mean structure of the Gaussian process needs to be specified as a function of the input points through prior_mean_str. The package has three options of prior mean functions available: linear prior mean, linear_prior(), constant, const_prior(), or quadratic, quad_prior() structures, or the user can insert their own prior mean function as a function of the input points. The function uses a nugget term, p, which is predefined in the function and used for computational reasons as explained in Section 3.2.6. The default value is p = 1e-4. We choose a small nugget term so it will not affect the results.

An example of how the function works is given in Section 6.6 for the two-level GP approximation of the GLMM example. A table of the arguments of the hyper_optim() function and their description can be found in Table A.1 in Appendix A. The function arguments with the asterisk are mandatory for the function.

Output of hyper_optim()

The function hyper_optim() returns a list containing the estimated hyperparameters (optim_hypers) of each level:

```
The signal variance parameter of the low-level is
The length-scale parameter of the low-level is
The signal variance parameter of the high-level is
The length-scale parameter of the high-level is
The autoregressive parameter is
```

Additional components include the optimisation result (optim_result). A table of the output of the hyper_optim() function and their description can be found in Table A.2 in Appendix A.

Arguments of hyper_optim_three()

The function hyper_optim_three() works in a similar way as the hyper_optim() function. The only difference is the addition of the training points and the evaluations of the third level of approximation given by train_points_mid and l_train_mid respectively.

A table of the arguments of the hyper_optim_three() function and their description can be found in Table A.3 in Appendix A. The function arguments with the asterisk are mandatory for the function.

Output of hyper_optim_three()

Similarly with the hyper_optim() function, hyper_optim_three() returns a list containing the estimated hyperparameters (optim_hypers) of each level:

```
The signal variance parameter of the low-level is
The length-scale parameter of the low-level is
The signal variance parameter of the middle-level is
The length-scale parameter of the middle-level is
The signal variance parameter of the high-level is
The length-scale parameter of the high-level is
The autoregressive parameter between the low and middle level is
The autoregressive parameter between the middle and high level is
```

Additional components include the optimisation result (optim_result). A table of the output of the hyper_optim_three() function and their description can be found in Table A.4 in Appendix A.

It is not compulsory for the remaining functions of the package to use the hyperparameter estimation functions presented in this section to estimate the hyperparameters. The user can estimate the hyperparameters of the GP using their own optimisation method. However, when a hela function has as argument the estimated hyperparameters the user needs to enter them in a vector form in the same format and order like the output optim_hypers of the hyper_optim() or the hyper_optim_three() functions. For example, for a two-level approximation the hyperparameters need to be in a vector of the form

```
estimated_hypers <- c(sigma_sq_1, l_1, sigma_sq_2, l_2, r_1)
# sigma_sq_1 is the signal variance parameter of the low-level
# l_1 is the length-scale parameter of low-level
# sigma_sq_2 is the signal variance parameter of the high-level
# l_2 is the length-scale parameter of high-level
# r_1 is the autoregressive parameter
```

6.3 Multi-level Gaussian process approximation in hela

Usage of gp_post()

The Gaussian process posterior distribution computed using the multi-level Gaussian process presented in Section 3.3 can be applied using the gp_post() function for two and three levels of approximation. The function has necessary and optional arguments. The user needs to define the

- training points of each level
- evaluations of each approximation level at the training points
- test points the user wants to estimate the intractable function at
- estimated hyperparameters as a result of hyper_optim() or hyper_optim_three() or any hyperparameter values the user estimated
- prior mean structure
- how many approximations levels are used

The usage of the gp_post() is given by

```
gp_post(train_points_low, train_points_high, l_train_low, l_train_high,
        test_points, hypers, prior_mean_str, approx_levels,
        train_points_mid = NULL, l_train_mid = NULL)
```

The explanation of the arguments of the function can be found below and in the documentation file of the function by typing in the R console the code

```
?gp_post
```

Arguments of gp_post()

The gp_post() function has as its arguments the training points of each level, train_points_low, train_points_mid, if the user computes a three-level approximation, default value is NULL, and train_points_high. Same holds for the evaluations of the approximation at the training points of each level l_train_low, l_train_mid and l_train_high.

The user needs to specify the points that they want to predict the intractable function at, test_points. Moreover, the function has as an argument the estimated hyperparameters resulting from the output of the hyper_optim() or hyper_optim_three() functions based on how many levels have been used. As previously mentioned, as an alternative, the user can enter their own estimated hyperparameters but they have to be in a specific format, like the output of hyper_optim().

The prior mean structure of the Gaussian process needs to be specified as a function of the input points using prior_mean_str. The package has three available options: linear prior mean, linear_prior, constant, const_prior, or quadratic, quad_prior, structures, or the user can insert their own prior mean function as a function of the input points. If the user used the hela functions for hyperparameter optimisation they need to make sure that they use the same prior mean structure for the two functions. Moreover, the approx_levels argument is a constant indicating how many levels of approximation are used.

A table of the arguments of the gp_post() function and their description can be found in Table A.5 in Appendix A. The function arguments with the asterisk are mandatory for the function.

Output of `gp_post()`

The function `gp_post()` returns a list containing the posterior mean (`gp_mean`) and posterior covariance (`gp_cov`) of the Gaussian process posterior distribution. The resulting posterior mean can be considered as the approximation of the high-level approximation of the intractable function and the posterior covariance as a measure of uncertainty. A table of the output of the `gp_post()` function and their description can be found in Table A.6 in Appendix A.

6.4 Multi-level Bayesian optimisation using expected gain in utility in `hela`

Usage of `egu()`

The expected gain in utility method for choosing the experimental design of each level of approximation presented in Chapter 5 can be applied using the `egu()` function for a two-level approximation and the `egu_three()` for a three-level approximation. The function has only necessary arguments. The user needs to specify the

- GP posterior mean and covariance at the initial design
- estimated hyperparameters from the initial design
- initial values of the hyperparameters for the estimation
- minimum tolerance value of the EGU as stopping criteria
- maximum number of the training points to be added to the design as stopping criteria
- candidate points
- approximation functions of each level having as their input the training points
- initial training points of each level
- evaluations of each approximation level at the initial training points
- test points
- computational cost of each level
- prior mean structure

The usage of the egu() is given by

```
egu(result_gp, hypers, hyper_initial, egu_min, max_points,
    cand_points, log_lik_high, log_lik_low,
    train_points_low, train_points_high, l_train_low, l_train_high,
    test_points, cost_h, cost_l, prior_mean_str)
```

The explanation of the arguments of the function can be found below and in the documentation file of the function by typing in the R console the code

```
?egu
```

Arguments of egu()

The egu() function has as its first argument a list containing the GP posterior mean vector and the posterior covariance matrix of the initial design, result_gp, which can be obtained using the gp_post function. We treat the posterior mean as the quantity we want to maximise through the EGU method. Also, the function asks for the estimated hyperparameters at the initial design resulting from the output of the hyper_optim() or hyper_optim_three() functions. Moreover, the user needs to specify the initial values of the hyperparameters for the estimation, hyper_initial.

There are are two quantities that act as stopping criteria of the EGU: a minimum tolerance value of EGU that the user is satisfied with, egu_min, and the maximum number of additional training points, max_points, to be added to the design. If one of the two stopping criteria is met then the EGU stops. The user needs to specify a set of candidate points that they want to include in the design, cand_points, and the approximation functions of each level as a function of the training points.

Moreover, the egu() function requires the initial training points of each level, train_points_low and train_points_high, the evaluations of the approximation of each level at the initial design, l_train_low and l_train_high and the computational cost of each level, cost_l and cost_h for the low-level and high-level respectively.

As before, the prior mean structure of the Gaussian process needs to be specified as a function of the input points using prior_mean_str. The user can use the package's prior mean structures or insert their own prior mean function as a function of the input points.

A table of the arguments of the egu() function and their description can be found in Table A.7 in Appendix A. The function arguments with the asterisk are mandatory for the function.

Output of egu()

The function egu() returns a list containing the updated training points of each level:

```
The training points of the low-level are
The training points of the high-level are
```

Some of the additional components of the output of egu() include the GP posterior mean and covariance, the estimated hyperparameters after each addition of a new point, the EGU of each level of approximation, the new points added to the design and at which level. This information is also useful for plotting purposes to produce the EGU plots given in Chapter 5. A table of all of the outputs of the egu() function and their description can be found in Table A.8 in Appendix A.

Arguments and output of egu_three()

The egu_three() function works in a similar way with the egu() function with the only difference that we now use one additional level of approximation. Tables of the arguments and outputs with their description of the egu_three() function can be found in Table A.9 and A.10 in Appendix A.

6.5 Plotting in hela

Usage of gp_posterior_plot()

The gp_posterior_plot() can be used to produce plots of the GP posterior distribution including the posterior mean, which we consider as the approximation of the high-level approximation of the log-likelihood and the posterior covariance as a measure of uncertainty in the form of credible intervals. The plot also includes the training points used for each level and the true or accurate approximation to the log-likelihood if this is available. The plots of the GP posterior in this thesis were produced using the gp_posterior_plot() function.

The function has necessary and optional arguments. The user needs to give the training points of each level, the function evaluations of each level at the training points, the test points used for prediction, the posterior mean vector and posterior covariance matrix as a result of using the gp_post() function. The usage of gp_posterior_plot() is given by

```
gp_posterior_plot(train_points_low, train_points_high, test_points,
                  l_train_low, l_train_high, gp_mean, gpcov,
                  accurate = NULL, train_points_mid  = NULL,
                  l_train_mid = NULL)
```

The explanation of the arguments of the function can be found below and in the documentation file of the functions by typing in the R console the code

```
?gp_posterior_plot
```

Arguments of gp_posterior_plot()

The gp_posterior_plot() function has as its arguments the training points of each level, train_points_low and train_points_high, the evaluations of the approximation of each level at the training points, l_train_low and l_train_high, of the low and high level respectively for the two-level approximation case. For the case where the user is using a three-level approximation the function has as optional arguments the training points, train_points_mid, and the log-likelihood approximation evaluations of the additional level training points, l_train_mid, whose default value is NULL.

Moreover, the function requires the test points used for the prediction, test_points, the posterior mean, gp_mean, and posterior covariance, gp_cov, as a result of the gp_post() function. The user can also inputs the true or accurate approximation of the high-level, accurate, if that is available, to plot for comparisons, the default value is NULL.

A table of the arguments of the gp_posterior_plot() function and their description can be found in Table A.11 in Appendix A. The function arguments with the asterisk are mandatory for the function.

Output of gp_posterior_plot()

The function gp_posterior_plot returns a plot of the Gaussian process posterior which includes the posterior mean (red curve), 95% interval estimation which is calculated based on the posterior mean and posterior covariance as a measure of uncertainty (blue shaded area) and the training points of each level (green, pink and blue dots for low, middle and high level respectively). Moreover, the function gives the option to plot an accurate approximation (orange dashed line) for comparisons. Examples can be found in previous chapters of the thesis and in Section 6.6.

6.6 Applications using hela package

6.6.1 Overview

In this section, the use of the hela package is demonstrated through the GLMM example. In general, the user can use any model and up to three levels of approximation of the intractable likelihood they wish to compute an approximation for the high-level of approximation using multi-level GP approximation.

6.6.2 Generalised linear mixed model example

We focus on log-likelihood approximations and we using as our example the GLMM example presented in Section 3.4.3 for a two-level GP approximation. For examples that use three-level approximation the user can follow the same procedure as the two-level case but using the respective functions for the three levels as described in this chapter. The examples presented in previous chapters of this thesis can be reproduce using the functions in the hela package.

The data for the GLMM example are taken from the glmmsr package (Ogden, 2019). The glmmsr package is not currently available on CRAN and can be obtained from GitHub using the R code

```
devtools::install_github("heogden/glmmsr")
library("glmmsr")
```

For the GLMM model we use three training points of the low-level and three training points of the high-level as in Example 3.11. The training points were generated using a LHD from the randomLHS function of the lhs package (Carnell, 2022). The low-level approximation of the log-likelihood was calculated using the LA method and the high-level using the AGQ approximation method with nAGQ = 10. The arguments for the GLMM example are

```
train_points_low <- c(2.996337891, 0.001464844, 0.963134766)
train_points_high <- c(2.998535156, 0.002929688, 1.378417969)
test_points <- seq(0, 3, length.out = 100)
l_train_low <- c(-72.67779, -66.61996, -66.01664)
l_train_high <- c(-70.14021, -66.61994, -65.82183)
```

More specifically, the evaluations of the log-likelihood approximations, l_train_low and l_train_high, are computed using the code

```
modfr <- find_modfr_glmm(response ~ covariate + (1 | cluster),
                         data = two_level, family = binomial)
m_est <- c(0.7167604, -1.2733840)
# high-level
lik_function_high <- find_lfun_glmm(modfr, method = "AGQ", list(nAGQ=10))
train_high_inputs <- cbind(train_points_high, matrix(rep(m_est,
                         each = length(train_points_high)),
                         nrow = length(train_points_high)))
l_train_high <- apply(train_high_inputs, 1, lik_function_high)
# low-level
lik_function_low <- find_lfun_glmm(modfr, method = "Laplace", NULL)
train_low_inputs <- cbind(train_points_low, matrix(rep(m_est,
                         each = length(train_points_low)),
                         nrow = length(train_points_low)))
l_train_low <- apply(train_low_inputs, 1, lik_function_low)
```

We consider the log-likelihood of the GLMM as a function of the standard deviation of the random effects, we fix the remaining of the model parameters and we set them equal to their estimated values after fitting the model, `m_est`. For more information on how to use the `find_modfr_glmm()` and `find_lfun_glmm()` functions please refer to the vignette of the `glmmsr` package (Ogden, 2019).

6.6.3 Hyperparameter estimation

The hyperparameter estimation for the two-level approximation is done using the `hyper_optim()` function. The initial values of the hyperparameters and the prior mean structure are given by

```
hypers <- rep(0.5, 5)
prior_mean_str <- quad_prior
```

where `quad_prior` is a function giving the quadratic prior mean structure. Then, the estimated hyperparameters can be calculated using

```
hyperparameters <- hyper_optim(hypers = hypers,
                               train_points_low = train_points_low,
                               train_points_high = train_points_high,
                               l_train_low = l_train_low,
                               l_train_high = l_train_high,
                               prior_mean_str = quad_prior)
```

The outcome hyper_optim() gives the estimated hyperparameters:

```
"The signal variance parameter of the low-level is 0"
"The length-scale parameter of the low-level is 0.73278"
"The signal variance parameter of the high-level is 0.53368"
"The length-scale parameter of the high-level is 0.69386"
"The autoregressive parameter is 0.59871"
```

or it can be accessed using the code

```
hyperparameters$optim_hypers
```

and the outcome of this line of code is a vector with entries

```
$optim_hypers
 ssquare1           l1     ssquare2           l2           r1
4.112279e-26 7.327787e-01 5.336828e-01 6.938563e-01 5.987097e-01
```

where ssquare1 and ssquare2 are the signal variance parameters and l1 and l2 are the length-scale parameters of the low and high levels respectively and r1 gives the autoregressive parameter between the low and high level. If the user wishes to use their own method for estimating hyperparameters they need to have them in this form to be used as an argument of the other functions.

6.6.4 Two-level approximation

Using the optimised hyperparameter values obtained from the hyper_optim() function, or any other hyperparameter values that the user calculated independently, we can approximate the high-level of approximation by combining evaluations from the high and low levels using the gp_post() function.

```
result_gp <- gp_post(train_points_low = train_points_low,
                     train_points_high = train_points_high,
                     l_train_low = l_train_low,
                     l_train_high = l_train_high,
                     test_points = test_points,
                     hypers = hyperparameters$optim_hypers,
                     prior_mean_str = prior_mean_str,
                     approx_levels = 2,
                     train_points_mid = NULL, l_train_mid = NULL)
```

Since we work with a two-level approximation `train_points_mid` and `l_train_mid` are kept to their default `NULL` values. The output of the `gp_post()` function can be extracted using the code

```
result_gp$gp_mean
result_gp$gp_cov
```

For this example, an accurate approximation of the log-likelihood is available, `accurate_high`, calculated using the code

```
test_inputs <- cbind(test_points,
                matrix(rep(m_est,each = length(test_points)),
                nrow = length(test_points)))
accurate_high <- apply(test_inputs, 1, lik_function_high)
```

The outcome of the `gp_post()` function can be plotted using the `gp_posterior_plot()` function, that is

```
gp_posterior_plot(train_points_low = train_points_low,
                train_points_high = train_points_high,
                test_points = test_points,
                l_train_low = l_train_low,
                l_train_high = l_train_high,
                gp_mean = result_gp$gp_mean,
                gp_cov = result_gp$gp_cov,
                accurate = accurate_high,
                train_points_mid  = NULL, l_train_mid = NULL)
```

The resulting plot can be seen in Figure 6.1. The red curve is the posterior mean of the Gaussian process which can be used as the approximation to the high-level, the orange dashed curve is the accurate approximation of the log-likelihood which is available in the example (added for comparison, the user has the option not to plot this in case that this is not available), the blue shaded area is the 95% credible interval and the blue and green dots are the training data for high and low level respectively.

6.6.5 Expected gain in utility

The expected gain in utility method for multi-level Bayesian optimisation has been presented in Chapter 5. We continue with the same example from the multi-level GP approximation and we consider as our initial design for the EGU the training points

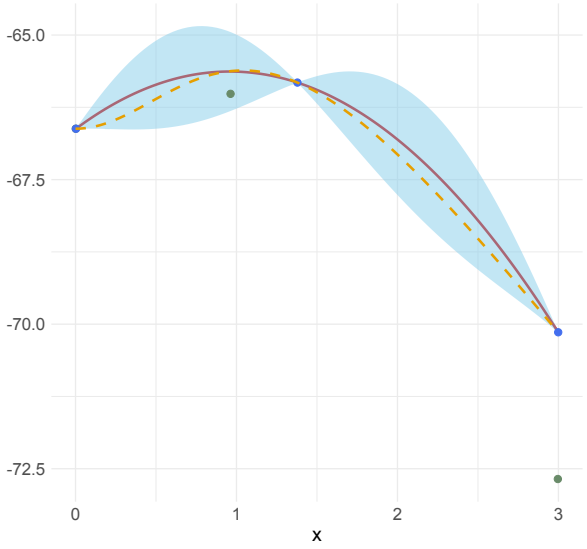


FIGURE 6.1: Posterior mean (red curve) as a two-level GP approximation to the log-likelihood for the GLMM example along with the accurate AGQ approximation of the log-likelihood (orange).

used in Section 6.6.2. The aim is to add new points to the existing design at each level which will benefit the prediction the most. This can be achieved using the `egu()` function.

The minimum tolerance value of the EGU is set to `1e-1` and we want up to three new points added to the design. The candidate points are generated from an equally spaced sequence of numbers covering the interval of interest. For the LA and AGQ approximation methods there is no significant difference in computational cost, therefore, we set an arbitrary cost to each level of approximation assigning the higher cost to the AGQ method.

More specifically, the arguments of the function are given by

```
cost_h <- 2.1; cost_l <- 2
hyper_initial <- rep(0.5, 5)
cand_points <- seq(0, 3, length.out = 20)

# approximation function of the high-level
# as a function of the training points
log_lik_high <- function(train_d2){
    m_est <- c(0.7167604, -1.2733840)
    train_high_inputs <- cbind(train_d2, matrix(rep(m_est,
                         each=length(train_d2)), nrow=length(train_d2)))
    l_train_exp <- apply(train_high_inputs, 1, lik_function_high)
  return(l_train_exp)
```

```
}
# approximation function of the low-level
# as a function of the training points
log_lik_low <- function(train_d1){
    m_est <- c(0.7167604, -1.2733840)
    train_low_inputs <- cbind(train_d1, matrix(rep(m_est,
                        each=length(train_d1)), nrow=length(train_d1)))
    l_train_ch <- apply(train_low_inputs, 1, lik_function_low)
    return(l_train_ch)
}
```

The egu() function can be used with the code

```
EGU_results <- egu(result_gp = result_gp,
                hypers = hyperparameters$optim_hypers,
                hyper_initial = hyper_initial, egu_min = 1e-1,
                max_points = 3, cand_points = cand_points,
                log_lik_high = log_lik_high, log_lik_low = log_lik_low,
                train_points_low = train_points_low,
                train_points_high = train_points_high,
                l_train_low = l_train_low,
                l_train_high = l_train_high,
                cost_h = cost_h, cost_l = cost_l,
                test_points = test_points,
                prior_mean_str = prior_mean_str)
```

While the egu() function is running the user gets an update how many new points have been added to the design and at which candidate point the egu() is at. The outcome of egu() gives the training points of each level:

```
"The training points of the low-level are"
2.996 0.001 0.963
"The training points of the high-level are"
2.999 0.003 1.378 0.947
```

The remaining of the outputs of the egu() can be extracted using the code

```
# hyperparameters of each level for each new point
EGU_results$ssquare1; EGU_results$l1;
EGU_results$ssquare2; EGU_results$l2; EGU_results$r1
# EGU per cost of each level of approximation for each new point
```

```
EGU_results$egu_high; EGU_results$egu_low
# updated training points of each level
EGU_results$train_points_low; EGU_results$train_points_high
# evaluations of the log-likelihood
EGU_results$l_train_low; EGU_results$l_train_high
# posterior mean and posterior covariance after each new point
EGU_results$post_covar; EGU_results$post_mean
# new points added to the design
EGU_results$new_point
```

Using these results we can produce a plot of the estimated hyperparameters after the addition of each new point as shown in Figures 5.14 and 5.18. Moreover, we can plot the EGU per cost after each iteration to assess the decision taken as shown in Figure 5.13. We favour the approximation level which has the greater EGU per cost since adding that point to the design will benefit the prediction more.

6.7 Summary

This chapter presented the `R` package `hela` for obtaining multi-level GP approximations based on a hierarchy of approximation levels. The package `hela` offers a statistical method for statistical design, modelling and inference using systems and approximations available on at least two hierarchical scales. Moreover, it provides a method for choosing the experimental design for experiments with multiple levels.

An example using GLMM was used to illustrate how the package can be used for a two-level GP approximation. The same procedures can be followed for a three-level GP approximation using the corresponding functions. We will continue to update and improve the `hela` package to incorporate a general multi-level GP approximation of intractable functions and to expand the EGU methodology.

Chapter 7

Discussion

7.1 Thesis synopsis

In this thesis, we developed statistical methods for statistical design, modelling and inference using systems and approximations available on multiple hierarchical scales. We also used Bayesian optimisation to choose the experimental design based on expected gain in utility and the cost of each level of approximation.

Chapter 2 gave a brief introduction of the preliminary material we used in the thesis. We presented an overview of generalised linear mixed models and Ising models.

In Chapter 3 we introduced the idea of hierarchical experiments and likelihood approximations. More specifically, we gave an overview of Gaussian process, Gaussian process regression and presented the methodology of how this could be combined with hierarchical experiments. As the running examples of the thesis, we considered a model where the likelihood is given in closed form, the simple linear regression model, in order to be able to demonstrate how the methodology works, and cases where the likelihood function of a model is intractable or computationally expensive, such as the likelihood of a generalised linear mixed model and Ising model.

Moreover, we demonstrated how hierarchical experiments and Gaussian process can be used to approximate a likelihood function. Considering the SLR model we did not have to use a likelihood approximation to fit the Gaussian process, instead we used some evaluations of the known likelihood and we worked with the single-level Gaussian process regression. For the GLMM example we used the multi-level GP approximation approach to obtain an approximation of the likelihood and subsequently to the maximum likelihood estimate by combining information from two levels of approximation using Gaussian process regression. We used the Laplace approximation as the lower-level and the adaptive Gaussian quadrature with ten quadrature points as the higher level. Using the same approach for the Ising model, we used the reduced-dependence

approximation method to approximate the normalising constant of the likelihood for different levels of approximation.

The GP multi-level approximation can also be expanded for multi-dimensions. We presented an example using GLMM where the log-likelihood function was considered as a function of two model parameters and we obtained the posterior predictive distribution using multi-level GP approximation.

As a general result, the posterior mean of the Gaussian process posterior distribution can be considered as a good approximation to a high-level approximation of an intractable or computationally costly likelihood using observations from multiple levels of approximation without having to use a lot of evaluations of the higher level of approximation.

In Chapter 4 we focused on conducting inference of the model parameters and dealing with different sources of uncertainty. We draw likelihood samples using the corresponding multivariate Gaussian distribution from the Gaussian process posterior distribution used as an approximation to the likelihood, and we obtained samples form the resulting posterior for the model parameter of interest by multiplying the resulting likelihood with a predefined prior. Hence, we were able to conduct inference for the model parameter and combine the underlying uncertainty about the model parameters given the data with the uncertainty from using the multi-level Gaussian process model as an approximation of the likelihood rather than the true likelihood itself.

In Chapter 5 we introduced expected gain in utility which is a Bayesian optimisation approach to choose the experimental design. We focused on the Gaussian process emulator of the function of interest, which in our case is the log-likelihood function with the aim to find the point maximising the log-likelihood. The reporting method, utility, we use was such that we choose the new point that maximises the posterior mean of the function of interest. The challenging part was to apply Bayesian optimisation to choose the experimental design for multi-level approximations of the likelihood, each with different cost. We expanded EGU to consider multi-level approximations and we added the point at the level which gives the largest EGU relative to the cost of each level.

Chapter 6 includes the vignette of our `R` package, `hela`. It includes the statistical methods presented in this thesis for statistical design, modelling and inference of the model parameters, choosing the experimental design using EGU using systems and approximations available on multiple hierarchical scales and plotting the GP approximation.

The development of new statistical methodology such as the one described in this thesis is important for the application of models with intractable likelihood functions. Our goal is to provide adequate inference for the model parameters, to combine various sources of uncertainty and to choose the experimental design by finding an optimal

balance between time and cost. There is also a lot of scope for future work in extending the work of this thesis.

7.2 Future work

7.2.1 Expected gain in utility

We believe EGU has potential to improve and generate high quality designs for hierarchical experiments across application areas, beyond the likelihood examples we have considered so far.

For this thesis, we have used the EGU for one-dimensional problems for single and multi-level Bayesian optimisation. It will be challenging to choose the experimental design in the higher dimensional parameter space case since the number of training points increase exponentially with the number of dimensions. Therefore, it is important to choose the training points wisely since we will not be able to have a fine full grid of training points at a low cost. Hence, it is vital to extend EGU for higher dimensional cases as well.

When we considered extending the GP regression to multi-dimensional there was a need for choosing the experimental region of each model parameter. Future work on the EGU could focus on how the area of experimentation can be expanded possibly using a response surface methodology (Box and Wilson, 1951).

The main application of the EGU for this thesis was to find the optimal design aiming to find a good approximation for the maximum likelihood estimate. The evaluation of the approximation mainly considers the point estimates and not the associated uncertainty. We would like to research on how to apply the same technique when the main focus is to accurately approximate the associated uncertainty at minimum cost and how the algorithms can be modified.

In the literature, there are other approaches as well for multi-level Bayesian optimisation such as two-stage learning method by Alexander et al. (2007), the MF-MI-Greedy method by Song et al. (2018) and the GP-UCB method by Kandasamy et al. (2017) which were described in the thesis. It would be useful to compare these methods with our EGU for multi-level BO method based on efficiency, accuracy and cost.

7.2.2 Assessing the accuracy of Gaussian process approximation

For the examples we examined in this thesis, we were able to have an accurate approximation of the likelihood function we were interested in. Using the accurate approximation we could assess the accuracy of the GP approximation graphically. However,

that is not applicable for most cases. Therefore, future work could also check the accuracy of the Gaussian process approximation and the hierarchical experiments method. To assess how accurately the Gaussian process approximation model will perform in practice we could use cross-validation methods (Hastie et al., 2009).

7.2.3 Working in multi-dimensions

When working in a multi-dimensional parameter space it is more challenging to choose a kernel of the Gaussian process. In general, kernels can be combined to compute new ones with different properties allowing the inclusion of high-level structure into the models. To model functions with more that one input it is useful to multiply kernels defined on each individual part (Duvenaud, 2014). The SE-ARD kernel is the result of the product of kernels over different dimensions with different length-scale parameters. It is usually used for most applications of Gaussian processes because it is simple and easily interpretable.

For this thesis, we have used a single length-scale parameter for all of the dimensions used. Therefore, as part of the future work, we would like to introduce and estimate different length-scale parameters for each dimension to distinguish between the relevance of each dimension. However, this becomes more complicated when working with multi-level GPs since for each level of approximation we will have to estimate multiple length-scale hyperparameters.

7.2.4 `R` package

Throughout the duration of the thesis, we created the `R` package, `hela` for developing the statistical methods presented in this thesis. We used the package for the purposes of the thesis' calculations and we will continue to improve the package to make it more general for multi-level GP approximations, include multi-dimensions, update it and integrate further elements.

7.2.5 Hierarchical experiments and optimal designs

In the future, it will be interesting to explore a way of integrating the hierarchical experiments into optimal design experiments. Working with optimal experiment design, the Fisher information is used to determine and optimise the design for maximising the expected accuracy of the model parameter estimates (Kreutz and Timmer, 2013).

The Fisher information can be considered as a measure of the amount of information about the parameters given the experimental data and is directly related to the accuracy

of the estimated parameters. The calculation of the Fisher information is based on the second order derivatives and it is only valid if a quadratic approximation of the log-likelihood exists. However, for models where the likelihood is intractable the Fisher information cannot be obtained analytically. Hence, using the methods and ideas presented in this thesis we would like to find optimal designs using the Fisher information. We can obtain all the derivatives analytically for a Gaussian process (Stephenson, 2010). Having the second derivatives available will be useful for the design problem. In this case the choice of design points will be different from what we presented so far since we will be interested in the behaviour of the log-likelihood around the maximum, not just in the location of the maximum.

7.2.6 Application to other examples

There are many areas of statistical research that require to understand and utilise hierarchical differences in the accuracy and cost of systems and approximations across different scales. This thesis, was mainly focused on likelihood approximations using hierarchical experiments.

The methodology developed for multi-level GP approximation and EGU for a multi-level BO can be expanded to be used for other examples. An example problem is the design of experiments across lab, pilot and manufacturing scales to understand and predict manufactured product performance in the pharmaceutical industry. The hierarchical levels for this example could be the scales of physical experiments, for instance, the lab scale could be the low-level and the physical scale could be the high-level, and multi-level GP approximations could be used to approximate the output of the production scale combining evaluations from both scales.

Appendix A

Documentation of the functions in `hela`

A.1 Arguments and output of the `hyper_optim()` function

Argument	Description
`hypers*`	A vector containing the initial values of the hyperparameters, used for optimisation, of the Gaussian process of each level: signal variance, length scale and autoregressive parameters.
`train_points_low*`	A vector containing the training points of the low-level.
`train_points_high*`	A vector containing the training points of the high-level.
`l_train_low*`	A vector containing the low-level function approximation at the low-level training points.
`l_train_high*`	A vector containing the low-level function approximation at the high-level training points.
`prior_mean_str*`	A function giving the structure of the prior mean.

TABLE A.1: Arguments of the function `hyper_optim()` with their description. The mandatory arguments are denoted with *.

Output	Description
optim_hypers	A vector containing the estimated hyperparameters: signal-variance parameter, length-scale parameter of each level and autoregressive parameter.
optim_result	The result of the hyperparameter optimisation using the Nelder-Mead method.

TABLE A.2: Output of the function hyper_optim() with a description.

A.2 Arguments and output of the hyper_optim_three() function

Argument	Description
hypers*	A vector containing the initial values of the hyperparameters, used for optimisation, of the Gaussian process of each level: signal variance, length scale and autoregressive parameters.
train_points_low*	A vector containing the training points of the low-level.
train_points_mid*	A vector containing the training points of the middle-level.
train_points_high*	A vector containing the training points of the high-level.
l_train_low*	A vector containing the low-level function approximation at the low-level training points.
l_train_mid*	A vector containing the low-level function approximation at the middle-level training points.
l_train_high*	A vector containing the low-level function approximation at the high-level training points.
prior_mean_str*	A function giving the structure of the prior mean.

TABLE A.3: Arguments of the function hyper_optim_three() with their description. The mandatory arguments are denoted with *.

Output	Description
optim_hypers	A vector containing the estimated hyperparameters: signal-variance parameter, length-scale parameter of each level and autoregressive parameters.
optim_result	The result of the hyperparameter optimisation using the Nelder-Mead method.

TABLE A.4: Output of the function hyper_optim_three() with a description.

A.3 Arguments and outputs of the gp_post() function

Argument	Description
train_points_low*	A vector containing the training points of the low-level.
train_points_high*	A vector containing the training points of the high-level.
l_train_low*	A vector containing the low-level function approximation at the low-level training points.
l_train_high*	A vector containing the low-level function approximation at the high-level training points.
test_points*	A vector containing the test points used for prediction.
hypers*	A vector containing the hyperparameters of the Gaussian process of each level: signal-variance, length scale and autoregressive parameters as a result of the hyper_optim() function.
prior_mean_str*	A function giving the structure of the prior mean.
approx_levels*	Constant indicating how many levels of approximation to use. Choose between 2 or 3.
train_points_mid	A vector containing the training points of the middle-level for a three-level approximation. Default value is NULL.
l_train_mid	A vector containing the low-level function approximation at the middle-level training points for a three-level approximation. Default value is NULL.

TABLE A.5: Arguments of the function gp_post() with their description. The mandatory arguments are denoted with *.

Output	Description
gp_mean	A vector containing Gaussian process posterior mean.
gp_cov	A matrix containing Gaussian process posterior covariance.

TABLE A.6: Output of the function gp_post() with a description.

A.4 Arguments and outputs of the egu() function

Argument	Description
result_gp*	A list containing the GP posterior mean and posterior covariance as a result of the gp_post() function.
hypers*	A vector containing the hyperparameters of the Gaussian process of each level: signal-variance, length scale and autoregressive parameters as a result of the hyper_optim() function.
hyper_initial*	A vector containing the initial values of the hyperparameters for estimation.
egu_min*	Expected gain in utility minimum tolerance as a stopping rule.
max_points*	Maximum points to be added to the experimental design as a stopping rule.
cand_points*	A vector containing the candidate points.
log_lik_high*	Function of the high-level points giving the high-level approximation.
log_lik_low*	Function of the low-level points giving the low-level approximation.
train_points_low*	A vector containing the training points of the low-level.
train_points_high*	A vector containing the training points of the high-level.
l_train_low*	A vector containing the low-level function approximation at the low-level training points.
l_train_high*	A vector containing the high-level function approximation at the high-level training points.
test_points	A vector containing the test points used for predictions.
cost_h*	Constant indicating the cost of the high-level.
cost_l*	Constant indicating the cost of the low-level.
prior_mean_str*	A function giving the structure of the prior mean.

TABLE A.7: Arguments of the function egu() with their description. The mandatory arguments are denoted with *.

Output	Description
ssquare1	List with the estimated signal variance hyperparameter of the low-level for each iteration of the EGU.
ssquare2	List with the estimated signal variance hyperparameter of the high-level for each iteration of the EGU.
l1	List with the estimated length-scale hyperparameter of the low-level for each iteration of the EGU.
l2	List with the estimated length-scale hyperparameter of the high-level for each iteration of the EGU.
r1	List with the estimated autoregressive parameter for each iteration of the EGU.
train_points_low	List containing the training points of the low-level for each iteration of the EGU.
train_points_high	List containing the training points of the high-level for each iteration of the EGU.
l_train_low	List containing the function evaluations at the low-level training points for each iteration of the EGU.
l_train_high	List containing the function evaluations at the high-level training points for each iteration of the EGU.
high_p	A logical vector indicating if the new point was added to the high-level.
low_p	A logical vector indicating if the new point was added to the low-level.
egu_high	List containing the EGU of the candidate points when added to the high-level for each iteration of the EGU.
egu_low	List containing the EGU of the candidate points when added to the low-level for each iteration of the EGU.
post_mean	List containing the GP posterior mean vector for each iteration of the EGU.
post_cov	List containing the GP posterior covariance matrix for each iteration of the EGU.
total_cost	Constant indicating the total cost.
points_n	Constant indicating the number of new points added to the design.
new_point	Vector containing the new points added to the design.
new_lik	Vector containing the evaluations of the new training points.

TABLE A.8: Output of the function egu() with a description.

A.5 Arguments and output of the egu_three() function

Argument	Description
result_gp*	A list containing the GP posterior mean and posterior covariance as a result of the gp_post() function.
hypers*	A vector containing the hyperparameters of the Gaussian process of each level: signal-variance, length scale and autoregressive parameters as a result of the hyper_optim() function.
hyper_initial*	A vector containing the initial values of the hyperparameters for estimation.
egu_min*	Expected gain in utility minimum tolerance as a stopping rule.
max_points*	Maximum points to be added to the experimental design as a stopping rule.
cand_points*	A vector containing the candidate points.
log_lik_high*	Function of the high-level points giving the high-level approximation.
log_lik_mid*	Function of the middle-level points giving the middle-level approximation.
log_lik_low*	Function of the low-level points giving the low-level approximation.
train_points_low*	A vector containing the training points of the low-level.
train_points_mid*	A vector containing the training points of the middle-level.
train_points_high*	A vector containing the training points of the high-level.
l_train_low*	A vector containing the low-level function approximation at the low-level training points.
l_train_mid*	A vector containing the middle-level function approximation at the middle-level training points.
l_train_high*	A vector containing the high-level function approximation at the high-level training points.
test_points	A vector containing the test points used for predictions.
cost_h*	Constant indicating the cost of the high-level.
cost_m*	Constant indicating the cost of the middle-level.
cost_l*	Constant indicating the cost of the low-level.
prior_mean_str*	A function giving the structure of the prior mean.

TABLE A.9: Arguments of the function egu_three() with their description. The mandatory arguments are denoted with *.

The output of egu_three() function is similar with the egu() function with the differences and additional outputs shown in Table A.10.

Output	Description
ssquare2	List with the estimated signal variance hyperparameter of the middle-level for each iteration of the EGU.
ssquare3	List with the estimated signal variance hyperparameter of the high-level for each iteration of the EGU.
l2	List with the estimated length-scale hyperparameter of the middle-level for each iteration of the EGU.
l3	List with the estimated length-scale hyperparameter of the high-level for each iteration of the EGU.
r1	List with the estimated autoregressive parameter between the low and middle levels for each iteration of the EGU.
r2	List with the estimated autoregressive parameter between the middle and high levels for each iteration of the EGU.
train_points_mid	List containing the training points of the middle-level for each iteration of the EGU.
l_train_mid	List containing the function evaluations at the middle-level training points for each iteration of the EGU.
mid_p	A logical vector indicating if the new point was added to the middle-level.
egu_mid	List containing the EGU of the candidate points when added to the middle-level for each iteration of the EGU.

TABLE A.10: Additional output of the function egu_three() with a description.

A.6 Arguments of the gp_posterior_plot() function

Argument	Description
train_points_low*	A vector containing the training points of the low-level.
train_points_high*	A vector containing the training points of the high-level.
test_points*	A vector containing the test points used for prediction.
l_train_low*	A vector containing the low-level function approximation at the low-level training points.
l_train_high*	A vector containing the low-level function approximation at the high-level training points.
gp_mean*	A vector containing Gaussian process posterior mean as a result of the gp_post() function.
gp_cov*	A matrix containing Gaussian process posterior covariance as a result of the gp_post() function.
accurate*	A vector containing the evaluations of the true or accurate approximation if available. Default value is NULL.
train_points_mid	A vector containing the training points of the middle-level for a three-level approximation. Default value is NULL.
l_train_mid	A vector containing the low-level function approximation at the middle-level training points for a three-level approximation. Default value is NULL.

TABLE A.11: Arguments of the function gp_posterior_plot() with their description. The mandatory arguments are denoted with *.

References

Ababou, R., Bagtzoglou, A. C., and Wood, E. F. (1994). On the condition number of covariance matrices in kriging, estimation, and simulation of random fields. *Mathematical Geology*, 26(1):99–133.

Abramowitz, M. and Stegun, I. (1965). *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. Applied mathematics series. Dover Publications.

Alexander, I. F., Sóbester, A., and Keane, A. (2007). Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463(2088):3251–3269.

Andrianakis, I. and Challenor, P. G. (2012). The effect of the nugget on gaussian process emulators of computer models. *Computational Statistics & Data Analysis*, 56(12):4215–4228.

Ba, S. and Joseph, V. R. (2015). *MaxPro: Maximum Projection Designs*. https://CRAN.R-project.org/package=MaxPro.

Bates, D., Mächler, M., Bolker, B., and Walker, S. (2015). Fitting Linear Mixed-Effects Models Using lme4. *Journal of Statistical Software*, 67(1):1–48.

Blum, M. and Riedmiller, M. (2013). Optimization of Gaussian Process Hyperparameters using Rprop. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 339–344.

Box, G. E. P. and Wilson, K. B. (1951). On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 13(1):1–45.

Browne, W. and Rasbash, J. (2004). *Handbook of Data Analysis AU*, chapter Multilevel Modelling, pages 460–479. SAGE Publications, Ltd.

Carnell, R. (2022). *lhs: Latin Hypercube Samples*. R package version 1.1.5.

Damianou, A. C. (2015). *Deep Gaussian processes and variational propagation of uncertainty*. PhD thesis, Department of Neuroscience, University of Sheffield.

Duvenaud, D. (2014). *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge.

Epskamp, S. and Boot, J. (2023). *IsingSampler: Sampling Methods and Distribution Functions for the Ising Model*. https://CRAN.R-project.org/package=IsingSampler.

Finnemann, A., Borsboom, D., Epskamp, S., and van der Maas, H. L. J. (2021). The theoretical and statistical ising model: A practical guide in r. *Psych*, 3(4):593–617.

Frazier, P., Powell, W., and Dayanik, S. (2009). The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, 21(4):599–613.

Frazier, P. I. (2018). A tutorial on bayesian optimization. *ArXiv*, abs/1807.02811.

Friel, N., Pettitt, A. N., Reeves, R., and Wit, E. (2009). Bayesian inference in hidden markov random fields for binary data defined on large lattices. *Journal of Computational and Graphical Statistics*, 18(2):243–261.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2013). *Bayesian Data Analysis*. Chapman and Hall/CRC, 3rd ed. edition.

Geyer, C. J. and Thompson, E. A. (1992). Constrained monte carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 54(3):657–699.

Gramacy, R. B. and Lee, H. K. H. (2010). Optimization under unknown constraints.

Handayani, D., Notodiputro, K. A., Sadik, K., and Kurnia, A. (2017). A comparative study of approximation methods for maximum likelihood estimation in generalized linear mixed models (GLMM). *AIP Conference Proceedings*, 1827(1):020033.

Hartzel, J., Agresti, A., and Caffo, B. (2001). Multinomial logit random effects models. *Statistical Modelling*, 1(2):81–102.

Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, chapter Model Assessment and Selection, pages 219–257. Springer, New York, 2nd edition.

Hennig, P. and Schuler, C. (2011). Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13:1809–1837.

Hernández-Lobato, J. M., Hoffman, M. W., and Ghahramani, Z. (2014). Predictive entropy search for efficient global optimization of black-box functions.

Ising, E. (1925). Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik*, 31(1):253–258.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492.

Kandasamy, K., Dasarathy, G., Schneider, J., and Póczos, B. (2017). Multi-fidelity Bayesian optimisation with continuous approximations. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1799–1808. PMLR.

Kaufman, B. (1949). Crystal Statistics. II. Partition Function Evaluated by Spinor Analysis. *Phys. Rev.*, 76:1232–1243.

Kennedy, M., Anderson, C., Conti, S., and O Hagan, A. (2006). Case studies in Gaussian process modelling of computer codes. *Reliability Engineering & System Safety*, 91.

Kennedy, M. C. and O'Hagan, A. (2000). Predicting the Output from a Complex Computer Code When Fast Approximations Are Available. *Biometrika*, 87(1):1–13.

Kreutz, C. and Timmer, J. (2013). *Optimal Experiment Design, Fisher Information*, pages 1576–1579. Springer New York, New York, NY.

Kushner, H. J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86:97–106.

Le Gratiet, L. (2011). Bayesian analysis of hierarchical multi-fidelity codes. *arXiv e-prints*, page arXiv:1112.5389.

Le Gratiet, L. (2012). Recursive co-kriging model for design of computer experiments with multiple levels of fidelity. *International Journal for Uncertainty Quantification*, 4.

Le Gratiet, L. (2013). *Multi-fidelity Gaussian process regression for computer experiments*. PhD thesis, Université Paris-Diderot-Paris VII.

Lenz, W. (1920). Beitrag zum verständnis der magnetischen erscheinungen in festen körpern. *Z. Phys.*, 21:613–615.

Liu, Q. and Pierce, D. A. (1994). A note on gauss-hermite quadrature. *Biometrika*, 81(3):624–629.

McCulloch, C. and Searle, S. (2004). *Generalized, Linear, and Mixed Models*. Wiley series in probability and statistics: Applied probability and statistics. Wiley.

McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). 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.

Močkus, J. (1975). On bayesian methods for seeking the extremum. In Marchuk, G. I., editor, *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, pages 400–404, Berlin, Heidelberg. Springer Berlin Heidelberg.

Myung, I. (2003). Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, 47(1):90 – 100.

Nelder, J. A. and Mead, R. (1965). A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313.

Oakley, J. and O'Hagan, A. (2002). Bayesian inference for the uncertainty distribution of computer model outputs. *Biometrika*, 89(4):769–784.

Ogden, H. (2019). *glmmsr: Fit a Generalized Linear Mixed Model*.

Ogden, H. E. (2017). On asymptotic validity of naive inference with an approximate likelihood. *Biometrika*, 104(1):153–164.

O'Hagan, A. (1978). Curve Fitting and Optimal Design for Prediction. *Journal of the Royal Statistical Society. Series B (Methodological)*, 40(1):1–42.

Orbanz, P. and Teh, Y. W. (2010). Bayesian Nonparametric Models. In *Encyclopedia of Machine Learning and Data Mining*.

Overstall, A. M. and Woods, D. C. (2013). A strategy for bayesian inference for computationally expensive models with application to the estimation of stem cell properties. *Biometrics*, 69(2):458–68.

Perdikaris, P., Raissi, M., Damianou, A., Lawrence, N. D., and Karniadakis, G. E. (2017). Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2198):20160751.

Pinheiro, J. and Chao, E. (2006). Efficient Laplacian and Adaptive Gaussian Quadrature Algorithms for Multilevel Generalized Linear Mixed Models. *Journal of Computational and Graphical Statistics - J COMPUT GRAPH STAT*, 15:58–81.

Polson, N. G. and Scott, J. G. (2012). On the Half-Cauchy Prior for a Global Scale Parameter. *Bayesian Analysis*, 7(4):887 – 902.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, USA, 3 edition.

Radford, M. N. (1997). Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification. Technical Report 9702, Department of Statistics, University of Toronto.

Rasmussen, C. (2003). Gaussian processes to speed up hybrid monte carlo for expensive bayesian integrals. *BAYESIAN STATISTICS*, 7:651–659.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, Cambridge.

Reeves, R. and Pettitt, A. N. (2004). Efficient recursions for general factorisable models. *Biometrika*, 91(3):751–757.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and Analysis of Computer Experiments. *Statistical Science*, 4(4):409 – 423.

Santner, T. J., Williams, B. J., and Notz, W. I. (2018). *The Design and Analysis of Computer Experiments*. Springer-Verlag New York, 2 edition.

Schulz, E., Speekenbrink, M., and Krause, A. (2018). A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1 – 16.

Song, J., Chen, Y., and Yue, Y. (2018). A general framework for multi-fidelity bayesian optimization with gaussian processes.

Stephenson, G. (2010). *Using derivative information in the statistical analysis of computer models*. PhD thesis, University of Southampton.

Tierney, L. and Kadane, J. B. (1986). Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, 81(393):82–86.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, Fourth edition.

Waite, T. and Woods, D. (2017). Closed-loop automatic experimentation for bayesian optimization of a flow chemistry reaction. Unpublished manuscript.

Wilson, J. T., Hutter, F., and Deisenroth, M. P. (2018). Maximizing acquisition functions for bayesian optimization.

Zhu, H., Gu, M., and Peterson, B. (2007). Maximum likelihood from spatial random effects models via the stochastic approximation expectation maximization algorithm. *Statistics and Computing*, 17(2):163–177.