Bayesian History Matching for Nuclear Data Evaluation

James M. Petticrew

School of Mathematics and Statistics
University of Sheffield

Dissertation submitted as part of the requirements for the award of MSc in Statistics, University of Sheffield, 2020–2022

Acknowledgements

Thank you to Jeremy for spending a lot of time listening to me say nonsense and replying with sense. Thank you to previous colleagues names redacted for reasons redacted. Thank you to Henrik, Georg and Joachim at Uppsala. Thank you to my Button.

Lay Summary of the Dissertation

In nuclear physics applications, cross-sections are used to quantify the probabilities of interactions between radiation and matter across energy states. They are commonly modelled using computer simulation tools as it is impossible to measure them comprehensively enough across energy fidelities and interaction scenarios. However the capacity of these tools to produce accurate results is limited, partially due to incomplete knowledge of the physical processes they try to model, and partially because the simulator takes a set of user-defined input parameters, the correct values of which are not known.

One way to learn more about these parameters is through history matching. In this dissertation, history matching was used to efficiently identify and discard values of the parameters that were unlikely to result in the simulator giving good match to experiments once the inadequacy of the simulator and experimental imprecision were accounted for. A Gaussian process (GP) statistical model was used to emulate the relationship between the input parameters and the simulator outputs. This greatly sped up computation at the expense of adding another source of uncertainty, which had to be considered when assessing the inputs. A GP emulator was built using a small number of simulator runs at carefully chosen values of the input parameters. Emulators that displayed good out-of-sample predictive performance were then used to identify parameters settings that could plausibly result in the the simulator well matching experiments, and further emulators were built for using simulator runs at those parameter settings. Iterating over this process, known as refocussing, was shown to greatly reduce the space of plausible values for the 28 input parameters considered, at the cost of only several hundred simulator runs.

Contents

Acknowledgements **iii**

Lay Summary of the Dissertation **v**

1 Introduction **1**
- 1.1 Introduction . . . 1
- 1.2 Aim of the dissertation . . . 1
- 1.3 Structure of the dissertation . . . 2

2 Nuclear data evaluation **3**
- 2.1 Introduction . . . 3
- 2.2 Nuclear cross-sections . . . 3
- 2.3 TALYS nuclear reaction simulation code . . . 4
- 2.4 Experimental cross-section data . . . 5
- 2.5 Nuclear data evalution . . . 7
- 2.6 Previous uses of Gaussian processes in nuclear data . . . 7
- 2.7 Conclusion . . . 7

3 History matching **9**
- 3.1 Introduction . . . 9
- 3.2 Motivation . . . 9
- 3.3 Notation . . . 10
- 3.4 Workflow . . . 12
- 3.5 Sources of uncertainty . . . 15
  - 3.5.1 Observation uncertainty . . . 15
  - 3.5.2 Simulator inadequacy . . . 16
  - 3.5.3 Emulator variance . . . 16
  - 3.5.4 Conclusion . . . 17
- 3.6 Implausibility measure . . . 17
- 3.7 Multidimensional implausibility . . . 19
- 3.8 Conclusion . . . 19

4 Gaussian process regression **21**
- 4.1 Introduction . . . 21
- 4.2 Gaussian process regression . . . 21
- 4.3 Parameter estimation . . . 22

4.4 Conditioning on the data 24
4.5 Covariance function 26
4.6 Mean function 28
4.7 Software implementations 30
4.8 Conclusion 31

5 Results 33
5.1 First wave methodology 33
5.2 First wave analysis 35
5.3 Minimum implausibility and optical depth analysis 37
5.4 Subsequent waves 39
5.5 Posterior sampling of active inputs 40

6 Discussion 45
6.1 Summary 45
6.2 Limitations of the analysis 46
6.3 Further work and conclusions 49

A Spherical optical model potential 51

B R code to generate implausibility/ optical depth heatmaps 53

C Research Ethics Approval 57

References 59

Chapter 1

Introduction

1.1 Introduction

Nuclear cross-sections are fundamental material properties describing the interactions of radiation with matter. Accurate characterisations of cross-sections are essential for computer simulations of radiation effects, which are used extensively in applications such as medical technology, energy and defence. Detailed energy-dependent cross-section spectra are commonly generated using a synthesis of computer simulation and experimental observations, in a process known as evaluation.

In this dissertation Bayesian history matching (Andrianakis et al. 2015) (Bower, Goldstein, and Vernon 2010) is examined as a tool for nuclear data evaluation. This involves identifying values of input parameters for computer simulators which give non-implausible outputs given relevant observations and all sources of uncertainty, such as the imprecision of the experimental measurement process, or the inability of the simulator to exactly reproduce the physical process it is intended to model. To do the analysis it is required that the simulator output is examined for many different values of the input parameters. It is often not feasible to run the simulator enough times to do this, even for simulators with moderate run-times. Consequently a statistical emulator must be used to model the relationship between the simulator's input parameters and its outputs. This introduces another source of uncertainty, which is fully characterised using the model.

1.2 Aim of the dissertation

The aim of the dissertation was to investigate history matching as tool for nuclear data evaluation. The particular case study of neutrons incident upon an Iron-56 (Fe-56) isotope with energy between 5 and 10 MeV. Bayesian history matching was used to iteratively discount regions in the input space and subsequently build improved emulators for the relationship between TALYS and its input parameters in input regions of interest.

1.3 Structure of the dissertation

The structure of the rest of the dissertation is as follows. In Chapter 2 some background on nuclear data is presented, with discussion on the availability of relevant experimental data and nuclear reaction software. History matching is described in Chapter 3 and a notation is developed. In Chapter 4 an important component of history matching, Gaussian process regression, is described, and the relative merits of different software implementations are discussed. In Chapter 5 the first wave of history matching analysis is described in some detail, followed by a summary of the analysis in subsequent waves. Finally in Chapter 6 the results are summarised, with discussion on the limits of the analysis and potential for further work.

Chapter 2

Nuclear data evaluation

2.1 Introduction

In this section is given a brief introduction to the concept of nuclear cross-sections, the software used in this dissertation to simulate them, and the data used to compare with the results of the simulator. The section ends with a brief discussion on nuclear data evaluation, intended to contextualise this work, and on some existing uses of Gaussian processes for nuclear data from the literature.

2.2 Nuclear cross-sections

Imagine a mono-energetic neutron beam of intensity I_0 neutrons/m^2 and energy E Mega electron-volts (MeV) incident upon a slab of thickness r metres, made up solely of isotope Z, and with an isotope number density N in units of isotopes/m^3 (see Figure @ref(fig:xs_schematic)). If the intensity of the beam can be measured after it has passed through the slab I, then the total cross-section σ_{tot} can be computed from

$$I = I_0 \exp\left(-rN\sigma_{tot}(Z, E)\right). \tag{2.1}$$

$\sigma_{tot}(Z, E)$ is the microscopic total neutron cross-section for isotope Z for incident neutron energy E and is proportional to the probability of any interaction occurring between a nucleus of isotope Z and a neutron of that energy incident upon the nucleus. Cross-section units are m^2, however 1m^2 is an enormous value for a cross-section, and consequently they are often measured in 'barns'[1] where a barn is equal to 10^{-28}m^2. A number of interactions can occur after a collision, such as the neutron changes its momentum (known as inelastic scatter) or the neutron is captured by the nucleus consequently releasing more neutrons (and energy - known as fission). Each interaction type has its own cross-section

[1]From the phrase 'as wide as a barn door'.

value in barns, which depends on the energy of the incident particle, the type of incident particle and the type of target nucleus. The capacity for scientists to measure all interaction, energy, particle and isotope combinations is limited, and as such computer simulations, verified using experimental measurements, are used extensively to model cross-sections with suitable fidelity. In this work the nuclear reaction simulation code TALYS (A. J. Koning, Hilaire, and Duijvestijn 2008) was used.

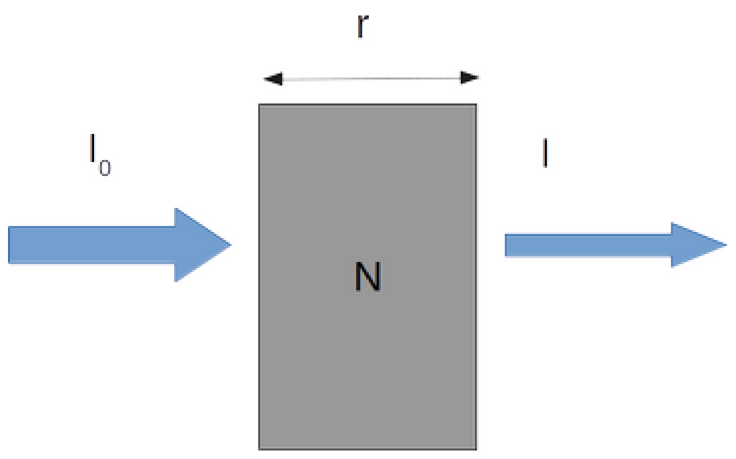


Figure 2.1: Illustration to aid interpretation of microscopic neutron cross-section. The ratio of the outgoing to the incident beam intensity is proprtional to the cross-section.

2.3 TALYS nuclear reaction simulation code

TALYS is a computer simulation tool which implements a suite of mathematical models to predict nuclear reactions. It has some 340 input parameters (A. J. Koning, Hilaire, and Duijvestijn 2008) which can be specified by the user. Here a distinction is made between state and design parameters. State parameters do not change between runs of the code, and define the particular scenario being examined. Examples of state parameters include incident particle type, target isotope and incident particle energies. Design parameters are changed between code runs. These are parameters in the mathematical model(s) implemented by the simulator, the true values of which are unknown. Hence we wish to compare observational data to the outputs of our simulator for different choices of design parameters. The hope is that design parameters settings which produce simulator outputs consistent with observational data will also produce simulator outputs consistent with reality for unobserved quantities.

In this dissertation 28 design parameters were examined, all part of the optical model potentials (OMP) implemented in TALYS (Appendix A), or extensions thereof. Parameters in the mathematical models implemented in TALYS have a default setting which is used if a value is not specified for it in the input file. This allows calculations to be carried out without having to specify values for all 340 input parameters. The defaults are by no means a 'best' set; the appropriate setting for each parameter depends, amongst other things, on the incident particle and target isotope being modelled. There are thousands of potential combinations for these. The actual value specified for a parameter in the input file acts as a multiplier for this default. For example, if the default value for a parameter is 4 and the user inputs a value of 0.5, the calculations will be carried out with the parameter equal to 2. Hence if a value for the multiplier is not specified in the input file, it is implicitly one. Each multiplier must be set to a value in the range $[0.1, 10]$. Initially, we assumed that every possible value for the design parameters was equally likely, which is not the same thing as the multipliers being equally likely (see Section **??**). In this dissertation, when we discuss values for the design parameters, we mean values of the multipliers, rather than the parameters themselves. We can think of the multipliers as being parameters in the model in Appendix A if we re-parametrise it with the default values as constant coefficients of the multipliers.

2.4 Experimental cross-section data

Experimental nuclear reaction measurements have been recorded since the discovery of the neutron, and a comprehensive database of these exists in EXFOR (Otuka et al. 2014), maintained and developed by International Network of Nuclear Reaction Data Centres (NRDC), coordinated by the International Atomic Energy Authority (IAEA). The data are open source, and are technically freely available to all. However, successfully querying the EXFOR database is a difficult task, which was much simplified using the framework created by the nuclear data evaluation pipeline software described in (Schnabel et al. 2021).

A total of 23183 data points pertaining to neutrons incident upon the Iron-56 isotope were pulled from the database. Each point is a measurement for a specific reaction at a specific incident neutron energy. It was decided to focus on incident neutrons with energies between five and 10 MeV, for which there were 1141 experimental observations. The data were then filtered further to include only measurements of reactions that TALYS could simulate directly; total cross-section (n,tot) - which is proportional to the probability of any reaction occurring, inelastic cross-section (n,n') - which is proportional to the incident neutron experiencing a change of momentum ('bouncing off' the nucleus), proton cross-section (n,p) - which is proportional to the probability of the incident neutron being absorbed by the nucleus and ejecting a proton as result, and alpha cross-section (n,a) - which is proportional to the probability of the incident neutron being absorbed by the nucleus and ejecting an alpha particle (a Helium nucleus) as result. A summary of the remaining relevant experimental data points is shown in Table 2.1.

The data are dominated by total cross-section measurements, which are the sums of the cross-sections for all possible reactions at a given energy. This is because total cross-sections are much easier to measure, (using the attenuation method

Table 2.1: Counts of relevant experimental Iron-56 neutron cross-section data points in the EXFOR database by reaction. The experimental data are dominated by total cross-section measurements.

Reaction	Number of observations
(n,a)	4
(n,n')	5
(n,p)	56
(n,tot)	1065

briefly described in Section 2.2) whereas it is much more difficult to measure, for example, the (n,p) reactions, where a proton is released from the nucleus, as it requires a specific ejectile to be detected. The experimental data covered a range of incident neutron energies, which can be set as state parameters (Section 2.3) in the TALYS input file. The run time of the simulator is proportional to the number of incident neutron energies it is required to simulate. Consequently, it was decided to only use half of the (n,tot) data points, both to reduce TALYS run time and to slightly reduce the dominance of (n,tot). The energies were filtered by ordering them and picking every second energy. Consequently 608 relevant data points were used in the analysis. Once duplicates were accounted for, there were 588 processes that needed to be computed for each TALYS run, where a 'process' in this context means a cross-section for a given reaction at a given (incident neutron) energy.

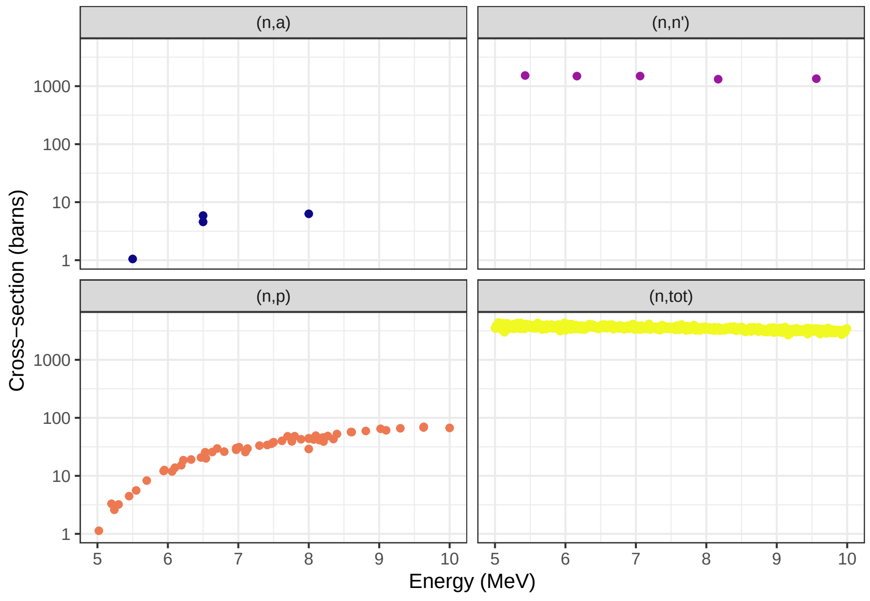


Figure 2.2: Plots of relevant Fe-56 nuetron cross-section data extracted from the EXFOR database. Each panel shows data for a different reaction cross-section. 1065 (n,tot) data points are plotted, it was decided to only use half of them in the analysis.

2.5 Nuclear data evalution

A single run of TALYS is capable of producing cross-sections for a number of reactions over a range of user defined incident particle energies. The aim of nuclear data evaluation is to produce the set of cross-sections believed to be most consistent with experiments and subject matter expert (SME) knowledge. When using a simulation tool such as TALYS, part of this process must consist of deciding at what value, or values of the design parameters at which to run the simulator. This is the aspect of nuclear data evaluation that we focus on in this dissertation. In Bayesian history matching we iteratively discount large regions of the space of possible design parameter values with the aid of Gaussian process regression. Evaluation is a huge task, and there are several prominent cross-section libraries that are created using different evaluation philosophies (see for example (Brown et al. 2018) (Shibata et al. 2011) (A. Koning et al. 2019)), and with different priorities. The methodology described in this dissertation represents a reasonably uncomplicated pragmatic way of generating simulator results that are consistent with empirical observations.

2.6 Previous uses of Gaussian processes in nuclear data

The use of Gaussian processes in nuclear data is not novel. In References (Schnabel et al. 2021) and (Helgesson and Sjöstrand 2018) the authors used Gaussian processes to describe the relationship between systematic errors and incident particle energies, and in Reference (Iwamoto 2020) there is a demonstration of how to generate cross-sections from experimental data only. A more recent Bayesian approach to evaluation employing maximum likelihood can be found in Reference (Alhassan et al. 2022).

2.7 Conclusion

In this section we gave a brief overview of nuclear cross-sections. We described the two main sources of data used in the dissertation, simulated cross-sections from TALYS and measured cross-sections from the EXFOR database, and gave some context for the application. The data described here was used in a process known as history matching, where we try to find values for the design parameters such that the simulated cross-sections produced by TALYS give good matches to the measured cross-sections extracted from the EXFOR database. We describe the history matching process in the next chapter.

Chapter 3

History matching

3.1 Introduction

In this section we give an overview of the main quantities involved in history matching and then develop the notation that we will use for the rest of the dissertation. We then describe the process workflow of history matching, and describe the various sources of uncertainty that need to be characterised in order to carry out the analysis. We then describe some of the metrics that we use to help us identify suitable values for TALYS design parameters.

3.2 Motivation

The primary goal of this dissertation is to identify a subset or subsets of the 28-dimensional design parameter space which could give rise to acceptable matches between TALYS outputs, denoted f, and the true cross-section which it attempts to simulate, which we denote y. The term 'acceptable' here implies that an exact match is not expected. One reason for this is that we cannot observe y perfectly, but that there is a discrepancy between the measured cross-section, denoted z, and y. A second reason is that we do not expect TALYS to perfectly recreate y. At third reason is that, as discussed in Section 4, history matching requires f to be evaluated an infeasibly large number of times and consequently we use a statistical emulator $\hat{f}$ to approximate f. The relationships between these four quantities are shown in Figure @ref(uncertainty sketch). If we are able to well quantify the discrepancies between the quantities (see Section 3.5) as well as the quantities themselves, then we can compute some metric to decide if the match been TALYS and the true process is acceptable (Section 3.6).

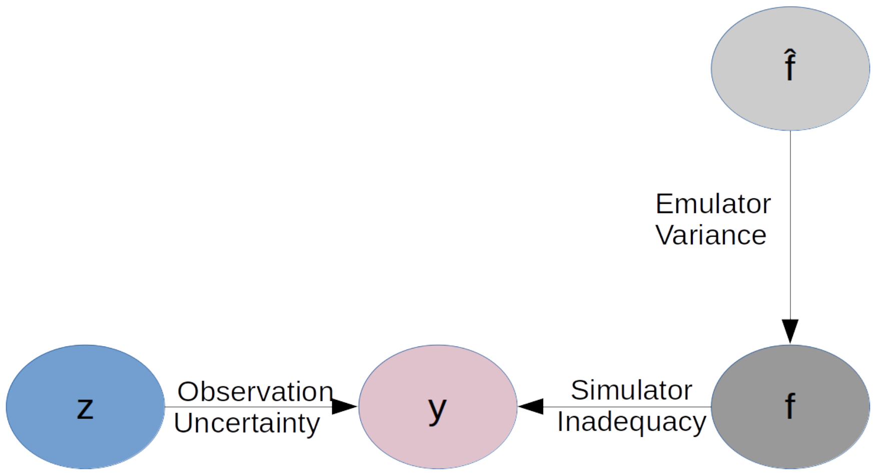


Figure 3.1: Both the experimental measurement z and the simulator f imperfectly capture the true process y. A further discrepancy occurs because the simulator itself most be imperfectly emulated by f.

History matching is a method which allows us to find values of design parameters for TALYS which give good agreement with empirical data. The assumption is that this increases the capacity of the simulator to generalise, giving good predictions for processes which have not, or cannot be, observed. This is pertinent for TALYS, where some of the cross-sections are almost impossible to measure. History matching works by iteratively discounting values of the design parameters as implausible give empirical observations and relevant uncertainties, and retraining sets of statistical emulators which focus on predicting well in the non-implausible design space, known as 'refocussing'. In order to describe it in more detail, we now develop our mathematical notation further.

3.3 Notation

In history matching we have a simulator with p design parameters, denoted by the p-vector $\mathbf{x} = [x^{(1)}, x^{(2)}, ..., x^{(p)}]$, that do not have fixed values and on which the simulator outputs depend. In this application $p = 28$. $\mathbf{x}$ can be any point in domain $\mathcal{X}$, which could be the set of all $\mathbf{x}$ that the simulator will accept as inputs, or a smaller subset of feasible inputs dictated by physical constraints or expert knowledge. In this application, $\mathcal{X} = [0.1, 10]^{28}$ as described in Section 2.3. The simulator implements a mathematical model describing some physical process y. The physical process is observed imperfectly through some measurement process z. We have a vector of experimental data points, denoted by the n-vector $\mathbf{z} = [z_1, z_2, ..., z_n]$, and index the individual measurements as z_i $i \in 1, 2, ..., n$. These are measurements of the n physical processes, denoted by the n-vector $\mathbf{y} = [y_1, y_2, ..., y_n]$. In this application $n = 608$. The simulator outputs a vector of quantities and consequently it can simulate all of $\mathbf{y}$ in one code run at some value of $\mathbf{x}$. We denote this output n-vector $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_n(\mathbf{x})]$. In the context of the discussion in Section 2.4, each element of $\mathbf{f}(\mathbf{x})$ corresponds to a cross-section for a given reaction at a given energy.

To represent the imperfection of the experimental measurement process we write

$$z_i = y_i + \delta_i \tag{3.1}$$

where δ_i is a univariate Gaussian random variable with expectation 0 and variance $V_i^{(obs)}$ which we use to represent our belief about the uncertainty of $y_i|z_i$. We also assume that simulator is an imperfect representation of reality and represent this by writing

$$f_i(\mathbf{x}) = y_i + \phi_i \tag{3.2}$$

where ϕ_i is a zero-centred univariate Gaussian random variable with variance $V_i^{(s)}$ used to represent out belief about the uncertainty of $y_i|f_i(\mathbf{x})$. Here we assume that ϕ_i is independent from $\mathbf{x}$.

The aim of history matching is to identify a subset or subsets of $\mathcal{X}$ for which the simulator could feasibly produce outputs consistent with the true process y. This requires comparing the simulator output to observations for m proposal inputs, $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_m}$, indexed using $j = 1, 2, ..., m$, where each of the $\mathbf{x_j}$ is a p-vector with a proposed scalar value for each of the p design parameters. Theoretically, if the models in TALYS were a perfect representation of reality, there would exist a $\mathbf{x_j}$ such that $f_i(\mathbf{x_j}) = y_i \ \forall i$, since all of the elements of $\mathbf{x}$ are observable quantities. This situation could also arise if TALYS were not a perfect representation of reality (getting the right answer for the wrong reasons).

In order to well explore $\mathcal{X}$ it is required that m is very large. It is often the case that one run of the simulator can take several hours to complete, and as such it is not possible to examine a very large number of proposal $\mathbf{x_j}$ directly. To address this, we use $k << m$ training runs of the simulator to build n emulators, one for each of the $f_1(\cdot), f_2(\cdot), ..., f_n(\cdot)$. We denote the subset of $\mathcal{X}$ used for the training runs as $\mathcal{X}^*$ and the corresponding outputs as $\mathbf{f}(\mathcal{X}^*) = [\mathbf{f_1}(\mathcal{X}^*), \mathbf{f_2}(\mathcal{X}^*), ..., \mathbf{f_n}(\mathcal{X}^*)]$, where each element of $\mathbf{f}(\mathcal{X}^*)$ is a k-vector, each element of which corresponds to the output of one of the k training runs. We use these emulators to estimate $\mathbf{f}(\mathbf{x_j})$ for proposal points $\mathbf{x_j}$ at which the simulator has not been evaluated. Each of the n elements in $\mathbf{f}(\mathbf{x_j})$ is estimated using its own emulator, and the n emulators are statistically independent. There is uncertainty about $f_i(\mathbf{x_j})$, and we represent this uncertainty using a random variable structure. $E[f_i(\mathbf{x_j})]$ denotes our belief about the expected value of $f_i(\mathbf{x_j})$ and $V[f_i(\mathbf{x_j})]$, the marginal emulator variance, represents our belief about the uncertainty of $f_i(\mathbf{x_j})|f_i(\mathcal{X}^*)$. In the case of a deterministic simulator, as we have here, if $\mathbf{x}_j \in \mathcal{X}^*$ then $V[f_i(\mathbf{x_j})] = 0$ and $E[f_i(\mathbf{x_j})] = f_i(\mathbf{x}_j)$.

A summary of the notation used hereafter is presented in Table 3.1.

Table 3.1: A summary of notation used to describe the history matching analysis

Symbol	**Meaning**
$\mathbf{x}$	Vector of simulator calibration parameters
p	Number of calibration parameters
$\mathcal{X}$	The set of all values that $\mathbf{x}$ can take
y_i	The ith physical process that the simulator attempts to reproduce
z_i	An experimental measurement of the process y_i
n	The total number of experimental measurements available in the analysis
$f_i(\mathbf{x_j})$	Simulated value of the physical process y_i for calibration input $\mathbf{x_j}$
δ_i	Random variable representing our uncertainty about the discrepancy between the process y_i and its measurement z_i
$V_i^{(obs)}$	Variance of δ_i
ϕ_i	Random variable representing our uncertainty about the discrepancy between the process y_i and its simulated value $f_i(\mathbf{x})$
$\mathcal{X}^*$	Subset of $\mathcal{X}$ used to build emulators for $\mathbf{f}(\cdot)$
k	Number of points in the training set $\mathcal{X}^*$
$E[f_i(\mathbf{x_j})]$	Emulator expectation value for the ith simulator output at input $\mathbf{x_j}$
$V[f_i(\mathbf{x_j})]$	Marginal emulator variance for the ith simulator output at input $\mathbf{x_j}$

3.4 Workflow

Here we give an overview of the steps carried out in history matching, following the schematic shown in Figure 3.2.

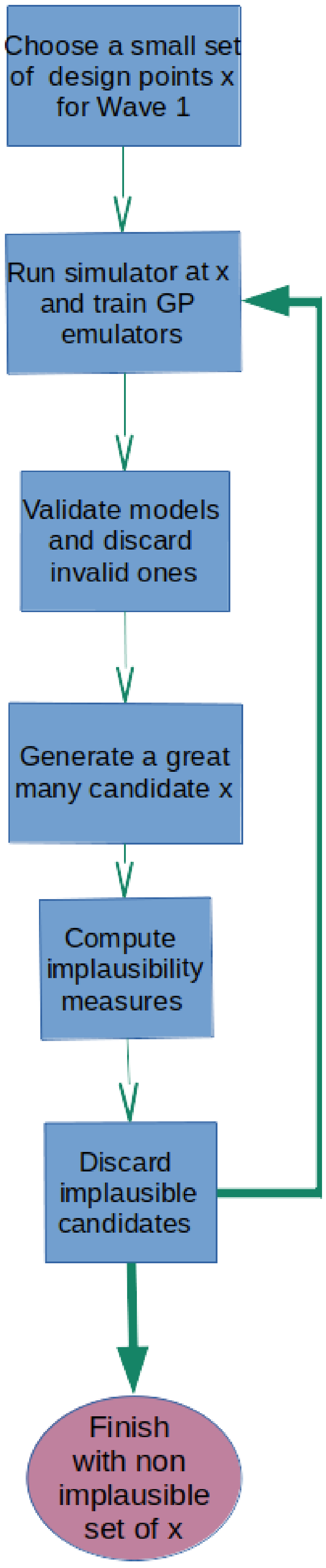

Choose a small set of design points x for Wave 1
Run simulator at x and train GP emulators
Validate models and discard invalid ones
Generate a great many candidate x
Compute implausibility measures
Discard implausible candidates
Finish with non implausible set of x

Figure 3.2: Schematic showing a typical history matching workflow. The process is iterative where each iteration is called a wave, and a wave ends either with refocussing, which is a reduction of the non-implasible input space, or with stoppinng with the current non-implausible space.

- **Choose design points for Wave 1**: The number of candidate input parameters that we wish to examine is very large but the number of times we can run TALYS to evaluate these inputs directly is relatively small. Consequently the best use of the limited number of runs we can afford is to build a statistical emulator for TALYS. In this dissertation we begin with a 28-dimensional uniform input parameter space $\mathcal{X}$. Initially, we have no idea of the relative plausibilities of samples from this space, and we want to build emulators from a limited number of TALYS runs that well represent its behaviour over the entirety of $\mathcal{X}$. To be in with the best chance of doing this, and to minimise the predictive variance of the emulators, we want to employ a space-filling design that provides coverage of $\mathcal{X}$ in some way that is optimised for the number of runs that we can afford. Typically this is not achieved from randomly sampling our design points in $\mathcal{X}$, or from a uniform design. In this dissertation we use a Latin Hypercube (LH) design. This is a high-dimensional extension of the Latin square experimental design.

- **Run simulator at design points and use results to build a emulators**: Once the design points have been chosen, TALYS needs to be run to obtain the $\mathbf{x} - y$ mappings needed to train the Gaussian process emulators. The results then need to be consolidated and formatted to suit the input format of the Gaussian process software package being used (see Section 4.7). In some cases, it may be desirable to scale $\mathbf{x}$ or y to help the modelling. For example, it is often useful to log-scale strictly non-negative outputs, particularly if they vary across orders of magnitude. When training many emulators, it may be the case that some models fail, perhaps due to convergence problems in the numerical estimation routine (see Section 4.3), especially if the design space is very large, as some outputs may be hard to emulate initially. One advantage of history matching is that the analysis is not invalidated if we simply drop these emulators at this stage of the analysis.

- **Validate models and discard invalid ones**: A small number of TALYS runs should also be carried out in order to test the out-of-sample predictive performance of the emulators. Any emulators not giving satisfactory out-of-sample performance should be dropped from this wave of history matching. More details on the validation methodology used in this dissertation are given in Section **??**.

- **Generate a great many candidate x**: The validated emulators can now predict the TALYS output a great number of times at a fraction of the cost of running the simulator. Consequently we generate very many samples for $\mathbf{x}$ and use the emulators to generate predictions for each one.

- **Compute implausibility measures**: As discussed in Section 3.6 below, we compute implausibility metrics for each candidate $\mathbf{x}$ as a function of the emulator predictions, the observed quantities, and the uncertainties present system. This typically leads to a large reduction in the non-implausible parameter space (Andrianakis et al. 2015) (Bower, Goldstein, and Vernon 2010), based on the implausibility metrics exceeding some acceptable threshold.

- **Discard implausible candidates**: The reduction in non-implausible parameter space is know as refocussing, and can lead to two outcomes. The non-implausible space (or a subset of it) can now be used as the values for design variables for the next set of TALYS runs, upon which a new set of emulators is built, and the process is repeated. This has the advantage of focussing the emulators in areas which have been shown to give acceptable matches to observations, leading to more precise emulation in these regions. As will be seen in Section 3.6 this can lead to fewer candidate points being accepted as non-implausible, refocussing even further. A second outcome from a given wave is that the process is stopped, and the current non-implausible space is accepted as the outcome of the analysis. Stopping may occur because the computational budget for TALYS runs has been reached, or because the proportional reduction in parameter space that occurred in this wave is not significantly different than those in previous waves, indicating that there is little to be gained in continuing.

History matching can be used as a way of helping to understand which parameter values are consistent with observations, which may be especially useful if the parameters represent observable quantities. It may be also used as a precursor to an uncertainty quantification study where, having found the values for the input parameters consistent with observations, we now wish to find the set of different cross-section values consistent with observations.

In order to decide if input parameter candidates $\mathbf{x}$ are implausible or not, we need to quantify the uncertainties that are present in the system. We discuss these, and how we model them, in the next section.

3.5 Sources of uncertainty

3.5.1 Observation uncertainty

In many ways we can think of the process of measurement as an aleatoric random process, in that if we take one measurement $z_i^{(1)}$ of the process y_i and then take a second measurement $z_i^{(2)}$ then in general $z_i^{(1)} \neq z_i^{(2)}$. We generally make an assumption that the measurement process is imprecise, but not systematically wrong. This implies that if we take n measurements $z_i^{(1)}, z_i^{(2)}, ..., z_i^{(n)}$, as n gets larger $\frac{1}{n}\sum_{j=1}^{n} z_i^{(j)} \rightarrow y_i$ and that the variance of the estimator for y_i is proportional to $\frac{1}{\sqrt{n}}$. In most practical cases, it is only possible to take one measurement z_i and the scientist must provide their own estimate of the observation uncertainty, based on their knowledge of the experimental process.

The authors of Reference (Schnabel et al. 2021) provide tools for extracting and contextualising the uncertainties that accompanied the experimental data used in this dissertation. However, the computational expense required was too large to undertake, and consequently a simple approach was taken in order allow progress to be made. It was decided that the observation uncertainty $V_i^{(obs)}$ for y_i should be $0.1z_i$. We write

$$y_i = z_i + \delta_i \tag{3.3}$$

where $\delta_i \sim N(0, V_i^{(obs)})$.

3.5.2 Simulator inadequacy

We accept that there can be a discrepancy between the TALYS output f and the true process y, and we expect this discrepancy to arise in several ways. First, TALYS implements some mathematical model, which is assumed to be an imperfect representation of reality. Second, the simulator itself may implement the mathematical model imperfectly.

The objective of history matching is not to find the 'correct' input parameters for a given simulation, and the method does not require the inputs to be physically meaningful. A consequence of this is that history matching can give the right answer for the wrong reasons. For example, the mathematical model may be imperfect, but for certain values of its parameters it might give a good match to observations. Consequently, simulator inadequacy is difficult to quantify, and even more difficult to characterise in terms of contributions from different sources. One approach would be to consult a subject matter expert (SME) or experts to attempt to elicit a probability distribution representing their belief about how great a discrepancy could arise between TALYS and reality. In this dissertation, in order to focus on the methodology, we take a very simple approach to uncertainty quantification, equating it to the corresponding measurement uncertainty of the process being simulated. We write

$$y_i = f_i(\mathbf{x}) + \phi_i \ \forall \mathbf{x} \tag{3.4}$$

where $\phi_i \sim N(0, V_i^{(s)})$ and $V_i^{(s)} = V_i^{(obs)}$. In particular, ϕ_i is independent of $\mathbf{x}$ in this model.

3.5.3 Emulator variance

As described in Section 4, we use an emulator $\hat{f}$ to represent the simulator f as it allows us to explore the input space much more efficiently. The price for this efficiency is that another source of uncertainty is introduced. The uncertainty depends on $\mathbf{x}$ in this case and we write

$$f_i(\mathbf{x_j}) \sim N\left(E[f_i(\mathbf{x_j}), V[f_i(\mathbf{x_j})]\right], \tag{3.5}$$

where $E[f_i(\mathbf{x_j})]$ is computed from Equation (4.7) and $V[f_i(\mathbf{x_j})]$ is computed from (4.8). We go into more detail on this in the next chapter.

3.5.4 Conclusion

In this dissertation we consider uncertainties that arise from the imprecisions of the measurement process, the inability of TALYS to perfectly simulate cross-sections, and the imprecise predictions that arise from using a statistical model to represent TALYS. In some applications, other sources of uncertainty may exist. For example, the application may required a simulator that employs Monte Carlo methods, and as such does not always give the same output when ran twice at the same input. This uncertainty could be estimated from multiple simulator runs. Once all sources of uncertainty have been quantified, it is possible to compute implausibility metrics to help assess if the simulator f_i could give an acceptable match to the true process y_i.

3.6 Implausibility measure

We want a metric that is a function of $\mathbf{x}$ and gives large values if $f(\mathbf{x})$ is unlikely to give an acceptable match to y and smaller values if it is more likely to give an acceptable match. Consequently it makes sense if the metric is proportional to $|f(\mathbf{x}) - y_i|$. But we also have to take into account the uncertainties discussed in Section 3.5. The greater the combined uncertainty in the system, the less sure we can be in branding values of $\mathbf{x_j}$ implausible. Consequently it makes sense if the metric is inversely proportional to the combined uncertainty. The standard approach to combining uncertainties (Andrianakis et al. 2015) (Bower, Goldstein, and Vernon 2010) is to assume that they are independent, and consequently that the variances are additive. We take that approach here, and hence the one-dimensional implausibility metric for $\mathbf{x_j}$ is

$$I_j = \frac{|E[f(\mathbf{x})] - z_i|}{\sqrt{V_i^{(obs)} + V_i^{(sim)} + V[f_i(\mathbf{x_j})]}}. \tag{3.6}$$

This metric is proportional to the difference between the observed and simulated process, inversely proportional to the total uncertainty in the system, and depends on $\mathbf{x_j}$. In order to decided what value of I_j to use as the cutoff for deeming $\mathbf{x_j}$ plausible/ implausible, we leverage Pukelsheim's 3σ rule, which states that for any continuous unimodal distribution, 95% of its probability mass lies within 3 standard deviations of its mean (Pukelsheim 1994). One way to proceed would then be to reject all values of $\mathbf{x_j}$ for which $I_j > 3$ as implausible. Assuming that $|E[f(\mathbf{x})] - z_i|$ meets Pukelsheim's requirements, we should then retain 95% of all plausible $\mathbf{x_j}$ on average, and pay the price of losing 5% of our non-implausible $\mathbf{x_j}$. In practice, this boundary can be moved to allow a more suitable number of candidate $\mathbf{x_j}$ to be accepted as not implausible if required.

Suppose that the function $y = x^2 - 5x$ represents a simulator with a single input x and we examine one of its outputs, y. We have made a measurement z of y and found that $z = 100$, with some uncertainty associated with it. We have run our simulator at five different values for x and built an emulator for y. We then predict the simulator output at a great many points. Figure 3.3 illustrates the scenario. The dots are the design point, which are interpolated by the simulators mean prediction. The emulator predictor variance has been summed with the observation uncertainty and simulator inadequacy variances and the corresponding 3σ intervals are shown in the Figure as the solid rectangle.

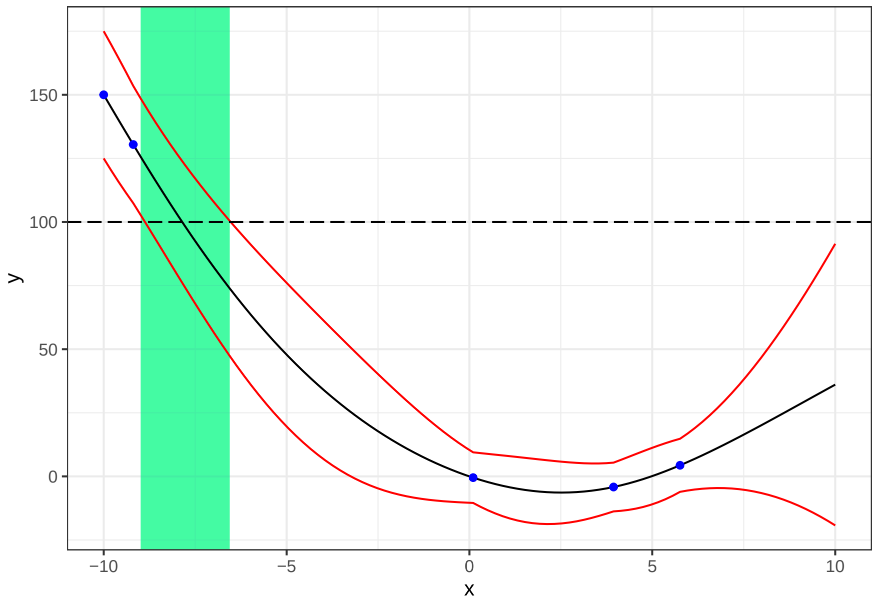


Figure 3.3: The simulator y is a function of one input x. The simulator has been run at the five points shown, and an emulator is built using these five points. Its predictions are shown interpolating the design points. An observation has been made of y=100. The uncertanties arising due to the emulator, simulator and act of observation are combined and the 3 sigma uncertainty intervals are plotted. If we use a 3 sigma cutoff for implausibility, only those values of x shown interessecting the rectangle are non-implausible.

If we set our implausibility cutoff at 3 σ, values for x that intersect the rectangle are non-implausible - roughly the interval [-8.99,-6.97]. The correct value is around -7.81. If we were then to run the simulator at a point in that region, the emulator variance, would shrink to zero at that point, reducing the width of the rectangle further. However, simulator inadequacy and observation uncertainty would still remain and so the correct input could not be found precisely. It is also worth noting that the non-implausible region need not be continuous. From Figure 3.3 we could imagine that were the domain of x to extend to +15, the mean prediction would intersect $y = 100$ again, and indeed a second, much wider interval in x would be deemed no implausible as the 3 σ intervals grow wider as the simulator predicts further from the data. In subsequent runs, candidate

x's from this interval would be evaluated using the simulator and used to build emulators that predict better in this region, consequently the non-implausible space would shrink.

I is a one-dimensional implausibility metric, computed with respect to a single TALYS output f. In this dissertation, we examine hundreds of TALYS outputs simultaneously, and it is the job of the simulator to give non-implausible matches to them all. Consequently we extend the idea of the implausibility measure in Equation (3.6) to two implausibility measures for multiple outputs.

3.7 Multidimensional implausibility

Hundreds of outputs $f_1(\mathbf{x_j}), f_2(\mathbf{x_j}), ..., f_n(\mathbf{x_j})$ are produced each time we run TALYS for the input $\mathbf{x_j}$. Each output $f_i(\mathbf{x})$ has a corresponding measurement z_i of a true process y_i. Some approaches exist for emulating multivariate outputs and accounting for correlations (Rougier 2008) (Fricker, Oakley, and Urban 2013). In this approach we take the simplest possible approach and emulate each of the f_i independently. As a consequence of this, there are n univariate implausibility measures for each input, and the criterion for labelling $\mathbf{x_j}$ plausible or non-implausible must be modified. One approach is to take the maximum of the n measures as I_j and to require that this be less than or equal to three for $\mathbf{x_j}$ to be implausible. The second or third largest I_j can also be taken - again, the sensitivity of the size of the non-implausible space to the choice of metric can be examined, and a pragmatic choice can be made.

A second multivariate implausibility metric is

$$I(\mathbf{x_j}) = ((z) - E[f(\mathbf{x_j})])^T \left(\mathbf{V}_i^{(obs)} + \mathbf{V}_i^{(sim)} + \mathbf{V}[f_i(\mathbf{x_j})]\right) ((z) - E[f(\mathbf{x_j})]). \quad (3.7)$$

Where we now have the vector of observations $\mathbf{z} = (z_1, z_2, ..., z_n)$ and the corresponding emulated quantities $\mathbf{f}(\mathbf{x_j}) = (f_1(\mathbf{x_j}), f_2(\mathbf{x_j}), ..., f_n(\mathbf{x_j}))$ and where $\left(\mathbf{V}_i^{(obs)} + \mathbf{V}_i^{(sim)} + \mathbf{V}[f_i(\mathbf{x_j})]\right)$ should now be a full $n \times n$ covariance matrix. $I(\mathbf{x_j})$ has an asymptotic χ^2 distribution and consequently we can choose a cut-off percentile from the appropriate distribution for deeming $\mathbf{x_j}$ plausible or implausible.

3.8 Conclusion

In this section we discussed the process workflow of history matching, and two of its important components, uncertainties and implausibility metrics. We also discussed the idea of using a Gaussian process emulator as a fast approximation for TALYS, enabling the space of design parameters to be explored more efficiently, at the cost of introducing another source of uncertainty. We dicuss Gaussian processes further in the next chapter.

Chapter 4

Gaussian process regression

4.1 Introduction

In this section we discuss some of the key ideas in Gaussian process regression (GPR). We start with a discussion of the theory, followed by two key components of GPR, the covariance kernel and mean function. We finish with a discussion on some of the software packages with with we attempted at least some of the modelling in this dissertation, highlighting the key features that led to them being accepted or rejected as the tool of choice.

4.2 Gaussian process regression

In this section we reason about a single output of TALYS $\hat{y}_i = \hat{f}_i(\cdot)$. There is no loss of generality in proceeding this way, as we chose to model all of the TALYS outputs independently, so we drop the i subscript for convenience.

We wish to use a statistical model as an emulator for TALYS in order to predict, with associated uncertainty estimates, the output of $\hat{f}(\cdot)$ at some unobserved inputs $\mathbf{x}^{\dagger}$. If we take some arbitrary input vector $\mathbf{x}$ and run TALYS twice for this input we get $y^{(1)} = \hat{f}(\mathbf{x})$ and $y^{(2)} = \hat{f}(\mathbf{x})$. Because TALYS is deterministic, $y^{(1)} = y^{(2)}\ \forall\ \mathbf{x}$. Consequently the uncertainty around $y = \hat{f}(\mathbf{x})$ reduces to zero once y has been observed, and there is no uncertainty due to random behaviour, often called aleatoric uncertainty, associated with the process $\hat{f}(\cdot)$. However, for $y = \hat{f}(\mathbf{x})$ where y is unobserved (the simulator has not been run at $\mathbf{x}$), there is uncertainty about the value of y due to incomplete knowledge about the process $\hat{f}(\cdot)$. This uncertainty is often described as epistemic, and in the Bayesian framework we use statistical models to describe our belief about the unobserved process, and the uncertainty surrounding it. If we denote as $\mathbf{x}^{\dagger} \subset \mathcal{X}$ the set of points at which we have not evaluated $\hat{f}(\cdot)$, then we treat the outputs at these points as random variables $Y = \hat{f}(\mathbf{x}^{\dagger})$ and $\hat{f}(\cdot)$ as a stochastic process.

We assume that Y is the sum of a deterministic trend function, $\mu(\cdot)$ and a zero mean Gaussian process, $Z(\cdot)$ (Roustant, Ginsbourger, and Deville 2012a):

$$Y(\mathbf{x}) = \mu(\mathbf{x}) + Z(\mathbf{x}). \tag{4.1}$$

A Gaussian process (GP) is a collection of random variables, any finite number of which has a multivariate Gaussian distribution (Williams and Rasmussen 2006). Although there is no upper bound to the number of random variables that can be a part of the GP, we are only ever concerned with two subsets of all possible Y, the set of y^* which we have observed, and the set of $Y^\dagger$ at which we wish to predict. Before we observe the y^*, we model out belief about the joint distribution of the two sets of random variables with the multivariate normal likelihood

$$\begin{bmatrix} \mathbf{Y}^* \\ \mathbf{Y}^\dagger \end{bmatrix} \sim MVN\left(\begin{bmatrix} \mu(\mathbf{x}^*) \\ \mu(\mathbf{x}^\dagger) \end{bmatrix}, \begin{bmatrix} \mathbf{\Sigma}^* & \mathbf{\Sigma}^{*\dagger} \\ \mathbf{\Sigma}^\dagger & \mathbf{\Sigma}^{\dagger *} \end{bmatrix} \right) \tag{4.2}$$

For some mean function $\mu(\cdot)$ (whose form is the same for all y but takes different values due to x) and some covariance matrices $\mathbf{\Sigma}^*, \mathbf{\Sigma}^\dagger$, which are functions of $\mathbf{x}^*$ and $\mathbf{x}^\dagger$ respectively, and $\mathbf{\Sigma}^{*\dagger} = (\mathbf{\Sigma}^{\dagger *})^{-1}$, which are functions of both $\mathbf{x}^*$ and $\mathbf{x}^\dagger$. $\mathbf{x}^* = \{\mathbf{x}_\mathbf{1}^*, \mathbf{x}_\mathbf{2}^*, ..., \mathbf{x}_\mathbf{k}^*\}$ denotes the set of k points in $\mathcal{X}$ for which we have observed simulator outputs $\mathbf{y}^*$. The function $Z(\cdot)$ is constrained to have specific properties, which we discuss in Section 4.5, in order to be considered a GP, as the Σ must have the same fundamental properties as the covariance matrix of a multivariate normal distribution. The function $\mu(\cdot)$ can be any normal linear regression function which we discuss in Section 4.6.

Both $\mu(\cdot)$ and $Z(\cdot)$ are functions with parameters that we must learn about from $(\mathbf{x}^*, \mathbf{y}^*)$. Parameter estimation methods vary between software implementations. After learning the parameters we can make inferences about the joint distributions of unobserved outputs $\mathbf{Y}^\dagger$, conditional on both the $\mathbf{y}^*$ and the learned parameters, which is what we wish to do in history matching to efficiently explore the space of possible values of $\mathbf{x}$.

4.3 Parameter estimation

The mean function in a GP takes the form of a linear model

$$\mu(\mathbf{x}) = \sum_{i=1}^{p} h_i(x_i)^T \beta_i. \tag{4.3}$$

Where p is the number of elements in the vector $\mathbf{x}$. The h_is are fixed basis functions. These basis functions are chosen by the user, some examples being $h_i(x_i) = x_i$ and $h_i(x_i) = \sin(x_i)$. Each element of the covariance matrix is usually given by $c(\mathbf{x}, \mathbf{x}') = \sigma^2 r(\mathbf{x}, \mathbf{x}')$, where r is some function that depends on

$|\mathbf{x} - \mathbf{x}'|$ where $\mathbf{x}$ and $\mathbf{x}'$ are two points from the input parameter space $\mathcal{X}$. So the covariance between two of the $\mathbf{y}$s depends in some way on the distance between their $\mathbf{x}$-coordinates. The function r has p hyperparameters, denoted γ which must be learned from the observed data, one for each element of $\mathbf{x}$. An optional noise parameter, sometimes called a nugget, can be learned (see Section 4.5 for more on γ and the noise parameter). We denote the vector of model parameters θ.

The mean function can simply be $\mu(\mathbf{x}) = 0 \ \forall \ \mathbf{x}$ and as such the minimum amount of parameters that must be learned from the data for a function with a $p-$dimensional input is $p + 1$. In Bayesian inference we derive an expression proportional to the joint posterior distribution of the the parameters θ given the observed data $\mathbf{y}$, $\pi(\theta|\mathbf{y})$ by multiplying together the likelihood of the data given the parameters $L(\mathbf{y}|\theta)$ and the priors for the parameters $\pi(\theta)$. One approach is to then use Markov Chain Monte Carlo (MCMC) to generate samples from $\pi(\theta|\mathbf{y})$. However this requires a great many proposal samples from the posterior to be generated. Consequently if it is computationally expensive to generate one proposal sample, this method becomes impractical, and an alternative is required. A common compromise is to find the mode of the posterior distributions and use these modal values as point estimates for the parameters in the model.

In this dissertation we use the Dice Kriging (Roustant, Ginsbourger, and Deville 2012b) software package, which implements improper uniform priors (Roustant, Ginsbourger, and Deville 2012a) for the θ and as a result $\pi(\theta|\mathbf{y}) \propto L(\mathbf{y}|\theta)$. Consequently maximum likelihood (ML) estimation is used to learn the θ, which is equivalent to finding the maximum posterior mode in the uniform prior case[1]. The observations $\mathbf{y}$ have a joint Gaussian distribution with mean $\mathbf{h}(\mathbf{x}^*)\beta$ and covariance $\mathbf{C} = \sigma^2\mathbf{R}$, where $\mathbf{R}$ is an $n \times n$ covariance matrix with the i, j-th element equal to $r(\mathbf{x_i}, \mathbf{x_j})$ where n is the number of observations and $i \in 1, 2, ..., n$, $j \in 1, 2, ..., n$. Hence the likelihood function to be maximised is

$$L(\mathbf{y}|\theta) = \frac{1}{(2\pi)^{n/2}|\sigma^2\mathbf{R}|^{1/2}} \exp\left(-\frac{1}{2}\left(\mathbf{y} - \mathbf{H}^T\beta\right)^T \left(\sigma^2\mathbf{R}\right)^{-1} \left(\mathbf{y} - \mathbf{H}^T\beta\right)\right) \tag{4.4}$$

Where $\mathbf{H}$ is the $n \times p$ matrix with each row $h(\mathbf{x})$ one of the n observations and each column one of the p parameters. Each time the likelihood is evaluated, the matrix $\mathbf{R}$ needs to be inverted, which requires $\mathcal{O}(n^3)$ computation[2] (Gelman et al. 2013), which means the computational expense grows very rapidly. This, and numerical issues that can arise from inverting $\mathbf{R}$ (see Section 4.5) are the two main hurdles that motivate the use of posterior mode/ ML point estimates of the model parameters, as opposed to 'full Bayesian' distributions.

[1]Provided the domain of the likelihood is the same as, or a subset of, the domain of the prior(s)

[2]The computational expense is proportional to n^3

Equation (4.4) is maximised with respect to β, σ and γ. Analytic expressions for ML estimators for β and σ can be found in terms of $\mathbf{R}$ and the data:

$$\hat{\beta} = \left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{R}\mathbf{y} \tag{4.5}$$

and

$$\hat{\sigma}^2 = \frac{1}{n}\left(\mathbf{y} - \mathbf{H}^T\hat{\beta}\right)^T\mathbf{R}^{-1}\left(\mathbf{y} - \mathbf{H}^T\hat{\beta}\right) \tag{4.6}$$

which look similar to the predictors for the similar terms in ordinary linear model theory, except for the presence of $\mathbf{R}$ because of the correlation of the residuals. The likelihood now need only be maximised with respect to the γ, a slightly easier numerical task. The maximum of the likelihood is then found using a gradient-based numerical optimisation algorithm, with multiple starting points to combat possible multi-modality in the likelihood (Roustant, Ginsbourger, and Deville 2012a). Once we have the parameter estimates we have fully characterised the joint distribution of $\mathbf{Y}^*$ and$\mathbf{Y}^\dagger$. We also have the observed $\mathbf{Y}^* = \mathbf{y}^*$. Consequently we can find the joint conditional distribution $\mathbf{Y}^\dagger|\theta, y^*$, which we use to emulate the simulator at unobserved points.

4.4 Conditioning on the data

Assume that $\mathbf{Y}^*$ and $\mathbf{Y}^\dagger$ in Equation (4.2) each have length one. Then, once we have estimators for the parameters $\hat{\sigma}$, their joint distribution would look something like Figure 4.1, which shows a bivariate Gaussian distribution where the ovals are lines of constant probability density. We can see that there is some positive correlation between these two simulator outputs; this may be because their x-coordinates are close together, or that the covariance hyperparameters $\hat{\gamma}$ for this process imply that there is not a great amount of variation in the function, or both. The left hand plot can be generated solely using the input parameter vectors $\mathbf{x_1}$ and $\mathbf{x_2}$. The right hand plot shows what happens when we observe $Y_1 = y_1$. Our belief about the simulator output at $\mathbf{x_2}$ changes. The joint bivariate density is three-dimensional, and we can imagine slicing it with the plane at $Y_1 = y_1$ and observing the exposed surface, whose dimensions are now the Y_2 axis and probability density. This is the conditional density of Y_2 given $Y_1 = y_1$. For some unevaluated simulator input $\mathbf{x}^\dagger$ with observed simulator outputs $\mathbf{y}^* = \hat{f}(\mathbf{x}^*)$ its mean, $y(\hat{\mathbf{x}^\dagger})$ and variance, $s^2(\mathbf{x}^\dagger)$ are given by

$$\hat{y}(\mathbf{x}^\dagger)|\mathbf{y}^*, \hat{\theta} = \mathbf{h}(\mathbf{x}^\dagger)^T\hat{\beta} + \mathbf{c}(\mathbf{x}^\dagger)\mathbf{C}^{-1}\left(\mathbf{y}^* - \mathbf{h}(\mathbf{x})^T\hat{\beta}\right) \tag{4.7}$$

and

$$s^2(\mathbf{x}^\dagger)|\mathbf{y}^*, \hat{\theta} = c(\mathbf{x}^*, \mathbf{x}^*) - c^T(\mathbf{x}^*)\mathbf{C}^{-1}c(\mathbf{x}^*) + \tag{4.8}$$
$$\left(\mathbf{h}(\mathbf{x}^\dagger) - \mathbf{h}^T(\mathbf{x}^*)\mathbf{C}^{-1}\mathbf{c}(\mathbf{x}^\dagger)\right)^T \left(\mathbf{h}^T(\mathbf{x}^*)\mathbf{C}^{-1}\mathbf{h}(\mathbf{x}^*)\right)^{-1} \left(\mathbf{h}(\mathbf{x}^\dagger) - \mathbf{h}^T(x^\dagger)\mathbf{C}^{-1}\mathbf{c}(\mathbf{x}^\dagger)\right)$$

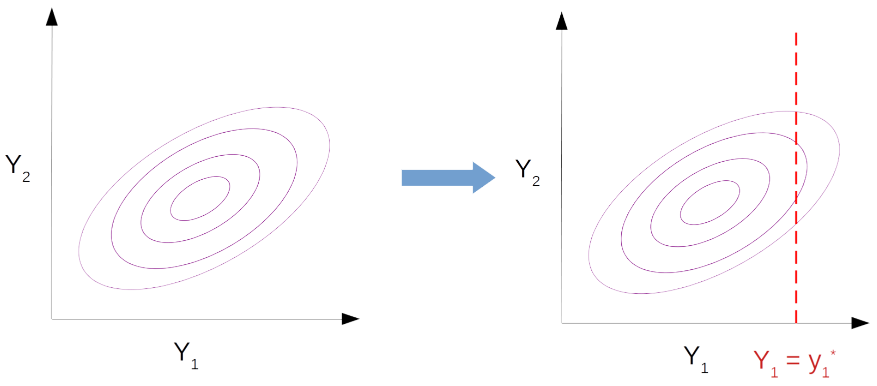


Figure 4.1: Projection of the joint distribution of Y1 and Y2 onto the Y1Y2 plane. The ovals are lines of constant probability density. Once we have observed y1, Y2 has a univariate Gaussian distribution.

where $\mathbf{c}(\mathbf{x}^\dagger)$ denotes the vector $\left(c(\mathbf{x}^\dagger, \mathbf{x}_1^*), c(\mathbf{x}^\dagger, \mathbf{x}_1^*), ..., c(\mathbf{x}^\dagger, \mathbf{x}_k^*)\right)$ where k is the number of observed $(\mathbf{x}^*, y^*)$. So $c(\mathbf{x}^\dagger)$ is the vector of covariances between $\mathbf{x}^\dagger$ and each of the k observed points. Chapter 2 of Reference Oakley (1999) shows how these equations arise from properties of multivariate normal distributions.

If we use Equations @(eq:pred-mean) and @(eq:refpred-var) to try and predict with the input vector $\mathbf{x_j}$ of one of the observed simulator runs then $\mathbf{c}^T(\mathbf{x_j})\mathbf{C}^{(-1)} = \mathbf{C}^{(-1)}\mathbf{c}(\mathbf{x_j}) = \mathbf{e_j}$ where $\mathbf{e_k}$ is the n-vector where the j-th element is one and all the other elements are zero. Consequently

$$\begin{aligned} \hat{y}(\mathbf{x_j}) &= \mathbf{h}(\mathbf{x_j})^T\hat{\beta} + y_j - \mathbf{h}(\mathbf{x_j})^T\hat{\beta} \\ &= y_j \end{aligned} \tag{4.9}$$

and

$$\begin{aligned} s^2(\mathbf{x_j}) &= c(\mathbf{x_j}, \mathbf{x_j}) - c(\mathbf{x_j}, \mathbf{x_j}) + (\mathbf{h}(\mathbf{x}^*) - \mathbf{h}(\mathbf{x}^*))\,\mathbf{h}(\mathbf{x}^*)\,(\mathbf{h}(\mathbf{x}^*) - \mathbf{h}(\mathbf{x}^*)) \\ &= 0. \end{aligned} \tag{4.10}$$

Consequently the observed $\mathbf{y}^*$ are interpolated exactly with zero uncertainty. This is desirable in cases where the data are observed with no error, as is the

Table 4.1: Correlation functions availabe in the software package DiceKriging.

Name	**Expression**
Gaussian	$\exp(-\frac{(x-x')^2}{2\gamma^2})$
Matérn $\nu = \frac{5}{2}$	$\left(1 + \frac{\sqrt{5}\lvert(x-x')\rvert}{\gamma} + \frac{5(x-x')^2}{3l\gamma 2}\right) \exp\left(-\frac{\sqrt{5}\lvert(x-x')\rvert}{\gamma}\right)$
Matérn $\nu = \frac{3}{2}$	$\left(1 + \frac{\sqrt{3}\lvert(x-x')\rvert}{\gamma}\right) \exp(-\frac{\sqrt{3}\lvert(x-x')\rvert}{\gamma})$
Exponential	$\exp\left(-\frac{\lvert(x-x')\rvert}{\gamma}\right)$
Power-Exponential	$\exp\left(-\left(\frac{\lvert(x-x')\rvert}{\gamma}\right)^p\right)$

case with a deterministic computer simulation. Intuitively we would expect this to be true as Equations @ref{eq:pred-mean} and (4.8) define the conditional distribution of $Y(\mathbf{x}^\dagger)|y^*$ and the distribution of $Y(\mathbf{x}^*)|\mathbf{y}^*$ is just $\mathbf{y}^*$. Stochastic behaviour can also be encapsulated by estimating a 'nugget' parameter form the data, which we discuss next, along with other properties of the covariance function.

4.5 Covariance function

The covariance between y_i and y_j is computed from the input parameters $\mathbf{x_i}$ and $\mathbf{x_j}$ and it is a key assumption of Gaussian processes that points that are close together in $\mathbf{x}$-space should be close together in y-space, and that if we wish to predict $Y^\dagger$ for some $\mathbf{x}^\dagger$, then observed points close to $\mathbf{x}^\dagger$ should tell us more about $Y^\dagger$ than those that are far away. The hyperparameters γ encapsulate this idea. The function $c(\cdot,\cdot)$ must produce covariances matrices for Y that are consistent with a multivariate normal distribution.

- Covariance matrices are symmetric in that for matrix $\mathbf{A}$, $\mathbf{A} = \mathbf{A}^{-1}$ and in particular $\mathbf{A}_{i,j} = \mathbf{A}_{j,i}$ where i and j index the rows and columns of $\mathbf{A}$. Consequently the covariance function $c(\cdot, cdot)$ must also be symmetric in that $c(\mathbf{x}, \mathbf{x}') = \cdot(\mathbf{x}', \mathbf{x})$.
- Covariance matrices are positive semidefinite (PSD), which means they have no negative eigenvalues. Hence $c(\cdot,\cdot)$ must produce covariance matrices which are PSD.
- A covariance function that is a function of $\mathbf{x} - \mathbf{x}'$ is stationary, in that the covariance between two points does not depend on where they are in $\mathcal{X}$ but only the distance between them.
- Furthermore a covariance function that is a function of $|\mathbf{x} - \mathbf{x}'|$ is isotropic.

If a covariance function is stationary, then we can express it as a product of a correlation matrix $\mathbf{R}$ and a constant variance σ^2. Table 4.1 shows the covariance functions available in DiceKriging (Roustant, Ginsbourger, and Deville 2012a), or more appropriately correlation functions, as each is multiplied by σ to get the required covariance. All of the covariance functions in Table 4.1 produce symmetric, PSD matrices, and they are all stationary and isotropic.

The Matérn correlation function with parameter $\nu = \frac{5}{2}$ is a popular choice and is the default option in both the DickeKriging and RobustGaSP (Gu, Palomo, and Berger 2022) software packages. The *nu* parameter in the Matérn family controls the smoothness of the function. As $\nu \to \infty$, the Matérn function begins to look more like the Gaussian covariance function, which is infinitely differentiable and can hence be unrealistically smooth. For $\nu = \frac{5}{2}$, the process is mean square twice differentiable, which guarantees the existence of first and second moments, and hence predictive means and variances, which is desirable. Achieving the same level of smoothness with the squared exponential kernel would come at the price of correlations quickly going to zero as $|\mathbf{x} - \mathbf{x}'|$ increases (Gu, Palomo, and Berger 2022). In this work we use the Matérn $\frac{5}{2}$ correlation function.

The correlation functions $r(\cdot,\cdot)$ in Table 4.1 take two scalar inputs and return a single scalar output[3]. To extend this to compute correlations for multivariate $\mathbf{x}$ the most common approach is to take the product of the univariate correlations:

$$\mathbf{c}(\mathbf{x}, \mathbf{x}') = \sigma^2 \prod_{i=1}^{d} c(x_i, x_i') \tag{4.11}$$

where d is the number of elements of $\mathbf{x}$. For each of element of $\mathbf{x}$ a separate γ is learned, and that controls how influential that input parameter is on determining the value and the precision of the emulator prediction at other points. The γ are often called length-scale parameters. As we predict at points in $\mathcal{X}$ further away from the observed $\mathbf{x}^*$, the predictive variance will tend to σ^2. If we changed γ from 1 to 0.5 for x_i, we would expect to see the same amount of variation in a fixed interval double along the x_i axis. Their is an interplay between the γ and σ, The smaller the γ are the more the function, and hence σ^2 would be smaller, as large fluctuations in y can be explained by function variation rather than noise. Higher values for γ are associated with higher values for σ^2 and imply a constant function with a lot noise. Lower values for γ are associated with lower values for σ^2 and imply a white-noise process (Williams and Rasmussen 2006).

Often the function $\hat{f}(\cdot)$ being emulated is non-deterministic. In this case, it is possible to learn another parameter from that data, called the 'nugget', which is the variance of a zero-centred Gaussian distribution. The univariate nugget is added to the diagonal elements of the covariance matrix for the observed data and the emulator is no longer constrained to interpolate the observed $\mathbf{y}^*$. The nugget can also be used to address problems with inverting $\mathbf{C}$ in Equation (4.4) which can arise when its elements are very close to 0 or 1. In this dissertation we allow for a nugget parameter to help with likelihood estimation.

There is a lot of flexibility in choosing covariance functions and they can be customised to reflect prior knowledge of the function being emulated. In particular, any sum or product of valid covariance functions is also a valid covariance

[3]A function that does this is called a kernel, and hence the $r(\cdot,\cdot)$ are sometimes called kernels

function (Williams and Rasmussen 2006). The covariance function is the defining element of a GP, but emulation is often boosted by the inclusion of a mean function, which we illustrate now with a simple example, along with some other properties of the GP discussed in this section.

4.6 Mean function

We can think of a GP of telling us how far we expect the true process to deviate from some given regression function. Any variation in the true process not captured by the regression function needs to be captured by the GP, which can often lead to less precision in emulator predictions. We illustrate this with an example. Assume we wish to emulate the function $y = x^2 - 5x$ shown in Figure @ref(fig:gp_example_data) using a GP. We are only able to observed the five points shown. There is a region with no observed points, roughly between -9 and 0 on the x-axis. The emulator will have to interpolate in this region. There are no observations in the region above about 6 on the x-axis, the emulator will have to extrapolate in this region. Luckily, we have a good amount of data around the point of inflection at $x = 2.5$. This is where the function varies the most.

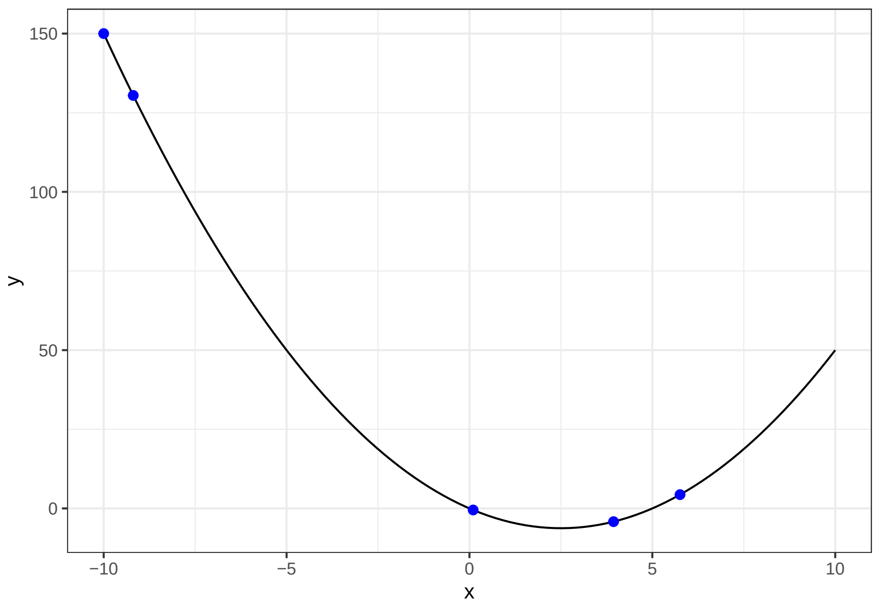


Figure 4.2: A simple function maths expression to be emulated, with the five points used to build the emulator shown. There is a region with no training data approximately maths expression.

We first build a zero-mean GP emulator, using a Matérn $\frac{5}{2}$ correlation function, without a 'nugget'. We then use the emulator to predict, with 95% posterior predictive intervals, the value of the function between x = -10 and 10. We then do the same again, but this time allow a simple first-order[4] mean function to be

[4]as in $y = mx + c$

estimated from the data. The results are shown in Figure **??**. The predictive means of both emulators do a good job of reproducing the function and both emulators have larger predictive intervals further away from the data. However, the constant-mean emulator (right pane) shows greater precision in predictions.

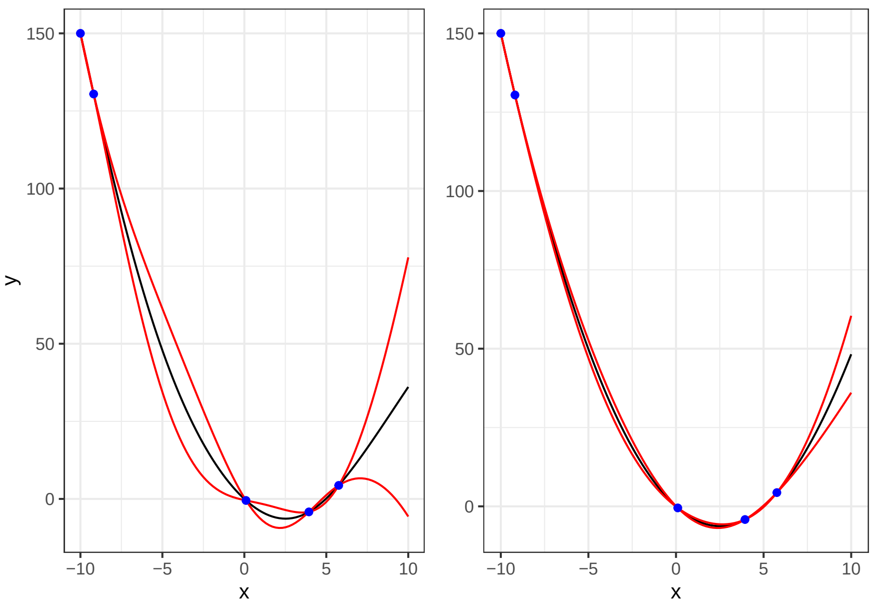


Figure 4.3: Predictions using a zero-mean (left) and first-order mean (right) GP emulator trained on the five points shown for the function, with 95 percent predictive intervals shown. The emulators predict the training points exactly, and the width of the predictive intervals increases further from the data. The first-order mean emulator is able to interpolate and extrapolate with greater precision.

As the emulator tries to predict at points further away from the training data, the influence of the training data on the predictions drops off, and the emulator reverts to the mean function, as show in Figure **??**, where the zero-mean emulator on the left tends to zero with larger x, and the constant-mean emulator on the right tends to the average of the training data, albeit both with such wide posterior predictive intervals as to render the results quite useless.

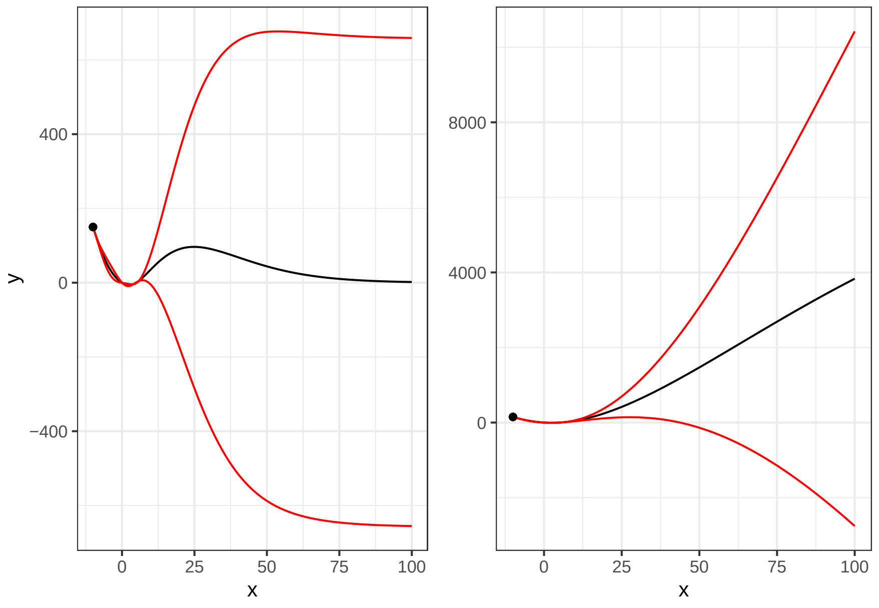


Figure 4.4: Behaviour of the emulator as it predicts further from the training data. Both the zero-mean (left) and first-order mean (right) emulators tend to their respective mean functions as the distance from the observed data grows larger.

Theoretically, the mean function can be any linear (in the coefficients) model of the inputs **x**. Some sources (Erickson, Ankenman, and Sanchez 2018) favour a mean-only model, describing the estimation of coefficients for higher order polynomials as possibly extraneous. Prior knowledge of the process to be emulated can be useful in choosing a suitable mean function. In this dissertation, where, 28 input parameters were being varied, and knowledge of the relationship between them and the simulator output was limited, a mean function of the kind

$$\mu(\mathbf{x}) = \beta_0 + \sum_{i=1}^{28} \beta_i x_i \tag{4.12}$$

was used, which meant that 29 parameters needed to be learned for the mean function. This was a practical choice - each emulator was trained using several hundred runs. Introducing second order terms may have required many more simulator runs to get good reliable estimates for the β, and the computational resource required for this was not available. The choice of mean function, and other modelling choices, may also be a function of the software package used, and we discuss the choice of software implementation in the next section.

4.7 Software implementations

Three software packages were tried to do the GP emulation in this dissertation, *RobustGaSP* (Gu, Palomo, and Berger 2022), the *scikit-learn* (Pedregosa et al. 2011) module *GaussianProcessRegressor*, and *DiceKriging* (Roustant, Ginsbourger, and Deville 2012b).

The *RobustGaSP* package distinguishes itself from the other two by implementing a non-uniform prior for the GP parameters and using the maximum posterior modes as point estimates for the parameters. This approach is supposed to protect against some of the numerical issues arising from parameter estimation (discussed briefly in Section 4.3). However, this software package was very slow to run, and as such it was impractical for training many hundreds of emulators at a time. Crucially, it did not expose a public method for generating a predictive covariance matrix, which was crucial for history matching, as discussed in Section ???.

The *scikit-learn GaussianProcessRegressor* module was the only Python module examined. It was very quick, and appeared to offer the greatest flexibility in building custom covariance functions. However, it failed very often in parameter estimation, and it only offered two mean function options: zero-mean or constant (equal to the mean of the training data), which may have been the cause of its problem in parameter estimation.

In this dissertation, we focussed on training many hundreds of GP emulators for processes with unknown behaviours. As such, it was impractical to have to wait a long time for parameters to be estimated, or to spend much time examining the impacts of different covariance functions. The preferred method was to build the emulators with sensible defaults, and to conservatively filter out models with poor generalisation performance. It was also crucial to be able to easily generate predictive covariance matrices. Consequently, the *DiceKriging* package was used, as it was relatively fast, and could compute covariance matrices using its 'predict' method.

The order of the information presented in this chapter was not indicative of the timeline of the modelling process, so we end with brief summary of GP modelling considerations, and a reminder of the purpose of GPs in this dissertation.

4.8 Conclusion

In this chapter we discussed some fo the theory and the main building blocks of GPs, while hopefully helping to build some intuition. In summary, once we have collected our data on the function we wish to emulate we

1. Choose a structure for the mean and covariance function,
2. Compute the parameters for the emulator by maximising the likelihood of the data with respect to the parameters,
3. We now have a model for the joint distribution of the unobserved and observed points, which means we have a model for the conditional distribution of the unobserved points, conditional on the observed points. It is this conditional distribution we use to predict at the unobserved points.

The prediction at an unobserved point is a Gaussian random variable, which means that we have a point estimate for the function there (the mean) and a quantification of the uncertainty of that point estimate (the variance). We can obtain these estimations in a fraction of the time it would take to run the simulator. These properties make a GP emulator a vital tool in history matching, which we discuss in the next section.

Chapter 5

Results

5.1 First wave methodology

A total of 28 active input parameters were considered. The design grid for the first wave training runs were generated as follows. A 300 point maxmin Latin hypercube design was generated on $U[0,1]^{28}$. These points then underwent a linear transformation onto $U[-1,1]^{28}$ and finally were exponentiated with base ten. This enabled exploration of the design space both above and below the default parameter values.

The simulator runs were carried out in parallel across 15 nodes on a personal laptop. The 300 simulator evaluations took approximately 36 hours to complete. The simulator produced cross-section spectra across the four of the reactions for which relevant experimental data existed, as shown in Table 2, where it can be seen that six of the runs failed to complete successfully for the reactions (n,a), (n,n') and (n,p) (see Section 2.4 for an explanation of this notation).

Individual univariate Gaussian process emulators were built for each of the 593 energy points for each of the four reactions in common between the observed and simulated cross-section data sets, 2372 models in all. This approach was inefficient as there did not exist observations at each of the 593 energy points across all four reactions, and much compute was wasted training and evaluating

Table 5.1: Counts of cross-section spectra generated by TALYS across 300 runs. Not all simulator runs were succesful, but this does not prevent history matching from being carried out.

Reaction	count
(n,a)	294
(n,n')	294
(n,p)	223
(n,tot)	300

models for which implausibility measures could not be computed. This was addressed in subsequent waves. The emulators used a first order linear mean function, which required learning 29 parameters from the data. Constant and 0 mean functions were also examined, but these led to an unacceptably large number of model failures arising from singular posterior correlation matrices. A Matern correlation function with smoothness parameter $\frac{5}{2}$ was used. The models were trained allowing a residual nugget to be estimated. This required learning a further 30 hyperparameters; 28 length-scale parameters, one variance and one residual nugget. None of the models failed to build. The inputs were transformed back to [-1,1]. The outputs were left unscaled in order to avoid potential errors when scaling the error terms for the implausibility measures.

A further set of 30 test runs were carried out, with the design matrix being generated using the same principles used to generate the training designs. The emulators were validated by predicting at the values of the inputs for the training runs and then computing the Mahalanobis distances between the predicted outputs and the simulator outputs. If the assumptions of the Gaussian process emulator are valid, these distances are hypothesised to have a scaled-$F_{m,n-q}$ distribution, where m is the number of test points (30), n is the number of training points, which varies across models, and q is the number of parameters in the emulator mean function (29). The distance metrics were compared to critical values corresponding to the one and 99 percentiles of the appropriate F distributions and the models were discarded if they lay outside this 98% probability interval. Of the 2372 models, 739 were discarded this way, leaving 1633.

Figure **??** illustrates the importance of carrying out evaluation. The Figure shows evaluation run results for an emulator for the simulated cross-section at 6.19 MeV for the reaction (n,p). The vertical lines show the two standard deviation predictive intervals for the emulator and the line x=y is plotted for reference. Also shown, at the intersection of the horizontal and vertical lines is the nominal measured cross-section value at 15.2 barns. The job of the emulator is to predict 15.2 barns at the same active parameter values as the simulator. From the plot, it appears that, if the emulator consistently under-predicts the emulator in this region and if that pattern continued, the emulator would predict that the cross-section would be close to 15.2 barns in a region of parameter space where the simulated cross-section would be considerably higher. As a consequence of this, a non-implausible value of the active parameters could be rejected as implausible, and the opportunity to run the simulator at this input and better emulate close to the observed value could be lost. Consequently, it is important to carry out validation on the emulators to ensure good out-of-sample predictive performance.

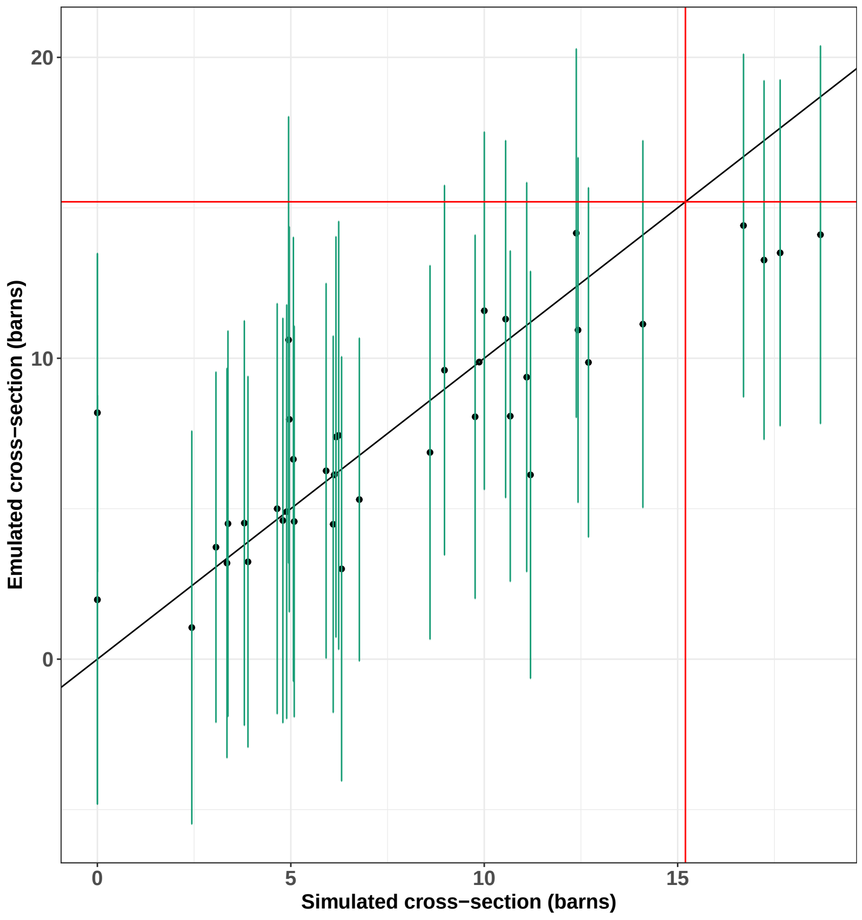


Figure 5.1: Validation run results for an emulator for the cross-section for the (n,p) reaction at energy 6.19 MeV. The vertical lines show the two standard deviation predictive intervals of the emulator and the x=y line has been plotted for reference. The observed cross section value is shown as the intersection of the vertical and horizontal line at (15.2,15.2). It appears that the emulator would underpredict the simulator output at this point.

5.2 First wave analysis

The computed Mahalanobis distances for the accepted models for (n,tot) are plotted in Figure **??**fig:mahalanobis). The hypothesised scaled $F_{30,271}$ curve is plotted over the top. The distribution of distances should resemble this curve under Gaussian process emulator assumptions. The histogram and the curve appear to have reasonably similar densities, although the mode is shifted slightly right.

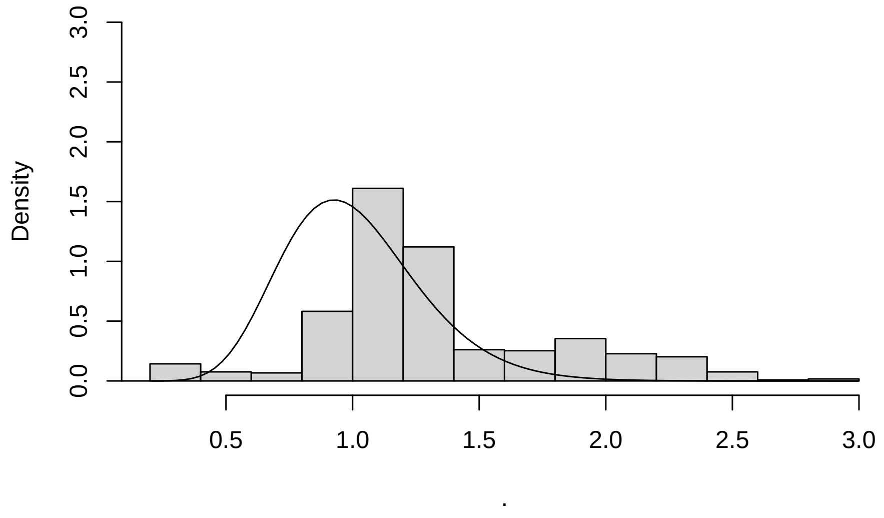


Figure 5.2: Histogram of Mahalanaobis distances for valid (n,tot) models. The hypothesised scaled-$F_{30,271}$ curve is plotted over the top. The distribution of Mahalanobis distances looks reasonably similar to the hypothesised distribution.

Of the 1633 validated models, 464 had corresponding observations allowing for implausibility analysis. Further waves trained models only at energy/ reaction combinations for which observations existed. The 464 models were used to find non-implausible design points for Wave 2. The number of potential design points that could be evaluated was limited by memory and compute power. A total of 142860 random points in the 28 dimensional design space were selected. Implausibility metrics were generated for each of the 464 models for each of the 142860 proposal points. In the implausibility analysis the assumption was made that observation uncertainty, $\sqrt{V_j^{(obs)}}$ for observation z_j was $0.1z_j$ and that the simulator inadequacy, $\sqrt{V_j^{(s)}}$ for simulator output $f_j(\mathbf{x_i})$ was also $0.1z_j$, where $j \in 1, 2, ..., 464$ indexes over the observations/ emulator pairs and $i \in 1, 2, ..., 142860$ indexes over the proposal points, implying that the simulator inadequacy is independent of the design point at which the simulator is evaluated. The third type of uncertainty considered was emulator uncertainty, $\sqrt{V[f_j(\mathbf{x_i})]}$ corresponding to mean emulator output $E[(f_j(\mathbf{x_i})]$. Hence the implausibility measure for proposal point i computed using simulator j is

$$I_{i,j} = \frac{|z_j - E[f_j(\mathbf{x_i})]|}{\sqrt{(V_j^{(obs)} + V_j^{(s)} + V[f_i(\mathbf{x_j})])}}. \tag{5.1}$$

Equation (1) was evaluated 464 times for each proposal point. The second largest of the 464 measures $I_i^{(2M)}$ for a proposal point $\mathbf{x_i}$ was used as the first implausibility metric for the proposal point. Another multivariate implausibility

Table 5.2: Number of proposal points accepted as a function of different cutoffs for implausibility metrics. I^{nM} indicates the nth largest univariate implausibility measure for that proposal point - the number of chosen points is more sensitive to the choice for this metric than to the choice of chi-squared cut-off percentile.

	$I_i^{(1M)}$	$I_i^{(2M)}$	$I_i^{(3M)}$
$\chi^2_{464,.95}$	0	268	4381
$\chi^2_{464,.99}$	0	276	4472

was also considered:

$$I_i = (\mathbf{z} - E[f(\mathbf{x_i})])^T \left(V_j^{(obs)} + V_j^{(s)} + V[f_i(\mathbf{x_j})]\right)^{-1} (\mathbf{z} - E[f(\mathbf{x_i})]) \tag{5.2}$$

where $\mathbf{z}$ and $E[f(\mathbf{x_i})]$ are j-vectors with the 464 observations and their corresponding emulator predictions respectively, and $V_j^{(obs)}$, $V_j^{(s)}$ and $V[f_i(\mathbf{x_j})]$ are now all covariances matrices. The simple approach was taken in assuming that the outputs are all uncorrelated. In this case, all three covariance matrices are diagonal, with the square of the denominator in Equation (1) making the j, j-th element of the matrix. Taking this approach, I_i for proposal point i is simply $\sum_{j=1}^{464} I_{i,j}^2$.

Two criteria were used for accepting a proposal point $\mathbf{x_i}$, First that $I_i^{(2M)} < 3$ and second that $I_i < \chi^2_{464,.95}$. 272 points met these criteria - 0.019% of the proposed points. The decision on how many points to examine, and what proportion of these to accept, was driven chiefly by practical considerations. The lists of implausibility measures for all the proposal points took up 5.4GB in memory, bringing an 8GB laptop close to capacity, and based on the 36 hour run time of the 300 wave one runs, a similar number of accepted points was desired to allow the analysis to be carried out in a reasonable amount of time. Given the size of the input space it would be desirable to generate a much larger proposal sample if resources allowed. Different acceptance criteria for the two implausibility measures were examined; Table 3 shows some of the results of this analysis, where it can be seen that the choice of maximal implausibility was the most important decision in determining the size of the non-implausible design space.

5.3 Minimum implausibility and optical depth analysis

Some sense of the sensitivity of implausibility to the inputs can be gained from examining plots such as those in Figure 5.3. The left column shows minimum implausibilities and the right column shows plots of probabilities for two of the inputs. The minimum implausibility plots were generated by subdividing the $[-1, 1]^2$ grid for two of the input parameters into 100 blocks. Each proposal input was then put into one of these intervals according to its values of the two relevant inputs. The minimum univariate implausibility measure (Equation (3.6) in each interval was then pulled out and used to determine the hue in that interval in the heat map. If the value of the minimum implausibility was high, this might indicate that it is not possible to get TALYS to match experimental measurements if the input parameters are set to those values. The optical depth plots are constructed by grouping the proposal inputs in the same way. An estimate of the probability of finding a non-implausible input in an interval is then computed as the ratio of the number of non-implausible inputs to the total number of proposal inputs in that interval. Taken together, the plots help in visualising the sensitivity of TALYS to different values of the inputs.

The code used to generate these plots is shown in Appendix B.

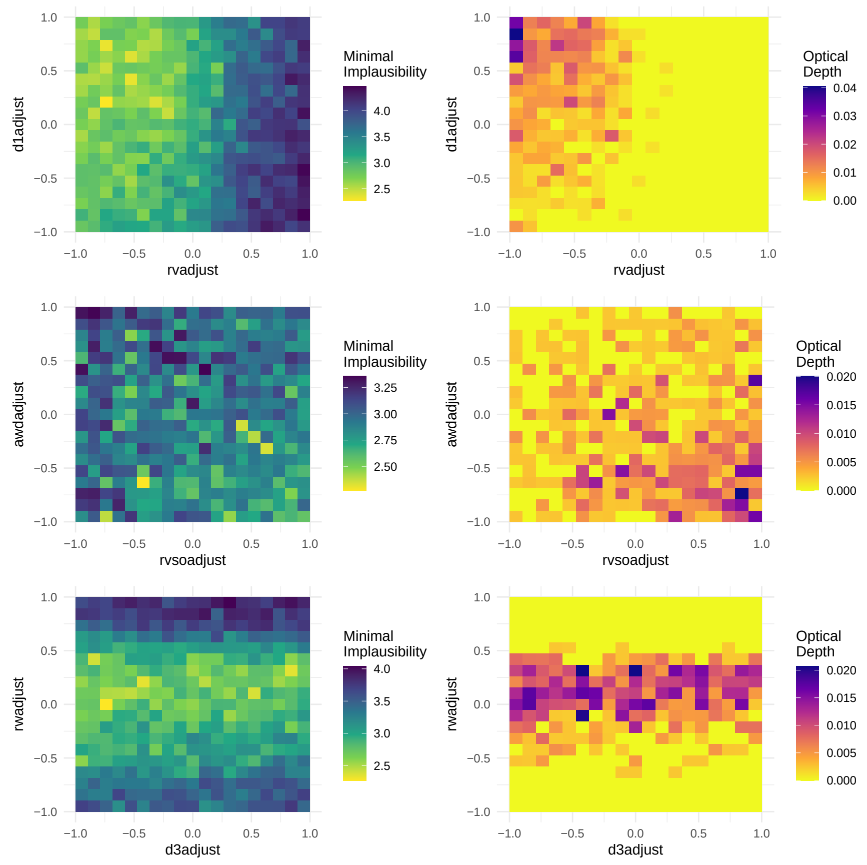


Figure 5.3: Minimum implausibility plots (left column) and optical depth plots (right column) for six of the inputs. The minimum implausibilty plots show estimates of the minimum univariate implausibility for values of two of the inputs and the optical depth plots show estimates of the probability of finding a non-implausible input for values of two of the inputs. Note that the heatmaps are not all on the same scale.

The first row of Figure 5.3 shows implausibility and optical depth plots for parameters ‘d1adjust’ and ‘rvadjust’. These plots indicate that larger positive values for ‘rvadjust’ are more likely to cause TALYS to provide a poor match to observations, and that values below zero (that is multipliers below one, so values for ‘rvadjust’ less than the default) are more likely to produce non-implausible results. The optical depth plot indicates that there might be some interaction effect between ‘d1adjust’ and ‘rvadjust’, as indicated by the higher probability region in the top left corner of the plot. The second row of the figure shows plots for parameters ‘awadjust’ and ‘rvsoadjust’. This plot also suggets that there may be an interaction between the two parameters, inputs with values of ‘awadjust’ close to -1 and ‘rvsoadjust’ close to +1 being more likely to produce non-implausible outputs. The bottom row shows plots for the parameters ‘rwadjust’ and ‘d3adjust’ and suggests that values of ‘rwadjust’ close to the TALYS default are more likely to give non-implausible outputs, and that perhaps the outputs are not very sensitive to the value for ‘d3adjust’.

5.4 Subsequent waves

The 276 non-implausible inputs chosen n Section **??** were used as the design for the training runs in Wave 2. 593 emulators were built, one for each experimental data point. A further set of test runs of the simulator were carried out. The active input variable settings for these test runs were also chosen by generating a large number of proposal samples and evaluating their implausibility using the Wave 1 emulators. The emulators were then validated in the same way as described in Section **??**. 168 were discarded, leaving 425 emulator-measurement pairs with which to carry out implausibility analysis for Wave 2. Less computational resource was available at the time of the Wave 2 implausibility analysis, and consequently only 35715 proposal points could be examined. Of these, 18043, which is just over 50%. The cut-offs used for the implausibility measures were the same as those used in Section **??**, although the relevant χ^2 distribution from which the 99th percentile was taken was different, due to their being a different number of validated model between the waves (see Section 3.7.

Having such a large number of non-implausible points for Wave 2 presented a new challenge, as it was impossible to run TALYS for all 18043 non-implausible inputs in an acceptable time interval. Consequently a subset of the inputs had to be selected. It was decided to run the simulator at 600 training points for Wave 3 (plus 60 test points). Instead of randomly sampling 600 of the 18043 non-implausible proposals, an attempt was made to find an optimal set of points by maximising the minimum distance between 28 input coordinates. This was carried out by first choosing a random sample of 600 points and computing their distance matrix (using the R function ‘dist’). The minimum value in the matrix was extracted. This process was repeated for 10^5 random samples of 600. The sample that had the maximum minimum distance was used as the design for Wave 3, which was a way of ensuring that the design points were as spread out across the non-implausible volume as possible.

Table 5.3: Summary of history matching waves. Column two indicates how many simulator runs were used to build the emulators at each wave. Columns three and four are the cutoff values for the univariate and multivariate implausibility measures used in each wave. Column five indicates the proportion of the original design space evaluated as non-implausible after each wave.

Wave	Runs	$I_i^{(2M)}$	$I_i^{(MV)}$	% of Original Space
1	300	3	$\chi^2_{464,0.99}$	0.19
2	276	3	$\chi^2_{425,0.99}$	0.09
3	600	2	$\chi^2_{440,0.95}$	0.03

The same for building and validating the emulators was carried out for Wave 3, in which 440 emulators were assessed as valid. In Wave 3, the cut-offs of the implausibility measures were changed form previous waves, as the ability of the emulators to refocus with the Wave 1 and 2 cut-offs was diminished. Table **??** shows a summary of the number of training runs and the values of the cut-offs used in each wave. The table also shows the proportion of the original input space evaluated as non-implausible after each run. This is computed as the proportion of proposal samples evaluated as non-implausible in the run multiplied by the proportion of the original space deemed non implausible in the previous run. For example, in Wave 2, $18043/35715 = 0.505$ of the runs were evaluated as non implausible, multiplied by 0.19 for Wave 1 gives 0.09. Implicitly, the proportion of the input space that is plausible before Wave 1 (wave 0) is 1. From Table **??** it can be seen that it was possible to go from a state of complete ignorance about the values of the state parameters to a subset of values 0.03% the size of the original volume for the price of about 1300 TALYS runs.

In Wave 3 3572 proposal inputs were examined, and 1115 were evaluated as non-implausible, about 31%. This was the final wave, and consequently the 1115 were the final non-implausible inputs. The joint distribution of these samples contained useful information about the relationship between the active parameters and the TALS outputs and could also be used to allow sampling from the conditional distribution of the parameters given the data, as discussed in the next Section.

5.5 Posterior sampling of active inputs

Eventually the iterative history matching process must stop. This might be because there is enough evidence to suggest that further waves will not reduce the non-implausible input space much further, or that there is a suitable probability of choosing a non-implausible random sample from the remaining active parameter space, such as in (Andrianakis et al. 2015). In this dissertation the process was stopped as all the time/compute resource available was used up. Results that arise in this way can still be useful due to the rapid reduction of the plausible parameter space that occurs in early waves, as can be seen in Table 5.4.

The endpoint of nuclear data evaluation is a set of multivariate normal distributions, one for each reaction, describing a suite of cross-sections on a fine energy grid. Consequently, the non-implausible active parameter space, denoted $\{\mathbf{x_{ni}}\}$, is of secondary interest, the primary interest is in the implication for the distribution of non-implausible TALYS outputs. One way to learn about the distribution of non-implausible TALYS outputs would be through Monte Carlo (MC) sampling, where samples are drawn from the posterior distribution of active parameters and TALYS is run for each sample. The resulting TALYS outputs can be used to learn the properties of the required multivariate normal distribution, and should converge in distribution towards it. This requires being able to generate samples from the required posterior. The non-implausible parameter space is not an estimate of the posterior distribution of $\mathbf{x}$, denoted $\pi(\mathbf{x}|z)$, but it does contain it. Consequently it was possible to sample from it, using the following method, taken from (**eremy_histmatch?**).

A proposal density was required to allow random sampling of proposal posterior samples. A sensible choice was

$$P(\mathbf{x}) = N\left(\mathbf{x}|\hat{\mu}_x, \kappa\hat{\mathbf{\Sigma}}_x\right) \tag{5.3}$$

where $\hat{\mu}_x$ and $\hat{\mathbf{\Sigma}}_x$ were the sample mean and covariance matrices of $\{\mathbf{x_{ni}}\}$. Random samples were drawn from Equation (5.3), each one a proposal for a sample from $\pi(\mathbf{x}|z)$. The constant κ was used to 'widen' the search area, which has been seen to make the posterior sampling more effective (Andrianakis et al. 2015). The first step was to draw a large set of proposal samples from Equation (5.3). In this dissertation, this was done with the aid of the R package 'mvtnorm' (Genz et al. 2021). κ was set equal to 2.

The likelihood of the data with respect to the proposal samples needed be computed. As an approximation to the likelihood

$$L(\mathbf{x}) = p\left(\mathbf{z}|\mathbf{x}\right) = N\left(\mathbf{z}|E[g(\mathbf{x})], V(\mathbf{x})\right) \tag{5.4}$$

was used.

Each time a proposal was generated from $P(\mathbf{x})$, it was used to predict the observations $\mathbf{z}$ using the appropriate GP emulator. The mean vector in Equation (4.4) was the vector of predictive emulator means at $\mathbf{x}$ corresponding to the observed $\mathbf{z}$. The covariance matrix was constructed by putting the sum of the emulator predictor variance, the observation uncertainty and simulator inadequacy as the diagonals, and zeros for the off diagonals. The likelihood of the data was then computed using 'mvtnorm'. This is the multivariate equivalent of using the 'dnorm' function from base R, that is the probability density, $L(\mathbf{x})$ with respect to Equation (4.4) was computed with respect to $\mathbf{z}$.

The likelihood $p(\mathbf{x})$ was also computed with respect to $\mathbf{x}$, and a weight $w(x) = \frac{L(\mathbf{x})}{p(\mathbf{x})}$ was assigned to $\mathbf{x}$. $L(\mathbf{x})$ is a measure of the probability of observing the data given the proposal $\mathbf{x}$ and $p(\mathbf{x})$ as the probability of having proposed $\mathbf{x}$ in the first place.

Once weights have been computed for all the proposal $\mathbf{x}$, if the proposals are randomly sampled with respect to their weights, this is approximately equivalent to sampling from the posterior $\pi(\mathbf{x}|\mathbf{z})$. The weighted sampling was achieved using the R function 'sample' with the 'prob' option set equal to the vector of weights.

This approach requires assuming that the simulator inadequacy and observation uncertainty terms were normally distributed, which is a common assumption, and is implied in Section 3.5. Implemented in this way, the sampling method also requires that the prior distribution of $\mathbf{x}$ is constant over the likelihood domain. This assumption is often reasonable giving how small the non-implausible space is with respect to the original parameter space after history matching.

In this dissertation, implementing this method proved challenging. Evaluating $L(\mathbf{x})$ almost always led to values below machine precision, effectively zero. One possible reason for this is that the history matching stopped at wave 3, and the non implausible space was not yet small enough to allow the emulators to predict very well. Hence the likelihood of the data given the parameters was vanishingly small. It may have also been due to the number of relevant observations, 440 for Wave 3. Given that the probability density must integrate to one it was perhaps not unusual that many instantaneous values for the density function of a 440-dimensional Gaussian would be vanishingly small. Some investigation revealed that reducing the number of relevant data points did indeed result in non-zero results for $L(\mathbf{x})$. Consequently it was decided to generate posterior samples of $\mathbf{x}$ with respect to a subset of the observations $\mathbf{z}$. As the elements of $\mathbf{z}$ each corresponded to a different reaction (see Section 2.4), a natural way to subset the observations was by reaction. From Section 4.2, any subset of a multivariate normal distribution is itself normal, consequently any marginal likelihood of Equation (4.4) is also normal, and it was possible follow the procedure to generate samples from the marginal posterior densities. This also provided an interesting way to investigate if different values of the parameters were better at reproducing different reactions. This sampling approach generated samples from the posterior that were non-zero outside of the allowable range of values ([-1,1] on the log scale). This was addressed by implying a uniform prior on [-1,1] and 0 otherwise, which translates in practical terms to zero-weighting any posterior samples generated outside the allowable. range

Plots for the posterior samples generated in this way are shown in Figures **??** and **??**. The likelihood was evaluated with respect to experimental data for these reactions for which there were valid emulators, that is for three data points for reaction (n,a) and fifty data points for reaction (n,p). The posterior samples are fairly uniform across most of the parameters, which may indicate that the two reaction types are fairly insensitive to the settings of most of the parameters, at least at the observed energy states, or that further waves of history matching may need to be carried out in order to build better emulators and increase the 'signal' in the posterior sampling. However, in both Figures, it is evident that lower values of rvadjust and rwdadjust are favoured, and this is particularly evident in Figure ??. Since there is evidence of this in both Figures, there is a good chance that lower values of these parameters are characteristics of the scenario being examined, neutrons incident upon Iron-56, rather than one of the reactions that can occur under this scenario. Contrastingly, the (n,a) reaction appears not to favour any particular values for the v3adjust parameter, whereas the (n,p) reaction appears to favour higher values, which could indicate that the parameter is of particular importance in modelling that reaction.

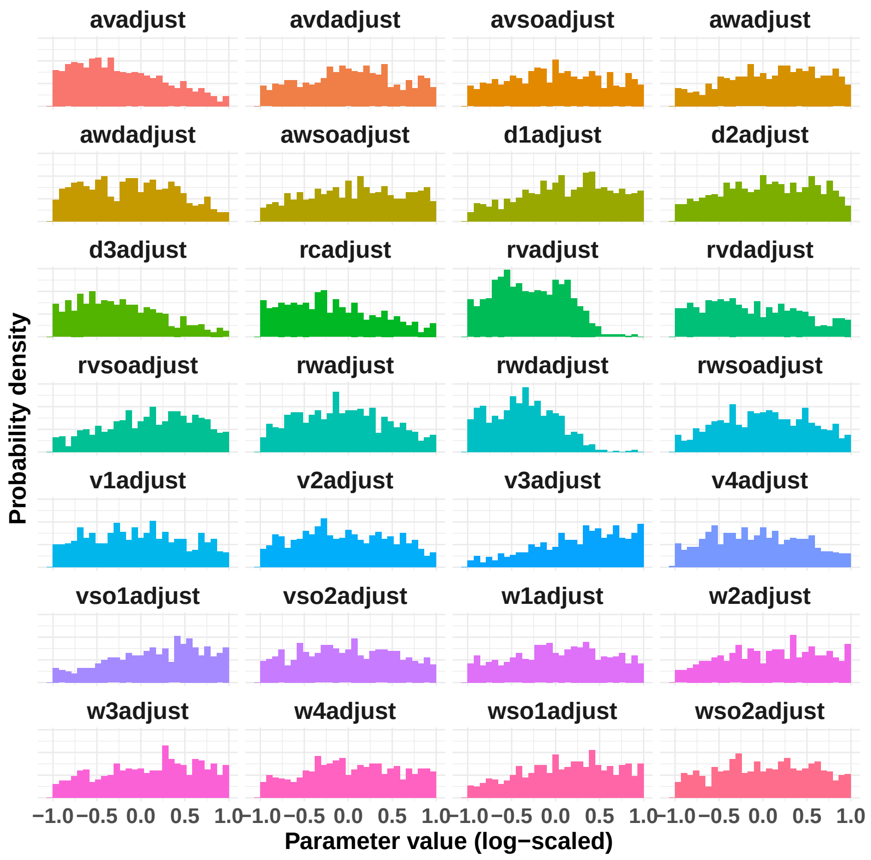


Figure 5.4: Posterior samples for active input parameters when the approximate likelihood is evaluated with respect to (n,p) experimental data. This reaction appears to favour lower values for parameters rvadjust and rwdadjust and higher values for v3adjust.

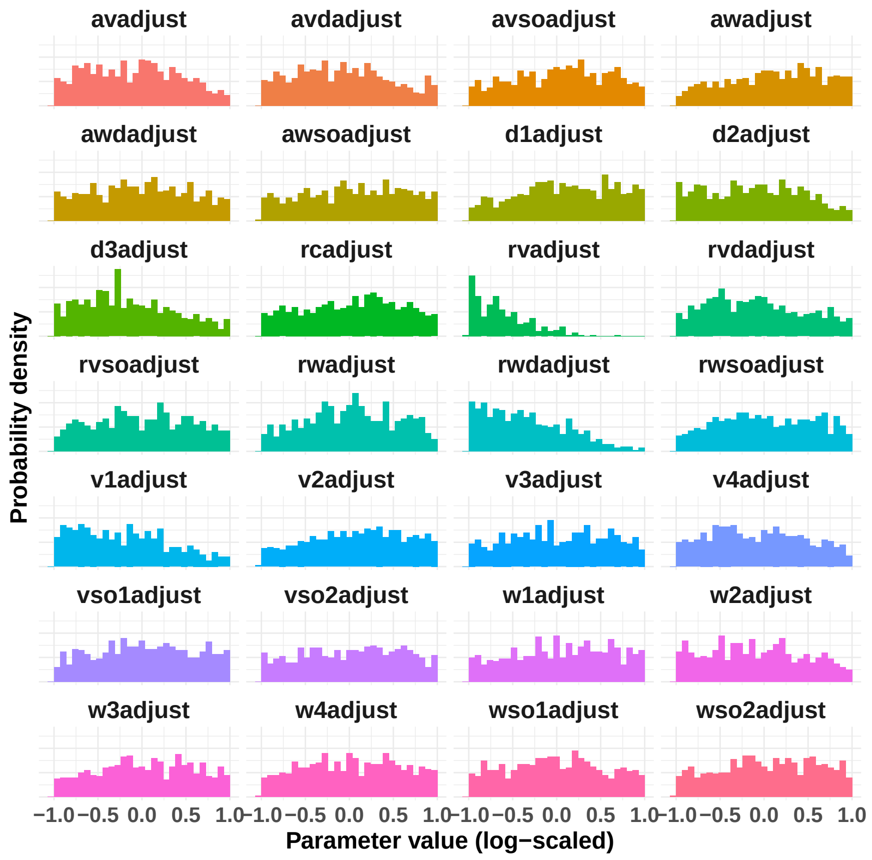


Figure 5.5: Posterior samples for active input parameters when the approximate likelihood is evaluated with respect to (n,a) experimental data. This reaction shows a preference for lower values of rvadjust and rwdadjust.

Chapter 6

Discussion

6.1 Summary

In this dissertation a discussion was presented on nuclear data evaluation, history matching, and Gaussian process emulation. Methodology and results on three waves of history matching were then discussed. History matching iteratively reduces the plausible space of simulator inputs by comparing its outputs to experimental measurements while accounting for uncertainties. The analysis is made tractable by employing a Gaussian process model to emulate the relationship between the simulator inputs and outputs. The discussed methodology included choosing an appropriate set of design points at which to run the simulator, training and validating waves of emulators of the simulator, using these emulators to identify non-implausible values for the input parameters, and sampling from the marginal posterior distributions of the active inputs conditional on subsets of the observed data.

The dissertation focussed on a particular case study examining nuclear reactions that take place when neutrons are incident upon the isotope Iron-56 at energies between 5 and 10 MeV. Experimental data on four reactions in this energy interval were extracted from EXFOR, a public database of nuclear reactions. The TALYS software was used to simulate nuclear data across the measured reactions and energies. A total of 28 TALYS parameters were examined across their ranges of possible values. No prior knowledge was employed to narrow the search space. After three waves of history matching, the non-implausible space was 0.03% of the volume of the original search space. The cost for this was approximately 1300 TALYS runs. Samples were then drawn from the marginal conditional posterior distributions of the parameters, using likelihoods functions for data on two of the observed reactions. The analysis showed that for many of the active parameters, a range of values were still consistent with the relevant data, which may indicate that more history matching waves would be required to reduce the plausible volume further, or that relevant outputs were insensitive to the settings for that parameter.

6.2 Limitations of the analysis

The intention of this dissertation was to demonstrate how Bayesian history matching could be used as a part of a nuclear data evaluation workflow. However, limitations in the analysis mean that the results themselves are only demonstrative and care should be taken in interpreting them too closely.

The main limitation of the analysis was the lack of availability of computational resources. Nuclear data evaluation is usually a high performance computing task, and this analysis was carried out on a personal laptop. Consequently, decisions on how long to run calculations for were made pragmatically. For example, in references (Andrianakis et al. 2015) and (Bower, Goldstein, and Vernon 2010), more simulator runs were carried out in successive waves to allow for higher order trend functions to be fitted. This was attempted in the second wave, where the time and opportunity arose to double the number of simulator from the first wave to 600. A trend function with first order and squared terms was fitted (but not interaction terms as the number of interactions between 28 parameters was too large). However a great many of these models failed, indicating that more runs may have been needed in order to fit the more complex models, so a first order linear trend term was used instead. For the third wave, there was only time to carry out 300 simulator runs for training the emulators. A similar problem arose when evaluating proposal samples for the non-implausible space, as for each proposal sample, predictions had to be carried out for hundreds of models, and consequently, even computing implausibility measures for a few tens of thousands of proposal inputs required running calculations for a day. Issues such as these were non-trivial when making progress on analysis depended on the availability of results. Consequently some decisions were made that were not consistent with history matching in the literature, such as decreasing the number of training points used for the second wave, and not examining more complex trend functions.

One important consequence of not using enough training data to build emulators is that there are large regions of space in which the emulator must interpolate, and as discussed in Section 4.5, predictive uncertainty grows as the distance from the observed data. This can result in emulators being validated and proposal points being evaluated as non-implausible by virtue of a great amount of predictive uncertainty, as opposed to making a good match to results. Figure **??** illustrates this point. The Figure shows the results of the validation runs for the emulator of the (n,tot) reaction at energy 5.49 MeV for the third wave. The vertical lines show the two standard deviation predictive intervals for the emulator, and the reference line x=y is plotted. This emulator validated successfully. The emulator certainly captured the general trend of the simulator, but the good match can be seen to be in no small part due to the wide predictive intervals, the majority of which intersect the x=y line, indicating that the emulator predictions are within two standard deviations of the simulator outputs. As an illustration, the dashed lines indicate the emulator predictions that would not intersect the x=y line if the predictive standard deviation were reduced by a half.

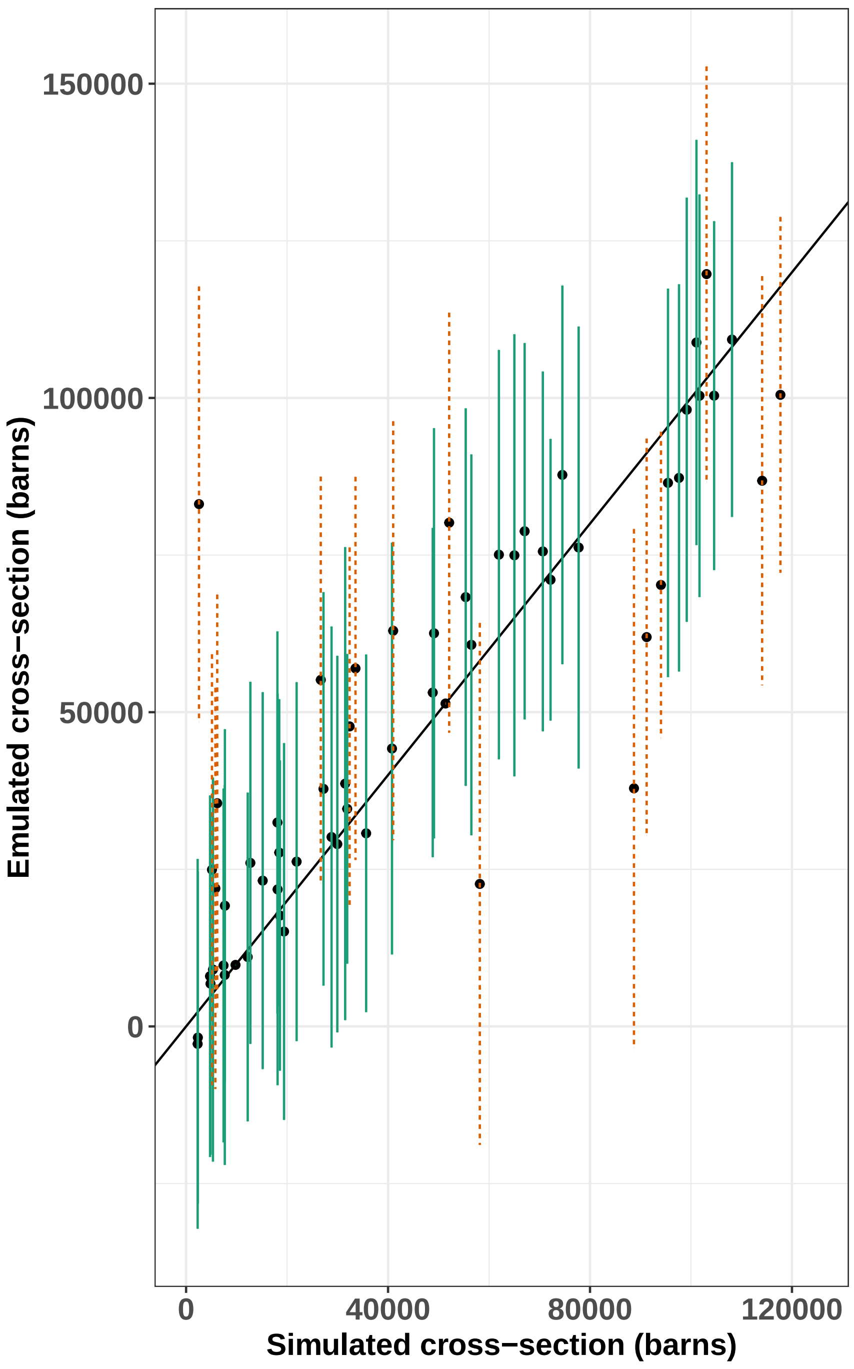


Figure 6.1: Validation run results for the emulator for the (n,tot) reaction at energy 5.49 MeV. The emulated versus simulated cross-sections are plotted with their two standard deviation error bars, as well as an x=y line for reference. Dashed error bars indicate emulator results than would not be within two standard deviations of the simulator results if the marginal predictive error were reduced by a half.

Figure **??** illustrates something else that could have been done better. The observed cross section for this reaction at this energy is 3732 barns. This is certainly within the range of this plot, however it can be seen that the simulator emulator can predict cross sections two orders of magnitude greater than this, depending on the values of the active parameters. It was decided early on in the analysis that, when training emulators, the simulator outputs should not be scaled. This decision was taken as it was thought that an error could easily have been made when rescaling the emulator predictions to compute implausibility measures such as Equation @ref((eq:chi-sq-impl) and due to a lack of certainty in how the assumptions pertaining to the measures would hold up under rescaling. However, Figure **??** proves good evidence that the simulator output would have benefited from logarithmic scaling to give the emulators a smaller predictive volume, and also to enforce non-negativity, which is a characteristic of cross-sections, and which can be seen to be violated in some of the predictions in the Figure.

In (Andrianakis et al. 2015) nine history matching waves are carried out over 18 active input parameters, and in (Bower, Goldstein, and Vernon 2010), five history matching waves were carried out over 17 input parameters, using many thousands of simulator runs. In this dissertation, three history matching waves were carried out over 28 input parameters. Consequently, it is expected that further waves would refocus the input volume yet further and produce more representative results.

In Section 4.7, different Gaussian process software implementations were discussed, with the R package 'DiceKriging' being chosen because of it's relatively quick run time and because it exposed a function that allowed access to predictive covariance matrices, an important component of model validation. However it is worth noting that (**comparison?**) found that 'DiceKriging' was one of the worst performers when tested against other Gaussian process software implementations. In particular, having to include a 'nugget' parameter (Section 4.5) to help the fitting procedure was unsatisfactory, but it was necessary in order to allow the analysis to take place.

It is expected that cross-sections at energies close together for the same reaction should be similar, and that an input parameter setting that gives good results for a given energy would also give good results for a neighbouring energy points. In this dissertation, independent emulators were built for every unique point, and no account was made for potential correlations of similar simulator outputs. This decision was taken because emulation of correlated multivariate outputs such as in (Fricker, Oakley, and Urban 2013) is a non-trivial extension to univariate emulation, and not easily implemented using out-of-the-box and well-known software implementations.

Finally, the values for observation uncertainties and simulator inadequacies (Section **??**) used in this work were chosen arbitrarily, and this would be reflected in the results. More representative uncertainties could be obtained by better mastering the software described in (Schnabel et al. 2021) and through expert elicitation to better understand how far the simulator predictions should be expected to deviate from reality, but both of these were outside the scope of the dissertation.

6.3 Further work and conclusions

As discussed in Section 5.5, identifying TALYS input parameter which give rise to good agreement with experiments is a step towards the goal of nuclear data evaluation, which is to have a set of cross-sections consistent with experimental data and with accompanying uncertainties. One way to achieve this with the results of history matching, is to run TALYS at many different values of the input parameters drawn from their joint posterior distribution. If a sufficient number of runs could be carried out, sample means and covariance matrices computed from the outputs of these runs could be used to define the evaluated dataset, one that would be consistent with experimental data and uncertainties, conditional on the modelling approach used. This task could also be made more efficient by using Gaussian process emulation in a similar way to that described in this work.

In conclusion, nuclear data evaluation is an enormous task and in this dissertation a small case study was examined to assess the usefulness of history matching as an addition to the evaluation toolkit. The analysis showed some good results, with a large volume of parameter space able to be analysed with the help of Gaussian process emulation. It is expected that better results could be seen with better computational resources, carefully defining relevant uncertainties, and in accounting for correlations between similar processes.

Appendix A

Spherical optical model potential

TALYS implements several nuclear models, the main one being th spherical optical model potential (A. J. Koning, Hilaire, and Duijvestijn 2008). All but two of the active input parameters in the analysis are parameters in this model. The two that are not are parameters in an alternative model which is used in certain incident neutron energy ranges. Fore more detail see Chapter 4 of reference (A. J. Koning, Hilaire, and Duijvestijn 2008). For completeness the equation, along with the mapping between TALYS and equation parameters, are given here.

$$\begin{aligned}
V_V(E) &= v_1\left[1 - v_2(E - E_f) + v_3(E - E_f)^2 - v_4(E - E_f)^3\right] \qquad \text{(A.1)}\\
W_V(E) &= w_1\frac{(E - E_f)^2}{(E - E_f)^2 + w_2^2}\\
r_V &= \text{constant}\\
a_V &= \text{constant}\\
W_D(E) &= d_1\frac{(E - E_f)^2}{(E - E_f)^2 + d_3^2}\exp\left[-d_2(E - E_f)\right]\\
r_D &= \text{constant}\\
a_D &= \text{constant}\\
V_{SO}(E) &= v_{so1}exp\left[-v_{so2}(E - E_f)\right]\\
W_{SO}(E) &= w_{so1}\frac{(E - E_f)^2}{(E - E_f)^2 + w_{so2}^2}\\
r_{SO} &= \text{constant}\\
a_{SO} &= \text{constant}\\
r_C &= \text{constant}
\end{aligned}$$

Table A.1: Active input parameters in the analysis and the parameters they adjust in Equation
eqrefeq:omp-equation

TALYS	**Equation**
v1adjust	v1
v2adjust	v2
v3adjust	v3
v4adjust	v4
rvadjust	rv
avadjust	av
rwadjust	rw
awadjust	aw
w1adjust	w1
w2adjust	w2
w3adjust	NA
w4adjust	NA
rvdadjust	rvd
avdadjust	avd
rwdadjust	rwd
awdadjust	awd
d1adjust	d1
d2adjust	d2
d3adjust	d3
vso1adjust	vso1
vso2adjust	vso2
wso1adjust	wso1
wso2adjust	wso2
rvsoadjust	rvso
avsoadjust	avso
rwsoadjust	rwso
awsoadjust	awso
rcadjust	rcadjust

Appendix B

R code to generate implausibility/ optical depth heatmaps

Below is the R code used to generate the implausibility and optical depth heat maps from Section 5.3. The function 'get_implausibilities' expects names of two of the active parameters as arguments 'v1' and 'v2'. The argument 'df' should be a data frame with at least two named parameters columns with the parameter values, and third column 'I2' with the corresponding implausibility measure. The function 'optical_depth_plot' requires a further column 'plausible' which is 1 of the input parameters were evaluated as plausible, and zero otherwise.

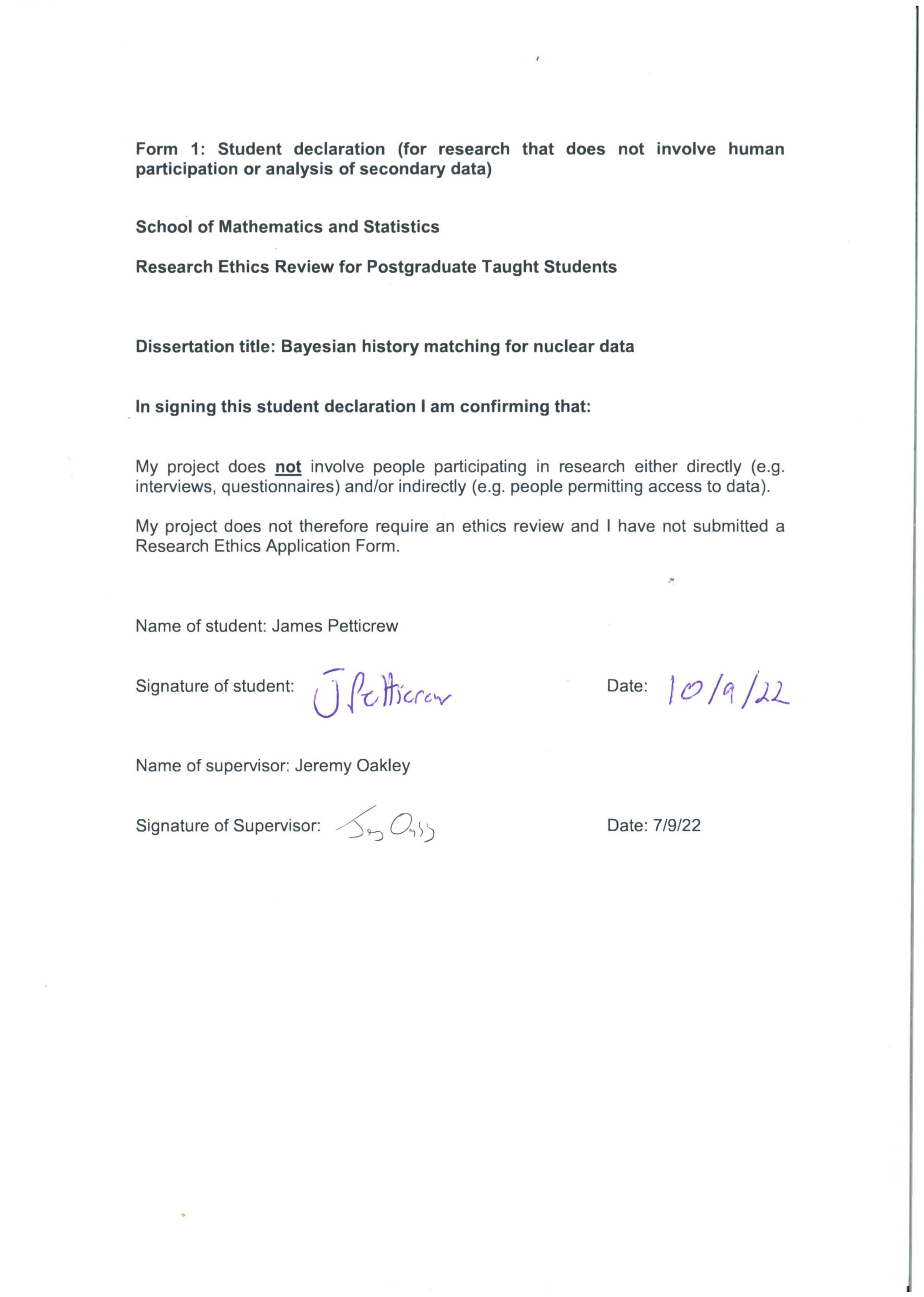
```r
# Pull out the minimum maximum implausibilities
get_implausibilites <- function(v1,v2,df){
  v1 = rlang::sym(v1)
  v2 = rlang::sym(v2)
  x <- seq(-1,1,length.out = 20)
  gap <- (x[2] - x[1])/2
  midpoint_1 <- x[1] + gap
  y <- seq(midpoint_1,-midpoint_1,length.out=19)
  gr <- expand.grid(y,y)
  # Slice up the parameter space and find the minimum
  # implausibility in each interval
  interval_mins <- purrr::pmap_dfr(gr, function(Var1,Var2){
    l1 <- Var1 - gap
    l2 <- Var2 - gap
    u1 <- Var1 + gap
    u2 <- Var2 + gap
    min_i <- df %>% dplyr::filter(!!v1 < u1 & !!v1 > l1 &
                                !!v2 < u2 & !!v2 > l2) %>%
      dplyr::pull(I2) %>% min()
    tibble::as_tibble(t(c(Var1,Var2,min_i))) %>%
      dplyr::rename(v1 = V1, v2 = V2, "Implausibility"=V3)
  } )
  interval_mins %>%
    ggplot2::ggplot(ggplot2::aes(x=v1,y=v2)) +
    ggplot2::geom_tile(ggplot2::aes(fill=Implausibility ) ) +
    ggplot2::theme_minimal() +
    ggplot2::scale_fill_viridis_c(direction=-1) +
    ggplot2::labs(fill= "Minimal\nImplausibility",
                  x = v1, y=v2)
}


# Compute optical depths in each interval and plot heatmap
optical_depth_plot <- function(v1,v2,df){
  v1 = rlang::sym(v1)
  v2 = rlang::sym(v2)
  x <- seq(-1,1,length.out = 20)
  gap <- (x[2] - x[1])/2
  midpoint_1 <- x[1] + gap
  y <- seq(midpoint_1,-midpoint_1,length.out=19)
  gr <- expand.grid(y,y)
  optical_depths <- purrr::pmap_dfr(gr, function(Var1,Var2){
    l1 <- Var1 - gap
    l2 <- Var2 - gap
    u1 <- Var1 + gap
    u2 <- Var2 + gap
    depth <- df %>% dplyr::filter(!!v1 < u1 & !!v1 > l1 &
                                    !!v2 < u2 & !!v2 > l2) %>%
      dplyr::summarise(count = dplyr::n(),
                       plausible = sum(plausible)) %>%
      dplyr::mutate(o_depth = plausible/count) %>%
      dplyr::pull(o_depth)
    tibble::as_tibble(t(c(Var1,Var2,depth))) %>%
      dplyr::rename(v1 = V1, v2 = V2, "Depth"=V3)
  } )
  optical_depths %>%
    ggplot2::ggplot(ggplot2::aes(x=v1,y=v2)) +
    ggplot2::geom_tile(ggplot2::aes(fill=Depth  ) )+
    ggplot2::theme_minimal() +
    ggplot2::scale_fill_viridis_c(direction=-1,
                                  option="C") +
    ggplot2::labs(fill= "Optical\nDepth",
                  x = v1, y=v2)
```

```
}
```

Appendix C

Research Ethics Approval

The research ethics approval process for the work described in this dissertation was completed in April 2020, before any work using data commenced. Upon completion of the required online ethics declaration form, it was decided that it was not necessary to gain formal research ethics approval for the dissertation, owing to the lack of data involving human subjects. A scan of a student declaration form saying as much, and signed by both the student and the supervisor, is included below.



Form 1: Student declaration (for research that does not involve human participation or analysis of secondary data)

School of Mathematics and Statistics

Research Ethics Review for Postgraduate Taught Students

Dissertation title: Bayesian history matching for nuclear data

In signing this student declaration I am confirming that:

My project does **not** involve people participating in research either directly (e.g. interviews, questionnaires) and/or indirectly (e.g. people permitting access to data).

My project does not therefore require an ethics review and I have not submitted a Research Ethics Application Form.

Name of student: James Petticrew

Signature of student: [illegible] Date: 10/9/22

Name of supervisor: Jeremy Oakley

Signature of Supervisor: [illegible] Date: 7/9/22

Figure C.1: The research ethics approval letter provided for the work outlined in this dissertation, following compliance with the University of Sheffield offical research ethics processes.

References

Alhassan, Erwin, Dimitri Rochman, Alexander Vasiliev, Mathieu Hursin, Arjan J Koning, and Hakim Ferroukhi. 2022. "Iterative Bayesian Monte Carlo for Nuclear Data Evaluation." *Nuclear Science and Techiques* 30 (50).

Andrianakis, Ioannis, Ian R Vernon, Nicky McCreesh, Trevelyan J McKinley, Jeremy E Oakley, Rebecca N Nsubuga, Michael Goldstein, and Richard G White. 2015. "Bayesian History Matching of Complex Infectious Disease Models Using Emulation: A Tutorial and a Case Study on HIV in Uganda." *PLoS Computational Biology* 11 (1): e1003968.

Bower, Richard G, Michael Goldstein, and Ian Vernon. 2010. "Galaxy Formation: A Bayesian Uncertainty Analysis." *Bayesian Analysis* 5 (4): 619–69.

Brown, David A, MB Chadwick, R Capote, AC Kahler, A Trkov, MW Herman, AA Sonzogni, et al. 2018. "ENDF/b-VIII. 0: The 8th Major Release of the Nuclear Reaction Data Library with CIELO-Project Cross Sections, New Standards and Thermal Scattering Data." *Nuclear Data Sheets* 148: 1–142.

Erickson, Collin B, Bruce E Ankenman, and Susan M Sanchez. 2018. "Comparison of Gaussian Process Modeling Software." *European Journal of Operational Research* 266 (1): 179–92.

Fricker, Thomas E, Jeremy E Oakley, and Nathan M Urban. 2013. "Multivariate Gaussian Process Emulators with Nonseparable Covariance Structures." *Technometrics* 55 (1): 47–56.

Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. 2013. *Bayesian Data Analysis, Third Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis.

Genz, Alan, Frank Bretz, Tetsuhisa Miwa, Xuefei Mi, Friedrich Leisch, Fabian Scheipl, and Torsten Hothorn. 2021. *mvtnorm: Multivariate Normal and t Distributions*. https://CRAN.R-project.org/package=mvtnorm.

Gu, Mengyang, Jesus Palomo, and James Berger. 2022. *RobustGaSP: Robust Gaussian Stochastic Process Emulation*. https://CRAN.R-project.org/package=RobustGaSP.

Helgesson, P., and H. Sjöstrand. 2018. "Treating Model Defects by Fitting Smoothly Varying Model Parameters: Energy Dependence in Nuclear Data Evaluation." *Annals of Nuclear Energy* 120: 35–47. https://doi.org/https://doi.org/10.1016/j.anucene.2018.05.026.

Iwamoto, Hiroki. 2020. "Generation of Nuclear Data Using Gaussian Process Regression." *Journal of Nuclear Science and Technology* 57 (8): 932–38.

https://doi.org/10.1080/00223131.2020.1736202.

Koning, A. J., S. Hilaire, and M. C. Duijvestijn. 2008. "TALYS-1.0." In *Proceedings of the International Conference on Nuclear Data for Science and Technology*, edited by O. Bersillon, F. Gunsing, E. Bauge, R. Jacqmin, and S. Leray, 211–14. EDP Sciences.

Koning, AJ, D Rochman, J-Ch Sublet, N Dzysiuk, M Fleming, and S Van der Marck. 2019. "TENDL: Complete Nuclear Data Library for Innovative Nuclear Science and Technology." *Nuclear Data Sheets* 155: 1–55.

Oakley, Jeremy. 1999. "Bayesian Uncertainty Analysis for Complex Computer Codes." PhD thesis, The University of Sheffield.

Otuka, N., E. Dupont, V. Semkova, B. Pritychenko, A. I. Blokhin, M. Aikawa, S. Babykina, et al. 2014. "Towards a More Complete and Accurate Experimental Nuclear Reaction Data Library (EXFOR): International Collaboration Between Nuclear Reaction Data Centres (NRDC)." *Nuclear Data Sheets* 120: 272–76. https://doi.org/https://doi.org/10.1016/j.nds.2014.07.065.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12 (Oct): 2825–30.

Pukelsheim, Friedrich. 1994. "The Three Sigma Rule." *The American Statistician* 48: 88–91.

Rougier, Jonathan. 2008. "Efficient Emulators for Multivariate Deterministic Functions." *Journal of Computational and Graphical Statistics* 17 (4): 827–43.

Roustant, Olivier, David Ginsbourger, and Yves Deville. 2012a. "DiceKriging, DiceOptim: Two r Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization." *Journal of Statistical Software* 51: 1–55.

———. 2012b. "DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization." *Journal of Statistical Software* 51 (1): 1–55. https://www.jstatsoft.org/v51/i01/.

Schnabel, G., H. Sjöstrand, J. Hansson, D. Rochman, A. Koning, and R. Capote. 2021. "Conception and Software Implementation of a Nuclear Data Evaluation Pipeline." *Nuclear Data Sheets* 173 (March): 239–84. https://doi.org/10.1016/j.nds.2021.04.007.

Shibata, Keiichi, Osamu Iwamoto, Tsuneo Nakagawa, Nobuyuki Iwamoto, Akira Ichihara, Satoshi Kunieda, Satoshi Chiba, et al. 2011. "JENDL-4.0: A New Library for Nuclear Science and Engineering." *Journal of Nuclear Science and Technology* 48 (1): 1–30.

Williams, Christopher KI, and Carl Edward Rasmussen. 2006. *Gaussian Processes for Machine Learning.* Vol. 2. 3. MIT press Cambridge, MA.