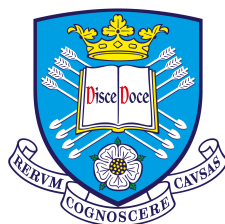


# Bayesian History Matching for Nuclear Data

James M. Petticrew

School of Mathematics and Statistics  
University of Sheffield



The  
University  
Of  
Sheffield.

Dissertation submitted as part of the requirements for the award of  
MSc in Statistics, University of Sheffield, 2021–2022



# 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. Gaussian process (GP) statistical models were used to emulate the relationships 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. GP emulators were 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

<b>Acknowledgements</b>	<b>iii</b>
<b>Lay Summary of the Dissertation</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Aim of the dissertation . . . . .	1
1.3 Structure of the dissertation . . . . .	2
<b>2 Nuclear data evaluation</b>	<b>3</b>
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 evaluation . . . . .	7
2.6 Previous uses of Gaussian processes in nuclear data . . . . .	8
2.7 Conclusion . . . . .	8
<b>3 History matching</b>	<b>9</b>
3.1 Introduction . . . . .	9
3.2 Motivation . . . . .	9
3.3 Notation . . . . .	10
3.4 Workflow . . . . .	12
3.5 Observation uncertainty . . . . .	15
3.6 Simulator inadequacy . . . . .	15
3.7 Emulator variance . . . . .	16
3.8 Other sources of uncertainty . . . . .	16
3.9 One-dimensional implausibility . . . . .	16
3.10 Worked example . . . . .	17
3.11 Multidimensional implausibility . . . . .	19
3.12 Advantages of history matching . . . . .	20
3.13 Conclusion . . . . .	20
<b>4 Gaussian process regression</b>	<b>23</b>
4.1 Introduction . . . . .	23
4.2 Gaussian process regression . . . . .	23

4.3	Parameter estimation . . . . .	24
4.4	Conditioning on the data . . . . .	26
4.5	Covariance function . . . . .	28
4.6	Mean function . . . . .	30
4.7	Model complexity and the predictive distribution . . . . .	34
4.8	Software implementations . . . . .	34
4.9	Conclusion . . . . .	35
<b>5</b>	<b>Results</b>	<b>37</b>
5.1	Introduction . . . . .	37
5.2	First wave methodology . . . . .	37
5.3	First wave analysis . . . . .	39
5.4	Minimum implausibility and optical depth analysis . . . . .	41
5.5	Subsequent waves . . . . .	43
5.6	Posterior sampling of active inputs . . . . .	44
5.7	Conclusion . . . . .	48
<b>6</b>	<b>Discussion</b>	<b>49</b>
6.1	Summary . . . . .	49
6.2	Limitations of the analysis . . . . .	50
6.3	Further work and conclusions . . . . .	54
<b>A</b>	<b>Spherical optical model potential</b>	<b>55</b>
<b>B</b>	<b>R code to generate implausibility/ optical depth heatmaps</b>	<b>57</b>
<b>C</b>	<b>Research Ethics Approval</b>	<b>59</b>
	<b>References</b>	<b>61</b>



# 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. Part of this process consist of deciding at what values of the simulator input parameters to run it at. This aspect of evaluation is the focus of this work.

In this dissertation Bayesian history matching [1], [2] was 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 emulator.

This work was carried out in compliance with the research ethics approval process of Sheffield University as described in Appendix C.

### 1.2 Aim of the dissertation

The aim of the dissertation was to investigate history matching as a tool for nuclear data evaluation. The particular case study used was of neutrons incident upon an

Iron-56 isotope with energies 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 the simulator 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 limitations 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<sup>2</sup> 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<sup>3</sup> (see Figure 2.1). If the intensity of the beam can be measured after it has passed through the slab  $I$ , then the total cross-section  $\sigma_{tot}$  can be computed from

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

$\sigma_{tot}(Z, E)$  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<sup>2</sup>, however one square metre is an enormous value for a cross-section to take, and consequently they are often measured in ‘barns’<sup>1</sup> where a barn is equal to 10<sup>-28</sup>m<sup>2</sup>.

There are a number of reactions can occur as a result of a collision between a neutron and an atomic nucleus. For example, the neutron can change its momentum (known as inelastic scatter) or the neutron can be captured by the nucleus and release more neutrons (and energy - known as fission) as a result.

---

<sup>1</sup>From the phrase ‘as wide as a barn door’.

Each interaction type has its own associated cross-section proportional to the probability of that reaction occurring. The value of the cross-section depends on the energy of the incident particle, the type of incident particle and the type of target nucleus. In this dissertation the focus is on neutrons with energies between 5 and 10 MeV ( $5 \leq E \leq 10$ ) incident upon the Iron-56 isotope<sup>2</sup> ( $Z=\text{Fe-56}$ ).

The capacity for scientists to measure all interaction, energy, particle and isotope combinations is limited, and as such computer simulations, validated using experimental measurements, are used extensively to model cross-sections with suitable fidelity. In this work the nuclear reaction simulation code TALYS version 1.95 [3] was used, which is explained further in the next section.

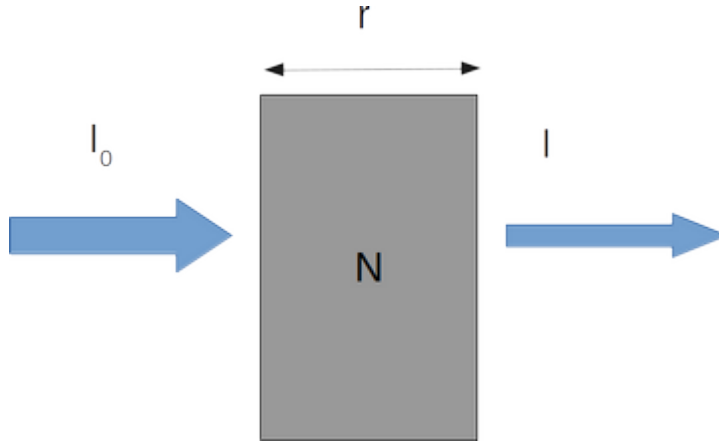


Figure 2.1: Illustration to aid interpretation of microscopic neutron cross-section. The ratio of the outgoing to the incident beam intensity is proportional to microscopic total cross-section.

## 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 [3] which can be specified by the user. Here a distinction is made between state and active 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. Active 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. Observational data can be compared to the outputs of the simulator for different choices of active parameter settings. The hope is that settings which produce simulator outputs consistent with observational data will also produce simulator outputs consistent with reality for unobserved quantities.

<sup>2</sup>An Iron atom with 26 protons and 30 neutrons in its nucleus. All Iron atoms have 26 protons in their nucleus but different isotopes have different numbers of neutrons. Iron-56 is the most common naturally occurring isotope of Iron.

In this dissertation 28 active 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, it is assumed that every possible value for the active parameters was equally likely, which is not the same thing as the multipliers being equally likely. In this dissertation, when values for the active parameters are discussed, the intended meaning is values of the multipliers, rather than the parameters themselves. Multipliers can be thought of as parameters in the model in Appendix A if it is re-parametrised 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 [4], 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 [5].

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 5 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

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

nucleus) as result. A summary of the 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 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.

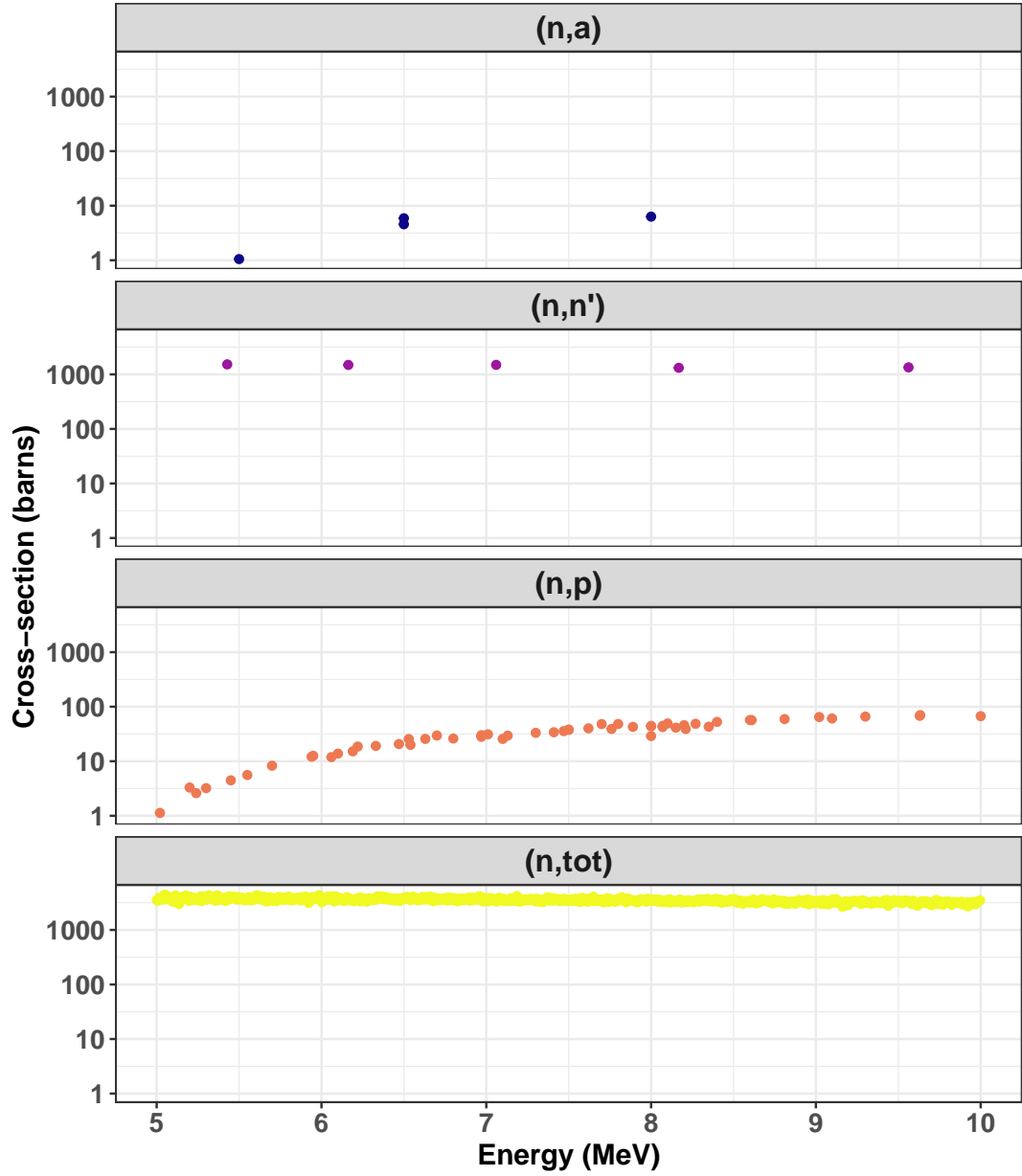


Figure 2.2: Plots of relevant Iron-56 neutron cross-section data extracted from the EXFOR database. Each panel shows data for a different reaction cross-section. 1065 (n,tot) data points are plotted, it was decided to only use half of them in the analysis.

## 2.5 Nuclear data evaluation

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 in evaluation, part of this process must

consist of deciding at what value, or values of the active parameters at which to run the simulator. This is the aspect of nuclear data evaluation that was focussed on in this dissertation. In Bayesian history matching large regions of the space of possible active parameter values are iteratively discarded 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 [6]–[8], 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 [5] and [9] the authors used Gaussian processes to describe the relationship between systematic errors and incident particle energies, and in [10] there is a demonstration of how to generate cross-sections from experimental data only. Another iterative Bayesian approach to evaluation can be found in [11].

## 2.7 Conclusion

In this section was given a brief overview of nuclear cross-sections. The two main sources of data used in the dissertation were described; simulated cross-sections from TALYS and measured cross-sections from the EXFOR database, and some context was given for the application. The data described here was used in a process known as history matching, where we try to find values for the active 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 history matching is described and relevant notation is developed. A description is given of the various sources of uncertainty that need to be characterised in order to carry out the analysis. Finally the metrics used to help identify suitable values for active TALYS parameters are described.

### 3.2 Motivation

The motivation for using history matching was to find a subset or subsets of TALYS's 28-dimensional active parameter space which could give rise to acceptable matches between TALYS outputs, denoted  $f$ , and the true cross-sections which it attempts to simulate, denoted  $y$ . The term 'acceptable' here implies that an exact match is not expected. One reason for this is that  $y$  cannot be measured perfectly, but there is a discrepancy between the measured cross-section, denoted  $z$ , and  $y$ . A second reason is that TALYS is not expected to perfectly simulate  $y$ . A third reason is that history matching requires  $f$  to be evaluated a large number of times and consequently a statistical emulator  $\hat{f}$  is used to approximate  $f$  (see Section 4) because it takes much less time to evaluate. The relationships between these four quantities are shown in Figure 3.1. If the discrepancies between the quantities are well quantified (see Sections 3.5-3.7), then there is enough information to compute some metric to decide if the match between TALYS and the true process is non-implausible (Section 3.9) for some value of the active input parameters.

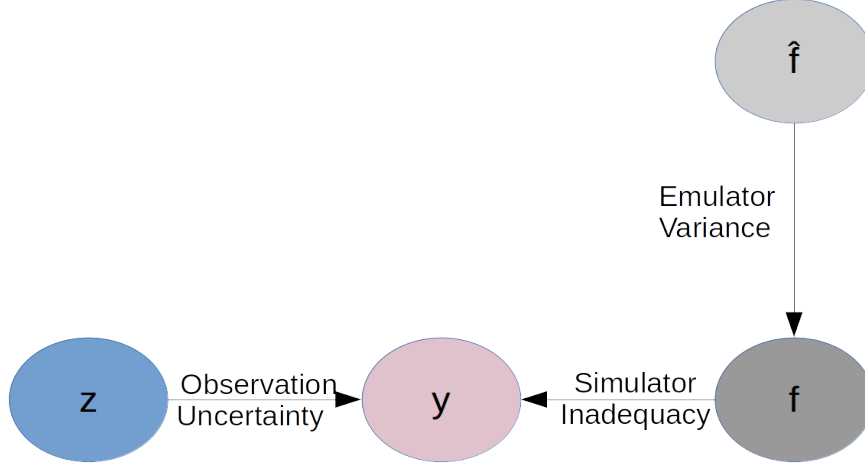


Figure 3.1: Both the experimental measurement  $z$  and the simulator  $f$  imperfectly capture the true process  $y$ . A further discrepancy occurs because the simulator itself must be imperfectly emulated by  $\hat{f}$ .

History matching works by iteratively discounting values of the active input parameters as implausible give measurements and relevant uncertainties. New sets of emulators are built which focus on predicting well in the non-implausible active parameter 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 there is a simulator with  $p$  active input parameters, denoted by the p-vector  $\mathbf{x} = [x^{(1)}, x^{(2)}, \dots, x^{(p)}]$ . In this application  $p = 28$ .  $\mathbf{x}$  can be any point in the 28-dimensional volume  $\mathcal{X}$ , which could be the set of all  $\mathbf{x}$  that the simulator accepts 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  $z$ . In general there is a vector of measurements, denoted by the n-vector  $\mathbf{z} = [z_1, z_2, \dots, z_n]$ , and the individual measurements are indexed as  $z_i$   $i \in 1, 2, \dots, n$ . Each  $z_i$  corresponds to a physical process  $y_i$ , denoted collectively as  $\mathbf{y} = [y_1, y_2, \dots, y_n]$ . In this application  $n = 588$  (see Section 2.4). The simulator simulates all of  $\mathbf{y}$  in one code run at some value of  $\mathbf{x}$ . The simulated output vector is denoted  $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_n(\mathbf{x})]$ . 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 from a single code run.

The imperfection of the experimental measurement process is represented by

writing

$$z_i = y_i + \delta_i \quad (3.1)$$

where  $\delta_i$  is a univariate Gaussian random variable with expectation 0 and variance  $V_i^{(obs)}$  representing our belief about the uncertainty of  $y_i|z_i$ .

The imperfection of the simulator is represented by writing

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

where  $\phi_i$  is a zero-centred univariate Gaussian random variable with variance  $V_i^{(s)}$  used to represent our belief about the uncertainty of  $y_i|f_i(\mathbf{x})$ . It is assumed that  $\phi_i$  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,  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m$ , indexed using  $j = 1, 2, \dots, m$ , where each of the  $\mathbf{x}_j$  is a  $p$ -vector with a proposed scalar value for each of the  $p$  active parameters. Theoretically, if the models in TALYS were a perfect representation of reality, there would exist a  $\mathbf{x}_j$  such that  $f_i(\mathbf{x}_j) = y_i \forall i$ . All of the elements of  $\mathbf{x}$  are theoretically observable quantities, so in that context a perfect match would arise if they could all be set to their ‘correct’ value. A perfect match could also arise for some proposal  $\mathbf{x}$  if TALYS were not a perfect representation of reality (getting the right answer for the wrong reasons). Regardless, a perfect match between  $f$  and  $y$  could not be detected, because  $y$  is never observed directly.

In order to well explore  $\mathcal{X}$  it is required that  $m$  be 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,  $k \ll m$  training runs of the simulator are used to build  $n$  emulators,  $\hat{f}_1(\cdot), \hat{f}_2(\cdot), \dots, \hat{f}_n(\cdot)$  one for each of the  $f_1(\cdot), f_2(\cdot), \dots, f_n(\cdot)$ . The  $\hat{f}_i$  are all statistically independent. The set of  $k$  training points used to train the emulators is denoted  $\mathcal{X}^*$ .

The  $\hat{f}_i$  can then be used to predict the simulator outputs  $f_i$  for many samples from  $\mathcal{X}$ . The emulators will predict the simulator outputs at points from  $\mathcal{X}^*$  exactly<sup>1</sup> but predictions at points not from  $\mathcal{X}^*$  will be imprecise. In Chapter 4 it will be seen that the  $\hat{f}_i$  are Gaussian random variables whose means and variances depend on  $\mathbf{x}$ .

Our uncertainty about the simulator output at unobserved  $\mathbf{x}_j$  is represented by

---

<sup>1</sup>In Section 4.5 this idea is modified slightly, but the ideas discussed here still stand

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

Symbol	Meaning
$\mathbf{x}$ or $\mathbf{x}'$	Vector of simulator calibration parameters
$p$	Number of active input parameters
$\mathcal{X}$	The set of all values that $\mathbf{x}$ can take
$y_i$	The $i$ th 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 and the dimensionality of the simulator output
$f_i(\mathbf{x}_j)$	Simulated value of the physical process $y_i$ for calibration input $\mathbf{x}_j$
$\hat{f}_i$	Emulator for $f_i$
$\delta_i$	Random variable representing our uncertainty about the discrepancy between the process $y_i$ and its measurement $z_i$
$V_i^{(obs)}$	Variance of $\delta_i$
$\phi_i$	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 $\mathbf{f}$
$\mathcal{X}^\dagger$	Subset of $\mathcal{X}$ at which to predict the behaviours of $\mathbf{f}$ using $\hat{\mathbf{f}}$
$k$	Number of points in the training set $\mathcal{X}^*$
$m$	Number of active parameters, the dimension of the input $\mathbf{x}$
$E[f_i(\mathbf{x}_j)]$	Emulator expectation value for the $i$ th simulator output at input $\mathbf{x}_j$
$V[f_i(\mathbf{x}_j)]$	Marginal emulator variance for the $i$ th simulator output at input $\mathbf{x}_j$

writing

$$f_i(\mathbf{x}_j) \sim \mathcal{N}(E[f_i(\mathbf{x}_j)], V[f_i(\mathbf{x}_j)]) \quad (3.3)$$

where  $E[f_i(\mathbf{x}_j)]$  and  $V[f_i(\mathbf{x}_j)]$  are the mean and variance of emulator  $\hat{f}_i(\mathbf{x}_j)$ .

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

### 3.4 Workflow

In this subsection an overview of the steps iterated over in history matching is given, following the schematic shown in Figure 3.2.

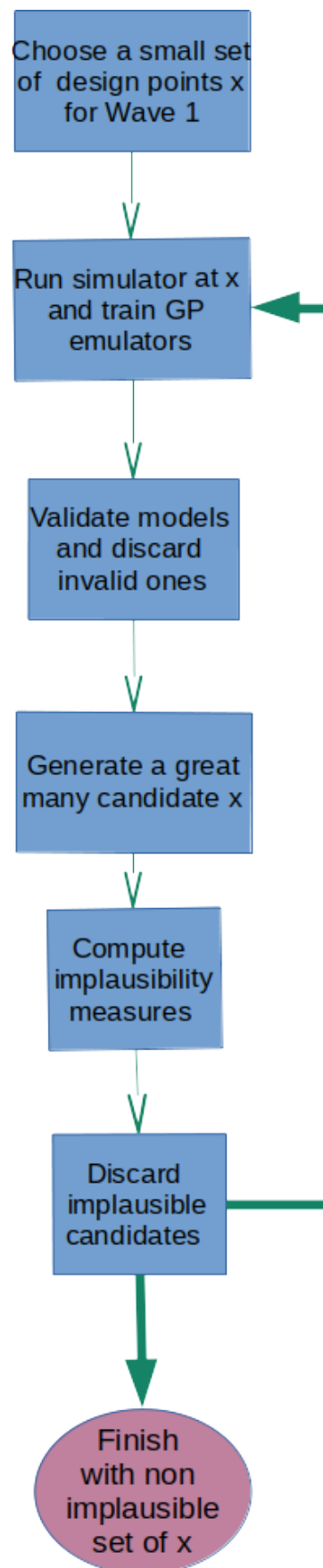


Figure 3.2: Schematic showing a typical history matching workflow. The process is iterative where each iteration is called a wave, and a wave ends either with refocussing the search space or stopping with the current set of non-implausible samples.

**Choose design points for Wave 1:** It was required that a great many candidate  $\mathbf{x}$  be examined, but the number of times that TALYS could be run to evaluate these inputs directly was relatively small. Consequently the limited runs available were used to build statistical emulators for TALYS. Initially, all samples from  $\mathcal{X}$  were equally plausible, and hence emulators were required that well represent TALYS's behaviour over all of  $\mathcal{X}$ . To maximise the chance of achieving this, and to minimise the predictive variance of the emulators, a space-filling design was employed that provided coverage of  $\mathcal{X}$  in some way that is optimised for the number of runs afforded. In this dissertation a Latin Hypercube (LH) design was used [12] to choose the  $\mathcal{X}^*$ . This is a high-dimensional extension of the Latin square experimental design.

**Run simulator at design points and use results to build emulators:** TALYS was run at all points in  $\mathcal{X}^*$  to obtain the  $\mathbf{x} \rightarrow f(\mathbf{x})$  mappings needed to train the emulators. The results were consolidated and formatted to suit the input format of the Gaussian process software package used (see Section 4.8). When training the first wave of emulators, some of them failed due to convergence problems in their numerical parameter estimation routines (see Section 4.3). This may have been because the active parameter space was very large at this stage, and some outputs were hard to emulate initially. One advantage of history matching is that the analysis was not invalidated by simply dropping these emulators.

**Validate models and discard invalid ones:** A small number of TALYS runs were carried out in order to test the out-of-sample predictive performance of the emulators. Any emulators not giving satisfactory out-of-sample performance were dropped. More details on the validation methodology used in this dissertation are given in Section 5.2.

**Generate a great many candidate  $\mathbf{x}$ :** The validated emulators were used to predict the TALYS output many times for many  $\mathbf{x}$  at a fraction of the cost of running the simulator. One TALYS runs took between 30 and 360 minutes to complete; contrastingly the emulators evaluated in a fraction of a second. Consequently a great many samples for  $\mathbf{x}$  were generated and the emulators were used to generate predictions for each one.

**Compute implausibility measures:** As discussed in Section 3.9, implausibility metrics were computed for each candidate  $\mathbf{x}$  based on the emulator predictions, the corresponding measurements, and the uncertainties present in the system. This typically leads to a large reduction in the non-implausible parameter space [1], [2], based on the implausibility metrics exceeding some acceptable threshold.

**Discard implausible candidates:** The reduction in non-implausible parameter space is known as refocussing, and resulted in one of two outcomes at the end of each wave. Either the non-implausible samples (or a subset of them) were used as the training points  $\mathcal{X}^*$  for the next set of TALYS runs, with which new emulators were built. 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. A second outcome is that the process was stopped, and the non-implausible samples were accepted as the outcome of the analysis. Stopping can 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 also be used as a precursor to an uncertainty quantification study where, having found the values for the input parameters consistent with observations, it is desired to find the set of simulator outputs consistent with those input parameters.

In order to decide if candidate  $\mathbf{x}$  are implausible, the uncertainties that are present in the system needed to be modelled. This is discussed in the next sections.

### 3.5 Observation uncertainty

The process of measurement resembles an aleatoric process, in that if one measurement  $z_i^{(1)}$  of the process  $y_i$  is taken, and then a second measurement  $z_i^{(2)}$ , in general  $z_i^{(1)} \neq z_i^{(2)}$ . The general assumption, though, is that the measurement process is imprecise, but not systematically wrong. This implies that if  $q$  measurements  $z_i^{(1)}, z_i^{(2)}, \dots, z_i^{(q)}$  were taken, as  $q$  gets larger  $\frac{1}{q} \sum_{j=1}^q z_i^{(j)} \rightarrow y_i$  and that the variance of the estimator for  $y_i$  is proportional to  $\frac{1}{\sqrt{q}}$ . 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 [5] provided tools for extracting and contextualising the uncertainties that accompanied the experimental data used in this dissertation. However, the computational expense required to use the tools was too large to undertake. Consequently a simple approach was taken in order allow progress to be made, and it was decided that the observation uncertainty  $V_i^{(obs)}$  from Equation (3.1) for  $y_i$  should be  $0.1z_i$ .

### 3.6 Simulator inadequacy

A discrepancy between the TALYS output  $f$  and the true process  $y$ , is expected to arise in at least two 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 an 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, a very simple approach was taken to uncertainty quantification, equating it to the corresponding measurement uncertainty of the process being simulated. Consequently  $V_i^{(s)} = V_i^{(obs)}$  and in particular,  $\phi_i$  is independent of  $\mathbf{x}$  in this model.

### 3.7 Emulator variance

As described in more detail in Section 4, an emulator  $\hat{f}$  was used to predict the output of the simulator  $f$  at unobserved  $\mathbf{x}$  as it allowed more efficient exploration of  $\mathcal{X}$ . The price for this efficiency was the introduction of another source of uncertainty. This uncertainty depends on  $\mathbf{x}$ . More detail is given in Chapter 4.

### 3.8 Other sources of uncertainty

In this dissertation we considered 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 an emulator to represent TALYS. In some applications, other sources of uncertainty may exist. For example, the application could require a simulator that uses Monte Carlo methods, and as such does not always give the same output when ran twice at the same input. This kind of uncertainty can be estimated from multiple simulator runs for the same  $\mathbf{x}$ , and computing the sample variance over these runs. Once all sources of uncertainty have been quantified, they are used to compute implausibility metrics to help assess if the simulator  $f_i$  could give non-implausible matches to the true process  $y_i$ .

### 3.9 One-dimensional implausibility

A good implausibility metric should be a function of  $\mathbf{x}$  and give extreme values if  $f_i(\mathbf{x})$  is unlikely to give an acceptable match to  $y_i$ . Consequently the metric should be proportional to  $|f_i(\mathbf{x}) - y_i|$ . It should also take into account uncertainties arising from measurement, simulation and emulation. The greater the combined uncertainties in the system, the less sure we can be in branding values of  $\mathbf{x}$  implausible. Consequently the metric should be inversely proportional to the combined uncertainties. The standard approach to combining uncertainties



[1] [2] is to assume that they are independent, and consequently that the variances are additive. This is the approach taken here, where the one-dimensional implausibility metric for  $\mathbf{x}_j$  for process  $y_i$  is defined as

$$I_j^{(i)} = \frac{|E[f_i(\mathbf{x}_j)] - z_i|}{\sqrt{V_i^{(obs)} + V_i^{(sim)} + V[f_i(\mathbf{x}_j)]}}. \quad (3.4)$$

Equation (3.4) is proportional to the difference between the observed and simulated process, inversely proportional to the total uncertainty in the system, and depends on  $\mathbf{x}_j$  as required. In order to decide what value of  $I_j$  to use as the cut-off for deeming  $\mathbf{x}_j$  plausible/ implausible, Pukelsheim's  $3\sigma$  rule is leveraged, which states that for any continuous unimodal distribution, 95% of its probability mass lies within 3 standard deviations of its mean [13]. One way to proceed would then be to reject all values of  $\mathbf{x}_j$  for which  $I_j > 3$  as implausible. Assuming that  $|E[f_i(\mathbf{x})] - z_i|$  meets Pukelsheim's requirements, 95% of all non-implausible  $\mathbf{x}_j$  should be retained on average, at the cost of losing 5% of 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.

## 3.10 Worked example

Suppose that the function  $f(x) = x^2 - 5x$  represents a simulator with a single input  $x$  and the interest lies in the output  $f$ , which attempts to simulate some process  $y$ . A measurement  $z$  of  $y$  has been made and found that  $z = 100$ , with some uncertainty associated with it. The simulator is run at five different values for  $x$  and these runs are used to build an emulator for  $f$ . The simulator output is then predicted at a great many points. Figure 3.3 illustrates the scenario. The dots are the design point, which are interpolated by the emulators mean predictions across the range  $[-10, 10]$ . The emulator predictor variance has been summed with the observation uncertainty and simulator inadequacy variances and the corresponding  $3\sigma$  intervals are shown in the Figure as lines above and below the mean prediction.

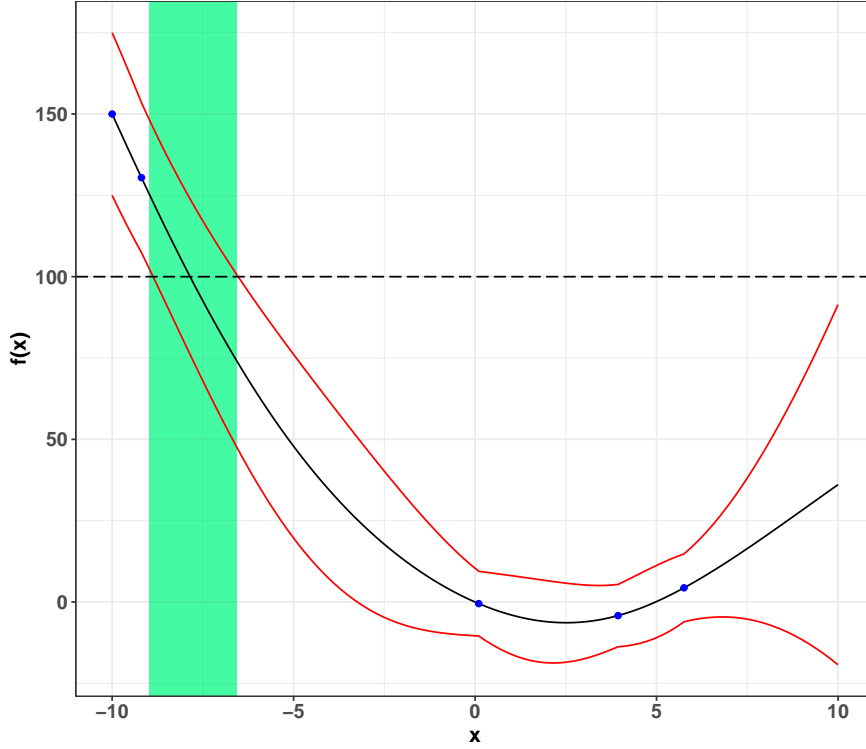


Figure 3.3: The simulator  $f$  is a function of one input  $x$ . The simulator has been run at the five points shown, and an emulator is built using these five points. Its predictions are shown interpolating the design points. An observation has been made of  $z=100$ , shown as the dashed horizontal line. The uncertainties arising due to the emulator, simulator and act of observation are combined and the 3 sigma uncertainty intervals are plotted above and below the mean predictions. If the 3 sigma cutoff is used for the implausibility measure, only those values of  $x$  shown intersecting the rectangle are non-implausible.

If the implausibility cut-off is set at  $3\sigma$ , values for  $x$  that intersect the rectangle in the plot are non-implausible; roughly the interval  $[-8.99, -6.97]$ . The correct value is around  $-7.81$ . If the simulator were then run at a point in that rectangular region, the emulator variance, would shrink to zero at that point, and greatly reducing the emulator variance at neighbouring points (see Section 4.5), 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 it could be imagined that were the domain of  $x$  to extend to  $+15$ , the mean prediction would intersect  $f = 100$  again, and indeed a second, much wider interval in  $x$  would be deemed non-implausible as the  $3\sigma$  intervals grow wider as the emulator 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.

Equation (3.4) is a one-dimensional implausibility metric, computed with respect to a single process  $y_i$ . In this dissertation, 588 processes were examined simultaneously, and ideally the simulator can give non-implausible matches to them all for some value or values of  $\mathbf{x}_j$ . Consequently the idea of the implausibility measure in Equation (3.4) is extended to compute an implausibility measure for  $\mathbf{x}_j$  with respect to multiple processes  $y_1, y_2, \dots, y_n$ .

### 3.11 Multidimensional implausibility

588 outputs  $f_1(\mathbf{x}_j), f_2(\mathbf{x}_j), \dots, f_{588}(\mathbf{x}_j)$  were produced each time TALYS was run for some  $\mathbf{x}_j$ . Each output  $f_i(\mathbf{x})$  had a corresponding measurement  $z_i$  of a true process  $y_i$ . In the literature there exist some methods for emulating multivariate outputs and accounting for correlations [14], [15] amongst the  $f$ s. In this dissertation, the simple approach of emulating each of the  $f$  independently was taken. As a consequence of this, there were up to 588 univariate implausibility measures for each input, and the criterion for labelling  $\mathbf{x}_j$  plausible or non-implausible had to be modified. One approach was to take the maximum of the 588 measures as  $I_j$  and to require that this be less than three for  $\mathbf{x}_j$  to non-implausible. A common approach [1], [2] is to take the second or third largest  $I_j^{(i)}$ . The choice is a pragmatic one and the sensitivity of the size of the non-implausible space to the choice of metric was examined in this dissertation, allowing a suitable choice can be made.

A second multivariate implausibility metric is

$$I(\mathbf{x}_j) = (\mathbf{z} - \mathbf{E}[\mathbf{f}(\mathbf{x}_j)])^T \left( \mathbf{V}_i^{(obs)} + \mathbf{V}_i^{(sim)} + \mathbf{V}[\mathbf{f}(\mathbf{x}_j)] \right) (\mathbf{z} - \mathbf{E}[\mathbf{f}(\mathbf{x}_j)]) \quad (3.5)$$

in which the vector of observations  $\mathbf{z} = (z_1, z_2, \dots, z_n)$  and the corresponding emulated quantities  $\mathbf{f}(\mathbf{x}_j) = (f_1(\mathbf{x}_j), f_2(\mathbf{x}_j), \dots, f_n(\mathbf{x}_j))$  are used 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. Under suitable assumptions,  $I(\mathbf{x}_j)$  has an asymptotic  $\chi^2$  distribution and consequently an appropriate percentile can be chosen from the  $\chi^2$  distribution with  $n$  degrees of freedom and use as the implausible/ non-implausible cut-off for  $\mathbf{x}_j$ .

One potential problem could arise in specifying the full covariance structure of  $\left( \mathbf{V}_i^{(obs)} + \mathbf{V}_i^{(sim)} + \mathbf{V}[\mathbf{f}(\mathbf{x}_j)] \right)$ . The simple approach was taken here to assume that all of the uncertainties were uncorrelated. In that case, all three covariance matrices are diagonal, with the  $i, i$ th element equal to the sum of the univariate uncertainties for that process, and all other elements zero. Taking this approach, it can be seen that the multivariate implausibility measure from Equation (3.5) is just the sum of the squared univariate implausibility measures from Equation (3.4),  $I_j(\mathbf{x}_j) = \sum_{i=1}^n \left( I_j^{(i)} \right)^2$

Here,  $I_j^{(uv)}$  is used to denote the univariate implausibility metric chosen for a given wave, and  $I_j^{(mv)}$  for the multivariate measure, where it is hoped that the

intended meaning should be clear from the superscript, and that these would not be mistaken for the  $uv$ th or  $mv$ th univariate implausibility metric for  $\mathbf{x}_j$ .

### 3.12 Advantages of history matching

History matching provides an efficient, pragmatic alternative to common inference methods such as maximum likelihood (ML) or Markov chain Monte Carlo (MCMC), which seek to learn a full distributional description of the active TALYS inputs from the data. In ML, the joint density of the inputs is modelled with some common likelihood function, commonly Gaussian. An optimisation algorithm is used to maximise this likelihood of the data with respect to the statistical model parameters. The maximum likelihood model parameters and the assumptions of the models together are sufficient to give a full probabilistic description of the active inputs. This method relies on the validity of the modelling assumptions and the success of the parameter estimation routine. Gradient-based, local optimisation algorithms risk converging on local, non-global maxima when trying to maximise the likelihood function, and this risk grows with dimensionality of the search space. Global optimisation algorithms can grow computationally very expensive and have no guarantee of successfully finding a global maximum for a finite run-time. In MCMC, the posterior distribution of the active inputs is not constrained to belong to a particular model family, instead samples can be numerically generated from the posteriors once the chains have converged on their joint stationary distribution. However, this convergence often requires a great many more samples than ML, and the chains can get stuck in regions of search space, both problems which can become worse in high dimensions.

Contrastingly, instead of searching for values of the active inputs that are most consistent with all of the observed data points, history matching identifies values of the active inputs that are inconsistent with one or more of the observed data points and discards them as implausible. This approach helps address some of the issues associated with high-dimensionality, and can rapidly reduce the non-implausible search space very quickly. The method is efficient because it is primarily concerned with simple functions of means and variances, which level of statistical detail is often appropriate given the performance of the simulators [1]. History matching does not require any likelihoods to be defined or any complex probabilistic calculations. The method is also efficient because the non-implausible search space is identified in iterations known as waves, where the search space shrinks each time. Because it is not a formal inference method, pragmatic decisions can be made on things such as implausibility cut-offs and how many samples to take.

### 3.13 Conclusion

In this section the process workflow of history matching was discussed, along with two of its important components, uncertainties and implausibility metrics. The

---

idea of using a Gaussian process emulator as a fast approximation for TALYS was also discussed, enabling the active parameter space to be explored more efficiently, at the cost of introducing another source of uncertainty. Gaussian processes are discussed further in the next chapter.



# Chapter 4

## Gaussian process regression

### 4.1 Introduction

In this section some of the key ideas in Gaussian process regression (GPR) are discussed. Section 4.2 gives a brief introduction to GPR and Section 4.3 discusses how parameters are estimated. This is followed by content on two key components of GPR: the covariance kernel and mean function. The chapter finishes with a discussion on the software packages with which some of the modelling in this dissertation was attempted, highlighting some of the key features that made them suitable or otherwise for the work.

### 4.2 Gaussian process regression

In this section a single output of TALYS  $f_i$  is reasoned about. No loss of generality is incurred by proceeding in this way, as all of the TALYS outputs were modelled independently, and the same reasoning applies to all of them equally, so the  $i$  subscript is dropped for convenience.

A GPR model was used as an emulator for TALYS in order to predict, with associated uncertainty estimates, the simulator output  $f$  at unobserved inputs  $\mathbf{x}^\dagger \in \mathcal{X}^\dagger$ . If TALYS is run twice for some  $\mathbf{x}$ , the two outputs  $f^{(1)}(\mathbf{x})$  and  $f^{(2)}(\mathbf{x})$  will be the same. Because TALYS is deterministic,  $f^{(1)}(\mathbf{x}) = f^{(2)}(\mathbf{x}) \forall \mathbf{x}$ . Consequently the uncertainty around  $f(\mathbf{x})$  reduces to zero once it has been observed<sup>1</sup>. However, for  $f(\mathbf{x}^\dagger)$  where the output has not been observed (the simulator has not been run at  $\mathbf{x}^\dagger$ ), there is uncertainty about its output due to incomplete knowledge about the process. The models and their implementation are understood, but not well enough that it is known what they will predict for a given  $\mathbf{x}$  without running the simulator. This uncertainty is often described as

---

<sup>1</sup>provided that, as is the case with TALYS, the simulator is deterministic and consequently there is no uncertainty due to random behaviour, often called aleatoric uncertainty, associated with the process  $f$ .

epistemic, and in the Bayesian framework, statistical models are used to describe our belief about unobserved quantities, and the uncertainty associated with that belief. Consequently, the outputs  $f(\mathcal{X}^\dagger)$  are modelled as random variables. It is assumed that  $f$  is the sum of a deterministic trend function,  $\mu(\cdot)$  and a zero mean Gaussian process,  $Z(\cdot)$  [16]:

$$f(\mathbf{x}) = \mu(\mathbf{x}) + Z(\mathbf{x}). \quad (4.1)$$

A Gaussian process (GP) is an infinite collection of random variables, any finite number of which has a multivariate Gaussian distribution [17]. In practice the interest lies with two subsets of all possible  $f(\mathbf{x})$ , the observed simulator outputs  $f(\mathcal{X}^*)$ , and the outputs we wish to predict  $f(\mathcal{X}^\dagger)$ . Before any simulator outputs are observed, our belief about the joint distribution of the observed and unobserved outputs is modelled as

$$\begin{bmatrix} \mathbf{f}(\mathcal{X}^*) \\ \mathbf{f}(\mathcal{X}^\dagger) \end{bmatrix} \sim MVN \left( \begin{bmatrix} \mu(\mathcal{X}^*) \\ \mu(\mathcal{X}^\dagger) \end{bmatrix}, \begin{bmatrix} \Sigma^* & \Sigma^{*\dagger} \\ \Sigma^{\dagger*} & \Sigma^\dagger \end{bmatrix} \right) \quad (4.2)$$

For some mean function  $\mu(\cdot)$  and some covariance matrices  $\Sigma^*, \Sigma^\dagger$ , which are functions of the  $\mathcal{X}^*$  and  $\mathcal{X}^\dagger$  respectively, and  $\Sigma^{*\dagger} = (\Sigma^{\dagger*})^{-1}$ , which are functions of both  $\mathcal{X}^*$  and  $\mathcal{X}^\dagger$ .

The function  $Z(\cdot)$  is constrained to have specific properties in order to qualify as a GP, as the  $\Sigma$  must have the similar fundamental properties as the covariance matrix of a multivariate normal distribution (Section 4.5). The function  $\mu(\cdot)$  can be any normal linear regression function (Section 4.6).

Both  $\mu(\cdot)$  and  $Z(\cdot)$  are functions with parameters that must be learned about from the  $(\mathcal{X}^*, f(\mathcal{X}^*))$ . Parameter estimation methods vary between software implementations (Section 4.8). After learning the parameters, the distribution in Equation (4.2) is fully characterised for a given  $\mathcal{X}^\dagger$ , and the outputs at these unknown points are emulated as the joint conditional distribution  $\mathbf{f}(\mathcal{X}^\dagger) | \mathbf{f}(\mathcal{X}^*)$ .

### 4.3 Parameter estimation

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

$$\mu(\mathbf{x}) = \sum_{i=1}^b h_b(\mathbf{x})^T \beta_b \quad (4.3)$$

Where  $b$  is the number of basis functions in the model. These basis functions are chosen by the user. Although any linear combination of basis functions is permissible, some software implementations offer more flexibility than others (Section 4.8). The  $\beta_b$  must be learned from the data.



Each element of the covariance matrix in Equation (4.2) is computed by using some function  $c(\mathbf{x}, \mathbf{x}') = \sigma^2 r(\mathbf{x}, \mathbf{x}')$ , where  $r$  is a correlation function that depends on  $|\mathbf{x} - \mathbf{x}'|$  where  $\mathbf{x}$  and  $\mathbf{x}'$  are two points from  $\mathcal{X}$ . So the covariance between two of the  $f$ s depends in some way on the distance between their  $\mathbf{x}$ -coordinates. The function  $r$  has  $p$  hyperparameters  $\gamma_1, \gamma_2, \dots, \gamma_p$ , one for each element of  $\mathbf{x}$ , which must be learned from the observed data, along with the variance parameter  $\theta$ . An optional noise parameter, sometimes called a nugget, can be learned (see Section 4.5 for more on the  $\gamma$  and the noise parameter). The vector of model parameters is denoted  $\theta$ .

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  $p + 1$ ,  $p$   $\gamma$ s and one  $\sigma$ . In a Bayesian context an expression proportional to the joint posterior distribution of the parameters  $\theta$  given the observed data<sup>2</sup>  $f(\mathcal{X}^*)$  is derived  $\pi(\theta|f(\mathcal{X}^*))$  using the product of the likelihood of the data given the parameters  $L(f(\mathcal{X}^*)|\theta)$  and the priors for the parameters  $\pi(\theta)$ . One approach is to then use MCMC to generate samples from  $\pi(\theta|f(\mathcal{X}^*))$ . 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 approach 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 the Dice Kriging [16] software package was used, which implements improper uniform priors [16] for the  $\theta$  and as a result  $\pi(\theta|f(\mathcal{X}^*)) \propto L(f(\mathcal{X}^*)|\theta)$ . Consequently ML estimation was used to learn the  $\theta$ , which is equivalent to finding the maximum posterior mode in the uniform prior case, provided the domain of the likelihood is the same as, or a subset of, the domain of the priors. The training points  $f(\mathcal{X}^*)$  are assumed to have a joint Gaussian distribution with mean  $\mu(\mathcal{X}^*) = \mathbf{h}(\mathcal{X}^*)\beta$  and covariance  $\Sigma^* = \mathbf{C} = \sigma^2 \mathbf{R}$ , where  $\mathbf{R}$  is a  $k \times k$  covariance matrix with the  $i, j$ -th element equal to  $r(\mathbf{x}_i, \mathbf{x}_j)$  and where  $k$  is the number of training points and<sup>3</sup>  $i \in 1, 2, \dots, k, j \in 1, 2, \dots, k$ . Hence the likelihood function that is maximised is

$$L(f(\mathcal{X}^*)|\theta) = \frac{1}{(2\pi)^{k/2} |\sigma^2 \mathbf{R}|^{1/2}} \exp \left( -\frac{1}{2} \left( f(\mathcal{X}^*) - \mathbf{H}^T \beta \right)^T \left( \sigma^2 \mathbf{R} \right)^{-1} \left( f(\mathcal{X}^*) - \mathbf{H}^T \beta \right) \right) \quad (4.4)$$

Where  $\mathbf{H}$  is the  $k \times p$  matrix with each row  $\mu(\mathbf{x})$  for one of the  $k$  training points and each column one of the  $p$  parameters. Each time the likelihood is evaluated,

<sup>2</sup>The observed data includes both the outputs  $f(\mathcal{X}^*)$  and the inputs  $\mathcal{X}^*$  but the inputs are omitted here for brevity.

<sup>3</sup>Indices  $i$  and  $j$  are borrowed here and not again; elsewhere  $i$  indexes over processes/measurements/emulator outputs and  $j$  indexes over proposal points from  $\mathcal{X}$ .

the matrix  $\mathbf{R}$  needs to be inverted, which requires  $\mathcal{O}(k^3)$  computation<sup>4</sup> [18], which means the computational expense can become prohibitively large if the likelihood has to be evaluated a great many times and/ or for a great many training points. 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, which usually requires the likelihood to be evaluated many more times.

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

$$\hat{\beta} = (\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{R} f(\mathcal{X}^*) \quad (4.5)$$

and

$$\hat{\sigma}^2 = \frac{1}{k} \left( f(\mathcal{X}^*) - \mathbf{H}^T \hat{\beta} \right)^T \mathbf{R}^{-1} \left( f(\mathcal{X}^*) - \mathbf{H}^T \hat{\beta} \right) \quad (4.6)$$

which look similar to the predictors for the similar terms in ordinary linear model theory, except for the presence of  $\mathbf{R}$  which accounts for correlation in the residuals. With this re-parametrisation, the likelihood need only be maximised with respect to the  $\gamma$ , 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 [16]. Once the parameter estimates are obtained the joint distribution of  $f(\mathcal{X}^*)$  and  $f(\mathcal{X}^\dagger)$  is fully characterised. Consequently the joint conditional distribution  $f(\mathcal{X}^\dagger | f(\mathcal{X}^*), \theta)$  can be found, which is used to emulate the simulator at unobserved  $\mathbf{x}$ .

## 4.4 Conditioning on the data

Assume that  $f(\mathcal{X}^*)$  and  $f(\mathcal{X}^\dagger)$  in Equation (4.2) each have length one. Once estimates for the parameters  $\hat{\theta}$  have been computed, 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. There is some positive correlation between these two simulator outputs; perhaps because their  $\mathbf{x}$ -coordinates are close together. The left hand plot represents our prior belief about  $f$  evaluated for any two active parameter vectors  $\mathbf{x}_1$  and  $\mathbf{x}_2$  before the simulator is run at these points, and can be founded by plugging  $\mathbf{x}_1$  and  $\mathbf{x}_2$  into the mean and covariance functions discussed in Sections 4.5 and 4.6. The right hand plot shows what happens when the simulator output at  $f(\mathbf{x}_1)$  is observed. Our belief about the simulator output at  $\mathbf{x}_2$  changes. The joint bivariate density is three-dimensional, and one can imagine slicing it with the plane at  $f(\mathbf{x}_1) = f(\mathbf{x}_1^*)$  and observing the exposed surface, whose two dimensions

<sup>4</sup>The computational expense is proportional to  $n^3$

are the  $f(\mathbf{x}_2)$  axis and a univariate Gaussian density. This is the conditional density of  $f(\mathbf{x}_2)|f(\mathbf{x}_1^*)$ . For a single unevaluated simulator output at  $\mathbf{x}^\dagger$  with observed simulator outputs  $f(\mathcal{X}^*)$  the conditional mean,  $E[f(\mathbf{x}^\dagger)]$  and variance,  $V[f(\mathbf{x}^\dagger)]$  are given by

$$E[f(\mathbf{x}^\dagger)] = \mathbf{h}(\mathbf{x}^\dagger)^T \hat{\beta} + \mathbf{c}(\mathbf{x}^\dagger) \mathbf{C}^{-1} \left( f(\mathcal{X}^*) - \mathbf{h}(\mathcal{X}^*)^T \hat{\beta} \right) \quad (4.7)$$

and

$$V[f(\mathbf{x}^\dagger)] = c(\mathbf{x}^*, \mathbf{x}^*) - c^T(\mathbf{x}^*) \mathbf{C}^{-1} c(\mathbf{x}^*) + \quad (4.8)$$

$$\left( \mathbf{h}(\mathbf{x}^\dagger) - \mathbf{h}^T(\mathbf{x}^*) \mathbf{C}^{-1} \mathbf{c}(\mathbf{x}^\dagger) \right)^T \left( \mathbf{h}^T(\mathbf{x}^*) \mathbf{C}^{-1} \mathbf{h}(\mathbf{x}^*) \right)^{-1} \left( \mathbf{h}(\mathbf{x}^\dagger) - \mathbf{h}^T(\mathbf{x}^\dagger) \mathbf{C}^{-1} \mathbf{c}(\mathbf{x}^\dagger) \right)$$

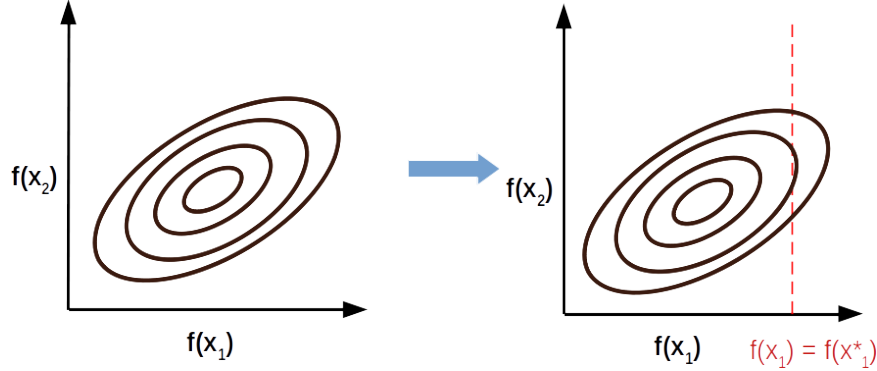


Figure 4.1: Projection of the joint distribution of a simulator output for two active parameter settings onto a plane. The ovals are lines of constant probability density. Once one of the outputs is observed, the other has a univariate Gaussian distribution, conditional on the observation.

where  $\mathbf{c}(\mathbf{x}^\dagger)$  denotes the vector  $(c(\mathbf{x}^\dagger, \mathbf{x}_1^*), c(\mathbf{x}^\dagger, \mathbf{x}_1^*), \dots, c(\mathbf{x}^\dagger, \mathbf{x}_k^*))$  where  $k$  is the number of observed training points. So  $c(\mathbf{x}^\dagger)$  is the vector of covariances between  $\mathbf{x}^\dagger$  and each of the  $k$  observed points. Chapter 2 of [19] shows how these equations arise from properties of multivariate normal distributions.

If Equations (4.7) and (4.8) are used to predict the simulator output at an observed point  $\mathbf{x}_d^*$  ( $d \in 1, 2, \dots, k$ ) from one of the training runs then  $\mathbf{c}^T(\mathbf{x}_d^*) \mathbf{C}^{(-1)} = \mathbf{C}^{(-1)} \mathbf{c}(\mathbf{x}_d^*) = \mathbf{e}_d$  where  $\mathbf{e}_d$  is the  $k$ -vector where the  $d$ -th element is one and all the other elements are zero. This arises from properties of the covariance function described in Section 4.5. Consequently

$$\begin{aligned} E[f(\mathbf{x}_d^*)] &= \mathbf{h}(\mathbf{x}_d^*)^T \hat{\beta} + f(\mathbf{x}_d^*) - \mathbf{h}(\mathbf{x}_d^*)^T \hat{\beta} \\ &= f(\mathbf{x}_d^*) \end{aligned} \quad (4.9)$$

and

$$\begin{aligned} V[f(\mathbf{x}_d^*)] &= c(\mathbf{x}_j, \mathbf{x}_j) - c(\mathbf{x}_j, \mathbf{x}_d^*) + (\mathbf{h}(\mathbf{x}^*) - \mathbf{h}(\mathbf{x}^*)) \mathbf{h}(\mathbf{x}^*) (\mathbf{h}(\mathbf{x}^*) - \mathbf{h}(\mathbf{x}^*)) \\ &= 0 \end{aligned} \tag{4.10}$$

and as such the observed  $f(\mathcal{X}^*)$  are interpolated exactly with zero uncertainty. This is desirable in cases where the data are observed with no error, as is the case with a deterministic computer simulation. Intuitively we would expect this to be true as Equations (4.7) and (4.8) define the conditional distribution of  $f(\mathbf{x}^\dagger)|f(\mathcal{X}^*)$ , and if  $f(\mathbf{x}^\dagger) = f(\mathbf{x}^*)$  then the distribution of  $f(\mathbf{x}^*)|f(\mathcal{X}^*)$  is just  $f(\mathbf{x}^*)$ . Stochastic behaviour in the function being emulated can also be encapsulated by estimating a ‘nugget’ parameter from the data, which is discussed next, along with other properties of the covariance function.

## 4.5 Covariance function

The covariance between  $f(\mathbf{x})$  and  $f(\mathbf{x}')$  is computed from the parameter vectors  $\mathbf{x}$  and  $\mathbf{x}'$ . A key assumption of Gaussian processes is that points that are close together in  $\mathbf{x}$ -space should be close together in  $f(\mathbf{x})$ -space, and that when predicting  $f(\mathbf{x}^\dagger)$ , then observed points close to  $\mathbf{x}^\dagger$  more strongly influence the prediction than those that are far away. The presence of the  $\mathbf{x} - \mathbf{x}'$  term in correlation functions encapsulates this idea. Another key assumption is that some elements of  $\mathbf{x}$  bear more influence on neighbouring points than others. and that information on this can be found from the data. The hyperparameters  $\gamma$  of the function  $c(\cdot, \cdot)$  encapsulate this idea. The function  $c(\cdot, \cdot)$  must produce covariances matrices for  $f$  that have several properties:

- Covariance matrices are symmetric in that for matrix  $\mathbf{A}$ ,  $\mathbf{A} = \mathbf{A}^{-1}$ . Consequently the covariance function  $c(\cdot, \cdot)$  must also be symmetric in that  $c(\mathbf{x}, \mathbf{x}') = c(\mathbf{x}', \mathbf{x})$ .
- 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 it can be expressed as a product of a correlation matrix  $\mathbf{R}$  and a constant variance  $\sigma^2$ . Table 4.1 shows the univariate version of the covariance functions available in DiceKriging [16], or more appropriately correlation functions, as each is multiplied by  $\sigma$  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.

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

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

The Matérn correlation function with parameter  $\nu = \frac{5}{2}$  is a popular choice and is the default option in both the DiceKriging and RobustGaSP [20] software packages. The  $\nu$  parameter in the Matérn family controls the smoothness of the function. As  $\nu \rightarrow \infty$ , 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 [20], which may be unrealistic, and cause numerical issues in the parameter estimation routine. In this dissertation, the Matérn  $\frac{5}{2}$  correlation function is used.

The correlation functions  $r(\cdot, \cdot)$  in Table 4.1 take two scalar inputs and return a single scalar output<sup>5</sup>. To extend this to compute correlations for multivariate  $\mathbf{x}$  the most common approach is to take the product of the univariate correlations:

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

where  $p$  is the number of elements of  $\mathbf{x}$ . For each of element of  $\mathbf{x}$  a separate  $\gamma$  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, by inflating or shrinking the effective distance between the points in the relevant  $\mathbf{x}$ -component. The  $\gamma$  are often called length-scale parameters. For predictions at points in  $\mathcal{X}$  further away from the observed  $\mathbf{x}^*$ , the predictive variance will tend to  $\sigma^2$ , as the  $(x - x')$  terms in the correlation functions go to infinity, and as such the exponential terms in Table 4.1 tend to one. If  $\gamma$  is changed from 1 to 0.5 for one of the components of  $\mathbf{x}$ , the average amount of variation observed in a fixed interval along the component axis would double. There is an interplay between the  $\gamma$  and  $\sigma$ . The smaller the  $\gamma$  are the more the variation arises due to the function, and hence  $\sigma^2$  would be smaller, as large fluctuations in  $f$  can be explained by function variation rather than noise. Higher values for  $\gamma$  are associated with

<sup>5</sup>A function that does this is called a kernel, and hence the  $r(\cdot, \cdot)$  are sometimes called kernels.

higher values for  $\sigma^2$  and imply a constant function with a lot noise. Lower values for  $\gamma$  are associated with lower values for  $\sigma^2$  and imply a white-noise process [17].

Often the function 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  $f(\mathcal{X}^*)$ . The nugget can also be used to address problems with inverting  $\mathbf{R}$  in Equation (4.4) which can arise when its elements are very close to 0 or 1. In this dissertation a nugget parameter was used when training emulators 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 function [17]. The covariance function is the defining element of a GP, but emulation is often boosted by the inclusion of a mean function, which is illustrated with a simple example, along with some other properties of the GP discussed in this section.

## 4.6 Mean function

A GP models our belief about how far a true process might 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. The following example gives an illustration of this.

Assume the simulator  $f(x) = x^2 - 5x$  from Section 3.10 and shown in Figure 4.2 is to be emulated using a GP and that the simulator can only be run at 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. There is a good amount of data around the point of inflection at  $x = 2.5$ . This is where the function varies the most.

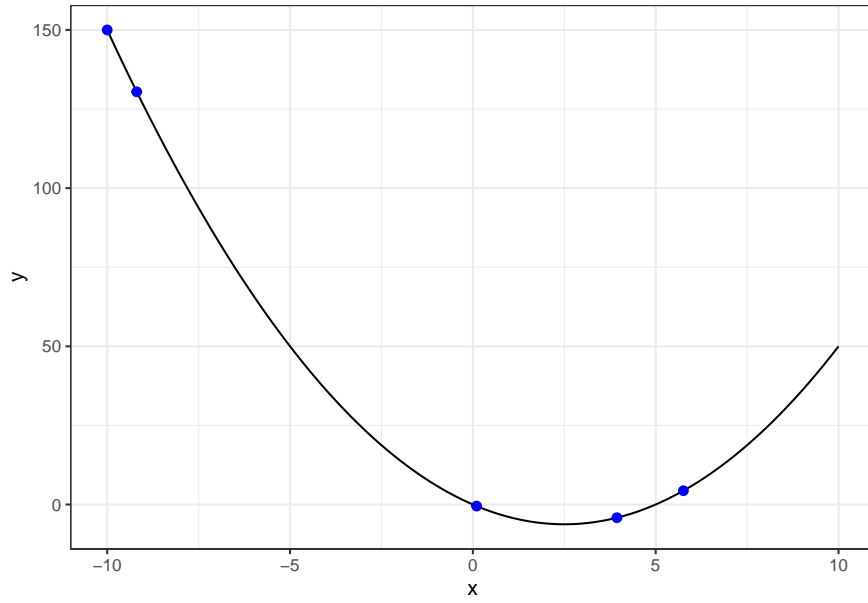


Figure 4.2: A simple function to be emulated, with the five points used to build the emulator shown. There is a region with no training data from approximately  $x=-9$  to  $x=0$ .

First a zero-mean GP emulator is built with a Matérn  $\frac{5}{2}$  correlation function, without a ‘nugget’. The emulator is then used to predict, with 95% posterior predictive intervals, the value of the function at many points between  $x = -10$  and 10. A second emulator is built using a simple first-order<sup>6</sup> mean function, which requires two more parameters to be estimated from the data. The results are shown in Figure 4.3. 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, because some part of the function variation is encapsulated by the mean function.

---

<sup>6</sup>as in  $y = mx + c$

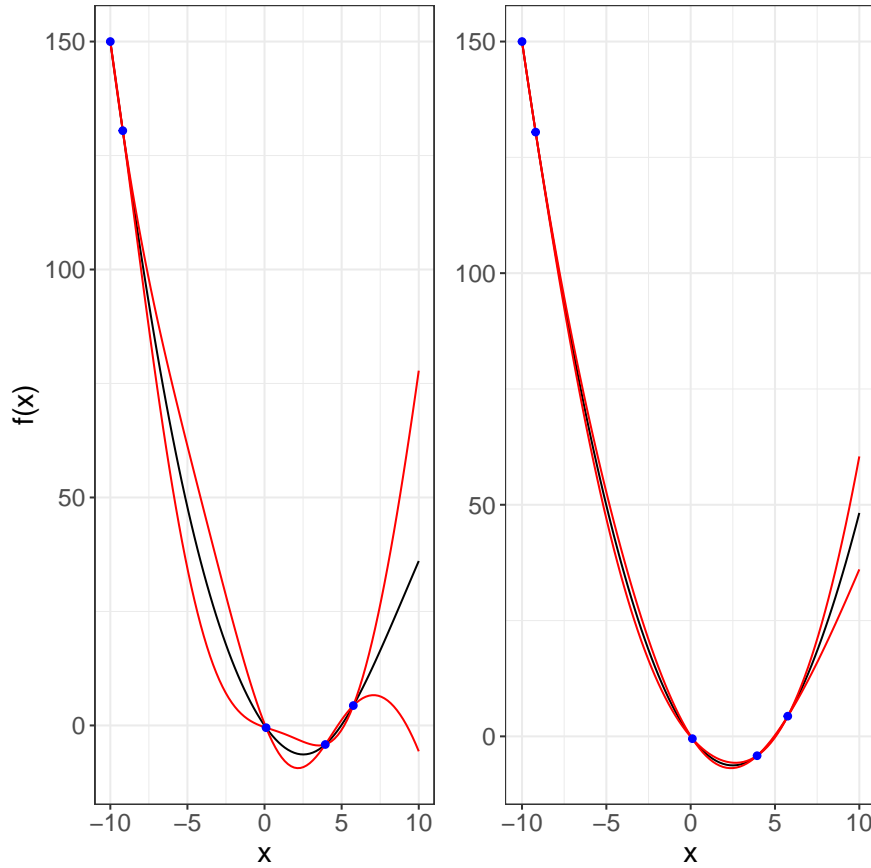


Figure 4.3: Predictions using a zero-mean (left) and first-order mean (right) GP emulator trained on the five points shown for the function, with 95 percent predictive intervals shown. The emulators predict the training points exactly, and the width of the predictive intervals increases further from 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 shown in Figure 4.4, 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 mean function, albeit both with such wide posterior predictive intervals as to render the results quite useless.



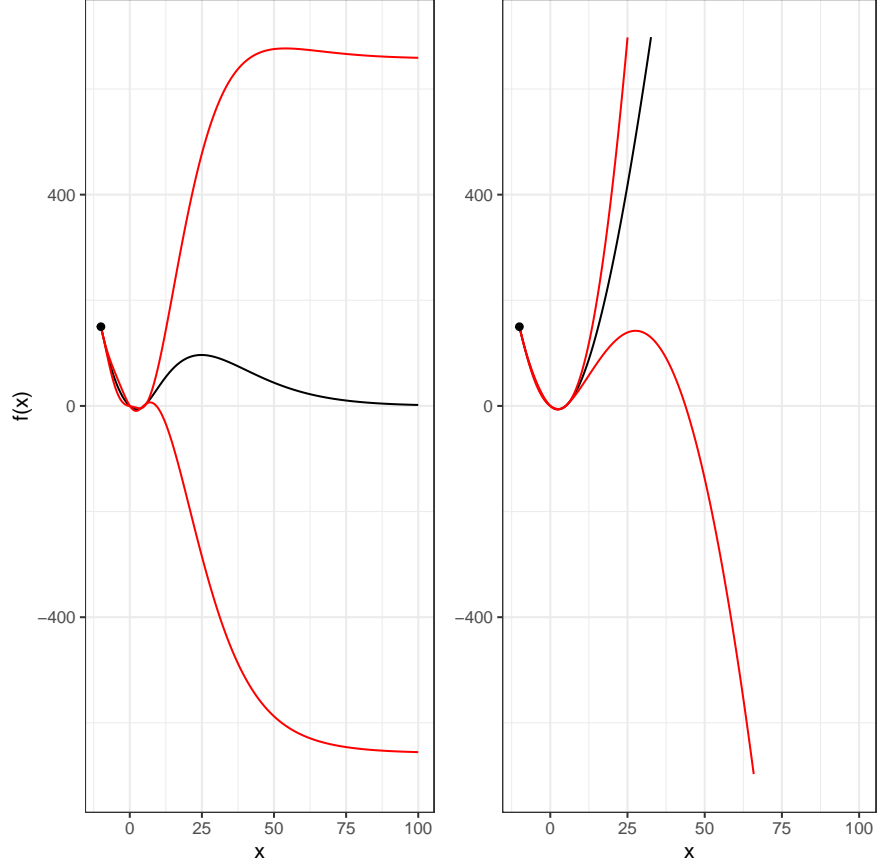


Figure 4.4: Behaviour of the emulator as it predicts further from the training data. Both the zero-mean (left) and first-order mean (right) emulators tend to their respective mean functions as the distance from the observed data grows larger.

Theoretically, the mean function can be any linear (in the coefficients) function of the inputs  $\mathbf{x}$ . Some sources [21] favour a first order-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

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

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  $\beta$ , and the computational resource required for this was not available. The choice of mean function, and

other modelling choices, can also be a function of the software package used, and we discuss the choice of software implementation in the next section.

## 4.7 Model complexity and the predictive distribution

The denominator in Equation (4.6) should be  $k - p$ , where  $p$  is the number of parameters in the GP mean function (Section 4.6), and consequently the predictive distribution of the GP emulator is a Student-t with  $k - p$  degrees of freedom. Often in practice, as is the case with the software used in this dissertation, it is assumed that  $k \gg p$  and as such the number of degrees of freedom is large enough that the predictive distribution is approximately Normal. In what follows it is assumed that the predictive distributions of the emulators are Gaussian, as the number of simulator runs used to train the emulators numbered in the hundreds, and the number of parameters was in the tens. Care must be taken with model complexity, if  $k$  is not much greater than  $p$ , an assumption of Gaussian predictions may not be valid, and the predictive intervals significantly wider than assumed.

## 4.8 Software implementations

Three software packages were trialled for the GP emulation in this dissertation, the R package RobustGaSP [20], the Python module scikit-learn [22] GaussianProcessRegressor, and the R package DiceKriging [16].

RobustGaSP package distinguishes itself from the other two by implementing a non-uniform prior for the GP parameters and finding their maximum posterior modes as point estimates. 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 588 emulators using hundreds of data points at a time. Crucially, it did not expose a public method for generating a predictive covariance matrix, which was required for model validation, as discussed in Section 5.2.

GaussianProcessRegressor from scikit-learn 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, 588 emulators were trained in batches 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 across hundreds of models. 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 modelling process workflow, so the chapter finishes with a summary of GP modelling considerations, and a reminder of the purpose of GPs in this dissertation.

## 4.9 Conclusion

In this chapter some of the theory and the main building blocks of GPs were presented. In summary, assuming adequate training data has been collected on the function to emulate:

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. This gives a model for the joint distribution of the unobserved and observed points, from which the conditional distribution of the unobserved points is derived. It is this conditional distribution that is used to predict at the unobserved points.

The simulator output an unobserved  $\mathbf{x}$  is a Gaussian random variable, which implies a point estimate for the function evaluated at that  $\mathbf{x}$  (the mean) and a quantification of the uncertainty of that point estimate (the variance). These estimates are obtained in a fraction of the time it would take to run the simulator. These properties make GP emulation a vital tool in history matching, which is discussed in the next section.



# Chapter 5

## Results

### 5.1 Introduction

In this section is described the history matching that was carried out using the simulator TALYS and experimental data from EXFOR as described in Chapter 2, following the methodology in described in Chapter 3, and emulating the simulator as described in Chapter 4. Three waves of history matching were carried out. The first wave and subsequent analysis is described in detail in Sections 5.2, 5.3 and 5.4. The results and analysis from subsequent waves are given in Sections 5.5 and 5.6.

### 5.2 First wave methodology

A total of 28 active input parameters were considered in the history matching analysis. The design grid for the first wave training runs were generated as follows. A 300 point Latin hypercube design was generated on  $U[0, 1]^{28}$  using the function ‘maximinLHS’ from the R package ‘lhs’ [12]. These points then underwent a linear transformation onto  $U[-1, 1]^{28}$  and finally were exponentiated with base 10. This enabled exploration of the active parameter space both above and below the default parameter values (see Section 2.3).

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 reactions for which relevant experimental data were available, as shown in Table 5.1, 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 588 TALYS outputs  $f$  corresponding to the 588 experimental observations. A first order linear mean function was used for each emulator, which required learning 29 parameters from the data. Constant and 0 mean functions were

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

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 to aid the parameter estimation routine (Section 4.3). 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  $[-1,1]$  for training the models. The outputs were left unscaled in order to avoid potential errors when re-scaling the error terms for computing 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 scaled 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 [23], 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. As a result of this, 2% of the valid emulators could be discarded on average, which is the price of protecting against the consequences of predicting with bad emulators. Of the 588 models, 124 were discarded this way, leaving 464.

Figure 5.1 illustrates the importance of carrying out emulator evaluation. The Figure shows evaluation run results for an emulator for the simulated cross-section at 6.19 MeV for the reaction (n,p). This emulator was rejected at evaluation. 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 try and predict 15.2 barns at the same active parameter values as the simulator predicts 15.2 barns. From the plot, it appears that the emulator consistently under-predicts the simulator in this region.

Consequently, 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. Hence, it is important to carry out validation on the emulators to ensure good out-of-sample predictive performance.

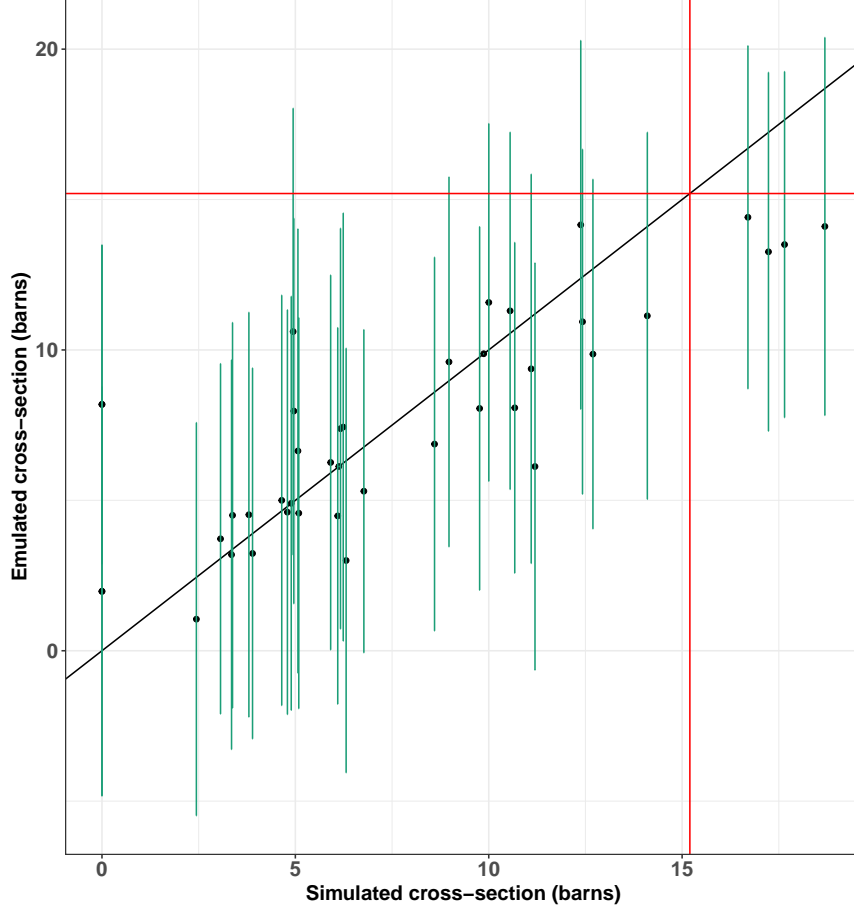


Figure 5.1: Validation run results for an emulator for the cross-section for the (n,p) reaction at energy 6.19 MeV. The vertical lines show the two standard deviation predictive intervals of the emulator and the  $x=y$  line has been plotted for reference. The observed cross section value is shown as 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.

### 5.3 First wave analysis

The computed scaled Mahalanobis distances for the accepted models for (n,tot) are plotted in Figure 5.2. 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.

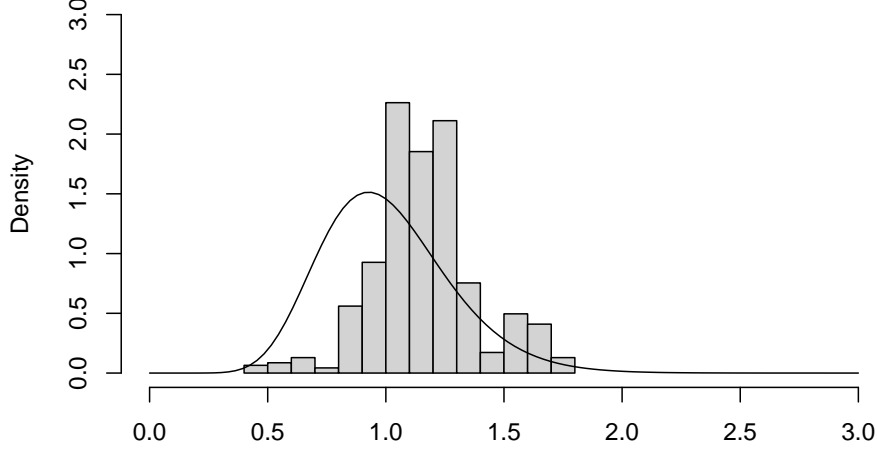


Figure 5.2: Histogram of scaled Mahalanobis distances for valid (n,tot) models. The hypothesised scaled- $F_{30,271}$  curve is plotted over the top. The distribution of Mahalanobis distances looks reasonably similar to the hypothesised distribution.

The 464 models were used to find non-implausible active parameter values for wave two. The number of potential points that could be evaluated was limited by memory and compute power. A total of 142860 points were selected at random from the 28 dimensional active parameter space were selected. Implausibility metrics were generated for each of the 464 models for each of the 142860 proposal points. Equation (3.4) was evaluated to compute 464 one-dimensional implausibility measures for each proposal  $\mathbf{x}$  and Equation (3.5) was evaluated to compute one multidimensional implausibility metric for each proposal  $\mathbf{x}$  using the corresponding experimental observations and uncertainties as described in Chapter 3.

Two criteria were used for accepting a proposal point  $\mathbf{x}_j$ , First that  $I_j^{(uv)} < 3$  and second that  $I_j^{(mv)} < \chi_{464, .95}^2$ . The second highest of the 464 implausibility measures for proposal  $\mathbf{x}_j$  was chosen as  $I_j^{(uv)}$ . 272 points met these criteria - 0.019% of the proposed points.

Different acceptance criteria for the two implausibility measures were examined; Table 5.2 shows some of the results of this analysis, where it can be seen that the choice which univariate implausibility measure to use was the most important decision in determining the size of the non-implausible active parameter space. Interestingly, choosing the highest univariate implausibility for each proposal resulted in zero non-implausible samples, and so no one sample was non-implausible for every process.



Table 5.2: Number of proposal points accepted as a function of different cut-offs for implausibility metrics.  $I^{uv=a}$  indicates the  $a$ th 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.

	$I_i^{(uv=1)}$	$I_i^{(uv=2)}$	$I_i^{(uv=3)}$
$\chi_{464,.95}^2$	0	268	4381
$\chi_{464,.99}^2$	0	276	4472

## 5.4 Minimum implausibility and optical depth analysis

The sensitivity of implausibility to the inputs can be visualised in 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 (the  $\log_{10}$  of the parameter space) 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 over all the candidate  $\mathbf{x}$ s from each interval was then pulled out and used to determine the hue in that interval in the heat map. The code used to generate these plots is shown in Appendix B.

If the value of the minimum implausibility was high, this might indicate that it is not possible to get TALYS to match experimental measurements when run at that combination of values for those elements of  $\mathbf{x}_j$ . The optical depth plots were constructed by grouping the proposal inputs in the same way. An estimate of the probability of finding a non-implausible input in an interval was 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.

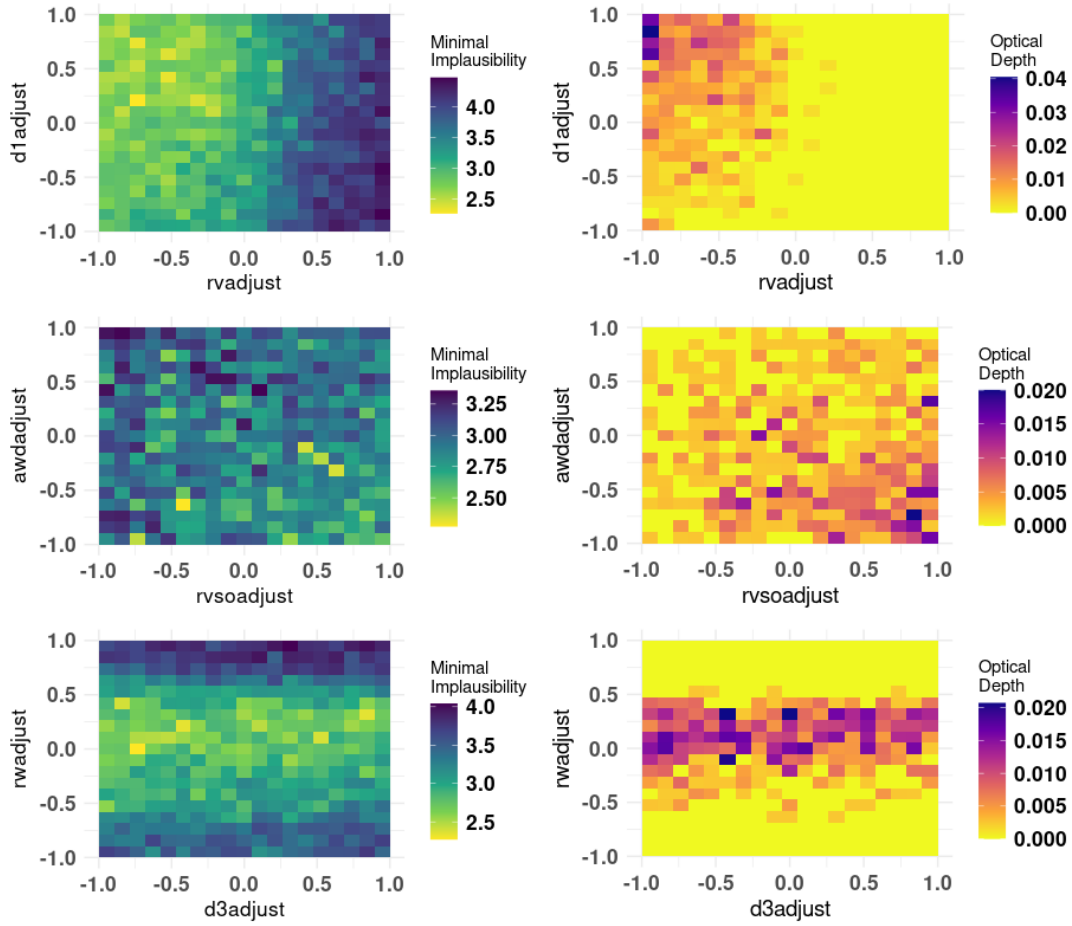


Figure 5.3: Minimum implausibility plots (left column) and optical depth plots (right column) for six of the inputs. The minimum implausibility plots show estimates of the minimum univariate implausibility for values of two of the inputs and the optical depth plots show estimates of the probability of 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 Figure 5.3 shows plots for parameters ‘awadjust’ and ‘rvsoadjust’. This plot also suggests 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’.

## 5.5 Subsequent waves

The 276 non-implausible inputs chosen in Section 5.3 were used as the design for the training runs in wave two. A new set of 588 emulators were built. A further set of test runs of the simulator were carried out. The active input parameter values for these test runs were chosen by generating a large number of proposal samples and evaluating their implausibility using the wave one emulators. The wave two emulators were then validated in the same way as described in Section 5.2. 163 were invalidated, 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 two implausibility analysis, and consequently only 35715 proposal points were examined. Of these, 18043 were accepted as non-implausible, which is just over 50%. The cut-offs used for the implausibility measures were the same as those used in Section 5.3, although the relevant  $\chi^2$  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 3.11).

Having such a large number of non-implausible points for wave two 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 three (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 three, which was a way of attempting to select a set of design points that were as spread out across the non-implausible volume as possible.

The same workflow for building and validating the emulators was carried out for Wave 3, in which 440 emulators were assessed as valid. In wave three, different cut-offs were used for the implausibility measures, as the ability of the emulators to refocus with the wave one and two cut-offs was diminished. Table 5.5 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 two,  $18043/35715 = 0.505$  of the runs were evaluated as non implausible. The proportion of non-implausible space from wave one was 0.19. The product of these proportions is 0.09. Implicitly, the proportion of the input

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 active parameter space evaluated as non-implausible after each wave.

Wave	Runs	$I_j^{(uv)}$	$I_j^{(mv)}$	% of Original Space
1	300	3	$\chi_{464,0.99}^2$	0.19
2	276	3	$\chi_{425,0.99}^2$	0.09
3	600	2	$\chi_{440,0.95}^2$	0.03

space that is plausible before wave one (wave zero) is 1. From Table 5.5 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 the third wave, 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. These samples contained useful information about the relationship between the active parameters and the TALYS outputs and could also be used to allow sampling from the posterior distribution of the active parameters (conditional on the data), as discussed in the next Section.

## 5.6 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 [1]. 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 5.5.

One 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 and the primary interest lies in the implication for the distribution of non-implausible TALYS outputs. One way to learn about the distribution of these non-implausible outputs would be through Monte Carlo (MC) sampling, where TALYS is run at samples of  $\mathbf{x}$  drawn from the posterior distribution of active parameters. The results can be used to learn the properties of the required multivariate normal distributions. 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 [1].

A proposal density was required to allow proposal posterior samples to be generated. A sensible choice was

$$P(\mathbf{x}) = N(\mathbf{x}|\hat{\mu}_x, \kappa\hat{\Sigma}_x) \quad (5.1)$$

where  $\hat{\mu}_x$  and  $\hat{\Sigma}_x$  were the sample mean and covariance matrices of  $\{\mathbf{x}_{ni}\}$ . Random samples were drawn from Equation (5.1), each one a proposal for a sample from  $\pi(\mathbf{x}|z)$ . The constant  $\kappa$  was used to ‘widen’ the search area, which has been seen to make the posterior sampling more effective [1]. The first step was to draw a large set of proposal samples from Equation (5.1). In this dissertation, this was done with the aid of the R package ‘mvtnorm’ [24].  $\kappa$  was set equal to 2.

The likelihood of the data  $\mathbf{z}$  with respect to the proposal samples needed be computed for each proposal. As an approximation to the likelihood

$$L(\mathbf{x}) = p(\mathbf{z}|\mathbf{x}) = N(\mathbf{z}|E[g(\mathbf{x})], V(\mathbf{x})) \quad (5.2)$$

was used.

Each time a proposal  $\mathbf{x}$  was generated from  $P(\mathbf{x})$ , the wave three emulator predictions at this point  $\hat{\mathbf{f}}(\mathbf{x})$  were computed. The mean vector in Equation (5.2) was the vector of predictive emulator means at  $\mathbf{x}$ , each element a prediction of an observation from  $\mathbf{z}$ . 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 (5.2) was computed with respect to  $\mathbf{z}$ .

The likelihood  $P(\mathbf{x})$  was also computed with respect to  $\mathbf{x}$ , and a weight  $w(\mathbf{x}) = \frac{L(\mathbf{x})}{P(\mathbf{x})}$  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 that the simulator inadequacy and observation uncertainty terms were normally distributed, an assumption that is commonly leveraged [2]. 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 three, 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 three. 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 of the active parameters. This also provided an interesting way to investigate if different values of the parameters were better at reproducing different reactions.

This approach resulted in samples from the posterior being generated 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 of the interval  $[-1,1]$ .

Plots for the posterior samples generated in this way are shown in Figures 5.4 and 5.5. The likelihood was evaluated with respect to the 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 be needed 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 5.5. 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 characteristic of a particular reaction. 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.

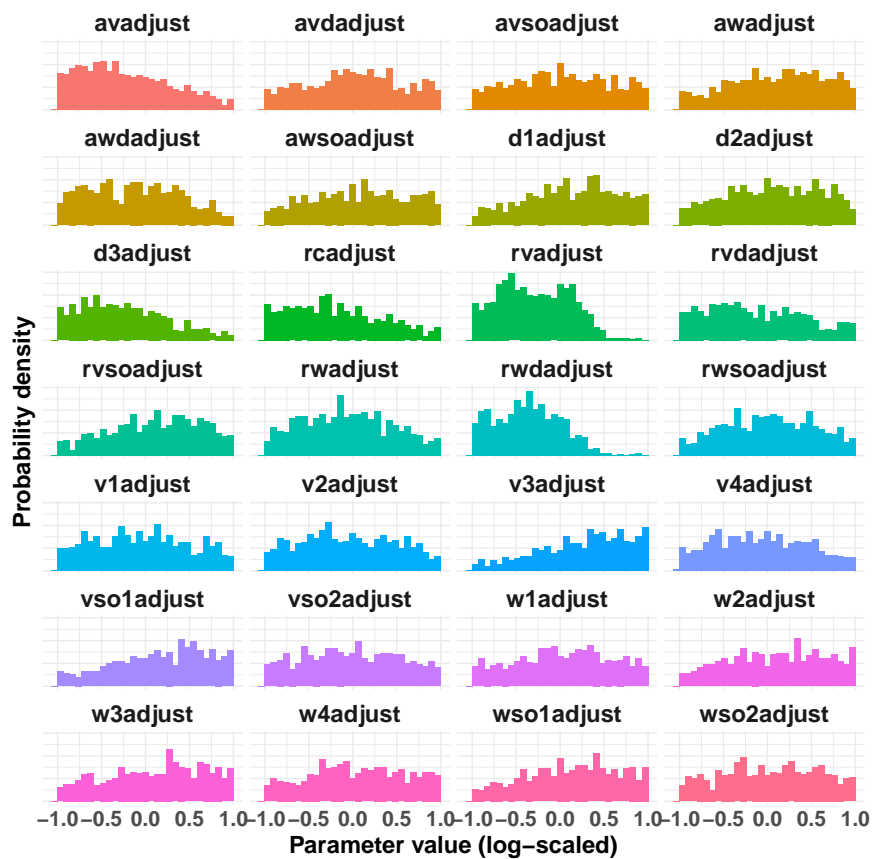


Figure 5.4: Posterior samples for active input parameters when the approximate likelihood is evaluated with respect to (n,p) experimental data. This reaction appears to favour lower values for parameters 'rvadjust' and 'rwdadjust' and higher values for 'v3adjust'.

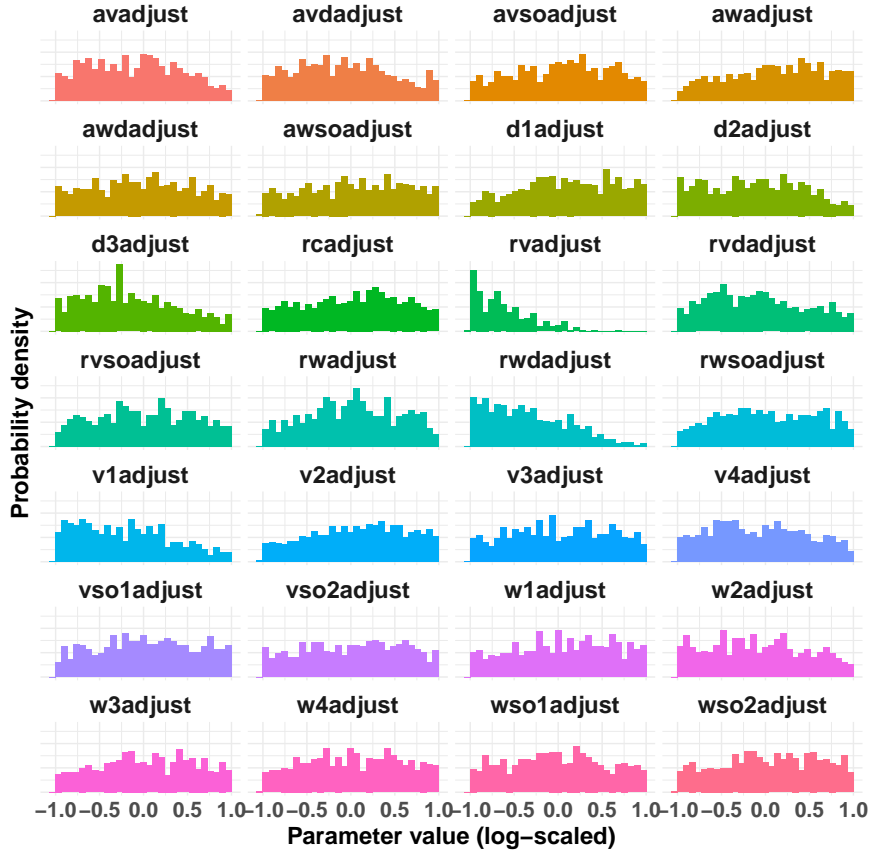


Figure 5.5: Posterior samples for active input parameters when the approximate likelihood is evaluated with respect to (n,a) experimental data. This reaction shows a preference for lower values of 'rvadjust' and 'rwdadjust'.

## 5.7 Conclusion

In this chapter, three waves of history matching were carried out, which was seen to shrink the non-implausible active parameter space to 0.03% of its original size. Sensitivity of the simulator output to a subset of the active parameters was analysed. Samples were drawn from the marginal posterior distributions of the active parameters.

The next chapter ends the dissertation with a discussion on the analysis and on how it could be improved upon and extended.



# 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, 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 [1] and [2], 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 resource 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 or more. 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 6.1 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. Although the two-sigma intervals intersecting the  $x=y$  line do not indicate that the emulator would be validated, it does help as a visual guide for how consistent the emulator is with the model outputs.

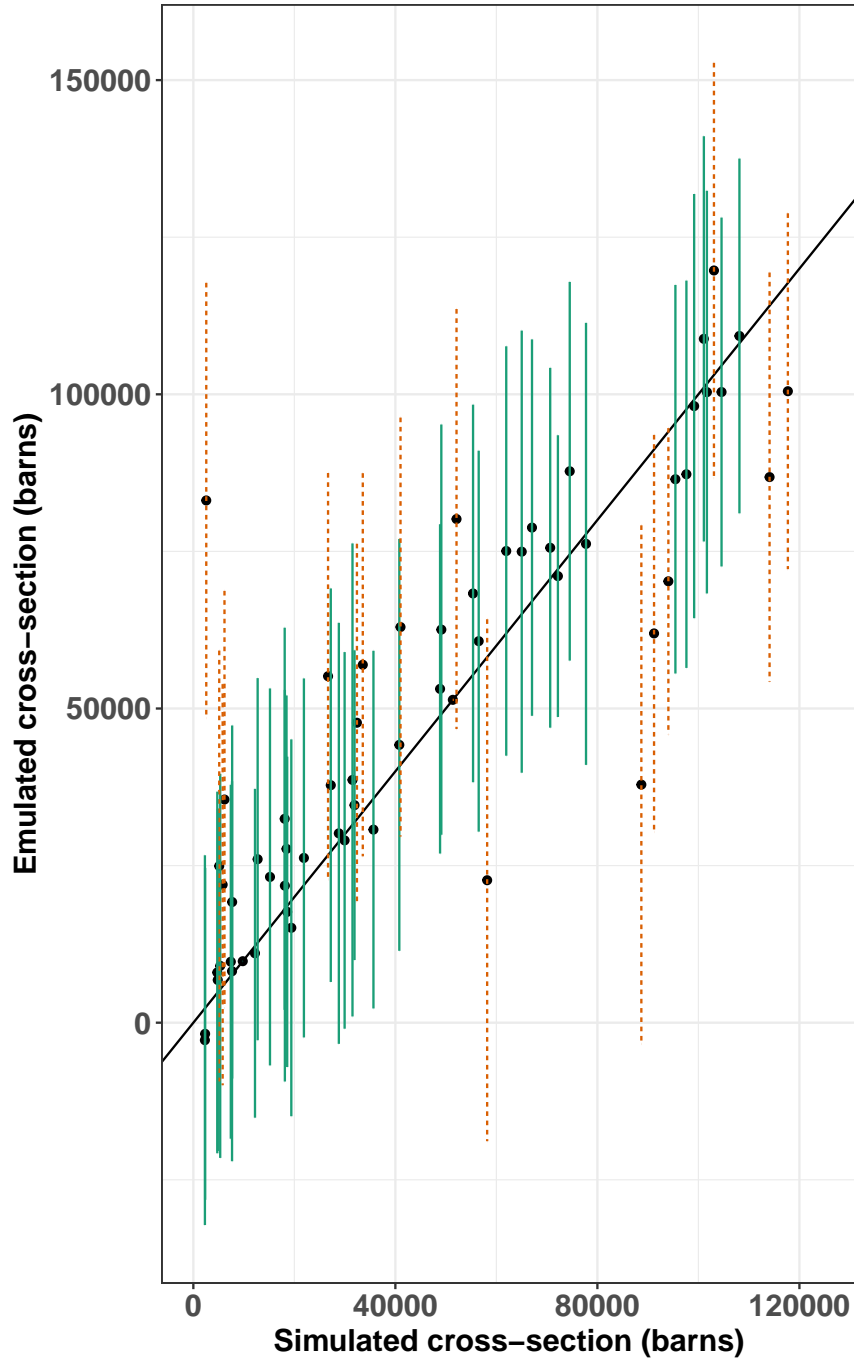


Figure 6.1: Validation run results for the emulator for the  $(n,\text{tot})$  reaction at energy 5.49 MeV. The emulated versus simulated cross-sections are plotted with their two standard deviation error bars, as well as an  $x=y$  line for reference. Dashed error bars indicate emulator results than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

In Figure 6.1, the nominal measured 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 emulator predicts at 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 (3.5) and due to a lack of certainty in how the assumptions pertaining to the measures would hold up under rescaling. However, Figure 6.1 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 [1] nine history matching waves are carried out over 18 active input parameters, and in [2], 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 a more focussed non-implausible space.

In Section 4.8, different Gaussian process software implementations were discussed, with the R package ‘DiceKriging’ being chosen because of its 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 [21] 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 many of the parameter estimation routine to complete successfully.

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 [15] 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 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 [5] and through expert elicitation to better understand how far the simulator predictions should be expected to deviate from reality.

## 6.3 Further work and conclusions

Identifying TALYS input parameter which give rise to good agreement with experiments is a step towards one of the endpoints 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 tool-kit. 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, by carefully defining relevant uncertainties, and by accounting for correlations between similar processes.

# Appendix A

## Spherical optical model potential

TALYS implements several nuclear models, the main one being the spherical optical model potential [3]. 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. For more detail see Chapter 4 of reference [3]. For completeness the equation, along with the mapping between TALYS and equation parameters, are given here.

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

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

<b>TALYS</b>	<b>Equation</b>
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 heatmaps from Section 5.4. 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 if the input parameters were evaluated as plausible, and zero otherwise.

```
# 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: 

Date: 10/9/22

Name of supervisor: Jeremy Oakley

Signature of Supervisor: 

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 official research ethics processes.

# References

- [1] I. Andrianakis *et al.*, “Bayesian history matching of complex infectious disease models using emulation: A tutorial and a case study on HIV in uganda,” *PLoS computational biology*, vol. 11, no. 1, p. e1003968, 2015.
- [2] R. G. Bower, M. Goldstein, and I. Vernon, “Galaxy formation: a Bayesian uncertainty analysis,” *Bayesian analysis*, vol. 5, no. 4, pp. 619–669, 2010.
- [3] A. J. Koning and D. Rochman, “Modern nuclear data evaluation with the TALYS code system,” *Nuclear Data Sheets*, vol. 113, no. 12, pp. 2841–2934, 2012, doi: <https://doi.org/10.1016/j.nds.2012.11.002>.
- [4] N. Otuka *et al.*, “Towards a more complete and accurate experimental nuclear reaction data library (EXFOR): International collaboration between nuclear reaction data centres (NRDC),” *Nuclear Data Sheets*, vol. 120, pp. 272–276, 2014, doi: <https://doi.org/10.1016/j.nds.2014.07.065>.
- [5] G. Schnabel, H. Sjöstrand, J. Hansson, D. Rochman, A. Koning, and R. Capote, “Conception and software implementation of a nuclear data evaluation pipeline,” *Nuclear Data Sheets*, vol. 173, pp. 239–284, Mar. 2021, doi: [10.1016/j.nds.2021.04.007](https://doi.org/10.1016/j.nds.2021.04.007).
- [6] D. A. Brown *et al.*, “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*, vol. 148, pp. 1–142, 2018.
- [7] K. Shibata *et al.*, “JENDL-4.0: A new library for nuclear science and engineering,” *Journal of Nuclear Science and Technology*, vol. 48, no. 1, pp. 1–30, 2011.
- [8] A. Koning, D. Rochman, J.-C. Sublet, N. Dzysiuk, M. Fleming, and S. Van der Marck, “TENDL: Complete nuclear data library for innovative nuclear science and technology,” *Nuclear Data Sheets*, vol. 155, pp. 1–55, 2019.
- [9] P. Helgesson and H. Sjöstrand, “Treating model defects by fitting smoothly varying model parameters: Energy dependence in nuclear data evaluation,” *Annals of Nuclear Energy*, vol. 120, pp. 35–47, 2018, doi: <https://doi.org/10.1016/j.anucene.2018.05.026>.
- [10] H. Iwamoto, “Generation of nuclear data using Gaussian process regression,” *Journal of Nuclear Science and Technology*, vol. 57, no. 8, pp. 932–938, 2020, doi: [10.1080/00223131.2020.1736202](https://doi.org/10.1080/00223131.2020.1736202).

- [11] E. Alhassan, D. Rochman, A. Vasiliev, M. Hursin, A. J. Koning, and H. Ferroukhi, “Iterative Bayesian Monte Carlo for nuclear data evaluation,” *Nuclear Science and Techiques*, vol. 30, no. 50, 2022.
- [12] R. Carnell, *Lhs: Latin hypercube samples*. 2022. Available: <https://CRAN.R-project.org/package=lhs>
- [13] F. Pukelsheim, “The three sigma rule,” *The American Statistician*, vol. 48, pp. 88–91, 1994.
- [14] J. Rougier, “Efficient emulators for multivariate deterministic functions,” *Journal of Computational and Graphical Statistics*, vol. 17, no. 4, pp. 827–843, 2008.
- [15] T. E. Fricker, J. E. Oakley, and N. M. Urban, “Multivariate gaussian process emulators with nonseparable covariance structures,” *Technometrics*, vol. 55, no. 1, pp. 47–56, 2013.
- [16] O. Roustant, D. Ginsbourger, and Y. Deville, “DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization,” *Journal of statistical software*, vol. 51, pp. 1–55, 2012.
- [17] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2. MIT press Cambridge, MA, 2006.
- [18] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis, third edition*. Taylor & Francis, 2013.
- [19] J. Oakley, “Bayesian uncertainty analysis for complex computer codes,” PhD thesis, The University of Sheffield, 1999.
- [20] M. Gu, J. Palomo, and J. Berger, *RobustGaSP: Robust gaussian stochastic process emulation*. 2022. Available: <https://CRAN.R-project.org/package=RobustGaSP>
- [21] C. B. Erickson, B. E. Ankenman, and S. M. Sanchez, “Comparison of Gaussian process modeling software,” *European Journal of Operational Research*, vol. 266, no. 1, pp. 179–192, 2018.
- [22] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in python,” *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.
- [23] L. S. Bastos and A. O’Hagan, “Diagnostics for gaussian process emulators,” *Technometrics*, vol. 51, no. 4, pp. 425–438, 2009.
- [24] A. Genz *et al.*, *mvtnorm: Multivariate normal and t distributions*. 2021. Available: <https://CRAN.R-project.org/package=mvtnorm>