A Bayes Linear Approach to the Branin-Hoo Function

Uncertainty Quantification and Emulator Development

Raul Unnithan

Durham University

January 20, 2025

Contents

Chapter 1 Introduction

1.1 Background

Uncertainty Quantification (UQ) plays a vital role in uncertainty reduction during both optimisation and decision-making.[1] This report uses the Bayes Linear (BL) approach to develop statistical emulators that approximate the outputs of computationally expensive simulators. It explicitly accounts for similar input uncertainty induced by links between models in large networks.[2] The BL approach is more efficient at building emulators than a complete Bayesian analysis due to primitive Expectation treatment, so the necessary objects required for specification are just the Expectations, Variances and Covariances of the random quantities of interest.[3] This report applies BL to the rescaled Branin-Hoo function, a benchmark problem in Emulation. It works for testing emulator performance because it has a complex, non-linear shape with multiple local minima.[4]

1.2 Outline

The first aim was to understand the rescaled Branin-Hoo function by establishing a foundation for emulator training and evaluation. Next, the report delved into the theory of and implements 1D and 2D emulators, applying BL methods to approximate the function. Surface plots and contour mappings are then plotted to compare emulator predictions and the actual function outputs. By approximating this function, this report assesses BL emulation effectiveness in capturing underlying model behaviour and addressing more uncertain prediction areas. This project also covers the strengths and limitations of the BL approach using 1D and 2D emulators. In 2D Emulation, Latin Hypercube Sampling (LHS) is used to select design points, ensuring thorough input space coverage by stratifying each input variable's range, reducing variance and improving estimate accuracy in cases where specific input components dominate outputs.[5] Variance surface plots are then analysed to refine the emulator by identifying the regions of high and low uncertainty across the input domain. Finally, history matching and implausibility analysis are used to iteratively rule out unfeasible parameter space regions, refining the emulator and improving predictions of plausible model outputs.[6]

Chapter 2 Methodology and Results

2.1 Model Exploration

2.1.1 Rescaled Branin-Hoo Function

The rescaled Branin-Hoo function was selected from the Virtual Library of Simulation Experiments. It is defined as:

$$f'(x_1, x_2) = \frac{\left(x_2' - \frac{5.1{x_1'}^2}{4\pi^2} + \frac{5x_1'}{\pi} - 6\right)^2 + \left(10 - \frac{10}{8\pi}\right)\cos(x_1') - 44.81}{51.95},$$

where: $x_1' = 15x_1 - 5$ and $x_2' = 15x_2$. This function is rescaled to the unit square to allow for different method comparisons.

The rescaled function has the same properties as the original, and it has three global minima located at $f'(x_1, x_2) = 0.398$ at the points:

$$x_1'^* \approx 0.698,\ 0.961,\ 0.902, \quad x_2'^* \approx 0.818,\ 0.152,\ 0.165.$$

2.1.2 Initial Exploration

The first step involved evaluating the function for different input parameter values to understand their influence on output behaviour. For the 1D case, the input x_2 was set to 0.5, which is not near any of the local minima, while x_1 was varied, see 2.1.

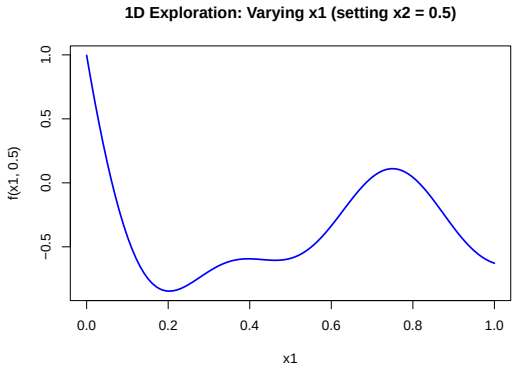


Figure 2.1: 1D Exploration of the Rescaled Branin-Hoo Function

At $x_1 \approx 0.7$, there is a peak, corresponding to the proximity to one of the minima at $x_1 \approx 0.698$, but since $x_2 = 0.5$ and not 0.818, it is not at the exact global minimum. This plot can also be used to check how well the emulator matches the actual function in 1D. The following exploration stage involved looking at the function's 2D contour plot:

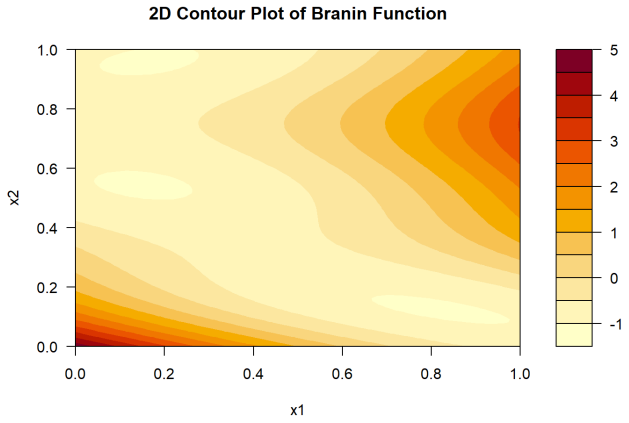


Figure 2.2: 2D Contour Plot

The curved contours in 2.2 show how both inputs interact, with all 3 of its global minima shown in the light yellow areas. This plot helps because it gives the "gold standard" of what Bayes Linear Emulation should be aiming for if the true function is available.

2.2 Bayes Linear Emulation

2.2.1 Theory

Before applying Bayes Linear Emulation (BLE), it is important to understand its theory. Given data containing simulator outputs D, from inputs $x_D = (x_1, x_2, \ldots, x_n)$, BLE predicts the output $f(x)$ at new points x_P by updating prior expectations and variances through a BL adjustment:

$$E_D[f(x)] = E[f(x)] + \text{Cov}[f(x), D]\text{Var}[D]^{-1}(D - E[D]), \text{ and}$$
$$\text{Var}_D[f(x)] = \text{Var}[f(x)] - \text{Cov}[f(x), D]\text{Var}[D]^{-1}\text{Cov}[D, f(x)],$$

where $E_D[f(x)]$ and $\text{Var}_D[f(x)]$ are the expectation and variance for $f(x)$, adjusted by data D and $\text{Cov}[f(x), D]$ is the covariance between new predictions and true data.[3] These performances are then evaluated through diagnostics:

$$S_D[f(x)] = \frac{E_D[f(x)] - f(x)}{\sqrt{\text{Var}_D[f(x)]}},$$

or, in other words, $S_D[f(x)]$ is the emulator's adjusted expectation subtracted from the true function value $f(x)$, normalised by the emulator's adjusted variance.

BLE comes with a few advantages:

- Computational Efficiency: They provide predictions of $f(x)$ in the form of the adjusted expectation and variance rapidly.[3]
- Flexibility: The method accommodates uncertainty, providing useful approximations even with limited simulator runs.[3]

Ultimately, BLE can be scaled to apply to large datasets where full Bayesian approaches may be less efficient. However, BLE has several limitations that can affect its accuracy and applicability:

- Covariance Dependence: Part of the accuracy of a BLE depends on the chosen covariance term. A poor choice results in unreliable predictions.
- Non-Gaussian Sensitivity: BLE relies on the Normal distributional assumptions of Gaussian Processes (GPs), which may not hold, hence making these estimates somewhat untrustworthy.[3]

2.2.2 Emulation Preparation

To prepare the function for Emulation, a few normalisation steps were carried out. These scale the input parameters to the $[0, 1]$ range, and these become the new emulator inputs. Here is how it is done throughout the code:

- The 2D emulator rescales inputs by dividing them by θ
- The rescaled Branin-Hoo function already assumes inputs x_1 and x_2 are in the $[0, 1]$ range due to its normalised form
- The design points generated using `randomLHS` are already normalised.

All of these adjustments ensured input data compatibility with emulator training.

2.3 1D Emulation

2.3 shows the resulting output of the 1D emulator. The emulator predictions (in blue) closely plot like the Branin-Hoo function. The narrow uncertainty bounds (in red) show the deviation between predicted and actual outputs. These minimal bounds and the emulator output smoothness indicate that the emulator captures the relationship between x_1 and $f(x_1, 0.5)$ mostly well. There are 2 noteworthy areas of uncertainty, though, which are the minima at $x_1 \approx 0.2$ and $x_1 \in [0.9, 1.0]$, as shown by their wider prediction intervals.

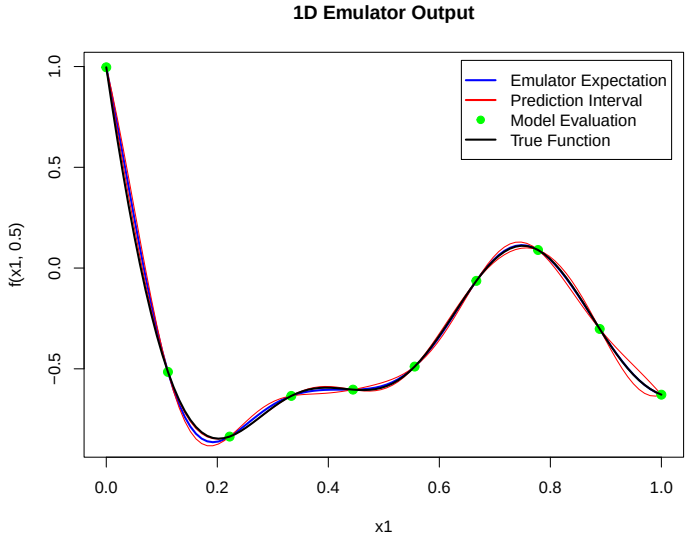


Figure 2.3: 1D Emulator Output showing predictions and uncertainty bounds.

2.4 2D Emulation

While 1D Emulation tests emulator performance, it does not consider a function's full complexity, which depends on the interaction between x_1 and x_2. To address this, 2D Emulation considers both parameters simultaneously, allowing the emulator to model their combined effects.

2.4.1 Latin Hypercube Design

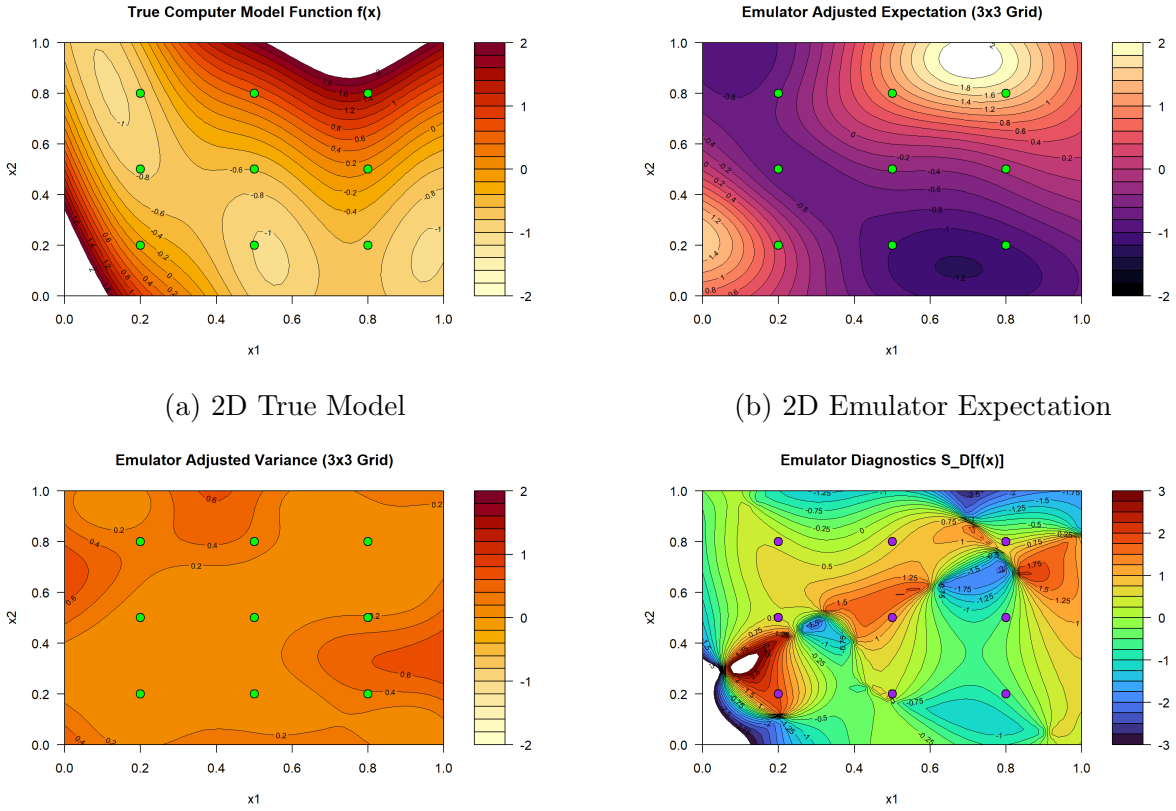
Before delving into 2D Emulation, understanding how it can be applied is essential. This report will apply it through Latin Hypercube Designs (LHDs), and here is how it works: For m input dimensions, a LHD for n runs, is formed by:

1. Dividing up the range of each input x_j into n equally sized intervals, and then
2. Placing the n runs so that each of the n intervals, for each of the inputs x_j, has exactly one run in it.[3]
3. This is done in turn for each of the m dimensions of the input space, i.e. repeat for x_j with $j = 1, \ldots, m$.[3]
4. Then either:
   - **Type 1:** add $1/(2n)$ to each $x_j^{(k)}$ entry to recentre intervals' middle points: $\{0, 1/n), [1/n, 2/n), \ldots, [(n-1)/n, 1)\}$, or
   - **Type 2:** add random i.i.d. uniform draws $u_{jk} \sim U[0, 1/n]$ to each entry $x_j^{(k)}$ to randomly sample each point from its chosen m dimensional "box".[3]

Type 2 is considered beneficial for several reasons; for example, the extra randomness avoids issues due to function periodicity.[3]

2.4.2 Extending to 2D

Here is the application of 2D Emulation to the Branin-Hoo function:



(a) 2D True Model

(b) 2D Emulator Expectation

(c) 2D Emulator Variance

(d) 2D Diagnostics

Figure 2.4: Comparison of 2D Emulator and True Model Outputs

These plots help to assess how well the emulator is performing. The contour lines, in general, represent regions of constant values, with tighter spacing reflecting more rapid value changes. Here are key insights from this set of plots:

- The true model, 2.4a, is used to refine BLE by focusing sampling near minima and using fewer points in flatter regions. It is used over 2.2 as the emulator only has limited training data, like the true model, avoiding unfair penalisation.
- 2.4b displays the 2D emulator's expectation. Compared to the true model, the grid points are largely accurate. However, there are some inaccuracies; for example, at $(0.5, 0.5)$, its predicted is 0.4 while its actual is 0.6.
- 2.4c shows the 2D emulator's variance. The grid points here mainly have low

variance, indicating estimate confidence. However, in the top left and bottom right regions, these variances are higher, so sampling should be focused here.

- The 2D emulator's diagnostics, see 2.4d, identify regions where additional training points may be necessary to refine the emulator's accuracy. The grid points largely display accurate emulator modelling with the only critique being at these points, the emulator is prone to slight overestimation.

Overall, these plots show that the emulator performs well but could benefit from additional training points and further tuning in high-gradient areas and this is where multi-wave history matching comes in.

2.5 Multi-Wave History Matching

2.5.1 History-Matching and Implausibility

History Matching is an iterative global parameter search method that exploits Emulation to efficiently explore $\mathcal{X}_0$ to find the set of acceptable inputs $\mathcal{X}$.[3] It compares model outputs with observed data to identify regions in the parameter space that are consistent with historical observations, hence refining model predictions and reducing uncertainty. History matching typically proceeds as follows: First, build an emulator of the model. Then, calculate implausibility, I, across the parameter space, where:

$$I^2(x) = \frac{(\mathrm{E}_D[f(x)] - z)^2}{\mathrm{Var}_D[f(x)] + \mathrm{Var}[\epsilon] + \mathrm{Var}[e]}.$$

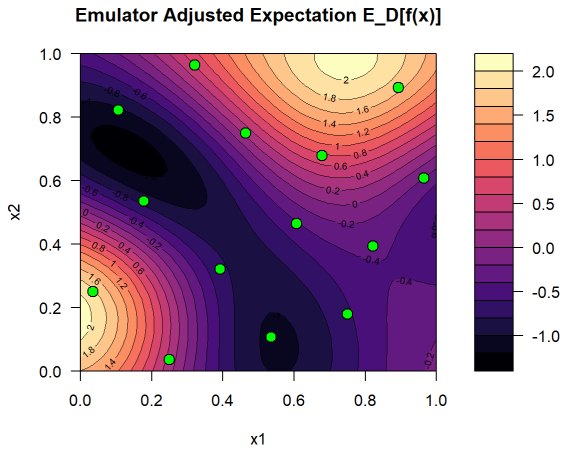
In this formula, z is the observed emulator value, $\mathrm{Var}[\epsilon]$ is the model discrepancy variance and $\mathrm{Var}[e]$ is the observation error variance.

Next, exclude regions where $I(\mathbf{x}) > I_{\text{threshold}}$ as they are set as unlikely to produce acceptable matches. This process is repeated iteratively, refining the parameter space in successive waves until only plausible input regions remain.[7]

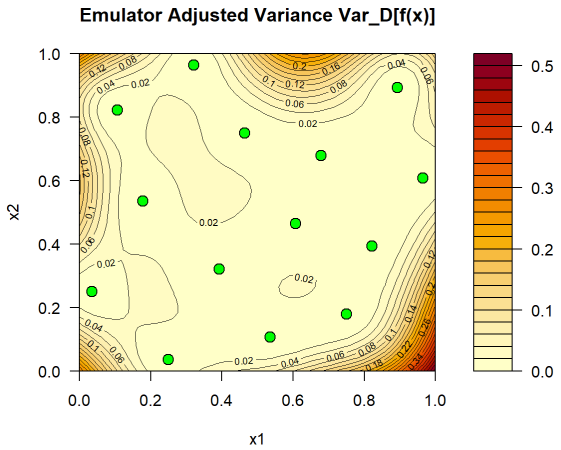
Implausibility is central to history matching. It is a measure that determines whether the input $\mathbf{x}$ is likely to result in an output that will match the observations.[7]

2.5.2 Multi-Wave History Matching and Implausibility Analysis

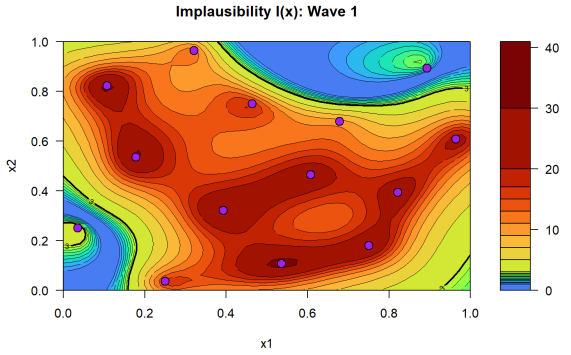
Instead of filtering the parameter space once, multi-wave history matching is performed iteratively across multiple waves. Each wave refines the emulator and updates the non-implausible parameter space before proceeding to the next iteration.[8] Early waves may have larger uncertainties, but later waves reduce these uncertainties by focusing emulator training on the non-implausible parameter space.[8] Here were the main results of applying Multi-Wave History Matching onto the rescaled Branin-Hoo Function:



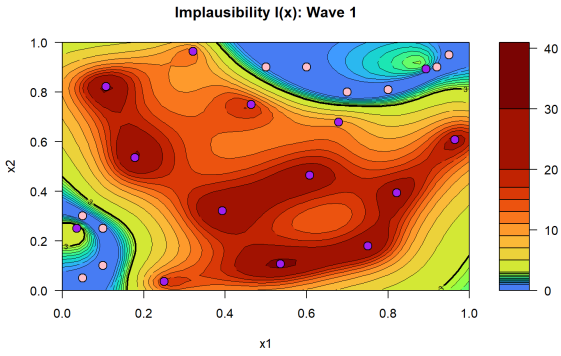
(a) Wave 1 Emulator Expectation



(b) Wave 1 Emulator Variance



(c) Wave 1 Implausibility



(d) Current Runs vs. Unevaluated Wave 2 Runs

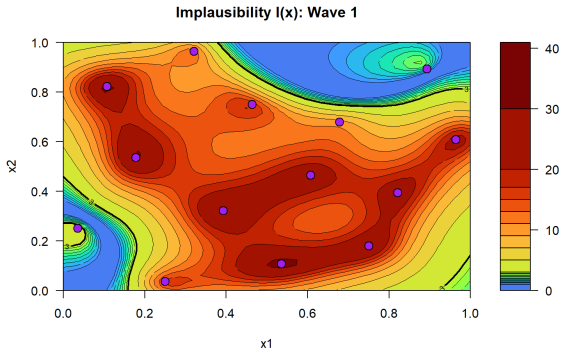
Figure 2.5: Wave 1 History-Matching

- 2.5a shows the emulator's predicted expectation. There is a minimum starting from $x_2 \in [0.8, 1.0]$ and $x_1 = 0$ and ending at $x_1 \in [0.4, 0.7]$ and $x_2 = 0$, indicating significant variation. Also, the Branin-Hoo function increases towards the top-right, suggesting a gradient.

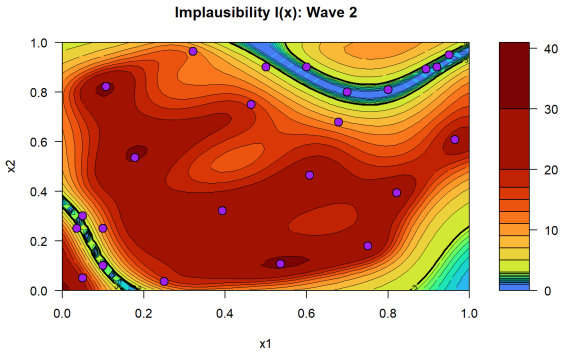
- 2.5b shows the emulator's uncertainty, representing confidence in predictions. Variance is low across most regions, indicating confidence in the emulator's predictions. Also, uncertainty is higher near the edges where no training points exist.

- 2.5c shows the emulator's implausibility, assessing emulator model fit. There are 2 clear areas of low implausibility, which the next wave will look at by taking design points in these regions.

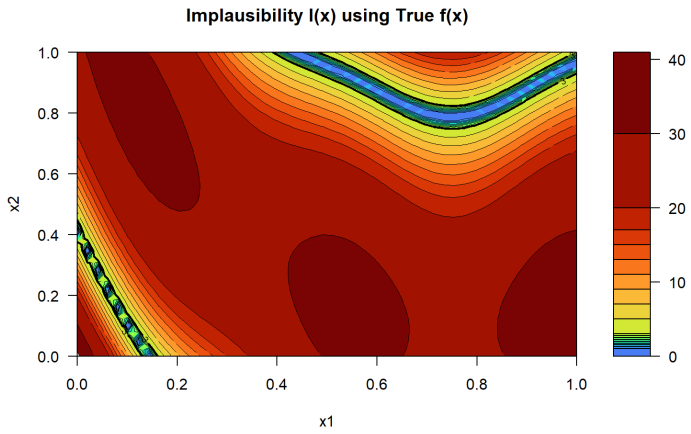
- 2.5d shows the locations of the first-wave runs (purple) and the 10 new second-wave candidate points (pink). The Wave 2 runs are mainly in low implausibility regions to help the emulator refine the model.



(a) Wave 1 Implausibility



(b) Wave 2 Implausibility



(c) Implausibility I(x) using True $f(x)$

Figure 2.6: Comparison of implausibility across Wave 1, Wave 2, and the true function.

- 2.6a is the implausibility after the initial grid points were applied. 2.6a is repeated for direct comparison with 2.6b, which is the implausibility after more grid points are added.
- The grid points have been added in key areas of high previous uncertainty along the upper-right and lower-left regions, matching where the true function had low implausibility values.
- These plots show how implausibility adjusts as the model explores the parameter space more and more. As shown by the blue region's precision in 2.6b.
- There are still some issues, however, as there is a blue region on the bottom right of 2.6b, and this should not be the case. However, this can be fixed with more grid points taken there.

Ultimately, taking more samples in low implausible regions has refined the emulator, leading to better alignment with the true function's implausibility, 2.6c.

Chapter 3 Conclusion & Further Research

3.1 Conclusion

The results presented in Figures 2.3 to 2.6 showcase the progression of Emulation from the simple 1D case to multi-wave history matching. The emulator's performance improves with each wave. The implausibility map highlights areas where emulator predictions may require improvements. By iteratively refining the parameter space, the emulator achieved tighter uncertainty bounds and improved observational data alignment on further waves.

By applying simple BLEs for single and multiple inputs, the project highlighted BLE's efficiency in approximating complex functions with limited computational resources. The emulator's performance was visualised through various plots, illustrating the expectation and variance of the emulated functions across the input space. LHD ensured adequate exploration of the parameter space, while the iterative history-matching process allowed for emulator refinement by excluding implausible regions. The results emphasise how Emulation can significantly improve computational models by providing insight into the variability and uncertainty of model outputs. Through multi-wave history matching, the emulator successfully adapted to new data points, improving its accuracy and narrowing down the plausible parameter space.

3.2 Further Research

Although this project covered BL Emulators in depth, there are several avenues for further research. While BLE is more computationally efficient, Gaussian Process Emulation (GPE) provides a full probabilistic specification across functions, meaning the posterior remains a full Gaussian Process.[3] Future research could involve applying GPE using packages such as `DiceKriging`. Comparing BLE and GPE provides insights into how the posterior distribution influences model predictions. Future work could also involve applying the developed emulator to real-world cases, such as in climate modelling.

Bibliography

1. Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.

2. S.E. Jackson and D.C. Woods. Bayes linear emulation of simulator networks. *arXiv preprint arXiv:1910.08003*, 2019.

3. Ian Vernon. *Uncertainty Quantification: Lecture Notes*. Durham University, 2024. Michaelmas Term.

4. S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets, 2013. `https://www.sfu.ca/~ssurjano/branin.html`.

5. M.D. McKay, R.J. Beckman, and W.J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

6. Andrew Iskauskas, Ian Vernon, Michael Goldstein, Danny Scarponi, Nicky McCreesh, Trevelyan J. McKinley, and Richard G. White. Emulation and history matching using the hmer package. *Journal of Statistical Software*, 109(10):1–48, 2024.

7. Ioannis Andrianakis, Ian R Vernon, Nicky McCreesh, TJ McKinley, Jeremy E Oakley, Richard G Bower, Michael Goldstein, R Nsubuga, and RG White. Bayesian history matching of complex infectious disease models using emulation: a tutorial and a case study on hiv in uganda. *PLOS Computational Biology*, 11(1):e1003968, 2015.

8. James M Salter and Daniel Williamson. A comparison of statistical emulation methodologies for multi-wave calibration of environmental models. *Environmetrics*, 27(8):507–523, 2016.