Universiteit Utrecht

MSc Mathematical Sciences

Master's Thesis

Emulation of Calibration Experiments in Agent-Based Models

Author:
Shaheen Ahmed-Chowdhury

Supervisors:
Dr. Martin Bootsma
Dr. Tristan van Leeuwen

October 6, 2021

Universiteit Utrecht

Abstract

As access to computational resources continues to grow, the scale of complex simulatory models only grows with it. Increased dimensionalities and runtimes for such models are a natural byproduct of this. This is an issue for tasks such as calibration and sensitivity analysis, as they often require a number of executions which grow with the runtime and input dimensionality of the underlying simulatory model.

In this thesis, we seek to remedy this with emulation methods, which seek to model the fitness surfaces of such experiments, and sequentially sample from the parameter space in a sample-efficient manner. We specifically examine a set of agent-based model (ABM) calibration experiments, both frequentist and Bayesian, and attempt to retrieve better fitness surface minima than schemes such as random search and MCMC, which are typically used. We emulate via Gaussian processes and Bayesian optimisation, a pair of methods commonly used for hyperparameter optimisation in expensive machine learning models.

We conduct such emulation experiments on a set of toy ABMs, of increasing dimensionality, ending with the baseline version of a leading, large-scale macroeconomic ABM. We show that such emulation schemes can typically locate better minima than the aforementioned methods, and thus often retrieve more suitable parameters for our ABMs, with less executions. This is crucial when the underlying model is expensive.

We end by introducing, and reporting the performance of, a set of state-of-the-art high-dimensional Bayesian optimisation methods, which attempt to remedy documented issues with Gaussian processes and Bayesian optimisation in regimes of high (> 20) input dimensionality.

Contents

Chapter 1

Introduction

What do economics, biology, sociology, physics, chemistry, ecology, urban planning, artificial intelligence and epidemiology all have in common? Well, amidst the myriad of sub-problems they all encapsulate, is the need to understand, model and predict the behaviour of complex, non-linear and multi-agent real-world systems. Such domains deal with analytically intractable systems far more often than mathematics, and arguably have more to gain from ambitious, large-scale simulatory exercises of the phenomena they encounter.

In this thesis, we shall be working with one such type of simulatory experiment, namely agent-based models (ABMs). ABMs are typically used in problems consisting of a large number of individual agents, entities or actors. These may be nations, banks, companies, vehicles, robots, consumers, animals, or even bacteria. However, most typically, the agents will require some form of decision-making process, in response to their environment and other agents. Such processes can often introduce path-dependence and stochasticity into the simulated systems, and expert domain knowledge is typically required to model such processes accurately.

When ABM practitioners get more ambitious in the size of the systems they'd like to model, and the scope of the phenomena they'd like to capture, two aspects of the ABM typically increase, their runtime and their input dimensionality. This makes calibration of the ABM more difficult. Finding a set of parameters which minimises some user-defined fitness function (essentially the task of calibration) becomes more arduous, as the volume of the parameter space increases exponentially with the number of parameters. In addition, an expensive ABM means that some time limit, or execution budget, will most likely exist for the ABM practitioner. In this thesis, we will look to tackle this problem of calibrating an expensive and high-dimensional ABM.

We will do this by first emulating the n-dimensional fitness surfaces within such calibration experiments, where n represents the number of parameters of the ABM. Emulation involves creating a surrogate model for the fitness surface, and interpolating between executed points in the parameter space. We will use a Gaussian process (GP) to do this. GPs allow us, via a Bayesian formulation, to first define a set of priors in function space, and then analytically obtain a posterior, post the observation of executed points. We then build upon this analytically tractable surrogate model, by combining it with a sequential sampling scheme. This will allow us to more efficiently gain information from the parameter space than say, random search or MCMC for instance.

The combination of these two ideas, a surrogate model and a sequential sampling scheme, create what is known in the machine learning literature as a Bayesian optimisation (BO) loop. Such schemes are commonplace in problems such as neural architecture search or hyperparameter optimisation, where models with long runtimes possess a large ($<$ 1000) input dimensionality, and a suitable parameter configuration must be identified.

Treating the ABM calibration problem with GPs and BO is not new, but applications are commonly restricted to base implementations. The ability of GPs to embed prior information about the fitness landscape is not typically utilised, sequential sampling schemes are often deemed costly and many authors are not aware of the rich array of applicable tools within the BO literature.

In this thesis, we shall discuss this gap between the implementation of GPs and BO in the ABM literature, and state-of-the-art BO methods. We will also discuss some pitfalls of such BO methods, namely high input dimensionalities (greater than 15-20) and large numbers of executed sample points, and also how to tackle these issues. We will provide numerical experiments incrementally, and compare the performance of our selected BO algorithms with those in the ABM literature.

Chapter 2 will define the notation used throughout this thesis, for our ABMs and the calibration experiments we implement. We provide details of all ABMs implemented, including their analytical forms (which can be skipped upon a first reading), parameter sets, dimensionalities and bounds (the last three of which the reader should pay more attention to). We also detail our frequentist and Bayesian calibration experiments in Chapter 2, providing algorithmic pseudocode for each. We end the chapter by illustrating said calibration experiments upon our toy models, and highlighting how they begin to struggle in higher input dimensions [1].

Chapter 3 will discuss emulation, GPs and BO, and also define our notation in these areas. We begin by building up an intuition of GPs, and how they can help us emulate our calibration experiments, before extending them with a sequential sampling scheme, and thus forming a BO loop. We then illustrate our emulated calibration experiments, and the consistent improvement in minima location over random search.

Chapter 4 discusses, in fair detail, many previous applications of emulation to similar ABM experiments, extending more generally to many other complex simulatory models. We discuss the frequency of the emulation methodology we use, and alternative methods that are often used. We find a handful of promising avenues for future exploration.

Chapter 5 extends upon this, by discussing our selection for many elements of our GP and BO loop, alongside multiple (quite promising) extensions to deal with features such as heteroscedasticity, more sample points and multiple model outputs. We also discuss one such area, high-dimensional BO, and outline a selected method which we implement in Chapter 6.

In this penultimate chapter, we emulate a large-scale macroeconomic model, discussed in Chapter 2, and examine the performance of random search, BO and our high-dimensional BO method upon our frequentist and Bayesian calibration experiments. Chapter 7 concludes.

[1] All code used in this thesis will be available at `https://github.com/shaheenahmedc/Emulation_ABMs`.

Chapter 2

Agent-Based Models

ABMs seek to simulate the behaviours and interactions of many interacting agents, through the inclusion of a set of (usually empirically grounded) behavioural rules. Such rules seek to create micro-interactions, which, at the macroscopic level, produce emergent behaviours in the system being modelled. A further quantitative definition is provided later (in Section 2.1), but some further key features, as detailed by Macal in [1], can serve to illuminate the ethos behind ABMs. These four features are:

- individualism - agents are represented individually, and thus require a 'state' to be stored;
- interactivity - agents can interact with other agents and their environment;
- autonomy - agents can independently interact without the need for centralised, distributed control;
- and adaptivity - autonomous, interacting agents are able to change their behaviours, as a result of some implementation of 'memory' within the state of the agent.

Such features are at the heart of agent-based modelling methodologies, which should become more evident when we outline the ABMs used in this thesis. For our purposes however, it suffices to treat all ABMs as black-box functions, with some input dimensionality and runtime.

Given that we in no way differentiate between ABMs and other simulatory models (apart from in the calibration literature we replicate), a further qualitative detailing of ABMs is not of utmost importance. Rather, we opt to provide a mathematical definition of such ABMs/simulators more generally in the next section. For the interested reader, we can recommending consulting ABMs such as von Neumann's cellular automaton [2], Conway's Game of Life [3], Schelling's segregation model [4], and Axtell's Sugarscape model [5], if seminal examples are sought.

Such ABMs have only increased in complexity in recent decades, as the field has matured, and computational resources have increased. Moore predicted the recently observed exponential growth in computational power back in 1965, in his now famous law [6], data for which is visualised in Figure 1. With such an explosion in resources (amongst much more) has come the ability to create extraordinary simulatory exercises, which at their grandest, seek to mimic the behaviour of entire epidemics [7], economies [8] and even galaxies [9]. However, with great power, of course comes great responsibility, and tasks such as calibration and sensitivity analysis, which the responsible practitioner would dedicate time to, quickly become infeasible with such ambitious simulations, and

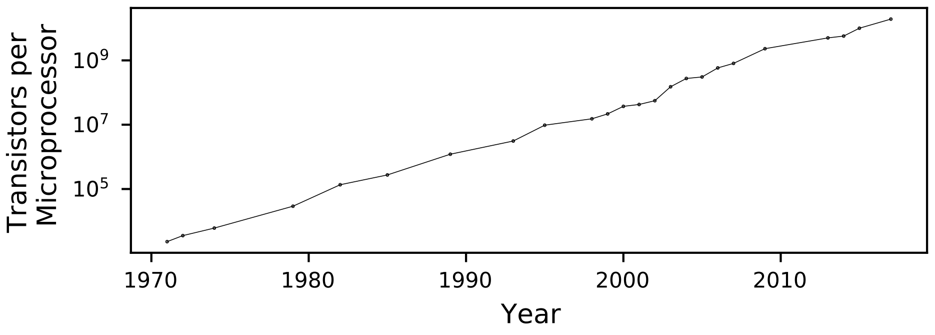

Figure 1: The maximum number of transistors fitted onto a microprocessor over time. The exponential trend clearly visualises Moore's Law. Data sourced from [10].

so sample-efficient schemes which can aid these tasks are of a necessity. Let's begin our path to such schemes, by first defining our notation, alongside the ABMs and calibration experiments implemented in this thesis.

2.1 ABM Notation

Let us begin by defining the notation we will be using. Lower-case roman and greek letters $(x, x_t, \theta, \theta_i)$ shall denote scalar quantities. Such letters, with arrows above them $(\vec{x}, \vec{\theta})$ shall denote vectors. Capital greek letters (Θ, Γ) shall denote matrices generally, and such notation shall be maintained if any dimension of a previously defined matrix becomes one (a set of parameter vectors becoming a set of one-dimensional parameter vectors for instance). Capital roman letters (A, T) are reserved for unique purposes. Some letters, such as X or T, are capital letters in the greek and roman alphabets, and whether these are matrices or unique specifications shall be provided to the reader at each occurence. The notation $\mathbb{E}_n[\cdot]$ denotes taking a sample average of the quantity $[\cdot]$, which itself shall depend on some variable n.

We shall now define ABMs for all subsequent discussion. Let

$$\vec{x} = A\big(\vec{\theta}, \varepsilon\big) \tag{1}$$

denote an ABM, where $\vec{\theta} = \{\theta_1, ..., \theta_n\}, \vec{\theta} \in \mathbb{R}^n$ denotes an input parameter vector of length n, and ε denotes a seed for the random number generator (RNG) used in the model[2]. The symbol A denotes our ABM (some computational simulator), while $\vec{x} = \{x_1, ..., x_T\}, \vec{x} \in \mathbb{R}^T$ denotes a user-selected output from the ABM, which consists of a univariate time-series of length T. In this thesis, we shall keep T constant over the course of each experiment, and we seek to take sufficient draws of the seed ε, such that its effect on $\vec{x}$ is negated. We discuss this more below. Note also, that here we work in the discretised time setting, as opposed to continuous-time environments that stochastic differential equations use for instance.

2.2 Stochasticity

If an ABM $A(\cdot)$ has a stochastic noise term, then that term will require an RNG, to generate random numbers at each timestep of the model. Some authors choose to denote

[2]More often than not, ABMs have stochastic noise components, and these require RNGs to generate such noise. We discuss this more in Section 2.2

this with a vector of random numbers being generated prior to the model, and that vector then being input to the model itself. Thus, Equation 1 would appear as $\vec{x} = A(\vec{\theta}, \vec{\varepsilon})$, with $\vec{\varepsilon}$ denoting such a vector of random numbers. However, for our purposes, the length of such a vector changes with each model, depending on how many stochastic noise terms there are in the ABM $A(\cdot)$. Also, in our implementations, we always set a temporary seed prior to running any ABM, the reasons for which shall become clear later. With these two factors in mind, we shall always input some random number seed ε to any ABM, which avoids having to define the length of the random number vector $\vec{\varepsilon}$ prior to every ABM execution. The ABM shall generate as many random numbers as it requires, starting from the random state defined by the random seed ε.

As shall become evident, when we hold the parameter vector $\vec{\theta}$ constant, and input different values of the seed ε in succession, we're able to control for the effects of the random seed by taking the output in expectation. Thus, when we subsequently change the parameter vector $\vec{\theta}$, any change observed in the sample average output can be attributed solely to such changes in $\vec{\theta}$, and not to any seed ε.

2.3 Toy ABMs

We've selected a set of ABMs which increase in complexity, both in structure and dynamics. We initially use them to display ABM calibration experiments, and then apply our BO schemes to the same experiments. This is done to improve the location of minima, and to better handle execution and/or time budgets. We draw heavily from the toy ABM selection of Platt [11]. Within his paper, Platt seeks to compare different ABM calibration methods, and uses a set of toy models similar in the range of input dimensionality and the range of complexity. Please note, the first-time reader may skip over many of the minute model details in this section. It is only important to gain an appreciation for the dimensionality, dynamics and typical parameters used in such models. We present them in full for completeness.

2.3.1 The AR(1) Model

The first model we use is the econometric AR(1) model. While this model wouldn't typically be classed as an ABM, it does help us illustrate calibration and emulation in a very simple system. It is defined as

$$x_{t+1} = \alpha x_t + \mathcal{N}(0,1)_{\varepsilon,t+1}, \tag{2}$$

where $\mathcal{N}(0,1)_{\varepsilon,t+1}$ denotes a normal random variable, with mean 0 and standard deviation 1. Its subscript denotes the timestep at which we observe such a variable, as well as the random number seed ε which is input to the model before execution. This initialises the sequence of random numbers which are then used by the model[3]. We also initialise the model at $x_0 = 0$.

In Equation 2, $\alpha \in [0,1]$ denotes a smoothing parameter (as it is referred to in the econometric literature), and is the only entry in this model's one-dimensional parameter vector $\vec{\theta}$, as denoted in Equation 1. Note also, for this thesis, that the resultant time-series $\vec{x} = \{x_1, ..., x_T\}$ from iterating Equation 2 is precisely the quantity $\vec{x}$ in Equation 1. Equation 2 entirely defines the data-generating process of the computational simulator A in Equation 1 (albeit a very simple one).

This model produces limited dynamics (as shown Figure 2-a), but possesses an input of dimensionality one, a negligible runtime, and a univariate time-series output $\vec{x}$, making it useful for the aforementioned illustrative purposes.

[3]Note that subsequent models may have multiple such noise elements. Each noise element increments the local random number state of the model, initialised by the random number seed ε.

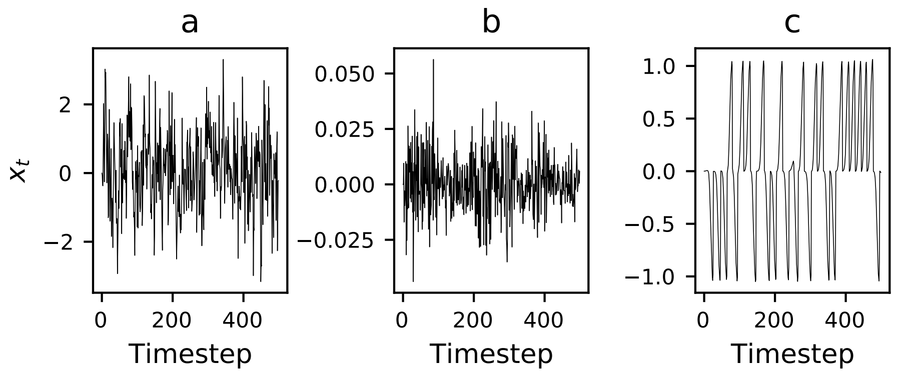

Figure 2: The AR(1), Franke-Westerhoff and Brock-Hommes models are simulated for 500 timesteps each, in subplots a, b and c respectively. The AR(1) model's α parameter was set to 0.5, while the Franke-Westerhoff and Brock-Hommes models' parameters were set to their *DCA-HPM* and BH_2 parameter sets, detailed in Tables 1 and 2 respectively. Note the different dynamics between the three models' outputs (which are more difficult to visualise for the first-differenced time-series' that we use in this thesis).

2.3.2 The Franke-Westerhoff Model

The second model we will be using is the nine-dimensional Franke-Westerhoff model (specifically the *DCA-HPM* variant), which is defined by Franke in [12], and summarised by Platt in [13], in a set of coupled equations, namely:

$$p_t = p_{t-1} + \mu\big(n^f_{t-1}d^f_{t-1} + n^c_{t-1}d^c_{t-1}\big), \tag{3}$$

$$d^f_t = \phi(p^* - p_t) + \mathcal{N}(0, \sigma_f^2)_{\varepsilon,t}, \tag{4}$$

$$d^c_t = \chi(p_t - p_{t-1}) + \mathcal{N}(0, \sigma_c^2)_{\varepsilon,t}, \tag{5}$$

$$n^f_t = \frac{1}{1 + exp(-\beta a_{t-1})}, \tag{6}$$

$$n^c_t = 1 - n^f_t, \tag{7}$$

$$a_t = \alpha_n(n^f_t - n^c_t) + \alpha_0 + \alpha_p(p_t - p^*)^2, \tag{8}$$

where $\mathcal{N}(0, \sigma_f^2)_{\varepsilon,t}$ and $\mathcal{N}(0, \sigma_c^2)_{\varepsilon,t}$ denote two normal random variables, with standard deviation σ_f and σ_c respectively (which themselves make up two of the Franke-Westerhoff model's nine input parameters). Again, we initialise the random state for the model with the random seed ε, and the subscript of the two random variables denotes this. Both random variables increment the same random state, initialised by ε, at each timestep of the model. The variable p_t (Equation 3) represents the log asset price at time t, p^* (Equation 4) is the log of the fundamental value, n^f_t and n^c_t (6 and 7) are market fractions of fundamentalists and chartists (agents believing in reversion to a fundamental value and agents following recent trends) respectively at time t, and d^f_t and d^c_t (4 and 5) are the corresponding average demands. The symbol a_t (8) represents the relative attractiveness of fundamentalist and chartism, and its calculation contains three more model parameters, weighting three behavioural heuristics: herding; predisposition; misalignment. The definitions of these can be found in Franke's original paper [12]. The parameter μ (3) weights the effect of trader strategy distributions on the previous price, ϕ and χ (4 and

	μ	β	ϕ	χ	α_n	α_0	α_p	σ_f	σ_c
DCA-HPM	0.01	1.0	0.12	1.50	1.79	-0.327	18.43	0.758	2.087
Bounds	[0,1]*	[0,1]*	[0,1]*	[0,10]*	[0,2]	[-1,1]	[0,20]	[0,1]*	[0,5]

Table 1: Parameter values for the *DCA-HPM* variant of the Franke-Westerhoff model. Bounds are also presented, with asterisks denoting assumed bounds. Those without an asterisk form a free parameter set in Platt's work [13], thus have well-defined bounds provided. Those with asterisks are held constant by Platt, and no such bounds are provided.

5) tune the impact of recent price information on each strategy's demand, while β (Equation 6) is a switching parameter, which tunes the intensity of agents switching between strategies. We present an example of the Franke-Westerhoff model's dynamics in Figure 2-b, along with initial values and boundaries of the model's nine-dimensional parameter vector $\vec{\theta}$, in Table 1.

2.3.3 The Brock-Hommes Model

The third model we shall be using is similar in subject area to the Franke-Westerhoff model, but different in dimensionality and structure. Known as the Brock-Hommes model, it possesses an 11-dimensional input space, and seeks to embed empirical heuristics of financial traders into a system of coupled equations. Many re-formulations of the Brock-Hommes model exist, so here we use the same version as Platt [11]. This allows us to validate our calibration experiments, by replicating a leading paper in the ABM calibration literature.

The model is defined by

$$x_{t+1} = \frac{1}{1+r}\left[\sum_{h=1}^{H} n_{h,t+1}(g_h x_t + b_h) + \mathcal{N}(0,\sigma^2)_{\varepsilon,t+1}\right], \tag{9}$$

$$n_{h,t+1} = \frac{exp(\beta U_{h,t})}{\sum_{h=1}^{H} exp(\beta U_{h,t})}, \tag{10}$$

$$U_{h,t} = (x_t - Rx_{t-1})(g_h x_{t-2} + b_h - Rx_{t-1}), \tag{11}$$

where $\mathcal{N}(0,\sigma^2)_{\varepsilon,t+1}$ is again, a normal random variable with mean 0 and standard deviation σ^2, the random number state for which is initialised by the random number seed ε.

We also set $H = 4$ (as is done by Platt in [11]), which represents the number of different trader strategies, each varying in fitness as the model evolves in time. The quantity $U_{h,t}$ (Equation 11) represents the fitness of any particular strategy $h \in 1, ..., H$. Note the two parameters in Equation 11, g_h and b_h. These are the trend-following and bias parameters respectively, for the strategy h. Thus, eight of the 11 parameters consist of these parameters (four apiece). In Equation 10, $n_{h,t}$ denotes the proportion of traders adopting a strategy h, based on the fitness $U_{h,t}$, while x_t denotes the realised asset price at time t, minus a fundamental price p^*. The parameter β is analogous to the Franke-Westerhoff model's use, namely as a switching intensity parameter. We refer the interested reader to the original article by Brock and Hommes [14], should they wish to see the derivation of this model. We have two more parameters left to define, namely the market interest rate r, and the standard deviation of the noise in Equation 9, σ. The variable $R = 1 + r$.

Once again, this model is relatively inexpensive to run, but now, far more complex dynamics can be produced, and a higher dimensional parameter space is available. Two

	g_1	g_2	g_3	g_4	b_1	b_2	b_3	b_4	r	β	σ
BH_1	0	0.6	0.7	1.01	0	0.2	-0.2	0	0.01	10	0.04
BH_2	0	0.9	0.9	1.01	0	0.2	-0.2	0	0.01	120	0.04
Bounds	[0,1]*	[0,1]	[0,1]	[0,2]*	[0,1]*	[0,1]	[-1,1]	[0,1]*	[0,1]*	[0,150]	[0,1]*

Table 2: Two sets of parameter values for the Brock-Hommes model, as used by Brock and Hommes in [14], and later by Platt in [11]. BH_1 represents a more realistic choice intensity, while BH_2 represents a more extreme regime, as can be seen by the increased values of g_2, g_3 and β. Bounds are also provided for each parameter. Asterisks denote assumed bounds. Non-asterisked bounds were provided by Platt in [11], as they were selected as free parameters, varied in these ranges.

common sets of parameter values are provided in Table 2, each of which define our 11-dimensional parameter vector $\vec{\theta}$ for the Brock-Hommes model. An example of the output this model can produce is presented in Figure 2-c. In using such a model, we attempt to mimic the dynamics of larger-scale ABMs.

2.3.4 The Keynes Meets Schumpeter Model

The fourth model we shall be using is a much larger macroeconomic ABM. Known as the Keynes Meets Schumpeter (KS) model, it seeks to combine Schumpetarian theories of innovation with Keynesian theories of demand-driven dynamics, and is defined by Dosi, Fagiolo and Roventini in [8]. Due to the complex nature of the model, it will only be briefly outlined in this thesis. We are mainly concerned with the model's dimensionality, and gaining brief insights into the parameters of the model, as well as an overview of how it works. We will be working with the 'benchmark' version of the model, as detailed by Dosi et al. in [8]. The parameters of this model are outlined in Table 3. This version consists of firms, consumers/workers and a government, with later iterations including banks (to study the role of credit) [15].

Firms belong to one of two industries, a capital-goods sector and a consumption-goods sector. Capital-goods firms perform R&D, and produce heterogeneous machines, with labour from consumers/workers. Consumption-goods firms buy machinery from capital-goods firms, and produce a single product for consumers/workers. A government simply collects taxes on profits and pays unemployment benefits. This outline is provided by Dosi et al. in [15], as well as the following timeline of microeconomic actions. At each timestep t:

1. Capital-goods firms perform R&D. They then advertise their machines to consumption-goods firms.
2. Consumption-goods firms choose how much to produce and invest. If the latter is positive, they choose suppliers and send orders.
3. All firms hire workers, according to their production plans, and begin production.
4. The consumption-goods market opens. The market shares of firms evolve, according to competition, via replicator dynamics.
5. Entry and exit of firms takes place. Any firms with (near) zero market share and/or negative net assets are removed, and replaced by new firms.
6. Machines ordered at the beginning of the period are delivered, and are added to a firm's assets at $t + 1$.

Description	Code	Base Value	Bounds
Tax rate	tr	0.1	[0, 0.3]
Multiple of net assets or sales for credit limit	$Lambda$	2.0	[0.01, 4.0]
Interest rate	r	0.01	[0.005, 0.05]
Average initial net worth in capital-good sector	$NW10$	1000	[200, 5000]
Lower/upper support for capital-good entrant net worth share	$Phi3, Phi4$	0.1, 0.9	[0, 0.5], [0.5, 0.1]
Beta distribution parameters for innovation drawing	$alpha1, beta1$	3.0, 3.0	[1.0, 5.0], [1.0, 5.0]
Beta distribution parameters for imitation drawing	$alpha2, beta2$	2.0, 4.0	[1.0, 5.0], [1.0, 5.0]
New customer share for firm in capital-good sector	$gamma$	0.5	[0.2, 0.8]
Mark-up of firms in the capital-good sector	$mu1$	0.08	[0.01, 0.2]
Share of revenue applied in R&D by firm in the capital-good sector	nu	0.04	[0.01, 0.2]
Lower/upper support for new machine productivity change	$x1inf, x1sup$	$-0.15, 0.15$	[-0.3, -0.1], [0.1, 0.30]
Upper share limit for productivity improvement of capital-good entrant	$x5$	0.3	[0.0, 1.0]
Share of R&D expenses in innovation	xi	0.5	[0.2, 0.8]
Expected elasticity of R&D expense in innovation/imitation success	$zeta1, zeta2$	0.3, 0.3	[0.1, 0.6], [0.1, 0.6]
Average initial net worth in consumption-good sector	$NW20$	1000	[200, 5000]
Lower/upper support for consumption-good entrant net worth share	$Phi1, Phi2$	0.1, 0.9	[0, 0.5], [0.5, 1.0]
Replicator dynamics selectivity coefficient	chi	1.0	[0.2, 5.0]
Minimum market share to stay in consumption-good sector	$f2min$	10^{-5}	[10^{-6}, 0.001]
Share of inventories on planned output	$iota$	0.1	[0, 0.3]
Machine output in consumption-good units per period	$m2$	40	[10, 100]
Initial mark-up of firms in the consumption-good sector	$mu20$	0.3	[0.1, 0.5]
Planned utilization of machinery	u	0.75	[0.5, 1.0]
Sensitivity of mark-up adjustment	$upsilon$	0.04	[0.01, 0.1]
Unemployment benefit as a share of average wage	phi	0.4	[0.0, 1.0]
Elasticity of wages to aggregate productivity growth	$psi2$	1.0	[0.95, 1.05]

Table 3: Parameter values for the 'benchmark' setup of the KS Model of Dosi et al. [8]. Note the use of codes for each parameter, rather than algebraic letters. This is done to mimic the notation in the software which implements the model, available via [16]. Not all parameters are provided in the literature, and so we opt to follow the notation of the software, to aid future work. The base values and bounds were gathered from a benchmark configuration file, and a sensitivity analysis file, which confirmed acceptable bounds for all parameters. This is discussed further in Chapter 6. After removing all discrete parameters, we are left with a 33-dimensional parameter space, three times that of the Brock-Hommes model.

At the end of each timestep, macroeconomic variables (such as GDP, investment and unemployment) are calculated for the system. In this thesis, we will only use the first-differenced form of one of these time-series $\vec{x}$, for our calibration and emulation experiments[4]. But we can see now, how such a macroeconomic time-series may be produced from a complex underlying ABM.

2.4 Calibration

ABMs seek to replicate some real-world data-generating process at varying levels of detail. As a result, ABMs will often contain parameters which have real-world analogues. ABM outputs will often have real-world time-series analogues also, as is described in the KS model above. Such mirroring of the real-world allows for the use of copious amounts of empirical data, with the goal of tuning our 'digital twin' of the real world to empirical observations. But how we do measure how well our ABM replicates the real world? Mustn't this precede parameter tuning? In other words, do we not need to validate that our model works as desired, before calibrating it according to some desired fitness function?

Within the ABM community, approaches to tackle this problem, commonly labelled as the 'validation' problem, have been somewhat segmented by field, from Ecology's pattern-oriented modelling [17] to Economics' matching of stylised facts. To get an indication of

[4]The first differences of the model's GDP output time-series, to be exact.

Code	Stylized Fact
SF1	Endogenous self-sustained growth with persistent fluctuations
SF2	Fat-tailed GDP growth-rate distribution
SF3	Recession duration exponentially distributed
SF4	Relative volatility of GDP, consumption and investment
SF5	Cross-correlations of macro variables
SF6	Pro-cyclical aggregate R&D investment
SF7	Cross-correlations of credit-related variables
SF8	Cross-correlation between firm debt and loan losses
SF9	Banking crises duration is right skewed
SF10	Fiscal costs of banking crises to GDP distribution is fat-tailed
SF11	Firm (log) size distribution is right-skewed
SF12	Fat-tailed firm growth-rate distribution
SF13	Productivity heterogeneity across firms
SF14	Persistent productivity differential across firms
SF15	Lumpy investment rates at firm-level
SF16	Firm bankruptcies are counter-cyclical
SF17	Firm bad-debt distribution fits a power-law

Table 4: A table of stylised facts, which the original KS model is able to replicate, as detailed in [15] (which also provides references to empirical studies for each stylized fact).

the detail of such stylised facts, we present those replicated by the KS model in Table 4. Empirical studies provide evidence for each stylised fact.

The fulfilment of such empirical criteria is typically measured by comparing moments of empirical and simulated data, and minimising a moments-based distance measure. We use similar methods below, but rather to tune the parameters of our model to pseudo-empirical data (as discussed in Section 2.5.1). Thus, while it would be a simple extension to attempt to validate ABMs with our emulation approach, we don't do so in this thesis, and leave it for a topic of further research. Thus, we refer to the task of parameter tuning, in an attempt to minimise some user-defined fitness function, as calibration going forward, and follow the terminology of Platt in [11].

2.5 Frequentist Calibration

In his 2020 work, Platt discusses two dominant strands to the the ABM calibration literature; frequentist and Bayesian [11]. We discuss the former here.

We begin by defining a few quantities we'll require. The vector $\vec{x}_{emp} = \{x_1, ..., x_{T_{emp}}\}$, $\vec{x}_{emp} \in \mathbb{R}^{T_{emp}}$ denotes an empirical dataset of length T_{emp}, an analogous version of which, $\vec{x}$, can be generated from our ABM via Equation 1. The matrix

$$\Theta = \{\vec{\theta_1}, ..., \vec{\theta_m}\} \in \mathbb{R}^{n \times m} \tag{12}$$

denotes a set of parameter vectors, generated by some sampling function

$$H_{os}(\vec{\theta}_{lower}, \vec{\theta}_{upper}) = \Theta, \tag{13}$$

where $\vec{\theta}_{lower}, \vec{\theta}_{upper} \in \mathbb{R}^n$ represent vectors of lower and upper bounds for each parameter. The symbol H_{os} denotes a one-shot sampling scheme, which produces m sample points in the n-dimensional parameter space, forming the $(n \times m)$ matrix Θ. Typical choices of H_{os} include random search, an n-dimensional grid search, Latin Hypercube [18] or nearly-orthogonal Latin Hypercube (NOLH) schemes [19]. Such one-shot schemes typically cast an initial set of points into the parameter space, with the goal of maximising (to varying degrees) the Euclidean distances between the points. The sampling scheme is then said to exhibit good space-filling properties [5].

Now, let us define the distance function y via

$$y = f(\vec{x}, \vec{x}_{emp}), \tag{14}$$

where $y \in \mathbb{R}$ represents a scalar measure of the distance between the two input time-series, as defined by f. We can equivalently write this is as

$$y = f(A(\vec{\theta}, \varepsilon), A(\vec{\theta}_{emp}, \varepsilon)), \tag{15}$$

via Equation 1, to see the explicit dependence of y on the parameter vector $\vec{\theta}$, and the random number seed ε. Note here, that we only use the parameter vector $\vec{\theta}$ (alongside its empirical analogue), and not the matrix Θ (which is used later). A further discussion of $\vec{\theta}_{emp}$ is coming very shortly.

Our goal in frequentist calibration is to minimize this distance function y, which can be formalised as finding the minimal value

$$y^* = \underset{\vec{\theta} \in \Theta}{\operatorname{argmin}} \underset{\varepsilon}{\mathbb{E}}[f(\vec{x}, \vec{x}_{emp})], \tag{16}$$

or otherwise, equivalently stated,

$$y^* = \underset{\vec{\theta} \in \Theta}{\operatorname{argmin}} \underset{\varepsilon}{\mathbb{E}}[f(A(\vec{\theta}, \varepsilon), A(\vec{\theta}_{emp}, \varepsilon))], \tag{17}$$

where $\underset{\varepsilon}{\mathbb{E}}[f(A(\vec{\theta}, \varepsilon), A(\vec{\theta}_{emp}, \varepsilon))]$ denotes the value of $f(A(\vec{\theta}, \varepsilon), A(\vec{\theta}_{emp}, \varepsilon))$ in expectation. We approximate this expectation by taking a sample-average over different values of ε. We then seek to minimise this quantity, by varying the parameter vector $\vec{\theta}$, but also by holding $\vec{\theta}_{emp}$ constant.

We also define the vector $\vec{y} = \{y_1, ..., y_m\}, \vec{y} \in \mathbb{R}^m$, which will store the mean value of the distance function for each parameter vector $\vec{\theta} \in \Theta$, where of course $\Theta \in \mathbb{R}^{n \times m}$. This is expressed algorithmically soon, in Section 2.5.2.

There is one last thing we must discuss now, and that is how to interpret $\vec{\theta}_{emp}$. What does it mean to have a parameter vector for empirically observed data? In order to discuss this, we now introduce the distinction between internal and external calibration.

2.5.1 Internal vs External Calibration

In this thesis, we define external calibration as tuning the parameter vector of our model $\vec{\theta}$, using empirical data $\vec{x}_{emp}$. Within internal calibration however, we select some 'ground-truth' parameter vector $\vec{\theta}_g \in \mathbb{R}^n$, and generate an output time-series $\vec{x}_g \in \mathbb{R}^{T_{emp}}$. We then withhold the 'ground-truth' parameter vector $\vec{\theta}_g$ from the calibration procedure, and attempt to tune $\vec{\theta}$ using our distance function $f(\vec{x},\ \vec{x}_g)$, with $\vec{x}_g$ acting as a proxy for actual empirical data $\vec{x}_{emp}$. Thus, we attempt to recover $\vec{\theta}_g$.

One may ask, why would we want to do this? Well, this methodology was developed recently by Platt in 2020 [11], to quantitatively assess the ability of different ABM calibration schemes (frequentist and Bayesian) to recover some 'ground-truth' parameter set $\vec{\theta}_g$. When we attempt to estimate the theorised real-world parameters $\vec{\theta}_{emp}$, underpinning some unknown data-generating process that produced $\vec{x}_{emp}$, we have no way of

[5]Later in the thesis, we will make use of sequential sampling schemes (denoted by H_{seq}), and while one-shot schemes are relatively trivial, sequential sampling schemes are not always so.

knowing with certainty that such a $\vec{\theta}_{emp}$ was responsible for producing some empirical data $\vec{x}_{emp}$. We do not know the real-world's 'analytical form' for the underlying data-generating process. Platt's internal calibration procedure addresses this, by introducing a 'ground-truth' parameter vector $\vec{\theta}_g$ to act as a known proxy for $\vec{\theta}_{emp}$, with which proxy data $\vec{x}_g$ can then be generated, in place of true empirical data $\vec{x}_{emp}$. By then creating a simple loss function (of absolute or squared absolute deviation between $\vec{\theta}$ and $\vec{\theta}_g$), one can quantitatively compare ABM procedures with ease. We discuss such a loss function in Section 2.8.

Thus, to address the use of $\vec{\theta}_{emp}$ and $\vec{x}_{emp}$ in Section 2.5, we actually use such proxies $\vec{\theta}_g$ and $\vec{x}_g$ as substitutes for some empirical time-series $\vec{x}_{emp}$, and an un-identifiable parameter vector $\vec{\theta}_{emp}$. Thus we perform internal calibration throughout this thesis. As a result, any references to $\vec{\theta}_{emp}$ or $\vec{x}_{emp}$ in this thesis, are references to such proxies $\vec{\theta}_g$ (which we select), and $\vec{x}_{emp}$ (which is generated using $\vec{\theta}_g$).

2.5.2 Frequentist Calibration - Algorithm

With the quantities we require now defined, we provide an algorithmic overview to frequentist calibration in Algorithm 1.

```
1:  Generate $\Theta = H_{os}(\vec{\theta}_{lower}, \vec{\theta}_{upper})$        ▷ $\Theta = \{\vec{\theta}_1, ..., \vec{\theta}_m\}$, $\Theta \in \mathbb{R}^{n \times m}$
2:  if External Calibration then
3:      Gather real-world data $\vec{x}_{emp}$                                 ▷ $\vec{x}_{emp} \in \mathbb{R}^{T_{emp}}$
4:  else if Internal Calibration then
5:      Select $\vec{\theta}_{emp}$, generate $\vec{x}_{emp}$                  ▷ $\vec{x}_{emp} \in \mathbb{R}^{T_{emp}}$
6:  end if
7:  Initialise fitness vector $\vec{y}$                                       ▷ $\vec{y} = \{y_1, ..., y_m\}$, $\vec{y} \in \mathbb{R}^m$
8:  for $i$ in $m$ do                                                          ▷ $m$ = number of sample points
9:      $y_i = 0$
10:     for $j$ in $r$ do                                                      ▷ $r$ = number of repetitions
11:         Set temporary seed, $\varepsilon = j$
12:         $\vec{x}_i = A(\vec{\theta}_i, \varepsilon)$
13:         $\vec{x}_{emp} = A(\vec{\theta}_{emp}, \varepsilon)$
14:         Calculate $f(\vec{x}_i, \vec{x}_{emp})$
15:         Add $f(\vec{x}_i, \vec{x}_{emp})$ to $y_i$
16:         Exit temporary seed state
17:     end for
18:     Divide $y_i$ by $r$
19: end for
20: $y^* = min(\vec{y})$
```

Algorithm 1: Frequentist Calibration with a one-shot sampling scheme. We use random search as our one-shot scheme.

The minimum fitness value y^*, resultant from Algorithm 1, represents the 'best' fitness value. That is to say we seek to minimise, rather than maximise the distance function here (and indeed throughout this thesis, unless otherwise stated). By then storing the parameter vector which generated y^*, we also receive the 'best' parameter vector $\vec{\theta}^* \in \mathbb{R}^n$. As one can imagine, the selection (and creation) of such a distance function is a well-studied topic. We use a single distance function from the ABM calibration literature, namely the Method of Simulated Moments (MSM)[12]. This distance function represents an intuitive and sensible first choice, as opposed to other advanced (but intricate) choices such as the Markov Information Criterion [20] and the GSL-div [21]. We discuss it below.

2.5.3 Method of Simulated Moments

The MSM has many implementations, but we use that of Franke in [22], which is also used by Platt in [11]. It is defined as

$$f(\vec{x}, \vec{x}_{emp}) = \vec{g}(\vec{x}, \vec{x}_{emp}) W \vec{g}(\vec{x}, \vec{x}_{emp}), \tag{18}$$

where

$$\vec{g}(\vec{x}, \vec{x}_{emp}) = (\vec{m}(\vec{x}) - \vec{m}_{emp}(\vec{x}_{emp})), \tag{19}$$

and

$$\vec{m}(\vec{x}) = \{m_1(\vec{x}), ..., m_k(\vec{x})\}, \tag{20}$$

where k represents the number of moments selected, $m_k(\vec{x})$ defines the kth moment being calculated over the time-series $\vec{x}$, and $\vec{x}$ represents the model-generated time-series defined in Equation 1.

The weighting matrix W, in Equation 18, is required to weight each moment according to its own variance, as Franke [22], Chen and Lux [23] and Platt [11] highlight. Ideally, the Newey-West estimator is used, as, in Chen and Lux's words, "the inverse of an unbiased and efficient estimator of the variance-covariance matrix of the moment conditions" needs to be used, and the Newey-West estimator fulfils these requirements optimally. However, as Platt writes, intuitively, we merely seek to weight moments according to their uncertainty. The higher the uncertainty, the lower the weight. This is captured by a simpler (but suboptimal) proposal of a diagonal form for W, with the elements defined as

$$w_{kl} = \begin{cases} \frac{1}{(1/T)\sum_{t=1}^{T}(m_k(\vec{z_t}^{emp}) - m_k(\vec{x}))^2}, & \text{if } k = l \\ 0, & \text{if } k \neq l. \end{cases} \tag{21}$$

Each moment of $\vec{m}(\vec{x})$ in Equation 20 is calculated via

$$m_k(\vec{x}) = \frac{1}{T}\sum_{t=L+1}^{T} m_k(\vec{z_t}), \tag{22}$$

where L is defined as the longest lag length amongst the selected moments, $m_k(\vec{z_t})$ defines the kth moment being calculated over the time-series $\vec{z_t} \subset \vec{x}$, where lastly, $\vec{z_t}$ represents a rolling window of the time-series $\vec{x}$, defined as

$$\vec{z_t} = \{x_t, x_{t-1}, ..., x_{t-L}). \tag{23}$$

The empirical moments $\vec{m}_{emp}(\vec{x}_{emp})$ in Equation 19 are defined analogously, with the only differences being that T_{emp} is used in place of T, and $\vec{x}_{emp}$ is used in place of $\vec{x}$. We refer the reader to Franke's work in [22] for a deeper discussion of the method.

We use the moments set of Chen and Lux, proposed in [23], as Platt also uses it in [13] for the predominantly financial and economic ABMs detailed above. These moments are, for any time-series $\vec{x}$ or $\vec{x}_{emp}$: the variance; the kurtosis; the autocorrelation function at lag 1; the autocorrelation function of the absolute value at lag 1; the autocorrelation function of the squared series at lag 1; the autocorrelation function of the absolute series at lag 5; and the autocorrelation function of the squared series at lag 5.

2.6 Bayesian Calibration

Frequentist methods are often criticised for requiring a pre-selected set of moments, and thus not using the full informational content within the data. What if we could rectify this, and even expand this method by including prior information on suitable parameter values? This is the task Grazzini sought to tackle in his 2017 paper, 'Bayesian estimation of agent-based models'.

At the heart of his method is of course, Bayes' theorem, which for our purposes, can be defined as

$$P(\vec{\theta}|\vec{x}_{emp}) = \frac{P(\vec{x}_{emp}|\vec{\theta})P(\vec{\theta})}{P(\vec{x}_{emp})}, \tag{24}$$

where $P(\vec{\theta})$ denotes a distribution modelling our prior beliefs in an ABM's parameters $\vec{\theta}$, and $P(\vec{x}_{emp}|\vec{\theta})$ denotes the likelihood of observing some empirical time-series $\vec{x}_{emp}$, given some vector of ABM parameters $\vec{\theta}$. Our prior beliefs in the parameter values are then updated by the likelihood of observing such data, to yield a posterior belief, in light of $\vec{x}_{emp}$, $P(\vec{\theta}|\vec{x}_{emp})$. We may ignore the normalisation constant, $P(\vec{x}_{emp})$, as Grazzini couples Bayes' theorem with a Markov chain Monte Carlo (MCMC) sampling procedure, which, as Platt writes [11], only requires the determination of a density function proportional to the posterior, which in Grazzini's case is $P(\vec{x}_{emp}|\vec{\theta})$.

Those familiar with applying Bayes' theorem to parameter estimation may be asking one question currently: how do we determine the likelihood? In analytically tractable models, likelihood functions can be analytically derived. For instance, in modelling some data-generating process as a Normal, Poisson or Bernoulli distribution. Such likelihood functions are exactly the quantity $P(\vec{x}_{emp}|\vec{\theta})$ in Equation 24, and thus allow us to sequentially update our posterior belief in $\vec{\theta}$, upon the arrival of new data throughout a calibration task. However, for complex ABMs, likelihood functions typically cannot be derived. So how did Grazzini deal with this, in order to apply Bayes' theorem to the ABM calibration problem?

Well, some proxy for the likelihood is required. Those familiar with likelihood-free inference methods (such as approximate Bayesian computation) will know that, similarly to the frequentist approaches previously, this proxy is usually some distance function again, measuring the similarity between empirical and model-generated data.[6] Let us now detail Grazzini's likelihood proxy.

2.6.1 Grazzini's Method

The first assumption Grazzini makes is that the ABM has reached some equilibrium state, with the selected output $\vec{x}$ now oscillating about some value

$$x^*(\vec{\theta}) = \mathbb{E}(x_t(\vec{\theta})|t > \tilde{T}), \tag{25}$$

where $\tilde{T}$ represents some timestep, after which equilibrium is assumed.

We then discard the first $\tilde{T}$ elements from $\vec{x}$, creating the vector $\vec{x}_{t>\tilde{T}} \in \mathbb{R}^{T-\tilde{T}}$. We repeat such generations of data $\vec{x}_{t>\tilde{T}}$ r times, and store the data in one vector $\vec{x}_{total} \in \mathbb{R}^{(T-\tilde{T})*r}$, where the $*$ symbol here denotes a scalar multiplication, rather than a second dimension to $\vec{x}_{total}$[7]. To be precise, by storing, we mean that at each generation of $\vec{x}_{t>\tilde{T}}$, we replace the first $(T-\tilde{T})$ zero elements of the zero-initialised vector $\vec{x}_{total}$ with $\vec{x}_{t>\tilde{T}}$. In this way $((T-\tilde{T})*r)$ zeros within $\vec{x}_{total}$ are replaced by $((T-\tilde{T})*r)$ datapoints from our ABM. We bin this data $\vec{x}_{total}$, and fit a non-parametric distribution to the resultant histogram via kernel density estimation (which, in essence, implements histogram smoothing) to provide us with a probability density function (PDF) for $\vec{x}_{total}$, which we denote with $g(x_t|\vec{\theta})$.

[6]We revisit this interesting similarity later.

[7]A small note is that we usually set $T = T_{emp}$ in such experiments, although this is to the discretion of the practitioner.

This PDF allows us to then estimate the probability of observing each empirical data point $x_t^{emp} \in \mathbb{R}$, by inputting x_t^{emp} to $g(x_t|\vec{\theta})$, and observing the density at each x_t^{emp}. We multiply each of these probabilities together, to arrive at a probability of observing the empirical data $\vec{x}_{emp}$, given our model parameters $\vec{\theta}$, i.e. $P(\vec{x}_{emp}|\vec{\theta})$. This is defined as

$$P(\vec{x}_{emp}|\vec{\theta}) = \prod_{t=1}^{T_{emp}} g(x_t^{emp}|\vec{\theta}), \tag{26}$$

and so we arrive at our proxy for the likelihood.

This Bayesian calibration procedure is not without its shortcomings however. The equilibrium assumption implies that the model-generated data arises from some i.i.d random process, and that $x_t(\vec{\theta}) \in \vec{x}$ is independent of $\{x_1(\vec{\theta}), ..., x_{t-1}(\vec{\theta})\}$ for all $t = \{1, ..., t\}$, as Platt states [11]. Thus, upon binning, intertemporal correlations are lost. This is undesirable as now, two PDFs $g(x_t|\vec{\theta}_1)$ and $g(x_t|\vec{\theta}_2)$ can be identical, yet have arisen from simulated data with vastly different temporal trends. Empirical data will then be assigned identical likelihoods such that $P(\vec{x}_{emp}|\vec{\theta}_1) = P(\vec{x}_{emp}|\vec{\theta}_2)$, even though one of the parameter sets is likely to reproduce the temporal trends of the empirical data better. Nevertheless, as Platt demonstrates [11], across a set of toy ABMs similar to ours, this method seems to perform better on ABM calibration tasks than the frequentist method outlined previously.

Extensions have been made (both by Grazzini [24] and Platt [13]) to this initial likelihood approximation, which now account for such temporal trends, but to limit complexity, we work with the original likelihood in this thesis.

2.6.2 Sampling in Bayesian Calibration

Earlier, in Section 2.5, we discussed one-shot sampling schemes, and noted the existence of the Latin Hypercube [18] and Nearly-Orthogonal Latin Hypercube schemes [19]. We now discuss sequential sampling schemes, an alternative to one-shot sampling which is more commonly used in Bayesian calibration[8]. Generally, these use some function

$$H_{seq}(\Theta_{hist}, \vec{\theta}_{lower}, \vec{\theta}_{upper}, \vec{y}) = \vec{\theta}, \tag{27}$$

whereby $H_{seq} \mapsto \mathbb{R}^n$, with n being the number of parameters. This function outputs the next parameter set $\vec{\theta} \in \mathbb{R}^n$ to execute, after taking the history of parameter vectors used, $\Theta_{hist} \in \mathbb{R}^{n\times m}$, both the lower and upper bounds for each parameter, $\vec{\theta}_{lower} \in \mathbb{R}^n$ and $\vec{\theta}_{upper} \in \mathbb{R}^n$, and a set of mean fitnesses $\vec{y} \in \mathbb{R}^m$, from the execution of the history of parameter vectors Θ_{hist}.

2.6.3 Bayesian Calibration Algorithm

With the definition of sequential sampling schemes above, we can now provide the Bayesian Calibration approach algorithmically in Algorithm 2. At Step 5, we refer to some sequential sampling scheme H_{seq}, which, while generally defined in Equation 27, has been purposefully separated from Algorithm 2 for clarity. We discuss such schemes below, particularly that used by Grazzini [25].

[8]But not always. Grazzini [24] for instance uses a one-shot scheme to reduce the complexity of his Bayesian calibration algorithm, while he details an improved likelihood. During our literature review, this added to the suspicion that the frequentist and Bayesian calibration schemes may be more similar than they appear initially.

```
1: Initialise prior $P(\vec{\theta})$
2: Initialise $\vec{\Theta}_{hist}$                              ▷ $\{\vec{\theta}_1, ..., \vec{\theta}_m\} = \Theta_{hist}$, $\vec{\Theta}_{hist} \in \mathbb{R}^{n \times m}$
3: Initialise $\vec{y}$                                          ▷ $\vec{y} \in \mathbb{R}^m$
4: Initialise $\vec{x}_{total}$                                  ▷ $\vec{x}_{total} \in \mathbb{R}^{(T-\tilde{T})*r}$
5: Initialise parameter vector $\vec{\theta}$ via $H_{seq}$          ▷ Step 1 in Alg. 4
6: Select pseudo-true parameter vector $\vec{\theta}_{emp}$
7: Select $\varepsilon$, generate $\vec{x}_{emp} = A(\vec{\theta}_{emp}, \varepsilon)$      ▷ $\vec{\theta}_{emp} \in \mathbb{R}^n$, $\vec{x}_{emp} \in \mathbb{R}^{T_{emp}}$, $\varepsilon \in \mathbb{R}$
8: for $i$ in $m$ do                                        ▷ $m$ = number of sample points
9:     Append parameter vector $\vec{\theta}_i$ to $\Theta_{hist}$
10:    $y_i = 0$                                            ▷ $y_i \in \vec{y}$
11:    for $j$ in $r$ do                                     ▷ $r$ = number of repetitions
12:        Set temporary seed, $\varepsilon = j$
13:        Generate data via $\vec{x} = A(\vec{\theta}_i, \varepsilon)$
14:        Store $\vec{x}_{t>\tilde{T}}$ in $\vec{x}_{total}$                ▷ $\vec{x}_{t>\tilde{T}} \in \mathbb{R}^{(T-\tilde{T})}$
15:        Exit temporary seed state
16:    end for
17:    Create PDF $g(x_t^{emp}|\vec{\theta})$, via KDE of $\vec{x}_{total}$
18:    for $j$ in $r$ do
19:        Set temporary seed, $\varepsilon = j$
20:        Calculate likelihood, $P(\vec{x}_{emp}|\vec{\theta}) = \prod_{t=1}^{T_{emp}} g(x_t^{emp}|\vec{\theta})$      ▷ Equation 26.
21:        Calculate posterior, $P(\vec{\theta}|\vec{x}_{emp}) = P(\vec{x}_{emp}|\vec{\theta}) \cdot P(\vec{\theta})$      ▷ Equation 24.
22:        Add $P(\vec{\theta}|\vec{x}_{emp})$ to $y_i$
23:        Exit temporary seed state
24:    end for
25:    Divide $y_i$ by $r$
26:    Input $\vec{y}, \Theta_{hist}, \vec{\theta}_{lower}, \vec{\theta}_{upper}$ to $H_{seq}$, return $\vec{\theta}_{i+1}$      ▷ Steps 4 - 13 in Alg. 4
27:    Store $\vec{\theta}_{i+1}$ in $\Theta_{hist}$
28: end for
29: Extract $\vec{\theta}^*$ from $\Theta_{hist}$
```

Algorithm 2: Bayesian Calibration with sequential sampling. We implement this algorithm upon our toy models, first via MCMC, and later via emulation. We separate the sequential sampling scheme from this algorithm, as we may use either MCMC or our Bayesian optimisation sampling scheme. Relevant steps in the multivariate MCMC algorithm are highlighted. Note Step 29, where $\vec{\theta}^*$ must be extracted from Θ_{hist}. For MCMC, this consists of discarding burn-in, binning the realised values of each parameter, and taking a selected moment from each histogram as the estimate of our parameter vector $\vec{\theta}^*$. For our emulation schemes, we simply use the parameter vector which yielded the lowest fitness y^*. Thus we don't specify how $\vec{\theta}^*$ is calculated from Θ_{hist} in this algorithm.

2.6.4 Our Sequential Sampling Schemes (MCMC)

The sequential sampling scheme that Grazzini [25] uses is Markov chain Monte Carlo (MCMC), specifically with the Metropolis-Hastings algorithm. While we're not interested in the precise details of MCMC, we provide a brief outline of univariate MCMC in Algorithm 3. Within the MCMC literature, it's typical to denote a chain of candidate and selected sample points, via $\theta_i^c, \theta_i^s \in \mathbb{R}^{1\times m}$, or equivalently, $\vec{\theta}_i^c, \vec{\theta}_i^s \in \mathbb{R}^{n\times m}$ in the multivariate case. In both cases, the subscript i denotes the i^{th} element of each chain. We maintain such notation for MCMC algorithms, and select only those steps in the algorithm which we require (in attempt to separate Bayesian calibration from MCMC).

Thus, at Step 4 in Algorithm 4, we may insert our current parameter history Θ_{hist}, alongside $\vec{\theta}_{lower}$, $\vec{\theta}_{upper}$ and $\vec{y}$, run Steps 4 - 13 of Algorithm 4, and output our next sample point. We then define all other required entities in Algorithm 2. This leaves some work to the reader, but should assist in isolating Bayesian calibration from MCMC.

We implement a rudimentary multivariate MCMC algorithm from Grazzini [25], which is a simpler variant upon Gibbs sampling. Each sample point is drawn by changing a single entry in the parameter vector, $\theta_{i,j}^s \in \vec{\theta}_i^s$, via $\theta_{i,j}^c = \theta_{i,j}^s + \eta_j\mathcal{N}(0,1)$, where $\theta_{i,j}^c$ and $\theta_{i,j}^s$ represent the j^{th} entries of the i^{th} candidate and current samples points, $\vec{\theta}_i^c$ and $\vec{\theta}_i^s$, in MCMC respectively. Each parameter η_j is an element in a vector $\vec{\eta} = \{\eta_1, ..., \eta_n\}, \vec{\eta} \in \mathbb{R}^n$, where each entry η_j represents an acceptance rate parameter, tuned to achieve roughly 25% acceptance in each dimension. Note here, that we don't seek to control the random state of the Normal random variable, and instead allow it to be drawn from the global state of our RNG. The acceptance mechanism following this (Step 8 in Algorithm 4 onwards) is analogous to that of the univariate MCMC algorithm. We now have everything we need to provide both the univariate and multi-variate MCMC algorithms implemented in this thesis, in Algorithms 3 and 4 respectively.

```
 1: Initialise $\theta_1^s$                                  ▷ $\theta_1^s \in \mathbb{R}$
 2: for $i$ in $m$ do                                        ▷ $m$ = number of sample points
 3:     Draw $\theta_i^c \sim \mathcal{N}(\theta_i^s, V)$    ▷ $V$ = proposal distribution variance
 4:     Calculate $\alpha = min\left\{1, \frac{P(\theta_i^c|\vec{x}_{emp})}{P(\theta_i^s|\vec{x}_{emp})}\right\}$
 5:     Generate $U \sim Uniform(0,1)$
 6:     if $\alpha < U$ then
 7:         Set $\theta_i^{s+1} = \theta_i^c$
 8:     else
 9:         Set $\theta_i^{s+1} = \theta_i^s$
10:     end if
11: end for
12: Discard burn-in from $\theta_i^s$
13: Bin $\theta_i^s$
14: Take selected moment of bin as parameter estimate
```

Algorithm 3: An outline of the univariate MCMC with the Metropolis-Hastings algorithm.

These sequential sampling schemes traverse the parameter space, in search of minima, which subsequently are likely to be oscillated about. By then binning the realised parameter values over the MCMC chain (one histogram for each parameter in $\vec{\theta}$), we arrive at distributions for each parameter, and can then take our selected moment as the parameter value post calibration.

Note however, that this procedure can be expensive in the number of evaluations, and the adaptive MCMC method Platt implements is an attempt at dealing with this [11]. This expensiveness is exacerbated as the dimensionality of the parameter space

```
1: Initialise $\vec{\theta}^s_1$                                    ▷ $\vec{\theta}^s_1 \in \mathbb{R}^n$
2: for $i$ in $m$ do                                   ▷ $n$ = number of parameters
3:     $\vec{\theta}^c_i = \vec{\theta}^s_i$
4:     for $j$ in $n$ do
5:         Set $\theta^c_{i,j} = \theta^s_{i,j} + \eta_j \mathcal{N}(0,1)$     ▷ $\theta^c_{i,j} \in \vec{\theta}^c_i$, $\theta^s_{i,j} \in \vec{\theta}^s_i$, $\eta_j$ = acceptance rate
6:         Calculate $\alpha = min\left\{1, \frac{P(\vec{\theta}^c_i|\vec{x}_{emp})}{P(\vec{\theta}^s_i|\vec{x}_{emp})}\right\}$
7:         Generate $U \sim Uniform(0,1)$
8:         if $\alpha < U$ then
9:             Set $\vec{\theta}^s_{i+1} = \vec{\theta}^c_i$
10:        else
11:            Set $\vec{\theta}^s_{i+1} = \theta^s_i$
12:        end if
13:    end for
14: end for
15: Discard burn-in from $\vec{\theta}^s$
16: Bin $\vec{\theta}^s$ in each dimension
17: Take selected moments of bins as parameter estimates
```

Algorithm 4: Multivariate MCMC with simplified Gibbs sampling. Note that $\vec{\theta}^c$ and $\vec{\theta}^s$ can indeed be written as matrices in this multivariate case, but we opt to maintain typical MCMC notation.

increases. Nevertheless, Platt shows in [11], that upon ABMs with input dimensionality less than roughly 15, and with moderate runtimes, that Bayesian Calibration can regularly outperform frequentist approaches.

2.7 Likelihood-Free Inference

Earlier, we mentioned how proxies to intractable likelihoods typically adopt the form of a distance function, which measures the similarity of empirical and simulated data. Let us elaborate on this now, with the guidance of a key source, Gutmann and Corander's work in [26].

Gutmann and Corander begin with a similar discussion of simulator models, intractable likelihoods and the creation of proxies for such likelihoods. Labelled as 'likelihood-free inference', Gutmann and Corander write that "For simulator-based models, likelihood-free inference methods have emerged in multiple disciplines. 'Indirect inference' originated in economics (...), 'approximate Bayesian computation' (ABC) in genetics (...), or the 'synthetic likelihood' approach in ecology (...)".

Indirect inference, as Platt writes in [11], is very similar in nature to MSM, with the main difference being the use of an auxiliary model, the parameters of which are then optimised by, again, some distance function, measuring the similarity between an empirically measured parameter vector (indeed an attempt to measure $\vec{\theta}_{emp}$), and one used to generate simulated data.

Approximate Bayesian computation (ABC), meanwhile, works within the Bayesian framework we outlined in Section 2.6. However, even as Grazzini writes [25], the only substantial difference is in the likelihood proxy. The sample generation phase of ABC is outlined in Algorithm 5. Grazzini recognises that ABC is a simulated minimum distance method (as MSM is), embedded in a Bayesian framework [25]. Thus, the methodological lines begin to blur.

Gutmann and Corander begin from a different starting point, namely that of the intractable likelihood for simulator models, and label (derivatives of) both frequentist

```
1:  Initialise $\vec{\theta}^s_1$
2:  Initialise tolerance threshold $h$                                  ▷ $h \in \mathbb{R}$
3:  Gather empirical data $\vec{x}_{emp}$
4:  for $i$ in $m$ do                                      ▷ $n$ = number of parameters
5:      $\vec{\theta}^c_i = \vec{\theta}^s_i$
6:      Generate $\vec{\theta}^c_i$ from selected multi-dimensional proposal distribution
7:      Generate data $\vec{x} = A(\vec{\theta}^c_i, \varepsilon)$
8:      Calculate $y = f(\vec{x}, \vec{x}_{emp})$, for selected ABC distance function
9:      if $y \leq h$ then
10:         Set $\vec{\theta}^s_{i+1} = \vec{\theta}^c_i$
11:     else
12:         Set $\vec{\theta}^s_{i+1} = \theta^s_i$
13:     end if
14: end for
15: Discard burn-in of $\vec{\theta}^s$
16: Bin remaining $\vec{\theta}^s$ in each dimension
17: Return selected moment of each dimension as parameter estimate, $\vec{\theta}^*$.
```
Algorithm 5: Sample generation in approximate Bayesian computation.

and Bayesian ABM calibration methods as solutions to this problem. The crucial insight of Gutmann and Corander's is that in place of the binning phase of ABC, a statistical model is used, which takes as input, the set of executed parameter sets Θ, and the vector of distance function outputs $\vec{y}$.

This, then, is incredibly similar to our frequentist calibration procedure. The statistical model for the likelihood allows for far more efficient parameter set selection, thus reducing the common expensiveness of ABC. Gutmann and Corander strip away many of the Bayesian elements of ABC, and focus solely on modelling the likelihood, which in ABC, is just some distance function anyway.

Thus we can use Gutmann and Corander's insight here to work within a unified framework for ABM calibration. The rest of the thesis looks at one particular type of statistical model for such a likelihood, Gaussian processes, and at how sequential sampling can be combined with it to form a Bayesian optimisation loop, a powerful tool within analogous machine learning hyperparameter search problems.

2.8 Loss Function

We've defined two forms of calibration experiment above, but in order to benchmark their performance, we need to define some measure for the quality of the resulting parameter vector, from each scheme. This extends to the emulation schemes which we discuss in the next chapter. We would like to show that our emulation schemes achieve similar values on such a measure, for less sample points executed.

The measure we shall use is defined by Platt in his 2020 paper [11], and is very simple. It is defined as

$$L(\vec{\theta}_{emp}, \vec{\theta}^*) = \|\vec{\theta}_{emp} - \vec{\theta}^*\|, \tag{28}$$

where $\vec{\theta}_{emp}$ is a ground-truth parameter value vector, as defined earlier in this chapter, and $\vec{\theta}^* \in \mathbb{R}^n$ is the parameter value selected by either our frequentist or Bayesian calibration scheme.

We make some minor adjustments to this loss function however. Given that the parameters are of different scales, small deviations in parameters with larger bounds may dominate this loss function. Thus, we normalise all parameters' bounds to $[0, 1]$, and normalise $\vec{\theta}_{emp}$ and $\vec{\theta}^*$ accordingly, before inputting them to the loss function above. Later in the thesis, we run our frequentist and Bayesian calibration schemes, and their emulated analogues, multiple times, and, unless otherwise stated, will calculate the loss for each run, calculate the mean of such losses, and report this mean as our estimate of $\vec{\theta}^*$. We will also provide the standard error to accompany such means, where relevant.

2.9 Numerical Experiments - Frequentist Calibration

In this section, we will conduct the frequentist calibration experiment outlined in Algorithm 1, using the MSM distance function. We will do so upon the AR(1), Franke-Westerhoff and Brock-Hommes models (in order of increasing dimensionality). We use the KS model later in the thesis, to test our selected high-dimensional Bayesian optimisation methods.

2.9.1 AR(1) Model

We seek to estimate the parameter α in Equation 2, which defines the AR(1) model. As noted in Section 2.5.1, we first define some 'ground-truth' value for α, α_{emp}, and generate a time-series $\vec{x}_{emp}$ from the AR(1) model using α_{emp}. We then generate $m = 50$ values of α via random search, within the bounds of α, [0,1].

We then input each value of α to the AR(1) model, and generate a time-series $\vec{x}$ of length 1000. We input each time-series $\vec{x}$, along with $\vec{x}_{emp}$, into our MSM distance function (as defined in Section 2.5.3), which yields an estimate of the time-series' distance. We repeat this $r = 5$ times, and calculate the mean of the resulting distances. We then store this mean in our results vector $\vec{y}$. We must be precise here, regarding the use of random number generators, and random number seeds, so let us take a moment to re-iterate this.

At the start of each of the r repetitions of this experiment (Step 10 in Algorithm 1), we set a temporary seed equal to the j^{th} iteration. This means that, when we generate a time-series $\vec{x}_i$, using the parameter set $\vec{\theta}_i$ (which for the AR(1) model is just a scalar θ_i, α_i), we also generate a new version of the pseudo-true time-series $\vec{x}_{emp}$, using the temporary seed j. Therefore, any variation in the two time-series is only due to the parameters input to each time-series, rather than the random numbers used in the stochastic element(s) of each time-series (which, for the AR(1) model, is the noise term in Equation 2). This idea is reflected in Algorithm 1.

We then plot the MSM values for each value of α used, in Figure 3. The vertical red line in Figure 3 denotes the value of the 'ground-truth' parameter, α_{emp}. Note the minimum value, roughly about α_{emp}. At this stage, we aren't interested in the precise value of the minimum MSM value (y^*), nor in the best parameter value $\vec{\theta}^*$. We merely want to see that our frequentist calibration experiment is able to recover a single parameter, in a one-dimensional parameter space, from a simple toy model. We will be interested in y^* and $\vec{\theta}^*$ however, when the dimensionality of the parameter space increases, as we won't we won't be able to visually inspect the MSM surfaces in such spaces. It's possible that random search, as implemented here, won't be able to recover such a higher-dimensional parameter vector.

2.9.2 Franke-Westerhoff model

We now present the same internal, frequentist calibration experiment for the Franke-Westerhoff model. This model has nine parameters, and we begin by visualising the MSM fitness surfaces for each of them in Figure 4.

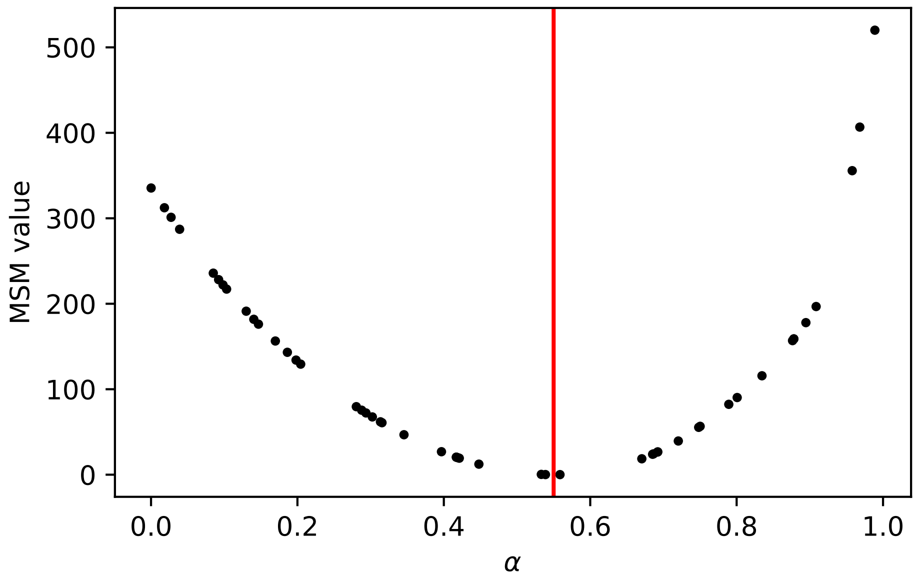

Figure 3: The results of running the frequentist calibration scheme outlined in Algorithm 1, on the AR(1) model. Note the minimum about the true parameter value, θ_{emp}. This shows that we have successfully recovered θ_{emp}.

We see that, for each parameter, we're again able to recover $\theta_{emp} \in \vec{\theta}_{emp}$. We see an MSM fitness surface clearly form in each case, with a minimum about θ_{emp}. It is precisely this surface we seek to emulate later in the thesis.

We then seek to extend this analysis to two-dimensions, and indeed for every dimensionality up to the full dimensionality of the model, ideally. But we encounter a problem in doing so. We now have to choose a subset of dimensions, the value of which we denote with $n_{sub} \in \mathcal{Z}^{+}$, $n_{sub} < n$. If we allow two parameters to vary, we set $n_{sub} = 2$. We now face a combinatorial problem, where we have to select two parameters to vary, but the number of combinations of which is given by

$$n_{combs} = \frac{n!}{n_{sub}!(n - n_{sub}!)}, \tag{29}$$

otherwise known as the binomial coefficient $(n \; n_{sub})^{\top}$, or n choose n_{sub}. For $n = 9$ and $n_{sub} = 2$, we have ($n_{combs} =$) 36 possible pairs of parameters to vary. This number peaks at 126 when $n_{sub} = 4$ and $n_{sub} = 5$. This is problematic, as any visual conclusions we reach regarding the performance of our internal, frequentist calibration procedure must now be made over n_{combs} parameter spaces.

For now, we merely note this, and still present a single two-dimensional MSM calbration experiment on the Franke-Westerhoff model in Figure 5.

Within this figure, we can determine visually that our frequentist calibration scheme is still working satisfactorily, via the minimum about $\vec{\theta}_{emp} = \{\alpha_n, \alpha_O\}$. These parameters were chosen as they are free parameters within Platt's analogous frequentist calibration experiments upon the Franke-Westerhoff model.

Now, how do such experiments extend to the full dimensionality of the Franke-Westerhoff model? In short, with less success. In order to show this, we first need to select one of the possible parameter spaces of each dimensionality. We have n_{combs} possible choices, and to simplify this, we simply define an ordering of parameters, through

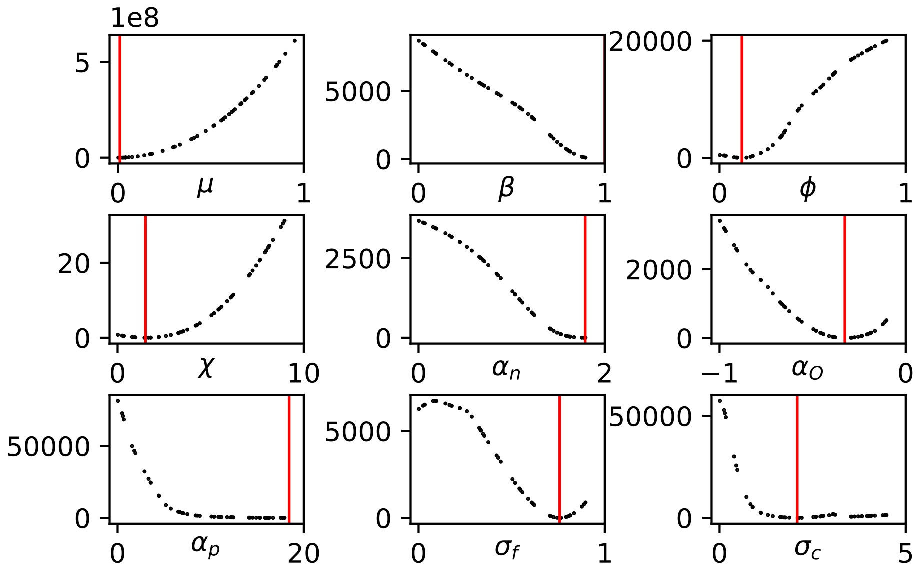

Figure 4: The results of running our frequentist calibration experiment upon the Franke-Westerhoff model, where we allow each dimension to vary in turn, and hold the rest to the values denoted in Table 1. We set $m = 50$, and thus draw 50 sample points randomly in each of the nine one-dimensional experiments. We set $r = 5$, and thus repeat each sample point five times (with five different seeds, as Algorithm 1 denotes). The lengths of the model-generated and pseudo-true time-series, T and T_{emp} respectively, were both set to 1000. Note that β has a true value very close to one. More importantly, note that each experiment results in a minima about each value in $\vec{\theta}_{emp}$, denoted by the red vertical line. Lastly, note the high sensitivity of μ, represented by the large (10^8) scale of its MSM values.

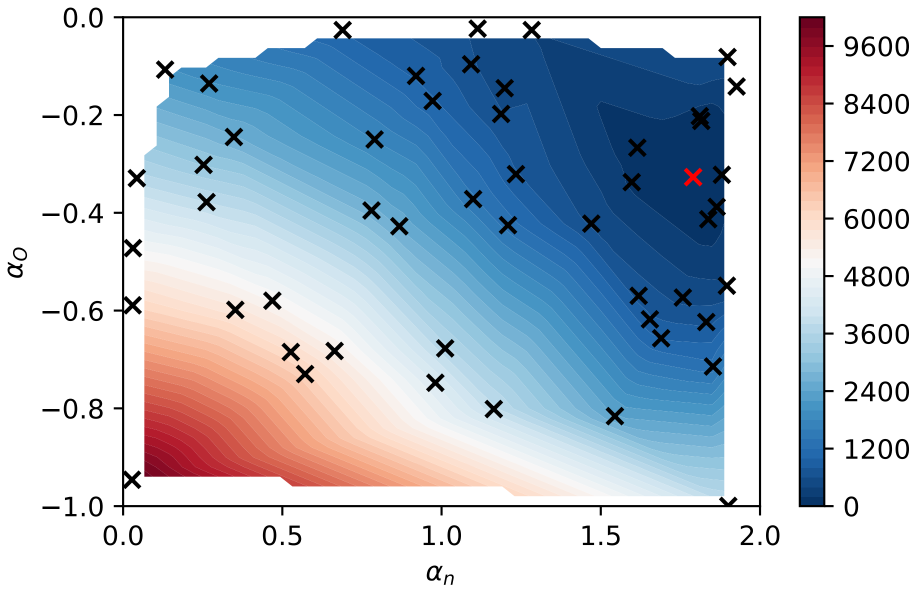

Figure 5: The two-dimensional MSM fitness surface, resulting from our frequentist calibration experiment upon the parameter set $\vec{\theta}_{emp} = \{\alpha_n, \alpha_O\}$. We set $m = 50$, thus drawing 50 samples from the two-dimensional parameter space. We also set T and T_{emp} to 1000, and $r = 5$, thus repeating each sample point five times, with a different seed each time once again. Note the lower MSM values about the true parameter vector, $\vec{\theta}_{emp}$, denoted by the red cross. The true values of $\vec{\theta}_{emp}$ are the values provided in Table 1.

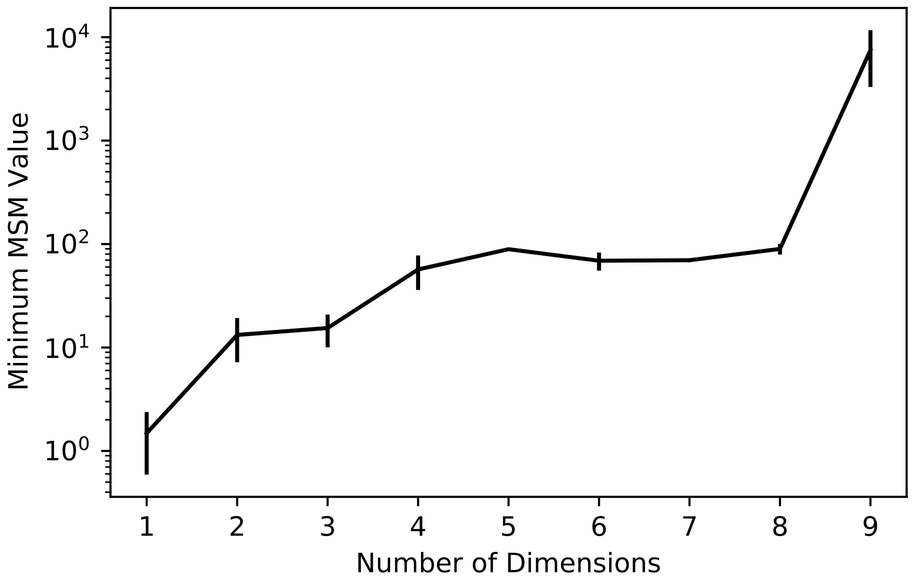

Figure 6: The minimum MSM value in our frequentist calibration experiment, upon the Franke-Westerhoff model, with incrementally increasing dimensionality. Each of the parameters of the Franke-Westerhoff model are freed, in the following order: [β, ϕ, χ, α_n, α_O, α_p, σ_f, σ_c, μ]. We run the experiment for 100 sample points in each dimensionality, with three repetitions at each sample point. All time-series are of length 100. We run this entire experiment five times, and report the mean and standard deviation of the resultant minima at each dimensionality.

which we incrementally add the next parameter, to provide us with a set of parameter spaces to test our internal, frequentist calibration experiment, with random search. The ordering is [β, ϕ, χ, α_n, α_O, α_p, σ_f, σ_c, μ].

We repeat our calibration experiment at each increasing dimension, and plot the mean and standard deviation of each set of minima, by dimension, in Figure 6. Note how as we increase the dimensionality, the mean MSM values also increase [9]. Random search is starting to struggle to explore the exponentially increasing n-dimensional parameter space sufficiently. The parameter set $\vec{\theta}_{emp}$ lies in an ever-increasing space, and random search has no way of utilising information from previous executions, to guide its search.

We complete our set of internal calibration experiments with a visualisation of the parameter values at each MSM calculated for Figure 6. In Figure 7, we see the extent to which random search is able to retrieve the parameters of the Franke-Westerhoff model, as dimensionality is increased in the previously defined order. We see relatively large deviations from the ground-truth vector $\vec{\theta}_{emp}$, which is in agreement with the large minimum MSM values seen in Figure 6. We now see the resultant best parameter values θ^*, that random search finds, in a parameter space of increasing dimensionality. Performance is degrading, and this is shown by the increasing values of y^* (Figure 6), and the deviation of $\vec{\theta}^*$ from $\vec{\theta}_{emp}$ (Figure 7).

[9]We should note, that this ordering has been selected to manage the MSM values large sensitivity to μ. While often taken as a fixed parameter in previous literature [11], we allow it to vary, but only after all other parameters have been freed. In preliminary experiments, we varied μ first, which led to all MSM values being dependant on the minimum MSM for μ, for any number of sample points m. For instance, setting $m = 1000$, and only varying μ still leads to a minimum MSM value of approximately $10^3 - 10^4$, which is far larger than the slowly increasing MSM minima from 1-4 dimensions.

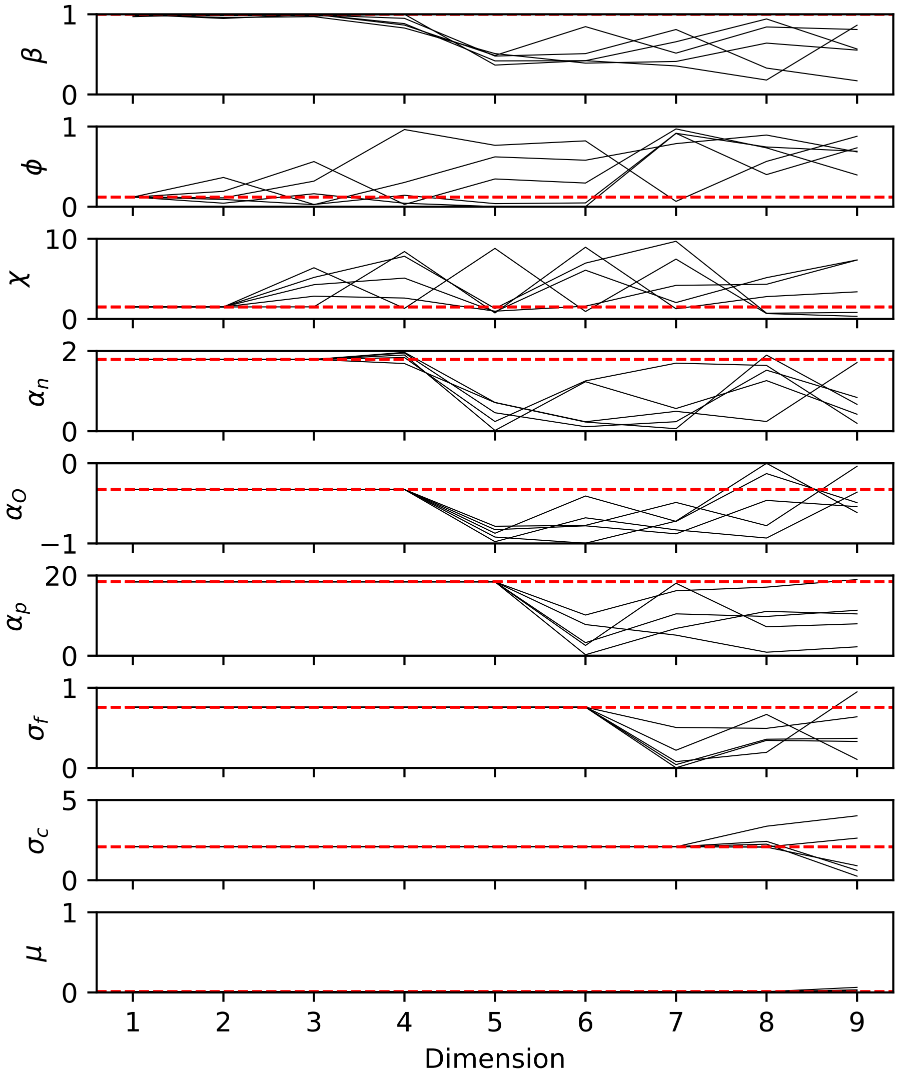

Figure 7: Parameter values for the sample points yielding the minimum MSM value, for each of the five runs of the frequentist calibration experiment in Figure 6. One parameter is freed incrementally at each dimensionality, in the previously defined order for the Franke-Westerhoff model. $\vec{\theta}_{emp}$ is represented by the horizontal red line on each subplot. Note the large deviations from $\vec{\theta}_{emp}$, which emerge as the dimensionality is increased.

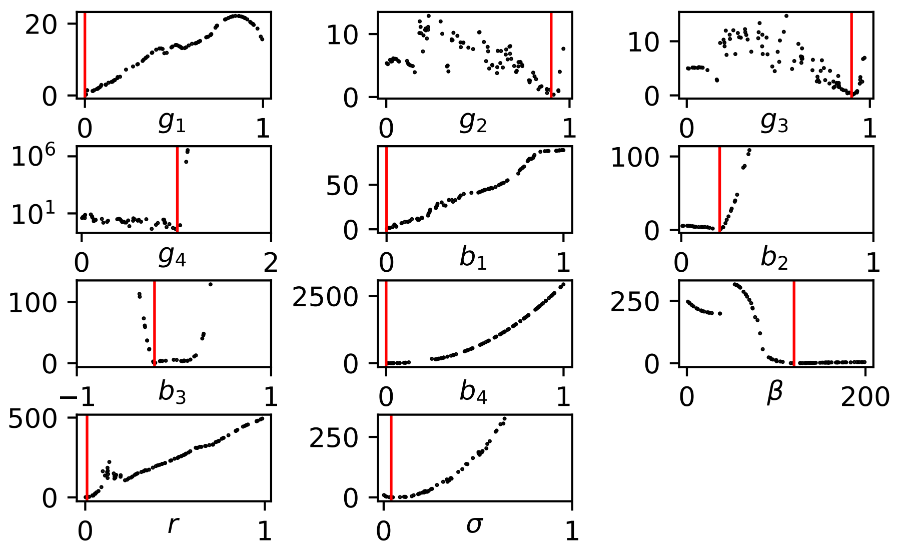

Figure 8: The results of running our frequentist calibration experiment upon the Brock-Hommes model, where we allow each dimension to vary in turn, and hold the rest to the values of the BH_2 configuration, denoted in Table 2. We draw 50 sample points randomly in each of the 11 one-dimensional experiments. We set $r = 5$, and thus repeat each sample point five times. The lengths of the model-generated and pseudo-true time-series were both set to 1000. Note that each experiment results in a minima about each value in $\vec{\theta}_{emp}$, denoted by the red vertical line. However, also note the increased noise and complexity of these fitness surfaces, in comparison to those of the Franke-Westerhoff model in Figure 4. Some parameters have regions where a valid time-series cannot be output, which is troublesome. We discuss this again later.

2.9.3 Brock-Hommes model

We reach similar conclusions by conducting the same set of internal, frequentist calibration experiments upon the Brock-Hommes model, in one, two and 11 dimensions (the full dimensionality of the Brock-Hommes model).

The results for each dimensionality are presented in Figures 8, 9, and 10, where again, we only present one of the possible two-dimensional parameter spaces in Figure 9. Once again, our calibration experiment is successful at low input dimensions, but performance degrades as dimensionality increases. We can see direct evidence of this in Figure 11, where the sample paths of the best performing parameter sets are plotted, from the five runs of the experiment visualised in Figure 10, just as for the Franke-Westerhoff model.

So how do we remedy this? One way would be to increase the number of parameter sets executed, but this is prohibitive when the ABM/simulator is expensive (in time or money). Later in the thesis, we discuss how Gaussian processes and Bayesian optimisation can help us guide this n-dimensional search through the MSM fitness surface, in a sample-efficient manner. But first, let us discuss the results of our Bayesian Calibration experiments, and see how they can also benefit from such methods.

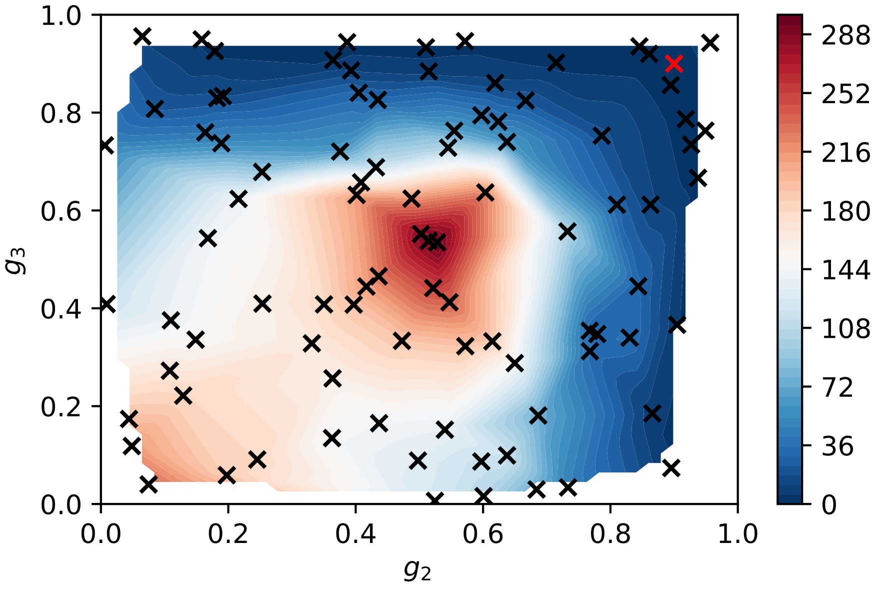

Figure 9: The two-dimensional MSM fitness surface, resulting from our frequentist calibration experiment upon the Brock-Hommes model, with the free parameter set $\vec{\theta}_{emp} = \{g_2, g_3\}$. We set $m = 50$, thus drawing 50 samples from the two-dimensional parameter space. We also set T and T_{emp} to 1000, and $r = 5$, thus repeating each sample point five times, with a different seed each time once again. Note the lower MSM values about the true parameter vector, $\vec{\theta}_{emp}$, denoted by the red cross. The true values of $\vec{\theta}_{emp}$ are the values provided in Table 2, under the BH_2 configuration.

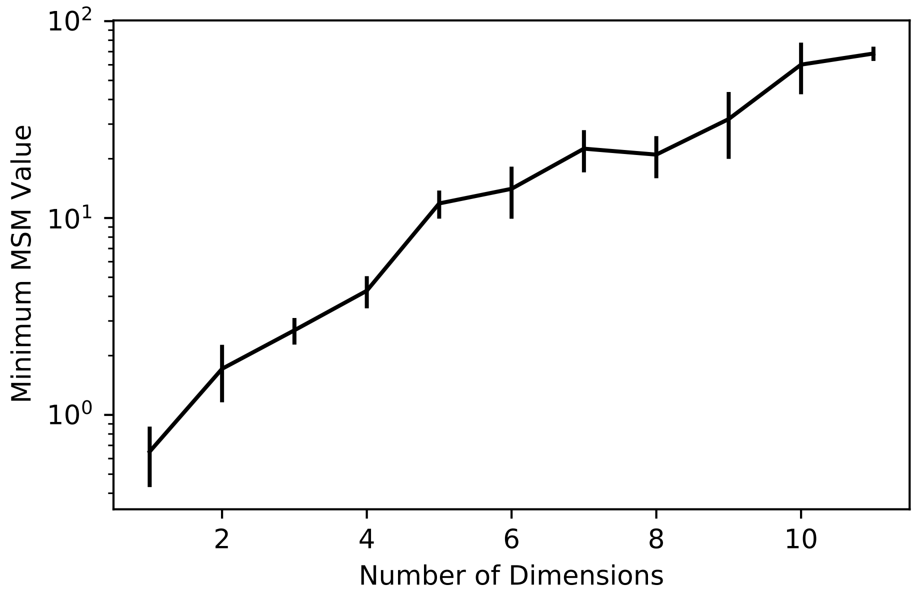

Figure 10: The frequentist calibration experiment, in Algorithm 1 was repeated 5 times, for the first n_{sub} entries in the following list of parameters, at each dimension n_{sub}: $[g_1, g_2, g_3, g_4, b_1, b_2, b_3, b_4, \beta, r, \sigma]$. Note the increasing MSM value as we increase dimension. The minimum MSM value in our frequentist calibration experiment, upon the Brock-Hommes model, with incrementally increasing dimensionality. Each of the parameters of the Franke-Westerhoff model are freed, in the following order: $[g_1, g_2, g_3, g_4, b_1, b_2, b_3, b_4, \beta, r, \sigma]$. We run the experiment for 100 sample points in each dimensionality, with three repetitions at each sample point. All time-series are of length 100. We run this entire experiment five times, and report the mean and standard deviation of the resultant minima at each dimensionality.

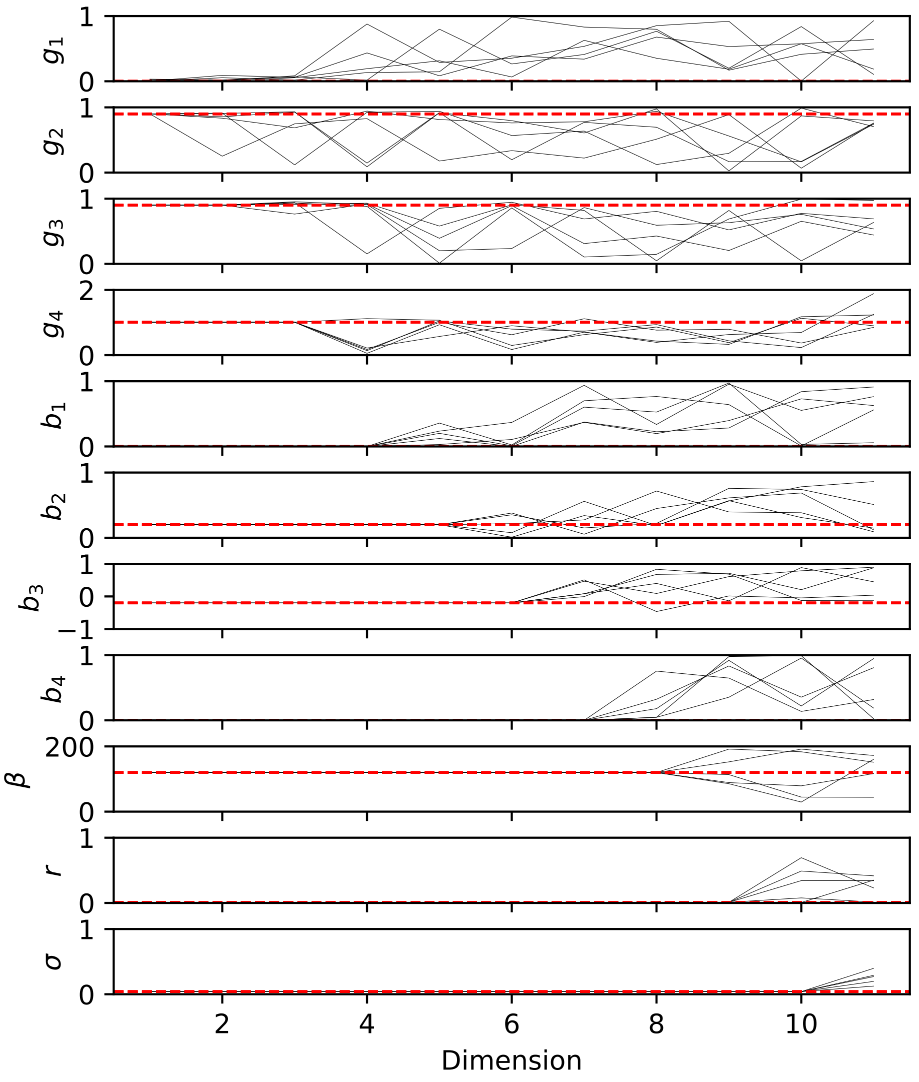

Figure 11: Parameter values for the sample points yielding the minimum MSM value, for each of the five runs of the frequentist calibration experiment in Figure 10, performed on the Brock-Hommes model. One parameter is freed incrementally at each dimensionality, in the previously defined order for the Brock-Hommes model. $\vec{\theta}_{emp}$ is represented by the horizontal red line on each subplot. Note the large deviations from $\vec{\theta}_{emp}$, which emerge as the dimensionality is increased

2.10 Numerical Experiments - Bayesian Calibration

In this section, we present illustrative examples of the Bayesian calibration experiment outlined in Algorithm 2. We do this for the AR(1), Franke-Westerhoff and Brock-Hommes models, again leaving treatment of the KS model till much later. We shall present histograms of the MCMC sample paths, for each parameter in each of the models, alongside the sample paths themselves. We shall also input the final parameter vector, realised from this Bayesian calibration scheme, to the loss function defined in Equation 28, along with a selected ground-truth parameter vector $\vec{\theta}_{emp}$. We do this only for the Franke-Westerhoff and Brock-Hommes models, as the AR(1) calibration problem is of too low dimensionality to pose a challenge to MCMC or Bayesian optimisation [10]. Thus we shall gain an estimate of the loss function for each of these models, under the Bayesian calibration scheme with MCMC.

In the next chapter, we discard this expensive MCMC scheme, and replace it with a GP and Bayesian optimisation scheme, and observe whether such an inexpensive scheme can generate similar loss values to that of MCMC, for the Franke-Westerhoff and Brock-Hommes models. We begin with the AR(1) model.

2.10.1 AR(1) Model

As an illustrative example of the Bayesian calibration scheme, let's take our one-dimensional AR(1) model, and run the Bayesian calibration scheme outlined in Algorithm 2 (with the MCMC scheme outlined in Algorithm 4), in an attempt to recover the 'ground-truth' one-dimensional parameter vector, α_{emp}[11]. In this experiment, we set $\alpha_{emp} = 0.55$ again. Algorithm 2 requires a number of sample points m, which we set to $m = 1000$. Algorithm 4 requires a set of tuning parameters $\vec{\eta} = \{\eta_1, ..., \eta_n\}$, which attempt to tune the acceptance rate to approximately 20-40%. The vector $\vec{\eta}$ is generated via

$$\vec{\eta} = \frac{(\vec{\theta}_{upper} - \vec{\theta}_{lower})}{c_\eta}, \tag{30}$$

where $c_\eta \in \mathbb{R}^+$ is some parameter which changes the weighting of the Normal noise in Step 5 in Algorithm 4. Thus, more generally, we create a set of rudimentary η_j parameters, to tune the proposal distribution in each dimension of our multivariate MCMC algorithm. However, in this AR(1) experiment, we only create a single η_j parameter. We set $c_\eta = 15$, and run the AR(1) model for 1000 timesteps each time, while setting $m = 1000$, thus running the MCMC chain for 1000 iterations. Lastly, we use a uniform prior here, and indeed throughout all Bayesian calibration experiments in this thesis. The resultant histogram of parameter values, sample path and posterior surface is plotted in Figure 12.

In Figure 12-a, we see our first example of a histogram of parameter values, realised from a single MCMC chain. The chain itself is visualised in Figure 12-b, and the posterior at each value of α used in the chain is plotted in Figure 12-c. In each subplot, the true parameter value, $\alpha = 0.55$, is given by the red line, while the mean value of α for the sample path is given in blue. Note, we only calculate the mean over the last two-thirds of the sample path, discarding the first third as burn-in.

Across the three subplots, we see that our selected moment, the mean, recovers the true parameter value very well. In this simple, one-dimensional Bayesian calibration

[10]We also don't calculate such losses for random search, as we shall see that Bayesian optimisation consistently produces lower MSM values than it. Thus we only calculate such losses for emulated frequentist calibration.

[11]Note, that while we have outlined a univariate MCMC scheme in Algorithm 3, we actually implement the multivariate MCMC algorithm of Algorithm 4 upon our one-dimensional AR(1) model, with the number of parameters $n = 1$. There should be no difference between the two schemes, apart from in the existence of the tuning parameter(s) η_j. We discuss how these are set above.

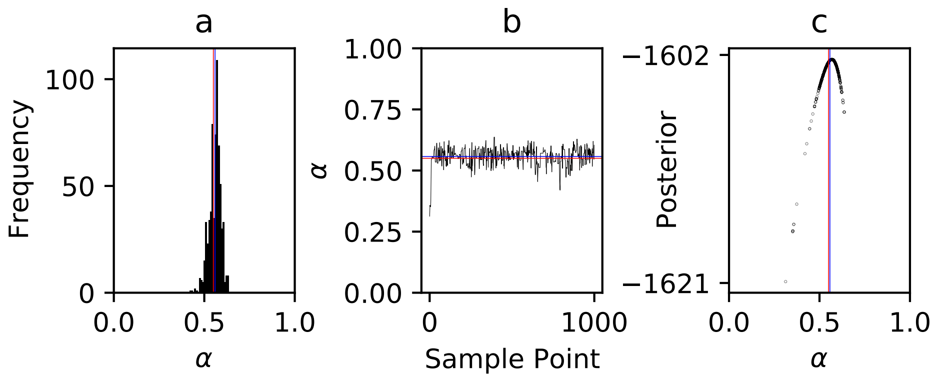

Figure 12: Results from our Bayesian calibration experiment, as outlined in Algorithm 2, upon the AR(1) model. In subplot (a), we plot the histogram of the single MCMC chain used, for the single parameter α in the AR(1) model. Subplot (b) visualises the realised values of α along the chain, while subplot (c) plots the posterior value for each point in the chain, against its value of α. The true parameter value, θ_{emp}, is denoted by the red line in all three subplots, and the mean of the chain (minus burn-in) is denoted in blue. Visually, it is very close to θ_{emp}.

problem, the MCMC chain is able to navigate to the region of the parameter space with high posterior, according to Equation 24, or equally (given our uniform prior) to the region of high likelihood, according to Equation 26. Such a posterior/likelihood surface is visualised in Figure 12-c, and this figure alone is of particular interest to us.

In Section 2.7, we detailed how Gutmann and Corander [26], instead of binning the sample path (as in 12-a), modelled the fitness surface of ABC with a Gaussian process, which, as we'll discuss thoroughly in the next chapter, allows for a much greater degree of sample efficiency. The fitness surface of ABC is formed by inputting model-generated data and empirical data to the distance function of ABC, as outlined in Algorithm 5. Here, we propose doing exactly the same thing, except, instead of ABC, we use Grazzini's Bayesian calibration method [25], and instead of a fitness surface formed from an ABC distance function, we use Grazzini's posterior, which we visualise in Figure 12-c.

This then allows us to adapt his scheme to models with long runtimes, and embrace the goal of sample efficiency, via the use of a Gaussian process. As will become evident also, Gaussian processes can be adapted to deal with large input dimensionalities, which isn't an area that literature following Grazzini's 2017 work (Grazzini in 2019 [24] and Platt in 2019 and 2020 [13] [11]) have attempted to tackle.

2.10.2 Franke-Westerhoff and Brock-Hommes

Now that we've demonstrated our Bayesian calibration experiment upon a simple one-dimensional model, let us expand it to our toy models of increased dimensionality. We'll then be able to see how well Bayesian calibration performs upon such models, and when we implement our more sample-efficient Bayesian calibration procedure (via emulation), hopefully the loss values will be comparable to that of the sample-expensive MCMC scheme, at a fraction of the cost.

We begin with the model of lower dimensionality; the Franke-Westerhoff model. We run the Bayesian calibration scheme in Algorithm 2, with the multivariate MCMC scheme of Algorithm 4. In this experiment, $\vec{\eta} \in \mathbb{R}^n$ is a vector rather than a scalar as previously, as $n = 11$. We set these as detailed in Equation 30, with $c_\eta = 15$ also. Due to the increased

Model	Method	Mean Loss	Standard Error
Franke-Westerhoff	Bayesian Calibration	0.908	0.089
Brock-Hommes	Bayesian Calibration	1.318	0.312

Table 5: Loss values for the Bayesian calibration experiments of Figures 13, 14, 35 and 36, the latter two of which are presented in Appendix A. We cannot compare such loss values between models, due to the different input dimensionalities, and instead compare them to emulated experiments later.

dimensionality, and thus the increased lengths of MCMC chains required ($m = 5000$), alongside the requirement of repetitions of each of these chains, we run the Franke-Westerhoff model for only 100 timesteps at each iteration of 10 MCMC chains.

We present the results of such an experiment in Figures 13 and 14, which visualise the histograms and sample paths for all chains, across each dimension of the Franke-Westerhoff model. We again, plot the true parameter values in red on each subplot, alongside the mean values for each parameter, over each MCMC chain, in blue. Note, such means are again only calculated for the last two-thirds of the data, as we again discard the first third as burn-in.

As we see in Figure 13, The combined histogram over the 10 MCMC chains does not peak about the true parameter value very often. A lot of exploration is visible in Figure 14, with the acceptance rates which control this provided in the last subplot of Figure 14. We attempt to tune c_η such that all chains have an acceptance rate of approximately 20-40%, but this doesn't always seem to be the case. In Figure 14 also, we can see the behaviour of all of the 10 MCMC chains, in each of the dimensions. In most dimensions, the chains cross over the true parameter value repeatedly, and yet don't converge about such values. This would indicate that the underlying fitness surface is likely to be flat around such true parameter values. This is most likely due to the simple i.i.d likelihood used in Equation 26.

It seems an adaptive MCMC scheme, which is able to maintain desired acceptance rates, alongside a more advanced likelihood, could aid the performance of this Bayesian calibration scheme in higher dimensions. This high level of crossover is again observed for the Brock-Hommes model's equivalent figures, which are relegated to Appendix A for brevity. The same issue of setting c_η to gain a desired acceptance rate is again observed, where after multiple iterations of preliminary experiments, it was evident the task would be difficult. The final acceptance rates vary between approximately 1-20%, yielding marginally better convergence, visually, about the true parameter values.

As discussed in Section 2.8, the loss function of Equation 28 allows us to quantitatively compare the quality of a set of calibration methods, over any single model. We input 10 mean parameter vectors, of length n (8 for the Franke-Westerhoff model, and 11 for the Brock-Hommes model) into the loss function, alongside the true parameter vector in each case. A set of 10 loss values are returned, from which we report their mean and standard error in Table 5. These loss values quantify the performance of our Bayesian calibration scheme on the full dimensionality of the Franke-Westerhoff and Brock-Hommes models[12].

Such values will only be of use when compared against analogous values for the emulated frequentist and Bayesian schemes in the next chapter. We will attempt to improve the sample-efficiency of Bayesian calibration, alongside replacing one-shot (random search) schemes. Both will be accomplished via our emulation scheme, again outlined in the next chapter.

[12]Bar the Franke-Westerhoff model's μ parameter of course.

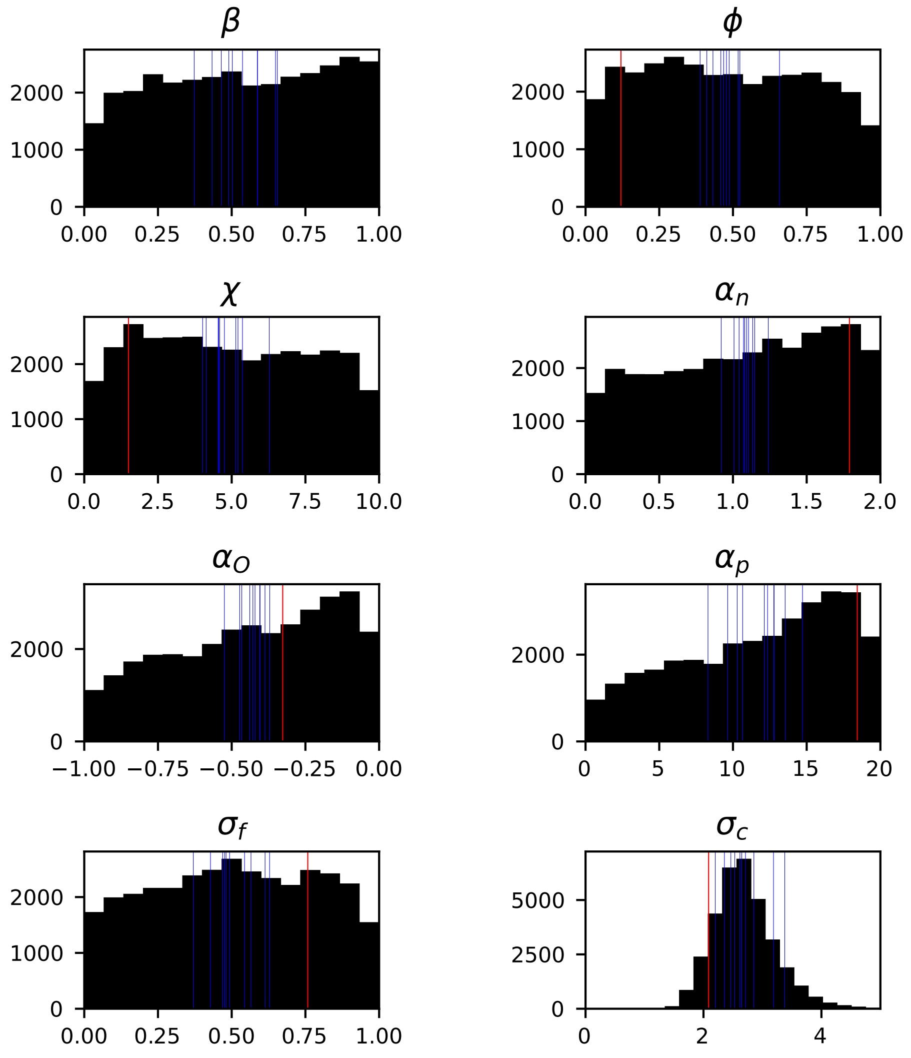

Figure 13: Histograms from our Bayesian calibration experiment, as outlined in Algorithm 2, upon the Franke-Westerhoff model. In each subplot, we plot the joint histogram, over 10 MCMC chains, for each denoted parameter. The true value of each parameter, within $\vec{\theta}_{emp}$, is denoted by the red line in each subplots, and the mean of each chain (minus burn-in) is denoted in blue. Note that β's true value is one. Less success is observed than in the one-dimensional Bayesian calibration of the AR(1) model.

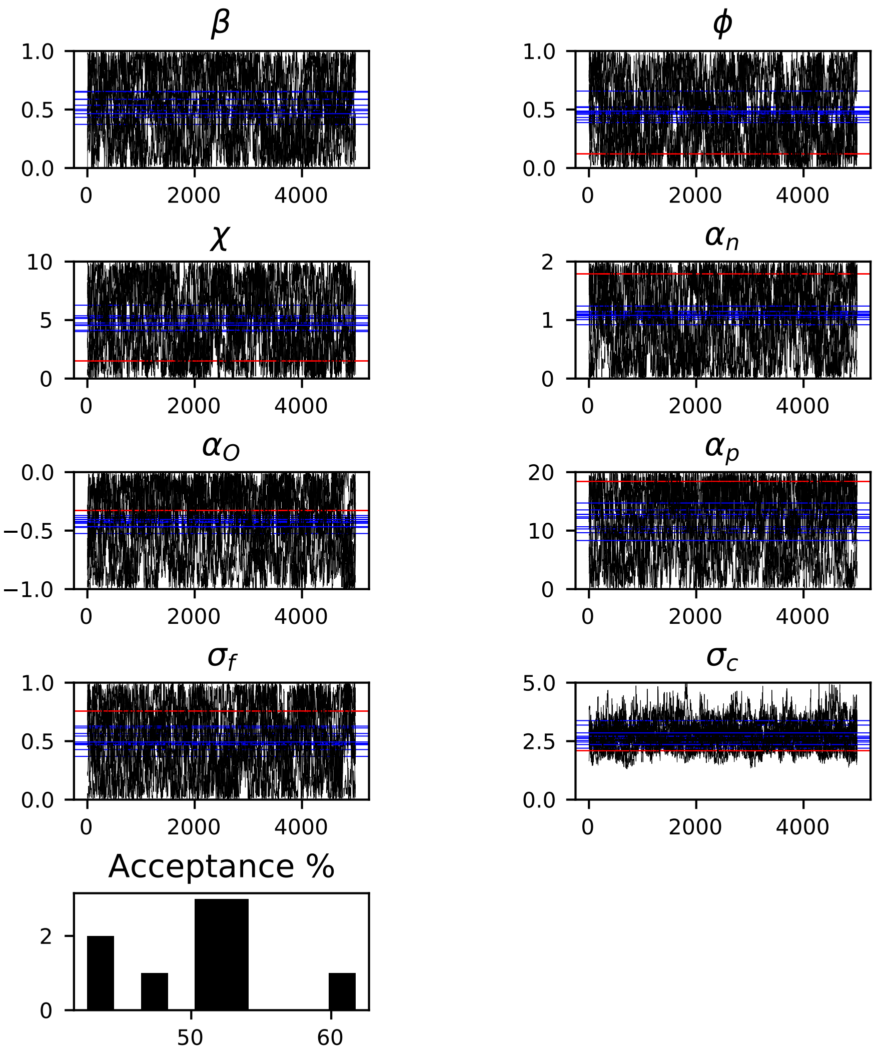

Figure 14: Sample paths from our Bayesian calibration experiment, as outlined in Algorithm 2, upon the Franke-Westerhoff model. In each subplot, the sample paths of 10 MCMC chains are plotted, for each denoted parameter. The true value of each parameter, within $\vec{\theta}_{emp}$, is denoted by the red line in each subplot, and the mean of each chain (minus burn-in) in blue. We plot the acceptance rates of the 10 chains in the last subplot.

Chapter 3

Emulation

In the previous chapter, we outlined the calibration experiments we'd like to conduct upon our toy ABMs. But more importantly, we showed how such experiments begin to struggle under higher input dimensionality. Commonly, when ABM practitioners (or rather practitioners of any complex simulatory models) conduct such experiments, some budget in time or number of executions exists. But how should we allocate such budgets, within the parameter space of the model? We described one-shot schemes in the previous chapter, which answer this question by maximising the Euclidean distance between sample points, via a single initial allocation of the entire budget. But is this not wasteful, given that as each sample point is executed, we retrieve more information about the shape of our fitness surface? This information could be exploited, in pursuit of minima on said surface.

Sequential sampling schemes (such as the MCMC schemes outlined previously) do just this. But given their Markovian nature (in that a standard MCMC chain's future state isn't conditional on the past states), they still only utilise a small amount of the information available. Information from past sample points isn't used to guide the search for future sample points, but instead the chain mostly searches for a point of higher fitness, almost always discarding points of lower fitness. We could benefit from methods which utilise all previous sample points to model the fitness surface, with the goal of increasing efficiency in the number of sample points used. If we allocate our budget in such a sequential scheme, can we reach better minima than our one-shot alternatives [13]?

This question provides motivation for the solution we discuss now, which is to emulate the fitness surface with a computationally inexpensive model, built around the sample points already executed. A large array of emulator models are available, the reviewing of which we won't undertake here. We work with a common emulation method in this thesis, namely Gaussian process regression (GPR). This method interpolates between sample points on the n-dimensional fitness surface, creating a direct surrogate, with analytical solutions for the value and uncertainty at all unsampled points. These features make it a popular choice for emulation across a myriad of simulatory fields. The availability of analytical uncertainty estimates in particular, is something that competing emulation techniques (such as random forest methods [27]) often lack.

Computational chemists have used GPR to form surrogates of multi-dimensional free-energy surfaces [28], while computational neuroscientists have implemented it to produce models of expensive-to-undertake EEG data-generating experiments [29]. Design engi-

[13]Tangentially, can we quantify the uncertainty in the output of our model, at unexplored points in parameter space? Such uncertainty estimates are unavailable in the Latin Hypercube Sampling and NOLH.

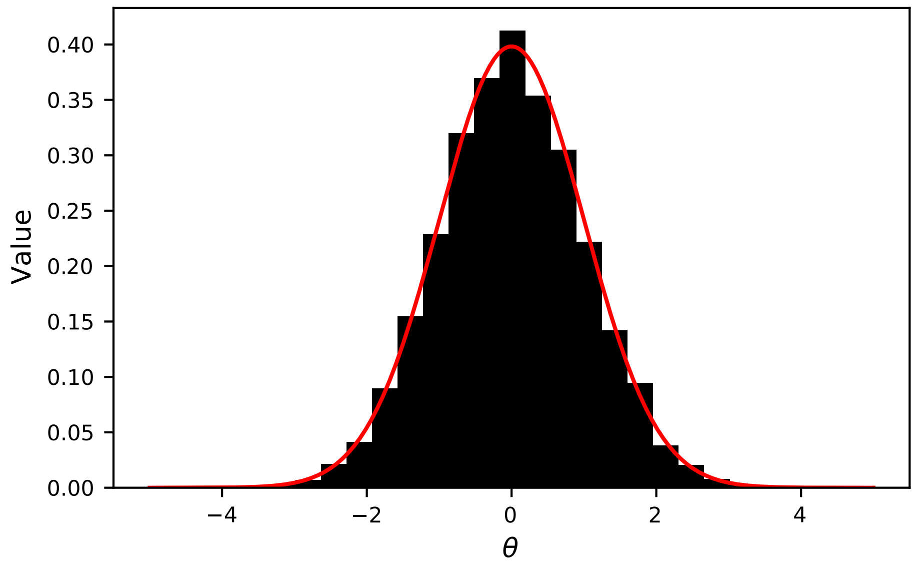

Figure 15: 10^4 samples from the Gaussian distribution of Equation 31, with $\mu = 0$ and $\sigma = 1$. A Gaussian distribution is then fitted to this data.

neers use it to emulate high-dimensional simulations of buildings [30], while computational cosmologists use it to constrain simulatory model parameters post data observation [30]. The list continues, with epidemiology [31], systems biology [32] and robotics [33] all benefiting from emulating fitness surfaces of expensive computer simulations. Thus, rewording an old adage, if it is good enough for them, it must be good enough for us?

We review the ABM field's attempts to answer this question in the next chapter, but not before detailing GPR, and its relative within the machine learning literature, Bayesian optimisation.

3.1 Gaussian Process Regression

We begin our discussion of GPR with the univariate Gaussian distribution, which lies at the heart of GPR. The method can be difficult to grasp for those new to it, and so we begin from early principles to remedy this.

Resultantly, let us begin with the univariate Gaussian distribution, defined as

$$f(\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(\theta-\mu)^2}{2\sigma^2}\right), \tag{31}$$

where $\sigma \in \mathbb{R}^+$ denotes the standard deviation of the distribution, and $\mu \in \mathbb{R}$ denotes the mean. The variable $\theta \in \mathbb{R}$ represents the input to our Gaussian distribution, the output of which is visualised in Figure 15.

We eventually want to work with multivariate Gaussian distributions in GPR, the dimensionality of which will be equal to our number of sample points, m. With this in mind, let us now define the bivariate Gaussian distribution, denoted by

$$f(\vec{\theta}) = \frac{1}{2\pi}|\Sigma|^{-1/2} \exp(-(\vec{\theta}-\vec{\mu})^T \Sigma^{-1} (\vec{\theta}-\vec{\mu})), \tag{32}$$

where $\vec{\theta}, \vec{\mu} \in \mathbb{R}^2$, $\Sigma \in \mathbb{R}^{2\times 2}$, and $|\Sigma| = \det \Sigma$.

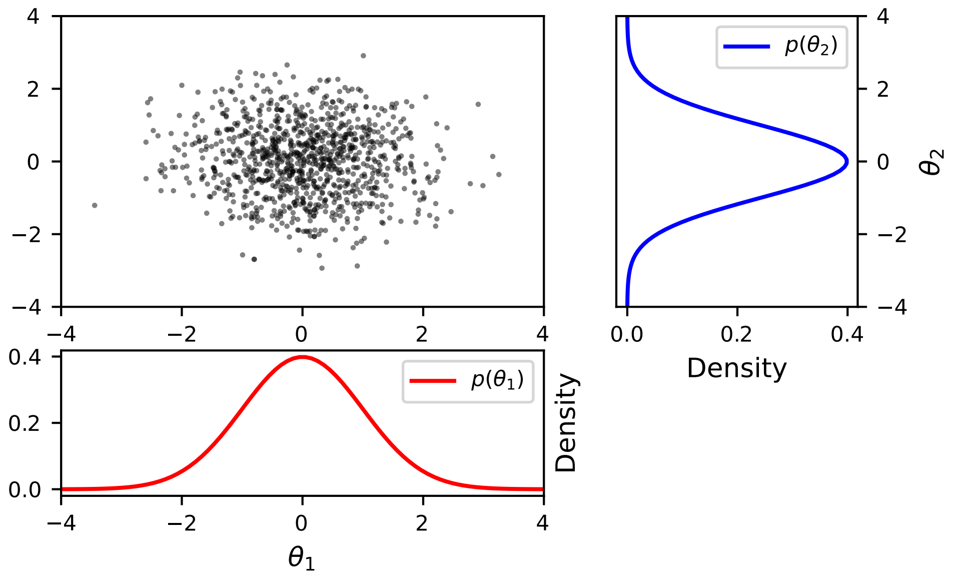

Figure 16: 1000 samples drawn from a bivariate Gaussian distribution, with $\vec{\mu}$ and Σ as in Equation 33.

Let us visualise some draws from this bivariate distribution now, in Figure 16. This data was generated using the following mean vector and covariance matrix:

$$\vec{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{33}$$

Note how the marginal distributions in Figure 16 both resemble Gaussian distributions themselves. Now, let's observe the effects of changing $\vec{\mu}$ and Σ. When we set them to

$$\vec{\mu} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 0.2 \end{pmatrix}, \tag{34}$$

the samples adopt the shape seen in Figure 17. Note how the mean of θ_2 has increased by 1, and its variance has shrunk. Now, let's display the effects of changing the off-diagonal elements of Σ. If we set

$$\vec{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix}, \tag{35}$$

the samples adopt the shape seen in Figure 18. Now, how do we interpret this different pattern, given our changes to the covariance matrix? Well, each element $\sigma_{ij} \in \Sigma$ (where i and j denote the row and column of Σ), tells us about the covariance between the random variables $\theta_i, \theta_j \in \vec{\theta}$. By increasing the off-diagonal elements from 0 to 0.9, we increase the correlation between the two random variables. That is to say, knowing the value θ_i now provides more information about the value of θ_j, and vice versa. This results in a narrower cloud of points, more highly centred about the axis $\theta_1 = \theta_2$. Such information 'flow' between random variables will be crucial to the construction of Gaussian processes.

3.1.1 Gaussians in Greater Than Two Dimensions

Now, one may be wondering, what does each random variable θ_i represent in our ABM calibration problem? In short, they will be the set of sample points in our ABM calibra-

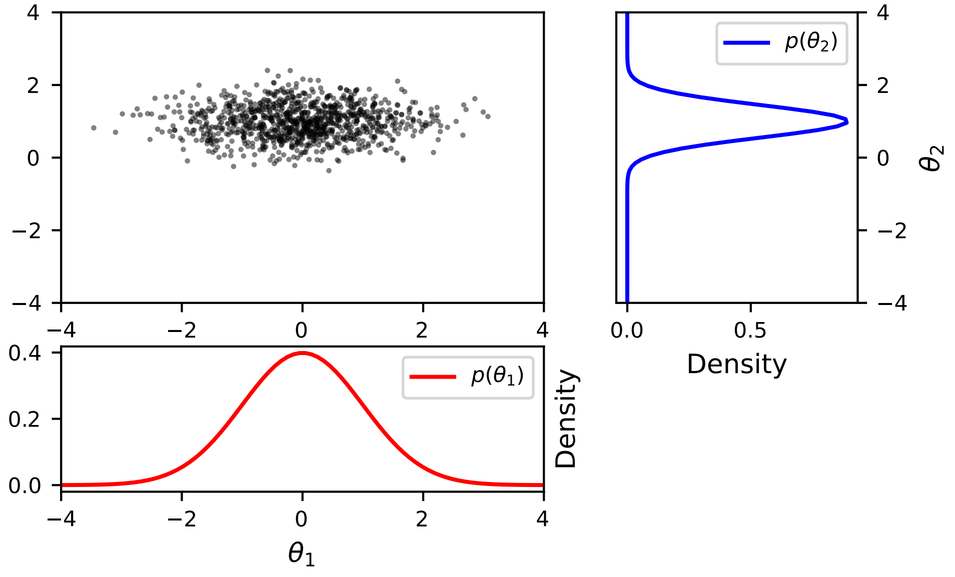

Figure 17: 1000 samples drawn from a bivariate Gaussian distribution, with $\vec{\mu}$ and Σ as in Equation 34.

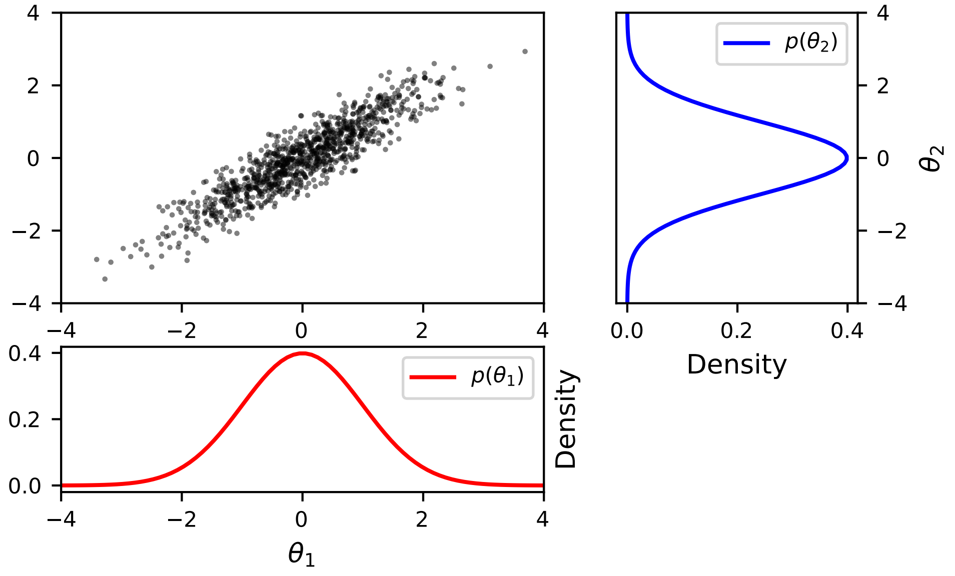

Figure 18: 1000 samples drawn from a bivariate Gaussian distribution, with $\vec{\mu}$ and Σ as in Equation 35.

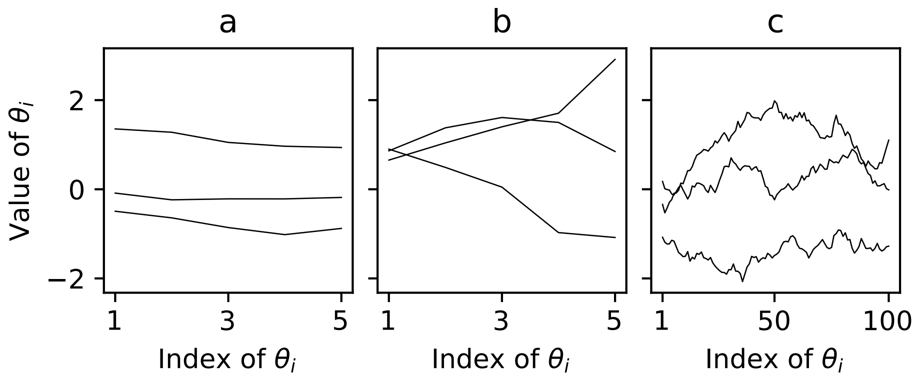

Figure 19: Three samples from multivariate Normal distributions, with changing dimensionality and covariance matrices. Subplot (a) presents draws from a five-dimensional Normal distribution, with $\vec{\mu}$ and Σ as in Equation 36. Subplot (b) presents draws from a five-dimensional Normal distribution again, with $\vec{\mu}$ as in Equation 36, and Σ as in Equation 37. Note the increased difference in value between index 1 and index 5. Subplot (c) presents draws from a 100-dimensional Normal distribution, with $\vec{\mu}$ and Σ as in Equation 38.

tion problem. This will become apparent soon however, as we extend our discussion to multivariate (> 2) Normal distributions (MVNs).

In order to progress towards MVNs of dimension m (where again, m is the number of sample points as in the previous chapter), we would like to re-produce figures such as Figure 16 for such MVNs, but this will prove difficult unless we reframe our plotting procedure, as we can't visualise more than two dimensions. Therefore, instead of plotting the value of θ_1 against θ_2, from repeated bivariate Gaussian samples, let us plot the *index* i of the random variable θ_i on one axis, and the random variable itself, θ_i, on the other. Let us also extend our MVN to a five-dimensional MVN, to illustrate this. We plot repeated samples from such a five-dimensional MVN in Figure 19-a, where our mean vector and covariance vector are now

$$\vec{\mu} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \Sigma = \begin{pmatrix} 1 & 0.99 & 0.98 & 0.97 & 0.96 \\ 0.99 & 1 & 0.99 & 0.98 & 0.97 \\ 0.98 & 0.99 & 1 & 0.99 & 0.98 \\ 0.97 & 0.98 & 0.99 & 1 & 0.99 \\ 0.96 & 0.97 & 0.98 & 0.99 & 1 \end{pmatrix}. \tag{36}$$

From our covariance matrix Σ, we see that the lowest value is 0.96, in the entry σ_{15}. This tells us that the covariance between the first random variable, θ_1 and the fifth random variable θ_5, is still relatively high. Thus, the values at index 5 are similar to those at index 1, for each separate five-dimensional sample from our five-dimensional MVN. If we lower these off-diagonal covariances, say to the following values:

$$\Sigma = \begin{pmatrix} 1 & 0.9 & 0.8 & 0.7 & 0.6 \\ 0.9 & 1 & 0.9 & 0.8 & 0.7 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.7 & 0.8 & 0.9 & 1 & 0.9 \\ 0.6 & 0.7 & 0.8 & 0.9 & 1 \end{pmatrix}, \tag{37}$$

and resample from our five-dimensional MVN, we see, in Figure 19-b, that values of θ_5 are now not as close to θ_1, for each respective sample path.

Now, if we extend this treatment to a 100-dimensional MVN, with

$$\vec{\mu} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}, \Sigma = \begin{pmatrix} 1 & \cdots & 0.5 \\ \vdots & \ddots & \vdots \\ 0.5 & \cdots & 1 \end{pmatrix}, \tag{38}$$

where $\vec{\mu} \in \mathcal{R}^{100}$ and $\Sigma \in \mathcal{R}^{100\times 100}$, and plot repeated samples from the 100-dimensional MVN as previously, we arrive at sample paths such as those plotted in Figure 19-c.

Such sample paths are discretised by the index i of each random variable $\theta_i \in \vec{\theta}$, and one could imagine, if we took the dimensionality to be very high, we could begin to see how we may form surrogate fitness surfaces with Gaussian processes.

At this point, we should mention that $\vec{\mu}$ and Σ are key to GPs, because they entirely define an MVN. Thus, when $\vec{\theta} = \{\theta_1, ..., \theta_m\}$ represents a vector of random variables, we could hypothesise that such a vector is drawn from a multivariate normal distribution (MVN), if we believed this to be so. In other words, we could hypothesise that

$$\vec{\theta} \sim \mathcal{N}(\vec{\mu}, \Sigma). \tag{39}$$

We can formalise this via the following definition for GPs:

Definition 3.1 (GP)**.** A Gaussian process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution [34].

Thus, we again see how a GP is completely defined by its mean vector $\vec{\mu}_m$, and its covariance matrix Σ_m. We could have started our discussion of GPs here in fact, and still have been able to define all required quantities. But it is hoped that through the discussion leading up to this, the reader is able to appreciate more intuitively, how MVNs are manipulated to create a Gaussian process.

The aware reader may note that Definition 3.1 is similar to that of a stochastic process, in being a collection of random variables. Indeed we can formalise this also, by stating that

$$Y = \{y(\theta) : \theta \in \Theta\}, \tag{40}$$

where θ represents the selected index, $y(\theta)$ represents the value of the MVN sample at index θ, and $\Theta \subset \mathbb{Z}^+$ represents the set of indices used.

If $\Theta = \mathbb{Z}^+$, that is to say that Θ becomes uncountable, then Y becomes an infinite dimensional stochastic process. In such scenarios, it makes sense to re-define the mean vector $\vec{\mu}$ as a mean function $\mu(\theta) \mapsto \mathbb{R}$, and the covariance matrix Σ as a covariance function $\Sigma(\theta, \theta') \mapsto \mathbb{R}$, where θ and θ' represent any two indices $\theta, \theta' \in \Theta$. Note also that we refer to the indices individually as θ and θ' now (as opposed to i and j), to highlight the shift to an infinite set of indices Θ.

The mean function and covariance function are then defined as the first and second-order statistics of the data-generating process $y(\theta)$ [35], namely

$$\mu(\theta) = \mathbb{E}[y(\theta)], \tag{41}$$

and

$$\Sigma(\theta, \theta') = Cov(y(\theta), y(\theta')), \tag{42}$$

where

$$Cov(y(\theta), y(\theta')) = \mathbb{E}[(y(\theta) - \mu(\theta)(y(\theta') - \mu(\theta'))]. \tag{43}$$

Thus, we define the GP, in this continuous domain, as

$$Y \sim \mathcal{N}(\mu(\theta), \Sigma(\theta, \theta')), \tag{44}$$

where Y denotes a set of random variables, as defined in Equation 40, and the notation '$\sim$' denotes that the set of random variables Y are sampled from the multivariate Normal distribution $\mathcal{N}(\vec{\mu}(\theta), \Sigma(\theta, \theta'))$, where $\theta, \theta' \in \Theta$.

3.2 Mean Function and Covariance Functions

Now, how do we select these mean and covariance functions? For the mean function, we shall set it to

$$\mu(\theta) = 0 \tag{45}$$

for much of this thesis. Often, data can be normalised to have a mean of zero, and unless there is some strong prior reason to introduce some constant $\mu(\theta) = c$, or trend, such as $\mu(\theta) = \beta\theta$, most practitioners set $\mu(\theta) = 0$[14].

Covariance functions are also referred to as kernels in the machine learning literature, and at this point, we see it fit to adopt such a notation. Henceforth, the covariance function $\Sigma(\theta, \theta')$ shall be denoted by $K(\theta, \theta')$, and where we refer to a covariance *function*, we also mean kernel. This allows us to use the letter Σ for the covariance *matrix* going forward. Such a notation is supported by Rasmussen's seminal textbook on Gaussian processes, [34], within which he writes the following: "A general name for a function K of two arguments mapping a pair of inputs $\theta \in \Theta$, $\theta' \in \Theta$ into $\mathbb{R}$ is a *kernel*." (adjusted to match the notation of this thesis).

The rules for defining such a covariance function/kernel are slightly more restrictive than that for the mean function, however. The kernel $K(\theta, \theta')$ must be a positive semidefinite kernel. That is to say, if we let Θ be a nonempty set, then the (symmetric) function $K(\theta, \theta') \mapsto \mathbb{R}$ is labelled a positive semidefinite kernel if

$$\sum_{i=1}^{m} \sum_{j=1}^{m} c_i c_j K(\theta_i, \theta_j) \geq 0$$

for all $\theta_i, \theta_j \in \Theta$, given $m = \mathbb{N}$ and $c_1, ..., c_m \in \mathbb{R}$.

It is very common to assume that the covariance function/kernel $K(\theta, \theta')$ is simply a function of the distance between the two points (θ - θ') (as opposed to the dot product for instance). Such kernels are typically labelled as stationary kernels, as they are invariant to translations in the input space. The Euclidean distance is all that matters.

The first kernel we shall define is the squared exponential kernel, which is defined as

$$K_{rbf}(\theta, \theta') = \exp(-\frac{1}{2}(\theta - \theta')^2). \tag{46}$$

This kernel is by far the most widely used kernel in GPR. By combining it with a mean function $\mu(\theta) = 0$, we define the GP

$$Y \sim \mathcal{N}(0, K_{rbf}(\theta, \theta')), \tag{47}$$

and draw three sample paths from it in Figure 20, over an (effectively) continuous set of indices Θ. We do this by instantiating a 100-dimensional MVN, with the mean and covariance function as in Equation 47, and drawing three 100-dimensional vectors from such an MVN, with a covariance matrix $\Sigma \in \mathbb{R}^{100 \times 100}$.

Note the smoothness of the sample path plotted. This is a direct result of the squared exponential kernel, which is infinitely differentiable, and so embeds an infinite smoothness assumption upon our GP model. Multiple samples from the GP defined in Equation 47 yield different sample paths, and in this way we can see how we sample from a function space, the priors of which are provided by the mean function and covariance function [34].

We provide a visualisation of the covariance matrix, for the GP of Figure 20, in Figure 21. Note the rapid reduction in covariance away from the diagonal, induced by the exponential nature of the RBF kernel.

[14]Note here, that we see the first signs of embedding prior information about our fitness landscape, into our GP model.

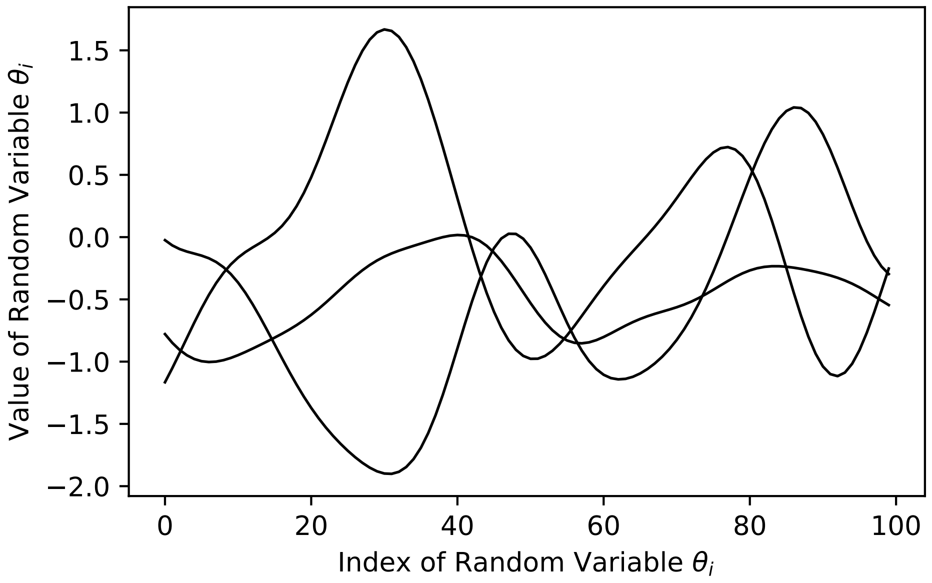

Figure 20: Three samples drawn from a Gaussian process, defined in Equation 47, with an RBF kernel, as detailed in Equation 46. The multivariate Normal distribution is 100-dimensional. We plot the index of each random variable, in each 100-dimensional draw, against the variable with said index's value.

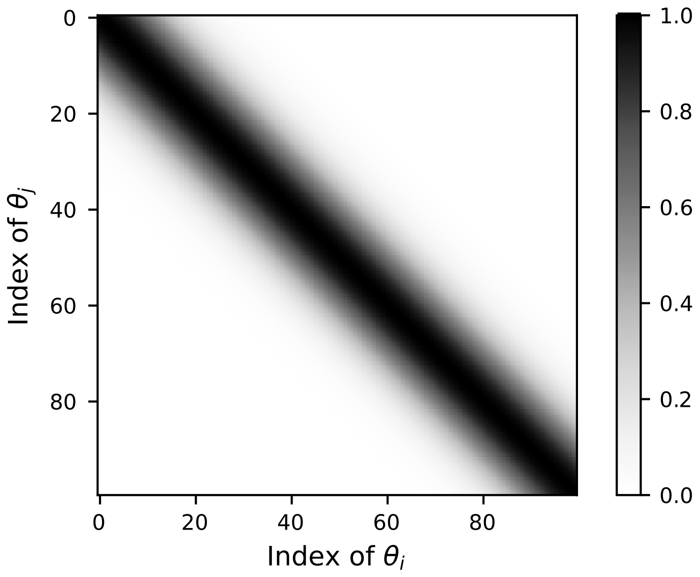

Figure 21: The covariance matrix for the Gaussian process in Figure 20. Note the speed of reduction of covariance away from the diagonal.

3.3 GP Posterior

Now, as Gramacy writes [36], "We're not in the business of generating random functions". For our purposes, GPs are made useful by their analytical treatment of data observations. Given data

$$D_m = (\Theta, \vec{y}), \tag{48}$$

where (for our current one-dimensional GP treatment), $\Theta = \{\theta_1, ..., \theta_m\}$, $\Theta \in \mathbb{R}^{1\times m}$, and $\vec{y} = \{y_1, ..., y_m\}$, $\vec{y} \in \mathbb{R}^m$, we want to know which random functions could have generated such data. That is to say, we want to know the conditional distribution of $Y(\theta)|D_m$. Let us elaborate on the dimensionalities here. So far in this chapter, we have only dealt with a set of one-dimensional random variables $Y \in \mathbb{R}^{1\times m}$. Now, we replace these for observed data $\Theta \in \mathbb{R}^{1\times m}$. Just as before, where each random variable was itself one-dimensional, our data here is, currently, one-dimensional also. We have m such one-dimensional datapoints, and will now look to model these with a GP. We discuss how to extend this to multi-dimensional data in Section 3.4.4. We also treat D_m as noiseless for now, but extend this later, alongside dimensionality.

Let's begin by considering an $(m+1)^{\text{th}}$ evaluation of our GP at the point θ_{m+1}, $Y(\theta_{m+1})$, post fitting our GP to D_m. We seek the conditional distribution $Y(\theta_{m+1})|D_m$. Let us now partition a vector $Y \in \mathbb{R}^{m+1}$ [15] as

$$Y = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}, \tag{49}$$

where $Y_1 \in \mathbb{R}$ and $Y_2 \in \mathbb{R}^m$. The vector Y represents the combination of $Y(\theta_{m+1})$ and $\vec{y}$, and partitioning it like so allows us to use results from Eaton [37] to condition $Y(\theta_{m+1})$ on D_m. We assume the vector Y has a joint MVN distribution $Y \sim \mathcal{N}(\vec{\mu}, \Sigma)$. Partitioning $\vec{\mu}$ as in Equation 49, the mean vector can be re-written as

$$\vec{\mu} = \begin{pmatrix} \vec{\mu}_1 \\ \vec{\mu}_2 \end{pmatrix}, \tag{50}$$

where $\vec{\mu}_1 \in \mathbb{R}$ and $\vec{\mu}_2 \in \mathbb{R}^{1\times m}$. The covariance matrix is then partitioned as

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}, \tag{51}$$

where

$$\Sigma_{11} = \Sigma(\theta_{m+1}, \theta_{m+1}), \Sigma_{11} \in \mathbb{R},$$
$$\Sigma_{12} = \Sigma(\theta_{m+1}, \Theta), \Sigma_{12} \in \mathbb{R}^{1\times m},$$
$$\Sigma_{21} = \Sigma(\Theta, \theta_{m+1}), \Sigma_{21} \in \mathbb{R}^{m\times 1},$$

and

$$\Sigma_{22} = \Sigma(\Theta, \Theta), \Sigma_{22} \in \mathbb{R}^{m\times m}.$$

The distribution of Y_1, conditional on $Y_2 = \vec{y}$, can then be derived as [37]

$$P(Y_1|Y_2) \sim \mathcal{N}(\mu_{Y_1|Y_2}, \sigma_{Y_1|Y_2}), \tag{52}$$

where

$$\mu_{Y_1|Y_2} = \vec{\mu}_1 + \Sigma_{12}\Sigma_{22}^{-1}(\vec{y} - \vec{\mu}_2), \tag{53}$$

with $\mu_{Y_1|Y_2} \in \mathbb{R}$ and

$$\sigma_{Y_1|Y_2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}, \tag{54}$$

[15] Here we momentarily stray from our notation convention for vectors.

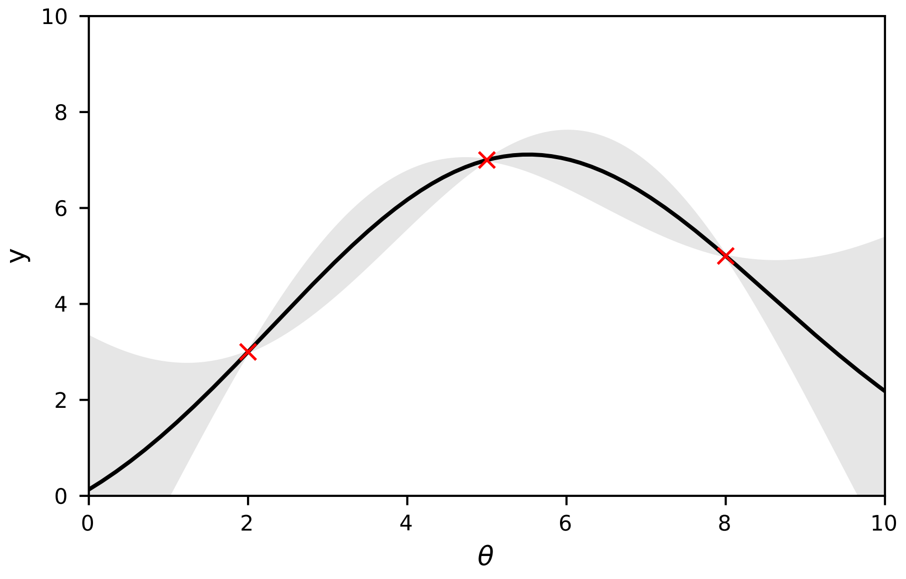

Figure 22: A zero-mean, RBF kernel GP, conditioned on three data points, denoted by red crosses. Note how the conditioned GP interpolates between the data. The kernel hyperparameters have been optimized by maximising the marginal likelihood, which we discuss in Section 3.6. Note that the horizontal axis now represents any (one-dimensional) real value of $\theta \in \Theta$, as we are now dealing a dataset D_m, rather than a set of indices which represent the dimensions of a multivariate Normal distribution.

where $\sigma_{Y_1|Y_2} \in \mathbb{R}$ also. Thus, we gain the mean and variance of Y_1 (the next sample point to evaluate), conditioned upon Y_2 (the previous data $\vec{y}$). Note that, in Equation 54, Σ_{11}, the variance of the next sample point, is reduced upon observation of the data $Y_2 = \vec{y}$[16]. Interestingly also, the amount this variance decreases isn't affected by the values of our data $Y_2 = \vec{y}$.

We can summarise these results by writing

$$Y(\theta_{m+1})|D_m \sim \mathcal{N}(\mu_{Y_1|Y_2}, \sigma_{Y_1|Y_2}). \tag{55}$$

Equations 53 and 54 completely define our posterior conditional distribution, and allow us to predict the value of our GP, conditional on data D_m, at any point θ_{m+1}. A visualisation of such a posterior is provided in Figure 22. Note how the posterior interpolates through the training data D_m, and produces error estimates away from training data. This is the statistical model we shall use to emulate our fitness surface in ABM calibration experiments.

3.4 Kernels

Previously we defined the squared exponential/radial basis function (RBF) kernel in Equation 46. While standard in much of the GPR literature, it is far from the only option. In this section, we detail some common alternatives, and also discuss how to fit the kernel hyperparameters (that will also be introduced).

[16]Also, for clarification, since Σ_{11} is just the variance of the next sample point, it is not just equal to unity.

3.4.1 RBF Kernel

The RBF kernel is commonly parametrised as

$$K(\theta,\theta') = \sigma_{rbf}^2 \exp\left(\frac{-(\theta-\theta')^2}{2l_{rbf}^2}\right), \tag{56}$$

where $l_{rbf} \in \mathbb{R}^+$, and represents a lengthscale parameter in the RBF kernel. Varying it tunes the rate of decay of the covariance, between data points the GPR model interpolates between. The second parameter in Equation 56, $\sigma_{rbf} \in \mathbb{R}^+$, represents a scale parameter, and simply tunes the amplitude of the GPR model.

3.4.2 Parameter Fitting

Now, how do we fit the parameters of the GPR model? Well, a common solution is to utilise the marginal likelihood of the MVN. However, derivations of the marginal likelihood typically use GPs with noise (which is often added in implementations, sometimes artificially, to avoid numerical instability within kernel inversions in the likelihood). We shall discuss the likelihood, and fitting the kernel parameters, once noisy GPs have been discussed.

3.4.3 Other Kernels

We mentioned previously that the squared exponential covariance function is infinitely smooth. It has been argued that this isn't realistic for simulations of real-world systems [38], and thus if the practitioner seeks to control this smoothness, they can look to the Matérn kernel, given by

$$K_{Matrn}(\theta,\theta') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left((\theta-\theta')\sqrt{\frac{2\nu}{l_{Matrn}}}\right)^{\nu}\zeta_\nu\left((\theta-\theta')\sqrt{\frac{2\nu}{l_{Matrn}}}\right), \tag{57}$$

where ζ_ν is a modified Bessel function, ν is a smoothness parameter, $\Gamma(\nu)$ is the factorial function $(= \nu(\nu-1)!)$[17], and l_{Matrn} represents the lengthscale parameter of the Matérn kernel.

We now discuss two more kernel types, which could be of use to us. These are additive and separable kernels. Separable kernels are generally of the form (when applied to the squared exponential covariance function)

$$K_S(\vec{\theta},\vec{\theta}') = \prod_{i=1}^{n} exp\Big(\frac{-||\theta_i-\theta_i'||^2}{2l_i^2}\Big), \tag{58}$$

where n represents the number of dimensions [39]. Note here, that in our entire GP discussion thus far, the input data $\Theta \in \mathbb{R}^{1\times m}$, that is to say that we have m data points, each of which is one-dimensional. Each of these data points represents a sample point on our ABM calibration fitness surface. However, in defining the separable kernel, which takes multi-dimensional input data, we extend our input data to $\Theta \in \mathbb{R}^{n\times m}$. This notation is exactly the same as that in Equation 12, back in Section 2.5. This multi-dimensional input is now also reflected in the use of the vectorised inputs $\vec{\theta},\vec{\theta}'$ to the kernel.

With regards to the separable kernel specifically, note the different length scales in each dimension, via the l_i parameters. This allows us to vary the decay of covariance in each dimension of our GPR surface.

[17]Note, we abuse our notation scheme here, to use the conventional symbol of the factorial function.

Additive kernels are slightly different, and take the form

$$K_A(\vec{\theta}, \vec{\theta}') = \frac{1}{n} \sum_{i=1}^{n} exp\Big(\frac{-||\theta_i - \theta_i'||^2}{2l_i^2}\Big). \tag{59}$$

Note the summation instead of product of exponentials. Additive kernels are said to work better in high dimensions than separable kernels [39]. In both kernels, the sets of vectorised l_i parameters can be calibrated via likelihood inferential schemes, which we shall discuss shortly.

These two kernels implement a more general tool, that is available with many other kernel machines, in that kernels can be added, multiplied and convoluted via tensor products to produce new kernels from old [34]. This begs the question of finding the best kernel for the problem at hand. Schemes to automatically find such kernels are discussed later in the thesis.

3.4.4 Multi-dimensional GPR

Let us briefly discuss this extension of GPs to multiple dimensions, which has arisen from the definitions of the separable and additive kernels above. It is actually rather simple to extend GPs to multiple dimensions. We still define a GP as we did in Definition 3.1, which is then discretised over some index set Θ, but the covariance function now changes, to define the covariance in each dimension, as done in Equations 58 and 59. The covariance matrix now simply defines the covariance between points in a higher-dimensional space, via the Euclidean distance. The additive kernel decomposes this covariance, such that it only depends on each individual dimension, while the separable kernel defines a covariance dependant on all dimensions.

3.5 Noisy GPs

Up to this point, we've assumed the value of each data point is noiseless. While this may be the case for some simple ABMs, and other simple simulatory experiments, it most likely isn't so for the high-dimensional, expensive-to-evaluate simulations that we're interested in. In addition, as mentioned before, the covariance matrix inversions [18] required in covariance matrix parameter tuning can suffer from numerical instabilities in deterministic situations, and so some small noise, or 'jitter' is often added, even in deterministic settings. So let's now discuss how noise is incorporated into GPs.

Quite simply, by adding a noise term to the diagonal of our covariance matrix, resultant from our kernel $K(\theta, \theta')$, we achieve the desired breaking of interpolation. The kernel that achieves this takes the general form

$$K(\theta, \theta') = K_{noiseless}(\theta, \theta') + \sigma_y^2 \delta_{\theta\theta'}, \tag{60}$$

where $\delta_{\theta\theta'}$ is the Kronecker delta, equalling one if $\theta = \theta'$, and otherwise zero. Such a term arises from assuming the noise affects the underlying function via

$$y = f(\theta) + \varepsilon, \tag{61}$$

where $\varepsilon \sim \mathcal{N}(0, \sigma_y^2)$.

[18] We reach an issue of convention here now, in referring to such a matrix inversion as the covariance matrix inversion. Typically, such inversions are referred to as kernel inversions, but this contradicts our use of of the kernel $K(\theta, \theta')$ as an analogue of the covariance function. Thus, if the covariance matrix inversion is referred to as a kernel inversion, we hope the reader may excuse this.

The posterior conditional equations, previously defined in Equations 53 and 54, can now be re-written as

$$\mu_{Y_1|Y_2} = \Sigma_{12}\Big[\Sigma_{22} + \sigma_y^2 I\Big]^{-1}\vec{y} \tag{62}$$

and

$$\Sigma_{Y_1|Y_2} = \Sigma_{11} - \Sigma_{12}\Big[\Sigma_{22} + \sigma_y^2 I\Big]^{-1}\Sigma_{21}. \tag{63}$$

Note that the covariance matrix Σ_{22} ($= \Sigma(\Theta, \Theta)$), the covariance matrix of the training data, has simply had a diagonal noise term $\sigma_y^2 I$ added.

3.6 Marginal Likelihood

Now, as mentioned previously, let us provide a brief overview of the marginal likelihood here. This likelihood is obtained by marginalising over the function values f, via

$$P(\vec{y}|\Theta) = \int P(\vec{y}|f, \Theta)P(f|\Theta)df. \tag{64}$$

The prior, $P(f|\Theta) = \mathcal{N}(0, \Sigma)$, which is a zero-centred MVN with the covariance matrix Σ (i.e. the GP prior defined in Equation 44). The likelihood is the probability of observing a set of i.i.d points about the model value f (as defined by our noise model in Equation 61. This adopts the value

$$P(\vec{y}|f, \Theta) = \prod_i \mathcal{N}(y_i|f_i, \sigma_y^2). \tag{65}$$

Thus, via laws of products of Gaussians, and after taking the resultant Gaussian, we arrive at the marginal likelihood, or rather, the log marginal likelihood,

$$\log P(\vec{y}|\Theta) = \frac{1}{2}\vec{y}^\top(\Sigma + \sigma_y^2 I)^{-1}\vec{y} - \frac{1}{2}log|\Sigma + \sigma_y^2 I| - \frac{n}{2}log 2\pi, \tag{66}$$

where $|\cdot|$ represents the determinant of the matrix within. This equation is of use to us, because it is the loss function by which we will tune our kernel's hyperparameters. By taking the derivative of Equation 66 with respect to each parameter in the covariance matrix Σ, and maximising said likelihood, we perform maximum likelihood estimation for our covariance matrix's hyperparameters.

Let us briefly now discuss the terms of the log marginal likelihood. Discarding the constant, we see that we have a term involving the inverse of our covariance matrix, which we will call the data fit term, and a term involving the determinant of our covariance matrix, which we will call the model complexity/capacity control term. Consider an RBF kernel, with lengthscale l and fixed scale σ. As l decreases, the fit of the model improves (as it gets closer to interpolation), and so the data fit term decreases. However, the model complexity term will increase, as the short length scale leads to a higher determinant of the resultant covariance matrix. For long length scales, the opposite is true. We get a poorer data fit, and thus a larger data fit term, but the model is less complex, so we get a smaller model complexity term. Thus, the log marginal likelihood allows us to tune the covariance matrix hyperparameters, while controlling for the complexity and fit of the GPR model.

With the log-likelihood and derivatives with respect to each parameter in hand (see [34] for these), we can use any standard gradient-based optimizer to tune our covariance matrix hyperparameters. Local minima can be an issue, and the selected optimizer should attempt to deal with this.

3.7 Challenges

The primary bottleneck in GPR are the kernel inversions[19], required in the posterior mean and covariance (in Equations 53, 54, 62 and 63), and in the log marginal likelihood in Equation 66.

The covariance matrix has dimensions $m \times m$, where m is the number of sample points executed. Such kernel inversions scale as $\mathcal{O}(m^3)$, and so m is limited to the low thousands. This isn't a disaster for ABM calibration experiments in itself, as such a number of executions can be towards the higher end of that available for expensive simulators. We discuss kernel approximation schemes later in the thesis, which will be able to reduce this $\mathcal{O}(m^3)$ scaling somewhat.

Another challenge is of high input dimensionality, and training GPs on such data. This is of particular interest to us, as we are looking to emulate the fitness surfaces of high-dimensional ABM calibration experiments. GPs are said to struggle to produce accurate fits beyond 15-20 dimensions, and we discuss how this can be tackled later, with assumptions on the structure of the parameter space, such as additivity or lower-dimensional effective subspaces. For now though, we move on to discussing Bayesian optimisation, which combines GPs with sequential sampling procedures, often making use of the analytical variance estimates we've now gained.

3.8 Bayesian Optimisation - Motivation

As machine learning's applicability to other fields continues to grow, the complexity of the models used is growing with it. Deep neural networks (DNNs) are a quintessential example. They build upon their predecessor by expanding the architecture of neural networks, and thus increasing the number of hyperparameters that need to be tuned within the DNN. But the evaluation of such DNNs is often expensive, whether this be in time or money. Bayesian optimisation (BO) has emerged as a common solution to this. By pairing GPs with a sequential sampling procedure, which efficiently uses the information gained from each sample point, hyperparameter optimisation problems can be approached methodically and efficiently.

As hinted at in the previous motivational sections, machine learning isn't the only field which has benefitted from BO. Many of the fields we discussed at the start of this chapter, couple GPR with sequential sampling schemes also, to create BO loops. One notable area we haven't mentioned yet, is that of direct User Interaction (UI) experiments in business. A/B tests seek to tune online shopping experiences, with expensive (in time) trails of online shop layouts [40]. Gaming companies seek to optimise player experiences of games by tuning parameters such as difficulty to maximise engagement [41]. Computer animators even seek to optimise fluid animation systems, which often have hyperparameters the animator is tasked with manually tuning, in pursuit of fluid dynamics that appear realistic [42]. Such simulators again, are often expensive in time. These problems have many similarities, between themselves and to our ABM calibration problem, and so it perhaps shouldn't be a surprise that BO can be useful in all of them.

3.9 Acquisition Functions

As mentioned previously, we've now seen the first step in most BO algorithms: the GP. We also saw, in Section 2.5, that we defined such sequential schemes H_{seq} in Equation 27. The MCMC algorithm, outlined in Algorithms 3 and 4, was our first example of such a H_{seq}, and in this section, we outline the sequential sampling scheme for BO, which uses

[19]This is an example of the underlying inversion being of the covariance matrix of course, but convention dictates that referring to is as the kernel inversion is desirable.

the variance estimates of GPR to guide its sequential selection. Such schemes are referred to as acquisition functions in the BO literature.

3.9.1 Expected Improvement

One popular acquisition function is the Expected Improvement (EI) [43], which, as the name suggests, maximises the expected value in the improvement of the minimum value found, y^*. We define a new proposal sample point, with $Y(\theta_{m+1})$ (as done previously), and the current best sample point with y^*. The EI is then defined as

$$EI_m(\theta_{m+1})) = \mathbb{E}_m \|y(\theta_{m+1}) - y^*\|, \tag{67}$$

where $\mathbb{E}_m$ denotes the expectation given by the posterior distribution of the GP, under m previously observed data points, D_m. In other words, the random variable $y(\theta_{m+1})$ arises from evaluating the GP, $Y(\theta_{m+1})|D_m$, at θ_{m+1}. The term $y(\theta_{m+1})$ in Equation 67 is itself a random variable (as per Definition 3.1 and Equation 40), the mean and variance of which, at any point θ_{m+1}, are provided in Equations 53 and 54 (in the noiseless setting). Intuitively, by modelling $y(\theta_{m+1})$ as a random variable, we see that there is some probability with which this next sample point has a lower value than the current minimum y^*. We want to know, in expectation, this improvement $\|y(\theta_{m+1}) - y^*\|$. We fulfil this request with some extended integration by parts (see [44] for details), resulting in the following expression:

$$EI(\theta_{m+1}) = \Delta(\theta_{m+1})\Phi(\frac{\Delta(\theta_{m+1})}{\sigma(\theta_{m+1})}) + \sigma(\theta_{m+1})\phi(\frac{\Delta(\theta_{m+1})}{\sigma(\theta_{m+1})}), \tag{68}$$

where $\Delta(\theta_{m+1}) = (\mu(\theta_{m+1}) - y^*)$, and $\mu(\theta_{m+1})$ represents the mean of the GP at θ_{m+1} (as provided by Equation 53 in the noiseless setting). The variance of the random variable $y(\theta_{m+1})$ is denoted by $\sigma(\theta_{m+1})$ (and again, is given by Equation 54 in the noiseless setting). The probability density function and cumulative density function of the standard normal distribution enter the analytical form for EI, and are denoted by $\phi(\cdot)$ and $\Phi(\cdot)$ respectively.

We then search for the point of maximum EI,

$$\theta_{m+1} = \arg\max_{\theta \in \Theta} EI(\theta) \tag{69}$$

via a global optimisation routine, such as L-BFGS, to search for this maxima. We present an illustration of this scheme in Figure 23. Note how the EI criterion is able to guide our sequential selection of sample points towards the minimum of the underlying function.

Now, a very valid question emerges from this. We've constructed a statistical model (the GP) and a sequential sampling scheme using the GP, to optimise some expensive simulator, according a selected fitness function. But now, we have to perform some global optimisation anyway, over the same parameter space as before. So, what have we gained? In short, evaluating the acquisition function is cheap, relative to evaluating the expensive simulator/ABM. We can afford to evaluate it many times, because of the analytical form of the GP that's still available to us. Running such an optimisation routine yields our next parameter set, θ_{m+1}, and so completes our BO loop.

Now, in order to provide the BO loop algorithmically, we must make some subtle notational adjustments. Throughout the entire analysis above, we've referred to the number of sample points via m, and the next sample point to evaluate via $m+1$. This would indicate that once m sample points have been executed, and their fitnesses calculated, we would seek to commence our BO algorithm at the $(m+1)^{\text{th}}$ sample point. In actuality, we need to take to take some set of sample points $m_{init} < m$, execute them via some

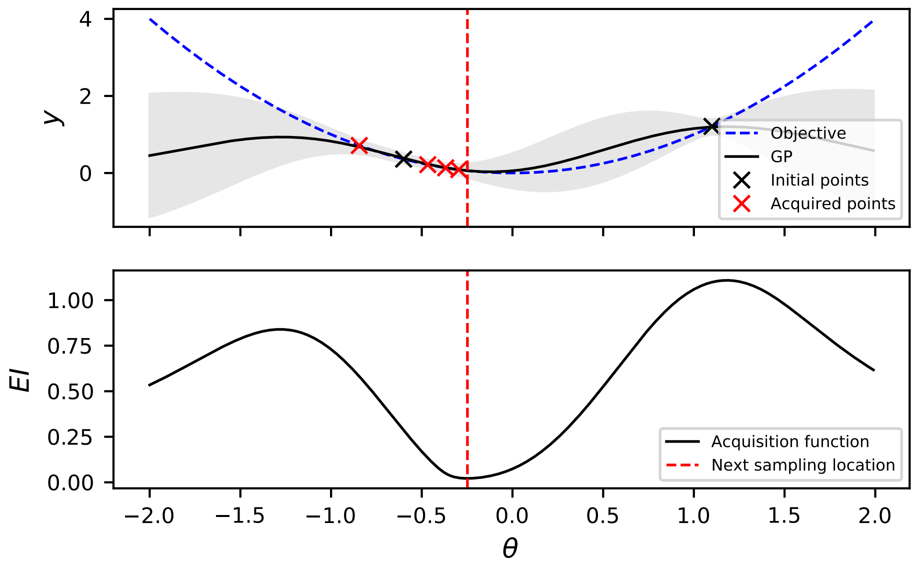

Figure 23: Two sample points are initially evaluated (and are denoted with black crosses). The Expected Improvement acquisition function is then evaluated five times, and is plotted in the bottom subplot for the last iteration. The GP is fit at each iteration. Note how the acquisition function guides our sample point selection, and drives it towards the minimum of the underlying quadratic function. The vertical red line in the bottom figure denotes the minimum of the acquisition function, and thus it denotes the next point to be executed in the top figure.

one-shot sampling scheme H_{os} (random search for our purposes), and then begin our BO scheme, to select the $(m_{init}+1)^{\text{th}}$ sample point. This is expressed algorithmically below. In addition to this, we must discuss how the fitnesses vector $\vec{y} \in \mathbb{R}^m$, and the set of parameter vectors $\Theta \in \mathbb{R}^{n\times m}$ are adjusted, in light of this. Given our requirement to initialise the BO scheme with m_{init} points, we may either denote analogous variables $\vec{y}_{init} \in \mathbb{R}^{m_{init}}$ and $\Theta \in \mathbb{R}^{n\times m_{init}}$, or initialise $\vec{y} \in \mathbb{R}^{m_{init}}$ and $\Theta \in \mathbb{R}^{n\times m_{init}}$, and append a single fitness ($y \in \mathbb{R}$) and sample point ($\vec{\theta} \in \mathbb{R}^n$) to each respectively, at each step of the BO loop. We opt for the latter, and such a BO algorithm is expressed in Algorithm 6.

```
1: Initialise $\vec{y}, \varepsilon$                                              ▷ $\vec{y} \in \mathbb{R}^{m_{init}}, \varepsilon \in \mathbb{Z}+$
2: Initialise $\Theta$ via random search                                   ▷ $\Theta \in \mathbb{R}^{m_{init}}$, Alg. 1(1), Alg. 2(2)
3: for $i$ in $m_{init}$ do
4:     Calculate $y_i = f(A(\vec{\theta},\varepsilon), A(\vec{\theta}_{emp},\varepsilon))\ \forall\ \vec{\theta} \in \Theta$       ▷ $\vec{y} = \{y_1, ..., y_{m_{init}}\}$
5: end for
6: Data $D = \{\Theta, \vec{y}\}$
7: for $i$ in $m - m_{init}$ do
8:     $Y(\vec{\theta}_i)|D \sim \mathcal{N}(\mu(\vec{\theta}_i), \Sigma(\vec{\theta}_i))$                           ▷ Generate GP
9:     $\vec{\theta}_{i+1} = \underset{\vec{\theta}\in[\vec{\theta}_{lower},\vec{\theta}_{upper}]}{\arg\max}\ EI(\vec{\theta})$
10:    Execute $\vec{\theta}_{i+1}$, return $y_{i+1}$                              ▷ Alg.'s 1(10-19), 2(11-25)
11:    Append $\theta_{i+1}$ to $\Theta$                                        ▷ $\Theta \in \mathbb{R}^{n\times(m_{init}+i)}$
12:    Append $y_{i+1}$ to $\vec{y}$                                         ▷ $\vec{y} \in \mathbb{R}^{(m_{init}+i)}$
13:
14: end for
15: Return $y^* = min(\vec{y})$
```

Algorithm 6: Bayesian optimisation. We denote the steps of the frequentist and Bayesian calibration algorithms to execute, at the relevant step of the Bayesian optimisation algorithm. Note Step 9, where $[\vec{\theta}_{lower}, \vec{\theta}_{upper}]$ denotes the bounds of the parameter vector $\vec{\theta}$, and thus we search in this space.

We now have everything we need to begin emulation of the fitness surfaces in our calibration experiments. Later in this chapter, we illustrate a simple BO implementation on the Franke-Westerhoff model, and show where BO begins to fail, in its current form. But before this, as we've added our sequential sampling scheme now, let's discuss some of its pitfalls, just as we did with GPR.

3.9.2 Challenges with BO

We've already discussed the challenges with GPR in Section 3.7. These obviously are problems for BO also. But the acquisition function has its own issue, which emerges in high-dimensional input settings. In short, it's difficult to optimise acquisition functions over many dimensions. The volume of the space grows exponentially, and we need exponentially more evaluations to sufficiently explore the n-dimensional acquisition function (again, with n being the input dimensionality). This mirrors GPs apparent struggles to produce accurate fits in high dimensions. Both can be dealt with by assuming a lower-dimensional structure to the parameter space, but we leave this till much later.

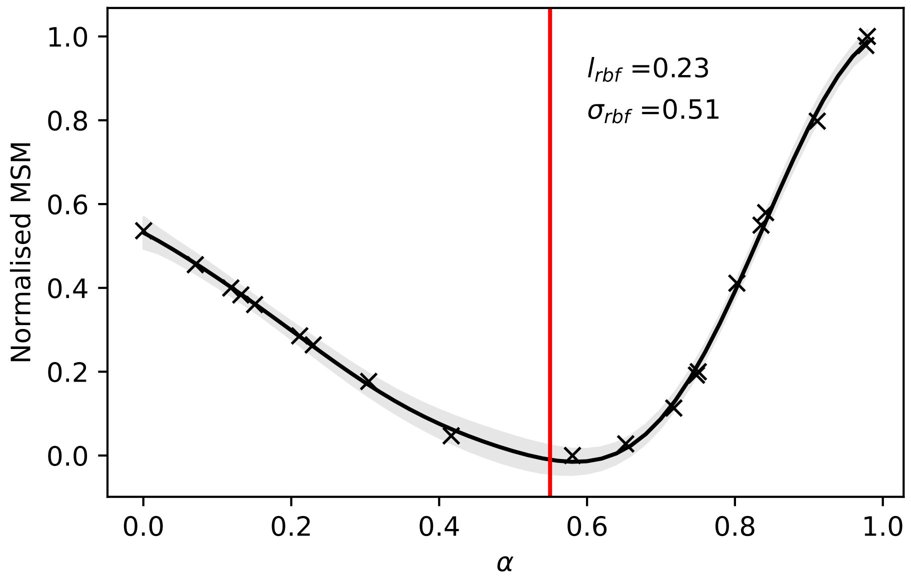

Figure 24: A zero-mean, RBF kernel GP, fit to the AR(1) fitness surface, resultant from a frequentist calibration experiment analogous to that undertaken for Figure 3. Note the interpolation of the sample points, and variance estimates in regions with no sample points. The kernel hyperparameters have been optimized by maximising the log marginal likelihood. The lengths of the pseudo-true and model-generated time-series were set to $T = T_{emp} = 100$, and the number of sample points $m = 20$. The number of repeats $r = 3$.

3.10 Numerical Experiments - Emulated Frequentist Calibration

3.10.1 AR(1)

Previously in Figure 3, we showed that the one-dimensional MSM surface, for the internal frequentist calibration experiment on the AR(1) model, has a minima about our selected 'ground-truth' parameter, $\vec{\theta}_{emp} = \alpha_{emp}$. As an illustrative exercise of applying GPs to emulate such surfaces, let us take a number of samples, $m = 20$, from this experiment, and fit a GP to the selected parameter sets and their fitnesses, $\Theta_m \in \mathbb{R}^{1\times m}$, $\vec{y}_m \in \mathbb{R}^m$. The resulting GP is visualised in Figure 24. Note how, even with a modest number of samples (20), we're able to gain a GP model which emulates the MSM surface with modest variance estimates in most regions of the parameter space. A zero-mean, RBF kernel GP was used in Figure 24, with the lengthscale and scale parameters, l_{rbf} and σ_{rbf} respectively, optimized via the 'L-BFGS-B' algorithm, which minimizes the log-likelihood outlined in Equation 66. We used the `sklearn.gaussian_process` module to produce these simple GP implementations [45]. As mentioned before, the log marginal likelihood of the GP balances model complexity against data fit, in its kernel parameter estimation.

Note that the noise is not estimated in this specific GP implementation. That is to say, σ_y^2 in Equation 60 is not estimated, but rather set beforehand. This is not the case in the GPs used in the remainder of this thesis, which do have such noise parameters estimated in their GP implementations. We set $\sigma_y^2 = 0.03$ in this GP implementation. We also repeat each sample point three times, in an attempt to mitigate the effects of noise about each sample point, as discussed previously. If we didn't do this, then using a larger value of σ_y^2 would be justified.

Also note, that via a random one-shot sampling scheme, we managed to locate the minimum of the MSM fitness surface with relative ease. In Figure 3, the sample point with the minimum MSM value represents our best parameter value $\vec{\theta}^*(= \alpha^*)$, and our best fitness value y^*. In higher dimensions, finding a minimum with such a low MSM value won't be as easy, particularly with limited evaluations. It is precisely this that motivates us to emulate with GPs, and sequentially search with BO. Note, that going forward, we should aim to study not only the value of y^*. but the values within $\vec{\theta}^*$ also.

3.10.2 Franke-Westerhoff model

Previously, we only modelled the MSM surface with a GP, and visually inspected the quality of the fit. We saw that the GP was able to model the fitness surface well. Let us extend this now, by applying sequential sampling, via Bayesian optimisation, to our GP fit of the MSM surface. We begin by applying the BO algorithm in Algorithm 6 to the Franke-Westerhoff model, first allowing just one parameter to vary, and then allowing all to vary.

We use the `BoTorch` package to implement all of our BO experiments [46] [20]. It is a recent addition to the set of BO packages available, but was regarded to be a useable, modern and stable choice, which is low-level enough (in the programming language sense) to allow high customisation, but high-level enough to abstract away many complexities of GP and BO implementation. The high-dimensional BO method we use later in this thesis is also built upon it, which was a further indicator of BoTorch's quality.

We should also note that the acquisition function optimisation possesses some hyperparameters of its own, in BoTorch. The underlying optimiser requires two quantities, `num_restarts` and `raw_samples`, which, as the name denotes, determines how many random restarts, and samples are used in the acquisition function optimisation of BoTorch. Two default values are 25 and 64 respectively, and so these values are used throughout all experiments using BoTorch in this thesis.

We use the `SingleTaskGP` model from `BoTorch`, with the `SingleTaskGP`'s default kernel, a separable Matérn kernel, with its smoothness parameter ν (as in Equation 57) equal to 2.5. This kernel is used in all our base BO implementations for the rest of the thesis. As we shall discuss in Chapter 5, for the sparsity of data that we have, we believe that such a base kernel selection is of less importance than the form of the kernel (separable, additive etc).

Let us now detail our emulated frequentist calibration experiment, upon the Franke-Westerhoff model. Over multiple runs of Algorithm 6, we plot y^* against the number of executions, for the α_n parameter in the Franke-Westerhoff model, noting that we are now using the Expected Improvement acquisition function to select our next sample point at each step of the BO algorithm. We plot curves of reducing y^* in Figure 25-a, for BO and random search, as the number of sample points increases. Note that, just as we saw for the AR(1) model, we quickly retrieve very low values of y^* via BO and random search, indicating that we've located the minimum of the one-dimensional MSM surface. However, the y^* curve for BO is already steeper than that of random search, in this one-dimensional space. This indicates that BO is able to find the minimum quicker, and the lower asymptotic value indicates that a higher quality minimum is found also.

Additionally, we track the values of $\vec{\theta}^*$ (just consisting of α_n^* currently) over the repeated BO experiments, and plot the sample paths in Figure 25-b, alongside the sample points executed in each repetition of the BO experiment. Note how the best parameter settings over the runs of the experiment are close to the ground-truth, plotted in red, with the sequential regime denoted between the blue dashed lines. We can also see the degree of exploration, by examining the distribution of sample points in Figure 25-b (denoted

[20]Available at https://github.com/pytorch/botorch.

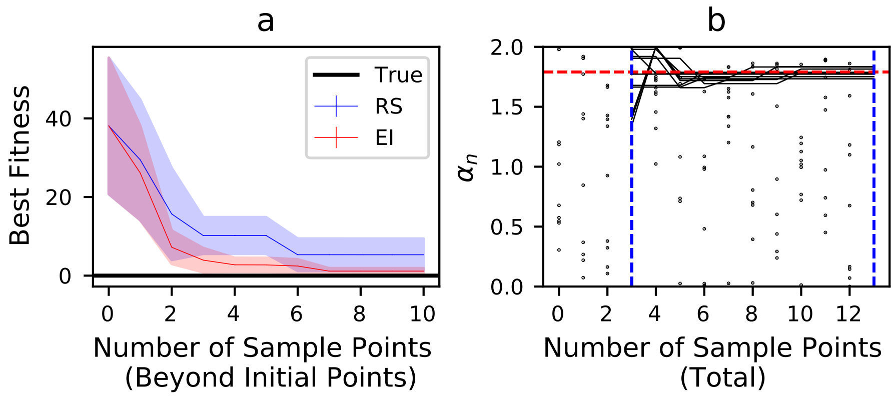

Figure 25: Results from an emulated one-dimensional frequentist calibration experiment, on the Franke-Westerhoff model. We only allow α_n to vary. The experiment was run 10 times, with three initial sample points, and 10 sequential points. Each sample point was repeated three times. All time-series were of length 100. The mean and standard error of the minima are plotted. Note the reduction in the mean minimum fitness obtained, over the 10 runs of the experiment. Also note, in comparison to random search, the faster convergence to the fitness of the true parameter vector (zero), denoted in black. We also plot the values of the best sample point after each sequential step in subplot (b), for each of the 10 runs. That is, at each sequential step, the parameter value of the sample point with lowest fitness, for each run. We also plot (with dots) the actual sample point executed at each sequential step. The dashed blue lines denotes the start and end of the sequential sampling period.

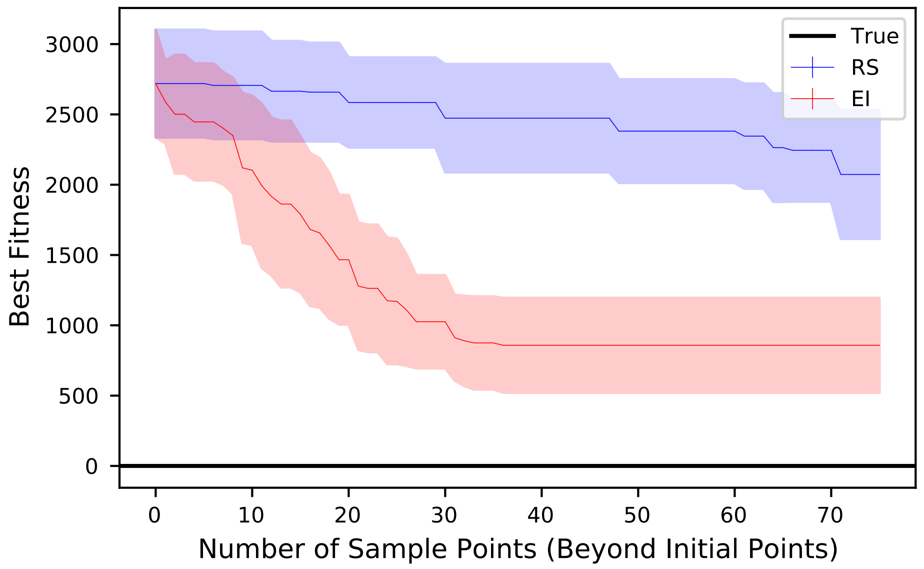

Figure 26: Results from our emulated frequentist calibration experiment, on the full-dimensionality of the Franke-Westerhoff model (minus μ). All parameters were allowed to vary. 25 initial sample points were used, and 75 subsequent sequential and random sample points were executed, for BO and random search respectively. The experiment was repeated 10 times, and the mean and standard error of the minimum fitness at each number of sample points is plotted. Each sample point was repeated three times.

by each dot plotted). The BO scheme continues to search for better minima, even when it has located one of (relative) quality.

Now, let's observe such y^* curves for the full dimensions in the Franke-Westerhoff model[21]. BO's performance on the full-dimensional, internal, frequentist calibration problem is compared to random search in Figure 26. Again, we see that BO is able to eventually locate better minima than random search, for the same number of sample points. Sample paths as in Figure 25 for the full-dimensionality of the Franke-Westerhoff model are provided in Appendix B. We observe mixed success in retrieving the true parameters. The performance of the BO algorithm has certainly degraded from the one-dimensional Franke-Westerhoff calibration, as to be expected. However, such qualitive conclusions from the figure in Appendix B are not required, as we can calculate the loss function of Equation 28 for the best sample point in each run of the algorithm, along with a standard error for such multiple runs. Such loss values are presented in Table 6, and we discuss these values once all experiments have been detailed, at the end of this chapter.

However, we can see that there is still room for improvement in the minimum MSM values reached. BO has improved upon the performance of random search, but can we do better? We explore this later in the thesis. For now, let us replicate these experiments on the Brock-Hommes model.

3.10.3 Brock-Hommes model

For the Brock-Hommes model, we omit a one-dimensional emulation of the analogous frequentist calibration experiment, for the sake of brevity, and skip to the results of a

[21]Minus the previously discussed μ parameter, the extreme sensitivity of which would have dominated this experiment.

full-dimensional experiment, as seen in Figure 27.

There is one thing that we must discuss however, and that is the Brock-Hommes model's instability at certain regions of the previously defined parameter space, which was realised post the illustrative experiments of Chapter 2. The selected output time-series (generated by Equation 9 in Section 2.3.3) can often possess numerical errors, even after dealing with the numerical underflow and overflow problems that could potentially arise from the Brock-Hommes model's analytical form. We suspect this is due to regions of the parameter space being explored, that shouldn't be so (as Figure 8 may hint at). In allowing all 11 parameters to vary, we are most likely varying parameters that were intended to have values about a narrow range, or at least remain fixed. This is supported by previous literature's use of free parameter sets for the Brock-Hommes model [11], as summarised in Table 2.

To deal with this, two amendments were made. First, any fitness values that were reported as a numerical error were cast to 10^3. From running experiments and tracking the fitnesses that were being output, this value was selected to minimise the chance of any successfully calculated fitnesses being larger than it, and for it not to be significantly larger than the largest successfully calculated fitness. In this way, our BO scheme wouldn't be led towards undesired regions of the parameter space, but it also wouldn't disproportionately weight certain regions due to a disproportionately large 'punishment' for a numerical error.

Second, to limit the effect of any possible non-Gaussian fitness distributions about any sample point, we began using the median of the sample point outputs, rather than the mean, solely for all experiments going forward which use the Brock-Hommes model. While this contradicts the theory previously, it was thought to be suitable to avoid outliers disproportionately affecting our final fitness value at each sample point. This also serves as an explanation for the noisy one-dimensional fitness surfaces observed in Figure 8.

The results of the subsequent experiment are presented in Figure 27. We observe the same trends as for the Franke-Westerhoff model's full dimensional experiment above. BO outperforms random search, and locates sample points of lower fitness (lower MSM value).

We should also examine the loss however, of the best sample points at the end of each iteration of such an experiment. The mean of these values, along with the mean's standard error, is again provided later. The loss values for random search were not calculated, as it was believed that the consistently better performance of BO made such calculations obsolete. In hindsight, these may have served to act as a benchmark for measuring the loss reduction between random search and BO. Nevertheless, this is a minor point, as we shall still compare BO's loss values, with the frequentist calibration experiment, against that of the Bayesian calibration experiment, using MCMC and emulation.

3.11 Numerical Experiments - Bayesian Calibration

Now, that we've shown BO's effectiveness on our frequentist calibration experiments, let's show how it performs on our Bayesian calibration (BC) experiments. Previously, we discussed how BC, when conducted via MCMC, was very expensive in the number of samples used. We showed how the scheme performed well for the AR(1) model, but began to exhibit poor performance for the Franke-Westerhoff and Brock-Hommes models at their full dimensionalities [22]. We calculated the loss values for repeated BC experiments upon each model, as seen in Table 6. In this section, we would like to emulate the posterior surfaces of these experiments, and aim to match, if not improve upon, the loss values that the BC experiment incurred upon the Franke-Westerhoff and Brock-Hommes models. To

[22] Apart from the μ parameter in the Franke-Westerhoff model of course, as discussed previously.

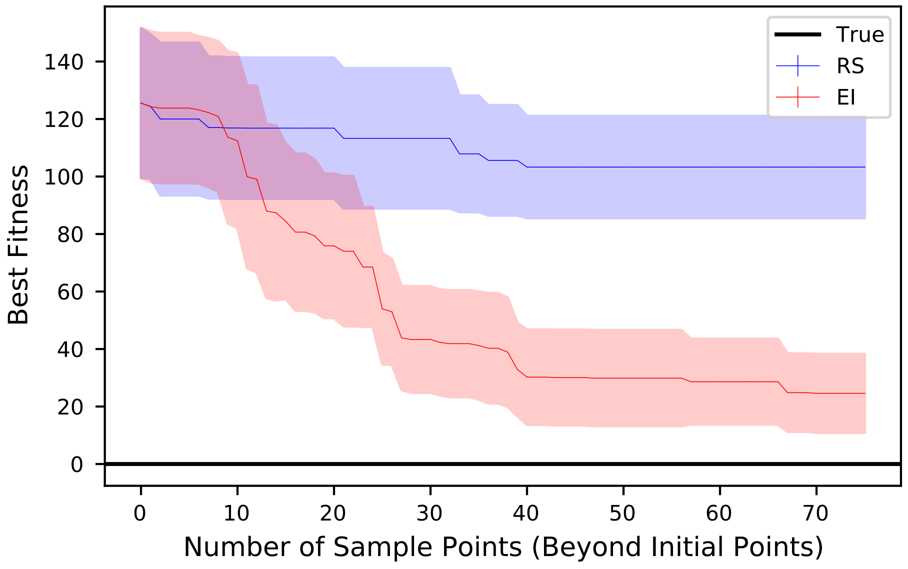

Figure 27: Results from our emulated frequentist calibration experiment, on the full-dimensionality of the Brock-Hommes model. All parameters were allowed to vary. 25 initial sample points were used, and 75 subsequent sequential and random sample points were executed, for BO and random search respectively. The experiment was repeated 10 times, and the mean and standard error of the minimum fitness at each number of sample points is plotted. Each sample point was repeated three times.

illustrate such an emulation, we present the emulated BC experiment upon the AR(1) model now.

3.11.1 AR(1) Model

Our first point of call is to construct our fitness surface, over which we shall sequentially sample, with the goal of reaching the minimum of said surface. Observe, that in Figure 12-c, we plotted the value of the posterior at each value of α used in the MCMC chain of Figure 12-b. We hypothesised that this surface could be emulated, and thus a much more sample-efficient calibration scheme devised. Given that we seek to minimise in our BO implementation, we need to transform this posterior, which has a maximum at the true value of α. We do this by simply taking the negative of the posterior expressed in Equation 24 [23].

Next, we run our BO algorithm, just as in the frequentist calibration experiments, alongside random search. We can then visualise the minima curves over multiple runs, alongside their standard error, and observe which method achieves a better minimisation. However, this question of a ‘better’ minimisation is a difficult one, for emulation upon BC. Previously, in frequentist calibration, our MSM distance function had a minimum of zero about the true parameter vector $\vec{\theta}_{emp}$. Here, we aren’t so lucky. Recall how for the BC experiment, we are required to bin multiple runs of our model-generated time-series $\vec{x}$, at some parameter vector $\vec{\theta}$. We then generated a single, pseudo-true time-series using

[23]In implementation, we take the negative of the sum of logarithms, of the likelihood outlined in Equation 26. This is done to avoid numerical underflow/overflow, and is a common feature of likelihood schemes with products. This is because the repeated multiplication of large or small numbers increases the probability of numerical instability.

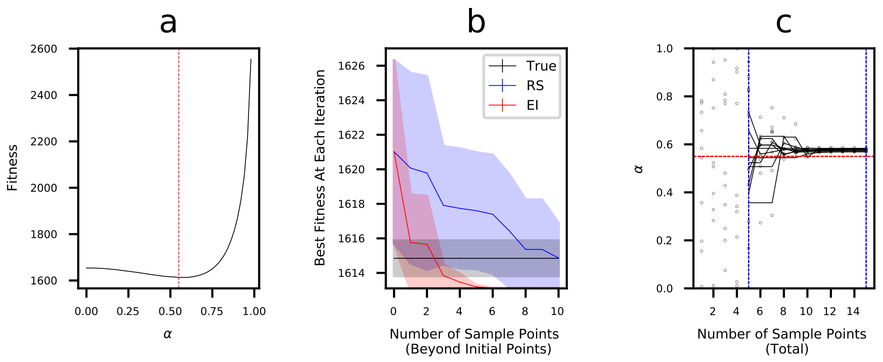

Figure 28: Our emulated Bayesian calibration experiment, upon the AR(1) model. The experiment was run 10 times, with two initial sample points, followed by 10 sequential points. In subplot (a), we plot the transformed posterior, calculated at 100 points on the one-dimensional parameter space of α. Note the high similarity to Figure 3. In subplot (b), we present our emulated Bayesian calibration experiment, using Bayesian optimisation and random search. Note how Bayesian optimisation, once again, locates minima quicker than random search. However, also note the fitness of the true parameter value ($\alpha = 0.55$) in black, with its standard error. Sample paths for the experiment are presented in subplot (c). Note the ability of our emulated Bayesian calibration scheme to recover the true parameter value. The sequential period is denoted in blue in (c), with the true parameter value in red, in (a) and (c).

our 'ground-truth' parameter vector, $\vec{\theta}_{emp}$, and calculated the probability of each of the pseudo-empirical data points within $\vec{x}_{emp}$, upon the PDF of our model-generated data (created via histogram-smoothing, itself via KDE).

Such a likelihood does not possess a minimum at zero. Instead we can only hope our likelihood is suitably constructed, such that it has a maximum when we set our executed parameter vector $\vec{\theta}$ equal to $\vec{\theta}_{emp}$ (or equivalently that is has a minimum at such a location for the negative likelihood). We see, in Figure 28-a, that this is almost the case, but not quite, for the AR(1) model. The minimum appears to be at a value of α which is slightly larger than α_{emp}.

Thus, when our BO scheme is employed, and it seeks to find the lowest value upon the fitness surface, it is likely to retrieve a value α slightly larger than 0.55. This can be seen in fact, in Figures 28-b and 28-c. In the former, we plot the lowest likelihood found at each iteration of the BO loop, by BO and random search. We also plot the likelihood of the true parameter vector $\vec{\theta}_{emp}$, whereby we set $\alpha = 0.55$, and calculate its likelihood multiple times, before returning the mean and standard deviation of this likelihood in black, in 28-b. Note how BO minimises past the true parameter value, and ends up at a point of lower fitness, but not at a point which is closer to the true value of α. [24]

This is better seen in Figure 28-c, where we plot the actual values of α selected by the BO algorithm, over the five iterations used to create 28-b. The dashed blue lines denote the start and end of the sequential sampling regime. Note the random α selections preceding the blue, sequential regime. We also plot the trajectories of the points which returned the lowest fitness, via the black lines, over the five runs. Note how the trajectories tend to values slightly higher than 0.55.

However, excusing the preliminary nature of the likelihood in Equation 26, this ex-

[24]Note the difference in fitness scales between Figure 28-a and 28-b.

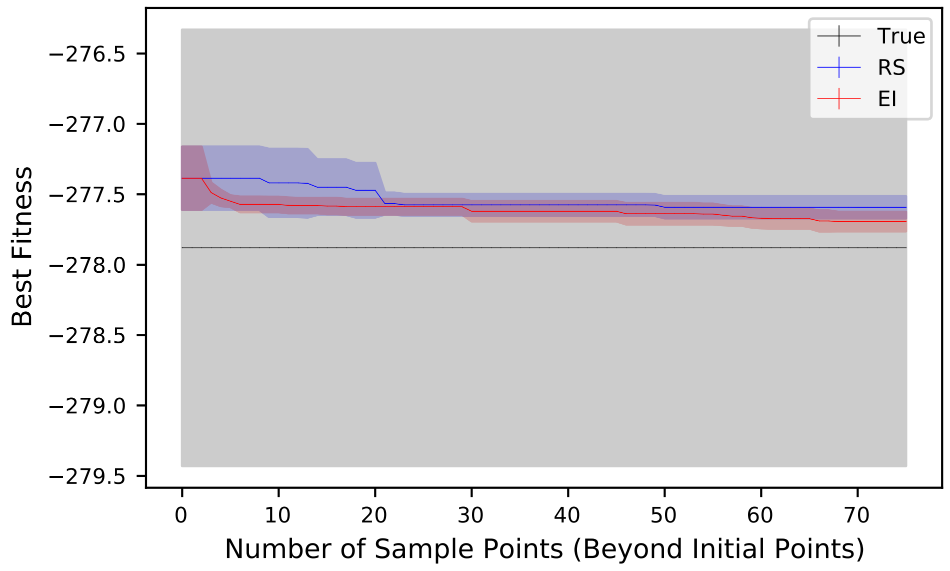

Figure 29: Our emulated Bayesian calibration scheme, upon the full dimensionality of the Franke-Westerhoff model (minus μ). 25 initial sample points were used, followed by 75 sequentially. The experiment was repeated 10 times, and each sample point was repeated 10 times also, within each experiment. The lengths of all time-series were 100. Note the fitness of the true parameter vector, denoted in black, as well as it's variance, denoted by the shaded region. Note Bayesian optimisation's ability to optimise marginally quicker than random search again.

periment shows how we may emulate the previous Bayesian calibration experiment. In this one-dimensional problem, BO was able to retrieve suitable values of α, for a very low number of executions. But of course, random search also manages this.

We now want to see how such an experiment behaves upon a model of higher dimensionality. We shall present analogous visualisations of Figure 28-b for the Franke-Westerhoff and Brock-Hommes models, calculate the loss values of the best sample point retrieved (in each run), and compare such values, with their standard deviations, to that found in the original Bayesian calibration experiments.

3.11.2 Franke-Westerhoff and Brock-Hommes Models

Emulating our Bayesian calibration experiment just as above, upon the Franke-Westerhoff and Brock-Hommes models, yields the y^* curves visualised in Figures 29 and 30. Again, we input the true parameter vector itself, as the sample point we seek to test, and generate its posterior as described in the previous section. Once again, just as seen above, this posterior for the true parameter vector possesses a variance, and it is hoped that the mean of this posterior, at the true parameter vector, is a minimum of the constructed fitness surface.

Our BO scheme sequentially selects points on this surface just as before, in an attempt to locate the minimum value of the fitness function. Such a BO scheme, upon the Franke-Westerhoff model, performs marginally better than random search for most iterations in the experiment. The scale of the variance in the true parameter vector's posterior is concerning however, as it is a few multiples of the entire posterior reduction, from the BO scheme, across the 75 sequential sample points. The loss values of the next section

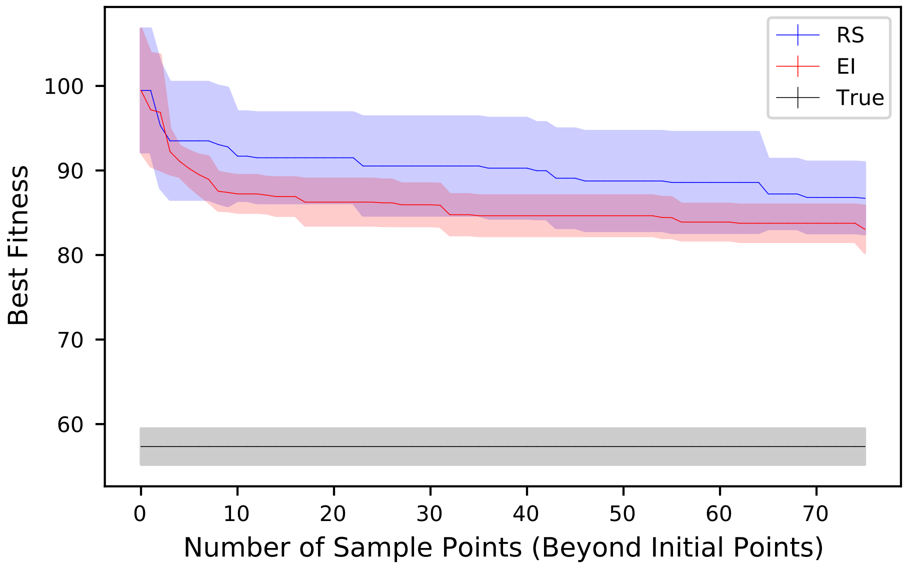

Figure 30: Our emulated Bayesian calibration scheme, upon the full dimensionality of the Brock-Hommes model. 25 initial sample points were used, followed by 75 sequentially. The experiment was repeated 10 times, and each sample point was repeated 10 times also, within each experiment. The lengths of all time-series were 100. Note the fitness of the true parameter vector, denoted in black, as well as it's variance, denoted by the shaded region. Lastly, note Bayesian optimisation's ability to optimise quicker than random search again.

Model	Method	Mean Loss	Standard Error
Franke-Westerhoff	Frequentist Calibration - Emulation	0.771	0.178
Franke-Westerhoff	Bayesian Calibration	0.908	0.089
Franke-Westerhoff	Bayesian Calibration - Emulation	1.301	0.186
Brock-Hommes	Frequentist Calibration - Emulation	1.419	0.251
Brock-Hommes	Bayesian Calibration	1.318	0.312
Brock-Hommes	Bayesian Calibration - Emulation	1.481	0.170

Table 6: Loss values, from Equation 28, for all experiments conducted thus far. Note the low loss value for emulated frequentist calibration, in comparison to the two flavours of Bayesian calibration. Also note the approximately equal losses incurred for all three methods upon the Brock-Hommes model.

also tell us that the proximity of the minima curves to the ground-truth is not a sign of successful calibration; a sample point with appreciable loss appears to have produced a very similar fitness to the ground-truth.

The same BO scheme also outperforms random search upon the Brock-Hommes model. Posterior fitness for the true parameter vector is much lower however, which indicates that our distance function is differentiating between the time-series produced at the ground-truth sample point, and other points in the parameter space.

3.12 Loss Values - Emulation

Let us now discuss the loss values of Table 6 in detail. First, let's examine those of the Franke-Westerhoff model. We see that emulated frequentist calibration produced significantly lower loss upon this model, than its Bayesian counterparts (via MCMC and emulation). Such a loss indicates the quality of the parameter vectors which the schemes returned [25]. A lower value indicates parameter vectors closer to the true parameter vector, $\vec{\theta}_{emp}$. Thus, the emulated frequentist calibration scheme appears to have definitively produced the best parameter vector, on average, and thus has most successfully calibrated our model.

In order to gain a rough indication of how close such parameter vectors are to the true values, we present figures of the loss for each parameter, over each of the runs, in Appendix C. However, more easily we can insert a vector of length n, where n is our model's dimensionality, consisting of a set of dummy fractional losses, and calculate the value manually.

For instance, if we would like to calculate the loss, whereby each parameter was a fraction of the normalised bounds $c \in [0, 1]$ away from the true parameter vector, then we could insert such a vector into Equation 28, and get an indicator for the loss incurred at such a fractional difference for each parameter. As an example, say each of our true parameter values were in the middle of their respective bounds. When normalised, such true parameter values would have all values of 0.5. Next, say our calibrated parameter vectors are always exactly half of our true parameter values. When normalised, such parameters would all have values of 0.25. We input the fractional differences, $c = 0.25$, to a vector of length n, pass such a vector into Equation 28, and we can calculate the loss of being 25% the distance of our parameters' bounds away from the true parameter values. Such a quantity is of course, dependant on the dimensionality of the model, so we present calculations of such loss indications for each model separately.

[25]These were the parameter set of lowest fitness, $\vec{\theta}^*$ for the emulated schemes, and the mean of each parameter distribution, with burn-in discarded, for the MCMC schemes

For the Franke-Westerhoff model for instance, if each parameter was 20% of its bounds' distance away from each true parameter value, a loss of 0.566 would have been incurred ($\sqrt{0.3^2 \times 8}$). At 50%, this value rises to 1.414. Thus, we gain intuition on how much better the calibrated parameter sets are for emulated frequentist calibration, in comparison to emulated Bayesian calibration for instance. These intuitions are supported by Appendix C.

For the Brock-Hommes model, no calibration scheme looks to have stood out significantly, with all mean losses being within the standard errors of the other schemes. Incurring 40% error for all parameters produces a loss value of 1.327, while a loss of 50% produces a loss of 1.658. Thus, we see to what degree, the calibration is worse for the Brock-Hommes model than the Franke-Westerhoff. Again, visualisations of such losses are available in Appendix C. Recall also, that while the emulation schemes ran just as intended, the lack of an adaptive MCMC scheme meant that undesirable acceptance rates were observed in both Bayesian calibration experiments using MCMC. Platt notes the importance of such schemes in his work [11], but in order to keep our implementation simple, we decided not to use such schemes. In addition, we are only interested in improving the performance of the Bayesian MCMC scheme, to provide us with a fair benchmark, against which to compare the emulated form of Bayesian calibration.

With that in mind, we see that in the experiments for both toy models, emulated Bayesian calibration incurs a larger loss than its MCMC counterpart, and by some distance for the Franke-Westerhoff model. This indicates that even if we used an adaptive MCMC scheme, the losses would have likely only decreased for both Bayesian MCMC calibration schemes. However, we should take into account the number of executions each scheme uses in one of its runs. MCMC uses 5000, while the emulated Bayesian calibration scheme we create only uses 100. This represents a 50-fold reduction in the number of sample points, which is rather remarkable alone. With a better likelihood, as developed by Platt [13] and Grazzini already [24], the performance of our emulated Bayesian calibration scheme could improve even further. Gutmann and Corander [26] present analogous multiple-fold reductions in sample points required, in their analogous scheme which improves upon approximate Bayesian computation, for a handful of simpler simulatory models. We merely attempt to show that such methods are directly applicable to the Bayesian calibration methods being developed in the ABM community. This point encapsulates one of the major contributions of this thesis.

In the remaining chapters, we shall discuss how emulation is implemented in an array of other fields, and how we may go about improving the performance of our base BO scheme. Bayesian optimisation, while better than random search in all of the above experiments, still has room for improvement in our internal calibration experiments, as indicated by the intuitive loss values. It is being bested by the increased dimensionality of our toy models, and so we seek to help it in this battle against an ever increasing parameter space volume.

Chapter 4

Emulation in ABMs

In the previous chapter, we detailed Gaussian processes, which we used to emulate our toy ABMs up to their full dimensionality. We combined these GPs with a sequential sampling scheme, which exploited the analytical variance estimates GPs provide, to then create a BO loop. We then showed how this led to greater sample efficiency in our ABM calibration experiments.

As hinted at previously, such methods are not unique to ABMs. Numerous fields dealing with analogous issues of sample efficiency, expensive simulations and increasing input dimensionality have benefited from Bayesian optimisation (BO), and even more so from emulation solely.

In this chapter, we discuss such fields' implementations, and some of the specific problems they may face. In the next chapter, we discuss extensions to our base BO scheme, hopefully guided by such implementations in other fields.

Before we progress however, it is important to recognise that the GP literature experienced an early bifurcation in the notation and terminology used. The historical origins of GPs date back to work in time-series extrapolation and interpolation by Kolmogorov [47] and Wiener [48] in the 1960s, before re-appearing in the Geostatistics literature, through works such as Matheron's [49] in the 1970s. It was in this domain, that the methods found a home. Given that Geostatistics is primarily concerned with spatial or spatiotemporal problems, it should be no surprise (in hindsight) that interpolation work from the likes of Matheron (and formerly of Krige [50]) would find use.

However, the field that developed from such early works became known as kriging, expressing many of the ideas of Chapter 3 with slightly different notation and terminology. This makes the reading of kriging and GP literature somewhat more difficult, particularly as the work we shall refer to in this chapter may use either. Thus, for the reader, we shall translate all kriging-inclined literature to the equivalent terminology of GPs. Those who seek to read more about kriging may look to the review of Kleijnen [51], or the seminal paper of Sacks [52], which explicitly links kriging/GPs to computer simulations.

4.1 Economics

Much of the calibration work conducted thus far, while in theory agnostic to field, originated from economic ABMs and related simulatory work. Thus, we begin our literature review here, specifically with the 2014 work of Salle and Yildizolu [53], which introduced many practitioners of computational economics to the ideas of emulation and kriging.

The authors emulate two economic ABMs with an NOLH one-shot sampling scheme (a variant of Latin Hypercubes), a constant mean function $\mu(\vec{\theta}) = c$ (where $c \in \mathbb{R}$ denotes said constant), and a separable RBF kernel. An exponential kernel, given by

$$K_{\mathrm{exp}}(\vec{\theta}) = \exp\Big(\frac{-\vec{\theta}}{l}\Big), \tag{70}$$

where l denotes a single length scale, is also used, alongside a Matérn kernel, as described in Chapter 3.

Interestingly, the paper looks at the RMSE (root mean square error) between the kriging meta-model, and the underlying ABM, in an attempt to validate a more efficient sensitivity analysis procedure. They conclude by stating the large reduction in sample points required to capture all desired effects and parameter interactions (85 as opposed to 10,000 Monte-Carlo samples), much like we do for our calibration experiments. However, note the lack of a sequential sampling scheme (which perhaps wasn't required for the authors' purposes). Salle and Yildizolu didn't use a BO loop, only a GP to emulate the ABM.

The next work we discuss is that of Dosi et al. in 2016 [54], within which a GP is again fit via a one-shot NOLH design, to an ABM investigating firm growith rates. The authors state that their use of emulation mimics Salle's methods directly, illustrating the significance of their work. The mean function is adapted to a polynomial of order zero or one (with order two yielding poorer results). As in Salle and Yildizolu's work, the primary objective is to aid a sensitivity analysis procedure upon their selected ABM.

Barde and van der Hoog's 2017 paper [55] also builds on the same work, applying emulation to a version of the EURACE macroeconomic model, a large-scale simulatory effort which, in the author's words, is "computationally heavy". They use an NOLH scheme again, alongside Stochastic Kriging, which bears much resemblance with heteroscedastic GPs [56], which we discuss in the next chapter. Both methods seek to deal with input-dependant noise in the parameter space. Interestingly, Barde and van der Hoog tackle a calibration problem for their expensive ABM. They use a well-known distance function in the economic ABM community, the Markov Information Criterion [20]. Detailing this distance function function is beyond our scope, but we note that it yields a much noisier fitness surface, which seems the be the rationale behind the authors' use of Stochastic Kriging.

Bargigli et al. [57] emulate an ABM of the Japanese credit market, for their external calibration (against empirical data) and sensitivity analysis experiments. Much like us, they seek the 'best' parameter set according to some moments-based distance function. They sample via NOLH, and test separable variants of the Matérn, RBF and exponential kernels. Interestingly, they also apply a diagonal noise matrix to their covariance matrix, the terms of which are estimated from repeated samples at each point in parameter space.

Lamperti et al. [27] present an innovative approach to emulation, in using a random forests metamodel, in combination with a unique sequential sampling scheme, centred around sample point labelling, according to user-defined criteria. Random forests metamodels are quicker to evaluate at high numbers of sample points than GPs, but they don't possess natural variance estimates as GPs do. Sensitivity analyses are also less naturally realised than in GPs, as one has to look to the number of times each parameter splits within the random forests' decision trees. The authors go in to display mean-squared error comparisons of kriging and random forests, emulating their selection (six-dimensional) economic ABM. They show that over 50 and 500 evaluations, their emulator is closer to the true fitness surface than kriging. However, it appears the authors use a standard RBF kernel, rather than a separable or additive kernel, which would allow for different length scales in each input dimension[26]. The underutilisation of such GP features motivates

[26]We see, in the authors' attached analysis that the `sklearn.GaussianProcessRegressor` Python module is used. However, no length scale parameter is passed, meaning the length scale initialises to unity, and the `GaussianProcessRegressor` object doesn't utilise its anisotropic kernel (another term for separable).

much of the remaining thesis. How good can GPs be, when more of such features are utilised?

Bayesian optimisation on the other hand, is beginning to be utilised within economic ABMs. Works of Kim [58] and Tran [59] present initial attempts, respectively utilising a GP with a Matérn kernel and a 'tree-structured Parzen Estimator' [60], the latter of which appears to hold similarities with the decision-tree based method of Lamperti et al. Both papers also used Expected Improvement to guide their sequential sampling, with Kim also utilising two alternatives (Predictive Mean and Predictive Variance, which respectively exploit and explore maximally). Minima curves similar to ours are presented, also comparing their calibration methods against random search.

4.2 Epidemiology

4.2.1 Covid-19

We take a brief detour at this point, to discuss an ABM which has gained large public attention, in the wake of the Covid-19 pandemic.

The spread of Covid-19 in early 2020, and the ensuing pandemic rapidly brought the need for analytical and simulatory prediction tools, with which policy makers could gain quantitative insight into the effectiveness of policies such as school and workplace closures, mask-wearing and social distancing. One such tool, an individual-based model (Epidemiology's ABM analogue) developed at Imperial College London under the leadership of Neil Ferguson, was widely reported to have re-directed the UK government's intended response to the pandemic [7]. The model, published in subsequent months on GitHub[27], became the source of public inquiry and scientific cross-examination for a number of weeks after. Documentation for the model is pending, but at least 91 parameters, continuous and discrete, are present in at least one of the input parameter files of the model[28]. Running the model in its entirety requires approximately 20GB of RAM, which puts it beyond our scope (given hardware constraints). But the model is an ideal candidate for the emulation methods discussed in this thesis, and an illustration of the need for tractable calibration and sensitivity analysis procedures, maximised for sample and resource efficiency.

4.2.2 Literature

We begin our review of the emulation literature in Epidemiology with the 'relatively' early work of Dancik, Jones and Dorman in 2010 [61], within which the authors apply GPs to emulate calibration and sensitivity analysis experiments, in an (at least 25-dimensional) ABM of parasitic infection[29]. A Latin Hypercube design is employed, alongside a constant non-zero mean function and a separable RBF kernel, with a diagonal noise matrix added. Very interestingly, it appears that the Bayesian calibration procedure implemented may hold many parallels to the method outlined by Grazzini in his 2017 work [25]. This perhaps shouldn't be a surprise. MCMC schemes have been around for decades, and when paired with the need to construct likelihoods for non-tractable simulatory models, we arrive at a problem which, as discussed a few times previously, permeates many fields. Indeed, their field observations are assumed to be i.i.d, and the posterior adopts an analogous form to that used in thesis.

Willem et al.'s 2014 work [62] possesses many of the same features of Dancik, Jones and Dorman's. An expensive IBM (of influenza) needs to be emulated, and a Latin Hypercube

[27]Available at github.com/mrc-ide/covid-sim.

[28]Again, available at github.com/mrc-ide/covid-sim/blob/master/data/param_files/p_NoInt.txt.

[29]20 parameters are held fixed, while five are varied.

design is combined with a surrogate model to do so. Symbolic Regression is used as the surrogate, which searches over a space of simple mathematical operations to find a suitable surrogate model. This is then combined with a genetic algorithm based generation of new symbolic regression metamodels, and is labelled as an active learning emulation scheme (or equally, just another sequential sampling scheme). As we're seeing, such sequential schemes seem to be less utilised than their one-shot counterparts. Willem's work does however, shift from the common Latin Hypercube (variant) plus separable RBF kernel emulation schemes, to aid calibration and/or sensitivity analysis exercises in expensive IBMs/ABMs.

Oyebamiji et al.'s 2017 work [63] fits firmly into such a mould however, with noisy GPs also being utilised. Interestingly, in 2019 they expanded upon their initial work by combining GPs with a Dynamic Linear Model, and modelling local variations from this model with a GP [31]. Discussing such an approach further is beyond our scope, but does hint to more complex emulation procedures being implemented on ABMs.

Fadikar et al.'s 2018 work [64] attempts to expand upon the aforementioned core method by implementing Quantile Kriging, which seeks to model heteroscedastic noise in the input space. Fadikar's ABM of Ebola spread produces noisy outputs, but such outputs are not well modelled with Gaussian noise at some regions of the parameter space. This concern should be taken into account for our experiments. We attempt to take the output at any sample point in expectation, via a sample average of multiple executions, but currently discard variance information about each sample in the interests of simplicity. We also occasionally use the median (and denote this when we do), in an attempt to control for non-Gaussian noise. It's understandable that if variances were available across each sample point, then such information could be included in a variant of our current acquisition function, the Expected Improvement, which uses the analytical (and Gaussian) variance estimates across the parameter space to guide its sequential sampling. We acknowledge this, and propose is at as a very fruitful topic for future investigation.

Davis et al.'s 2020 work [65] adopts a flavour of neural network, namely Mixture Density Networks (MDNs), to model the relationship between input parameters and a set of mixture components, for the probability density function that the MDN outputs. In short, such methods seem to deal with non-Gaussian outputs more naturally than GPs, but require much more data [30].

We complete this section by discussing the work of Andrianakis et al. [66], within which the authors attempt to calibrate a 22-dimensional ABM of HIV transmission. The authors fit separate GPs to the mean and the variance of sample points, requiring a higher amount of replication at each sample point to do so. A Matérn kernel and third-order polynomial mean function is used to do this. Their most interesting innovation is in what the authors describe as History Matching. In sequential 'waves'/iterations, regions of the parameter space are discarded, according to an implausibility measure built by the authors. At each wave, a Latin Hypercube design is used, the two GPs are re-fit, and regions of the parameter space are discarded. This allows the authors to deal with high input dimensionality, with a 96-input IBM of HIV transmission being emulated in their subsequent 2017 paper [67]. However, the authors also replace their GP emulator with a linear regression model in this latter paper (seemingly to reduce complexity), but note that GPs could equally have been used. The sequential reduction of the input space is a particularly relevant idea for us, which we note as another very fruitful avenue for future work.

[30] Very interestingly however, the author discusses how an MDN could be used to produce a synthetic likelihood for Bayesian inference schemes such as approximate Bayesian computation. This appears to be precisely what Platt does in the follow-up work [13] to his ABM calibration methods comparison paper [11], which we draw upon in earlier chapters.

4.3 Health Economics and Social Sciences

Emulation can be found in the Heath Economics literature as far back as 2004, where Stevenson, Oakley and Chilcott [68] use GP emulation to improve previously prohibitive sensitivity analyses of expensive individual-patient models, hence allowing estimation of the cost-effectiveness of certain osteoporosis treatments. This vein of work in Health Economics, of expensive simulations running cost-benefit analyses of different treatments, and the subsequent emulation requirement, is perhaps unexpected to those external to the field. But emulation methods again find uses here. Rojnik and Naveršnik's's 2008 work [69] is another example of this, where a Latin Hypercube design is again selected, along with a GP to aid resource allocation in breast cancer research.

Agricultural studies have even utilised emulation. Parry et al.'s work [70] attempts to make the sensitivity analysis of an expensive ecosystem ABM more tractable. Here however, an initial input dimensionality of 35 was reduced to 10 before emulation.

Sociology is not exempt from emulation either, with Bijak et al.'s ABM of marriage formation again benefiting from a more tractable sensitivity analysis, as a result of such methods. Latin Hypercube schemes seem to dominate these areas of the literature. In such computer experiments, these schemes are obvious candidates for calibration and sensitivity analyses.

4.4 Engineering

Engineering itself can be divided up into a myriad of sub-disciplines, and so in this section, we attempt to outline some of the emulation work in this area. Park and Jeon's 2002 work is an early example of such work, within which the parameters of a transport model for nuclear fusion must be determined, from an expensive simulation. But, as perhaps to be expected, in such early implementations, no techniques beyond a base GP and Latin Hypercube design were used. Interestingly, Gengembre et al. [71] present a use in the building design field, where simulatory models are used to model features such as heat loss of a building. They use a full BO loop (while not labelled as such), adjusting their Expected Improvement acquisition function with constraints, involving features such as energy cost, mean window ratio and life-cycle cost. These examples serve as illustrations of engineering's use of emulation, but for a comprehensive review of such literature, we direct the reader to Greenhill et al.'s comprehensive 2019 review work [30]. Within it, the authors review the use of emulation, and more specifically BO, over the following sub-fields/topics: incorporating prior knowledge; high-dimensional BO; multi-objective BO; constraints; parallel BO; multi-fidelity BO; and mixed-type inputs. The final citations list names 125 papers, the majority of which are BO implementations. This illustrates the maturity of engineering emulation techniques, the comprehensive reviewing of which is sadly beyond our scope. We are particularly interested in the high-dimensional BO methods Greenhill discusses, and leave this as another avenue of further exploration.

4.5 Cosmology

Once again, the use of emulation in Cosmology appears to be extensive, given the exorbitant scope of their simulatory exercises. Replicating the universe appears to be expensive. We therefore discuss a few key works from this body of literature.

We begin by discussing Vernon, Goldstein and Bower's 2010 work [72], which was the recipient of the 2010 Mitchell Prize for the Best Applied Bayesian Article worldwide. Within it, they pioneer the previously discussed approach of History Matching, which enables a previously infeasible sensitivity analysis to be conducted on a state-of-the-art computational cosmology model known as Galform, which "models the creation and evolution of approximately one million galaxies from the beginning of the Universe until the current day".

In this model, 17 parameters are present. These are reduced to between six and nine 'active' parameters, which are believed to generate most of the output variance. We note that again, a separable RBF kernel is used. Interestingly, more weight is placed on determining the mean function than the covariance function. The authors recognise this ongoing debate, and cite strong prior knowledge of physically interpretable monotonicities (which are more easily expressed through the mean function) as a major reason for this focus. A further detailing of their work is beyond our scope, but we can note that History Matching certainly poses an interesting method for future work to investigate. They build upon their work in the 2010 paper of Bower et al. [9], where the authors mention how such procedures cope better with high-dimensionalities than "more traditional methods such as MCMC".

Very interestingly, the work of Gutmann and Corander, which we discussed in Section 2.7, is implemented by Leclerq in 2018 [73] to improve the sample efficiency of a Bayesian inference scheme of a six-dimensional cosmological simulation. It also appears, from preliminary readings, that the likelihood (again, intractable for the simulator) may be similar to that of Grazzini's, and indeed it is then extended via the creation of a Gaussian-Gamma likelihood, which we omit in the interest of brevity. Only two of the six parameter are varied however, and Leclerq notes the difficulties of performing BOLFI (see Section 2.7) in high dimensions (increasing parameter space volume, kernel hyperparameter optimisation and acquisition function optimisation). Finally, some high-dimensional BO methods (HDBO), which we discuss in the next chapter, are cited by Leclerq as remedies to such problems. Such references confirm our suspicions that such HDBO methods are promising areas to explore.

4.6 Conclusions

In this chapter, we have attempted to provide the reader with a sense of where our emulation (and secondarily our calibration) work sits amongst the wider body of literature, crossing over from ABMs into more general simulators. We discussed how some authors opt for other emulator models such as random forests and neural network methods, and how, if a sequential sampling scheme is to be used with such emulators, then schemes different to the GP-specific Expected Improvement must be derived.

We also noted the difficulties that are incurred from the dual sets of notation between kriging and GPs. While theoretically equivalent, such a duality increases the risk of duplicating ideas between fields, and decreases the ease of interdisciplinary emulation work. We noted the dimensionality of the simulations in most cases, and only History Matching seemed to embrace the problem of high input dimensionality, amongst our brief review of an expansive body of literature spanning multiple fields. Emulation of simulations is a sizeable topic, implemented in many domains.

Something which is of interest to us, post this chapter's review, is the possible lack of cross-pollination between the aforementioned emulation work, and machine learning's BO literature, which commonly finds uses in hyperparameter optimisation problems. The issue of high-dimensionality in these BO schemes is clearly one we'd like to focus on, and such HDBO algorithms indeed seem to be a potentially very fruitful subfield of BO, as discussed by Leclerq [73].

In addition, questions such as kernel and acquisition function selection remain. Given the vast space of kernels available (expanded only by the ability to multiply and add kernels to create new ones), how do we approach the task of selecting a kernel, and is this an important question to ask? The same could be asked of the acquisition function. We examine such questions in the next chapter, and for now leave the reader with the following note. Emulation is being utilised across a vast number of fields, and is a highly customisable science. Different emulators, sampling schemes, acquisition functions and dimensionalities are of interest each time, for a (seemingly finite) set of inference and uncertainty quantification tasks, and in this thesis, we work within just one configuration of such an experiment, albeit agnostically to the underlying simulator.

Chapter 5

Advanced Bayesian Optimisation

In Chapters 2 and 3, we showed how random search and BO struggle to find MSM minima in higher dimensions. We will discuss how we deal with such problems of high-dimensional BO (HDBO) shortly, but before this, we thought it useful to discuss the extensions to base BO which we don't implement in this thesis, and the reasons for this. We hope this may aid future ABM researchers in their navigation of the GP and BO literature.

We will begin by discussing the kernel selection problem (and the possible automation of this), before moving on to selecting the acquisition function, dealing with heteroscedasticity, and finally a set of discarded/unused features to deal with high numbers of data points, multiple outputs, parallelisation, multi-fidelity data, and alternative meta-models to GPs.

5.1 Kernels

If one was to observe the base GP and BO loop holistically, and select a feature which could yield performance improvements, the kernel wouldn't be an absurd place to begin. This is particularly so in this thesis, where we use a zero-mean function throughout, and allow all prior knowledge about the GP to be embedded in the kernel. That last statement is important, so let us re-iterate it.

Prior knowledge is inserted to the GP via the two functions that define it, the mean and covariance functions, $\mu(\vec{\theta})$ and $\Sigma(\vec{\theta}, \vec{\theta}')$. If we suspect a certain hyperplane in our parameter space is linearly increasing for instance, or quadratic about the midpoint, or monotonic perhaps, the practitioner must decide whether to embed this information in the mean function, or most likely not at all in the covariance function, which is more suited (as the name suggests) to embedding prior covariance information/knowledge. How much information does a single data point $\vec{\theta}$ provide about another? Is this the same across all dimensions of the parameter space? Is there periodicity? Is there noise in the underlying space? Is it infinitely differentiable? Does the covariance decay as a function of the Euclidean distance or the dot product of two points, or perhaps something more exotic? The answers to these questions are different for every problem, but one thing that (usually) can be stated is that the covariance function controls much of the behaviour of the GP, and it's far more common to work on selecting its form than that of the mean function [31].

This task, of covariance function(/kernel) selection is made even more complex, when we account for the ability to add, multiply and convolute any two valid kernels $K_1(\vec{\theta}, \vec{\theta}')$

[31] Although of course, not all GP practitioners will agree with this, with Vernon's 2010 work [72] being one example of where more effort is placed on selecting the mean function than the covariance function.

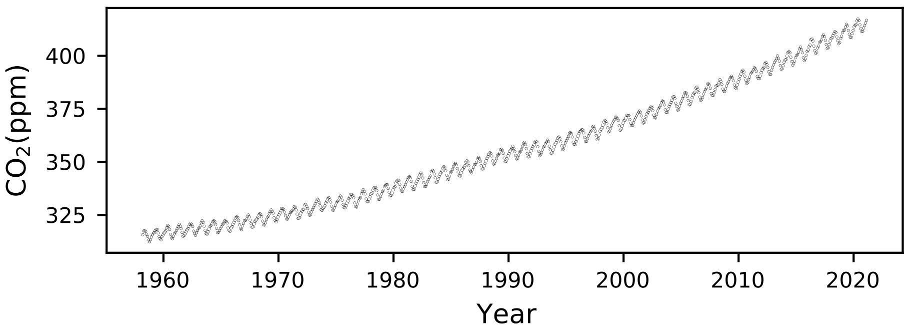

Figure 31: Data on CO$_2$ concentration over time, in parts per million. Note the combination of a linear trend, as CO$_2$ levels increase over long timescales, with a periodic trend, as the levels fluctuate per year.

and $K_2(\vec{\theta}, \vec{\theta}')$ to produce a composite kernel [34]. We now have an expansive space of possible kernels, which we'd ideally search over for a kernel that suits our data. That is to say, any periodicities, non-infinite differentiability, scales, lengthscales or covariance decay over specific hyperplanes would be detected by a goodness-of-fit measure, ran on the resulting GP from each kernel, fit to said data.

The Automatic Statistician project is precisely this [74]. Through a tree-based greedy search over a pre-defined kernel 'grammar' (a set of fixed base kernels, and some mathematical operations), the search for a kernel is automated. Such a method isn't guaranteed to find the 'best' kernel in the pre-defined grammar, but rather emphasis is placed on automating the search procedure. A dataset which such a method is ideal for is the Mauna Loa CO2 dataset, which is plotted in Figure 31[32]. In this dataset, we see a very obvious linearly increasing trend, with a periodic pattern from seasonal CO$_2$ fluctuations. Thus, a linear and periodic kernel will be of use, and are defined (in one dimension) as

$$K_{lin} = \sigma_{lin}^2(\theta - c)(\theta' - c), \tag{71}$$

where the constant $c \in \mathbb{R}$ determines the θ value for which there is zero variance, and all posterior GP draws will pass through. The scale parameter $\sigma_{lin}^2 \in \mathbb{R}^+$ is interpreted as previously, for other kernels. The periodic kernel, meanwhile, is given by

$$K_{per}(\theta, \theta') = \sigma_{per}^2 \exp\left(\frac{-2\sin^2\left(\frac{\pi|\theta-\theta'|}{p}\right)}{l_{per}^2}\right), \tag{72}$$

where $l_{per} \in \mathbb{R}^+$ and $\sigma_{per} \in \mathbb{R}^+$ retain their previous interpretations, and $p \in \mathbb{R}^+$ determines the periodicity of the kernel.

Such methods may appear to be a promising line of inquiry, but a few things make this not so. Firstly, we must consider the density of data available to us. In Figure 31, we have a high enough density of data to observe the periodic trend, and potentially even the smoothness of the underlying surface. This is most likely not the case in our high-dimensional ABM calibration experiments. Secondly, such greedy searches are typically expensive in the fitting of GPs. While (usually) less so than the time expense of running large-scale simulators, as the number of the sample points increases to the low hundreds, the fitting expense (primarily from the $\mathcal{O}(m^3)$ kernel inversion, where m is again the number of sample points) becomes particularly cumbersome in such a search scheme.

[32]Sourced from https://www.esrl.noaa.gov/gmd/ccgg/trends/data.html.

Thirdly, and perhaps most importantly, prioritising the kernel search problem neglects now known issues in our current BO scheme. Fitting a gradually increasing number of kernel hyperparameters (in the separable kernel case) and optimising an acquisition function in an increasing parameter space are the main issues we need to address. Searching for a kernel, over a pre-defined kernel grammar, may help, but not until more pressing issues are addressed.

We outline our proposed solutions to these problems in the following section, but not before outlining our selection process for acquisition functions, and a set of other possible BO extensions.

5.2 Acquisition Function

Our choice of the Expected Improvement (EI) acquisition function in Chapter 3 was driven more from desiring simplicity than maximal performance. EI represents an intuitive, popular and reasonably effective first choice to illustrate BO with. However, given how crucial the acquisition function is to the BO loop, we now discuss a few common alternatives.

5.2.1 Probability of Improvement

The EI is an extension of a much simpler acquisition function; the Probability of Improvement (PI) [75]. This acquisition function seeks to maximise the probability that a new sample point $\vec{\theta}$ leads to an improvement in the current best value $y^* = y(\vec{\theta}^*)$, as denoted in Section 2.5. Also as previously, via the analytical tractability of GPs, the PI can be derived as

$$PI_m(\vec{\theta}_{m+1}) = \Phi\left(\frac{\mu(\vec{\theta}_{m+1}) - y^*}{\sigma(\vec{\theta}_{m+1})}\right), \tag{73}$$

which is actually just one part of the first term of the EI in Equation 68. Unfortunately, the PI often results in an undesirably high level of exploitation [75], and so the EI, or rather the *Expected* Improvement, is preferred to the *Probability of* Improvement.

5.2.2 GP-UCB

Another popular acquisition function is GP-UCB (Upper Confidence Bound). As the name suggests, by taking an upper quantile of the GP variance σ_{m+1}, say the 95th, at a sample point candidate $\vec{\theta}_{m+1}$, and adding this to the mean, the GP-UCB attempts to "optimise in the face of uncertainty" [75]. A large variance could yield a high reward, in the form of better optima [33]. With this in mind, we can denote the GP-UCB acquisition function as

$$UCB(\vec{\theta}_{m+1}) = \mu(\vec{\theta}_{m+1}) + \beta\sigma(\vec{\theta}_{m+1}), \tag{74}$$

where $\beta \geq 0$. The setting of β is rather involved, and can benefit from theoretical work on the cumulative regret/error bounds of GP-UCB, from the like of Srinivas [76]. It's due to complexities such as these that we don't use GP-UCB, and instead opt for the more interpretable EI.

5.2.3 Acquisition Functions in High Dimensions

Acquisition function optimisation in high dimensions is one of the main challenges of high-dimensional BO, and results from an increased parameter space volume, over which

[33]The GP-UCB literature typically maximises, whereas in this thesis we seek to minimise our distance function. Such differences can be easily dealt with in implementation, but should be acknowledged.

optimisers must search for minima. As we will see later in this chapter, by either imposing an additive structure on the space, or by projecting it into a lower-dimensional subspace, we're able to cast such a high-dimensional acquisition function optimisation to one of lower dimensionality. Thus, we hold further discussion of this until such high-dimensional BO methods have been discussed.

5.3 Unused GP features

A GP and BO loop, as mentioned previously, is very customisable according to the problem at hand. In this section, we briefly discuss some further extensions we haven't used, why we haven't done this, and the impact we think they could have on our problem.

5.3.1 Heteroscedastic GPs

The extension of GPs to deal with input-varying noise is generally recognised to have first been dealt with by Goldberg, Williams and Bishop in 1998 [77]. Their method of applying a second GP to model such noise is intuitive, but hasn't been implemented due to time constraints. It also adds a lot of notational complexity, which we felt was slightly beyond our scope. Ankenman's 2010 work [78] is well recognised for exploiting replication to deal with heteroscedasticity, by assigning replications proportionally to the perceived level of noise in a region of the parameter space. However, as Binois, Gramacy and Ludkovski write in their 2018 work [56], the inference for Ankenman's introduced noise parameters uses simulated data, of which a minimum number of repeats is required. This is in addition to to the closed-form, marginal likelihood-based inference of parameters such as the lengthscales in the same GP. Binois, Gramacy and Ludkovski note the suboptimality of this, and instead produce an entirely analytical, likelihood-based inference scheme for all heteroscedastic GP hyperparameters.

Not utilising heteroscedastic GPs in this thesis is a shame, as repeated executions at sample points are already performed to control for sample point variance. These most likely would be useful to a selected heteroscedastic GP scheme, whether that of Ankenman's or Binois, Gramacy and Ludkovski's.

5.3.2 Sparse GPs

We've mentioned multiple times now that the $\mathcal{O}(m^3)$ kernel inversion in the GP fitting procedure quickly compromises computational speed, acting as a bottleneck to the number of sample points that can typically be executed in a BO scheme. Sparse GP methods are one way of dealing with this [79]. These methods seek to generate a set of "support points" which can summarise the information held by all the sample points in fewer, easing the $\mathcal{O}(m^3)$ inversion somewhat. We never exceed 10^3 sample points in any our experiments, but practitioners with more computational resources, thus allowing more sample points, are (ironically) more likely to run into such computational constraints. Sparse GPs present a suitable starting point to deal with such issues.

5.3.3 Multi-output GPs

Another aspect we haven't exploited is the emulation of multiple output variables. ABMs, and indeed most complex simulators, are able to produce multiple different output variables, and work has been done to use all such outputs to aid calibration and sensitivity analyses at unexplored points in parameter space. Such additional outputs are usually gained at minimal expense, and can be gathered from a single simulator run. Making use of such a set of outputs is known as co-kriging in the kriging literature, and as Liu writes in 2018 [80], overlaps with multi-task learning in machine learning. The key idea is to exploit correlations between the outputs, to either improve the emulation for all outputs, or just for a single primary output. The latter borders into another unexplored area, known as multi-fidelity GPs, where data of varying quality (earlier versus later/more advanced versions of a simulator for instance), and varying quantities are available, and we seek to use all such datasets to improve the GP emulation of our primary simulator, and its output data.

5.3.4 Parallel GPs

Another extension which we couldn't implement (due to hardware constraints) is parallel GPs. The sequential scheme outlined previously is typically performed in batches at each BO loop, with each batch being distributed over a set of cores or workers. This can be extended to bulding custom BO schemes which are able to exploit in multiple locations of the parameter space in tandem [81].

5.3.5 Other Emulators

Lastly, as seen in Chapter 4, GPs are far from the only emulator that can be used in a BO loop. Random forests and neural network methods are popular, with each method having its own advantages and disadvantages. Emulation with GPs benefits from analytical variances over the entire space, better interpretability than say, neural networks for instance, and easily derivable sensitivity analysis measures. For these reasons, along with the belief that high-dimensional GP and BO methods seem to be under-utilised in the ABM calibration literature, we chose GPs as our emulator model.

5.4 High-Dimensional Bayesian Optimisation

The aforementioned issues an increasing input dimensionality induces are typically combatted by two different methods. The first is to impose an additive structure upon the parameter space, decomposing the problem into a set of lower (but typically one-dimensional) subspaces. The second is to project the entire space down to a single, lower-dimensional subspace. The former, which we shall call additive high-dimensional Bayesian optimisation (HDBO), is well-suited to spaces with low parameter interactivity; each dimension is assumed to be independent of the others. The latter, which we shall label as subspace-embedding HDBO, is typically used when it's believed that a few parameters are responsible for the majority, if not all, of the variance in the selected ABM output. Thus, the output may be insensitive to many parameters, and the problem can be cast to one of lower (effective) dimensionality.

5.4.1 Additive HDBO

Let's begin our additive HDBO discussion with a method introduced earlier; additive kernels. We reproduce Equation 59 here for the reader's benefit:

$$K_A(\vec{\theta},\vec{\theta}') = \frac{1}{n}\sum_{i=1}^{n} exp\Big(\frac{-||\theta_i - \theta_i'||^2}{2l_i^2}\Big).$$

Note how each term of the summation depends on only the i^{th} dimension of the sample points $\vec{\theta}$ and $\vec{\theta}'$.

Now suppose, instead of varying each kernel in the above summation along one dimension, we allowed them to vary across any number of dimensions n', where $n' < n$, but with the caveat that each dimension could only be used once, across the set of kernels?

For instance, if three parameters were known to have interactions, but the remaining parameters had no such interactions, we could produce a summation where the first term depended on these three parameters, and each of the other terms only depended on the remaining parameters separately. Well, as Gardner writes [82], the possible additive kernels that could then be constructed grows super-exponentially, with $n = 10$ already yielding 115,975 combinations. It is computationally infeasible to calculate the marginal likelihood for the resultant GP model from each of these kernels, so rather some search through such a model space must be undertaken, as is precisely done by Kandasamy in 2016 [83], Gardner in 2017 [82] and Wang in 2017 [84]. However, such schemes are typically expensive, requiring many kernel evaluations via MCMC sampling in Gardner's case, or Gibbs sampling in Wang's. For these reasons, we opted not to pursue such methods.

With regards to the acquisition function however, this can now be decomposed over the same set of dimensions used in the additive kernel. This produces a much easier set of lower(one)-dimensional optimisation problems. Kandasamy constructs an additive form of the GP-UCB acquisition function for his additive HDBO solution, while Gardner reformulates the more common EI criterion.

5.4.2 Subspace-embedding HDBO

Of more interest to us, and our HDBO problem, are subspace-embedding methods, which seek to project the entire parameter space down to a lower-dimensional subspace, while still capturing most (if not all) of the the output variance of the original space. Such projections reduce the volume of the space, and thus aid problems stemming from this (acquisition function and kernel hyperparameter optimisations). However, the relative distances between the sample points must be preserved in the embedding space, to avoid distortions to the fitness function. Fortunately, the seminal result of Johnson and Lindenstrauss [85] confirms that the distances remain preserved, up to some error. As Nayebi, Munteanu and Poloczek write [86], the Johnson-Lindenstrauss lemma "states that any set of m points in an n-dimensional space can be embedded into $d \in \mathcal{O}(\log(n)/\varepsilon^2)$ dimensions, such that all pairwise l_2 distances are preserved up to a $(1 \pm \varepsilon)$ factor." (where notation has been adjusted to match that of this thesis). Such a theorem has found widespread applications across machine learning, where algorithms typically begin to struggle as dimensionality increases.

At this point, we think it fitting to provide evidence for why we believe such subspace-embedding methods to be of interest, for our set of models. In actuality, we only suspected this may be true for one of our models, namely the KS model discussed in Chapter 2. As discussed in Section 5.4, subspace-embedding methods are most effective when the effective dimensionality of the model is low. That is to say, most, if not all, of the variance of the output time-series $\vec{x}$ is due to a subset of the model's parameters. This was observed to be the case for a similar ABM, for which a sensitivity analysis was undertaken by Dosi, Pereira and Virgillito in 2016 [54]. We present a figure from such work in Figure 32, where a variance decomposition procedure is performed upon their ABM's parameters, in a bid to quantify the sensitivity of each parameter to their model's output.

Note the single parameter $sMin$ which contributes most of the variance (denoted by the Sobol index on the vertical axis). Many of the other parameters seem ineffectual, while a select few dominate the variance decomposition. Subspace-embedding methods are designed for precisely this situation, and so we elected to implement one such subspace-embedding scheme upon the KS model (as discussed in the next chapter).

Let us now outline how such subspace-embedding methods generally work. We define the projection matrix $\Omega \in \mathbb{R}^{d\times n}$, where d represents the dimensionality of the subspace we project to, and n represents the original dimensionality (the number of parameters)

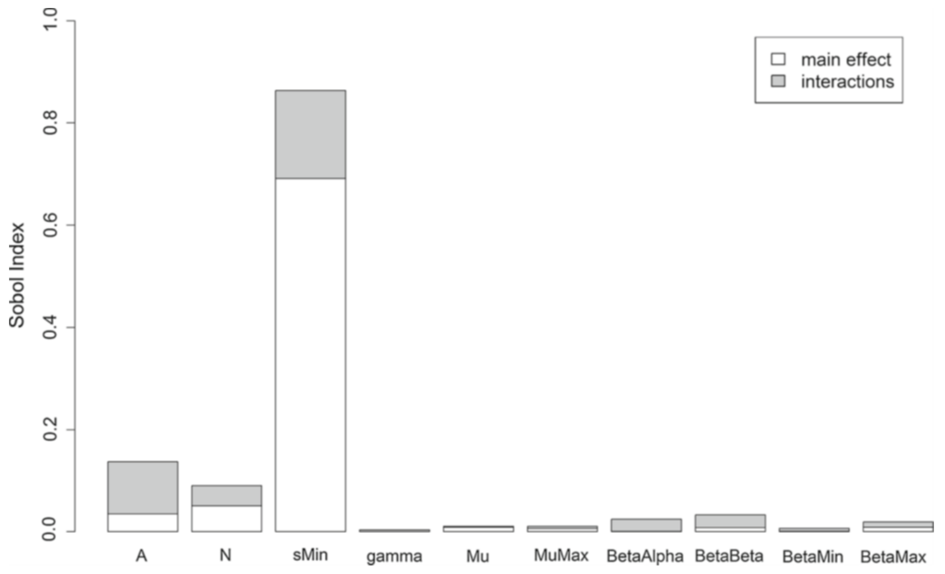

Figure 32: A variance decomposition from Dosi, Pereira and Virgillito [54], for a large-scale ABM similar in nature to the KS model. Note the few parameters contributing most of the variance in the output (measured by a quantity known as a Sobol index). Such variance decompositions can tell us how much of the variance is caused by each parameter, as well quantifying the interactivity (or lack thereof) between the parameters.

of our simulator. The matrix Ω applies a projection from n dimensions down to d. This matrix is generally applied to each of our sample points $\vec{\theta} \in \Theta$, $\Theta \in \mathbb{R}^{n\times m}$, resulting in a set of sample point in the embedding space

$$\Theta_d = \Omega \cdot \Theta, \tag{75}$$

where $\Theta_d \in \mathbb{R}^{d\times m}$. We also denote the pseudo-inverse matrix $\Omega^\dagger$ as the inverse operation, projecting a sample point $\vec{\theta}_d$ from the embedding space back up to the original space.

Lastly, recall the notational shift previously in Algorithm 6, whereby we initialised our fitness vector $\vec{y} \in \mathbb{R}^{m_{init}}$, and our set of parameter vectors $\Theta \in \mathbb{R}^{n\times m_{init}}$, to accommodate for the m_{init} sample points required to initialise each BO algorithm. These vectors are then appended to at each step of the BO loop, and eventually are of size (m) and $(n \times m)$ respectively. This notation is maintained for the subspace-embedding BO algorithms presented in the remainder of this thesis. We can now provide a generalised form of such subspace embedding methods in Algorithm 7.

Now, a few ideas remain to be dealt with in Algorithm 7. Bounds on both spaces are crucial. Lower and upper bounds in the original space are present in our problem, and projections back up from the embedding space may not be within such bounds. In addition, the bounds of the embedding space haven't been provided yet. Another crucial point is how to select the projection matrix Ω. Will we need to expensively learn this, as in the additive HDBO scheme? We discuss an early, seminal treatment of such matters in the next section.

5.4.3 REMBO

Wang et al. present REMBO, an algorithm to perform such a subspace embedding procedure, in their 2016 paper [87]. Their projection matrix Ω consists of elements drawn from a standard Normal distribution $\mathcal{N}(0, 1)$, hence the acronym REMBO ('Random

```
1:  Generate projection matrix $\Omega \in \mathbb{R}^{d\times n}$
2:  Initialise $\vec{y}$                                                    ▷ $\vec{y} \in \mathbb{R}^{m_{init}}$
3:  Generate $\Theta_d \in \mathbb{R}^{d\times m_{init}}$ in embedding space            ▷ $\Theta_d \in \mathbb{R}^{d\times m_{init}}$
4:  Calculate $\vec{y} = f(A(\vec{\theta},\varepsilon), A(\vec{\theta}_{emp},\varepsilon))\ \forall\ \vec{\theta} \in \Omega^{\dagger}\Theta_d$
5:  for $i$ in $m - m_{init}$ do
6:      Construct GP upon data $\Theta_d$ and $\vec{y}$
7:      Maximise selected acquisition function, return next sample point $\vec{\theta}_{d,i} \in \mathbb{R}^d$
8:      Project $\vec{\theta}_{d,i}$ up via $\vec{\theta}_{D,i} = \Omega^{\dagger}\vec{\theta}_{d,i}$
9:      Execute $\vec{\theta}_{D,i}$, return $y(\vec{\theta}_{D,i})$                    ▷ Alg.'s 1(10-19), 2(11-25)
10:     Append $\vec{\theta}_{d,i}$ to $\Theta_d$                                  ▷ $\Theta_d \in \mathbb{R}^{d\times(m_{init}+i)}$
11:     Append $y(\vec{\theta}_{D,i})$ to $\vec{y}$                                 ▷ $\vec{y} \in \mathbb{R}^{(m_{init}+i)}$
12: end for
13: Return $y^* = min(\vec{y})$
```

Algorithm 7: A generalised subspace-embedding BO algorithm. Again, as with our Bayesian optimisation algorithm, in Algorithm 6, we denote the steps of our frequentist and Bayesian calibration schemes, which should be accessed at the sample point execution step in this algorithm. This allows us to avoid over-complicating the subspace-embedding algorithm, with ideas such as seed repetitions.

Embedding Bayesian Optimisation'). Their box bounds for the embedding space must also be selected carefully. First, if they are too small, then the embedding space is unlikely to contain an optimum. However, if the bounds are too large, then their BO scheme will begin to struggle in the larger space. The bounds $[-\sqrt{d}, \sqrt{d}]$ are used in each embedding dimension to balance such requirements.

Secondly, not all points in the random embedding subspace will project up back into the box bounds of the original space. In such situations, a boundary projection operator, $P(\Omega^{\dagger}\vec{\theta}_d)$, maps the sample point projection up to the original space, $\Omega^{\dagger}\vec{\theta}_d$, onto the boundary of the original space.

Thirdly, in order to improve the chance of the embedding containing an optimum, multiple (four) runs of the REMBO algorithm are conducted, with the authors writing that the probability of failure decreases exponentially with the number of runs.

The kernel, differently to other multi-dimensional GP implementations, is not the separable RBF kernel, but the standard RBF kernel, with one lengthscale parameter l. While a breakthrough in the HDBO literature, REMBO has since been improved upon multiple times, and we present one such promising extension in the next section.

5.4.4 ALEBO

In their 2020 paper, Letham et al. attempt to tackle multiple issues with REMBO [33]. The first is with regards to the frequency and amount of induced distortion from projections to the boundary of the original space, via the boundary projection operator $P(\Omega^{\dagger}\vec{\theta}_d)$. Letham et al. show that for even moderate dimensionalities ($n \approx 20$) of the original space, the probability of a point in the embedding being projected up into the original bounds is as low as around 10% when the embedding dimensionality $d = 2$. This high frequency of mapping to the facet of the original space is only made worse by the severity of the non-linear distortions in the embedding space, when points that were meant to be approximately distance-preserved in the embedding space, end up mapping to the boundaries of the original space. In other words, points in the embedding space, which were meant to be close, can end up being very far apart when projected up to the original space, and thus, when these far-apart sample points are executed, and the fitnesses for these sample points used as the fitnesses for the points in embedding space, non-linear distortions occur in the fitness surface.

Letham et al. deal with this by only allowing sample points in the embedding space which project up to points inside the original space. They constrain their selected acquisition function (EI in most cases) via the following constraint:

$$\vec{\theta}_{m+1} = \underset{\vec{\theta}_d \in \Theta_d}{\arg\max} EI(\vec{\theta}_d), \quad \text{subject to} -1 \leq \Omega^{\dagger}\vec{\theta}_d \leq 1. \tag{76}$$

Note, no box bounds on the embedding space are required, as opposed to REMBO's $[-\sqrt{d}, \sqrt{d}]$ bounds for each dimension. But the constraint bounds are indeed the box bounds upon the normalised original space.

The second problem is less severe, but is due to distortions in the distances between points in the embedding space from Ω. A Mahalanobis kernel, of the form

$$K_{Mah} = \sigma_{Mah}^2 \exp(-(\vec{\theta}_d - \vec{\theta}'_d)^{\top} \Lambda (\vec{\theta}_d - \vec{\theta}'_d)) \tag{77}$$

is used, where $\Lambda \in \mathbb{R}^{d \times d}$ is symmetric and positive definite. Discussing the exact form of Λ quickly becomes involved, so we note a few key ideas here to aid the reader. If we take Λ to be a diagonal matrix, where each of the diagonal elements represents a lengthscale parameter, then we retrieve the matrix form of the separable RBF kernel outlined in Equation 58. In the separable kernel, we learned the lengthscale in each dimension, and a diagonal matrix allowed us to do this. If we use such a diagonal matrix in the embedding space, where each of the dimensions d are a composite of the original dimensions n, then such a kernel is ill-defined in the embedding space. As a solution to this, Letham et al. propose a $(d \times d)$ matrix Λ where the off-diagonal elements are non-zero, and each of the entries in Λ must be fitted, just as the individual lengthscales in our separable RBF kernel were. This is done by performing an upper Cholesky decomposition on the matrix Λ, resulting in $d(d+1)/2$ parameters to fit. Similar kernels are used in other randomly projected GPs such as that of Garnett in 2014 [88] and Delbridge in 2020 [89].

The final problem involves the probability that the embedding contains an optimum, p_{opt}. The above constraint in Equation 76 only reduces this quantity. Via assumptions on the location of minima in the original space, p_{opt} can be calculated as a function of the projection matrix Ω, the original dimensionality n, the true subspace dimensionality d_{true}, and the embedding dimensionality d.

Letham et al. find that hypersphere sampling, that is, sampling each row of Ω from the unit hypersphere $\mathbb{S}^d$, produces a higher p_{opt} than REMBO's $\mathcal{N}(0,1)$ sampling, to construct Ω. Thus, this represents the third and final REMBO extension.

The final algorithm of Letham et al. is labelled ALEBO (Adaptive Linear Embedding BO), and an algorithm summarising its implementation is presented in Algorithm 8.

We implement it, alongside base BO, upon the KS model in the next chapter. We also implement random sampling again as our one-shot benchmark. ALEBO should help us with the aforementioned HDBO problems we are facing, and hopefully will lead to higher quality minima in our ABM calibration experiments. We explore this in the next chapter.

```
1: Generate random projection matrix $\Omega \in \mathbb{R}^{d\times n}$ by sampling $n$ points from $\mathbb{S}^d$
2: Initialise $\Theta_d$, $\vec{\theta}_d \in \Theta_d$ via rejection sampling, with constraints of Equation 76
3: Select $\vec{\theta}_{emp}$                                                    ▷ $\vec{\theta}_{emp} \in \mathbb{R}^n$
4: Calculate $\vec{y} = f(A(\vec{\theta}, \varepsilon), A(\vec{\theta}_{emp}, \varepsilon))\ \forall\ \vec{\theta} \in \Omega^\dagger \Theta_d$      ▷ $\vec{y} \in \mathbb{R}^{m_{init}}$
5: Data $D = \{\Theta_d, \vec{y}\}$
6: for $i$ in $m - m_{init}$ do
7:     Construct GP upon data $D$, with Mahalanobis kernel.
8:     Maximise acquisition function in embedding space, via Equation 76. Return $\vec{\theta}_{d,i}$
9:     Project $\vec{\theta}_{d,i}$ up via $\vec{\theta}_{D,i} = \Omega^\dagger \vec{\theta}_{d,i}$
10:    Execute $\vec{\theta}_{D,i}$, return $y(\vec{\theta}_{D,i})$                ▷ Alg.'s 1(10-19), 2(11-25)
11:    Append $\vec{\theta}_{d,i}$ to $\Theta_d$                                ▷ $\Theta_d \in \mathbb{R}^{d\times(m_{init}+i)}$
12:    Store $y(\vec{\theta}_{D,i})$ in $\vec{y}$                                  ▷ $\vec{y} \in \mathbb{R}^{(m_{init}+i)}$
13: end for
14: Return $y^* = min(\vec{y})$
```

Algorithm 8: ALEBO. Note the reference to our frequentist and Bayesian calibration schemes, at the sample point execution step here.

Chapter 6

Numerical Experiments - KS and HDBO

Up to now, we've tested our base BO implementation upon the Franke-Westerhoff and Brock-Hommes toy ABMs, which, while of fair dimensionality (nine and 11), aren't near the input dimensionality of the most complex ABMs and simulatory models (as discussed in Chapter 4). Such models can reach dimensionalities of up to 100 (and most likely beyond), so it is important to test our emulation scheme with a model possessing a dimensionality at least somewhat closer to such values.

Let us now extend our analysis to one such model, namely the KS model detailed in Chapter 2.

6.1 KS Model

This model is a leading macroeconomic ABM, later iterations of which are being actively used to perform macroeconomic research at the University of Pisa in Italy [90]. The model is available within publicly available software on GitHub [16]. We use the 8.0 version of the software in this thesis.

Within the software (labelled LSD by its authors), the base version of the model is available for experimentation, alongside later (and even more complex) iterations, which add features such as labour markets to the base KS model. Documentation is available, and tutorials present to aid practitioners. The largest difficulty is in implementing a sequential sampling scheme, as the software doesn't do this currently. Thus it must interface with (in our case) a Python script, in order to sequentially run samples point which are output from our acquisition functions. Once this is overcome, we may interact with the model in exactly the same way as done for the Franke-Westerhoff and Brock-Hommes models.

LSD has a slightly larger set of parameters for the KS model, to that denoted in early literature [8]. These parameters have benchmark values, and bounds for sensible sensitivity analyses available in configuration files that accompany LSD. Such data are presented in Table 3 in Chapter 2. Only using the parameter values presented in such sensitivity analysis files is of utmost importance. Not doing so, could lead to the collapse of the simulated economy, and problems such as flat distance functions in large regions of the parameter space. This would subsequently cause problems in the base BO implementation, such as with length scale parameter fitting and acquisition function optimisation.

We filter the parameters to retain only those which are continuous, and are left with 33, which also have bounds and benchmark values available. This represents a three-fold increase in the largest model dimensionality explored previously, but we should note that it still isn't of the order of some the largest dimensionalities discussed in Chapter 4 (which approach 100 on occasion). The model takes an input seed ε, just as previously, and this is varied r times, across r runs at each sample point we execute.

With regards to the output time-series $\vec{x}$ to use, we chose the differences in GDP that the model readily outputs, as this represents a stationary time-series, which our frequentist and Bayesian calibration methods require. Luckily, most output time-series (which ABM practitioners are concerned with) can be made stationary by taking such first differences, so this isn't prohibitive.

The KS model is run for 100 timesteps in all following experiments, due to hardware and time constraints. A single model execution at this length takes approximately two to three seconds, but given additional time to commence the model executable, to store relevant data, the number of repetitions at each sample, running random search with the same number of repetitions alongside BO, the running of ALEBO, and the GP fitting and BO acquisition function costs, total simulation time increases quicker than is desirable.

Once again however, the executions at each sample point seemed to exhibit very non-Gaussian behaviour. Sample points didn't return numerical errors, as in the Brock-Hommes model, but would be very high on semi-rare occasions. Thus, once again, it was deemed sensible to stray from our earlier discussed theory, and utilise the median of the sample points, to account for such leptokurtic data.

We run all experiments in BoTorch again, as discussed in Section 3.10.2 for 25 initial sample points (via random search), followed by 50 sequential points, with five repetitions at each sample point. We set the aforementioned hyperparameters of the acquisition function optimisation, `num_restarts` and `raw_samples` to 25 and 64 respectively, again, as discussed previously in Section 3.10.2. This is also the case for our use of ALEBO, which we discuss next.

6.1.1 ALEBO

While not of extremely high dimensionality, the KS model still (theoretically) warrants a high-dimensional BO method, due to the widely reported degradation of BO in dimensionalities greater than 20. The ALEBO algorithm [33] is available as part of the Facebook Research package, `AX` [34]. The acronym stands for adaptive experimentation, which is better understood when we consider Facebook's interest in using `AX` to tune A/B tests [40] automatically. ALEBO is built into `AX`, but its underlying implementation uses `BoTorch`. In essence, practitioners looking to use ALEBO must choose between simplicity, but inflexibility in their BO loop (with `AX`), or higher customisability at the expense of increased difficulty and coding requirements (with `BoTorch`). We opted for the former, but efforts were made with the latter, which would be required to embed previously discussed features into ALEBO, should that be desired in future work.

The subspace dimensionality d, as denoted in Algorithms 7 and 8, is a hyperparameter of ALEBO, which must be selected by the practitioner. As Letham et al. state [33], it should be larger than the true subspace dimensionality (which we can only guess), but small enough so that the selected amount of iterations has a good chance of optimising a space of said dimensionality. Specifically, they write, "With a budget of 50 iterations and $d = 15$, it will be unlikely to get good model fit quickly enough to effectively optimize, so smaller values like 8 or 10 would be warranted. On the other hand, with the 500 iteration budget of the Daisy problem, one could set d in the 1520 range (the maximum supported by normal BO) to maximize p_{opt}". Such a statement hints that standard BO should

[34]See https://github.com/facebook/Ax and https://github.com/facebookresearch/alebo.

struggle in the full dimensionality of the KS model, but as we shall soon see, this appears to not be the case. Preliminary experiments indeed seemed to indicate that ALEBO's performance upon the KS model was insensitive to values of d between 5 and 20. With such preliminary discoveries in mind, we decided to set $d = 15$ for the presented results of ALEBO, in a bid to maximise the probability of containing an optimum, p_{opt}, which generally increases with d.

We now discuss the results of ALEBO, base BO and random search upon the KS model, for our frequentist and calibration experiments.

6.2 Numerical Results

We present the results of running our emulated frequentist calibration schemes in Figure 33, with all hyperparameters of the experiment confirmed in the figure caption. The mean loss over the 10 runs, along with the standard error, is reported in Table 7. First, let's discuss Figure 33. We can see that our base BO implementation (in red) consistently locates better minima than ALEBO and random search (in green and blue respectively). ALEBO performs better than random search, but it seems base BO has not begun to struggle with the dimensionality enough for ALEBO to best its performance. This is perhaps the largest surprise of this experiment. It is widely reported within the BO literature that a simple separable RBF GP, with an Expected Improvement acquisition function should struggle at dimensionalities exceeding 20, and yet, at 33 dimensions, it continues to best both random search and a state-of-the-art high-dimensional BO algorithm.

If we look to the loss values in Table 7, for frequentist calibration, we see that the mean loss is also (marginally) lower, which means the lower minima that base BO finds (on average) are better quality sample points, in that they are closer to the true parameter vector $\vec{\theta}_{emp}$. Just as done previously, some quick, approximate calculations can give an idea of how far away, on average, the resulting parameters are from $\vec{\theta}_{emp}$. Merely by inserting vectors, of a repeated value between zero and one, and of dimensionality n, into Equation 28, we can see how the loss values of Table 7 fare. If each parameter was 40% away from its true value, the loss incurred would be $\sqrt{0.4^2 \times 33}$, which is equal to 2.298. Similarly, a 30% loss per parameter incurs a total loss of $\sqrt{0.3^2 \times 33}$, which equals 1.723. Thus we can see that the calibration is approximately as good as that on the Franke-Westerhoff and Brock-Hommes models.

If we now look to the minima curves of our Bayesian calibration experiment, upon the KS model, with our emulation scheme (in Figure 34), we see that our base BO implementation again bests ALEBO and random search, when it comes to minima location. This time however, ALEBO performs even more poorly, in just about matching random search's performance. The loss values between base BO and ALEBO tell us that slightly better minima are also recovered, again between 30 and 40% away (on average) from their true values. While such poor performance of ALEBO wasn't expected, the continued performance of base BO, even in a 33-dimensional parameter space, is an equivalent surprise. Future work should continue to investigate the failure mode of such base BO implementations in high dimensions.

ALEBO's average, if not slightly sub-par, performance could be due to a number of reasons. The dimensionality of the KS model may still not be high enough to warrant such subspace-embedding methods being used. The effective dimensionality of the model may also be higher than expected, which could be quantitatively investigated through sensitivity analysis procedures on the KS model (if this hasn't already been completed by Dosi's group, such as was done in [54] for a similar model). Similarly, quantifying the interactivity between the parameters would help determine whether additive HDBO methods would be worth investigating upon the KS model.

A large number of alternative methods have also been proposed in this thesis, which

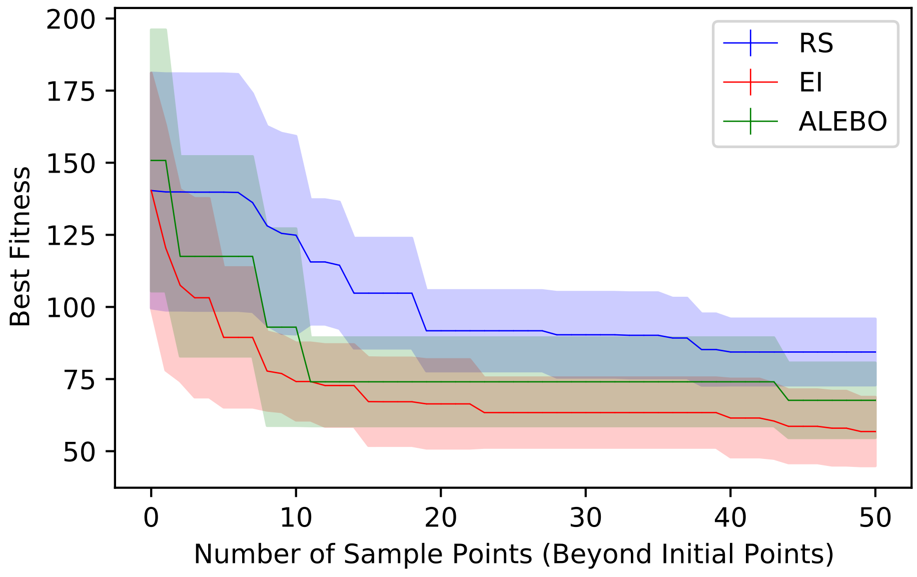

Figure 33: The results of our frequentist calibration experiment upon the KS macroeconomic model. The experiment was emulated via ALEBO, base Bayesian optimisation and finally sampled via random search. 25 initial sample points were ran, followed by 50 sequential points over 10 runs. Each sample point was repeated 10 times, and the median of the resulting fitnesses was taken (to control for strongly non-Gaussian fitnesses about each sample point). An embedding dimensionality of $d = 15$ was used for ALEBO. Note the continued performance of base BO, and ALEBO's performance between that of base BO random search. The mean minimum fitness from over the 10 runs, at each timestep, is plotted, alongside the standard error at each sample point.

could improve further upon base BO. Of these, heteroscedastic GPs are an immediate candidate. Utilising information about the noise across the sample space could be very fruitful, given the non-Gaussian sample point noise that was observed for the Brock-Hommes and KS models.

Model	Method	Mean Loss	Standard Error
KS	Frequentist Calibration - BO	1.976	0.188
KS	Frequentist Calibration - ALEBO	2.018	0.243
KS	Bayesian Calibration - BO	2.037	0.241
KS	Bayesian Calibration - ALEBO	2.141	0.349

Table 7: Final loss values for the KS macroeconomic model, across our frequentist and Bayesian calibration experiments. Base BO under frequentist calibration produced the best mean loss, but marginally. Most losses seem to be within each other's standard error.

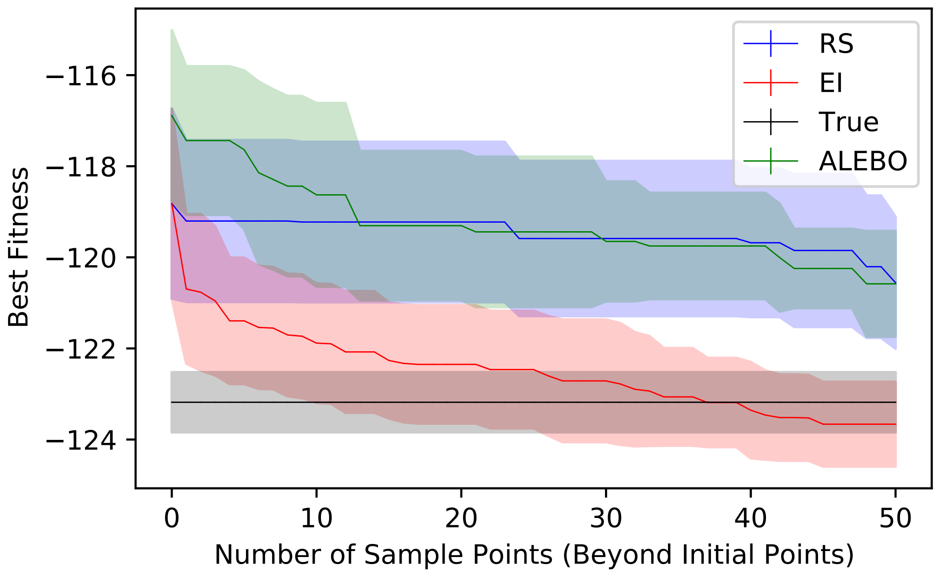

Figure 34: The results of our Bayesian calibration experiment upon the KS macroeconomic model. The experiment was emulated via ALEBO, base Bayesian optimisation and finally sampled via random search. 25 initial sample points were ran, followed by 50 sequential points over 10 runs. Each sample point was repeated 10 times, and the median of the resulting fitnesses was taken (to control for strongly non-Gaussian fitnesses about each sample point). An embedding dimensionality of $d = 15$ was used for ALEBO. Note the continued performance of base BO, and ALEBO's performance being comparable to that of random search. The mean minimum fitness from over the 10 runs, at each timestep, is plotted, alongside the standard error at each sample point.

Chapter 7

Conclusions and Future Work

In this thesis, we sought to replicate two leading strands of agent-based model (ABM) calibration literature, frequentist and Bayesian, and extend both schemes to models of increasing dimensionality and runtime. Such models have recently found their way into the public sphere, given the Covid-19 crisis [7], and creating tractable calibration and sensitivity analysis schemes, which prioritise sample-efficiency, are of utmost importance.

We begin with an overview of both frequentist and Bayesian calibration schemes, within which we propose an emulation methodology for recent Bayesian calibration procedures, themselves first proposed by Grazzini in [25]. The emulation scheme we propose is adapted from applications upon approximate Bayesian computation experiments, which hold many similarities to such Bayesian calibration methods. By combining this with previous emulation work in frequentist methods, we attempt to unify both schemes in an emulation methodology designed with sample-efficiency as a priority. Thus, we hope our work can bring both frequentist and Bayesian calibration schemes closer to working upon high-dimensional ABMs with long runtimes, which is an open problem currently.

We emulate via Gaussian processes and Bayesian optimisation, and show that such an emulation method consistently outperforms random search in minima location, across all calibration experiments upon the AR(1), Franke-Westerhoff, Brock-Hommes and KS ABMs, the last of which is actively used for macroeconomic research at the University of Pisa [15].

While our emulation scheme is able to obtain better minima on our constructed frequentist and Bayesian fitness surfaces, the parameter values that are retrieved (in our internal calibration experiments) are of modest quality. Our emulation scheme can optimise fitness surfaces in a sample-efficient manner, but such fitness surfaces don't seem to yield parameter sets satisfactorily close to true parameter sets, which we define prior to the experiments.

We couple such experiments with a literature review of previous emulation work in ABM calibration (and other simulatory models more generally), and discover a multitude of previous applications, across a number of fields. History Matching from the cosmology literature is one particularly interesting high-dimensional emulation method, which could be of interest for subsequent works [72]. Heteroscedastic Gaussian processes are also a very valid vein for future emulation research, given the non-Gaussian noise observed for the Brock-Hommes and KS models in this thesis [56]. Multi-output Gaussian processes are also another very promising area, providing more information about fitness surfaces for minimal computational cost, which is of utmost importance in emulation [80].

We end by discussing high-dimensional Bayesian optimisation methods, which seek to impose structure upon the parameter space, from a priori knowledge such as parameter independence and low effective dimensionalities of the space. We implement a leading method which assumes the latter; ALEBO [33]. This method seeks to embed the parameter space in one of lower dimensionality, thus aiding theorised performance restrictions for acquisition function and kernel hyperparameter optimisation within our Bayesian optimisation procedure. However, performance upon the KS model, in a 33-dimensional space of its larger input domain, is not found to be better than base implementations of Bayesian optimisation. This is a surprise, given the strongly documented failure mode of standard Bayesian optimisation in dimensions greater than 15-20.

Nevertheless, through our unifying emulation framework, and the set of extensions to base Bayesian optimisation which we outline, it is hoped that this thesis serves as an aid to future researchers, who seek to create tractable calibration and sensitivity analysis schemes upon simulatory models of large input dimensionality, expensive runtime and ambitious scope.

References

[1] C. M. Macal, "Everything you need to know about agent-based modelling and simulation," *Journal of Simulation*, vol. 10, pp. 144–156, 2016.

[2] J. Neumann, A. W. Burks, *et al.*, *Theory of self-reproducing automata*, vol. 1102024. University of Illinois press Urbana, 1966.

[3] M. Games, "The fantastic combinations of john conways new solitaire game life by martin gardner," *Scientific American*, vol. 223, pp. 120–123, 1970.

[4] T. C. Schelling, "Models of segregation," *The American Economic Review*, vol. 59, no. 2, pp. 488–493, 1969.

[5] J. M. Epstein and R. Axtell, *Growing artificial societies: social science from the bottom up.* Brookings Institution Press, 1996.

[6] G. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, 1965.

[7] N. Ferguson *et al.*, "impact of non-pharmaceutical interventions (npis) to reduce covid-19 mortality and healthcare demand."

[8] G. Dosi, G. Fagiolo, and A. Roventini, "Schumpeter meeting keynes: A policy-friendly model of endogenous growth and business cycles," *Journal of Economic Dynamics and Control*, vol. 34, pp. 1748–1767, 2010.

[9] R. G. Bower, I. Vernon, M. Goldstein, A. J. Benson, C. G. Lacey, C. M. Baugh, S. Cole, and C. S. Frenk, "The parameter space of galaxy formation," *Monthly Notices of the Royal Astronomical Society*, vol. 407, pp. 2017–2045, 08 2010.

[10] K. Rupp, "40 Years of Microprocessor Trend Data." `https://ourworldindata.org/grapher/transistors-per-microprocessor/`, 2017.

[11] D. Platt, "A comparison of economic agent-based model calibration methods," *Journal of Economic Dynamics and Control*, vol. 113, p. 103859, 2020.

[12] R. Franke and F. Westerhoff, "Structural stochastic volatility in asset pricing dynamics: Estimation and model contest," *Journal of Economic Dynamics and Control*, vol. 36, pp. 1193–1211, 2012.

[13] D. Platt, "Bayesian estimation of economic simulation models using neural networks," 2019.

[14] W. A. Brock and C. H. Hommes, "Heterogeneous beliefs and routes to chaos in a simple asset pricing model," *Journal of Economic Dynamics and Control*, vol. 22, pp. 1235 – 1274, 1998.

[15] G. Dosi, M. Napoletano, A. Roventini, and T. Treibech, “Micro and macro policies in the keynes + schumpeter evolutionary models,” *Journal of Evolutionary Economics*, vol. 27, p. 6390, 2017.

[16] M. Pereira *et al.*, “Lsd.” `https://github.com/SantAnnaKS/LSD`, 2020.

[17] V. Grimm, “Pattern-oriented modeling of agent-based complex systems: Lessons from ecology,” *Science*, vol. 310, 5750, pp. 987–991, 2005.

[18] J. C. Helton and F. J. Davis, “Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems,” *Reliability Engineering and System Safety*, vol. 81, pp. 23–69, 2003.

[19] T. Cioppa and T. Lucas, “Efficient nearly orthogonal and space-filling latin hypercubes,” *Technometrics*, vol. 49, pp. 45–55, 2007.

[20] S. Barde, “A practical, accurate, information criterion for nth order markov processes,” *Computational Economics*, vol. 50, pp. 281–324, 2017.

[21] F. Lamperti, “An information theoretic criterion for empirical validation of simulation models,” *Econometrics and Statistics*, vol. 5, pp. 83–106, 2018.

[22] R. Franke, “Applying the method of simulated moments to estimate a small agent-based asset pricing model,” *Journal of Empirical Finance*, vol. 16, pp. 804–815, 2009.

[23] Z. Chen and T. Lux, “Estimation of sentiment effects in financial markets: A simulated method of moments approach,” *Computational Economics*, vol. 52, pp. 711–714, 2016.

[24] J. Grazzini, M. G. Richiardi, and M. Tsionas, “Rising to the challenge: Bayesian estimation and forecasting techniques for macroeconomic agent-based models,” *Cesifo Working Papers*, 2019.

[25] J. Grazzini, M. G. Richiardi, and M. Tsionas, “Bayesian estimation of agent-based models,” *Journal of Economic Dynamics and Control*, 2017.

[26] M. U. Gutmann and J. Corander, “Bayesian optimization for likelihood-free inference of simulator-based statistical models,” *Journal of Machine Learning Research*, vol. 17, pp. 1–47, 2016.

[27] F. Lamperti, A. Roventini, and A. Sani, “Agent-based model calibration using machine learning surrogates,” *Journal of Economic Dynamics and Control*, 2018.

[28] L. Mones, N. Bernstein, and G. Csányi, “Exploration, sampling, and reconstruction of free energy surfaces with gaussian process regression,” *Journal of Chemical Theory and Computation*, vol. 12, no. 10, pp. 5100–5110, 2016.

[29] M. S. Caywood, D. M. Roberts, J. B. Colombe, H. S. Greenwald, and M. Z. Weiland, “Gaussian process regression for predictive but interpretable machine learning models: An example of predicting mental workload across tasks,” *Frontiers in Human Neuroscience*, vol. 10, p. 647, 2017.

[30] S. Greenhill, S. Rana, S. Gupta, P. Vellanki, and S. Venkatesh, “Bayesian optimization for adaptive experimental design: A review,” *IEEE Access*, vol. 8, pp. 13937–13948, 2020.

[31] O. Oyebamiji, D. Wilkinson, B. Li, P. Jayathilake, P. Zuliani, and T. Curtis, "Bayesian emulation and calibration of an individual-based model of microbial communities," *Journal of Computational Science*, vol. 30, pp. 194–208, 2019.

[32] I. Vernon, S. E. Jackson, and J. A. Cumming, "Known boundary emulation of complex computer models," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 7, no. 3, pp. 838–876, 2019.

[33] B. Letham, R. Calandra, A. Rai, and E. Bakshy, "Re-examining linear embeddings for high-dimensional bayesian optimization," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 1546–1558, Curran Associates, Inc., 2020.

[34] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning.* MIT Press, 2006.

[35] M. Seeger, "Gaussian processes for machine learning," *International Journal of Neural Systems*, vol. 14, pp. 69–106, 2004.

[36] R. B. Gramacy, *Surrogates.* CRC Press, 2020.

[37] M. L. Eaton, "Multivariate statistics: a vector space approach.," 1983.

[38] M. L. Stein, *Interpolation of Spatial Data.* Springer-Verlag, New York, NY, 1999.

[39] N. Durrande, D. Ginsbourger, and O. Roustant, "Additive covariance kernels for high-dimensional gaussian process modelling," *Annales de la Faculté des sciences de Toulouse : Mathématiques*, 2012.

[40] B. Letham, B. Karrer, G. Ottoni, and E. Bakshy, "Efficient tuning of online systems using Bayesian optimization." `https://research.fb.com/efficient-tuning-of-online-systems-using-bayesian-optimization/`, 2018.

[41] M. M. Khajah, B. D. Roads, R. V. Lindsey, Y.-E. Liu, and M. C. Mozer, *Designing Engaging Games Using Bayesian Optimization*, p. 55715582. New York, NY, USA: Association for Computing Machinery, 2016.

[42] E. Brochu, T. Brochu, and N. de Freitas, "A bayesian interactive optimization approach to procedural animation design," SCA '10, (Goslar, DEU), p. 103112, Eurographics Association, 2010.

[43] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, pp. 455–492, 1998.

[44] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *CoRR*, vol. abs/1012.2599, 2010.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[46] M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, "Botorch: A framework for efficient monte-carlo bayesian optimization," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[47] A. N. Kolmogorov, W. L. Doyle, and I. Selin, *Interpolation and extrapolation of stationary random sequences.* Santa Monica, CA: RAND Corporation, 1962.

[48] N. Wiener, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series.* The MIT Press, 1964.

[49] G. Matheron, "The intrinsic random functions and their applications," *Advances in Applied Probability*, vol. 5, no. 3, p. 439468, 1973.

[50] D. Krige, *A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand.* 1951.

[51] J. P. Kleijnen, "Regression and kriging metamodels with their experimental designs in simulation: A review," *European Journal of Operational Research*, vol. 256, no. 1, pp. 1–16, 2017.

[52] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and Analysis of Computer Experiments," *Statistical Science*, vol. 4, no. 4, pp. 409 – 423, 1989.

[53] I. Salle and M. Yildizoglu, "Efficient sampling and meta-modeling for computational economic models," *Computational Economics, 44:507-536*, 2014.

[54] G. Dosi, M. C. Pereira, and M. E. Virgillito, "On the robustness of the fat-tailed distribution of firm growth rates: a global sensitivity analysis," *Journal of Economic Interaction and Coordination*, 2016.

[55] S. Barde and S. van der Hoog, "An empirical validation protocol for large-scale agent-based models," *Bielefeld Working Papers in Economics and Management No. 04*, 2017.

[56] M. Binois, R. B. Gramacy, and M. Ludkovski, "Practical heteroscedastic gaussian process modeling for large simulation experiments," *Journal of Computational and Graphical Statistics*, vol. 27, p. 808821, Jul 2018.

[57] L. Bargigli, L. Riccetti, A. Russo, and M. Gallegati, "Network calibration and metamodeling of a financial accelerator agent based model," *Journal of Economic Interaction and Coordination*, vol. 15, pp. 413–440, April 2020.

[58] D. Kim, T. Yun, and I. Moon, "Automatic calibration of dynamic and heterogeneous parameters in agent-based model," *CoRR*, vol. abs/1908.03309, 2019.

[59] M. Tran, M. Ngo, D. Pham-Hi, and M. Bui, "Bayesian calibration of hyperparameters in agent-based stock market," in *2020 RIVF International Conference on Computing and Communication Technologies (RIVF)*, pp. 1–6, 2020.

[60] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems* (J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, eds.), vol. 24, Curran Associates, Inc., 2011.

[61] G. M. Dancik, D. E. Jones, and K. S. Dorman, "Parameter estimation and sensitivity analysis in an agent-based model of leishmania major infection," *Journal of Theoretical Biology*, vol. 262, no. 3, pp. 398–412, 2010.

[62] L. Willem, S. Stijven, E. Vladislavleva, J. Broeckhove, P. Beutels, and N. Hens, "Active learning to understand infectious disease models and improve policy making," *PLOS Computational Biology*, vol. 10, pp. 1–10, 04 2014.

[63] O. Oyebamiji, D. Wilkinson, P. Jayathilake, T. Curtis, S. Rushton, B. Li, and P. Gupta, "Gaussian process emulation of an individual-based model simulation of microbial communities," *Journal of Computational Science*, vol. 22, pp. 69–84, 2017.

[64] A. Fadikar, D. Higdon, J. Chen, B. Lewis, S. Venkatramanan, and M. Marathe, "Calibrating a stochastic, agent-based model using quantile-based emulation," *SIAM/ASA Journal on Uncertainty Quantification, Vol. 6, No. 4, pp 1685-1706*, 2018.

[65] C. N. Davis, T. D. Hollingsworth, Q. Caudron, and M. A. Irvine, "The use of mixture density networks in the emulation of complex epidemiological individual-based models," *PLOS Computational Biology*, vol. 16, pp. 1–16, 03 2020.

[66] I. Andrianakis, I. R. Vernon, N. McCreesh, T. J. McKinley, J. E. Oakley, R. N. Nsubuga, M. Goldstein, and R. G. White, "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, pp. 1–18, 01 2015.

[67] I. Andrianakis, N. McCreesh, I. Vernon, T. McKinley, J. Oakley, R. Nsubuga, M. Goldstein, and R. White, "Efficient history matching of a high dimensional individual based hiv transmission model," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 5, 03 2017.

[68] M. D. Stevenson, J. Oakley, and J. B. Chilcott, "Gaussian process modeling in conjunction with individual patient simulation modeling: A case study describing the calculation of cost-effectiveness ratios for the treatment of established osteoporosis," *Medical Decision Making*, vol. 24, no. 1, pp. 89–100, 2004. PMID: 15005958.

[69] K. Rojnik and K. Navernik, "Gaussian process metamodeling in bayesian value of information analysis: A case of the complex health economic model for breast cancer screening," *Value in Health*, vol. 11, no. 2, pp. 240–250, 2008.

[70] H. R. Parry, C. J. Topping, M. C. Kennedy, N. D. Boatman, and A. W. Murray, "A bayesian sensitivity analysis applied to an agent-based model of bird population response to landscape change," *Environmental Modelling and Software*, vol. 45, pp. 104–115, 2013. Thematic Issue on Spatial Agent-Based Models for Socio-Ecological Systems.

[71] E. Gengembre, B. Ladevie, O. Fudym, and A. Thuillier, "A kriging constrained efficient global optimization approach applied to low-energy building design problems," *Inverse Problems in Science and Engineering*, vol. 20, no. 7, pp. 1101–1114, 2012.

[72] R. G. Bower, M. Goldstein, and I. Vernon, "Galaxy formation: a Bayesian uncertainty analysis," *Bayesian Analysis*, vol. 5, no. 4, pp. 619 – 669, 2010.

[73] F. Leclercq, "Bayesian optimization for likelihood-free cosmological inference," *Physical Review D*, vol. 98, Sep 2018.

[74] C. Steinruecken, E. Smith, D. Janz, J. Lloyd, and Z. Ghahramani, *The Automatic Statistician*, pp. 161–173. Cham: Springer International Publishing, 2019.

[75] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[76] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Gaussian process bandits without regret: An experimental design approach," *CoRR*, vol. abs/0912.3995, 2009.

[77] P. Goldberg, C. Williams, and C. Bishop, "Regression with input-dependent noise: A gaussian process treatment," in *Advances in Neural Information Processing Systems* (M. Jordan, M. Kearns, and S. Solla, eds.), vol. 10, MIT Press, 1998.

[78] B. Ankenman, B. L. Nelson, and J. Staum, "Stochastic kriging for simulation metamodeling," *Oper. Res.*, vol. 58, p. 371382, Mar. 2010.

[79] E. Snelson and Z. Ghahramani, "Local and global sparse gaussian process approximations," in *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics* (M. Meila and X. Shen, eds.), vol. 2 of *Proceedings of Machine Learning Research*, (San Juan, Puerto Rico), pp. 524–531, PMLR, 21–24 Mar 2007.

[80] H. Liu, J. Cai, and Y.-S. Ong, "Remarks on multi-output gaussian process regression," *Knowledge-Based Systems*, vol. 144, pp. 102–121, 2018.

[81] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, "Scalable global optimization via local bayesian optimization," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[82] J. Gardner, C. Guo, K. Weinberger, R. Garnett, and R. Grosse, "Discovering and Exploiting Additive Structure for Bayesian Optimization," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (A. Singh and J. Zhu, eds.), vol. 54 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 1311–1319, PMLR, 20–22 Apr 2017.

[83] K. Kandasamy, J. Schneider, and B. Poczos, "High dimensional bayesian optimisation and bandits via additive models," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 295–304, PMLR, 07–09 Jul 2015.

[84] Z. Wang, C. Li, S. Jegelka, and P. Kohli, "Batched high-dimensional Bayesian optimization via structural kernel learning," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, (International Convention Centre, Sydney, Australia), pp. 3656–3664, PMLR, 06–11 Aug 2017.

[85] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," *Contemporary Mathematics*, vol. 26, 1984.

[86] A. Nayebi, A. Munteanu, and M. Poloczek, "A framework for bayesian optimization in embedded subspaces," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 4752–4761, PMLR, 09–15 Jun 2019.

[87] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. De Freitas, "Bayesian optimization in a billion dimensions via random embeddings," *Journal of Artificial Intelligence Research*, vol. 55, p. 361387, Jan. 2016.

[88] R. Garnett, M. A. Osborne, and P. Hennig, "Active learning of linear embeddings for gaussian processes," in *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, p. 230239, 2014.

[89] I. Delbridge, D. Bindel, and A. G. Wilson, "Randomly projected additive Gaussian processes for regression," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 2453–2463, PMLR, 13–18 Jul 2020.

[90] G. Dosi, M. C. Pereira, A. Roventini, and M. E. Virgillito, “What if supply-side policies are not enough? the perverse interaction of flexibility and austerity,” *Journal of Economic Behavior and Organization*, 2019.

Appendices

Appendix A. Bayesian Calibration

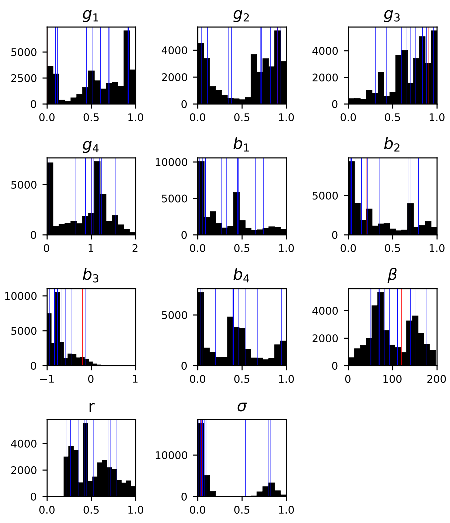

Figure 35: Histograms from our Bayesian calibration experiment, as outlined in Algorithm 2, upon the Brock-Hommes model. In each subplot, we plot the joint histogram, over 10 MCMC chains, for each denoted parameter. The true value of each parameter, within $\vec{\theta}_{emp}$, is denoted by the red line in each subplots, and the mean of each chain (minus burn-in) is denoted in blue. Note that β's true value is one. Less success is observed than in the one-dimensional Bayesian calibration of the AR(1) model.

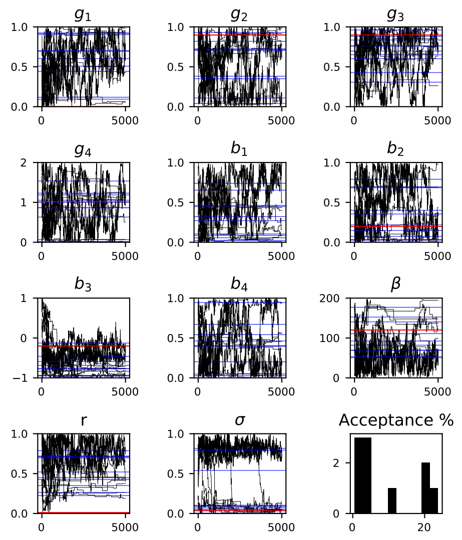

Figure 36: Sample paths from our Bayesian calibration experiment, as outlined in Algorithm 2, upon the Brock-Hommes model. In each subplot, the sample paths of 10 MCMC chains are plotted, for each denoted parameter. The true value of each parameter, within $\vec{\theta}_{emp}$, is denoted by the red line in each subplot, and the mean of each chain (minus burn-in) in blue. We plot the acceptance rates of the 10 chains in the last subplot.

Appendix B. Sample Paths

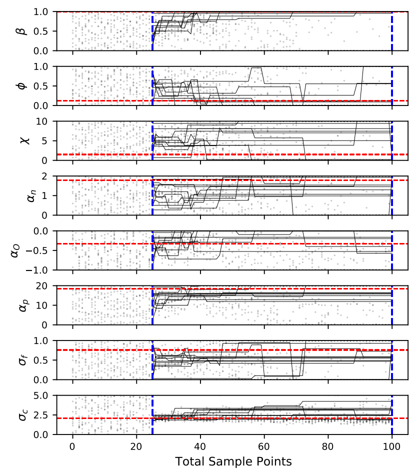

Figure 37: Sample paths for base BO, in the emulated frequentist calibration experiment of Figure 26, upon the Franke-Westerhoff model.

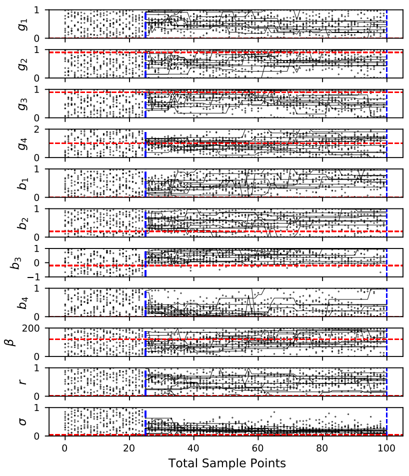

Figure 38: Sample paths for base BO, in the emulated frequentist calibration experiment of Figure 27, upon the Brock-Hommes model.

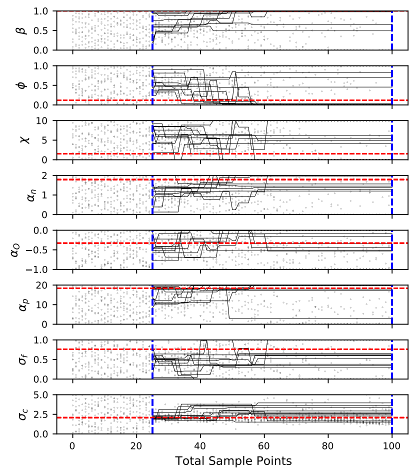

Figure 39: Sample paths for base BO, in the emulated Bayesian calibration experiment of Figure 29, upon the Franke-Westerhoff model.

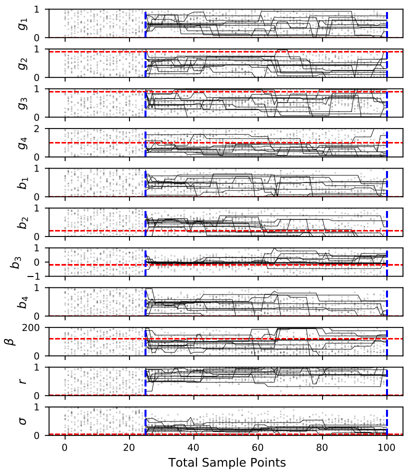

Figure 40: Sample paths for base BO, in the emulated Bayesian calibration experiment of Figure 30, upon the Brock-Hommes model.

Appendix C. Loss Figures

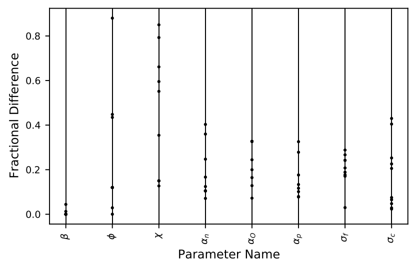

Figure 41: Fractional differences, for Bayesian optimisation, between each element of $\vec{\theta}^*$ (our selected parameter vector), and the accompanying element of $\vec{\theta}_{emp}$ (the ground-truth parameter vector), for each run of the emulated frequentist calibration experiment, upon the Franke-Westerhoff model, in Figure 26. Note, in many of these scatter plots, there are less points than the number of experiment repetitions. This is because, when we examine the parameter values, some end up being set to the maximum or minimum of the parameter boundary, and so when the fractional difference to $\vec{\theta}_{emp}$ is taken, multiple sample points sometimes possess the same value.

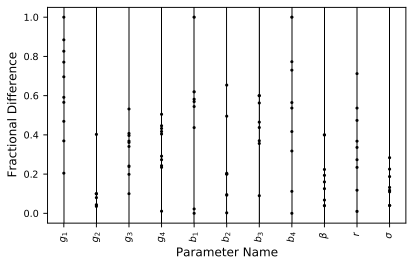

Figure 42: Fractional differences, for Bayesian optimisation, between each element of $\vec{\theta}^*$ (our selected parameter vector), and the accompanying element of $\vec{\theta}_{emp}$ (the ground-truth parameter vector), for each run of the emulated frequentist calibration experiment, upon the Brock-Hommes model, in Figure 27.

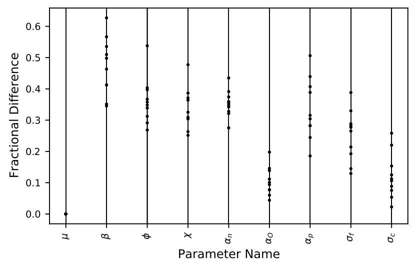

Figure 43: Fractional differences between each element of $\vec{\theta}^*$ (our selected parameter vector), and the accompanying element of $\vec{\theta}_{emp}$ (the ground-truth parameter vector), for each run of the Bayesian calibration experiment via MCMC, upon the Franke-Westerhoff model, in Figure 13.

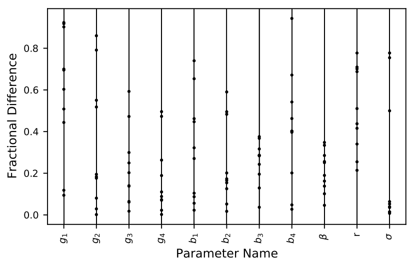

Figure 44: Fractional differences between each element of $\vec{\theta}^*$ (our selected parameter vector), and the accompanying element of $\vec{\theta}_{emp}$ (the ground-truth parameter vector), for each run of the Bayesian calibration experiment via MCMC, upon the Brock-Hommes model, in Figure 35.

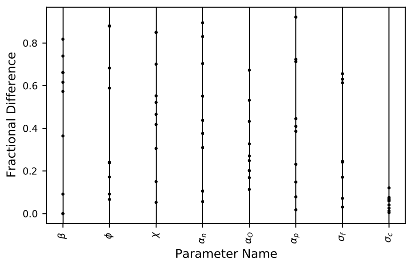

Figure 45: Fractional differences, for Bayesian optimisation, between each element of $\vec{\theta}^*$ (our selected parameter vector), and the accompanying element of $\vec{\theta}_{emp}$ (the ground-truth parameter vector), for each run of the emulated Bayesian calibration experiment, upon the Franke-Westerhoff model, in Figure 29.

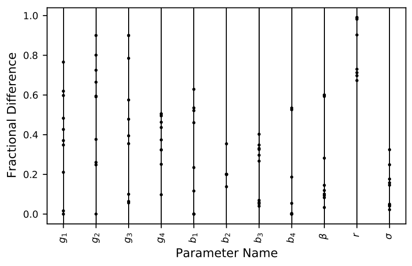

Figure 46: Fractional differences, for Bayesian optimisation, between each element of $\vec{\theta}^*$ (our selected parameter vector), and the accompanying element of $\vec{\theta}_{emp}$ (the ground-truth parameter vector), for each run of the emulated Bayesian calibration experiment, upon the Brock-Hommes model, in Figure 30.

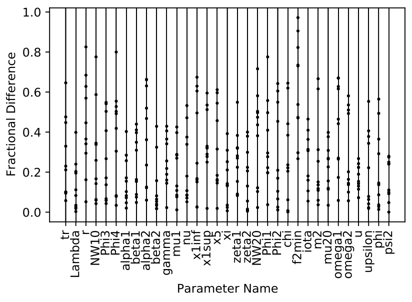

Figure 47: Fractional differences, for Bayesian optimisation, between each element of $\vec{\theta}^*$ (our selected parameter vector), and the accompanying element of $\vec{\theta}_{emp}$ (the ground-truth parameter vector), for each run of the emulated frequentist calibration experiment, upon the KS model, in Figure 33.

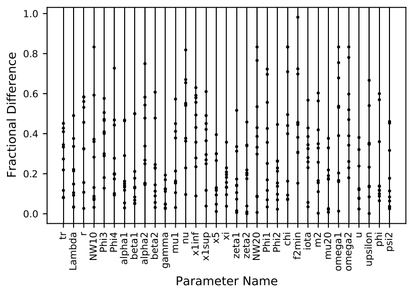

Figure 48: Fractional differences, for ALEBO, between each element of $\vec{\theta}^*$ (our selected parameter vector), and the accompanying element of $\vec{\theta}_{emp}$ (the ground-truth parameter vector), for each run of the emulated frequentist calibration experiment, upon the KS model, in Figure 33.

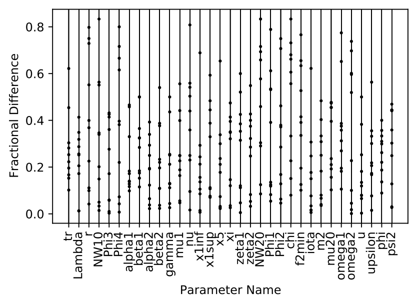

Figure 49: Fractional differences, for base BO, between each element of $\vec{\theta}^*$ (our selected parameter vector), and the accompanying element of $\vec{\theta}_{emp}$ (the ground-truth parameter vector), for each run of the emulated Bayesian calibration experiment, upon the KS model, in Figure 34.

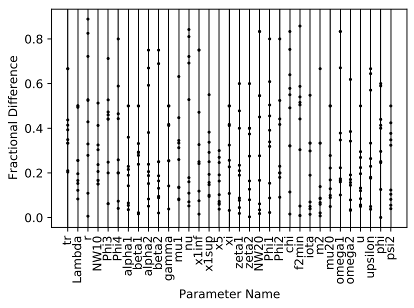

Figure 50: Fractional differences, for ALEBO, between each element of $\vec{\theta}^*$ (our selected parameter vector), and the accompanying element of $\vec{\theta}_{emp}$ (the ground-truth parameter vector), for each run of the emulated Bayesian calibration experiment, upon the KS model, in Figure 34.