Efficient Parameter Inference with Bayesian Optimization for Agent-Based Biological Simulations

Vincenzo Incutti
under the supervision of Dr. Andrea Y. Weiße

4th Year Project Report
Artificial Intelligence and Software Engineering
School of Informatics
University of Edinburgh

2022

Abstract

This thesis explores the performance of Bayesian Optimization as a method to infer parameters of agent-based biological simulations. Its motivation lies with the high computational costs posed by traditional methods adopted for parameter inference in the field of computational biology, such as Markov Chain Monte Carlo. Bayesian Optimization is by design suited to optimize expensive black-box functions by minimising the number of evaluations. I compare different agent-based modelling software packages for multi-cellular simulations, identifying Bsim as the best alternative due to its balance between complexity and customizability. Bayesian Optimization is implemented and tested under a number of experimental conditions: increasing sampling range size, noisy function evaluations and multiple parameters to infer. This work shows promising initial results and lays the groundwork for further research and applicability of Bayesian Optimization to the field of systems and computational biology.

Acknowledgements

I would like to thank my supervisor, Dr. Andrea Y. Weiße, for the constant help and support throughout this work. She gave me the necessary guidance and reassurance to spark my interest in the field of computational biology, as well as being an amazing mentor.

I would also like to thank the entire group of wonderful people that Andrea brought together for our journal club and multidisciplinary discussions: Holly, Rohan, Ash, Elena, Sandy and Fiona. You showed me how much we can learn from each other and provided invaluable help in carrying out this work.

Finally, I would like to thank my family, friends and flatmates for being there throughout these challenging months and giving me the strength to always do my best.

Table of Contents

Chapter 1

Introduction

1.1 Context and Motivation

Recent advances in the field of molecular biology, such as accurate and fast genome sequencing techniques and high-throughput measurements, are providing extensive amounts of data. Not only has this allowed biologists to better understand the underpinnings of individual cells, it has also shed light on the behaviour of entire populations. In turn, this has renewed interest in a system-level understanding of biological processes, which focuses on the emergent properties that arise as a consequence of numerous individual cells interacting with each other. These traits, such as robustness to changing and adverse environmental conditions, are what allows species to withstand evolutionary pressure and survive.

In Systems Biology, mathematical models are used to represent biological processes. By converting the chemical reactions to systems of differential equations, one obtains a representation of the system that can be run *in silico*. Computer simulations provide predictions to be tested *in vitro*, and given the relatively lower cost of running them, significantly accelerate the process of developing and refining hypotheses.

The parameters used in computational models are often unknown and must be inferred from experimental data. A number of techniques to perform parameter inference are available, and much research has been devoted to efficient parameter determination in systems and computational biology. One of the most widely-adopted and successful techniques is Markov Chain Monte Carlo. Although accurate, this method has the major drawback of being significantly computationally intensive, requiring tens of thousands of iterations and ensemble methods to achieve satisfactory results. Given that the simulations of fairly complex biological models often contain many differential equations to be solved via numerical integration for each iteration, scalability quickly becomes an issue.

In this thesis, I propose Bayesian Optimization (BayesOpt) as an alternative method to perform parameter inference for biological simulations. BayesOpt has recently attracted much attention in the field of deep learning as a hyperparameter tuning method for neural networks. However, it has not seen many applications elsewhere. At its

core, BayesOpt is a global optimization method for expensive 'black-box' functions. Black-box here refers to the fact that we have no knowledge about the shape of the function, and therefore cannot rely on any particular property, e.g. its gradient, which is often the case for biological models. We would like to minimise the number of expensive evaluations required to obtain accurate parameter estimates. I provide quantitative evidence for the viability of BayesOpt to perform parameter inference, and test the algorithm under multiple conditions to measure its robustness and scalability.

1.2 Contributions

This thesis makes the following contributions:

- Critical review of multiple agent-based modelling software packages for multi-cellular simulations;
- Implementation of the Bayesian Optimization algorithm to perform parameter inference for multi-cellular simulations;
- Experiments with Bayesian Optimization parameter inference under multiple conditions: increasingly large sampling space, noisy function evaluations and multiple parameters inference.

An implementation of Markov Chain Monte Carlo for parameter inference was among the initial objectives outlined at the beginning of this work. While implemented, this has not been included in the analysis of the results for two reasons. First, no thorough testing has been conducted on the implementation. Second, no conclusive data has been obtained due to computational and time constraints under which this work has been carried out. I leave these as objectives for future research on the topic.

1.3 Thesis Structure

The remainder of this thesis is structured as follows:

- Chapter 2 provides both the biological and mathematical foundations for the following chapters. Section 2.1 presents the fields of systems and synthetic biology, while Section 2.2 shows how multi-cellular simulations are implemented *in silico* via agent-based modelling. Section 2.3 covers ordinary differential equations and how these are used in computational biology. Finally, Section 2.4 presents the task of parameter inference, with Bayesian inference covered in Section 2.4.1, Markov Chain Monte Carlo in Section 2.4.2 and Bayesian Optimization in Section 2.4.3.
- Chapter 3 illustrates the practical contributions of this thesis. First, a number of agent-based simulation software packages are analysed and compared in Section 3.1: Gro (Section 3.1.1), CompuCell3D (Section 3.1.2), CellModeller (Section 3.1.3) and Bsim (Section 3.1.4). Then, the implementation details of Bayesian Optimization (Section 3.2) and Markov Chain Monte Carlo (Section 3.3) are described. Finally, the experimental setup and the conditions under which Bayesian

Optimization has been tested are outlined: increasingly large sampling space (Section 3.4.1), noisy function evaluations (Section 3.4.2) and multiple parameters inference (Section 3.4.3).

- Chapter 4 presents the result of the experiments outlined in Section 3.4: increasingly large sampling space (Section 4.1), noisy function evaluations (Section 4.2) and multiple parameters inference (Section 4.3).
- Finally, Chapter 5 gives a summary of the topics discussed and the experiments undertaken in this thesis, leaving open questions for further research.

Chapter 2

Background and Literature Review

This chapter outlines the biological and mathematical concepts necessary to understand further discussion in the thesis. Firstly, it provides an overview of the field of systems biology and its practical implications in the design of synthetic molecular machinery. It then describes agent-based modelling as a tool to simulate cell populations *in silico*, and how ordinary differential equations are used to represent intracellular dynamics. Finally, it focuses on parameter inference and how it is typically undertaken in systems biology, proposing Bayesian Optimization as an efficient alternative to traditional frameworks.

2.1 Systems and Synthetic Biology

Systems biology refers to the comprehensive analysis of the manner in which all the components of a biological system interact functionally over time. Such systems are more than a mere assembly of genes and proteins. Drawing the interconnections between these is a fundamental prerequisite to a broader understanding of complex biological networks, however it is analogous to sketching a static road map. Holistic knowledge can only be derived by observing the traffic patterns along the connections, how and why these emerge, and how we can manipulate them [16].

A system-oriented approach is ideologically and pragmatically opposed to a reductionist one. According to *reductionism*, the behaviour of a biological system can be explained by the properties of its constituent components alone. While the reductionist paradigm has dominated the field of biology and allowed scientists to achieve breakthrough findings in identifying genes, molecules and bio-chemical reactions, it falls short of explanations that link low-level processes to higher biological phenomena [17]. Systems biology bridges this gap by studying the evolution of a biological system from a global perspective, incorporating into its analysis observed characteristics such as stochasticity, sensitivity to initial conditions and chaotic behaviour. Consequently, it allows to explain important evolutionary properties such as:

- *Emergence*. So-called 'emergent properties' arise from the interplay between multiple components. An understanding of such properties cannot be gleaned

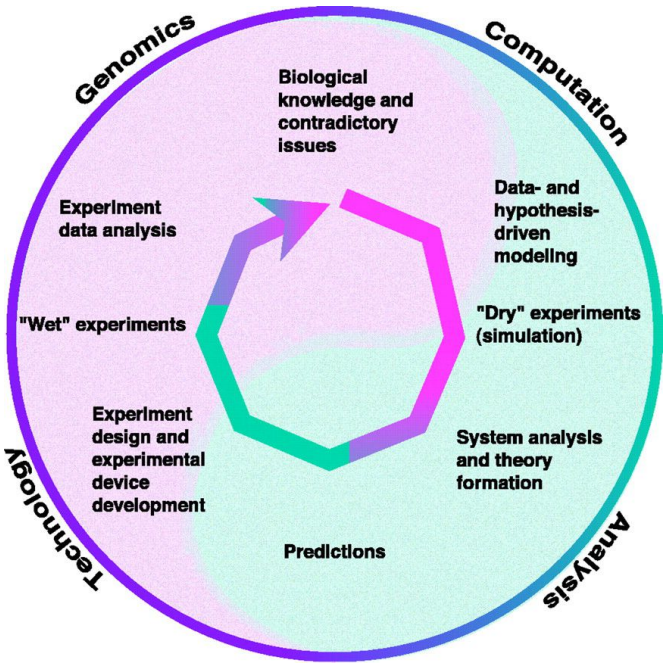

Figure 2.1: Combining Systems and Synthetic biology. Hypotheses of interest (e.g. the behaviour of a bacterial population after modifying a metabolic pathway) are used to create computational models. These are tested *in silico*, or via 'dry' experiments, which can either support or deny the set of assumptions embedded in the model. When computer simulations reveal inadequacies in the hypotheses, these are adjusted with the information gleaned from the predicted behaviours. The revised models are then used to design new 'wet' experiments to perform, whose output is analysed and used to refine understanding of the issue. Figure from [16].

from a reductionist approach. For example, one cannot understand the behaviour of water molecules simply by observing atoms of hydrogen and oxygen;

- *Robustness*. Positive and negative feedback loops allow biological systems to maintain phenotypic stability when perturbed by adverse environmental conditions, stochasticity, and genetic variations;
- *Modularity*. Functionalities are grouped into isolated collections of components, so that the failure of one does not spread to others, preventing catastrophic damage to the entire system [18].

A complete understanding of biological systems can be harnessed to modify their normal course of action for beneficial applications. The field of synthetic biology, by combining technology, engineering and biological sciences, aims to alter existing cellular machinery for a variety of purposes. Bacteria, due to their unicellular nature and short regeneration time, are particularly suited for engineering complex metabolic pathways. Particularly, engineered microbes have been extensively used to produce therapeutic proteins, industrial enzymes, small molecular pharmaceuticals, chemicals, biofuels, and materials [19]. Together, systems and synthetic biology provide a new research framework that combines 'dry' (*in silico* computer simulations) and 'wet'

(carried out in the lab) experiments and data to iteratively propose and amend hypotheses (see Fig. 2.1). This novel approach promises to significantly speed up the development cycle, both for theoretical research questions and practical use cases, as well as reducing the costs involved in running expensive experiments. Therefore, it is crucial to develop faithful computational descriptions to yield insightful observations.

Designing dry experiments requires mathematical models to describe the behaviour of the processes being investigated. Owing to our ability to observe and measure many diverse aspects of individual cells, much of the modelling in synthetic biology to date has focused on intracellular dynamics (e.g. variations in concentrations of proteins over time). As the reductionist school of thought became unable to explain the robustness and modularity shown by biological systems, the modelling paradigm began to shift to capture collective population-level features that extend beyond individual cells, thus becoming the perfect application for agent-based modelling.

2.2 Agent-Based Modelling

Agent-Based Modelling (ABM) is a computational modelling approach which captures emergent properties of dynamical systems by considering interactions between 'agents' and the environment. Agents can represent any entity of interest, such as a molecule, cell or multicellular organism, and each independently follows a prescribed set of rules. In a biological setting, these rules are often encoded as genetic circuits that drive cellular responses to particular stimuli. By simulating the behaviour of these virtual populations in realistic environments, it is possible to gain an understanding of how low-level cellular rules lead to the emergence of collective population-level behaviours [8].

ABM offers two advantages over traditional modelling frameworks. Firstly, it captures minor differences that exist or can arise between agents, such as the varying levels of expression of a particular protein across a population due to intracellular noise. These differences are not trivial, as they can be leveraged to achieve novel functions. In *Escherichia Coli* populations, for instance, cells can either be active or inactive depending on the concentration of stimulants, a property which improves the fitness of the population as a whole by ensuring that at least some cells can exploit changes in environmental conditions [20]. Other modelling approaches, by averaging out differences and assuming a uniform behaviour, fail to capture these features. The other major benefit of ABM is the ability to capture the multiplicity of interactions that agents can have. Direct physical interactions, where two agents meet, can be made more realistic by modifying them to be probabilistic, so that they do not always lead to the same behavioural responses. The environment itself can also act as a means for indirect interactions, such as as when pheromones are deposited to be sensed later by other individuals [21]. ABM easily incorporates this in the simulation, whereas traditional frameworks often struggle. A schematic overview of ABM can be seen in Figure 2.2.

Both agents and the environment can be specified via a collection of static properties (e.g. the cell walls' diffusivity rate or the dimensions of the space in which the agents are allowed to evolve) or dynamical ones. The latter are often in the form of systems

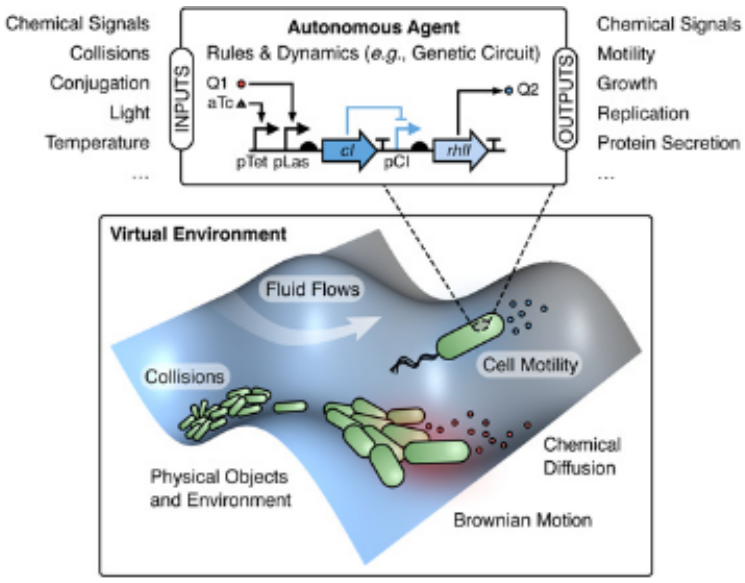

Figure 2.2: Agent-Based Modelling of cell populations. Agents' behaviour is determined by their internal state, the behaviour of other agents and the conditions of the surrounding environment. The system develops emergent properties as a result of the local interactions. The intracellular dynamics of each agent are specified by gene circuits with variable inputs such as chemical signals, collisions or temperature, and lead to different behaviours, e.g. replication, growth or protein secretion. The dynamics of gene circuitry is often specified as a system of ordinary differential equations (ODEs), specifying the rates of bio-chemical reactions. Figure from [8].

of ordinary differential equations (ODEs), representing the rates of change for the concentration of chemical species. At each timestep in the simulation, the ODE model contained in each agent is run with input values determined by the state of the system.

The model proposed in [2], for instance, is a system of 12 ODEs representing cellular processes competing for three types of finite resources: *cellular energy*, required to launch new biochemical processes, *ribosomes*, necessary to translate mRNA, and *proteome*, such that the expressing one type of protein reduces expression of others (Fig. 2.3). The model relates gene expression with growth rate, and growth rate with a growing population of cells. For this reason, it is particularly suited to an agent-based exploration approach.

2.3 Ordinary Differential Equations

The behaviour of agents in computational biology simulations is often described by a system of ordinary differential equations (ODEs). An ODE describes the rate of change of one or more dependent variables, y, as a function of a single independent variable, x, more commonly referred to as t, or time. In the context of computational and systems biology, ODEs are used the describe the concentrations of reactants used by chemical processes over time.

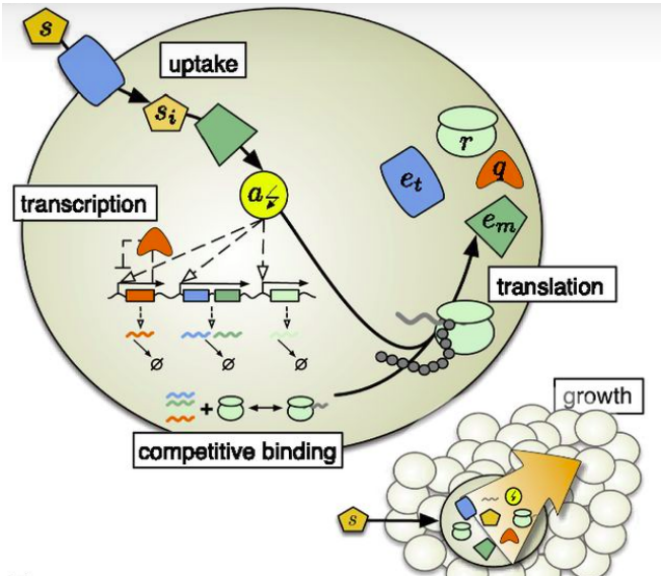

Figure 2.3: Model presented in [2]. ODEs are used to describe the intake of an external nutrient, s, which, after being internalised as s_i, is used to produce cellular energy, a, which fuels a variety of cellular processes such as transcription and translation. The inset shows how every cell runs its own ODE system and together they all contribute to the growth of the population as a whole.

Most published models of biochemical reactions are non-linear and closed form solutions are not available. Accordingly, numerical integration methods have to be employed to study them [14]. These typically take the form of an *initial value problem*, where the value of y is provided at t_0. The final solution is obtained iteratively over a specified number of timesteps, by applying the numerical method at time t_i to the result of the integration at t_{i-1}.

The timescales for the reaction rate of the molecular processes can differ by orders of magnitude [15], a phenomenon captured by so-called *stiff* ODEs. Stiffness occurs whenever there are slowly changing solutions combined with rapidly changing components. An example of a non-stiff and stiff ODE is shown in Figure 2.4 and Figure 2.5, respectively, illustrating the exact solutions for the van der Pol equation, defined as

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= \mu(1-y_1^2)y_2 - y_1. \end{aligned} \tag{2.1}$$

where y' refers to the first derivative of y.

The step size taken by an ODE solver is forced down to an unreasonably small level in comparison to the interval of integration when dealing with stiff systems. The step size can be so small that traversing a short time interval might require millions of evaluations, if not leading to a failure of the integration altogether. Fortunately, stiff ODE solvers that take this problem into account do exist and have been extensively benchmarked [14]. The choice of an appropriate ODE solving routine is particularly

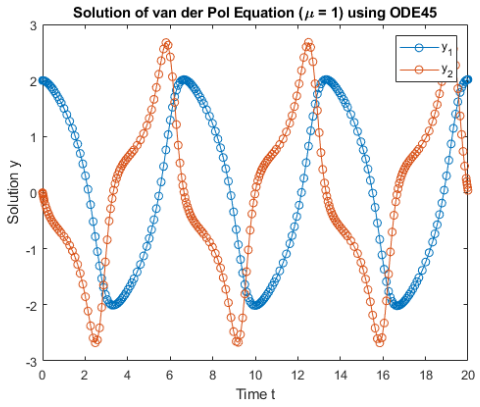

Figure 2.4: van der Pol equation with $\mu = 1$, solved with MATLAB's ode45 function [22]. Note how y_1 and y_2 change over the same timescale.

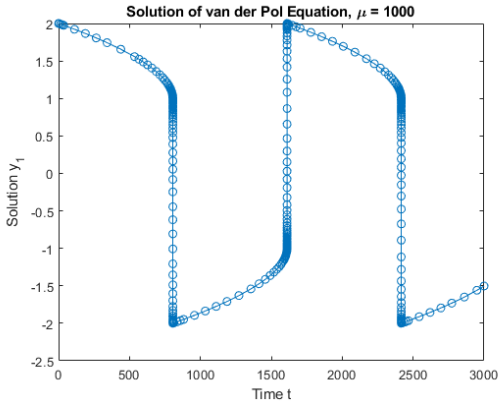

Figure 2.5: van der Pol equation with $\mu = 1000$, solved with MATLAB's ode15s function [23]. The same equation becomes stiff when changing the value of μ. Note how the behaviour of y_1 changes compared to its stiff counterpart with $\mu = 1$ in Fig. 2.4. y_2 is not plotted for scaling reasons.

important when the ODE system has to be simulated and solved a significant number of times, such as in the context of parameter inference.

2.4 Parameter Inference

The systems of ODEs used to describe the dynamics of biological processes introduce various unknown parameters that need to be estimated efficiently from a set of experimental observations. One of the major challenges faced by parameter inference in the context of computational systems biology is the limited size of the set of available observations, given the complexity of the models and the time and resources needed to run them. This problem is compounded by the fact that some species evolve over timescales orders of magnitude apart, a property represented by stiff ODEs. Furthermore, the data available frequently contains noise, caused by either the instruments used to capture it (e.g. the microscope's measurement error), or simply by the stochas-

ticity of the processes themselves.

Formally, we are given a system of ODEs representing the biochemical processes as follows

$$\frac{d\mathbf{x}}{dt} = \boldsymbol{f}(\mathbf{x}, t, \mathbf{p}), \tag{2.2}$$

where the state vector $\mathbf{x} = (x_1, x_2, ..., x_N)$ represents the concentrations of N species and $\mathbf{p} = (p_1, p_2, ..., p_K)$ are K parameters of the model [24]. We are also given the set of initial observations at time T_0

$$\mathbf{x}(T_0) = \mathbf{x}_0, \tag{2.3}$$

which is used to make an initial random guess as to what the values of the parameters $\mathbf{p}$ could be. From here, by repeatedly running the model and collecting observations, the parameter inference procedure refines its estimate of the values of the variables. This is represented as follows

$$\mathbf{y} = \boldsymbol{h}(x) + \varepsilon(t), \quad \varepsilon(t) \sim \mathcal{N}(\mu, \sigma^2), \tag{2.4}$$

where $\mathbf{y}$ are experimental observations consistent with the ODE system, which possibly contains noise ε of Gaussian type with mean μ and standard deviation σ. The output function $\boldsymbol{h}$ is defined as part of the experiment design, however it is often of the form

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \dots \\ \varepsilon_N \end{bmatrix}. \tag{2.5}$$

Assuming there is a set of parameters $\mathbf{p}^*$ such that $\boldsymbol{f}(\mathbf{x}, t, \mathbf{p}^*)$ is a sufficiently accurate mathematical description approximating reality, i.e. all the relevant knowledge about the biological processes is incorporated correctly into the vector function $\boldsymbol{f}$, we define a vector of discrepancies between the model values $\boldsymbol{p}^*$ and the experimental values $\hat{\boldsymbol{p}}$ as

$$\mathbf{e}(\hat{\boldsymbol{p}}) = |\boldsymbol{f}(\mathbf{x}, t, \mathbf{p}^*) - \boldsymbol{f}(\mathbf{x}, t, \hat{\boldsymbol{p}})|. \tag{2.6}$$

The task of parameter inference can then be expressed as finding the set of parameters $\boldsymbol{p}$ that minimizes $\mathbf{e}(\boldsymbol{p})$, often referred to as the *objective* function,

$$\boldsymbol{p} = \arg\min_{\boldsymbol{p}} \mathbf{e}(\boldsymbol{p}). \tag{2.7}$$

The task of minimising the objective function is called *optimization*, and although a variety of approaches exist to perform it, these can broadly be classified into two categories: *local* and *global* optimization methods.

A parameter vector $\hat{\mathbf{p}}$ is said to be a local minimizer of $\mathbf{e}$ if it gives the lowest obtainable objective function value in the neighbourhood of a starting point

$$\hat{\mathbf{p}}_{local} = \arg\min_{\boldsymbol{p}} \mathbf{e}(\boldsymbol{p}) \quad \forall ||\boldsymbol{p} - \hat{\mathbf{p}}_{start}|| < \delta, \quad \delta > 0. \tag{2.8}$$

A global optimizer, on the other hand, gives the lowest obtainable objective function value from an arbitrary starting point:

$$\hat{\mathbf{p}}_{global} = \arg\min_{\boldsymbol{p}} \mathbf{e}(\boldsymbol{p}) \quad \forall \mathbf{p} \text{ in the parameter space.} \tag{2.9}$$

Local optimizers work by computing the gradient of the objective function with respect to the parameter vector. The gradient ∇ is simply a vector containing the partial derivative of the function with respect to each p_i in $\boldsymbol{p}$

$$\nabla \mathbf{e}(\hat{\mathbf{p}}) = \left[\frac{\partial \mathbf{e}}{\partial \mathbf{p}} (\hat{\mathbf{p}}) \right]. \tag{2.10}$$

At its minimum, the gradient of the objective function with respect to the parameters vanishes, i.e. is equal to zero. Local optimization methods employ this fact to explore the parameter space by following the opposite direction of the steepest gradient, which is proven to converge to a local minimum [25].

Multiple issues arise when adopting local optimization in the context of systems biology. Firstly, the objectives to optimize are often 'black-box' functions, meaning that no assumptions about their shape can be made, including the computability of their gradient. This is even more emphasised in agent-based simulations by the stochastic nature of the interactions and evolution of individual agents and the entire population. Secondly, assuming that the gradients can be computed, the non-linearity of biological systems can lead to multimodal landscapes (more than one minimum are present) [26]. Local methods tend to get trapped in local minima and, although a number of high-performing software packages are available (Copasi [27], PottersWheel [28]), these often require multi-start strategies (repeatedly applying the local method starting from a number of different initial vectors) to address the presence of multiple minima [29]. When we consider the additional burden of numerically solving the system of ODEs for every simulation iteration, local optimization methods often become not viable due to their high computational costs. It is therefore desirable to minimise the number of stochastic simulations required to perform parametric inference for noisy biological systems [30].

Global optimization methods are designed to address the issues described above. They do not require knowledge of the gradients, only an order relation of the form $\mathbf{e}(\mathbf{p}_1) < \mathbf{e}(\mathbf{p}_2)$ for the points in the parameter space [31]. Certain global optimizers are stochastic in nature, meaning that although the theoretical global minimum cannot not be

reached, solutions in its vicinity can be located with relative efficiency. In practice, the solutions are satisfactory enough and can be achieved in modest computation times. Finally, global optimization methods do not required any transformation of the original problem, which can be thus treated as a black box [32]. This work focuses on Bayesian inference, a particular type of stochastic global optimization.

2.4.1 Bayesian Inference

Bayesian methods are a family of statistical modelling approaches used, among other applications, for parameter inference. They are particularly suited to the context of systems biology [3] due to the problem of parameter *identifiability*. Because of the high-dimensionality of the parameter space and the limited, noisy amounts of data available from expensive biological experiments, multiple sets of parameters equivalently describe the experimental data. When parameters are non-identifiable, a single parameter set is insufficient to describe the feasible space of parameters associated with the model [33].

Bayesian methods solve this problem by modelling the entire probability distribution of the unknown parameters, rather than just a point-estimate, given the experimental data. This allows both determination of best-fit parameters, as well as allowing the modeller to incorporate and prior information they might have about the parameters [34].

Bayesian inference estimates a set of parameters θ by merging the available (prior) information about the model with the information obtained by observing new data $\mathcal{D}$. This is represented by Bayes' theorem [4]:

$$\pi(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta)\pi(\theta)}{P(\mathcal{D})}. \tag{2.11}$$

Where the *prior* $\pi(\theta)$ and the *posterior* $\pi(\theta|\mathcal{D})$ summarise the available information about θ before and after considering the data $\mathcal{D}$, respectively. The other two terms are $P(\mathcal{D})$, the *marginal probability* of observing $\mathcal{D}$, and $P(\mathcal{D}|\theta)$, the *likelihood*. The former is a normalization constant, whereas the latter represents the probability of observing the data $\mathcal{D}$ given the current parameterization.

One commonly adopted instance of Bayesian methods in the field of systems biology is Markov Chain Monte Carlo. Example applications include parameter inference for influenza transmission [35] or finite resource allocation in molecular processes [2], illustrated in Section 2.2.

2.4.2 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is a sampling technique that generates a chain of parameters $\theta^1, \theta^2, ..., \theta^{d_s}$ by exploring the posterior distribution $\pi(\theta|\mathcal{D})$ [6]. The samples are generated form a Markov chain that serves as an approximation of the

distribution of interest, or *target*. It can be proven that given enough sampling the stationary distribution of the chain (a distribution π that does not change as more samples are obtained, i.e. $\pi = \pi\mathbf{P}$ for a given transition matrix $\mathbf{P}$) is the target [7].

Each iteration of an MCMC algorithm follows two steps. Firstly, a point θ^k in the parameter space is proposed by sampling it from a proposal distribution

$$\theta^k \sim Q(\theta^k, \theta^{k-1}). \tag{2.12}$$

The proposal density $Q(\theta^k, \theta^{k-1})$ depends on the previous chain member, θ^{k-1}, and might be any distribution from which we can sample, such as a Gaussian centered at θ^{k-1}.

In the second step, the proposal is either accepted or rejected based on whether the value of the posterior evaluated at the proposal is higher than its evaluation at the previous proposal. The rule is represented by the following

$$p_a = min\left\{1, \frac{P(\mathcal{D}|\theta^k)}{P(\mathcal{D}|\theta^{k-1})}\frac{\pi(\theta^k)}{\pi(\theta^{k-1})}\frac{Q(\theta^{k-1},\theta^k)}{Q(\theta^k,\theta^{k-1})}\right\}. \tag{2.13}$$

If the proposal is accepted, it is used as the reference point for the next iteration of the algorithm, otherwise it is discarded.

As already mentioned, the stationary distribution of the chain is the target, however it is not straightforward to determine the convergence of the chain at any instant t in time [57]. This is because the convergence is not to a point, but to a distribution of a sequence of generated values to another distribution. No single diagnostic tool exists to assess convergence, and several methods are used together to approximate convergence measurement [58]. However, the number of iterations recommended to be reasonably confident that MCMC has converged is highly computationally demanding, and not viable for commercial hardware [5]. This limitation is even more restricting when obtaining each sample is expensive, which, as already mentioned, is the case for complex ODE systems in computational biology.

Another consideration to be made on MCMC regards the choice of the proposal distribution. Because of the dependence of each sample on the previous one, the choice of an appropriate proposal distribution becomes crucial to minimise the correlation between samples which could lead to slower convergence or even becoming trapped in local optima during the search for global ones [59]. The hyperparameters of the proposal are often manually tuned by field experts and in general what works for one domain cannot be extended to others. A solution to the problem would be to employ an approach that is inherently designed to minimise the amount of sampling required and does not require exhaustive tuning of a proposal function. Bayesian Optimization is designed to achieve this goal.

2.4.3 Bayesian Optimization

Bayesian Optimization (BayesOpt) is a class of algorithms aimed at optimizing objective functions that are very costly (in terms of time, money or other resources) to evaluate [60].

BayesOpt algorithms try to minimise the number of evaluations of the objective by employing an acquisition function which identifies the next point to sample in order to maximize the informational gain of the sampling procedure by balancing the "exploration-exploitation" trade-off (sample points that are far from the current sample to learn about the function in an unobserved area vs. pick the next sample near the current one when this is believed to be near an optimum).

BayesOpt consists of two main components:

- A Bayesian statistical model for modelling the objective function, a "surrogate" of it. The goal of the surrogate is to provide a posterior probability distribution that describes potential values for the objective function at a specific candidate point (Fig.2.6, Top panel).
- An acquisition function for deciding where to sample next. It measures the value that would be generated by evaluation of the objective function at a new point, based on the current posterior distribution over the target (Fig.2.6, Bottom panel).

BayesOpt is often used in conjunction with other parameter estimation methods. For instance, it is used in adaptive MCMC to tune the hyperparameters of the Markov Chain [1]. However, to the best of my knowledge, it has not been applied in isolation for the purpose of parameter inference in the field of Systems and Computational biology.

BayesOpt can solve many of the problems encountered by traditional MCMC since it is, by design, a 'black-box, derivative-free global optimizer' [60]:

- Black-box, since no special structure about the function, like convexity or linearity, is known and can therefore be exploited;
- Derivative-free, since no first- or second-order derivatives are required, only the objective itself;
- Global, meaning it globally samples over the objective, rather than locally like in based on Markov Chains-based methods.

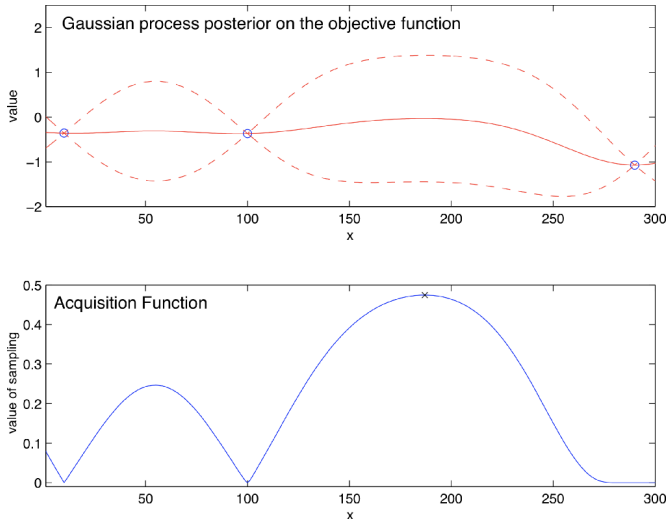

Figure 2.6: *Top panel*: The surrogate model, in this case a Gaussian Process, used to approximate the target. The blue dots represent observations of the objective function. The solid line is the estimate of the target. The dashed lines are Bayesian credible intervals, representing the confidence on the range of possible values that the target evaluates to at a specific point. *Bottom panel*: The acquisition function, whose maximum indicated by the 'x' represents the next point to sample on the surrogate. The acquisition function balances exploration (leading to sample in areas with bigger confidence intervals, representing uncertainty in that area) and exploitation (leading to sample in areas where an optimum is thought to be present based on previous samples).

Chapter 3

Implementation & Experiments

This chapter presents the software libraries, algorithms and implementation decisions adopted in the comparative analysis of Bayesian Optimization and Markov Chain Monte Carlo techniques for parameter inference. It discusses alternative tools and approaches for individual components of the experimental setup: Agent-Based Modelling (Section 3.1), BayesOpt (Section 3.2) and MCMC (Section 3.3). Finally, it outlines the experimental conditions under which the two methods have been explored (Section 3.4).

3.1 Agent-based Cell Simulators

The first step to build an agent-based simulation is to find the correct software that meets our modelling criteria. Sections 3.1.1 - 3.1.4 illustrate the advantages and limitations of different biological modelling packages. Although the perfect tool does not exist, we would like our candidates to strike a good balance between complexity of the simulation and ease of deployment. The properties to look for are:

- Customisability of growth rate and division rules;
- Stochasticity in agent-agent and agent-environment interactions;
- Support for stiff ODE solving.

Table 3.1 shows an overview of the packages I have tested and their properties.

3.1.1 Gro

Gro [9], developed at the University of Washington, was the first candidate ABM tool I adopted. This choice was driven by the simplicity of gro which provides a friendly user interface to the underlying C++ source code, while allowing for detailed model descriptions. After its first release in 2013, gro significantly surpassed the complexity of the biological modelling tools available at the time. It introduced, for instance, customisable morphology for growing microcolonies of bacteria and design of gene circuit interactions.

		Gro	CellModeller	CompuCell 3D	Bsim
Agent dynamics and features	Simple rules	✓	✓	✓	✓
	Advanced rules	✓	✓	✓	✓
	ODEs		✓	✓	✓
	DDEs				✓
	Chemical equations				
	Stochastic dynamics	✓	✓	✓	✓
	Motility	✓		✓	✓
	Chemotaxis	✓		✓	✓
	Cell replication	✓	✓	✓	✓
	Cell morphology	✓	✓	✓	
Environment	2D	✓		✓	
	3D		✓	✓	✓
	Chemical diffusion	✓	✓	✓	✓
	Complex objects			✓	✓
	Language	C++	Python	C++/Python	Java
	Reference(s)	[9]	[10]	[11]	[12]

Table 3.1: Overview of the Agent-Based Modelling tools tested and their properties. Table adapted from [8].

Description of properties

Simple Rules: a limited subset of commands are available to control agent behaviours;
Advanced Rules: access to a full programming language is provided to control agents;
ODEs: agents can use ordinary differential equations to describe their internal state;
DDEs: agents can use delay differential equations to describe their internal state;
Chemical equations: cellular chemical reaction networks can be simulated;
Stochastic dynamics: the internal state of an agent and the interactions with other agents can be stochastic, i.e. upon meeting another agent, there is a probability that they interact;
Motility: agents can move freely within the environment and functionality to manage collisions/interactions is available;
Chemotaxis: a realistic implementation of chemotaxis, i.e. chemically induced motility, is available to control cellular movement;
Cell replication: agents are able to replicate over time;
Cell morphology: agents can take an arbitrary shape or have the option to take one of multiple predefined shapes;
2D: visualizations in 2D of the simulations are available;
3D: visualizations in 3D of the simulations are available;
Chemical diffusion: movements of chemical species in the environment can be customised;
Complex objects: augment behaviour and properties of individual agents by extending desired classes.

A simple example showcasing the simplicity of gro is shown below, where production, dilution and degradation of the gfp protein produced by *E. coli* are simulated.

```
program make_gfp ( k1, k2, m ) := {

  // initialize gfp copy number
  gfp := m;

  // produce gfp
  rate ( k1 ) : { gfp := gfp + 1 }

  // degrade gfp
  rate ( k2 * gfp ) : { gfp := gfp - 1 }

};

// compute production rate needed for 100 copies of gfp per cell
alpha := - log ( 0.5 ) / 20.0; // dilution rate
k1 := 100 * alpha;             // production rate

// make a new cell
ecoli ( [ x := 0, y := 0 ], program make_gfp ( k1, 0.001, 0 ) );
```

As part of the inspection for the suitability of gro, I learned its specification language. The overall logic and syntax are easily provided within a programming background, however I used the opportunity to familiarise myself with fundamental biological concepts and how these are implemented in code. This proved to be an important stepping stone towards more complex tools and simulations explored in later stages.

Eventually, I discarded gro as the ABM tool of choice for two reasons. Firstly, it can only run on Windows or Mac operating systems, thus being incompatible with Linux, the os powering the machine on which the experiments that have been run. I attempted to overcome this obstacle by setting up a Windows Virtual Machine, however the virtualization software proved to be a considerable speed bottleneck. As Bayesian Optimization requires multiple simulation iterations to be run, such a slowdown was not viable. Secondly, gro does not offer the possibility of specifying agents' internal state via a system of ordinary differential equations (ODEs). This limitation is quite restrictive in this work's context, as a thorough parameter inference analysis would treat variables exhibiting more complex behaviours.

3.1.2 CompuCell3D

CompuCell3D (CC3D) [11], originally developed at the University of Indiana in 2012, is a powerful ABM tool that allows rapid construction of multi-scale simulations for a variety of biological domains, including bacterial populations. Compared to gro, CC3D offers extensive instruction manuals, installation guides and support forums. More crucially for the purpose of this work, it supports ODE-specified agent behaviour. It is actively maintained and utilized in different research areas as of the time of writing.

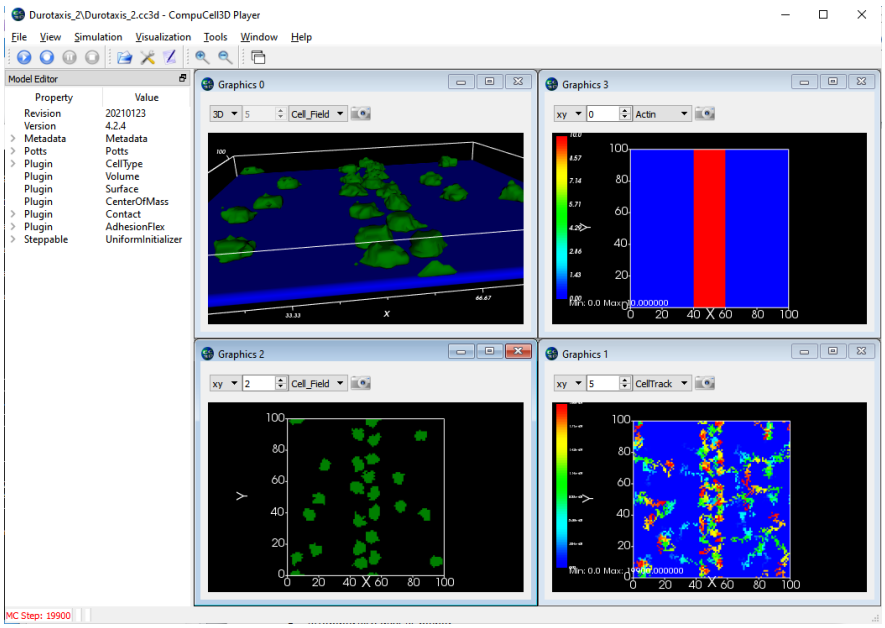

Figure 3.1: CC3D's GUI offers quick shortcuts to customize running simulations, as well as rich representations for qualitative visual analysis.

I decided to experiment with CC3D for multiple reasons:

1. Rich graphical user interface which allows for quick customizability of the simulations, as well as providing detailed representations for qualitative visual analysis. An example of CC3D's GUI can be seen in Fig. 3.1;
2. Source code for simulation directly written in Python, which brings two benefits. Firstly, since Python is the language I am most familiar with, it held promise of speeding up the development process. Secondly, it would be easier to interface with BayesOpt libraries also written in Python;
3. Runs on Linux, thus avoiding the need for a virtual machine and associated computational bottleneck;
4. Supports Systems Biology Markup Language (SBML) [13]. Many systems biological models are disseminated in SBML, a widely adopted protocol in the computational biology field. Importing a SBML file directly rather than having to manually write code for the simulation significantly reduces development time.

Although yielding some promising initial results, I encountered difficulties in installing and linking the external modules for ODE support (Tellerium, libRoadRunner). After extensive research and reaching out to some of the CompuCell3D developers, I was not able to solve the problem. As the process was taking significantly long, I eventually decided to explore further alternatives.

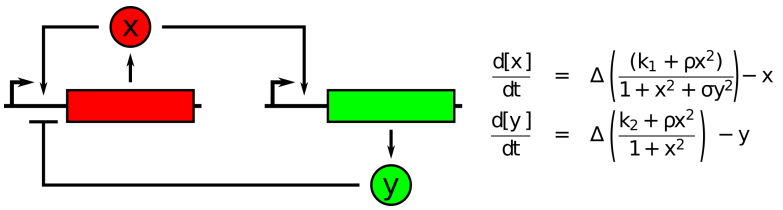

Figure 3.2: Simple gene circuit in which two species, x and y, are produced at the rates specified by the corresponding equations. Out of the four variables k_1, k_2, ρ and σ, only one, ρ, was inferred via BayesOpt as part of a proof-of-concept.

3.1.3 CellModeller

CellModeller [10], developed at the University of Cambridge, is a Python-based ABM framework offering customisable intra- and extracellular dynamics with ODE support. Although not maintained anymore, the concise tutorials allow to quickly deploy complex simulations. I have adopted CellModeller in the early stages of BayesOpt experimentation by making use of existing examples (specifically the circuit described in Fig. 3.2) that contain ODE-defined behaviour. This allowed me to create a successful proof-of-concept without the burden of implementing the simulation itself.

The major limitation of CellModeller is that it does not support stiff ODE solvers. As mentioned in Section 2.3, these are often present in systems biology, as parameters vary over different orders of magnitude. I therefore decided to discard CellModeller as the final ABM tool of choice, while looking for alternatives that would support more complex simulations with stiff ODEs.

3.1.4 Bsim

Bsim [52, 12] is a biological ABM tool that overcomes the limitations posed by the alternatives discussed above. Written in Java, it follows an object-oriented paradigm that requires knowledge of software engineering fundamentals, thus making not ideal for practitioners outside the computing field. It contains a variety of reusable, isolated components representing common bacterial traits applicable to numerous kinds of models, thus saving time and effort in the development process; it supports complex intracellular dynamics and stiff ODEs; it allows customisability of agents' morphology, division rules and growth rate, as well as environmental properties and interactions; finally, it offers an attractive GUI for 3D visualizations. For these reasons, I considered it the best ABM package candidate and I adopted it throughout the BayesOpt experiments.

3.2 Bayesian Optimization

The package I have chosen to perform Bayesian Optimization is GPyOpt [53, 54]. It is one of the first libraries for Bayesian Optimization written in Python. Despite its age and end of maintenance, it offers enough functionality to perform basic BayesOpt. Its main advantage, however, is the readability of the code and how easily the source files

can be modified to include new functionalities, e.g. new acquisition functions or sampling mechanisms. More recent alternatives such as BoTorch [55] are optimized for speed and parallelism at the cost of a steeper learning curve. I decided to adopt GPyOpt as it strikes a good balance between efficacy, speed and ease of implementation.

3.3 Markov Chain Monte Carlo

The initial objectives of this thesis included an implementation of parameter inference via Markov Chain Monte Carlo to compare against BayesOpt. I started developing MCMC via the PyMC3 package [56]. However, two problems forced me abandon this task. Firstly, difficulties in installing the right dependencies to make the library work took considerable time, and I preferred to shift my attention to BayesOpt. Although the documentation appears extensive at first, it lacks detailed explanation for a correct and complete installation. Secondly, and most pressingly, MCMC is known to take a considerable amount of iterations to obtain reliable estimates. Given the limited computational resources under which this work has been carried out, I could not afford to test the correctness of my implementation given the time required by MCMC, especially for more complex experiments (noisy function evaluations - Section 3.4.2 - and multiple parameters - Section 3.1). I have provided the implementation of MCMC in the supplemental material to this thesis, and I leave it to future research to use my code to perform a thorough analysis and comparison between MCMC and BayesOpt.

3.4 Experiments

The pipeline adopted throughout the experiments is shown in Figure 3.3. As a first step, the 'synthetic' data is produced. This corresponds to the data that we would obtain if we were to run the experiment *in vitro*. To obtain it, the simulation is run with the 'true value' of the parameters, which are the ground truths we would like to infer with Bayesian Optimization. The data can represent any quantity of interest, such as the total number of cells in the population, their growth rates or the concentrations of chemical species. I have chosen the average concentration of the LacI protein across the population at each timestep (see Section 3.4.4 for more details). Although this fails to capture the differences between individual cells that, as described in Section 2.2 can have significant consequences on population-level dynamics, I decided to adopt a less complex quantity to create a baseline to improve upon.

The BayesOpt loop then starts by making a random guess about the value of the parameter. This is a value sampled at random from a uniform distribution whose extremes are the bounds of the specified sampling range. If one were to have any prior knowledge about the distribution of the parameter, this could easily be exploited by changing the distribution the initial guess is sampled from. However, here I assume that no such prior is available, therefore a uniform distribution seemed a sensible choice.

The simulation is then run with the proposed value and the data, or 'predictions' from the model, are collected. Together with the synthetic data, the predictions are used to

compute the error, or objective function to minimise. The error function I chose is the Mean Squared Error (MSE), as widely adopted throughout the literature.

The evaluation of the objective function at the specified value is used by BayesOpt to update the surrogate and propose a new sample by maximising the acquisition function. This proposal is used to run the simulation again, and the loop continues until the converge criterion is met or a maximum number of iterations is reached. As explained in Sections 3.4.1 - 3.4.3, this choice is determined by multiple factors such as noise or high dimensionality of the search space.

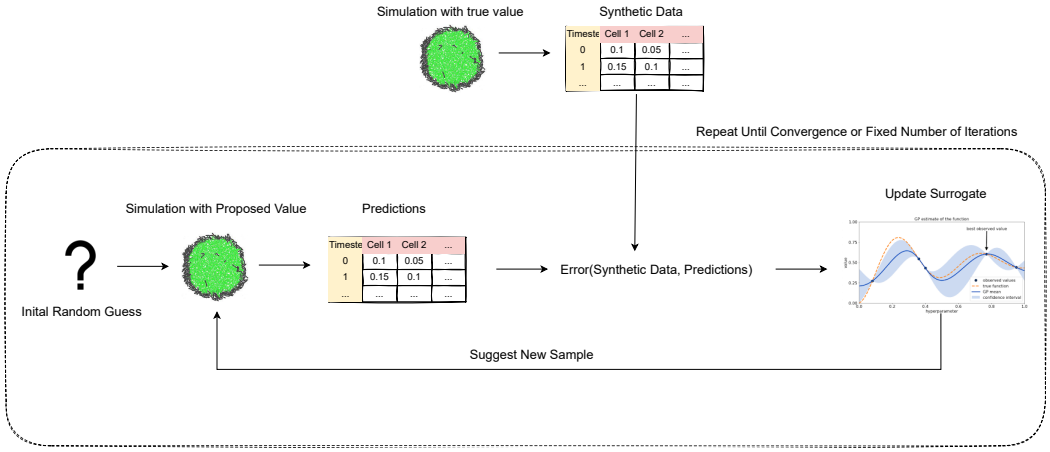

Figure 3.3: Schematic representation of the pipeline adopted throughout the experiments. Firstly, the synthetic data is produced by using the true values of the parameters: this is the ground truth. Then, starting from an initial random guess, BayesOpt uses the error computed between predictions and synthetic data to refine its surrogate and propose new samples based on the acquisition function. The loop is repeated until the convergence criterion is met.

I have used the experimental setup described to test the efficacy of BayesOpt under different conditions: increasingly large sampling spaces (Section 3.4.1), noisy function evaluations (Section 3.4.2), and multiple parameters to infer simultaneously (Section 3.4.3).

3.4.1 Effects of Sampling range on Inference

Biological parameters can often vary over several orders of magnitude, which increases the search space of viable parameter values [36]. By allowing parameters to vary over large ranges, biological systems develop the 'robustness' described in Section 2.1 to survive against changing environmental conditions [37]. For this reason, I performed a series of experiments to systematically test the efficacy of BayesOpt in inferring parameters within variable-sized search spaces.

I selected one key parameter of the ODE system and run a simulation for each sampling range with value 10^x for $x \in [1, 1.5, 2.0, 3.0, 3.5, 4.0]$. This means that for a sampling range value of 10^2, for instance, the search space for the parameter n extends from $n - 100$ to $n + 100$. Additionally, there is cut-off on the lower bound at zero,

as biochemical rates are constrained to non-negativity. The BO algorithm runs until convergence, which I have manually defined to be met when the last 10 samples are all within 0.1 of the nominal value of the parameter. I repeat each experiment five times to to obtain an average result and observe any differences across runs. Results for this set of experiments are presented in Section 4.1.

3.4.2 Effect of Noisy Evaluations on Inference

Noise permeates biology on all levels, from the most basic molecular, sub-cellular processes to the dynamics of tissues, organs, organisms and populations [38]. At a molecular level, control strategies based on dynamic equilibrium of stochastic events have been shown to offer robustness and tunability in processes such as mRNA translation [39, 40]. At a higher level, regulatory networks alter their topology to contribute to higher adaptability to changing environments and external noise (the 'robustness' property described in Sections 2.1 and 3.4.1). Although connections between noise at these different levels are still to be elucidated, there is strong evidence for its importance in the evolutionary process [41]. For this reason, I have conducted parameter inference via BayesOpt with different levels of artificial noise added to the simulated data. Note that an element of stochasticity is already present in the simulation itself, however an additional noise term could represent, for instance, external factors such as the microscope measurement error.

The amount of noise introduced is defined by the Signal-to-Noise Ratio (SNR) [42]. The SNR is the ratio of the signal power (in this context the concentration of the LacI protein) to the noise power:

$$SNR = \frac{P_{signal}}{P_{noise}}, \tag{3.1}$$

It is often expressed in decibels in the general context of signal processing. A ratio greater than 1:1 (or 0 dB) indicates more signal than noise. In order to add noise to the data, I first obtain the 'power of the signal', which for a random variable X is equal to $E[X^2]$, where E represents the expected value, or mean. In our case, this is the average concentration of LacI across all cells at time t

$$P_{signal} = E[(\text{LacI concentration at time } t)^2], \tag{3.2}$$

Since dB are used in the context of SNR, I convert the raw signal power to dB according to the following formula:

$$P_{signal,dB} = 10log_{10}(P_{signal}), \tag{3.3}$$

In a similar manner,

$$SNR_{dB} = 10log_{10}(SNR), \tag{3.4}$$

and when we substitute 3.1 in the above, together with the quotient rule for logarithms, we obtain

$$SNR_{dB} = 10log_{10}(P_{signal}) - 10log_{10}(P_{noise}). \tag{3.5}$$

I then provide a desired value of SNR in dB and use 3.5 to obtain the desired amount of noise power. For the same reasoning in 3.2, this is equal to $E[(noise)^2]$. In turn, $E[(noise)^2] = \mu_{noise}^2 + \sigma_{noise}^2$ [43] which, if we assume that the noise has mean of 0, reduces to σ_{noise}^2, its variance. Noise is finally sampled from a Gaussian distribution centered at zero with standard deviation σ_{noise} and added to the data produced by the simulation.

I have chosen SNR as a way to introduce noise because, by definition, it takes into account the values of the data the simulation produces, rather than blindly sampling gaussian noise with a prespecified standard deviation. This means that the method can easily be adapted to any other quantities one wished to perform parameter inference on without any fine-tuning required.

A visual representation of how much noise corresponds to different values of SNR is shown in Figure 3.4.

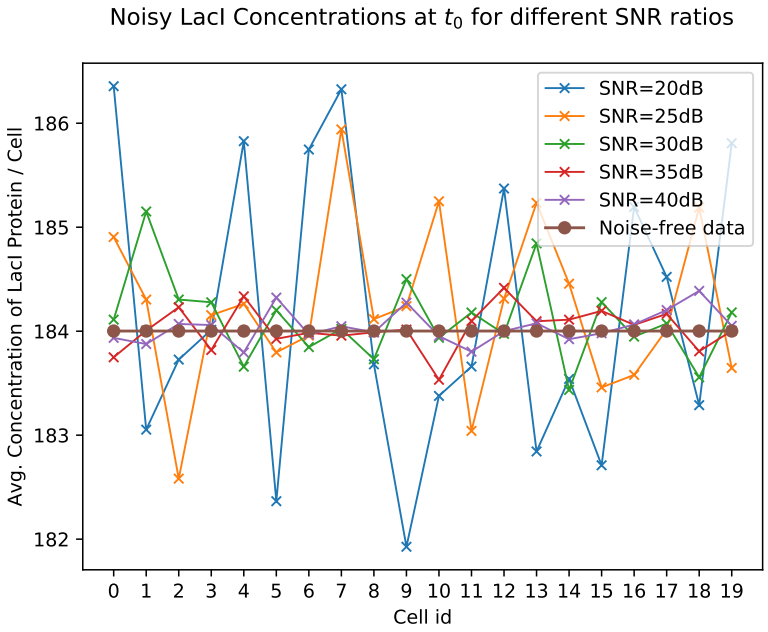

Figure 3.4: Gaussian noise is added to the average concentration of LacI protein in each cell for each timestep. The noise is sampled starting from a Signal-to-Noise (SNR) ratio, as described in equations 3.1 - 3.5.

The convergence criterion I initially adopted for this set of experiments was the same described in Section 3.4.1, i.e. sampling for 10 consecutive iterations within 0.1 of the nominal value of the parameter. However, I found that BayesOpt would never meet the criterion and therefore terminate. The first step I took towards a solution was

relaxing the convergence criterion, which did not lead to any improvements. This suggested that the problem could lie within the hyperparameter settings of the BayesOpt algorithm itself. Based on this insight, I modified the acquisition function from Expectation Improvement (EI), to Noisy Expectation Improvement (NEI), as proposed in [44]. The main issue with EI is that it computes the next sample based on the best function evaluation so far, which becomes challenging in a noisy setting given that the best estimate is unreliable. The authors of [44] provide compelling evidence for how NEI outperforms both EI and various heuristics used in combination with EI to deal with noise. The GPyOpt library does not support NEI, however I found a partially working implementation at [45] which I used as a starting point to integrate with the other modules in the library and overall pipeline. NEI allowed BayesOpt to terminate for the 40dB SNR experiment, however noise levels from 35dB to 20dB would still pose problems. Therefore, I modified the convergence criterion altogether, by running BayesOpt for a fixed number of iterations and observing the frequency counts of the sampled values. This choice was also dictated by the fact that NEI is significantly slower than EI, and I could not afford to let the experiments run for large number of iterations due to time and computational constraints under which this work has been carried out.

Results for this set of experiments are presented in Section 4.2.

3.4.3 Effects of Multiple Variables on Inference

A common difficulty encountered in parameter inference is that the model equations usually have a large number of unknowns, which cannot be determined individually with reasonable accuracy due to correlations between the parameters [31, 49]. Moreover, as already mentioned in Section 3.4.1 and as shown by Gutenkunst *et al.* [48], multiple parameterizations can reflect the model behaviour, with some parameters being 'stiff' (accurately identifiable) and others being 'sloppy' (varying by orders of magnitude without significantly influencing the quality of the fit). On the other hand, one could argue that often the precise value of a parameter is not required to draw biological conclusions [49]. I therefore considered testing BayesOpt inference with multiple parameters important.

Inference of two parameters successfully terminates according to the convergence criterion defined in Section 3.4.1. When inferring three and four variables, however, I had to modify the convergence criterion for the algorithm to terminate, similarly to the experiments described in Section 3.4.2. This was not done due to noisy function evaluations, however, but for the identifiability problem of sloppy parameters described above, as well as the toll imposed by the curse of dimensionality [50]: the prohibitive rise in computational costs due to the exponential growth of the search space as the number of parameters increases.

Results for this set of experiments are presented in Section 4.3.

3.4.4 Biological Model

The simulation I chose to use throughout the experiments models a 'synthetic biological clock' in populations of *E. Coli* [51]. Bacterial populations exhibit diverse biochemical rhythms, behaviours that repeat themselves by following regular patterns. These arise from the synchronous coordination of the individual cells' oscillations.

The reason I chose this particular model is that it is contained in the bsim installation files, therefore allowing me to avoid implementing the bio-chemical details. I still had to modify the source code to log the relevant data and interface bsim with the BayesOpt module. Initially, I started to implement the model described in Section 2.2, representing allocation of finite resources by molecular processes. However, due to time constraints and the fact the core topic of the project is Bayesian Optimization, rather than any specific biological model, I decided to forgo the idea to speed up development.

The biological clock model is described by seven ODEs and six parameters (equations 3.6 - 3.12). The nominal values of the parameters are reported in Table 3.2.

$$\frac{da_i}{dt} = -a_i + \frac{\alpha}{1+C_i^n}, \tag{3.6}$$

$$\frac{db_i}{dt} = -b_i + \frac{\alpha}{1+A_i^n}, \tag{3.7}$$

$$\frac{dc_i}{dt} = -c_i + \frac{\alpha}{1+B_i^n} + \frac{kS_i}{1+S_i}, \tag{3.8}$$

$$\frac{dA_i}{dt} = \beta(a_i - A_i), \tag{3.9}$$

$$\frac{dB_i}{dt} = \beta(b_i - B_i), \tag{3.10}$$

$$\frac{dC_i}{dt} = \beta(c_i - C_i), \tag{3.11}$$

$$\frac{dS_i}{dt} = -k_{s0}S_i + k_{s1}A_i - \eta(S_i - S_e). \tag{3.12}$$

For the biological meaning of the equations and parameters, I refer to the original work in which the model was presented [51].

I let the model run for 10 (simulated) seconds. This value has been determined empirically to strike a balance between complexity of the data collected and speed when run multiple times throughout BayesOpt. I leave for future research the task of running the simulation for longer and performing parameter inference on vaster datasets.

Name	Value
α	216.0
n	2.0
η	2.0
k_{s0}	1.0
k_{s1}	0.01
κ	20.0

Table 3.2: Nominal values of the parameters for the model presented in [51]. These represent the 'true values' used to obtain the synthetic data, or ground truth, used by BayesOpt top update the surrogate.

Chapter 4

Results

This chapter presents the results of the experiments illustrated in Chapter 3. Bayesian Optimization is shown to provide accurate estimates when sampling from increasingly large sampling spaces, while struggling when gaussian noise is added to the function evaluations. It correctly infers simultaneously two parameters, whereas for three and four parameters it is difficult to asses its efficacy due to the limited iterations the algorithm was run for.

4.1 Effects of Sampling Range on Inference

Bayesian Optimization has been adopted to infer parameters from an increasingly large sampling space following the considerations in Section 3.4.1. I have arbitrarily chosen κ, with nominal value of 20.0, to be inferred. The convergence criterion is met when the last 10 proposed samples are all within 0.1 of the nominal value of the parameter. The acquisition function used is expectation improvement. Five experiments are run for each sampling range and the results averaged.

Figure 4.2 illustrates the progression of the algorithm. Table 4.1 and Figure 4.1 summarise the results.

The number of iterations needed to converge increases as the sampling space expands. The average number of iterations, computed over the five runs of each set of experiments, stays stable between 10^1 and 10^2, while showing a 5-fold increase from 10^2 and 10^3, and a 1.5 increase to 10^4. I could not identify any clear trend in the relationship between iterations and range due to the limited size of the data. However, these initial results are promising as for practical applications, in which prior knowledge about the parameter by the modeller can further reduce the search space, successfully sampling over a range of $\pm 10^4$ is good enough. A more through analysis would include sampling over even vaster ranges as well as repeating each set of experiments for a greater number of times to obtain more statistically significant results. However, due to computational and time constrains encountered during this work, it is left up to future research.

The standard deviation across experiment sets rises at a much faster rate. This can be

observed in Figure 4.1, where a Gaussian distribution has been fitted to each set of experiments and appears to capture the variability of the individual data points. This can be explained by the fact that the algorithm has fewer chances of sampling in the vicinity of the minimum early on, given the vastness of the search space.

Range ($\pm 10^{\mathbf{x}}$)	**Average**	**Standard Deviation**
1.0	13	1.41
1.5	13	0.97
2.0	15	3.57
2.5	32	13.04
3.0	82	32.98
3.5	90	35.00
4.0	126	86.98

Table 4.1: Number of iterations (average and standard deviation for five runs of each experiment) needed for BayesOpt to converge to the nominal value of $\kappa = 20 \pm 0.1$ when sampling search spaces of different sizes.

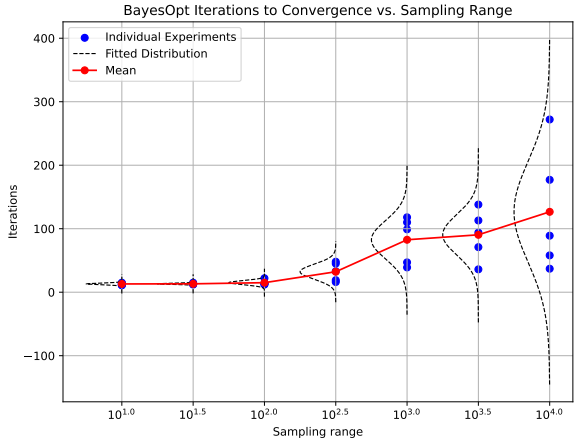

Figure 4.1: Number of iterations needed for BayesOpt to converge to nominal value of $\kappa = 20 \pm 0.1$ when sampling search spaces of different sizes. Both individual experiments and averages are shown. Note the increase in iterations as the sampling space expands, as well as the higher variability in the number of iterations required. The number of iterations required for each sampling range is normally distributed.

4.2 Effects of Noisy Evaluations on Inference

Bayesian Optimization has been adopted to infer parameters with gaussian noise added to the function evaluations, following the considerations in Section 3.4.2. I have arbitrarily chose κ to be inferred, with nominal value of 20.0, and a sampling space of $\pm$ 10. The algorithm would not converge according to the criterion adopted for the experiments in Section 4.1, therefore BayesOpt is run for a fixed number of iterations (50) and the resulting sampling frequencies inspected.

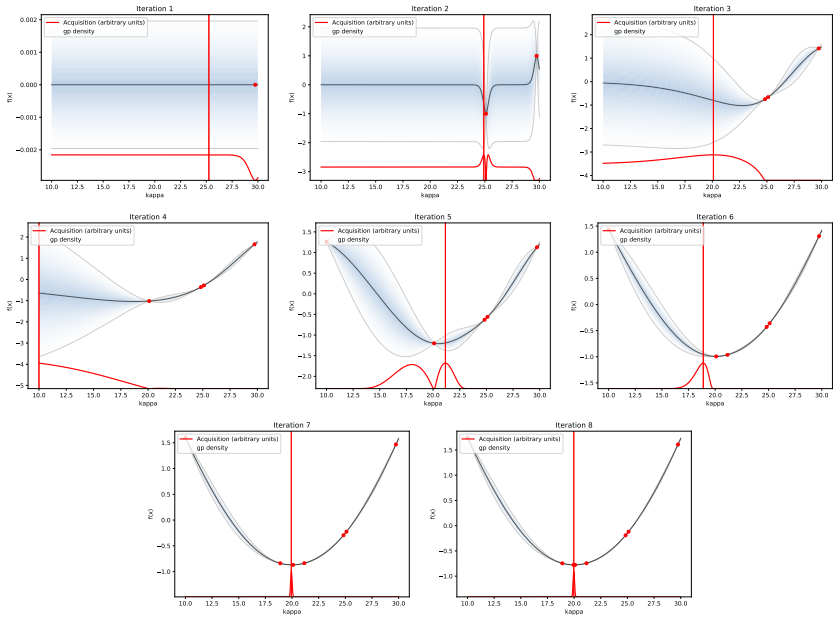

Figure 4.2: BayesOpt inference of κ (nominal value = 20.0), with sampling space ± 10, expected improvement as acquisition function, starting point randomly chosen in $[\kappa - 10, \kappa + 10]$, and no noise added to the function evaluations. The figure shows the surrogate (black), the confidence intervals (shaded blue), and the acquisition function (red). At each iteration the value of κ that maximizes the acquisition function is chosen as the next sampling point, the simulation is run with that value and the error computed between the simulated data and the synthetic data is used to update the surrogate.

As Figure 4.3 illustrates, the algorithm successfully converges to the true value of the parameter with SNR = 40dB, with the sampled values appearing to be normally distributed. The estimates quickly start to deteriorate for increasing levels of noise, starting with SNR = 30dB. A possible explanation for this behaviour is that the noise levels are too high, therefore multiple function evaluations $f(x)$ will yield different results for the same x. This leads BayesOpt to keep sampling around the same x multiple times until the uncertainty around that value is reduced, which is hard given the variability of the evaluations at that point. This explains why, for high levels of noise, there is a concentration of sampling around one or two values, whereas for SNR = 40dB the samples are more spread out. This hypothesis if further corroborated by Figure 4.4, showing the posterior and acquisition functions, as well as the function evaluations at the proposed samples, for SNR = 40dB and SNR = 30dB. In the case of SNR = 40dB, we can already observe how the variability of the evaluations leads to many peaks in the acquisition functions, indicating that there a are still a number of areas the algorithm wishes to explore. From the overall shape of the posterior, however, it is clear that BayesOpt has a rough idea of where the minimum is (compare this with the noise-free posterior in Figure 4.2). In the case of SNR = 30 dB, however, the evaluations for the same $\kappa \approx 22.5$ yield significantly different values, leading to BayesOpt getting stuck.

There are a number of considerations to be made regarding the experiments. Note that

these have not been implemented due to time and computational constraints and are left for future research.

1. A SNR of 40dB might be enough in certain contexts; the levels of noise to be simulated are application-specific and fine-tuning is required by modellers;
2. BayesOpt could be run for a number of iterations greater than 50. The panel in Figure 4.3 corresponding to SNR = 35dB, for instance, shows that the algorithm has sampled in the vicinity of the true value, and a more accurate estimate could have been obtained given a greater number of iterations;
3. My implementation of the NEI acquisition function requires more thorough testing to be confident of its efficacy;
4. A number of other acquisition functions can deal with noisy evaluations, such as entropy search [46] and knowledge gradient [47]. These have not been implemented and a thorough comparative analysis would be required to determine whether the problem lies with the type of acquisition rather than the levels of noise.

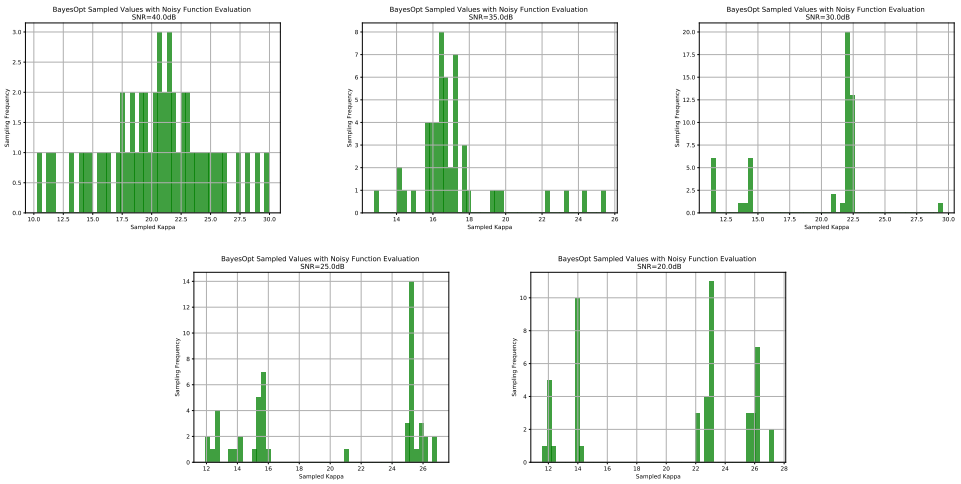

Figure 4.3: Sampling frequency of BayesOpt-proposed values with Gaussian noise added to function evaluations. Noise is added based on Signal-to-Noise (SNR) ratio, as described in Section 3.4.2. Note how the parameter is correctly inferred with a SNR of 40dB, and estimates become progressively worse for greater levels of noise.

4.3 Effects of Multiple Variables on Inference

Bayesian Optimization has been adopted to infer multiple parameters at the same time following the considerations in Section 3.4.3. I have arbitrarily chosen the parameters to infer to be $k_{s0} = 1.0$, $\alpha = 216.0$, $\kappa = 20.0$, $nExp = 2.0$.

For two-parameter inference, the convergence criterion of sampling ten consecutive values within 0.1 of the nominal values of the parameters is successfully met by

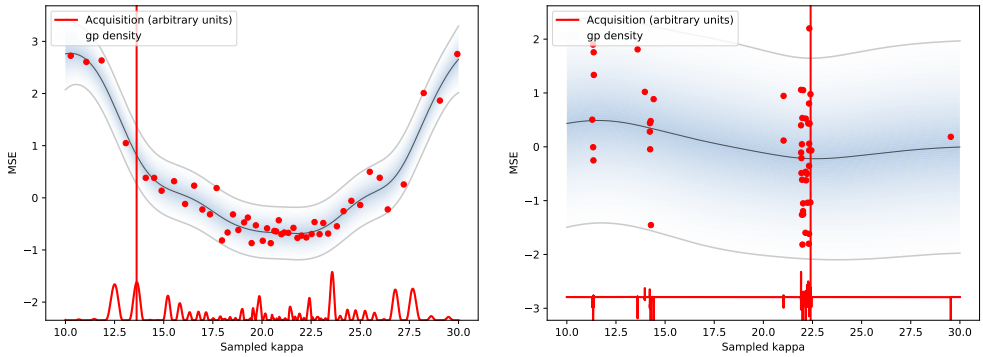

Figure 4.4: Surrogate and NEI acquisition function after 50 iterations with noisy evaluations. *Left panel*: SNR=40dB; *Right panel*: SNR=30dB.

BayesOpt. A progression of the algorithm is shown in Figure 4.6: although the objective function shows a 'valley' (area where the objective is uniformly low), BayesOpt is able to find the global minimum. Figure 4.5 shows how the algorithm, after identifying the correct value of α around iteration 20, keeps this fixed and explores the 'valley' by varying k_{s0}. BayesOpt converges in around 30 iterations (experiments were repeated five times and results averaged).

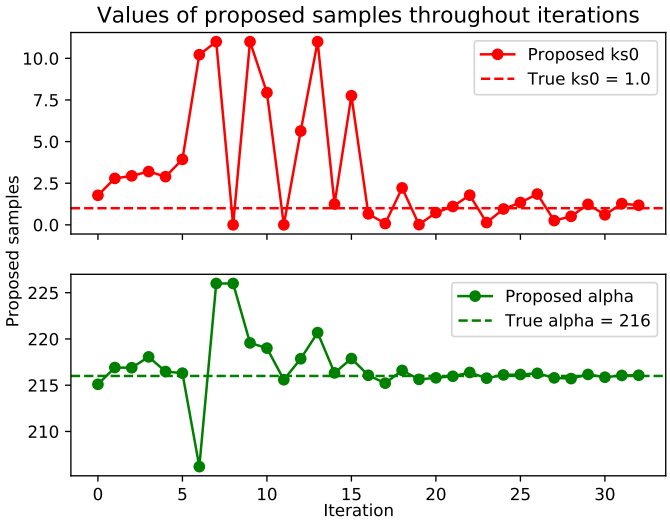

Figure 4.5: Sampled values of k_{s0} and α (nominal values 1.0 and 216.0, respectively) throughout iterations. The algorithm successfully terminates after roughly thirty iterations.

For three-parameter inference, the algorithm would not converge. Therefore, I ran it for a fixed number of iterations (500) and inspected frequency counts similarly to the noisy setting (Section 4.2). While BayesOpt is relatively confident about the viable region for k_{s0}, its estimates for α and κ are far from their nominal values (Fig. 4.7). In the case of α, there is a concentration of samples around 222.5, whereas the samples for κ are spread out and cover more or less uniformly the entire sampling space, except

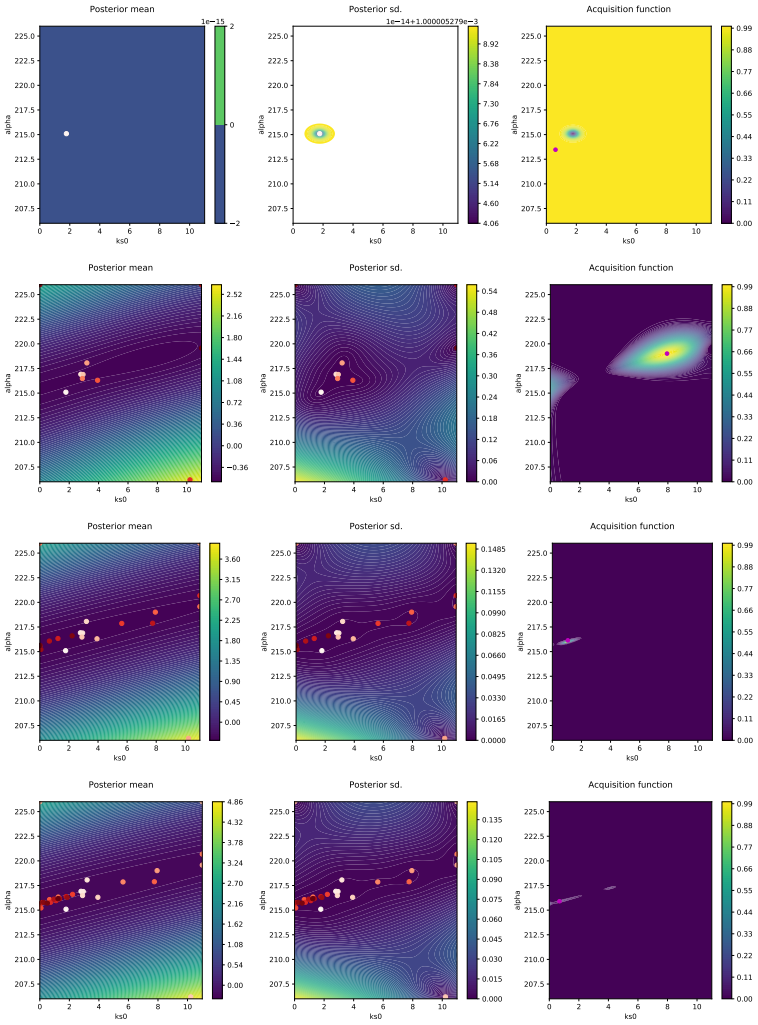

Figure 4.6: Surrogate and acquisition function for BayesOpt inference of two parameters, k_{s0} and α (nominal values 1.0 and 216.0, respectively), for successive iterations. The surrogate's mean and standard deviation are plotted separately. Iteration number from top to bottom: 1, 10, 20, 30.

for a burst of samples around 10.0.

Similar results can be seen for the inference of four parameters (Fig. 4.8, note that here BayesOpt has been run for 1,000 iterations given the greater search space and curse of dimensionality problem discusses in Section 3.4.3). In this case, it is the *nExp* variable BayesOpt is confident about, while other values are sampled uniformly across the entire search space. This suggests the *nExp* might be a 'stiff' parameter, while others might be 'sloppier' (extending over extended ranges with limited impact on overall objective landscape). An interesting feature to glean from the histograms is the presence of high sampling at the extremes of the search spaces for $k_{s)}$, α and κ. An explanation for this could be that BayesOpt fixes all parameter values but one, and explores the objective by varying only one parameter. This would also explain the spread of the samples between the two extremes.

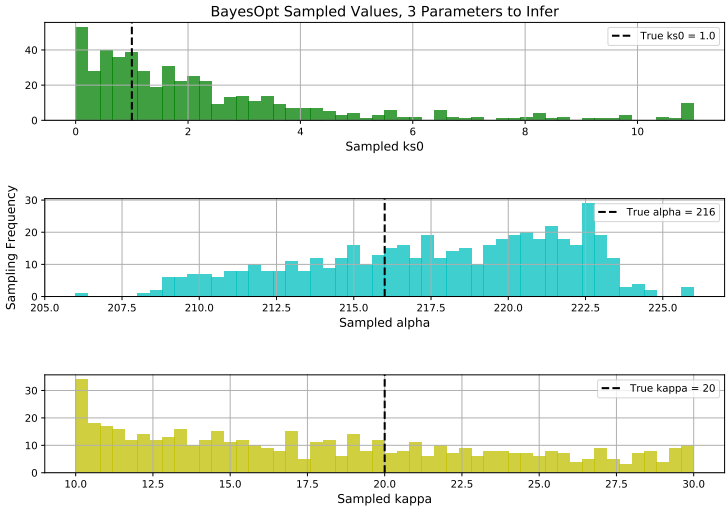

Figure 4.7: Sampling frequency of BayesOpt-proposed values for 3-variable parameter inference ($k_{s0} = 1.0$, $\alpha = 216.0$, $\kappa = 20.0$). The algorithm has been run for a fixed number (500) of iterations.

The current experimental setting could be augmented in multiple ways:

1. Run BayesOpt for a greater number of iterations. For the three- and four-parameter experiments, the iterations (500 and 1,000 respectively) could simply not be enough for the algorithm to converge. The computer used to perform the experiments posed serious limitations to their extent, and I had to balance a trade-off between breadth and depth for each one of them;

2. The sampled solutions would need to undergo *a posteriori* identifiability analysis, i.e. testing whether they are statistically significant by inspecting the objective landscape in the neighbourhood of the proposed vector parameters, as shown, for instance, in [49].

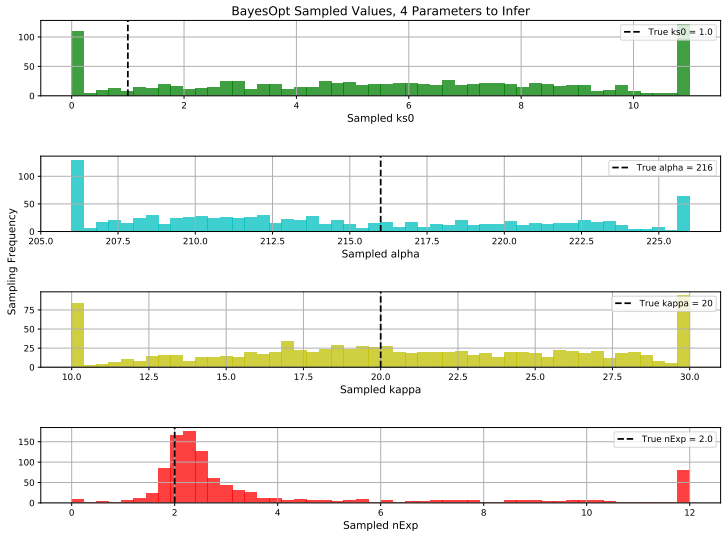

Figure 4.8: Sampling frequency of BayesOpt-proposed values for 4-variable parameter inference ($k_{s0} = 1.0$, $\alpha = 216.0$, $\kappa = 20.0$, $nExp = 2.0$). The algorithm has been run for a fixed number (1,000) of iterations.

Chapter 5

Conclusion

5.1 Summary

This thesis has explored the efficacy of Bayesian Optimization in the field of Systems Biology to infer parameters in multi-cellular simulations.

Chapter 2 provided a brief overview of the fields of Systems, Computational and Synthetic biology. It highlighted how a system-level understanding yields novel insights compared to a reductionist approach. It then delved into the computational and mathematical details of *in silico* modelling, describing agent-based simulations and ordinary differential equations. Finally, it highlighted the importance of efficient parameter estimation techniques by comparing Markov Chain Monte Carlo and Bayesian Optimization.

Chapter 3 described different software packages for agent-based simulations, identifying Bsim as the optimal choice, as well as the libraries used for BayesOpt and MCMC. It also outlined the structure of the pipeline adopted throughout the experiments, and the conditions under which BayesOpt has been tested.

Finally, Chapter 5 presented the results of the experiments. Bayesian Optimization has been shown to successfully estimate the correct parameter values under certain conditions. With variable-size search spaces, BayesOpt can correctly estimate parameters with a sampling range spanning up to four orders of magnitude. Noisy function evaluations make inference more difficult, and although an alternative acquisition function (Noisy Expected Improvement) has been implemented to deal with the issue, results show that a SNR $\approx$ 30-35dB still poses problems. Finally, simultaneous inference of two parameters terminates successfully in reasonable time, while for three- and four-parameter estimation results are inconclusive.

5.2 Future Work

Interesting directions for future research on the topic can extend this thesis in multiple ways:

- Provided more computational resources, run BayesOpt for a greater number of iterations. I had to carefully balance breadth and depth of this work given time and computational constraints. A more complete analysis would inspect the behaviour of the algorithm over longer runs;
- Adopt different libraries. I have adopted GPyOpt as the library of choice due to its ease of implementation and relatively complete suite of functions. More complex alternatives such as BoTorch [55], optimized for GPU and parallel computing, could be used to conduct more intensive experiments:
- Test the implementation of MCMC provided in the supplemental material of this thesis. For time reasons, this was not thoroughly tested and therefore its results excluded from the final analysis. As the primary objective of this thesis was to compare BayesOpt against other frameworks, a natural continuation of this work would be to terminate implementation of MCMC and compare its accuracy and running time against BayesOpt's.

Bibliography

[1] N. Mahendran, Z. Wang, F. Hamze, and N. De Freitas, "Adaptive mcmc with bayesian optimization," in *Artificial Intelligence and Statistics*, pp. 751–760, PMLR, 2012.

[2] A. Y. Weiße, D. A. Oyarzún, V. Danos, and P. S. Swain, "Mechanistic links between cellular trade-offs, gene expression, and growth," vol. 112, pp. E1038–E1047, Feb. 2015.

[3] D. J. Wilkinson, "Bayesian methods in bioinformatics and computational systems biology," vol. 8, pp. 109–116, Dec. 2006.

[4] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*. USA: Cambridge University Press, 2002.

[5] S. Sinharay, "Experiences with markov chain monte carlo convergence assessment in two psychometric examples," vol. 29, pp. 461–488, Dec. 2004.

[6] C. Andrieu, A. Doucet, and R. Holenstein, "Particle markov chain monte carlo methods," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 3, pp. 269–342, 2010.

[7] C. Robert and G. Casella, "A short history of markov chain monte carlo: Subjective recollections from incomplete data," vol. 26, Feb. 2011.

[8] T. E. Gorochowski, "Agent-based modelling in synthetic biology," *Essays Biochem*, vol. 60, pp. 325–336, 11 2016.

[9] S. S. Jang, K. T. Oishi, R. G. Egbert, and E. Klavins, "Specification and simulation of synthetic multicelled behaviors," *ACS Synthetic Biology*, vol. 1, pp. 365–374, Aug 2012.

[10] T. J. Rudge, P. J. Steiner, A. Phillips, and J. Haseloff, "Computational modeling of synthetic microbial biofilms," *ACS Synth Biol*, vol. 1, pp. 345–352, Aug 2012.

[11] M. H. Swat, G. L. Thomas, J. M. Belmonte, A. Shirinifard, D. Hmeljak, and J. A. Glazier, "Multi-scale modeling of tissues using CompuCell3D," *Methods Cell Biol*, vol. 110, pp. 325–366, 2012.

[12] A. Matyjaszkiewicz, G. Fiore, F. Annunziata, C. S. Grierson, N. J. Savery, L. Marucci, and M. di Bernardo, "BSim 2.0: An Advanced Agent-Based Cell Simulator," *ACS Synth Biol*, vol. 6, pp. 1969–1972, 10 2017.

[13] M. Hucka, F. T. Bergmann, C. Chaouiya, A. Dräger, S. Hoops, S. M. Keating, M. König, N. L. Novère, C. J. Myers, B. G. Olivier, S. Sahle, J. C. Schaff, R. Sheriff, L. P. Smith, D. Waltemath, D. J. Wilkinson, and F. Zhang, "The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 2 Core Release 2," *J Integr Bioinform*, vol. 16, Jun 2019.

[14] P. Stadter, Y. Schalte, L. Schmiester, J. Hasenauer, and P. L. Stapor, "Benchmarking of numerical integration methods for ode models of biological systems," *Scientific Reports*, vol. 11, p. 2696, Jan 2021.

[15] A. Raue, M. Schilling, J. Bachmann, A. Matteson, M. Schelker, M. Schelke, D. Kaschek, S. Hug, C. Kreutz, B. D. Harms, F. J. Theis, U. Klingmüller, and J. Timmer, "Lessons learned from quantitative dynamical modeling in systems biology," *PLoS One*, vol. 8, no. 9, p. e74335, 2013.

[16] H. Kitano, "Systems biology: A brief overview," 2002.

[17] S. Kesić, "Systems biology, emergence and antireductionism," *Saudi Journal of Biological Sciences*, vol. 23, no. 5, pp. 584–591, 2016.

[18] A. Aderem, "Systems biology: its practice and challenges," *Cell*, vol. 121, pp. 511–513, May 2005.

[19] D. Liu, A. Hoynes-O'Connor, and F. Zhang, "Bridging the gap between systems biology and synthetic biology," *Front Microbiol*, vol. 4, p. 211, 2013.

[20] E. M. Ozbudak, M. Thattai, H. N. Lim, B. I. Shraiman, and A. Van Oudenaarden, "Multistability in the lactose utilization network of Escherichia coli," *Nature*, vol. 427, pp. 737–740, Feb 2004.

[21] T. O. Richardson and T. E. Gorochowski, "Beyond contact-based transmission networks: the role of spatial coincidence," *J R Soc Interface*, vol. 12, p. 20150705, Oct 2015.

[22] "Matlab ode45." https://www.mathworks.com/help/matlab/ref/ode45.html. Accessed: 2022-04-02.

[23] "Matlab ode15s." https://uk.mathworks.com/help/matlab/ref/ode15s.html. Accessed: 2022-04-02.

[24] A. Yazdani, L. Lu, M. Raissi, and G. E. Karniadakis, "Systems biology informed deep learning for inferring parameters and hidden dynamics," *PLoS Comput Biol*, vol. 16, p. e1007575, 11 2020.

[25] H. B. Curry, "The method of steepest descent for non-linear minimization problems," *Quarterly of Applied Mathematics*, vol. 2, pp. 258–261, 1944.

[26] B. Calderhead and M. Girolami, "Estimating bayes factor via thermodynamic integration and population mcmc," *Computational Statistics Data Analysis*, vol. 53, pp. 4028–4045, 10 2009.

[27] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer, "Copasi—a complex pathway simulator," *Bioinformatics*, vol. 22, no. 24, pp. 3067–3074, 2006.

[28] T. Maiwald and J. Timmer, "Dynamical modeling and multi-experiment fitting with potterswheel," *Bioinformatics*, vol. 24, no. 18, pp. 2037–2043, 2008.

[29] G. I. Valderrama-Bahamóndez and H. Fröhlich, "Mcmc techniques for parameter estimation of ode based models in systems biology," *Frontiers in Applied Mathematics and Statistics*, vol. 5, 2019.

[30] I. G. Johnston, "Efficient parametric inference for stochastic biological systems with measured variability," 2014.

[31] M. Ashyraliyev, Y. Fomekong-Nanfack, J. A. Kaandorp, and J. G. Blom, "Systems biology: parameter estimation for biochemical models," *FEBS J*, vol. 276, pp. 886–902, Feb 2009.

[32] C. G. Moles, P. Mendes, and J. R. Banga, "Parameter estimation in biochemical pathways: a comparison of global optimization methods," *Genome Res*, vol. 13, pp. 2467–2474, Nov 2003.

[33] S. Gupta, L. Hainsworth, J. Hogg, R. Lee, and J. Faeder, "Evaluation of parallel tempering to accelerate bayesian parameter estimation in systems biology," in *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pp. 690–697, 2018.

[34] J. Liepe, P. Kirk, S. Filippi, T. Toni, C. P. Barnes, and M. P. Stumpf, "A framework for parameter estimation and model selection from experimental data in systems biology using approximate Bayesian computation," *Nat Protoc*, vol. 9, pp. 439–456, Feb 2014.

[35] J. D. Mathews, C. T. McCaw, J. McVernon, E. S. McBryde, and J. M. McCaw, "A biological model for influenza transmission: pandemic planning implications of asymptomatic infection and immunity," *PLoS One*, vol. 2, no. 11, p. e1220, 2007.

[36] J. Stelling, U. Sauer, Z. Szallasi, F. J. Doyle, and J. Doyle, "Robustness of cellular functions," *Cell*, vol. 118, pp. 675–685, Sep 2004.

[37] E. Zamora-Sillero, M. Hafner, A. Ibig, J. Stelling, and A. Wagner, "Efficient characterization of high-dimensional parameter spaces for systems biology," *BMC Syst Biol*, vol. 5, p. 142, Sep 2011.

[38] L. S. Tsimring, "Noise in biology," *Reports on Progress in Physics*, vol. 77, no. 2, p. 026601, 2014.

[39] V. Pancaldi, "Biological noise to get a sense of direction: an analogy between chemotaxis and stress response," *Frontiers in Genetics*, vol. 5, 2014.

[40] A. Sanchez and I. Golding, "Genetic determinants and cellular constraints in noisy gene expression," *Science*, vol. 342, no. 6163, pp. 1188–1193, 2013.

[41] B. Lehner, "Selection to minimise noise in living systems and its implications for the evolution of gene expression," *Mol Syst Biol*, vol. 4, p. 170, 2008.

[42] D. H. Johnson, "Signal-to-noise ratio," *Scholarpedia*, vol. 1, no. 12, p. 2088, 2006.

[43] R. Durrett, *Probability: theory and examples*, vol. 49. Cambridge university press, 2019.

[44] B. Letham, B. Karrer, G. Ottoni, and E. Bakshy, "Constrained bayesian optimization with noisy experiments," 2017.

[45] "Nei implementation." https://github.com/SheffieldML/GPyOpt/issues/192. Accessed: 2022-04-05.

[46] P. Hennig and C. J. Schuler, "Entropy search for information-efficient global optimization," *J. Mach. Learn. Res.*, vol. 13, p. 1809–1837, jun 2012.

[47] W. Scott, P. Frazier, and W. Powell, "The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression," *SIAM Journal on Optimization*, vol. 21, no. 3, pp. 996–1026, 2011.

[48] R. N. Gutenkunst, J. J. Waterfall, F. P. Casey, K. S. Brown, C. R. Myers, and J. P. Sethna, "Universally sloppy parameter sensitivities in systems biology models," *PLoS Comput Biol*, vol. 3, pp. 1871–1878, Oct 2007.

[49] M. Ashyraliyev, J. Jaeger, and J. G. Blom, "Parameter estimation and determinability analysis applied to Drosophila gap gene circuits," *BMC Syst Biol*, vol. 2, p. 83, Sep 2008.

[50] F. Y. Kuo and I. H. Sloan, "Lifting the curse of dimensionality," *Notices of the AMS*, vol. 52, no. 11, pp. 1320–1328, 2005.

[51] J. Garcia-Ojalvo, M. B. Elowitz, and S. H. Strogatz, "Modeling a synthetic multicellular clock: Repressilators coupled by quorum sensing," *Proceedings of the National Academy of Sciences*, vol. 101, no. 30, pp. 10955–10960, 2004.

[52] T. E. Gorochowski, A. Matyjaszkiewicz, T. Todd, N. Oak, K. Kowalska, S. Reid, K. T. Tsaneva-Atanasova, N. J. Savery, C. S. Grierson, and M. di Bernardo, "BSim: an agent-based tool for modeling bacterial populations in systems and synthetic biology," *PLoS One*, vol. 7, no. 8, p. e42790, 2012.

[53] "Gpyopt github." https://github.com/SheffieldML/GPyOpt. Accessed: 2022-04-07.

[54] "Gpyopt." https://sheffieldml.github.io/GPyOpt/. Accessed: 2022-04-07.

[55] "Botorch." https://botorch.org/. Accessed: 2022-04-07.

[56] "Pymc3." https://docs.pymc.io/en/v3/. Accessed: 2022-04-07.

[57] M. Plummer, N. Best, K. Cowles, and K. Vines, "Coda: convergence diagnosis and output analysis for mcmc," *R News*, vol. 6, pp. 7–11, March 2006.

[58] S. Brooks, P. Giudici, and A. Philippe, "Nonparametric convergence assessment for mcmc model selection," *Journal of Computational and Graphical Statistics*, vol. 12, no. 1, pp. 1–22, 2003.

[59] J. S. Rosenthal *et al.*, "Optimal proposal distributions and adaptive mcmc," *Handbook of Markov Chain Monte Carlo*, vol. 4, no. 10.1201, 2011.

[60] 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.