BAYESIAN STATISTICAL INFERENCE FOR ORDINARY DIFFERENTIAL EQUATIONS

THOMAS ARMSTRONG

Abstract. Mathematical models using Ordinary Differential Equations (ODEs) are very common in areas of applied mathematics because they allow us to specify how the behaviour of a system evolves in time. These ODE systems often involve parameters which, in part, characterise how a variable in the system will develop, based on interactions with other variables in the system or with itself. The act of inferring the values of parameters that occur in these ODEs can be a difficult task when faced with noisy experimental data. This report looks at applying methods of approximate inference to the posterior distribution of unknown parameters, after observing simulated data from a system. In particular, we use the grid approximation and the Metropolis-Hastings algorithm in the estimation of low-dimensional posterior distributions. We consider posterior distributions arising from two small ODE models, one being the Logistic, or Verhulst-Pearl, equation and the other being a two-dimensional, artificially created, predator-prey model. In each case of posterior approximation, we will also explore how the convergence of Markov Chain Monte Carlo techniques can be assessed using the Gelman-Rubin $\widehat{R}$ diagnostic. We conclude by discussing certain problems faced by the grid approximation and the Metropolis-Hastings algorithm, their applications to systems biology, and suggestions for further work.

Contents

1. Introduction

The values of parameters can be very important in determining the dynamics of a system; for certain values, we may obtain limit cycles and oscillatory solutions, whilst for other values we have variables that escape to infinity. When faced with experimental data from a system that is governed by a set of Ordinary Differential Equations (ODEs), we may want to find the values of the parameters that produced such dynamics. This task of parameter identifiability can be approached in a variety of ways. From the classical frequentist arsenal, we could look to methods like Maximum Likelihood Estimation to provide the parameter estimates. A contemporary approach for parameter estimation in systems biology, and the approach used in this report, is to instead utilise the Bayesian framework. The Bayesian paradigm is particularly attractive to work with because of the ability to incorporate our pre-experiment beliefs into the inference problem, through the use of a prior distribution. Due to the complexity of the models arising from the Bayesian framework and ODE system, we are typically faced with the problem that the posterior is not available in a closed form. This report will look at methods that produce approximations to the posterior distribution, be it analytical or computational approximations; this task is known as *approximate inference*. More specifically, we will look at the grid approximation, a brute-force method for the estimation of low-dimensional posteriors, and the Metropolis-Hastings algorithm [1], a member of the Markov Chain Monte Carlo (MCMC) methods. The development of accurate and efficient methods for parameter estimation in ODE models has become a prominent area of work in computational systems biology. ODE systems are particularly useful in providing a deterministic framework to explain the dynamics of complex biological interactions. These systems typically depend on several parameters, which can be hard, or impossible, to measure directly. A conference and competition named Competitive Statistical Inference for Differential Equations (CSIDE), held at the University of Glasgow, gave important practical applications to which approximate inference could be applied [2]. In particular, it looked at how accurate parameter estimation could aide in the understanding of cell migration during chemotaxis.

This report will discuss how the grid approximation can be implemented, and the practical issues that arise in medium to high-dimensional problems. It will also build on the structures set out in [3] and [4] for the implementation of MCMC methods to ODE-based models. We will apply both methods of approximate inference to the posterior distributions arising from two different systems, namely the Logistic Equation and a two dimensional predator-prey model.

Throughout this report, we will be applying certain methods of approximate inference to a variety of models. To view the code used to produce the following results, the reader is directed to my GitHub which hosts a set of Python notebooks [5]. These notebooks show how the following algorithms can be implemented in practice, they can also be downloaded and altered if the reader wishes to experiment with the models further.

1.1. Ordinary Differential Equations.

We will be considering situations in which we have obtained experimental data from a system which we know follows a set of ODEs. This system of equations, however, depends on a set of unknown parameters $\boldsymbol{\theta}$.

$$\dot{\boldsymbol{x}} = \mathbf{f}(\boldsymbol{x}(t); \boldsymbol{\theta})$$

Here $\boldsymbol{x} \in \mathbb{R}^n$ is the vector of variables in the system and $\mathbf{f} = (f_1, f_2, \ldots, f_n)$ is a vector of functions describing the derivatives of these variables. We make m observations of the system, denoted $\boldsymbol{y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_m})$ at times $t_1, t_2, ..., t_m$ respectively. In this report, we wish to observe how experimental data can update our beliefs about the values of parameters in a system of ODEs. To do so, we will be working in a Bayesian framework. We will be heavily reliant on this framework, so a brief description of the Bayesian methodology will be provided in the following section.

1.2. **The Bayesian Framework.**

1.2.1. *Bayes Rule.* Inferential statistics makes use of data to help us decide what we should believe about the probability of certain events or the values of variables. If we toss a coin ten times and obtain nine heads, should we believe that this coin is fair? By using observations, we can make inferences to update our past beliefs. To use this new information, we can use Bayes Rule, displayed in Eq.(1); this formula allows us to update our beliefs about the probability of an event, given that another event has occurred. If we consider an event space Ω, partitioned into a collection of sets $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$, then for an event E, Bayes Rule is written as

$$P(\omega_i \mid E) = \frac{P(\omega_i)P(E \mid \omega_i)}{P(E)} = \frac{P(\omega_i)P(E \mid \omega_i)}{\sum_{i=1}^{K} P(\omega_i)P(E \mid \omega_i)} \tag{1}$$

where $P(\cdot)$ denotes the probability of an event occurring and $P(\cdot \mid \cdot)$ denotes conditional probability. In the last step, we have used the rule of marginal probabilities [6].

As well as for discrete events, Bayesian inference can also be used for random variables. We consider a random variable to be defined as an unknown quantity that takes a value within an interval with a particular probability. A more formal definition of a random variable along with what it means to be continuous or discrete can be found in [7]. If we consider two continuous random variables X and Y, then Bayes Rule can be re-written as

$$p_{X|Y}(x \mid y) = \frac{p_X(x)\, p_{Y|X}(y \mid x)}{p_Y(y)} = \frac{p_X(x)\, p_{Y|X}(y \mid x)}{\int_\chi p_X(x)\, p_{Y|X}(y \mid x)\, dx} \tag{2}$$

where $p_X(\cdot)$ denotes the probability density function of a variable X, and χ the support of X. In some cases, we will consider an unnormalised posterior density $p^*_{X|Y}$ by omitting the denominator of Eq.(2), which is known as the marginal-likelihood.

$$p^*_{X|Y}(x \mid y) = p_X(x)\, p_{Y|X}(y \mid x) \tag{3}$$

From this point forward, we will drop the subscripts on our probability density functions, so that $p(x)$ represents $p_X(x)$.

1.2.2. *The Bayesian Approach.* For a statistical approach to be Bayesian, we require more than just using Bayes Rule in our working. An approach is considered to be Bayesian based on how we consider parameters. A frequentist approach considers a parameter, such as a chemical reaction rate, to have one fixed, true value. Meanwhile a Bayesian believes that, although a parameter may have one true value, we can never know truly what that value is based on experimental results. This view allows us to treat parameters as random variables, characterised by particular probability distributions. The Bayesian framework therefore uses Bayes Rule to perform inference on parameters and models given a set of observations $\boldsymbol{y}$. In order to make probability statements about a set of parameters $\boldsymbol{\theta}$, we require a parametric statistical model (the likelihood function) $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ and a prior distribution $\pi(\boldsymbol{\theta})$. Since we treat $\boldsymbol{\theta}$ as a set of random variables and $\boldsymbol{y}$ as realisations of random variables, we can use Bayes Rule to obtain a posterior density $p(\boldsymbol{\theta} \mid \boldsymbol{y})$.

$$\overbrace{p(\boldsymbol{\theta} \mid \boldsymbol{y})}^{\text{Posterior}} = \frac{\overbrace{\pi(\boldsymbol{\theta})}^{\text{Prior}}\ \overbrace{p(\boldsymbol{y} \mid \boldsymbol{\theta})}^{\text{Likelihood}}}{\underbrace{\int_{\boldsymbol{\Theta}} \pi(\boldsymbol{\theta})\, p(\boldsymbol{y} \mid \boldsymbol{\theta})\, d\boldsymbol{\theta}}_{\text{Marginal-likelihood}}}$$

The labelled posterior density characterises our updated beliefs about the probability density of $\boldsymbol{\theta}$ given a set of realisations of $\boldsymbol{y}$, through the use of the prior distribution placed on $\boldsymbol{\theta}$ and the likelihood function. It is particularly important to understand that the likelihood function is actually a function of $\boldsymbol{\theta}$ for a realisation $Y = \boldsymbol{y}$, this concept is often called the *likelihood principle* [8]. The prior distribution $\pi(\boldsymbol{\theta})$ characterises the belief in the values of $\boldsymbol{\theta}$ before observing the data $\boldsymbol{y}$. The choice of $\pi(\boldsymbol{\theta})$ is subjective and allows us to describe our current beliefs and incorporate any previous knowledge obtained from past experiments.

With certain choices of prior and likelihood distributions, we can analytically derive the posterior distribution by analysing the unnormalised posterior probability density given by Eq.(3). Such cases require prior-conjugate pairs, hence restricting the flexibility with which we can characterise the likelihood and our current beliefs in $\boldsymbol{\theta}$. In this report, we will work as though the posterior is not available in a closed form. This assumption puts us on a path towards methods in *approximate inference*, that is, we do not have a normalised posterior that we can evaluate.

The Bayesian framework is particularly attractive because of the nature of the resulting posterior. Unlike frequentist approaches like Maximum Likelihood Estimation, or non-linear optimisation algorithms, the posterior gives a full distribution for the updated beliefs about a particular variable, rather than a single point estimate. Having a full distribution to work with also allows us to easily compute the probability that a parameter falls within a certain interval, known as a *credible interval* or *credible region.*

1.2.3. *Bayesian Credible Intervals.* Typically, after approximating the posterior, we wish to define some region that contains a significant portion of the posterior. This introduces the idea of a Bayesian credible interval, which is a region designed so that the probability that a model parameter lies within the region is a certain percentage. This should not be confused with confidence intervals, a frequentist approach in which the uncertainty is assigned to the region rather than the value of interest. Within this report, we will use 95% credible regions to give bounds to the estimates for the parameters, after observing the data. More information regarding the Bayesian approach to credible regions, and how they differ from classical confidence intervals, can be found here in [9].

1.2.4. *Approximate Inference.* Approximate inference methods are used when exact inference, such as obtaining a normalised posterior, is either analytically or computationally intractable. The analytical intractability occurs when the marginal-likelihood

$$p(\boldsymbol{y}) = \int \pi(\boldsymbol{\theta})\, p(\boldsymbol{y} \mid \boldsymbol{\theta}) d\boldsymbol{\theta} \tag{4}$$

is not obtainable in a closed form, requiring us to utilise approximations in order to recover the posterior [10]. Within the Bayesian community, there are many methods associated to the broad class of approximate inference, however within this report we will consider only the grid approximation and a class of samplers called Markov Chain Monte Carlo.

1.3. **Parameter Estimation.** To be able to perform parameter estimation, we require the following components:

- An ODE system describing the experiment, with initial conditions.
- Noise model for the experimental data.
- Prior distribution for the set of parameters.

We specify an ODE system $\mathbf{f}$, defined in Section 1.1 so that we can analytically derive or numerically compute a solution with particular parameters at specified time points.

1.3.1. *Noise Model.* A noise model is required to explain the experimental errors that may occur when observing a system. From this noise model, we construct a likelihood function. If we consider an observation $\boldsymbol{y_i}$, then the probability that this data was produced by a particular set of parameters $\boldsymbol{\theta}$ is the likelihood function. In this report, we will assume that this likelihood takes the form of a Normal distribution, that is

$$\boldsymbol{y_i} \mid \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{x}(t_i; \boldsymbol{\theta}), \Delta)$$

where $\boldsymbol{x}(t_i; \boldsymbol{\theta})$ denotes the solution of the ODE system at time t_i, dependent on the particular set of parameters $\boldsymbol{\theta}$. We consider the noise models of the variables in the system to be independent, that is, the covariance matrix Δ is diagonal. We also assume that all observations are conditionally independent given $\boldsymbol{\theta}$. By this construction, we can use the inverse of de Finetti's Theorem [7] to treat the data as *exchangeable*; this means that the order in which one considers the observations does not matter. This assumption is important in the construction of the inference model we will use. We cannot treat the observations as independent, since knowing the value of $\boldsymbol{y_1}$ can influence the plausible range of values that $\boldsymbol{y_2}$ can take. Conditional independence, however, implies that if we know the true value of $\boldsymbol{\theta}$, then knowing $\boldsymbol{y_1}$ does not provide any additional information about $\boldsymbol{y_2}$.

$$p(\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_m} \mid \boldsymbol{\theta}) = \prod_{i=1}^{m} p(\boldsymbol{y_i} \mid \boldsymbol{\theta})$$

Most of our inference will be computationally demanding, and so we will actually be calculating the log-likelihood for a proposed $\boldsymbol{\theta}$; doing so improves the numerical stability of the calculations by replacing the product of small floating point numbers with a sum, which can later be exponentiated. The log likelihood for the entire set of observations is therefore

$$\log(p(\boldsymbol{y} \mid \boldsymbol{\theta})) = \sum_{i=1}^{m} \log(\mathcal{N}(\boldsymbol{y_i} \mid \boldsymbol{x}(t_i, \boldsymbol{\theta}), \Delta))$$

where the notation $\mathcal{N}(\boldsymbol{y_i} \mid \boldsymbol{x}(t_i, \boldsymbol{\theta}), \Delta)$ denotes the probability density function of $\boldsymbol{y}_i$ with mean $\boldsymbol{x}(t_i)$ and the diagonal covariance matrix Δ.

1.3.2. *Data Simulation.* In the inference model, we assume that we know how the errors in the observations are distributed. Because of this, we can only perform meaningful inference on data that is truly a sample, or set of samples, from this exact noise model. If we believe the errors form a different distribution, then the likelihood function must also change to suit the new distribution. To ensure that the errors in the data match the specific noise model, we will be using data that has been artificially simulated. To generate this data, we first define a set of time points $t = (t_1, t_2, \ldots, t_m)$, these points represent the times at which we make measurements of the system. Within this report, the time points are linearly spaced, however this is not a requirement to perform inference. We set the true value of the parameters to be $\boldsymbol{\theta^*}$ and prescribe the initial conditions of the system, $\boldsymbol{x}(0; \boldsymbol{\theta^*}) = \boldsymbol{x}_0$. The solution of the system can then be computed at each time point, with these parameters and initial conditions; that is, at each time t_i, we compute $\boldsymbol{x}(t_i; \boldsymbol{\theta^*})$ to obtain a set of vectors $(\boldsymbol{x}(t_1; \boldsymbol{\theta^*}), \boldsymbol{x}(t_2; \boldsymbol{\theta^*}), \ldots, \boldsymbol{x}(t_m; \boldsymbol{\theta^*}))$. To compute the solution of a particular system, we either input t_i, $\boldsymbol{\theta^*}$ and $\boldsymbol{x}_0$ into the solution, if it is available in closed form, or we numerically approximate the solution. To numerically approximate the solution, we will use the `integrate` module from the SciPy library [11]. This package uses the Runge-Kutta method (RK45) for numerical integration by default; more information about this method, and how it is implemented within the package, can be found in [12]. We then add independent Gaussian noise to the solutions at each time point to obtain the simulated data. This noise has a mean of zero and a pre-specified variance or diagonal co-variance matrix, matching the noise model.

$$\boldsymbol{y}(t_i) = \boldsymbol{x}(t_i, \boldsymbol{\theta^*}) + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \Delta)$$

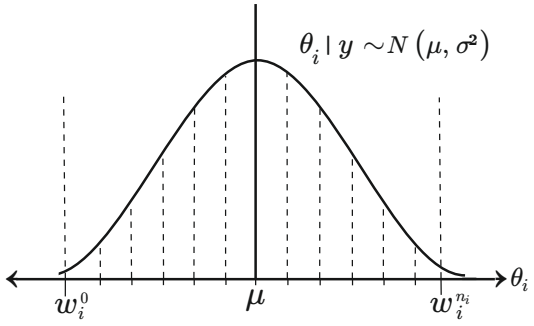


FIGURE 1. Toy example displaying suggested upper and lower bounds for a grid approximation.

2. Posterior Grid Approximation

The grid approximation, also known as the Riemann approximation, is a method of approximate inference that can be used for complex posterior distributions in low dimensions. If performed correctly, the accuracy of this method increases in polynomial time. The grid approximation is a method that approximates the marginal-likelihood, which is needed as the normalisation constant of the posterior distribution in Eq.(2) [6]. This approximation comes in the form of using the Riemann sum to estimate the integral in Eq.(5). In addition to the grid approximation, Monte Carlo techniques such as importance sampling also look to approximate this integral, these are discussed further in Section 3.1.1.

$$p(\boldsymbol{y}) = \int \pi(\boldsymbol{\theta})\, p(\boldsymbol{y} \mid \boldsymbol{\theta}) d\boldsymbol{\theta} \tag{5}$$

2.1. **Defining a grid.** First, we must discretise the parameter space for $\boldsymbol{\theta}$ in the region of interest. The region $\boldsymbol{\Theta}$ represents the subset of $\mathbb{R}^k$ in which all of the parameters are defined. That is, $\boldsymbol{\Theta}$ is the Cartesian product of all Θ_is, where Θ_i is defined as the support for the parameter θ_i.

$$\boldsymbol{\Theta} = \bigtimes_{i=1}^{k} \Theta_i$$

In order to discretise the entire parameter space $\boldsymbol{\Theta}$, we first approximate the support for each parameter. To do so, we create a grid $\boldsymbol{w_i}$ which discretises Θ_i. For each $\boldsymbol{w_i}$ we specify an n_i, the number of intervals we are dividing the grid into. That is

$$\boldsymbol{w_i} = [w_i^0, w_i^1, \ldots, w_i^{n_i}]$$

where w_i^0 and $w_i^{n_i}$ denote the lower and upper bound respectively for the grid of parameter θ_i, which we must also specify. The aim of defining w_i^0 and $w_i^{n_i}$ is to allow us to obtain an accurate view of the posterior. If we choose w_i^0 and $w_i^{n_i}$ too narrow, we will miss areas of non-negligible density in the approximation, whilst if we define them too wide then we increase the computing time unnecessarily. Figure 1 suggests some suitable bounds that could be used to approximate the area under the posterior density of a normally distributed parameter. Having defined a grid $\boldsymbol{w_i}$ for each θ_i, we can now define the k-dimensional hyper-grid $\boldsymbol{W}$, which is the Cartesian product of the $\boldsymbol{w_i}$s.

$$\boldsymbol{W} = \bigtimes_{i=1}^{k} \boldsymbol{w_i}$$

The hyper-grid $\boldsymbol{W}$ is therefore a discretised approximation to the parameter space $\boldsymbol{\Theta}$, containing $n_W = \prod_{i=1}^{k} n_i$ points. For notational convenience, we let $\boldsymbol{W}_{a,b,\ldots,p}$ denote the point $(\boldsymbol{w_1}^a, \boldsymbol{w_2}^b \ldots, \boldsymbol{w_k}^p)$ when $a \in \{0, \ldots, n_1\}$, $b \in \{0, \ldots, n_2\}, \ldots, p \in \{0, \ldots, n_k\}$. In Figure 2, we

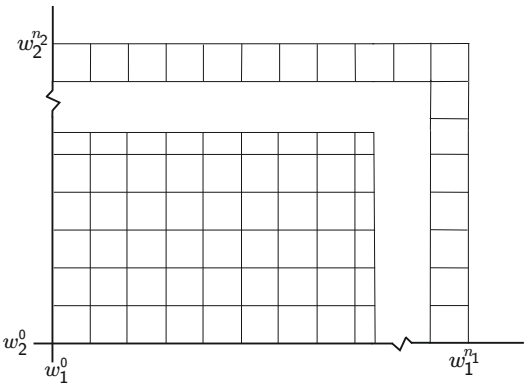


FIGURE 2. An example two-dimensional hypergrid $\boldsymbol{W}$, defined by the Cartesian product of grids for arbitrary parameters θ_1 and θ_2.

consider an example in which $k = 2$ and both grids $\boldsymbol{w_1}$ and $\boldsymbol{w_2}$ are define; $\boldsymbol{W}$ is therefore the collection of all points, displayed as a two-dimensional grid.

2.2. **Approximation and Implementation.** After defining the hyper-grid $\boldsymbol{W}$, we calculate the unnormalised posterior density at each point of $\boldsymbol{W}$. For a specific point, we first evaluate the log-likelihood; this is then exponentiated and multiplied by the prior dsitribution evaluated at that point. When applied at all points in $\boldsymbol{W}$, we obtain a k-dimensional matrix of unnormalised posterior densities.

$$p^*(\boldsymbol{W}_{a,b,\ldots,p} \mid \boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_m}) = \prod_{i=1}^{m} p(\boldsymbol{y}_i \mid \boldsymbol{W}_{a,b,\ldots,p}) \pi(\boldsymbol{W}_{a,b,\ldots,p})$$

To obtain a hyper-grid of normalised posterior densities, we have to approximate the normalisation constant, the marginal-likelihood. The marginal-likelihood, as denoted by the integral over the region $\boldsymbol{\Theta}$, signifies the area under the unnormalised posterior density. We approximate this integral with a multi-dimensional Riemmann sum over the hyper-grid $\boldsymbol{W}$ [6].

$$\begin{aligned}\int_{\boldsymbol{\Theta}} p(\boldsymbol{y} \mid \boldsymbol{\theta}) \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} &= \int_{\Theta_1} \cdots \int_{\Theta_k} \pi(\theta_1, \ldots, \theta_k)\, p(\boldsymbol{y} \mid \theta_1, \ldots, \theta_k)\, d\theta_1 \cdots d\theta_k \\ &\approx \sum_{w_1 = w_1^0}^{w_1^{n_1}} \cdots \sum_{w_k = w_k^0}^{w_k^{n_k}} \pi(w_1, \ldots, w_k)\, p(\boldsymbol{y} \mid w_1, \ldots, w_k)\, \Delta w_1 \cdots \Delta w_k\end{aligned}$$

The Δw_i terms represent the step size for the grid of parameter θ_i respectively, and can be calculated as:

$$\Delta w_i = \frac{w_i^{n_i} - w_i^0}{n_i}, \qquad \text{for } i = 1, \ldots, k$$

Defining the the hyper-grid $\boldsymbol{W}$ to approximate $\boldsymbol{\Theta}$ is an ambiguous process since we typically do not know before doing our calculations where the posterior will be centered and what shape it may take. This incurs a level of trial and error when performing this approximation. In most cases it is best to make the lower and upper bounds wide originally and then re-run the algorithm with more precise, narrower bounds. Having wide grid boundaries will also reduce the chance of viewing a single peak of a multi-modal posterior. To obtain an accurate approximation to the marginal-likelihood, however, wide bounds are not recommended; this is due to the number of points considered in the summation. With wide bounds, it is likely that many points in our hyper-grid $\boldsymbol{W}$ will have negligible density, and so evaluating the unormalised posterior at these points adds nothing to the value of the summation. There is a continuous trade-off between setting the bounds wide enough to capture the parameter space whilst ensuring the grid is computationally efficient [13].

2.3. **Grid Approximation for the Logistic Equation.**

2.3.1. *ODE System.* Now we will consider applying the grid approximation to some experimental data. In this application, we will consider a one-dimensional system which follows the Logistic equation [14] (also known as the Verhust-Pearl equation), that is

$$\frac{dN}{dt} = r\,N\left(1 - \frac{N}{K}\right), \tag{6}$$

where $N(t)$ denotes the population or density of a population, depending on the use of the model. We would like to perform inference on the two parameters $(\theta_1, \theta_2) = (r, K)$ that occur in the ODE system. The parameter r denotes the intrinsic growth rate of the system whilst K is the carrying capacity. This system has a closed form solution, which can be written as

$$N(t) = \frac{\left(\frac{KN_0}{K-N_0}\right)e^{rt}}{1 + \frac{1}{K}\left(\frac{KN_0}{K-N_0}\right)e^{rt}}, \quad \text{where } N(0) = N_0 \text{ is the initial condition.}$$

For this particular problem, we will consider the initial value $N(0)$ to be known. This allows us to obtain a unique solution for the system at the specified time points, given values of K and r. We will simulate data from this system using the method described in Section 1.3.2. For this particular example, we set the parameter values $r = 1.2$ and $K = 4.6$, whilst adding Gaussian noise to the solution with a standard deviation of $\sigma = 0.5$, to obtain the experimental data which is displayed in Figure 3.

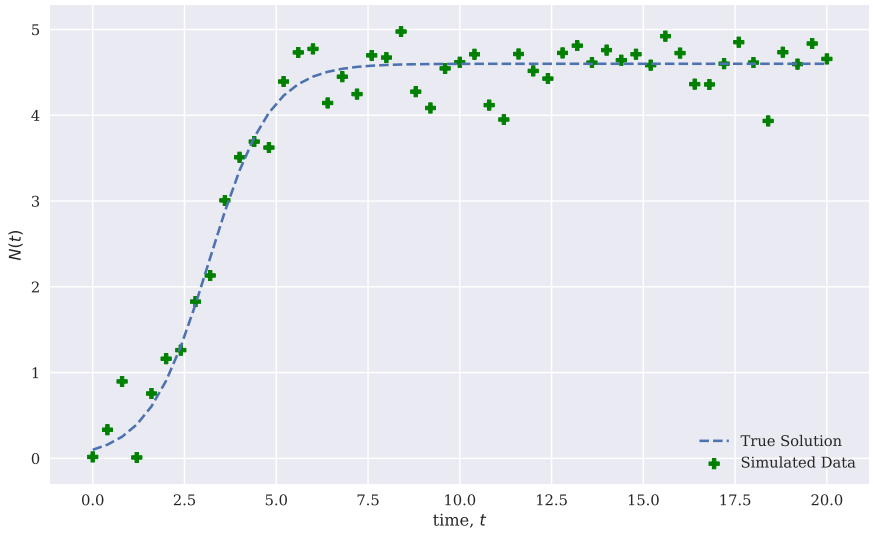
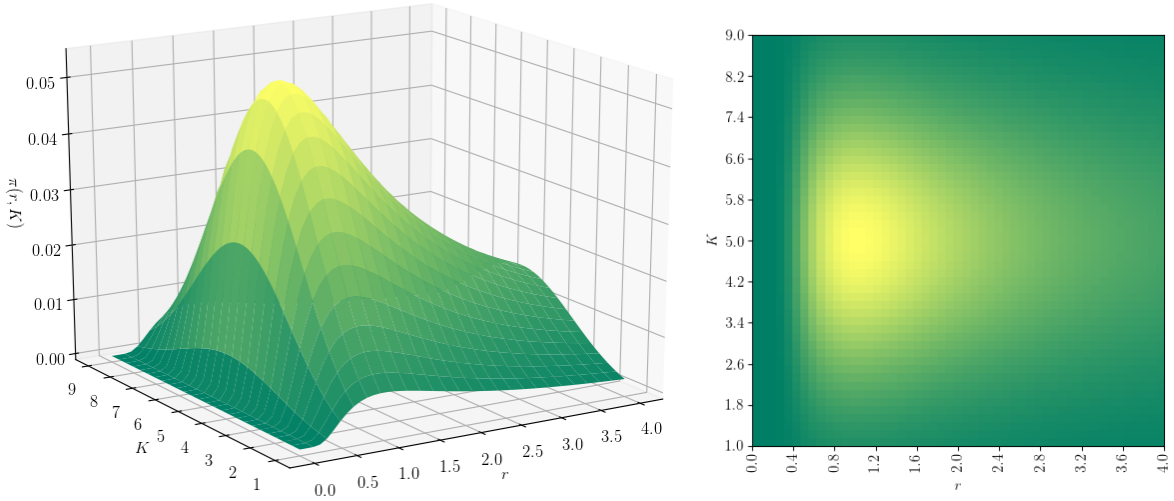


FIGURE 3. The simulated data from the ODE given by Eq.(6) with $r = 1.2$, $K = 4.6$ and $N_0 = 0.1$.

2.3.2. *Priors.* Prior distributions are then assigned to the parameters r and K independently. We place an Inverse-Gamma distribution on r with shape parameter $a = 1$ and scale 2. We use an Inverse-Gamma distribution prior for r because we believe it is a non-negative growth rate, the shape of the Inverse-Gamma distribution captures this belief. For K we place a Normal distribution prior, with a mean of 5 and standard deviation of 2. Figure 4 displays the probability density function of the joint prior $\pi(r, K)$ for the model parameters r and K.

$$r \sim \text{Inv-Gamma}(1, 2), \qquad K \sim \mathcal{N}(5, 4)$$



(A) Surface plot

(B) Heat plot

FIGURE 4. Plots of the joint prior $\pi(r, K)$.

2.3.3. *Inference.* After several trial runs, we define a grid for each parameter by choosing the upper and lower bounds as well as the number of points on each grid. For this example, we choose $n_1 = n_2 = 100$ and the bounds displayed below.

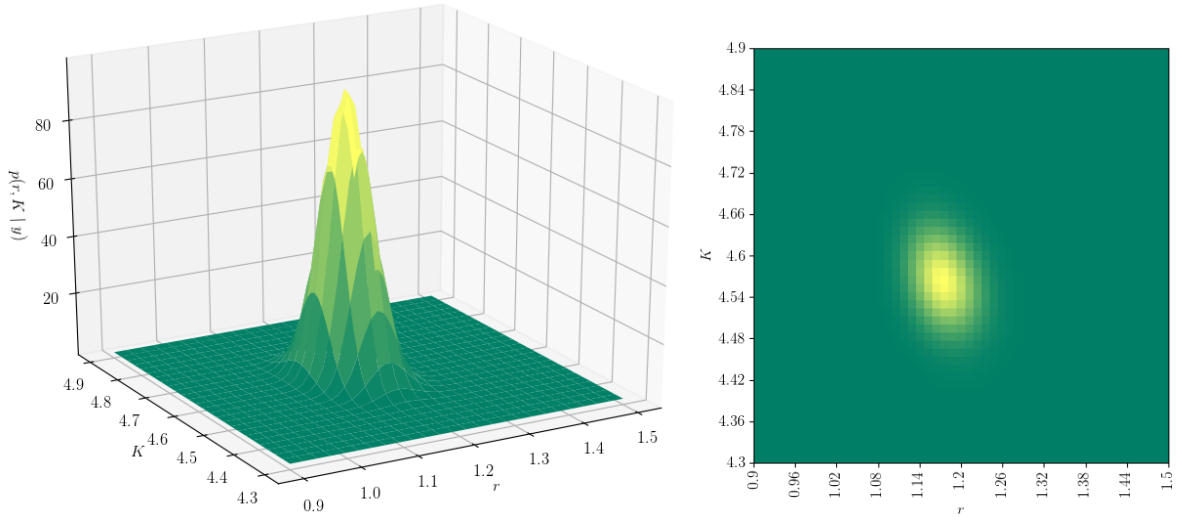
$$\begin{aligned} \theta_1 = r: \quad & w_1^0 = 0.9, \quad w_1^{100} = 1.5 \implies \Delta w_1 = 0.006 \\ \theta_2 = K: \quad & w_2^0 = 4.3, \quad w_2^{100} = 4.9 \implies \Delta w_2 = 0.006 \end{aligned}$$

These bounds are chosen after observing the posterior on a wide two dimensional grid and then visually identifying the region of non-negligible density. The plots for these grids along with the required code can be found in the `two_param_logitic_grid_approx.ipynb` file of my GitHub [5]. Now that we have approximated the parameter space $\boldsymbol{\Theta}$ to a region of non-negligible density, we can perform the grid approximation. First, we calculate the k-dimensional un-normalised posterior matrix. We then approximate the marginal-likelihood using the Riemann sum over the approximated parameter space, multiplied by the step-sizes Δw_1 and Δw_2. Finally, we divide the un-normalised posterior matrix by this approximation to obtain the posterior hyper-grid; this matrix can then be plotted, as shown in Figure 5.

2.3.4. *Evaluation.* By comparing the axes of the heat plots for both the prior and the posterior, we observe that the uncertainty about $\boldsymbol{\theta}$ has become more localised. To use the results of the grid approximation, we can consider the 95% credible intervals for each parameter, as discussed in Section 1.2.3. An alternative summary statistic would be to calculate a Highest Posterior Density Region for the joint posterior; this would produce an ellipse in which the posterior density of every (r, K) is above a certain value. Instead, I chose to calculate marginalised credible regions for each parameter because the marginalisation step also allows us to obtain an approximation to the posterior for a single parameter. To obtain the posterior for a single parameter, one again approximates the marginalisation integral by a Riemann sum.

$$\begin{aligned} p(\theta_1 \mid \boldsymbol{y}) &= \int_{\boldsymbol{\Theta}_2} p(\theta_1, \theta_2 \mid \boldsymbol{y})\, d\theta_2 \\ &\approx \sum_{w_2 = w_2^0}^{w_2^{n_2}} p(w_1, w_2 \mid \boldsymbol{y}) \Delta w_2 \end{aligned}$$

Now that we have obtained the posterior distribution for a single parameter, we can compute the respective 95% credible region. This statistic gives a region in which there is a 95% probability



(A) Surface plot (B) Heat plot

FIGURE 5. Plots of the posterior distribution $p(r, K \mid \boldsymbol{y})$ after the grid approximation.

that the model parameter is contained within the interval. By giving a credible region, we can quanitfy a region, smaller than the graphical axis above, in which it is likely that the parameter is contained. For this example of the Logistic Equation, we obtain the 95% credible region for both parameters, displayed in Figure 6 and Table 1.

	r	K
95% Credible Region	[1.146, 1.278]	[4.534, 4.720]

TABLE 1. The credible regions obtained from the grid approximation for parameters r and K.

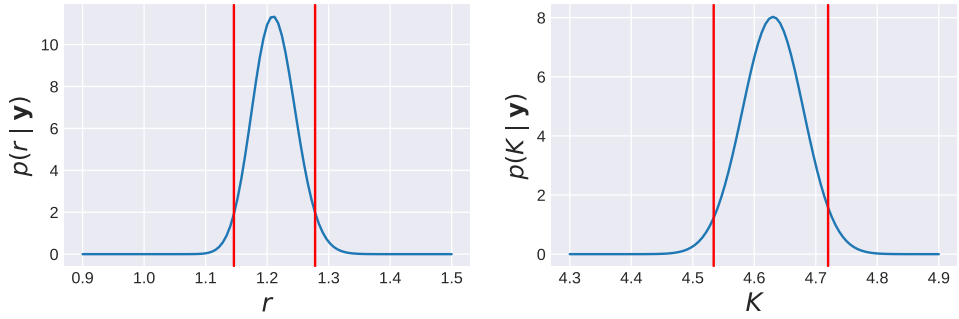


FIGURE 6. The marginalised probability density functions for r and K respectively, with the bounds of their 95% credible regions overlayed.

Using the bounds of the 95% credible region for both parameters, we can look at the range of solutions that arise from the values of parameters within these intervals. We do so by considering the pairs of boundary points that are the minimum $(1.146, 4.534)$ and maximum $(1.278, 4.720)$ of each 95% credible interval respectively. By computing and plotting the solution of the ODE system for both pairs of parameters we find a range for which the solutions within our credible regions are contained. We plot this range, with the experimental data overlayed, in Figure 7. We need only to compute the maximum and minimum pairs of the boundaries and not the other two combinations of extrema, $(1.146, 4.720)$ and $(1.278, 4.534)$, as they are contained within this region.

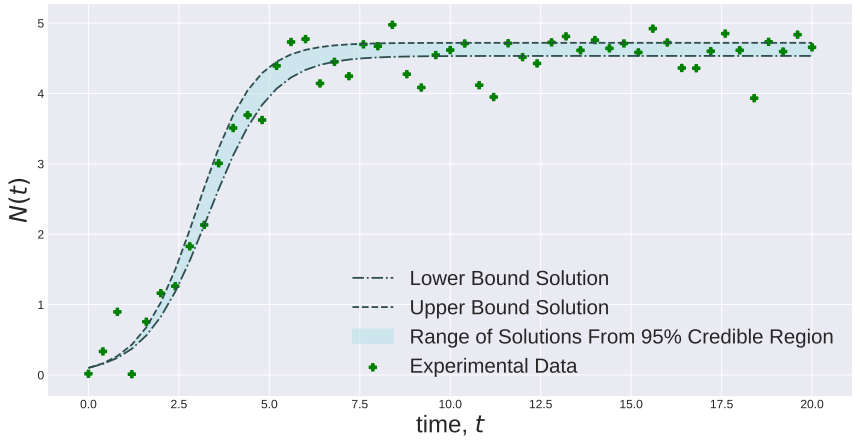


FIGURE 7. Range of solutions produced by the maximum and minimum bounds of the confidence intervals for parameters r and K.

2.4. **Evaluation of the Grid Approach.** The grid approximation allows us to compute an approximation to the posterior by discretising the parameter space and calculating the marginal likelihood for those points. Because we are considering the posterior at every point of the hyper-grid, this approach is computationally demanding when we consider a system which contains many parameters. For every element we add to the set of parameters $\boldsymbol{\theta}$, we add an entire dimension to the hyper-grid $\boldsymbol{W}$ we must consider. For example if we consider a system with 5 parameters, and for each parameter we choose 51 grid points then we must calculate the un-normalised posterior density at $n_{\boldsymbol{W}} \approx 340,000,000$ points. Doubling the number of parameters we consider (each with a grid of 51 points) means we have to consider approximately $340,000,000^2 \approx 1.2 \times 10^{17}$ grid points. For an average computer, this task will not be completed in any remotely reasonable time frame, even if we move from Python to a higher performance language. It may seem logical to therefore increase the step size Δw_i, so that we are considering fewer grid points. Increasing the step size however, will cause the Riemann sum approximation to become less accurate. In addition to the computational difficulties presented by the grid approach, we have also assumed that finding a bounded region for the parameter space $\boldsymbol{\Theta}$ is a relatively simple task. In low dimensions, one can easily identify regions of non-negligible density by visually analysing the heat plots of the posterior with wide grids. In higher dimensions, where multi-modal posteriors typically occur, visually analysing the heat plots for pairs of parameters can become cumbersome and ineffective. Despite these computational bottlenecks, when considering a low-dimensional posterior, the grid approximation has an advantage over sampling techniques like Markov Chain Monte Carlo. Unlike Markov Chain Monte Carlo, using the grid approximation to infer a complex or multi-modal posterior distribution is not more computationally demanding than a simple uni-modal distribution, provided that the time to evaluate the unnormalised posterior is the same. This brute-force method of approximate inference can yield accurate results for complex distributions in low dimensions, provided the grids are appropriately tuned to approximate the parameter space [15].

3. Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is a family of algorithms designed to produce samples from a distribution that is too difficult to sample directly. As the name suggests, they do so by using a large number of random samples, produced by a Markov process. The development and improvement of MCMC algorithms is a current topic of research in the field of Information Theory and Statistics, with its history dating back to Nicolas Metropolis in 1953 [1]. The applications of MCMC can be found in statistical physics and chemistry, however its most common use is in the Bayesian approach to probabilistic inference for complex models.

3.1. **Monte Carlo Integration.** Suppose we have a set of k random variables $\boldsymbol{\theta}$ under a distribution $P(\cdot)$ with a state space χ. Our task is to evaluate the expectation of a function of interest h under this distribution. Analytically, this is

$$\mathbb{E}_P\left[h(\boldsymbol{\theta})\right] = \frac{\int_\chi h(\boldsymbol{\theta})\, P(\boldsymbol{\theta})\, d\boldsymbol{\theta}}{\int_\chi P(\boldsymbol{\theta})\, d\boldsymbol{\theta}}. \tag{7}$$

We can use Monte Carlo integration to evaluate $\mathbb{E}_P\left[h(\boldsymbol{\theta})\right]$. We do this by drawing a set of samples $\{\boldsymbol{\theta}^{(t)},\, t = 1, \ldots, n\}$ directly from the distribution $P(\cdot)$. By the Law of Large Numbers, the empirical average of these samples under the function h will converge to the expectation in Eq.(7) [16].

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} h(\boldsymbol{\theta}^i) \to \mathbb{E}_P\left[h(\boldsymbol{\theta})\right]$$

3.1.1. *Application.* In Bayesian inference, when we are faced with non-conjugate likelihood-prior pairs, we wish to find an estimate for the marginal-likelihood $p(\boldsymbol{y}) = \int_{\boldsymbol{\Theta}} p(\boldsymbol{y} \mid \boldsymbol{\theta})\, \pi(\boldsymbol{\theta})\, d\boldsymbol{\theta}$, which can be given in the form of Eq.(7),

$$\mathbb{E}_\pi[p(\boldsymbol{y} \mid \boldsymbol{\theta})] = \int_{\boldsymbol{\Theta}} p(\boldsymbol{y} \mid \boldsymbol{\theta})\, \pi(\boldsymbol{\theta})\, d\boldsymbol{\theta}, \qquad \text{since} \int_{\boldsymbol{\Theta}} \pi(\boldsymbol{\theta})\, d\boldsymbol{\theta} = 1.$$

Therefore, by drawing independent samples from the prior distribution

$$\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(n)} \sim \pi(\boldsymbol{\theta})$$

we can approximate this expectation [17].

$$\mathbb{E}_\pi[p(\boldsymbol{y} \mid \boldsymbol{\theta})] \approx \frac{1}{n} \sum_{i=1}^{n} p(\boldsymbol{y} \mid \boldsymbol{\theta}^{(i)}) \tag{8}$$

In practice, however, when we are are considering a finite number of samples, this estimate can be unstable and inefficient. This is due to many of the samples from π not visiting regions of high likelihood [10]. There are other techniques for which we can estimate the marginal-likelihood using independent samples, namely importance and rejection sampling [16], however they will not be discussed here. For the set of samples to satisfy Eq.(8), they must be independent. Typically, due to their Markovian nature, MCMC methods produce *dependent* samples from their target distribution but, when the algorithms satisfy certain criteria, we can produce a result similar to the one above.

3.2. **Properties of MCMC Algorithms.** In this section we will briefly discuss the properties of Markov Chain Monte Carlo algorithms that allow them to generate samples from a probability distribution. We will be considering the general case of MCMC sampling, looking at how MCMC produces samples from any probability density $P(\boldsymbol{\theta})$ (rather than our particular posterior problem). Since the aim of this report is to explore computational techniques for parameter inference in ODE systems, the details of many of the following proofs, statements and remarks can be found in their respective references. To aide in the explanation of Markov chains and their stochastic nature, we will also make use of terminology and theory associated with Markov processes, described in detail in [18]. Prior to dissecting the theory behind MCMC algorithms, we must introduce some requirements for the probability function that we wish to sample. First, we should be able to directly evaluate either the probability density itself or the un-normalised probability density of this distribution, $P^*(\boldsymbol{\theta})$. Secondly, $P^*(\boldsymbol{\theta})$ must be a non-negative valued function that integrates to a finite, positive value [16]. Hence,

$$P(\boldsymbol{\theta}) = \frac{P^*(\boldsymbol{\theta})}{Z}, \quad \text{where } Z = \int_\chi P^*(\boldsymbol{\theta}) d\boldsymbol{\theta}.$$

Definition 1. A *Markov Chain Monte Carlo* method for the simulation of a distribution P is any method that constructs an *ergodic* Markov chain $(X^{(t)})$ such that P is an invariant, or stationary, distribution of the chain [17].

After specifying an initial starting value $x^{(0)}$, a chain $(X^{(t)})$ is generated using a *transition kernel* T. The transition kernel is a function that describes how we move from one state in the chain to the next.

$$X^{(t)} \xrightarrow{T} X^{(t+1)}$$

Definition 2. If a Markov chain with transition kernel T satisfies the *detailed balance condition*, with respect to a distribution P, then the chain is *reversible* and P is an invariant distribution of the chain.

$$T(\boldsymbol{x}, \boldsymbol{y})P(\boldsymbol{y}) = T(\boldsymbol{y}, \boldsymbol{x})P(\boldsymbol{x}) \qquad \text{for every } (\boldsymbol{x}, \boldsymbol{y}) \tag{9}$$

To satisfy the detailed balance condition, it therefore must be just as likely to draw a state $\boldsymbol{x}$ from the target distribution P, then move from the state $\boldsymbol{x}$ to $\boldsymbol{y}$ under the transition kernel T, as it so to draw $\boldsymbol{y}$ from P then move from $\boldsymbol{y}$ to $\boldsymbol{x}$ under T [16].

3.2.1. *Markov Chain Strong Law of Large Numbers.*

Definition 3. A Markov chain is *aperiodic* if the transition kernel T allows for the event $\{X^{t+1} = X^t\}$ with a non-zero probability. More formally, if the greatest common divisor of the the time taken for the chain to return to any state is equal to one then the chain is aperiodic.

Definition 4. A Markov chain is *irreducible* if it is possible for every state i to visit every other state j in finite time. That is, for all i and j, there exists an m such that

$$\mathbb{P}(X^{(n+m)} = j \mid X^{(n)} = i) > 0.$$

Because of the continuous nature of the Markov chains in MCMC, this definition has been adapted slightly to be more applicable, the reader is directed to [19] for further details.

Lemma 1. *If a Markov chain $(X^{(t)})$ is aperiodic and irreducible (or Harris recurrent with respect to a particular measure* [19]*), and has a stationary distribution P, then a "Strong Law of Large Numbers" holds* [17]*:*

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} h(X^{(i)}) = \int_{\chi} h(\boldsymbol{\theta})P(\boldsymbol{\theta})\, d\boldsymbol{\theta} = \mathbb{E}_P[h(\boldsymbol{\theta})]. \tag{10}$$

What it means to be Harris-recurrent, along with a proof of the above statement, can be found in the survey [19].

In addition to the result from Lemma 1, it can be shown that the distribution of an aperiodic and irreducible chain $(X^{(t)})$ converges to the target distribution as $n \to \infty$. If we consider the n-th sample of the chain, denoted $X^{(n)}$, then

$$\lim_{n\to\infty} ||X^{(n)} - P|| \to 0, \tag{11}$$

where $||\cdot||$ denotes total variation distance [19]. This implies that although the distribution of the chain $X^{(t)}$ will reach that of the target distribution when we consider infinite time, the chain will never reach this distribution in finite time. Samples produced from a chain are therefore *approximate* samples, rather than true samples. In this report we will say that a chain reaching, or converging to, its target distribution to mean that, according to some diagnostic tools discussed in Section 3.4, the total variation distance is smaller than some acceptable bound. Once a chain has converged to the target distribution, we will also say that it produces samples and approximate samples interchangeably.

For an MCMC algorithm to produce approximate samples from the target distribution, the chain must have first converged to this distribution from its initial state. The time taken for the chain to reach its stationary distribution is non-deterministic in nature. Since practicality dictates that we can only work with a finite set of samples, we have to consider what proportion

of these samples are prior to the chain converging to the target distribution. Only after the chain has reached the target distribution will we be able to approximate Eq.(7). If we consider a chain of length n where b is the point in which the chain has reached the stationary distribution, then we can construct a practical version of Eq.(8),

$$\mathbb{E}_P[h(\boldsymbol{\theta})] \approx \frac{1}{(n-b+1)} \sum_{i=b}^{n} h(X^{(i)}).$$

In addition to approximating the marginal-likelihood, the samples can be used to provide credible intervals for the random variables. For low-dimensional $\boldsymbol{\theta}$ (or subsets of $\boldsymbol{\theta}$), we can also construct histograms of our samples, approximating the surface of the posterior distribution. The construction of histograms, or the density estimation obtained from the histograms, allow us to visually identify areas of multi-modality in the posterior, or correlation between model parameters.

3.3. **Metropolis-Hastings.** The Metropolis-Hastings (M-H) algorithm is a Markov Chain Monte Carlo method that allows us to produce approximate samples from multi-dimensional probability distributions. The name of the algorithm is derived from the contributions of two scientists; a physicist named Nicholas Metropolis, responsible for creating the base of the algorithm [1] and a statistician W.K Hastings who later generalised the method [20]. The Metropolis-Hastings algorithm, along with the Gibbs Sampler [21] (which will not be discussed) are two of the most popular MCMC methods due to their implementation in computer programmes like `WinBUGS` [22] or coding packages such as `PyMC3` [23].

3.3.1. *The Algorithm.* We first initialise the chain $(X^{(t)})$ at an initial state $x^{(0)}$, with the requirement that $x^{(0)}$ is chosen so that $P^*(x^{(0)})$ is non-zero. To move from one point to another within the Markov chain, the algorithm requires us to specify a *proposal distribution* $q(y \mid x)$ which, as its name suggests, proposes the next step of the chain. The proposal distribution is a probability density function which depends on the current state of the chain and, when sampled from, gives the next proposed step. The following algorithm decides how we move from our starting initial value to the next state, this is then repeated for a desired number of steps.

Algorithm 1 Metropolis-Hastings

Given $X^{(t)} = x^{(t)}$
Generate $Y_t \sim q(y \mid x^{(t)})$
Take

$$X^{(t+1)} = \begin{cases} x^{(t)} \text{ with probability } \alpha(x^{(t)}, Y_t) \\ Y_t \text{ with probability } 1 - \alpha(x^{(t)}, Y_t) \end{cases}$$

Where

$$\alpha(x^{(t)}, Y_t) = \min\left\{ \frac{P^*(Y_t)}{P^*(x^{(t)})} \frac{q(x^{(t)} \mid Y_t)}{q(Y_t \mid x^{(t)})},\ 1 \right\}$$

3.3.2. *Detailed Balance and Convergence.* As described in the previous section, if the transition kernel T satisfies the *detailed balance condition* given by Eq.(9), then P is a stationary distribution of the chain produced. The transition kernel for the M-H algorithm can be written as

$$T(x, y) = q(y \mid x)\, \alpha(x, y) + \delta_x(y)(1 - r(x)) \tag{12}$$

where $r(x) = \int_\chi \alpha(x, y)\, q(y \mid x)\, dy$ and $\delta_x(y)$ is the Dirac mass in x [24]. We arrive at this form by considering the cases in which $y \neq x$ and $y = x$ separately. When $y \neq x$, we must first propose the new step y and then accept with it according to the method in Algorithm 1. For $y \neq x$ we therefore have $T(x, y) = \alpha(x, y) q(y \mid x)$; this occurs with probability $r(x)$. When we consider the case in which the chain remains at x, we look to two separate sub-cases. Firstly, we

consider the case when the proposed new step y is the same as the current state x, this occurs with probability $q(y \mid x)$ and is automatically accepted by the chain. Secondly, we consider the event where the chain does do not accept the proposed step, according to the rule described in Algorithm 1, this occurs with probability $1 - r(x)$. Combining the two sub-cases and the generic case of $y \neq x$, we obtain Eq.(12). A full proof and explanation of how we arrive at this transition kernel can be found in [24]. Now that we have a transition kernel for the algorithm, we can test that it satisfies the detailed balance condition. We will consider only the case when $x \neq y$, since the case of equality is trivial.

$$\begin{aligned} T(x,y)P(x) &= P(x)\, q(x \mid y)\, \min\left\{\frac{P^*(y)}{P^*(x)}\frac{q(x \mid y)}{q(y \mid x)},\, 1\right\} \\ &= Z^{-1}\, P^*(x)\, q(x \mid y)\, \min\left\{\frac{P^*(y)}{P^*(x)}\frac{q(x \mid y)}{q(y \mid x)},\, 1\right\} \\ &= Z^{-1}\, \min\left\{P^*(y)\, q(y \mid x),\; P^*(x)\, q(x \mid y)\right\} \end{aligned}$$

Here, Z is the integral previously defined in Section 3.2. We see that the last line is symmetric with respect to x and y and therefore satisfies the detailed-balance equations.

In addition to satisfying Eq.(9), the chain produced by the M-H method must also be an *ergodic* Markov Chain (a chain that is both aperiodic and irreducible). It can be shown that the Metropolis-Hastings algorithm satisfies these properties. Samples produced by this algorithm will therefore satisfy the Law of Large Numbers described in Lemma 1. Details, including the proof of such properties, can be found in [25].

3.3.3. *Practical Implementation of Metropolis-Hastings.* Although the Metropolis-Hastings algorithm is guaranteed to converge to the target distribution in infinite time, we only have the luxury of working with a finite set of samples. Because of this, there has been some developments in the optimisation of MCMC methods, that either reduce the time taken for the chain to converge to the target distribution, or maximise the region of $\boldsymbol{\Theta}$ which the chain explores. Typically, as is the convention in this report, a Gaussian distribution is used for the proposal distribution, that is

$$q(y \mid x) \sim \mathcal{N}(x, \Sigma), \quad \Sigma = \begin{pmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_k^2 \end{pmatrix}$$

where Σ is a given, diagonal, covariance matrix. The optimisation of the M-H algorithm comes in the form of tuning this covariance matrix, since the values in this matrix are arbitrarily set by the user.

Large variances within Σ can to the algorithm proposing points that are far away from the current point. Typically, many of these proposed points will fall into regions of negligible density, causing a low proportion of proposed steps to be accepted. On the other hand, small variance values in Σ will mean proposals are near to the current point and are more likely to be accepted, due to their comparable densities. In addition, for small variances, the chain will take a long time to fully explore the parameter space and converge to the target distribution [26]. The proportion of steps being accepted is commonly referred to as the *acceptance ratio* of the chain. A method proposed by Gelman [26], suggests that the the covariance matrix should be tuned to so that we are accepting the proposed steps of the chain with a rate of approximately $\frac{1}{4}$. It is also suggested that by Roberts [27] that values within Σ should be updated prior to the chain's convergence. More precisely, we estimate each σ_i^2 by calculating the variance of the previously accepted states; the variance estimate is then multiplied by a particular scaling factor, given in Eq.(13). This method implies that we must continually monitor the acceptance ratio prior to convergence and, if the ratio strays too far from $\frac{1}{4}$, re-estimate the covariance matrix for the proposal distribution using the previously accepted states. It is important to

note that this is only performed prior to the chain's convergence to the target distribution; once the chain reaches its stationary distribution, we must keep Σ constant. Failing to do so can alter the ergodic properties of the chain. In the estimation of the covariance matrix for the proposal distribution, we set

$$\Sigma = \frac{2.4}{\sqrt{k}}\hat{I}, \quad \text{where } k \text{ is the dimension of } \boldsymbol{\theta} \text{ and } \hat{I} = \begin{pmatrix} \hat{\sigma_1^2} & 0 & \dots & 0 \\ 0 & \hat{\sigma_2^2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \hat{\sigma_k^2} \end{pmatrix}. \tag{13}$$

These alterations are given for the general use of the Metropolis-Hastings method. Target distributions with a complex topology may require different approach. In particular, the previously described covariance matrix tuning may not be optimal if the parameters are highly correlated [27]. We may also encounter target distributions with surfaces that may require a proposal distribution different to a Gaussian.

3.4. **Diagnostics of the Posterior.** As discussed in the previous subsection, we found that certain MCMC methods, like Metropolis-Hastings, will produce samples from the target distribution once the Markov chain has converged to its stationary distribution. The time taken for the chain to reach this point of convergence is known as the *burn in* period. In practice, however, finding the point at which the chain converges to the target distribution can be difficult. There are several popular diagnostic algorithms and statistics that can help determine the convergence of a chain, proposed by Geweke [28], Gelman and Rubin [29] and Raftery and Lewis [30]. In this report, we will be considering only the Gelman-Rubin $\widehat{R}$ diagnostic. The topic of convergence diagnostics is a current area of research in the theory of Markov Chain Monte Carlo methods, with modifications of the $\widehat{R}$ diagnostic having been proposed in early 2019 [31].

3.4.1. *The Gelman-Rubin $\widehat{R}$ Diagnostic.* The $\widehat{R}$ method proposed in 1992 [29] entails running m Markov chains, each from a different starting points x_m^0, for n iterations. The method tries to diagnose convergence by assessing the *mixing* and *stationarity* of the chains [26]. The reason for using multiple chains can be explained using Figure 8. In this toy example, we have run two chains from two different starting points. By looking at each chain separately, it would appear that the chains have converged to a stationary distribution, however when we consider them together, we can see that they are exploring two different parts of the target distribution and therefore have not converged. Problems like this, named *slow-mixing problems* [26], often occur when the target distribution is multi-modal; this can cause the a chain to get stuck at a particular peak. From Figure 8, we can see that we need to consider the *mixing* of multiple chains in order to ensure we have reached a common stationary distribution. The approach proposed by Gelman and Rubin is to consider the *between* and *within* variances of the chains. Suppose we run m changes independently, let $X_j^{(i)}$ denote the i-th sample of the j-th chain, that is

$$X_j^{(1)}, X_j^{(2)}, \dots, X_j^{(n)}$$

is the history of the j-th Markov Chain. For each chain, we calculate the within-sequence variance W and the between-sequence variance B:

$$B = \frac{n}{m-1}\sum_{j=1}^{m}(\bar{X}_j - \bar{X})^2, \quad \text{where} \quad \bar{X}_j = \frac{1}{n}\sum_{i=1}^{n} X_j^{(i)} \quad \text{and} \quad \bar{X} = \frac{1}{m}\sum_{j=1}^{m}\bar{X}_j,$$

$$W = \frac{1}{m}\sum_{j=1}^{m} s_j^2, \quad \text{where} \quad s_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_j^{(i)} - \bar{X}_j)^2.$$

From these statistics, we can estimate the variance of the target distribution by using a weighted average of W and B, given by $\hat{\text{var}}(X)$. Such an estimator overestimates the target variance but

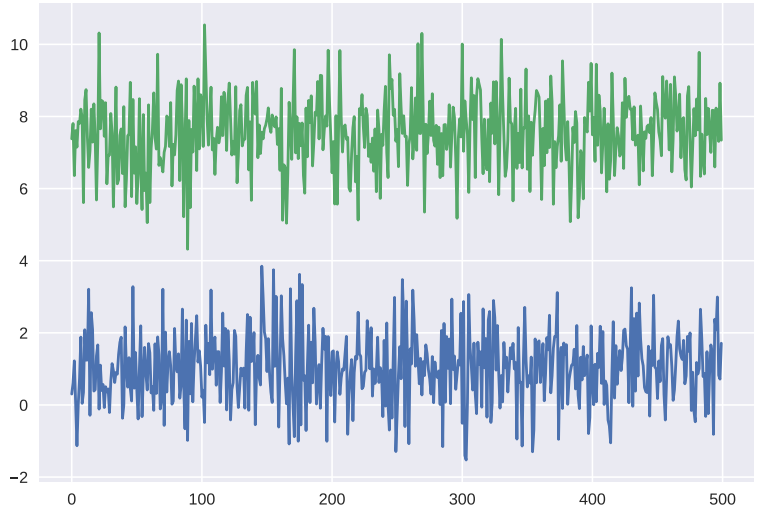


FIGURE 8. A toy example displaying how two chains looked to have converged to a stationary distribution but are in fact exploring two different peaks.

is unbiased when the chain starts in the stationary distribution or when $n \to \infty$ [29].

$$\hat{\text{var}}(X) = \frac{n-1}{n}W + \frac{1}{n}B$$

Meanwhile, the within-sequence variance W underestimates $\hat{\text{var}}(X)$, since the individual chains will never explore the full parameter space with finite n. When we take $n \to \infty$ however, the expectation of $W \to \hat{\text{var}}(X)$. We asses convergence by monitoring the ratio of both estimators, which approaches 1 from above as $n \to \infty$.

$$\widehat{R} = \sqrt{\frac{\hat{\text{var}}(X)}{W}} \tag{14}$$

Typically, when performing simulation, we exit the burn-in period when $\widehat{R} \leq 1 + \delta$ for some chosen δ (typically 0.1 [26]). To implement this statistic, we iteratively perform simulation of m chains using an MCMC method, stopping after some predetermined n to calculate Eq.(14). If the chain does not exit the burn-in period, then we could continue the chain and re-evaluate Eq.(14) after another n; if, instead, the chain does exit the burn-in period, then we can run the chain to obtain approximate samples from the stationary, target distribution.

3.5. **2-Parameter Logistic Equation Revisited with MCMC.** Now, we will revisit the inference problem that we explored with the grid approximation in Section 2.3. This time we will be using the Metropolis-Hastings algorithm to perform inference on the posterior $p(\boldsymbol{\theta} \mid \boldsymbol{y})$. We will approach the problem with the same data generated from the ODE model given in Eq.(6), and apply the same priors to the model parameters. To asses the convergence of the algorithm, we will use the $\widehat{R}$ diagnostic described in the previous section. We will exit the burn in period of the chains when the $\widehat{R} \leq 1.05$, a slightly stricter value than the that suggested in [29]. We consider this stricter value because of the discussion put forward in [32] regarding the use of $\widehat{R}$ on simple uni-modal models. We will use a Gaussian proposal distribution with a burn-in estimated covariance matrix, discussed in Section 3.3.3, after starting with an initial covariance matrix Σ.

$$\Sigma = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.5 \end{bmatrix}$$

For this particular case of inference, we will run 5 chains consecutively for 1500 iterations each. For the i-th chain, we must specify an initial starting point $\boldsymbol{x_i^0}$. To specify such a starting point, we will draw 5 samples from the priors of our parameters using samplers from the `scipy.stats` package, making sure that each $\boldsymbol{x^0}$ satisfies $p^*(\boldsymbol{x^0} \mid \boldsymbol{y}) > 0$.

θ_i	$\boldsymbol{x_1^0}$	$\boldsymbol{x_2^0}$	$\boldsymbol{x_3^0}$	$\boldsymbol{x_4^0}$	$\boldsymbol{x_5^0}$
r	1.272	0.827	0.517	1.782	1.525
K	6.630	4.617	3.561	6.135	4.488

TABLE 2. Starting points for each chain in the Metropolis-Hastings algorithm, sampled from the prior $\pi(r, K)$.

After specifying these initial conditions, we then run the Metropolis-Hastings algorithm on each chain. We run the 5 chains consecutively rather than simultaneously due to the estimation of the proposal distribution's covariance matrix. This means that we run the first chain for n iterations; over this time, the chain will estimate the covariance matrix if the acceptance ratio falls below 0.15 or rises above 0.35, this criterion is assessed every one hundred iterations. The covariance matrix will then be passed on to the proposal distribution of the second chain, which is also run for n iterations before being passed to the third chain, all the way to the fifth chain. This method reduces computing time when compared to running the chains simultaneously, since the covariance matrix estimation typically only occurs early in the first chain rather than the beginning of all five chains. Figure 9 shows the samples from all five chains for both model parameters, r and K. Such samples are often referred to as the *trace* of the MCMC algorithm and plots like Figure 9, *trace plots*.

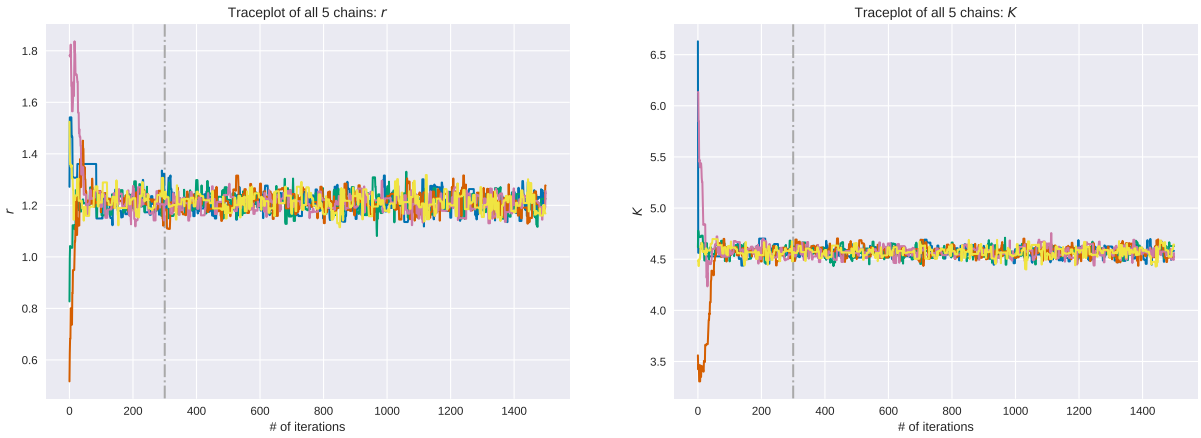
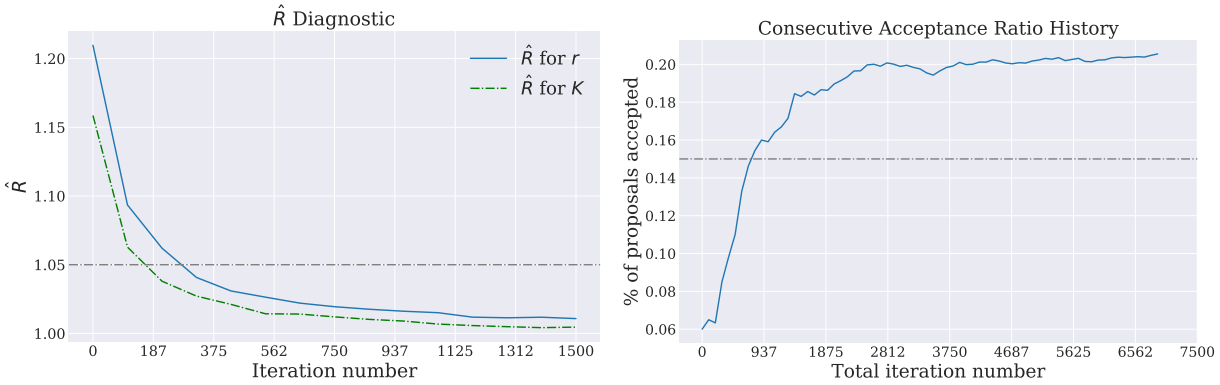


FIGURE 9. Trace plots for model parameters r and K respectively after running the Metropolis-Hastings algorithm.

To determine how many samples should be discarded as the burn in period, we look to the Gelman-Rubin $\widehat{R}$ diagnostic for the five chains. We compute the $\widehat{R}$ statistic for each parameter at every one hundred iterations of the chains. By doing so, we can monitor how the $\widehat{R}$ of the chains changes over time. To identify the region of burn in, we look to when the $\widehat{R}$ falls below the desired value of 1.05 for both parameters. We consider the point in which this happens for the first time as the start of our samples from the posterior; that is, all samples prior to that are discarded as the burn in. Since we are running the chains consecutively rather than simulanteously, the $\widehat{R}$ diagnostic is assessed after we have run the chains for n iterations. If the $\widehat{R}$ values for both parameters have not fallen below the desired range then we restart the algorithm and increase the value of n. From Figure 10a, one can identify that the first time



(A) $\widehat{R}$ Diagnostic for both parameters.

(B) Acceptance ratio over the history of all five chains, considered end to end.

FIGURE 10. Plots of the $\widehat{R}$ diagnostic and acceptance ratio for 5 chains produced, using the Metropolis-Hastings algorithm.

the $\widehat{R}$ falls below 1.05 for both parameters is at the third evaluation of the $\widehat{R}$ statistic. This implies that we discard the first 299 samples of each chain, for both parameters, as a burn in. The vertical grey dashed line in Figure 9 shows where the 300-th sample lies.

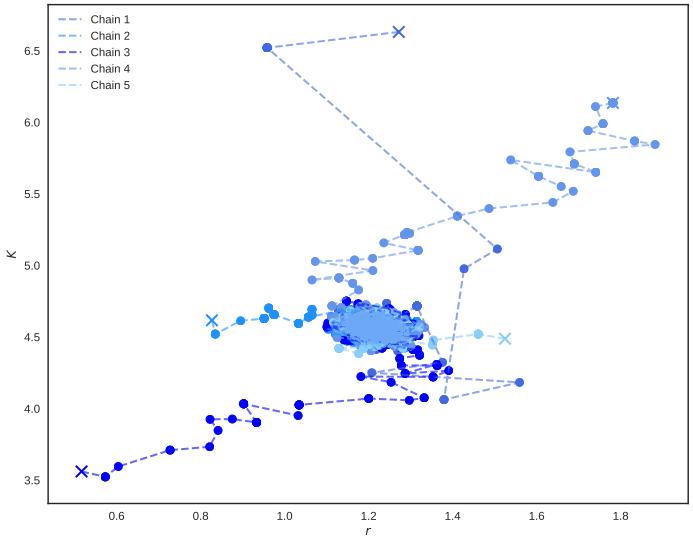


FIGURE 11. Monitoring how all five chains explore the parameter space from their initial states.

3.5.1. *Inference.* Now that we separated the chains into the burn in period and the samples, we can use the samples to make approximations to expectations, like those in Section 3.2.1. In particular we can approximate the posterior mean for both parameters by calculating the empirical average of the samples across all the chains, that is

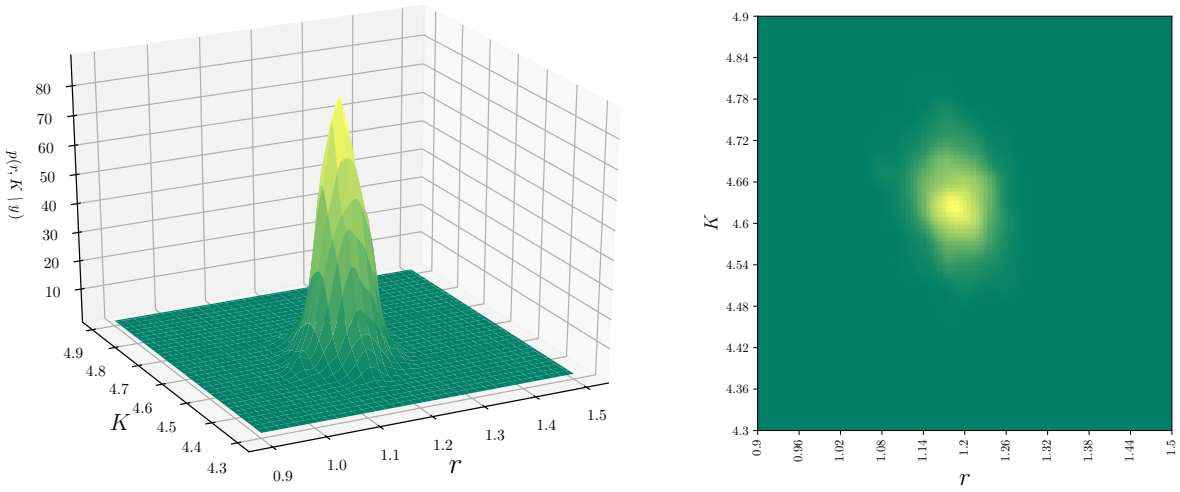
$$\hat{\boldsymbol{\theta}} = \frac{1}{n\,m}\sum_{j=1}^{m}\sum_{i=1}^{n}\boldsymbol{\theta}_j^{(i)}$$

where we denote $\boldsymbol{\theta}_j^{(i)}$ as the i-th sample of chain j produced by the MCMC method, after discarding the burn in period. In addition to computing the empirical average, we can also approximate the 95% credible intervals, given in Table 3.

θ_i	$\hat{\theta}$	95% Credible Interval
r	1.28	[1.144, 1.282]
K	4.689	[4.471, 4.664]

TABLE 3. Posterior mean and 95% credible intervals obtained from running the Metropolis-Hastings algorithm.

Since $\boldsymbol{\theta}$ is low-dimensional, we could use a three-dimensional histogram to visually analyse how the samples approximate the surface of the posterior. Instead of using histograms, we can obtain a smoothed approximation to the posterior surface by using Kernel Density Estimation (KDE). Kernel Density Estimation is a non-parametric smoothing method that can be used to estimate the probability density function of a random variable. A detailed account of KDE and its implementation can be found in [33]. For this specific case we will be using a multivariate Gaussian density kernel with a bandwidth of 0.3. By using KDE, we can take the samples from the Metropolis-Hastings algorithm and produce a grid on which we have estimated the probability density function of the posterior at each point, displayed in Figure 12. To view the practical implementation of the Metropolis-Hastings algorithm to this ODE model, and observe the results given in this section, the reader is directed to the `two_param_mcmc_logisitc_eqn.ipynb` file of [5].



(A) Surface Plot

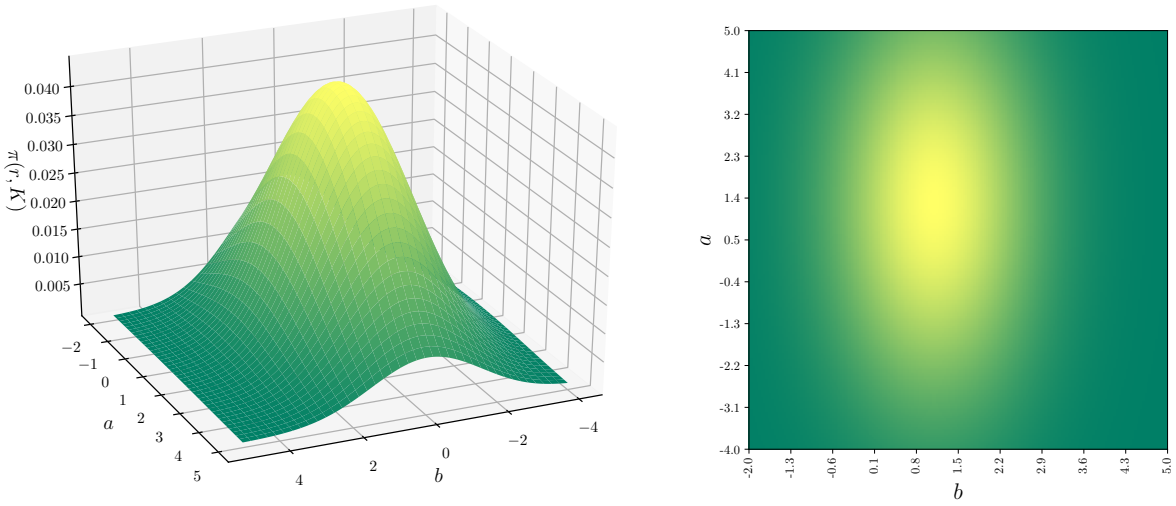
(B) Heat Plot

FIGURE 12. Surface and Heat plots of the approximation to the posterior for parameters r and K, after performing Kernel Density Estimation.

3.6. **Two Dimensional Model.** In this section, we will explore an ODE system which produces a more complex posterior surface. In this case, we will run the Metropolis-Hastings algorithm to see if any issues arise when the surface is not uni-modal.

3.6.1. *ODE System.* We will be considering an artificially created ODE, described below.

$$\begin{cases} \dot{x} = -x + g(a)\,xy, & g(a) = -\dfrac{1}{20}\left(a - \dfrac{1}{2}\right)^2 + 2 \\ \dot{y} = y - f(b)\,xy, & f(b) = 1 + \dfrac{3}{20}\sin(b) \end{cases}$$



(A) Surface Plot

(B) Heat Plot

FIGURE 13. Surface and Heat plot of the joint prior, $\pi(a, b)$.

We wish to perform inference on the parameters $\boldsymbol{\theta} = (a, b)$. In the case of this ODE, we could set $\alpha = g(a)$ and $\beta = f(b)$ and perform inference on α and β, however we pretend that this is an ODE with some physical meaning, be it biological or mechanical. We therefore assume that $g(a)$ and $f(b)$ denote some type of predation or competition rates between the variables. This then implies that a and b have some physical interpretation, and so we would like to perform direct inference on the parameters themselves. In addition, a complex posterior distribution only arises when we consider $\boldsymbol{\theta} = (a, b)$ rather than $\boldsymbol{\theta} = (\alpha, \beta)$.

3.6.2. *Priors and Data Simulation.* In this particular setting, we define the priors for a and b independently. Figure 13 displays a heat and surface plot of the joint prior over a subset of $\mathbb{R}^2$.

$$a \sim \mathcal{N}(1.3, 9), \quad b \sim \mathcal{N}(1.1, 1.44)$$

We also simulate some experimental data from this ODE system. First, we define the true values of the parameters, $\boldsymbol{\theta} = (2.1, 0.6)$. We then numerically integrate the system from $t = 0$ to $t = 20$ at 21 time points using the `odeint` function of the `scipy.integrate` package. Again, we consider the initial conditions to be known, in this case we set $x(0) = 2$ and $y(0) = 1$. After obtaining the numerical solutions to the system, we then disrupt this by adding zero-mean gaussain noise to both x and y variables, with variances $\sigma_x^2 = 0.3$ and $\sigma_y^2 = 0.15$ respectively. This data is displayed in Figure 14.

3.6.3. *Inference.* To obtain samples from the M-H algorithm, we run the sampler in the same manner as described in the previous section. For each proposed step, however, we must now numerically integrate the system over the time span using the proposed model parameters; this drastically increases the computing time of each chain, since in the previous example we simply had to input the parameters into the closed solution. In addition to performing the Metropolis-Hastings algorithm on this system, we will also perform the grid approximation. The grid approximation is used as a benchmark for the M-H algorithm due to its accuracy when we consider a fine grid in low dimensions. When running the M-H algorithm, we will again initialise five chains with starting points sampled from the joint prior, given in Table 4. We run each chain for 5000 iterations and exit the burn in period when the $\widehat{R}$ falls below 1.05. Figure 15 displays the traceplots produced for both parameters. The plots given in this section as well as the code used to implement the Metropolis-Hastings algorithm on this ODE model can be found in the file `complex_lotka_volterra.ipynb` of [5]. From analysing the $\widehat{R}$ plot for

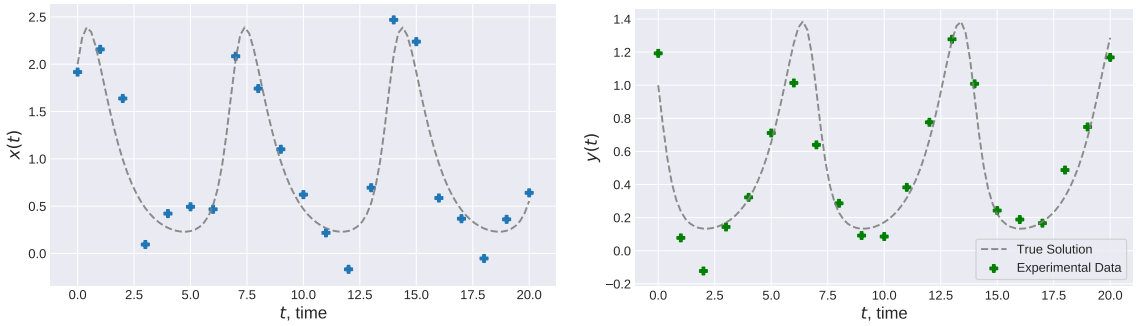


FIGURE 14. Plots of the true solution of $x(t)$ and $y(t)$, from the ODE model given in Section 3.6.1. Simulated data generated with true parameter values $(a, b) = (2.1, 0.6)$ and initial conditions $x(0) = 2$ and $y(0) = 1$.

θ_i	x_1^0	x_2^0	x_3^0	x_4^0	x_5^0
a	-0.110	3.814	2.363	1.259	0.865
b	-1.104	1.156	1.638	4.044	2.796

TABLE 4. Initial states for each chain in the Metropolis-Hastings algorithm, sampled from the prior $\pi(a, b)$.

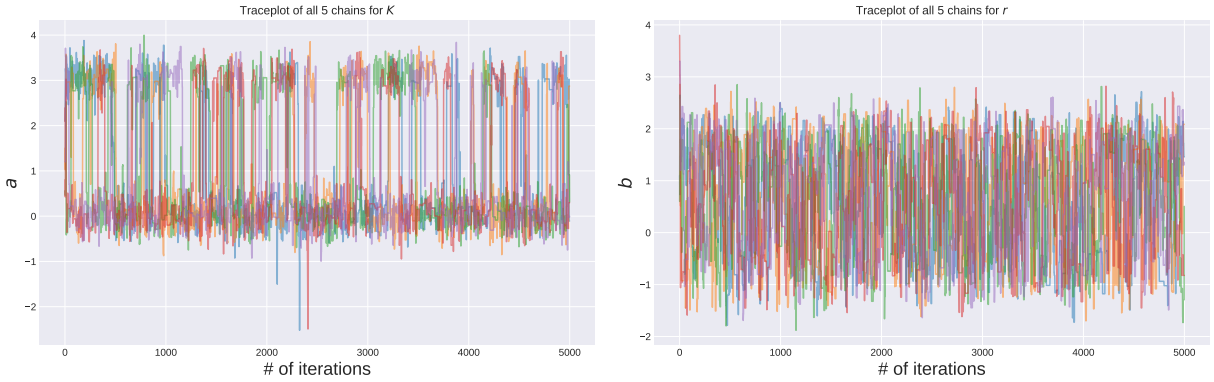
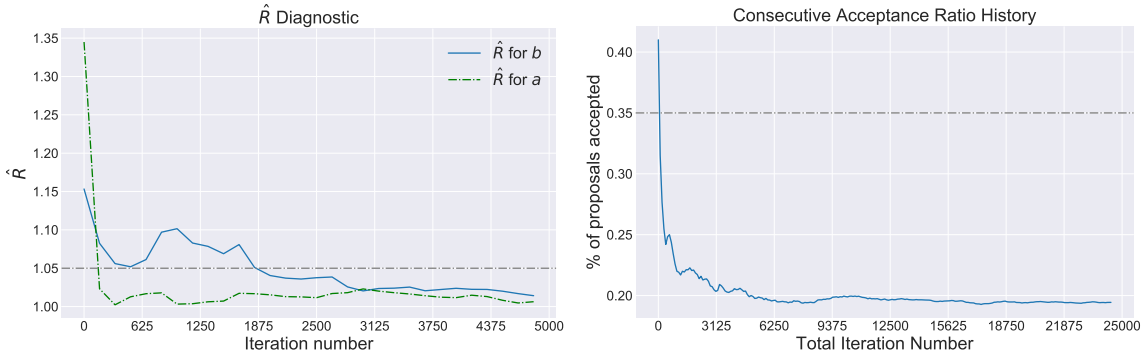


FIGURE 15. Trace plots for parameters a and b respectively from the Metropolis-Hastings algorithm on the inference problem in Section 3.6.1.

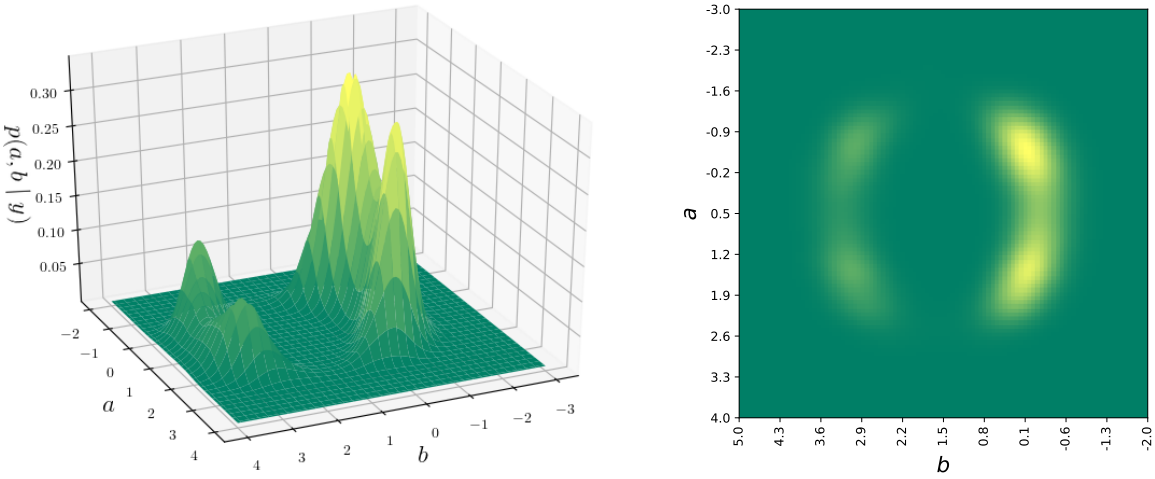
both parameters, displayed in Figure 16a, we see that the first time they both fall under 1.05 is at sample point 1900. It is important to understand that posteriors like the one presented here require a longer run time to obtain a low $\widehat{R}$ value, this is because of the mutli-modality of the surface. Having an $\widehat{R}$ close to 1 implies that all chains have visited the peaks in a similar proportion. This is why we have had to run the chains for significantly longer than in Section 3.5. We see in Figure 16b, where we consider the acceptance ratio of the chains end-to-end, that the ratio falls into the acceptable bounds early in the history of the first chain. After discarding the initial 1899 samples as the burn in period, we again perform Kernel Density Estimation on the remaining samples, to produce a heat and surface plot of the posterior, displayed in Figure 17. In addition, we also produce a heat plot of a finely discretised grid approximation to this posterior, to asses the accuracy of the Metropolis-Hastings, shown in Figure 18. This example displays the influence a prior distribution can have on our inference. By construction, the likelihood function is symmetric in both a and b, yet evidently the posterior displayed in



(A) Acceptance Ratio for Chains

(B) Plot of $\widehat{R}$ Statistic

FIGURE 16. Plots of the $\widehat{R}$ diagnostic and acceptance ratio for 5 chains produced using the Metropolis-Hastings algorithm.



(A) KDE Surface Plot

(B) KDE Heat Plot

FIGURE 17. Surface and Heat Plots of the Density Estimation for the joint posterior of parameters a and b, after performing Metropolis-Hastings.

Figure 17 is not symmetric in b. This asymmetry is due to the pre-experiment beliefs being shifted more towards one side of the likelihood surface.

3.7. **Evaluation of Metropolis-Hastings.** Markov Chain Monte Carlo algorithms are just one class of methods used to perform approximate inference of a posterior distribution. There is no go-to, one-beats-all class of methods because of the varying degree of model complexity and computing power between each user. Because of this competition between techiques, attractive MCMC methods are those which converge to the their stationary distribution, the posterior, quickly and need minimal tuning in order to be effective. The advantage of the Metropolis-Hastings is that it is simple and relatively easy to self-implement through the use of a programming language. In addition, for rather simple low dimensional problems, the method produces reasonable results in a short period of time as demonstrated in the previous examples. As described in [4], the Metropolis-Hastings algorithms is one of the most effective MCMC methods when we consider the Effective Sample Size, another convergence diagnostic, versus CPU time. We will now briefly discuss some of the shortcomings of the M-H algorithm and

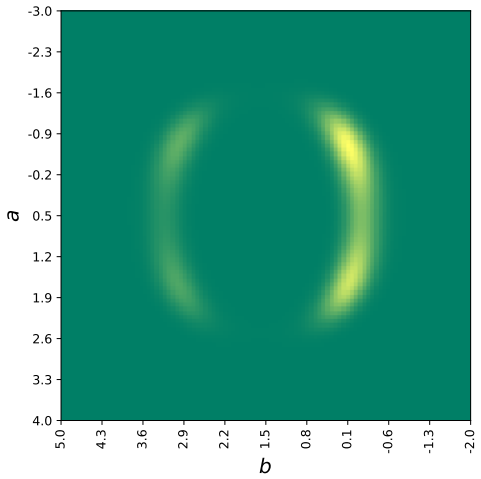


FIGURE 18. Fine grid approximation to the ODE problem in Section 3.6.1

certain probability distributions which it struggles with, before looking at how these issues are addressed in contemporary MCMC methods.

3.7.1. *ODE Parameter Sensitivity.* The Metropolis-Hastings algorithm encounters a problem with the sensitivity of the parameters in a system of differential equations. A parameter is considered to be *sensitive* in an ODE system if, when we change the value of the parameter slightly, the output of the ODE system changes substantially [34]. If a parameter is sensitive we are typically faced with a narrowly peaked posterior, since surrounding values will produce data that is considerably different from that of the true of parameter, thus giving a them a low likelihood. The M-H algorithm encounters difficulty with sensitive parameters due to its random-walk nature. For us to explore a narrowly peaked region with the Metropolis-Hastings method, many attempts at tuning the proposal distribution are needed. An example of this would be considering inference on a parameter which denotes the frequency of a system with cyclical solutions, when observations are made over multiple wavelengths.

$$\begin{cases} \dot{x} = y \\ \dot{y} = -a^2\, x \end{cases}$$

The parameter a represents the frequency of the cyclical solutions. Below, we observe the solution $y(a,t) = A\,\sin(a\,t) + B\,\cos(a\,t)$ with A and B equal to one. If we consider observing the solutions over multiple wavelengths then we see that a is sensitive, since the solutions generated using a and $a+\delta$, where δ is small, will diverge as time increases.

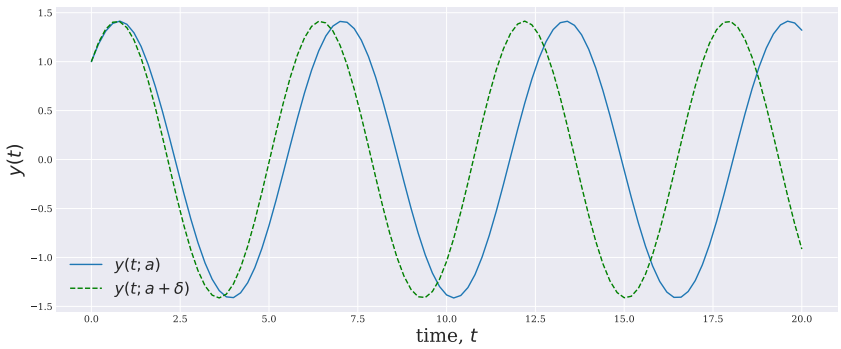


FIGURE 19. Observing how small change in a can cause significant change in output.

As previously explained, this typically causes posterior distributions with narrow peaks of high probability density. Tuning the covariance matrix of a proposal distribution to find and explore this narrow peak can require many attempts when compared with other MCMC methods. Methods like the Hamiltonian Monte Carlo (HMC) [35] calculate the sensitivity of the unnormalised posterior using partial derivatives, to make an informed choice about the proposed steps. When proposing new steps in a chain, the partial derivatives allow the algorithm to consider the local curvature of the surface. By using gradient-based information, methods like HMC typically converge to the target distribution faster.

3.7.2. *Multimodal Distributions.* In addition to the issue described above, the Metropolis-Hastings algorithm can falter when faced with narrow yet sparsely-distributed multi-modal distributions. The constant nature of proposal distribution's covariance matrix means it can be hard for the chain to explore both peaks with the correct proportion in finite time. We will explore this problem through a simple, one dimensional toy example. We consider a posterior which has a probability density function, displayed in Figure 20. This posterior, and the following results, can be found in the notebook titled `problems_with_mh_mcmc.ipynb` of [5].

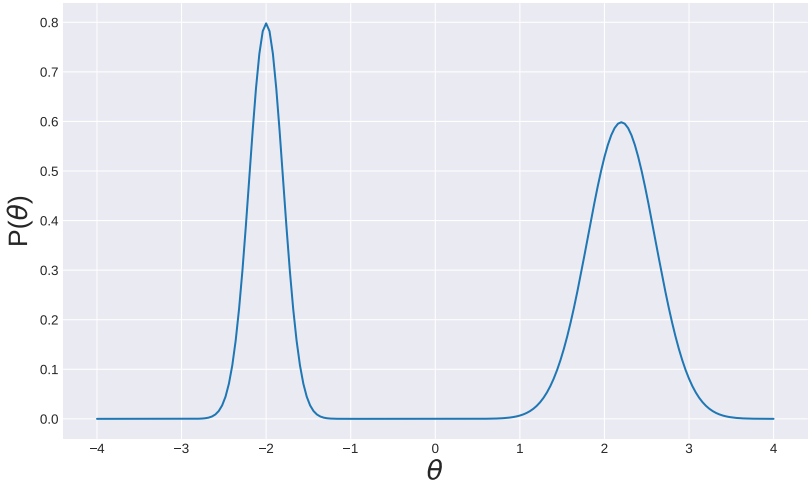
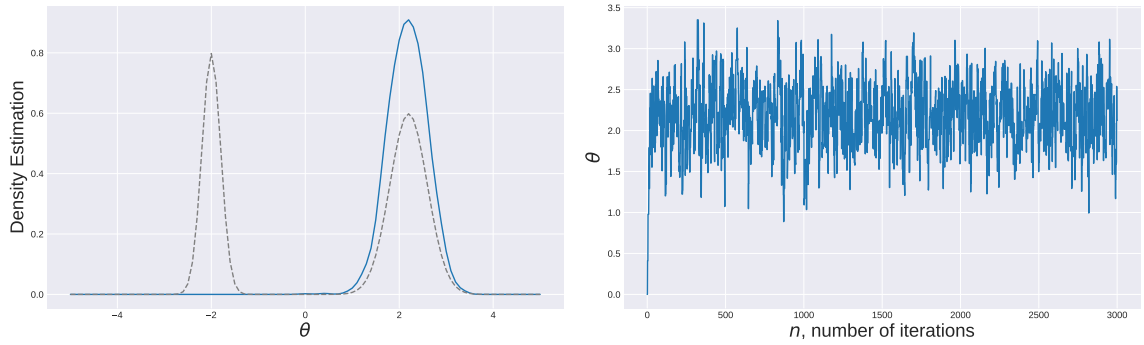


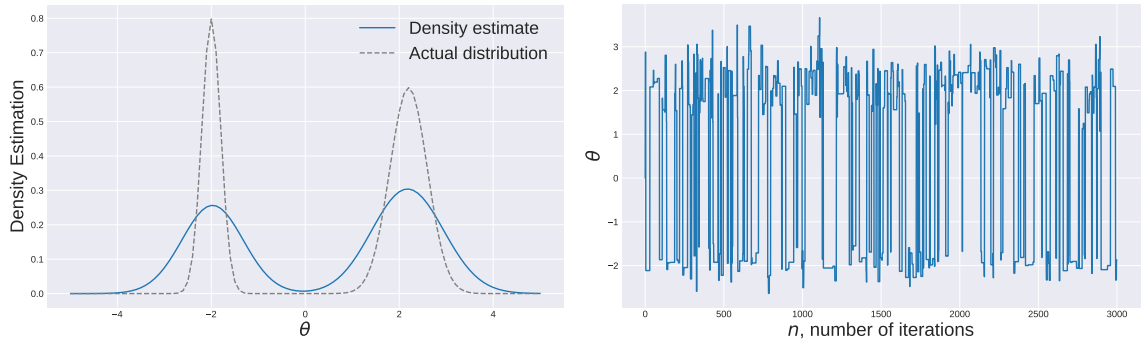
FIGURE 20. An arbitrary bi-modal posterior used to explain issues arising from the Metropolis-Hasting algorithm.

As described in Section 3.3.3, we only alter the covariance matrix of the proposal distribution during the burn in and when the current acceptance ratio falls out of the pre-defined bounds. Metropolis-Hastings algorithms that use this type of covariance matrix estimation struggle with multi-modal distributions as we will now show. For example, suppose we have a proposal distribution q with a small variance, $\sigma = 0.5$ and we run the M-H algorithm on the posterior example defined above. After a long set of iterations the chain will meander from the initial state and enter one of the peaks. Because q has a small variance, we will be able to explore that particular peak well. For the chain to enter the other peak, we will require many more iterations since it is unlikely that the proposal distribution will suggest a point so far away. The number of iterations for this to happen can often be beyond any reasonable waiting time. Figure 21 shows the traceplot of a single chain exploring the above posterior, we see that the chain does not enter the second peak when run for 1000 iterations. On the other hand, if we define the proposal distribution q to have a large variance, $\sigma = 5$, we will visit both peaks regularly, however it will take a long time for us to explore the local details of each peak. This is due to a lower probability of a proposed point being close to the current point. We also run the M-H algorithm using a proposal distribution with a large variance and observe the density estimation generated from the samples, plotted in Figure 22.



(A) Plot of density estimation from samples (B) Trace plot

FIGURE 21. Plots generated from running the M-H on this toy example with a small variance proposal distribution.



(A) Plot of density estimation from samples (B) Trace plot

FIGURE 22. Plots generated from running the M-H on this toy example with a large variance proposal distribution.

This problem can be again addressed with the Hamiltonian Monte Carlo algorithm. The method of HMC is similar to that of the Metropolis-Hastings, in that we compare the proposed step to the current state. However, in addition to using gradient information, the HMC algorithm occasionally introduces a random kick to the next proposed step. This random kick, although infrequent, means we suggest a point in the parameter space that is far away from the current point. By spontaneously adding a kick to the proposal step, we can find and explore distant peaks in a much shorter time when compared to the standard Metropolis-Hastings. This flexibility of the Hamiltonian Monte Carlo algorithm has led it to become the base algorithm for popular probabilistic programming language `Stan` [36].

4. Discussion and Further Work

Bayesian inference has become a popular approach to statistical modelling because of the ability to portray one's pre-experiment beliefs and the flexibility to describe how the data was produced. For complex models to be used, we must find a computationally efficient and accurate method that allows us to bypass or approximate an intractable marginal-likelihood. In addition to Markov-Chain Monte Carlo, there are several other notable approaches to the problem of approximate inference which may appeal to the reader. In particular, Variational Inference (VI) is a popular alternative to MCMC that uses Laplace's Approximation to the posterior. This method is typically favoured when computational run time is an important factor to consider. In fact, many Variational approaches produce accurate results while an MCMC algorithm is still within its burn in period. Although Variational methods can be applied to a broad class of

models, it performs poorly for problems with highly correlated parameters. For a detailed survey on Variational Inference and its uses, the reader is directed to [37]. In addition to VI, topics such as Slice Sampling [38] and Tempered Annealing [3] are also current branches of research, which have uses in approximate inference. The flexibility of Markov Chain Monte Carlo allows us to perform inference and parameter estimation for intricate biological and mechanical systems. For a reader interested in the applications of MCMC, they are directed towards the CSIDE (Competitive Statistical Inference for Ordinary Differential Equations) webpage [2]. CSIDE, as mentioned in the introduction, is an annual conference hosted to asses what "inference methods are currently state of the art" [2]. In particular, the competition explored in-depth parameter estimation in various settings of systems biology, from cardiac excitation to cell migration. In addition to the work of CSIDE, [39] gives a well-written account on the practical implementation of the Metropolis-Hastings algorithm in geography and [40] shows how the Hamiltonian Monte Carlo can be used in ODEs relating to neurology. To develop the ideas presented in this report further, comparing the performance of an MCMC method using gradient-based information to that of the Metropolis-Hastings would be an obvious next step. In addition, exploring the use of MCMC algorithms on systems that naturally produce complex posterior distributions would provide more insight into how these methods of approximate inference can be used in real biological problems. Rather than using artificially generated complex distributions, like in Section 3.6.1, we could consider systems that produce limit-cycle behaviour for specific values of parameters, such as the circadian oscillator ODE model proposed by Goodwin [41]. It has been shown in [3] that for specific combinations of true parameters, models of this form produce complex log-likelihood surfaces. An alternative suggestion for further work could be to explore posterior inference on a system when we have less information available. In particular, we consider the case where the initial conditions, and error variance are unknown; doing so would simulate a more realistic experimental environment. We would then consider these as two extra parameters in the inference model, after assigning relevant priors. Such models could be used to identify plausible values of initial conditions in ecological systems, as described in [42].

5. Summary

This report has focused on investigating computational methods of approximate inference for ODE based models. We explored the merits and pitfalls of both the grid approximation, a brute-force method for low-dimensional models, and the Metropolis-Hastings algorithm, a member of the Markov Chain Monte Carlo family. In our discussion of the grid approximation, we gave an original description for its implementation to ODE based models. For the Metropolis-Hastings algorithm, we built upon several pre-exisiting structures from computational systems biology and examined the convergence of the MCMC method to simulated data. We applied both methods to inference problems arising from two low dimensional ODE models, namely the Logistic Equation and an artificial, two-dimensional predatory-prey model. In the final discussion, we emphasised the need for accurate and efficient methods of approximate inference. This is particularly present in the field of systems biology and ecological modelling, where good parameter estimates can aide in our understanding of many biological interactions, from a cellular-level understanding of enzyme reactions to the predation and competition rates of large ecosystems.

Acknowledgements

Special thanks are due to Benn MacDonald and the Competitive Statistical Inference for Differential Equations 2018 conference, for them kindly organising and funding my trip to the event as well as exposing me to the current advances of approximate inference in computational systems biology. The Competitive Statistical Inference for Differential Equations event was funded by The Biometrika Trust, Fellowship No. B0003, and sponsored by the EPSRC Centre for Multiscale soft tissue mechanics with application to heart and cancer, Reference No. EP/NO14642/1.

References

[1] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[2] Benn MacDonald. Competitive statistical inference for differential equations 2018. `https://www.gla.ac.uk/schools/mathematicsstatistics/events/conferences/cside2018/`, 2018. [Online; accessed 13-April-2019].

[3] Ben Calderhead. A Study of Population MCMC for estimating Bayes Factors over Nonlinear ODE Models. Master's thesis, University of Glasgow, Scotland, 2007.

[4] Irena Kuzmanovska. Markov Chain Monte Carlo Methods in Biological Mechanistic Models. Master's thesis, ETH Zurich, 2012.

[5] Thomas Armstrong. GitHub: Bayesian Statistical Inference for Ordinary Differential Equations. `https://github.com/thomasarmstrong98/thirdyearproj`, 2019. [Online; accessed 28-April-2019].

[6] John Kruschke. *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press, 2014.

[7] Peter D Hoff. *A First Course in Bayesian Statistical Methods*, pages 4–40. Springer, 2009.

[8] Anthony W F Edwards. *Likelihood*. CUP Archive, 1984.

[9] James M. Curran William M. Bolstan. *Introduction to Bayesian Statistics*, chapter 8, pages 162–164. Wiley, 3rd edition, 2016.

[10] Girolami M. Rattray M. Lawrence, N. D. and G. Sanguinetti. *Learning and Inference in Computational Systems Biology*. MIT Press, 2010.

[11] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 13-April-2019].

[12] J.R. Dormand and P.J. Prince. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19 – 26, 1980.

[13] Oksana A Chkrebtii, David A Campbell, Ben Calderhead, Mark A Girolami, et al. Bayesian solution uncertainty quantification for differential equations. *Bayesian Analysis*, 11(4):1239–1267, 2016.

[14] Steven Strogatz, Mark Friedman, A John Mallinckrodt, and Susan McKay. Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering. *Computers in Physics*, 8(5):532–532, 1994.

[15] Joanna Dunkley and Rupert Allison. Comparison of sampling techniques for Bayesian parameter estimation. *Monthly Notices of the Royal Astronomical Society*, 437(4):3918–3928, 2013.

[16] David J C MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[17] Gareth O Roberts, Jeffrey S Rosenthal, et al. General state space Markov chains and MCMC algorithms. *Probability Surveys*, 1:20–71, 2004.

[18] Nicolas Privault. *Understanding Markov Chains: examples and Applications*. Springer Singapore, 2013.

[19] Luke Tierney. Markov Chains for Exploring Posterior Distributions. *The Annals of Statistics*, 22(4):1701–1728, 1994.

[20] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 04 1970.

[21] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. In *Readings in Computer Vision*, pages 564–584. Elsevier, 1987.

[22] David J. Lunn, Andrew Thomas, Nicky Best, and David Spiegelhalter. WinBUGS - A Bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing*, 10(4):325–337, Oct 2000.

[23] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2:e55, 2016.

[24] Siddhartha Chib and Edward Greenberg. Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.

[25] Gareth O Roberts and Richard L Tweedie. Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms. *Biometrika*, 83(1):95–110, 03 1996.

[26] Andrew Gelman, John B. Carlin, et al. *Bayesian Data Analysis*. Chapman & Hall/CRC Texts in Statistical Science, 3 edition, 2004.

[27] Gareth O Roberts, Jeffrey S Rosenthal, et al. Optimal scaling for various Metropolis-Hastings algorithms. *Statistical science*, 16(4):351–367, 2001.

[28] John Geweke et al. *Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments*, volume 196. Federal Reserve Bank of Minneapolis, Research Department Minneapolis, MN, 1991.

[29] Andrew Gelman, Donald B Rubin, et al. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–472, 1992.

[30] Adrian E Raftery and Steven M Lewis. Practical Markov Chain Monte Carlo: Implementation strategies for Markov Chain Monte Carlo. *Statistical Science*, 7(4):493–497, 1992.

[31] Aki Vehtari, Andrew Gelman, et al. Rank-normalization, folding, and localization: An improved $\widehat{R}$ for assessing convergence of MCMC. *arXiv preprint arXiv:1903.08008*, 2019.

[32] Dootika Vats and Christina Knudson. Revisiting the Gelman-Rubin Diagnostic. *arXiv preprint arXiv:1812.09384*, 2018.

[33] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[34] Robert P. Dickinson and Robert J. Gelinas. Sensitivity analysis of ordinary differential equation systems — a direct method. *Journal of Computational Physics*, 21(2):123 – 143, 1976.

[35] Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

[36] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76(1), 2017.

[37] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[38] Radford M Neal et al. Slice sampling. *The annals of statistics*, 31(3):705–767, 2003.

[39] Bryson C Bates and Edward P Campbell. A markov chain monte carlo scheme for parameter estimation and inference in conceptual rainfall-runoff modeling. *Water Resources Research*, 37(4):937–947, 2001.

[40] Biswa Sengupta, Karl J Friston, and Will D Pennny. Gradient-based MCMC samplers for dynamic causal modelling. *NeuroImage*, 125:1107 – 1118, 2016.

[41] Brian C Goodwin. Oscillatory behavior in enzymatic control processes. *Advances in Enzyme Regulation*, 3:425–437, 1965.

[42] S Schliehe-Diecks, PM Kappeler, and R Langrock. On the application of mixed hidden Markov models to multiple behavioural time series. *Interface focus*, 2(2):180–189, 2012.