STATISTICAL INFERENCE FOR ORDINARY DIFFERENTIAL EQUATIONS

THOMAS ARMSTRONG

Abstract. The use of ordinary differential equations (ODEs) in applied mathematics and systems biology is crucial in describing how the position, density, or population of certain objects change in time. These ODEs often involve parameters which, in part, characterise how an object may develop based on interactions with other objects in the system or with itself. The act of inferring the values of parameters that govern these ODEs can be a difficult task when faced with experimental data, obscured by noise. This survey looks at how we approach the task of approximate inference of a posterior distribution when given simulated experimental data. In particular, we use the grid approximation and the Metropolis-Hastings algorithm in the estimation of low-dimensional posterior distributions. Such posterior distributions arise from two small ODE models, one being the Logistic, or Verhulst-Pearl, equation and the other being a two-dimensional, artificially created, competition model. In each case of posterior approximation, we will also explore how convergence of Markov Chain Monte Carlo techniques can be assessed using the Gelman-Rubin $\hat{R}$ diagnostic. We conclude by discussing certain problems faced by the grid approximation and the Metropolis-Hastings algorithm as well as suggestions for further work.

Contents

1. Introduction

Probabilistic inference, also referred to solely as inference, is the task of assigning probabilities to a random variable taking a value within a certain range. We are interested in being able to perform inference so that we can consider which regions a random variable is likely to take values in. In this survey, we will be utilising a Bayesian approach to our inference, this is a common approach when considering parameter estimation of Ordinary Differential Equation (ODE) models [1], [2]. The use of mathematical models, such as ODEs, has allowed us to understand the movement, birth or growth of many observable systems in nature. Typically, we will be considering ODE models with non-conguate, prior-likelihood pairs. Such models mean that we are unable to perform exact inference and, instead, must resort to analytical or computational approximations, known as *approximate inference*. Our task is to construct methods of parameter estimation having observed one of these systems; to do so, we will be considering the grid-approximation, a brute force method of approximation, and the Metropolis-Hastings algorithm, a member of Markov Chain Monte Carlo methods. The motivation for having accurate and computationally efficient methods for parameter estimation in ODE models can be determined by the practical implementations of such methods. 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 [3]. In particular, it discussed various aspects of systems biology, in which this branch of research could be used.

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 parameters $\boldsymbol{\theta}$ which we do not know the value of.

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

Here $\mathbf{x} \in \mathbb{R}^n$ is the vector of variables in our system and $\mathbf{f} = (f_1, f_2, \ldots, f_n)$ is a vector of functions describing the derivatives of the system. We measure m observations of the system, denoted $\mathbf{y} = (\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_m})$ at times $t_1, t_2, ..., t_m$ respectively. The aim of this report, as mentioned, is to find how experimental data can update our beliefs about the values of the parameters in our ODEs. This inductive approach makes it natural for us to consider a Bayesian framework to work in. We will be heavily reliant on this Bayesian framework, so a description will be provided in the next 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 be believe that this coin is fair? By using observations, we can make inferences to update our past beliefs. To quantify new information, we can use Bayes Rule (1), this formula allows us to update our beliefs about the probability on an event, given that another event has occurred. If we consider a probability space Ω, which denotes the set of all possible events, that can be partitioned into a collection of sets $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$, then for an event E, we can write Bayes Rule as:

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

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 [4].

As well as 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 with a particular probability. In the Bayesian framework, we can treat fixed but unknown population

parameters as random variables [5]. A more formal definition of a random variable along with what it means to be continuous or discrete can be found here [6]. If we consider two continuous random variables X and Y, then Bayes Rule can be re-written as:

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

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 (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.* The Bayesian framework makes use of Bayes Rule to perform statistical 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 then we can use Bayes Rule to obtain a posterior density $p(\boldsymbol{\theta} \mid \boldsymbol{y})$.

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

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 [7]. Our prior distribution $\pi(\boldsymbol{\theta})$ characterises our belief in the values of $\boldsymbol{\theta}$ before observing any data. The choice of $\pi(\boldsymbol{\theta})$ is subjective and therefore allows us to accurately 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 (3). Such cases require prior-conjugate pairs, hence restricting the flexibility with which we can characterise our current beliefs in $\boldsymbol{\theta}$. In this report, we will work as though the posterior is not available in a closed form, this however may not be the case in some examples. This assumption puts us on a path towards methods in *approximate inference*, that is, we do not have a normalised posterior for which we can evaluate.

Although we are using Bayes Rule to update our beliefs about the values of a parameter, this does not make our approach inherently Bayesian. 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 be 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 a particular probability distribution. This framework is particularly attractive for us to work with because of the nature of the resulting posterior. Unlike frequentist approaches like Maximum Likelihood Estimation, our posterior gives a full distribution for our 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 at *credible intervals*.

1.2.3. *Bayesian Credible Intervals.* Typically, after approximating our posterior, we wish to define some region in which 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 of a model parameter lies within the region is a certain percentage. 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 be using 95% credible regions to give bounds to our estimates for our 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 [4].

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 [5]. 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 our parameter estimation, we require the following components:

- An ODE system describing the experiment, with initial conditions.
- Noise model for our 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 experimental errors that may occur when observing a system. From this noise model, we construct a likelihood function. If we have an observation $\boldsymbol{y_i}$, then the probability that this data was produced by a particular set of parameters $\boldsymbol{\theta}$ is our 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 our ODE system at time t_i, dependent on the particular set of parameters $\boldsymbol{\theta}$. We assume that all observations are conditionally independent given $\boldsymbol{\theta}$. By this construction, we can use the inverse of de Finetti's Theorem [6] to treat our data as *exchangeable*, that is, the order in which we consider the observations does not matter.

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

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 our calculations by replacing the product of small floating point numbers with a sum, which can later be exponentiated. Our log likelihood for the entire set of observations is therefore

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

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

1.3.2. *Data Simulation.* In our assumptions, we assume that we know exactly how the errors in our measurements are distributed. Because of this approach, 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 generate data so that we can perform parameter estimation, we first define a set of points time points $t = (t_1, t_2, \ldots, t_m)$ which represent the times at which we make measurements of our system. Within this report, our time points are linearly spaced, however this is not a requirement to perform inference. We set the "true" value of our parameters to be $\boldsymbol{\theta^*}$ and then compute the solution of the system with these parameters at each time point. 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 and $\boldsymbol{\theta^*}$ into the solution if it available in closed form or numerically approximate the solution. To numerically approximate the solution, we will be use the `integrate` module from the SciPy library [8]. 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 here [9]. We then add independent Gaussian noise to the solutions at each time point to obtain our observed data. This noise has a mean of zero and a pre-specified variance or co-variance matrix, matching our noise model.

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

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, its accuracy is proportional to the computing time of the algorithm. The grid approximation is a method that approximates the marginal-likelihood (4) of the posterior to obtain the normalisation constant [4]. This approximation comes in the form of using the Riemann sum to estimate the integral (4). In addition to the grid approximation, Monte Carlo techniques such as importance sampling also look to approximate this integral, discussed further in 3.1.1.

2.1. **Defining a grid.** First, we must discretise our parameter space for $\boldsymbol{\theta}$ in our region of interest. The region $\boldsymbol{\Theta}$ represents the subset of $\mathbb{R}^k$ in which all of our 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.

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

To approximate 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 discrete points we are dividing our grid into. That is

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

where w_i^0 and $w_i^{n_i-1}$ denote the lower and upper bound respectively for the grid of parameter θ_i which we must also specify. If we consider the toy example, displayed in Figure 1 below, where we have plotted the posterior density for a parameter θ_i which is normally distributed. The aim of defining w_i^0 and $w_i^{n_i-1}$ is to allow us to obtain an accurate view of the posterior. If we chose w_i^0 and $w_i^{n_i-1}$ too narrow, we will miss areas of non-negligible density in our approximation, whilst if we define them too wide then we can increase our computing time drastically. Figure 1 suggests some suitable bounds that could be used to approximate the area under the posterior density.

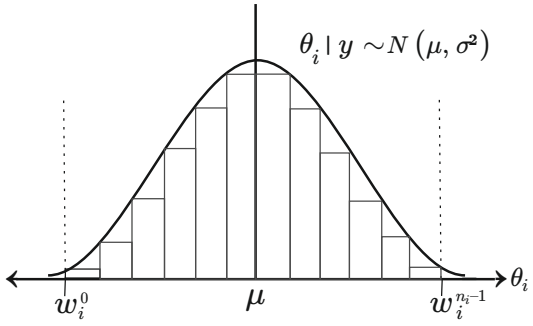


FIGURE 1. Toy example displaying suggested upper and lower bounds for grid approximation.

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.

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

If we consider another example in which $k = 2$ and we have defined both $\boldsymbol{w_1}$ and $\boldsymbol{w_2}$, then $\boldsymbol{W}$ is the set of all combination of points, displayed as a 2-dimensional grid below.

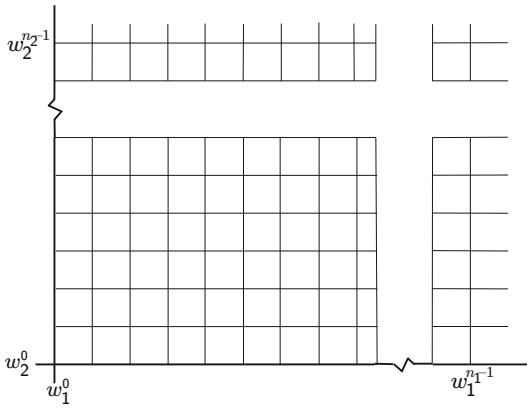


FIGURE 2. A two-dimensional hypergrid $\boldsymbol{W}$ defined by Cartesian product of grids for arbitrary parameters θ_1 and θ_2.

The evidence term, as denoted by the integral over the region $\boldsymbol{\Theta}$, signifies the area under our unnormalised posterior density. We approximate this integral with a multi-dimensional Riemmann sum over our hyper-grid $\boldsymbol{W}$.

$$\begin{aligned}\int_{\boldsymbol{\Theta}} p(\boldsymbol{y} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta} &= \int_{\Theta_1} \cdots \int_{\Theta_k} \pi(\theta_1, \ldots, \theta_k)\, p(\boldsymbol{y} \mid \theta_1, \ldots, \theta_k)\, d\theta_1 \cdots d\theta_k \\ &\approx \sum_{w_1=w_1^0}^{w_1^{n_1-1}} \cdots \sum_{w_k=w_k^0}^{w_k^{n_k-1}} \pi(w_1, \ldots, w_k)\, p(\boldsymbol{y} \mid w_1, \ldots, w_k)\, \Delta w_1 \cdots \Delta w_k\end{aligned}$$

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

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

Defining this region 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 our 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 this point adds nothing to the value of our summation. There is a constant trade-off between setting the bounds wide enough to capture the parameter space whilst ensuring the grid is computationally efficient [10].

2.2. **Implementation.** Once we have defined the the grid $\boldsymbol{w_i}$ for each parameter θ_i we can calculate the hyper-grid $\boldsymbol{W}$ 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 $a \in [0, n_1 - 1]$, $b \in [0, n_2 - 1], \ldots, p \in [0, n_k - 1]$. We then calculate the unnormalised posterior density at each point of the hyper-grid $\boldsymbol{W}$.

$$p(\boldsymbol{W}_{a,b,\ldots,p} \mid \boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_m}) \propto \prod_{l=1}^{m} p(\boldsymbol{y}_l \mid \boldsymbol{W}_{a,b,\ldots,p}) \pi(\boldsymbol{W}_{a,jb,\ldots,p})$$

We now have a k-dimensional matrix of unnormalised posterior densities. To normalise them, we calculate the Riemann sum described in the previous sub-section and divide every entry of $\boldsymbol{W}$ by this value [4].

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 [11] (also known as the Verhust-Pearl equation), that is:

$$\frac{dN}{dt} = r\,N\left(1 - \frac{N}{K}\right) \tag{5}$$

where N(t) denotes the population or density of a population, depending on the use of the model. This system contains two parameters $(\theta_1, \theta_2) = (r, K)$, which we will be performing inference on. 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:

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

For this particular problem, we will consider the initial condition for our system $N(0)$ to be known, allowing us to obtain a unique solution for the system at our time points, with particular values of K and r. We will simulate data from this system using the method described in 1.3.2. For this particular example, we set our 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 our experimental data which is displayed in Figure 3.

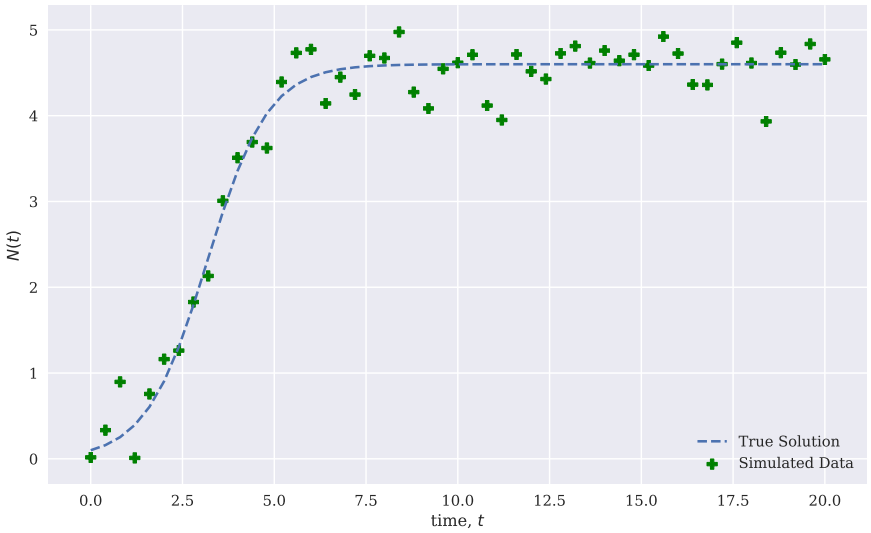
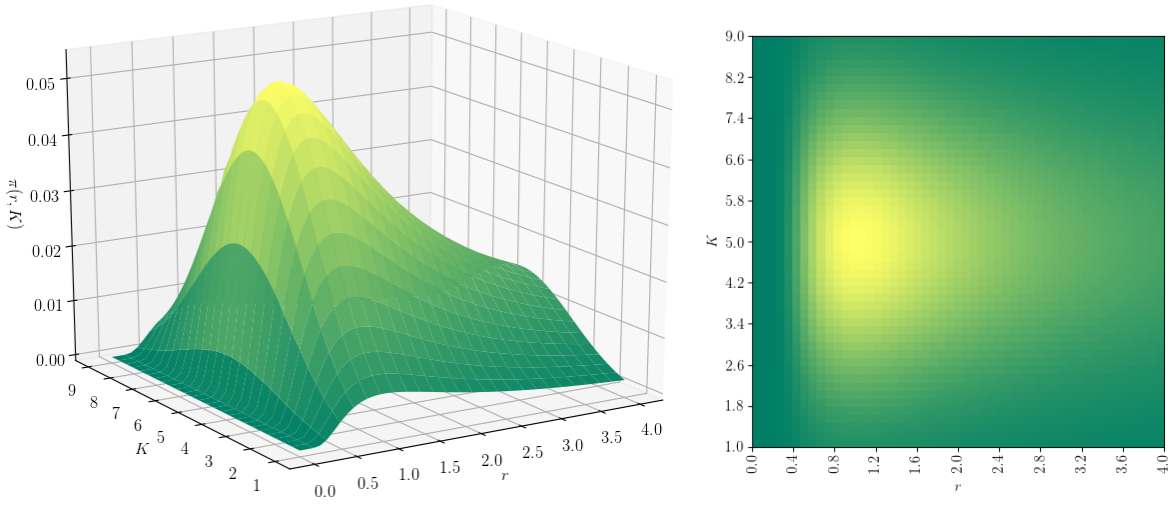


FIGURE 3. Our simulated data from the ODE specified above with $r = 1.2$, $K = 4.6$ and $N_0 = 0.1$.

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, and this distribution captures this fact. 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 our model parameters r and K.

$$r \sim \text{Inv-Gamma}(a, b), \qquad K \sim \text{N}(5, 2)$$



(A) Surface plot (B) Heat plot

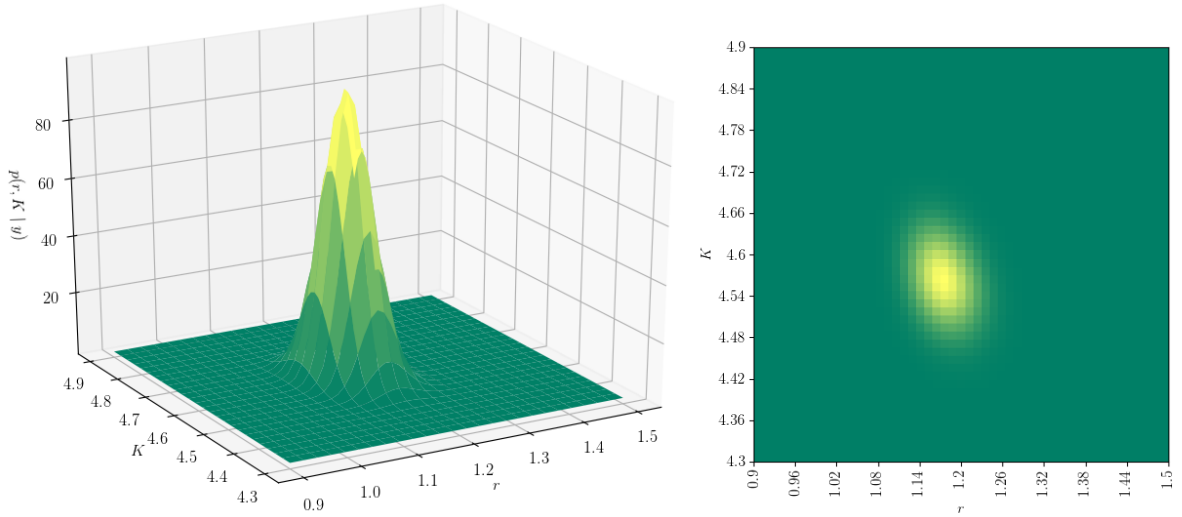
FIGURE 4. Plots of the joint prior $\pi(r, K)$

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 $n_1 = n_2 = 101$ and our bounds as displayed below:

$$\theta_1 = r: \quad w_1^0 = 0.9, \quad w_1^{50} = 1.5 \implies \Delta w_1 \approx 0.006$$
$$\theta_2 = K: \quad w_2^0 = 4.3, \quad w_2^{50} = 4.9 \implies \Delta w_2 \approx 0.006$$

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 can be found at my GitHub. Now that we have approximated our 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 evidence term using the Riemann sum over our approximated parameter space, multiplied by our step-sizes Δw_1 and Δw_2. Finally, we divide our un-normalised posterior matrix by this approximation to obtain our posterior hyper-grid which can then be plot as shown in Figure 5.



(A) Surface plot (B) Heat plot

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

2.3.4. *Evaluation.* By comparing the axes of our heat plots for both the prior and the posterior, we observe that our uncertainty about $\boldsymbol{\theta}$ has become more localised. To use the results of our grid approximation, we can consider the 95% credible intervals for each parameter, as discussed in 1.2.3. To obtain the posterior of one parameter only, we must marginalise the posterior with respect to the remaining parameters. We again approximate the marginalisation integral by a Reimann sum.

$$\begin{aligned} p(\theta_1 \mid \boldsymbol{y}) &= \int_{\boldsymbol{\Theta}_2} p(\theta_1, \theta_2 \mid \boldsymbol{y})\, d\theta_2 \\ &\approx \sum_{w_2 = w_2^0}^{w_2^{n_2-1}} p(w_1, w_2 \mid \boldsymbol{y}) \Delta w_2 \end{aligned}$$

Now that we have obtained the posterior distribution for a single parameter, we can compute the respective 95% credible region (CI). This statistic gives a region in which there is a 95% probability that our model parameter is contained within our interval. By giving a CI, we can quanitfy a region, smaller than our graphical axis above, in which it is likely that our parameter is contained. For this example of the Logistic Equation, we obtain our 95% credible region for both parameters, displayed in Figure 6 and Table 1.

	r	K
95% CI	[1.146,1.278]	[4.534, 4.720]

TABLE 1. Displays our Credible Interval bounds for model parameters r and K.

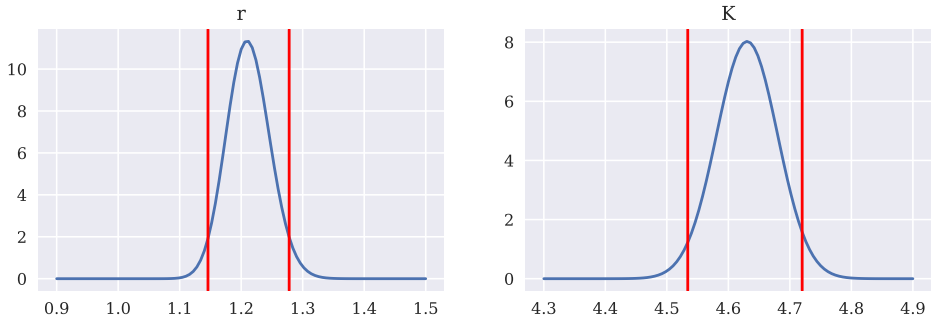


FIGURE 6. The marginalised probability density functions for r and K respectively, with the bounds of their 95% CI overlayed.

Using the bounds of the 95% CI for both parameters, we can look at the range of solutions that arise from the values of parameters within this interval. 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 the 95% CI respectively. By computing and plotting the solution of our ODE system for both pairs of parameters we can see the range in which the solutions of our 95% CI would lie, displayed 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.

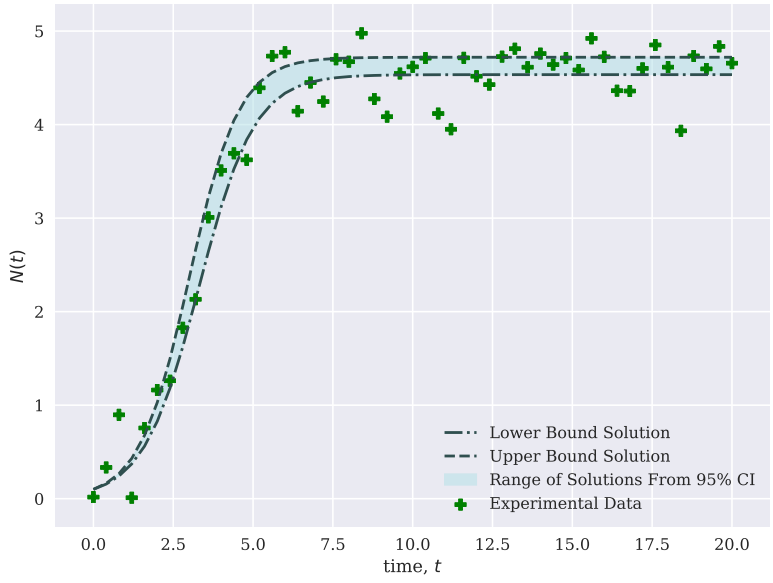


FIGURE 7. Range of solutions produced by the maximum and minimum bounds of the confidence interval.

2.4. **Evaluation of the Grid Approach.** The Grid approximation allows us to compute an approximation to the posterior by subsetting our parameter space and discretising the marginal likelihood. 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 our 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 $340,000,000^2 \approx 1.2e+17$ 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 reduce the total number of grid points considered, by making Δw_i larger, but this will cause our Riemann sum approximation to the integral 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 our parameter space $\boldsymbol{\Theta}$ is a relatively simple task. In low dimensions, we 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. When considering a low-dimensional posterior, however, the grid approximation has an advantage over techniques like Markov Chain Monte Carlo being that the complexity or multi-modality of the posterior is not more computationally demanding than a simple uni-modal distribution. This brute-force method of approximate inference can yield accurate results for complex distributions in low dimensions [12], provided the grids are appropriately tuned to approximate the parameter space.

3. Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is a family of algorithms designed to produce samples from a particular probability distribution, known as the target distribution, without directly sampling from the distribution itself. 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 field of research in the School of Information Theory and Statistics, with its history dating back to Nicolas Metropolis in 1953 [13]. The applications of MCMC can be found in statistical physics and chemistry, however the most common application is in the Bayesian approach to probabilistic inference.

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 looks like:

$$\mathbb{E}_P\left[h(\boldsymbol{\theta})\right] = \frac{\int_\chi h(\boldsymbol{\theta})\, P(\boldsymbol{\theta})\, d\boldsymbol{\theta}}{\int_\chi P(\boldsymbol{\theta})\, d\boldsymbol{\theta}} \tag{6}$$

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 required expectation (6) [14].

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

3.1.1. *Application.* In our setting, 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 is of the form (6).

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

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 [15].

$$\mathbb{E}_\pi[p(\boldsymbol{y} \mid \boldsymbol{\theta})] \approx \frac{1}{n}\sum_{i=1}^{n} p(\boldsymbol{y} \mid \boldsymbol{\theta}^{(i)}) \tag{7}$$

In practise, 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 [5]. There are other sampling techniques for which we can estimate the marginal-likelihood using independent samples, namely importance and rejection sampling [14], however they will not be discussed here. For our set of samples to satisfy (7), they must be independent. Typically, MCMC methods produce *dependent* samples from their target distribution but, when the algorithms satisfy certain criterion, 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 here [16]. 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 [14].

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

Definition 1. A *Markov Chain Monte Carlo* method for the simulation of a distribution P is any method which constructs an *ergodic* Markov chain $(X^{(t)})$ such that P is an invariant, or stationary, distribution of the chain [15].

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 our 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* (8) with respect to a distribution P, then the chain is *reversible* and P is an invariant distribution of the chain.

$$T(\boldsymbol{x}, \boldsymbol{y})P(\boldsymbol{y}) = T(\boldsymbol{y}, \boldsymbol{x})P(\boldsymbol{x}) \qquad \text{for every } (\boldsymbol{x}, \boldsymbol{y}) \tag{8}$$

We can infer from the above equation that it must be just as likely that we pick a state $\boldsymbol{x}$ from our target distribution P, then move from the state $\boldsymbol{x}$ to $\boldsymbol{y}$ under the transition kernel T, as it so to pick $\boldsymbol{y}$ from P then move from $\boldsymbol{y}$ to $\boldsymbol{x}$ under T [14].

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 for every state i, it is possible to visit every other state j in finite time.

$$\forall\, i,\ j,\ \exists\, m \text{ such that } \mathbb{P}(X^{(n+m)} = j \mid X^{(n)} = i) > 0$$

Because of the continuous nature of our Markov chains, this definition has been adapted slightly to be more applicable, the reader is directed to [17] for further details.

Lemma 1. *If a Markov chain is* aperiodic *and* irreducible *(or Harris recurrent with respect to a particular measure* [17]*), and has a stationary distribution P, then a "Strong Law of Large Numbers" holds* [15]*:*

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} h(X^{(i)}) = \int_{\chi} h(\boldsymbol{\theta})P(\boldsymbol{\theta})\, d\boldsymbol{\theta} = \mathbb{E}_P[h(\boldsymbol{\theta})] \tag{9}$$

What is means to be Harris-recurrent along with a proof of the above statement, can be found in this survey [17].

The result displayed in Definition 1 shows that when our sample size $n \to \infty$, estimators using samples from the Markov chain converge to that particular expectation. Given sufficient computing time, MCMC methods that satisfy the required properties in Definition 1 are guaranteed to provide samples from our target distribution, and therefore can be used to approximate the marginal-likelihood. In addition to approximating the marginal-likelihood, the samples can be used to provide credible intervals. 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.

For an MCMC algorithm to produce 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 our samples are prior to the chain converging to our target distribution. Only after the chain has reached the target distribution will we be able to approximate (6). If we consider a chain of length n where b is the point in which our chain has reached the stationary distribution, then we can construct a practical version of (7).

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

3.3. **Metropolis-Hastings.** The Metropolis-Hastings (M-H) algorithm is a Markov Chain Monte Carlo method that allows us to produce samples of a multi-dimensional probability distribution without having to sample from the distribution directly. 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 [13] and a statistician W.K Hastings who later generalised the method [18]. The Metropolis-Hastings algorithm, along with the Gibbs Sampler [19] (which will not be discussed) are two of the most popular MCMC methods due to their implementation in computer programmes like `WinBUGS` [20] or coding packages such as `PyMC3` [21].

3.3.1. *The Algorithm.* We first initialise the our 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 then decides how we move from our starting initial value to the next state, which 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 our transition kernel T satisfies the *detailed balance equations* (8), then P is a stationary distribution of the chain produced. The transition kernel for the M-H algorithm can be written:

$$T(x, y) = q(y \mid x)\, \alpha(x, y) + \delta_x(y)(1 - r(x)) \tag{10}$$

where $r(x) = \int_\chi \alpha(x, y)\, q(y \mid x)\, dy$ and $\delta_x(y)$ is the Dirac mass in x [22]. We arrive at this form by considering the cases in which $y \neq x$ and $y = x$ seperately. When $y = x$, we 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 have when the proposed new step y is the same as the current state x, and therefore automatically accepted, this occurs with probability $q(y \mid x)$. Secondly, we consider when we 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 (10). A full proof and explanation of how we arrive at this transition kernel can be found here [22]. Now that we have a transition kernel for our 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.

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

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 the satisfying (8), the M-H method must also create an *ergodic* Markov Chain (that is a chain that is both aperiodic and irreducible). It can be shown that the Metropolis-Hastings algorithm satisfies these properties and so the samples produced will obey the Law of Large Numbers, described in Definition 1. Details, including the proof of such properties, can be found here [23].

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, which either reduce the time taken for the chain to converge to the target distribution, or maximises the region of $\boldsymbol{\Theta}$ which the chain explores. Typically, as is the convention in this report, we will be using a Gaussian distribution for our 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 lead to the proposed points that are, typically, further away from the current point. This can lead to a low proportion of proposed steps being accepted, as many of the proposed points will fall into regions of negligible density. On the other hand, small variances value will mean proposals are near to the current point and are more likely to be accepted, due to the comparable denisties at both the current and the proposed point. In addition, for small variances, the chain will take a long time to full explore the parameter space and converge to the target distribution [24]. A method proposed by Gelman [24], 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 [25] that Σ should be updated prior to the chain reaching convergence. More precisely, we estimate each σ_i^2 using a variance estimator of the previously accepted states, multiplied by a particular scaling factor (11). In the M-H method, this would imply 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 covaraince matrix 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 keep Σ constant. Failing to do so can alter the ergodic properties of the chain.

$$\Sigma = \frac{2.4}{\sqrt{k}}\hat{I}, \text{ where } k \text{ is the dimension of } \boldsymbol{\theta} \text{ and } \hat{I} = \begin{pmatrix} \hat{\sigma_1^2} & 0 & \dots & 0 \\ 0 & \hat{\sigma_2^2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \hat{\sigma_k^2} \end{pmatrix} \tag{11}$$

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 work if the parameters are highly correlated [25]. We also 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 section, 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 practise, 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 the chain proposed by Geweke [26], Gelman and Rubin [27] and Raftery and Lewis [28]. In this report, we will be considering only the Gelman-Rubin $\hat{R}$ 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 $\hat{R}$ diagnostic being proposed at the time of writing.

3.4.1. *The Gelman-Rubin $\hat{R}$ Diagnostic.* The $\hat{R}$ method proposed in 1992 [27] 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 stationary' of the chains [24]. The reason for using multiple can be explained using Figure 8 below. In this toy example, we have ran two chains from 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 our target distribution and therefore have not converged. Problems like this, named *slow-mixing problems* [24], often occur when the target distribution is multi-modal; this can cause the a chain to get stuck at a particular peak.

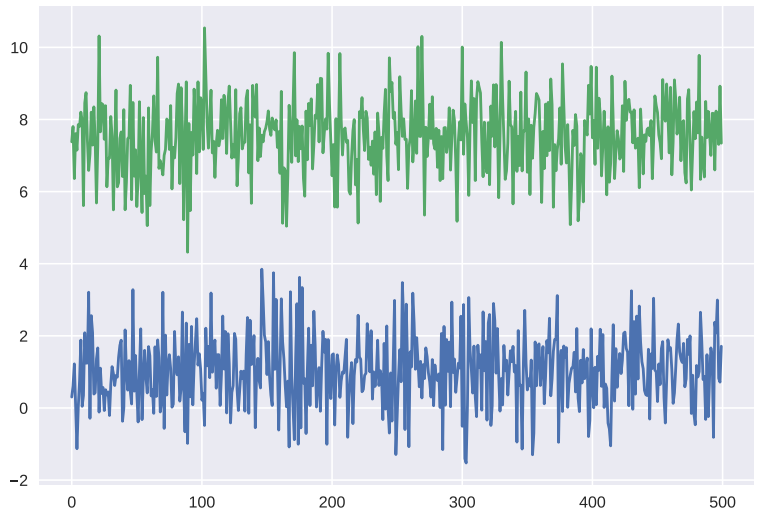


FIGURE 8. A toy example displaying how two chains looked to have converged to a stationary distribution but are in fact exploring two different peaks.

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

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

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, as given below:

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

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

From these statistics, we can estimate the variance of target distribution by using a weighted average of W and B. Such an estimator, overestimates the target variance but is unbiased when the chain starts in the stationary distribution or when $n \to \infty$ [27]:

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

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 (12) which approaches 1 from above as $n \to \infty$.

$$\hat{R} = \sqrt{\frac{\hat{\text{var}}(X)}{W}} \tag{12}$$

Typically, when performing simulation, we exit the burn-in period when $\hat{R} \leq 1 + \delta$ for some chosen δ (typically 0.1 [24]). To implement this statistic, we iteratively perform simulation of an MCMC method, stopping after some predetermined n to calculate (12). If the chain does does not exit the burn-in period then we continue the chain and re-evaluate after another n; if,

θ_i	x_1^0	x_2^0	x_3^0	x_4^0	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 our M-H algorithm.

instead, the chain does exit the burn-in period, then we run the chain to obtain samples from the stationary, target distribution.

3.5. **2-Parameter Logistic Equation Revisited with MCMC.** Now, we will revisit the inference problem that we will 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 (5) and apply the same priors to the model parameters. To asses the convergence of the algorithm, we will use the $\hat{R}$ diagnostic described in the previous section. We will finish the burn in period of our chain when the $\hat{R} \leq 1.05$, a slightly stricter value than the one suggested in [27] because of the discussion put forward in [29] when we consider 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 simultaneously for 1000 iterations each. For the i-th chain, we must specify an initial starting point 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 x^0 satisfies $p^*(x^0 \mid \boldsymbol{y}) > 0$. After specifying these initial conditions, we then run the Metropolis-Hastings algorithm on each chain. 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 figures like 9, *trace plots*.

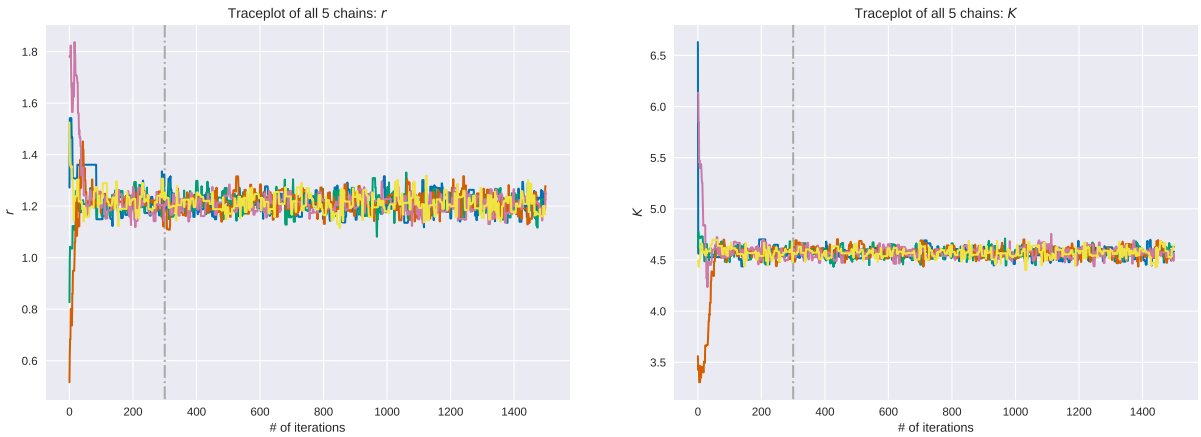
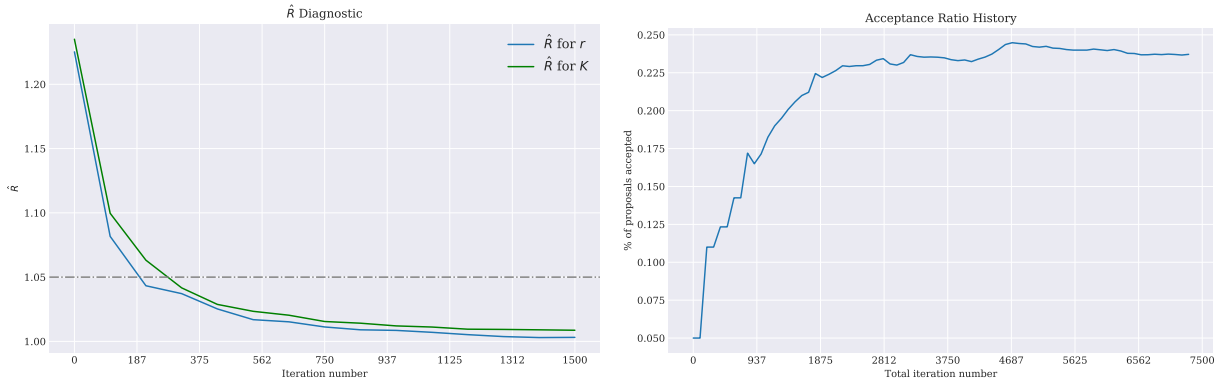


FIGURE 9. Trace plots for model parameters r and K respectively.

To determine when how many samples should be discarded as the burn in period, we look to the Gelman-Rubin $\hat{R}$ diagnostic between the five chains for each parameter. Here, we compute the $\hat{R}$ statistic for each parameter at every one hundred iterations of the chains. By doing so, we can monitor how the $\hat{R}$, and therefore the convergence, of the chains changes over time. To identify the region of burn in, we look to when the $\hat{R}$ 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. We can identify that the first time the $\hat{R}$ falls below 1.05 for both parameter is at the third time



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

(B) Acceptance ratio over the history of all five chains.

FIGURE 10

we evaluate the $\hat{R}$ 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.

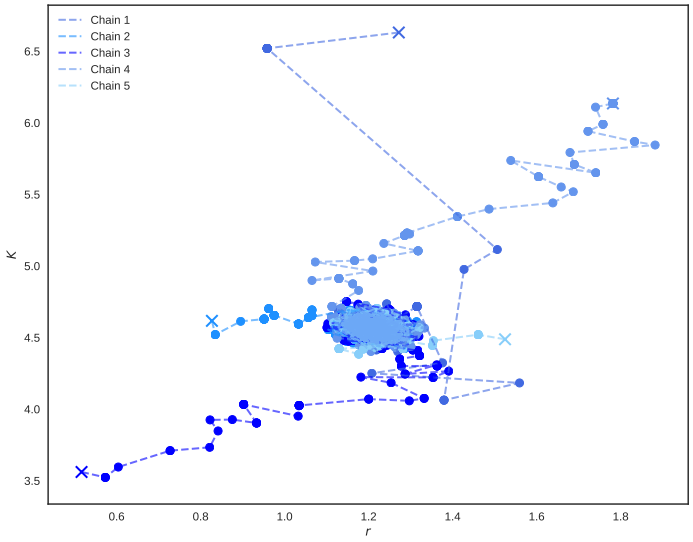


FIGURE 11. Monitoring how all five chains explore the parameter space from their initial states.

3.5.1. *Inference.* Now that we separated the chains into the burn in and the samples for each parameter, 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

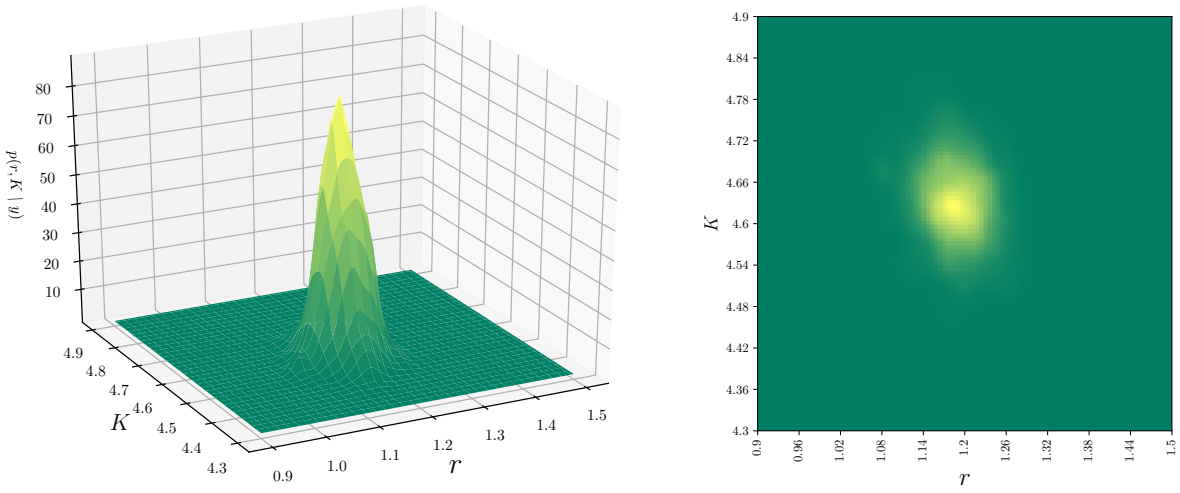
$$\hat{\boldsymbol{\theta}} = \frac{1}{n\,m} \sum_{j=1}^{m} \sum_{i=1}^{n} \boldsymbol{\theta}_j^{(i)}$$

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% (Bayesian) credible intervals, given in Table 3.

θ_i	$\hat{\theta}$	95% C.I.
r	1.28	[1.144, 1.282]
K	4.6887	[4.471, 4.664]

TABLE 3. 95% Credible Intervals obtain from our samples.

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 instead 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 here [30]. 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 our samples from the Metropolis-Hastings algorithm and produce a grid in which we have estimated the probability density function of the posterior at each point, displayed in Figure 12.



(A) Surface Plot

(B) Heat Plot

FIGURE 12. Surface and Heat plot of our approximation to the posterior after perforing KDE.

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

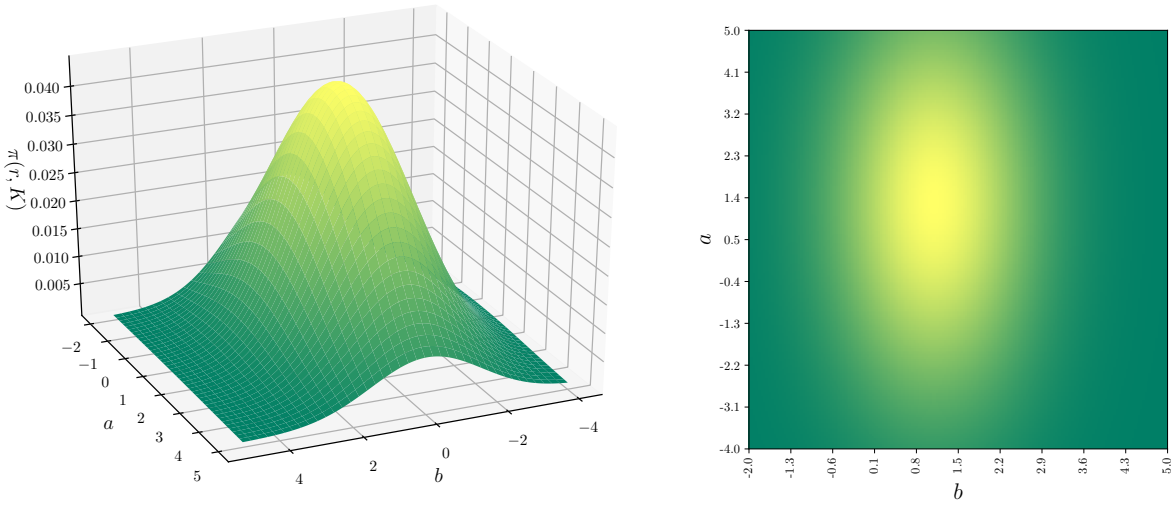
3.6.1. *ODE System.* We will be considering an artificially created ODE, described below.

$$\dot{x} = -x + g(a)\,xy, \quad g(a) = -\frac{1}{20}(a - \frac{1}{2})^2 + 2$$
$$\dot{y} = y - f(b)\,xy, \quad f(b) = 1 + \frac{3}{20}\sin(b)$$

We wish to perform inference on the posterior of 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. In this case, g and f are interaction functions, akin to that of functional responses in ecological modelling [31]. We therefore also assume 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 an b independently. Figure 13 displays a heat plot of our prior over a subset of $\mathbb{R}^2$.

$$a \sim \mathcal{N}(1.3, 4), \quad b \sim \mathcal{N}(1.1, 2)$$



(A) Surface Plot (B) Heat Plot

FIGURE 13. Surface and Heat plot of our joint prior.

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 our 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 on Figure 14.

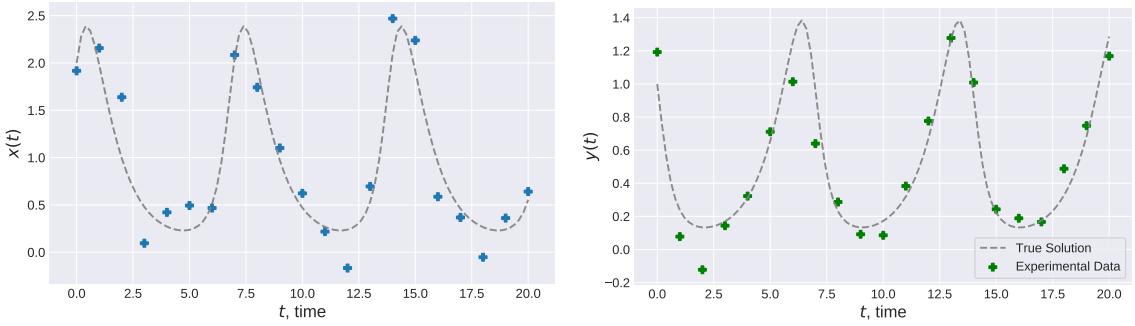


FIGURE 14. Plots of the true solution of $x(t)$ and $y(t)$ and experimental data. Data generated with true parameter values $(a, b) = (2.1, 0.6)$ and initial conditions $x(0) = 2$ and $y(0) = 1$.

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. Starting points for each chain in our M-H algorithm.

3.6.3. *Inference.* To obtain samples from our 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 our time span using the proposed model parameters; this drastically increases the computing time of each chain, since in our previous example we simply had to input our parameters into the analytical 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 use five chains with starting points sampled from the joint prior, shown below. We again terminate the burn in period of our chains when the $\hat{R}$ falls below 1.05. Below shows the results of the five chains when we run our each for 5000 iterations.

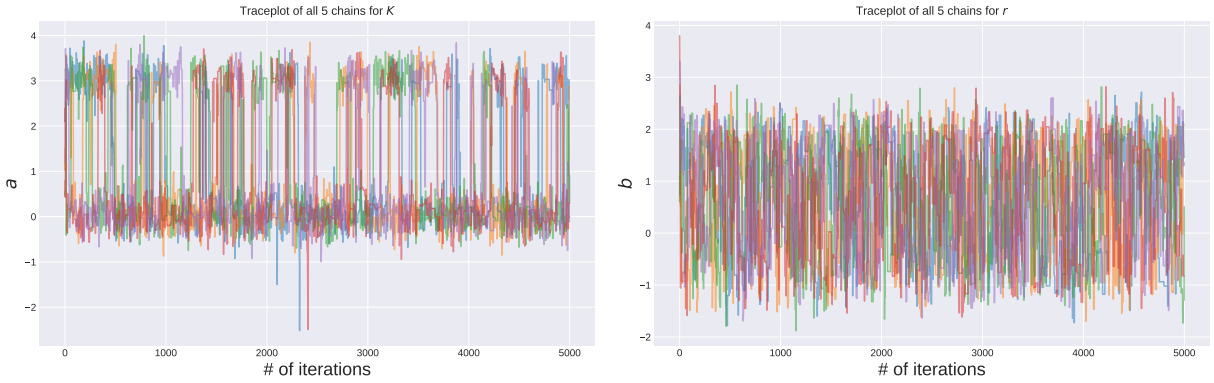
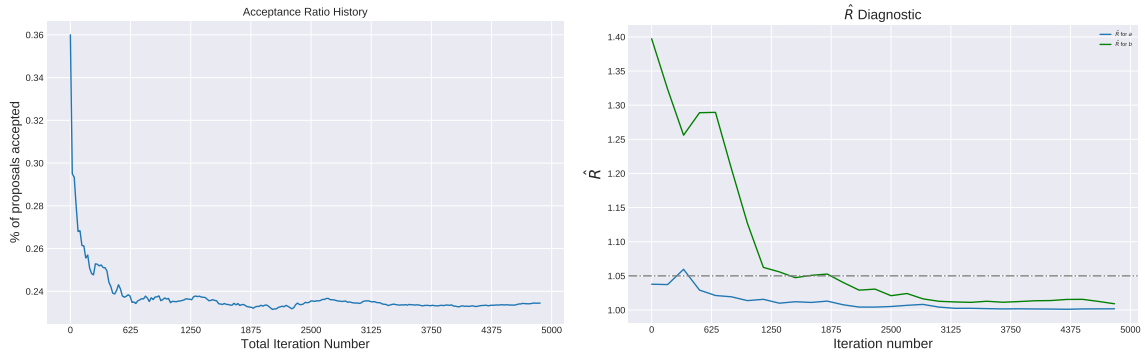


FIGURE 15. Trace plots for parameters a and b respectively.

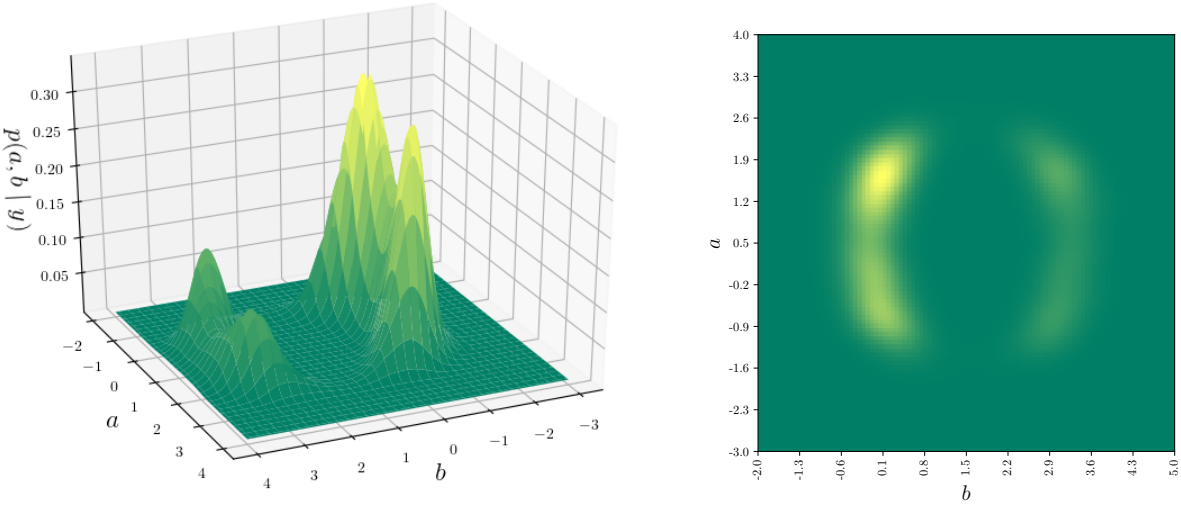
From analysing the $\hat{R}$ plot for both parameters, displayed in Figure 16, we see that the first time they both fall under 1.05 is at sample point 1900. It is important to understand that posteriors like the one presented here require require a longer run time to obtain a low $\hat{R}$, this is because of the bi-modality of our surface. Having an $\hat{R}$ close to 1 implies that all chains have visited both the peaks in a similar proportion. This is why we have had to run our chain for significantly longer than in Section 3.5. After discarding the initial burn in samples, we again perform Kernel Density Estimation from our samples to produce a heat and surface plot of our posterior respectively, shown in Figure 17. In addition, we also produce a heat plot of the grid approximation to this posterior to asses the accuracy of the Metropolis-Hastings, shown in Figure 18.



(A) Acceptance Ratio for Chains

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

FIGURE 16



(A) KDE Surface Plot

(B) KDE Heat Plot

FIGURE 17. Surface and Heat Plots of the Density Estimation after performing Metropolis-Hastings

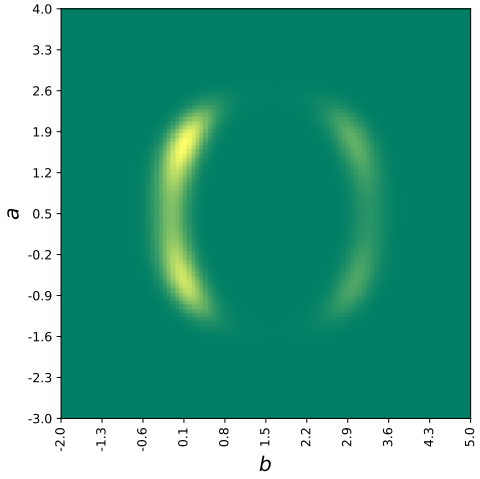


FIGURE 18. Fine grid approximation to ODE problem in Section 3.6.1

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 computational resources available 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 a minimal tuning in order to 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 here [2], 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 certain probability distributions which it struggles with, before looking at how these issues are addressed in new MCMC methods.

3.7.1. *ODE Parameter Sensitivity.* The Metropolis-Hastings algorithm encounters a problem with the sensitivity of the parameters in the system of differential equations. A parameter is considered to be *sensitive* in an ODE system if, when we change we change the value of the parameter slightly, the output of the ODE system changes substantially [32]. 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, giving a low 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.

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

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.

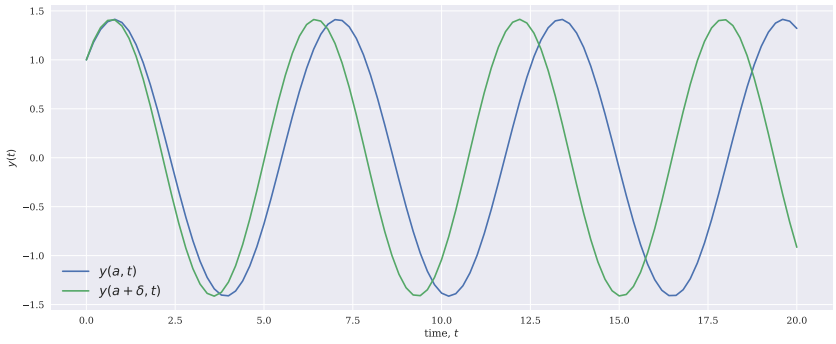


FIGURE 19. Observing how small change in a can cause significant change in output.

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) [33] calculate the sensitivity of the unnormalised posterior using partial derivatives to make an informed choice about the proposed steps, allowing them to consider the local curvature of the surface. By using gradient-based information, methods like HMC typically converge to the target distribution quicker.

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 below. This posterior can be generated from a problem hosted on my GitHub.

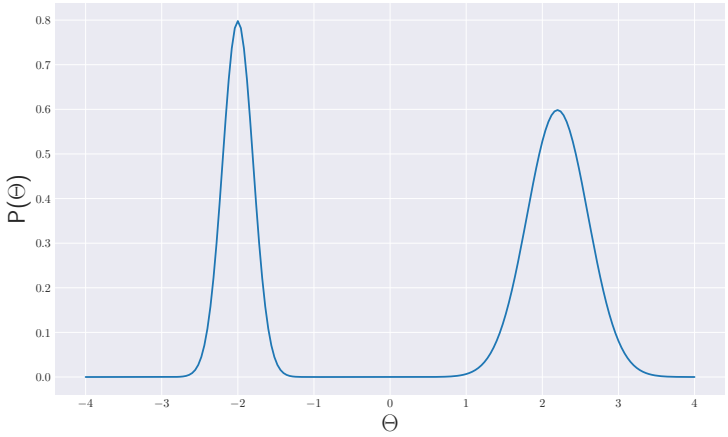
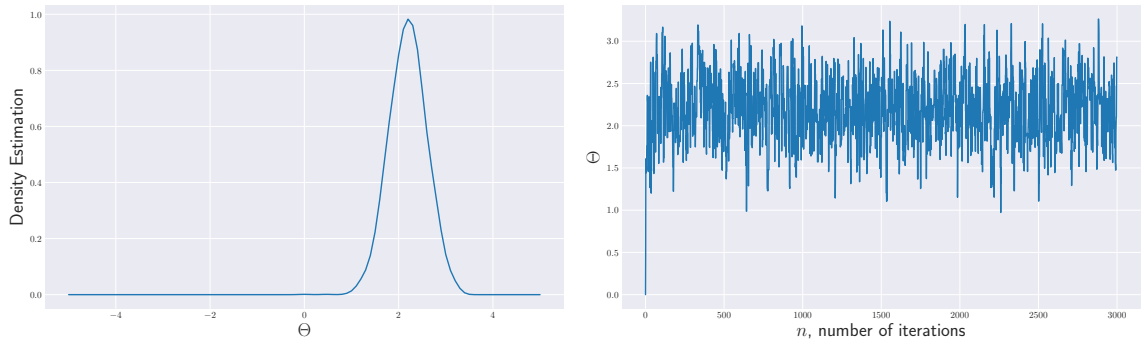


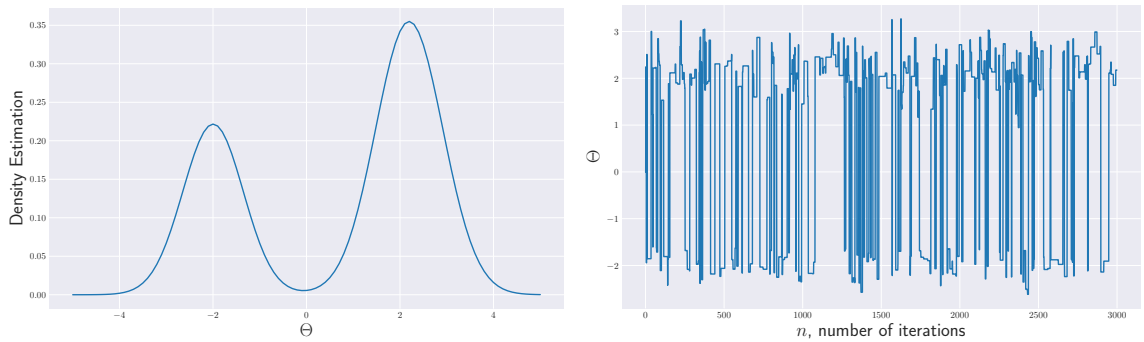
FIGURE 20. An arbitrary bi-modal posterior

As described in Section 3.3.3, we only alter the covariance matrix of our proposal distribution during the burn in and when the current acceptance ratio falls out of our pre-defined bounds. Multi-modal distributions struggle present problems for random-walk MCMC methods, like Metropolis Hastings, with constant proposal distributions. 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 our 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 ran for 1000 iterations. On the other hand, if we define our proposal distribution q to have a large variance, $\sigma = 5$, we may visit the 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 histogram generated from the samples, shown 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 type of problem can be addressed using further modifications to the Metropolis-Hastings algorithm. Methods such as the Adaptive-Metropolis algorithm have been suggested [34]. This particular method estimates the covariance matrix of the proposal distribution using the history of accepted samples, similar to our approach during the burn in period. For this algorithm, however, the updating of the covariance matrix occurs after the burn-in period also. Algorithms such as the Adaptive-Metropolis must be carefully created to ensure that they do not violate the ergodicity properties required for 3.2. The advantages of such methods allows the chain to explore uni-modal peaks in more detail upon convergence.

4. SUMMARY

Bayesian inference has become a popular approach to statistical modelling due to the combination of model flexibility and the use of a prior distribution, allowing the user to fully explain their pre-experiment knowledge. For complex models to be used, we must find a computationally efficient and accurate method that allows us to work with models with 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 problems, there are certain problems in which VI provides

poor approximations to (7), particularly in higher dimensions. For a detailed survey on Variational Inference and its uses, the reader is directed to [35]. In addition to VI, topics such as Slice Sampling and Temperatured Annealing are also current branches of research, which have uses in approximate inference [36][1]. 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, the read is directed towards the CSIDE (Competitive Statistical Inference for Ordinary Differential Equations) webpage [3]. CSIDE, as mentioned in the introduction, is an annual conference hosted to asses what "inference methods are currently state of the art" [3]. In particular, the competition explored in-depth parameter estimation in various settings, from cardiac excitation to cell migration. In addition to the work of CSIDE, [37] gives a well-written account on the practical implementation of the Metropolis-Hastings algorithm in geography and [38] shows how the Hamiltonian Monte Carlo can be used in ODEs relating to neurology.

4.1. **Further Work.** To further explore the space of MCMC methods and their uses in posterior approximation, looking at space itself would be an interesting problem. In the setting of astrophysics and astronomy, MCMC and the Python package `emcee` [39] are popular when considering Bayesian parameter estimation. These problems are particularly interesting because of the large number of parameters we are dealing with, along with intricate systems that require accurate numerical approximations. This problem would look at how to make MCMC effective in higher dimensions and how to efficiently integrate it into high performance computing.

References

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

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

[3] 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].

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

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

[6] Peter D Hoff. *A first course in Bayesian statistical methods*, pages 4–40. Springer, 2009.

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

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

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

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

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

[19] 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.

[20] 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.
[21] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2:e55, 2016.
[22] Siddhartha Chib and Edward Greenberg. Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.
[23] 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.
[24] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC Texts in Statistical Science, 3 edition, 2004.
[25] Gareth O Roberts, Jeffrey S Rosenthal, et al. Optimal scaling for various Metropolis-Hastings algorithms. *Statistical science*, 16(4):351–367, 2001.
[26] 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.
[27] Andrew Gelman, Donald B Rubin, et al. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–472, 1992.
[28] 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.
[29] Dootika Vats and Christina Knudson. Revisiting the Gelman-Rubin Diagnostic. *arXiv preprint arXiv:1812.09384*, 2018.
[30] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.
[31] C S Holling. The Components of Predation as Revealed by a Study of Small-Mammal Predation of the European Pine Sawfly. *The Canadian Entomologist*, 91(5):293–320, 1959.
[32] 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.
[33] Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.
[34] Heikki Haario, Eero Saksman, and Johanna Tamminen. An Adaptive Metropolis Algorithm. *Bernoulli*, 7(2):223–242, 2001.
[35] 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.
[36] Radford M Neal et al. Slice sampling. *The annals of statistics*, 31(3):705–767, 2003.
[37] 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.
[38] Biswa Sengupta, Karl J Friston, and Will D Pennny. Gradient-based MCMC samplers for dynamic causal modelling. *NeuroImage*, 125:1107 – 1118, 2016.
[39] Daniel Foreman-Mackey, David W Hogg, Dustin Lang, and Jonathan Goodman. emcee: the MCMC hammer. *Publications of the Astronomical Society of the Pacific*, 125(925):306, 2013.