Gradient-Free Optimal Postprocessing of MCMC Output

by

Artem Glebov

Department of Mathematics
King's College London
The Strand, London WC2R 2LS
United Kingdom

Abstract

The dissertation reports the results of implementing the gradient-free kernel Stein discrepancy of Fisher and Oates (2024) within the Stein thinning algorithm of Riabiz et al. (2022). We evaluate the performance of the proposed algorithm on a Gaussian mixture target distribution and in a Bayesian inverse problem for the Lotka-Volterra model. The results confirm the feasibility of the proposed algorithm and highlight the primary challenge in its practical application: the requirement for an auxiliary distribution to be provided by the user.

Contents

Introduction

The Stein thinning algorithm was recently proposed by Riabiz et al. (2022) motivated by the problem of selecting a subsample from Markov Chain Monte Carlo (MCMC) output for further expensive processing. The algorithm relies on the kernel Stein discrepancy (KSD), whose calculation involves the gradients of the log-density of the target distribution. The requirement to provide the gradients imposes a significant computational cost on the practitioner attempting to use it for non-trivial distributions, such as posteriors in Bayesian inverse problems. Fisher and Oates (2024) developed a way around this challenge by formulating a gradient-free version of KSD, which the authors applied in the context of importance sampling and variational inference.

We adopt the proposal of Fisher and Oates (2024) to implement a gradient-free version of the original Stein thinning algorithm, and evaluate the newly developed algorithm on simple but illustrative cases of the Gaussian mixture and Bayesian inference for the parameters of the Lotka-Volterra model, demonstrating the feasibility of the algorithm, and highlighting practical challenges for its application.

The report is organised as follows. Chapter 1 reviews MCMC methods and discusses the challenges of applying MCMC to practical problems. It proceeds to explain how the Stein thinning algorithm of Riabiz et al. (2022) offers a solution for retrospective bias removal and compression of samples. The gradient-free KSD of Fisher and Oates (2024) is then introduced. Chapter 2 describes our proposed approach, supported by numerical experiments. Finally, Chapter 3 summarises the finding of the experiments and suggests ideas for further research. Additional derivations and figures are included in the Appendices.

The Python code reproducing the experiments in this report can be found at:

`https://github.com/aglebov/gradient-free-mcmc-postprocessing/code`.

The implementation of the gradient-free kernel Stein thinning was contributed by the author directly to the `stein-thinning` Python library:

`https://github.com/wilson-ye-chen/stein_thinning`.

Notation

For a matrix A, we use use A^T to denote the transpose of the matrix. The matrix $A^{1/2}$ is the square root of matrix A if it satisfies $A^{1/2}A^{1/2} = A$. A^{-1} is the inverse of matrix A, if it exists. I stands for the identity matrix of an appropriate dimension.

We use $\langle x, y\rangle$ to denote the scalar product of x and y, and $\|x\|$ to denote the norm of x.

For a function $f : \mathbb{R}^d \to \mathbb{R}$, $\nabla f(x)$ denotes the gradient of $f(x)$ given by the vector

$$\nabla f(x) \coloneqq \left(\frac{\partial f(x)}{\partial x_1}, \ldots, \frac{\partial f(x)}{\partial x_d}\right)^T,$$

where x_i are the components of x for $i \in \{1, \ldots, d\}$. For a function of two arguments $f(x, y) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, we define the gradient w.r.t. to each argument:

$$\nabla_x f(x, y) \coloneqq \left(\frac{\partial f(x, y)}{\partial x_1}, \ldots, \frac{\partial f(x, y)}{\partial x_d}\right)^T,$$
$$\nabla_y f(x, y) \coloneqq \left(\frac{\partial f(x, y)}{\partial y_1}, \ldots, \frac{\partial f(x, y)}{\partial y_d}\right)^T,$$

where x_i and y_i are the components of x and y, respectively, for $i \in \{1, \ldots, d\}$. The divergence operator $\nabla_x \cdot \nabla_y$ is then defined as

$$(\nabla_x \cdot \nabla_y) f(x, y) \coloneqq \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i \, \partial y_i} f(x, y).$$

For a sequence of probability measures P_m and a probability measure P on a measurable space $\mathcal{X}$, $P_m \Rightarrow P$ as $m \to \infty$ denotes weak convergence:

$$\int_{\mathcal{X}} f \, \mathrm{d}P_m \to \int_{\mathcal{X}} f \, \mathrm{d}P \quad \text{as } m \to \infty \quad \text{for all bounded, continuous functions } f.$$

Chapter 1

Background

1.1 Markov chain Monte Carlo (MCMC)

Markov chain Monte Carlo (MCMC) are a popular class of algorithms for sampling from complex probability distributions.

The need to sample from a probability distribution arises in exploratory analysis as well as when analytical expressions are unavailable for quantities of interest, such as the modes or quantiles of the distribution, or for expectations with respect to the distribution, so a Monte Carlo simulation is used to obtain approximations instead. Such cases are frequent in Bayesian analysis, where the posterior density often has a complex structure with an analytically intractable normalising constant.

In the simplest cases where the inverse cumulative density function of a distribution is available, sampling can be done using the straightforward inverse method. Bespoke sampling methods, such as the Box-Muller transform for the normal distribution, can occasionally be developed by exploiting the properties of specific distributions. See Chapters 2 and 3 in Robert and Casella (2004) for an overview of these methods. In many cases, however, the complexity of the target distribution and the dimensionality of the parameter space render the simple methods infeasible or inefficient. The class of MCMC algorithms has arisen to address the challenge.

Describe alternatives: accept-reject and importance sampling

Mention what makes MCMC different from the other methods

An MCMC algorithm proceeds by sequentially constructing a chain of samples[1] $x_1, x_2, \ldots$, where each sample is drawn from a Markov transition kernel Q conditional on the preceding value:

$$x_{n+1} \sim Q(x_{n+1}|x_n).$$

The distribution Q is known as the transition kernel and is selected so that it is easy to sample from and to ensure asymptotic convergence to the target distribution Π:

$$x_n \xrightarrow{d} \Pi \quad \text{as} \quad n \to \infty.$$

Two classical variations of this technique are the Metropolis-Hastings and Gibbs algorithms.

Metropolis-Hastings algorithm. The algorithm due to Metropolis et al. (1953) and Hastings (1970) uses an auxiliary distribution q to sample a proposed value

$$x' \sim q(x'|x_n),$$

[1] These are also sometimes called "draws". In this report, we follow the literature in using the term "sample" both for a single element in an MCMC chain and for all such elements taken together as a sample from the target distribution.

which is then accepted with probability

$$\alpha(x_n, x') = 1 \wedge \frac{\pi(x')}{\pi(x_n)} \frac{q(x_n|x')}{q(x'|x_n)}.$$

If x' is accepted, the algorithm sets $x_{n+1} = x'$. If x' is rejected, the value remains unchanged: $x_{n+1} = x_n$.

Consider using a different notation to avoid the confusion between the density of the proposal q and the transition kernel Q.

The common choice for the proposal distribution q is a symmetric proposal satisfying $q(x'|x_n) = q(x_n|x')$, so that the ratio of these two quantities disappears from the expression for the acceptance probability:

$$\alpha(x_n, x') = 1 \wedge \frac{\pi(x')}{\pi(x_n)}.$$

In the special case where $q(x'|x_n) = q(x' - x_n)$ we obtain a random walk proposal:

$$x' = x_n + z,$$

where the step z is sampled from the proposal distribution of the algorithm, such as the multivariate normal distribution. The operation of the algorithm then resembles a random walk across the domain of the target distribution where steps towards areas of lower probability are more likely to be rejected. The scale of the step distribution determines the average size of the jump that the algorithm can make at each iteration and thus the speed of traversal of the target domain.

An alternative to symmetric proposals is an independence proposal satisfying $q(x'|x_n) = q(x')$.

Cite the ST03 lecture notes or Robert & Casella

Gibbs algorithm. Suppose x is a d-dimensional vector and the components $x^{(1)}$, $x^{(2)}$, …, $x^{(d)}$ can be partitioned in such a way that we can sample the components belonging to each partition while keeping the components in other partitions fixed. That is, let $I_i \subset \{1, \ldots, d\}$ with $\cup_{i=1}^{k} I_i = \{1, \ldots, d\}$ for some k and $I_i \cap I_j = \emptyset$ for $i \neq j$, and assume we can sample

$$x^{(I_i)} \sim f_i\left(x^{(I_i)}|x^{(I_1,\ldots,I_{i-1},I_{i+1},\ldots,I_k)}\right).$$

The sample x_{n+1} can then be constructed by sequentially sampling for each partition:

$$x_{n+1}^{(I_i)} \sim f_i\left(x^{(I_i)}|x_{n+1}^{(I_1,\ldots,I_{i-1})}, x_n^{(I_{i+1},\ldots,I_k)}\right).$$

Note that the newly sampled values $x_{n+1}^{(I_1,\ldots,I_{i-1})}$ enter the computation for subsequent partitions.

Read and cite the original paper for Gibbs sampler

Consider simplifying this description

Mention HMC and other recent variations

Development of general-purpose and specialised MCMC algorithms remains an active area of research.

1.2 Challenges of running MCMC

While the asymptotic convergence of MCMC samples to the target distribution is guaranteed, no general guarantee is available for finite samples, resulting in several interrelated challenges that a practitioner faces when applying this class of algorithms:

Consider adding a reference

1. The choice of a starting point for a chain affects the speed of convergence to the target distribution.

2. For a multimodal distribution, the algorithm might struggle to move between the modes within a feasible time. This problem becomes especially acute in high dimensions.

3. The scale of the proposal distribution must be calibrated to ensure that the algorithm is able to explore the domain of the target distribution efficiently.

4. Assessing how close an MCMC chain is to convergence is difficult, since the knowledge about the target distribution often comes from the chain itself.

5. In order to eliminate the impact of the starting point, it can be useful to discard the initial iterations of an MCMC chain, which are considered as "burn-in". Selecting the optimal length of the burn-in period is contingent on being able to detect convergence.

6. The sequential procedure of constructing a chain introduces autocorrelation between the samples, which leads to increased variance of resulting estimators.

7. The large number of samples resulting from an MCMC algorithm might need to be summarised for subsequent analysis, particularly when the cost of using all available samples is too high. Such situations arise when samples obtained from MCMC are used as starting points for further expensive simulations.

The first three challenges require decisions to be made upfront before running the algorithm or adaptively during its run. In order to address the impact of the starting point, running multiple chains with starting points sampled from an overdispersed distribution is recommended (Gelman and Rubin (1992)). This approach has the added benefit of increasing the chance of discovering the modes of the target distribution, although it does not provide a guarantee in this respect.

Mention perfect sampling.

The scaling of the step distribution in the random-walk Metropolis-Hastings algorithm is commonly tuned to target the acceptance rate of roughly 0.234 for proposed samples (Gelman et al. (1996, 1997); Roberts and Rosenthal (2001)), which balances the speed of traversal and the computational effort generating samples that end up rejected.

Comparing the summary statistics of several chains (Gelman and Rubin (1992); Brooks and Gelman (1998); Vehtari et al. (2021)) offers a way to detect a lack of convergence at the cost of additional computation. Alternatively, the comparison can be applied to batches of samples from a single chain, as proposed by Vats and Knudson (2021). Convergence detection can be used to terminate the algorithm once a chosen criterion is satisfied. It should be noted that convergence criteria establish a necessary but not sufficient condition for convergence, so the outcomes need to be interpreted accordingly.

The last three challenges are typically addressed by post-processing a sample from a completed MCMC run. A recent proposal by Riabiz et al. (2022) addresses these challenges by selecting a fixed-size subset of samples from an MCMC run such that the empirical distribution given by the subset best approximates the target distribution. In the following section, we consider their approach in greater detail.

Read and cite Cowles and Carlin (1996) regarding the choice of burn-in length.

1.3 Stein thinning

Given MCMC output $(x_i)_{i=1}^n$ of length n, the empirical approximation of the target posterior distribution is given by

$$\frac{1}{n}\sum_{i=1}^{n}\delta(x_i), \tag{1.1}$$

where $\delta(x)$ is the Dirac delta function. Riabiz et al. (2022) set out to identify $m \ll n$ indices $\pi(j) \in \{1,\dots,n\}$ with $j \in \{1,\dots,m\}$, such that the approximation provided by the subset of samples

$$\frac{1}{m}\sum_{j=1}^{m}\delta(x_{\pi(j)}) \tag{1.2}$$

is closest to the target distribution. The criterion of proximity is based on the kernel Stein discrepancy, itself a special case of the integral probability metric.

The integral probability metric (Müller (1997)) between two distributions P and P' is defined as

$$\mathcal{D}_{\mathcal{F}}(P,P') \coloneqq \sup_{f\in\mathcal{F}}\left|\int_{\mathcal{X}} f\,\mathrm{d}P - \int_{\mathcal{X}} f\,\mathrm{d}P'\right|, \tag{1.3}$$

where $\mathcal{X}$ is a measurable space on which both P and P' are defined and $\mathcal{F}$ is a set of test functions. The set $\mathcal{F}$ needs to be rich enough to detect differences between P and P', that is:

$$\mathcal{D}_{\mathcal{F}}(P,P') = 0 \iff P = P'.$$

If this property holds, the metric is said to be measure determining. Depending on the choice of $\mathcal{F}$, the definition (1.3) gives rise to different classes of probability metrics, including the well-known Kolmogorov distance, Wasserstein distance and total variation distance.

If P is taken to be the target distribution of an MCMC algorithm, evaluating (1.3) poses two practical challenges: the integral $\int_{\mathcal{X}} f\,\mathrm{d}P$ is often analytically intractable, and the supremum requires a non-trivial optimisation procedure to find.

The need to integrate with respect to P can be eliminated if we find a set of function $\mathcal{F}$ for which $\int_{\mathcal{X}} f\,\mathrm{d}P = 0$ for all $f \in \mathcal{F}$. The expression (1.3) then simplifies to

$$\mathcal{D}_{\mathcal{F}}(P,P') = \sup_{f\in\mathcal{F}}\left|\int_{\mathcal{X}} f\,\mathrm{d}P'\right|. \tag{1.4}$$

Gorham and Mackey (2015) propose choosing such a set $\mathcal{F}$ based on the observation that the infinitesimal generator

$$(\mathcal{L}u)(x) \coloneqq \lim_{t\to 0}\frac{\mathbb{E}[u(Z_t)|Z_0 = x] - u(x)}{t} \quad \text{for } u : \mathbb{R}^d \to \mathbb{R}$$

of a Markov process $(Z_t)_{t\geq 0}$ with stationary distribution P satisfies

$$\mathbb{E}[(\mathcal{L}u)(Z)] = 0 \tag{1.5}$$

under mild conditions on $\mathcal{L}$ and u. In the specific case of an overdamped Langevin diffusion

Check the conditions.

$$\mathrm{d}Z_t = \frac{1}{2}\nabla \log p(Z_t)\,\mathrm{d}t + \mathrm{d}W_t,$$

where W_t is the standard Brownian motion, the infinitesimal generator becomes

$$(\mathcal{L}_P u)(x) = \frac{1}{2}\langle \nabla u(x), \nabla \log p(x)\rangle + \frac{1}{2}\langle \nabla, \nabla u(x)\rangle.$$

Denoting $g = \frac{1}{2}\nabla u$, they obtain the Stein operator

$$\mathcal{A}_P g \coloneqq \langle g, \nabla \log p \rangle + \langle \nabla, g \rangle = \langle p^{-1}\nabla, pg \rangle, \tag{1.6}$$

and rewrite (1.4) via (1.5) as

$$\mathcal{D}_{P,\mathcal{G}}(P') = \sup_{g \in \mathcal{G}} \left| \int_{\mathcal{X}} \mathcal{A}_P g \, \mathrm{d}P' \right| \tag{1.7}$$

for a suitably chosen set $\mathcal{G}$. Since p only appears in (1.6) through $\nabla \log p$, the normalising constant of p is not required to evaluate $\mathcal{D}_{P,\mathcal{G}}(P')$: for any constant $c > 0$ we have $\nabla \log(cp(x)) = \nabla \log p(x)$.

To remove the optimisation step in the calculation of (1.7), Gorham and Mackey (2017) employ a reproducing kernel Hilbert space (RKHS) $\mathcal{H}(k)$ with kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ satisfying the reproducing property:

$$f(x) = \langle f, k(x, \cdot) \rangle \quad \text{for } f \in \mathcal{H}(k).$$

Taking the set

$$\mathcal{G} \coloneqq \left\{ \mathrm{g} : \mathbb{R}^d \to \mathbb{R}^d \,\middle|\, \sum_{i=1}^{d} \|g_i\|^2_{\mathcal{H}(k)} \leq 1 \right\} \tag{1.8}$$

which defines a unit-ball in a Cartesian product of d copies $\mathcal{H}(k)$, Proposition 2 in Gorham and Mackey (2017) establishes that

$$\mathcal{D}_P^2(P') \coloneqq= \mathcal{D}_{P,\mathcal{G}}(P') = \iint_{\mathcal{X}} k_P(x, y) \, \mathrm{d}p'(x) \, \mathrm{d}p'(y), \tag{1.9}$$

where p' is the density of P', and $k_P(x, y)$ is given by

$$\begin{aligned} k_P(x, y) \coloneqq& (\nabla_x \cdot \nabla_y) k(x, y) \\ &+ \langle \nabla_x k(x, y), \nabla_y \log p(y) \rangle + \langle \nabla_y k(x, y), \nabla_x \log p(x) \rangle \\ &+ k(x, y) \langle \nabla_x \log p(x), \nabla_y \log p(y) \rangle. \end{aligned} \tag{1.10}$$

If P' is a discrete distribution, as in (1.1), the squared discrepancy (1.9) becomes

$$\mathcal{D}_P^2 \left(\frac{1}{n} \sum_{i=1}^{n} \delta(x_i) \right) = \frac{1}{n^2} \sum_{i,j=1}^{n} k_P(x_i, x_j), \tag{1.11}$$

i.e. the average of the elements in the Gram matrix of $k_P(x_i, x_j)$, which is straightforward to compute.

When $k(x, y)$ is chosen to be the inverse multiquadratic kernel (IMQ)

$$k(x, y) = \left(c^2 + \|\Gamma^{-1/2}(x - y)\| \right)^{\beta} \tag{1.12}$$

with $\beta \in (-1, 0)$ and $\Gamma = I$, Gorham and Mackey (2017) demonstrate that $\mathcal{D}_P(P')$ provides convergence control: if $\mathcal{D}_P(P'_m) \to 0$, then $P'_m \Rightarrow P$ for a sequence of distributions P'_m under suitable conditions (Theorem 8). Theorem 4 by Chen et al. (2019) justifies the introduction of Γ in IMQ. The expression for $k_P(x, y)$ when $k(x, y)$ is taken to be the inverse multiquadratic kernel is derived in Appendix A.1.

Check why it's not obvious that Γ can be introduced.

The hyperparameters c, Γ and β in (1.12) must be provided by the user. The constant c in (1.12) can be set to 1 without loss of generality, and the positive-definite preconditioner matrix Γ can be chosen to exploit the geometry of the parameter space. Riabiz et al. (2022) suggest several choices for Γ, in particular the identity matrix scaled by the median Euclidean distance between points in the sample, possibly further rescaled by $(\log m)^{-1/2}$ where m is the desired cardinality of the thinned sample, or the sample covariance matrix. Based on empirical evaluation, Gorham and Mackey (2017) and Riabiz et al. (2022) settle on the value $\beta = -\frac{1}{2}$.

Algorithm 1: Stein thinning.

Data: Sample $(x_i)_{i=1}^n$ from MCMC, desired cardinality $m \in \mathbb{N}$.
Result: Indices π of a sequence $(x_{\pi(j)})_{j=1}^m \subset \{x_i\}_{i=1}^n$ where $\pi(j) \in \{1, \ldots, n\}$.
for $j = 1, \ldots, m$ **do**

$$\pi(j) \in \operatorname*{arg\,min}_{i=1,\ldots,n} \frac{k_P(x_i, x_i)}{2} + \sum_{j'=1}^{j-1} k_P(x_{\pi(j'}, x_i)$$

end

Other choices of kernels are possible and offer convergence control, as demonstrated by Chen et al. (2018), however IMQ performed on par or better than the alternatives considered by the authors.

Equipped with the kernel Stein discrepancy (KSD) as defined above, Riabiz et al. (2022) develop a greedy optimisation algorithm to select a subset of points from a sample that minimises the total kernel Stein discrepancy. Rather than attempting to evaluate KSD for all $\binom{n}{m}$ combinations of points, they construct a subsample iteratively, each time picking a point that minimises the KSD with previously selected points. We reproduce their procedure verbatim in Algorithm 1 for the reader's convenience.

The algorithm was implemented by the authors and made available in the open-source library `stein-thinning`. The computational complexity of the provided implementation is $O(nm^2)$, allowing the algorithm to be used for large n when m is fixed.

The strength of the method lies in its ability to correct for bias in the input sample, as established by Theorem 3 in Riabiz et al. (2022), meaning that the algorithm can be applied to samples from MCMC chains that have not converged to the target distribution, provided that its domain is sufficiently explored by the chains. The limitation of the method comes from its reliance on the gradients of the log-target, which may be expensive to compute.

1.4 Gradient-free kernel Stein discrepancy

Cite cases where gradient cannot be computed easily.

To address the limitation of the kernel Stein discrepancy, Fisher and Oates (2024) propose the gradient-free Stein operator $\mathcal{S}_{P,Q}$ defined for any differentiable function g as

$$\mathcal{S}_{P,Q} g \coloneqq \frac{q}{p}(\langle g, \nabla \log q \rangle + \langle \nabla, g \rangle) = \frac{q}{p}\langle q^{-1}\nabla, qg \rangle. \tag{1.13}$$

This definition generalises expression (1.6) by introducing an auxiliary distribution Q with density q chosen such that its gradient is easily computable. When $q = p$, we recover the original Langevin Stein operator (1.6).

Fisher and Oates (2024) proceed to show in their Proposition 1 that, under certain regularity conditions,

$$\int_{\mathcal{X}} \mathcal{S}_{P,Q} g \, \mathrm{d}P = 0$$

for any function g whose first derivatives exist and are bounded, allowing them to define the gradient-free Stein discrepancy

$$\mathcal{D}_{P,Q}(P') = \sup_{g \in \mathcal{G}} \left| \int_{\mathcal{X}} \mathcal{S}_{P,Q} g \, \mathrm{d}P' \right|$$

by analogy with (1.7). Taking $\mathcal{G}$ again to be the unit-ball (1.8), Proposition 7 in Fisher and Oates (2024) establishes that

$$\mathcal{D}^2_{P,Q}(P') = \iint_{\mathcal{X}} \frac{q(x)}{p(x)} \frac{q(y)}{p(y)} k_Q(x, y) \, \mathrm{d}p'(x) \, \mathrm{d}p'(y),$$

where p' is the density of distribution P' and $k_Q(x, y)$ is given by (1.10) but with Q and its density q replacing P and its density p, respectively. For a discrete P', this translates to

$$\mathcal{D}^2_{P,Q}\left(\frac{1}{n}\sum_{i=1}^{n}\delta(x_i)\right) = \frac{1}{n^2}\sum_{i,j=1}^{n}\frac{q(x_i)}{p(x_i)}\frac{q(x_j)}{p(x_j)}k_Q(x_i, x_j) = \frac{1}{n^2}\sum_{i,j=1}^{n}k_{P,Q}(x_i, x_j), \tag{1.14}$$

where

$$k_{P,Q}(x_i, x_j) \coloneqq \frac{q(x_i)}{p(x_i)}\frac{q(x_j)}{p(x_j)}k_Q(x_i, x_j).$$

Similar to (1.11), expression (1.14) averages the elements of a Gram matrix, and this fact can be exploited to provide an efficient implementation of the greedy search described in Section 1.3.

When $k_Q(x, y)$ is based on IMQ with $\Gamma = I$, Theorem 2 in Fisher and Oates (2024) asserts convergence control under regularity conditions on Q: $\mathcal{D}^2_{P,Q}(P'_m) \to 0$ for a sequence of distributions P'_m implies $P'_m \Rightarrow P$ as $m \to \infty$.

While removing the need to calculate the gradient of log-target, the approach by Fisher and Oates (2024) replaces it with the requirement to choose a suitable auxiliary distribution Q. The authors provide several examples, but stop short of recommending a particular option:

- where the target distribution is the posterior in a Bayesian inference problem, the prior distribution can serve as Q,
- where p can be differentiated, the Laplace approximation of P could be used,
- where samples from P are available, it can be approximated by either the Gaussian mixture model (GMM) or a kernel density estimator (KDE).

The choice of an auxiliary distribution Q is non-trivial, and Fisher and Oates (2024) warn of possible failure of convergence control when the density of Q has either a substantially heavier or a substantially lighter tail than p. Two other situations are identified by the authors as detrimental to their approach: high dimension of the domain $\mathcal{X}$ and well separated high-probability regions.

In their paper, Fisher and Oates (2024) apply gradient-free KSD for importance resampling and variational inference problems. In what follows, we adopt their approach for the thinning problem.

Chapter 2

Methodology and Results

2.1 Gradient-free Stein thinning

We modify Algorithm 1 to use the gradient-free Stein kernel $k_{P,Q}(x, y)$ in place of $k_P(x, y)$, where

$$\begin{aligned} k_{P,Q}(x,y) =& \frac{q(x)}{p(x)}\frac{q(y)}{p(y)}\times \\ & \left[-4\frac{\beta(\beta-1)\|\Gamma^{-1}(x-y)\|^2}{(c^2+\|\Gamma^{-1/2}(x-y)\|^2)^{-\beta+2}}\right. \\ & -2\beta\frac{\operatorname{trace}(\Gamma^{-1})+\langle\Gamma^{-1}(x-y),\nabla_x\log q(x)-\nabla_y\log q(y)\rangle}{(c^2+\|\Gamma^{-1/2}(x-y)\|^2)^{-\beta+1}} \\ & \left.+\frac{\langle\nabla_x\log q(x),\nabla_y\log q(y)\rangle}{(c^2+\|\Gamma^{-1/2}(x-y)\|^2)^{-\beta}}\right]. \end{aligned} \tag{2.1}$$

The resulting procedure is shown in Algorithm 2. The implementation of Algorithm 2 is added directly to the Python library `stein-thinning` and is publicly available.

We proceed to compare the performance of the proposed algorithm against naïve thinning (retaining each i-th element of the sample) and the Stein thinning algorithm of Riabiz et al. (2022) for several target distributions. The outline of the evaluation procedure for each test case is as follows:

1. obtain a sample from the target distribution,
2. apply naïve thinning, Stein thinning and the proposed algorithm to get a thinned sample of a given cardinality,
3. evaluate the result of thinning using an impartial metric.

In order to assess how well the selected sample approximates the target distribution, we use the energy distance. Following Rizzo and Székely (2016), the squared energy distance is defined for two distributions P and Q as

$$D^2(P,Q) \coloneqq 2\mathbb{E}\|X-Y\| - \mathbb{E}\|X-X'\| - \mathbb{E}\|Y-Y'\|, \tag{2.2}$$

where $X, X' \sim P$ and $Y, Y' \sim Q$. The rationale for this expression comes from Theorem 2 of Szekely (2002), which connects $D^2(P,Q)$ with the difference of characteristic functions $\hat{p}$ and $\hat{q}$ of P and Q, respectively:

$$D^2(P,Q) = \frac{1}{C_d}\int_{\mathbb{R}^d}\frac{(\hat{p}-\hat{q})^2}{\|t\|^2}\,\mathrm{d}t,$$

Algorithm 2: Gradient-free Stein thinning.

Data: Sample $(x_i)_{i=1}^n$ from MCMC, desired cardinality $m \in \mathbb{N}$.
Result: Indices π of a sequence $(x_{\pi(j)})_{j=1}^m \subset \{x_i\}_{i=1}^n$ where $\pi(j) \in \{1, \ldots, n\}$.
for $j = 1, \ldots, m$ **do**

$$\pi(j) \in \underset{i=1,\ldots,n}{\arg\min} \frac{k_{P,Q}(x_i, x_i)}{2} + \sum_{j'=1}^{j-1} k_{P,Q}(x_{\pi(j'}, x_i)$$

end

where C_d is a constant specific to the dimension d. In the univariate case, this coincides with the Cramer distance (Cramér (1928))

$$D^2(P, Q) = 2 \int_{-\infty}^{\infty} (F(t) - G(t))^2 dt,$$

where F and G are cumulative density functions of P and Q. For samples $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$ from X and Y, respectively, the statistic corresponding to (2.2) is given by

$$\mathcal{E}_{n,m}(P, Q) \coloneqq \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \|x_i - y_j\| - \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \|x_i - x_j\| - \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \|y_i - y_j\|.$$

We use the implementation of the energy distance from the `dcor` Python library by Ramos-Carreño and Torrecilla (2023).

The advantages of choosing the energy distance as our quality metric are its relative ease of computation and its objectivity, since the energy distance is not directly optimised by the thinning algorithm.

We use synthetic data in order to have control over the ground truth in our experiments.

2.2 Bivariate Gaussian mixture

The purpose of the first test case is to confirm the expected behaviour of the proposed algorithm under the favourable conditions of an i.i.d. sample. Here we take the target distribution to be the bivariate Gaussian mixture with means

$$\mu_1 = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \qquad \mu_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

covariance matrices

$$\Sigma_1 = \begin{pmatrix} 0.5 & 0.25 \\ 0.25 & 1 \end{pmatrix}, \qquad \Sigma_2 = \begin{pmatrix} 2 & -0.8\sqrt{3} \\ -0.8\sqrt{3} & 1.5 \end{pmatrix}$$

and weights

$$w = \begin{pmatrix} 0.3 \\ 0.7 \end{pmatrix},$$

and obtain 1000 samples by directly drawing from the target. A scatter plot of the sample as well as the contour plot of the probability density are shown in Figure 2.1.

Naïve thinning. We select elements of the sample with uniformly-spaced indices. Since the samples are i.i.d., naïve thinning is equivalent to drawing a smaller sample from the bivariate Gaussian mixture directly.

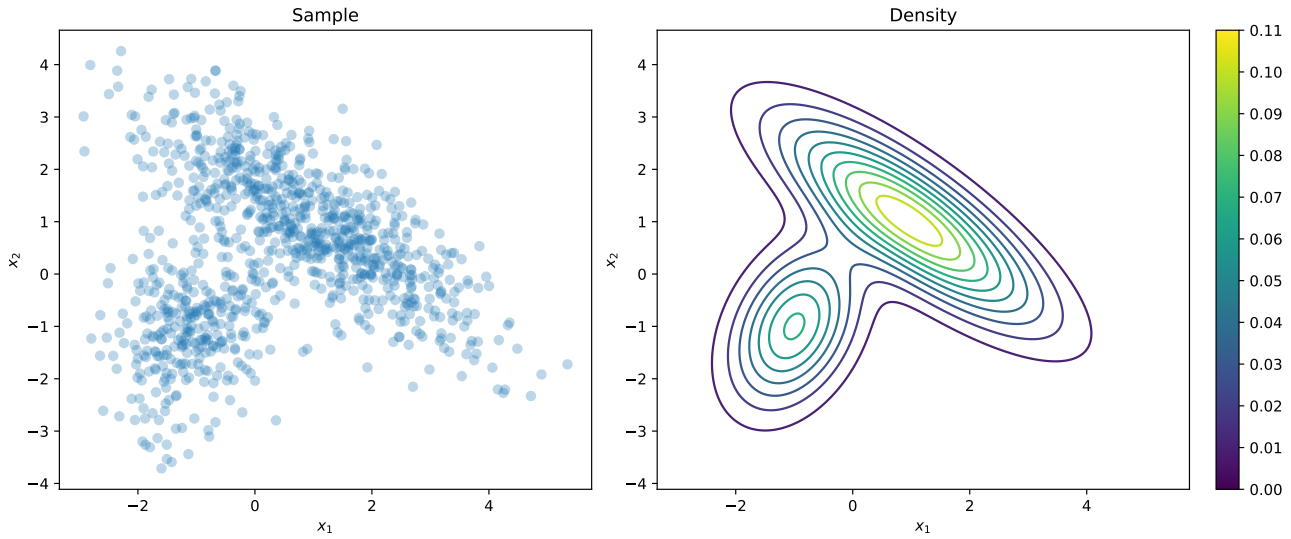

Figure 2.1: A sample from the bivariate Gaussian mixture with two components and its probability density function.

Stein thinning. The gradients of the log-target required by the Stein thinning algorithm can be obtained analytically in this case (see Appendix A.2).

Gradient-free Stein thinning. This approach requires us to select an auxiliary distribution. We consider several options:

- a bivariate Gaussian with the mean and covariance matrix matching the sample mean and covariance of the sample,
- a Laplace approximation of the target distribution,
- a KDE approximation of the target constructed from the sample.

For the KDE estimator, we choose the Gaussian kernel with bandwidth selected according to Silverman's rule.

The results are shown in Figure 2.2. The failure of the Laplace approximation is striking, so we investigate it further. The other methods produce visually sensible results. One can notice some clustering of points selected by naïve thinning, whereas the Stein thinning and the gradient-free approach using the Gaussian auxiliary distribution and the KDE appear to spread points more evenly.

To see why the Laplace approximation fails in this case, note the multiplier $q(x)/p(x)$ in (2.1). Since the thinning algorithm seeks to minimise the sum, it will tend to prefer elements with low values of $q(x)/p(x)$. In Figure 2.3, we plot the values $\log q(x) - \log p(x) = \log(q(x)/p(x))$ for points in the sample. It is clear that the points in the lower left corner of the plot become very attractive for the algorithm to select due to their miniscule values of the multiplier $q(x)/p(x)$. The low ratio $q(x)/p(x)$ for those points is in turn due to the fact that the Laplace approximation picks the larger of the two modes in the mixture, so it cannot serve as a good auxiliary distribution for the samples coming from the other component. By contrast, the simple Gaussian distribution achieves more uniform values of the multiplier.

Finally, Figure 2.4 compares the energy distance achieved by thinned samples using the approaches above. We see that the gradient-free approach using the KDE is competitive with Stein thinning for small cardinalities, but loses out as the required sample size grows. The gradient-free approach using the simple Gaussian approach proves inferior to the naïve approach.

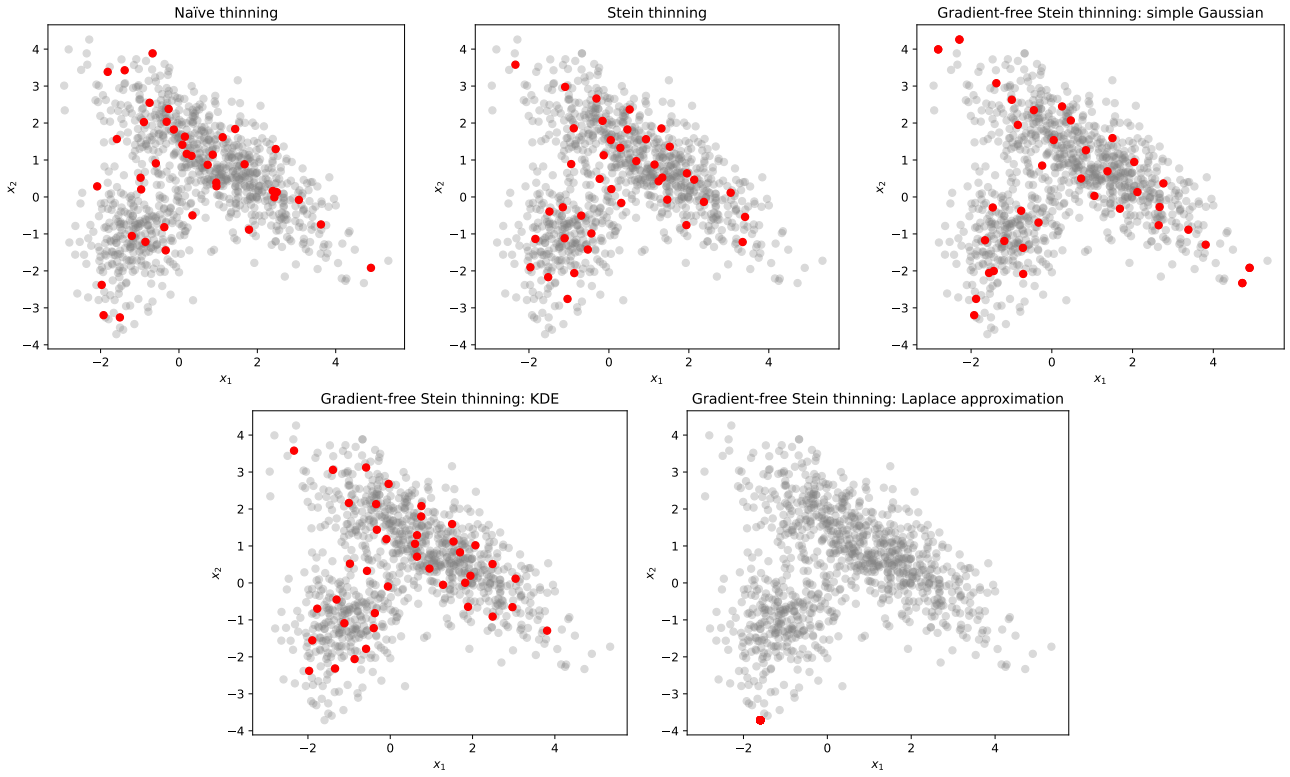

Figure 2.2: Thinning results for the bivariate Gaussian mixture.

We conclude that even in such a favourable setup, the performance of the gradient-free algorithm depends crucially on the choice of the auxiliary distribution.

2.3 Lotka-Volterra inverse problem

A more challenging case is presented by the inverse problem of parameter inference for the Lotka-Volterra model.

Cite sources on inverse Bayesian problems.

The Lotka-Volterra model describes the evolution of an idealised ecosystem with two species: predator and prey. The predator population grows while prey is abundant, and the prey population shrinks when there are too many predators. Denoting the size of prey population by u_1, and the predator population by u_2, the model postulates the following dynamic:

$$\begin{aligned} \frac{\mathrm{d}u_1}{u_1} &= (\theta_1 - \theta_2 u_2)\,\mathrm{d}t, \\ \frac{\mathrm{d}u_2}{u_2} &= (-\theta_3 + \theta_4 u_1)\,\mathrm{d}t, \end{aligned} \tag{2.3}$$

with $\theta_1, \ldots, \theta_4 > 0$. The resulting behaviour of the system is thus driven by the four parameters $\theta_1, \ldots, \theta_4$. It is convenient to denote the solution to this system as $u(t;\theta) = (u_1(t;\theta), u_2(t;\theta))^T$, where θ is the vector of parameter values $(\theta_1, \ldots, \theta_4)^T$.

If a realisation of $u(t;\theta)$ is observed with noise for an interval $t \in [0, T]$ with θ unknown, one may wish to infer the values of the parameters that best describe the observed behaviour. In a practical setting, this corresponds to formulating a parametric model of a natural phenomenon and inferring the parameters of the model from the measurements of the phenomenon. We emulate this situation by fixing the true model, generating a realisation perturbed by noise and then attempting to recover the parameters of the true model from the realisation.

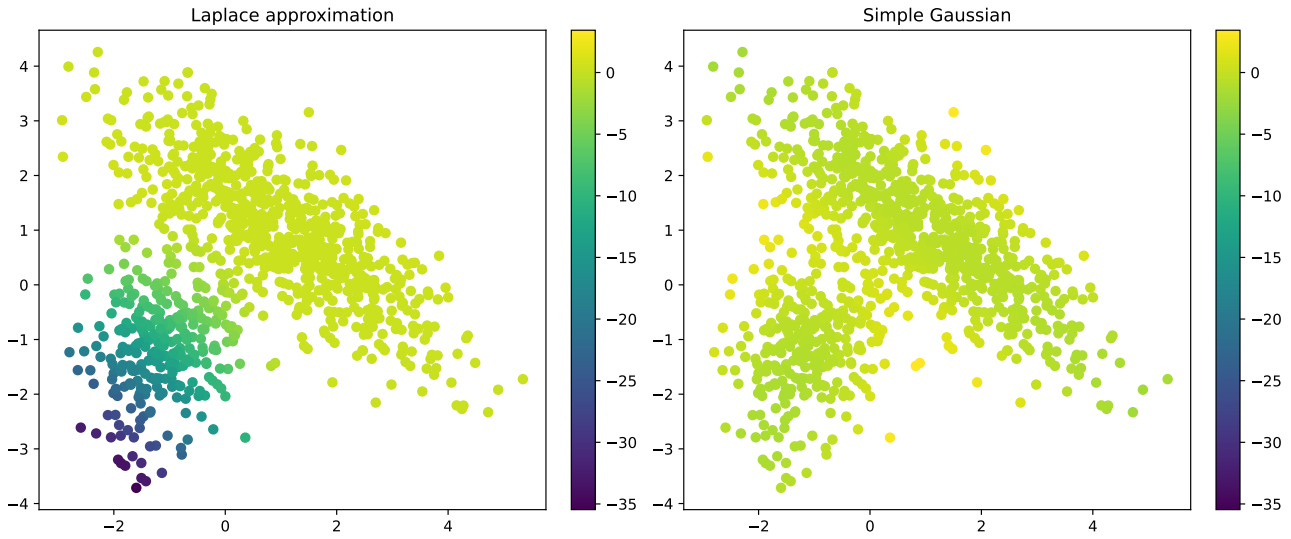

Figure 2.3: The values of $\log q(x) - \log p(x)$ for the points in the Gaussian mixture sample under the Laplace approximation and the simple Gaussian using the sample mean and covariance.

We take $u(t;\theta^*)$ to be generated for $t \in [0, 25]$ by the model (2.3) with parameters $\theta^* = (0.67, 1.33, 1, 1)^T$ and initial value $u(0;\theta^*) = (1, 1)^T$. The interval $[0, 25]$ is discretised into $N = 2400$ points, so that $t_i = 25 \cdot i/(N-1)$ for $i \in \{0, N-1\}$. Bivariate i.i.d. Gaussian noise $\varepsilon(t) \sim \mathcal{N}\left(0, \mathrm{diag}(0.2^2, 0.2^2)\right)$ is then added to all data points to emulate the measurement error, and we take the resulting time series $y(t) = u(t;\theta^*) + \varepsilon(t)$ as our observed data. Figure 2.5 displays the values $y(t)$. For comparability of results, the parameter values above match those used by Riabiz et al. (2022).

Posterior inference is then performed by means of a random-walk Metropolis-Hastings algorithm. Assuming independent observations, we take the likelihood to be

$$\mathcal{L}(\theta) = \prod_{i=1}^{N} \phi_i(u(t_i;\theta)), \tag{2.4}$$

where

$$\phi_i(u(t_i;\theta)) \propto \exp\left(-\frac{1}{2}(y(t_i) - u(t_i;\theta))^T C^{-1}(y(t_i) - u(t_i;\theta))\right) \tag{2.5}$$

with $C = \mathrm{diag}(0.2^2, 0.2^2)$. Since $\theta_i > 0$, we put independent log-normal priors on each θ_i, so

$$\pi(\theta) \propto \exp\left(-\frac{1}{2}(\log\theta)^T(\log\theta)\right), \tag{2.6}$$

where the logarithm in $\log\theta$ is applied component-wise. By the Bayes theorem, the posterior is then

$$p(\theta) \propto \mathcal{L}(\theta)\pi(\theta). \tag{2.7}$$

We follow Riabiz et al. (2022) in selecting the starting values for the chains (Table 2.1).

Since the parameters of the Lotka-Volterra model are non-negative, we run inference in the log-space. The trace plots from running 500,000 iterations of the random-walk Metropolis-Hastings algorithm for each chain are shown in Figure B.1 in the Appendix. Figure 2.6 confirms that all chains reach the high-probability region, albeit after considerable burn-in, and Figure 2.7 demonstrates the sample obtained from the first chain. Having obtained samples from the posterior distribution, we proceed to thin them.

Naïve thinning. We select elements of the sample with uniformly spaced indices. The results are shown in Figure B.2 in the Appendix.

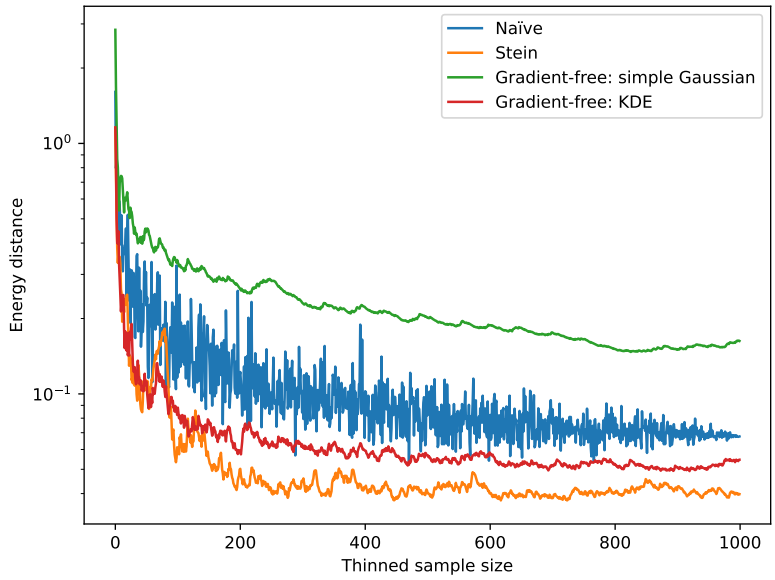

Figure 2.4: Energy distance comparison between thinning approaches for the bivariate Gaussian mixture sample.

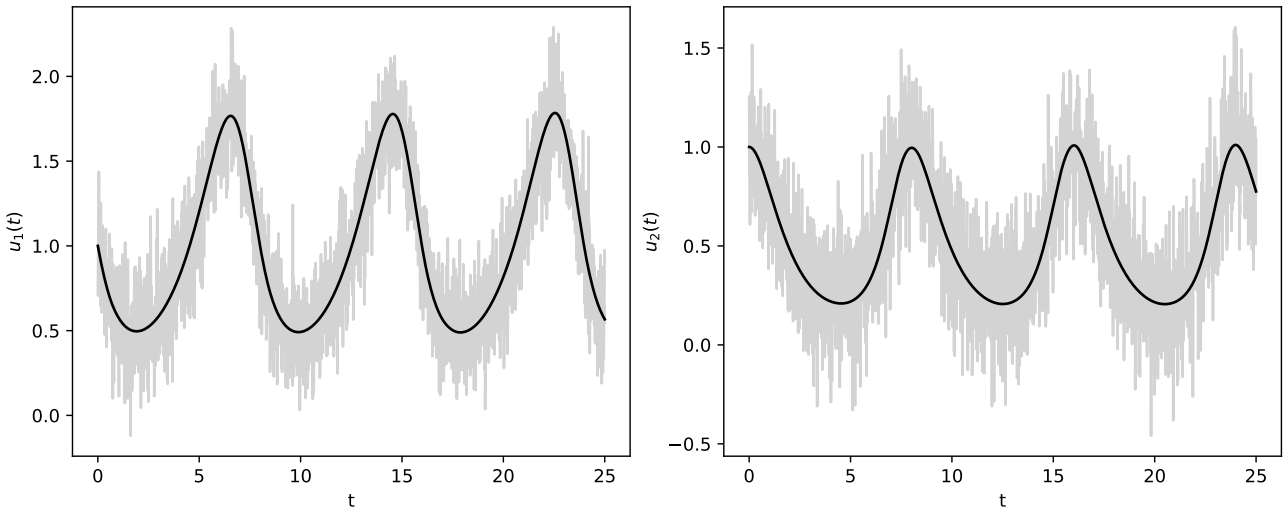

Figure 2.5: Solution to the Lotka-Volterra ODEs (black) with added Gaussian noise (grey).

Chain	θ_1	θ_2	θ_3	θ_4
1	0.55	1	0.8	0.8
2	1.5	1	0.8	0.8
3	1.3	1.33	0.5	0.8
4	0.55	3	3.	0.8
5	0.55	1	1.5	1.5

Table 2.1: Starting values for the random-walk Metropolis-Hastings algorithm in the Lotka-Volterra inference problem.

Stein thinning. In order to use this approach, we need the gradients of the log-posterior. From (2.7), we have

$$\nabla_\theta \log p(\theta) = \nabla_\theta \log \mathcal{L}(\theta) + \nabla_\theta \pi(\theta).$$

Equation (2.4) yields

$$\frac{\partial}{\partial \theta_s} \log \mathcal{L}(\theta) = \sum_{i=1}^{N} \frac{\partial}{\partial \theta_s} \log \phi_i(u(t_i;\theta)) = \sum_{i=1}^{N} \sum_{r=1}^{q} \frac{\partial}{\partial u_r} (\log \phi_i) \frac{\partial u_r}{\partial \theta_s},$$

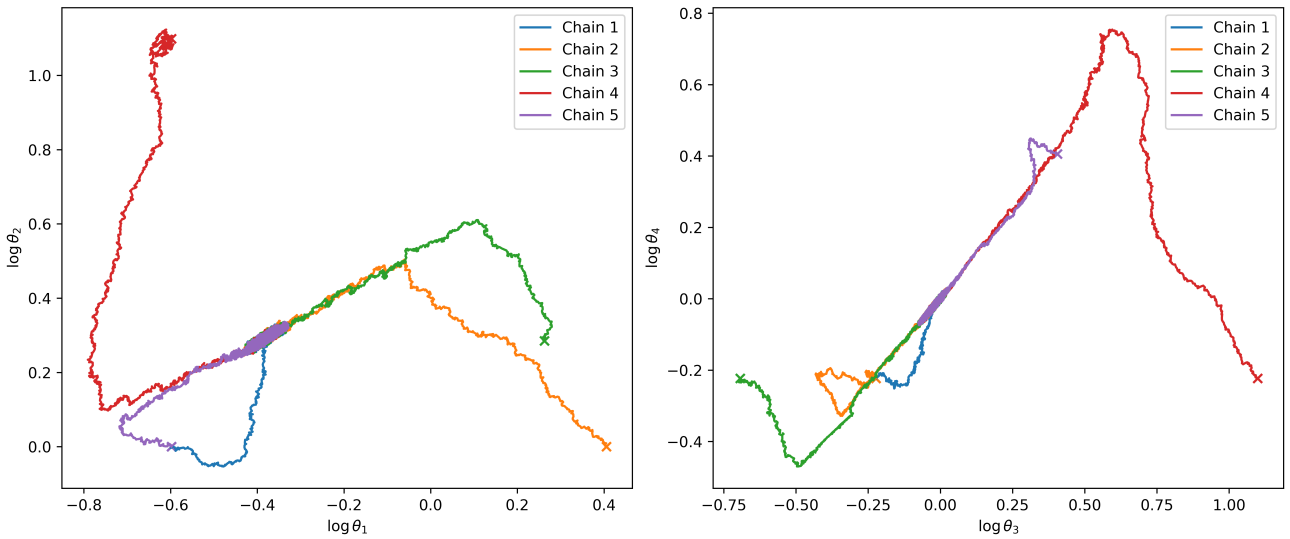

Figure 2.6: Chain paths from the random-walk Metropolis-Hastings algorithm for the Lotka-Volterra inference problem.

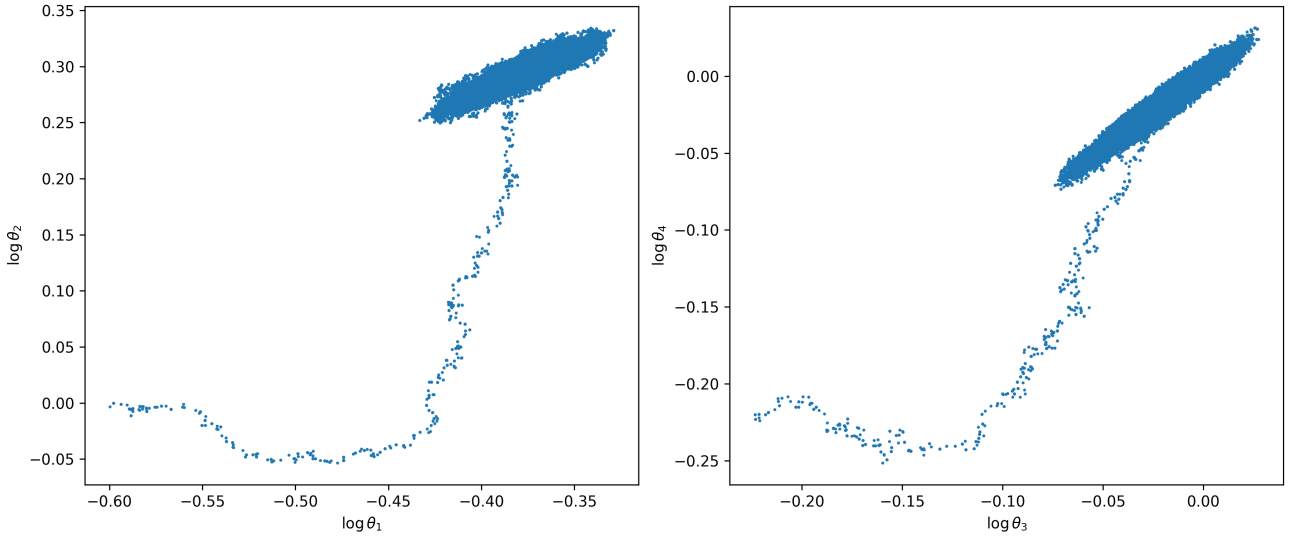

Figure 2.7: Sample of size 500,000 from the first chain from an MCMC run for the Lotka-Volterra inverse problem.

where $q = 2$ is the dimension of the vector $u(t)$ and $s \in \{1, \ldots, 4\}$. This can be written in the matrix form as

$$(\nabla_\theta \log \mathcal{L})(\theta) = \sum_{i=1}^{N} (S(t_i))^T (\nabla_u \log \phi_i)(u(t_i; \theta)),$$

where the elements of the sensitivities matrix are given by

$$S_{r,s} := \frac{\partial u_r}{\partial \theta_s},$$

where $r \in \{1, 2\}$ and $s \in \{1, \ldots, 4\}$. The gradient

$$(\nabla_u \log \phi_i)(u(t_i; \theta)) = C^{-1}(y(t_i) - u(t_i; \theta))$$

is derived directly from (2.5). In order to obtain the sensitivities $\partial u_r / \partial \theta_s$, we augment the system (2.3) with additional forward sensitivity equations and solve it numerically. The derivation of the sensitivity equations can be found in Appendix A.3. Alternatively, the sensitivities can be obtained by numerical differentiation, for example using the Python library `JAX`[1]. Since the gradient is calculated indepen-

[1] `https://jax.readthedocs.io`

dently for each element of the sample, these calculations can easily be parallelised at postprocessing stage[2].

The gradient of the log-prior is obtained immediately from (2.6):

$$\nabla_\theta \log \pi(\theta) = -\frac{\log \theta}{\theta},$$

where the logarithm and division are component-wise.

The results of applying Stein thinning are shown in Figure B.3 in the Appendix. We compare the performance of Stein thinning in linear and log space of parameters in Figure 2.8:

- there is no discernible difference between the linear and logarithmic parameterisations,
- Stein thinning performs consistently for all chains, whereas naïve thinning fails for chain 4, which spent most of its time away from the high-probability region,
- Stein thinning improves on naïve thinning for sample sizes below 20 and above 2000, and is otherwise similar.

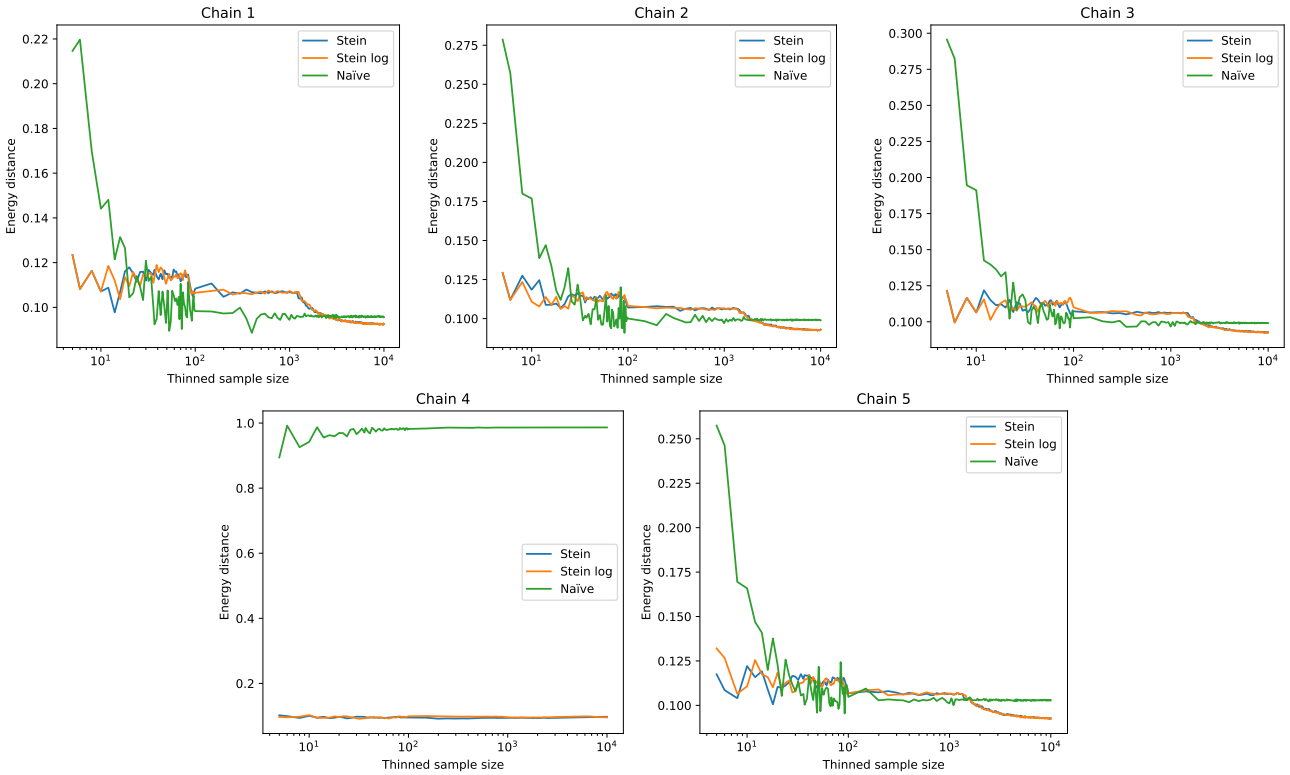

Figure 2.8: Energy distance comparison between Stein thinning and naïve thinning for MCMC chains in the Lotka-Volterra inverse problem.

Gradient-free Stein thinning. The choice of KDE as the auxiliary distribution, which performed best for the Gaussian mixture in Section 2.2, becomes computationally infeasible here due to the size of the sample. Indeed, if KDE is built based on the entire sample, and is evaluated for each point of the sample, the resulting computational cost grows as $O(n^2)$, where n is the sample size. One could try to build a KDE based on a subsample of size $m < n$, however if the computational budget is fixed (i.e. $mn = \text{const}$) then larger n implies lower m and thus inferior approximation of the target density.

[2]Parallelisation can be implemented with minimal effort using the Python `Dask` library (`https://docs.dask.org`) and run either locally or on Amazon Web Services (`https://aws.amazon.com`). See `https://github.com/aglebov/gradient-free-mcmc-postprocessing/blob/main/code/examples/Dask_AWS.ipynb` for a demonstration.

As in the case of the Gaussian mixture, the Laplace approximation fails to produce a thinned sample here due to ratio $q(x)/p(x)$ vanishing away from the mode. Instead, we attempt to use the Student's t distribution, which exhibits polynomially-decaying tails. To select the parameters of the t distribution, we perform numerical optimisation in the spirit of Section 7.13.3 of Ruppert and Matteson (2015). As an alternative strategy, we also evaluate the following configuration: set the location of the Student's t distribution to be the mode of the sample (i.e. the point with the highest target probability), set the scale parameter to the sample covariance multiplied by 3, and the degrees of freedom parameter to 4. The values were hand-picked by trial and error to achieve smaller resulting energy distance.

Figure 2.9 shows that our hand-picked choice of the auxiliary distribution out-performs the gradient-based approach in terms of energy distance, while the optimised choice fares worse, in fact for the first chain it comes out even worse than naïve thinning. While it is surprising that the gradient-free approach being an approximation of the gradient-based algorithm ends up beating it on energy distance, we note that the energy distance looks at the similarity of distributions from a different angle compared to the KSD, so the two are not necessarily maximised by the same configuration. Figure 2.10 confirms that the gradient-based algorithm consistently achieves the lowest values of the kernel Stein discrepancy among the variations considered.

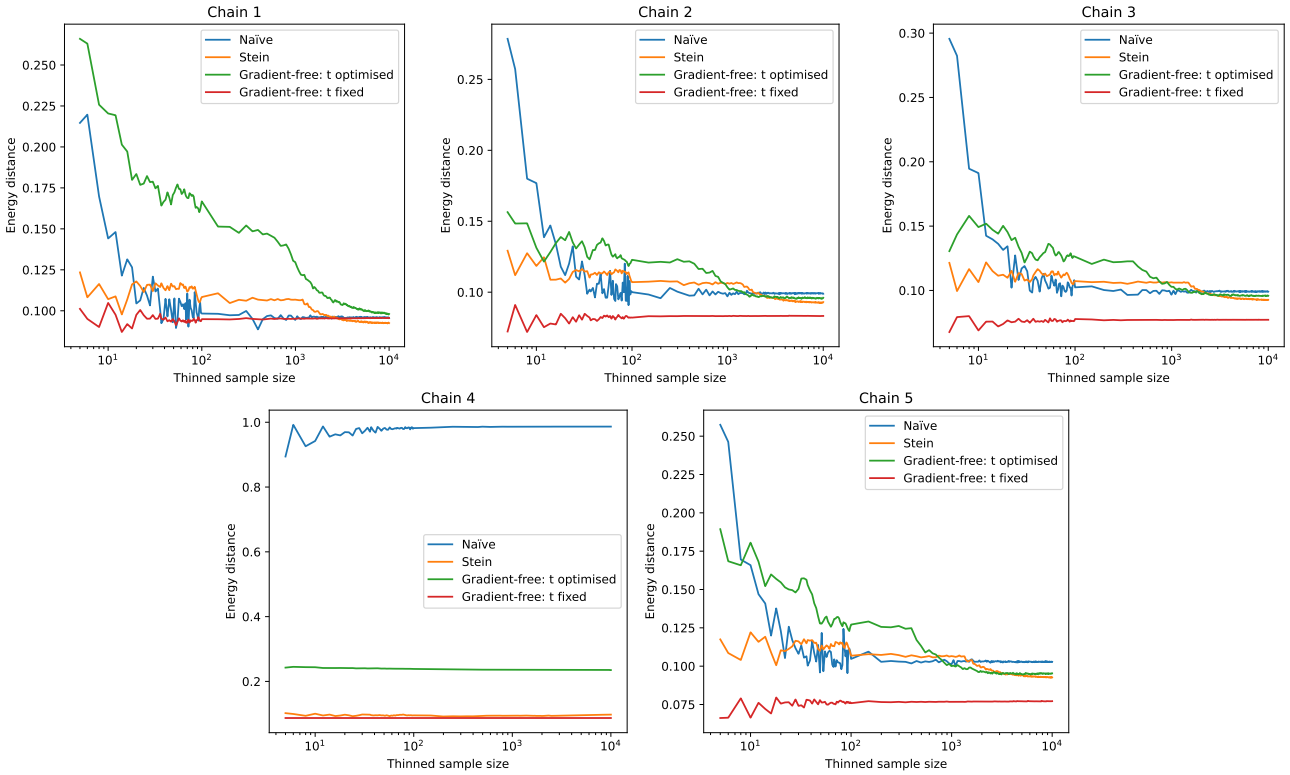

Figure 2.9: Energy distance comparison between gradient-free and standard Stein thinning for the Lotka-Volterra inverse problem.

Echoing the observations from the preceding section, we conclude that a good choice of an auxiliary distribution is key for successfully applying the gradient-free algorithm. It becomes clear, however, that a good choice might be difficult to come up with in practice, as it relies on the MCMC sample to guide the search. In a practical setting, however, one might not have access to high-quality samples, as was the case for the calcium signalling model considered by Riabiz et al. (2022).

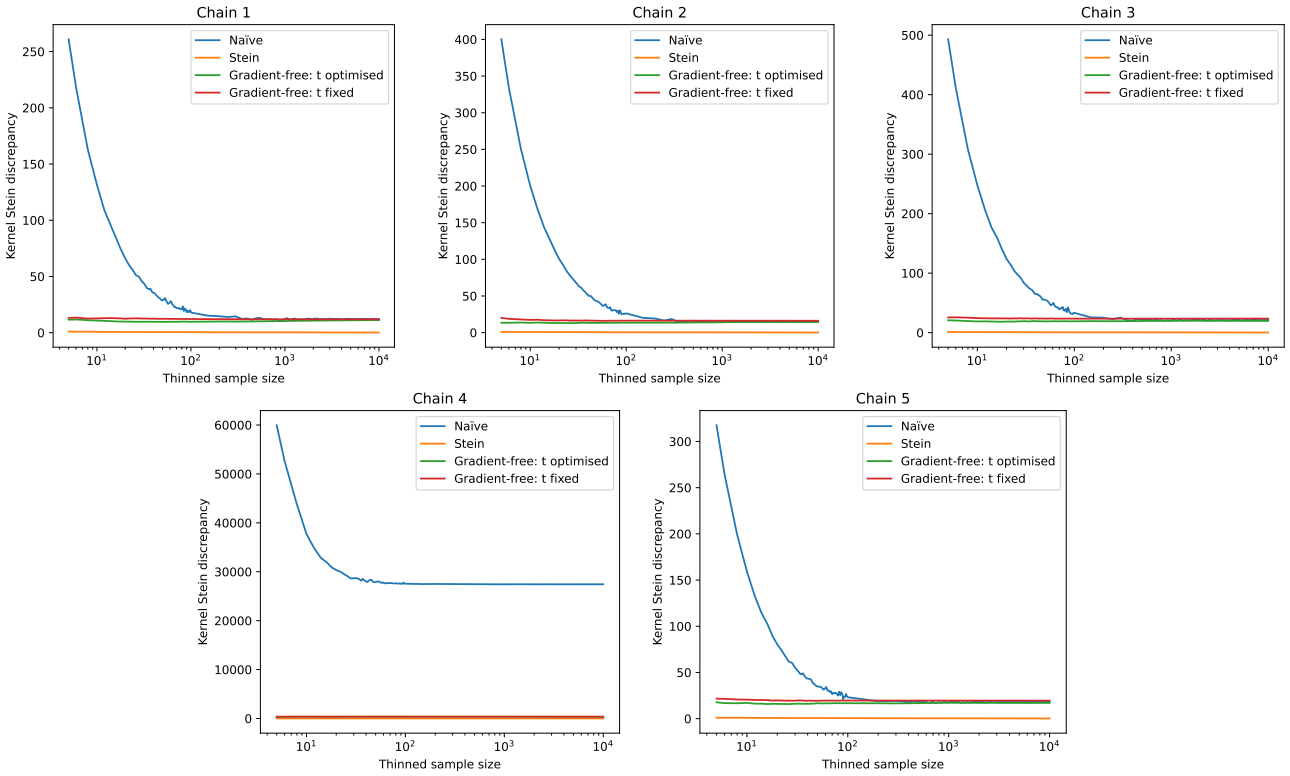

Figure 2.10: Kernel Stein discrepancy comparison between gradient-free and standard Stein thinning for the Lotka-Volterra inverse problem.

Chapter 3

Conclusions and Further Work

This dissertation makes two contributions. We implement the gradient-free Stein thinning algorithm in the Python library `stein-thinning` and evaluate the performance of the proposed algorithm for i.i.d. samples from a Gaussian mixture and MCMC samples from the Lotka-Volterra inverse problem.

We conclude that the gradient-free approach is feasible and performs similarly to the Stein thinning algorithm of Riabiz et al. (2022) for small thinned sample sizes, however its performance depends crucially on the choice of the auxiliary distribution. Even in the highly favourable setting of i.i.d. samples from a Gaussian mixture for instance, choosing the auxiliary distribution based on the Laplace approximation fails to produce a thinned sample. Bespoke treatment is required for more complex problems, as demonstrated by the case of parameter inference in the Lotka-Volterra model. In deciding whether to use the new algorithm as opposed to the gradient-based approach, the effort involved in selecting a good auxiliary distribution must be weighed against the computational cost of obtaining gradients.

Many interesting avenues for further research remain open.

Evaluate the choices of KDE kernels other than Gaussian for constructing the auxiliary distribution. Kernels with bounded support (uniform, saw-tooth, Epanechnikov) are expected to be of limited use, since they do not provide gradient estimates outside of their support, however fat-tailed kernels (e.g. exponential or Student's t) are an interesting option to consider.

Parallelise the computation of KDE. As noted in Section 2.3, the computational cost of the KDE approach rises as $O(n^2)$, where n is the size of the sample. Evaluations of the density, however, can be performed in parallel, extending the range of sample sizes that can be handled within a given time budget.

Perform thinning in a lower-dimensional space. Samples from the MCMC runs for the Lotka-Volterra problem (see Figure 2.7, for example) indicate correlation between the parameters. Principal component analysis (PCA) could be used to identify a lower-dimensional representation of the samples, and thinning could be done in the transformed space.

Develop a principled procedure for selecting the parameters of a Student's t auxiliary distribution. In Section 2.3, we hand-picked values for the parameters by calculating the energy distance for the resulting sample repeatedly. The approach is dissatisfying in its arbitrariness. Instead, one could, for example, try choosing the degrees of freedom parameter based on the tail behaviour of the target distribution and then optimising the location and scale of the t distribution.

Investigate the feature of Figure 2.8, whereby the energy distance for the gradient-based algorithm remains approximately constant up to sample sizes around 1000, and then starts reducing further. Riabiz et al. (2022) plot the corresponding dependency for sample sizes up to 200, so we cannot be sure if the same effect was observed in their experiments.

Compare the performance of the approaches in terms of estimating the true parameters of the Lotka-Volterra model. While the energy distance offers a richer metric of dissimilarity between distributions, parameter inference is often the goal in practice, so a simple comparison between first moment estimates could be useful.

Building on the preceding idea, run an experiment initialising a large number of chains with starting values drawn from some distribution, perform sampling and thinning, and compare the mean-squared errors of the resulting parameter estimates.

Repeat the experiments in Section 2.3 with more advanced MCMC algorithms. For example, the MALA sampler considered in Riabiz et al. (2022) or the HMC sampler implemented in the `Stan` library could be used.

Check how running a gradient-free MCMC sampling algorithm (such the random-walk Metropolis-Hastings) followed by Stein thinning of the sample compares to running a gradient-based sampling algorithm (e.g. HMC). Given the sequential nature of MCMC, the costs of calculating the gradients add up in the later case, but can be parallelised in the former. It would be interesting to see which approach produces higher-quality samples under a fixed time budget.

Demonstrate the bias-correction result of Theorem 3 in Riabiz et al. (2022) for the gradient-free algorithm.

Bibliography

S. P. Brooks and A. Gelman. General Methods for Monitoring Convergence of Iterative Simulations. *Journal of Computational and Graphical Statistics*, 7(4):434–455, December 1998.

W. Y. Chen, L. Mackey, J. Gorham, F.-X. Briol, and C. J. Oates. Stein Points. In *Proceedings of the 35th International Conference on Machine Learning*, pages 843–852. PLMR, 2018.

W. Y. Chen, A. Barp, F.-X. Briol, J. Gorham, M. Girolami, L. Mackey, and Chris. J. Oates. Stein Point Markov Chain Monte Carlo. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1011–1021, Long Beach, California, 2019. PLMR.

H. Cramér. On the composition of elementary errors: First paper: Mathematical deductions. *Scandinavian Actuarial Journal*, 1928(1):13–74, January 1928.

M. A. Fisher and C. J. Oates. Gradient-Free Kernel Stein Discrepancy. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 23855–23885, May 2024.

A. Gelman and D. B. Rubin. Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 7(4):457–472, November 1992.

A. Gelman, G. O. Roberts, and W. R. Gilks. Efficient Metropolis Jumping Rules. In *Bayesian Statistics*, volume 5, pages 599–608. Oxford University Press, Oxford, May 1996.

A. Gelman, W. R. Gilks, and G. O. Roberts. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120, February 1997.

J. Gorham and L. Mackey. Measuring Sample Quality with Stein's Method. In *Advances in Neural Information Processing Systems*, volume 28, pages 226–234. MIT Press, 2015.

J. Gorham and L. Mackey. Measuring Sample Quality with Kernels. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1292–1301, Sydney, Australia, 2017. PLMR.

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

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, June 1953.

A. Müller. Integral Probability Metrics and Their Generating Classes of Functions. *Advances in Applied Probability*, 29(2):429–443, June 1997.

C. Ramos-Carreño and J. L. Torrecilla. Dcor: Distance correlation and energy statistics in Python. *SoftwareX*, 22:101326, May 2023.

M. Riabiz, W. Y. Chen, J. Cockayne, P. Swietach, S. A. Niederer, L. Mackey, and Chris. J. Oates. Optimal Thinning of MCMC Output. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(4):1059–1081, September 2022.

M. L. Rizzo and G. J. Székely. Energy Distance. *WIREs Computational Statistics*, 8(1):27–38, January 2016.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer, New York, 2nd edition, 2004.

G. O. Roberts and J. S. Rosenthal. Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, 16(4):351–367, November 2001.

D. Ruppert and D. S. Matteson. *Statistics and Data Analysis for Financial Engineering: With R Examples*. Springer, New York, 2nd edition, 2015.

G. Szekely. E-statistics: The Energy of Statistical Samples. Technical report, Bowling Green State University, 2002.

D. Vats and C. Knudson. Revisiting the Gelman–Rubin Diagnostic. *Statistical Science*, 36(4):518–529, November 2021.

A. Vehtari, A. Gelman, D. Simpson, B. Carpenter, and P.-C. Bürkner. Rank-Normalization, Folding, and Localization: An Improved R for Assessing Convergence of MCMC (with Discussion). *Bayesian Analysis*, 16(2):667–718, June 2021.

Appendix A

Derivations

A.1 Stein kernel based on inverse multiquadratic kernel

When $k(x, y)$ is the inverse multiquadratic kernel (1.12) with $x, y \in \mathbb{R}^d$, the corresponding Stein kernel $k_P(x, y)$ is obtained from (1.10). We start by evaluating the terms in (1.10). First, for $r \in \{1, \dots, d\}$,

$$\begin{aligned}\frac{\partial^2}{\partial x_r \, \partial y_r} k(x,y) = &-4\beta(\beta-1)\left(c^2 + \sum_{i=1}^{d}\sum_{j=1}^{d}(x_i - y_i)\Gamma^{-1}_{ij}(x_j - y_j)\right)^{\beta-2}\left(\sum_{j=1}^{d}\Gamma^{-1}_{rj}(x_j - y_j)\right)^2 \\ &-2\beta\left(c^2 + \sum_{i=1}^{d}\sum_{j=1}^{d}(x_i - y_i)\Gamma^{-1}_{ij}(x_j - y_j)\right)^{\beta-1}\Gamma^{-1}_{rr}\end{aligned} \tag{A.1}$$

which gives us

$$\begin{aligned}(\nabla_x \cdot \nabla_y)k(x,y) = &-4\beta(\beta-1)\left(c^2 + \|\Gamma^{-1/2}(x-y)\|^2\right)^{\beta-2}\|\Gamma^{-1}(x-y)\|^2 \\ &-2\beta\left(c^2 + \|\Gamma^{-1/2}(x-y)\|^2\right)^{\beta-1}\operatorname{trace}(\Gamma^{-1})\end{aligned} \tag{A.2}$$

Now,

$$\begin{aligned}\frac{\partial}{\partial x_r}k(x,y) &= \beta\left(c^2 + \sum_{i=1}^{d}\sum_{j=1}^{d}(x_i - y_i)\Gamma^{-1}_{ij}(x_j - y_j)\right)^{\beta-1}\sum_{j=1}^{d}(\Gamma^{-1} + \Gamma^{-T})_{rj}(x_j - y_j) \\ &= 2\beta\left(c^2 + \sum_{i=1}^{d}\sum_{j=1}^{d}(x_i - y_i)\Gamma^{-1}_{ij}(x_j - y_j)\right)^{\beta-1}\sum_{j=1}^{d}\Gamma^{-1}_{rj}(x_j - y_j),\end{aligned} \tag{A.3}$$

where we used that Γ is a symmetric matrix. The gradient is then

$$\nabla_x k(x,y) = 2\beta\left(c^2 + \|\Gamma^{-1/2}(x-y)\|^2\right)^{\beta-1}\Gamma^{-1}(x-y). \tag{A.4}$$

and similarly

$$\nabla_y k(x,y) = -2\beta\left(c^2 + \|\Gamma^{-1/2}(x-y)\|^2\right)^{\beta-1}\Gamma^{-1}(x-y). \tag{A.5}$$

Substituting (A.4), (A.5) and (A.2) into (1.10), we obtain

$$\begin{aligned} k_P(x,y) =& -4\beta(\beta-1)\left(c^2+\|\Gamma^{-1/2}(x-y)\|^2\right)^{\beta-2}\|\Gamma^{-1}(x-y)\|^2 \\ & -2\beta\left(c^2+\|\Gamma^{-1/2}(x-y)\|^2\right)^{\beta-1}\operatorname{trace}(\Gamma^{-1}) \\ & +2\beta\left(c^2+\|\Gamma^{-1/2}(x-y)\|^2\right)^{\beta-1}\langle\Gamma^{-1}(x-y),\nabla_y\log p(y)\rangle \\ & -2\beta\left(c^2+\|\Gamma^{-1/2}(x-y)\|^2\right)^{\beta-1}\langle\Gamma^{-1}(x-y),\nabla_x\log p(x)\rangle \\ & +\left(c^2+\|\Gamma^{-1/2}(x-y)\|^2\right)^{\beta}\langle\nabla_x\log p(x),\nabla_y\log p(y)\rangle \\ =& -4\beta(\beta-1)D^{\beta-2}\|\Gamma^{-1}(x-y)\|^2 \\ & -2\beta D^{\beta-1}(\operatorname{trace}(\Gamma^{-1})+\langle\Gamma^{-1}(x-y),\nabla_x\log p(x)-\nabla_y\log p(y)\rangle) \\ & +D^{\beta}\langle\nabla_x\log p(x),\nabla_y\log p(y)\rangle, \end{aligned} \tag{A.6}$$

where in the last equation we have denoted $D = c^2 + \|\Gamma^{-1/2}(x-y)\|^2$.

A.2 Gradient of the Gaussian mixture distribution

For multivariate normal distributions with density functions

$$f_i(x) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)\right),$$

where $x \in \mathbb{R}^d$, the mixture density with k components is given by

$$f(x) = \sum_{i=1}^{k} w_i f_i(x),$$

thus the gradient of its log-density is obtained as

$$\nabla_x \log f(x) = \frac{\sum_{i=1}^k w_i \nabla_x f_i(x)}{\sum_{i=1}^k w_i f_i(x)} = -\frac{\sum_{i=1}^k w_i f_i(x)\Sigma_i^{-1}(x-\mu_i)}{\sum_{i=1}^k w_i f_i(x)}.$$

A.3 Forward sensitivity equations for the Lotka-Volterra model

Given a system of ODEs of the form:

$$\frac{\mathrm{d}u_r}{\mathrm{d}t} = F_r(t, u_1, \ldots, u_q; \theta_1, \ldots, \theta_d), \qquad r = 1, \ldots, q,$$

where q is the dimension of the state space (i.e. the number of observed variables) and d is the dimension of the parameter space, the sensitivities can be found by solving forward sensitivity equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial u_r}{\partial \theta_s}\right) = \frac{\partial}{\partial \theta_s}\frac{\mathrm{d}u_r}{\mathrm{d}t} = \frac{\partial F_r}{\partial \theta_s} + \sum_{l=1}^{q}\frac{\partial F_r}{\partial u_l}\frac{\partial u_l}{\partial \theta_s}.$$

Double-check the notation above.

For the Lotka-Volterra model (2.3), this yields eight additional equations:

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial u_1}{\partial \theta_1}\right) &= u_1 + (\theta_1 - \theta_2 u_2)\frac{\partial u_1}{\partial \theta_1} - \theta_2 u_1 \frac{\partial u_2}{\partial \theta_1},\\
\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial u_1}{\partial \theta_2}\right) &= -u_1 u_2 + (\theta_1 - \theta_2 u_2)\frac{\partial u_1}{\partial \theta_2} - \theta_2 u_1 \frac{\partial u_2}{\partial \theta_2},\\
\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial u_1}{\partial \theta_3}\right) &= (\theta_1 - \theta_2 u_2)\frac{\partial u_1}{\partial \theta_3} - \theta_2 u_1 \frac{\partial u_2}{\partial \theta_3},\\
\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial u_1}{\partial \theta_4}\right) &= (\theta_1 - \theta_2 u_2)\frac{\partial u_1}{\partial \theta_4} - \theta_2 u_1 \frac{\partial u_2}{\partial \theta_4},\\
\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial u_2}{\partial \theta_1}\right) &= \theta_4 u_2 \frac{\partial u_1}{\partial \theta_1} + (\theta_4 u_1 - \theta_3)\frac{\partial u_2}{\partial \theta_1},\\
\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial u_2}{\partial \theta_2}\right) &= \theta_4 u_2 \frac{\partial u_1}{\partial \theta_2} + (\theta_4 u_1 - \theta_3)\frac{\partial u_2}{\partial \theta_2},\\
\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial u_2}{\partial \theta_3}\right) &= -u_2 + \theta_4 u_2 \frac{\partial u_1}{\partial \theta_3} + (\theta_4 u_1 - \theta_3)\frac{\partial u_2}{\partial \theta_3},\\
\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial u_2}{\partial \theta_4}\right) &= u_1 u_2 + \theta_4 u_2 \frac{\partial u_1}{\partial \theta_4} + (\theta_4 u_1 - \theta_3)\frac{\partial u_2}{\partial \theta_4}.
\end{aligned}
$$

Appendix B

Additional figures

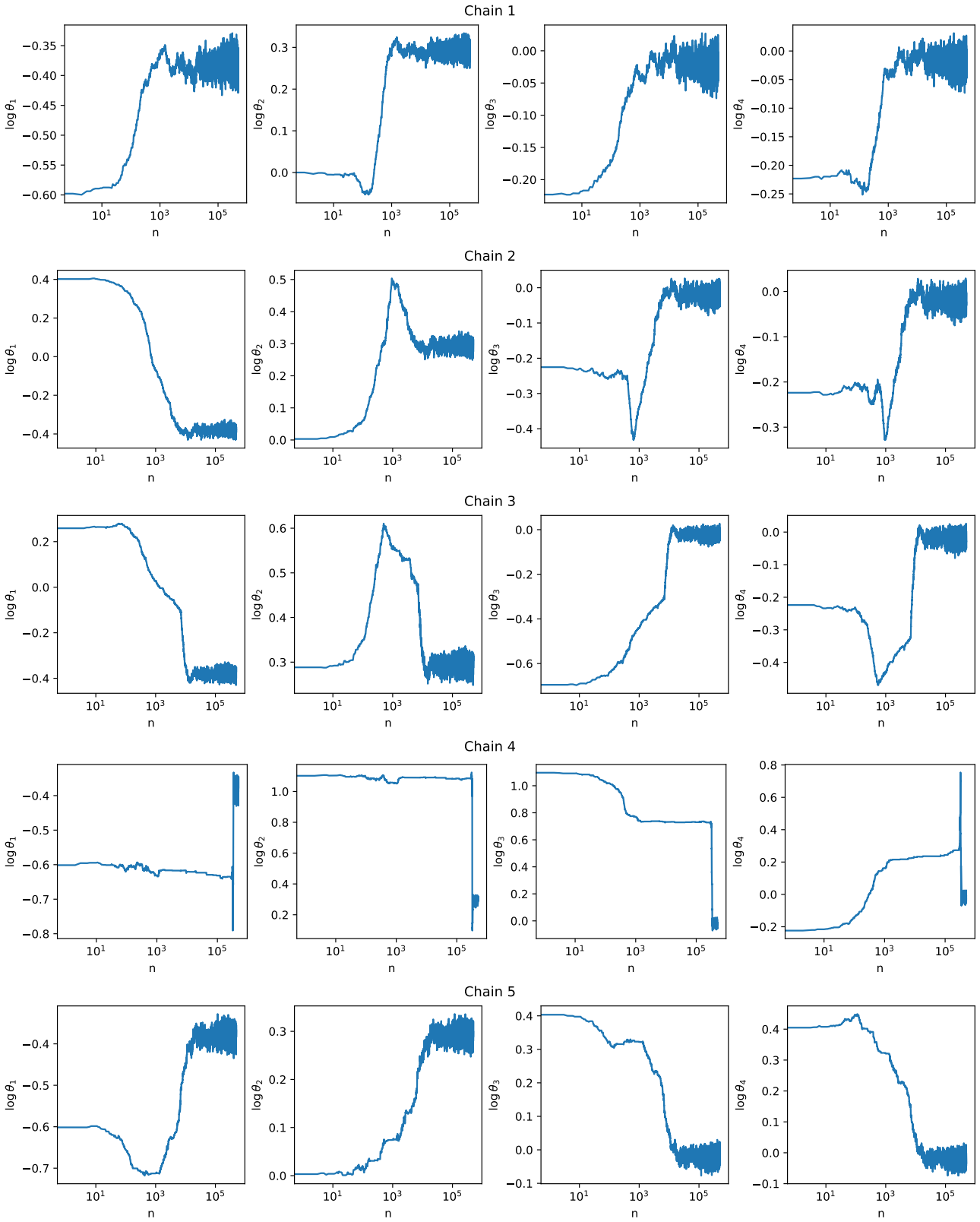

Figure B.1: Trace plots from the random-walk Metropolis-Hastings algorithm for the Lotka-Volterra inference problem.

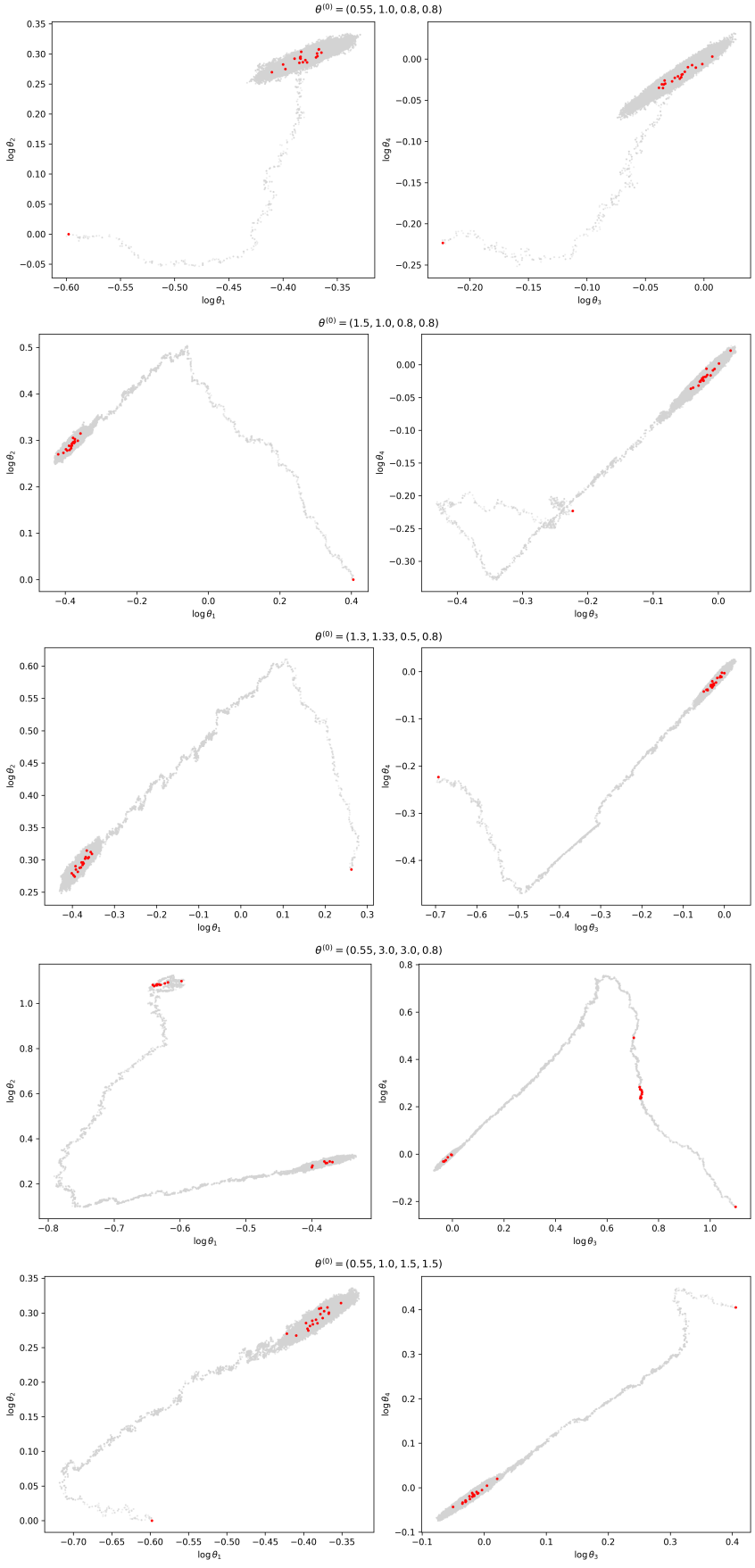

Figure B.2: Results of naïve thinning of the sample from the Lotka-Volterra model.

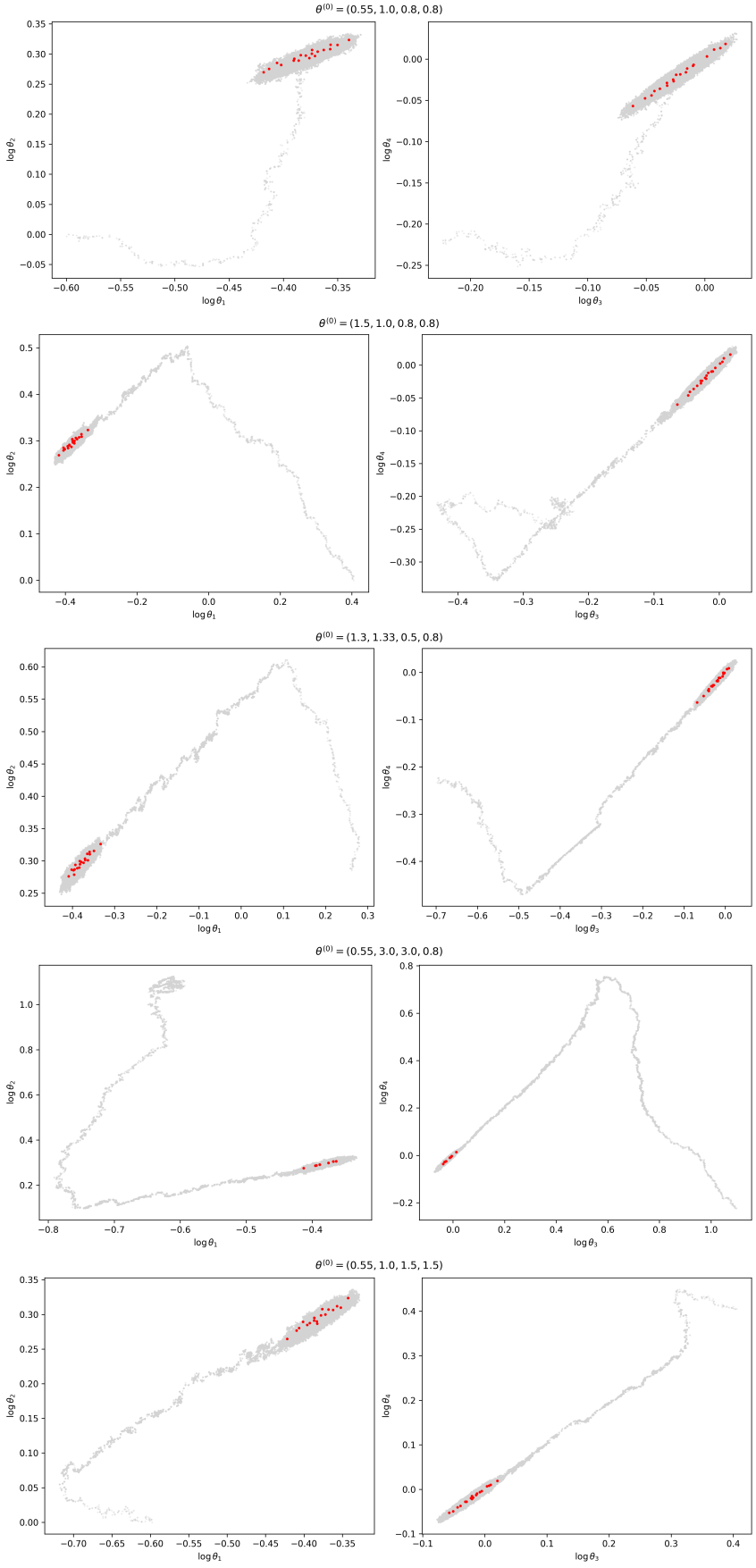

Figure B.3: The first 20 points selected by the Stein thinning algorithm for each MCMC chain in the Lotka-Volterra inverse problem.