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

Contents

Introduction

The Python code accompanying this report can be found at:

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

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`.

Chapter 1

Background

1.1 Markov chain Monte Carlo

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 numerical 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.

> Describe alternatives: the inverse method, accept-reject and importance sampling

An MCMC algorithm proceeds by sequentially constructing a chain of samples[1] $x_1, x_2, \ldots$, where each sample is drawn from a transition distribution 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

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

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: $x_{n+1} = x_n$.

---

[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.

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 Z is the distribution of the step taken by the algorithm, e.g. a 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 Z determines the average size of the jump that the algorithm can make in one iteration and thus the speed of traversal of the target domain.

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

Gibbs algorithm. Suppose x is a d-dimensional vector and the components $x^{(1)}, x^{(2)}, \ldots, x^{(d)}$ 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.

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:

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 induces autocorrelation between the samples, which leads to increased variance of derived estimators.

7. The large number of samples resulting from an MCMC algorithm needs 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.

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, or to assess the quality of a sample retrospectively. 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 scaling of the step distribution in a 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.

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 distribution resulting from the full sample. 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 Optimal thinning

Given a Markov chain $(X_i)_{i \in \mathbb{N}}$ and its realisation $(x_i)_{i=1}^n$ 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 $\pi(j) \in \{1, \ldots, n\}$ with $j \in \{1, \ldots, m\}$, 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 approximation given by the full set, and thus to the target distribution. The criterion of proximity is based on the kernel Stein discrepancy, itself a special case of the integral probability metric.

1.3.1 Kernel Stein discrepancy

The integral probability metric between two distributions P and P' is defined as

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

where $\mathcal{X}$ is a measurable space on which both P and P' are defined and $\mathcal{F}$ is a set of test functions. 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,

- 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

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

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

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

under mild conditions on $\mathcal{L}$ and u

Check the conditions.

. In the specific case of an overdamped Langevin diffusion

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

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

$$\mathcal{A}_P g = \langle g, \nabla \log p\rangle + \langle \nabla, g\rangle = \langle p^{-1}\nabla, pg\rangle, \tag{1.5}$$

and rewrite (1.4) as

$$\mathcal{D}_{P,\mathcal{G}}(P') = \sup_{g \in \mathcal{G}} \left| \int_{\mathcal{X}} \mathcal{A}_P g\,\mathrm{d}P' \right| \tag{1.6}$$

for a suitably chosen set $\mathcal{G}$. Note that p enters (1.5) via $\nabla \log p$, so the knowledge of its normalising constant is not required to evaluate the operator.

To remove the optimisation step in the calculation of (1.6), 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

$$\mathcal{G} := \left\{ \mathrm{g} : \mathbb{R}^d \to \mathbb{R}^d \,\middle|\, \sum_{i=1}^{d} \|g_i\|_{\mathcal{H}(k)}^2 \leq 1 \right\} \tag{1.7}$$

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

$$\mathcal{D}_P^2(P') = \mathbb{E}_{P' \times P'}[k_P(X, \tilde{X})] = \iint_{\mathcal{X}} k_P(x, y)\,\mathrm{d}p'(x)\,\mathrm{d}p'(y), \tag{1.8}$$

where $X, \tilde{X} \sim P'$, p' is the density of P', and $k_P(x, y)$ is given by

$$\begin{aligned} k_P(x, y) :=&(\nabla_x \cdot \nabla_y)k(x, y) \\ &+ \langle \nabla_x k(x, y), \nabla_y \log p(y)\rangle + \langle \nabla_y k(x, y), \nabla_x \log p(x)\rangle \\ &+ k(x, y)\langle \nabla_x \log p(x), \nabla_y \log p(y)\rangle. \end{aligned} \tag{1.9}$$

Here ∇_x and ∇_y are gradients w.r.t. x and y, respectively, and the operator $\nabla_x \cdot \nabla_y$ is defined as:

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

Note that we have dropped $\mathcal{G}$ in the subscript of (1.8) as the choice of $\mathcal{G}$ does not depend on P or P'. If P' is a discrete distribution, as in (1.1), the squared discrepancy (1.8) becomes

$$\mathcal{D}_P^2\left(\frac{1}{n}\sum_{i=1}^{n} \delta(x_i)\right) = \frac{1}{n^2}\sum_{i,j=1}^{n} k_P(x_i, x_j). \tag{1.10}$$

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

$$k(x, y) = \left(c^2 + \|\Gamma^{-1/2}(x - y)\|\right)^{\beta} \tag{1.11}$$

with $\beta \in (-1, 0)$, Gorham and Mackey (2017) demonstrate for $\Gamma = I$ that $\mathcal{D}_P(P')$ provides convergence control:

- if $\mathcal{D}_P(P'_m) \to 0$, then P'_m converges in distribution to P (Theorem 8),

- if P'_m converges to P in Wasserstein distance, then $\mathcal{D}_P(P'_m) \to 0$ (Proposition 9)

for a sequence of distributions P'_m under suitable conditions. Theorem 4 by Chen et al. (2019) justifies the introduction of Γ in IMQ.

The constant c in (1.11) 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 $\beta = -\frac{1}{2}$.

The expression for $k_P(x, y)$ when $k(x, y)$ is taken to be the inverse multiquadratic kernel is derived in section A.1 and is coded directly in the Python library `stein-thinning`[2].

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.

1.3.2 Stein thinning

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.

Data: Sample $(x_i)_{i=1}^n$ from MCMC, Stein kernel k_P, 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**

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

end

Algorithm 1: Stein thinning.

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 $O(nm)$.

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.

1.3.3 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

---

[2]Available from `https://github.com/wilson-ye-chen/stein_thinning`.

gradient-free Stein operator $\mathcal{S}_{P,Q}$ defined for any differentiable function g as

$$\mathcal{S}_{P,Q}g := \frac{q}{p}(\langle g, \nabla \log q \rangle + \langle \nabla, g \rangle) = \frac{q}{p}\langle q^{-1}\nabla, qg \rangle. \tag{1.12}$$

This definition generalises expression (1.5) by introducing a proxy distribution Q with density q chosen such that its gradient is easily computable. When $q = p$, we recover the original Langevin Stein operator (1.5).

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 kernel 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 (1.6), taking $\mathcal{G}$ again to be the unit-ball (1.7). Futhermore, Proposition 7 in Fisher and Oates (2024) establishes that

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

where p' is the density of distribution P' and $k_Q(x,y)$ is given by (1.9) but with q replacing p. For a discrete P', this translates to

$$\mathcal{D}_{P,Q}^2 \left(\frac{1}{n} \sum_{i=1}^{n} \delta(x_i) \right) = \frac{1}{n^2} \sum_{i,j=1}^{n} \frac{q(x_i)}{p(x_i)} \frac{q(x_j)}{p(x_j)} k_Q(x_i, x_j). \tag{1.13}$$

Both expression (1.10) and (1.13) can be viewed as averaging the elements of matrices, and this fact is used to provide an efficient implementation of the greedy search described in Section 1.3.2.

When $k_Q(x,y)$ is based on IMQ with $\Gamma = I$, Theorem 2 in Fisher and Oates (2024) proves that $\mathcal{D}_{P,Q}^2(P'_m) \to 0$ implies that the sequence of distributions P'_m converges weakly to P as $m \to \infty$, thus ensuring convergence control.

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 proxy distribution Q. The authors provide several examples:

- 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 a proxy is non-trivial, and Fisher and Oates (2024) warn of possible failure of convergence control when the proxy density 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 this dissertation, we adopt their approach for the thinning problem.

Chapter 2

Methodology and Results

We modify Algorithm 1 to use the gradient-free Stein kernel $k_{P,Q}(x,y)$ in place of $k_P(x,y)$. With inverse multiquadratic kernel (1.11), this results in the expression

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

We aim 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. calculate the energy distance between the thinned sample and the target distribution and compare the values across the algorithms.

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

$$D^2(F,G) := 2\mathbb{E}\|X - Y\| - \mathbb{E}\|X - X'\| - \mathbb{E}\|Y - Y'\|,$$

where $X, X' \sim F$, $Y, Y' \sim G$, and $\|\cdot\|$ denotes the Euclidean norm. For samples $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$ from X and Y, respectively, the corresponding statistic is given by

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

Advantages: objective and easy to calculate.

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

2.1 Bivariate Gaussian mixture

The purpose of the first test case is to confirm the expected behaviour of the proposed algorithm under favourable conditions. 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.
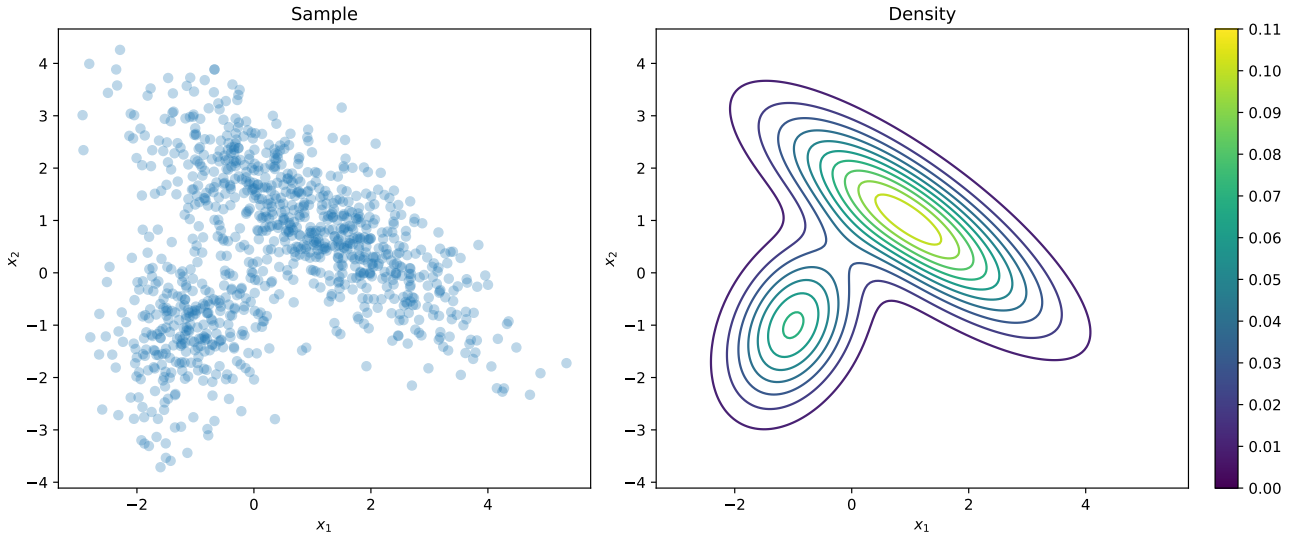

Figure 2.1: A sample from the bivariate Gaussian mixture with two components and its probability density function.

Naïve thinning. We select elements of the sample with equally-spaced indices.

Stein thinning. The gradients of the log-target required by the Stein thinning algorithm can be obtained analytically in this case. 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 the log-density is obtained as

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

Gradient-free Stein thinning. This approach requires us to select a proxy 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.

The results are shown in Figure 2.2. The failure of the Laplace proxy 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 proxy and the KDE proxy appear to spread points more evenly.

To see why the Laplace approximation fails in this case, note the multiplier $q(x)/p(x)$ in (1.13). 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 proxy for the samples coming from the other component. By contract, the simple Gaussian proxy 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 proxy 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 proxy proves inferior to the naïve approach.

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

2.2 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 when 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:

$$
\begin{aligned}
\frac{\mathrm{d}u_1}{u_1} &= (\quad \theta_1 - \theta_2 u_2)\,\mathrm{d}t \\
\frac{\mathrm{d}u_2}{u_2} &= (-\theta_3 + \theta_4 u_1)\,\mathrm{d}t
\end{aligned}
\tag{2.1}
$$

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

If a realisation of $\mathbf{u}(t; \boldsymbol{\theta})$ is observed for an interval $t \in [0, T]$ with $\boldsymbol{\theta}$ 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 for 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.

We take $\mathbf{u}(t;\boldsymbol{\theta}^*)$ to be generated for $t \in [0,25]$ by the model (2.1) with parameters $\boldsymbol{\theta}^* = (0.67, 1.33, 1, 1)^T$ and initial value $\mathbf{u}(0;\boldsymbol{\theta}^*) = (1,1)^T$. The interval $[0,25]$ is discretised into $N = 2400$ points, so that $t_i = 25 \cdot i/2400$ for $i \in \{0, 2400\}$. Bivariate i.i.d. Gaussian noise $\boldsymbol{\varepsilon}(t) \sim \mathcal{N}\left(\mathbf{0}, \mathrm{diag}(0.2^2, 0.2^2)\right)$ is then added to all data points to emulate the measurement error, and we take the resulting time series $\mathbf{y}(t) = \mathbf{u}(t;\boldsymbol{\theta}^*) + \boldsymbol{\varepsilon}(t)$ as our observed data. Figure 2.5 displays the values $\mathbf{y}(t)$. For comparability of results, the parameter values above match those used by Riabiz et al. (2022).

Inference is then performed by means of a random-walk Metropolis-Hastings algorithm. Note that the likelihood is not available in an explicit form in this case, but is instead determined by the deviation of the solution $\mathbf{u}(t;\boldsymbol{\theta})$ from $\mathbf{y}(t)$ for each $\boldsymbol{\theta}$. Specifically, we take

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{N} \phi_i(\mathbf{u}(t_i;\boldsymbol{\theta})), \tag{2.2}$$

where

$$\phi_i(\mathbf{u}(t_i;\boldsymbol{\theta})) \propto \exp\left(-\frac{1}{2}(\mathbf{y}(t_i) - \mathbf{u}(t_i;\boldsymbol{\theta}))^T C^{-1}(\mathbf{y}(t_i) - \mathbf{u}(t_i;\boldsymbol{\theta}))\right) \tag{2.3}$$

with $C = \mathrm{diag}(0.2^2, 0.2^2)$. We also choose the normal prior for $\boldsymbol{\theta}$:

$$\pi(\boldsymbol{\theta}) \propto \exp\left(-\frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta}\right). \tag{2.4}$$

By the Bayes theorem, the posterior is then

$$p(\boldsymbol{\theta}) \propto \mathcal{L}(\boldsymbol{\theta})\pi(\boldsymbol{\theta}). \tag{2.5}$$

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

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.

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. Figure 2.6 confirms that all chains reach the high-probability region, albeit after considerable burn-in. 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.

Stein thinning. In order to use this approach, we need the gradients of the log-posterior. From (2.5), we have

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

Equation (2.2) yields

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

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

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

where the elements of the Jacobian matrix are given by

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

The gradient

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

is derived directly from (2.3). In order to obtain the sensitivities $\frac{\partial u_r}{\partial \theta_s}$, we augment the system (2.1) with additional forward sensitivity equations and solve it numerically. The derivation of the sensitivity equations can be found in Appendix A.2. Alternatively, the sensitivities can be obtained by numerical differentiation, for example using the Python library JAX[1]. Since the gradient is calculated independently for each element of the sample, the calculations can easily be parallelised.

The gradient of the log-prior is obtained immediately from (2.4):

$$\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta}) = -\boldsymbol{\theta}.$$

The results of applying Stein thinning are shown in Figure B.3 in the Appendix.

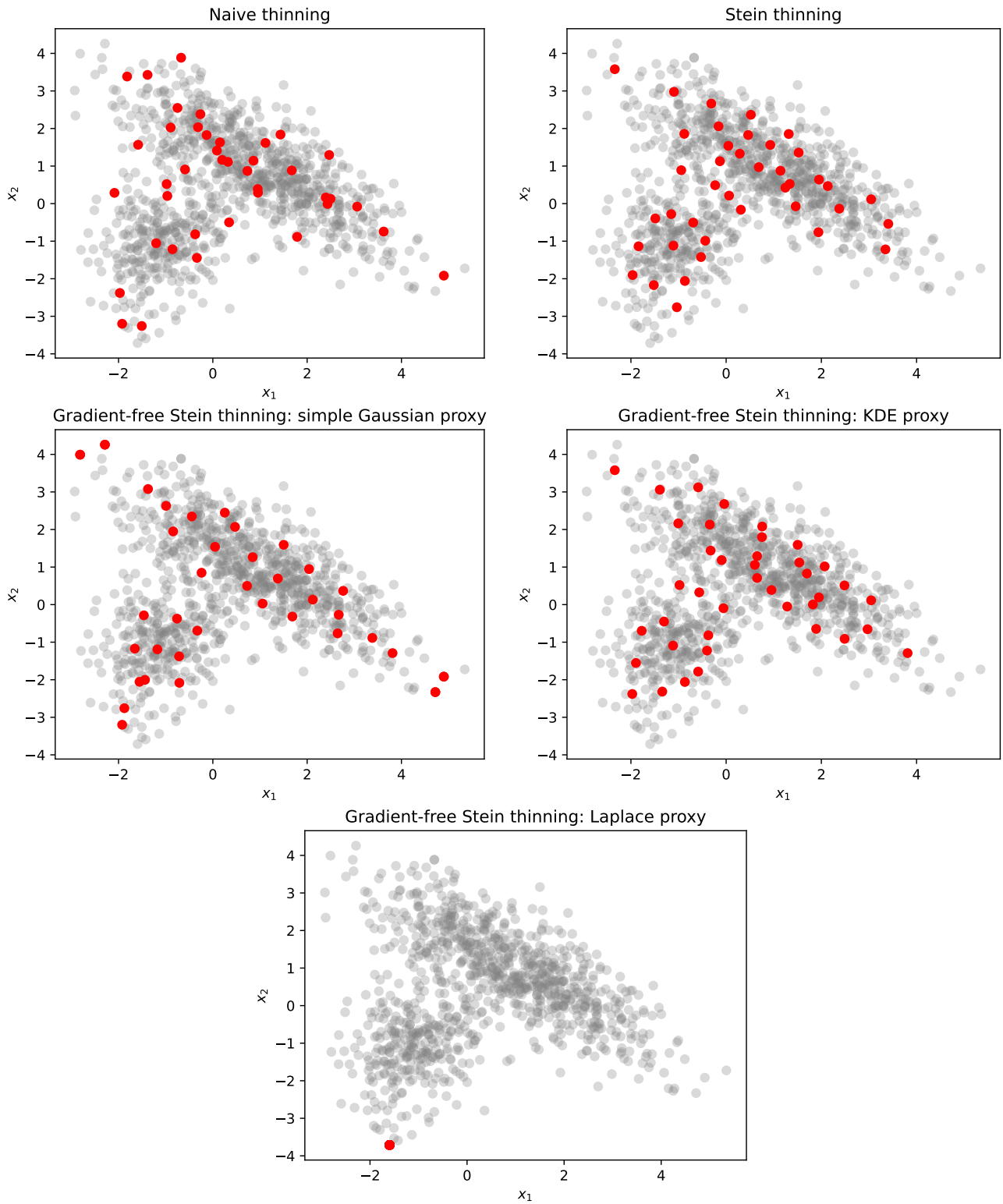[1]https://jax.readthedocs.io

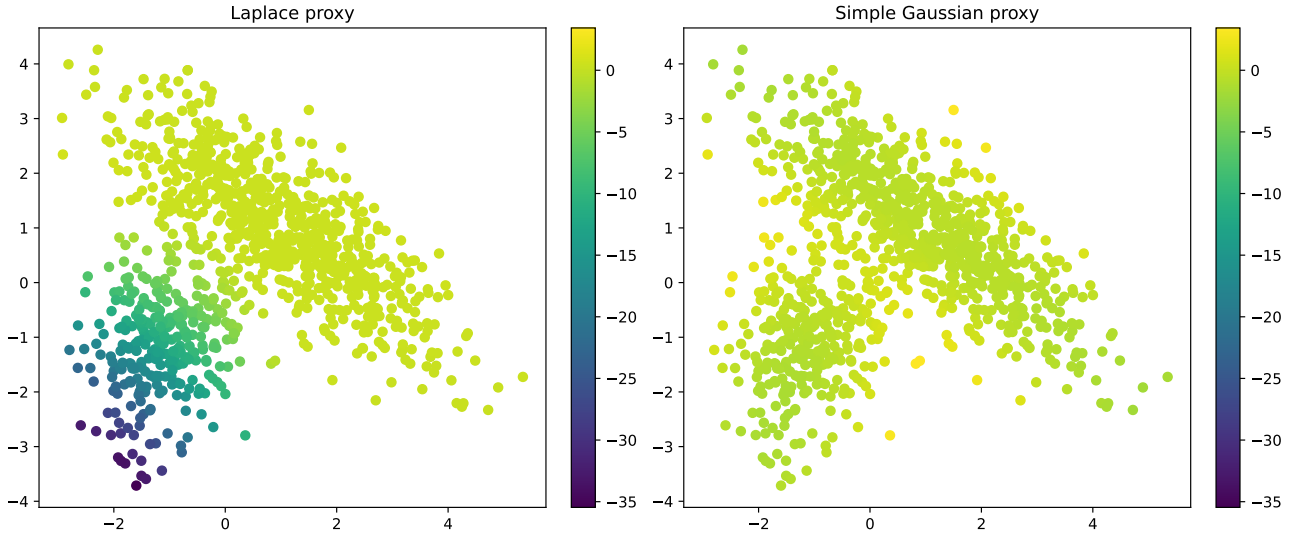Figure 2.2: Thinning results for the bivariate Gaussian mixture.

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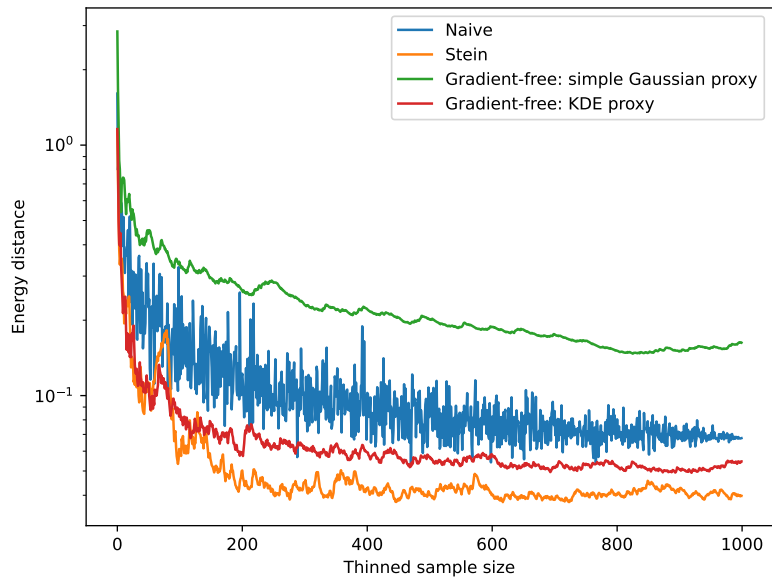

Figure 2.4: Energy distance comparison between thinning approaches for the bivariate Gaussian mixture sample.
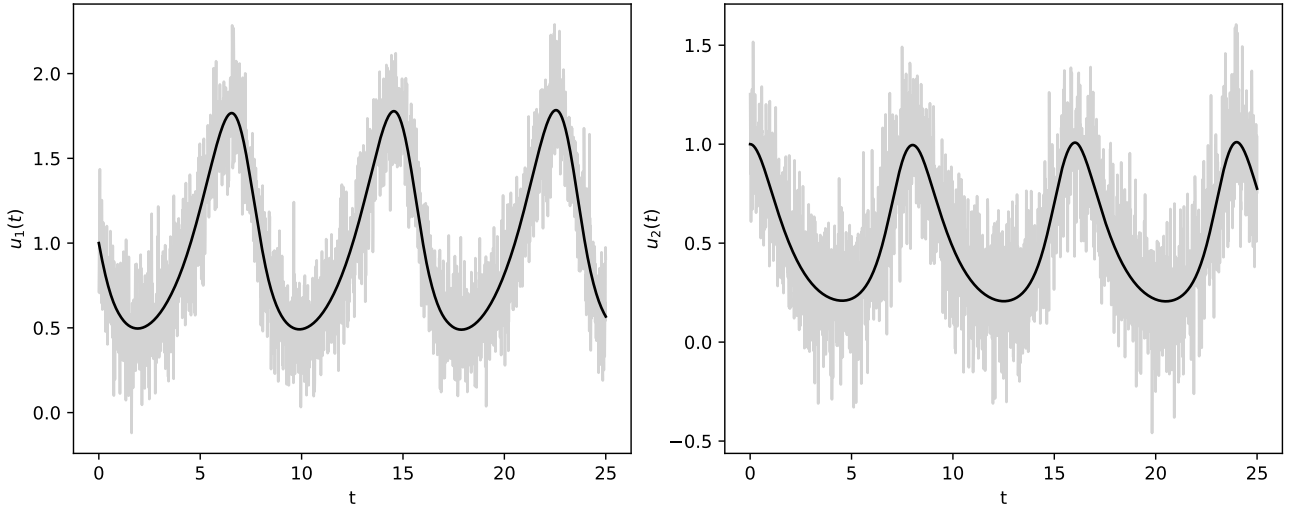
Figure 2.5: Solution to the Lotka-Volterra ODEs (black) with added Gaussian noise (grey).
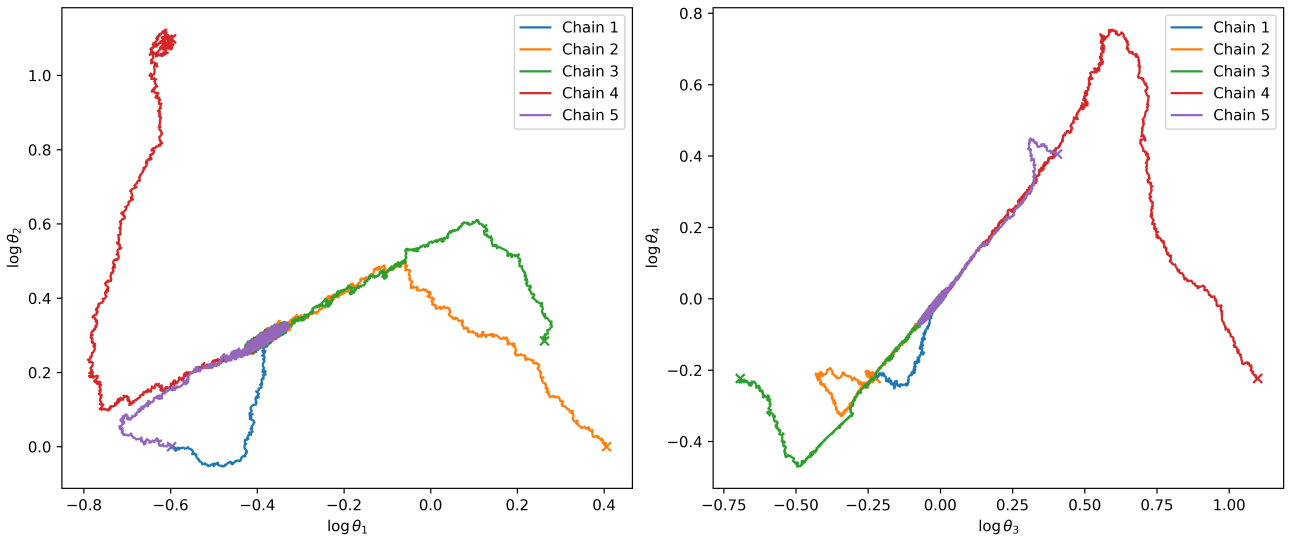

Figure 2.6: Chain paths from the random-walk Metropolis-Hastings algorithm for the Lotka-Volterra inference problem.

Chapter 3

Conclusions

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.

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.

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.

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

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

Given a kernel $k(x, y)$, the corresponding Stein kernel is given by (1.9). For the inverse multiquadratic kernel (1.11) we obtain

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

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

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

and similarly

$$
\nabla_y k(x, y) = -2\beta \left(c^2 + \|\Gamma^{-1/2}(x - y)\|^2 \right)^{\beta-1} \Gamma^{-1}(x - y).
\tag{A.3}
$$

Now

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

which gives us

$$
\begin{aligned}
(\nabla_x \cdot \nabla_y) k(x, y) = &-4\beta(\beta - 1) \left(c^2 + \|\Gamma^{-1/2}(x - y)\|^2 \right)^{\beta-2} \|\Gamma^{-1}(x - y)\|^2 \\
&- 2\beta \left(c^2 + \|\Gamma^{-1/2}(x - y)\|^2 \right)^{\beta-1} \text{trace}(\Gamma^{-1})
\end{aligned}
\tag{A.5}
$$

Substituting (A.2), (A.3) and (A.5) into (1.9), we obtain

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

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

A.2 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 (2.1), this yields eight additional equations:

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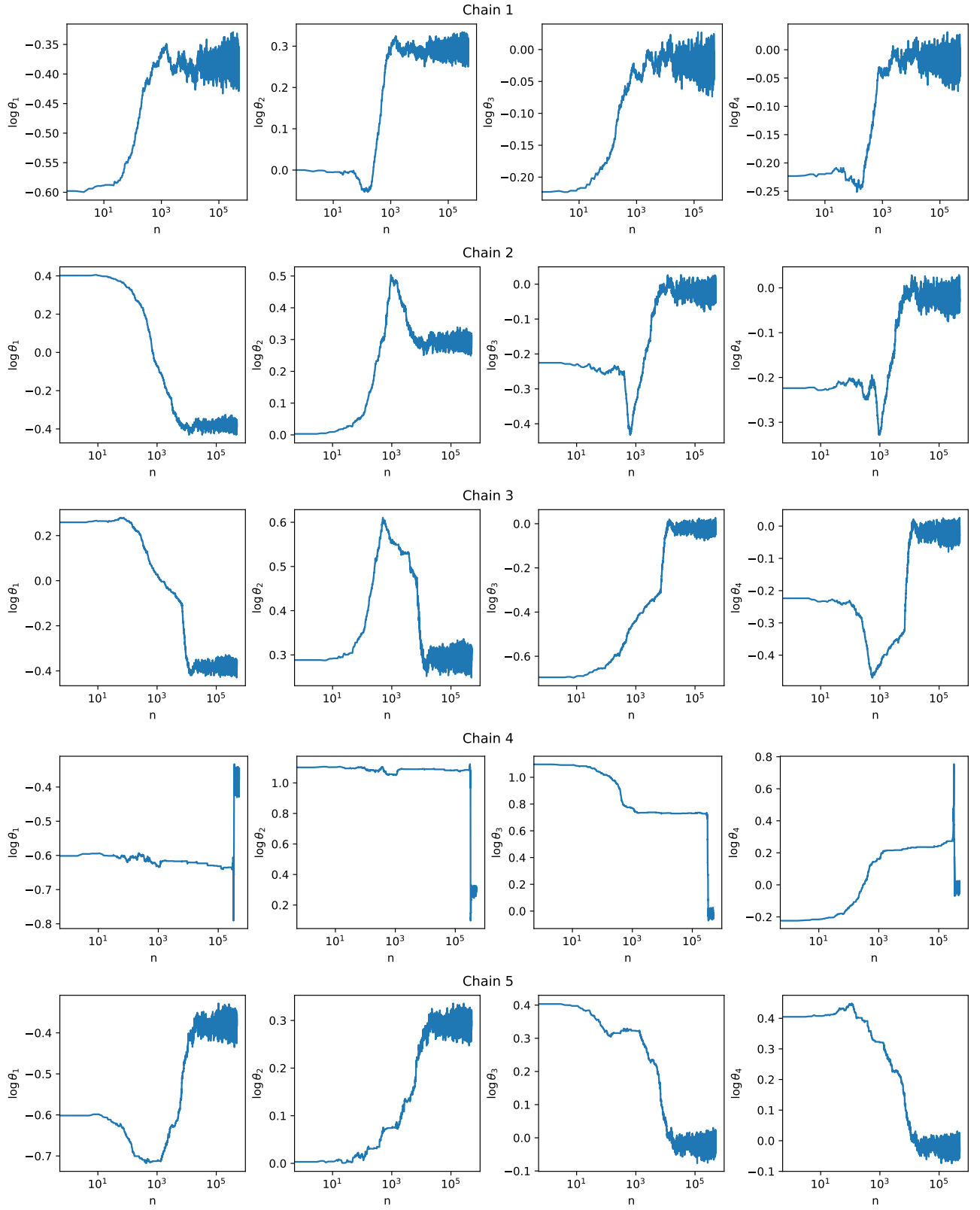$$

Appendix B

Additional figures

Figure B.1: Trace plots from the random-walk Metropolis-Hastings algorithm for the Lotka-Volterra inference problem.
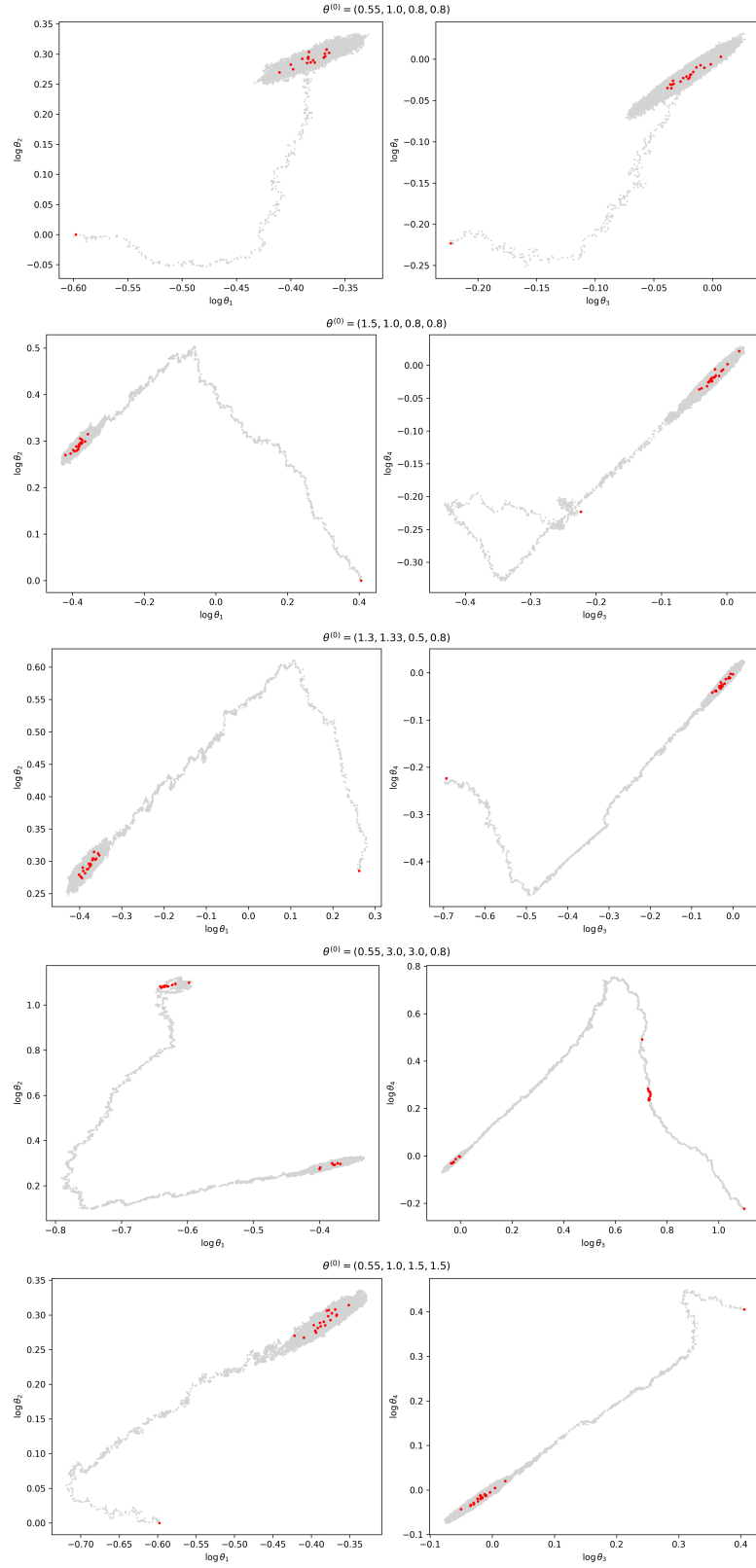
Figure B.2: Results of naïve thinning of the sample from the Lotka-Volterra model.
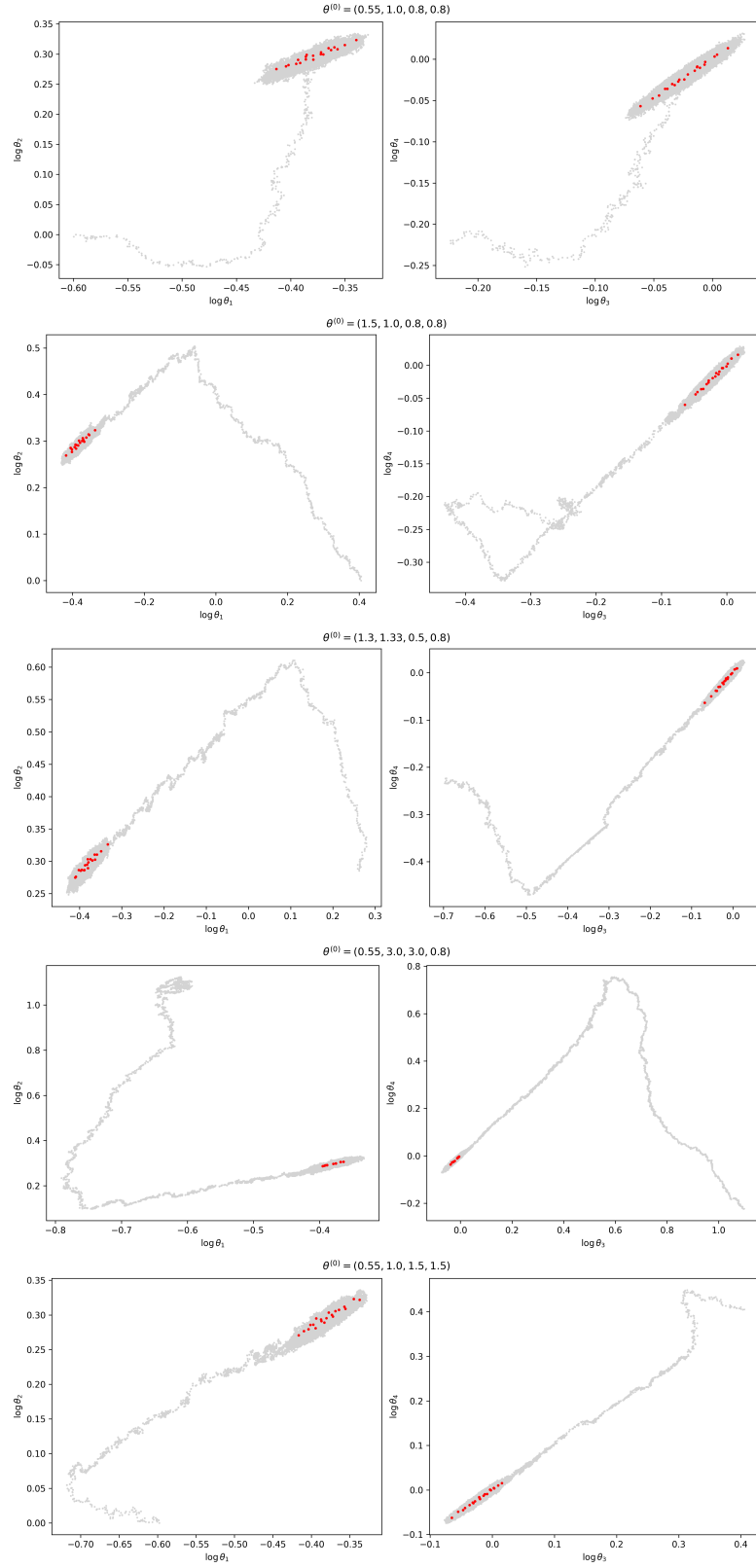
Figure B.3: Results of Stein thinning of the sample from the Lotka-Volterra model.