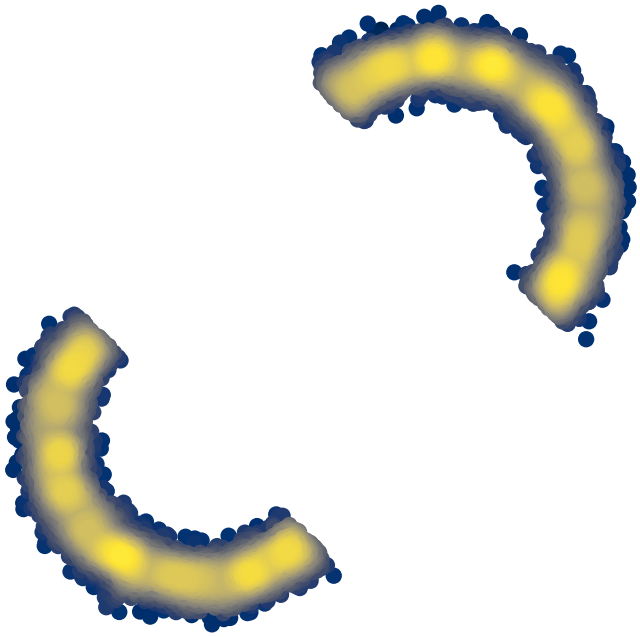


Sequential Bayesian inference with intractable likelihoods

A sequential mixture model method for posterior and likelihood estimation

Master's thesis in Engineering Mathematics and Computational Science

HENRIK HÄGGSTRÖM

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Sequential Bayesian inference with intractable likelihoods

A sequential mixture model method for posterior and likelihood estimation

HENRIK HÄGGSTRÖM

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
Gothenburg, Sweden 2023

Sequential Bayesian inference with intractable likelihoods
A sequential mixture model method for posterior and likelihood estimation
HENRIK HÄGGSTRÖM

Supervisor: Umberto Picchini, Depertment of Mathematical Sciences
Examiner: Moritz Schauer, Depertment of Mathematical Sciences

Master's Thesis 2023
Depertment of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Posterior samples from the Two Moons model in section 4.1.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Sequential Bayesian inference with intractable likelihoods
A sequential mixture model method for posterior and likelihood estimation
HENRIK HÄGGSTRÖM
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

In a Bayesian setting with realistic models it is not unusual that it is not possible to perform exact parameter inference. This is typically due to the fact that the likelihood function is not available in closed form or is intractable. Likelihood-free methods perform parameter inference in models where evaluating the likelihood is an intractable problem, but sampling data from a generative model is possible. With the expansion of machine learning, recent approaches learn the posterior distribution of the parameters by sequential updates of neural-network based density estimators. While these methods perform well, they require a network architecture to be specified and training the neural-network can be computationally demanding and time consuming. In this work we present a Bayesian inference method which, in place of neural networks, uses Gaussian mixtures sequentially learned through an expectation-maximization procedure. Posterior samples are then obtained via MCMC through an informative and self-tuned proposal sampler. Only the number of components in the Gaussian mixture needs to be specified to run the algorithm. We show the feasibility of this method and benchmark it against two neural-network based state-of-the-art Bayesian methods in 4 simulation studies. The results show that the proposed method is competitive and in some cases even outperforms the other methods in terms of simulation efficiency. Additionally, it is in most cases significantly faster to run.

Keywords: Bayesian inference, simulation-based inference, likelihood-free methods, multimodal posteriors, posterior estimation, likelihood estimation, R, Python.

Acknowledgements

I would like to express my gratitude to my supervisor Umberto Picchini for the continuous support given during the work on this thesis as well as introducing me to the subject of likelihood-free inference. Thank you for always taking time to discuss things and your enthusiasm in the project.

Henrik Häggström, Gothenburg, May 2023

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ABC	Approximate Bayesian Computation
APT	Automatic Posterior Transformation
BIC	Bayesian Information Criterion
EM	Expectation Maximization
GLLiM	Gaussian Locally Linear Mapping
KDE	Kernel Density Estimate
MCMC	Markov chain Monte Carlo
SDE	Stochastic Differential Equation
SeMPLE	Sequential Mixture Posterior and Likelihood Estimation
SMC-ABC	Sequential Monte Carlo Approximate Bayesian Computation
SNL	Sequential Neural Likelihood
SNPE	Sequential Neural Posterior Estimation

Contents

List of Figures

1

Introduction

This chapter introduces the subject and gives some background into why this is of interest to study. Additionally, the aim of this thesis is stated and a short overview of the chapters is provided.

1.1 Background

In many areas of science, complex simulations have been developed to describe phenomena of interest. Such simulator models is a natural way of modeling mechanistic processes in fields such as physics, protein folding, evolutionary biology and economics. With the power of modern computing even complex simulator models can be run forward to generate synthetic data which enables the use of new statistical methods.

A simulator, in the context of this thesis, is a generative model $\mathcal{M}(\boldsymbol{\theta})$ taking as input a vector of parameters $\boldsymbol{\theta}$ and producing an output vector of data $\boldsymbol{y}$ as following $\boldsymbol{\theta} \underset{\mathcal{M}}{\longrightarrow} \boldsymbol{y}$. The data $\boldsymbol{y}$ produced this way is such that $\boldsymbol{y} \sim p(\boldsymbol{y} \mid \boldsymbol{\theta})$. To give a simple example of how this works let us assume that we do not know how to sample from a generic Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, but we do know how to sample from a standard Gaussian distribution $X \sim \mathcal{N}(0, 1)$. With $\boldsymbol{\theta} = (\mu, \sigma)$ the generative model

$$\mathcal{M}(\boldsymbol{\theta}) = \mu + \sigma \times x, \tag{1.1}$$

allows for generation of samples $y \sim \mathcal{N}(\mu, \sigma^2)$. Note that sampling y from the generative model $\mathcal{M}(\boldsymbol{\theta})$ is equivalent to drawing samples from the likelihood $p(y \mid \boldsymbol{\theta})$, even if the likelihood is analytically unavailable.

In the setting of statistical inference one wants to perform inference about the parameters of the simulation model given observed data of the phenomena of interest. The difficulty is that the likelihood function is in many realistic cases intractable. The likelihood function may be unavailable in closed form or may be a high dimensional integral, making it intractable to evaluate. In Bayesian inference, the likelihood function is essential and without it traditional Bayesian methods, such as Markov Chain Monte Carlo (MCMC), are not applicable.

Statistical inference in the setting where the likelihood function is intractable has been named likelihood-free inference and simulation-based inference. A compact account of the current state of simulation-based inference is provided in [4]. One

popular method in likelihood-free inference is called Approximate Bayesian Computation (ABC) [1]. A more detailed introduction is given in section 2.3, but the idea is that observed and simulated data is compared based on some distance measure to learn about which parameter values likely could have generated the observed data.

Likelihood estimation and posterior estimation are two other approaches to likelihood-free inference. The idea with these methods is to use the simulator to train a likelihood estimator or posterior estimator. In these methods, parameters are sampled from a prior distribution or other proposal distribution and corresponding data is generated by the model simulator. The likelihood or posterior estimate is then obtained by training or fitting a surrogate distribution to the data set of proposed parameters and simulated data. Likelihood estimation and posterior estimation methods differ from ABC methods in the way that one obtains a parametric estimate of the likelihood or posterior respectively that can be used for sampling from the posterior.

Instead of sampling parameters from the possibly uninformative prior distribution, the current posterior estimate or posterior sample can be used to guide the proposal distribution to propose parameters that increases our knowledge about the posterior distribution the most. In [4] this is referred to as active learning and it can reduce the number of simulations required to obtain accurate inference drastically.

Recently there has been an accelerated expansion of simulation-based methods coming from the introduction of machine learning based methods. Two such algorithms are Sequential Neural Posterior Estimation C (SNPE-C), first denoted Automatic Posterior Transformation (APT) in [12], and Sequential Neural Likelihood (SNL) [19]. SNPE-C and SNL are described in section 2.7.1 and 2.7.2 respectively. These methods use neural networks trained on parameter and simulated data pairs to approximate the posterior density and likelihood respectively, by a surrogate distribution. Using neural networks to approximate probability densities is called neural density estimation. Stochastic gradient descent is typically employed altogether with the backpropagation algorithm towards obtaining such surrogate functions.

In this thesis, a new simulation-based algorithm called Sequential Mixture Posterior and Likelihood Estimation (SeMPLE) is introduced. It is similar to SNPE-C and SNL in the sense that likelihood estimation and posterior estimation is used and it also uses active learning to guide the proposal distribution. One major difference is that the introduced method uses an expectation-maximization (EM) procedure to fit Gaussian mixtures to approximate the likelihood and posterior, as opposed to neural density estimation. This means that no network architecture has to be defined, only the number of components in the Gaussian mixture. Additionally, training the neural-network can be computationally demanding and time consuming, which is avoided with SeMPLE. It is of interest to investigate if the EM procedure is powerful enough to provide exact density estimations while being computationally less demanding.

1.2 Aim

The aim of this thesis is to investigate this proposed method of simulation-based inference. We want to see how well this alternative approach performs compared to existing state-of-art Bayesian algorithms such as SNPE-C and SNL. It is of interest to find the limitations of SeMPLE and possibly expand the initial implementation according to these findings.

The intended outcome is an implementation of the method and comparisons of results from applications to several different simulation models with results from SNPE-C and SNL.

We will consider questions such as:

- Does SeMPLE produce samples from a posterior distribution similar to the true posterior?
- How can the different simulation-based algorithms be compared in a meaningful way?
- Is SeMPLE competitive compared to, or even outperforming, existing state-of-art simulation-based inference methods?
- What are the limitations of SeMPLE, are there ways to improve it by modifying the initial implementation?

1.3 Overview

In Chapter 2, the SeMPLE method and its underlying theoretical components are introduced. Additionally, the state-of-the-art algorithms SNPE-C and SNL are explained. Next, Chapter 3 explains how computations were performed in R and Python. Furthermore, the methods of benchmarking the algorithms are explained and motivated. Chapter 4 contains simulation studies with 4 different models and results. Lastly, Chapter 5 summarizes the main conclusions and presents open questions and future work.

2

Theory

In this chapter, a fundamental introduction to Bayesian inference is given and the Metropolis-Hastings algorithm used to sample from a probability distribution is explained. Next, Approximate Bayesian Computation (ABC) is introduced to give insight into a simple likelihood-free method. The SeMPLE method and its underlying theoretical components is introduced. Lastly, the neural density estimation methods SNPE-C and SNL are explained.

2.1 Bayesian inference

Bayesian inference methods are a way of updating one's beliefs upon observing data. The way this is performed is given by Bayes' theorem. Let $\boldsymbol{\theta}$ be the (possibly multidimensional) parameter of interest and let $\boldsymbol{y}_0$ be the observed data. Furthermore, let $p(\boldsymbol{\theta})$ be the prior probability density function which incorporates the prior beliefs about $\boldsymbol{\theta}$ before observing $\boldsymbol{y}_0$, and let $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ be the likelihood function. Bayes' theorem formulated in terms of probability density functions is then given as

$$p(\boldsymbol{\theta} \mid \boldsymbol{y}_0) = \frac{p(\boldsymbol{y}_0 \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{y}_0)}. \tag{2.1}$$

The probability density function $p(\boldsymbol{\theta} \mid \boldsymbol{y}_0)$ is called the posterior distribution of $\boldsymbol{\theta}$ given the observation $\boldsymbol{y}_0$ and incorporates one's beliefs about $\boldsymbol{\theta}$ after observing the data $\boldsymbol{y}_0$. The goal of Bayesian inference is to compute the posterior distribution of some parameter of interest. It can also be of interest to obtain a sample from the posterior distribution.

From equation (2.1) it is clear that the likelihood function plays a central role in Bayesian inference. But what if the likelihood function for some reason is intractable? Computing the posterior by using Bayes theorem explicitly becomes impossible if the likelihood function can not be evaluated. It is even the case that other traditional methods such as the Metropolis-Hastings algorithm, explained further in section 2.2, can not be used to sample from the posterior if the likelihood function can not be evaluated. This is where likelihood-free inference methods are required. There are different likelihood-free methods and they all provide inference without evaluation of the likelihood function. This is the setting that this thesis is concerned with, the likelihood function is assumed to be intractable and it can not be evaluated.

In section 2.2, we introduce the Metropolis-Hastings MCMC method. It would ideally be used to obtain a posterior sample if the likelihood function is explicitly available for a given model, which is not our working scenario. However, we will use it to produce samples from the exact posterior distribution of toy models to be used as ground truth in comparisons. We also use it in the implementation of SeMPLE in order to obtain posterior samples with this method.

2.2 Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm is a MCMC method to obtain random samples from a probability distribution. It is commonly used in settings where direct sampling is difficult, and has the advantage that it is sufficient to evaluate a function proportional to probability density function of the target distribution in order to obtain a sample from the target distribution. If one wants to sample from the posterior distribution, this means that the marginal likelihood $p(\boldsymbol{y})$ in equation (2.1), which can be computationally expensive to compute, does not have to be computed. The Metropolis-Hastings algorithm is given in Algorithm 1.

Algorithm 1 Metropolis-Hastings

Pick an initial value $\boldsymbol{\theta}_1$
for $j = 2 : N$ **do**
    Sample $\boldsymbol{\theta}^* \sim g(\boldsymbol{\theta} \mid \boldsymbol{\theta}_{j-1})$
    $\alpha = \min\left(1, \frac{P(\boldsymbol{\theta}^*)}{P(\boldsymbol{\theta}_{j-1})} \frac{g(\boldsymbol{\theta}_{j-1} \mid \boldsymbol{\theta}^*)}{g(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}_{j-1})}\right)$
    Sample $u \sim \mathcal{U}[0, 1]$
    if $u \leq \alpha$ **then**
        $\boldsymbol{\theta}_j = \boldsymbol{\theta}^*$
    else
        $\boldsymbol{\theta}_j = \boldsymbol{\theta}_{j-1}$
    end if
end for

Let $P(\boldsymbol{\theta})$ be the target density one wants to sample from and let $g(\boldsymbol{\theta} \mid \boldsymbol{\theta}_{j-1})$ be a proposal density that suggest a new sample value given the previous value $\boldsymbol{\theta}_{j-1}$. The algorithm produces a Markov chain by proposing values, possibly influenced by the previous value, and accepting or rejecting them according to the acceptance probability α which is based on the proposal and target distribution probability density functions. It can be shown that the limiting and stationary distribution of the produced Markov chain will be the target distribution $P(\boldsymbol{\theta})$. Note that due to the fraction of target distribution densities in α, the target distribution is only required to be known up to a constant factor with respect to $\boldsymbol{\theta}$.

In section 2.3, one method of likelihood-free inference called approximate Bayesian computation is introduced. It gives insight into a simulation-based inference method and provides some background to what problems with such methods that newer methods try to solve.

2.3 Approximate Bayesian Computation

The simplest version of Approximate Bayesian Computation (ABC) is the ABC rejection algorithm [21]. It is given in Algorithm 2 and presented similarly to the introduction given in [16]. The observed data is denoted as $\boldsymbol{y}_0$. The algorithm repeatedly draws parameters $\boldsymbol{\theta}' \sim p(\cdot)$ from the prior distribution and simulates corresponding data $\boldsymbol{y} \sim p(\cdot \mid \boldsymbol{\theta}')$ from the generative model. The proposed parameter $\boldsymbol{\theta}'$ is then accepted or rejected based on the distance of the simulated data $\boldsymbol{y}$ to the observed data $\boldsymbol{y}_0$ determined by some distance measure ρ. The distance measure ρ can for example be the Euclidian distance. The distance threshold of acceptance is denoted as $\epsilon > 0$. In the limit $\epsilon \to 0$, the ABC rejection algorithm will provide samples from the exact posterior distribution under the assumption that no summary statistic of the data is used, but at the same time the acceptance probability vanishes. In practical applications, small values of ϵ requires unfeasibly many simulations, leading to a trade-off between sample efficiency and inference quality.

ABC methods suffer from the curse of dimensionality. When the data is high-dimensional, all data will have a high distance to the observed data, introducing the need for summary statistics to reduce the dimensionality of the data. Finding powerful summary statistics that preserves much of the information in the data is case specific and can require domain knowledge.

Improvements to the ABC rejection algorithm include Sequential Monte Carlo (SMC) methods, thus giving rise to SMC-ABC [2] that make proposal of parameters more efficient than sampling from the prior by using the knowledge of previously accepted parameters. This is the same idea that was previously referred to as active learning and it can reduce the number of model simulations needed to obtain high quality inference drastically.

Algorithm 2 ABC rejection

1: **for** $i = 1 : N$ **do**
2: **repeat**
3: Generate $\boldsymbol{\theta}'$ from the prior distribution $p(\cdot)$
4: Simulate $\boldsymbol{y}$ from the generative model $p(\cdot \mid \boldsymbol{\theta}')$
5: **until** $\rho(\boldsymbol{y}_0, \boldsymbol{y}) < \epsilon$
6: set $\boldsymbol{\theta}_i = \boldsymbol{\theta}'$
7: **end for**

An important goal with SeMPLE is to incorporate a method of active learning. Section 2.4 introduces a core component of SeMPLE, named GLLiM, that will be used to produce a well-targeted proposal function as well as likelihood and posterior approximations.

2.4 GLLiM

Gaussian Locally-Linear Mapping (GLLiM) was originally introduced in [5]. The problem the authors intend to solve is high-dimensional to low-dimensional regression. Their approach is to exchange the roles of the input and response variables in the regression, such that the low-dimensional variable becomes the regressor. The parameters of the low-to-high regression model are estimated and the authors derive the forward parameters, characterizing the high-to-low regression, in closed form. By using (Gaussian) mixture models, this inverse-then-forward strategy becomes tractable. The mixture model parameters are estimated using a maximum likelihood principle with an expectation-maximization (EM) procedure that can be found in [5].

To formalize the explanation above with much of the notation borrowed from [9], let $\boldsymbol{\theta} \in \mathbb{R}^l$ be (presumably low-dimensional) parameters and $\boldsymbol{y} \in \mathbb{R}^d$ be (possibly high-dimensional) observations. Assume that the observed $\boldsymbol{y}$ is the image of the parameter $\boldsymbol{\theta}$ by an affine transformation τ_k, among K, plus an error term. This is modeled by a latent variable z such that z=k if and only if $\boldsymbol{y}$ is the image of $\boldsymbol{\theta}$ by the transformation τ_k. Let $\tilde{\boldsymbol{\phi}}$ denote the parameters of the model. The following decomposition of the joint probability distribution is used in [5]:

$$p(\boldsymbol{y}, \boldsymbol{\theta}; \tilde{\boldsymbol{\phi}}) = \sum_{k=1}^{K} p(\boldsymbol{y} \mid \boldsymbol{\theta}, z = k; \tilde{\boldsymbol{\phi}}) p(\boldsymbol{\theta} \mid z = k; \tilde{\boldsymbol{\phi}}) p(z = k; \tilde{\boldsymbol{\phi}}). \tag{2.2}$$

The locally affine function mapping $\boldsymbol{\theta}$ onto $\boldsymbol{y}$ is

$$\boldsymbol{y} = \sum_{k=1}^{K} \mathbb{1}\{z = k\}(\tilde{\boldsymbol{A}}_k \boldsymbol{\theta} + \tilde{\boldsymbol{b}}_k + \tilde{\boldsymbol{\epsilon}}_k), \tag{2.3}$$

where $\mathbb{1}$ denotes the indicator function, and $\tilde{\boldsymbol{A}}_k$ and $\tilde{\boldsymbol{b}}_k$ define the affine transformation τ_k. The variable $\tilde{\boldsymbol{\epsilon}}_k$ corresponds to an error term capturing both the observation noise and the reconstruction error due to the affine approximation.

Assuming $\tilde{\boldsymbol{\epsilon}}_k \sim \mathcal{N}_d(\mathbf{0}, \tilde{\boldsymbol{\Sigma}}_k)$ does not depend on $\boldsymbol{\theta}$, $\boldsymbol{y}$ or z we obtain

$$p(\boldsymbol{y} \mid \boldsymbol{\theta}, z = k; \tilde{\boldsymbol{\phi}}) = \mathcal{N}_d(\boldsymbol{y}; \tilde{\boldsymbol{A}}_k \boldsymbol{\theta} + \tilde{\boldsymbol{b}}_k, \tilde{\boldsymbol{\Sigma}}_k). \tag{2.4}$$

For the posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{y})$ and the likelihood $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ to be easily derived it is important to control the joint distribution $p(\boldsymbol{y}, \boldsymbol{\theta})$. In [5] the GLLiM model is derived assuming the joint distribution $p(\boldsymbol{y}, \boldsymbol{\theta})$ is a mixture of Gaussian distributions. To complete the hierarchical definition in (2.2), $\boldsymbol{\theta}$ is assumed to follow mixture of Gaussians specified by

$$p(\boldsymbol{\theta} \mid z = k; \tilde{\boldsymbol{\phi}}) = \mathcal{N}_l(\boldsymbol{\theta}; \tilde{\boldsymbol{c}}_k, \tilde{\boldsymbol{\Gamma}}_k), \tag{2.5a}$$

$$p(z = k; \tilde{\boldsymbol{\phi}}) = \pi_k. \tag{2.5b}$$

This assumption specifies a prior distribution of $\boldsymbol{\theta}$, however our main interest is in the the posterior distribution and the likelihood that can be derived with this prior assumption.

The mixture model parameters can be summarized as

$$\tilde{\boldsymbol{\phi}} = \{\pi_k, \tilde{c}_k, \tilde{\boldsymbol{\Gamma}}_k, \tilde{\boldsymbol{A}}_k, \tilde{\boldsymbol{b}}_k, \tilde{\boldsymbol{\Sigma}}_k\}_{k=1}^K. \tag{2.6}$$

Once the parameter vector $\tilde{\boldsymbol{\phi}}$ has been estimated by an EM procedure explained in detail in [5] and [9], one obtains the conditional density

$$q_{\tilde{\phi}}(\boldsymbol{y} \mid \boldsymbol{\theta}) = \sum_{k=1}^{K} \tilde{\eta}_k(\boldsymbol{\theta}) \mathcal{N}_d(\boldsymbol{y}; \tilde{\boldsymbol{A}}_{\boldsymbol{k}}\boldsymbol{\theta} + \tilde{\boldsymbol{b}}_k, \tilde{\boldsymbol{\Sigma}}_k), \tag{2.7}$$

with

$$\tilde{\eta}_k(\boldsymbol{\theta}) = \frac{\pi_k \mathcal{N}_l(\boldsymbol{\theta}; \tilde{\boldsymbol{c}}_k, \tilde{\boldsymbol{\Gamma}}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}_l(\boldsymbol{\theta}; \tilde{\boldsymbol{c}}_j, \tilde{\boldsymbol{\Gamma}}_j)}, \tag{2.8}$$

that we will use as a likelihood estimation. The change in notation of the likelihood to $q_{\tilde{\phi}}$ is to emphasize that this is the density estimator with parameters $\tilde{\boldsymbol{\phi}}$ that will be used in SeMPLE. The forward relation parameters $\boldsymbol{\phi} = \{\pi_k, \boldsymbol{c}_k, \boldsymbol{\Gamma}_k, \boldsymbol{A}_k, \boldsymbol{b}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ can easily be deduced from $\tilde{\boldsymbol{\phi}}$ and are given in [5] by

$$\boldsymbol{c}_k = \tilde{\boldsymbol{A}}_k \tilde{\boldsymbol{c}}_k + \tilde{\boldsymbol{b}}_k \tag{2.9a}$$

$$\boldsymbol{\Gamma}_k = \tilde{\boldsymbol{\Sigma}}_k + \tilde{\boldsymbol{A}}_k \tilde{\boldsymbol{\Sigma}}_k \tilde{\boldsymbol{A}}_k{}^\top \tag{2.9b}$$

$$\boldsymbol{\Sigma}_k = \left(\tilde{\boldsymbol{\Gamma}}_k^{-1} + \tilde{\boldsymbol{A}}_k{}^\top \tilde{\boldsymbol{\Sigma}}_k^{-1} \tilde{\boldsymbol{A}}_k\right)^{-1} \tag{2.9c}$$

$$\boldsymbol{A}_k = \boldsymbol{\Sigma}_k \tilde{\boldsymbol{A}}_k{}^\top \tilde{\boldsymbol{\Sigma}}_k^{-1} \tag{2.9d}$$

$$\boldsymbol{b}_k = \boldsymbol{\Sigma}_k \left(\tilde{\boldsymbol{\Gamma}}_k^{-1} \tilde{\boldsymbol{c}}_k - \tilde{\boldsymbol{A}}_k{}^\top \tilde{\boldsymbol{\Sigma}}_k^{-1} \tilde{\boldsymbol{b}}_k\right). \tag{2.9e}$$

With the forward relation parameters $\boldsymbol{\phi}$ we obtain the conditional density

$$q_{\phi}(\boldsymbol{\theta} \mid \boldsymbol{y}) = \sum_{k=1}^{K} \eta_k(\boldsymbol{y}) \mathcal{N}_l(\boldsymbol{\theta}; \boldsymbol{A}_k \boldsymbol{y} + \boldsymbol{b}_k, \boldsymbol{\Sigma}_k), \tag{2.10}$$

with

$$\eta_k(\boldsymbol{y}) = \frac{\pi_k \mathcal{N}_d(\boldsymbol{y}; \boldsymbol{c}_k, \boldsymbol{\Gamma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}_d(\boldsymbol{y}; \boldsymbol{c}_j, \boldsymbol{\Gamma}_j)}, \tag{2.11}$$

that we will use as a posterior estimate. Note that once $\tilde{\boldsymbol{\phi}}$ has been estimated by the EM procedure $\boldsymbol{\phi}$ can easily be computed by the equations (2.9) and we obtain both a likelihood estimate and posterior estimate in the same GLLiM run.

In the context of this thesis, the conditional densities (2.7) and (2.10) are used to approximate the likelihood and posterior respectively. They will be referred to as the surrogate likelihood and surrogate posterior, respectively. The way this is performed and integrated with the rest of the SeMPLE method is explained in section 2.6.

In section 2.5 a neural density estimation based algorithm is introduced. SeMPLE builds strongly on this method, which is why it is important to introduce it prior to introducing SeMPLE. The SNPE-C and SNL algorithms are also strongly connected to this algorithm.

2.5 SNPE-A

Sequential Neural Posterior Estimation A (SNPE-A) [17] is a parametric approach to likelihood-free inference where a parametric approximation to the exact posterior is learned by sequential updates. This approximation is acquired by training a mixture density network (MDN) [3], which is a Gaussian mixture where the mean and covariances are parameterised via neural networks. However, no method of estimating the number of mixture components is provided in SNPE-A.

What would later be named the SNPE-A algorithm is given in Algorithm 3 and 4. The MDN with parameters $\boldsymbol{\phi}$ is denoted by $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y})$ and the proposal distribution that parameters are drawn from is denoted $\tilde{p}(\boldsymbol{\theta})$. The intuitive explanation of SNPE-A is that parameters $\boldsymbol{\theta}_n$ are drawn from a proposal distribution $\tilde{p}(\boldsymbol{\theta})$ and corresponding data $\boldsymbol{y}_n$ are simulated from the generative model $p(\boldsymbol{y} \mid \boldsymbol{\theta}_n)$, the MDN $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y})$ is then trained on the data set $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$ to obtain a posterior approximation.

The authors of [17] show that using preliminary posterior fits to guide future simulation reduces the number of simulations required to learn an accurate posterior approximation drastically. In terms of notation, this means that parameters are proposed from $\tilde{p}(\boldsymbol{\theta})$, which can be different from the prior $p(\boldsymbol{\theta})$. However, not sampling from the prior distribution requires a correction factor to target the true posterior distribution. The motivation behind this is given in the proposition below, which is the same as Proposition 1 in [17].

Proposition 1 *We assume that each of a set of N pairs $(\boldsymbol{\theta}_n, \boldsymbol{y}_n)$ was independently generated by*

$$\boldsymbol{\theta}_n \sim \tilde{p}(\boldsymbol{\theta}) \quad \textit{and} \quad \boldsymbol{y}_n \sim p(\boldsymbol{y} \mid \boldsymbol{\theta}). \tag{2.12}$$

In the limit $N \to \infty$, the probability parameter vectors $\prod_n q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_n \mid \boldsymbol{y}_n)$ is maximized w.r.t $\boldsymbol{\phi}$ if and only if

$$q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y}) \propto \frac{\tilde{p}(\boldsymbol{\theta})}{p(\boldsymbol{\theta})} p(\boldsymbol{\theta} \mid \boldsymbol{y}), \tag{2.13}$$

provided a setting of $\boldsymbol{\phi}$ that makes $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y})$ proportional to $\frac{\tilde{p}(\boldsymbol{\theta})}{p(\boldsymbol{\theta})} p(\boldsymbol{\theta} \mid \boldsymbol{y})$ exists.

The proof of Proposition 1 can be found in [17]. The intuition is that if we simulate enough parameters from the prior, the density estimator $q_{\boldsymbol{\phi}}$ would learn a conditional of the joint prior model over parameters and data, which is the posterior $p(\boldsymbol{\theta} \mid \boldsymbol{y})$. When we draw parameters from another distribution it is necessary to "importance reweight" the result.

Based on Proposition 1 the authors of [17] suggest to estimate the posterior by

$$\hat{p}(\boldsymbol{\theta} \mid \boldsymbol{y}_0) \propto \frac{p(\boldsymbol{\theta})}{\tilde{p}(\boldsymbol{\theta})} q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y}_0), \tag{2.14}$$

as this will target the the true posterior.

In SNPE-A the proposal prior $\tilde{p}(\boldsymbol{\theta})$ trained in Algorithm 3 is restricted to be Gaussian distribution. The prior $p(\boldsymbol{\theta})$ is assumed to be a uniform or Gaussian to allow

analytical calculation of $\tilde{p}(\boldsymbol{\theta} \mid \boldsymbol{y} = \boldsymbol{y}_0)$. While it is convenient to be able to compute $\tilde{p}(\boldsymbol{\theta} \mid \boldsymbol{y} = \boldsymbol{y}_0)$ analytically, it is quite restrictive in terms of the the proposal prior and prior distribution.

Algorithm 3 Training of proposal prior

initialize $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y})$ with one component
$\tilde{p}(\boldsymbol{\theta}) \leftarrow p(\boldsymbol{\theta})$
repeat
  for $n = 1 : N$ **do**
    sample $\boldsymbol{\theta}_n \sim \tilde{p}(\boldsymbol{\theta})$
    sample $\boldsymbol{y}_n \sim p(\boldsymbol{y} \mid \boldsymbol{\theta}_n)$
  end for
  retrain $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y})$ on $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$
  $\tilde{p}(\boldsymbol{\theta}) \leftarrow \frac{p(\boldsymbol{\theta})}{\tilde{p}(\boldsymbol{\theta})}\, q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y}_o)$
until $\tilde{p}(\boldsymbol{\theta})$ has converged

Algorithm 4 Training of posterior

initialize $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{x})$ with K components
{if $q_{\boldsymbol{\phi}}$ available by Algorithm 3
initialize by replicating its
one component K times}
for $n = 1 : N$ **do**
  sample $\boldsymbol{\theta}_n \sim \tilde{p}(\boldsymbol{\theta})$
  sample $\boldsymbol{y}_n \sim p(\boldsymbol{y} \mid \boldsymbol{\theta}_n)$
end for
train $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y})$ on $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$
$\hat{p}(\boldsymbol{\theta} \mid \boldsymbol{y} = \boldsymbol{y}_o) \leftarrow \frac{p(\boldsymbol{\theta})}{\tilde{p}(\boldsymbol{\theta})}\, q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y}_o)$

After explaining SNPE-A, we are now ready to introduce the SeMPLE algorithm, which builds strongly on SNPE-A, in section 2.6

2.6 SeMPLE

The sequential Bayesian inference method explained in this section is the main contribution of this thesis. It is likelihood-free in the sense that the true likelihood function does not have to be evaluated at any point to obtain inference. Instead it relies on simulations from a simulator that are used to approximate the likelihood and posterior by a mixture of Gaussians through GLLiM. The method is suitable for multimodal posteriors due to the mixture distribution nature of GLLiM.

The full SeMPLE algorithm is given in Algorithm 5 and the details of the MCMC step in line 4 is given in Algorithm 6. With $p(\boldsymbol{\theta})$ being the prior distribution and $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ being the generative model that enables simulating data given a parameter $\boldsymbol{\theta}$. The intuitive explanation to Algorithm 5 is that parameters $\boldsymbol{\theta}_n \sim \tilde{p}(\boldsymbol{\theta})$ are sampled from the proposal distribution and corresponding simulated data $\boldsymbol{y}_n \sim p(\boldsymbol{y} \mid \boldsymbol{\theta}_n)$ are generated from the generative model, GLLiM is then used to fit a surrogate likelihood $q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{y} \mid \boldsymbol{\theta})$ and surrogate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y})$ to this data set $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$. Next, the proposal distribution is updated to the surrogate likelihood times the prior $\tilde{p}_{r+1}(\boldsymbol{\theta}) = q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{y}_0 \mid \boldsymbol{\theta})\; p(\boldsymbol{\theta})$, which is the current posterior estimate. In the next algorithm iteration, MCMC is used to sample from the proposal distribution and the procedure is repeated. In each algorithm iteration the algorithm output is the approximate posterior sample $\{\boldsymbol{\theta}_n\}_{n=1}^N$ from the proposal distribution $\tilde{p}_r(\boldsymbol{\theta})$.

Algorithm 5 SeMPLE

1: $\tilde{p}_1(\boldsymbol{\theta}) \leftarrow p(\boldsymbol{\theta})$
2: **for** $r = 0 : R$ **do**
3: **for** $n = 1 : N$ **do**
4: sample $\boldsymbol{\theta}_n \sim \tilde{p}_r(\boldsymbol{\theta})$ via MCMC using Algorithm 6
5: sample $\boldsymbol{y}_n \sim p(\boldsymbol{y} \mid \boldsymbol{\theta}_n)$
6: **end for**
7: **if** r = 0 **then**
8: collect $\mathcal{D}_0 = \{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$
9: train $q_{\tilde{\phi}_0}(\boldsymbol{y} \mid \boldsymbol{\theta})$ and $q_{\phi_0}(\boldsymbol{\theta} \mid \boldsymbol{y})$ on $\mathcal{D}_0$ using GLLiM
10: $\tilde{p}_{r+1}(\boldsymbol{\theta}) \leftarrow q_{\phi_r}(\boldsymbol{\theta} \mid \boldsymbol{y}_0)$
11: **else**
12: collect $\mathcal{D}_r = \mathcal{D}_{r-1} \cup \{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N \setminus \mathcal{D}_0$
13: train $q_{\tilde{\phi}_r}(\boldsymbol{y} \mid \boldsymbol{\theta})$ and $q_{\phi_r}(\boldsymbol{\theta} \mid \boldsymbol{y})$ on $\mathcal{D}_r$ using GLLiM
14: $\tilde{p}_{r+1}(\boldsymbol{\theta}) \leftarrow q_{\tilde{\phi}_r}(\boldsymbol{y}_0 \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})$
$((optional)\ \tilde{p}_{r+1}(\boldsymbol{\theta}) \leftarrow \frac{p(\boldsymbol{\theta})}{\tilde{p}_r(\boldsymbol{\theta})} q_{\phi_r}(\boldsymbol{\theta} \mid \boldsymbol{y}_0))$
15: **end if**
16: **end for**

Algorithm 6 Metropolis-Hastings with GLLiM surrogate posterior as proposal distribution

1: Pick an initial value $\boldsymbol{\theta}_1$
2: **for** $j = 2 : N$ **do**
3: Sample $\boldsymbol{\theta}^* \sim q_{\phi_{r-1}}(\boldsymbol{\theta} \mid \boldsymbol{y}_0)$
4: $\alpha = \min\{1, \frac{\tilde{p}_r(\boldsymbol{\theta}^*)}{\tilde{p}_r(\boldsymbol{\theta}_{j-1})} \frac{q_{\phi_{r-1}}(\boldsymbol{\theta}_{j-1} \mid \boldsymbol{y}_0)}{q_{\phi_{r-1}}(\boldsymbol{\theta}^* \mid \boldsymbol{y}_0)}\}$
5: Sample $u \sim \mathcal{U}[0, 1]$
6: **if** $u \leq \alpha$ **then**
7: $\boldsymbol{\theta}_j = \boldsymbol{\theta}^*$
8: **else**
9: $\boldsymbol{\theta}_j = \boldsymbol{\theta}_{j-1}$
10: **end if**
11: **end for**

A benefit of using GLLiM is that one easily obtains approximations of both the likelihood and posterior. This means that one can choose to target the (corrected) posterior distribution, similarly to SNPE-A, or the likelihood times prior in line 14 of Algorithm 5. In theory, this flexibility could be beneficial if either the likelihood or posterior is easier to learn for a given model. While we report the optional procedure of targeting the corrected surrogate posterior, we anticipate that we actually discourage its use as from the suboptimal results it produces. Note that we select one of the choices about the proposal distribution in line 14 of the algorithm and stick with it throughout all algorithm iterations.

The exception of MCMC being used to sample from the proposal distribution is in iteration r=0, when $\tilde{p}_0(\boldsymbol{\theta}) = p(\boldsymbol{\theta})$, and in iteration r=1, when $\tilde{p}_1(\boldsymbol{\theta}) = q_{\phi_0}(\boldsymbol{\theta} \mid \boldsymbol{y}_0)$. In

iteration r=0, there is no need to use MCMC since the prior distribution is assumed to be simple enough for explicit sampling. The reason for the surrogate posterior being used as proposal distribution in iteration r=1, is that the MCMC step can be avoided by sampling explicitly from the Gaussian mixture surrogate posterior $q_{\phi_0}(\boldsymbol{\theta} \mid \boldsymbol{y}_0)$. In later iterations, the correction factor in equation (2.14) is needed to continue sampling from the approximate posterior, which makes the use of MCMC necessary.

By using MCMC to sample from the proposal distribution $\tilde{p}(\boldsymbol{\theta})$ we do not rely on being able to compute the proposal distribution analytically, instead being able to evaluate it is sufficient. Hence, the restrictions on the prior distribution and proposal prior in SNPE-A are not present in SeMPLE, which allows more flexibility.

The initial training set $\mathcal{D}_0$ is disregarded in the coming data sets $\mathcal{D}_r$, $r > 0$. This is because in $\mathcal{D}_0$, the parameters $\boldsymbol{\theta}_n$ are sampled from the prior, which is typically not very informative, and if the number of samples N is large, too much weight of the cumulative training set $\mathcal{D}_r$ would be put on $\mathcal{D}_0$. The idea of gathering data cumulatively in $\mathcal{D}_r$ is to improve the GLLiM fit by giving it more data, as compared to using the parameter, data pairs $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$ from the most recent iteration only.

The MCMC sampling in Algorithm 5 is performed by the Metropolis-Hastings algorithm. A benefit of acquiring both the surrogate likelihood $q_{\tilde{\phi}}(\boldsymbol{y} \mid \boldsymbol{\theta})$ with GLLiM, and the surrogate posterior $q_{\phi}(\boldsymbol{\theta} \mid \boldsymbol{y})$ through the closed formulas in equation (2.9), is that the surrogate posterior can be used as a proposal distribution in the Metropolis-Hastings algorithm. This procedure is given in detail in Algorithm 6. Using the surrogate posterior as a proposal sampler in the Metropolis-Hastings algorithm has the benefit of avoiding manual tuning of a proposal distribution. Another benefit is that the surrogate posterior should improve in the sense that proposals become closer to true posterior samples with further SeMPLE iterations. This could result in high Metropolis-Hastings acceptance ratios and good mixing of the produced Markov chain without the need for manual tuning of the proposal distribution.

One thing to note with using the surrogate posterior as proposal distribution is that both the proposal distribution and the target distribution are influenced by the GLLiM fit. This could mean that Algorithm 5 converges relatively quickly because the proposal distribution will most likely propose parameter values that results in high surrogate likelihood values and high surrogate posterior values due to the forward/backwards relationship between the surrogate likelihood and surrogate posterior. One way of mitigating the effect of a too narrow proposal distribution and too early convergence is to inflate the covariance of the proposal distribution by a multiplicative factor.

In GLLiM, the number of components K in the Gaussian mixture model has to be specified prior to performing the EM procedure. In [5], the Bayesian Information Criterion (BIC) is proposed as a method of selecting the value of K. This procedure is outlined in section 3.4.

Next, the two neural density estimation methods SNPE-C and SNL to be used for comparisons are introduced in section 2.7.

2.7 Neural density estimation methods

Note that in this section the posterior and likelihood estimators denoted $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{y} \mid \boldsymbol{\theta})$ are neural density estimators with parameters $\boldsymbol{\phi}$, instead of Gaussian mixtures as in equation (2.10) and (2.7).

2.7.1 SNPE-C

The SNPE-C algorithm was introduced in [12] under the name Automatic Posterior Transformation (APT) with the name SNPE-C first being used in [6]. As the name suggests, SNPE-C is a successor to the SNPE-A algorithm. It shares the iterative refinement of the posterior estimate and proposal distribution, which is what is known as sequential neural posterior estimation (SNPE). SNPE-C is given in Algorithm 7, which is the same as Algorithm 1 in [12] with some adaptations in the notation. To account for the fact that the proposal distribution $\tilde{p}(\boldsymbol{\theta})$ is different from the prior distribution, they introduce

$$\tilde{q}_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y}) = q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y}) \frac{\tilde{p}(\boldsymbol{\theta})}{p(\boldsymbol{\theta})} \frac{1}{Z(\boldsymbol{y}, \boldsymbol{\phi})}, \tag{2.15}$$

where $Z(\boldsymbol{y}, \boldsymbol{\phi}) = \int_{\theta} q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y}) \frac{\tilde{p}(\boldsymbol{\theta})}{p(\boldsymbol{\theta})}$ is a normalization constant. In SNPE-C a neural network with weights $\boldsymbol{\phi}$ is trained on a data set of parameters and simulated data $\{\boldsymbol{\theta}_j, \boldsymbol{y}_j\}$ to minimize the loss function $\tilde{\mathcal{L}}(\boldsymbol{\phi}) = -\sum_{j=1}^{N} \log \tilde{q}_{\boldsymbol{\phi}}(\boldsymbol{\theta}_j \mid \boldsymbol{y}_j)$ over the weights $\boldsymbol{\phi}$. This results in a neural density estimate $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y})$ of $p(\boldsymbol{\theta} \mid \boldsymbol{y})$ that approximates the true posterior when the observed data $\boldsymbol{y}_0$ is plugged in. Note that the correction for not sampling from the prior happens during the learning step in SNPE-C as opposed to after the learning step in SNPE-A.

Algorithm 7 SNPE-C

Input: simulator with (implicit) density $p(\boldsymbol{y} \mid \boldsymbol{\theta})$, data $\boldsymbol{y}_0$, prior $p(\boldsymbol{\theta})$, density family $q_{\boldsymbol{\phi}}$, simulations per round N, number of rounds R.

$\tilde{p}_1(\boldsymbol{\theta}) := p(\boldsymbol{\theta})$
for $r = 1 : R$ **do**
  for $j = 1 : N$ **do**
    Sample $\boldsymbol{\theta}_{r,j} \sim \tilde{p}_r(\boldsymbol{\theta})$
    Simulate $\boldsymbol{y}_{r,j} \sim p(\boldsymbol{y} \mid \boldsymbol{\theta}_{r,j})$
  end for
  $\boldsymbol{\phi} \leftarrow \arg\min_{\boldsymbol{\phi}} \sum_{i=1}^{r} \sum_{j=1}^{N} -\log \tilde{q}_{\boldsymbol{\phi}}(\boldsymbol{\theta}_{i,j} \mid \boldsymbol{y}_{i,j})$ ▷ using (2.15)
  $\tilde{p}_{r+1}(\boldsymbol{\theta}) := q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y}_0)$
end for
return $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y}_0)$

Despite the similarities an important difference between SNPE-C and SeMPLE is that SNPE-C uses a neural network to fit the posterior approximation to the pa-

rameter and simulated data pairs, whereas SeMPLE uses GLLiM to fit mixture distributions to approximate the likelihood and posterior.

2.7.2 SNL

Sequential Neural Likelihood (SNL) [19] is similar to SNPE-C in the sense that the it trains a conditional neural density estimator and a proposal density is updated sequentially. The most important difference is that SNL estimates the likelihood instead of the posterior. This eliminates the need for corrections stemming from the proposal distribution not being the prior, but introduces an additional MCMC step to sample from the posterior distribution. In SNL, MCMC is used to sample from the posterior using the estimation $q_{\boldsymbol{\phi}}(\boldsymbol{y}_o \,|\, \boldsymbol{\theta})$ of the likelihood. SNL is detailed in Algorithm 8, which is the same as Algorithm 1 in [19].

SeMPLE share evident analogies with SNL when $\tilde{p}_{r+1}(\boldsymbol{\theta}) \leftarrow q_{\boldsymbol{\phi}_r}(\boldsymbol{y}_0 \,|\, \boldsymbol{\theta})p(\boldsymbol{\theta})$ in SeMPLE. The main difference is that SeMPLE uses GLLiM to approximate the likelihood instead of using a neural density estimator.

Algorithm 8 Sequential Neural Likelihood (SNL)

Input: observed data $\boldsymbol{y}_0$, estimator $q_{\boldsymbol{\phi}}(\boldsymbol{y} \,|\, \boldsymbol{\theta})$, number of rounds R, simulations per round N
Output: approximate posterior $\hat{p}(\boldsymbol{\theta} \,|\, \boldsymbol{y}_0)$

set $\hat{p}_0(\boldsymbol{\theta} \,|\, \boldsymbol{y}_0) = p(\boldsymbol{\theta})$ and $\mathcal{D} = \{\}$
for $r = 1 : R$ **do**
  for $n = 1 : N$ **do**
    sample $\boldsymbol{\theta}_n \sim \hat{p}_{r-1}(\boldsymbol{\theta} \,|\, \boldsymbol{y}_0)$ with MCMC
    simulate $\boldsymbol{y}_n \sim p(\boldsymbol{y} \,|\, \boldsymbol{\theta}_n)$
    add $(\boldsymbol{\theta}_n, \boldsymbol{y}_n)$ into $\mathcal{D}$
  end for
  (re-)train $q_{\boldsymbol{\phi}}(\boldsymbol{y} \,|\, \boldsymbol{\theta})$ on $\mathcal{D}$ and set $\hat{p}_r(\boldsymbol{\theta} \,|\, \boldsymbol{y}_0) \propto q_{\boldsymbol{\phi}}(\boldsymbol{y}_0 \,|\, \boldsymbol{\theta})\, p(\boldsymbol{\theta})$
end for
return $\hat{p}_R(\boldsymbol{\theta} \,|\, \boldsymbol{y}_0)$

3

Methods

This chapter explains the method of producing inference results with the SeMPLE, SNPE-C and SNL methods. Additionally, it explains the method of comparing inference results using metrics. Furthermore, we introduce a method of selecting the number of mixture components in GLLiM.

3.1 Simulation studies

Four simulation studies with different models were performed to asses and compare the performance of SeMPLE, SNPE-C and SNL. The Two Moons model and the multiple hyperboloid model provide the challenge of multimodal posterior distributions. The Ornstein-Uhlenbeck process gives a high dimensional time series as output from the simulation, and the Simple Likelihood Complex Posterior (SLCP) model is purposely designed to have a simple likelihood function, but complex posterior distribution. All models are introduced in Chapter 4. The use of several models with different characteristics provides insight into how SeMPLE performs in different scenarios.

The goal of all simulation-based inference algorithms is to obtain a posterior distribution as similar as possible to the true posterior, that one would compute analytically if possible. The similarity between the true posterior and the one inferred by an algorithm can be quantified with a suitable distance measure between posterior samples. However, this requires access to a reference sample from a ground-truth posterior, obtained by some other method. All models considered in this thesis makes it possible to obtain such a reference sample that can be used to measure performance.

SeMPLE was implemented in the programming language R with help of the *xLLiM* package [20] that provides a GLLiM implementation. The output $\{\boldsymbol{\theta}_n\}_{n=1}^N$ from the MCMC step in line 4 of Algorithm 5 was saved from every iteration r, except r=0 because this is simply a sample from the prior distribution. Obtaining an approximate posterior sample from every algorithm iteration r enables evaluation of how the posterior approximation improves with further iterations.

To avoid having a too narrow proposal distribution in SeMPLE (explained in section 2.6), the covariance of the proposal distribution was multiplied by 1.2 from algorithm iteration r=2 and forward in all the simulation studies. The value of 1.2 was found

to be a reasonable trade-off between widening the proposal distribution without decreasing the acceptance rate of the Metropolis-Hastings step to problematically low levels.

3.2 Benchmarking

To perform inference with SNPE-C and SNL the Python package *SBIBM* (Simulation-Based Inference Benchmark) [15] built on the *SBI* (Simulation-Based Inference) [23] package was used. One major challenge is how to make fair comparisons with algorithms that are inherently different. SBIBM tries to solve this problem by providing a framework where different algorithms can be run and compared on the same problems with publicly available settings. SBIBM has been used both to produce the SNPE-C and SNL posterior samples and to compute the C2ST metric introduced in section 3.3.1. The SNL default neural density estimator in SBIBM, Masked Autoregressive Flow (MAF) [18], was used when running SNL. Similarly, the SNPE-C default neural density estimator in SBIBM, Neural Spline Flow (NSF) [7], was used when running SNPE-C.

SBIBM provides true posterior reference samples corresponding to 10 different observed data sets of the Two Moons and SLCP models. The multiple hyperboloid model and the Ornstein-Uhlenbeck process were not implemented in SBIBM by default, but have been implemented in the framework to make comparisons possible for these models. In the case of the multiple hyperboloid model and the Ornstein-Uhlenbeck process, reference samples were obtained by the Metropolis-Hastings algorithm using the *mcmc* package [11] in R. The Metropolis-Hastings algorithm is likelihood-based which means that the models considered are for benchmarking purposes only and likelihood-based methods can be used to provide parameter inference.

A posterior sample size of 10^4 was used for both the reference samples and the samples from posteriors approximated by the different algorithms. The chosen sample size is a balance of being large enough to draw accurate conclusions about the quality of the approximation while being small enough for computational convenience.

As explained in [15], it is typically assumed that the total computation cost is dominated by simulation cost when the simulation model is complex. Therefore, SBIBM is based around reporting performance at different simulation budgets and the same approach has been used in this thesis to benchmark different algorithms. However, one should note that this is not the only way of comparing algorithms and sometimes simulation budget comparisons does not provide the full picture.

SBIBM is based around setting a total simulation budget and obtaining a posterior sample after all those simulations at the final iteration of the algorithm (if the algorithm is sequential). Because all the algorithms considered in this thesis are sequential, the SBIBM code of the algorithms SNPE-C and SNL was modified to output a posterior sample after each iteration instead of only from the last algorithm iteration. This provides more information about how the posterior estimation changes over iterations without the need to run each algorithm multiple times with different simulation budgets.

Another modification that was implemented in the SBIBM code of the SNPE-C and SNL algorithms was to output the runtime after each iteration r. This was done to be able to compare runtimes, as the runtimes with the same simulation budgets could vary heavily between different algorithms. However, there are some problems with this way of comparing runtime because the algorithms are implemented in different programming languages. This means that the implementations and internal optimizations are inherently different and can not be compared directly. However, it can still provide an idea of differences in runtime if the difference is very large. All computations were run on a desktop computer with a 6-core (12 threads) AMD Ryzen 5 2600 CPU.

3.3 Performance metrics

To quantify how well the posterior distribution is approximated there are a number of performance metrics. In the setting of this thesis, inference quality is measured by comparing samples from the true posterior, usually obtained by some MCMC method, with samples from the posterior approximation. SBIBM has the same approach and provides true posterior reference samples for the implemented tasks and the ability to compute the C2ST metric. The benefit of using performance metrics is that average results over multiple algorithm runs easily can be summarized. However, the performance metric values do not always correspond to what one can observe visually when comparing samples. For example, different metrics are affected differently by outliers in the samples. With this in mind, it can be problematic to consider one metric only when assessing performance, which is why both the C2ST metric and the Wasserstein distance presented in sections 3.3.1 and 3.3.2 were used.

3.3.1 C2ST

Classifier 2-Sample Tests (C2ST) [10], [14] trains a classifier to discriminate samples from the true and inferred posterior. The accuracy of the classifier can then be used to quantify how well the inferred posterior approximates the true posterior. An accuracy of 1 means that the classifier always manages to correctly discriminate samples from the two distributions, and an accuracy of 0.5 means that the classifier is as bad as randomly guessing which posterior distribution the samples come from. A high accuracy means that the classifier finds that there is a significant difference between the two set of samples, rejecting the hypotheses that the two sets are generated from the same distribution. On the other hand, an accuracy of 0.5 means that the classifier is unable to reject such hypothesis. Since the comparison is between samples from an approximate method and samples from an exact method, we wish to get scores close to 0.5. The C2ST implementation in SBIBM, using a multi-layer perceptron classifier, was used.

3.3.2 Wasserstein distance

The Wasserstein distance, originally defined in [13] in the context of optimal transport, is a distance function between probability distributions. Viewing each distribution as a unit of mass, it can intuitively be understood as the minimum total "cost" of turning one distribution into the other. The cost in this analogy is defined as the amount of mass moved times the distance moved. This optimal transport problem can only make sense if the distributions have the same mass. In the context of this thesis, this means that the samples to be compared have to be of equal size. Naturally, a lower Wasserstein distance means that the distributions are more similar. The implementation in the Python package POT [8] was used to compute the Wasserstein distance.

3.4 BIC

In GLLiM, the number of components in the Gaussian mixture model K has to be specified prior to performing the EM procedure. A too large value of K can result in overfitting and unnecessary computational effort, while a too small value of K could mean that the relationship between input and response variables can not be represented well enough. The Bayesian Information Criterion (BIC) [22] is a criterion for model selection that guides the selection of a parameter such as K. It is given in equation (3.1) where $\mathcal{L}(\boldsymbol{\phi})$ is the maximal value of the likelihood function, $D(\boldsymbol{\phi})$ is the total number of parameters in the model and N is the number of data points in the data set. The BIC is based on the likelihood function and the idea is that increasing the number of parameters of a model can increase the maximum likelihood value $\mathcal{L}(\boldsymbol{\phi})$, but may result in overfitting. The BIC tries to resolve this problem by introducing the penalty term $D(\boldsymbol{\phi}) \log N$ for the number of parameters in the model. By construction, the aim is to to minimize the BIC when selecting a model, as this becomes a balance between maximizing the likelihood while limiting the number of model parameters.

$$BIC = -2\mathcal{L}(\boldsymbol{\phi}) + D(\boldsymbol{\phi}) \log N \tag{3.1}$$

In [5], the BIC is suggested as a method of selecting the value of K. One problem with the BIC and the likelihood-free setting is that the true likelihood is assumed to be intractable and one can not assume that it can be evaluated. A solution to this is to produce a data set $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$ by sampling parameters $\boldsymbol{\theta}_n \sim p(\boldsymbol{\theta})$ from the prior and simulating corresponding $\boldsymbol{y}_n \sim p(\boldsymbol{y}|\boldsymbol{\theta}_n)$ from the generative model, then use GLLiM to fit surrogate likelihoods $q_{\tilde{\phi}}(\boldsymbol{y}|\boldsymbol{\theta})$ to the data set $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$ for different values of K. The surrogate likelihood can then be used in place of the true likelihood in the BIC computations. In [5], the log-likelihood

$$\mathcal{L}(\boldsymbol{\phi}) = \sum_{n=1}^{N} \log q_{\tilde{\phi}}(\boldsymbol{y}_n, \boldsymbol{\theta}_n), \tag{3.2}$$

with

$$q_{\tilde{\phi}}(\boldsymbol{y}_n, \boldsymbol{\theta}_n) = \sum_{j=1}^{K} \pi_j q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{y}_n, \boldsymbol{\theta}_n | z_n = j). \tag{3.3}$$

Note that

$$\begin{aligned} q_{\tilde{\phi}}(\boldsymbol{y}_n, \boldsymbol{\theta}_n | z_n = k) &= q_{\tilde{\phi}}(\boldsymbol{y}_n | \boldsymbol{\theta}_n, z_n = k) q_{\tilde{\phi}}(\boldsymbol{\theta}_n | z_n = k) \\ &= \mathcal{N}(\boldsymbol{y}_n; \tilde{\boldsymbol{A}}_k \boldsymbol{\theta} + \tilde{\boldsymbol{b}}_k, \tilde{\boldsymbol{\Sigma}}_k) \mathcal{N}(\boldsymbol{\theta}_n; \tilde{\boldsymbol{c}}_k, \tilde{\boldsymbol{\Gamma}}_k). \end{aligned} \tag{3.4}$$

The number of parameters that GLLiM needs to estimate

$$D(\boldsymbol{\phi}) = (K-1) + K(DL + D + L + \text{nbpar}_{\Sigma} + \text{nbpar}_{\Gamma}), \tag{3.5}$$

where nbpar_{Σ} and nbpar_{Γ} are the number of parameters in the covariance matrices $\tilde{\Sigma}_k$ and $\tilde{\Gamma}_k$, respectively, in equation (2.6). The covariance structure of the matrices $\tilde{\Sigma}_k$ and $\tilde{\Gamma}_k$ can be can be constrained to reduce the number of parameters that GLLiM needs to estimate. One possible constraint to the covariance structure is to set the covariance matrices to be isotropic, which means that they must be proportional to the identity matrix. The BIC computations were implemented and performed in the programming language R.

The GLLiM implementation in the *xLLiM* package removes mixture component k if the posterior mixture probability $\tilde{\eta}_k(\boldsymbol{\theta})$ in equation (2.8) becomes zero. Empirically it was found that when a posterior mixture probability becomes very small, GLLiM would sometimes crash with errors about failed internal matrix inversions. To prevent this, a threshold was set to remove mixture components with posterior mixture probability below this threshold. This also shows the need of having a method to select the initial number of mixture components K, as setting a too large value inflates the problem of small mixture component probabilities.

4

Results

This chapter containts simulation studies with the Two Moons model, the multiple hyperboloid model, the Ornstein-Uhlenbeck process and the SLCP model. The chapter is divided into 4 sections, one for each simulation study. In each section, the model is explained, the implementation details are provided and finally the results are presented.

4.1 Two Moons

The Two Moons model is a relatively simple inference task that serves well as an initial example. The purpose of using this model is to illustrate how algorithms deal with multimodality and the local crescent shape of the posterior distribution. In this simulation study, results with different SeMPLE settings are presented, something that is not done to the same extent in the other simulation studies to avoid too much repetition.

4.1.1 Model

The Two Moons model, described in [12], generates $\boldsymbol{y} \in \mathbb{R}^2$ for a given parameter $\boldsymbol{\theta} \in \mathbb{R}^2$ according to

$$a \sim \mathcal{U}(-\frac{\pi}{2}, \frac{\pi}{2}) \tag{4.1}$$

$$r \sim \mathcal{N}(0.1, 0.01^2) \tag{4.2}$$

$$\boldsymbol{p} = (r\cos(a) + 0.25,\ r\sin(a)) \tag{4.3}$$

$$\boldsymbol{y}^\top = \boldsymbol{p} + \left(-\frac{|\theta_1 + \theta_2|}{\sqrt{2}},\ \frac{-\theta_1 + \theta_2}{\sqrt{2}}\right). \tag{4.4}$$

A uniform prior $\mathcal{U}(-\mathbf{1}, \mathbf{1})$ independent on each component of $\boldsymbol{\theta}$ was used, as suggested in [12]. By construction, this toy example has a posterior distribution with two crescent shapes for a given observation, an example is shown in Figure 4.1.

4.1.2 Implementation

The BIC was used to determine a suitable value of the number of components K in the Gaussian mixture distribution in GLLiM. All BIC values are computed on the

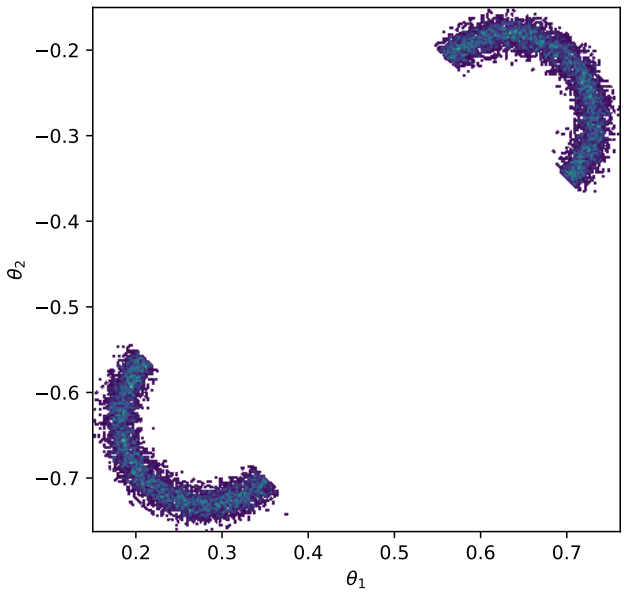


Figure 4.1: Example scatter plot of a Two Moons posterior sample. Bright colors indicates higher density.

same prior-predictive data set $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$ with $N = 2\,500$ prior to running SeMPLE. Figure 4.2 shows BIC as a function of K. The minimum BIC of the evaluated values is obtained at K=50 but the BIC values decreases the most from K=10 to K=30. By the Occam's razor principle, K=30 was used when running SeMPLE. Empirical experiments showed that a larger value of K did not improve results. The runtime to compute all BIC values in Figure 4.2 was 93 seconds $\approx$ 1.5 minutes. Note that the BIC computation is independent of the observed data set and only has to be performed once before running SeMPLE.

The three algorithms were run with the 10 different observed data sets and corresponding posterior reference samples provided in SBIBM. Hence, the metric results are averaged over 10 different data sets.

The relatively easy Two Moons inference task was set up with 10^4 total model simulations uniformly distributed across R=4 algorithm iterations. In the notation of the algorithms this corresponds to $N = 2\,500$. This was the same for all algorithms to make the comparison as fair as possible. A comparison with a similar number of model simulations but with a different set of algorithms can be found in [12].

4.1.3 Results

Figure 4.3 shows the C2ST metric and the Wasserstein distance as a function of number of model simulations for different SeMPLE settings. The settings compared differ in the choice of using the surrogate likelihood to update the proposal distribution in line 14 of Algorithm 5 to $\tilde{p}_{r+1}(\boldsymbol{\theta}) = q_{\tilde{\phi}_r}(\boldsymbol{y}_0 \,|\, \boldsymbol{\theta})p(\boldsymbol{\theta})$, or using the sur-

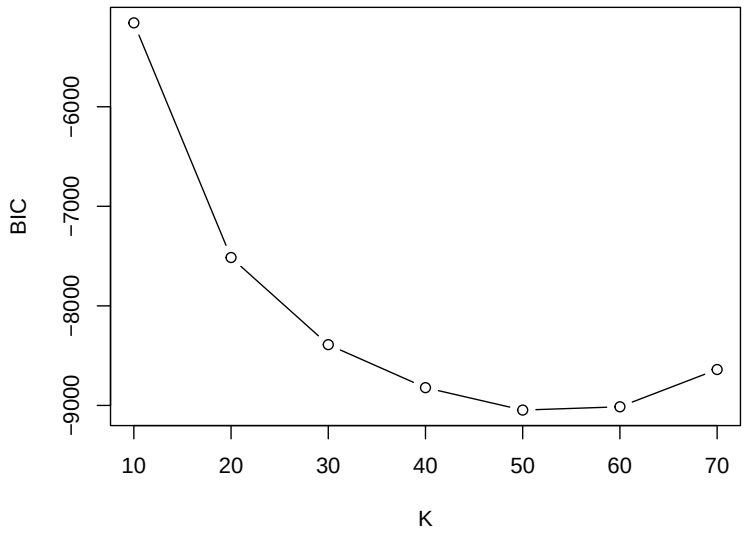
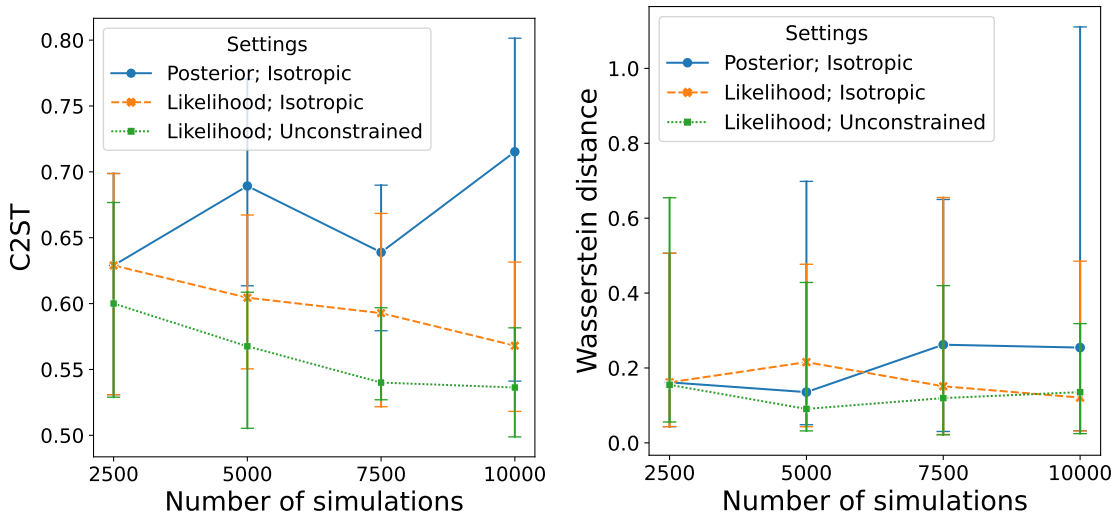


Figure 4.2: BIC as a function of number of Gaussian mixture components K for the Two Moons model.

rogate posterior to update the proposal distribution to $\tilde{p}_{r+1}(\boldsymbol{\theta}) = \frac{p(\boldsymbol{\theta})}{\tilde{p}_r(\boldsymbol{\theta})} q_{\phi_r}(\boldsymbol{\theta} \mid \boldsymbol{y}_0)$. In both settings, the GLLiM covariance matrices were set to be isotropic. Additionally, the combination of using fully-specified covariance matrices in GLLiM with the surrogate likelihood in the proposal distribution is included. It is clear that running MCMC using the surrogate likelihood to sample from $q_{\tilde{\phi}_r}(\boldsymbol{y}_0 \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})$ yields better results than running MCMC targeting the corrected surrogate posterior $\frac{p(\boldsymbol{\theta})}{\tilde{p}_r(\boldsymbol{\theta})} q_{\phi_r}(\boldsymbol{\theta} \mid \boldsymbol{y}_0)$. Targeting the corrected surrogate posterior gives higher variance in the metric results and both the C2ST metric and the Wasserstein distance increases with further iterations instead of decreasing (and improving). A possible explanation to why targeting the surrogate posterior results in inaccurate inference is that at iteration s the proposal distribution $\tilde{p}_s(\boldsymbol{\theta})$ will contain factors from all previous surrogate posteriors $q_{\phi_r}(\boldsymbol{\theta} \mid \boldsymbol{y}_0)$, $r < s$. Since we use MCMC, sampling from such a distribution is not a problem, but it could be that the variance of the MCMC target distribution becomes high as a consequence of the many surrogate posterior factors from previous iterations.

Based on these results, it was concluded that using the surrogate likelihood to target $q_{\tilde{\phi}_r}(\boldsymbol{y}_0 \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})$ in the MCMC step, with the surrogate posterior $q_{\phi_r}(\boldsymbol{\theta} \mid \boldsymbol{y}_0)$ as proposal distribution in the Metropolis-Hastings algorithm is a superior method overall. This will be used in all the other simulation studies. Using fully-specified covariance matrices in GLLiM and hence increasing the number of mixture model parameters seems to yield slightly better results with the Two Moons model.

To provide further insight into what happens in the Metropolis-Hastings step of SeMPLE, Figure 4.4 shows the acceptance rates when using the surrogate likelihood or surrogate posterior in the target distribution. The acceptance rates are higher



(a) (b)

Figure 4.3: Two Moons: Median C2ST and Wasserstein distance of 10 SeMPLE runs with different data sets as a function of number of model simulations. The three different SeMPLE settings refers to if the surrogate likelihood or the corrected surrogate posterior is used in the MCMC target distribution, and constraints used on the GLLiM covariance matrices. Error bars show min/max values.

and increases when the surrogate likelihood is used to sample from $q_{\tilde{\phi}_r}(\boldsymbol{y}_0 \mid \boldsymbol{\theta})p(\boldsymbol{\theta})$ in the MCMC step. The higher acceptance rates means that there are more unique samples in the posterior samples and hence more of the posterior distribution could potentially be explored. The increase in acceptance rate with further iterations could indicate that the algorithm converges when the surrogate posterior proposes parameters with high surrogate likelihood values.

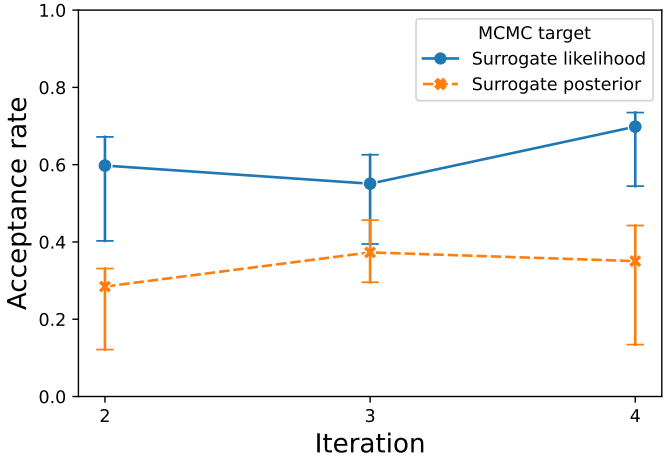
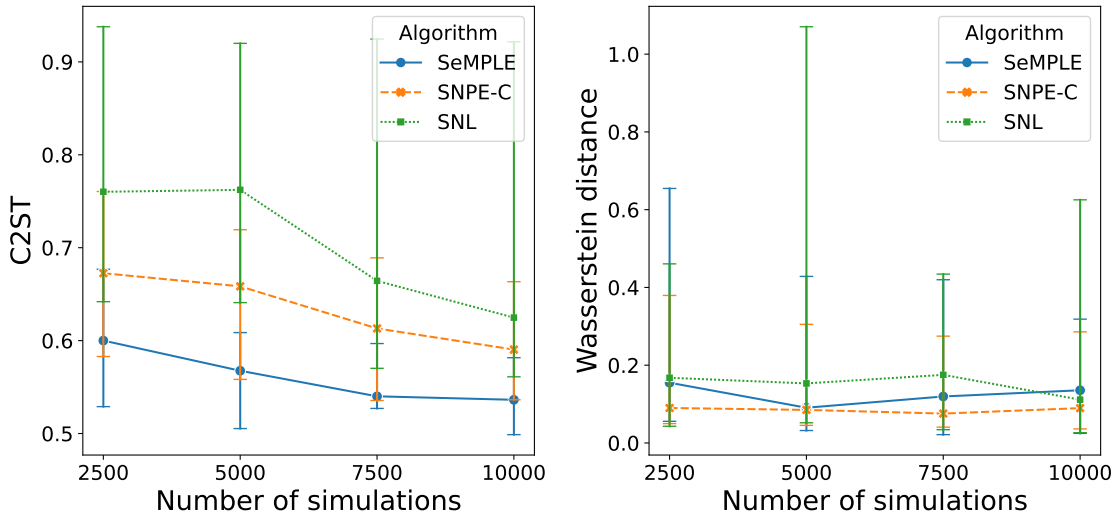


Figure 4.4: Two Moons: Median acceptance rate of the Metropolis-Hastings step of 10 SeMPLE runs with different data sets. The MCMC target refers to the choice of using the surrogate likelihood or surrogate posterior in line 14 of Algorithm 5. Error bars show min/max values.

With the insights of what SeMPLE settings seem to be the most effective, it is natural to present the performance comparison with the SNPE-C and SNL algorithms. Figure 4.5 shows the C2ST metric and the Wasserstein distance as a function of number of model simulations for SeMPLE, SNPE-C and SNL. One can immediately identify that all three algorithms manages solve the inference task and approximate the posterior distribution quite well, although the SNL results have high variance. In terms of the C2ST metric SeMPLE outperforms both SNPE-C and SNL and achieves better results with fewer model simulations. In terms of the Wasserstein distance the difference between algorithms is much smaller.



(a) **(b)**

Figure 4.5: Two Moons: Median C2ST and Wasserstein distance of 10 runs with different data sets as a function of number of model simulations. Error bars show min/max values.

As mentioned previously, performance metrics are a useful way of summarizing multiple results, but can not present all details. Figure 4.6 shows a pair plot with Kernal Density Estimate (KDE) marginals of the posterior samples of the final iteration r=4 of each of the algorithms. This is using observed data set number 1 in SBIBM. With this specific data set, SNPE-C is very close to the reference posterior sample when looking at the parameter marginals. However, the SNPE-C sample has many outliers outside of the crescent shapes of the reference posterior sample. The SeMPLE posterior sample has almost no such outliers outside of the reference posterior region and in this regard approximates the posterior shape very well. This can likely be contributed to the mixture distribution structure of SeMPLE.

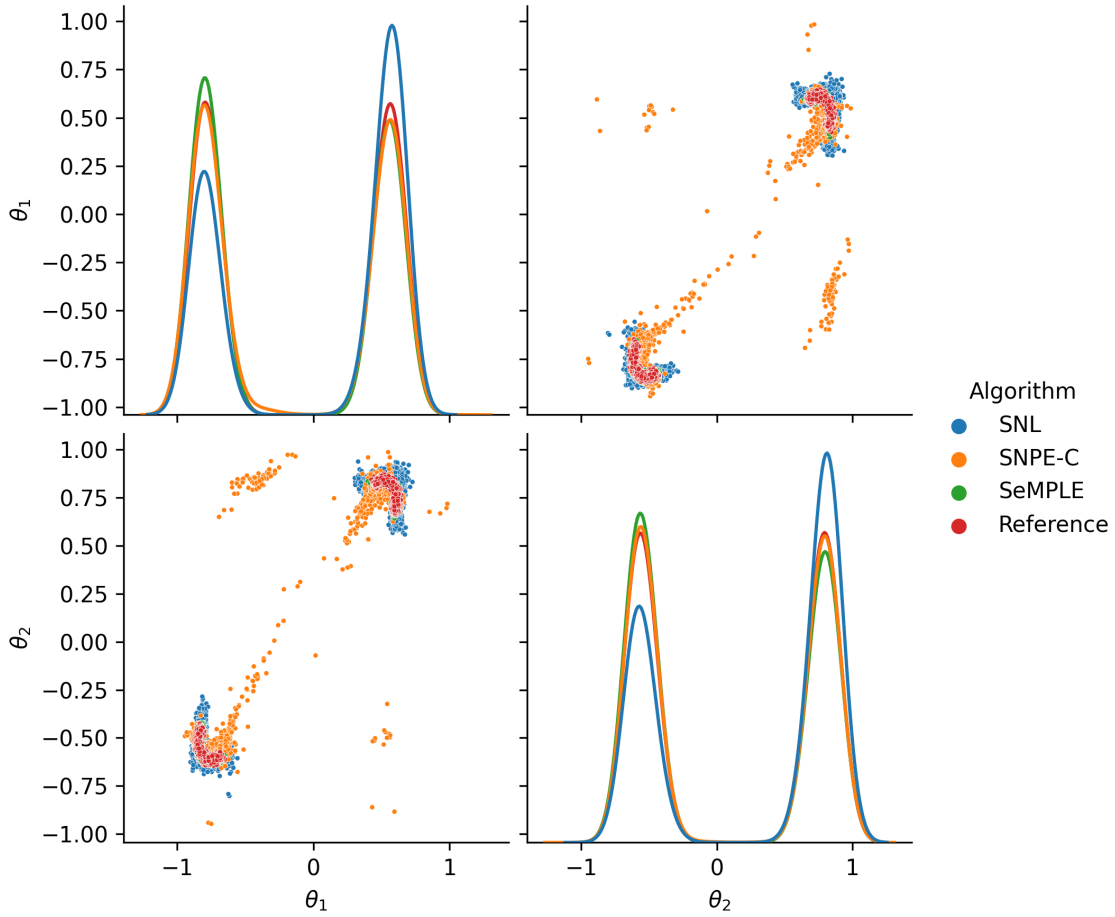


Figure 4.6: Two Moons: Pair plot of posterior samples from the last iteration (r=4) of each algorithm. Observed data set number 1 in SBIBM.

Pair plots and scatter plots colored by density of posterior samples from the final iteration r=4 of all algorithms using all 10 different SBIBM data sets can be found in the Appendix sections A.1.1 and A.1.2. The median runtimes of the algorithms can be found in Figure A.1 in the Appendix.

4.2 Multiple hyperboloid

The multiple hyperboloid model in [9] is constructed from a real sound source localization problem in audio processing, however the connection to audio processing is only illustrative. A detailed explanation of the model is provided in the supplementary material of [9] Section S4.5. In the context of this thesis, the multiple hyperboloid model serves as an example of more challenging inference task with a complex symmetrical and multimodal posterior distribution.

4.2.1 Model

Referring to the audio processing illustration the unknown parameter $\boldsymbol{\theta} \in \mathbb{R}^2$ can be interpreted as a sound source location and the two pairs of 2-dimensional parameters $\boldsymbol{m}^1 = (\boldsymbol{m}_1^1, \boldsymbol{m}_2^1)$ and $\boldsymbol{m}^2 = (\boldsymbol{m}_1^2, \boldsymbol{m}_2^2)$ can be interpreted as the location of two microphone pairs. An observation $\boldsymbol{y} \in \mathbb{R}^d$ has the likelihood function

$$p(\boldsymbol{y} \mid \boldsymbol{\theta}) = \frac{1}{2}\mathcal{S}(\boldsymbol{y}; F_{\boldsymbol{m}^1}(\boldsymbol{\theta})\mathbb{1}_d, \sigma^2 I_d, \nu) + \frac{1}{2}\mathcal{S}(\boldsymbol{y}; F_{\boldsymbol{m}^2}(\boldsymbol{\theta})\mathbb{1}_d, \sigma^2 I_d, \nu), \tag{4.5}$$

where

$$F_m(\boldsymbol{\theta}) = (||\boldsymbol{\theta} - \boldsymbol{m}_1||_2 - ||\boldsymbol{\theta} - \boldsymbol{m}_2||_2), \text{ if } \boldsymbol{m} = (\boldsymbol{m}_1, \boldsymbol{m}_2). \tag{4.6}$$

The likelihood in equation (4.5) is a mixture of two Student's t-distributions with a d-dimensional location parameter with all dimensions equal to $F_{\boldsymbol{m}^1}$, $F_{\boldsymbol{m}^2}$ respectively, a diagonal scale matrix σI_d and ν degrees of freedom. It is assumed that $\nu = 3$ and $\sigma = 0.01$. Generating data $\boldsymbol{y}$, given a parameter $\boldsymbol{\theta}$, is as simple as sampling from a mixture of Student's t-distributions. The prior is set to $\mathcal{U}(-2, 2)$ for both components of $\boldsymbol{\theta}$ with the true parameter $\boldsymbol{\theta}^* = (1.5, 1)$ as in [9]. The number of dimensions of the data $d = 10$.

4.2.2 Implementation

Figure 4.7 shows the BIC for different values of K computed by fitting surrogate likelihoods using the same prior-predictive data set $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$ with $N = 10\,000$ prior to running SeMPLE. Isotropic covariance matrices were used in GLLiM as it was found that using fully-specified covariance matrices did not improve results. The decrease in the BIC slows down significantly at K=30 which motivates a starting value of K=40 in SeMPLE. This is close to the value of K=38 in [9] selected for the same multiple hyperboloid model by BIC. The runtime to compute all the BIC values in Figure 4.7 was 270 seconds = 4.5 minutes. Note once again that the BIC computation is independent of the observed data set and only has to be performed once before running SeMPLE.

Since the multiple hyperboloid model is not implemented in SBIBM there exist no observed data sets or reference posterior samples in SBIBM. To produce an observed data set $\boldsymbol{y}_0$, the model simulator $p(\boldsymbol{y} \mid \boldsymbol{\theta}^*)$ was run with the true parameter value $\boldsymbol{\theta}^*$. Since the posterior shape varies widely depending on the true parameter value and to keep the connection to the experiment in [9], only a single such observed data set was produced. Instead, the metric results were averaged over 10 independent runs with this same observed data set. To produce a reference posterior sample corresponding to the produced observed data set, the Metropolis-Hastings algorithm was used. The sample size of the reference posterior sample was set to 10^4 to match the reference sample size in SBIBM.

As the multiple hyperboloid model is a more challenging inference task compared to the Two Moons model, the total number of model simulations when running the algorithms was set to 40 000, while the number of algorithm iterations was kept to R=4. We spread the simulations uniformly across the R=4 algorithm iterations which results in $N = N = 10\,000$.

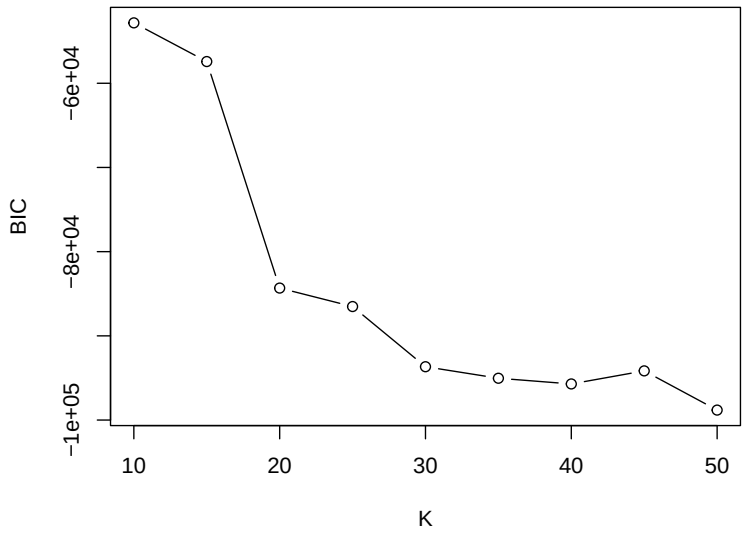
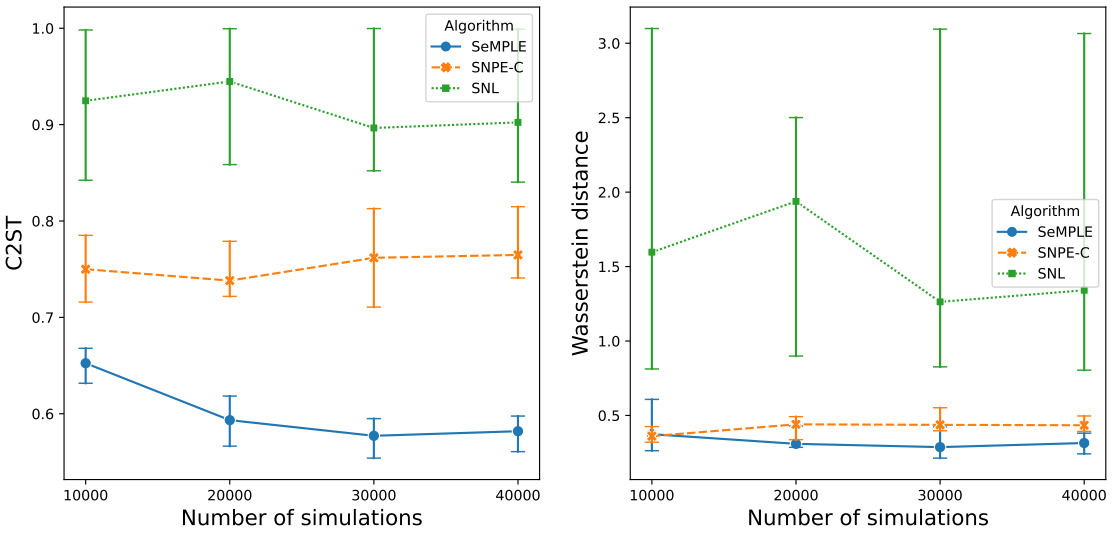


Figure 4.7: Multiple hyperboloid: BIC as a function of number of Gaussian mixture components K.

4.2.3 Results

Figure 4.8 shows the C2ST metric and the Wasserstein distance as a function of the number of model simulations for each algorithm. Regardless of metric, SNL has poor results with high variance, and it can clearly be seen in the scatter plots of each run in section A.2.1 of the Appendix that SNL completely fails to approximate the posterior in this simulation study. In terms of both performance metrics, SeMPLE performs better than SNPE-C. The SNPE-C scores in terms of the C2ST metric and Wasserstein distance actually increases sligthly with increasing number of model simulations, suggesting that SNPE-C struggles with approximating the posterior even though more model simulations are provided. The Metropolis-Hastings acceptance rate plot of the internal SeMPLE step can be found in Figure A.22 in the Appendix.

In terms of simulation efficiency, SeMPLE performs very well compared to SNPE-C and SNL. Another aspect to compare is the runtime of the algorithms. Figure 4.9 shows the elapsed time as a function of iteration number for each algorithm. The SNL results are not that relevant because of the poor inference results, but there is a considerable difference in computational time between SeMPLE and SNPE-C in the favor of SeMPLE, even after taking the 4.5 minutes of BIC computational time into consideration. In this case study, SeMPLE is outperforming both SNPE-C and SNL in terms of inference quality, simulation efficiency and runtime.



(a) (b)

Figure 4.8: Multiple hyperboloid: Median C2ST and Wasserstein distance of 10 independent runs with the same observed data set as a function of number of model simulations. Error bars show min/max values.

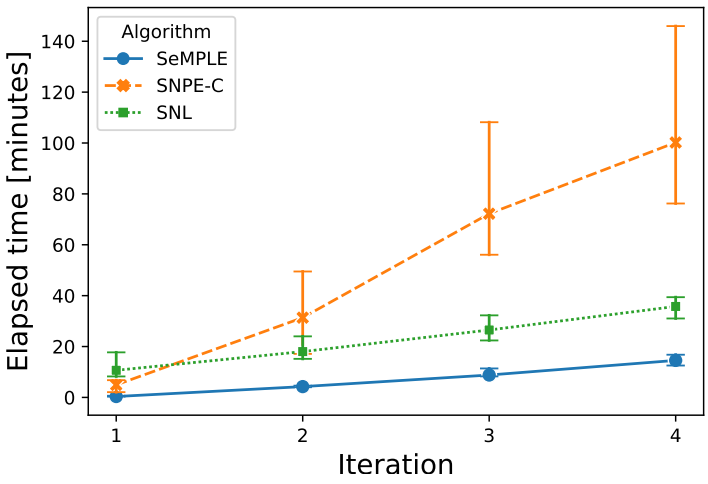


Figure 4.9: Multiple hyperboloid: Median elapsed time of 10 independent runs as a function of algorithm iteration. Error bars show min/max values.

Figure 4.10 shows a pair plot of a multiple hyperboloid posterior sample from the final iteration (r=4) of each algorithm. Note that this is only one of the 10 independent runs that make up the averaged metric results. The SeMPLE and SNPE-C posterior samples appear to be quite similar to the reference sample when looking at the scatter plots of this figure. The SeMPLE sample contains some samples near the origin that are not present in the reference sample. However, when looking at the marginals it appears that the SeMPLE sample is in fact more similar to the reference sample compared to SNPE-C and SNL. The pair plots of all repetitions

can be found in the Appendix section A.2.2.

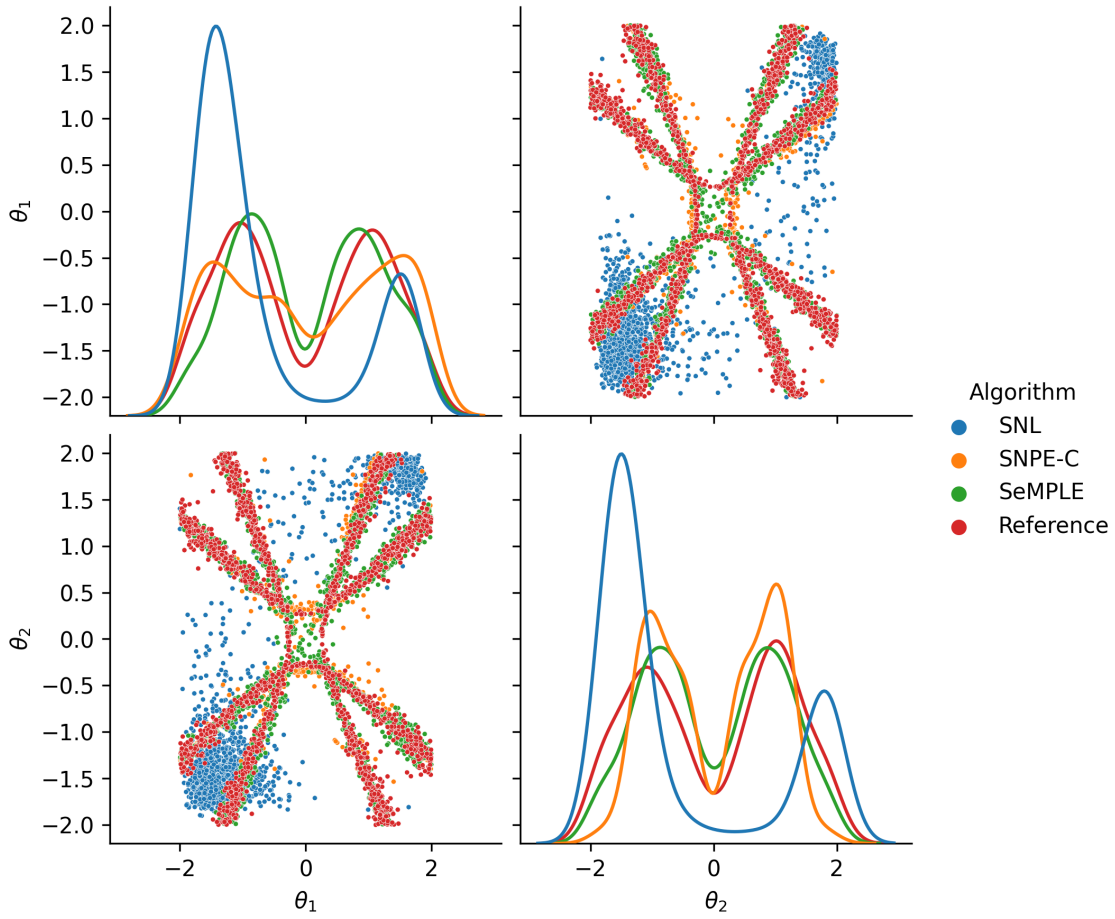


Figure 4.10: Multiple hyperboloid: Pair plot of posterior samples from the last iteration (r=4) of each algorithm. Run 2 out of 10.

4.3 Ornstein-Uhlenbeck

The Ornstein-Uhlenbeck process [24] is a stochastic process with original applications in physics to model the velocity of a Brownian particle. In this thesis it serves as an application where the data can be set to a high-dimensional time series, while the likelihood function is tractable and can be used to produce true posterior reference samples.

4.3.1 Model

The Ornstein-Uhlenbeck process X_t is defined by the stochastic differential equation (SDE)

$$dX_t = -\beta(X_t - \alpha)dt + \sigma dW_t, \tag{4.7}$$

where $\beta > 0$, $\alpha \in \mathbb{R}$, $\sigma > 0$, $X_0 = x_0$ and W_t denotes the Wiener process. The solution to equation 4.7 is

$$X_t = \alpha + (x_0 - \alpha)e^{-\beta t} + \sigma \int_0^t e^{-\beta(t-s)} dW_s. \tag{4.8}$$

The Ornstein-Uhlenbeck process has known Gaussian transition densities which enables evaluation of the exact likelihood function. This means that a reference sample from the true posterior can be produced by MCMC. For a generic time t and an arbitrary state x_0 at time $t_0 = 0$ the conditional transition density is given as

$$(X_t \,|\, X_0 = x_0) \sim \mathcal{N}(\alpha + (x_0 - \alpha)e^{-\beta t}, \frac{\sigma^2}{2\beta}(1 - e^{-2\beta t})). \tag{4.9}$$

The transition density in equation (4.9) allows exact simulation of a solution path to the Ornstein-Uhlenbeck SDE. Starting at $t_0 = 0$ one can simulate $n - 1$ further points of the solution at instants $t_1, \ldots, t_{n-1}$, letting $\Delta_i = t_i - t_{i-1}$, by using the transition recursively as following

$$(X_{t_i} \,|\, X_{t_{i-1}} = x_{t_{i-1}}) \sim \mathcal{N}(\alpha + (x_0 - \alpha)e^{-\beta \Delta_i}, \frac{\sigma^2}{2\beta}(1 - e^{-2\beta \Delta_i})), \quad i = 1, \ldots, n-1. \tag{4.10}$$

The likelihood function for the parameters $\boldsymbol{\theta} = (\alpha, \beta, \sigma)$ can due to the Markovanity of the solution of an SDE be written as

$$\mathcal{L}(x_0, \ldots, x_{n-1} \,|\, \boldsymbol{\theta}) = \prod_{i=1}^{n-1} p(x_i \,|\, x_{i-1}; \boldsymbol{\theta}), \tag{4.11}$$

where $p(x_i \,|\, x_{i-1}; \boldsymbol{\theta})$ is the transition density in equation (4.10).

The interpretation of the parameter α is the asymptotic mean. This can be seen by letting $t \to \infty$ in equation (4.9) which shows that asymptotically $X_t \sim \mathcal{N}(\alpha, \frac{\sigma^2}{2\beta})$. The parameter σ can be interpreted as the variation or the size of the noise while β can be interpreted as the growth-rate or how strongly the system react to perturbations.

4.3.2 Implementation

The fixed initial value was set to $x_0 = 0$. Uniform priors distributions according to $\alpha \sim \mathcal{U}(0, 10)$, $\beta \sim \mathcal{U}(0, 5)$ and $\sigma \sim \mathcal{U}(0, 2)$ were used with the true parameter values $\boldsymbol{\theta} = (\alpha, \beta, \sigma) = (3, 1, 0.5)$. In this study, 50 points were simulated from the process to obtain a time series of length $n = 51$ (including the fixed starting value) in each simulation. The time frame of the process was set to $[t_0, T] = [0, 10]$ and the discrete time points $t_1, \ldots, t_{n-1}$ set to be equally spaced in this interval.

Similarly to the multiple hyperboloid model a single observed data set was produced by running the simulation model with the true parameter values $\boldsymbol{\theta} = (3, 1, 0.5)$ once. Metrics are instead averaged over 10 independent algorithm runs with the same

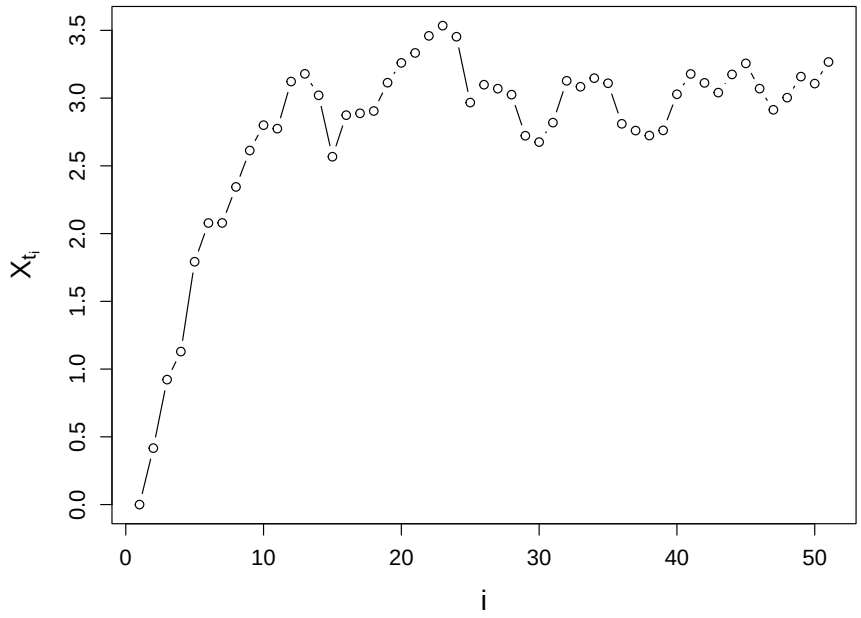


Figure 4.11: Ornstein-Uhlenbeck: Observed time series data of the process.

observed data set. The observed data set is shown in Figure 4.11. Note how the time series moves towards to the asymptotic mean $\alpha = 3$.

The covariance structure in GLLiM was set to be fully-specified with this model, because through empirical experiments, it was found that this improved the inference results substantially. This could be because the data is correlated due to the Markovanity of the time series data.

Figure 4.12 shows the BIC for different values of K, computed by fitting surrogate likelihoods using the same prior-predictive data set $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$ with $N = 10\,000$ prior to running SeMPLE. The minimum value is obtained at K=8 but because of this relatively small value, K was instead set to 15 to have some buffer to remove unnecessary mixture components if needed. The runtime to compute all BIC values in Figure 4.12 was 973 seconds $\approx$ 16 minutes. Note that the BIC computation is independent of the observed data set and only has to be performed once before running SeMPLE.

4.3.3 Results

Figure 4.13 shows the C2ST metric and Wasserstein distance as a function of number of model simulations for each algorithm. At iteration r=2 (20 000 model simulations) SNL consistently shows an unexpected behaviour where the posterior sample is clearly worse than in the first iteration (r=1). This happened in some scenarios with SNL where the number of simulations in the first iteration is low enough. Apart from this, SNL scores best in both metrics after the full 40 000 model simulations (R=4 algorithm iterations). In terms of the C2ST metric it clearly performs the best, but the difference compared to SeMPLE is much smaller in terms of the Wasserstein

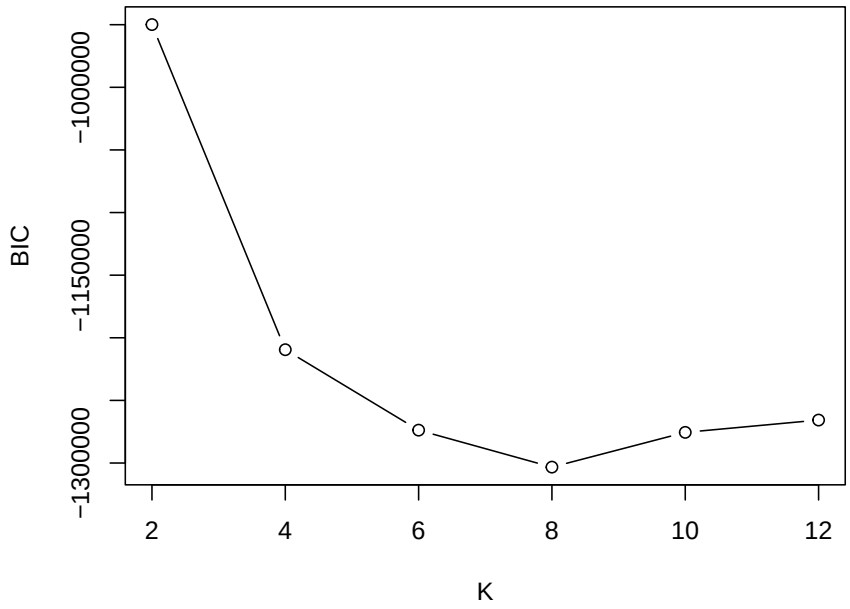


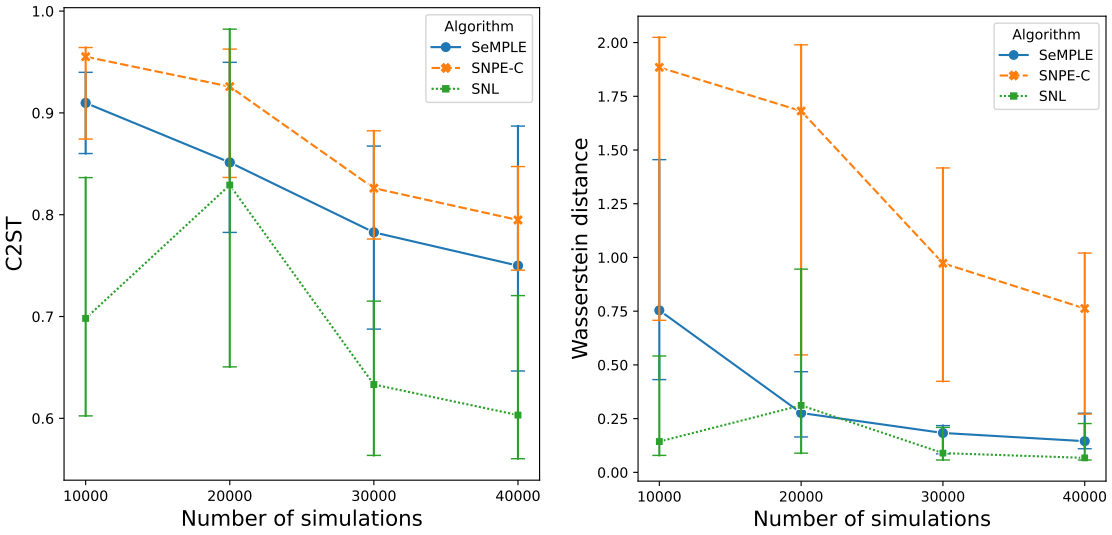
Figure 4.12: Ornstein-Uhlenbeck: BIC as a function of number of Gaussian mixture components K.

distance. SeMPLE obtains better metric values than SNPE-C with the difference between the algorithms being larger in terms of the Wasserstein distance compared to the C2ST metric. The Ornstein-Uhlenbeck process could be a simulation model where it is easier to estimate the likelihood compared to the posterior, which could explain the outcome of metric scores since both SNL and SeMPLE uses a surrogate likelihood to obtain posterior samples. Based on the C2ST metric score, one could assume that providing more model simulations or running the algorithms for more iterations could improve the inference quality.

The increasing acceptance rate of the Metropolis-Hastings step of SeMPLE in Figure 4.14 suggests that SeMPLE is converging to a surrogate likelihood and surrogate posterior in the later algorithm iterations.

Figure 4.15 shows that the runtime of SeMPLE is much lower compared to both SNL and SNPE-C, even after taking the 16 minutes of BIC computations into account. Especially SNPE-C has much longer runtimes with this model, combined with the worst performance metric results out of all three algorithms. However, SNL interesting in comparison to SeMPLE when we consider the added time of the BIC computations for SeMPLE and the promising performance metric results of SNL.

Pair plots of the posterior sample from the last algorithm iteration (r=4) of all 10 algorithm runs, as well as the Markov chains of the Metropolis-Hastings output can be found in the Appendix section A.3.



(a) **(b)**

Figure 4.13: Ornstein-Uhlenbeck: Median C2ST and Wasserstein distance of 10 independent runs with the same observed data set as a function of number of model simulations. Error bars show min/max values.

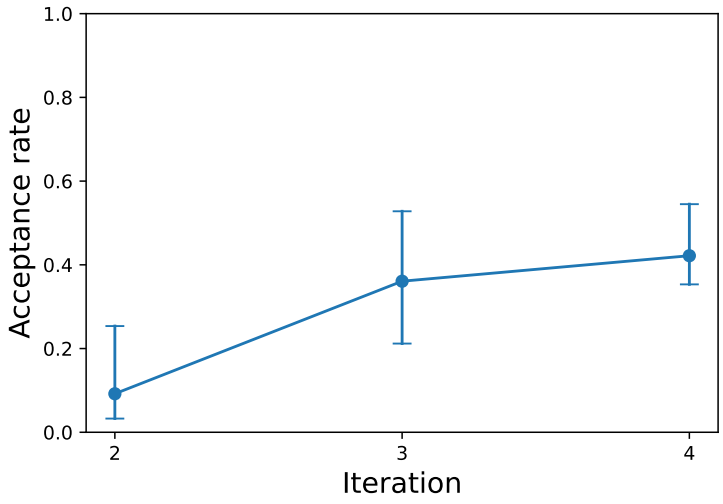


Figure 4.14: Ornstein-Uhlenbeck: Median acceptance rate of the Metropolis-Hastings step of 10 independent SeMPLE runs with the same data set. Error bars show min/max values.

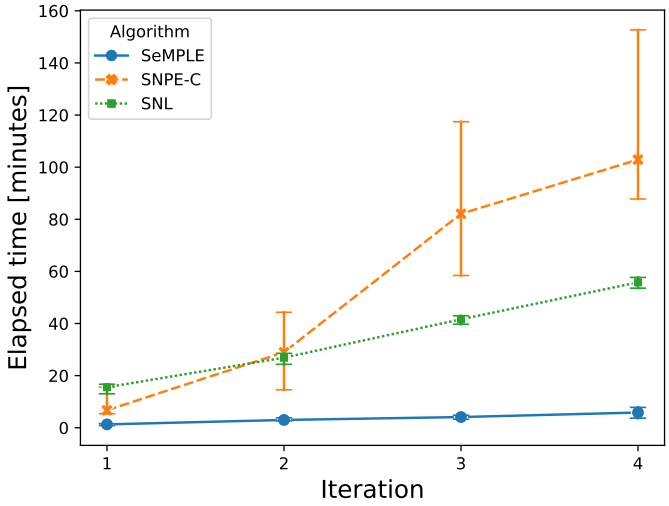


Figure 4.15: Ornstein-Uhlenbeck: Median elapsed time of 10 independent runs as a function of algorithm iteration. Error bars show min/max values.

4.4 SLCP

The Simple Likelihood Complex Posterior (SLCP) model [19] [12] is a challenging inference task that is designed to have a simple likelihood function but a complex posterior distribution. In [19], it is used as an example where SNL performs better than posterior estimation methods. In this thesis, it serves as an example of a model where the likelihood function is supposed to be considerably easier to approximate compared to the posterior distribution.

4.4.1 Model

Given parameters $\boldsymbol{\theta} \in \mathbb{R}^5$, the data $\boldsymbol{y} \in \mathbb{R}^8$ can be generated according to the following equations:

$$\boldsymbol{m}_{\boldsymbol{\theta}} = [\theta_1, \theta_2] \tag{4.12}$$
$$s_1 = \theta_3^2 \tag{4.13}$$
$$s_2 = \theta_4^2 \tag{4.14}$$
$$\rho = \tanh(\theta_5) \tag{4.15}$$
$$\boldsymbol{S}_{\boldsymbol{\theta}} = \begin{bmatrix} s_1^2 & \rho s_1 s_2 \\ \rho s_1 s_2 & s_2^2 \end{bmatrix} \tag{4.16}$$
$$\boldsymbol{y}_j \sim \mathcal{N}(\boldsymbol{m}_{\boldsymbol{\theta}}, \boldsymbol{S}_{\boldsymbol{\theta}}) \quad \text{for} \quad j = 1, \ldots, 4 \tag{4.17}$$
$$\boldsymbol{y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_4]. \tag{4.18}$$

The data $\boldsymbol{y} \in \mathbb{R}^8$ is a set of four two-dimensional points independently sampled from a Gaussian likelihood. The model is (pairwise) multimodal by construction and the variance $\boldsymbol{S}_{\boldsymbol{\theta}}$ is a nonlinear function of $\boldsymbol{\theta}$. The prior distribution was set to $\mathcal{U}(-3, 3)$ independently on each component of $\boldsymbol{\theta}$ as in [19].

4.4.2 Implementation

Figure 4.16 shows the BIC for different values of K computed by fitting surrogate likelihoods using the same prior-predictive data set $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$ with $N = 20,000$ prior to running SeMPLE. The minimum value is obtained at K=10, but similarly to the case with the Ornstein-Uhlenbeck model, K was set to 15 to have some buffer to remove mixture components with small posterior mixture probabilities. The runtime to compute all BIC values in Figure 4.16 was 276 seconds $\approx$ 5 minutes. Note that the BIC computation is independent of the observed data set and only has to be performed once before running SeMPLE.

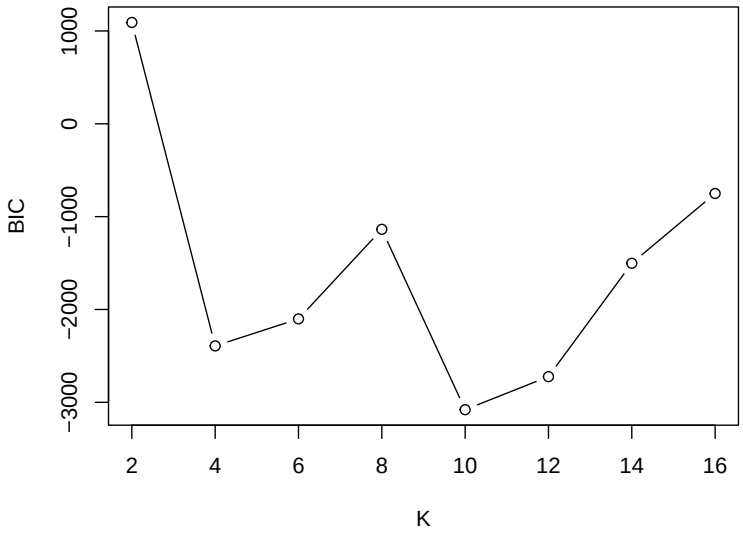


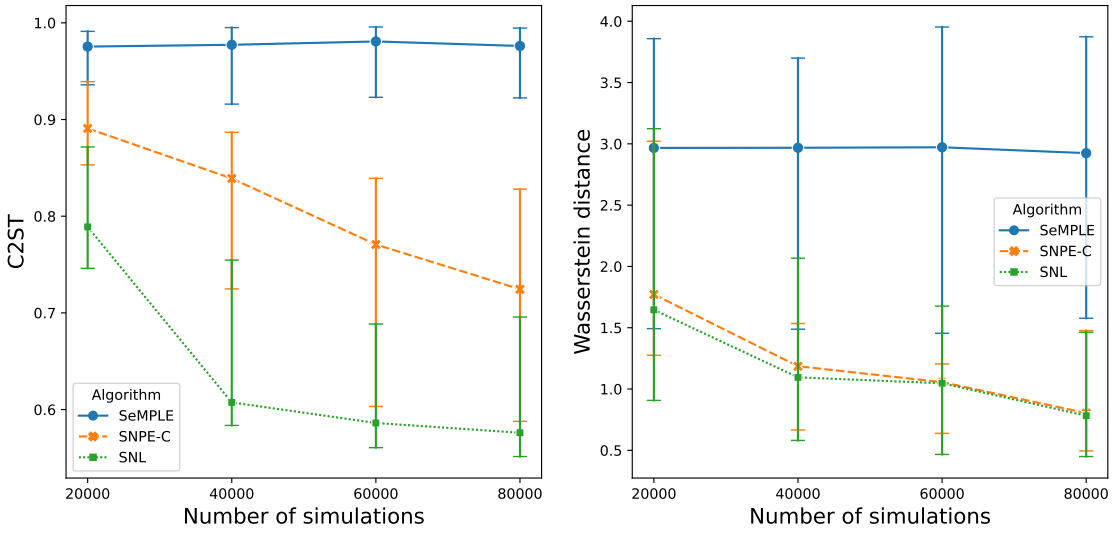
Figure 4.16: SLCP: BIC as a function of number of Gaussian mixture components K.

The three algorithms were run with the 10 different observed data sets and corresponding posterior reference samples provided in SBIBM. Since the initial inference results with SeMPLE were poor, the total number of model simulations was increased to 80 000 to show the results with a larger budget of model simulations. The number of algorithm iterations was set to R=4. The covariance structure in GLLiM was found to not have any considerable impact on the result with this model and was set to be isotropic.

4.4.3 Results

Figures 4.17 shows the C2ST metric and Wasserstein distance as a function of number of model simulations for each algorithm. While SNL and SNPE-C both perform relatively well, SeMPLE obtains poor results in terms of both metrics. It seems like SeMPLE is not able to learn much of the posterior or likelihood even though more model simulations are provided. This could indicate that GLLiM struggles

to approximate the likelihood and posterior with this specific model. We expect SNL to perform better than SNPE-C, since the simple likelihood should be easier to approximate compared to the complex posterior, which seems to be the case. The difference between SNL and SNPE-C is much larger in terms of the C2ST metric compared to the Wasserstein distance where the results are more similar.



(a) **(b)**

Figure 4.17: SLCP: Median C2ST and Wasserstein distance of 10 runs with different data sets as a function of number of model simulations. Error bars show min/max values.

The acceptance rates in Figure 4.18 are surprisingly high considering the poor inference results with SeMPLE. However, the acceptance rates remains relatively constant with increasing algorithm iterations, which suggests that the algorithm is not converging or learning the likelihood or posterior.

The computational time displayed in Figure 4.19 shows once again that SeMPLE is much faster to run compared to SNPE-C and SNL. However, this is not of much use in this case when the inference quality of SeMPLE is very poor.

Pair plots of posterior samples from the final iteration r=4 of all algorithms, using all 10 different SBIBM data sets, can be found in the Appendix section A.4.

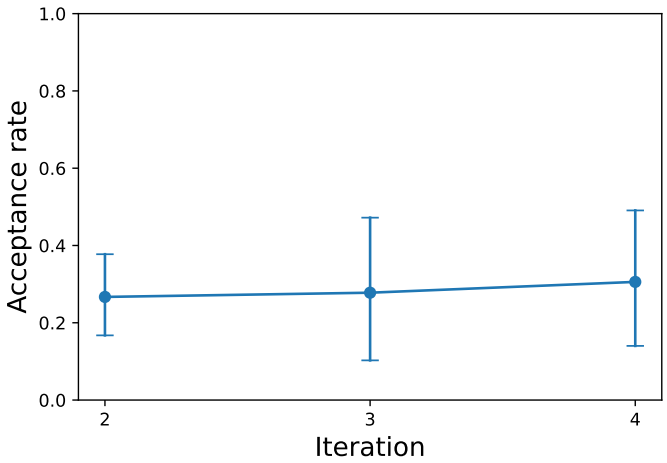


Figure 4.18: SLCP: Median acceptance rate of the Metropolis-Hastings step of 10 SeMPLE runs with different data sets. Error bars show min/max values.

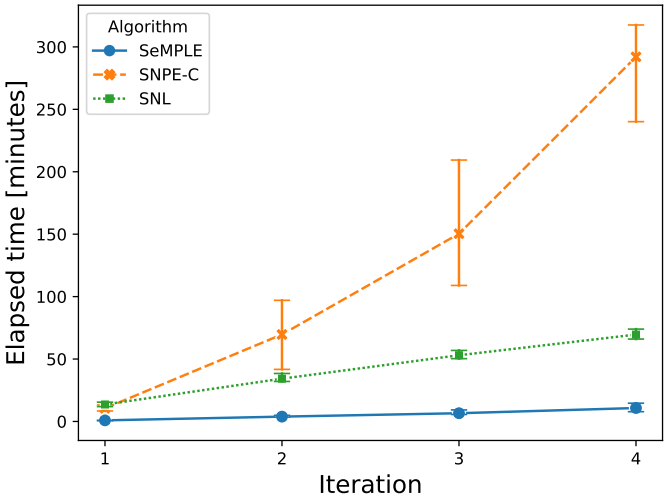


Figure 4.19: SLCP: Median elapsed time of 10 runs with different data sets as a function of number of algorithm iterations. Error bars show min/max values.

5

Conclusion

The simulation studies show that SeMPLE performs well compared to state-of-art methods such as SNPE-C and SNL in a variety of different inference tasks. In some scenarios even outperforming SNPE-C and SNL in both simulation efficiency and runtime. In many cases the EM procedure in GLLiM seems to be powerful enough to provide accurate likelihood and posterior estimations, while in all the simulations studies considered SeMPLE was significantly faster to run compared to SNPE-C and SNL.

Apart from the promising results in the Two Moons, Hyperboloid and Ornstein-Uhlenbeck simulation studies, SeMPLE did not perform well in the SLCP inference task. There could be a number of possible explanations to this as there were no signs of improved metric results with increasing number of model simulations. This could be due to the GLLiM procedure having problems estimating the likelihood or posterior due to the design of the model. Recall that the SLCP model is designed to have a complex posterior and simple likelihood and because SeMPLE relies on estimating both, this might cause problems. A further investigation by someone having deeper knowledge in GLLiM would probably be required to fully understand why SeMPLE does not perform well in the SLCP inference task.

A general question is weather it is preferable to learn the likelihood or posterior. Depending on the model considered the posterior or likelihood might be easier to approximate. A likelihood approximation can be reused with different priors, but on the other hand an additional MCMC step is required to obtain a posterior sample, which increases the computational cost. SeMPLE provides the flexibility of targeting either the surrogate likelihood times the prior or the corrected surrogate posterior in the MCMC step, but it was found that using the likelihood approximation in general provided better inference results.

With some of the models it was found that a sufficiently large number of parameter and simulation pairs N was needed in the SeMPLE iteration r=0. If this number was not high enough the acceptance rate of the first Metropolis-Hastings iteration (r=2) would be very low with many samples being identical. This then leads to internal matrix inversion problems in the next GLLiM fit as the input data is heavily correlated. This lower limit of prior predictive samples varied between models and could often be resolved by adjusting other parameters such as the number of mixture components K. Still, it could be a problem if there exists a lower limit of first iteration model simulations N under which the SeMPLE algorithm can not complete. With

SNPE-C and SNL, no such problems occurred even with very low number of model simulations. However, with a very low number of model simulations the quality of the resulting inference is usually very low.

SeMPLE is still a new algorithm with room for improvements and modifications. For example, the xLLiM package provides the possibility of using a mixture of Student's t-distributions instead of a mixture of Gaussians. This could be beneficial if the likelihood or posterior is heavy-tailed. It could be interesting to see if Student's t mixtures are more expressive and better at handling more disperse observations compared to Gaussian mixtures. Experiments with different mixture distributions has not been conducted and is left as a question of further research.

In the Ornstein-Uhlenbeck model, the length of the time series and dimension of the data was 51, showing that SeMPLE works well with higher dimensional data compared to the other models tested. However, if the dimension of the observed data set $d \gg 51$, SeMPLE would not work because of numerical issues with inversion of large matrices in the internal GLLiM computations. One way of dealing with this would be to consider summary statistics of the data to reduce the dimensionality of the data and use these in place of the raw data. This raises the question of how to create informative summary statistics of the data prior to starting SeMPLE, something that is also left as a question of further research.

Bibliography

[1] M A Beaumont, W Y Zhang, and D J Balding. "Approximate Bayesian computation in population genetics". English. In: *Genetics* 162.4 (2002). ID number: ISI:000180502300043, pp. 2025–2035. ISSN: 0016-6731.

[2] Mark A. Beaumont et al. "Adaptive approximate Bayesian computation". In: *Biometrika* 96.4 (Oct. 2009), pp. 983–990. ISSN: 0006-3444. DOI: 10.1093/biomet/asp052. eprint: https://academic.oup.com/biomet/article-pdf/96/4/983/588237/asp052.pdf. URL: https://doi.org/10.1093/biomet/asp052.

[3] Christopher M. Bishop. *Mixture density networks*. English. WorkingPaper. Aston University, 1994.

[4] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. "The frontier of simulation-based inference". In: *Proceedings of the National Academy of Sciences* 117 (May 2020), p. 201912789. DOI: 10.1073/pnas.1912789117.

[5] Antoine Deleforge, Florence Forbes, and Radu Horaud. "High-dimensional regression with gaussian mixtures and partially-latent response variables". In: *Statistics and Computing* 25.5 (Mar. 2014), pp. 893–911. DOI: 10.1007/s11222-014-9461-5. URL: https://doi.org/10.1007%5C%2Fs11222-014-9461-5.

[6] Conor Durkan, Iain Murray, and George Papamakarios. "On Contrastive Learning for Likelihood-free Inference". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 2771–2781. URL: https://proceedings.mlr.press/v119/durkan20a.html.

[7] Conor Durkan et al. "Neural Spline Flows". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/7ac71d433f282034e088473244df8c02-Paper.pdf.

[8] Rémi Flamary et al. "POT: Python Optimal Transport". In: *Journal of Machine Learning Research* 22.78 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-451.html.

[9] Florence Forbes et al. "Summary Statistics and Discrepancy Measures for Approximate Bayesian Computation via Surrogate Posteriors". In: *Statistics and Computing* 32.5 (Oct. 2022). ISSN: 0960-3174. DOI: 10.1007/s11222-022-10155-6. URL: https://doi.org/10.1007/s11222-022-10155-6.

[10] Jerome Friedman. "On multivariate goodness-of-fit and two-sample testing". In: *Conference on Statistical Problems in Particle Physics, Astrophysics and Cosmology*. 2004.

[11] Charles J. Geyer and Leif T. Johnson. *mcmc: Markov Chain Monte Carlo*. R package version 0.9-7. 2020. URL: https://CRAN.R-project.org/package=mcmc.

[12] David S. Greenberg, Marcel Nonnenmacher, and Jakob H. Macke. *Automatic Posterior Transformation for Likelihood-Free Inference*. 2019. DOI: 10.48550/ARXIV.1905.07488. URL: https://arxiv.org/abs/1905.07488.

[13] L. V. Kantorovich. "Mathematical Methods of Organizing and Planning Production". In: *Management Science* 6.4 (1960), pp. 366–422. DOI: 10.1287/mnsc.6.4.366. eprint: https://doi.org/10.1287/mnsc.6.4.366. URL: https://doi.org/10.1287/mnsc.6.4.366.

[14] David Lopez-Paz and Maxime Oquab. "Revisiting Classifier Two-Sample Tests". In: *5th International Conference on Learning Representations, ICLR*. 2017.

[15] Jan-Matthis Lueckmann et al. "Benchmarking Simulation-Based Inference". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Ed. by Arindam Banerjee and Kenji Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, 13–15 Apr 2021, pp. 343–351. URL: https://proceedings.mlr.press/v130/lueckmann21a.html.

[16] Jean-Michel Marin et al. "Approximate Bayesian computational methods". In: *Statistics and Computing* 22.6 (2012), pp. 1167–1180. URL: https://arxiv.org/abs/1101.0955.

[17] George Papamakarios and Iain Murray. *Fast ϵ-free Inference of Simulation Models with Bayesian Conditional Density Estimation*. 2016. DOI: 10.48550/ARXIV.1605.06376. URL: https://arxiv.org/abs/1605.06376.

[18] George Papamakarios, Theo Pavlakou, and Iain Murray. "Masked Autoregressive Flow for Density Estimation". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/6c1da886822c67822bcf3679d04369fa-Paper.pdf.

[19] George Papamakarios, David Sterratt, and Iain Murray. "Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows". In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. Ed. by Kamalika Chaudhuri and Masashi Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, 16–18 Apr 2019, pp. 837–848. URL: https://proceedings.mlr.press/v89/papamakarios19a.html.

[20] Emeline Perthame et al. *xLLiM: High Dimensional Locally-Linear Mapping*. R package version 2.2.1. 2022.

[21] J K Pritchard et al. "Population growth of human Y chromosomes: a study of Y chromosome microsatellites." In: *Molecular Biology and Evolution* 16.12 (Dec. 1999), pp. 1791–1798. ISSN: 0737-4038. DOI: 10.1093/oxfordjournals.molbev.a026091. eprint: https://academic.oup.com/mbe/article-pdf/16/12/1791/9592173/mbe1791.pdf. URL: https://doi.org/10.1093/oxfordjournals.molbev.a026091.

[22] Gideon Schwarz. "Estimating the Dimension of a Model". In: *The Annals of Statistics* 6.2 (1978), pp. 461–464. DOI: 10.1214/aos/1176344136. URL: https://doi.org/10.1214/aos/1176344136.

[23] Alvaro Tejero-Cantero et al. "sbi: A toolkit for simulation-based inference". In: *Journal of Open Source Software* 5.52 (2020), p. 2505. DOI: 10.21105/joss.02505. URL: https://doi.org/10.21105/joss.02505.

[24] G. E. Uhlenbeck and L. S. Ornstein. "On the Theory of the Brownian Motion". In: *Phys. Rev.* 36 (5 Sept. 1930), pp. 823–841. DOI: 10.1103/PhysRev.36.823. URL: https://link.aps.org/doi/10.1103/PhysRev.36.823.

A

Appendix 1

This appendix contains figures and is divided into four sections, one for each simulation study.

A.1 Two Moons

Figure A.1 shows the median algorithm runtime of 10 runs with different data sets as a function of algorithm iteration.

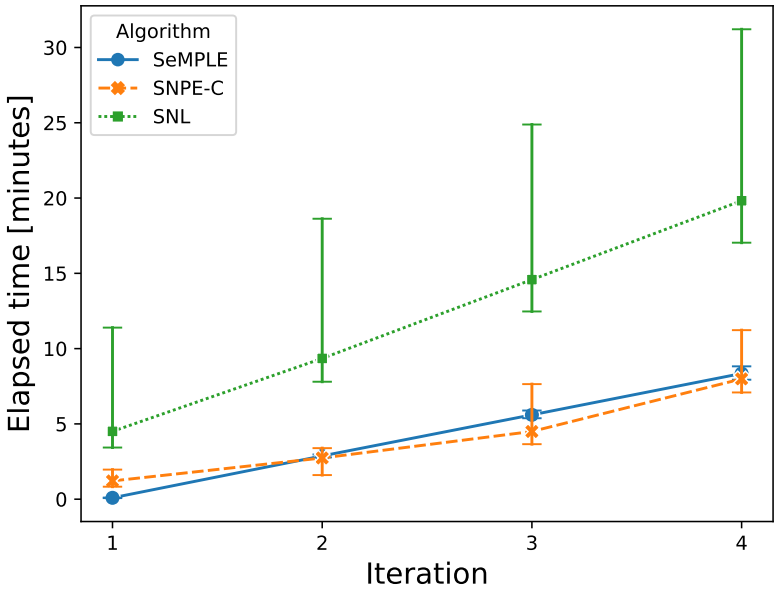


Figure A.1: Median elapsed time of 10 runs with different data sets as a function of algorithm iteration. Error bars show min/max values.

A.1.1 Density scatter plots

Figures A.2 - A.11 shows scatter plots colored by density of the Two Moons posterior sample from the last iteration (r=4) of each algorithm for each of the 10 observed data sets in SBIBM.

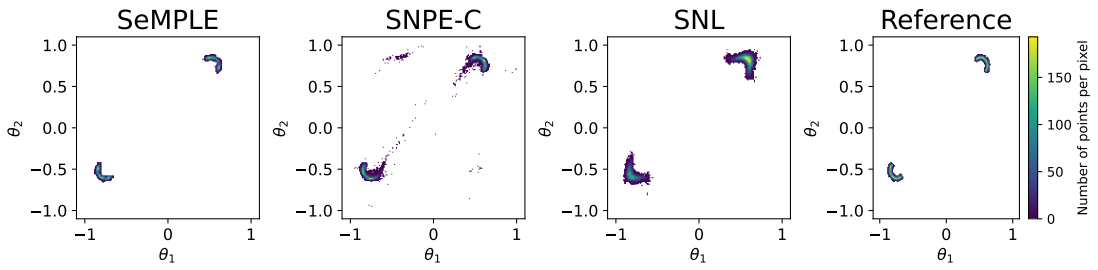


Figure A.2: Two Moons: Density scatter plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 1 in SBIBM.

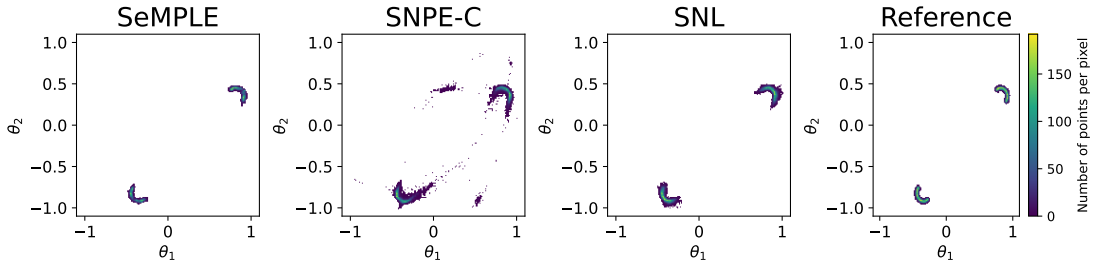


Figure A.3: Two Moons: Density scatter plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 2 in SBIBM.

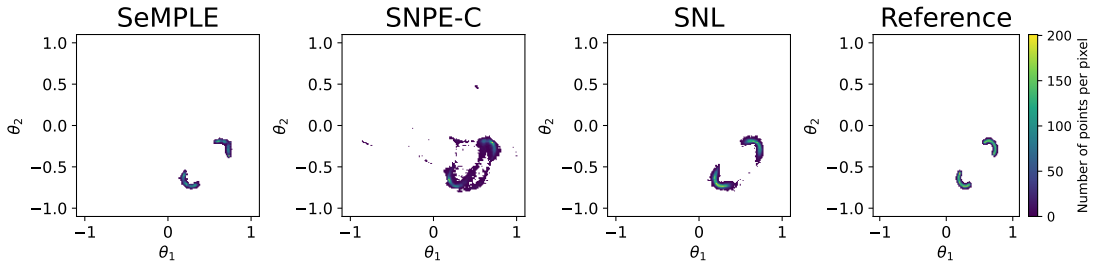


Figure A.4: Two Moons: Density scatter plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 3 in SBIBM.

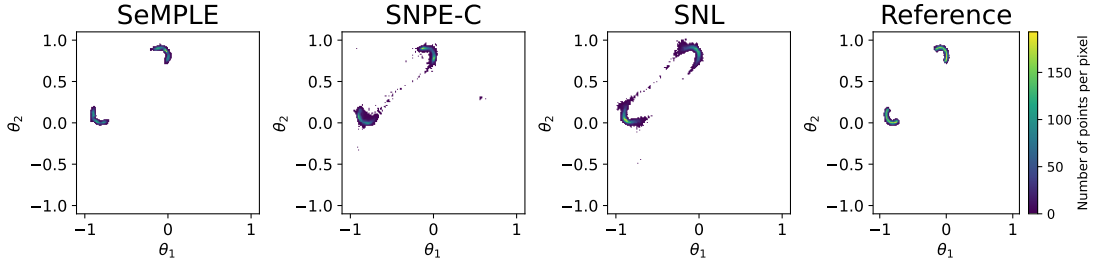


Figure A.5: Two Moons: Density scatter plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 4 in SBIBM.

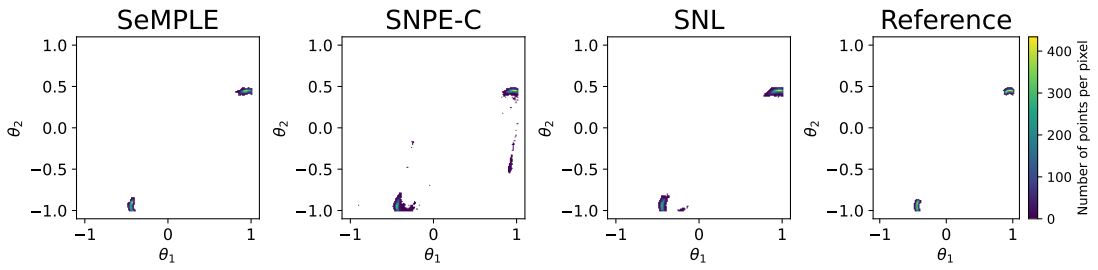


Figure A.6: Two Moons: Density scatter plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 5 in SBIBM.

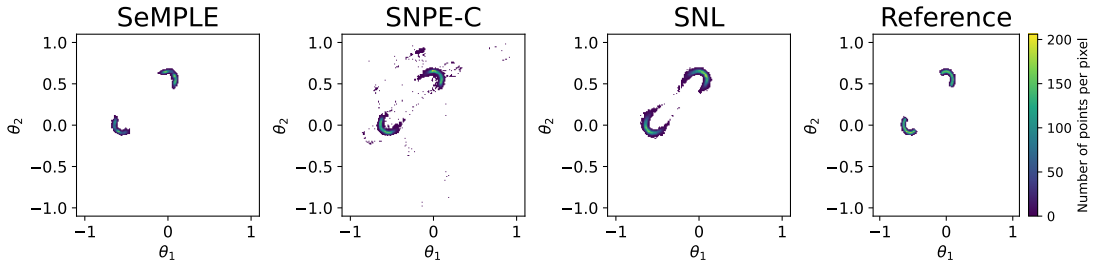


Figure A.7: Two Moons: Density scatter plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 6 in SBIBM.

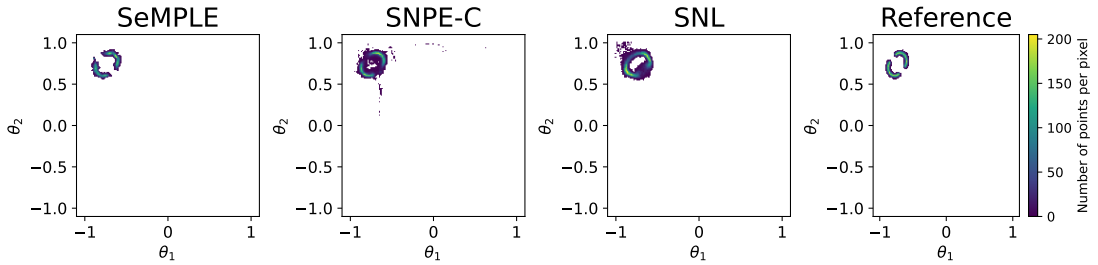


Figure A.8: Two Moons: Density scatter plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 7 in SBIBM.

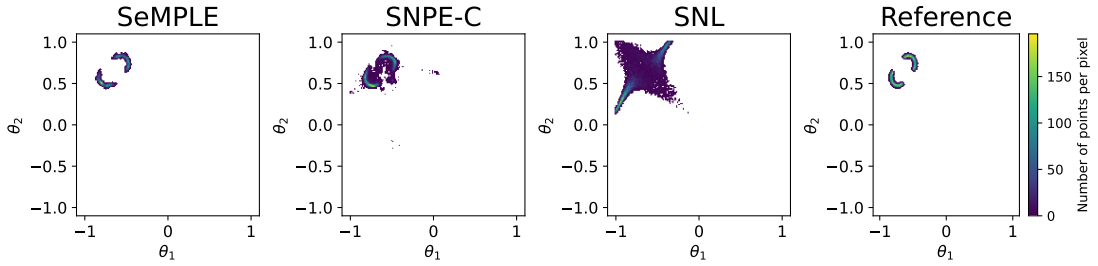


Figure A.9: Two Moons: Density scatter plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 8 in SBIBM.

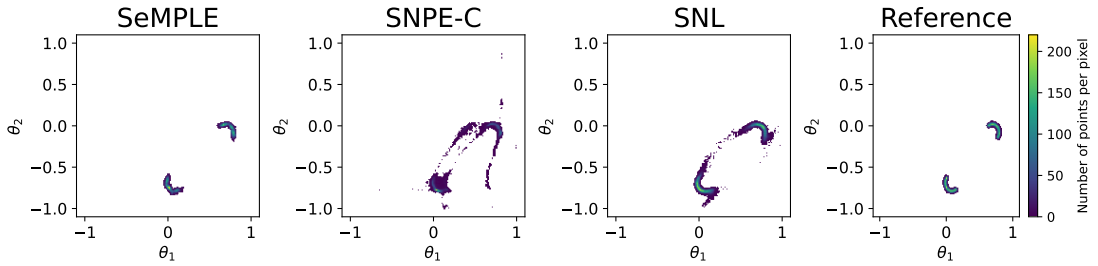


Figure A.10: Two Moons: Density scatter plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 9 in SBIBM.

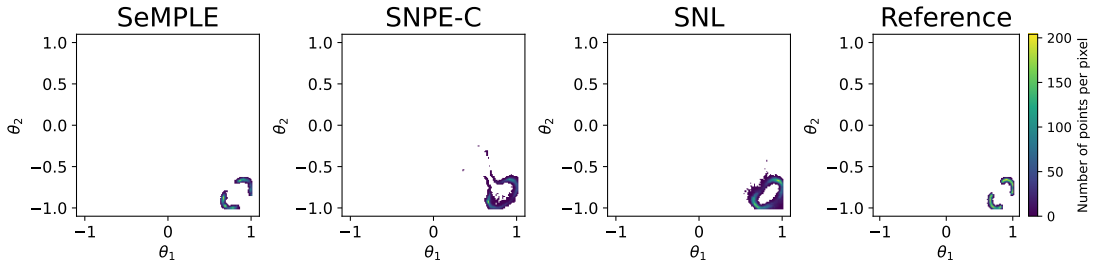


Figure A.11: Two Moons: Density scatter plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 10 in SBIBM.

A.1.2 Pair plots

Figures A.12 - A.21 shows pair plots of posterior samples from the last iteration (r=4) of each algorithm for each of the 10 observed data sets in SBIBM.

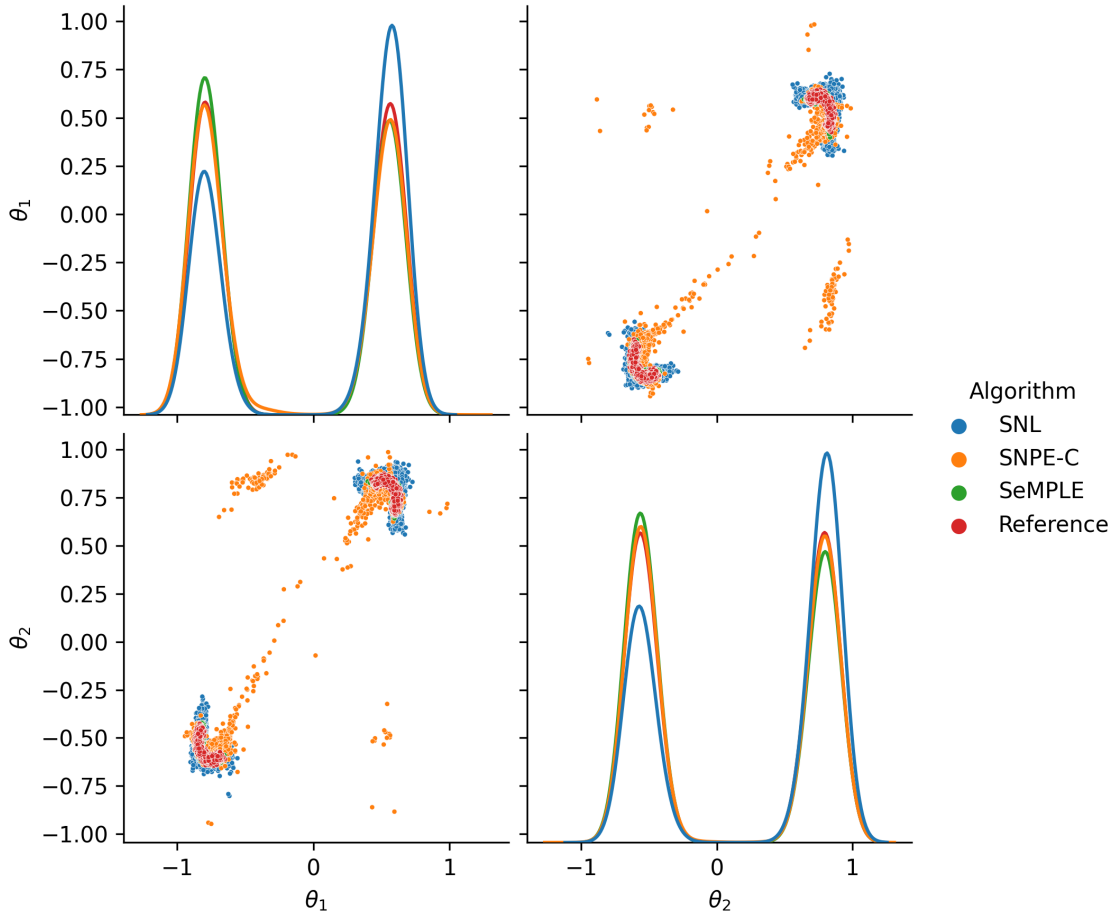


Figure A.12: Two Moons: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 1 in SBIBM.

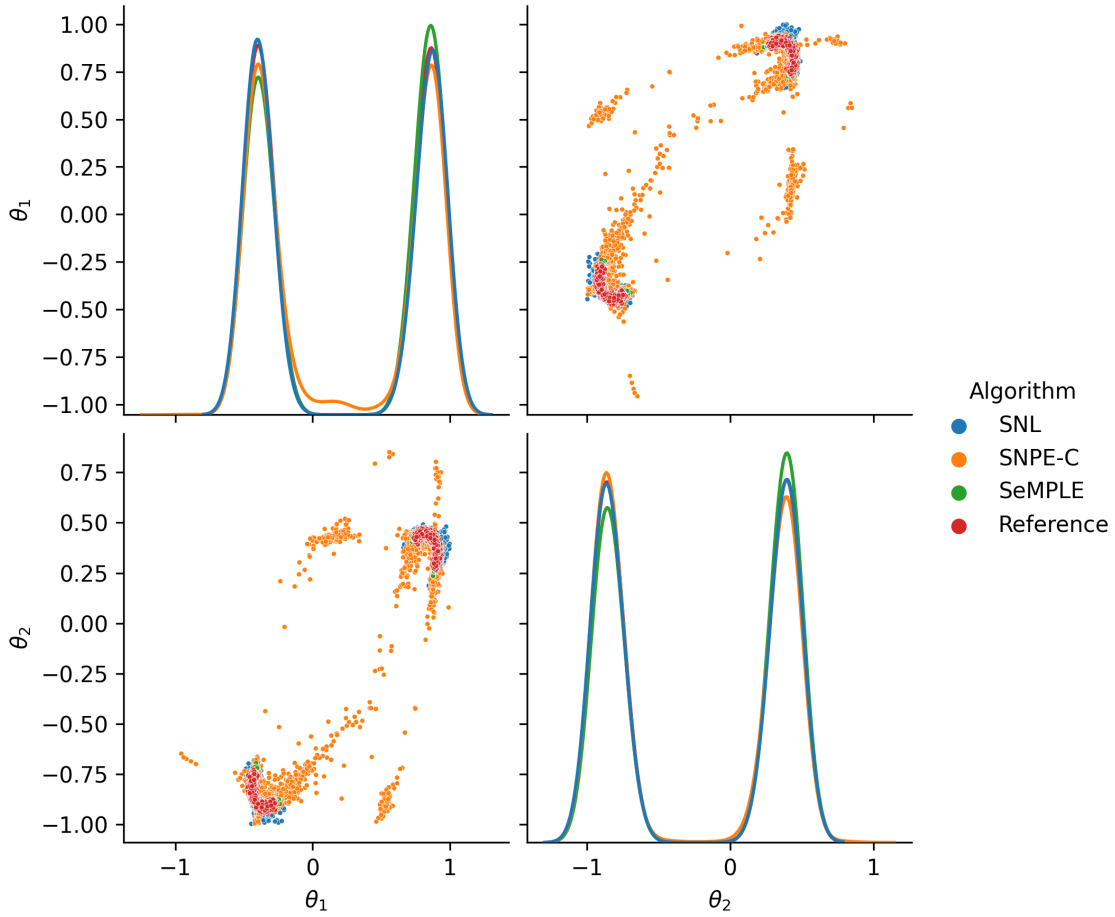


Figure A.13: Two Moons: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 2 in SBIBM.

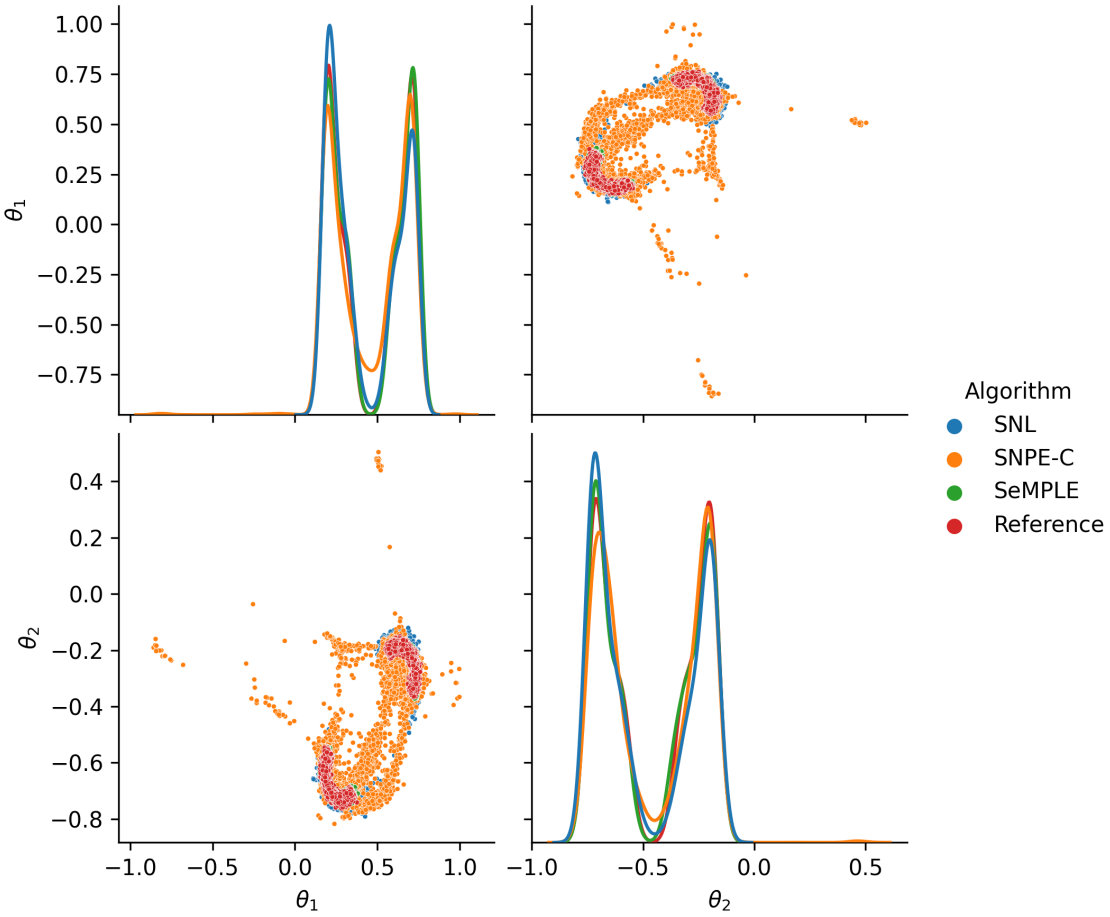


Figure A.14: Two Moons: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 3 in SBIBM.

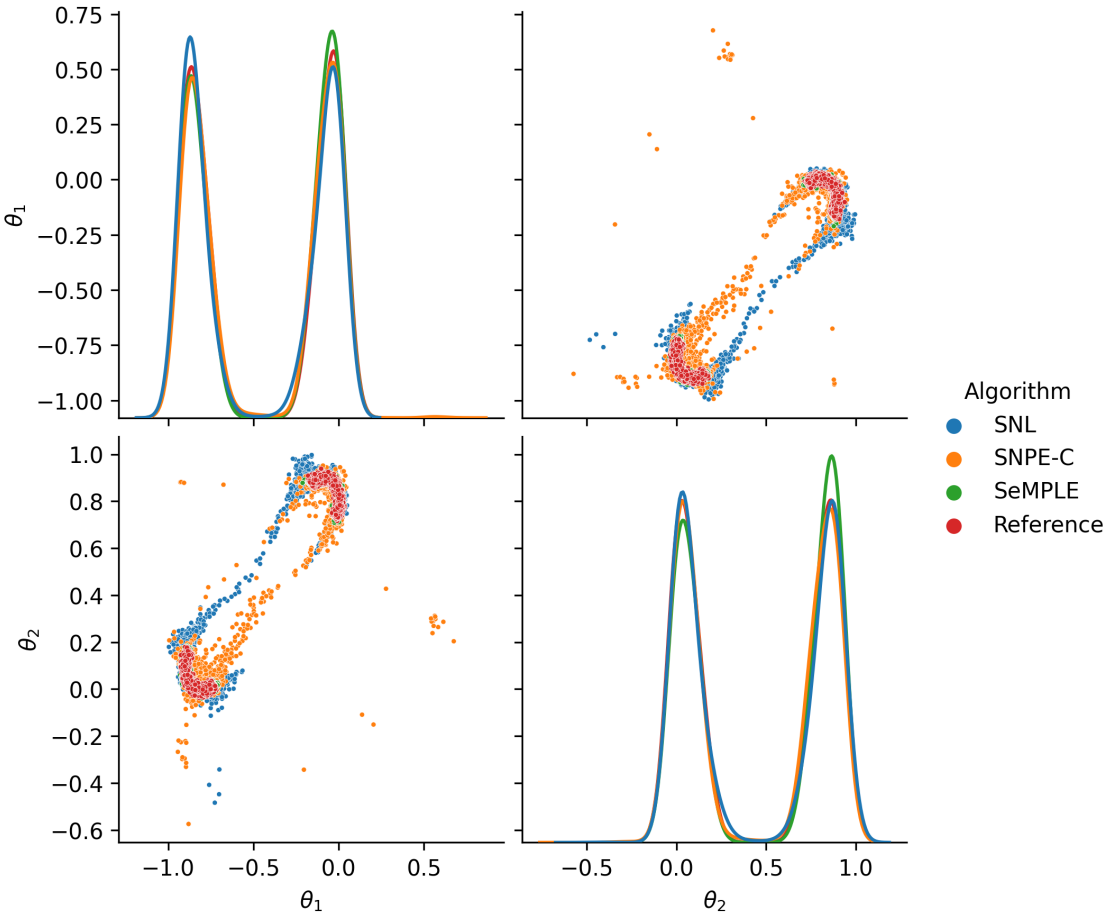


Figure A.15: Two Moons: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 4 in SBIBM.

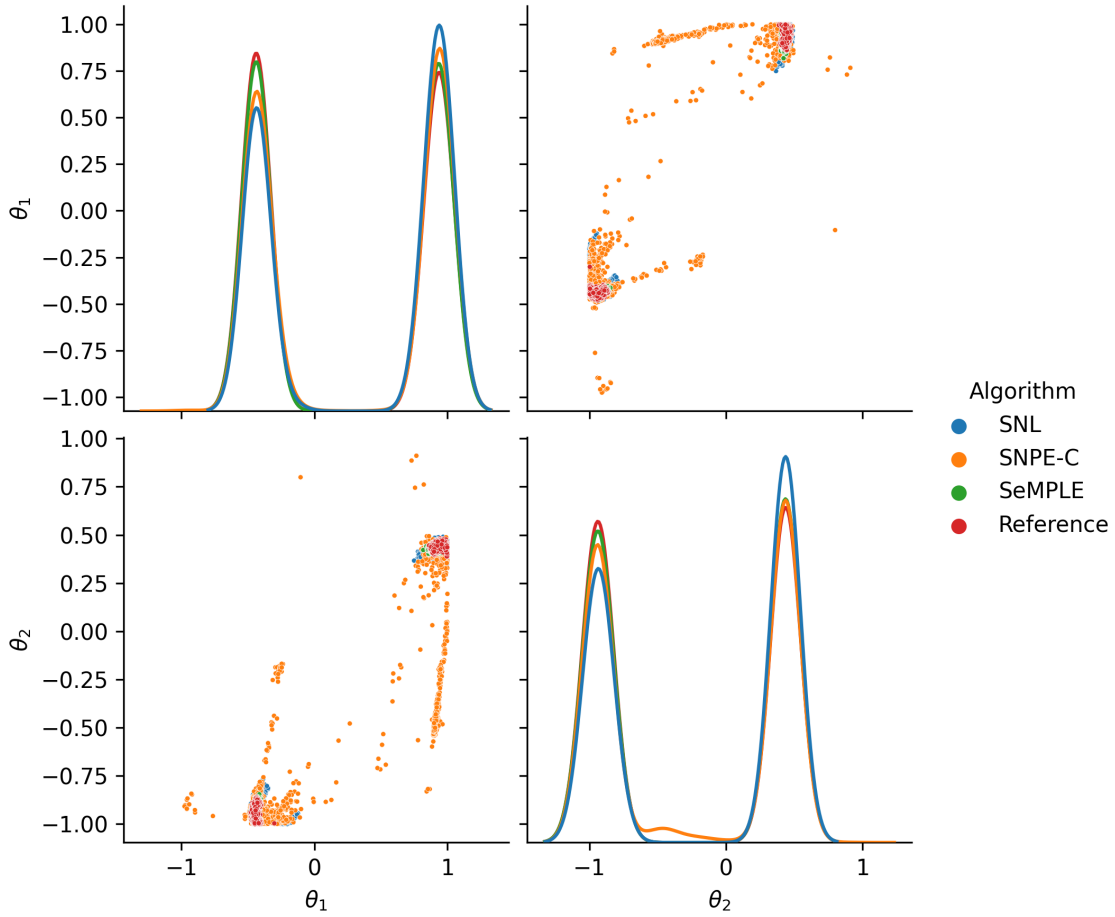


Figure A.16: Two Moons: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 5 in SBIBM.

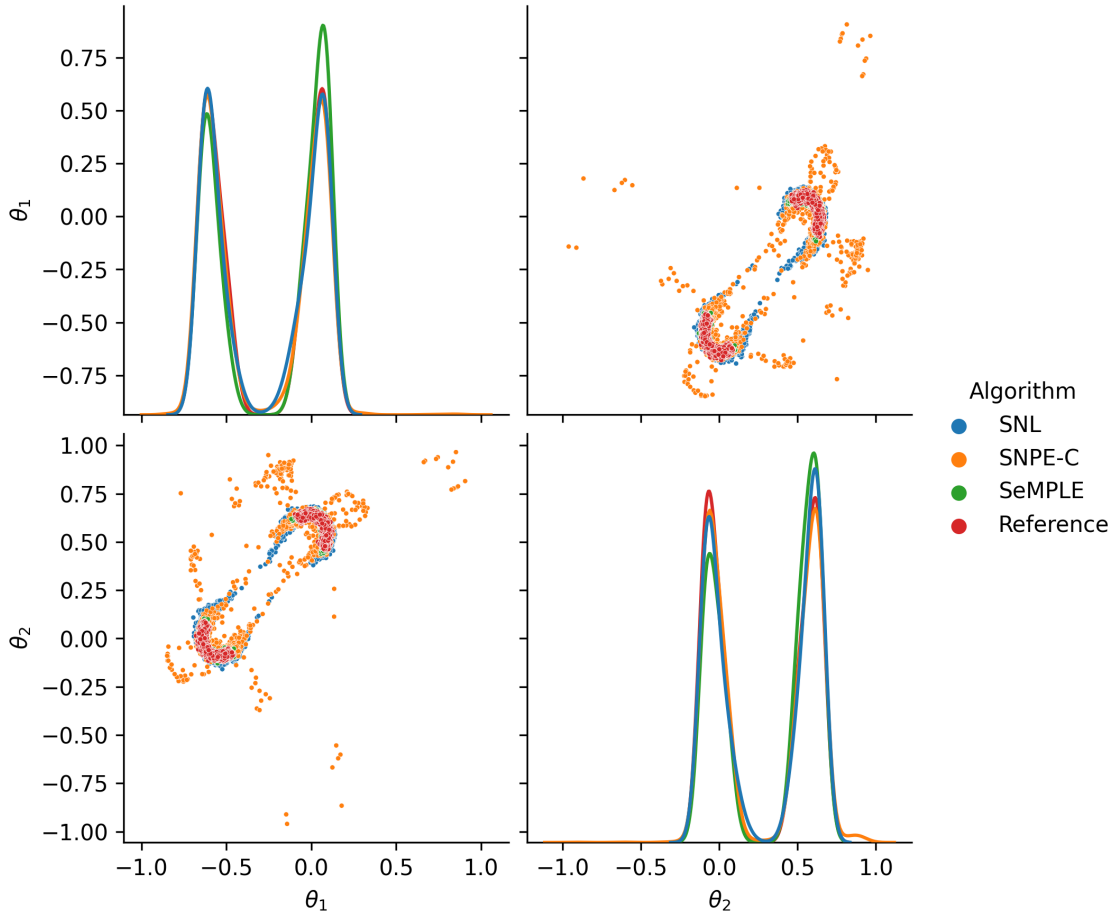


Figure A.17: Two Moons: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 6 in SBIBM.

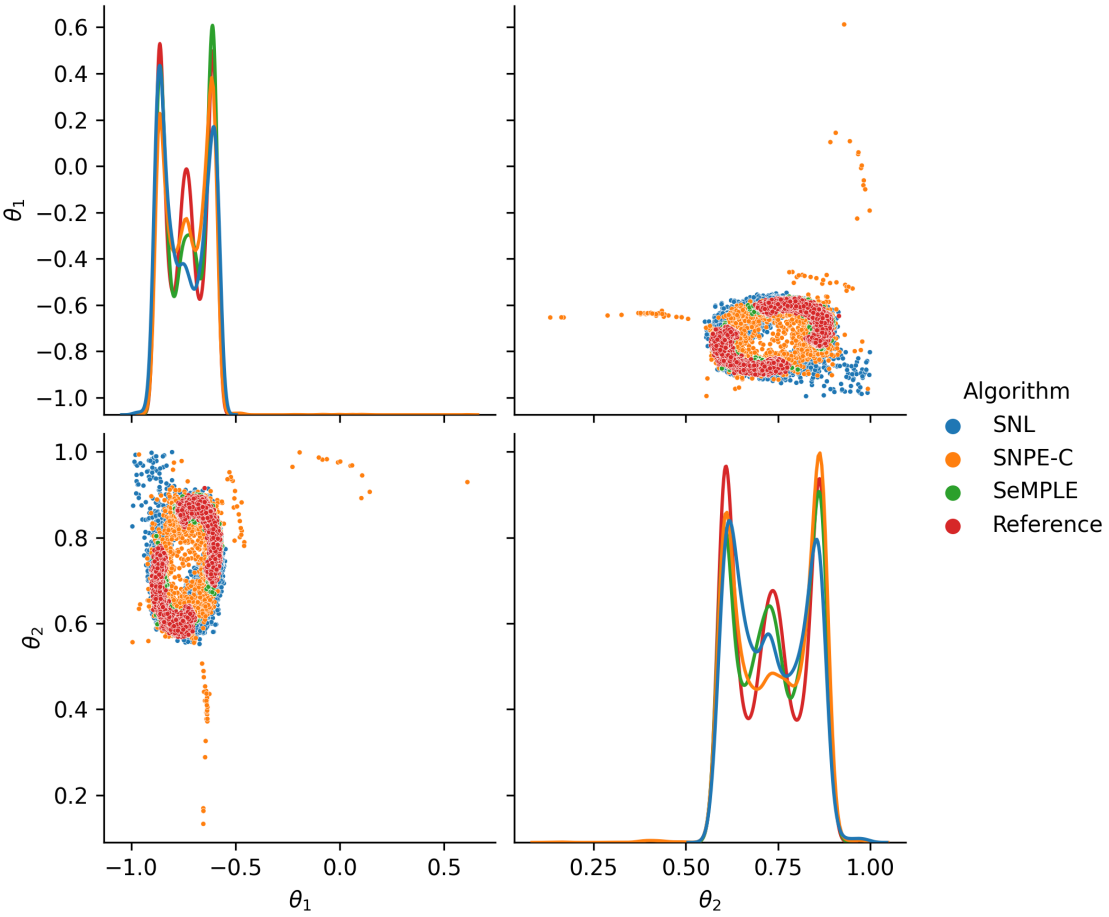


Figure A.18: Two Moons: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 7 in SBIBM.

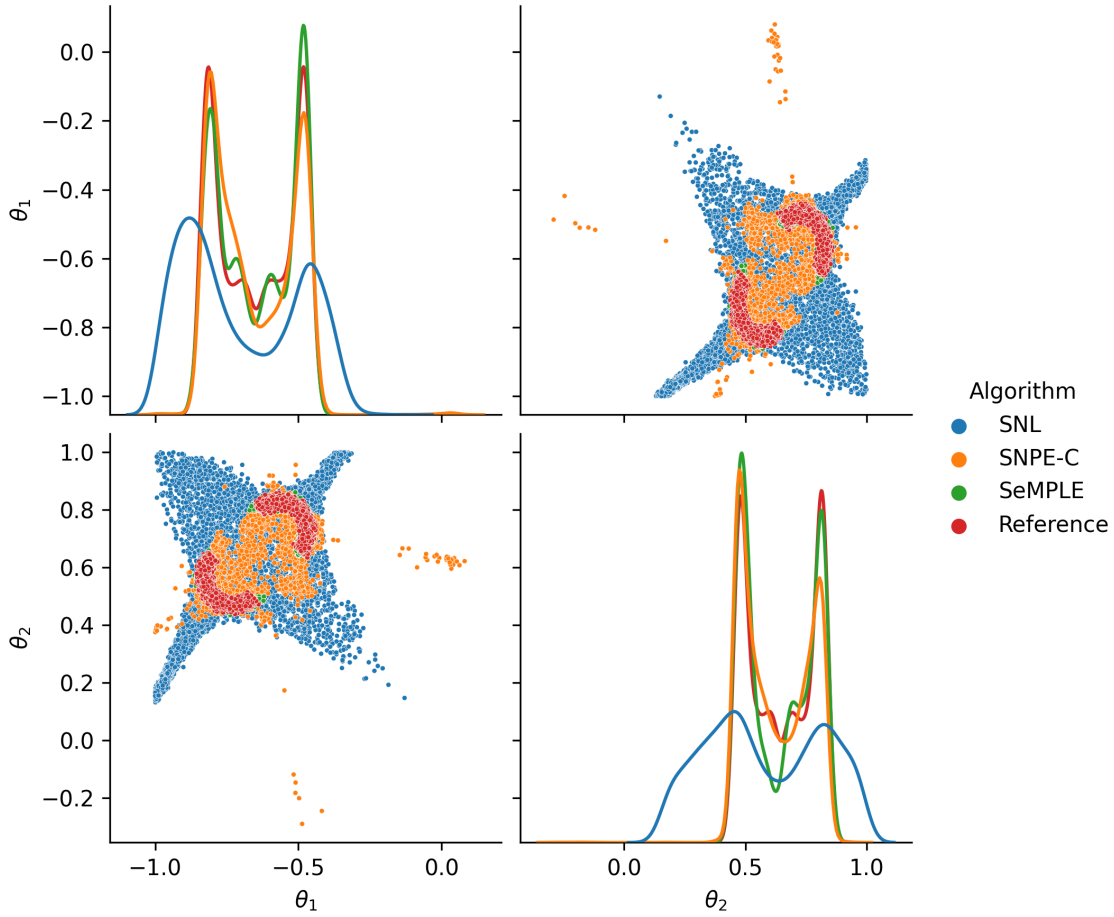


Figure A.19: Two Moons: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 8 in SBIBM.

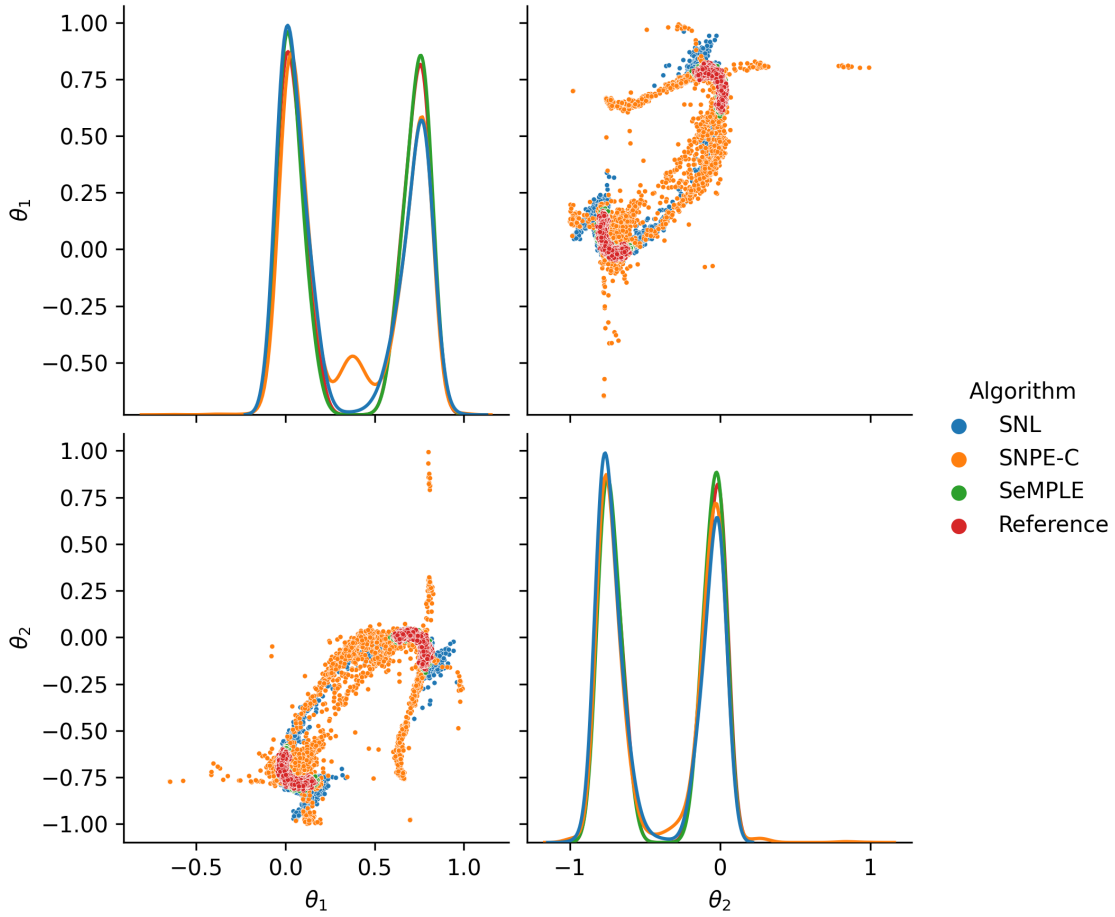


Figure A.20: Two Moons: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 9 in SBIBM.

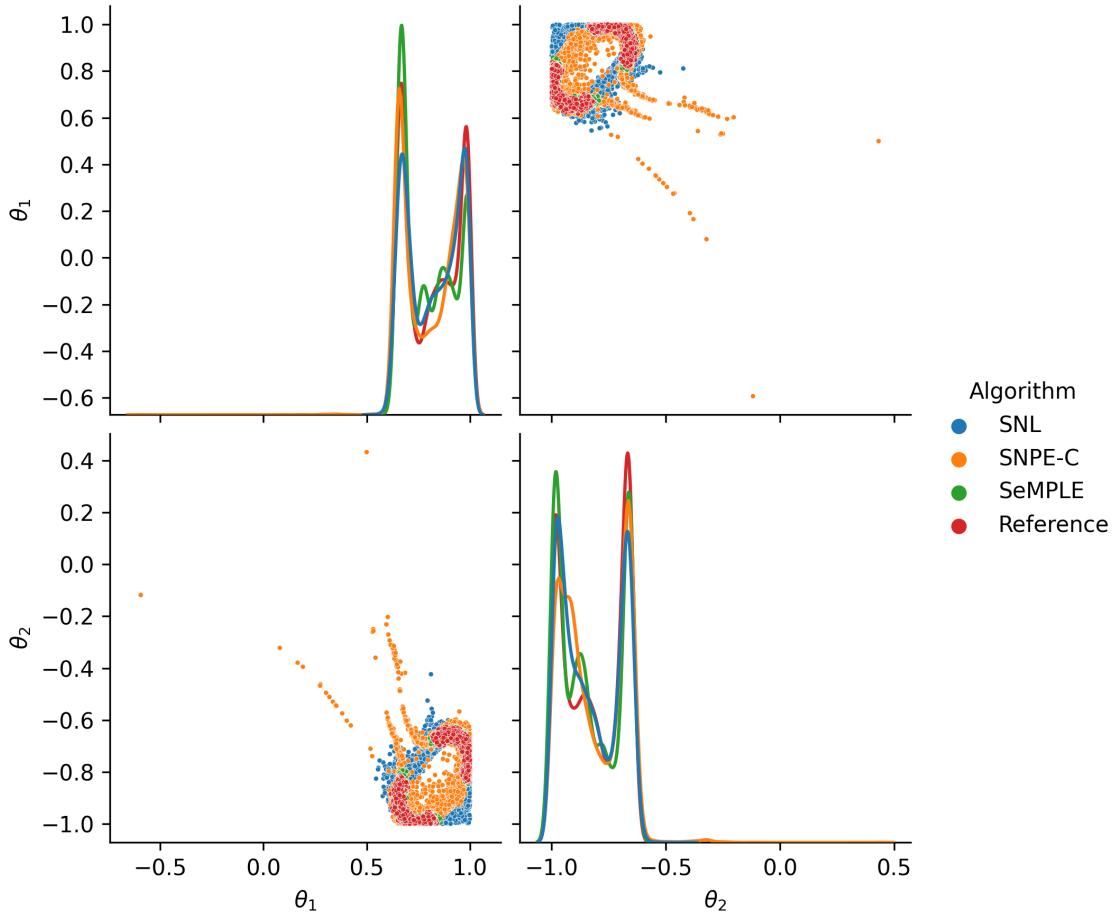


Figure A.21: Two Moons: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 10 in SBIBM.

A.2 Multiple hyperboloid

Figure A.22 shows the median acceptance rate of the Metropolis-Hastings step of 10 independent SeMPLE runs with the Multiple hyperboloid model using the same data set.

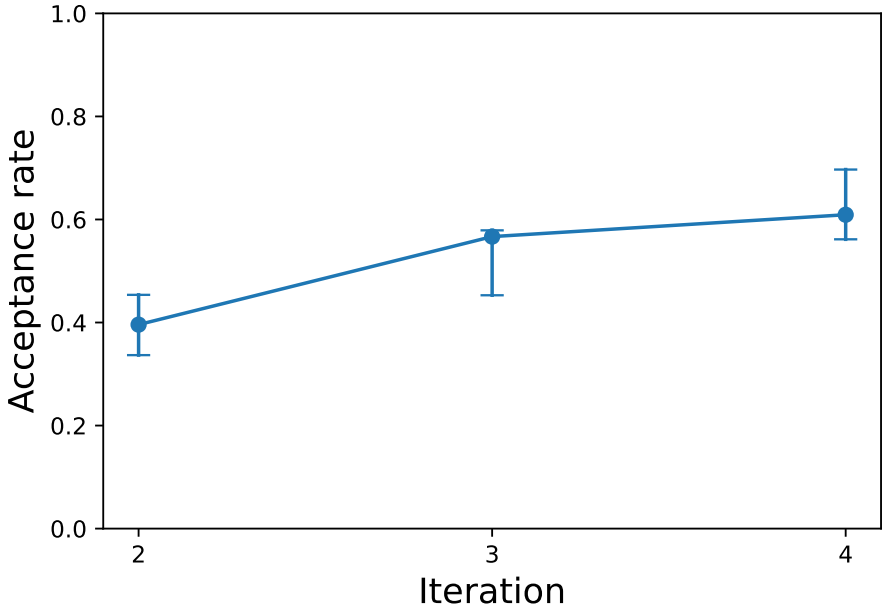


Figure A.22: Multiple hyperboloid: Median acceptance rate of the Metropolis-Hastings step of 10 independent SeMPLE runs with the same data set. Error bars show min/max values.

A.2.1 Density scatter plots

Figures A.23 - A.32 shows scatter plots colored by density of the Multiple hyperboloid posterior sample from the last iteration (r=4) of each algorithm for each of the 10 independent algorithm runs.

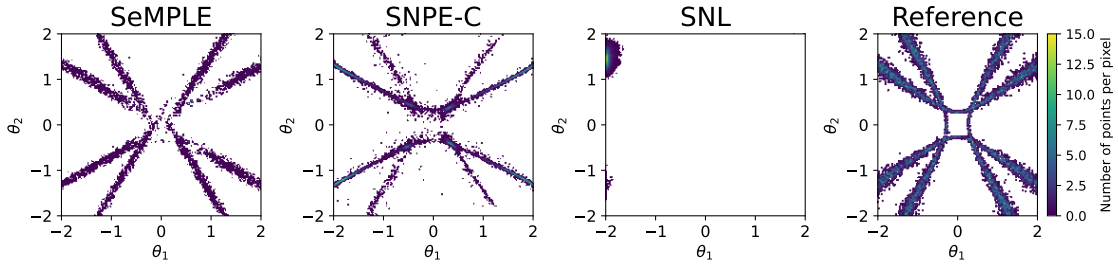


Figure A.23: Multiple hyperboloid: Density scatter plots of posterior sample from the last iteration (r=4) of each algorithm. Repetition 1.

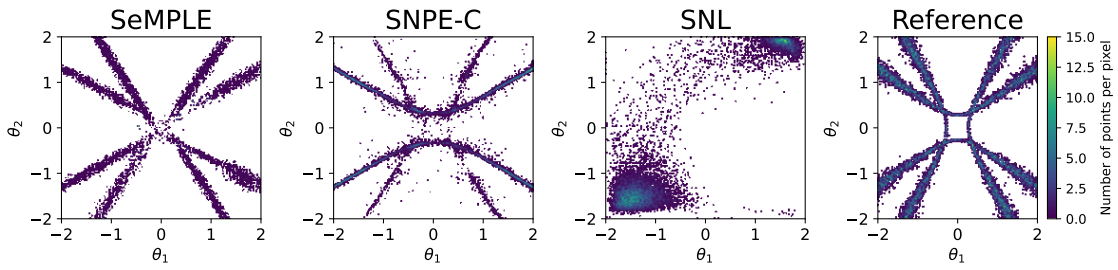


Figure A.24: Multiple hyperboloid: Density scatter plots of posterior sample from the last iteration (r=4) of each algorithm. Repetition 2.

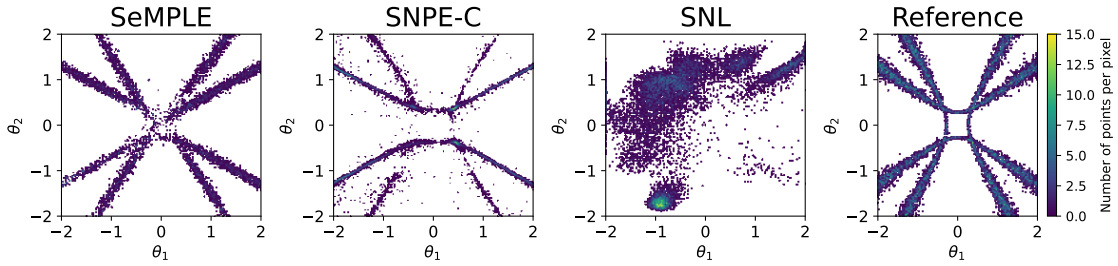


Figure A.25: Multiple hyperboloid: Density scatter plots of posterior sample from the last iteration (r=4) of each algorithm. Repetition 3.

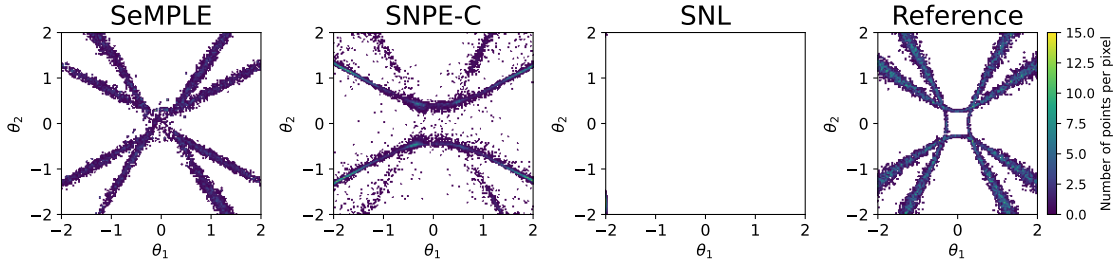


Figure A.26: Multiple hyperboloid: Density scatter plots of posterior sample from the last iteration (r=4) of each algorithm. Repetition 4.

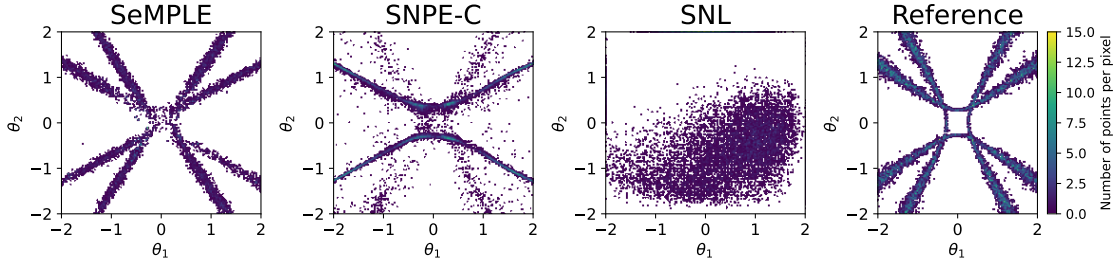


Figure A.27: Multiple hyperboloid: Density scatter plots of posterior sample from the last iteration (r=4) of each algorithm. Repetition 5.

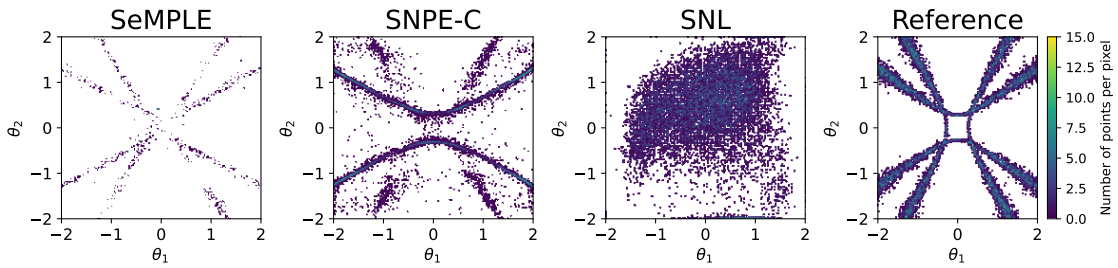


Figure A.28: Multiple hyperboloid: Density scatter plots of posterior sample from the last iteration (r=4) of each algorithm. Repetition 6.

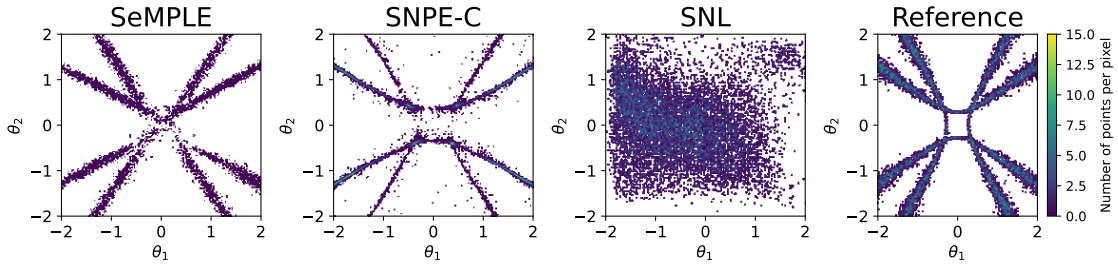


Figure A.29: Multiple hyperboloid: Density scatter plots of posterior sample from the last iteration (r=4) of each algorithm. Repetition 7.

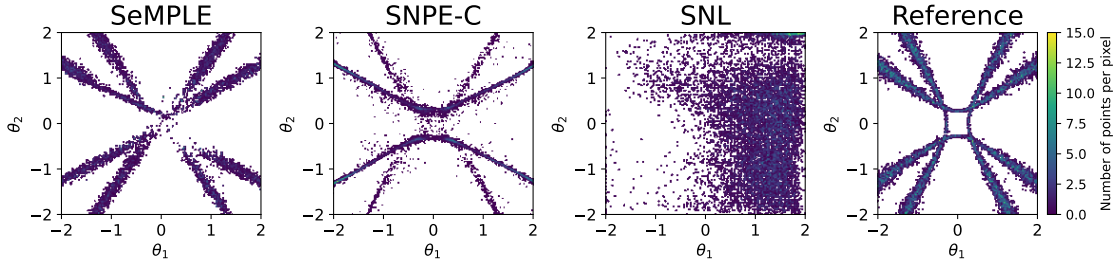


Figure A.30: Multiple hyperboloid: Density scatter plots of posterior sample from the last iteration (r=4) of each algorithm. Repetition 8.

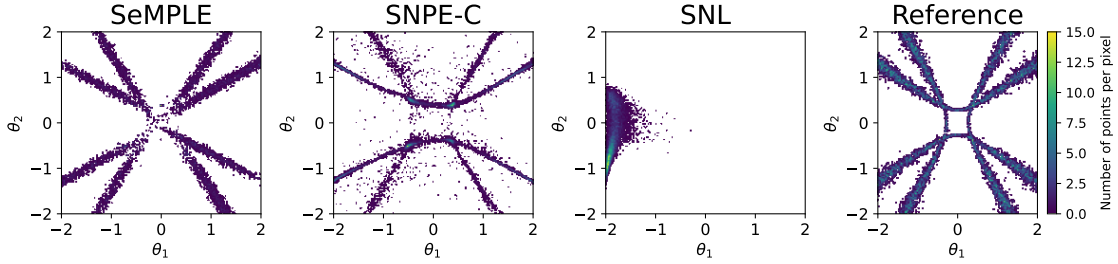


Figure A.31: Multiple hyperboloid: Density scatter plots of posterior sample from the last iteration (r=4) of each algorithm. Repetition 9.

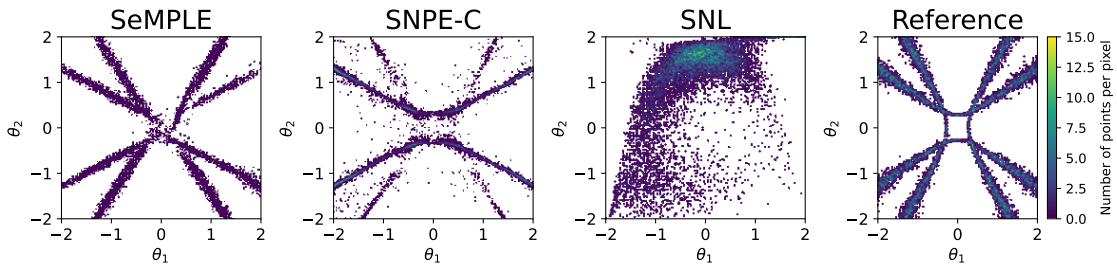


Figure A.32: Multiple hyperboloid: Density scatter plots of posterior sample from the last iteration (r=4) of each algorithm. Repetition 10.

A.2.2 Pair plots

Figures A.33 - A.42 shows pair plots of Multiple hyperboloid posterior samples from the last iteration (r=4) of each algorithm for each of the 10 independent algorithm runs.

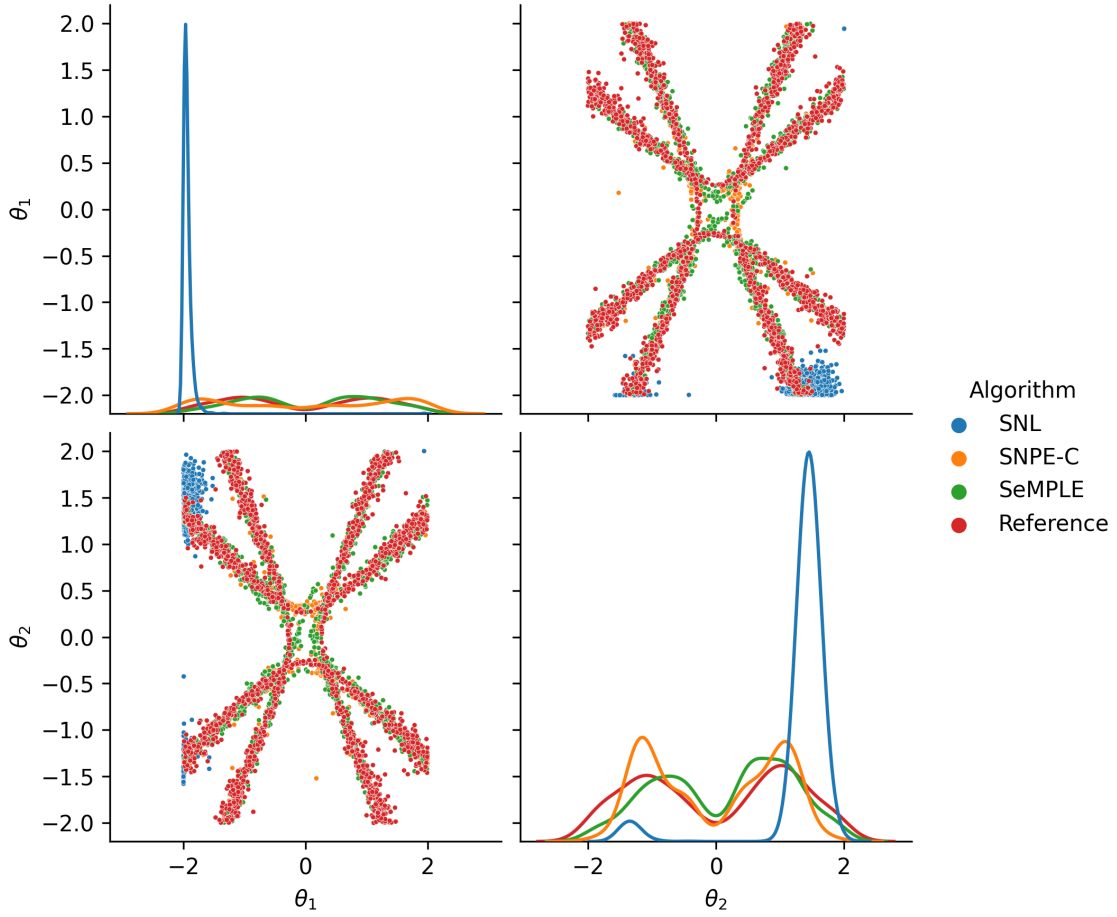


Figure A.33: Multiple hyperboloid: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Repetition 1.

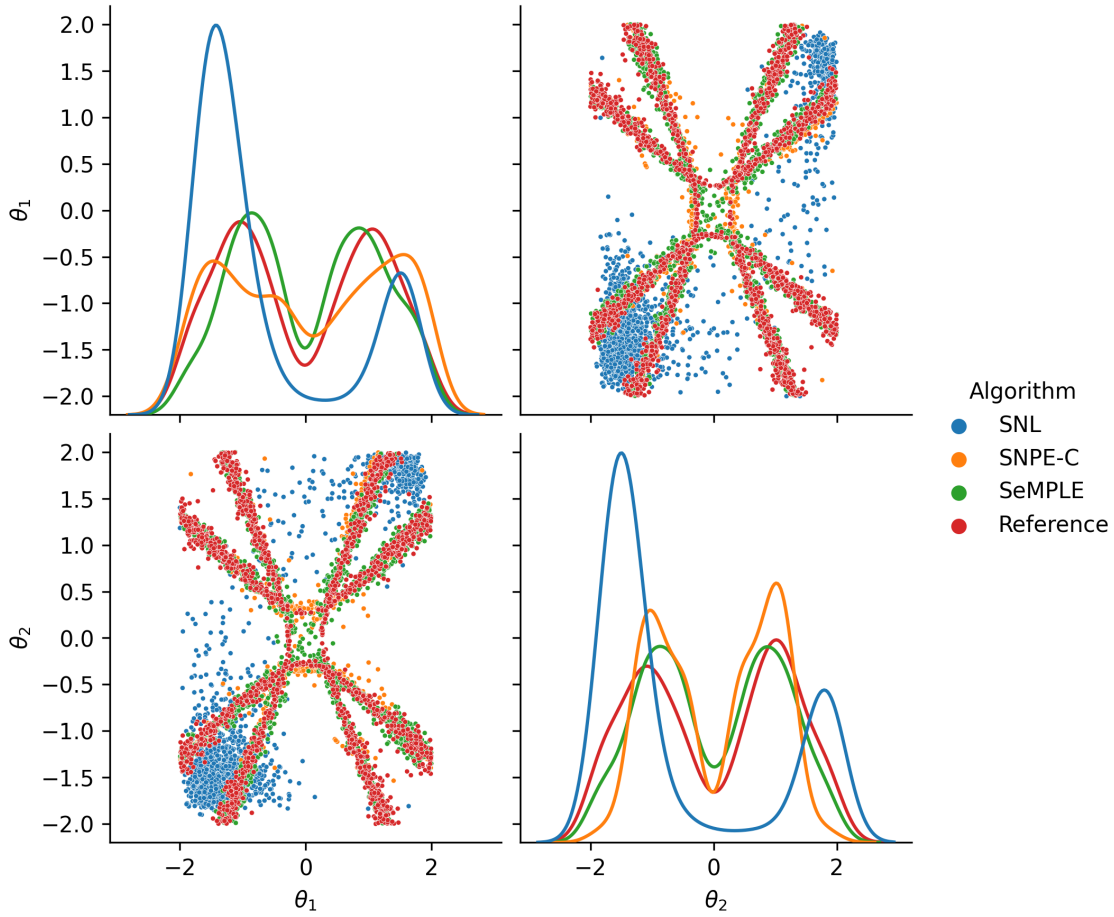


Figure A.34: Multiple hyperboloid: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Repetition 2.

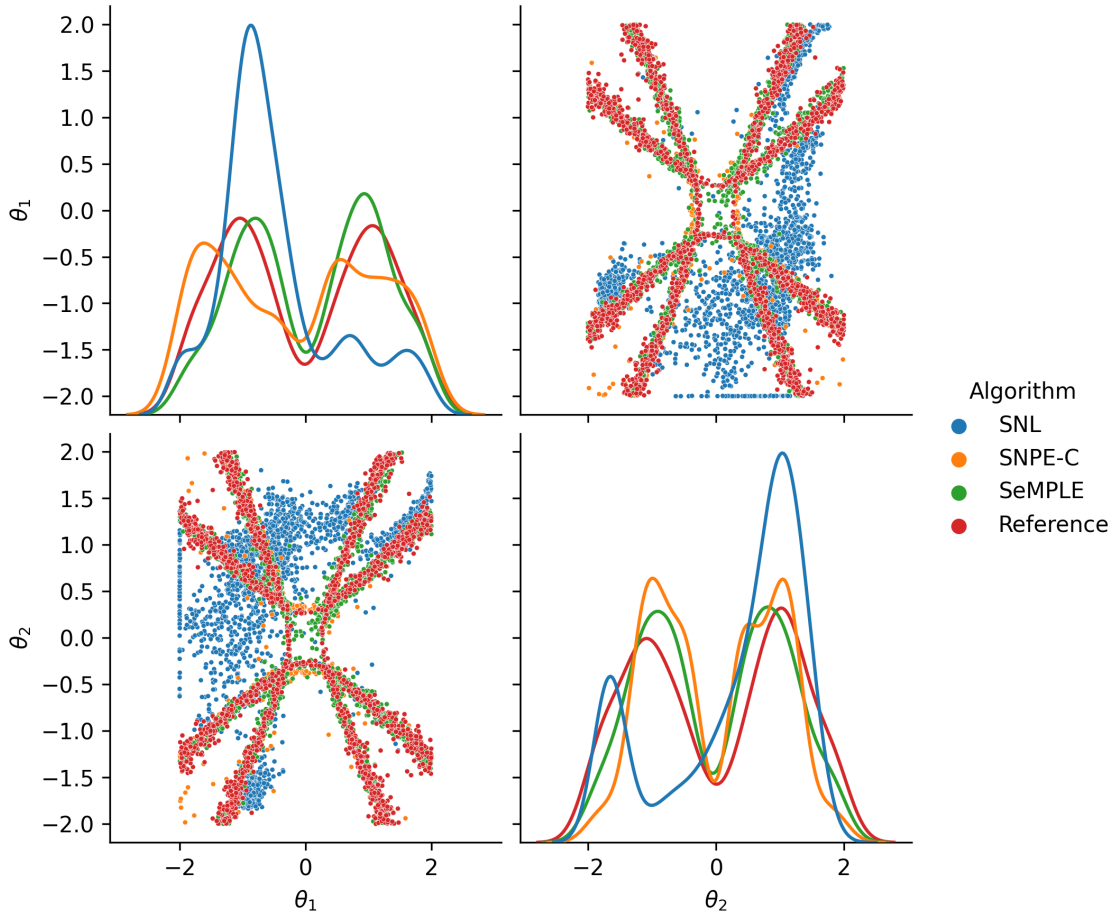


Figure A.35: Multiple hyperboloid: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Repetition 3.

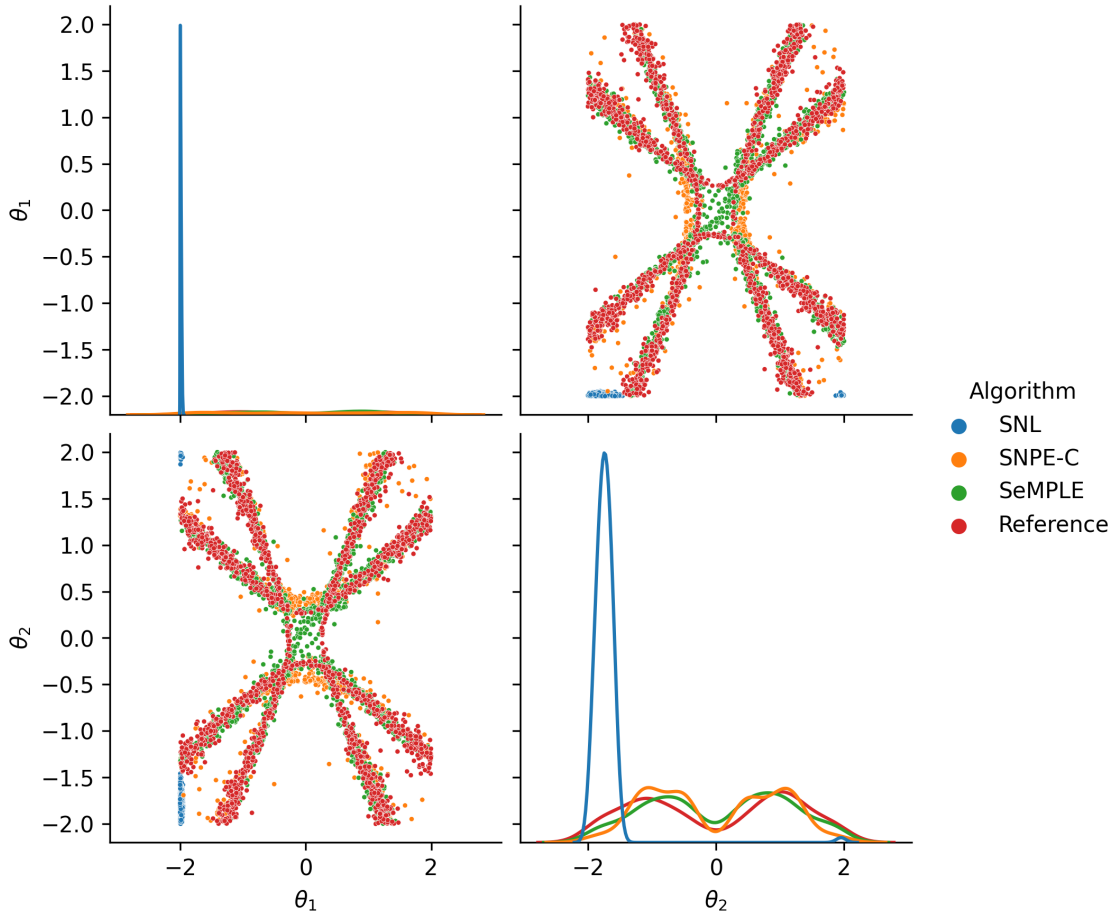


Figure A.36: Multiple hyperboloid: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Repetition 4.

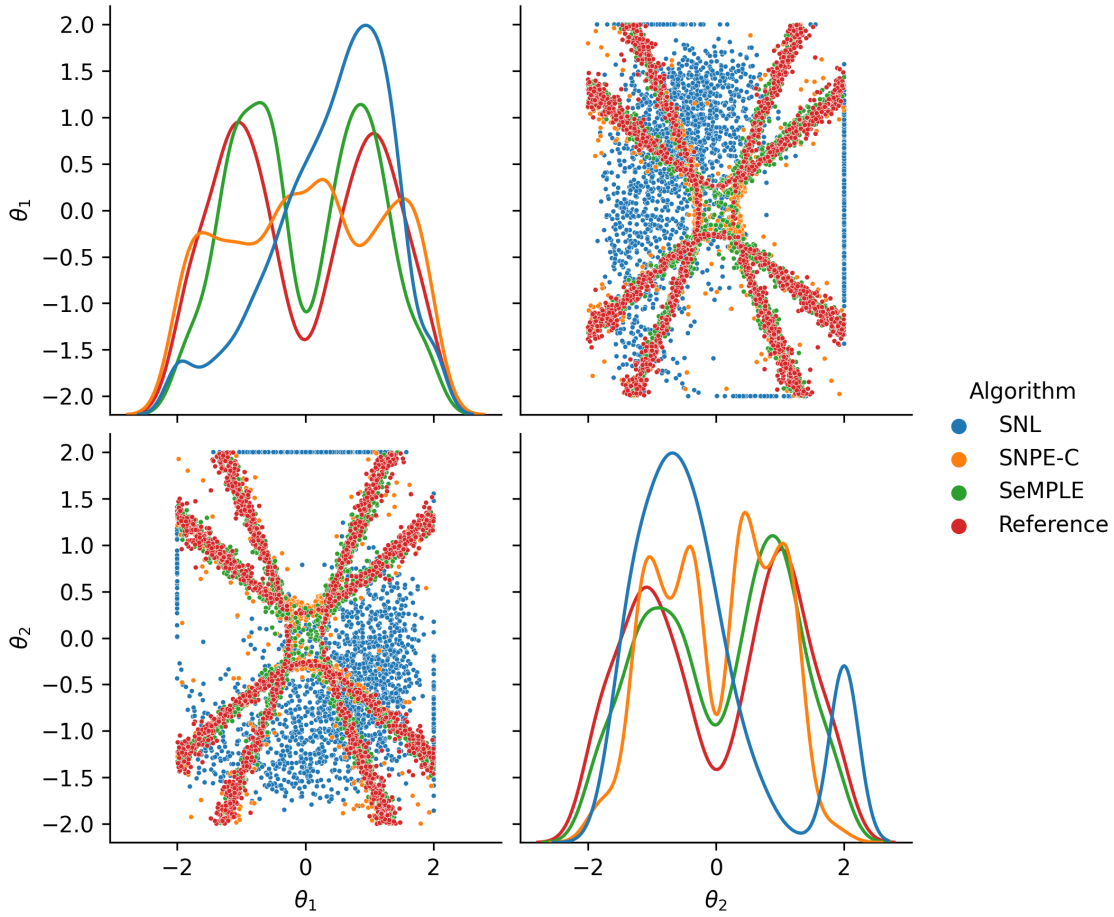


Figure A.37: Multiple hyperboloid: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Repetition 5.

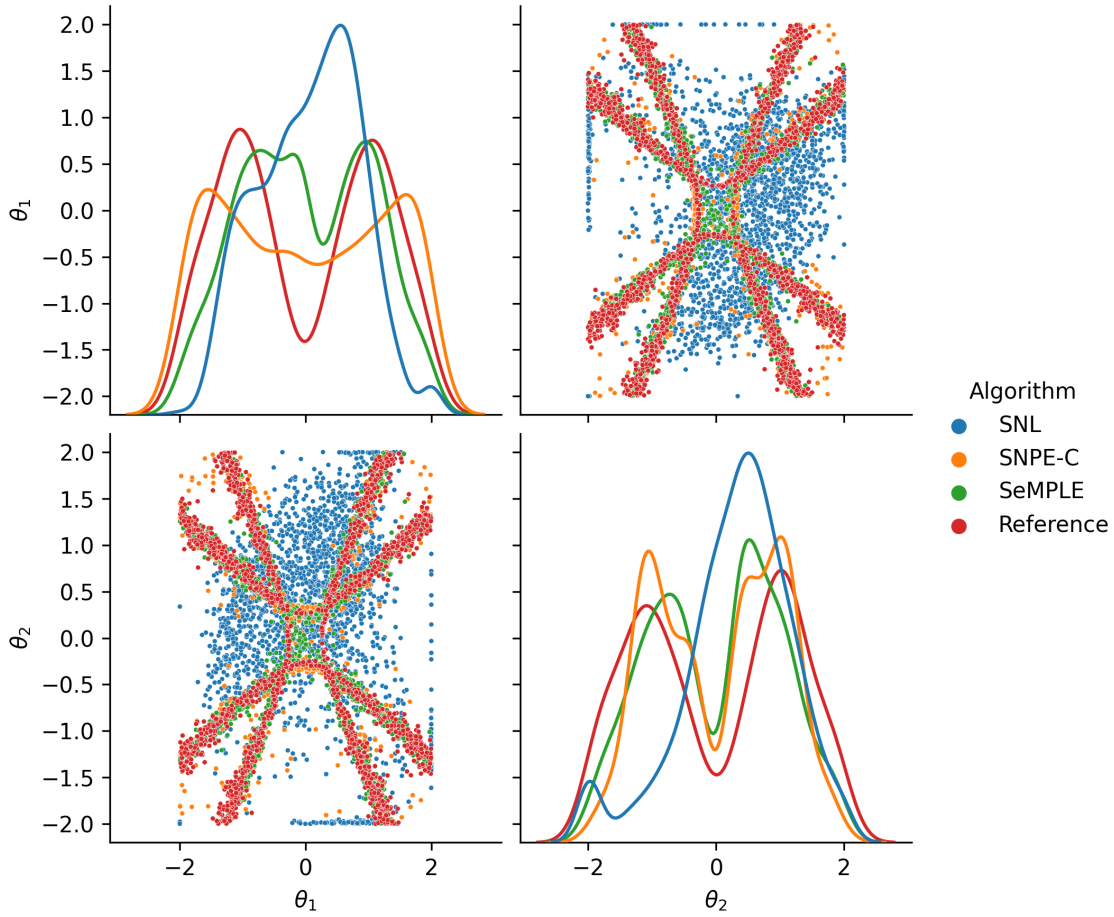


Figure A.38: Multiple hyperboloid: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Repetition 6.

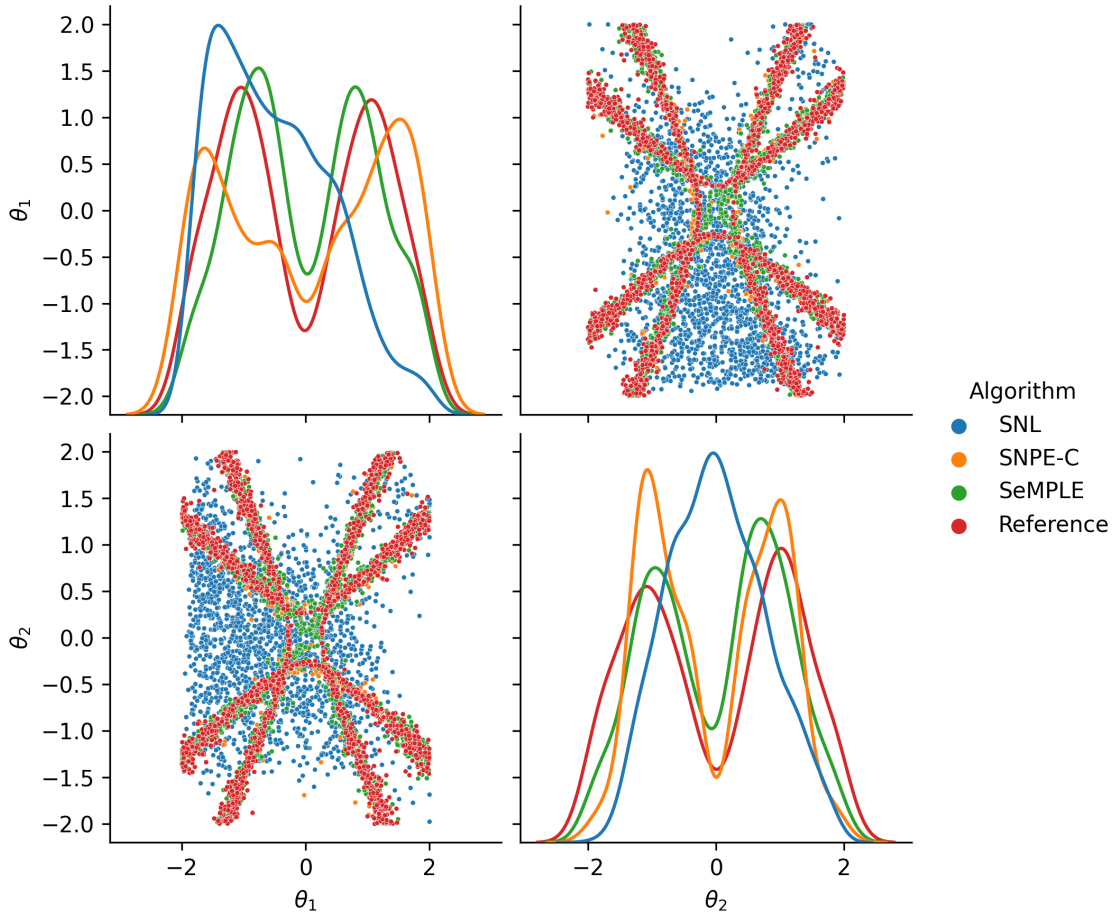


Figure A.39: Multiple hyperboloid: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Repetition 7.

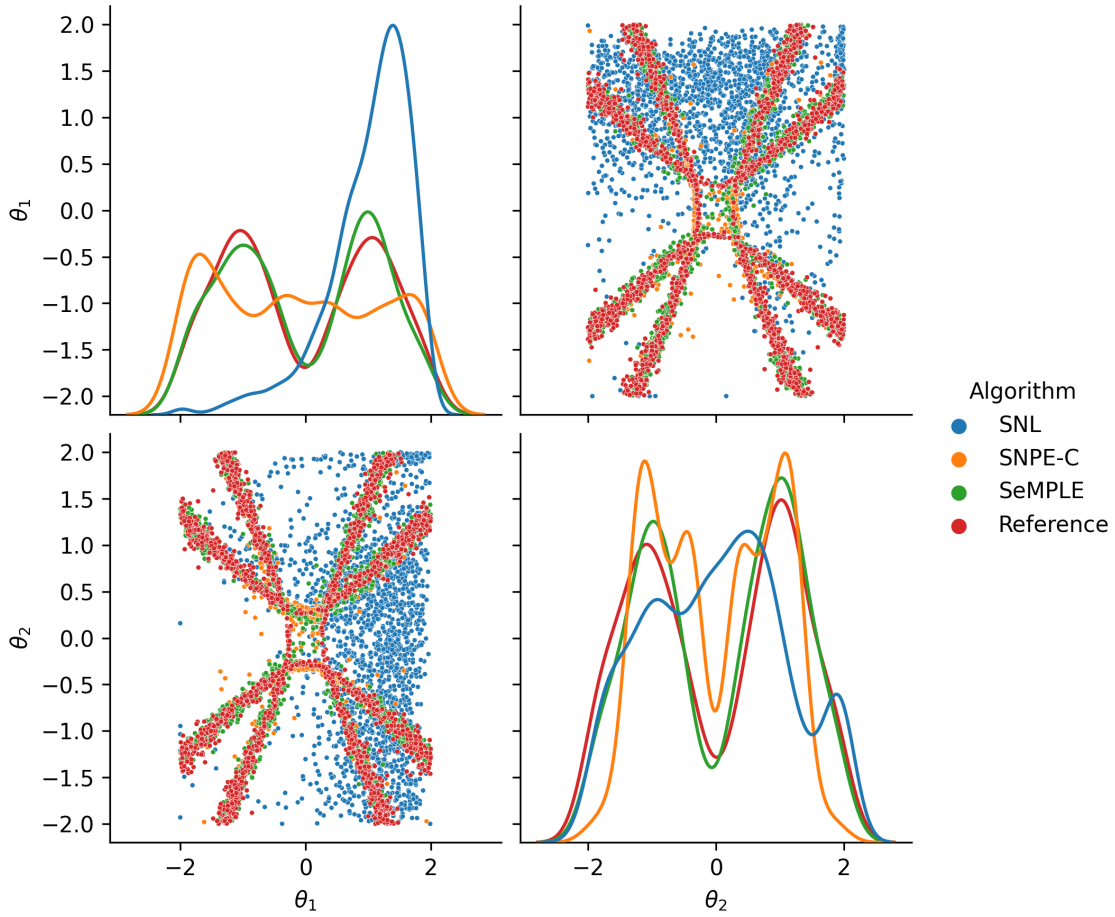


Figure A.40: Multiple hyperboloid: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Repetition 8.

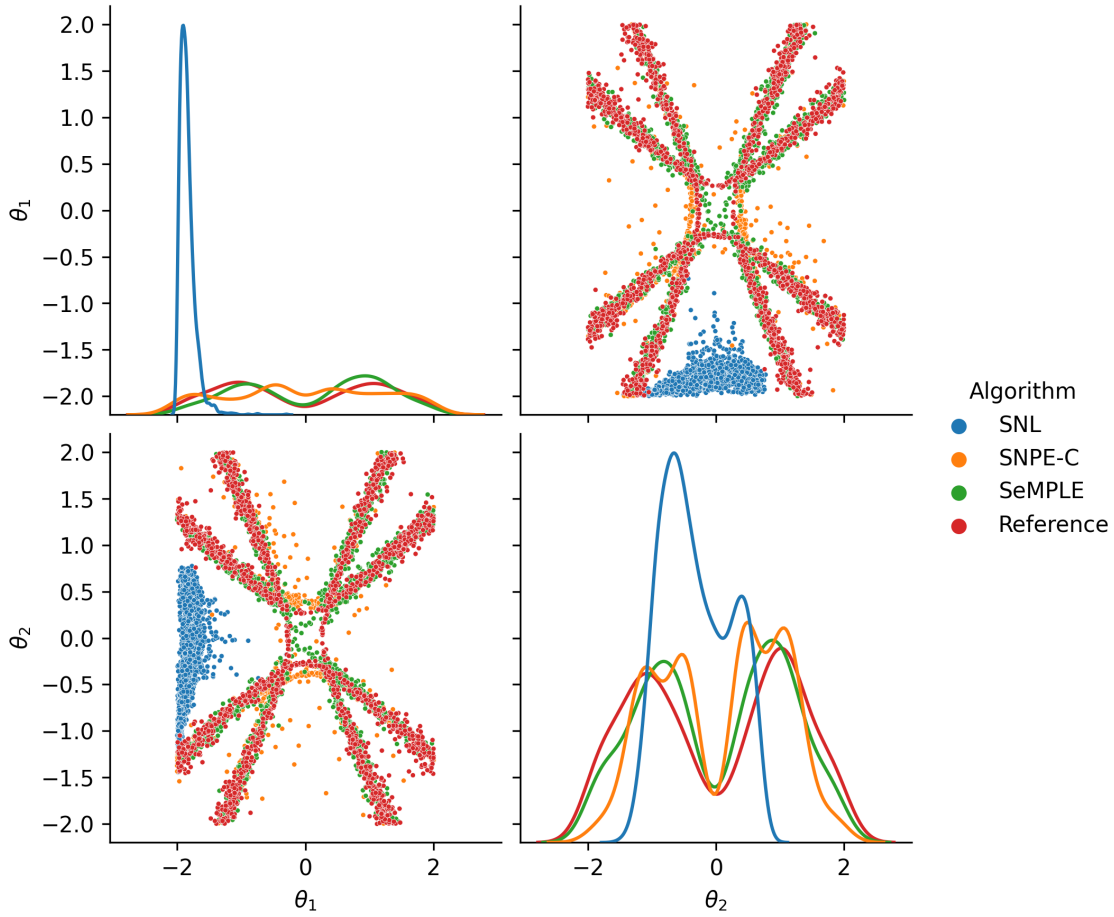


Figure A.41: Multiple hyperboloid: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Repetition 9.

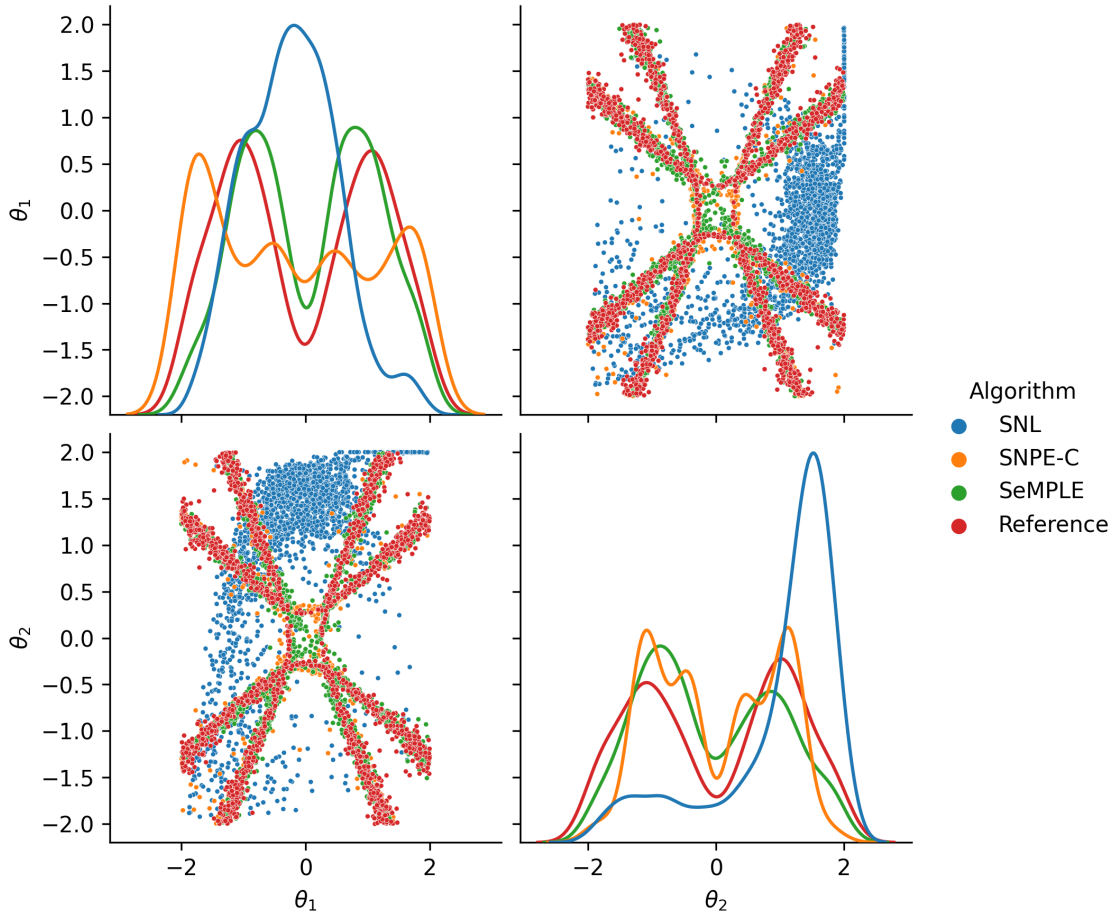


Figure A.42: Multiple hyperboloid: Pair plot of posterior sample from the last iteration (r=4) of each algorithm. Repetition 10.

A.3 Ornstein-Uhlenbeck process

Figures A.43 - A.52 shows pair plots of Ornstein-Uhlenbeck posterior samples from the last iteration (r=4) of each algorithm for each of the 10 independent algorithm runs.

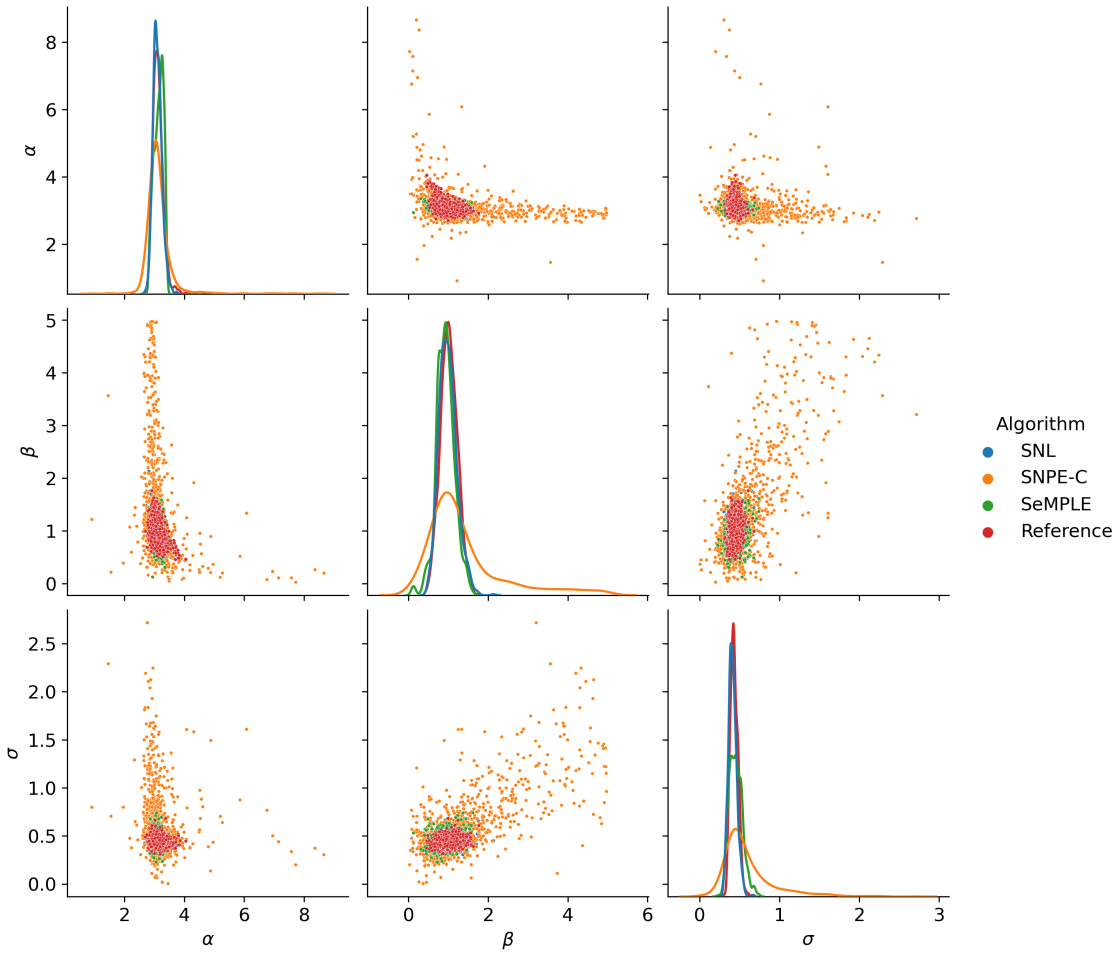


Figure A.43: Pair plot of Ornstein-Uhlenbeck posterior sample from the last iteration (r=4) of each algorithm. Repetition 1.

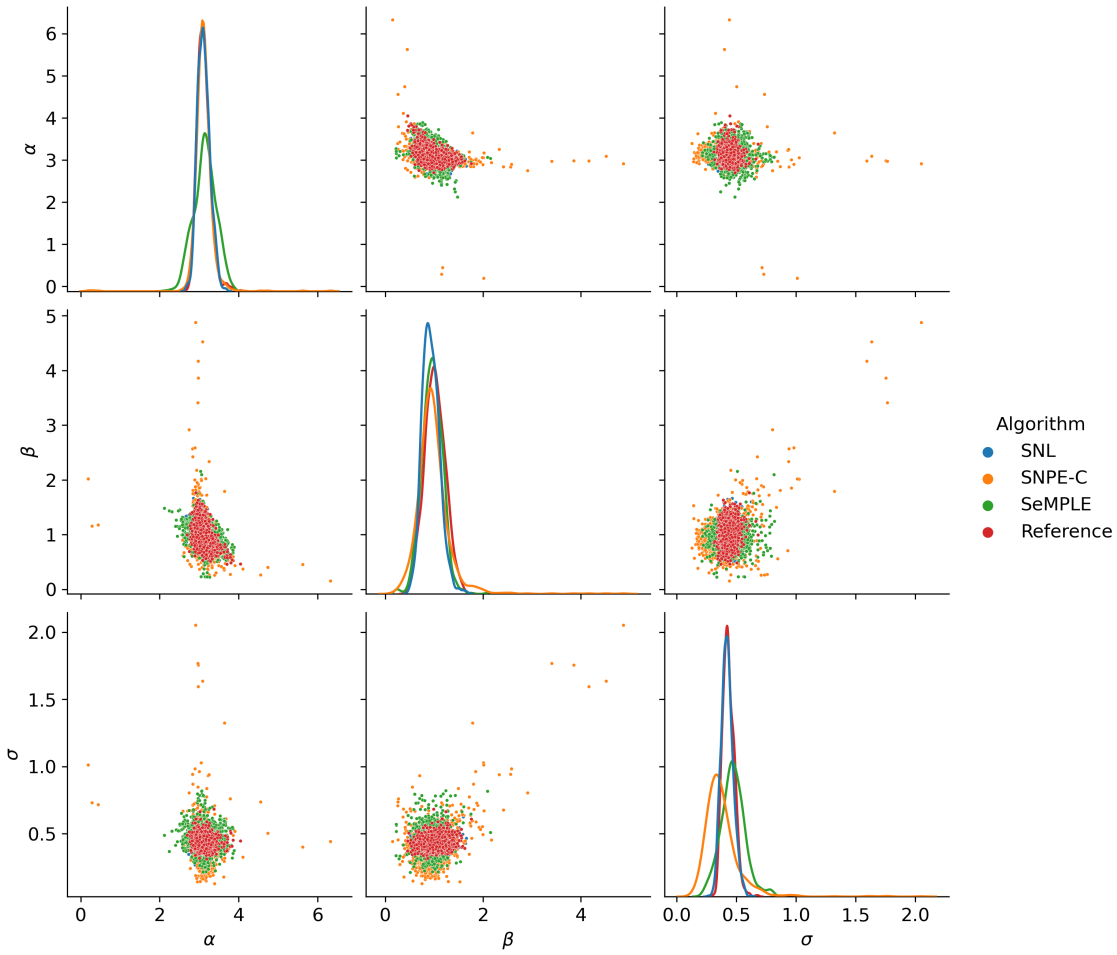


Figure A.44: Pair plot of Ornstein-Uhlenbeck posterior sample from the last iteration (r=4) of each algorithm. Repetition 2.

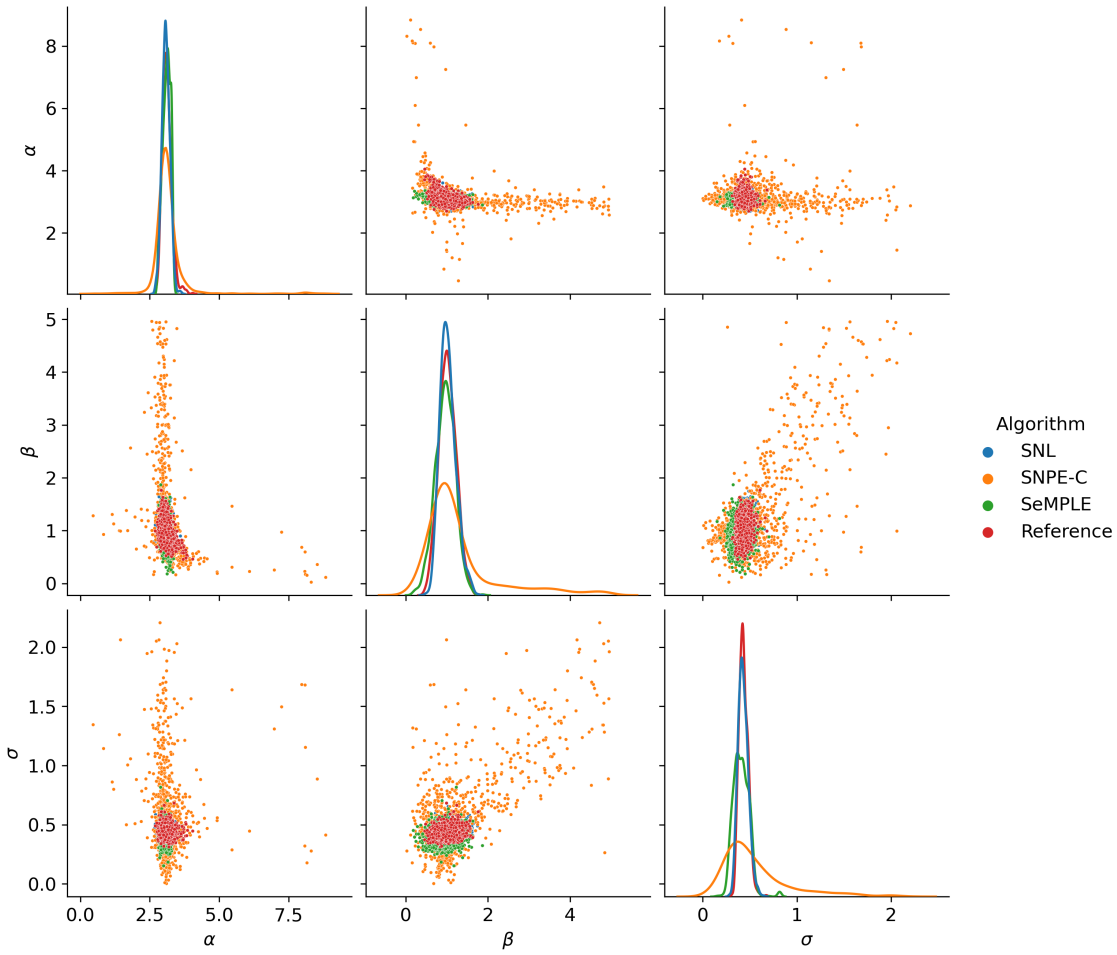


Figure A.45: Pair plot of Ornstein-Uhlenbeck posterior sample from the last iteration (r=4) of each algorithm. Repetition 3.

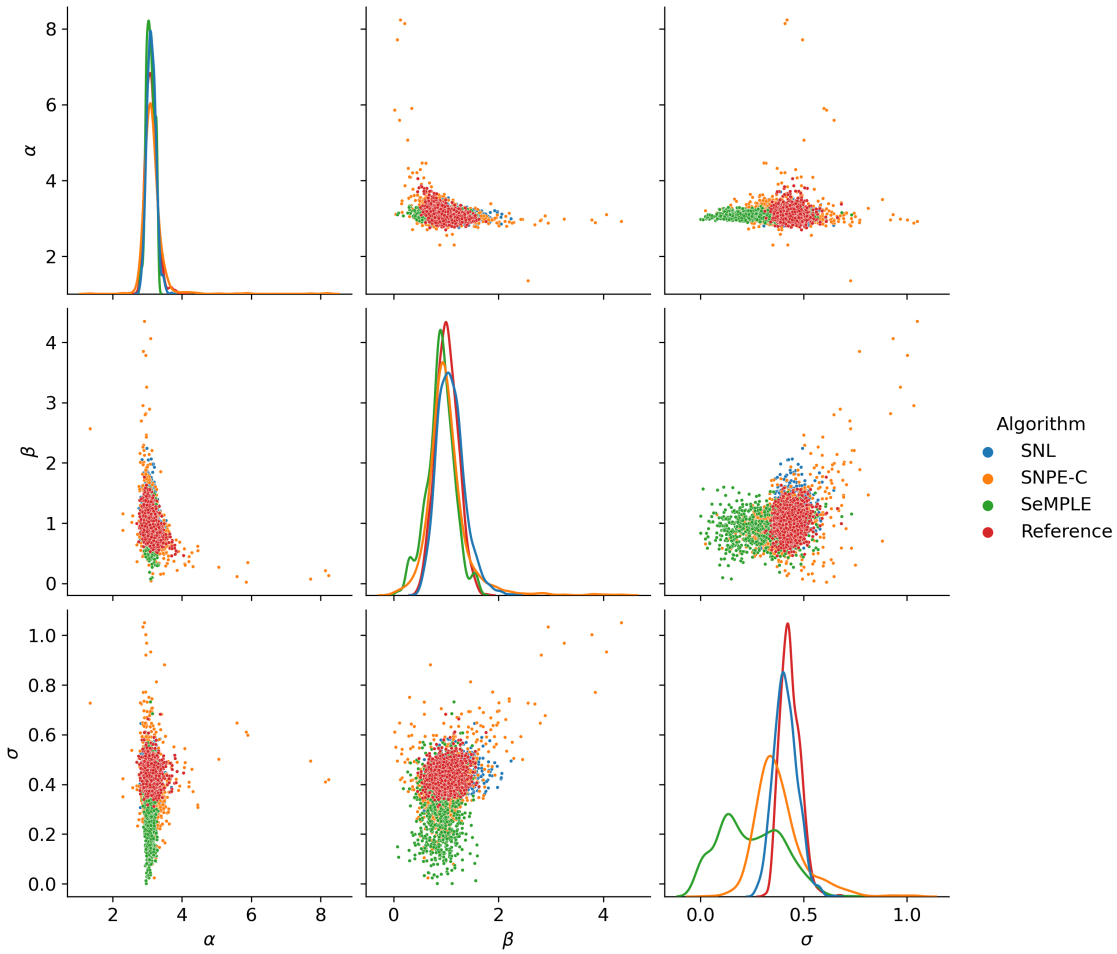


Figure A.46: Pair plot of Ornstein-Uhlenbeck posterior sample from the last iteration (r=4) of each algorithm. Repetition 4.

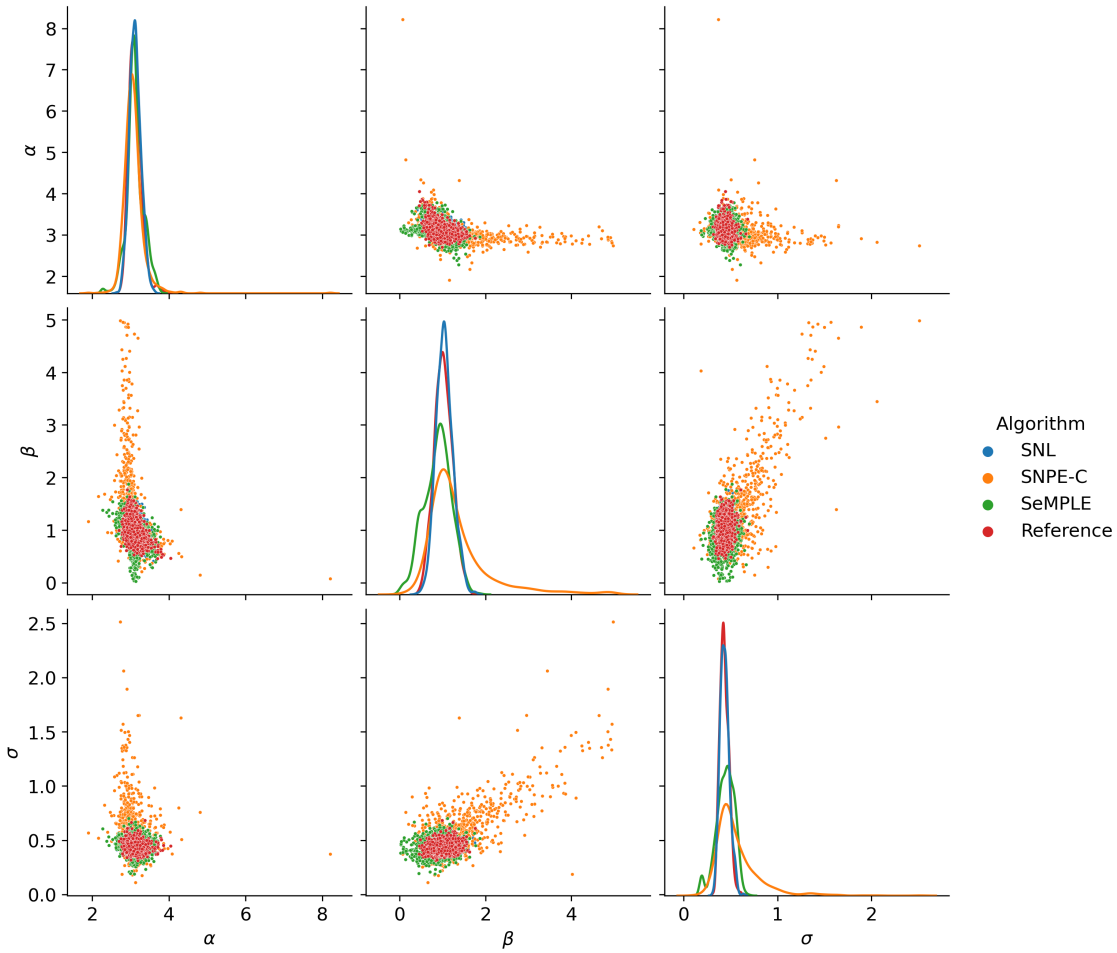


Figure A.47: Pair plot of Ornstein-Uhlenbeck posterior sample from the last iteration (r=4) of each algorithm. Repetition 5.

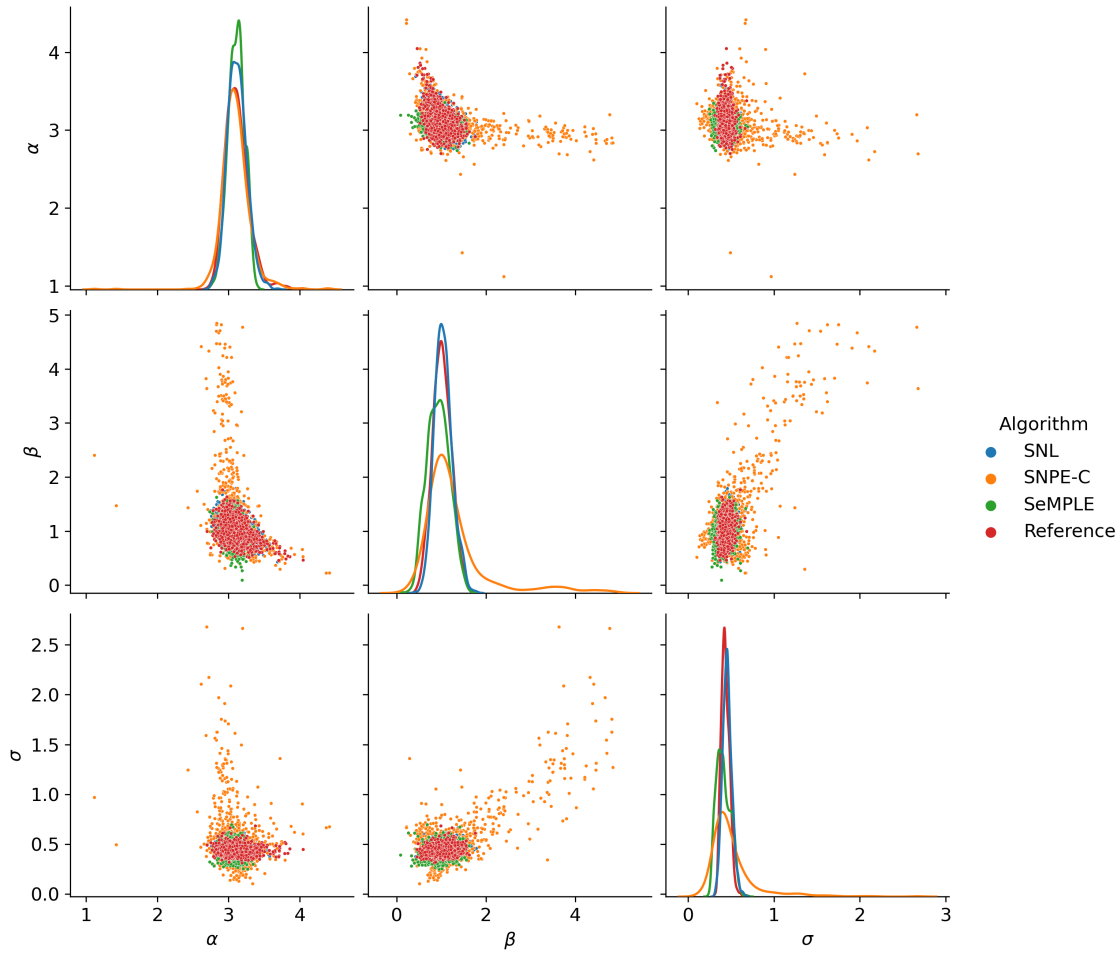


Figure A.48: Pair plot of Ornstein-Uhlenbeck posterior sample from the last iteration (r=4) of each algorithm. Repetition 6.

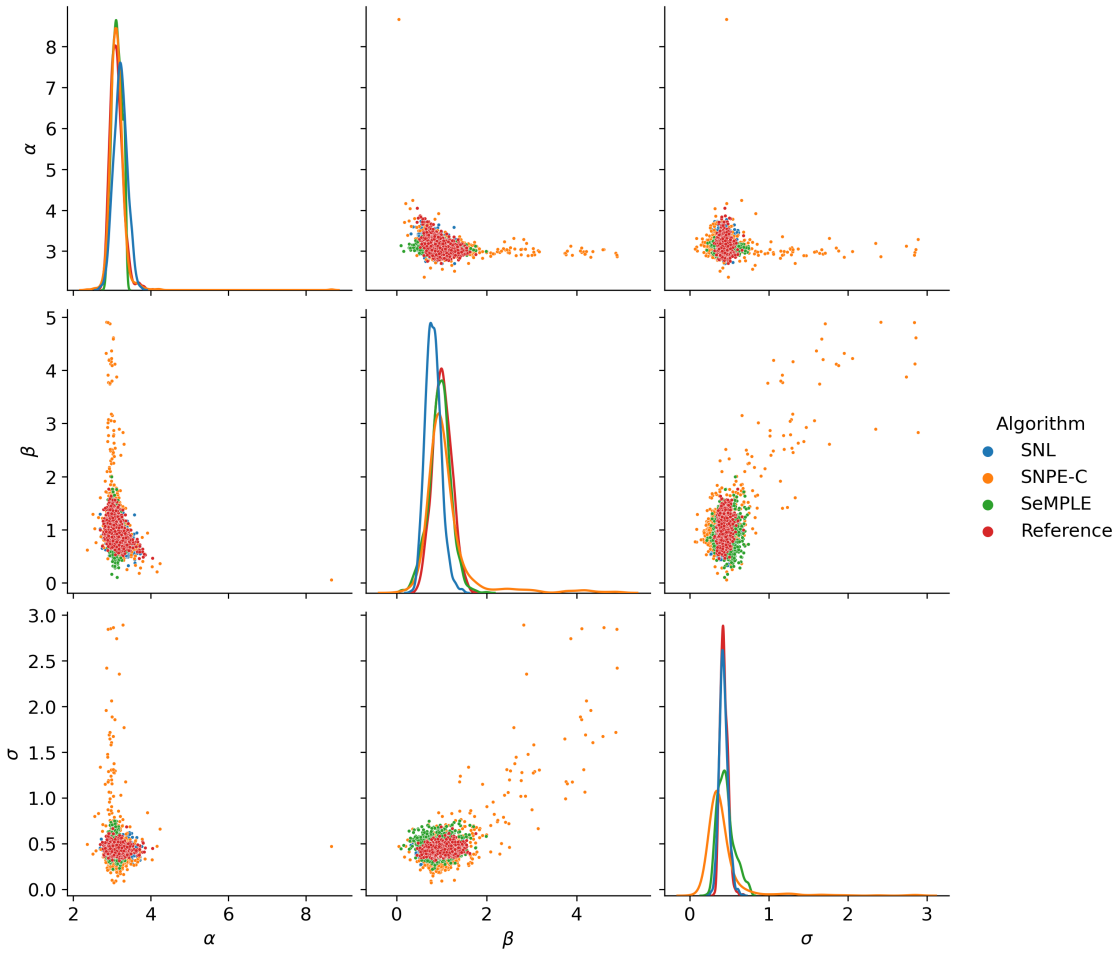


Figure A.49: Pair plot of Ornstein-Uhlenbeck posterior sample from the last iteration (r=4) of each algorithm. Repetition 7.

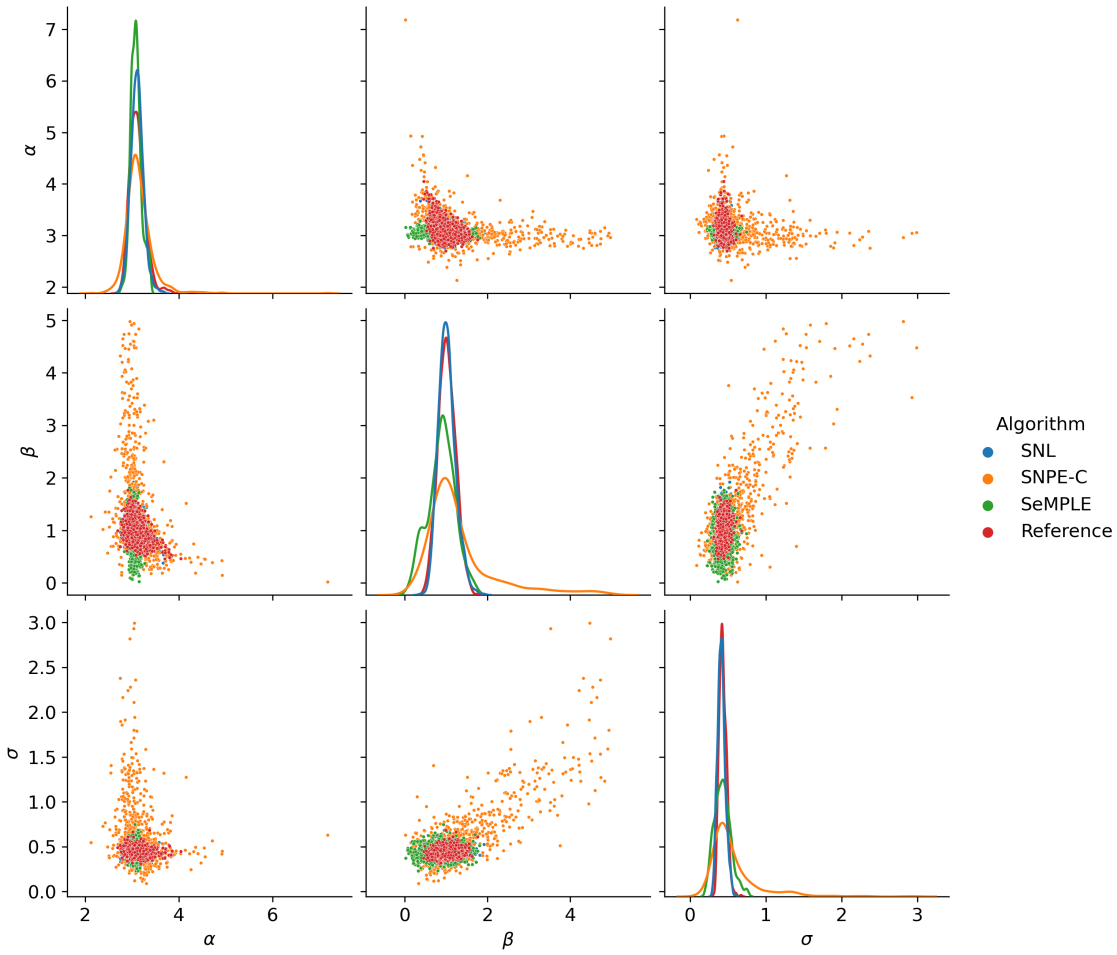


Figure A.50: Pair plot of Ornstein-Uhlenbeck posterior sample from the last iteration (r=4) of each algorithm. Repetition 8.

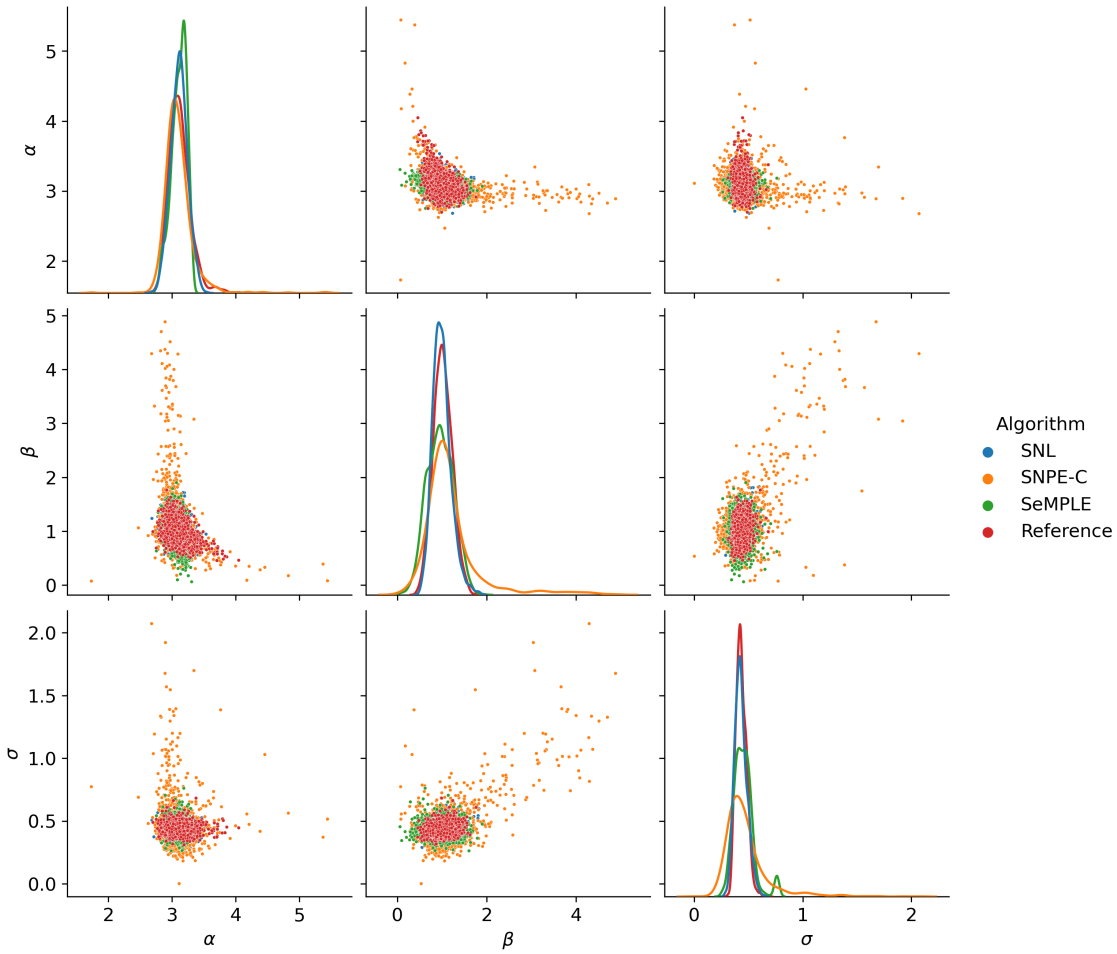


Figure A.51: Pair plot of Ornstein-Uhlenbeck posterior sample from the last iteration (r=4) of each algorithm. Repetition 9.

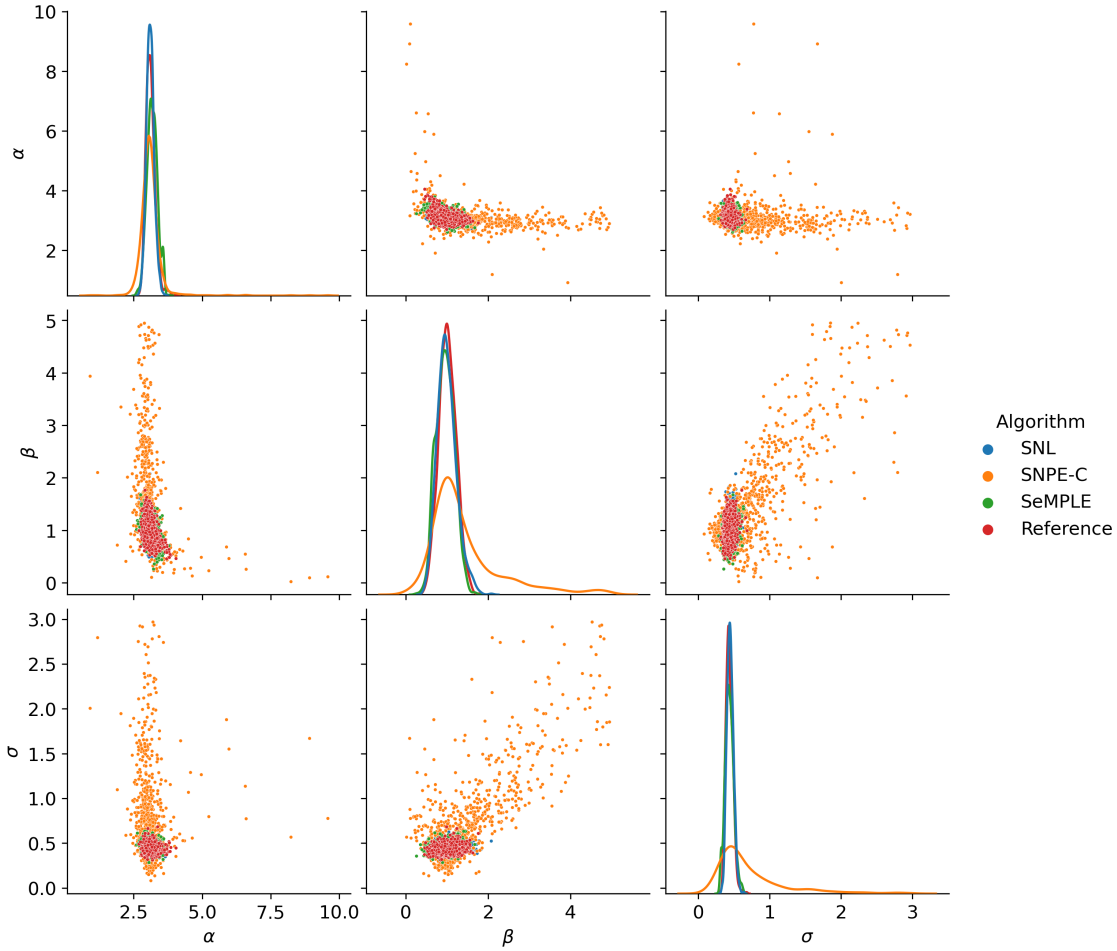


Figure A.52: Pair plot of Ornstein-Uhlenbeck posterior sample from the last iteration (r=4) of each algorithm. Repetition 10.

Figures A.53, A.54 and A.55 shows the Markov chains of each model parameter respectively of the true posterior sample used in the performance metric calculations.

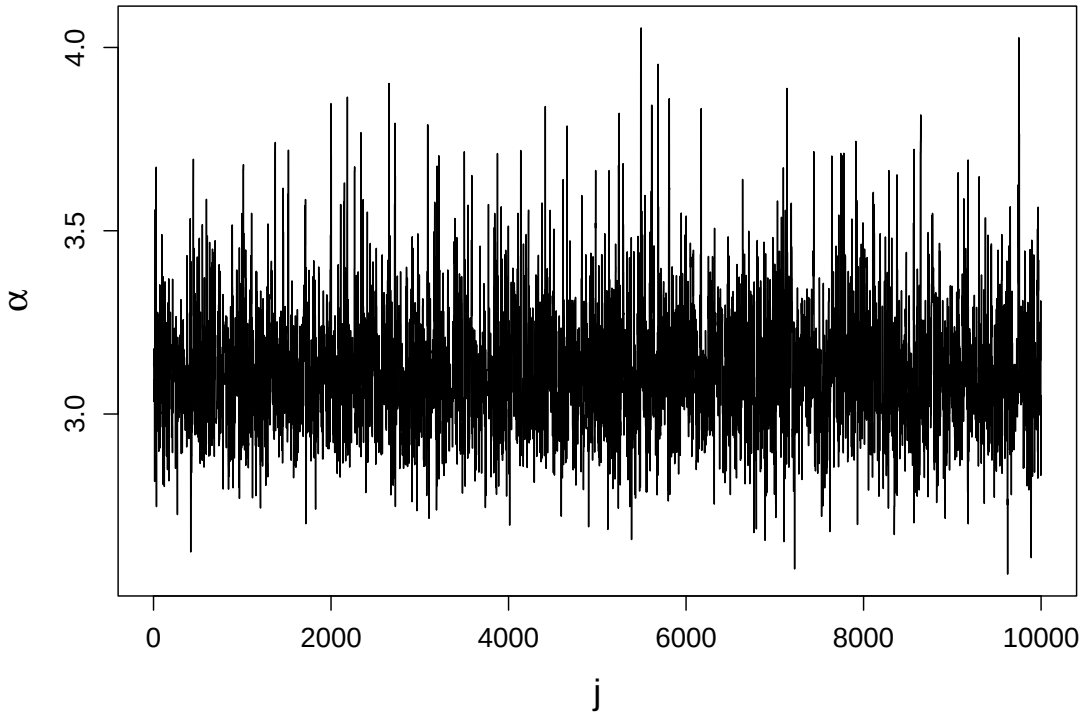


Figure A.53: Markov chain of parameter
x obtained by the Metropolis-Hastings algorithm as the true posterior sample.

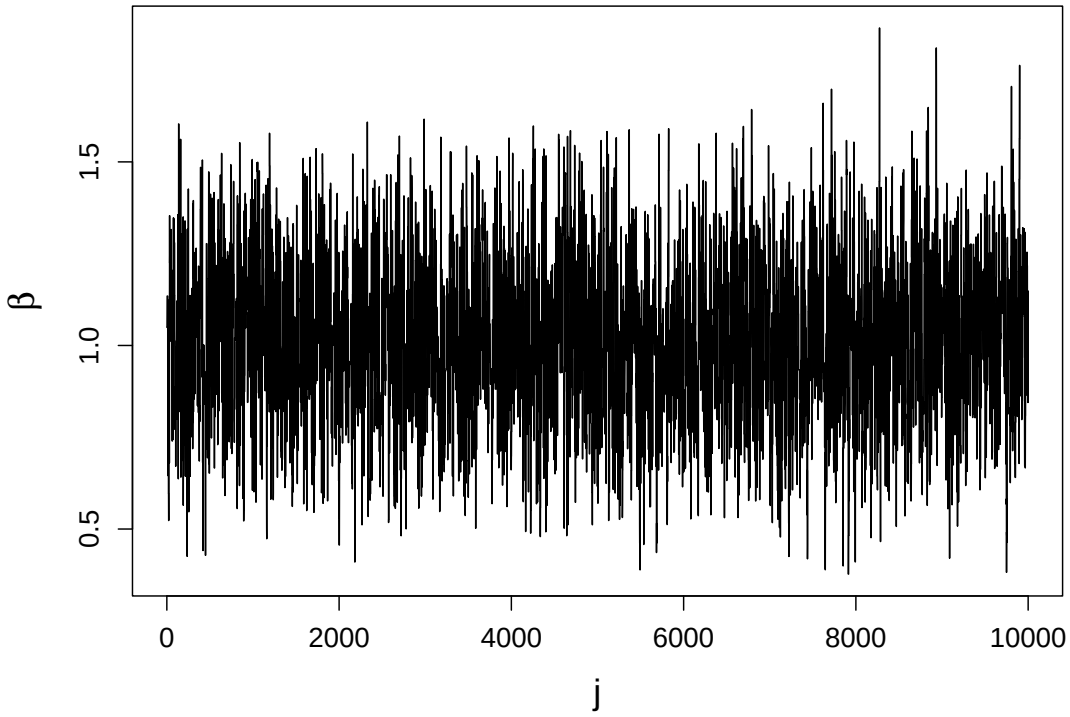


Figure A.54: Markov chain of parameter
x obtained by the Metropolis-Hastings algorithm as the true posterior sample.

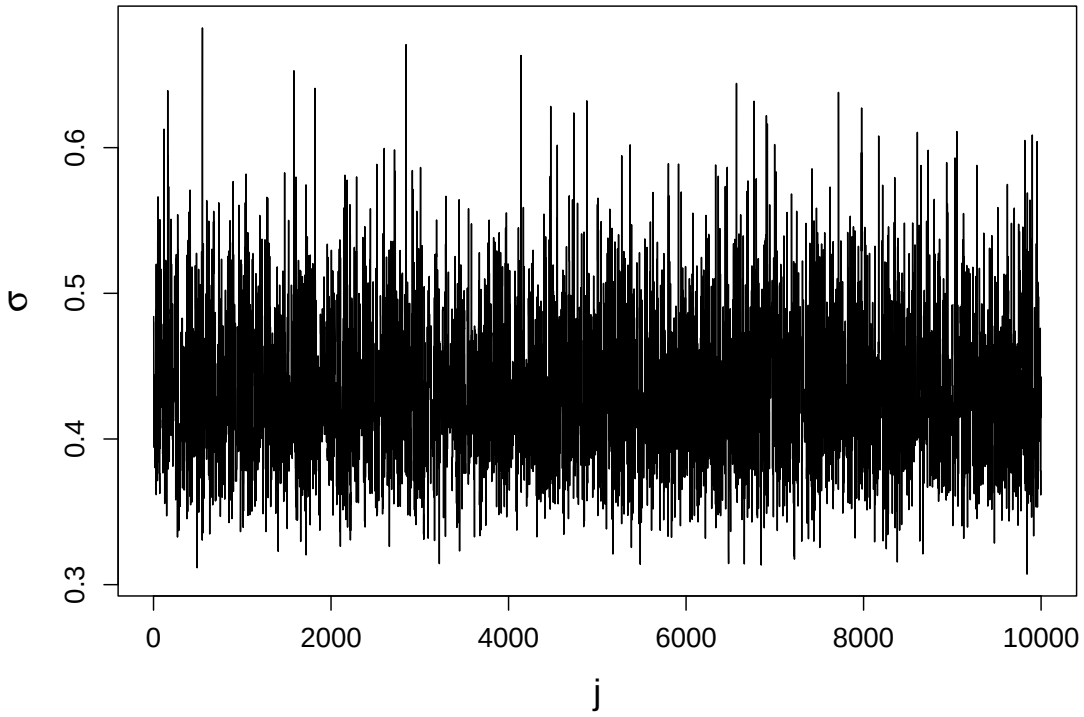


Figure A.55: Markov chain of parameter
x obtained by the Metropolis-Hastings algorithm as the true posterior sample.

A.4 SLCP

Figures A.56 - A.65 shows pair plots of SLCP posterior samples from the last iteration (r=4) of each algorithm for each of the 10 observed data sets in SBIBM.

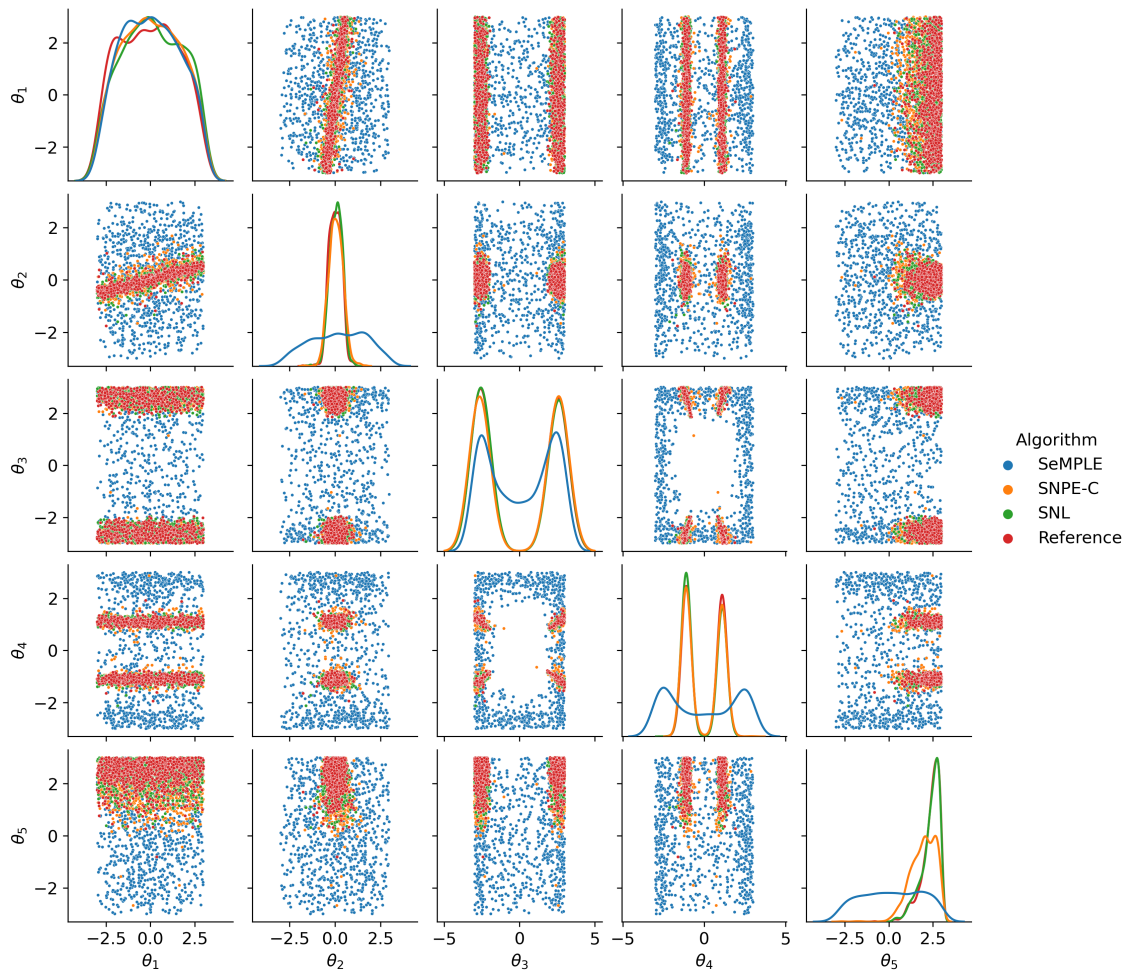


Figure A.56: Pair plot of SLCP posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 1 in SBIBM.

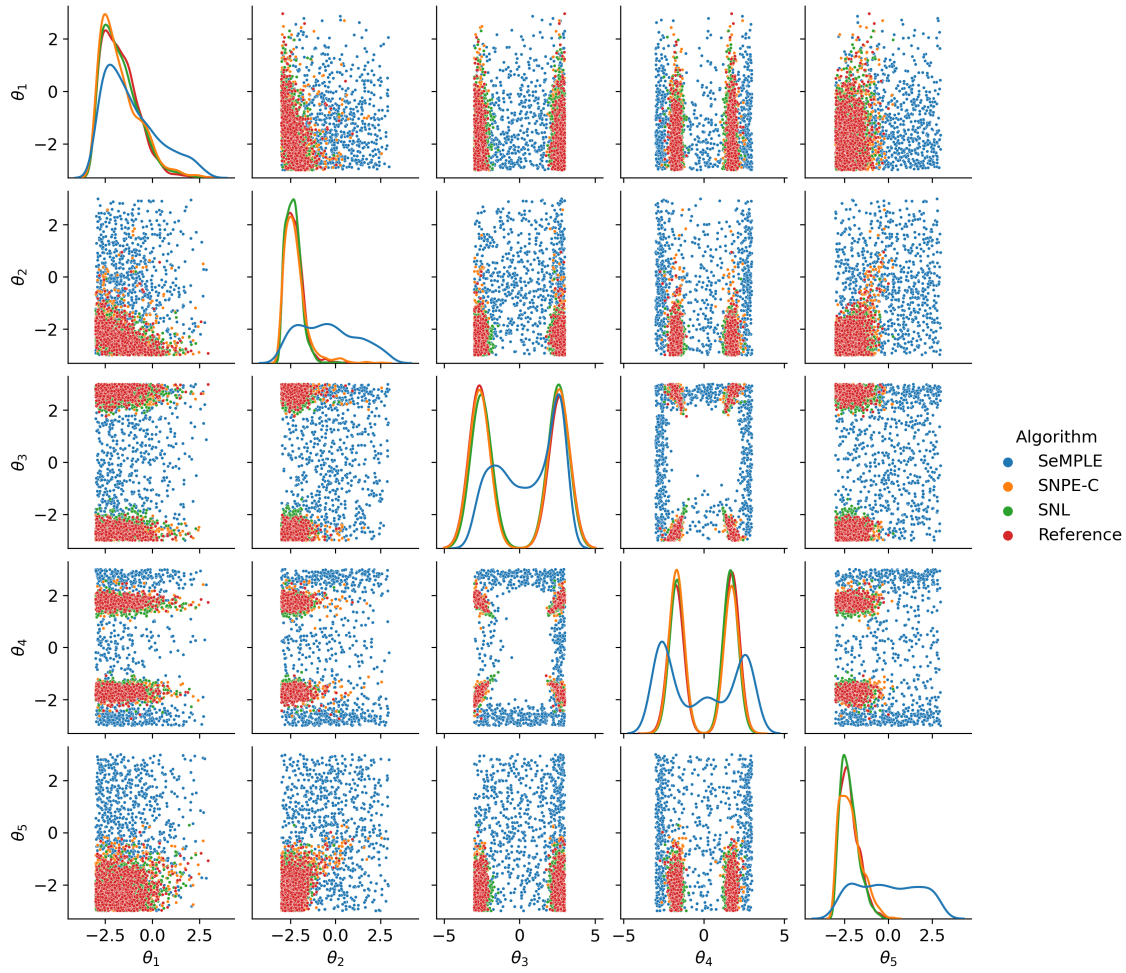


Figure A.57: Pair plot of SLCP posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 2 in SBIBM.

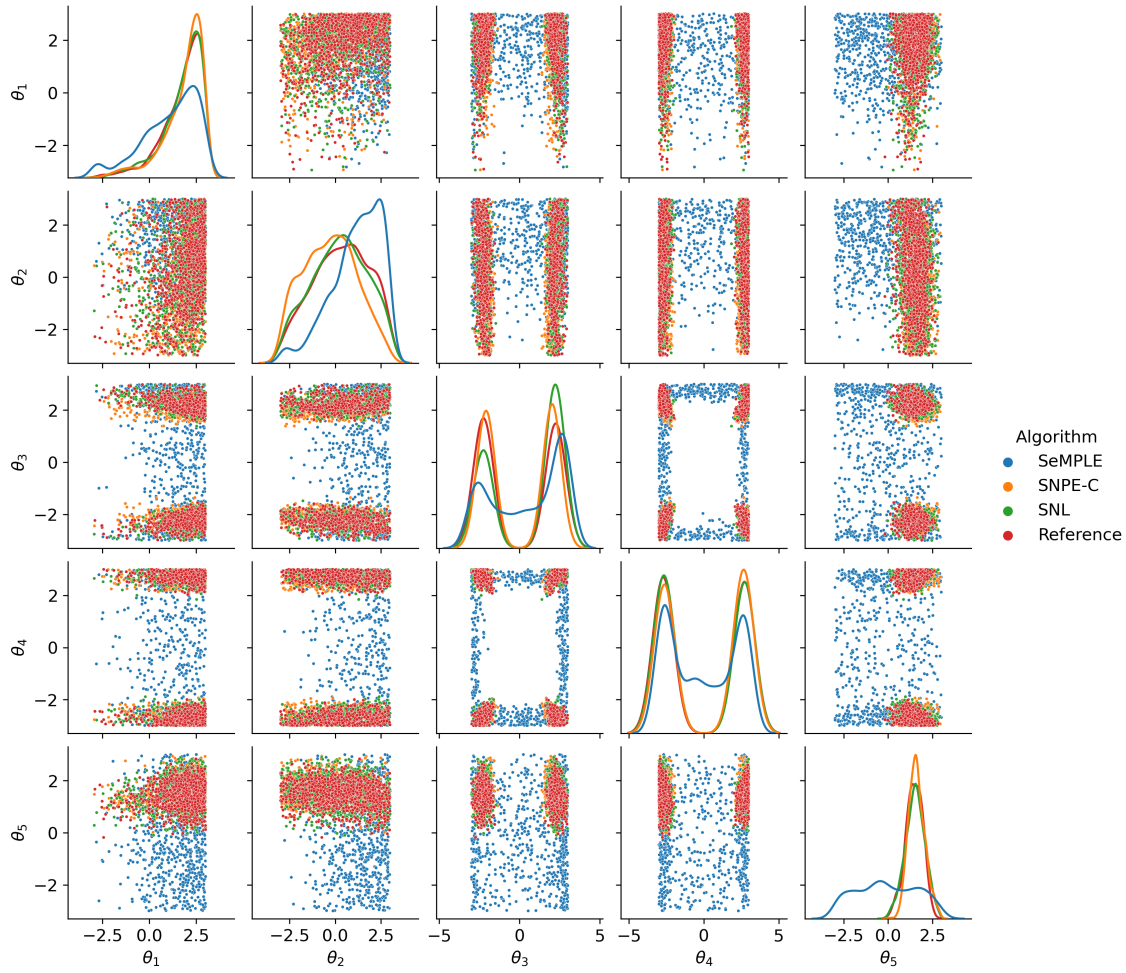


Figure A.58: Pair plot of SLCP posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 3 in SBIBM.

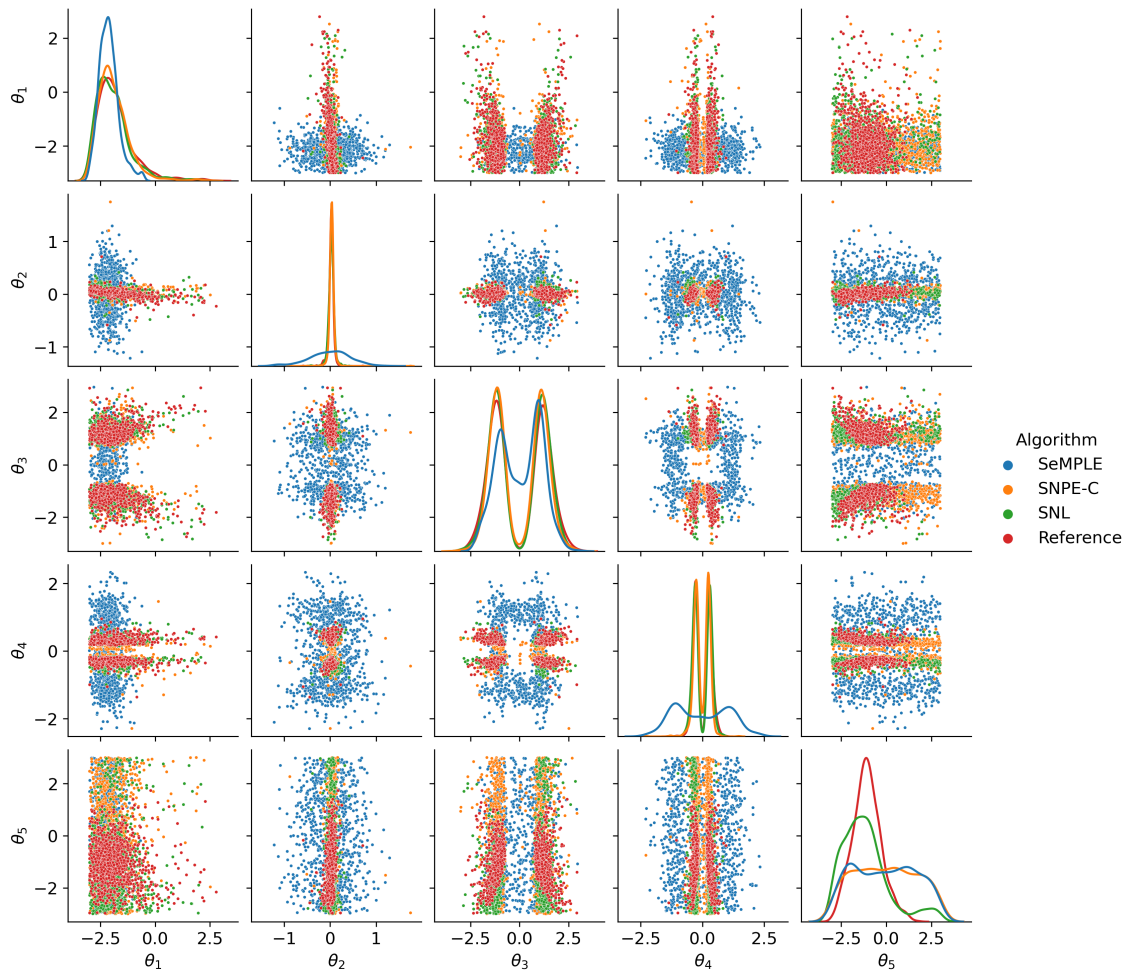


Figure A.59: Pair plot of SLCP posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 4 in SBIBM.

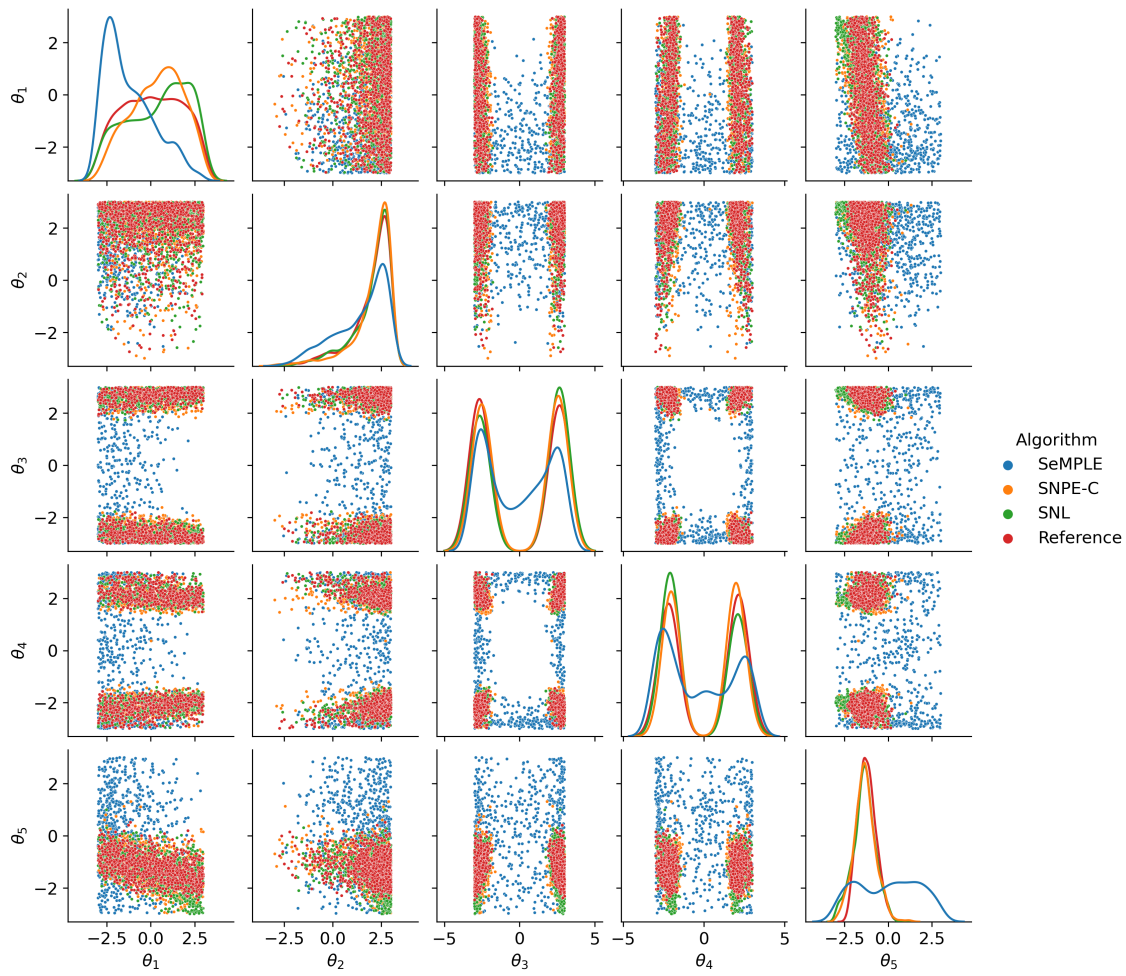


Figure A.60: Pair plot of SLCP posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 5 in SBIBM.

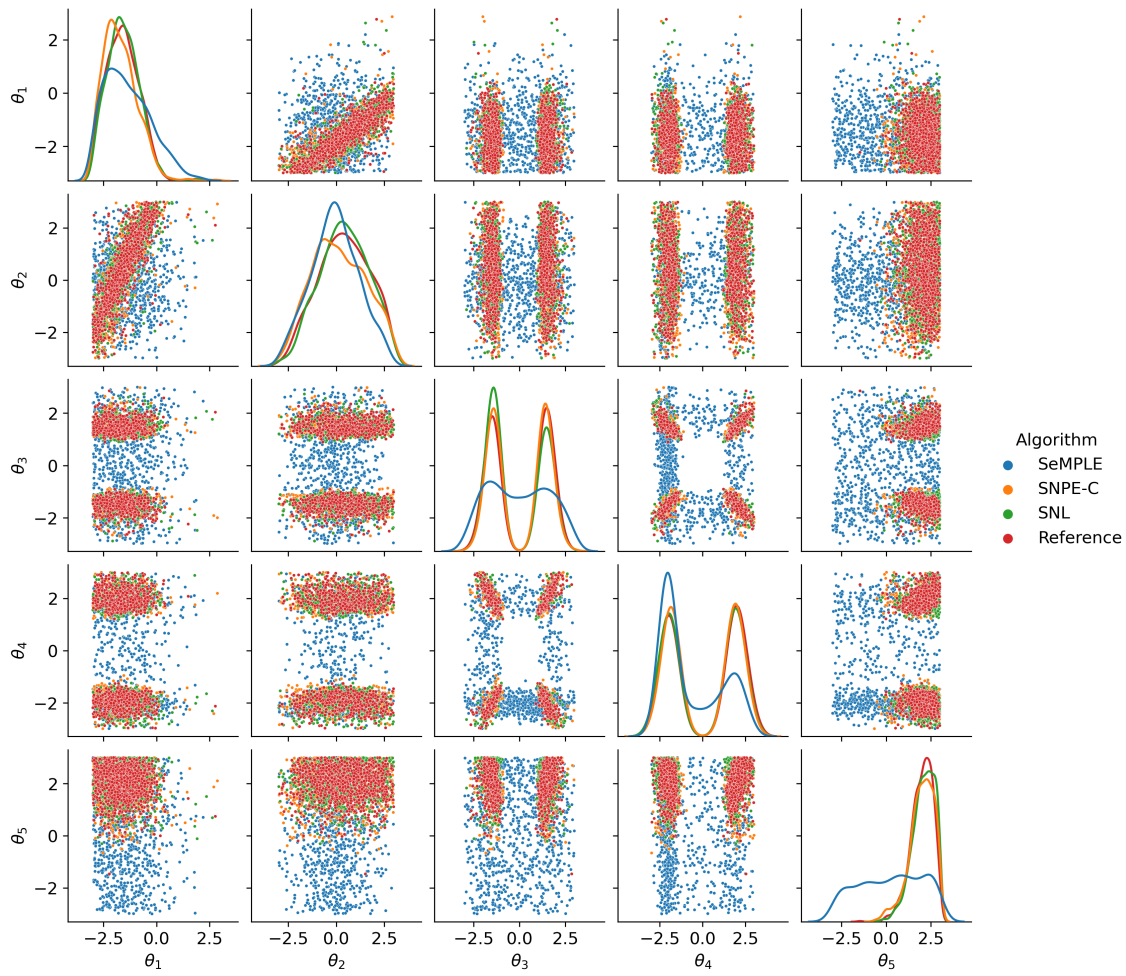


Figure A.61: Pair plot of SLCP posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 6 in SBIBM.

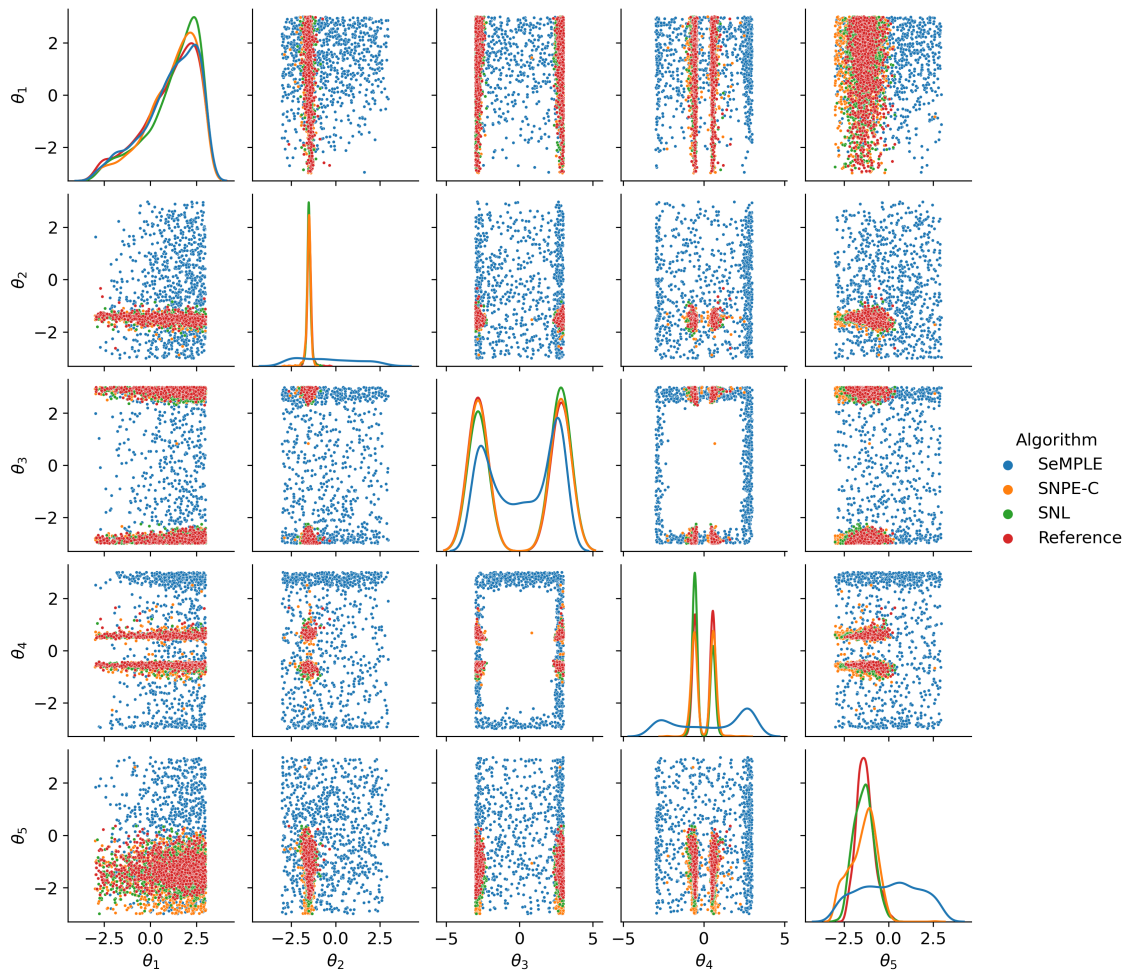


Figure A.62: Pair plot of SLCP posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 7 in SBIBM.

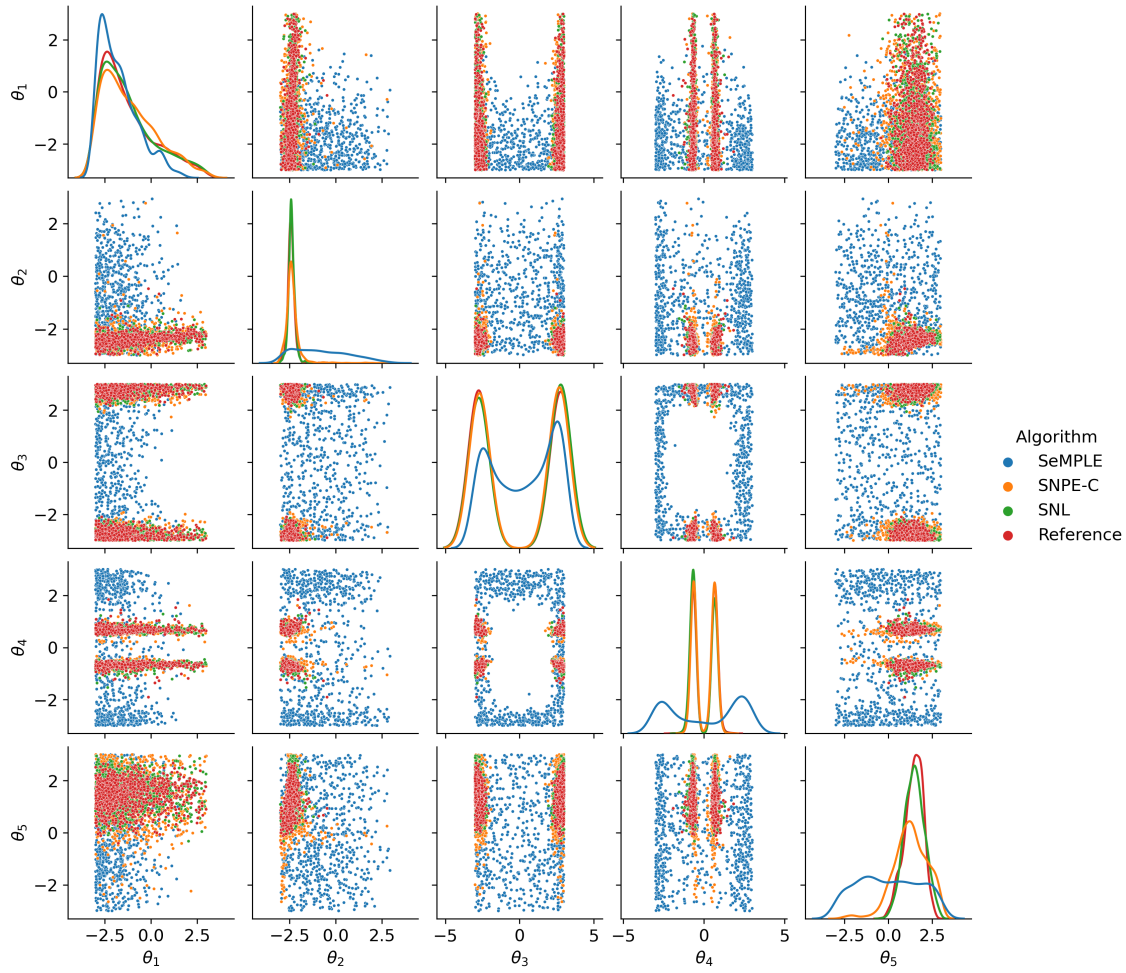


Figure A.63: Pair plot of SLCP posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 8 in SBIBM.

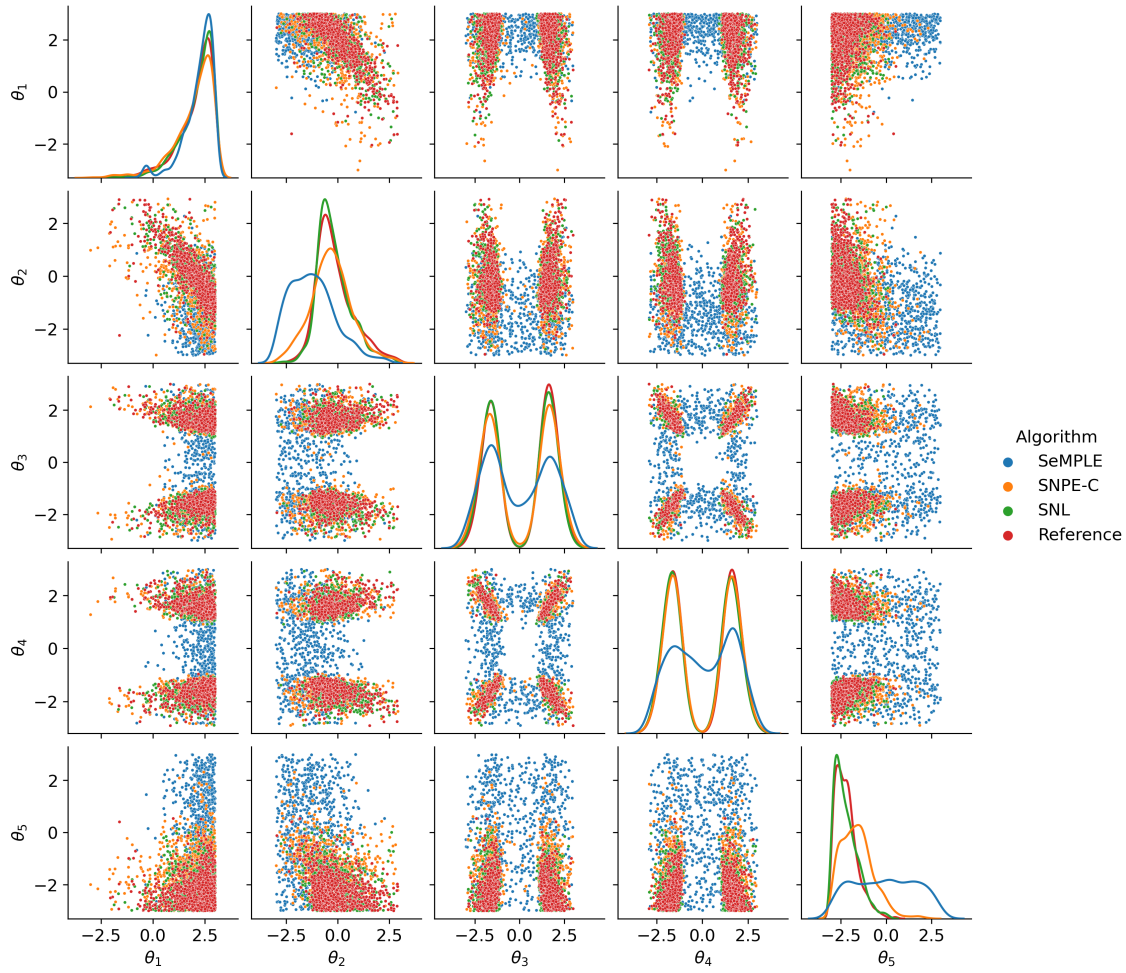


Figure A.64: Pair plot of SLCP posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 9 in SBIBM.

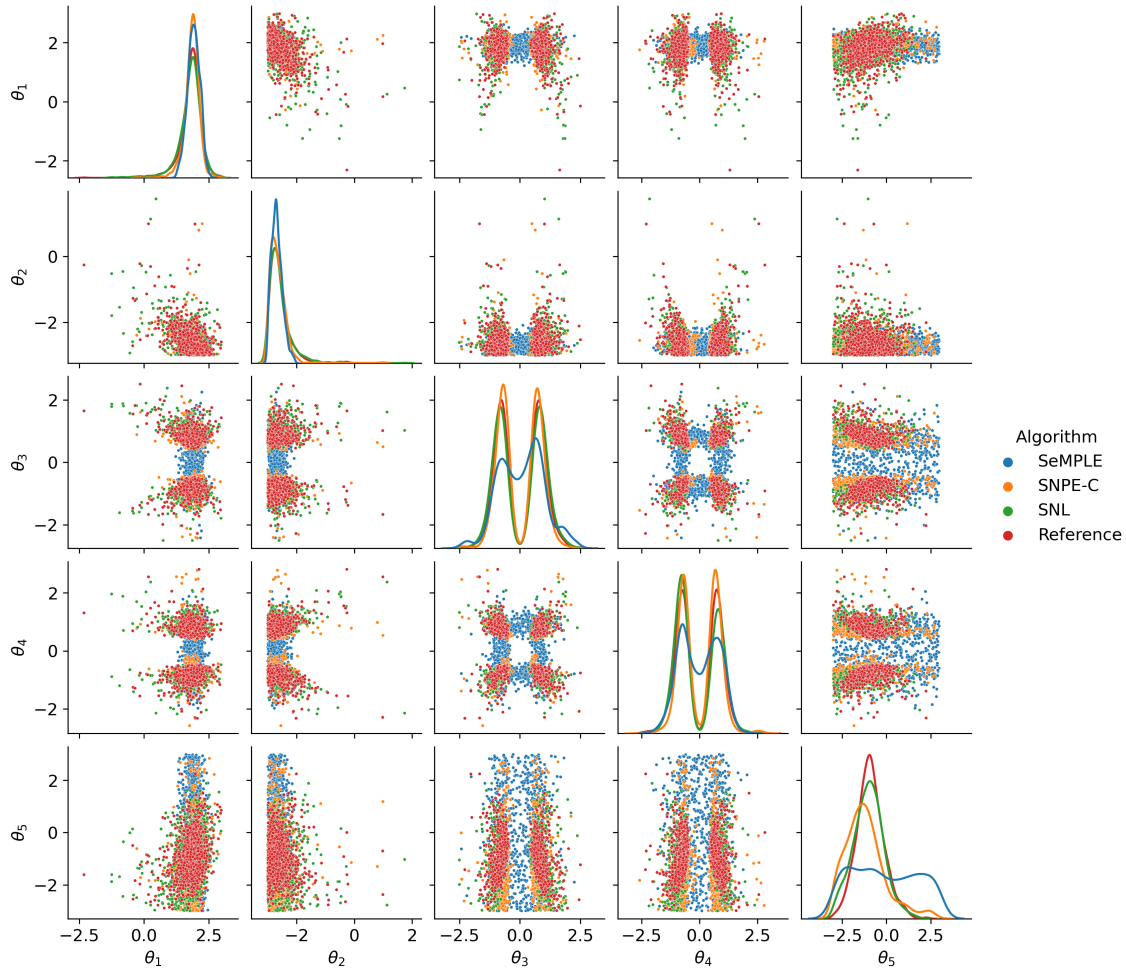


Figure A.65: Pair plot of SLCP posterior sample from the last iteration (r=4) of each algorithm. Observed data set number 10 in SBIBM.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY