Fast, accurate and lightweight sequential simulation-based inference using Gaussian locally linear mappings

Henrik Häggström *henhagg@chalmers.se*
Dept. Mathematical Sciences
Chalmers and University of Gothenburg

Pedro L. C. Rodrigues *pedro.rodrigues@inria.fr*
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, Inserm U1216, CHU Grenoble Alpes,
Grenoble Institut des Neurosciences, LJK

Geoffroy Oudoumanessah *Geoffroy.Oudoumanessah@inria.fr*
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, Inserm U1216, CHU Grenoble Alpes,
Grenoble Institut des Neurosciences, LJK
Univ. Lyon, CNRS, Inserm, INSA Lyon, UCBL, CREATIS

Florence Forbes *florence.forbes@inria.fr*
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, Inserm U1216, CHU Grenoble Alpes,
Grenoble Institut des Neurosciences, LJK

Umberto Picchini *picchini@chalmers.se*
Dept. Mathematical Sciences,
Chalmers and University of Gothenburg

Reviewed on OpenReview: `https://openreview.net/forum?id=QOnzpRcwWn`

Abstract

Bayesian inference for complex models with an intractable likelihood can be tackled using algorithms performing many calls to computer simulators. These approaches are collectively known as *simulation-based inference* (SBI). Recent SBI methods have made use of neural networks (NN) to provide approximate, yet expressive constructs for the unavailable likelihood function and the posterior distribution. However, the trade-off between accuracy and computational demand leaves much space for improvement. In this work, we propose an alternative that provides both approximations to the likelihood and the posterior distribution, using structured mixtures of probability distributions. Our approach produces accurate posterior inference when compared to state-of-the-art NN-based SBI methods, even for multimodal posteriors, while exhibiting a much smaller computational footprint. We illustrate our results on several benchmark models from the SBI literature and on a biological model of the translation kinetics after mRNA transfection.

1 Introduction

We propose a novel methodology achieving state-of-the-art Bayesian inference for models with an intractable likelihood function. We illustrate, via several benchmark models, how our methodology returns accurate inference, even in presence of multimodal posteriors, with a small runtime. Moreover, for numerous simulation studies and applications, machine and statistical learning solutions show an ever growing performance, but are not generally evaluated in terms of resource and energy consumption. Although their environmental impact is more and more documented (Schwartz et al., 2020; Strubell et al., 2019; Thompson et al., 2022), the question of resource efficiency and energy consumption when learning model parameters is not as much studied as the quality of the inference itself, due to the multiple aspects to be taken into account and to the lack of clear metrics that limit the algorithms evaluation. When comparing with widely employed inference engines based on neural-networks, our method considerably lower the energy and memory required for the inference task, while showing competitive inference quality.

We consider simulation-based inference (SBI) for model parameters $\boldsymbol{\theta}$ in stochastic modelling. SBI refers to approaches that bypass the need to have a likelihood function $p(\boldsymbol{y}|\boldsymbol{\theta})$ (for data $\boldsymbol{y}$) that is available in closed form, or one that is cheap to approximate, by instead considering a generative model $f(\boldsymbol{\theta})$ or *computer simulator*. The simulator depends on $\boldsymbol{\theta}$ and possibly on other inputs such as covariates and a stream of pseudorandom numbers. Generating a dataset $\boldsymbol{y}^*$ by running the simulator $\boldsymbol{y}^* = f(\boldsymbol{\theta}^*)$ at $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ is then equivalent to simulating $\boldsymbol{y}^*$ from the unavailable likelihood function. In the following this simulation is written as $\boldsymbol{y}^* \sim p(\boldsymbol{y}|\boldsymbol{\theta}^*)$. Established SBI methods include approximate Bayesian computation (ABC, reviewed in Sisson et al., 2018), synthetic likelihoods (Wood, 2010; Price et al., 2018), particle Markov chain Monte Carlo (Andrieu & Roberts, 2009; Andrieu et al., 2010), and several more that we discuss in Section 2. More recently, methods exploiting neural networks (NN) have gained considerable attention. Neural-based estimators have been used to produce approximated likelihoods or approximated posterior distributions, typically using (discrete) normalizing flows (Rezende & Mohamed, 2015; Papamakarios et al., 2021). NN have been used to approximate the likelihood function in implicit models (Papamakarios et al., 2019; Chen et al., 2021), the posterior distribution (Papamakarios & Murray, 2016; Greenberg et al., 2019; Durkan et al., 2020; Chen et al., 2021) or the likelihood and the posterior distribution simultaneously (Wiqvist et al., 2021; Radev et al., 2023a).

These neural-based posterior and likelihood estimation methods have produced exceptionally flexible inference strategies for models with intractable likelihoods. In this work, we propose to investigate more *frugal* strategies (Evchenko et al., 2021) that can run with limited resources and a much smaller computational footprint, while returning inference of similar quality as neural-based approaches. For SBI, we propose to investigate structured mixtures of probability distributions whose versatility has been widely recognized for a variety of data and tasks, while not requiring excessive design effort or tuning. Their expressivity makes them good candidates to account for complex multivariate models both for the likelihood and posterior distribution. Moreover, mixture models have a much smaller number of parameters $\boldsymbol{\phi}$ (see Table 1), compared to NN, and this makes them more amenable to interpretation and efficient learning. More specifically, we design a new simulation-based inference method named "Sequential Mixture Posterior and Likelihood Estimation" (SeMPLE). SeMPLE learns approximations for both the posterior $p(\boldsymbol{\theta}|\boldsymbol{y})$ and the likelihood $p(\boldsymbol{y}|\boldsymbol{\theta})$ using a structured mixture model obtained via the Gaussian Locally Linear Mapping (GLLiM, Deleforge et al., 2014) method. Interestingly, both approximations are jointly returned, simultaneously, following a run of an Expectation-Maximization (EM) algorithm within GLLiM. We then use the GLLiM approximate posterior as a well-informed proposal distribution in a tuning-free Metropolis-Hastings sampler. The procedure is embedded in a sequential strategy, which we compare with state-of-the-art NN-based sequential algorithms for posterior inference, namely SNL (Papamakarios et al., 2019) and SNPE-C (Greenberg et al., 2019). Other algorithms could be considered for comparison, however we stick with SNL and SNPE-C and a motivation for this is given in Section 2. Our comparisons are in terms of inference quality as well as resource and energy requirements. We did not compare against ABC samplers such as SMC-ABC, except for the Lotka-Volterra example where such comparison was more suitable, as in Lueckmann et al., 2021 it was shown that standard implementations of ABC approaches usually require a larger number of model simulations, compared to NN-based approaches, although recent research improved the SMC-ABC performance (Picchini & Tamborrino, 2024). We found that, for a given computational budget specified by the total number of model simulations, our method is cheaper to run and produces accurate inference thanks to (i) the informed proposal sampler created by SeMPLE and (ii) the fast model training. As an illustration, Figure 1 shows inference for three datasets of the *Two Moons* model, a model with bimodal posterior that we further analyse in Section 5. The runtime, energy (consumed by the CPU and the DRAM) and memory requirements for SeMPLE are considerably lower, see Table 1, Figure 2 (right) and section 5.5 for details on how energy and memory requirements were measured. In other examples, (*e.g.* Figure 3), the speedup provided by SeMPLE is even more dramatic (for the model in Supplementary section G.4 we show that SeMPLE is 74 times faster then SNL and 26 times faster than SNPE-C). Overall SeMPLE shows low resources costs that are always of the same magnitude (< 10 kJ and < 1 GB), while the requirements for SNL and SNPE-C vary much more with the model dimension or the inference complexity. These results highlight the advantage brought by SeMPLE, which can be run on a minimal configuration while maintaining good performance.

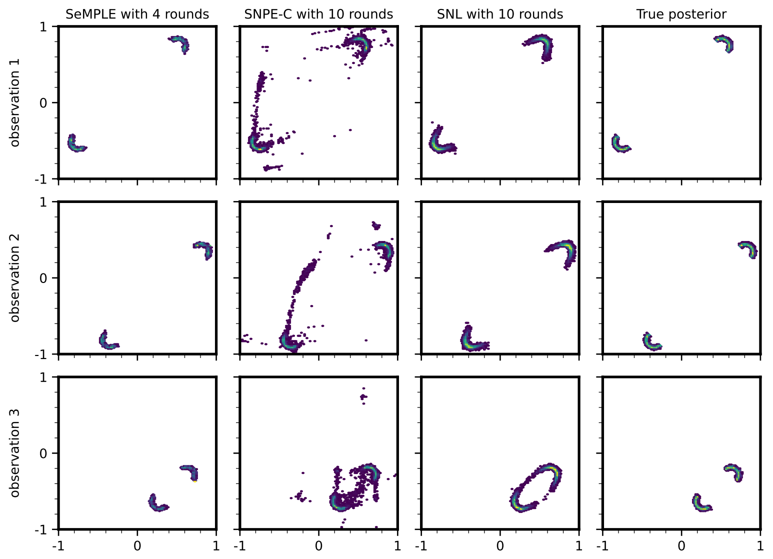


Figure 1: Two Moons model: inference using datasets #1 (top), #2 (middle) and #3 (bottom) from `SBIBM`. Posterior samples after 4 rounds of SeMPLE (column 1), 10 rounds of SNPE-C and SNL (columns 2 and 3). All methods used a total budget of 10K model simulations. The true posteriors are in column 4.

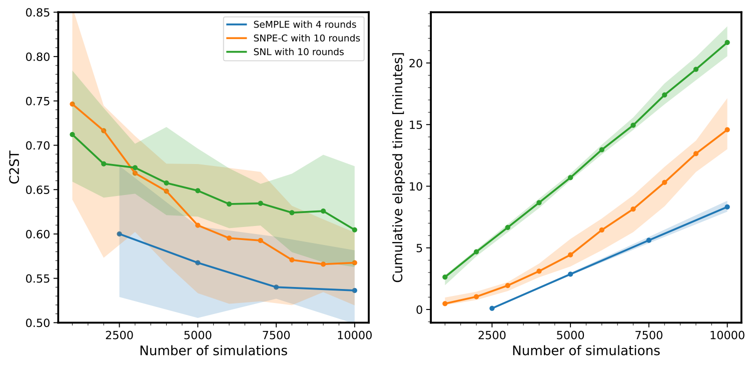


Figure 2: Two Moons. (left) Median C2ST (the lower the better) and median cumulative runtime in minutes (right) for 10 runs with different data sets vs the number of model simulations. Shaded bands enclose the min and max values. C2ST is the two-samples classifier test, taking values in [0.5,1], see Section 5.

Section 2 gives a brief overview of the main SBI approaches. Section 3 describes how GLLiM can be used to construct surrogate posteriors and surrogate likelihoods, while the complete SeMPLE methodology is explained in Section 4. Results from SeMPLE, SNL and SNPE-C are then illustrated in Section 5. Test models, partly reported in the Appendices, include simulation studies comprising multimodal surfaces (two moons, multiple hyperboloid), stochastic differential equations (Ornstein-Uhlenbeck), generalised linear models (Bernoulli GLM), a Markov jump process with additional measurement error (Lotka-Volterra), highly correlated posteriors with 20 parameters (twisted-prior), and a challenging and partially-observed biological model of the translation kinetics after mRNA transfection. Code is available at `https://github.com/henhagg/semple`.

Table 1: Median resource requirements (across 10 independent repetitions) and number of parameters ϕ for posterior inference with a budget of 10K (two moons and Bernoulli GLM) and 40K (hyperboloid and Ornstein-Uhlenbeck) model simulations. Best (lowest) values are in bold. SNPE-C and SNL used the default `SBIBM` setup described in Appendix A.

Method	Energy Consumption [kJ]	Peak Memory Usage [GB]	#parameters ϕ
Two moons			
SNL	24.5	19	28,020
SNPE-C	15	9	60,925
SeMPLE	**7.1**	**0.71**	**449**
Hyperboloid			
SNL	54	30	30,020
SNPE-C	120	15	66,925
SeMPLE	**3.2**	**0.88**	**1,479**
Bernoulli GLM			
SNL	108	75	36,100
SNPE-C	41	10	98,025
SeMPLE	**4.8**	**0.93**	**2,309**
Ornstein-Uhlenbeck			
SNL	66	39	41,030
SNPE-C	134	15	102,242
SeMPLE	**6.1**	**0.93**	**30,799**

2 Related work

Simulation-based inference, also referred to as *likelihood-free inference*, is reviewed *e.g.* in Cranmer et al. (2020) and Pesonen et al. (2023). In a Bayesian framework the focus is on posterior estimation, which in SBI directly relates to the quality of the approximations made to the unavailable likelihood, the posterior distribution or both. This aspect raises the issue of how to evaluate the quality of posterior inference when a reference ground truth posterior is absent by definition. This is still an open question and is further discussed in Section 6. A second important ingredient is sampling efficiency, as simulation-based methods are typically relying on exploration, and on Monte Carlo principles that can be prohibitively expensive. Active or sequential learning is a way to address sample efficiency by gradually discovering and guiding where simulations should be made for a specific given observation. Our proposed method is sequential and we focus on comparisons with NN-based sequential methods for posterior inference, namely SNL (Papamakarios et al., 2019) and SNPE-C (Greenberg et al., 2019). Many other approaches exist for Bayesian inference with intractable likelihoods, such as sequential neural ratio estimation (SNRE, Hermans et al., 2020), the several variants of ABC such as SMC-ABC (Toni et al., 2009; Del Moral et al., 2012), synthetic likelihoods (Wood, 2010; Price et al., 2018) and particle MCMC (Andrieu & Roberts, 2009; Andrieu et al., 2010). Comparisons between several of the above mentioned methods are available in previous works. For example, Greenberg et al. (2019) show that SNPE-C is computationally more efficient than the predecessors[1] SNPE-A (Papamakarios & Murray, 2016) and SNPE-B (Lueckmann et al., 2017), where *efficient* means that, for a given computational budget, SNPE-C approaches the true posterior using a smaller number of model simulations. Lueckmann et al. (2021) show that, for nontrivial case-studies, both SNPE-C and SNRE are more efficient than SNL and SMC-ABC. Moreover, SNPE-C and SNRE perform similarly. The above motivates why in this work we focus on the comparison with SNPE-C using the `SBIBM` package implementation (Lueckmann et al., 2021) (an exception is the Lotka-Volterra Markov jump process, where we compare against SMC-ABC, as an objective comparison with NN-based methods is difficult here, see section 5.3 for more details). Since

[1]The acronyms SNPE-A and SNPE-B follow the taxonomy in Papamakarios et al. (2019).

SeMPLE includes a Markov chain Monte Carlo (MCMC) step, same as SNL, it is also relevant to compare against SNL, using its implementation in `SBIBM`, which is an ameliorated one, compared to the original version of Papamakarios et al. (2019).

Note that there are crucial differences between our SeMPLE and SNPE-A, as detailed in Appendix C. Briefly, in SNPE-A a K component mixture density network (MDN) is used as a proposal sampler. As such, the MDN means and covariances are parametrised via neural networks. However SNPE-A does not have a built-in strategy to learn each of the K components. Instead, the GLLiM procedure embedded into SeMPLE automatically determines the means and covariance matrices for each of the K components, thus increasing flexibility to the shapes of the posteriors that can be targeted (see also Wang et al., 2018, where MDNs are employed as proposal functions within a Gibbs sampler). The fact that SeMPLE does not use MDNs, and more generally no NN architectures are employed in SeMPLE[2], it results in an easy to train framework, with fewer parameters to learn (see Table 1). Other architectures using NN within proposal samplers are e.g. the *neural proposal* of Stites et al. (2021) and the mixture of Gaussian copulas of Chen & Gutmann (2019).

Recent work that is relevant for SBI, but is less comparable with our work, includes amortized strategies. Amortization is a property that refers to the ability of algorithms to make use of previous computational efforts, not to repeat computationally expensive steps for each new observation considered. Examples in this class are the approaches in Radev et al. (2023a), Radev et al. (2023b), Dyer et al. (2022), Gao et al. (2023). Although GLLiM is by design an amortized procedure, this property is partly lost when used in a sequential approach such as SeMPLE, as amortization and sequential learning are somewhat opposite objectives. The next section gives a brief introduction to the Gaussian local linear mapping (GLLiM) procedure, which allows to obtain amortized surrogates of the posterior and of the likelihood in closed form.

3 Amortized surrogates for likelihoods and posteriors via GLLiM

The *inverse regression* approach via Gaussian local linear mapping (GLLiM) was first introduced in Deleforge et al. (2014). GLLiM is a parametric mixture model on the joint distribution of $(\boldsymbol{y}, \boldsymbol{\theta})$, denoted by $q_{\tilde{\boldsymbol{\phi}}}$ and depending on a set of parameters $\tilde{\boldsymbol{\phi}}$. For interpretation or visualization, the model can also be presented as an invertible regression model as follows. In GLLiM the relationship between observations $\boldsymbol{y} \in \mathbb{R}^d$ and $\boldsymbol{\theta} \in \mathbb{R}^l$ is assumed to be locally-linear with K components, defined through an additional latent variable $z \in \{1, ..., K\}$ as,

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

where $\mathbb{1}$ denotes the indicator function, and $\tilde{\boldsymbol{A}}_k \in \mathbb{R}^{d \times l}$ and $\tilde{\boldsymbol{b}}_k \in \mathbb{R}^d$ are matrices and vectors, respectively, defining the affine transformations, while $\tilde{\boldsymbol{\epsilon}}_k \in \mathbb{R}^d$ corresponds to an error term capturing both the observational noise and the modelling error due to assuming an affine approximation for the data. In the following we consider Gaussian noise, $\tilde{\boldsymbol{\epsilon}}_k \sim \mathcal{N}_d(\mathbf{0}, \tilde{\boldsymbol{\Sigma}}_k)$, and assume $\tilde{\boldsymbol{\epsilon}}_k$ to not depend on $\boldsymbol{\theta}$, $\boldsymbol{y}$ nor z. Therefore, conditionally on component $z = k$, an approximate generative model (while the assumed generative model is denoted $p(\boldsymbol{y}|\boldsymbol{\theta})$) is given by

$$q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{y} \mid \boldsymbol{\theta}, z = k) = \mathcal{N}_d(\boldsymbol{y}; \tilde{\boldsymbol{A}}_k \boldsymbol{\theta} + \tilde{\boldsymbol{b}}_k, \tilde{\boldsymbol{\Sigma}}_k). \tag{2}$$

To complete the hierarchical model, $\boldsymbol{\theta}$ is assumed to follow a mixture of Gaussian distributions specified by

$$q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{\theta} \mid z = k) = \mathcal{N}_l(\boldsymbol{\theta}; \tilde{\boldsymbol{c}}_k, \tilde{\boldsymbol{\Gamma}}_k), \tag{3a}$$

$$q_{\tilde{\boldsymbol{\phi}}}(z = k) = \pi_k. \tag{3b}$$

The GLLiM model hierarchical construction above (eq.(1) to (3b)) defines then a joint Gaussian mixture model on $(\boldsymbol{y}, \boldsymbol{\theta})$:

$$q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{y}, \boldsymbol{\theta}) = \sum_{k=1}^{K} q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{y}|\boldsymbol{\theta}, z=k) q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{\theta}|z=k) q_{\tilde{\boldsymbol{\phi}}}(z=k), \tag{4}$$

[2]While NN could potentially be incorporated into SeMPLE for the purpose of obtaining summary statistics of the data, as done in eg ABC, this is a separate problem that is not considered in the present work.

where the full vector of mixture model parameters is

$$\tilde{\boldsymbol{\phi}} = \{\pi_k, \tilde{c}_k, \tilde{\boldsymbol{\Gamma}}_k, \tilde{\boldsymbol{A}}_k, \tilde{\boldsymbol{b}}_k, \tilde{\boldsymbol{\Sigma}}_k\}_{k=1}^K. \tag{5}$$

From this joint model, we can deduce straightforwardly both its conditional distributions $q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{y} \,|\, \boldsymbol{\theta})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \boldsymbol{y})$ in closed-form. First, it comes that,

$$q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{y} \,|\, \boldsymbol{\theta}) = \sum_{k=1}^K \tilde{\eta}_k(\boldsymbol{\theta}) \mathcal{N}_d(\boldsymbol{y}; \tilde{\boldsymbol{A}}_{\boldsymbol{k}}\boldsymbol{\theta} + \tilde{\boldsymbol{b}}_k, \tilde{\boldsymbol{\Sigma}}_k), \tag{6}$$

with

$$\tilde{\eta}_k(\boldsymbol{\theta}) = \frac{\pi_k \mathcal{N}_l(\boldsymbol{\theta}; \tilde{\boldsymbol{c}}_k, \tilde{\boldsymbol{\Gamma}}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}_l(\boldsymbol{\theta}; \tilde{\boldsymbol{c}}_j, \tilde{\boldsymbol{\Gamma}}_j)}, \tag{7}$$

and similarly,

$$q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \boldsymbol{y}) = \sum_{k=1}^K \eta_k(\boldsymbol{y}) \mathcal{N}_l(\boldsymbol{\theta}; \boldsymbol{A}_k \boldsymbol{y} + \boldsymbol{b}_k, \boldsymbol{\Sigma}_k), \tag{8}$$

with

$$\eta_k(\boldsymbol{y}) = \frac{\pi_k \mathcal{N}_d(\boldsymbol{y}; \boldsymbol{c}_k, \boldsymbol{\Gamma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}_d(\boldsymbol{y}; \boldsymbol{c}_j, \boldsymbol{\Gamma}_j)}, \tag{9}$$

where $\boldsymbol{\phi} = \{\pi_k, \boldsymbol{c}_k, \boldsymbol{\Gamma}_k, \boldsymbol{A}_k, \boldsymbol{b}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ is just a convenient equivalent reparameterization that can easily be deduced from $\tilde{\boldsymbol{\phi}}$ (Deleforge et al., 2014) as

$$\boldsymbol{c}_k = \tilde{\boldsymbol{A}}_k \tilde{\boldsymbol{c}}_k + \tilde{\boldsymbol{b}}_k \tag{10a}$$
$$\boldsymbol{\Gamma}_k = \tilde{\boldsymbol{\Sigma}}_k + \tilde{\boldsymbol{A}}_k \tilde{\boldsymbol{\Gamma}}_k \tilde{\boldsymbol{A}}_k^\top \tag{10b}$$
$$\boldsymbol{\Sigma}_k = \left(\tilde{\boldsymbol{\Gamma}}_k^{-1} + \tilde{\boldsymbol{A}}_k^\top \tilde{\boldsymbol{\Sigma}}_k^{-1} \tilde{\boldsymbol{A}}_k\right)^{-1} \tag{10c}$$
$$\boldsymbol{A}_k = \boldsymbol{\Sigma}_k \tilde{\boldsymbol{A}}_k^\top \tilde{\boldsymbol{\Sigma}}_k^{-1} \tag{10d}$$
$$\boldsymbol{b}_k = \boldsymbol{\Sigma}_k \left(\tilde{\boldsymbol{\Gamma}}_k^{-1} \tilde{\boldsymbol{c}}_k - \tilde{\boldsymbol{A}}_k^\top \tilde{\boldsymbol{\Sigma}}_k^{-1} \tilde{\boldsymbol{b}}_k\right). \tag{10e}$$

The modelling power of GLLiM comes essentially from the fact that both conditional distributions above are mixtures of Gaussian affine experts (MoE), which are known for their modelling and approximation capabilities (Nguyen et al., 2019; 2021; 2023). As such, they are good candidates to provide surrogate likelihoods and posteriors. All the more so as, given a training dataset made of many pairs of $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$, surrogate models specified via GLLiM can be estimated via an EM algorithm, see Deleforge et al. (2014) and Forbes et al. (2022) for details. In fact a surrogate likelihood and posterior can be obtained with $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$ prior-predictive realizations simulated from $p(\boldsymbol{\theta})p(\boldsymbol{y} \,|\, \boldsymbol{\theta})$, but more generally, as exploited in SeMPLE, GLLiM can be used in a sequential way, by changing the training set to include posterior draws.

Once an estimate for the parameter vector $\tilde{\boldsymbol{\phi}}$ (or equivalently $\boldsymbol{\phi}$) is available, we have an *amortized* procedure providing a surrogate likelihood function $q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{y} \,|\, \boldsymbol{\theta})$, which can be evaluated for any data $\boldsymbol{y}$. In particular, if we specify $\boldsymbol{y}$ to be a given *observed dataset* $\boldsymbol{y}_o$, and plug $\boldsymbol{y}_o$ into (6), we obtain an approximate (surrogate) likelihood function $q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{y}_o \,|\, \boldsymbol{\theta})$ for observation $\boldsymbol{y}_o$. Similarly, and using the same training data, an amortized surrogate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \boldsymbol{y})$, for a generic $\boldsymbol{y}$, can also be derived as a mixture of Gaussian experts distributions from (8). If we plug $\boldsymbol{y}_o$ into (8), then $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \boldsymbol{y}_o)$ is an approximate posterior based on observed data. In summary, once $\tilde{\boldsymbol{\phi}}$ is estimated via EM, we obtain surrogates of both the likelihood and the posterior in the same GLLiM run, simultaneously and using the same training data. The way this is performed and integrated with the rest of the SeMPLE method is explained in Section 4. Note that it is possible to assume a different distribution for $\tilde{\boldsymbol{\epsilon}}_k$ than the Gaussian one. For example (1) can be a mixture of generalized Student distributions, resulting in the so-called SLLiM method (Perthame et al., 2018). Both GLLiM and SLLiM are implemented in the R package `xLLiM` (Perthame et al., 2022) which we employ in our experiments, see Section 5 for details. Notice that GLLiM has already been used in SBI, see Forbes et al. (2022), namely they produced an amortized sampler whose posterior draws were then "refined" via a single ABC step. However, in Forbes et al. (2022) only surrogate posteriors were investigated and, in addition, they did not sequentially refine the surrogate posterior and surrogate likelihood. In contrast, our SeMPLE approach makes use of both posterior and likelihood surrogates, to then provide an automatically "tuned" proposal function for MCMC (as described in Section 4) but also to provide a surrogate likelihood, which finds excellent use in the MCMC we construct in Section 4. Then, SeMPLE sequentially refines, in multiple rounds, both the surrogate likelihood and the surrogate posterior (and hence the proposal function), without requiring an ABC step, and hence without the critical tuning/setup decisions that ABC inference entails (eg the choice of distances, threshold values, the determination of suitable proposal functions), and without the large number of simulations that ABC requires.

Another approach that, similarly to GLLiM, could be considered for closed-form approximations of densities without involving neural networks, is the "elliptical process" of Bånkestad et al. (2020), in particular when using their piecewise constant mixing distribution.

Algorithm 1 SeMPLE

```
1:  $\tilde{p}_0(\boldsymbol{\theta}) \leftarrow p(\boldsymbol{\theta})$, set starting $K$ and number of rounds $R$.
2:  for $r = 0 : R$ do
3:      if $r = 0$ or $r = 1$ then
4:          sample $\{\boldsymbol{\theta}_n\}_{n=1}^N$ iid $\boldsymbol{\theta}_n \sim \tilde{p}_r(\boldsymbol{\theta})$ (without MCMC), $n = 1, ..., N$.
5:      else
6:          sample $\{\boldsymbol{\theta}_n\}_{n=1}^N$ via $\boldsymbol{\theta}_n \sim \tilde{p}_r(\boldsymbol{\theta})$ using Algorithm 2.
7:      end if
8:      Conditionally on every $\boldsymbol{\theta}_n$, obtain $\{\boldsymbol{y}_n\}_{n=1}^N$ by sampling (implicitly) $\boldsymbol{y}_n \sim p(\boldsymbol{y} \mid \boldsymbol{\theta}_n)$ $(n = 1, ..., N)$.
9:      if r = 0 then
10:         Collect $\mathcal{D}_0 = \{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$.
11:         Train $q_{\tilde{\boldsymbol{\phi}}_0}(\boldsymbol{y} \mid \boldsymbol{\theta})$ on $\mathcal{D}_0$. Update $K$.
12:         Obtain $q_{\boldsymbol{\phi}_0}(\boldsymbol{\theta} \mid \boldsymbol{y})$.
13:         Set $\tilde{p}_1(\boldsymbol{\theta}) \leftarrow q_{\boldsymbol{\phi}_0}(\boldsymbol{\theta} \mid \boldsymbol{y} = \boldsymbol{y}_o)$.
14:     else
15:         Collect $\mathcal{D}_r = \mathcal{D}_{r-1} \cup \{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N \setminus \mathcal{D}_0$.
16:         Train $q_{\tilde{\boldsymbol{\phi}}_r}(\boldsymbol{y} \mid \boldsymbol{\theta})$ on $\mathcal{D}_r$. Update $K$.
17:         Obtain $q_{\boldsymbol{\phi}_r}(\boldsymbol{\theta} \mid \boldsymbol{y})$.
18:         Set $\tilde{p}_{r+1}(\boldsymbol{\theta}) \leftarrow q_{\tilde{\boldsymbol{\phi}}_r}(\boldsymbol{y}_o \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})$
19:             $((optional)\ \tilde{p}_{r+1}(\boldsymbol{\theta}) \leftarrow \frac{p(\boldsymbol{\theta})}{\tilde{p}_r(\boldsymbol{\theta})} q_{\boldsymbol{\phi}_r}(\boldsymbol{\theta} \mid \boldsymbol{y}_o))$
20:     end if
21: end for
```

Algorithm 2 Independence MH with GLLiM surrogate posterior as proposal distribution

```
1:  For the initial value $\boldsymbol{\theta}_1$, pick the last accepted value from the currently available Markov chain.
2:  for $j = 2 : N$ do
3:      Propose $\boldsymbol{\theta}^* \sim q_{\boldsymbol{\phi}_{r-1}}(\boldsymbol{\theta} \mid \boldsymbol{y}_o)$,
4:      $\alpha = \min\{1, \frac{\tilde{p}_r(\boldsymbol{\theta}^*)}{\tilde{p}_r(\boldsymbol{\theta}_{j-1})} \frac{q_{\boldsymbol{\phi}_{r-1}}(\boldsymbol{\theta}_{j-1} \mid \boldsymbol{y}_o)}{q_{\boldsymbol{\phi}_{r-1}}(\boldsymbol{\theta}^* \mid \boldsymbol{y}_o)}\}$.
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
```

4 SeMPLE: sequential learning of likelihoods and posteriors

We have shown how GLLiM can provide amortized estimations of both the likelihood and the posterior distribution at no additional cost, given training data. This GLLiM feature is exploited in our proposed Sequential Mixture Posterior and Likelihood Estimation (SeMPLE), to obtain a sequence of increasingly improved samplers, to more accurately and more efficiently sample from the targeted posterior of observed data $\boldsymbol{y}_o$. Similarly to SNL (Papamakarios et al., 2019), given a likelihood approximation $q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{y}_o \,|\, \boldsymbol{\theta})$, SeMPLE targets an approximate posterior $\tilde{p}(\boldsymbol{\theta}) \propto q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{y}_o \,|\, \boldsymbol{\theta})p(\boldsymbol{\theta})$ with posterior sampling being carried out using a MCMC procedure. Typically, the MCMC scheme used in SNL is slice sampling (Neal, 2003), which has the advantage to be tuning-free but is likely to have difficulties in exploring high dimensional posteriors with complex multimodalities. As a matter of fact, Papamakarios et al. (2019) recommend SNL "at most for tens of parameters that do not have pathologically strong correlations". In SeMPLE, since GLLiM is able to learn a closed-form approximation of the posterior distribution (in addition to a likelihood function), such posterior is used as a proposal distribution within a Metropolis-Hastings (MH) algorithm. SeMPLE, by using an efficient EM procedure to fit a flexible proposal distribution, is expected to benefit from a MH step that better guides simulations towards more likely parameter values. To ease the comparison with SNL, the latter is recalled in Appendix D, while SeMPLE is summarized in Algorithms 1-2.

Algorithm 1 shows that the targeted approximate posterior for $\boldsymbol{\theta}$ is a function denoted with $\hat{p}_r(\boldsymbol{\theta}) \propto q_{\tilde{\boldsymbol{\phi}}_r}(\boldsymbol{y}_o \,|\, \boldsymbol{\theta})p(\boldsymbol{\theta})$ which is sequentially obtained, where $r = 0, ..., R$ denotes the SeMPLE *rounds* for the current approximation[3]. In the following we denote by $\boldsymbol{\theta}^{(r)}$ and $\boldsymbol{y}^{(r)}$ a parameter and an observation simulated in round r, respectively. At first ($r = 0$) $\hat{p}_0(\boldsymbol{\theta})$ is initialised at the prior $p(\boldsymbol{\theta})$. An initial batch of N pairs $\mathcal{D}_0 = \{\boldsymbol{\theta}_n^{(0)}, \boldsymbol{y}_n^{(0)}\}_{n=1}^N$ produced from the prior-predictive distribution (step 4) constitutes the starting training data for GLLiM, and the parameters $\tilde{\phi}_0$ and ϕ_0 are obtained via EM within GLLiM to produce the initial surrogate likelihood and posterior $q_{\tilde{\boldsymbol{\phi}}_0}(\boldsymbol{y} \,|\, \boldsymbol{\theta})$ and $q_{\boldsymbol{\phi}_0}(\boldsymbol{\theta} \,|\, \boldsymbol{y})$ respectively (steps 11-12). The initial surrogate posterior becomes the current target (step 13) $\tilde{p}_1(\boldsymbol{\theta}) \equiv q_{\boldsymbol{\phi}_0}(\boldsymbol{\theta} \,|\, \boldsymbol{y}_o)$. In the next round ($r = 1$), the latter can be sampled from without MH (step 4) since $\tilde{p}_1(\boldsymbol{\theta})$ is defined as a Gaussian mixture, from which it is trivial to sample N parameters $\{\boldsymbol{\theta}_n^{(1)}\}_{n=1}^N$. These N draws are then plugged into the model simulator to produce $\boldsymbol{y}_n^{(1)} \sim p(\boldsymbol{y} \,|\, \boldsymbol{\theta}_n^{(1)})$ and define $\mathcal{D}_1 = \{\boldsymbol{\theta}_n^{(1)}, \boldsymbol{y}_n^{(1)}\}_{n=1}^N$ as the new training data (we discard $\mathcal{D}_0$ for future rounds as it is deemed too uninformative for training). GLLiM is then fitted to $\mathcal{D}_1$ and the corresponding surrogate likelihood and surrogate posterior are obtained. From this step onwards, the surrogate posterior is employed as a proposal distribution within a tuning-free MH algorithm (Algorithm 2) since, for an arbitrary prior $p(\boldsymbol{\theta})$, when $r \geq 2$ we do not explicitly know which distribution is proportional to the product $p(\boldsymbol{\theta})q_{\tilde{\boldsymbol{\phi}}_r}(\boldsymbol{y}_o \,|\, \boldsymbol{\theta})$ (for $r = 1$ we know by construction that the targeted posterior is a Gaussian mixture), except for the special case of Gaussian and Uniform priors (see Papamakarios & Murray, 2016). In Algorithm 2, the acceptance probability of a proposed $\boldsymbol{\theta}^*$ is (for $r > 1$)

$$\alpha = \min\left\{1, \frac{\tilde{p}_r(\boldsymbol{\theta}^*)}{\tilde{p}_r(\boldsymbol{\theta}_{j-1})} \frac{q_{\boldsymbol{\phi}_{r-1}}(\boldsymbol{\theta}_{j-1} \,|\, \boldsymbol{y}_o)}{q_{\boldsymbol{\phi}_{r-1}}(\boldsymbol{\theta}^* \,|\, \boldsymbol{y}_o)}\right\}.$$

Samples from the posterior $\tilde{p}_r$ are obtained by first proposing parameters $\boldsymbol{\theta}^*$ from a Gaussian mixture model. From the current proposal function $q_{\phi_{r-1}}(\boldsymbol{\theta} \,|\, \boldsymbol{y}_o)$, which is the mixture

$$q_{\phi_{r-1}}(\boldsymbol{\theta} \,|\, \boldsymbol{y}_o) = \sum_{k=1}^K \eta_k^{(r-1)}(\boldsymbol{y}_o)\mathcal{N}_l(\boldsymbol{\theta}; \boldsymbol{A}_k^{(r-1)}\boldsymbol{y}_o + \boldsymbol{b}_k^{(r-1)}, \boldsymbol{\Sigma}_k^{(r-1)}),$$

a component k^* is randomly sampled from the set $\{1, ..., K\}$ with associated probabilities $\{\eta_1^{(r-1)}(\boldsymbol{y}_0), ..., \eta_K^{(r-1)}(\boldsymbol{y}_0)\}$. Then $\boldsymbol{\theta}^*$ is sampled from the k^*-th component $\mathcal{N}_l(\boldsymbol{\theta}; \boldsymbol{A}_{k^*}^{(r-1)}\boldsymbol{y} + \boldsymbol{b}_{k^*}^{(r-1)}, \gamma\boldsymbol{\Sigma}_{k^*}^{(r-1)})$ by (optionally) slightly inflating its covariance matrix with a factor $\gamma \geq 1$, which we set to $\gamma = 1$ (no inflation) or $\gamma \in [1.1, 1.2]$ in our experiments. The inflation factor is to ensure that the proposal is sampled from a broad-enough distribution covering the support of the target (no inflation factor is used when $r = 1$).

[3]We use the term *round* instead of the most common *iteration*, as the latter could be confused with the MCMC iterations that are run within a given SeMPLE round.

Also, note that the N retained draws via MH are N post-burnin draws, where the burnin consists of 100 iterations, and the last accepted draw from round r of SeMPLE is used to initialize the chain at round $r+1$.

Importantly, our MCMC procedure is tuning-free. We use an *independence sampler* (Robert & Casella, 2004), relying on GLLiM to provide an informative proposal function. This proposal function is trained on draws accepted in previous rounds which are, reasonably, more spread than the currently targeted posterior. In addition, being a mixture model, it is suitable to deal with multimodalities in the target. This greatly simplifies the construction of SeMPLE as no complex tuning is necessary for the MCMC sampler. While K could in principle be set arbitrarily large, in practice this may slow the EM algorithm convergence. In SeMPLE, the value of K is allowed to decrease between rounds, as expressed with "*update K*" in lines 11 and 16 of Algorithm 1. In fact, mixture components having probabilities $\tilde{\eta}_k$ smaller than a threshold are discarded (ie these gets removed from the mixture model). In most cases we find that setting a starting value of $K = 20$ or 30 produces good results, with a threshold for $\tilde{\eta}_k$ typically set to 0.005 (multiple hyperboloid, Ornstein-Uhlenbeck, Lotka-Volterra, biological SDE model), 0.03 (Bernoulli GLM model) or simply 0 for no deletion (Two Moons model). This is to prevent having too many irrelevant components, that could be problematic when fitting the mixture to training data produced via MCMC, as the latter may not manage to visit tiny modes. It is also possible to choose the starting value of K in a principled way. In Deleforge et al. (2014), the Bayesian Information Criterion (BIC) is used to select K, as we do in some of our examples, see Appendix B. An appropriate value for K can be learned by repeatedly computing the BIC on realizations of the prior-predictive distribution, before SeMPLE is started. This use of the BIC is an *amortized* procedure, as the selected values can be reused regardless the specific observed data $\boldsymbol{y_o}$. The impact on the overall computational budget is negligible. We have also experimented with a version of SeMPLE that learns a *corrected posterior* (see the *optional* part in step 19 of Algorithm 1) using the correction formula from Papamakarios & Murray (2016). However, we found that learning a surrogate likelihood was much more beneficial, providing better inference coupled with a higher acceptance rate. Notice that none of our reported results use the optional corrected-posterior approach.

In summary, a major difference with the several variants of sequential neural posterior estimation (where the most used one is SNPE-C, Greenberg et al., 2019), and with SNL, is that SeMPLE uses Gaussian mixtures learned via EM to approximate both the likelihood function and the posterior distribution of $\boldsymbol{\theta}$, as opposed to neural density estimation. Thus no NN architecture has to be designed, nor special stochastic gradient descent (SGD) setup has to be considered to fit NNs. Only the number of components K and the structure of the covariance matrices $\tilde{\boldsymbol{\Sigma}}_k$ (as the $\boldsymbol{\Sigma}_k$ are obtained as an algebraic by-product), according to the options offered by the `xLLiM` package[4], have to be set. As we show in our examples, inference based on procedures using SGD can be much more computationally demanding and time consuming.

5 Numerical illustrations

We illustrate SeMPLE on several simulation studies, see also the Appendices for extra results and additional models. SeMPLE is written in R and uses the GLLiM implementation from the `xLLiM` package (Perthame et al., 2022). Comparisons with neural-based likelihood and posterior estimation methods (SNL and SNPE-C) use the Python implementations in the `SBIBM` package for SBI benchmarking (Lueckmann et al., 2021), and architecture specifications for `SBIBM` are in Appendix A. To compare approximate posteriors with a suitable reference posterior, we consider the classifier two-samples test (C2ST), as implemented in `SBIBM`, and Wasserstein distances via the `POT` Python package (Flamary et al., 2021). C2ST varies in the interval [0.5,1], the lower the better. It equals 0.5 when samples from two distributions appear statistically indistinguishable and equals 1 when the classifier is able to perfectly separate samples as belonging to two different distributions. We also report runtimes in figures and resource requirements in Table 1. For SeMPLE, the time required to select K via BIC (when the BIC procedure is performed, which is not the case for all studies) is reported separately and not taken into account in the figures below. In fact, for some case studies, such as Lotka-Volterra, we merely picked a starting value of K without running the BIC procedure. When we used the BIC, adding this overhead time does not change the overall conclusion that SeMPLE is much

[4]The $\tilde{\boldsymbol{\Sigma}}_k$'s can be isotropic, diagonal or full matrices, and set all equal or varying with k.

more time-efficient than the other methods. Moreover, the selection of K is an amortized procedure, hence it does not need to be repeated for different observations of the same model.

In the next two sections 5.1 and 5.2, we illustrate results from two models having multimodal posteriors, in section 5.3 we summarise the Lotka-Volterra experiment, and in section 5.6 and Table 2 we summarise results from other experiments detailed in the Appendices. For comparison, all methods are given the same budget of model simulations. For SNL and SNPE-C the `SBIBM` default of distributing these simulations uniformly across $R = 10$ rounds is used, but to ease comparison we also run SNL and SNPE-C with the same number of rounds used for SeMPLE, which is $R = 4$, except for the Bernoulli GLM model where $R = 2$ suffices. In fact, we observe that SeMPLE does not need as many sequential steps as the other methods. It may be argued that SNL could perform better if a larger number of MCMC iterations was used, at the cost of a smaller number of rounds R. Results in the Appendices show that both SNL and SNPE-C perform worse under this alternative configuration. Regarding SeMPLE, when the computational budget limits the number of model simulations, we found it beneficial to keep the number of rounds R between 2 and 3, where $R = 2$ implies that we have exactly one round where MCMC is used, and this is typically already enough for satisfying inference, though usually an additional round with MCMC (hence $R = 3$) produces further refined inference. In our work we show results up to $R = 4$, but in general, four rounds are not recommended. The algorithm performance relies first on the GLLiM ability to learn reasonably accurate likelihood and posterior approximations and then on the MCMC sampler to explore the parameter space. Both steps require a large enough number of model simulations, which depends mainly on the model dimension.

5.1 Two Moons

The Two Moons model is a standard benchmark example in SBI (Greenberg et al., 2019; Lueckmann et al., 2021; Picchini & Tamborrino, 2024) to illustrate how algorithms deal with multimodality and the local crescent shapes of the posterior distribution. The model itself is defined in Appendix E. Our experiments are run on the same 10 datasets provided in `SBIBM`, each of them being a vector $\boldsymbol{y}_0$ of length two. We use $K = 30$, a value we determine as described in Appendix B (this determination only required 90 seconds). We run SeMPLE, SNL and SNPE-C algorithms for a total budget of 10^4 model simulations, uniformly distributed across inference rounds, and we used both $R = 4$ and $R = 10$ rounds. The obtained samples are then compared with the corresponding posterior reference samples (obtained via MCMC, since the likelihood is tractable) provided in `SBIBM`. The performance metric results are averaged over the 10 different data sets. Figure 2 shows C2ST and runtime values. SeMPLE provides the best approximation while SNL performs the worst. The comparison with SNL is particularly interesting as while both SNL and SeMPLE incorporate a tuning-free MCMC step, SeMPLE is able to visit both modes with an acceptance rate of 60-70%, while SNL often struggles in efficiently exploring a multimodal posterior as demonstrated by the higher C2ST values. Moreover, in this simple example the runtime for SeMPLE is about half the SNPE-C runtime, and is about three times lower than the SNL runtime. In other examples, (*e.g.* Figure 3), the speedup provided by SeMPLE is even more dramatic. All these benefits from SeMPLE come with a very small impact on the memory usage, see Section 5.5 for details.

5.2 Multiple hyperboloid model

The multiple hyperboloid model (Forbes et al., 2022) is constructed from a sound source localization problem in audio processing. This is a challenging inference task with a complex symmetrical and multimodal posterior distribution. The model is provided in Appendix F. The observation is a vector of length 10 and $\boldsymbol{\theta}$ has dimension 2. SeMPLE is run with $K = 40$, a value that is selected via BIC in 4.5 minutes. The total number of model simulations is set to 4×10^4. C2ST and runtime results are reported in Figure 3. An exemplary posterior is shown in Figure 4. SeMPLE shows the best performance according to C2ST, and its runtime is about half that of SNL, and is about 10 times more efficient than SNPE-C. Adding the BIC overhead does not change the conclusion that SeMPLE is much more efficient. The runtime comparison with SNL is not particularly relevant, since the latter does not generally manage to explore the multimodal posterior (see Figure 4 and the posteriors in Appendix F).

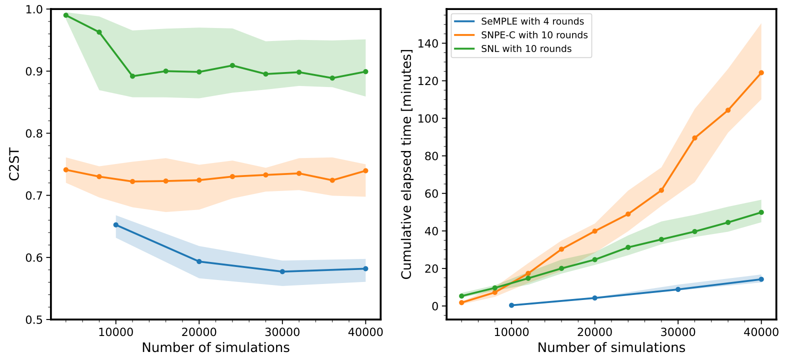


Figure 3: Hyperboloid. Median C2ST (left) and median cumulative runtime in minutes (right) for 10 runs with the same data vs the number of model simulations. Shaded bands enclose the min and max values.

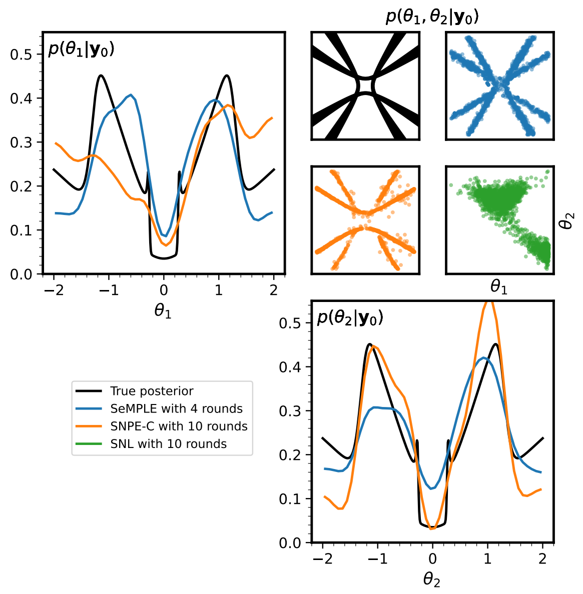


Figure 4: Hyperboloid. An example of marginal posteriors and samples from the last round of each algorithm. The SNL marginal posteriors are not reported as their inference is completely off.

5.3 Lotka-Volterra

The Lotka-Volterra predator prey model is a classical example of a population dynamics model. We consider the stochastic Markov jump process version of the model, which is defined in Appendix I. This model, although being relatively simple and characterized by a set of three interactions on the two species, each with a corresponding rate θ_j $(j = 1, 2, 3)$, encompasses many of the difficulties associated with larger, more complex systems. The observations consider the addition of non-negligible measurement error to the simulated dynamics, and the standard deviation σ of this noise term is inferred as an additional parameter, resulting in a total of four model parameters to infer $(\theta_1, \theta_2, \theta_3, \sigma)$. For this model, we are not primarily focusing on the performance comparison to other neural-based methods, as this version of the model, as a Markov jump process, is not included in `SBIBM`. Moreover, running times for a single model call are random even for repeated calls using the same parameter value. We use the Gillespie algorithm, as explained in Appendix I, and the number of “reactions” at any given $\boldsymbol{\theta}$ is stochastic. Due to the reasons above, the primary focus is to show that high quality inference can be obtained with SeMPLE using a comparably small computational effort, and compare our results against two versions of sequential Monte Carlo ABC (SMC-ABC). A “reference posterior” is obtained with an efficient version of SMC-ABC called `blockedopt` in Picchini & Tamborrino

(2024), which is run for very many model simulations, around 3.3 million, see Appendix I for details. We also compare it with a more standard off-the-shelf SMC-ABC named `standard`. SeMPLE inference is obtained with only 30,000 model simulations. In Figure 5 we also include the corresponding inference obtained with around 30,000 model simulations for both `blockedopt` and `standard` SMC-ABC. With only 30,000 model simulations, SeMPLE manages to produce inference for all parameters that is similar to the inference returned after 3.3 million model simulations from `blockedopt` SMC-ABC, a 100-fold computational saving against the most efficient version of SMC-ABC. We observe that, unlike SMC-ABC, SeMPLE produces accurate posterior inference for all model parameters, including the noise parameter σ, but with a much smaller number of model simulations compared to SMC-ABC. The latter fact, namely the SMC-ABC inefficiency in terms of model simulations, was not unexpected and known from Greenberg et al. (2019); Lueckmann et al. (2021). However it is reassuring to notice the excellent inference quality brought by SeMPLE in this challenging case study.

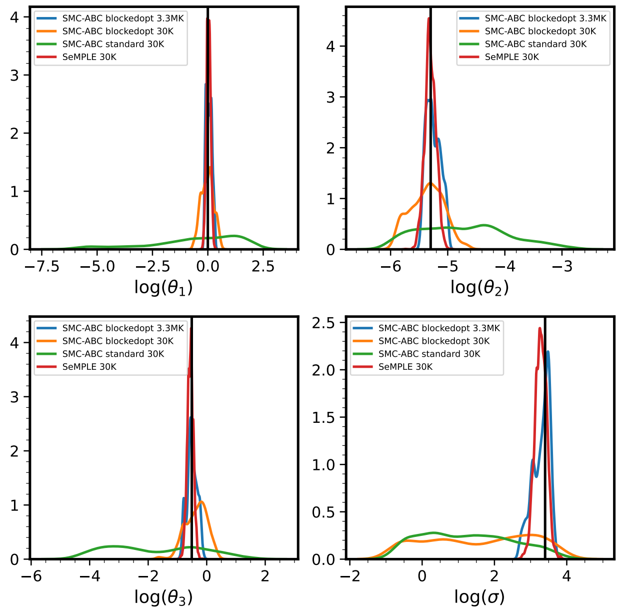


Figure 5: Lotka-Volterra. Marginal posteriors from SMC-ABC and SeMPLE. SeMPLE is run for a total of 30,000 model simulations. We report posterior samples from `blockedopt` SMC-ABC after a total of 32,865 and 3,337,640 model simulations. We also report the posterior samples from `standard` SMC-ABC after a total of 30,436 model simulations. True parameter values are indicated with black vertical lines.

5.4 Biological model of the translation kinetics after mRNA transfection

We consider a two-dimensional stochastic differential equation (SDE) model of the translation kinetics after mRNA transfection, as studied in Pieschner et al. (2022). Only one of the two coordinates of the process is observed (with measurement noise). Details about the model are in Appendix K. Pieschner et al. (2022) perform several experiments, where data are either exact observations of the system or are perturbed with measurement error, and inferences for both an ODE and an SDE model are compared. However, here we only study the SDE model with data perturbed with measurement error. The parameter to infer is (we consider parameters on log-scale, same as for their priors) $\theta = (\log \delta, \log \gamma, \log k, \log m_0, \log \text{scale}, \log t_0, \log \text{offset}, \log \sigma)$. We ran inference for five datasets, each consisting of a time-series of length 60, each generated with different parameter values. Details about data-generating parameters, and other settings, are in Appendix K. For one of the five datasets, in Figure 6 we show the marginal posteriors from the third round of SeMPLE, in comparison to the parameter priors, while in Appendix K we also show comparisons with SNL and SNPE-C. For all datasets, SeMPLE accurately infers $(\delta, \gamma, t_0, \text{offset}, \sigma)$, and we confirm the findings in Pieschner et al. (2022) (which they obtain using exact Bayesian inference via the Hamiltonian Monte Carlo method implemented in `Stan`), namely that k, m_0 and scale suffer from identifiability issues (see the supplementary section A.4.1 in Pieschner et al., 2022 where their parameter θ_2 is our k). In Appendix K we show that only SeMPLE can identify t_0 accurately, while SNL and SNPE-C provide a biased estimation of t_0. Moreover, SNPE-C is often unable to infer the measurement error variability σ, while this is not a problem for SeMPLE.

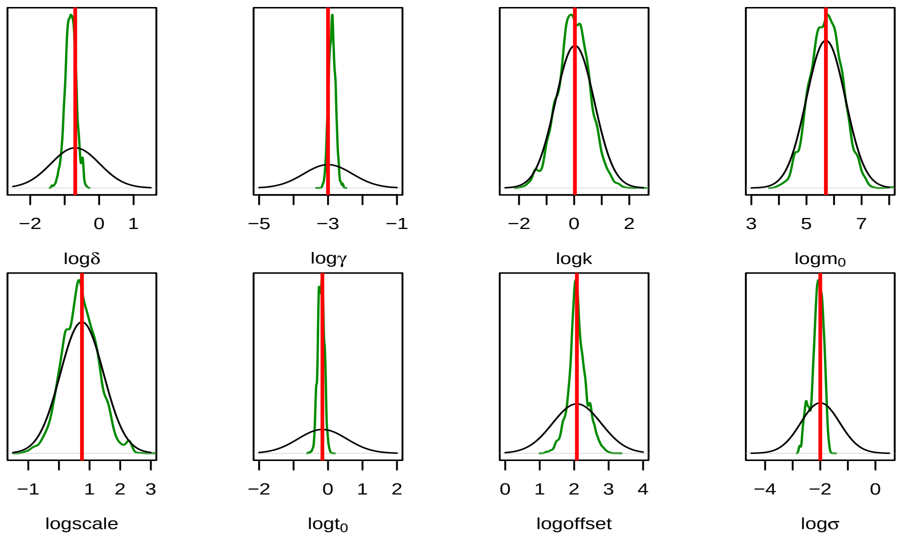


Figure 6: Translation kinetics model: in green are the marginal posteriors from SeMPLE (at round $r = 3$), priors (solid black lines) and ground truth parameters (vertical red lines).

5.5 Resource requirement metrics

The metrics reported in Table 1 correspond to four models and have been measured in kiloJoules (kJ) using the `PyJoules`[5] module found in the `PowerAPI` package (Bourdon et al., 2013) in Python, which makes use of the *Running Average Power Limit* (RAPL) technology available on Intel® CPUs. It is important to note that these energy measurements are based on hardware readings, and not on estimations depending on eg running time. The experiments were run on a machine with a CPU Intel® Core$^{\mathrm{TM}}$ i7-4790 CPU @ 3.60GHz and 16GB of DRAM. Memory cost and peak memory usage results have been measured by the `tracemalloc` Python library. These results highlight the advantage brought by SeMPLE, which can be run on a minimal configuration while maintaining good performance. For instance, in the Two Moons case, the memory requirement (Peak Memory Usage) for SeMPLE is 13 times lower than SNPE-C and 27 times lower than SNL. The gain is even larger for the other models in Table 1. Similarly, for the Ornstein-Uhlenbeck model the energy consumption (CPU+DRAM) is more than 10 times smaller with SeMPLE.

5.6 Results summary

In Table 2 we give a summary of the main results from all experiments where an objective comparison with NN-based methods was possible, including those in the Appendices. Instead, for the Lotka-Volterra model we did not run NN-based methods and we reported comparisons with SMC-ABC both in section 5.3 and in Appendix I. Generally, SeMPLE returns accurate inference, either by producing the smallest metric value (the smaller the better), as for Two Moons and Hyperboloid, or being second best after SNL, as for Bernoulli GLM and Ornstein-Uhlenbeck, also considering the larger variation of the SNPE-C results. However, even when SNL produces a smaller Wasserstein distance, it has to be balanced with respect to the running time. For Bernoulli GLM, SNL is between 18 and 75 times slower than SeMPLE, depending on the number of rounds, as SNL has to perform an expensive neural network training at each round. For the same reason, when looking at Bernoulli GLM, SNPE-C is between 8 and 26 times slower than SeMPLE,

[5]`https://pyjoules.readthedocs.io/en/latest/`

Table 2: Median metric results, across 10 independent runs (brackets include minimum and maximum values), and median runtime for posterior inference. Best (lowest) values are in bold. Notice, “Metric (final)” reports the metric at the final inference round, instead “#Model sims” gives the total number of model simulations across all rounds for a single run, and similarly for “Runtime”. SNPE-C and SNL used the default `SBIBM` setup except when the number of algorithm rounds is reduced from 10. The C2ST metric is reported for multimodal posterior distributions and the Wasserstein distance for unimodal posterior distributions according to the discussion in Section 5.6.

Method	Metric (final)	#Model sims	Runtime (minutes)	#Rounds
	Two moons (multimodal)			
SNL	C2ST=0.62 [0.56, 0.92]	10K	19 [17, 31]	4
SNL	C2ST=0.60 [0.56, 0.68]	10K	22 [21, 23]	10
SNPE-C	C2ST=0.59 [0.54, 0.66]	10K	**8.1** [7.1, 12]	4
SNPE-C	C2ST=0.57 [0.52, 0.60]	10K	15 [13, 17]	10
SeMPLE	C2ST=**0.54** [0.50, 0.58]	10K	8.3 [7.9, 8.8]	4
	Hyperboloid (multimodal)			
SNL	C2ST=0.90 [0.84, 1.0]	40K	35 [30, 39]	4
SNL	C2ST=0.90 [0.86, 0.95]	40K	50 [45, 57]	10
SNPE-C	C2ST=0.76 [0.74, 0.81]	40K	100 [76, 150]	4
SNPE-C	C2ST=0.74 [0.70, 0.75]	40K	124 [110, 151]	10
SeMPLE	C2ST=**0.58** [0.56, 0.60]	40K	**14** [13, 17]	4
	Bernoulli GLM (unimodal)			
SNL	Wasserstein=**0.58** [0.44, 0.71]	10K	22 [22, 24]	2
SNL	Wasserstein=0.69 [0.46, 0.89]	10K	91 [91, 92]	10
SNPE-C	Wasserstein=0.94 [0.65, 1.6]	10K	9.5 [4.9, 15]	2
SNPE-C	Wasserstein=0.71 [0.49, 1.78]	10K	31 [25, 40]	10
SeMPLE	Wasserstein=0.77 [0.43, 1.0]	10K	**1.2** [0.98, 1.2]	2
	Ornstein-Uhlenbeck (unimodal)			
SNL	Wasserstein=**0.067** [0.057, 0.23]	40K	56 [54, 60]	4
SNL	Wasserstein=0.12 [0.079, 0.20]	40K	62 [58, 71]	10
SNPE-C	Wasserstein=0.76 [0.27, 1.0]	40K	100 [83, 150]	4
SNPE-C	Wasserstein=0.34 [0.19, 0.48]	40K	130 [100, 150]	10
SeMPLE	Wasserstein=0.15 [0.11, 0.28]	40K	**5.9** [3.7, 14]	4

again depending on how many times it needs to be retrained. When considering the Ornstein-Uhlenbeck model, the Wasserstein distances have a good degree of overlap between SNL and SeMPLE, but SNL is between 9.5 and 10.5 times slower than SeMPLE, while SNPE-C is between 17 and 22 times slower than SeMPLE. In Table 2 we highlight whether an experiment has a unimodal or a multimodal target, as this determines the type of metric we report. Namely we use the C2ST metric for multimodal targets and the Wasserstein distance for unimodal targets. The reasons for this are now discussed, however notice that in the Appendices we report results for both metrics in each experiment. In our experiments, we found the C2ST metric (advocated in Lueckmann et al., 2021) useful to evaluate the inference quality with multimodal targets, when draws from a reference posterior are available. However, we also found that at times C2ST produced contradicting results with unimodal targets. An example is the Bernoulli GLM model considered in Appendix G. There, it is obvious that the marginal posteriors returned by SeMPLE are accurate and that inference improved between the first two rounds, however C2ST showed that inference was apparently deteriorating for SeMPLE from round $r = 1$ to $r = 2$, which was not the case in reality. We also computed Wasserstein distances, which could confirm that the inference quality via SeMPLE did improve from round 1 to round 2. However, for multimodal targets, it is true that Wasserstein distances may not be a suitable tool, see *e.g.* Balaji et al. (2019). We therefore warn the reader from blindly relying on a single performance metric. It is also important to examine marginal and joint posterior densities. For example, in Bischoff et al. (2024) it is mentioned that interpreting results for C2ST requires knowledge of the classifier used and its appropriateness for the data at hand, and that its effectiveness is critically dependent on the selection and training of a suitable classifier.

6 Conclusion and perspectives

We have addressed the challenging question of Bayesian inference in the absence of explicit likelihoods. We have investigated mixture models through the GLLiM method, and shown that their expressivity could account for multivariate distributions, while their much simpler structure made them more amenable to efficient learning than NN solutions. The resulting new inference strategy, named SeMPLE, yielded inference that was on par with inference from the state-of-art neural-network (NN)-based methods that we examined (SNL and SNPE-C), while being more energy and memory efficient. For the considered examples, and the considered simulation setup, SeMPLE produced inference of superior or similar quality than NN-based methods, and when it did occasionally under-perform, it was by a negligible amount that can largely be compensated by the much smaller runtime. In fact, in all tested cases, SeMPLE displayed much lower running times and memory requirements. The energy gains appear to increase with the complexity of the model simulator, and we expect to obtain even larger gains with real-life simulators. For example, for the Ornstein-Uhlenbeck model, SeMPLE achieved a 17-fold runtime speedup compared to SNPE-C. Although very preliminary, we hope with this attempt to meet the constraints of "Green AI" (Schwartz et al., 2020), and open the way to the development of more methods that can achieve accurate posterior inference, balancing the environmental impact of growing energy costs with the obtained results quality.

Of course, SNL and SNPE-C could certainly be improved for each specific case study, but we did not examine or fine-tune SNL and SNPE-C on the many other possible configurations, as this can be influenced by many factors. While examining several case studies implemented in `SBIBM`, we restricted to simulation settings provided in the `SBIBM` package, including reusing the same NN architectures and training setups, see Appendix A for details.

Some open questions, that can be addressed in future research, follow. SeMPLE can benefit from the summarization of data when the dimension of $\boldsymbol{y}$ is large, just like other SBI methods. In addition to reducing the dimension, using summary statistics (as we did with the Lotka-Volterra model) can also improve the GLLiM performance. Indeed the performance of SeMPLE depends on the availability of good first approximations of the posterior and likelihood, which themselves depend on the ability of GLLiM to find probabilistic mappings between parameters and observations, in a regression-like manner using a Gaussian mixture structure. The construction of informative summary statistics is an important, but independent, research question that affects not only standard ABC methods, but also SBI methods based of neural networks, *e.g.* Chen et al. (2021); Radev et al. (2023b). Finally, following the discussion in Section 5.6, we wish to emphasize that measuring the quality of posterior inference in SBI is certainly an important question deserving further research, and that multiple metrics should be considered when evaluating experiments. For a recent study on metrics to evaluate the quality of SBI, see Bischoff et al. (2024) .

Acknowledgements

We thank Emeline Perthame (Institut Pasteur) for assistance with the xLLiM package. HH and UP acknowledge funding from the Swedish National Research Council (Vetenskapsrådet 2019-03924). UP acknowledges funding from the Chalmers AI Research Centre.

References

Christophe Andrieu and Gareth O Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. The Annals of Statistics, 2009.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. Journal of the Royal Statistical Society Series B: Statistical Methodology, 72(3):269–342, 2010.

Yogesh Balaji, Rama Chellappa, and Soheil Feizi. Normalized Wasserstein distance for mixture distributions with applications in adversarial learning and domain adaptation. arXiv preprint arXiv:1902.00415, 2019.

Maria Bånkestad, Jens Sjölund, Jalil Taghia, and Thomas Schön. The elliptical processes: a family of fat-tailed stochastic processes. arXiv preprint arXiv:2003.07201, 2020.

Mark A. Beaumont, Jean-Marie Cornuet, Jean-Michel Marin, and Christian P. Robert. Adaptive approximate Bayesian computation. Biometrika, 96(4):983–990, 10 2009.

Sebastian Bischoff, Alana Darcher, Michael Deistler, Richard Gao, Franziska Gerken, Manuel Gloeckler, Lisa Haxel, Jaivardhan Kapoor, Janne K Lappalainen, and Jakob H Macke. A practical guide to statistical distances for evaluating generative models in science. arXiv preprint arXiv:2403.12636, 2024.

Christopher M. Bishop. Mixture density networks. Technical report, Aston University, 1994.

Aurélien Bourdon, Adel Noureddine, Romain Rouvoy, and Lionel Seinturier. PowerAPI: A software library to monitor the energy consumed at the process-level. ERCIM News, 2013.

Yanzhi Chen and Michael U Gutmann. Adaptive gaussian copula ABC. In The 22nd International Conference on Artificial Intelligence and Statistics, pp. 1584–1592. PMLR, 2019.

Yanzhi Chen, Dinghuai Zhang, Michael U. Gutmann, Aaron Courville, and Zhanxing Zhu. Neural approximate sufficient statistics for implicit models. In International Conference on Learning Representations, 2021.

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. Proceedings of the National Academy of Sciences, 117:201912789, 05 2020.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. An adaptive sequential Monte Carlo method for approximate Bayesian computation. Statistics and Computing, 22:1009–1020, 2012.

Antoine Deleforge, Florence Forbes, and Radu Horaud. High-dimensional regression with Gaussian mixtures and partially-latent response variables. Statistics and Computing, 25(5):893–911, mar 2014.

Conor Durkan, Iain Murray, and George Papamakarios. On contrastive learning for likelihood-free inference. In International conference on machine learning, pp. 2771–2781. PMLR, 2020.

Ritabrata Dutta, Marcel Schoengens, Lorenzo Pacchiardi, Avinash Ummadisingu, Nicole Widmer, Jukka-Pekka Onnela, and Antonietta Mira. ABCpy: A high-performance computing perspective to approximate Bayesian computation. Journal of Statistical Software, 100:1–38, 2017.

Joel Dyer, Patrick W Cannon, and Sebastian M Schmon. Amortised likelihood-free inference for expensive time-series simulators with signatured ratio estimation. In International Conference on Artificial Intelligence and Statistics, pp. 11131–11144. PMLR, 2022.

Dirk Eddelbuettel, Romain Francois, JJ Allaire, Kevin Ushey, Qiang Kou, Nathan Russell, Inaki Ucar, Douglas Bates, and John Chambers. Rcpp: Seamless R and C++ Integration, 2024. URL `https://CRAN.R-project.org/package=Rcpp`. R package version 1.0.12.

Mikhail Evchenko, Joaquin Vanschoren, Holger H. Hoos, Marc Schoenauer, and Michèle Sebag. Frugal machine learning. arXiv preprint arXiv:2111.03731, 2021.

Sarah Filippi, Chris P Barnes, Julien Cornebise, and Michael PH Stumpf. On optimality of kernels for approximate Bayesian computation using sequential Monte Carlo. Statistical applications in genetics and molecular biology, 12(1):87–107, 2013.

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, et al. POT: Python optimal transport. The Journal of Machine Learning Research, 22(1):3571–3578, 2021.

Florence Forbes, Hien Duy Nguyen, TrungTin Nguyen, and Julyan Arbel. Summary Statistics and Discrepancy Measures for Approximate Bayesian Computation via Surrogate Posteriors. Statistics and Computing, 32(5), oct 2022.

Richard Gao, Michael Deistler, and Jakob H Macke. Generalized Bayesian inference for scientific simulators via amortized cost estimation. arXiv preprint arXiv:2305.15208, 2023.

Gillespie. Exact stochastic simulation of coupled chemical reactions. The Journal of Physical Chemistry 81 (25), 2340–2361, 1977.

Andrew Golightly and Darren J Wilkinson. Bayesian parameter inference for stochastic biochemical network models using particle markov chain monte carlo. Interface Focus, 1(6):807–820, 2011.

David Greenberg, Marcel Nonnenmacher, and Jakob Macke. Automatic posterior transformation for likelihood-free inference. In International Conference on Machine Learning, pp. 2404–2414. PMLR, 2019.

Heikki Haario, Eero Saksman, and Johanna Tamminen. Adaptive proposal distribution for random walk Metropolis algorithm. Computational Statistics, 14(3):375–395, 1999.

Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free MCMC with amortized approximate ratio estimators. In International conference on machine learning, pp. 4239–4248. PMLR, 2020.

Jingjing Li, David J Nott, Yanan Fan, and Scott A Sisson. Extending approximate Bayesian computation methods to high dimensions via a Gaussian copula model. Computational Statistics & Data Analysis, 106: 77–89, 2017.

Jan-Matthis Lueckmann, Pedro J Goncalves, Giacomo Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H Macke. Flexible statistical inference for mechanistic models of neural dynamics. Advances in neural information processing systems, 30, 2017.

Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking simulation-based inference. In International conference on artificial intelligence and statistics, pp. 343–351. PMLR, 2021.

Radford M Neal. Slice sampling. The Annals of Statistics, 31(3):705–767, 2003.

Hien D Nguyen, Faicel Chamroukhi, and Florence Forbes. Approximation results regarding the multiple-output Gaussian gated mixture of linear experts model. Neurocomputing, 366:208–214, 2019.

Hien Duy Nguyen, TrungTin Nguyen, Faicel Chamroukhi, and Geoffrey John McLachlan. Approximations of conditional probability density functions in Lebesgue spaces via mixture of experts models. Journal of Statistical Distributions and Applications, 8:1–15, 2021.

Huy Nguyen, TrungTin Nguyen, and Nhat Ho. Demystifying Softmax Gating Function in Gaussian Mixture of Experts. In Thirty-seventh Conference on Neural Information Processing Systems, 2023.

D. J. Nott, V. M.-H. Ong, Y. Fan, and S.A. Sisson. Handbook of approximate Bayesian computation, chapter High-dimensional ABC. Chapman and Hall/CRC, 2018.

Jamie Owen, Darren J. Wilkinson, and Colin S. Gillespie. Likelihood free inference for markov processes: a comparison. Statistical Applications in Genetics and Molecular Biology, 14(2):189–209, 2015a.

Jamie Owen, Darren J Wilkinson, and Colin S Gillespie. Scalable inference for Markov processes with intractable likelihoods. Statistics and Computing, 25:145–156, 2015b.

George Papamakarios and Iain Murray. Fast ε-free inference of simulation models with Bayesian conditional density estimation. Advances in Neural Information Processing Systems, 29, 2016.

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, volume 89, pp. 837–848. PMLR, 16–18 Apr 2019.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. The Journal of Machine Learning Research, 22(1):2617–2680, 2021.

E. Perthame, F. Forbes, and A. Deleforge. Inverse regression approach to robust nonlinear high-to-low dimensional mapping. Journal of Multivariate Analysis, 163(C):1–14, 2018.

Emeline Perthame, Florence Forbes, Antoine Deleforge, Emilie Devijver, and Melina Gallopin. xLLiM: High Dimensional Locally-Linear Mapping, 2022. R package version 2.2.1.

Henri Pesonen, Umberto Simola, Alvaro Köhn-Luque, Henri Vuollekoski, Xiaoran Lai, Arnoldo Frigessi, Samuel Kaski, David T Frazier, Worapree Maneesoonthorn, Gael M Martin, and Jukka Corander. ABC of the future. International Statistical Review, 91(2):243–268, 2023.

Umberto Picchini and Massimiliano Tamborrino. Guided sequential ABC schemes for intractable Bayesian models. 2024. Forthcoming in *Bayesian Analysis* doi:10.1214/24-BA1451. Also available as arXiv preprint, arXiv:2206.12235.

Susanne Pieschner, Jan Hasenauer, and Christiane Fuchs. Identifiability analysis for models of the translation kinetics after mRNA transfection. Journal of Mathematical Biology, 84(7):56, 2022.

Leah F Price, Christopher C Drovandi, Anthony Lee, and David J Nott. Bayesian synthetic likelihood. Journal of Computational and Graphical Statistics, 27(1):1–11, 2018.

Stefan T. Radev, Marvin Schmitt, Valentin Pratz, Umberto Picchini, Ullrich Koethe, and Paul Buerkner. JANA: Jointly amortized neural approximation of complex Bayesian models. In Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence, pp. 1695–1706. PMLR, 2023a.

Stefan T Radev, Marvin Schmitt, Lukas Schumacher, Lasse Elsemüller, Valentin Pratz, Yannik Schälte, Ullrich Köthe, and Paul-Christian Bürkner. BayesFlow: Amortized Bayesian workflows with neural networks. arXiv preprint arXiv:2306.16015, 2023b.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In International conference on machine learning, pp. 1530–1538. PMLR, 2015.

Christian P Robert and George Casella. Monte Carlo Statistical Methods. Springer, 2004.

R. Schwartz, J. Dodge, N.A. Smith, and O. Etzioni. Green AI. Commun. ACM, 63(12):54–63, 2020.

Yannik Schälte, Emmanuel Klinger, Emad Alamoudi, and Jan Hasenauer. pyABC: Efficient and robust easy-to-use approximate bayesian computation. Journal of Open Source Software, 7:4304, 2022.

Scott A Sisson, Yanan Fan, and Mark Beaumont (eds.). Handbook of approximate Bayesian computation. CRC Press, 2018.

Sam Stites, Heiko Zimmermann, Hao Wu, Eli Sennesh, and Jan-Willem van de Meent. Learning proposals for probabilistic programs with inference combinators. In Uncertainty in Artificial Intelligence, pp. 1056–1066. PMLR, 2021.

E. Strubell, A. Ganesh, and A. McCallum. Energy and Policy Considerations for Deep Learning in NLP. In 57th Meeting of Assoc. Computational Linguistics, 2019.

Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. The computational limits of deep learning. ArXiv, abs/2007.05558, 2022.

Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael PH Stumpf. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. Journal of the Royal Society Interface, 6(31):187–202, 2009.

G. E. Uhlenbeck and L. S. Ornstein. On the Theory of the Brownian Motion. Phys. Rev., 36:823–841, Sep 1930.

Tongzhou Wang, Yi Wu, Dave Moore, and Stuart J Russell. Meta-learning MCMC proposals. Advances in neural information processing systems, 31, 2018.

Darren Wilkinson. smfsb: Stochastic Modelling for Systems Biology, 2024. URL `https://CRAN.R-project.org/package=smfsb`. R package version 1.5.

Darren J. Wilkinson. Summary stats for ABC, 2013. `https://darrenjw.wordpress.com/2013/09/01/summary-stats-for-abc/`.

Samuel Wiqvist, Jes Frellsen, and Umberto Picchini. Sequential neural posterior and likelihood approximation. arXiv preprint arXiv:2102.06522, 2021.

Simon N Wood. Statistical inference for noisy nonlinear ecological dynamic systems. Nature, 466(7310): 1102–1104, 2010.

Appendix

A SBIBM

Our runs of SNL (Papamakarios et al., 2019) and SNPE-C (Greenberg et al., 2019) used the implementation given in the Python package `SBIBM` (Lueckmann et al., 2021), which we ran with default parameters. However, we produced a fork of the `SBIBM` GitHub repository which can be found at `https://github.com/henhagg/sbibm`, and conveniently modified this to output posterior samples at each round of the inference, instead of only from the last round as given in the default implementation. This made it easier to display the performance of SNL and SNPE-C at intermediate rounds. Additionally, we added time measurements to the SNL and SNPE-C algorithms in `SBIBM` for comparison of runtimes. Note that the multiple hyperboloid model and the Ornstein-Uhlenbeck process had to be implemented as new tasks in `SBIBM` to run these models with SNL and SNPE-C. The implementation of these tasks can also be found in the `SBIBM` fork mentioned above.

The default setting for SNPE-C in `SBIBM` is to utilize Neural Spline Flows (NSF) as density estimator with 50 hidden features and 10 atoms. Correspondingly, SNL by default uses Masked Autoregressive Flow (MAF) for density estimation with 50 hidden features. The MCMC step of SNL utilizes vectorized slice sampling with 100 Markov chains and a thinning interval of 10 by default.

B BIC computation for selecting K

In GLLiM, the number K of components in the Gaussian mixture model has to be specified prior to performing the EM procedure. A too large value of K may result in overfitting and unnecessary computational effort, while a too small value of K may limit the ability to represent the relationship between $\boldsymbol{\theta}$ and $\boldsymbol{y}$. The Bayesian Information Criterion (BIC), used in Deleforge et al. (2014) to guide the selection of K, is given in equation (11)

$$BIC = -2\mathcal{L}(\hat{\boldsymbol{\phi}}) + D(\tilde{\boldsymbol{\phi}}) \log N \tag{11}$$

where $\tilde{\boldsymbol{\phi}}$ is the GLLiM model parameter, $\mathcal{L}(\hat{\boldsymbol{\phi}})$ is the maximised value of the GLLiM log-likelihood function at the MLE $\hat{\boldsymbol{\phi}}$, $D(\tilde{\boldsymbol{\phi}})$ is the total number of parameters in the model and N is the number of observations in the training dataset. We wish to select a K returning a small BIC when GLLiM is fitted with K components to a training dataset $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$ obtained by sampling parameters $\boldsymbol{\theta}_n \sim p(\boldsymbol{\theta})$ from the prior and simulating the corresponding $\boldsymbol{y}_n \sim p(\boldsymbol{y}|\boldsymbol{\theta}_n)$ from the generative model. The GLLiM log-likelihood is specified as (see eq. (4) in the main body),

$$\mathcal{L}(\tilde{\boldsymbol{\phi}}) = \sum_{n=1}^{N} \log q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{y}_n, \boldsymbol{\theta}_n), \tag{12}$$

with

$$q_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{y}_n, \boldsymbol{\theta}_n) = \sum_{k=1}^{K} \mathcal{N}(\boldsymbol{y}_n; \tilde{\boldsymbol{A}}_k\boldsymbol{\theta} + \tilde{\boldsymbol{b}}_k, \tilde{\boldsymbol{\Sigma}}_k)\, \mathcal{N}(\boldsymbol{\theta}_n; \tilde{\boldsymbol{c}}_k, \tilde{\boldsymbol{\Gamma}}_k)\, \pi_k\ . \tag{13}$$

The number of parameters that GLLiM needs to estimate is

$$D(\boldsymbol{\phi}) = (K-1) + K(DL + D + L + \text{nbpar}_\Sigma + \text{nbpar}_\Gamma), \tag{14}$$

where nbpar_Σ and nbpar_Γ are the number of parameters in the covariance matrices $\tilde{\Sigma}_k$ and $\tilde{\Gamma}_k$, respectively. The covariance structure of the matrices $\tilde{\Sigma}_k$ and $\tilde{\Gamma}_k$ can be constrained to reduce the number of parameters that GLLiM needs to estimate. One possible constraint is to set the covariance matrices to be isotropic, which means that they are set to be proportional to the identity matrix, which is the default in the `xLLiM` package (Perthame et al., 2022).

The GLLiM implementation in the `xLLiM` package removes a mixture component k if the mixture probability becomes zero. Empirically it was found that when mixture probabilities becomes very small, GLLiM could occasionally crash with errors about failed internal matrix inversions. To prevent this, a threshold was set within SeMPLE to remove mixture components with probabilities $\tilde{\eta}_k(\boldsymbol{\theta})$ below this threshold, which we set to be 0.005 for the multiple hyperboloid, Ornstein-Uhlenbeck, Lotka-Volterra and the biological SDE model, zero for Two Moons (hence no component was deleted) or 0.03 for Bernoulli GLM and SLCP models. This check was repeated in every round of SeMPLE. This shows the need for setting reasonable initial number of mixture components K, as setting a too large value inflates the problem of small mixture component probabilities. However, in several cases our inference tasks worked out-of-the-box by setting the initial value to $K = 20$ or 30.

C SNPE-A

In Papamakarios & Murray (2016) a parametric approximation to the exact posterior is learned via sequential updates, and the method is denoted SNPE-A (that is, Sequential Neural Posterior Approximation A[6]). This approximation is acquired by training a mixture density network (MDN, Bishop, 1994), which is a Gaussian mixture where the means and covariances are parameterised via neural networks. However, no method of estimating the number of mixture components is provided in SNPE-A. SNPE-A 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 for parameters $\boldsymbol{\theta}$ is denoted $\tilde{p}(\boldsymbol{\theta})$ and called "proposal prior". Therefore, parameters $\boldsymbol{\theta}_n \sim \tilde{p}(\boldsymbol{\theta})$ are drawn and corresponding data $\boldsymbol{y}_n$ are simulated from the generative model $p(\boldsymbol{y} \mid \boldsymbol{\theta}_n)$, and the MDN $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y})$ is trained on the data set $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$ to obtain a posterior approximation.

Papamakarios & Murray (2016) show that using preliminary posterior fits to guide future simulations drastically reduces the number of simulations required to learn an accurate posterior approximation. 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 Proposition 1 in Papamakarios & Murray (2016), and ultimately they suggest to estimate the posterior by

$$\hat{p}(\boldsymbol{\theta} \mid \boldsymbol{y}_o) \propto \frac{p(\boldsymbol{\theta})}{\tilde{p}(\boldsymbol{\theta})} q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y}_o).$$

As mentioned in Papamakarios & Murray (2016), in Algorithm 3 as long as $q_\phi(\boldsymbol{\theta}|\boldsymbol{y})$ has only one Gaussian component ($K = 1$) then $\tilde{p}(\boldsymbol{\theta})$ remains a single Gaussian throughout. This Gaussian approximation can be used as a rough but cheap approximation to the true posterior, or it can serve as a good proposal prior in Algorithm 4 for fine-tuning a non-Gaussian multi-component posterior. When the second strategy is used, *i.e.* Algorithm 4 is employed, they reuse the single-component neural density estimator learnt in Algorithm 3 to initialize q_ϕ in Algorithm 4, and in this case the weights in the final layer of the MDN are replicated K times, with small random perturbations to break symmetry. Notice, in SNPE-A the proposal prior $\tilde{p}(\boldsymbol{\theta})$ trained in Algorithm 3 is restricted to be a Gaussian distribution, and the prior $p(\boldsymbol{\theta})$ is assumed to be a uniform or Gaussian distribution to allow analytical calculation of $\tilde{p}(\boldsymbol{\theta} \mid \boldsymbol{y} = \boldsymbol{y}_o)$.

In SeMPLE we do not have such restrictions, meaning that we can consider any prior $p(\boldsymbol{\theta})$, however this comes at the price of introducing MCMC sampling (as in SNL). However, unlike for SNPE-A, in SeMPLE we are not constrained to reuse the estimated one-component Gaussian and replicate this K times to obtain a K-components mixture, instead the GLLiM algorithm automatically determines the means and covariance matrices for each of the K components, which as such can be different between components, thus adding flexibility to the shapes of the posterior distributions that can be targeted.

[6]This is not the name given in Papamakarios & Murray (2016) but follows the taxonomy established in Durkan et al. (2020).

Algorithm 3 SNPE-A: Training of proposal prior

```
Initialize $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y})$ with one component ($K = 1$).
$\tilde{p}(\boldsymbol{\theta}) \leftarrow p(\boldsymbol{\theta})$.
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 SNPE-A: 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 $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$
$\hat{p}(\boldsymbol{\theta} \mid \boldsymbol{y} = \boldsymbol{y}_o) \leftarrow \frac{p(\boldsymbol{\theta})}{\tilde{p}(\boldsymbol{\theta})} q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{y}_o)$
```

D SNL

Sequential Neural Likelihood (SNL) (Papamakarios et al., 2019) sequentially trains a conditional neural density estimator for the likelihood function, instead of training an estimator for the posterior as in SNPE-A. This eliminates the need for corrections stemming from the proposal distribution not being the prior, but introduces an additional MCMC step (in the form of slice sampling) to sample from the posterior distribution, by employing the estimation $q_{\boldsymbol{\phi}}(\boldsymbol{y}_o \mid \boldsymbol{\theta})$ of the likelihood. SNL is detailed in Algorithm 5, which is the same as Algorithm 1 in Papamakarios et al. (2019). The main difference with SeMPLE is that, while SNL trains $q_\phi(\boldsymbol{y}|\boldsymbol{\theta})$ and then plugs the latter into MCMC for posterior sampling, SeMPLE trains $q_{\tilde{\phi}}(\boldsymbol{y}|\boldsymbol{\theta})$ and also obtain (as an *immediate* algebraic by-product of the $q_{\tilde{\phi}}(\boldsymbol{y}|\boldsymbol{\theta})$ training) a proposal distribution $q_\phi(\boldsymbol{\theta}|\boldsymbol{y}_o)$ that is used in the MCMC step.

Algorithm 5 Sequential Neural Likelihood (SNL)

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

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

E Two moons model

E.1 Two moons model definition

The Two Moons model, described in Greenberg et al. (2019), generates $\boldsymbol{y} \in \mathbb{R}^2$ for a given parameter $\boldsymbol{\theta} \in \mathbb{R}^2$ according to

$$a \sim \mathcal{U}(-\frac{\pi}{2}, \frac{\pi}{2}) \tag{15}$$

$$r \sim \mathcal{N}(0.1, 0.01^2) \tag{16}$$

$$\boldsymbol{p} = (r\cos(a) + 0.25,\ r\sin(a)) \tag{17}$$

$$\boldsymbol{y}^\top = \boldsymbol{p} + \left(-\frac{|\theta_1 + \theta_2|}{\sqrt{2}},\ \frac{-\theta_1 + \theta_2}{\sqrt{2}}\right). \tag{18}$$

Same as in Greenberg et al. (2019), we set uniform priors $\theta_1 \sim \mathcal{U}(-1, 1)$, $\theta_2 \sim \mathcal{U}(-1, 1)$. By construction, this toy example has a posterior distribution with two crescent shapes for a given observation.

E.2 Two moons: targeting the posterior directly vs targeting the likelihood

In the main paper we discussed the two possibilities of targeting the posterior $\tilde{p}_{r+1}(\boldsymbol{\theta}) \propto q_{\tilde{\boldsymbol{\phi}}_r}(\boldsymbol{y}_o \,|\, \boldsymbol{\theta})p(\boldsymbol{\theta})$ by first obtaining an estimate $q_{\tilde{\boldsymbol{\phi}}_r}(\boldsymbol{y}_o \,|\, \boldsymbol{\theta})$ of the likelihood and then sampling via MCMC from $\tilde{p}_{r+1}(\boldsymbol{\theta})$, as opposed to the option of using the GLLiM surrogate posterior $q_{\boldsymbol{\phi}_r}(\boldsymbol{\theta} \,|\, \boldsymbol{y}_o)$ to update the posterior $\tilde{p}_{r+1}(\boldsymbol{\theta}) = \frac{p(\boldsymbol{\theta})}{\tilde{p}_r(\boldsymbol{\theta})} q_{\boldsymbol{\phi}_r}(\boldsymbol{\theta} \,|\, \boldsymbol{y}_o)$, and then sampling from the latter via MCMC. We hinted at the fact that it is preferable to follow the first route, which we used to obtain all our results, and here we provide results in support of our recommendation. Figure 8 shows the C2ST metric and the Wasserstein distance as a function of the number of model simulations for different SeMPLE settings, including different choices for the covariance matrices $\tilde{\boldsymbol{\Sigma}}_k$ ("isotropic" or "full unconstrained") of the mixture models. We first considered the "isotropic" covariances case (specified in `xLLiM` via the option `cstr$Sigma="i"`), which means that all the K covariance matrices are proportional to the diagonal matrix, however these matrices are all different between the K components. With this option, we ran SeMPLE by targeting $\tilde{p}_{r+1}(\boldsymbol{\theta}) \propto q_{\tilde{\boldsymbol{\phi}}_r}(\boldsymbol{y}_o \,|\, \boldsymbol{\theta})p(\boldsymbol{\theta})$ ("likelihood; isotropic" in the legend of Figure 8) and also ran SeMPLE by targeting $\tilde{p}_{r+1}(\boldsymbol{\theta}) = \frac{p(\boldsymbol{\theta})}{\tilde{p}_r(\boldsymbol{\theta})} q_{\boldsymbol{\phi}_r}(\boldsymbol{\theta} \,|\, \boldsymbol{y}_o)$ ("posterior; isotropic" in the legend of Figure 8). Targeting $\tilde{p}_{r+1}(\boldsymbol{\theta}) \propto q_{\tilde{\boldsymbol{\phi}}_r}(\boldsymbol{y}_o \,|\, \boldsymbol{\theta})p(\boldsymbol{\theta})$ is clearly the preferred route according to C2ST. This is less clear by looking at the Wasserstein distance, however the computation of this particular metric may not be accurate with multimodal targets (while we recommend to use Wasserstein distances for unimodal targets), and we refer the reader to section 5.6 in the main paper for a discussion about metrics to evaluate posterior inference. Additionally, we explored the results of assuming full matrices $\tilde{\boldsymbol{\Sigma}}_k$ without constraints (specified in `xLLiM` via the option `cstr$Sigma=""`). This setting is run together with our (recommended) method of sampling from $\tilde{p}_{r+1}(\boldsymbol{\theta}) \propto q_{\tilde{\boldsymbol{\phi}}_r}(\boldsymbol{y}_o \,|\, \boldsymbol{\theta})p(\boldsymbol{\theta})$ ("likelihood; unconstrained" in the legend of Figure 8). This SeMPLE configuration produces the best results. This may not be surprising, as for two-moons $\boldsymbol{y}_0$ has dimension 2, therefore the covariances $\tilde{\boldsymbol{\Sigma}}_k$ have dimensions 2×2, thus not inducing a large number of parameters $\tilde{\boldsymbol{\phi}}$ to estimate via EM. Notice that the C2ST value, when using our recommended method but with isotropic covariance matrices, is still lower than C2ST as obtained with SNPE-C (see the main paper). Regarding the performance of the MCMC algorithm, Figure 7 shows that the acceptance rate is noticeably higher with the "likelihood" option rather than the "posterior" one, which strengthens our choice. While a higher acceptance rate does not denote, per-se, any special ability for an algorithm to efficiently explore the posterior, however this fact coupled with the already displayed high quality inference provided by SeMPLE for this example, means that Figure 7 further shows that we can obtain a larger number of representative posterior samples when targeting $\tilde{p}_{r+1}(\boldsymbol{\theta}) \propto q_{\tilde{\boldsymbol{\phi}}_r}(\boldsymbol{y}_o \,|\, \boldsymbol{\theta})p(\boldsymbol{\theta})$.

Based on these results, it was concluded that using the surrogate likelihood to target $q_{\tilde{\boldsymbol{\phi}}_r}(\boldsymbol{y}_o \,|\, \boldsymbol{\theta})p(\boldsymbol{\theta})$ in the MCMC step, with the surrogate posterior $q_{\phi_r}(\boldsymbol{\theta} \,|\, \boldsymbol{y}_o)$ as proposal distribution in the Metropolis-Hastings algorithm, is a superior method overall, which we used in all the other simulation studies.

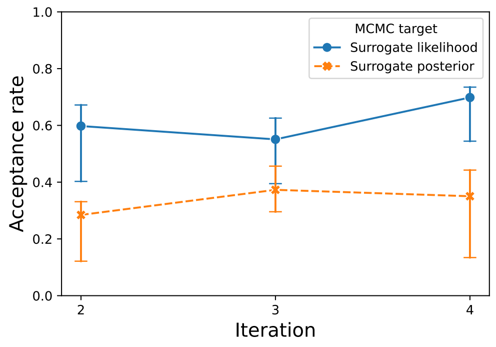
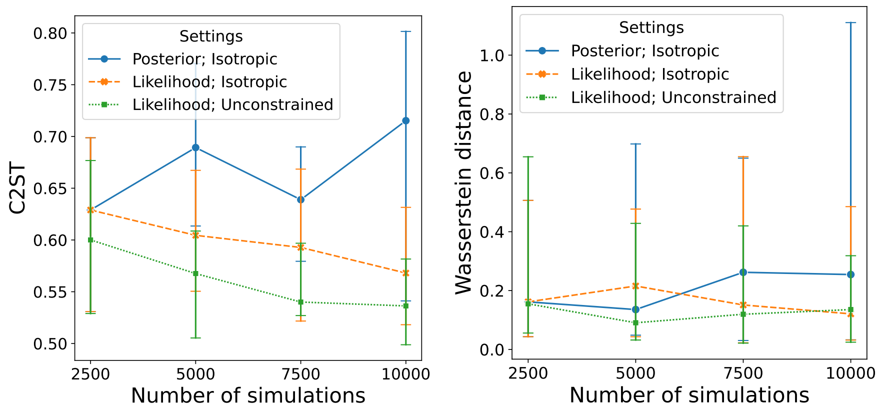


Figure 7: Two Moons: Median acceptance rate of the Metropolis-Hastings step of 10 SeMPLE runs with different data sets. Error bars show min/max values.



(a) (b)

Figure 8: Two Moons: Median C2ST and Wasserstein distance on 10 SeMPLE runs with different data sets vs number of model simulations. The three different SeMPLE settings refers to whether the surrogate likelihood (reported as "likelihood; isotropic" or "likelihood; unconstrained") or the "corrected" surrogate posterior (reported as "Posterior; isotropic") is used in the MCMC target distribution, and constraints used for the GLLiM covariance matrices ("isotropic" or "unconstrained"). Error bars show min/max values.

E.3 Two moons: selection of K via BIC

We illustrate how to select a starting value for K using BIC. However, we recall that it is not strictly necessary to choose K in a very precise way, as its value is let decrease during the SeMPLE run. All BIC values are computed on the same prior-predictive data set $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$ with $N = 2\,500$, obtained prior to running SeMPLE. Figure 9 shows BIC as a function of K. The minimum BIC of the evaluated values is obtained at $K = 50$ but it decreases sharply from $K = 10$ to $K = 30$. By the Occam's razor principle, $K = 30$ was used as input to SeMPLE. Empirical experiments showed that a larger value of K did not improve results. The runtime to compute all BIC values in Figure 9 was 93 seconds (on a desktop computer with a 6-core (12 threads) AMD Ryzen 5 2600 CPU). Note that the BIC computations are "amortised", that is are independent of the observed data set $\boldsymbol{y}_o$, and only have to be performed once before running SeMPLE, and can thus be recycled whenever a different observed data set is considered.

E.4 Two moons. Results with $R = 4$

In the main paper, we have considered the inference results based on a total of 10^4 model simulations, uniformly distributed across $R = 10$ rounds for SNPE-C and SNL (as per `SBIBM` defaults), and $R = 4$ rounds

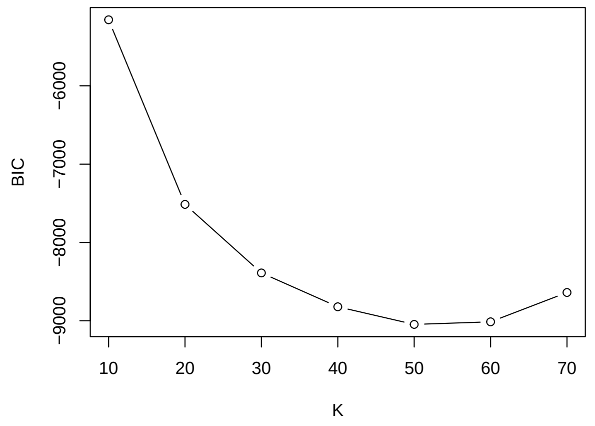


Figure 9: Two Moons. BIC with respect to the number of mixture components K.

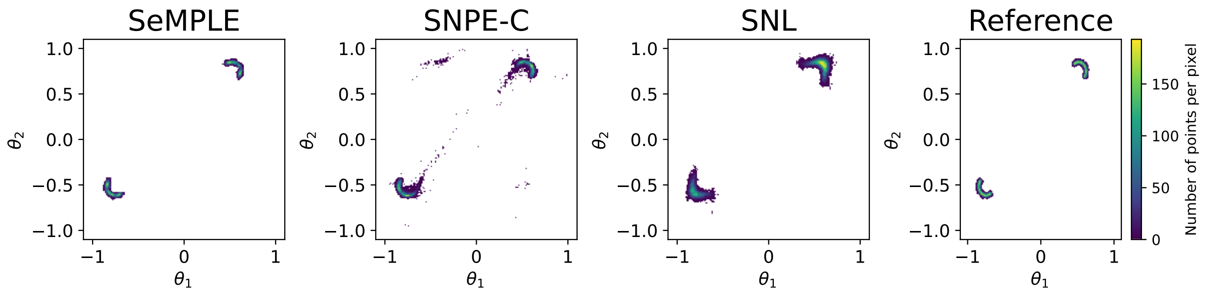


Figure 10: Two Moons using $R = 4$ rounds for every method. Density scatter plot of posterior samples from the last round (r=4) of each algorithm. Observed data set number 1 in SBIBM.

for SeMPLE. Results showed that SeMPLE performed better and in particular that SNL (which is also an MCMC-based procedure, similarly to SeMPLE) was returning the worst performance. Here we show the performance of both SNPE-C and SNL with $R = 4$, again using a total of 10^4 model simulations for each method. That is, for each method we show the inference obtained after 2500, 5000, 7500 and 10000 model simulations. Figures 10-19 shows scatter plots (colored by density value) of the posterior samples from the last round (r=4) of each algorithm and for each of the 10 observed data sets in SBIBM. SeMPLE consistently matches the reference posterior, unlike SNL and SNPE-C.

Moreover, it is even more interesting to note that SeMPLE performs already well after a single round, see Figures 20-29 showing posterior samples from every round of each algorithm and for each of the 10 observed data sets. While SNL improves across rounds, this is not the case for SNPE-C.

Figure 30 shows C2ST and runtime values in the case when $R = 4$ is used for all algorithms. Compared to the results in the main paper with $R = 10$ rounds, SNPE-C and SNL perform slightly worse with $R = 4$ rounds in terms of the C2ST metric. The SNPE-C and SNL runtimes are however reduced, with SNPE-C having essentially identical runtime as SeMPLE, although with significantly higher variance. SNL is approximately twice as slow as the other algorithms.

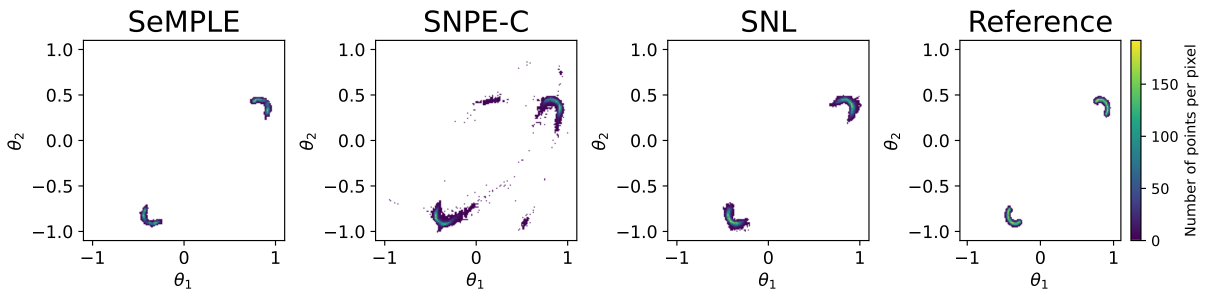


Figure 11: Two Moons using $R = 4$ rounds for every method. Density scatter plot of posterior samples from the last round (r=4) of each algorithm. Observed data set number 2 in SBIBM.

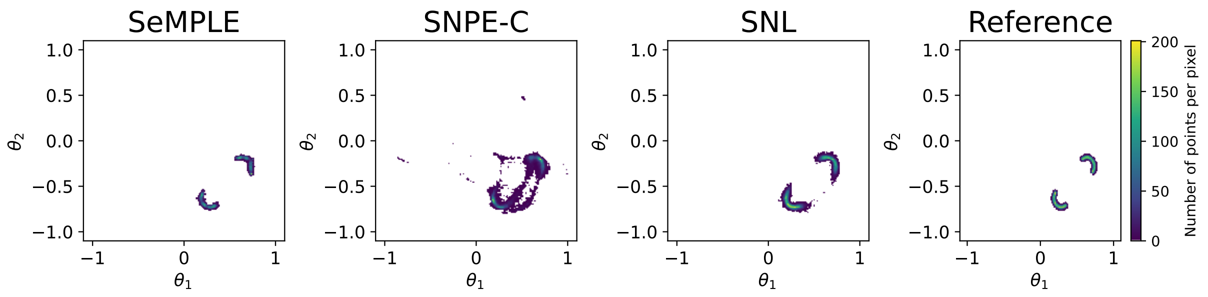


Figure 12: Two Moons using $R = 4$ rounds for every method. Density scatter plot of posterior samples from the last round (r=4) of each algorithm. Observed data set number 3 in SBIBM.

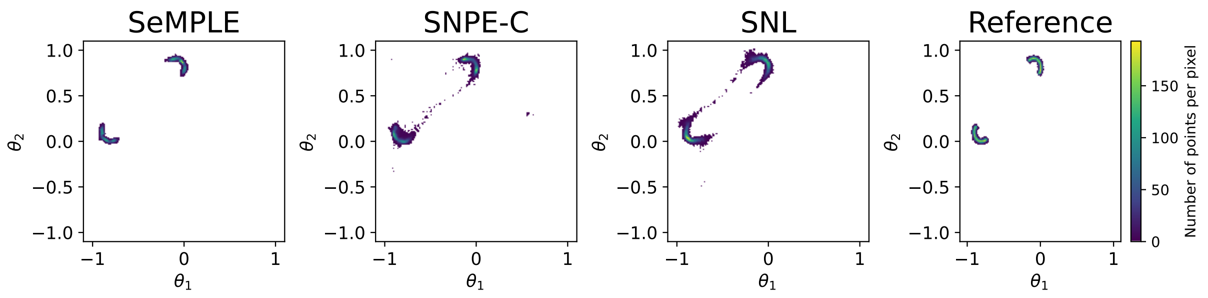


Figure 13: Two Moons using $R = 4$ rounds for every method. Density scatter plot of posterior samples from the last round (r=4) of each algorithm. Observed data set number 4 in SBIBM.

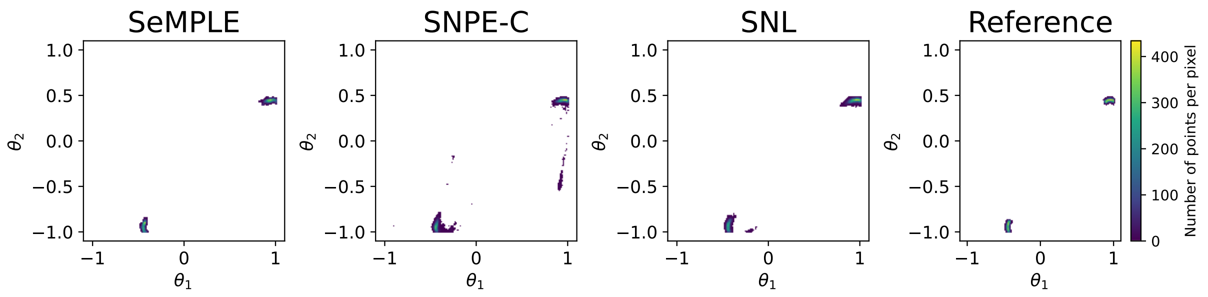


Figure 14: Two Moons using $R = 4$ rounds for every method. Density scatter plot of posterior samples from the last round (r=4) of each algorithm. Observed data set number 5 in SBIBM.

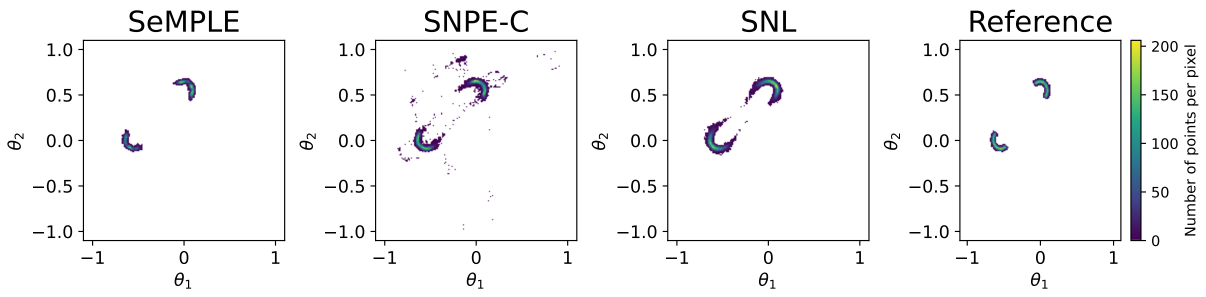


Figure 15: Two Moons using $R = 4$ rounds for every method. Density scatter plot of posterior samples from the last round (r=4) of each algorithm. Observed data set number 6 in SBIBM.

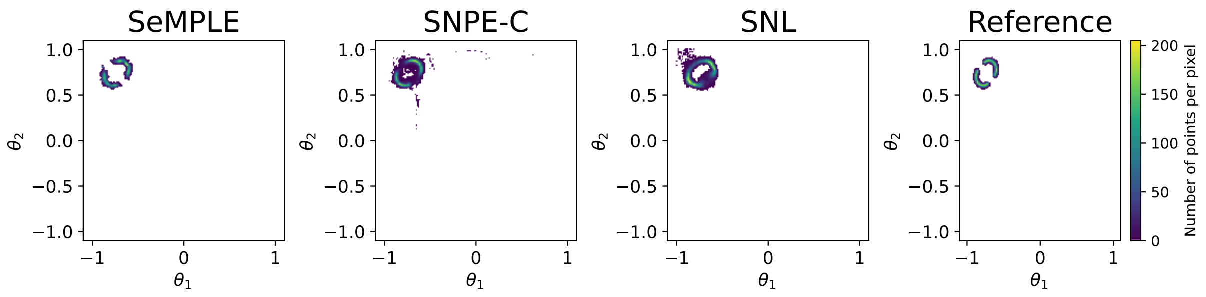


Figure 16: Two Moons using $R = 4$ rounds for every method. Density scatter plot of posterior samples from the last round (r=4) of each algorithm. Observed data set number 7 in SBIBM.

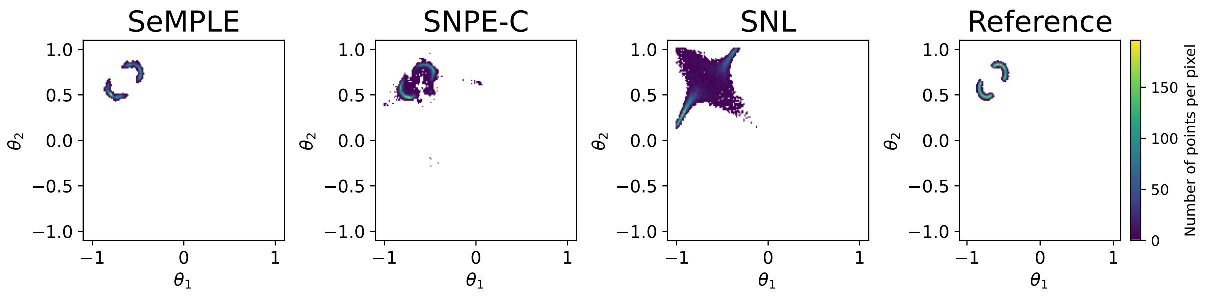


Figure 17: Two Moons using $R = 4$ rounds for every method. Density scatter plot of posterior samples from the last round (r=4) of each algorithm. Observed data set number 8 in SBIBM.

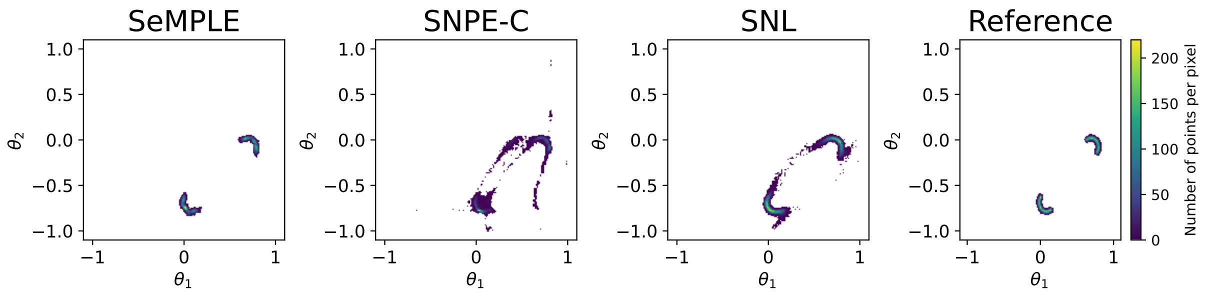


Figure 18: Two Moons using $R = 4$ rounds for every method. Density scatter plot of posterior samples from the last round (r=4) of each algorithm. Observed data set number 9 in SBIBM.

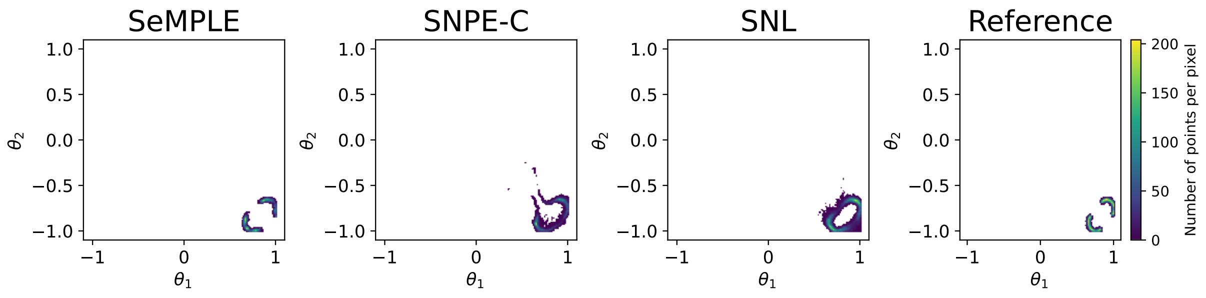


Figure 19: Two Moons using $R = 4$ rounds for every method. Density scatter plot of posterior samples from the last round (r=4) of each algorithm. Observed data set number 10 in SBIBM.

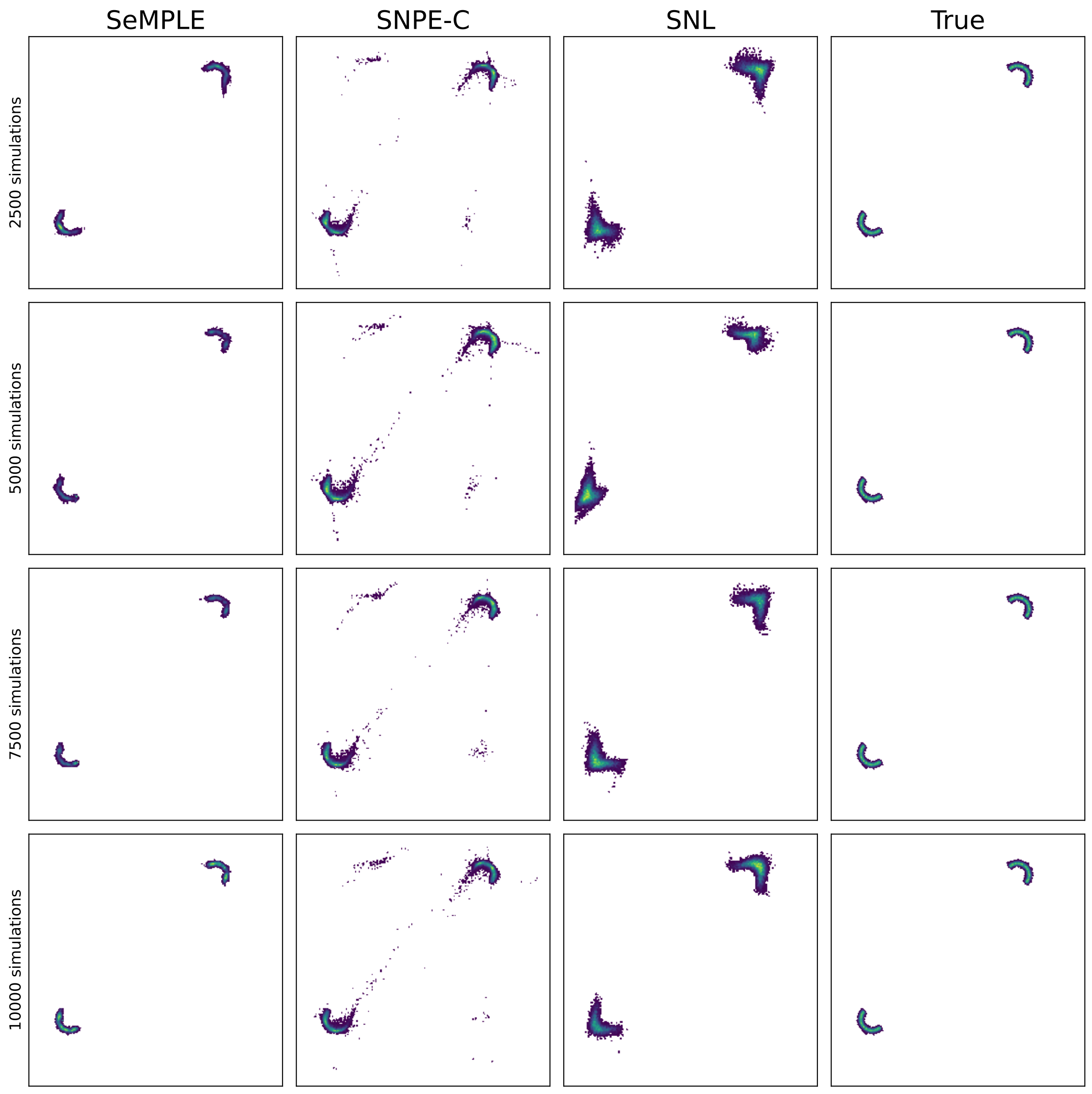


Figure 20: Two Moons using $R = 4$ rounds for every method: inference using observed dataset number 1 from SBIBM. Posterior samples from each algorithm round, starting from the first round (upper row) to the fourth (bottom row).

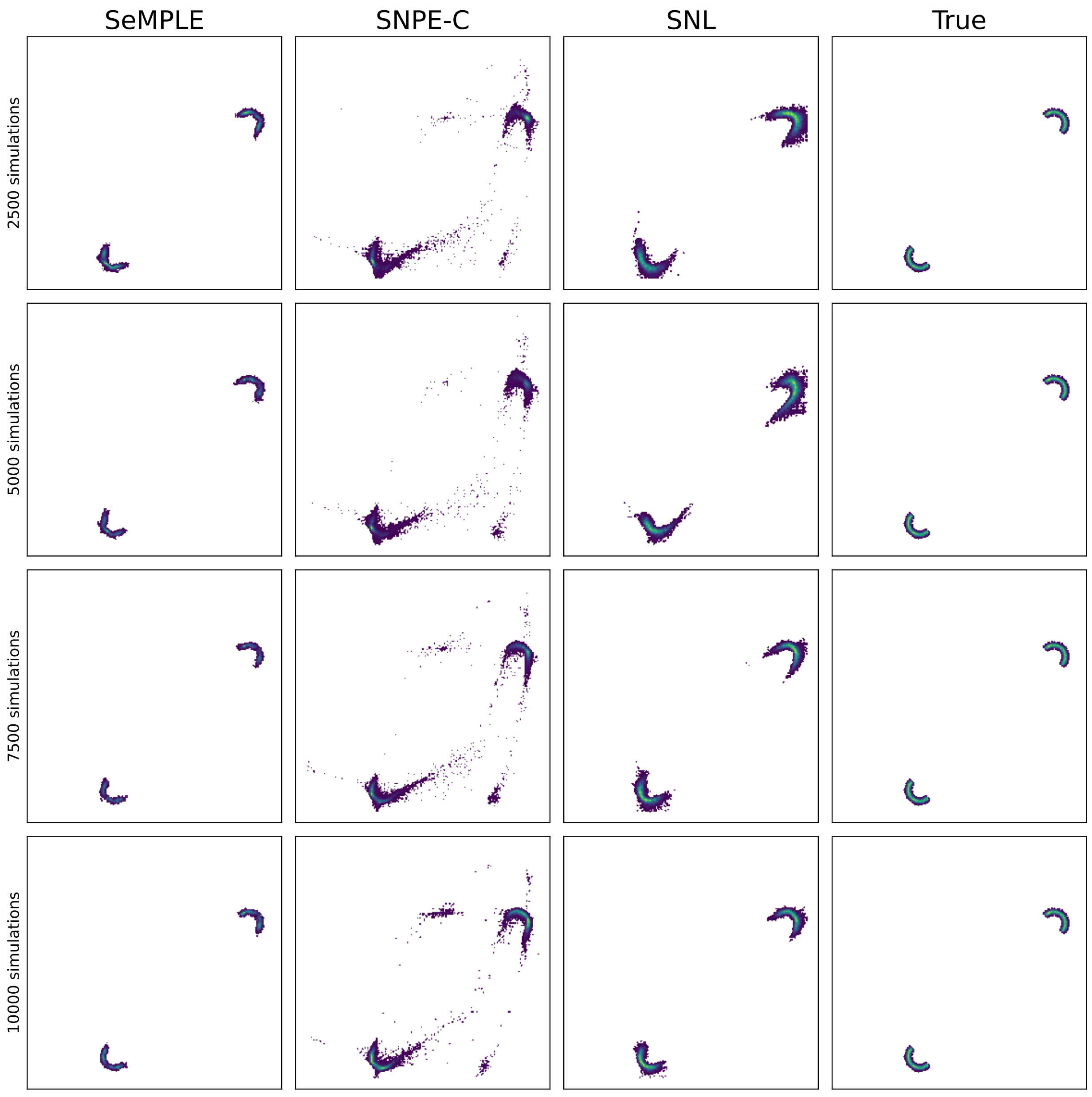


Figure 21: Two Moons using $R = 4$ rounds for every method: inference using observed dataset number 2 from SBIBM. Posterior samples from each algorithm round, starting from the first round (upper row) to the fourth (bottom row).

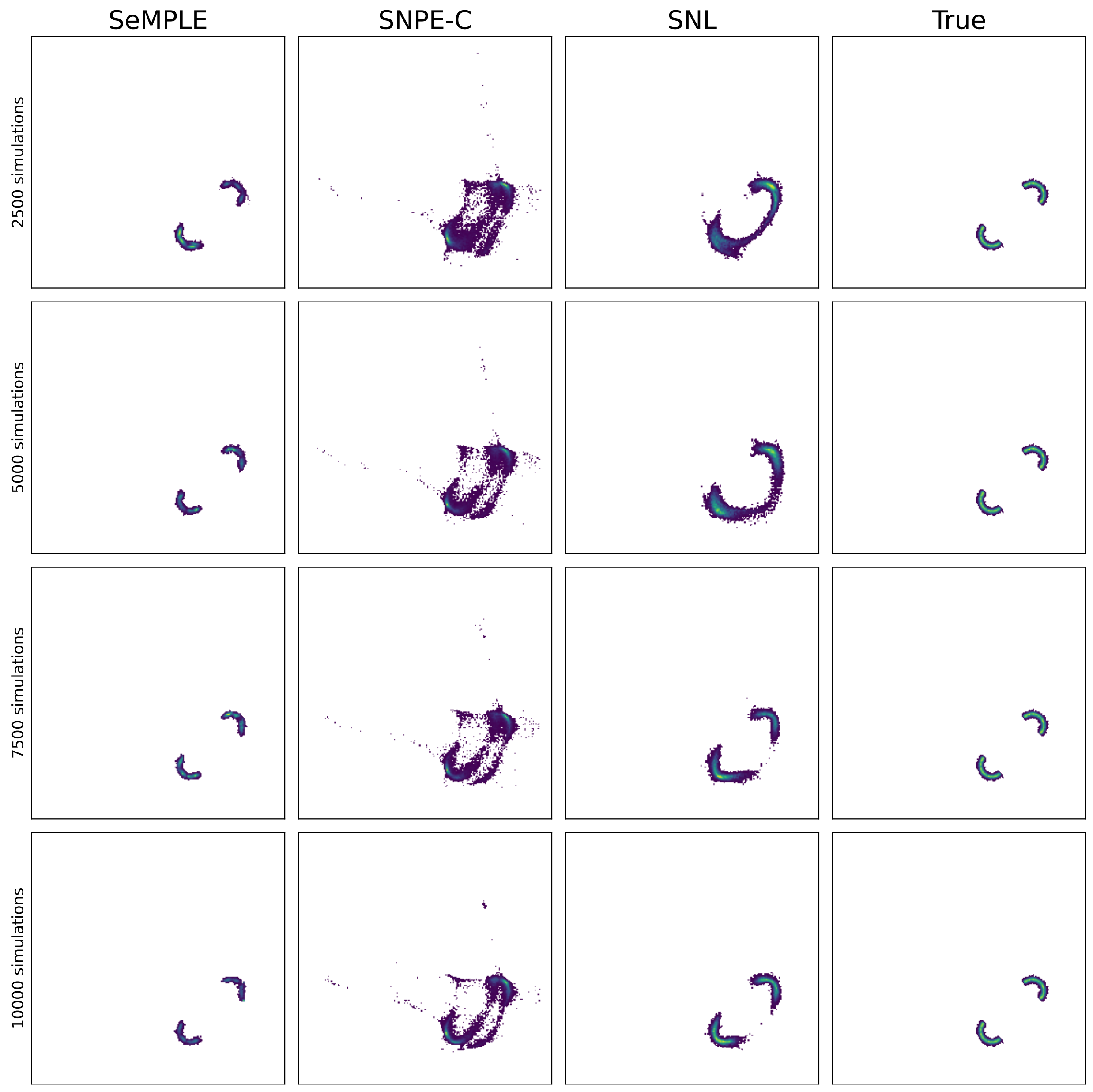


Figure 22: Two Moons using $R = 4$ rounds for every method: inference using observed dataset number 3 from SBIBM. Posterior samples from each algorithm round, starting from the first round (upper row) to the fourth (bottom row).

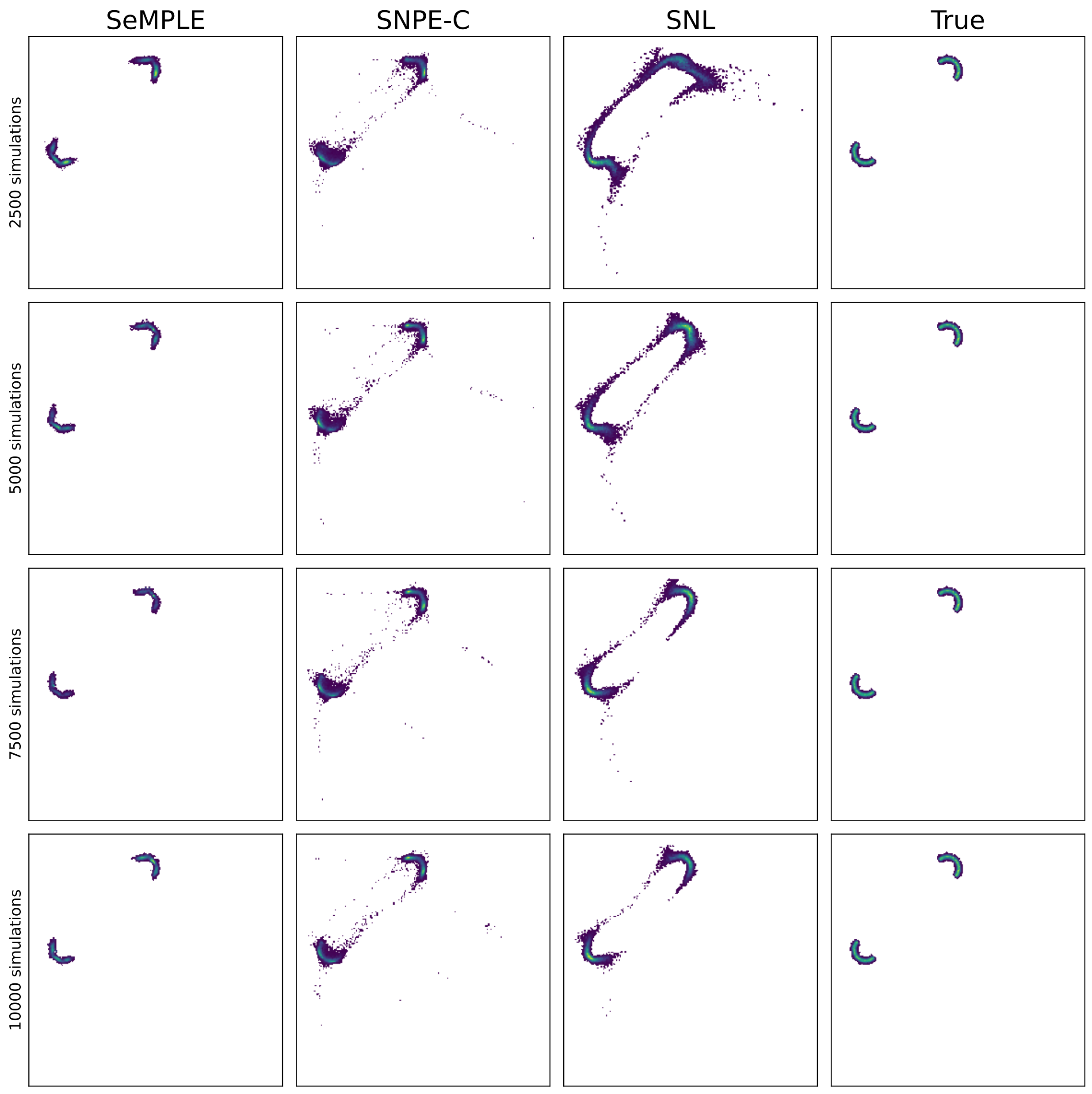


Figure 23: Two Moons using $R = 4$ rounds for every method: inference using observed dataset number 4 from SBIBM. Posterior samples from each algorithm round, starting from the first round (upper row) to the fourth (bottom row).

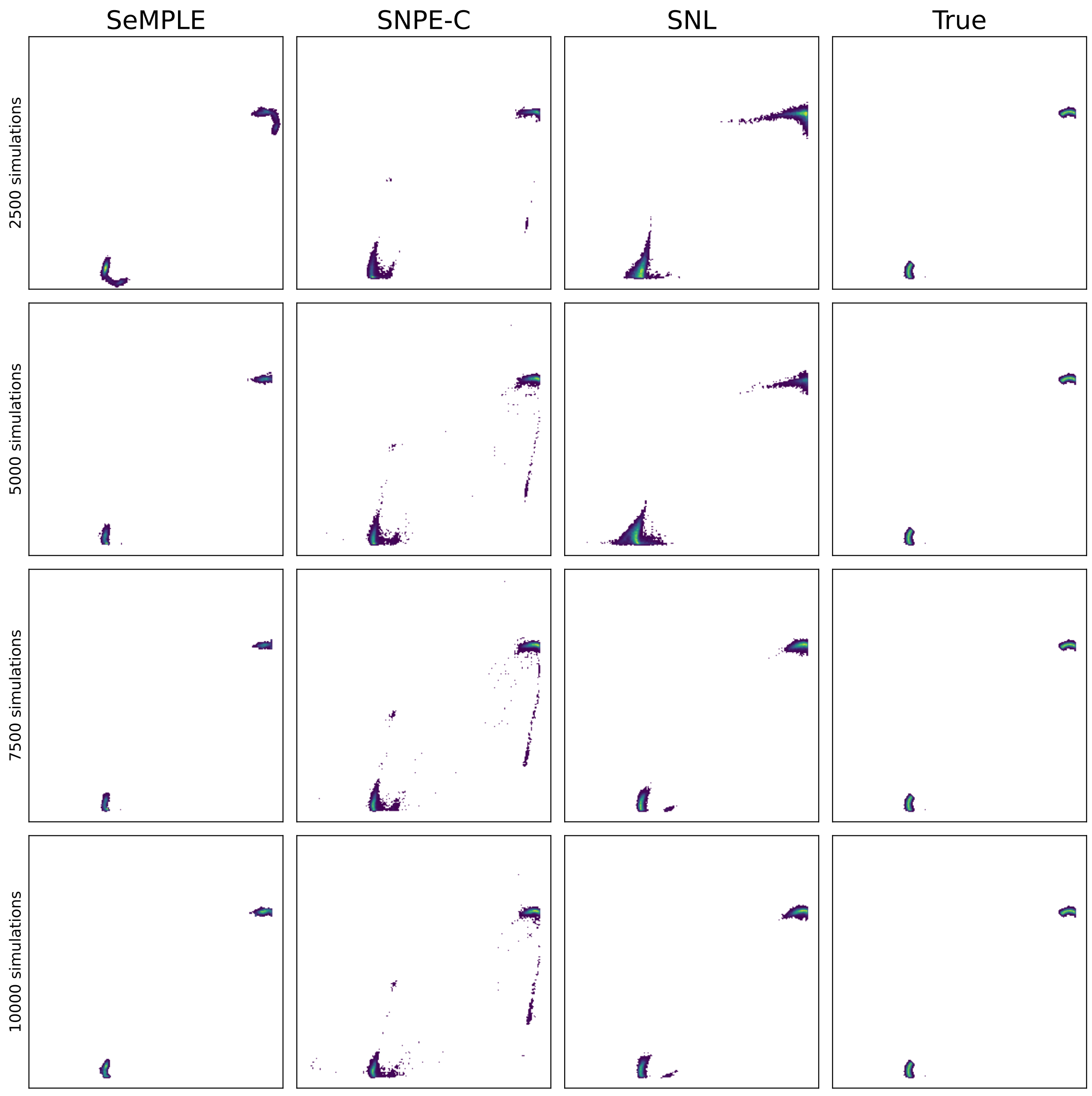


Figure 24: Two Moons using $R = 4$ rounds for every method: inference using observed dataset number 5 from SBIBM. Posterior samples from each algorithm round, starting from the first round (upper row) to the fourth (bottom row).

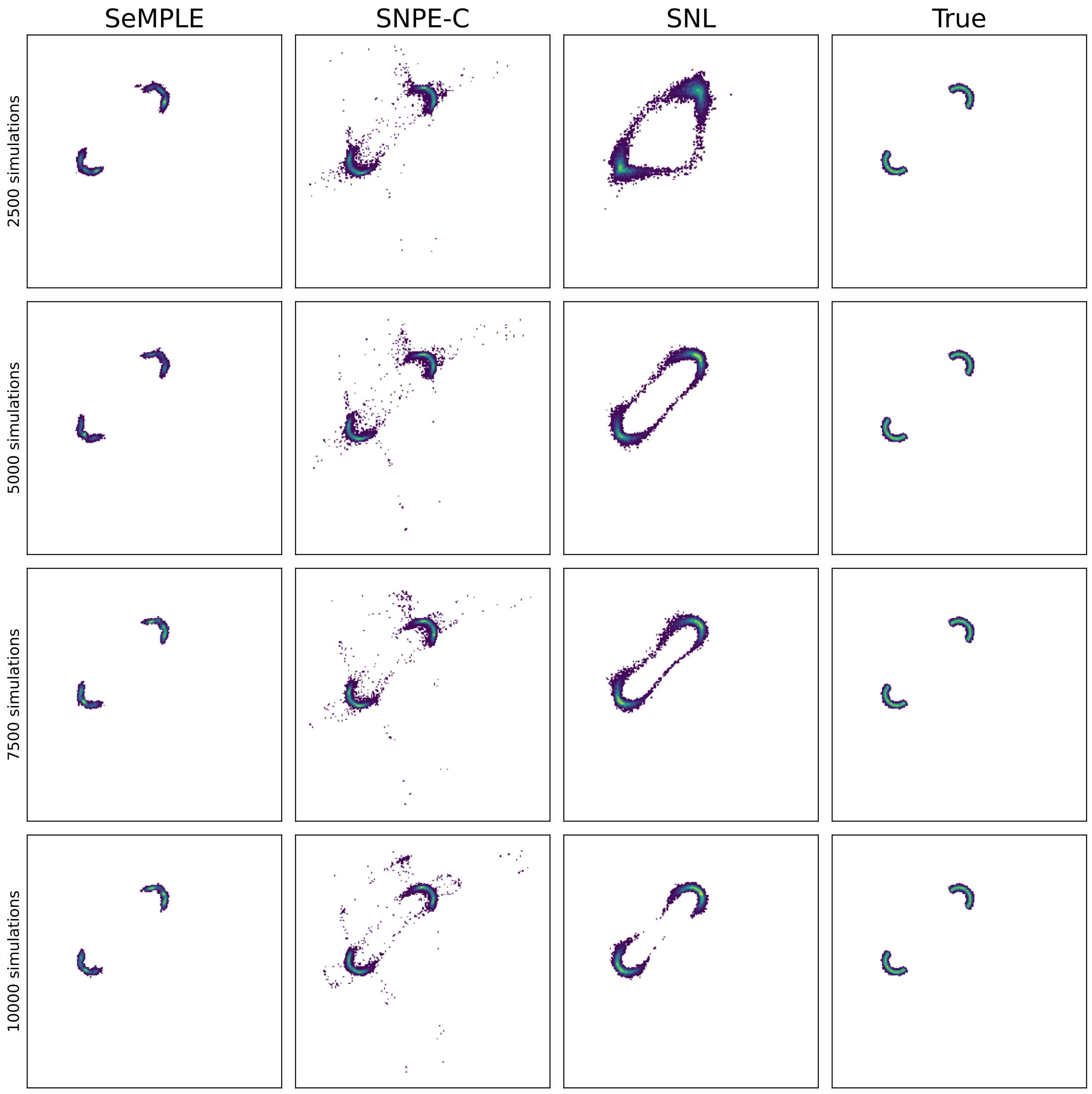


Figure 25: Two Moons using $R = 4$ rounds for every method: inference using observed dataset number 6 from SBIBM. Posterior samples from each algorithm round, starting from the first round (upper row) to the fourth (bottom row).

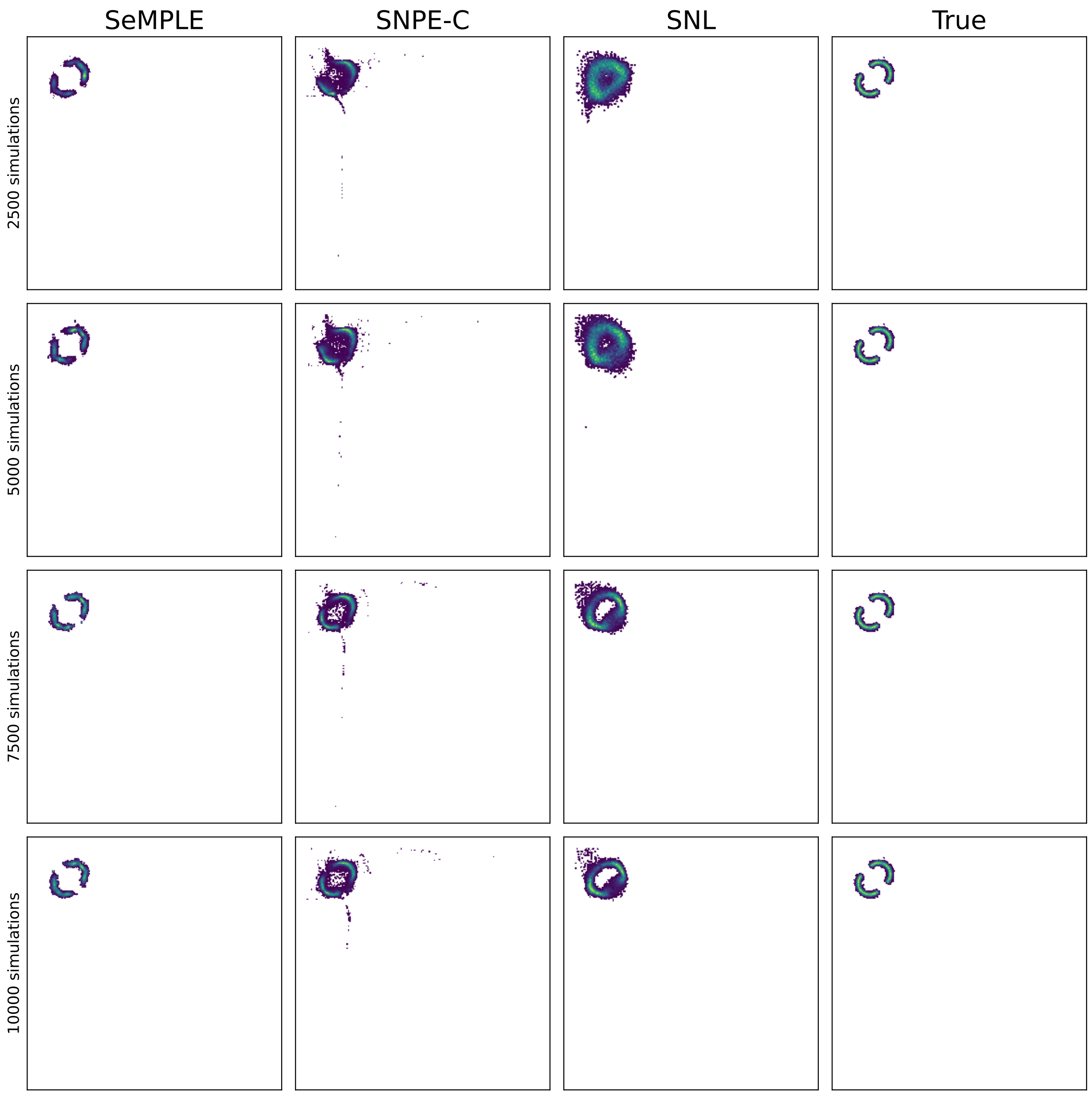


Figure 26: Two Moons using $R = 4$ rounds for every method: inference using observed dataset number 7 from SBIBM. Posterior samples from each algorithm round, starting from the first round (upper row) to the fourth (bottom row).

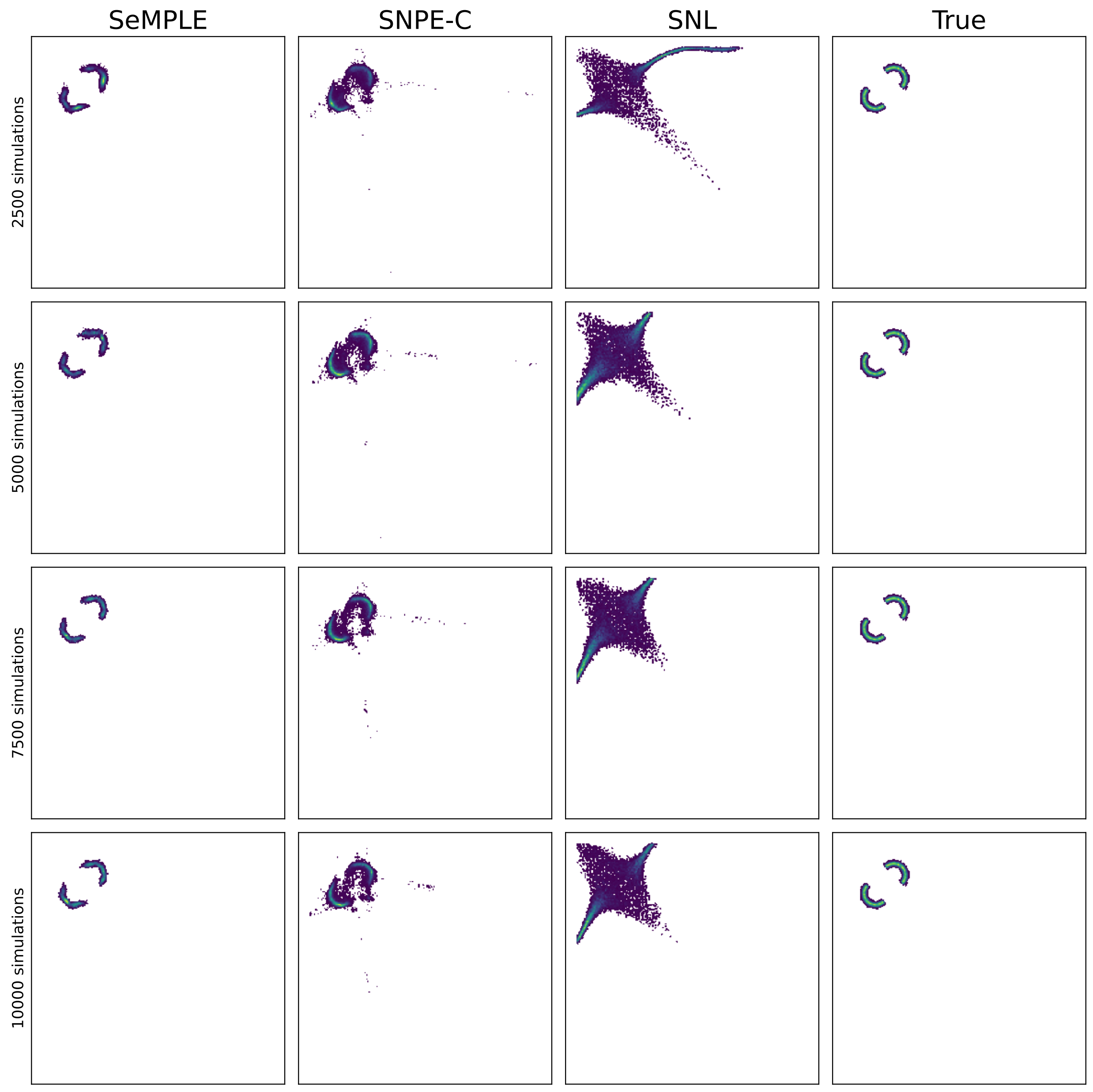


Figure 27: Two Moons using $R = 4$ rounds for every method: inference using observed dataset number 8 from SBIBM. Posterior samples from each algorithm round, starting from the first round (upper row) to the fourth (bottom row).

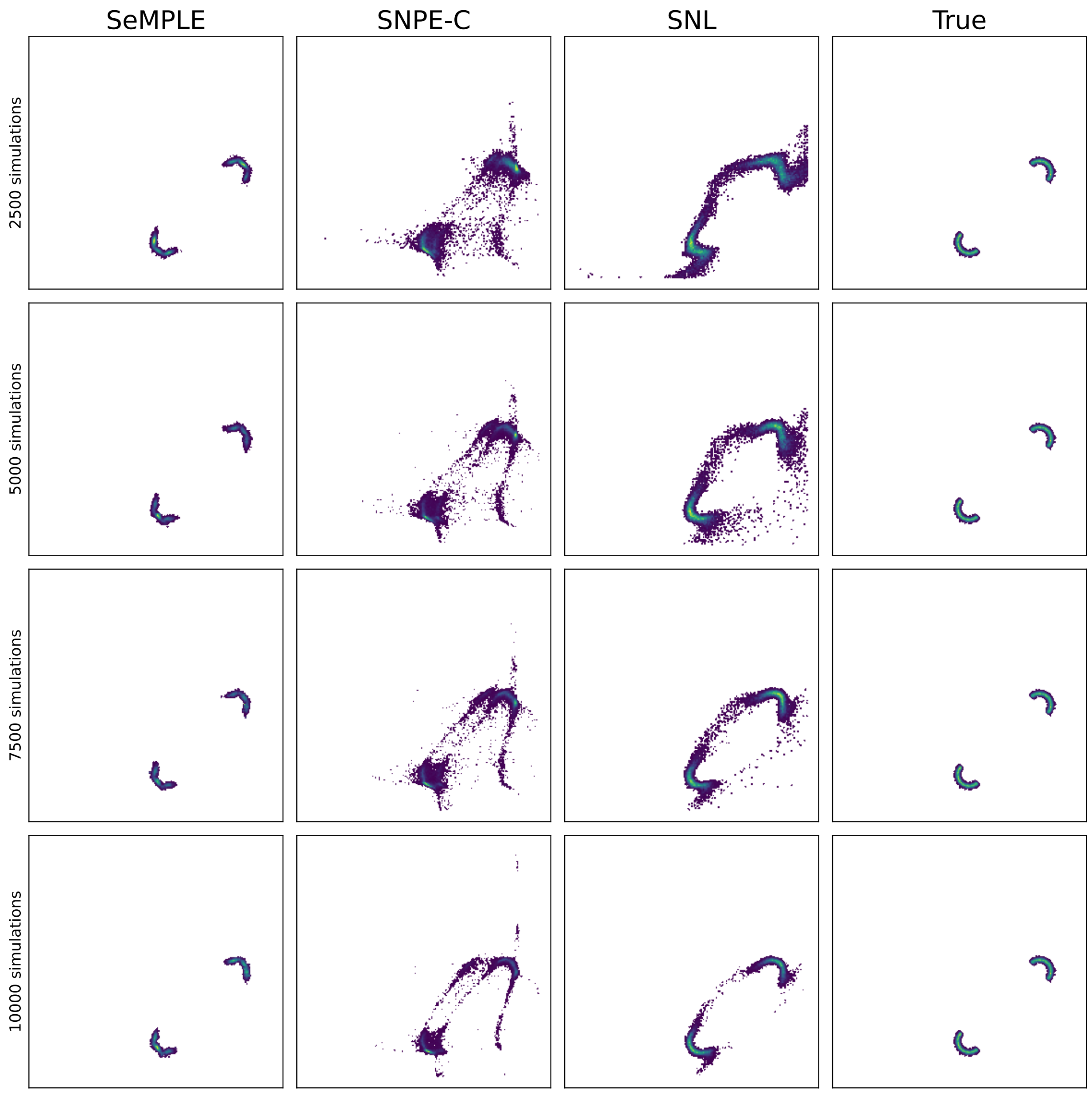


Figure 28: Two Moons using $R = 4$ rounds for every method: inference using observed dataset number 9 from SBIBM. Posterior samples from each algorithm round, starting from the first round (upper row) to the fourth (bottom row).

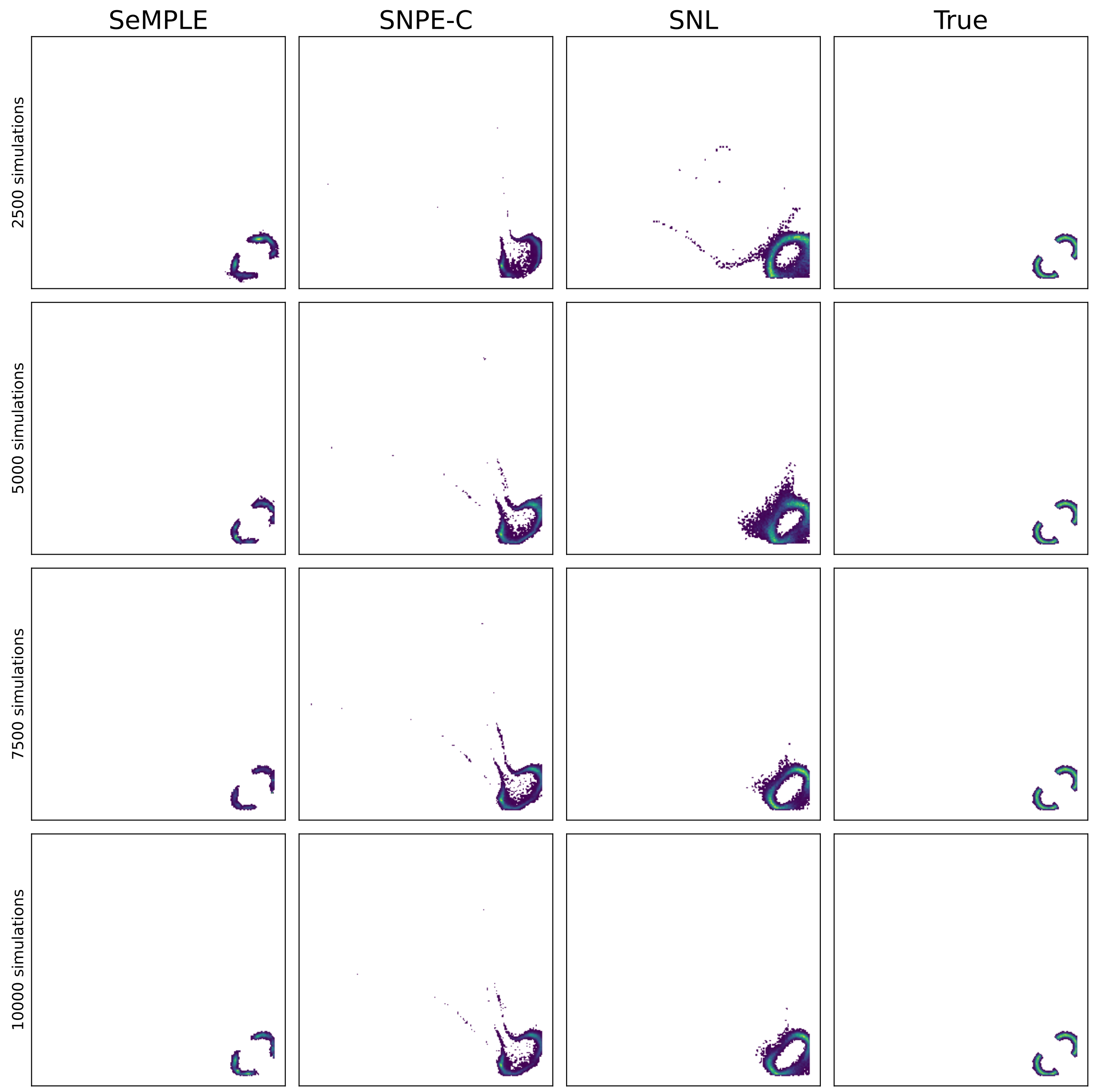


Figure 29: Two Moons using $R = 4$ rounds for every method: inference using observed dataset number 10 from SBIBM. Posterior samples from each algorithm round, starting from the first round (upper row) to the fourth (bottom row).

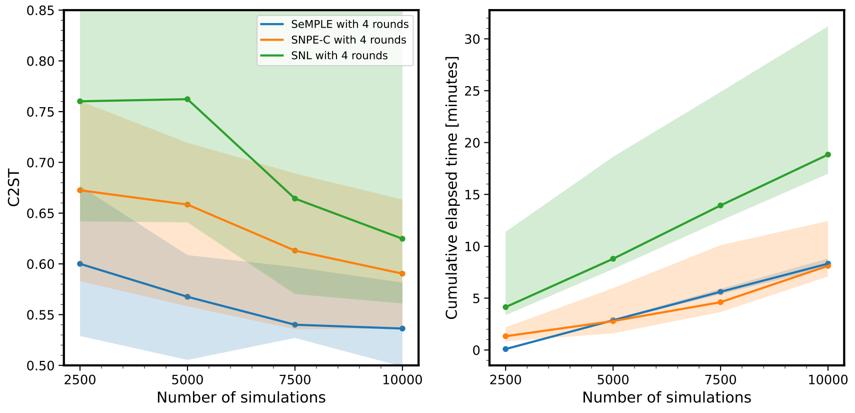


Figure 30: Two Moons using $R = 4$ rounds for every method. Median C2ST (left) and median cumulative runtime in minutes (right) for 10 runs with different data sets vs the number of model simulations. Shaded bands enclose the min and max values.

F Multiple hyperboloid model

F.1 Multiple hyperboloid model definition

This model was introduced by Forbes et al. (2022). With reference to applications in audio processing, the unknown parameter $\boldsymbol{\theta} \in \mathbb{R}^2$ can be interpreted as the coordinates of a sound source location on the plane, while 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 pairs of microphones aimed at detecting the sound source. An observation $\boldsymbol{y} \in \mathbb{R}^d$ has the likelihood function

$$p(\boldsymbol{y} \,|\, \boldsymbol{\theta}) = \frac{1}{2}\mathcal{S}(\boldsymbol{y}; F_{\boldsymbol{m}^1}(\boldsymbol{\theta})\mathbb{1}_d, \sigma^2 I_d, \nu) + \frac{1}{2}\mathcal{S}(\boldsymbol{y}; F_{\boldsymbol{m}^2}(\boldsymbol{\theta})\mathbb{1}_d, \sigma^2 I_d, \nu), \tag{19}$$

where

$$F_m(\boldsymbol{\theta}) = (||\boldsymbol{\theta} - \boldsymbol{m}_1||_2 - ||\boldsymbol{\theta} - \boldsymbol{m}_2||_2), \text{ with } \boldsymbol{m} = (\boldsymbol{m}_1, \boldsymbol{m}_2). \tag{20}$$

The likelihood in equation (19) is a mixture of two Student's t-distributions with a d-dimensional location parameter having all dimensions equal to $F_{\boldsymbol{m}^1}(\boldsymbol{\theta})$, $F_{\boldsymbol{m}^2}(\boldsymbol{\theta})$ respectively, a diagonal scale matrix σI_d and ν degrees of freedom. In our example, the observed data vector $\boldsymbol{y}_o$ has length $d = 10$, and $\nu = 3$, $\sigma = 0.01$, $\boldsymbol{m}_1^1 = (-0.5, 0)^T$, $\boldsymbol{m}_2^1 = (0.5, 0)^T$, $\boldsymbol{m}_1^2 = (0, -0.5)^T$ and $\boldsymbol{m}_2^2 = (0, 0.5)^T$. The prior is set to be a uniform $\mathcal{U}(-2, 2)$ for both components of $\boldsymbol{\theta}$, and $\boldsymbol{\theta}$ is the only parameter to infer. Data generating parameter is $\boldsymbol{\theta}^* = (1.5, 1)$ as in Forbes et al. (2022), which represents the coordinates of the location of a sound source.

Since the multiple hyperboloid model is not implemented in `SBIBM`, there exists no observed data sets or reference posterior samples in `SBIBM`. To produce an observed data set $\boldsymbol{y}_0$, the model simulator $p(\boldsymbol{y} \,|\, \boldsymbol{\theta}^*)$ was run with the true parameter value $\boldsymbol{\theta}^* = (1.5, 1)$. Since the posterior shape varies widely depending on the true parameter value, and to keep the connection to the experiment in Forbes et al. (2022), only a single 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.

F.2 Multiple hyperboloid model: selection of K via BIC

Figure 31 shows the BIC for different values of K computed by fitting GLLiM, prior to running SeMPLE, using the same prior-predictive data set $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$ with $N = 10\,000$. Isotropic covariance matrices were used in GLLiM as it was found that using fully-specified covariance matrices did not improve the 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 used in Forbes et al. (2022) (which was also selected for the same multiple hyperboloid model via BIC). The runtime to compute all the BIC values in Figure 31 was 270 seconds = 4.5 minutes. Note once again that the BIC computation is independent of the observed data set and has to be performed only once before running SeMPLE.

F.3 Multiple hyperboloid model: results with $R = 4$

In the main paper we have considered the inference results based on a total of 4×10^4 model simulations, uniformly distributed across $R = 10$ rounds for SNPE-C and SNL (as per `SBIBM` defaults), and $R = 4$ rounds for SeMPLE. Results showed that SeMPLE performed better and in particular that SNL (which is also an MCMC-based procedure, similarly to SeMPLE) was returning the worst performance. Here we show the performance of both SNPE-C and SNL with $R = 4$, again using a total of 4×10^4 model simulations for each method. Figures 32-34 show posterior samples from the last round (r=4) of each algorithm for each of the 10 independent algorithm runs. We can clearly notice that SNL fails completely and therefore, for ease of reading, the SNL marginal posteriors are not reported. However, the differences between SNPE-C and SeMPLE are more subtle. It appears that occasionally SeMPLE samples from central areas, where there should not be any posterior mass, while SNPE-C seems to undersample some of the *branches*. Therefore, we also plot the corresponding kernel-smoothed marginals in Figures 32-34. There, it appears that the

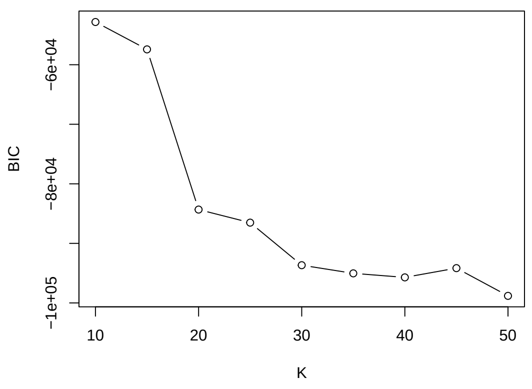
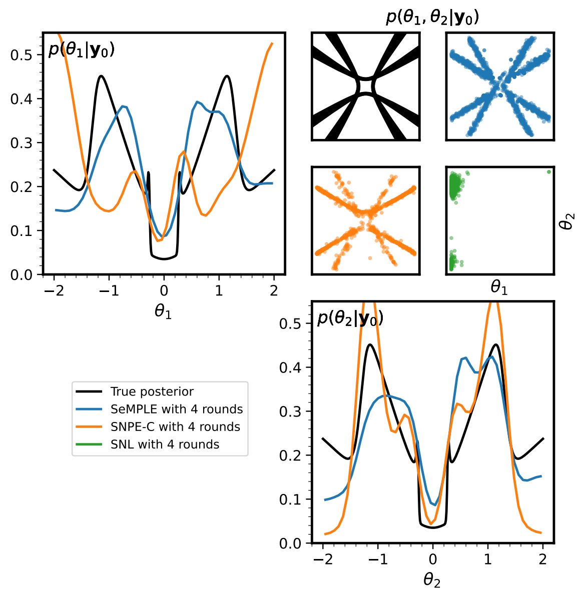


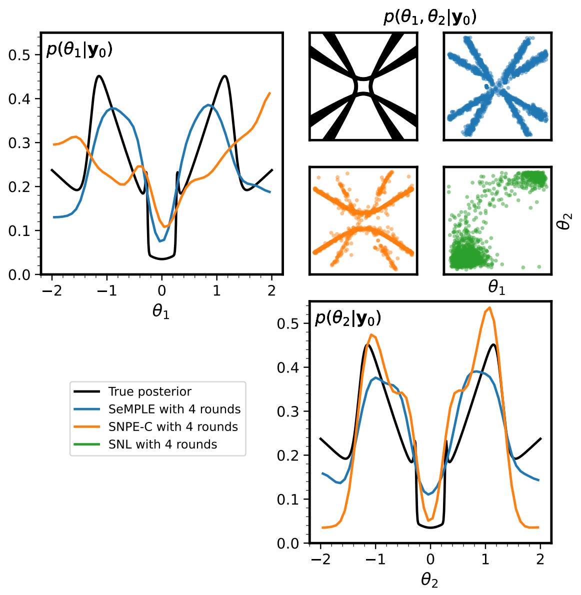
Figure 31: Multiple hyperboloid. BIC with respect to the number of Gaussian mixture components K.

marginals from SeMPLE are overall closer to the reference marginals, which is in agreement with the better C2ST attained by SeMPLE (see the main text).

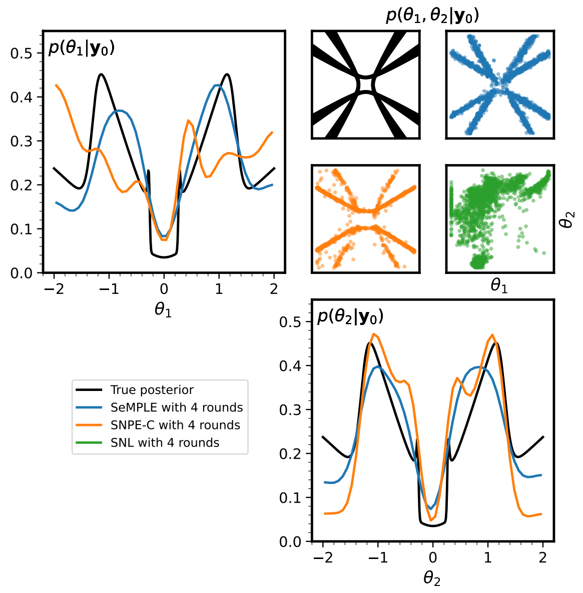
Figure 35 shows C2ST and runtime values in the case when $R = 4$ rounds for all algorithms. The C2ST results of SNPE-C and SNL are very similar with $R = 4$, meaning that SNL fails completely and SeMPLE still performs much better than SNPE-C. In terms of runtime, SNPE-C is still about 7 times slower than SeMPLE, disregarding SNL as the inference from this method is not useful.



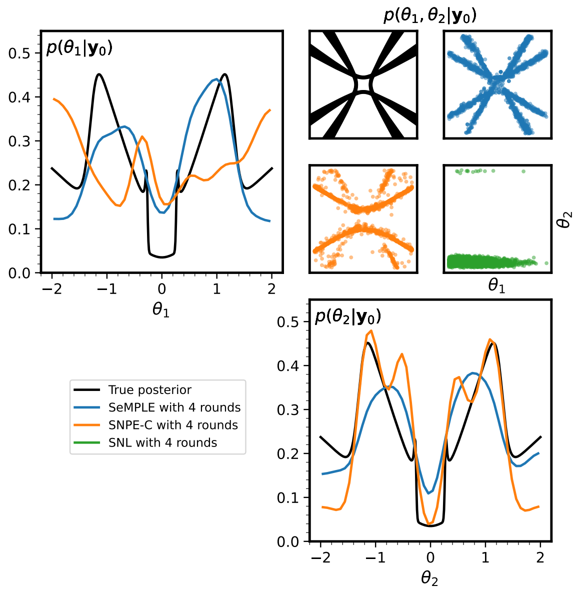
(a) Repetition 1



(b) Repetition 2

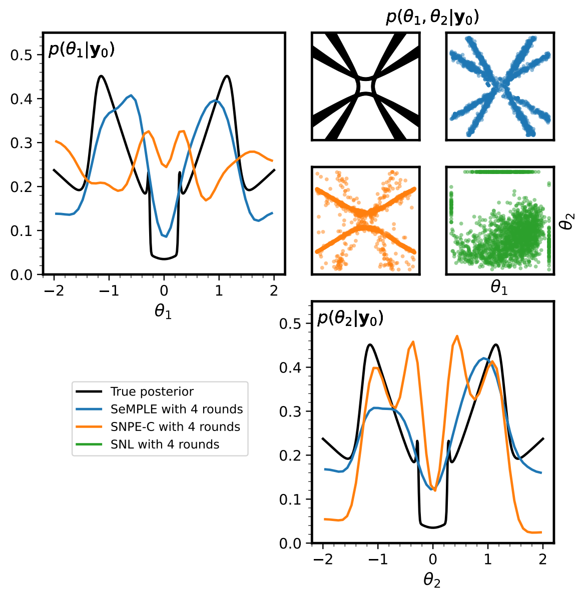


(c) Repetition 3

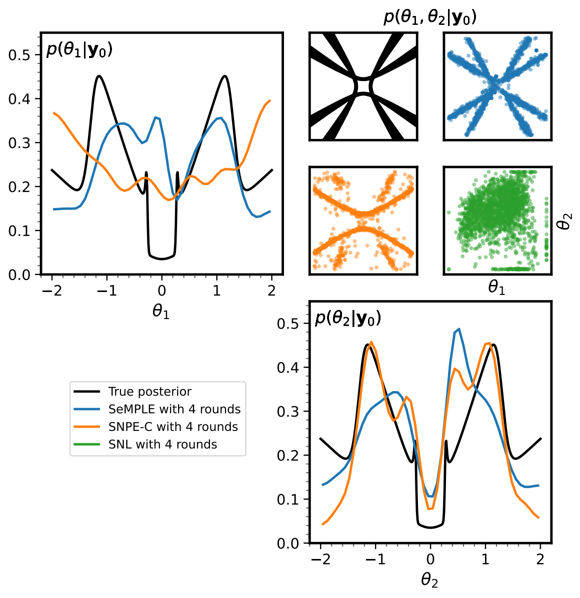


(d) Repetition 4

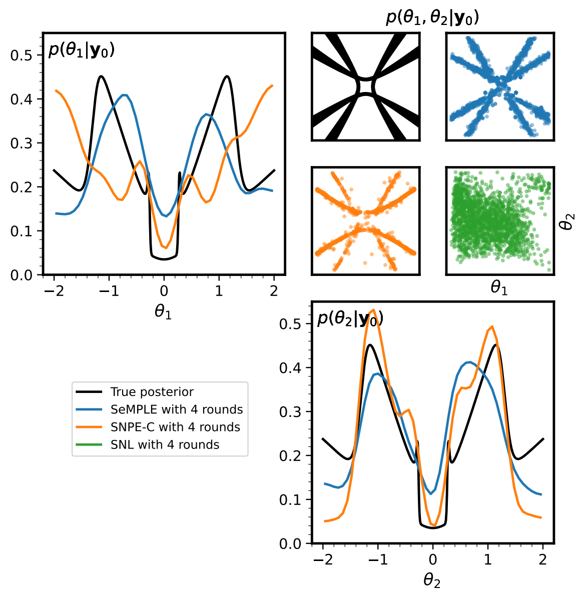
Figure 32: Multiple hyperboloid model. Pair plots of posterior samples from the last round ($r = 4$) of each algorithm. The SNL marginal posteriors are not reported for ease of reading, since the SNL inference fails.



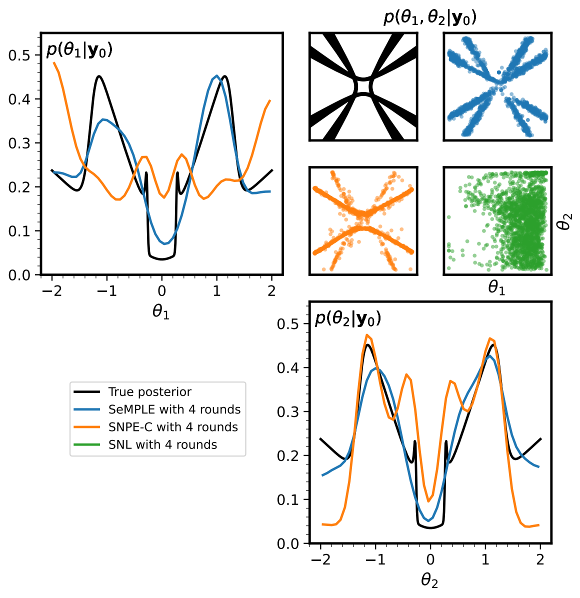
(a) Repetition 5



(b) Repetition 6

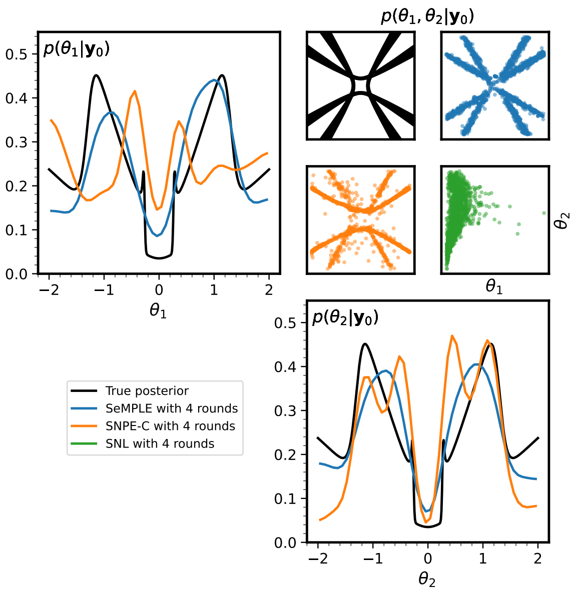


(c) Repetition 7

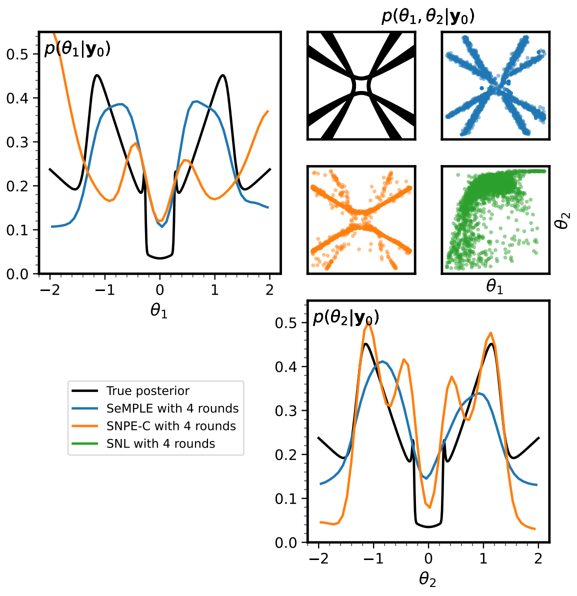


(d) Repetition 8

Figure 33: Multiple hyperboloid. Pair plots of posterior samples from the last round ($r = 4$) of each algorithm. The SNL marginal posteriors are not reported for ease of reading, since the SNL inference fails.



(a) Repetition 9



(b) Repetition 10

Figure 34: Multiple hyperboloid. Pair plots of posterior samples from the last round ($r = 4$) of each algorithm. The SNL marginal posteriors are not reported for ease of reading, since the SNL inference fails.

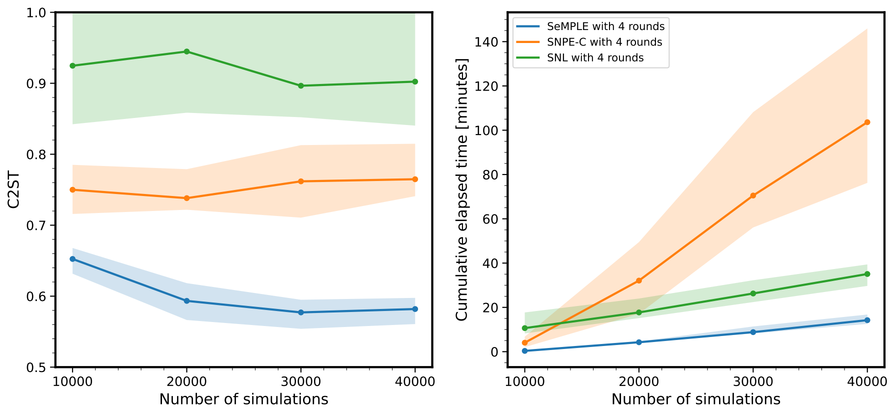


Figure 35: Hyperboloid. Median C2ST (left) and median cumulative runtime in minutes (right) for 10 runs with the same data vs the number of model simulations. Shaded bands enclose the min and max values.

G Bernoulli GLM model

G.1 Bernoulli GLM model definition

For the model definition, please see section T.5 in the appendix of Lueckmann et al. (2021). The prior distribution and model parameters can also be found there.

G.2 Inference setup

For all inference algorithms, we executed runs with the 10 different observed data sets provided in `SBIBM`. The total simulation budget was always set to $10,000$ model simulations, but two different setups were used for the number of algorithm rounds. In the first setup all algorithms used $R = 2$ algorithm rounds and in the second setup SNL and SNPE-C used $R = 10$ rounds while SeMPLE still used $R = 2$ rounds. The reason for running $R = 2$ rounds with SeMPLE was that for this model the output from $r = 2$ produced accurate inference but had some stickiness in the Markov chain that caused problems in the GLLiM learning in round $r = 3$. Hence, the best inference results with SeMPLE were obtained with $R = 2$. The GLLiM covariance structure was set to be a full unconstrained matrix as our experiments showed that this improved the inference results compared to a constrained covariance matrix structure. Figure 36 shows BIC for several values of K, computed by fitting GLLiM using the same prior-predictive data set $\{\boldsymbol{\theta}_n, \boldsymbol{y}_n\}_{n=1}^N$ with $N = 2\,500$ prior to running SeMPLE. The minimum value is obtained at $K = 2$ but because of this small value, K was instead set to 10 as this gives more flexibility to the model without resulting in significantly longer runtimes. The runtime to compute all BIC values in Figure 36 was 16 seconds. Note that the BIC computation is independent of the observed data set and only has to be performed once before running SeMPLE.

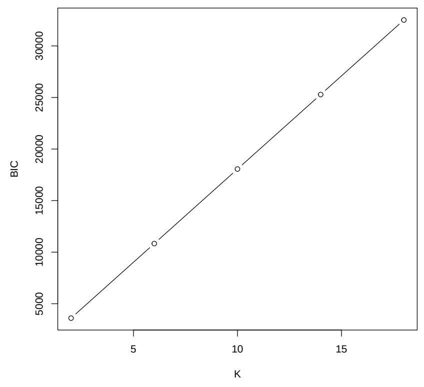
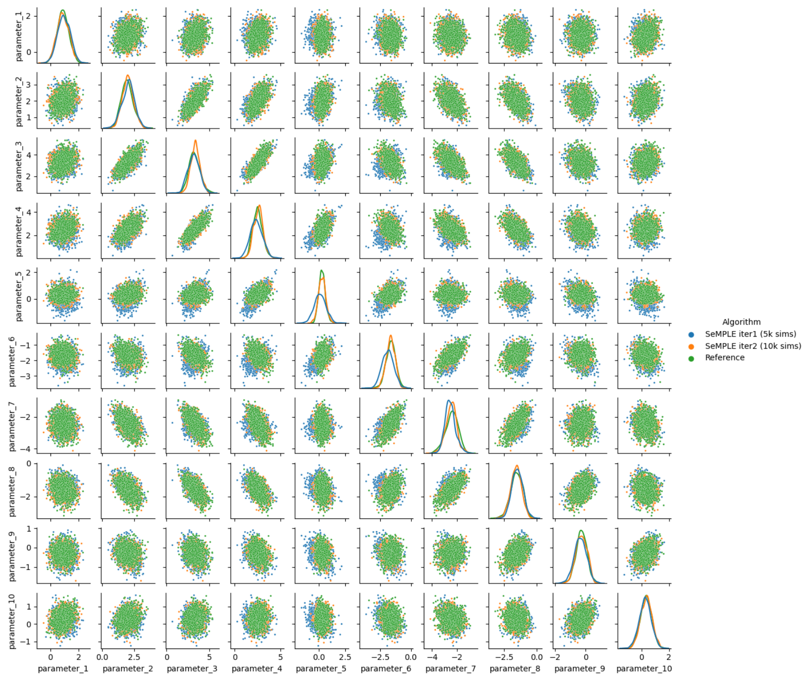


Figure 36: Bernoulli GLM model. BIC with respect to the number of Gaussian mixture components K.

G.3 Results when using $R = 2$ for all algorithms

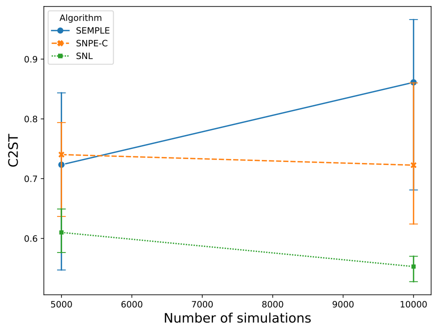
Here we compare results via SNL, SNPE-C and SeMPLE with the true posterior using $R = 2$ rounds for all algorithms. Since we consider a total budget of 10,000 model simulations, Figure 38 shows the C2ST and Wasserstein performance evaluated at 5,000 an 10,000 model simulations. We found that C2ST can be problematic with unimodal targets, and we see this also with the Ornstein-Uhlenbeck results (which also has a unimodal posterior), where the performance of the several algorithms seems to be better evaluated using Wasserstein distances. In fact, as from Figure 37 it is clear that the results from SeMPLE at $r = 2$ are noticeably improved compared to $r = 1$, and this is captured by the Wasserstein distances in Figure 38(b), whereas C2ST produces a contradicting result in Figure 38(a). We therefore alert the reader that with unimodal targets C2ST is at times not always consistent with other measures, such as posterior plots (see also the Ornstein-Uhlenbeck section), while we found C2ST useful with multimodal targets. Moreover, the inference from SeMPLE is not only accurate, but the computational gain provided by SeMPLE is striking,



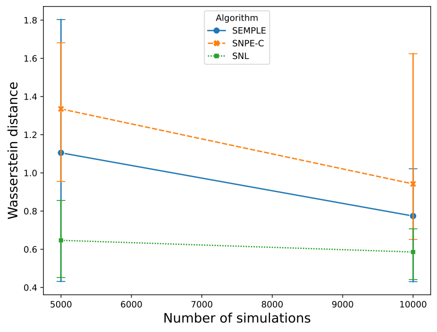
(a)

Figure 37: Bernoulli GLM model. Posterior plots obtained with SeMPLE and obtained at round $r = 1$ after 5,000 model simulations, and at $r = 2$ after 10,000 model simulations.

as illustrated in Figure 39, where SeMPLE is 8 times faster than SNPE-C and 19 times faster than SNL. These figures increase dramatically for SNL and SNPE-C in next section, where these methods are run with the default `SBIBM` setup.



(a)



(b)

Figure 38: Bernoulli GLM model with $R = 2$ for all algorithms. (a) Median C2ST of 10 runs with different data sets vs number of model simulations. (b) Median Wasserstein distance of 10 runs with different data sets vs number of model simulations. Error bars show min/max values. **For unimodal posteriors (as in this case) Wasserstein distances should be preferred to C2ST.**

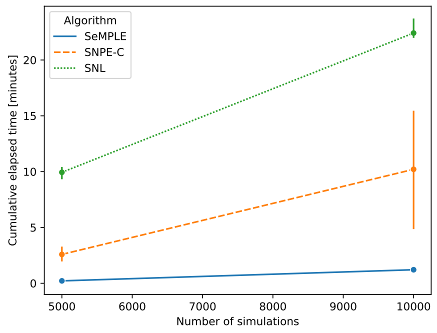
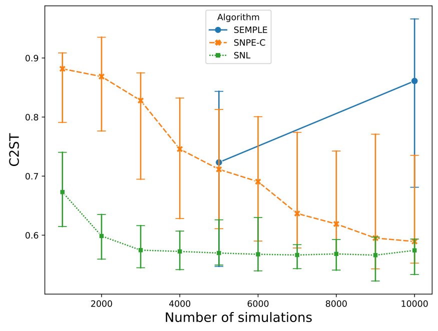
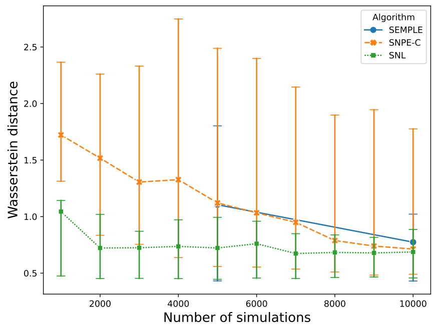


Figure 39: Bernoulli GLM model with $R = 2$. Median running times of 10 runs using several methods. Error bars show min/max values.



(a)



(b)

Figure 40: Bernoulli GLM model with $R = 10$ (SNL/SNPE-C) and $R = 2$ (SeMPLE). (a) Median C2ST of 10 runs with different data sets vs number of model simulations. (b) Median Wasserstein distance of 10 runs with different data sets vs number of model simulations. Error bars show min/max values. **For unimodal posteriors (as in this case) Wasserstein distances should be preferred to C2ST.**

G.4 Results when using $R = 10$ for SNL/SNPE-C and $R = 2$ for SeMPLE

We executed the experiment anew, but this time by allocating the 10^4 model simulations uniformly across 2 rounds for SeMPLE and 10 rounds for SNL and SNPE-C (as per `SBIBM` defaults). Results are in Figure 40 and Figure 41. Once more, we recommend the reader to consider the Wasserstein distances and not C2ST for this example, since that posterior is unimodal, as motivated in the previous section. The distances show that on average SeMPLE is as accurate as SNL but not before 10^4 model simulations are executed, while SNPE-C shows a large variability in the results. Once more, SeMPLE comes with the benefit of a much faster inference (Figure 41): in this case SeMPLE is 26 times faster than SNPE-C and 74 times faster than SNL.

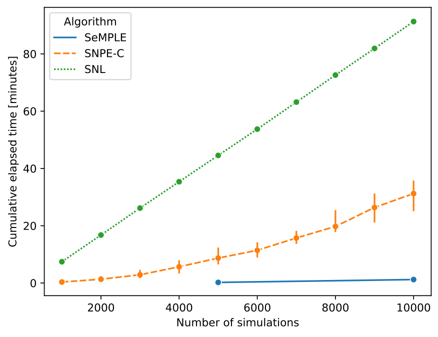


Figure 41: Bernoulli GLM model with $R = 10$ (SNL/SNPE-C) and $R = 2$ (SeMPLE). Median running times of 10 runs using several methods. Error bars show min/max values.

H Ornstein-Uhlenbeck

The Ornstein-Uhlenbeck process has original applications in physics to model the velocity of a Brownian particle (Uhlenbeck & Ornstein, 1930), and has found application in many contexts such as neuronal modelling and finance, to mention a few. In our work, it serves as an application where the data is a time series, while the likelihood function is tractable and therefore it is possible to obtain samples from the true posterior, to use as reference. The Ornstein-Uhlenbeck process $\{X_t\}_{t\geq 0}$ is defined by the stochastic differential equation (SDE)

$$dX_t = -\beta(X_t - \alpha)dt + \sigma dW_t, \tag{21}$$

where $\beta > 0$, $\alpha \in \mathbb{R}$, $\sigma > 0$, $X_0 = x_0$ and W_t denotes the Wiener process (Brownian motion). The solution to equation (21) is

$$X_t = \alpha + (x_0 - \alpha)e^{-\beta t} + \sigma \int_0^t e^{-\beta(t-s)} dW_s. \tag{22}$$

The process has known Gaussian transition densities: for a generic time t and an arbitrary state x_0 at time $t_0 = 0$, the transition density is given as

$$(X_t \,|\, X_0 = x_0) \sim \mathcal{N}\Big(\alpha + (x_0 - \alpha)e^{-\beta t}, \frac{\sigma^2}{2\beta}(1 - e^{-2\beta t})\Big). \tag{23}$$

The transition density (23) 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

$$(X_{t_i} \,|\, X_{t_{i-1}} = x_{t_{i-1}}) \sim \mathcal{N}\Big(\alpha + (x_0 - \alpha)e^{-\beta \Delta_i}, \frac{\sigma^2}{2\beta}(1 - e^{-2\beta \Delta_i})\Big), \quad i = 1, \ldots, n-1. \tag{24}$$

This makes the Ornstein-Uhlenbeck model one of the very few examples of SDEs where it is trivial to evaluate the exact likelihood function. The likelihood function for the parameters $\boldsymbol{\theta} = (\alpha, \beta, \sigma)$ is written as (due to the Markovianity of the solution of an SDE)

$$\mathcal{L}(x_0, \ldots, x_{n-1} \,|\, \boldsymbol{\theta}) = \prod_{i=1}^{n-1} p(x_i \,|\, x_{i-1}; \boldsymbol{\theta}), \tag{25}$$

where $p(x_i \,|\, x_{i-1}; \boldsymbol{\theta})$ is the transition density in equation (24) (we assume x_0 a fixed constant throughout, otherwise a multiplicative term $p(x_0|\boldsymbol{\theta})$ would have to appear in (25)). The interpretation of the parameter α is the asymptotic mean. This can be seen by letting $t \to \infty$ in equation (23) which shows that the process' stationary distribution is $X_t \sim \mathcal{N}(\alpha, \frac{\sigma^2}{2\beta})$. 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 reacts to perturbations.

H.1 Inference setup

The fixed initial value was set to $x_0 = 0$. We set uniform priors $\alpha \sim \mathcal{U}(0, 10)$, $\beta \sim \mathcal{U}(0, 5)$ and $\sigma \sim \mathcal{U}(0, 2)$. In this study, 50 points were simulated from the process to obtain a time series of length $n = 51$ (including the fixed starting value x_0) using $\boldsymbol{\theta} = (\alpha, \beta, \sigma) = (3, 1, 0.5)$ as ground-truth. 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. The resulting observed data is shown in Figure 42. Similarly to the multiple hyperboloid model, a single observed data set was produced by running the simulation model once with the true parameter values $\boldsymbol{\theta} = (3, 1, 0.5)$.

The covariance structure in GLLiM was set to be a full unconstrained matrix, which is reasonable given the time-dependent structure, and in your experiments it was found that this improved the inference results substantially. Figure 43 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 20 to have some buffer to remove unnecessary mixture components if needed. The runtime to compute all BIC values in Figure 43 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.

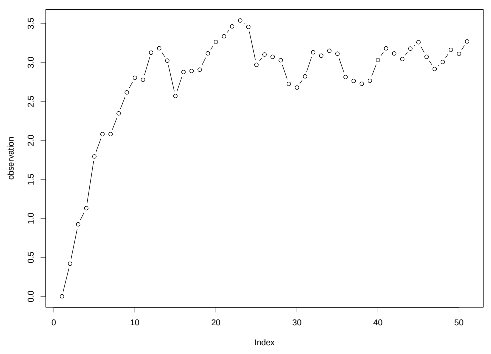


Figure 42: Ornstein-Uhlenbeck: Observed time series data of the process.

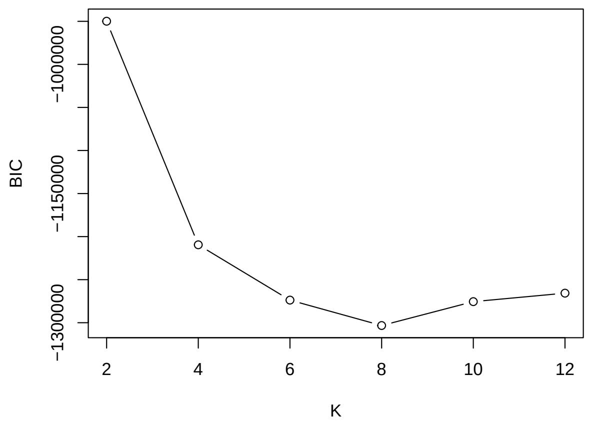


Figure 43: Ornstein-Uhlenbeck: BIC as a function of number of Gaussian mixture components K.

H.2 Results

The posterior is unimodal and therefore, following the results for the Bernoulli GLM example, we find it useful to report both the C2ST metric and Wasserstein distances. For all inference algorithms, we executed 10 independent runs using the same observed data set. We ran experiments using two configurations by always allocating the same total number of 4×10^4 model simulations for all methods: the first configuration uses $R = 4$ rounds for all methods (section H.3). The second configuration uses $R = 4$ rounds for SeMPLE and $R = 10$ for both SNL and SNPE-C (section H.4), as $R = 10$ is the default implementation in `SBIBM` for SNL and SNPE-C. It is instructive to look first at the case where $R = 4$ is used for all algorithms, as there we can make useful remarks about the behaviour of C2ST and the Wasserstein distance.

H.3 Results when using $R = 4$ for all algorithms

Figure 44 shows the C2ST metric and Wasserstein distance as a function of the number of model simulations for each algorithm. At round r=2 (20 000 model simulations) SNL consistently shows an unexpected behaviour where the posterior sample is clearly worse than in the first round (r=1). This happened in some scenarios with SNL where the number of simulations in the first round is low enough. Apart from this, SNL scores best in both metrics after the full 40 000 model simulations (R=4 algorithm rounds). However, notice that the difference between SeMPLE and SNL is much smaller in terms of the Wasserstein distance.

In fact, by looking at the marginal posteriors (Figures 46–49) it is clear that while SNL posteriors better approximate the true posterior, SeMPLE is a strong second best alternative, and provides clearly better posteriors compared to SNPE-C. However the latter result is not apparent when looking at the C2ST metric, where SeMPLE and SNPE-C do not seem strikingly different. Therefore, for this experiment with a unimodal target, similarly to the Bernoulli GLM example, we find that the Wasserstein distance provides a better summary of the quality of the posterior inference. 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.

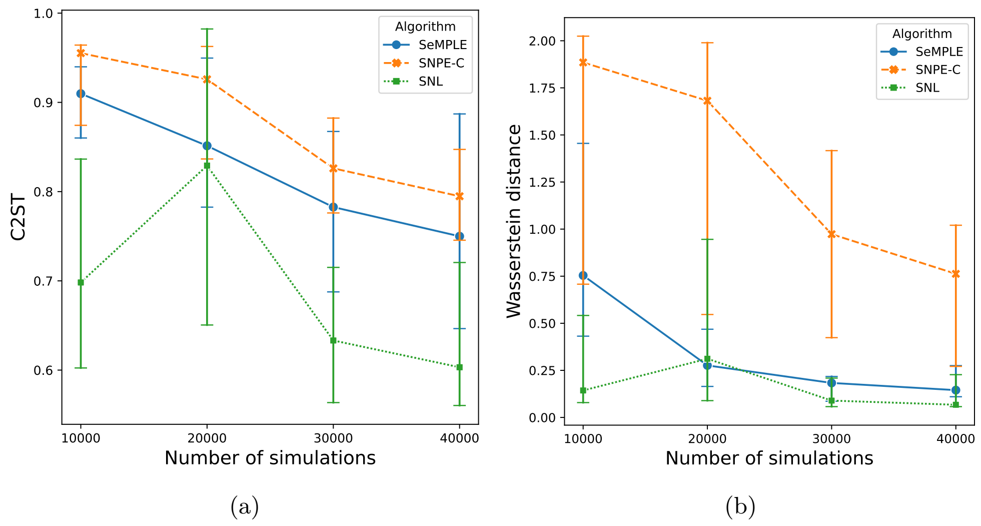


Figure 44: Ornstein-Uhlenbeck, results when $R = 4$ for all algorithms: median C2ST and Wasserstein distance across 10 independent runs (with the same observed data set) as a function of the number of model simulations. Error bars show min/max values. **For unimodal posteriors (as in this case) Wasserstein distances should be preferred to C2ST.**

In terms of runtime, Figure 45 shows that SeMPLE has a much much smaller runtime compared to both SNL and SNPE-C, even if were to include 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. Plots of the posterior marginals and posterior draws from the last round (r=4) of each algorithm are in Figures 46-47, for each of the 10 independent runs. Acceptance rates of the Metropolis-Hastings step of SeMPLE are in Figure 48, showing that, particularly for the third and fourth round, the acceptance rate is fairly high (compatibly with being an MCMC algorithm).

H.4 Results when using $R = 10$ for SNL/SNPE-C and $R = 4$ for SeMPLE

We executed the experiment anew, but this time by allocating the 4×10^4 model simulations uniformly across 4 rounds for SeMPLE and 10 rounds for SNL and SNPE-C (as per `SBIBM` defaults). Results are in Figure 49 and Figure 50. As explained in previous sections, for this experiment it is more trustworthy to consider the Wasserstein distances, hence Figure 49(b) in this case. Once again we show that, even by comparing with SNL and SNPE-C trained over many inference rounds (as per `SBIBM` defaults), SeMPLE is competitive. Moreover, in terms of runtime, if we include the 16 minutes required to run the BIC procedure to select K (recall this is amortized so it can be reused with other observations), the median runtime for SeMPLE is around 21 minutes, while SNPE-C requires around 136 minutes to complete the 10 rounds on average. Hence SeMPLE implies a more than 6-fold acceleration, which becomes around 27-fold if we exclude the BIC procedure.

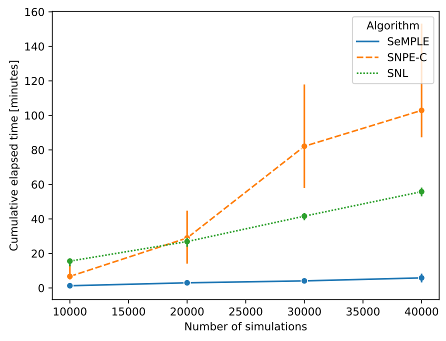
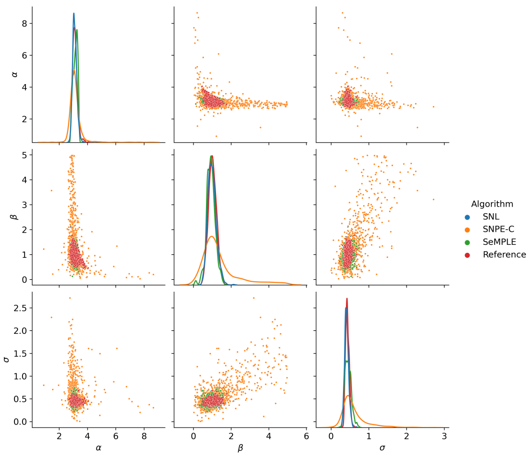
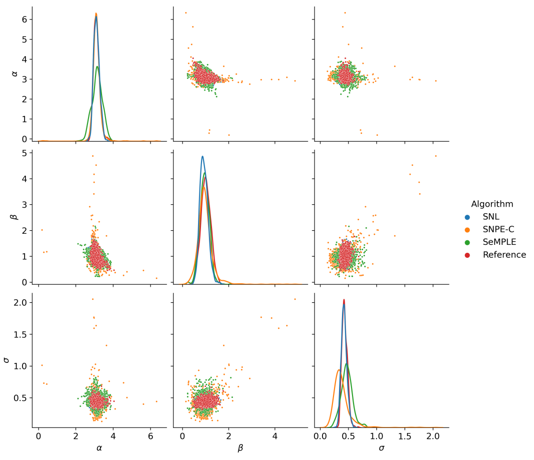


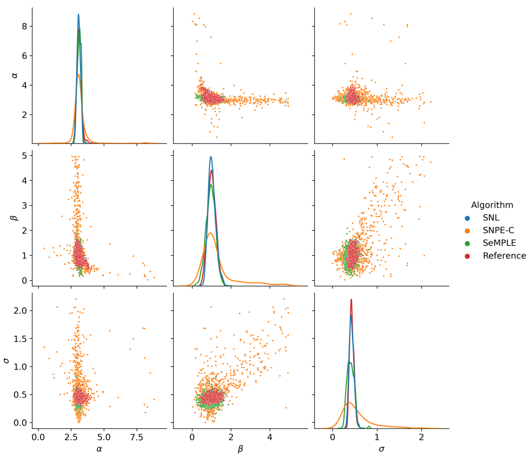
Figure 45: Ornstein-Uhlenbeck, results when $R = 4$ for all algorithms: median cumulative runtime of 10 independent runs as a function of number of model simulations. Error bars show min/max values.



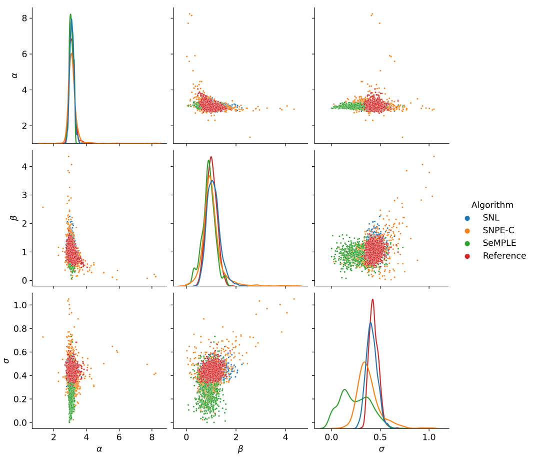
(a) Repetition 1.



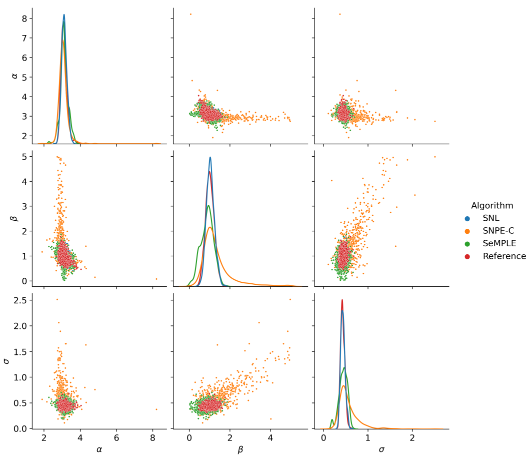
(b) Repetition 2.



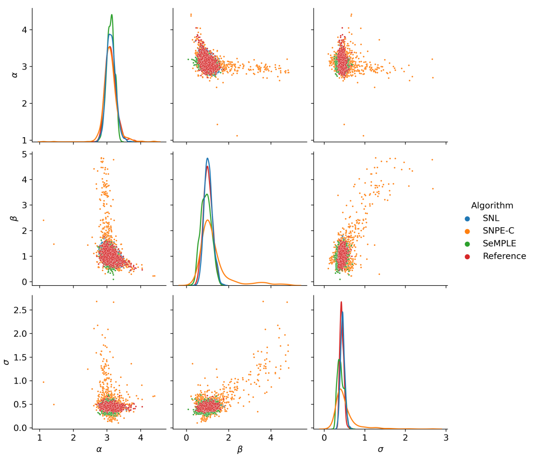
(c) Repetition 3.



(d) Repetition 4.

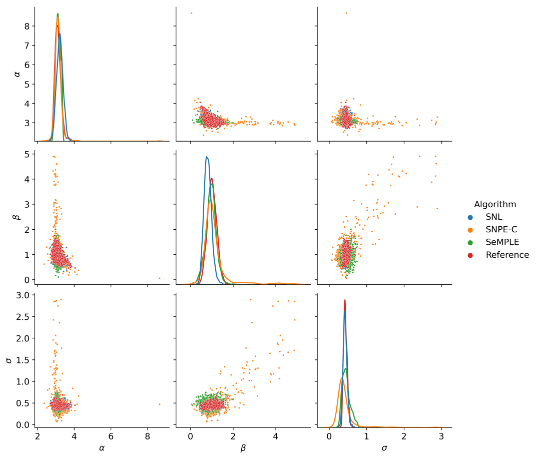


(e) Repetition 5.

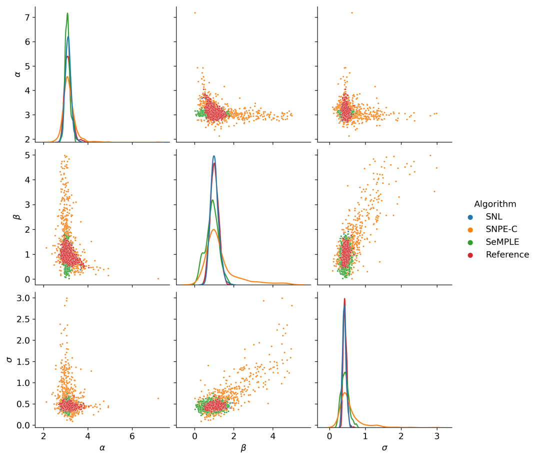


(f) Repetition 6.

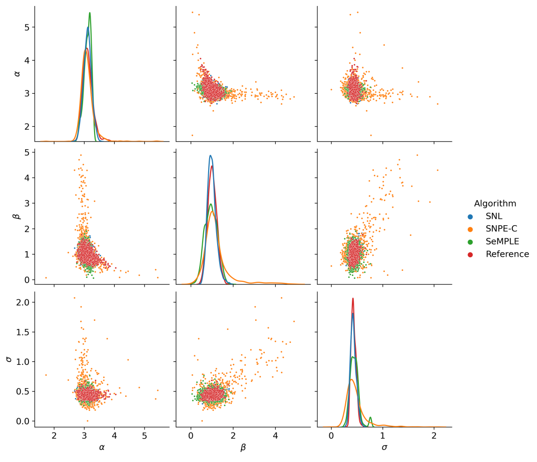
Figure 46: Ornstein-Uhlenbeck, results when $R = 4$ for all algorithms, runs 1–6.: marginal posteriors and posterior draws from the last round (R=4) of each algorithm.



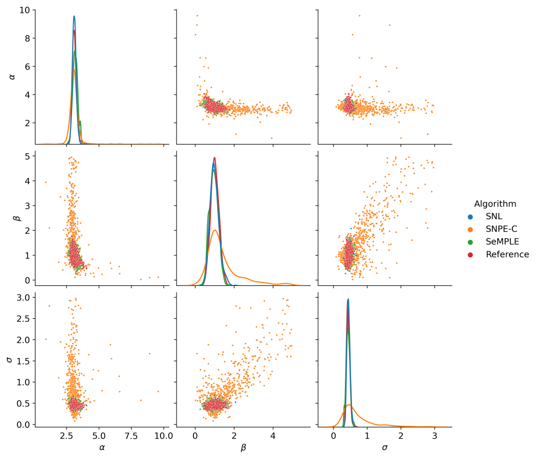
(a) Repetition 7.



(b) Repetition 8.



(c) Repetition 9.



(d) Repetition 10.

Figure 47: Ornstein-Uhlenbeck, results when $R = 4$ for all algorithms, runs 7-10: marginal posteriors and posterior draws from the last round (R=4) of each algorithm.

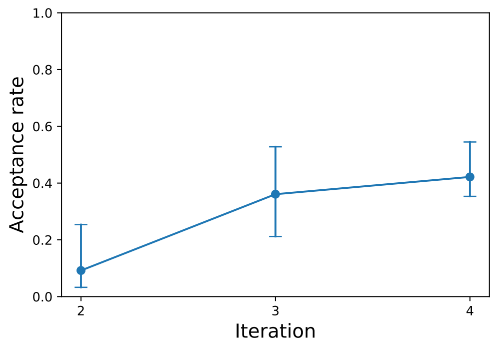
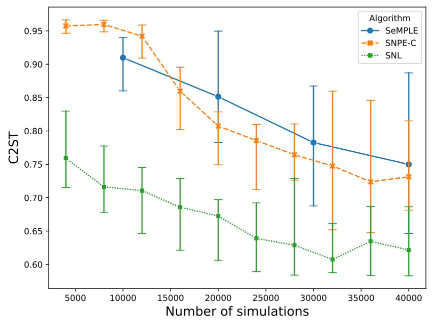
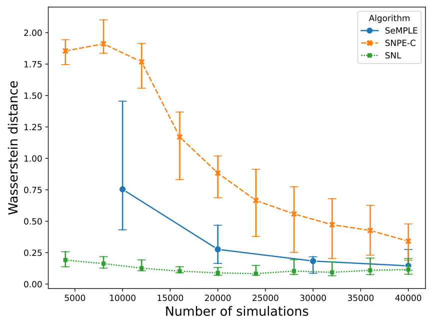


Figure 48: Ornstein-Uhlenbeck: Median acceptance rate of the Metropolis-Hastings step of 10 independent SeMPLE runs with the same data set. Error bars show min/max values.



(a)



(b)

Figure 49: Ornstein-Uhlenbeck, results with $R = 4$ (SeMPLE) and $R = 10$ (SNL/SNPE-C): median C2ST and Wasserstein distance across 10 independent runs (with the same observed data set) as a function of the number of model simulations. Error bars show min/max values. **For unimodal posteriors (as in this case) Wasserstein distances should be preferred to C2ST.**

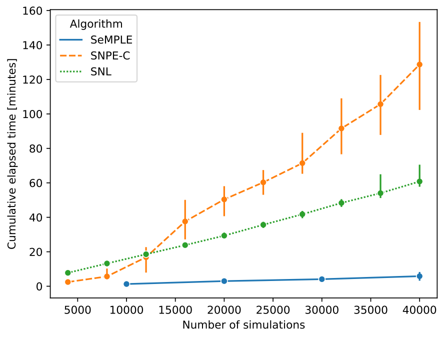


Figure 50: Ornstein-Uhlenbeck, results with $R = 4$ (SeMPLE) and $R = 10$ (SNL/SNPE-C): median cumulative runtime of 10 independent runs as a function of number of simulations. Error bars show min/max values.

I Lotka-Volterra (Markov jump process with measurement error)

The Lotka-Volterra predator prey model is a population dynamics model with two species. Its definition varies, depending on whether it is defined via an ordinary differential equation (as in Lueckmann et al., 2021), a continuous time Markov Jump Process (MJP, Owen et al., 2015a;b) or a continuous time approximation to the MJP given by a stochastic differential equation (as in Golightly & Wilkinson, 2011). Moreover, it could be seen as a model involving three (as in Owen et al., 2015a) or four (as in Papamakarios & Murray, 2016) reactions. We employ a stochastic MJP version of this model as described in Owen et al. (2015a), characterized by a set of 3 reactions on the two species, with additional measurement error.

I.1 Model definition with three reactions

Denoting the two species, preys in number X_1 and predators in number X_2, evolution of the system is governed by the following three reactions:

$$\begin{array}{lccc} \mathbf{R}_1: & X_1 & \rightarrow & 2X_1 \\ \mathbf{R}_2: & X_1+X_2 & \rightarrow & 2X_2 \\ \mathbf{R}_3: & X_2 & \rightarrow & \emptyset. \end{array} \tag{26}$$

These reactions can be interpreted respectively as a prey birth, a predator prey interaction resulting in the death of a prey and a predator birth, and a predator death respectively. Let $X_t = (X_{1,t}, X_{2,t})$ denote the number of both species at time t. Each reaction $\mathbf{R}_i$ is assumed to have an associated constant θ_i and hazard function $h_i(X_t, \theta_i)$ which gives the propensity for a reaction event of type i at time t to occur. Reaction events are dependent on the current state of the system as well as the reaction rate parameters. Hence the evolution of species counts corresponds to a Markov process on a discrete state space. This reaction network is summarised by its stoichiometry matrix, S and hazard function $h(\mathbf{X}, \theta)$:

$$S = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix}, \qquad h(\mathbf{X}, \theta) = (\theta_1 X_1, \theta_2 X_1 X_2, \theta_3 X_2). \tag{27}$$

I.2 Inference setup

To simulate the exact dynamics, we use the `smfsb` R package (Wilkinson, 2024), implementing the "Gillespie algorithm" (Gillespie, 1977). Whilst the Gillespie algorithm allows exact simulation of the time and type of each reaction event that occurs, observed data $\mathbf{Y} = (y_0, y_1, \ldots, y_T)$ are typically noisy observations at discrete time intervals, where each y_t is 2-dimensional with $y_t^j \sim \pi(|X_{j,t}, \sigma)$ *i.i.d.* for $j = 1, 2$. We corrupt each $X_{j,t}$ with *i.i.d.* Gaussian error with mean 0 and variance σ^2, $\pi(y_t^j | X_{j,t}, \sigma^2) \sim \mathcal{N}(X_{j,t}, \sigma^2)$. The true parameter values generating the observed data is set to $\boldsymbol{\theta} = (1, 0.005, 0.6)$ and the starting values $X_0 = (50, 100)$ are used throughout. When generating the observed data we let $\sigma = 30$ and later infer σ along with the model parameters $\boldsymbol{\theta}$, resulting in a total of 4 parameters.

When computing model realisation with the Gillespie algorithm, we let $T = 30$ and observe the Markov jump process at uniformly spaced intervals with step length 0.2, resulting in a time series of length 150 for each species. After corrupting the time series data with the previously described Gaussian error we compute a total of 9 summary statistics that is used when performing the inference. These summary statistics, first suggested in Wilkinson (2013), consist of the mean, (log) variance, and the lag 1 and lag 2 two auto-correlations of the time series of each species. Additionally the cross-correlation between the two time series is included in the summary statistic.

Following the setup in Owen et al. (2015a), we place uniform prior information on $\log(\theta)$,

$$\log(\theta_i) \sim \mathcal{U}(-6, 2), \quad i = 1, 2, 3. \tag{28}$$

and $\log(\sigma)$

$$\log(\sigma) \sim \mathcal{U}(\log(0.5), \log(50)). \tag{29}$$

Prior to running the inference, for each of the nine summary statistics we computed trimmed means, where 1.25% of the values were trimmed from each end and corresponding trimmed standard deviations, obtained from 10,000 prior predictive simulations, and we used these to standardise both the observed summaries and the summaries simulated during the inference. For each SeMPLE run we set a budget of 30 000 model simulations, uniformly distributed across 3 algorithm rounds. The number of mixture components is set to $K = 15$ (a number which was picked without any specific tuning) and we set the covariance matrices in GLLiM to be full unconstrained matrices, all different across the K components. Moreover, we do not inflate the covariance in the Metropolis-Hastings step (ie we set $\gamma = 1$). The mixture probability threshold to remove negligible mixture components during the SeMPLE run was set to 0.005.

For a MJP Lotka-Volterra model, we did not find a ready implementation in `SBIBM` (there, an ODE-based Lotka-Volterra model is provided, not a MJP version), and this is likely because with such a model it is impossible to perfectly control the computational effort, as for a given value of θ the time spent to complete a trajectory is random. For this reason, we found more practical to compare SeMPLE with approximate Bayesian computation (ABC) experiments using the code from Picchini & Tamborrino (2024). For SMC-ABC, we used the same observed data and the same standardised summary statistics as with SeMPLE. We consider sequential Monte Carlo ABC (SMC-ABC), and also use the posterior from SMC-ABC as "reference posterior". SMC-ABC is the state-of-art approach to ABC inference and is the algorithm of choice in recent inference platforms such as `ABCpy` (Dutta et al., 2017) and `pyABC` (Schälte et al., 2022). We consider two versions of SMC-ABC that differ in terms of proposal function: the first one is the "classic" SMC-ABC, in that it uses as proposal distribution a Gaussian random walk of the type $\theta^{**} \sim N(\theta^*, 2\Sigma)$ (Beaumont et al., 2009, Filippi et al., 2013), where θ^* is a parameter "particle" randomly sampled from the most recent set of "parameter particles" (these providing an ABC approximation to the posterior), and Σ is the empirical weighted covariance matrix of these "parameter particles". This first sampler is named `standard` in Picchini & Tamborrino (2024), as it is the most common SMC-ABC proposal sampler, as found in most literature and in available software packages. A second SMC-ABC sampler is the one denoted `blockedopt` in Picchini & Tamborrino (2024), which is also a Gaussian proposal sampler but not of random-walk type, and is much more resource-effective, as it more rapidly "guide" the parameters towards the bulk of the posterior region (see Picchini & Tamborrino, 2024 for details). We ran both SMC-ABC versions with 1,000 particles.

I.3 Results

For this case study, we consider the posterior returned by the `blockedopt` SMC-ABC as "reference posterior", as it is obtained after more than three million model simulations (specifically 3,337,640 simulations), that is a computational budget more than 100 times larger than the budget used for SeMPLE. In Figure 51 we show that, while the "guided" `blockedopt` SMC-ABC can infer $(\theta_1, \theta_2, \theta_3)$ for a smaller budget than millions of simulations, however inferring also σ does requires many additional simulations. When we use the `standard` SMC-ABC, results after 3,110,349 model simulations are in Figure 52, and it is clear that the variability in the inference is much larger, and in particular `standard` struggles in inferring σ.

It is of interest to compare results for SeMPLE and SMC-ABC for a similar number of model simulations. We anticipate to observe something that is well known already from Greenberg et al. (2019); Lueckmann et al. (2021), namely the SMC-ABC inefficiency in terms of model simulations. What we mostly wish to verify is the ability for SeMPLE to return accurate inference. In Figure 53 we compare results when using around 30,000 model simulations, however notice that it is impossible to compare the methods for the exact same budget, as SMC-ABC employs while-loops that keep repeating model simulations until a prefixed number of parameter proposals is accepted. It is clear that when the number of model simulations is limited to around 30,000, neither of the SMC-ABC versions can produce accurate inference. The SMC-ABC methods struggle especially with estimating the noise parameter σ when the number of model simulations is limited. The SeMPLE posterior after 30,000 model simulations is more similar to the `blockedopt` SMC-ABC posterior obtained after approximately 3.3 million model simulations, showing the computational efficiency of SeMPLE with a much smaller simulation budget.

To give an idea of the variation in the SeMPLE posterior sample output, in Figure 54 we report 5 independent SeMPLE runs with the same observed data as in previous analyses. We see that SeMPLE consistently provides accurate inference of the model parameters $\boldsymbol{\theta}$ in multiple repetitions. There is a larger variation

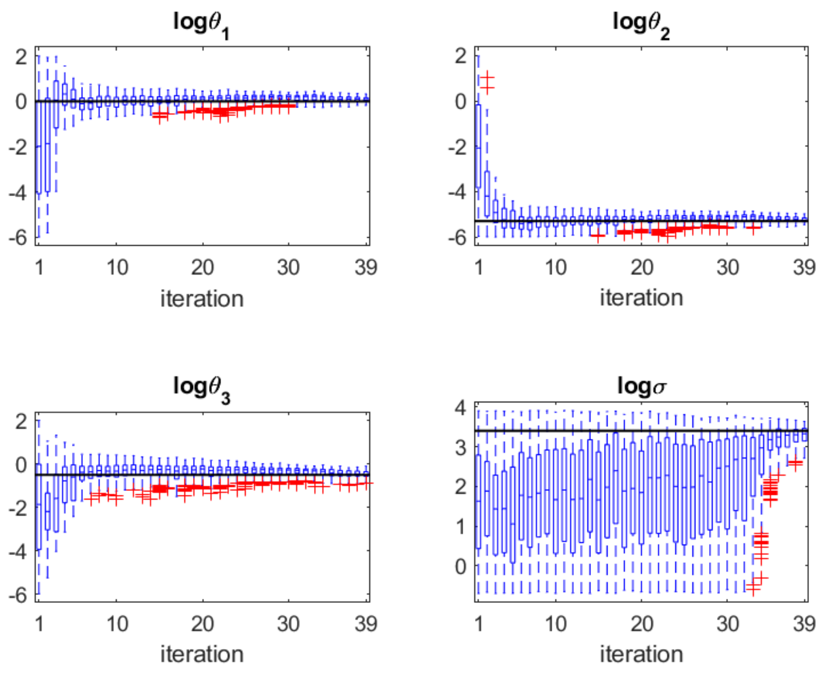


Figure 51: Lotka-Volterra: evolution of ABC posteriors using the "guided" `blockedopt` SMC-ABC across 39 rounds and 3,337,640 model simulations. Ground-truth parameters are marked with horizontal lines.

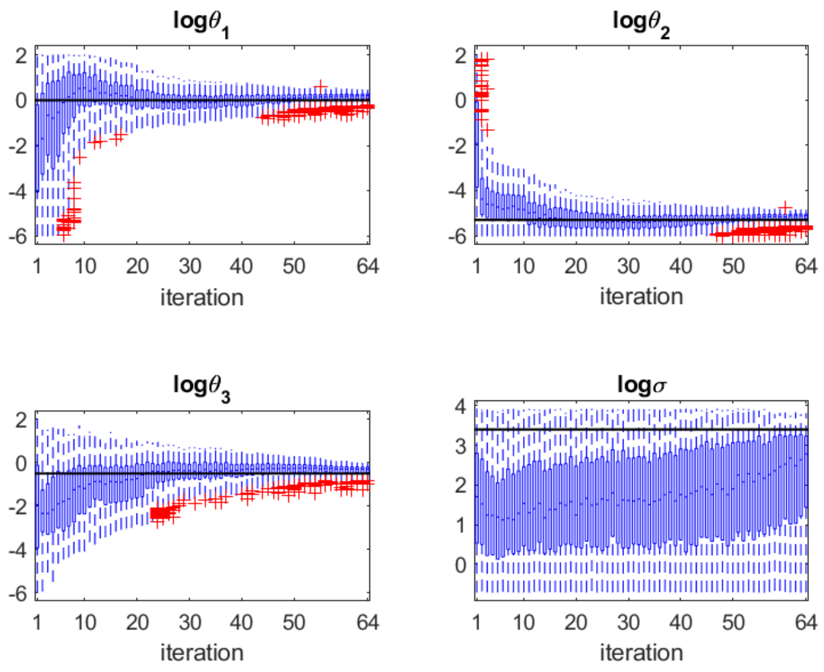


Figure 52: Lotka-Volterra: evolution of ABC posteriors using `standard` SMC-ABC across 64 rounds and 3,110,349 model simulations. Ground-truth parameters are marked with horizontal lines.

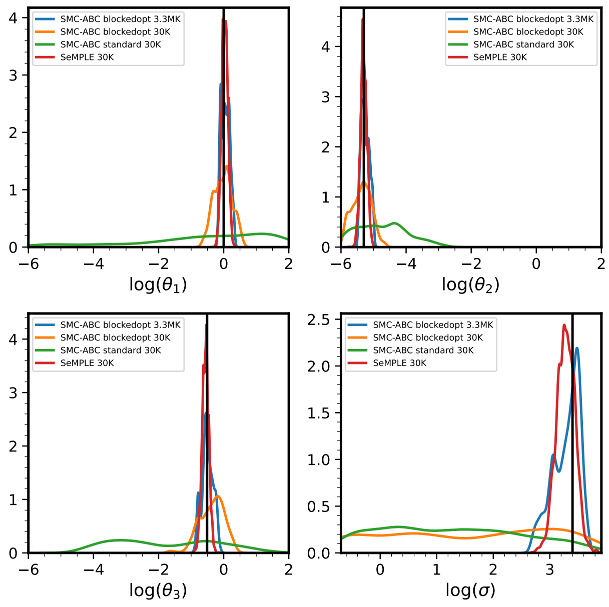


Figure 53: Lotka-Volterra: marginal posterior plots from SMC-ABC and SeMPLE. The SeMPLE posterior sample is based on 10,000 posterior draws, and was returned after a total of 30,000 model simulations. The `blockedopt` SMC-ABC posterior samples from round 10 and 39 used a total 32,865 and 3,337,640 model simulations respectively. The posterior samples from `standard` SMC-ABC used a total of 30,436 model simulations.

in terms of the noise parameter σ, suggesting that this is more difficult to infer with the limited budget of model simulations. However, when comparing with the SMC-ABC posterior distributions for σ, see Figure 53, the SeMPLE results are much more informative for the given model simulation budget of 30,000.

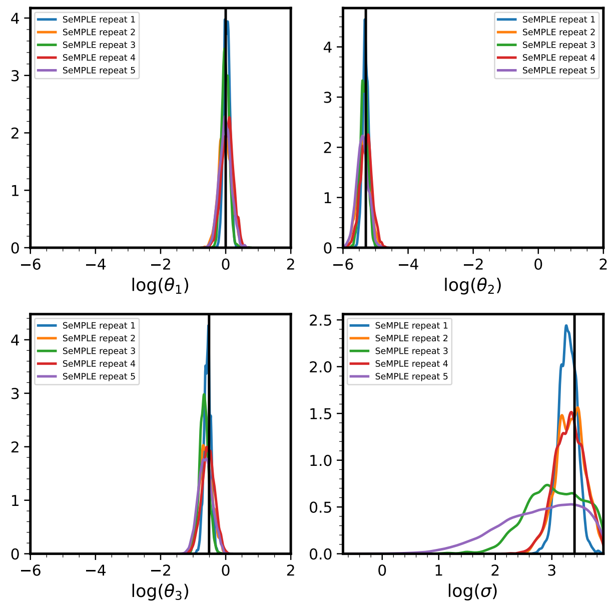


Figure 54: Lotka-Volterra: marginal posterior plots from 5 repeated runs of SeMPLE using the same observed data. The posterior sample is based on 10,000 posterior draws, and was returned after a total of 30,000 model simulations. The x-axes are adjusted to the support of the uniform prior of each parameter.

J Twisted prior model with 20 parameters

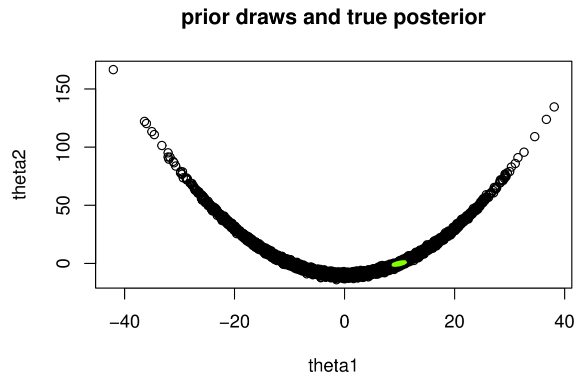
This case study is merely illustrative of the ability of SeMPLE to rapidly return sensible inference for a posterior with highly correlated components. Unlike with other examples, here we did not perform a full comparison with other methods, as we found it difficult to specify the non-standard prior used here to work within the framework provided by `SBIBM`.

J.1 Model definition

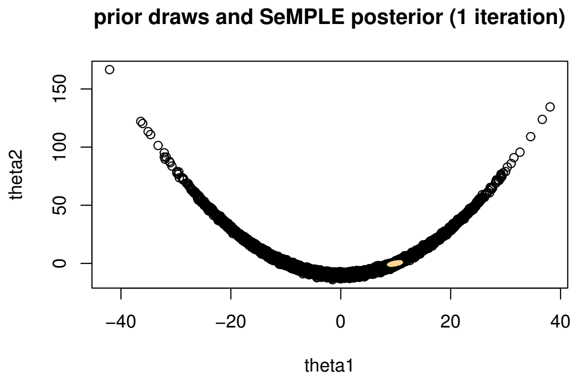
We now consider a model with 20 parameters and a challenging posterior characterised by a strong correlation between some of the parameters. This case study is particularly interesting, as the likelihood only provides location information about the unknown parameter while the dependence structure in the posterior comes mostly from the prior. For this reason, the posterior dependence changes direction depending on whether the likelihood locates the posterior in the left or right tail of the prior (see Nott et al., 2018 for a graphical illustration). This case study was analysed in an ABC context in Li et al. (2017) and Picchini & Tamborrino (2024). The model assumes $\boldsymbol{y} = (y_1, ..., y_{d_\theta})$ drawn from a d_θ-dimensional Gaussian $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\Psi})$, with $\boldsymbol{\theta} = (\theta_1, ..., \theta_{d_\theta})$ and diagonal covariance matrix $\Psi = \text{diag}(\sigma_0, ..., \sigma_0)$. The prior is the "twisted-normal" prior of Haario et al. (1999), with density function proportional to $p(\boldsymbol{\theta}) \propto \exp\Bigg\{-\theta_1^2/200 - (\theta_2 - b\theta_1^2 + 100b)^2/2 - \mathbb{1}_{\{d_\theta>2\}} \sum_{j=3}^{d_\theta} \theta_j^2/2\Bigg\}$, where $\mathbb{1}_B$ denotes the indicator function of the set B. This prior is essentially a product of independent Gaussian distributions with the exception that the component for (θ_1, θ_2) is modified to produce a "banana shape", with the strength of the bivariate dependence determined by the parameter b. Simulation from $p(\boldsymbol{\theta})$ is achieved by first drawing $\boldsymbol{\theta}$ from a d_θ-dimensional multivariate Gaussian as $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{A})$, where $\boldsymbol{A} = \text{diag}(100, 1, ..., 1)$, and then placing the value $\theta_2 + b\theta_1^2 - 100b$ in the slot for θ_2. We set $b = 0.1$, a value inducing a strong correlation in the prior between the first two components of $\boldsymbol{\theta}$. We consider $d_\theta = 20$, i.e. both $\boldsymbol{y}$ and $\boldsymbol{\theta}$ have length 20, and (same as Li et al., 2017; Picchini & Tamborrino, 2024) we set $\sigma_0 = 1$ as fixed and known, with observations given by the 20-dimensional vector $\boldsymbol{y}_o = (10, 0, ..., 0)$, where only the first entry is non-zero.

J.2 Results

We now show how a single round of SeMPLE ($r = 1$), hence without any MCMC, can in this case return excellent inference (notice, generally we recommend to run at least 2 rounds of SeMPLE, as MCMC is typically needed to ensure that samples from $r = 1$ do not "leak" outside the prior's support). Figure 55 compares the banana-shaped prior for the first two components of $\boldsymbol{\theta}$ with draws from the true posterior obtained via MCMC (panel (a)), and then with draws from a single round ($r = 1$) of SeMPLE initialised with $K = 10$ components (panel (b)). Here SeMPLE has been trained on 10,000 prior predictive simulations. Despite the true posterior occupies a tiny region within the prior's support, SeMPLE immediately identifies the bulk of the posterior in one round, that is Figure 55(b) shows samples via the learned GLLiM posterior model. Then, in Figure 56(a) we zoom on the posterior region to better show the quality of the SeMPLE approximation for both $r = 1$ and $r = 2$: see Figure 56(a) for (θ_1, θ_2) and Figure 56(b) for (θ_3, θ_4), the latter being an illustrative example of the non-correlated components (inference behaves the same for all the remaining components, hence further marginals are not reported). It appears that, for this model, the correction brought by MCMC at $r = 2$ is not necessary.

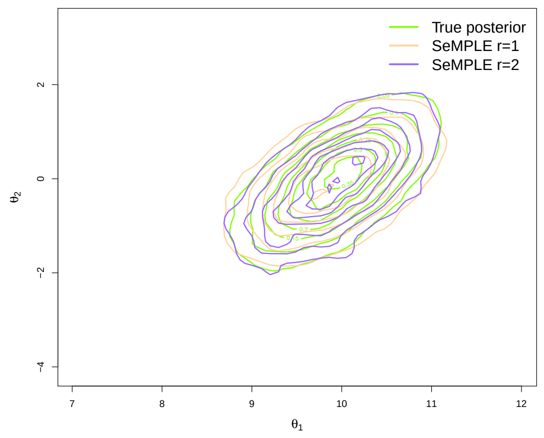


(a) Draws from prior $p(\theta_1, \theta_2)$ (black) and exact posterior (green)

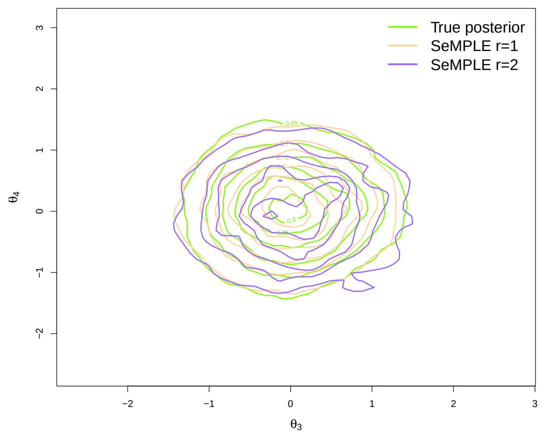


(b) Draws from prior $p(\theta_1, \theta_2)$ (black) and from SeMPLE (brown)

Figure 55: Twisted prior example: prior, true posterior and draws from SeMPLE at round $r = 1$.



(a) Posteriors for (θ_1, θ_2)



(b) Posteriors for (θ_3, θ_4)

Figure 56: Twisted prior example: (a) Contour plots of the exact posterior for (θ_1, θ_2) (green) and SeMPLE posterior at $r = 1$ (brown) and $r = 2$ (blue). (b) Same as (a) but for (θ_3, θ_4).

K Biological model of the translation kinetics after mRNA transfection

This model serves as an example of a multidimensional SDE model with a higher number of model parameters, compared to the Ornstein-Uhlenbeck model. Moreover here the model is challenging as it is partially observed (only one coordinate is observed), and observations incorporate measurement measurement error. Reference posterior samples cannot be obtained exactly, as the transition densities are not known for this model.

K.1 Model definition

We consider the SDE model from Pieschner et al. (2022) in a simulation study. In Pieschner et al. (2022) the model was used to describe translation kinetics after mRNA transfection. Specifically, in a real data scenario, time lapse microscopy images of the cells are taken for several hours, and for the first hour, the cells would be incubated with mRNA lipoplexes. Released mRNA gets translated into a green fluorescent protein (GFP) which causes the cells to fluoresce. The production of GFPs is the *translation process*. The SDE below is used to study the translation kinetics of one cell, based on one observed fluorescence trajectory, where the two-dimensional stochastic process $(m(t), p(t))$ has $m(t)$ to represent the amount of mRNA molecules at time t, and $p(t)$ is the amount of GFP molecules at time t. The SDE model is given by

$$d\begin{pmatrix} m \\ p \end{pmatrix}(t) = \begin{pmatrix} -\delta \cdot m(t) \\ k \cdot m(t) - \gamma \cdot p(t) \end{pmatrix} dt + \begin{pmatrix} \sqrt{\delta \cdot m(t)} & 0 \\ 0 & \sqrt{k \cdot m(t) + \gamma \cdot p(t)} \end{pmatrix} dB_t, \tag{30}$$

where dB_t is a two-dimensional standard Brownian motion. It is assumed that all mRNA molecules (within one cell) are released at once from the lipoplexes and denote this initial time point by t_0. Before t_0, there are neither mRNA nor GFP molecules, and at t_0, an amount m_0 of mRNA molecules is released, i.e

$$\begin{pmatrix} m(t) \\ p(t) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \text{for } t < t_0$$

while $m(t_0) = m_0$ and $p(0) = 0$.

We take as observable mapping

$$y(t_i) = \log(\text{scale} \cdot p(t_i) + \text{offset}) + \varepsilon(t_i) \qquad i = 1, ..., n,$$

where $\varepsilon(t_i)$ is iid Gaussian measurement error $\varepsilon(t_i) \sim \mathcal{N}(0, \sigma^2)$, and $t_1, ..., t_n$ are observational time instants (see further below). Note that the observations depend only indirectly on process $\{m(t)\}$, and only $\{p(t)\}$ is observed. The parameter vector we wish to infer is $\boldsymbol{\theta} = (\delta, \gamma, k, m_0, \text{scale}, t_0, \text{offset}, \sigma)$.

It has been shown by Pieschner et al. (2022) that SDE modelling improves identifiability of parameters, compared to using a corresponding ODE model. Nevertheless, some parameters are still unidentifiable, see in particular supplementary section A.4.1 in Pieschner et al. (2022).

Solutions to the SDE (30) are simulated using an Euler-Maruyama scheme implemented in `Rcpp` (Eddelbuettel et al., 2024) from $t = t_0$ to $t = 30$, with step size 0.01. The observed time series is interpolated from the Euler-Maruyama approximation at $n = 60$ equidistant time points from $t_1 = 0.5$ to $t_n = 30$. Note that this implies that observations can be made prior to $t = t_0$ where $p(t_i) = 0$, $t_i < t_0$.

Inference is performed on the parameters log-scale, and the prior distributions are all Gaussian as follows

$$\begin{aligned} \log \delta &\sim \mathcal{N}(-0.694, 0.5), & \log \gamma &\sim \mathcal{N}(-3, 0.5) \\ \log k &\sim \mathcal{N}(0.027, 0.5), & \log m_0 &\sim \mathcal{N}(5.704, 0.5) \\ \log \text{scale} &\sim \mathcal{N}(0.751, 0.5), & \log t_0 &\sim \mathcal{N}(-0.164, 0.5), \\ \log \text{offset} &\sim \mathcal{N}(2.079, 0.5), & \log \sigma &\sim \mathcal{N}(-2, 0.5). \end{aligned}$$

SeMPLE, SNL and SNPE-C were run for five different observed data sets, each generated with a different set of parameters given in Table 3. For each method, the total simulation budget was set to 30k model simulations that were uniformly distributed across the number of algorithm rounds. For SeMPLE, the number of algorithm rounds was set to $R = 3$, the number of starting mixture components was set to $K = 10$, and the covariance matrices in GLLiM were set to be full unconstrained matrices. We did not inflate the covariance in the Metropolis-Hastings step ($\gamma = 1$). The mixture probability threshold to remove negligible mixture components was set to 0.005. For SNPE-C and SNL the number of algorithm rounds was set to 10.

	$\log \delta$	$\log \gamma$	$\log k$	$\log m_0$	$\log \text{scale}$	$\log t_0$	$\log \text{offset}$	$\log \sigma$
Data set 1	-0.694	-3	0.027	5.704	0.751	-0.164	2.079	-2
Data set 2	-1.827	-3.138	-0.426	6.869	0.697	0.465	2.787	-1.993
Data set 3	-1.153	-3.501	-0.181	4.601	0.386	-2.121	2.185	-2.578
Data set 4	-1.297	-2.134	1.118	5.577	0.938	0.12	1.122	-2.854
Data set 5	-0.243	-4.596	0.168	5.573	-0.086	1.234	1.392	-1.655

Table 3: Translation kinetics model: parameter values used to generate observations for the five data sets.

K.2 Results

Figures 57-61 compare the marginal posteriors obtained by SeMPLE, SNPE-C and SNL, where each figure corresponds to a different dataset. All results are obtained after 30k model simulations but the number of algorithm rounds is 10 for both SNPE-C and SNL, while we ran 3 rounds for SeMPLE. Inference is very similar across the three methods, however it is clear that only SeMPLE can identify t_0 accurately, while SNL and SNPE-C provide a biased estimation of t_0 (an exception being the inference for dataset 3 in Figure 59, where we deliberately challenged the inference by setting the ground truth value for t_0 far in the left tail of the prior). Moreover, SNPE-C is often unable to infer the measurement error variability σ (datasets 2, 4, 5). As previously found in Pieschner et al. (2022), the lack of identifiability of parameters (k, m_0, scale) is confirmed in our analyses. However the other parameters are correctly inferred by SeMPLE. Finally, in Figures 62-66 we focus on displaying the performance of SeMPLE for each round.

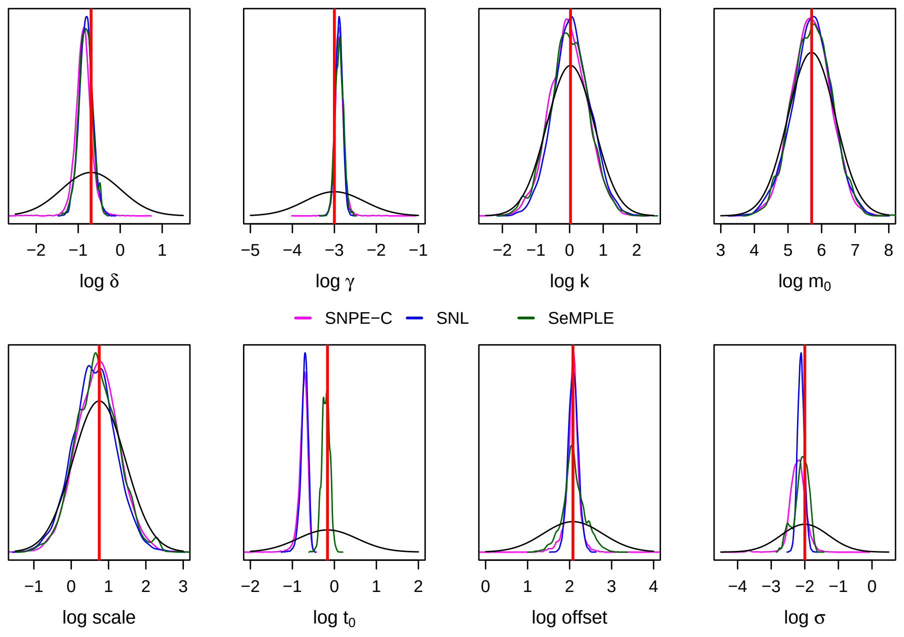


Figure 57: Translation kinetics model, observed data set #1 out of 5: marginal posteriors from SeMPLE, SNPE-C and SNL (after 30k model simulations), priors (black lines) and ground truth parameters (vertical red lines). For SNPE-C and SNL we show results after 10 algorithm rounds, and after 3 rounds for SeMPLE.

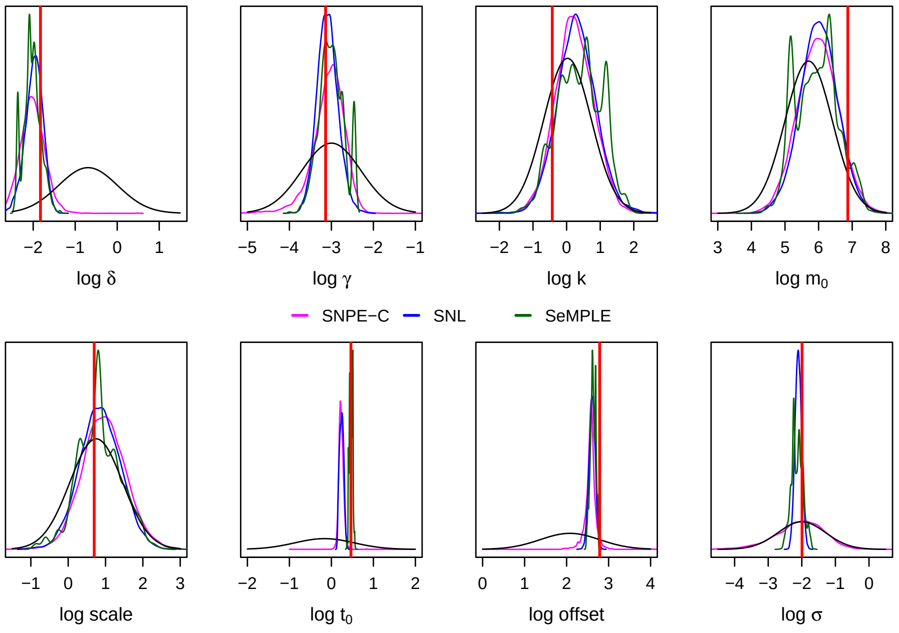


Figure 58: Translation kinetics model, observed data set #2 out of 5: marginal posteriors from SeMPLE, SNPE-C and SNL (after 30k model simulations), priors (black lines) and ground truth parameters (vertical red lines). For SNPE-C and SNL we show results after 10 algorithm rounds, and after 3 rounds for SeMPLE.

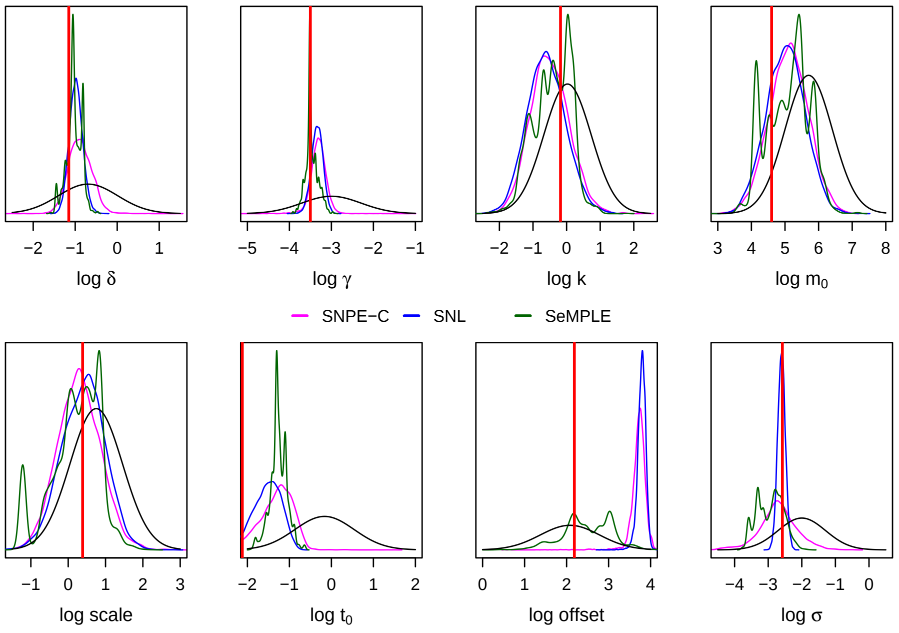


Figure 59: Translation kinetics model, observed data set #3 out of 5: marginal posteriors from SeMPLE, SNPE-C and SNL (after 30k model simulations), priors (black lines) and ground truth parameters (vertical red lines). For SNPE-C and SNL we show results after 10 algorithm rounds, and after 3 rounds for SeMPLE.

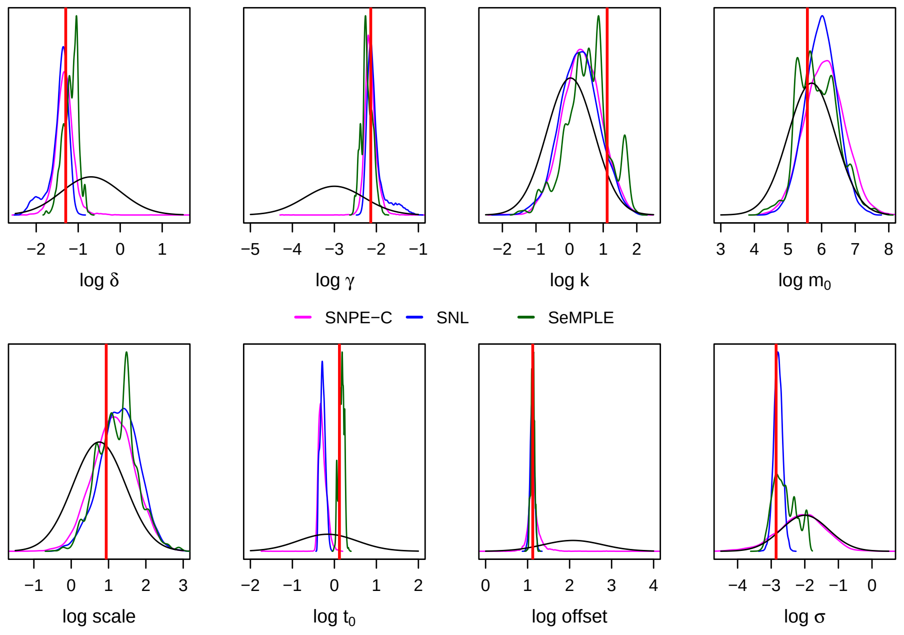


Figure 60: Translation kinetics model, observed data set #4 out of 5: marginal posteriors from SeMPLE, SNPE-C and SNL (after 30k model simulations), priors (black lines) and ground truth parameters (vertical red lines). For SNPE-C and SNL we show results after 10 algorithm rounds, and after 3 rounds for SeMPLE.

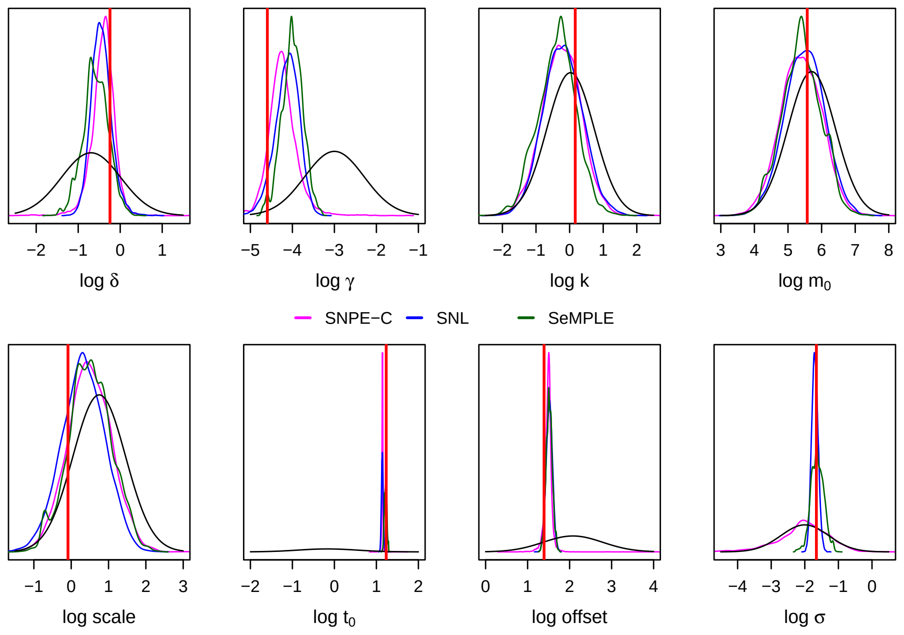


Figure 61: Translation kinetics model, observed data set #5 out of 5: marginal posteriors from SeMPLE, SNPE-C and SNL (after 30k model simulations), priors (black lines) and ground truth parameters (vertical red lines). For SNPE-C and SNL we show results after 10 algorithm rounds, and after 3 rounds for SeMPLE.

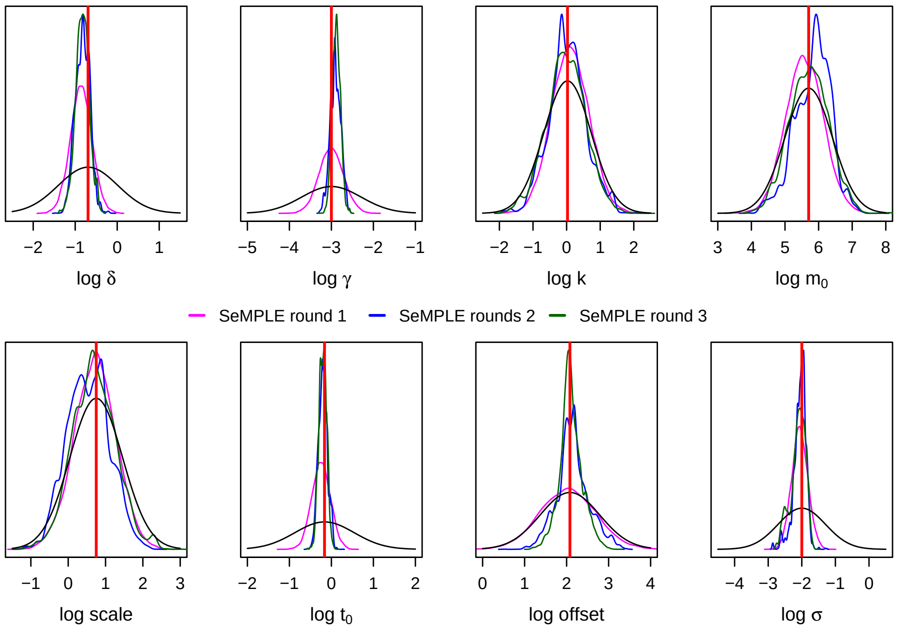


Figure 62: Translation kinetics model, observed data set #1 out of 5: marginal posteriors from SeMPLE for every algorithm round (corresponding to 10k, 20k and 30k model simulations), priors (black lines) and ground truth parameters (vertical red lines).

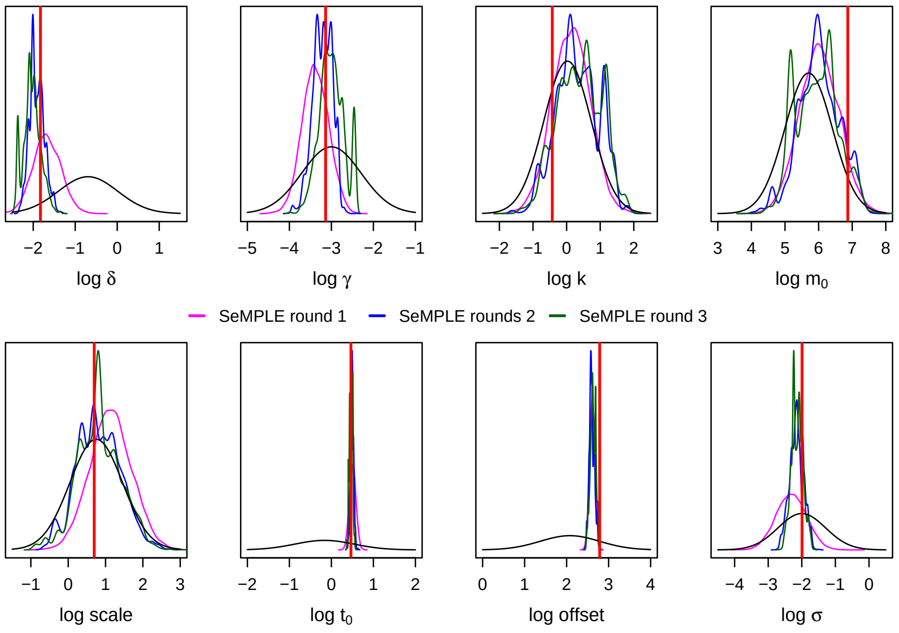


Figure 63: Translation kinetics model, observed data set #2 out of 5: marginal posteriors from SeMPLE for every algorithm round (corresponding to 10k, 20k and 30k model simulations), priors (black lines) and ground truth parameters (vertical red lines).

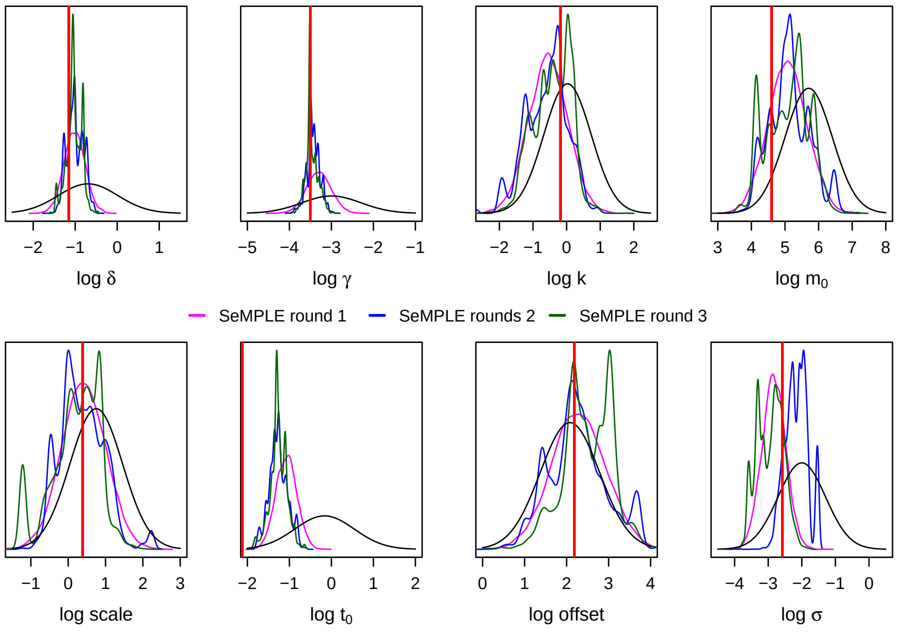


Figure 64: Translation kinetics model, observed data set #3 out of 5: marginal posteriors from SeMPLE for every algorithm round (corresponding to 10k, 20k and 30k model simulations), priors (black lines) and ground truth parameters (vertical red lines).

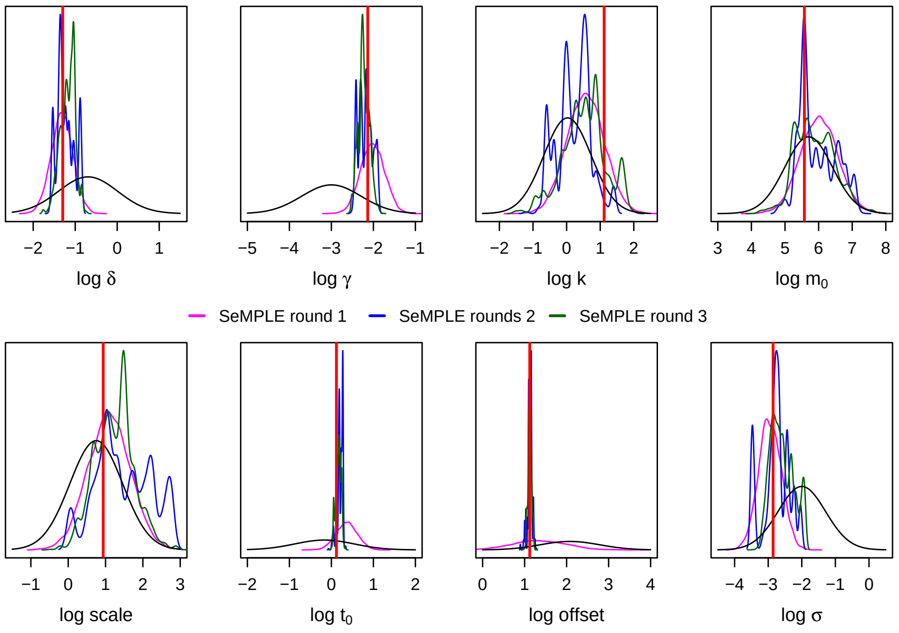


Figure 65: Translation kinetics model, observed data set #4 out of 5: marginal posteriors from SeMPLE for every algorithm round (corresponding to 10k, 20k and 30k model simulations), priors (black lines) and ground truth parameters (vertical red lines).

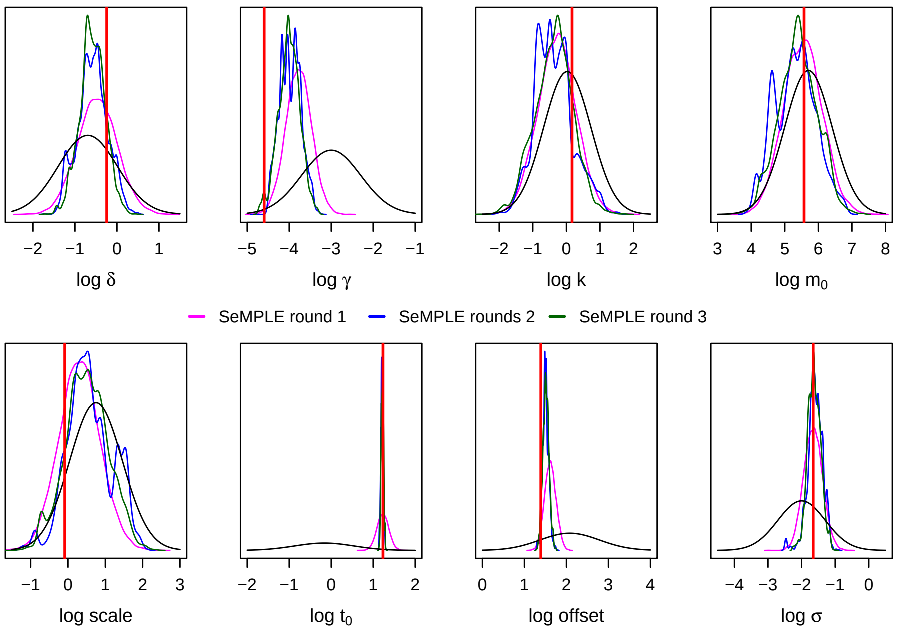


Figure 66: Translation kinetics model, observed data set #5 out of 5: marginal posteriors from SeMPLE for every algorithm round (corresponding to 10k, 20k and 30k model simulations), priors (black lines) and ground truth parameters (vertical red lines).