Diffusion posterior sampling for simulation-based inference in tall data settings

Anonymous Authors[1]

Abstract

Determining which parameters of a non-linear model could best describe a set of experimental data is a fundamental problem in science and it has gained much traction lately with the rise of complex large-scale simulators (a.k.a. *black-box* simulators). The likelihood of such models is typically intractable, which is why classical MCMC methods can not be used. Simulation-based inference (SBI) stands out in this context by only requiring a dataset of simulations to train deep generative models capable of approximating the posterior distribution that relates input parameters to a given observation. In this work, we consider a *tall data* extension in which multiple observations are available and one wishes to leverage their shared information to better infer the parameters of the model. The method we propose is built upon recent developments from the flourishing score-based diffusion literature and allows us to estimate the *tall data* posterior distribution simply using information from the score network trained on individual observations. We compare our method to recently proposed competing approaches on various numerical experiments and demonstrate its superiority in terms of numerical stability and computational cost.

1. Introduction

Inverting non-linear models describing natural phenomena is a fundamental problem in science and has been tackled by many fields (Gonçalves et al., 2020; Vasist et al., 2023; Dax et al., 2023). In this work, we consider a Bayesian approach (Tarantola, 2005) in which the input parameters $\theta \in \mathbb{R}^m$ of a model $\mathcal{M}$ that best explain a set of output observations $x \in \mathbb{R}^d$ are described via a posterior distribution $p(\theta \mid x)$. Obtaining samples of the posterior can, however, be very challenging when the outputs of $\mathcal{M}$ are obtained through complex simulations (e.g. solutions of non-linear differential equations (Jansen & Rit, 1995)). Indeed, in these cases the likelihood function $p(x \mid \theta)$ is often impossible to evaluate and MCMC procedures can not be used.

Simulation-based inference (SBI) (Cranmer et al., 2020) is a promising approach that bypasses the difficulties of likelihood evaluations via simulations from the model and leverages the recent advances from deep generative learning. The procedure relies on the choice of a prior distribution $\lambda(\theta)$ encoding knowledge about the values of $\mathcal{M}$'s parameters and a simulated dataset generated as per:

$$\Theta_i \sim \lambda(\theta)\,, \quad X_i \sim p(x \mid \Theta_i)\,, \quad (\Theta_i, X_i) \sim p(\theta, x)\,.$$

Letting N_s be a fixed simulation budget, one can build a dataset $\mathcal{D} = \{(\Theta_i, X_i)\}_{i=1}^{N_s}$ and train conditional deep generative models to generate samples from $p(\theta \mid x^\star)$ for any new observation $x^\star$. Usual approaches approximate either directly the posterior distribution (NPE) (Greenberg et al., 2019), the likelihood function (NLE) (Papamakarios et al., 2019), or the ratio of likelihoods (NRE) (Hermans et al., 2020). In NPE, the posterior samples can be obtained directly by sampling a conditional normalizing flow (Papamakarios et al., 2021), whereas in NLE and NRE it is necessary to use a MCMC sampler.

In this work, we consider a natural extension of the above Bayesian framework to a *tall data* setting (Bardenet et al., 2015), in which multiple observations $x^\star_{1:n} = (x_1, \dots, x_n)$ are available and the posterior distribution

$$p(\theta \mid x^\star_{1:n}) \propto \lambda(\theta)^{1-n} \prod_{j=1}^{n} p(\theta \mid x^\star_j)\,, \tag{1}$$

is expected to provide more precise information about how to invert $\mathcal{M}$ (see Fig 1). Despite being a setup with much practical interest, there has not been many previous works providing a satisfactory extension to usual SBI procedures. In Rodrigues et al. (2021), the authors merge a fixed number of extra observations via a deepset (Zaheer et al., 2017) and fall back to NPE with an augmented training dataset with samples $\mathcal{D}_n = \{(\Theta_i, X_{i,1:n})\}$. Important drawbacks of such approach are the lack of flexibility on the number of extra observations at inference time and the potentially

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

heavy cost of generating extra observations from the simulator. In Hermans et al. (2020), the authors show how to handle the *tall data* setting with NRE and an equivalent extension can be done for NLE as well (Geffner et al., 2023). Note, however, that both approaches still require a MCMC sampler to obtain posterior samples, which is often undesirable in SBI applications.

Recently, Sharrock et al. (2022) proposed to use score-based generative models (SBGM) (Ho et al., 2020; Song et al., 2021b) to approximate the conditional posterior distribution targeted in SBI, called *Neural Posterior Score estimation* (NPSE). SBGM introduces an easy-to-train objective for learning the *score* of a sequence of perturbed versions of the target distribution. This relies on a network $s_\phi(\theta, x_i, t)$ trained using the simulated dataset. SBGM rivals state-of-the-art generative models such as generative adversarial networks (GANs) (Goodfellow et al., 2014) and normalizing flows in high-dimension challenging datasets without the need of adversarial training or special network architectures.

However, extending NPSE to a *tall data* setting is challenging, as a direct application of the method would require training a score network $s_\phi(\theta, x_{1:n}, t)$ to approximate $\nabla_\theta \log p_t(\theta \mid x_{1:n})$ on an augmented dataset $\mathcal{D}_n$, leading to the same drawbacks from the augmented NPE approach. Recently, Geffner et al. (2023) proposed a method named F-NPSE in which an annealed Langevin procedure is used to approximate the target tall posterior using a sequence of probability distributions from which the score can be computed using the individual scores $s_\phi(\theta, x_i, t)$, thus avoiding the requirement of an augmented training dataset. The major drawback of this approach, however, is reintroducing the need of MCMC for sampling from the posterior.

In this paper, we propose an algorithm that approximates the score of the SBGM associated with the *tall data* posteriors using only the individual scores from the individual SBGMs and without a Langevin sampler. We demonstrate the superiority of our approach as compared to F-NSPE on several numerical experiments: two Gaussian toy models for which all quantities of interest are known analytically, three examples from the SBI benchmark (Lueckmann et al., 2021a), and the inversion of a challenging model from neuroscience (Jansen & Rit, 1995).

2. Background

2.1. Score based generative models (SBGM)

Score estimation. The goal of score-based generative modelling is to learn how to sample new data from a m-dimensional target distribution q_{data} using an approximation of the score on "noisy versions" of samples from q_{data}. In the variance preserving (VP) framework (Ho et al., 2020; Yang et al., 2023), a sequence of noisy versions $\{\theta_t\}_{t\in[1:T]}$ of q_{data} is defined for $t \in [1:T]$, by

$$\theta_t = \sqrt{\alpha_t}\theta_0 + \sqrt{1-\alpha_t}\epsilon_t\,,$$

where $\theta_0 \sim q_{\text{data}}$, $\{\epsilon_t\}_{t\in[1:T]}$ are i.i.d. with distribution $\mathcal{N}(0, \mathbf{I}_m)$, $\{\alpha_t\}_{t\in[1:T]} \in [0,1]^T$ is a decreasing sequence of positive real numbers. Let $q_{t|0}(\theta_t|\theta_0) = \mathcal{N}(\theta_t; \sqrt{\alpha_t}\theta_0, (1-\alpha_t)\mathbf{I}_m)$ be the probability density function of the conditional distribution of θ_t given θ_0 and q_t the density of θ_t. Following the seminal works of Hyvärinen & Dayan (2005); Vincent (2011), we can learn the score of q_t via a neural network s_ψ by minimising $\mathcal{L} : \psi \mapsto \mathbb{E}_{\theta_t\sim\mathsf{q}_t}\left[\|\mathrm{s}_\psi(\theta_t, \alpha_t) - \nabla \log \mathsf{q}_t(\theta_t)\|^2\right]$ without knowledge of the ground-truth data score. Using Fisher's identity, we can define the SBGM loss as

$$\mathcal{L}_{\text{score}}(\psi) = \sum_{t=1}^{T} \gamma_t^2 \mathbb{E}_{\theta_0\sim q_{\text{data}}, \theta_t\sim q_{t|0}(\cdot|\theta_0)}\Big[\|\mathrm{s}_\psi(\theta_t, \alpha_t) - \nabla \log q_{t|0}(\theta_t|\theta_0)\|^2\Big]\,, \quad (2)$$

where γ_t is a positive weighting function introduced in Vincent (2011).

Backward sampling. Once we have the score approximation $\mathrm{s}_\psi(\theta_t, \alpha_t)$ of $\nabla \log \mathsf{q}_t(\theta_t)$, our goal is to sample *backwards* from the distributions $\mathsf{q}_T, \ldots, \mathsf{q}_1$. There are many ways of carrying out this sampling, such as using annealed Langevin dynamics (Song & Ermon, 2019), stochastic differential equations (Song et al., 2021b), or ordinary differential equations (Karras et al., 2022). In this work, we follow the approach proposed in Song et al. (2021a), which yields the denoising diffusion implicit models (DDIM) sampler. DDIM introduces a set inference distributions indexed by $\sigma = \{\sigma_t \in (0, \alpha_{t-1}^{1/2})\}_{t\in[1:T-1]}$ defined for $t \in [1:T-1]$ as:

$$q^\sigma_{t|t+1,0}(\theta_t|\theta_0, \theta_{t+1}) = \mathcal{N}\left(\theta_{t-1}; \boldsymbol{\mu}_t(\theta_0, \theta_t), \sigma_t^2 \mathbf{I}_m\right)\,,$$

with $\boldsymbol{\mu}_t(\theta_0, \theta_t) = {\alpha_{t-1}}^{1/2}\theta_0 + (\upsilon_{t-1} - \sigma_t^2)^{1/2}(\theta_t - {\alpha_t}^{1/2}\theta_0)/(1-\alpha_t)^{1/2}$. Note that $\boldsymbol{\mu}_t$ is chosen so that $q^\sigma_{t|0}(\theta_t|\theta_0) = \mathcal{N}(\theta_T; \sqrt{\alpha_t}\theta_0, (1-\alpha_t)\mathbf{I}_m)$ (Song et al., 2021a, Lemma 1, Appendix B). This property allows us to write

$$\mathsf{q}_{t-1}(\theta_{t-1}) = \int q^\sigma_{t-1|t,0}(\theta_{t-1}|\theta_t, \theta_0) \times q_{0|t}(\theta_0|\theta_t)\mathsf{q}_t(\theta_t)\mathrm{d}\theta_0\mathrm{d}\theta_t\,.$$

Even though the equation above suggests a way of passing from q_t to q_{t-1} it involves an intractable kernel $\int q^\sigma_{t-1|t,0}(\theta_{t-1}|\theta_t, \theta_0) q_{0|t}(\theta_0|\theta_t)\mathrm{d}\theta_0$. The marginal distribution can be approximated by

$$\hat{\mathsf{q}}_{t-1}(\theta_{t-1}) = \int q^\sigma_{t-1|t,0}(\theta_{t-1}|\theta_t, \mu_t(\theta_t))\mathsf{q}_t(\theta_t)\mathrm{d}\theta_t\,, \quad (3)$$

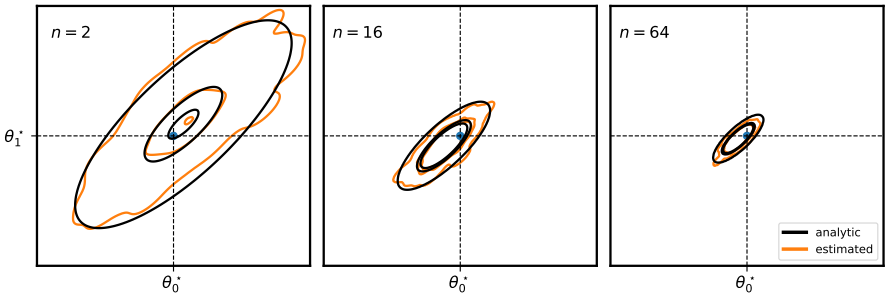


Figure 1. The posterior distribution of a model with a Gaussian simulator and Gaussian prior concentrates around the **true parameter** $\theta^\star$ as the number n of observations $x_i \sim p(x \mid \theta^\star)$ increases. In this simple setting we can calculate the **analytic** posterior distribution and compare it to the posterior **estimated** with our score-based proposal (**GAUSS**).

where $\mu_t(\theta_t)$ is the conditional mean of θ_0 given θ_t under the inference distribution; i.e $\mu_t(\theta_t) := \mathbb{E}_{\theta_0 \sim q^\sigma_{0|t}(\cdot|\theta_t)}[\theta_0]$. As $\mu_t(\theta_t)$ is not available explicitly, in order to effectively sample from (3), we can estimate $\mu_t(\theta_t)$ using the approximate score $\mathrm{s}_\psi(\theta_t, t)$. By noting that

$$\alpha_t^{1/2}\mu_t(\theta_t) - \theta_t = (1-\alpha_t)\mathbb{E}_{\theta_0 \sim q^\sigma_{0|t}(\cdot|\theta_t)}\left[\nabla \log q_{t|0}(\theta_t|\theta_0)\right] ,$$

we can use (2) to define the following approximation for all $t \in [1:T]$

$$\mu_{\psi,t}(\theta_t) := \alpha_t^{1/2}\Big(\theta_t + (1-\alpha_t)\mathrm{s}_\psi(\theta_t, \alpha_t)\Big) . \tag{4}$$

We can finally define the backward Markov chain

$$p_{\psi,0:T}(\theta_{0:T}) = p_T(\theta_T)\prod_{t=1}^{T} p_{\psi,t-1|t}(\theta_{t-1}|\theta_t) , \tag{5}$$

where $p_T(\theta_T) = \mathrm{q}_T(\theta_T)$, $p_{\psi,t-1|t}(\theta_{t-1}|\theta_t) = q^\sigma_{t-1|t,0}(\theta_{t-1}|\theta_t, \mu_{\psi,t}(\theta_t))$ and $p_{\psi,0|1}(\theta_0|\theta_1) = \mathcal{N}(\theta_0; \mu_{\psi,1}(\theta_1), \sigma_0^2\mathbf{I}_m)$, with $\sigma_0 > 0$ a free parameter.

Note that we use the procedure described above to learn the score of the conditional distribution of θ given x. To do so, we learn an ammortized score $\mathrm{s}_\psi(\theta, x, \alpha_t)$ as in Batzolis et al. (2021).

2.2. Factorized neural posterior score estimation

The F-NPSE method proposed in Geffner et al. (2023) defines a sequence of distributions $\{\varrho_t(\cdot \mid x)\}_{t\in[0:T]}$ as per

$$\nabla_\theta \log \varrho_t(\theta \mid x^\star_{1:n}) = (1-n)(1-t)\nabla_\theta \log \lambda(\theta) + \sum_{j=1}^{n} \mathrm{s}_\psi(\theta, x_i, \alpha_t) , \tag{6}$$

where the score of ϱ_0 is the score of the tall posterior defined in (1) and $\nabla_\theta \log \lambda(\theta)$ can be computed analytically for common prior choices; see Appendix D. F-NPSE uses Langevin dynamics to sample ϱ_t for each $t = T, \dots, 1$ and has shown superior performance as compared to all competing methods for *tall data* settings (i.e. augmented versions of NPE, NLE, and NRE) in terms of posterior reconstruction, achieving the best trade-off between sample efficiency and accumulation of errors when the number of observations grows. However, this performance is at the cost of an increased number of neural net evaluations. The main limitation of F-NPSE is the use of annealed Langevin dynamics which is very sensitive to the choices of step-size at each sampling iteration, as well as the total number of steps.

3. Diffusion posterior sampling for tall data

3.1. Exact computation of the tall data posterior score

Let $x^\star_{1:n} = \{x^\star_1, \dots, x^\star_n\} \sim_{\text{iid}} p(x \mid \theta^\star)$ be observations sampled using the same parameter $\theta^\star \in \mathbb{R}^m$. Our goal is to sample from the *tall data* posterior $p(\theta \mid x^\star_{1:n})$ via the backward sampling Markov chain defined in (5), while only relying on a score estimate $s_\phi(\theta, x, \alpha_t)$ of $\nabla_\theta \log p_t(\theta \mid x)$ of the diffused posterior for a single observation. To do so, we need to build an approximation of the diffused *tall data* posterior score

$$\nabla_{\theta_t} \log p_t(\theta_t \mid x^\star_{1:n}) = \nabla_{\theta_t} \log \int p(\theta \mid x^\star_{1:n}) q_{t|0}(\theta_t|\theta)\mathrm{d}\theta$$

that solely depends on $\nabla_\theta \log p_t(\theta \mid x^\star_j)$, for all $j \in [1, n]$. Using (1) we can write

$$p_t(\theta \mid x^\star_{1:n}) = \int p(\theta_0 \mid x^\star_{1:n}) q_{t|0}(\theta|\theta_0)\mathrm{d}\theta_0$$
$$\propto \int \left(\lambda(\theta_0)^{1-n}\prod_{j=1}^{n} p(\theta_0 \mid x^\star_j)\right) q_{t|0}(\theta|\theta_0)\mathrm{d}\theta_0 . \tag{7}$$

Given the diffused prior $p^\lambda_t(\theta) = \int \lambda(\theta_0) q_{t|0}(\theta|\theta_0)\mathrm{d}\theta_0$[1], we now introduce the following backward transition kernels obtained via Bayes' rule:

$$p_{0|t}(\theta_0|\theta, x) = \frac{p(\theta_0 \mid x) q_{t|0}(\theta|\theta_0)}{p_t(\theta \mid x)} \tag{8}$$

$$\text{and} \quad p^\lambda_{0|t}(\theta_0|\theta) = \frac{\lambda(\theta_0) q_{t|0}(\theta|\theta_0)}{p^\lambda_t(\theta)} . \tag{9}$$

[1]The diffused prior can be computed analytically for simple (e.g. uniform or Gaussian) prior distributions as in Sharrock et al. (2022).

Rearranging terms in equation (7) and replacing them with the quantities in (8), gives us

$$
\begin{aligned}
&p_t(\theta \mid x^\star_{1:n}) \\
&\quad \propto \int \left(\lambda(\theta_0) q_{t|0}(\theta|\theta_0)\right)^{1-n} \prod_{j=1}^n p(\theta_0 \mid x_j^\star) q_{t|0}(\theta|\theta_0) \mathrm{d}\theta_0 \\
&\quad = \int \left(p^\lambda_{0|t}(\theta_0|\theta) p_t^\lambda(\theta)\right)^{1-n} \\
&\qquad\qquad \times \prod_{j=1}^n p_{0|t}(\theta_0|\theta, x_j^\star) p_t(\theta \mid x_j^\star) \mathrm{d}\theta_0 \\
&\quad = \left(p_t^\lambda(\theta)^{1-n} \prod_{j=1}^n p_t(\theta \mid x_j^\star) \right) L_\lambda(\theta, x^\star_{1:n}) \,,
\end{aligned}
$$

with

$$
L_\lambda(\theta, x^\star_{1:n}) := \int p^\lambda_{0|t}(\theta_0|\theta)^{1-n} \prod_{j=1}^n p_{0|t}(\theta_0|\theta, x_j^\star) \mathrm{d}\theta_0 \,.
$$

The corresponding Fisher score therefore writes

$$
\begin{aligned}
\nabla_\theta \log p_t(\theta \mid x^\star_{1:n}) &= (1-n) \nabla_\theta \log p_t^\lambda(\theta) \\
&+ \sum_{j=1}^n \nabla_\theta \log p_t(\theta \mid x_j^\star) + \nabla_\theta \log L_\lambda(\theta, x^\star_{1:n}) \,.
\end{aligned}
$$

The first two terms in the above equation are known: the prior score can be computed analytically in most cases[2] and the (single) posterior scores are approximated by evaluating the learned score model $s_\phi(\theta, x, \alpha_t)$ in every $x_j^\star$. This leaves us with the last term, that we still need to approximate.

3.2. Second order approximation of $\log L_\lambda$

In this section, we consider a Gaussian approximation of the backward kernels defined in (8) via

$$
\hat{p}_{0|t}(\theta_0|\theta, x) = \mathcal{N}(\theta_0; \mu_t(\theta, x), \Sigma_t(\theta, x)) \tag{10}
$$

$$
\text{and} \quad \hat{p}^\lambda_{0|t}(\theta_0|\theta) = \mathcal{N}(\theta_0; \mu_{t,\lambda}(\theta), \Sigma_{t,\lambda}(\theta)) \,, \tag{11}
$$

where $\mu_t(\theta, x)$, $\mu_{t,\lambda}(\theta)$ and $\Sigma_t(\theta, x)$, $\Sigma_{t,\lambda}(\theta)$ are the means and covariance matrices of the backward processes respectively associated with the diffused posterior $p_t(\theta \mid x)$ and prior $p_t^\lambda(\theta)$ distributions.[3] This leads us to the following estimator of $\log L_\lambda(\theta, x^\star_{1:n})$:

$$
\ell_\lambda(\theta, x^\star_{1:n}) = \log \int \hat{p}^\lambda_{0|t}(\theta_0|\theta)^{1-n} \prod_{j=1}^n \hat{p}_{0|t}(\theta_0|\theta, x_j) \mathrm{d}\theta_0 \,. \tag{12}
$$

From now on, we alleviate the dependency on x_j and write $\mu_{t,j}(\theta) = \mu_{t,j}(\theta, x_j)$ and $\Sigma_{t,j}(\theta) = \Sigma_t(\theta, x_j)$. We now state the two Lemmas that are the foundation of our approximation of the score of the tall posterior. All proofs are postponed to Appendix A.

Lemma 3.1. *For all $\theta \in \mathbb{R}^m$, let $\Lambda(\theta) = \sum_{j=1}^n \Sigma_{t,j}^{-1}(\theta) + (1-n)\Sigma_{t,\lambda}^{-1}(\theta)$. Assume that $\Lambda(\theta)$ is positive definite. Then, for all $\theta \in \mathbb{R}^m$ and $x_i^\star \in \mathbb{R}^d$, $1 \le i \le n$,*

$$
\ell_\lambda(\theta, x^\star_{1:n}) = \sum_{j=1}^n \zeta_j(\theta) + (1-n)\zeta_\lambda(\theta) - \zeta_{\text{all}}(\theta) \,, \tag{13}
$$

where

$$
\zeta(\mu, \Sigma) = -\left(m \log 2\pi - \log|\Sigma^{-1}| + \mu^\top \Sigma^{-1} \mu\right)/2 \,,
$$

and $\zeta_j(\theta) = \zeta(\mu_{t,j}(\theta), \Sigma_{t,j}(\theta))$ for all $j \in [1:n]$, $\zeta_\lambda(\theta) = \zeta(\mu_{t,\lambda}(\theta), \Sigma_{t,\lambda}(\theta))$ and $\zeta_{\text{all}}(\theta) = \zeta(\Lambda(\theta)^{-1}\eta(\theta), \Lambda(\theta)^{-1})$ with $\eta(\theta) = \sum_{j=1}^n \Sigma_{t,j}^{-1}(\theta)\mu_{t,j}(\theta) + (1-n)\Sigma_{t,\lambda}^{-1}(\theta)\mu_{t,\lambda}(\theta)$.

Lemma 3.2. *For all $\theta \in \mathbb{R}^m$ and $x_i^\star \in \mathbb{R}^d$, $1 \le i \le n$, the full gradient can be written as*

$$
\begin{aligned}
\nabla_\theta \log p_t(\theta \mid x^\star_{1:n}) &= \Lambda(\theta)^{-1} \sum_{j=1}^n \Sigma_{t,j}^{-1}(\theta) \nabla_\theta \log p_t(\theta \mid x_j) \\
&+ (1-n)\Lambda(\theta)^{-1}\Sigma_{t,\lambda}^{-1}(\theta) \nabla_\theta \log p_t^\lambda(\theta) + F(\theta, x^\star_{1:n}) \,,
\end{aligned}
$$

where $F(\theta, x_{1:n}) = 0$ if $\nabla_\theta \Sigma_{t,j}(\theta) = 0$ for all $j \in [1:n]$ and $\nabla_\theta \Sigma_{\lambda,t}(\theta) = 0$.

We now focus on the particular choice of $\Sigma_{t,i}(\theta)$ that we use in our approximation. It can be shown that $\Sigma_{t,i}(\theta_t) = \nabla_{\theta_t} \mu_{t,i}(\theta_t)$ (Boys et al., 2023). This corresponds to Algorithm 2, which we refer to as JAC. However, this approach possesses two main drawbacks. The first is that it scales poorly with the dimension; indeed, we need to calculate a $m \times m$ matrix which is prohibitive for large m. The second is that we have to take the derivative w.r.t. the inputs of the score neural network, which is known to be unstable. Note that as proposed in Boys et al. (2023), we do not propagate gradients through $\Sigma_{t,i}(\theta_t)$, rendering $F(\theta_t, x_{1:n}) = 0$ in Lemma 3.2.

As an alternative, we consider a constant approximation of the covariance matrix. To do so, we start by noting that in the case where q_{data} is a Gaussian distribution with mean μ_0 and variance Σ_0, we have $\Sigma_t = (\Sigma_0^{-1} + \alpha_t(1-\alpha_t)^{-1}\mathbf{I}_m)^{-1}$, see Appendix D. Note that choosing $\Sigma_0 = \mathbf{I}_m$ results in the approximation proposed by Song et al. (2023). We use

[2] See Appendix D for the Gaussian and Uniform cases. If not, it can be learned via the classifier-free guidance approach (Ho & Salimans, 2021), at the same time as the posterior score (more details and experimental results are provided in Appendix K).

[3] Note that $\mu_t(\theta) = \mathbb{E}[\theta_0 \mid \theta]$ and $\Sigma_t(\theta) = \mathbb{E}\left[(\theta_0 - \mu_t(\theta))(\theta_0 - \mu_t(\theta))^\top \mid \theta\right]$. They are explicitly known for certain distribution choices and are respectively functions of the score and its derivatives; see Boys et al. (2023).

the formula for the Gaussian case as an approximation of the real covariance matrix, yielding Algorithm 1 which we refer to as GAUSS. To do so, we need to first estimate Σ_0 for each x, which we do by first running a DDIM with a small number of iterations (100) for each i and using the empirical covariance matrix of the samples as Σ_0. Note that in this case, we also have $F(\theta_t, x_{1:n}) = 0$ in Lemma 3.2.

Algorithm 1 GAUSS (Gaussian approximation)

Input: θ, $x_{1:n}$, t, $\hat{\Sigma}_{1:n}$, prior_fn
Output: $s_{1:n}$
$\Sigma_{t,\lambda}^{-1}, s_\lambda \leftarrow \text{prior\_fn}(\theta, t)$
for $j \leftarrow 1$ to n **do**
  $s_j \leftarrow \mathrm{s}_\psi(\theta, x_j, \alpha_t)$
  $\hat{\Sigma}_{t,j}^{-1} \leftarrow \hat{\Sigma}_j^{-1} + \frac{\alpha_t}{(1-\alpha_t)}\mathbf{I}_m$
end for
$\Lambda \leftarrow (1-n)\Sigma_{t,\lambda}^{-1} + \sum_{i=1}^n \hat{\Sigma}_{t,i}^{-1}$
$\tilde{s}_{1:n} \leftarrow (1-n)\Sigma_{t,\lambda}^{-1} s_\lambda + \sum_{i=1}^n \hat{\Sigma}_{t,i}^{-1} s_i$
$s_{1:n} \leftarrow \text{LinSolve}(\Lambda, \tilde{s}_{1:n})$

Algorithm 2 JAC (Jacobian approximation)

Input: θ, $x_{1:n}$, t, prior_fn
Output: $s_{1:n}$
$\Sigma_{t,\lambda}^{-1}, s_\lambda \leftarrow \text{prior\_fn}(\theta, t)$
for $j \leftarrow 1$ to n **do**
  $s_j \leftarrow \mathrm{s}_\psi(\theta, x_j, \alpha_t)$
  $\hat{\Sigma}_{t,j}^{-1} \leftarrow \frac{\alpha_t}{(1-\alpha_t)}\left(\mathbf{I}_m + (1-\alpha_t)\nabla_\theta \mathrm{s}_\psi(\theta, x_j, \alpha_t)\right)^{-1}$
end for
$\Lambda \leftarrow (1-n)\Sigma_{t,\lambda}^{-1} + \sum_{i=1}^n \hat{\Sigma}_{t,i}^{-1}$
$\tilde{s}_{1:n} \leftarrow (1-n)\Sigma_{t,\lambda}^{-1} s_\lambda + \sum_{i=1}^n \hat{\Sigma}_{t,i}^{-1} s_i$
$s_{1:n} \leftarrow \text{LinSolve}(\Lambda, \tilde{s}_{1:n})$

4. Experiments

In this section, we investigate the performance of GAUSS and JAC for different tasks from SBI literature with increasing difficulty. We take F-NPSE (Geffner et al., 2023) as a baseline for comparisons, which uses unadjusted Langevin dynamics (ULD) with $L = 5$ steps and $\tau = 0.5$. We have decided to not compare our methods to augmented versions of NPE, NLE, and NRE because this has already been done in Sharrock et al. (2022) and Geffner et al. (2023) with the score-diffusion framework demonstrating clearly superior performance. Of course, this choice reduces the range of comparisons of our experimental section, but allows us to focus solely on the best alternative method from current literature. For GAUSS and JAC we sample with the backward Markov chain defined in Section 2 with $\sigma_t^2 = \eta^2(1-\alpha_{t-1})/(1-\alpha_t)(1-\alpha_t/\alpha_{t-1})$ where $\eta = 0.2, 0.5, 0.8, 1$ for a number of steps $T = 50, 150, 400, 1000$ respectively. We use a uniform scheduling $\{t_i = i/T\}_{i=1}^T$. The code reproducing all experiments is available in the following repository.[4]

4.1. Gaussian toy models

We consider two toy examples for which the analytic form of the posterior and corresponding score distributions are known. Our first example is a multivariate Gaussian simulator $p(x \mid \theta) = \mathcal{N}(x; \theta, (1-\rho)\mathbf{I}_m + \rho \mathbf{1}_m)$ with correlation factor $\rho = 0.8$. The second is a Mixture of Gaussians (GMM) $p(x \mid \theta) = 0.5\,\mathcal{N}(x; \theta, 2.25\Sigma) + 0.5\,\mathcal{N}(x; \theta, \Sigma/9)$, where Σ is a diagonal matrix with values increasing linearly between 0.6 and 1.4, following Geffner et al. (2023). Both examples are carried out with a Gaussian prior $\lambda(\theta) = \mathcal{N}(\theta; 0, \mathbf{I}_m)$ in dimension $m = 10$. Here, the samples of the true posterior for multiple observations are obtained either directly for Gaussian or via Metropolis Adjusted Langevin (MALA) for GMM (see Appendix E for further details). We use the sliced Wasserstein (sW) distance to measure how close the obtained posterior samples are to the true samples.[5] The sW is computed with 10^4 slices, on 10^3 samples and results are reported over 5 different seeds.

Consider the following approximate score model

$$\tilde{\mathrm{s}}_\psi(\theta, x, \alpha_t) = \mathrm{s}_\psi(\theta, x, \alpha_t) + \epsilon(1-\alpha_t) r(\theta, x, \alpha_t)\ ,$$

where $\epsilon \geq 0$, $\mathrm{s}_\psi(\theta, x, \alpha_t)$ is the analytical posterior score and r is a randomly initialized neural net with outputs in range $[-1, 1]$, see Appendix D. This construction leads to an error of ϵ over the "noise predictor" network defined as $-\tilde{\mathrm{s}}_\psi(\theta, x, \alpha_t)/(1-\alpha_t)$, which is the neural network that one actually optimizes when training a score model. Note that unlike the Gaussian case, Tweedie's approximation is not exact for GMM, i.e. $p_{0|t|}(\theta_0|\theta_t)$ is not a Gaussian density for all θ_t and all $t \in [1:T]$. Therefore, this example allows us to quantify the effect of Tweedie's approximation while still controlling the score estimation error (through ϵ).

Table 1 displays the total running time and sW for each algorithm on the Gaussian example (Table 3 in Appendix G shows similar results for GMM). Based on this table, we consider "equivalent time settings", namely we run our algorithm with 400 and 1000 steps for **JAC** and **GAUSS** respectively and **LANGEVIN** for 400 steps. We can see that our algorithm yields smaller sW than the equivalent Langevin sampler while accounting for 5 times less neural network evaluation (NNE), thus 5 times faster. We show in Appendix G the differences in speed for each tested setting.

Figure 2 portrays the effect of the perturbation ϵ in the

[4] https://anonymous.4open.science/r/diffusions-for-sbi-7675

[5] In order to account for the finite-sample effects on sW, we considered a normalized version of the distance by removing the expected sW over different sample sets from the true distribution.

Algorithm	N steps	Δt (s)	sW
GAUSS	50	0.45 +/- 0.00	0.17 +/- 0.08
JAC	50	0.41 +/- 0.00	3.14 +/- 4.12
LANGEVIN	50	0.83 +/- 0.00	nan +/- nan
GAUSS	150	0.90 +/- 0.00	0.17 +/- 0.06
JAC	150	1.22 +/- 0.00	1.57 +/- 2.33
LANGEVIN	150	2.50 +/- 0.01	0.65 +/- 0.42
GAUSS	400	2.04 +/- 0.00	0.20 +/- 0.11
JAC	400	3.26 +/- 0.01	0.85 +/- 1.20
LANGEVIN	400	6.65 +/- 0.02	0.65 +/- 0.43
GAUSS	1000	4.77 +/- 0.01	0.22 +/- 0.10
JAC	1000	8.18 +/- 0.03	0.25 +/- 0.09
LANGEVIN	1000	16.65 +/- 0.03	0.65 +/- 0.42

Table 1. Sliced Wasserstein and total elapsed time for the Gaussian toy problem with $m = 10$, $n = 32$ and $\epsilon = 10^{-2}$ per algorithm and number of sampling steps. Mean and std over 5 different seeds.

posterior approximation for each algorithm for $n \in [1, 100]$. In the Gaussian example we see that **GAUSS** is better in all settings. This is the expected behaviour, since in this example our method based on second order approximations fits perfectly the score of the tall posterior. **JAC** is exact for zero or very small perturbations ($\epsilon = 0, 0.001$), but becomes unstable when it increases ($\epsilon = 0.01$). In GMM, our approximation **GAUSS** is competitive with **LANGEVIN**, achieving smaller sW for small n and then reaching the same sW for $n = 90$, while being the best for all n in the setting with the highest amount of perturbation ($\epsilon = 10^{-1}$). We see that **JAC** is extremely precise for $\epsilon = 0$ (which corresponds to the case where the analytical posterior score is available) and becomes unstable for $\epsilon > 0$. This suggests that **GAUSS** offers a good trade-off between precision and robustness and thus the better algorithm choice in SBI settings where the posterior score is unknown and has to be approximated.

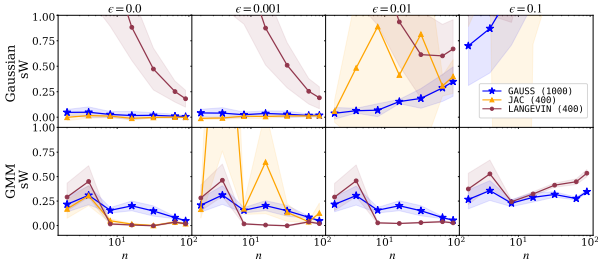


Figure 2. Normalized sW distance as a function of the number of observations n for the Gaussian and GMM toy examples, for each algorithm (**GAUSS**, **JAC** and **LANGEVIN**) and with different levels of ϵ. Mean and std over 5 different seeds.

We include in Appendix G a complete analysis for the Gaussian example, comparing all algorithms for different choices of dimensions m, number of observations n, and precision ϵ.

4.2. Benchmark SBI examples

We consider three examples from the increasingly popular SBI benchmark presented in Lueckmann et al. (2021b).

- SLCP ($m = 5$, $d = 8$): Uniform prior and Gaussian simulator whose mean and covariance are non-linear functions of input parameter θ.
- SIR ($m = 2$, $d = 10$): LogNormal prior and simulator based on a set of differential equations and outputs sampled from a Binomial distribution.
- Lotka-Volterra ($m = 4$, $d = 20$): LogNormal prior and simulator based on a set of differential equations and outputs sampled from a LogNormal output.

The score corresponding to each prior distribution is analytically computable; see Appendix D.[6] However, contrarily to the toy models considered in Section 4.1, the analytical posterior score is not available for the chosen examples and is, therefore, learned via score-matching (considering the loss function from Section 2.1).

Note that the purpose of the experiments in this section is not to find the best score model and the best posterior approximations for each task, which has already been done by Sharrock et al. (2022) and Geffner et al. (2023), but to evaluate the robustness of the different sampling algorithms. This is why we use the same score model architecture and training hyper parameters for all tasks, which may lead to different quality for the score estimator in each example (the data dimension and shape of the posterior impact the training of the score model). The used score model is always a MLP with 3 hidden layers, trained on N_{train} samples over 5000 epochs using the Adam optimizer. We standardize the train data to have zero-mean and $\mathbf{I}_m$ covariance matrix, which ensures that all the random variables θ_t created during the generative process have zero-mean and $\mathbf{I}_m$ covariance. Further details on the model architecture and training procedure are provided in Appendix E.

According to the *equivalent time setting* defined in Section 4.1, we use 1 000 sampling steps for **GAUSS** and 400 steps for **JAC** and **LANGEVIN**. We also consider *clipped* versions for all proposed algorithms (represented with dotted lines in all figures), where the generated samples at each step are *clipped* between (-3,3) so to ensure that all samples stay within the region of high probability for the standard Gaussian distribution. We introduce this numerical procedure to stabilize the **JAC** and **LANGEVIN** algorithms, that we found to be less robust than **GAUSS**, with quickly diverging sW for poor score estimators; see Section 4.1. However, this

[6] The LogNormal distribution can easily be transformed into a Gaussian distribution since when $\theta \sim \text{LogNormal}(m, s)$ then $\log \theta \sim \mathcal{N}(m, s)$.

procedure significantly slows down the sampling procedure and can introduce some bias in the posterior approximation.

Our empirical evaluation samples twenty-five[7] different ground truth parameters $\theta^\star \sim \lambda(\theta)$ from the prior distribution, each of which used to simulate observations $x_j^\star \sim p(x \mid \theta^\star)$ for $j = 1, \dots, n$, later plugged in as conditioning variables in the tall posterior $p(\theta \mid x_{1:n}^\star)$. For all three examples, samples from the true tall posterior can be obtained via MCMC using the `numpyro` package (Phan et al., 2019; Bingham et al., 2019) and used to compute the sliced Wasserstein (sW) distance in every setting corresponding to varying N_{train} and n.

Figure 3 portrays the sW distance for each task as a function of the size N_{train} of the training set for the score model. Larger values of N_{train} are expected to yield better score estimates and thus more accurate posterior approximations. This is the case for all three examples, where we observe a decreasing tendency and convergence to 0 for all n. Overall, we observe that our algorithm `GAUSS` outperforms all others. Indeed, it scales to high n values, while `LANGEVIN` diverges (or is non-decreasing) for $n \geq 14$ for the `Lotka-Volterra` and `SIR` examples. Note that the unclipped version of `JAC` diverges in every case as soon as $n > 1$, which is why we didn't include it in the plots. `GAUSS` yields consistently lower distance values than `LANGEVIN`. `JAC-clip` is more or less equivalent to `GAUSS`, with slightly better results for `SLCP`. In Appendix H, Figure 11 display the same results, but as a function the number of observations $n \in [1, 8, 14, 22, 30]$ for a given score model (i.e. fixed N_{train}). We also report results for the Maximum Mean Discrepancy (MMD) metric in Figures 12 and 13. Additionally, Figures 14 and 15 showcase the ability of the obtained posterior samples to concentrate (MMD to dirac) around the ground truth parameter $\theta^\star$.

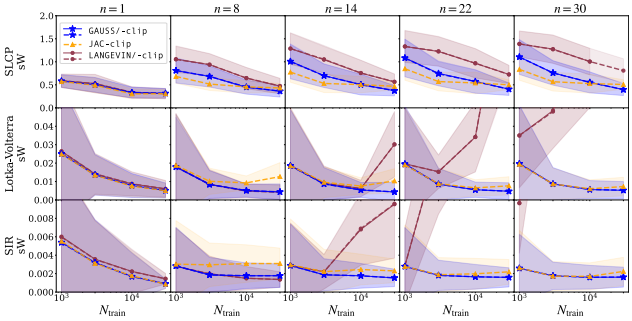


Figure 3. Curves with the sliced Wasserstein distance as a function of $N_{\text{train}} \in [10^3, 3.10^3, 10^4, 3.10^4]$ between the samples obtained by each algorithm (**GAUSS**, **JAC** and **LANGEVIN**) and the true tall posterior distribution $p(\theta \mid x_{1,n}^\star)$ for $n \in [1, 8, 14, 22, 30]$. Mean and std over 25 different parameters $\theta^\star \sim \lambda(\theta)$.

[7] We discarded outliers (max. 3% of the 25 points), which correspond to a sW for **GAUSS** above the 99% quantile (or `NaNs`).

4.3. Inverting a non-linear model from computational neuroscience

We illustrate our proposal on a classic model from computational neuroscience and consider the Bayesian inversion of the Jansen & Rit neural mass model (JRNMM) (Jansen & Rit, 1995). Neural mass models are non-linear models constructed based on physiologically motivated stochastic differential equations and are able to replicate oscillatory electrical signals experimentally observed with electroencephalography (EEG) . Neural mass models are commonly used in large-scale simulators of the brain (Sanz Leon et al., 2013) and are present in several simulation studies in cognitive and clinical neuroscience (Aerts et al., 2018). We follow the setup proposed by Rodrigues et al. (2021). Specifically, we consider a stochastic version of the JRNMM (Ableidinger et al., 2017) and use the C++ code provided by the authors of Buckwar et al. (2019). The output $x(t)$ of this generative model is a time series obtained by taking as input a set of four parameters $\theta = (C, \mu, \sigma, g)$. To help with the interpretations of the results, we provide some details on the parameters: Parameter C influences the general shape of the oscillatory behavior, (μ, σ) drive the statistics of the signals $s(t)$ generated by the physiological model, and g represents a gain factor of an amplifier (resp. attenuator) used to measure the signals and produce the actual model output $x(t) = 10^{g/10} s(t)$. The coupling-effect of parameters g and (μ, σ) on the amplitude of the output signal $x(t)$ means that the model is non-injective: the same observed signal $x^\star(t)$ could be generated with larger (smaller) values of g and smaller (larger) values of μ and σ.

We now apply the different sampling algorithms **GAUSS**, **JAC** and **LANGEVIN** (and their clipped versions[8]), again with respectively 1000, 400 and 400 steps, to infer the posterior distribution of the JRNMM for a set of n observations $x_j^\star \sim p(x \mid \theta^\star)$ independently simulated with $\theta^\star = (C^\star, \mu^\star, \sigma^\star, g^\star) = (125, 220, 2000, 0)$. A uniform prior distribution is placed over the range of physiologically meaningful values of the simulator parameters as done in Buckwar et al. (2019); Rodrigues et al. (2021). The posterior score is again estimated using the model architecture and training procedure from Section 4.2. We evaluate the ability of our posterior approximator to concentrate around the true parameters $\theta^\star$, which we quantify using the Maximum Mean Discrepancy (MMD) between the marginals of the approximate posterior and the Dirac distribution $\delta_{\theta^\star}$ centered at the true parameter $\theta^\star$.

We first consider a simplified setting for which the gain parameter is fixed $g = g^\star = 0$ so to lift the indeterminacy on the estimation of parameters (μ, σ). Results are shown in Figure 4. On the left, we plot the MMD as a function of the number n of observations. **JAC** yields values outside

[8] Numerical procedure used to stabilize `JAC`; see Section 4.2.

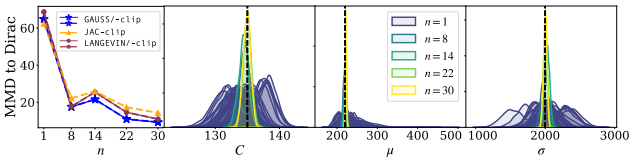


Figure 4. Inference on the 3D JRNMM (fixed $g = 0$). Left: MMD between the marginals of the approximate posterior obtained for for **GAUSS**, **JAC**, and **LANGEVIN**, and the Dirac of the true parameters $\theta^\star = (125, 220, 2000)$ (black dotted lines) used to simulate the observations $x^\star_{1:n}$. Right: Histograms of the 1D marginals of the posterior samples obtained with **GAUSS** for 30 single observations $x^\star_1, \dots, x^\star_{30}$ ($n = 1$) and for observation sets $x^\star_{1:n}$ of increasing size. We see that $p(\theta \mid x^\star_{1:n})$ concentrates around $\theta^\star$.

of the plot limits. All other methods perform as wanted, with decreasing MMD. On the right are displayed the 1D marginals corresponding to the posterior samples obtained with **GAUSS** for observation sets $x^\star_{1:n}$ of increasing size. We observe a progressive concentration around $\theta^\star$.

Figure 5 displays the results for the full 4-dimensional JRNMM. On the left plot, we can see that **GAUSS** yields consistenty lower MMD values, with a decrease for lower n, **LANGEVIN** is unstable and **JAC** yields values outside of the plot limits. The 1D and 2D marginals for samples obtained for **GAUSS** are shown on the right. The non-decreasing behaviour of MMD for higher n can be explained by the indeterminacy in the inference task for the (μ, σ) parameters, and is illustrated on the right plot, showing the 1D and 2D marginals for samples obtained for **GAUSS**. Indeed, we observe a "sharpened banana" shaped posterior distribution, containing the true parameters, concentrated around C and g, but dispersed along the dimensions of (μ, σ).

5. Conclusion

In every experimental setting it is always desirable to leverage information from as much data as possible. Model inversions with SBI are no exception and the *tall data* extension considered in this paper provides a way of parsing extra observations in an efficient way. Indeed, our method only requires the training of an initial score model for a single observation to construct the score of the *tall data* posterior. This allows `GAUSS` and `JAC` to be more simulation efficient, avoiding the shortcomings of the augmented datasets required by NPE, NLE, and NRE. Moreover, we tackle directly the challenge of constructing a diffused version of the posterior distribution for *tall data* based on recently proposed second order approximations of the backward diffusion kernels. This is considerably different from what was proposed in Geffner et al. (2023), where the problem was simplified by constructing a sequence of distributions relying on the individual scores, but at the cost of drastically reducing the arsenal of samplers available for sampling

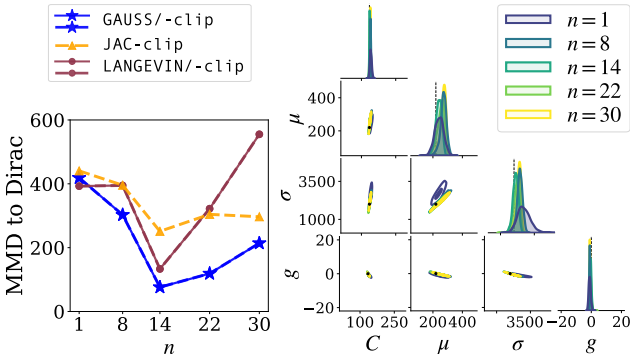


Figure 5. Inference on the full JRNMM. Left: MMD between the marginals of the approximate posterior obtained for **GAUSS**, **JAC**, and **LANGEVIN**, and the Dirac of the true parameters $\theta^\star = (125, 220, 2000, 0)$ used to simulate the observations $x^\star_{1,n}$. Right: Histograms of the 1D and 2D marginals of the posterior samples obtained with **GAUSS** for single observations $x^\star$ ($n = 1$) and for observation sets $x^\star_{1:n}$ of increasing n. We observe a progressive convergence of the inferred posterior mean towards $\theta^\star$ (black dots and lines), a sharpening around (C, g) and a sustained dispersion along the dimensions of (μ, σ) due to JRNMM's intrinsic indeterminacy.

from a score based generative model. Indeed, our methodology can benefit from recent advances of score-diffusion literature, such as DDIM (Song et al., 2021b), to perform backward sampling of the diffusion path and obtain samples from the posterior distribution in a much faster and stable way than the annealed Langevin procedure used in Geffner et al. (2023). We have illustrated the performance of our methods on different settings of increasing complexity, starting with toy examples for which the analytic posterior and scores were known, considering three examples from the SBI benchmark, and then finally applying the methodology to a challenging non-linear model from computational neuroscience. We demonstrated the superiority of our methods in terms of sample quality for almost all examples, and the reduced computational cost and increased numerical stability in every instance. Our work has confirmed the flexibility and viability of score-diffusion methods to approximate posterior distributions conditioned on sets of variable sizes and we expect that it will encourage other researchers to explore the score-diffusion framework for SBI problems.

Acknowledgement

Numerical computation was enabled by the scientific Python ecosystem: `numpy` (Harris et al., 2020), `scipy` (Virtanen et al., 2020), `matplotlib` (Hunter, 2007), `seaborn` (Waskom, 2021), and `pytorch` (Paszke et al.).

Impact statement

This paper presents work whose goal is to advance the field of machine learning applied to science. Although we base our approach on deep generative models, for which many ethical and societal questions are of interest, we do not feel that our specific application could generate any concerning issues. It is worth noting that one of our main contributions is the reduction in neural network evaluations for the *tall data* posterior sampling, which can be immediately translated into reduced energy consumption.

References

Ableidinger, M., Buckwar, E., and Hinterleitner, H. A stochastic version of the Jansen and Rit neural mass model: Analysis and numerics. *The Journal of Mathematical Neuroscience*, 7(1), August 2017. doi: 10.1186/s13408-017-0046-4.

Aerts, H., Schirner, M., Jeurissen, B., Van Roost, D., Achten, E., Ritter, P., and Marinazzo, D. Modeling brain dynamics in brain tumor patients using the virtual brain. *eNeuro*, 5(3), June 2018. ISSN 2373-2822. Society for Neuroscience.

Bardenet, R., Doucet, A., and Holmes, C. On Markov chain Monte Carlo methods for tall data. *Journal of Machine Learning Research*, 18:1–43, 5 2015. ISSN 15337928.

Batzolis, G., Stanczuk, J., Schönlieb, C.-B., and Etmann, C. Conditional image generation with score-based diffusion models, 2021.

Bhatia, R. *Positive definite matrices*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, December 2006.

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P. A., Horsfall, P., and Goodman, N. D. `Pyro`: Deep universal probabilistic programming. *J. Mach. Learn. Res.*, 20: 28:1–28:6, 2019.

Boys, B., Girolami, M., Pidstrigach, J., Reich, S., Mosca, A., and Akyildiz, O. D. Tweedie moment projected diffusions for inverse problems. *arXiv preprint arXiv:2310.06721*, 2023.

Brosse, N., Durmus, A., Moulines, E., and Sabanis, S. The tamed unadjusted langevin algorithm. 2017.

Buckwar, E., Tamborrino, M., and Tubikanec, I. Spectral density-based and measure-preserving ABC for partially observed diffusion processes. an illustration on hamiltonian SDEs. *Statistics and Computing*, 30(3):627–648, November 2019. doi: 10.1007/s11222-019-09909-6.

Cranmer, K., Brehmer, J., and Louppe, G. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences (PNAS)*, 117:30055–30062, 2020. ISSN 0027-8424. doi: 10.1073/pnas.1912789117.

Dax, M., Green, S. R., Gair, J., Pürrer, M., Wildberger, J., Macke, J. H., Buonanno, A., and Schölkopf, B. Neural importance sampling for rapid and reliable gravitational-wave inference. *Phys. Rev. Lett.*, 130:171403, Apr 2023. doi: 10.1103/PhysRevLett.130.171403.

Geffner, T., Papamakarios, G., and Mnih, A. Compositional Score Modeling for Simulation-Based Inference. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11098–11116. PMLR, 23–29 Jul 2023.

Gonçalves, P. J., Lueckmann, J.-M., Deistler, M., Nonnenmacher, M., Öcal, K., Bassetto, G., Chintaluri, C., Podlaski, W. F., Haddad, S. A., Vogels, T. P., Greenberg, D. S., and Macke, J. H. Training deep neural density estimators to identify mechanistic models of neural dynamics. *eLife*, 9:e56261, sep 2020. ISSN 2050-084X. doi: 10.7554/eLife.56261.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Greenberg, D., Nonnenmacher, M., and Macke, J. Automatic posterior transformation for likelihood-free inference. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 2404–2414. PMLR, 09–15 Jun 2019.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. 585 (7825):357–362, 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-2649-2. Number: 7825 Publisher: Nature Publishing Group.

Hermans, J., Begy, V., and Louppe, G. Likelihood-free MCMC with amortized approximate ratio estimators. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp.

4239–4248. PMLR, 13–18 Jul 2020. doi: 10.48550/arxiv.1903.04057.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Hunter, J. D. Matplotlib: A 2D graphics environment. 9 (3):90–95, 2007. ISSN 1558-366X. doi: 10.1109/MCSE.2007.55. Conference Name: Computing in Science & Engineering.

Hyvärinen, A. and Dayan, P. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Jansen, B. H. and Rit, V. G. Electroencephalogram and visual evoked potential generation in a mathematical model of coupled cortical columns. *Biological Cybernetics 1995 73:4*, 73:357–366, 9 1995. ISSN 1432-0770. doi: 10.1007/BF00199471.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 26565–26577, 2022.

Lueckmann, J.-M., Boelts, J., Greenberg, D., Goncalves, P., and Macke, J. Benchmarking simulation-based inference. In Banerjee, A. and Fukumizu, K. (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 343–351. PMLR, 13–15 Apr 2021a.

Lueckmann, J.-M., Boelts, J., Greenberg, D., Goncalves, P., and Macke, J. Benchmarking simulation-based inference. In Banerjee, A. and Fukumizu, K. (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 343–351. PMLR, 13–15 Apr 2021b.

Papamakarios, G., Sterratt, D., and Murray, I. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In Chaudhuri, K. and Sugiyama, M. (eds.), *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 837–848. PMLR, 16–18 Apr 2019.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22:1–64, 2021. ISSN 15337928. doi: 10.48550/arxiv.1912.02762.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Phan, D., Pradhan, N., and Jankowiak, M. Composable effects for flexible and accelerated probabilistic programming in NumPyro. *arXiv preprint arXiv:1912.11554*, 2019.

Roberts, G. O. and Tweedie, R. L. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, pp. 341–363, 1996.

Rodrigues, P. L. C., Moreau, T., Louppe, G., and Gramfort, A. HNPE: Leveraging Global Parameters for Neural Posterior Estimation. In *NeurIPS 2021*, Sydney (Online), Australia, December 2021.

Sanz Leon, P., Knock, S., Woodman, M., Domide, L., Mersmann, J., McIntosh, A., and Jirsa, V. The Virtual Brain: a simulator of primate brain network dynamics. *Frontiers in Neuroinformatics*, 7:10, 2013. ISSN 1662-5196. doi: 10.3389/fninf.2013.00010.

Sharrock, L., Simons, J., Liu, S., and Beaumont, M. Sequential neural score estimation: Likelihood-free inference with conditional score based diffusion models. 2022.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.

Song, J., Vahdat, A., Mardani, M., and Kautz, J. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2023.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.

Tarantola, A. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, 2005. doi: 10.1137/1.9780898717921.

Vasist, M., Rozet, F., Absil, O., Mollière, P., Nasedkin, E., and Louppe, G. Neural posterior estimation for exoplanetary atmospheric retrieval. *Astronomy and Astrophysics*, 672:A147, April 2023. doi: 10.1051/0004-6361/202245263.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, I., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., and van Mulbregt, P. `scipy` 1.0: fundamental algorithms for scientific computing in python. 17(3):261–272, 2020. ISSN 1548-7105. doi: 10.1038/s41592-019-0686-2. Number: 3 Publisher: Nature Publishing Group.

Waskom, M. L. `seaborn`: statistical data visualization. 6 (60):3021, 2021. ISSN 2475-9066. doi: 10.21105/joss.03021.

Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zhang, W., Cui, B., and Yang, M.-H. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

A. Proofs

A.1. Proof of Lemma 3.1

Let $(\mu_k)_{1\leq k\leq K} \in (\mathbb{R}^m)^K$ and $(\Sigma_k)_{1\leq k\leq K}$ be covariance matrices in $\mathbb{R}^{m\times m}$. Denote by p_k the Gaussian pdf with mean μ_k and covariance matrix Σ_k. Note that

$$p_k : \theta_0 \mapsto \exp\left(\tilde{\zeta}_k + (\Sigma_k^{-1}\mu_k)^\top\theta_0 - \frac{1}{2}\theta_0^\top\Sigma_k^{-1}\theta_0\right) ,$$

where

$$\tilde{\zeta}_k = -\frac{1}{2}\left(m\log 2\pi - \log|\Sigma_k^{-1}| + \mu_k^\top\Sigma_k^{-1}\mu_k\right) .$$

Therefore,

$$\begin{aligned}\prod_{k=1}^K p_k(\theta_0) &= \exp\left(\sum_{k=1}^K \tilde{\zeta}_k - \tilde{\zeta}_{\text{all}}\right)\exp\left(\tilde{\zeta}_{\text{all}} + \tilde{\eta}^\top\theta_0 - \frac{1}{2}\theta_0^\top\tilde{\Lambda}\theta_0\right)\\ &= \exp\left(\sum_{k=1}^K \tilde{\zeta}_k - \tilde{\zeta}_{\text{all}}\right)\mathcal{N}(\theta_0;\tilde{\Lambda}^{-1}\varsigma\tilde{\eta},\tilde{\Lambda}^{-1}) ,\end{aligned}$$

with $\tilde{\eta} = \sum_{k=1}^K \Sigma_k^{-1}\mu_k$, $\tilde{\Lambda} = \sum_{k=1}^K \Sigma_k^{-1}$, and $\tilde{\zeta}_{\text{all}} = -(m\log 2\pi - \log|\tilde{\Lambda}| + \tilde{\eta}^\top\tilde{\Lambda}^{-1}\eta)/2$. We can apply this result to equation (12) with

$$\begin{aligned}\eta(\theta) &= \sum_{j=1}^n \Sigma_t^{-1}(\theta,x_j)\mu_t(\theta,x_j) + (1-n)\Sigma_{t,\lambda}^{-1}(\theta)\mu_{t,\lambda}(\theta) ,\\ \Lambda(\theta) &= \sum_{j=1}^n \Sigma_t^{-1}(\theta,x_j) + (1-n)\Sigma_{t,\lambda}^{-1}(\theta) ,\end{aligned}$$

since by assumption $\Lambda(\theta)$ is definite positive. This provides the following reformulation of equation (12):

$$\ell_\lambda(\theta, x_{1:n}^\star) = \log\int \exp\left(\sum_{j=1}^n \zeta_j(\theta) + (1-n)\zeta_\lambda(\theta) - \zeta_{\text{all}}(\theta)\right)\mathcal{N}(\theta_0;\Lambda^{-1}(\theta)\eta(\theta),\Lambda^{-1}(\theta))\mathrm{d}\theta_0 ,$$

which concludes the proof.

A.2. Proof of Lemma 3.2

Let $\zeta(\mu,\Sigma^{-1}) = -\left(m\log 2\pi - \log|\Sigma^{-1}| + \mu^\top\Sigma^{-1}\mu\right)/2$. Then,

$$\begin{aligned}\nabla_\theta\zeta_j(\theta,x_j) &= \nabla_\theta\zeta(\mu_t(\theta,x_j),\Sigma_t^{-1}(\theta,x_j))\\ &= \nabla_\mu\zeta(\mu_t(\theta,x_j),\Sigma_t^{-1}(\theta,x_j))^\top\nabla_\theta\mu_t(\theta,x_j)\\ &\qquad + \nabla_{\Sigma^{-1}}\zeta(\mu_t(\theta,x_j),\Sigma_t^{-1}(\theta,x_j))\nabla_\theta\Sigma_t^{-1}(\theta,x_j) ,\end{aligned}$$

where

$$\begin{aligned}\nabla_\mu\zeta(\mu,\Sigma^{-1}) &= -\Sigma^{-1}\mu\\ \nabla_\Sigma^{-1}\zeta(\mu,\Sigma^{-1}) &= (1/2)\left(\Sigma - \mu\mu^\top\right) .\end{aligned}$$

Therefore, we obtain

$$\begin{aligned}\nabla_\theta\zeta_j(\theta,x_j) = &-\mu_t(\theta,x_j)^\top\Sigma_t^{-1}(\theta,x_j)^\top\nabla_\theta\mu_t(\theta,x_j)\\ &+ (1/2)\left(\Sigma_t(\theta,x_j) - \mu_t(\theta,x_j)\mu_t(\theta,x_j)^\top\right)\nabla_\theta\Sigma_t^{-1}(\theta,x_j) .\end{aligned}$$

Note that $\nabla_\theta \mu_t(\theta, x_j) = (\sqrt{\alpha_t}/\upsilon_t)\Sigma_t(\theta, x_j)$, which leads to

$$\begin{aligned}\nabla_\theta \zeta_j(\theta, x_j) &= -\frac{\sqrt{\alpha_t}}{\upsilon_t}\mu_t(\theta, x_j) + (1/2)\left(\Sigma_t(\theta, x_j) - \mu_t(\theta, x_j)\mu_t(\theta, x_j)^\top\right)\nabla_\theta \Sigma_t^{-1}(\theta, x_j) \\ &= -\upsilon_t^{-1}\theta - \nabla_\theta \log p_t(\theta) x_j + (1/2)\left(\Sigma_t(\theta, x_j) - \mu_t(\theta, x_j)\mu_t(\theta, x_j)^\top\right)\nabla_\theta \Sigma_t^{-1}(\theta, x_j)\,.\end{aligned}$$

This leads to

$$\begin{aligned}\nabla_\theta \ell_\lambda(\theta, x_{1:n}) = -(1-n)\nabla_\theta \log p_t^\lambda(\theta) - \sum_{j=1}^n \nabla_\theta \log p_t(\theta) x_j^\star - \upsilon_t^{-1}\theta - \nabla_\theta \zeta_{\text{all}}(\theta) \\ + (1/2)\left[\sum_{j=1}^n \left(\Sigma_t(\theta, x_j) - \mu_t(\theta, x_j)\mu_t(\theta, x_j)^\top\right)\nabla_\theta \Sigma_t^{-1}(\theta, x_j)\right] \\ + \frac{1-n}{2}\left(\Sigma_{t,\lambda}(\theta) - \mu_{t,\lambda}(\theta)\mu_{t,\lambda}(\theta)^\top\right)\nabla_\theta \Sigma_{t,\lambda}^{-1}(\theta, x_j)\,.\end{aligned}$$

The score is then given by

$$\begin{aligned}\nabla_\theta \log p_t(\theta) x_{1:n} = -\upsilon_t^{-1}\theta - \nabla_\theta \zeta_{\text{all}}(\theta) \\ + (1/2)\left[\sum_{j=1}^n \left(\Sigma_t(\theta, x_j) - \mu_t(\theta, x_j)\mu_t(\theta, x_j)^\top\right)\nabla_\theta \Sigma_t^{-1}(\theta, x_j)\right] \\ + \frac{1-n}{2}\left(\Sigma_{t,\lambda}(\theta) - \mu_{t,\lambda}(\theta)\mu_{t,\lambda}(\theta)^\top\right)\nabla_\theta \Sigma_{t,\lambda}^{-1}(\theta, x_j)\,.\end{aligned}$$

We now estimate $\nabla_\theta \zeta_{\text{all}}(\theta)$ by noting that $\zeta_{\text{all}} = \zeta(\Lambda(\theta)^{-1}\eta(\theta), \Lambda(\theta))$. We obtain

$$\begin{aligned}\nabla_\theta \zeta_{\text{all}}(\theta) &= -\nabla_\theta(\Lambda(\theta)^{-1}\eta(\theta))\eta(\theta) + (1/2)\left[\mathbf{I}_m - \Lambda(\theta)^{-1}\eta(\theta)\eta(\theta)^\top\right]\Lambda(\theta)^{-1}\nabla_\theta \Lambda(\theta) \\ &= -\Lambda(\theta)^{-1}\nabla_\theta \eta(\theta)\eta(\theta) - \eta(\theta)^\top \nabla \Lambda(\theta)^{-1}\eta(\theta) + (1/2)\left[\mathbf{I}_m - \Lambda(\theta)^{-1}\eta(\theta)\eta(\theta)^\top\right]\Lambda(\theta)^{-1}\nabla_\theta \Lambda(\theta)\,.\end{aligned}$$

Note now that

$$\begin{aligned}\nabla_\theta \eta(\theta) &= \sum_{j=1}^n \Sigma_t^{-1}(\theta, x_j)\nabla_\theta \mu_t(\theta, x_j) + \mu_t(\theta, x_j)^\top \nabla_\theta \Sigma_t^{-1}(\theta, x_j) \\ &\quad + (1-n)\left(\Sigma_{t,\lambda}^{-1}(\theta)\nabla_\theta \mu_{t,\lambda}(\theta) + \mu_{t,\lambda}(\theta)^\top \nabla_\theta \Sigma_{t,\lambda}^{-1}(\theta)\right) \\ &= \frac{\sqrt{\alpha_t}}{\upsilon_t}\mathbf{I}_m + \sum_{j=1}^n \mu_t(\theta, x_j)^\top \nabla_\theta \Sigma_t^{-1}(\theta, x_j) + (1-n)\mu_{t,\lambda}(\theta)^\top \nabla_\theta \Sigma_{t,\lambda}^{-1}(\theta)\,,\end{aligned}$$

which leads to

$$\nabla_\theta \eta(\theta)\eta(\theta) = \frac{\sqrt{\alpha_t}}{\upsilon_t}\eta(\theta) + \left[\sum_{j=1}^n \mu_t(\theta, x_j)^\top \nabla_\theta \Sigma_t^{-1}(\theta, x_j) + (1-n)\mu_{t,\lambda}(\theta)^\top \nabla_\theta \Sigma_{t,\lambda}^{-1}(\theta)\right]\eta(\theta)\,. \tag{14}$$

Note that

$$\begin{aligned}\sqrt{\alpha_t}\eta(\theta) &= \sum_{j=1}^n \Sigma_t^{-1}(\theta, x_j)\left(\theta + \upsilon_t \nabla_\theta \log p_t(\theta) x_j\right) + (1-n)\Sigma_{t,\lambda}^{-1}(\theta)\left(\theta + \upsilon_t \nabla_\theta \log p_t^\lambda(\theta)\right) \\ &= \Lambda(\theta)\theta + \upsilon_t\left[\sum_{j=1}^n \Sigma_t^{-1}(\theta, x_j)\nabla_\theta \log p_t(\theta) x_j + (1-n)\Sigma_{t,\lambda}^{-1}(\theta)\nabla_\theta \log p_t^\lambda(\theta)\right]\,,\end{aligned}$$

which finally leads to

$$\nabla_\theta \log p_t(\theta \mid x^\star_{1:n}) = \Lambda(\theta)^{-1} \left[\sum_{j=1}^{n} \Sigma_t^{-1}(\theta, x_j) \nabla_\theta \log p_t(\theta) x_j + (1-n)\Sigma_{t,\lambda}^{-1}(\theta) \nabla_\theta \log p_t^\lambda(\theta) \right] + F(\theta, x^\star_{1:n}) \,,$$

where

$$\begin{aligned} F(\theta, x^\star_{1:n}) = \eta(\theta)^\top \nabla\Lambda(\theta)^{-1}\eta(\theta) + (1/2)\left[\mathbf{I}_m - \Lambda(\theta)^{-1}\eta(\theta)\eta(\theta)^\top\right]\Lambda(\theta)^{-1}\nabla_\theta\Lambda(\theta) \\ + (1/2)\left[\sum_{j=1}^{n}\left(\Sigma_t(\theta, x_j) - \mu_t(\theta, x_j)\mu_t(\theta, x_j)^\top\right)\nabla_\theta\Sigma_t^{-1}(\theta, x_j)\right] \\ + \frac{1-n}{2}\left(\Sigma_{t,\lambda}(\theta) - \mu_{t,\lambda}(\theta)\mu_{t,\lambda}(\theta)^\top\right)\nabla_\theta\Sigma_{t,\lambda}^{-1}(\theta, x_j)\,. \end{aligned}$$

B. Positive Definiteness of $\Lambda(\theta)$

The approximation described in Section 3.2 is only valid if matrix $\Lambda(\theta)$ is symmetric positive definite (SPD), i.e. it is a valid covariance matrix. In what follows, we will show what are the conditions on the choice of the pdf for the prior distribution that will ensure such property. Using the partial ordering defined by the convex cone of SPD matrices (Bhatia, 2006) it follows that:

$$\Lambda(\theta) \succ 0 \iff \sum_{j=1}^{n} \Sigma_t^{-1}(\theta, x_j) + (1-n)\Sigma_{t,\lambda}^{-1}(\theta) \succ 0 \,, \tag{15}$$

$$\iff \sum_{j=1}^{n} \Sigma_t^{-1}(\theta, x_j) \succ (n-1)\Sigma_{t,\lambda}^{-1}(\theta) \,, \tag{16}$$

$$\iff \Sigma_{t,\lambda}(\theta) \succ \left(\frac{1}{(n-1)}\sum_{j=1}^{n}\Sigma_t^{-1}(\theta, x_j)\right)^{-1} . \tag{17}$$

Note that as n increases, the R.H.S. of the inequality converges to the harmonic mean of the $\Sigma_t(\theta, x_j)$, which helps building an intuition to the correct choice for the covariance $\Sigma_{t,\lambda}(\theta)$ of the prior. For instance, a sufficient choice (not necessarily optimal) would be to have a $\Sigma_{t,\lambda}(\theta)$ whose associated ellipsoid[9] covers the ellipsoids generated by all other covariance matrices $\Sigma_t(\theta, x_j)$.

C. Influence of the correction term $\nabla_\theta \ell_\lambda(\theta, x^\star_{1:n})$.

Intuition following the definition of the backward kernels in equation (8):

- For $t \to 1$: the forward kernel and the diffused data distribution get close to the noise distribution: $p_t(\theta \mid x) \underset{t\to 1}{\to} \mathcal{N}(\theta; 0, \mathbf{I}_m)$ and $q_{t|0}(\theta \mid \theta_0) \underset{t\to 1}{\to} \mathcal{N}(\theta; 0, \mathbf{I}_m)$. Therefore the backward kernel is approximately the target data distribution:
$$p_{0|t}(\theta_0 \mid \theta, x) = \frac{p(\theta_0 \mid x) q_{t|0}(\theta \mid \theta_0)}{p_t(\theta \mid x)} \underset{t\to 1}{\sim} p(\theta_0 \mid x)\,. \tag{18}$$
Therefore the backward kernels vary very little with θ, and because they define $\ell_\lambda(\theta, x^\star_{1:n})$ (see eq. (12)), its gradient is close to zero. In other words, $\nabla_\theta \ell_\lambda(\theta, x^\star_{1:n})$ has no significant impact at the beginning of the backward diffusion (aka. sampling or generative process).
- For $t \to 0$: the denominator is the diffused distribution that gets close to the target data distribution $p_t(\theta \mid x) \underset{t\to 0}{\to} p(\theta_0 \mid x)$. Therefore the backward kernel is approximately
$$p_{0|t}(\theta_0 \mid \theta, x) = \frac{p(\theta_0 \mid x) q_{t|0}(\theta \mid \theta_0)}{p_t(\theta \mid x)} \underset{t\to 0}{\sim} q_{t|0}(\theta \mid \theta_0) \underset{t\to 0}{\to} \delta_{\theta_0}(\theta)\,. \tag{19}$$

[9] The ellipsoid $\mathcal{E}_A$ associated with SPD matrix A is defined as $\mathcal{E}_A = \{\mathbf{x} \,:\, \mathbf{x}^\top A^{-1}\mathbf{x} < 1\}$

Here, the dependence on θ is convergence to a dirac function. Which means that the gradient of $\ell_\lambda(\theta, x^\star_{1:n})$ will increase during the sampling process and finally explode when t approaches 0. The correction term therefore plays an important role as we approach the target tall data posterior distribution, at the end of the sampling process.

D. Analytical formulas for score and related quantities

D.1. Gaussian case

The considered Bayesian Inference task is to estimate the mean $\theta \in \mathbb{R}^m$ of a Gaussian simulator model $p(x \mid \theta) = \mathcal{N}(x; \theta, \Sigma)$, given a Gaussian prior $\lambda(\theta) = \mathcal{N}(\theta; \mu_\lambda, \Sigma_\lambda)$. For a single observation $x^\star$, the true posterior is also a Gaussian obtained using Bayes formula, as the product of two Gaussian distributions:

$$p(\theta \mid x^\star) = \mathcal{N}(\theta; \mu_{\text{post}}(x^\star), \Sigma_{\text{post}}) \tag{20}$$

with $\mu_{\text{post}}(x^\star) = \Sigma_{\text{post}}(\Sigma^{-1}x^\star + \Sigma_\lambda^{-1}\mu_\lambda)$ and $\Sigma_{\text{post}} = (\Sigma^{-1} + \Sigma_\lambda^{-1})^{-1}$. Note that in this case, the full posterior can be written as

$$p(\theta \mid x^\star_{1:n}) = \mathcal{N}(\theta; \mu_{\text{post}}(x^\star_{1:n}), \Sigma_{\text{post},n}) \tag{21}$$

with $\Sigma_{\text{post},n} = (n\Sigma^{-1} + \Sigma_\lambda^{-1})^{-1}$ and $\mu_{\text{post}}(x^\star_{1:n}) = \Sigma_{\text{post},n}(\sum_{j=1}^n \Sigma^{-1}x_j^\star + \Sigma_\lambda^{-1}\mu_\lambda)$.

Assume that $q_{t|0}(\theta \mid \theta_0) = \mathcal{N}(\theta; \sqrt{\alpha_t}\theta_0, \upsilon_t \mathbf{I}_m)$; see Section 2. Using standard results (e.g. equation 2.115 in Bishop, 2006), we can derive the analytic formula of the diffused prior

$$\begin{aligned} p_t^\lambda(\theta) &= \int \lambda(\theta_0) q_{t|0}(\theta \mid \theta_0) \mathrm{d}\theta_0 & (22)\\ &= \mathcal{N}(\theta; \sqrt{\alpha_t}\mu_\lambda, \alpha_t\Sigma_\lambda + \upsilon_t \mathbf{I}_m) & (23) \end{aligned}$$

and of the diffused posterior

$$\begin{aligned} p_t(\theta \mid x^\star) &= \int p(\theta_0 \mid x^\star) q_{t|0}(\theta \mid \theta_0) \mathrm{d}\theta_0 & (24)\\ &= \mathcal{N}(\theta; \sqrt{\alpha_t}\mu_{\text{post}}(x^\star), \alpha_t\Sigma_{\text{post}} + \upsilon_t \mathbf{I}_m). & (25) \end{aligned}$$

The corresponding Fisher scores are

$$\begin{aligned} \nabla_\theta \log p_t^\lambda(\theta) &= -(\alpha_t\Sigma_\lambda + \upsilon_t \mathbf{I}_m)^{-1}(\theta - \sqrt{\alpha_t}\mu_\lambda)\,, & (26)\\ \nabla_\theta \log p_t(\theta \mid x^\star) &= -(\alpha_t\Sigma_{\text{post}} + \upsilon_t \mathbf{I}_m)^{-1}(\theta - \sqrt{\alpha_t}\mu_{\text{post}}(x^\star))\,. & (27) \end{aligned}$$

Replacing the score from (27) in the formulas from (Boys et al., 2023), we get the following expressions for the mean and covariance matrix of the posterior backward diffusion kernel $p_{0|t}(\theta_0 \mid \theta, x)$:

$$\begin{aligned} \Sigma_t(\theta, x^\star) &= \frac{\upsilon_t}{\alpha_t}\left(1 - \upsilon_t\Big(\alpha_t\Sigma_{\text{post}} + \upsilon_t \mathbf{I}_m\Big)^{-1}\right) = \Sigma_t\,,\\ \mu_t(\theta, x^\star) &= \frac{1}{\sqrt{\alpha_t}}\left(1 - \upsilon_t\Big(\alpha_t\Sigma_{\text{post}} + \upsilon_t \mathbf{I}_m\Big)^{-1}\right)\theta + \upsilon_t\Big(\alpha_t\Sigma_{\text{post}} + \upsilon_t \mathbf{I}_m\Big)^{-1}\mu_{\text{post}}(x^\star)\\ &= \frac{\sqrt{\alpha_t}}{\upsilon_t}\Sigma_t\theta + \upsilon_t\Big(\alpha_t\Sigma_{\text{post}} + \upsilon_t \mathbf{I}_m\Big)^{-1}\mu_{\text{post}}(x^\star)\,. \end{aligned}$$

The same goes for the prior backward diffusion kernel $p^\lambda_{0|t}(\theta_0 \mid \theta)$:

$$\begin{aligned} \Sigma_{t,\lambda}(\theta) &= \frac{\upsilon_t}{\alpha_t}\left(1 - \upsilon_t\Big(\alpha_t\Sigma_\lambda + \upsilon_t \mathbf{I}_m\Big)^{-1}\right) = \Sigma_{t,\lambda}\,,\\ \mu_{t,\lambda}(\theta) &= \frac{1}{\sqrt{\alpha_t}}\left(1 - \upsilon_t\Big(\alpha_t\Sigma_\lambda + \upsilon_t \mathbf{I}_m\Big)^{-1}\right)\theta + \upsilon_t\Big(\alpha_t\Sigma_\lambda + \upsilon_t \mathbf{I}_m\Big)^{-1}\mu_\lambda\\ &= \frac{\sqrt{\alpha_t}}{\upsilon_t}\Sigma_{t,\lambda}\theta + \upsilon_t\Big(\alpha_t\Sigma_\lambda + \upsilon_t \mathbf{I}_m\Big)^{-1}\mu_\lambda\,. \end{aligned}$$

We now consider the tall data setting with n i.i.d. observations $x^\star_{1:n} = \{x^\star_1, \dots, x^\star_n\}$ and compute the correction term $\nabla_\theta \ell_\lambda(\theta, x^\star_{1:n})$. Note that the covariance matrices above are independent of θ and therefore $\Lambda(\theta)$ does not depend on θ and is referred to as Λ. Consequently, we only need to consider the terms that depend on the means $\mu_t(\theta, x^\star_j)$ or $\mu_{t,\lambda}(\theta)$ when computing the gradient w.r.t. θ.

$$\nabla \zeta_{0|t}(\theta, x) = -\frac{\sqrt{\alpha_t}}{\upsilon_t}\mu_t(\theta, x)\,, \tag{28}$$

$$\nabla \zeta_{0|t,\lambda}(\theta) = -\frac{\sqrt{\alpha_t}}{\upsilon_t}\mu_{t,\lambda}(\theta)\,. \tag{29}$$

For $\zeta_{0|t,\text{all}}$, write

$$\begin{aligned}
\eta_{0|t}(\theta) = &\sum_{j=1}^{N} \Sigma_t^{-1}\left[\frac{\sqrt{\alpha_t}}{\upsilon_t}\Sigma_t\theta + \upsilon_t\Big(\alpha_t\Sigma_{\text{post}} + \upsilon_t\mathbf{I}_m\Big)^{-1}\mu_{\text{post}}(x^\star_j)\right] \\
&+ (1-n)\Sigma_{t,\lambda}^{-1}\left[\frac{\sqrt{\alpha_t}}{\upsilon_t}\Sigma_{t,\lambda}\theta + \upsilon_t\Big(\alpha_t\Sigma_\lambda + \upsilon_t\mathbf{I}_m\Big)^{-1}\mu_\lambda\right] \\
&= \frac{\sqrt{\alpha_t}}{\upsilon_t}\theta + \upsilon_t\Sigma_t^{-1}\Big(\alpha_t\Sigma_{\text{post}} + \upsilon_t\mathbf{I}_m\Big)^{-1}\sum_{j=1}^{N}\mu_{\text{post}}(x^\star_j) \\
&+ (1-n)\upsilon_t\Sigma_{t,\lambda}^{-1}\Big(\alpha_t\Sigma_\lambda + \upsilon_t\mathbf{I}_m\Big)^{-1}\mu_\lambda\,.
\end{aligned}$$

We need to assure the positive definiteness of $\Lambda(\theta)$. Let $v \in \mathbb{R}^d$ such that $\|v\| = 1$. Thus,

$$v^\top\Lambda(\theta)v \propto \frac{1}{\upsilon_t} + \alpha_t^2(1-n)v^\top\Sigma_\lambda v + n\alpha_t^2 v^\top\Sigma_{\text{post}}v\,. \tag{30}$$

Therefore, for $\Lambda(\theta)$ to be positive definite is equivalent to

$$v^\top\Sigma_{\text{post}}v > \frac{n-1}{n}v^\top\Sigma_\lambda v - \frac{1}{n\upsilon_t\alpha_t^2}\,, \tag{31}$$

for all $v \in \mathbb{R}^d$ such that $\|v\| = 1$. Note that it is not trivial to find meaningful worst case scenario sufficient condition for the inequality above. To see why, let $e^{\min}_{\text{post}}$ be the smallest eigenvalue of Σ_{post} and $e^{\max}_\lambda$ the biggest eigenvalue of Σ_λ. Then, a sufficient condition would be $e^{\max}_\lambda < e^{\min}_{\text{post}}$, since

$$v^\top\Sigma_{\text{post}}v > e^{\min}_{\text{post}} > e^{\max}_\lambda > \frac{n-1}{n}v^\top\Sigma_\lambda v - \frac{1}{n\upsilon_t\alpha_t^2}\,. \tag{32}$$

But this does not make much sense, since we are asking the prior to be more concentrated than the posterior! Therefore, we obtain $\nabla\zeta_{0|t,\text{all}} = -\sqrt{\alpha_t}\Lambda^{-1}\eta_{0|t}(\theta)/\upsilon_t$, which yields

$$\nabla\ell_\lambda(\theta, x^\star_{1:n}) = -\frac{\sqrt{\alpha_t}}{\upsilon_t}\left(\sum_{j=1}^{n}\mu_t(\theta, x^\star_j) + (1-n)\mu_{t,\lambda}(\theta) - \Lambda^{-1}\eta_{0|t}(\theta)\right)\,. \tag{33}$$

D.2. Score of the diffused Uniform prior

Consider the case where the prior is a Uniform distribution $\lambda(\theta) = \mathcal{U}(\theta; a, b)$, with $(a, b) \in \mathbb{R}^m \times \mathbb{R}^m$, the lower and upper bounds respectively. Assume that $q_{t|0}(\theta \mid \theta_0) = \mathcal{N}(\theta; \sqrt{\alpha_t}\theta_0, \upsilon_t\mathbf{I}_m)$; see Section 2 with $\upsilon_t = 1 - \alpha_t$. We then get the following analytic formula for the diffused prior:

$$\begin{aligned}
p^\lambda_t(\theta) &= \int \lambda(\theta_0)q_{t|0}(\theta \mid \theta_0)\mathrm{d}\theta_0 && (34)\\
&= \frac{1}{\prod_{i=1}^m(b_i - a_i)}\int_{[a_1,b_1]\times\cdots\times[a_m,b_m]} \mathcal{N}\left(\theta; \sqrt{\alpha_t}\theta_0, \upsilon_t\mathbf{I}_m\right) && (35)\\
&= \frac{1}{\sqrt{\alpha_t}\prod_{i=1}^m(b_i - a_i)}\prod_{i=1}^{m}\left(\Phi\left(\sqrt{\alpha_t}b_i; \theta_i, \upsilon_t\right) - \Phi\left(\sqrt{\alpha_t}a_i; \theta_i, \upsilon_t\right)\right)\,, && (36)
\end{aligned}$$

where $\Phi(.;\mu,\sigma^2)$ is the c.d.f. of a univariate Gaussian with mean μ and variance σ^2. The score of the above quantity is then simply obtained by computing the score of each one-dimensional element in the above product. For $i \in [1,m]$, $\nabla_\theta \log p_t^\lambda(\theta)_i = \nabla_{\theta_i} \log f(\theta_i) = \frac{\nabla_{\theta_i} f(\theta_i)}{f(\theta_i)}$ with

$$f(\theta_i) = \left(\Phi\left(\sqrt{\alpha_t}b_i;\theta_i,\upsilon_t\right) - \Phi\left(\sqrt{\alpha_t}a_i;\theta_i,\upsilon_t\right)\right) \tag{37}$$

$$\nabla_{\theta_i} f(\theta_i) = -\frac{1}{\sqrt{\alpha_t\upsilon_t}}\left(\mathcal{N}\left(\sqrt{\alpha_t}b_i;\theta_i,\upsilon_t\right) - \mathcal{N}\left(\sqrt{\alpha_t}a_i;\theta_i,\upsilon_t\right)\right). \tag{38}$$

D.3. Mixture of Gaussians

In the case of the Mixture of Gaussians, the prior is $\lambda = \mathcal{N}(0,\mathbf{I}_m)$, the simulator is given by $p(x|\theta) = (1/2)\mathcal{N}(x;\theta,\Sigma_1) + (1/2)\mathcal{N}(x;\theta,1/9\Sigma_2)$ where $\Sigma_1 = 2.25\Sigma$, $\Sigma_2 = 1/9\Sigma$ and Σ is a diagonal matrix with values increasing linearly between 0.6 and 1.4. In this case, the posterior pdf is

$$p(\theta|x) = \omega_1(x)\mathcal{N}(\theta;\mu_1,\Sigma_{1,p}) + \omega_2(x)\mathcal{N}(\theta;\mu_2,\Sigma_{2,p}) \tag{39}$$

where for $i=1,2$, $\Sigma_{i,p} = \left(\Sigma_i^{-1} + \mathbf{I}_m\right)^{-1}$, $\mu_i = \Sigma_{i,p}\Sigma_i^{-1}x$, $\tilde{\omega}_i(x) = \mathcal{N}(x;\theta,\Sigma_i + \mathbf{I}_m)$ and $\omega_i = \tilde{\omega}_i / \sum_{j=1}^2 \tilde{\omega}_j$.

Therefore, the diffused marginals are

$$p_t(\theta|x) = \omega_1(x)\mathcal{N}(\theta;\alpha_t^{1/2}\mu_1,\alpha_t\Sigma_{1,p} + (1-\alpha_t)\mathbf{I}_m) + \omega_2(x)\mathcal{N}(\theta;\alpha_t^{1/2}\mu_2,\alpha_t\Sigma_{2,p} + (1-\alpha_t)\mathbf{I}_m), \tag{40}$$

from which the score is

$$\nabla_\theta \log p_t(\theta|x) = -\omega_1(x)(\alpha_t\Sigma_{1,p} + (1-\alpha_t)\mathbf{I}_m)^{-1}(\theta - \alpha_t^{1/2}\mu_1) - \omega_2(x)(\alpha_t\Sigma_{2,p} + (1-\alpha_t)\mathbf{I}_m)^{-1}(\theta - \alpha_t^{1/2}\mu_2). \tag{41}$$

E. Implementation details

All experiments are implemented with Python combined with PyTorch.

The score network. The same score model architecture and training hyper parameters are used for all tasks. Our implementation of the score model $s_\phi(\theta, x, t)$ is an MLP with layer normalization and 3 hidden layers of 256 hidden features. It takes as input the variables (θ, x, t) and outputs a vector of the same size as $\theta \in \mathbb{R}^m$. Before being passed to the score network, a positional embedding is computed for $t \in [0, 1]$ as per: $(\cos(ti\pi), \sin(ti\pi))_{1 \leq i \leq F}$, where we chose $F = 3$ for the number of frequencies. Optionally, θ and x can each be passed to an embedding network. Note that for the benchmark experiment in section 4.2, no embedding networks were used, as we chose to use the same score model architecture across all tasks, each with different data dimensions. For the more complex neuroscience application in section 4.3, θ and x were each embedded with with a MLP with 3 hidden layers of 64 hidden features and output embedding dimension of 32.

The score model is trained by minimizing the denoising score-matching (DSM) loss $\mathbb{E}\left[\|s_\phi(\theta, x, t) - \nabla_\theta \log q_{t|0}(\theta \mid \theta_0)\|^2\right]$, parametrized with the VP-SDE and a linear noise schedule $\beta(t) = 32t$ (Song et al., 2021b). For more stability we actually learn the *noise distribution* ϵ_ϕ (Ho et al., 2020; Song et al., 2021a), which is related to the score function via $\epsilon_\phi(\theta, x, t) = -\sigma(t) s_\phi(\theta, x, t)$. Given a training dataset $\left(\{(\theta_{0,i}, x_i)\}_{i=1}^{N_s} \sim p(\theta, x),\ t_i \sim \mathcal{U}(0, 1),\ z_i \sim \mathcal{N}(0, \mathbf{I}_m)\right)$, the empirical loss function writes:

$$\mathcal{L}(\phi) = \frac{1}{N_s} \sum_{i=1}^{N_s} \|\epsilon_\phi(\theta_i, x_i, t_i) - z_i\|^2 , \quad \text{with} \quad \theta_i = \sqrt{\alpha(t_i)}\theta_{0,i} + \sigma(t_i) z_i . \tag{42}$$

Training is done using the Adam optimizer over 5 000 epochs and early stopping using 20% of the training data. Note that early stopping requires to be done with care, since the loss function is stochastic. For each example and training set N_{train}, we therefore trained several models for different learning rates and batch sizes, and selected the ones that correspond to the best trade-off between smallest validation loss and latest stopping epoch (we want the score network to be trained over as many epochs as possible). See Figure 6 that displays the train and validation losses for all cases and explains the choice of the learning rates and batch sizes.

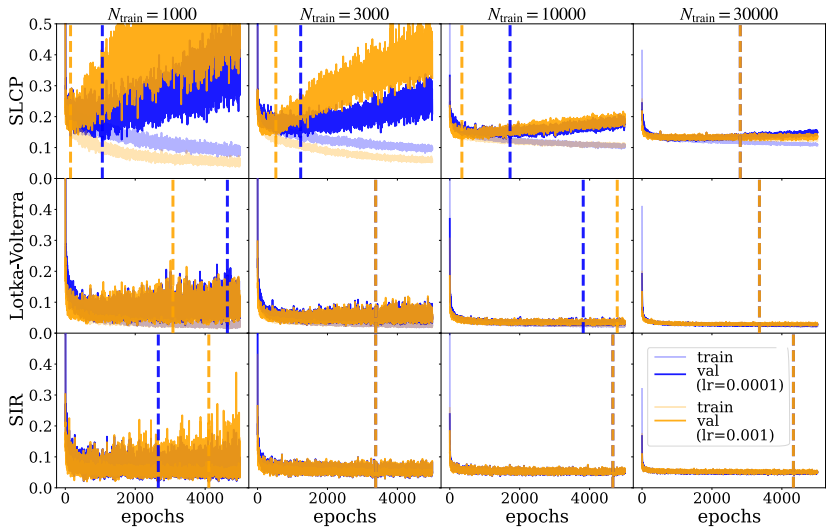
The training dataset is always normalized to zero mean and standard variance (for variables θ and x). The same transformation has to be applied to the observations $x^\star_{1:n}$ and the prior score function $\nabla_\theta \log p_t^\lambda(\theta)$ during sampling. After sampling, the inverse transformation is applied to the obtained samples, which are then compared to samples from the true posterior distribution $p(\theta \mid x^\star_{1:n})$. For the cases where the prior and / or the simulator corresponds to a LogNormal distribution, an additional log-transformation is applied to θ and / or x.[10]

(Diffusion) Posterior Sampling. For sampling the approximate tall posteriors, we use the score-based samplers chosen in section 4.1 according to the *equivalent time setting*, i.e. DDIM with respectively 1 000 steps, $\eta = 1$ and 400 steps, $\eta = 0.8$ for `GAUSS` and `JAC`, and 400 steps for `LANGEVIN` with the the same hyper-parameters as in (Geffner et al., 2023). The true posterior samples are obtained via MCMC using `numpyro` with `jax`., except for the Gaussian Mixture Toy Model from section 4.1, for which we consider the Metropolis-Adjusted Langevin Algorithm; see (Roberts & Tweedie, 1996). For a given number of observations n, we initialize the samples at $n^{-1} \sum_{j=1}^n x_j^\star + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \mathbf{I}_m)$. We run the MALA algorithm for 10^3 iterations with a learning rate of $n^{-1}10^{-3}$. We adapt the learning rate in the first 500 iterations to target an acceptance ratio of 0.5. To do so, we increase the learning rate by a factor of 1.01 if the acceptance rate is larger than 0.5 or by a factor of 0.99 if the acceptance rate is smaller than 0.45.

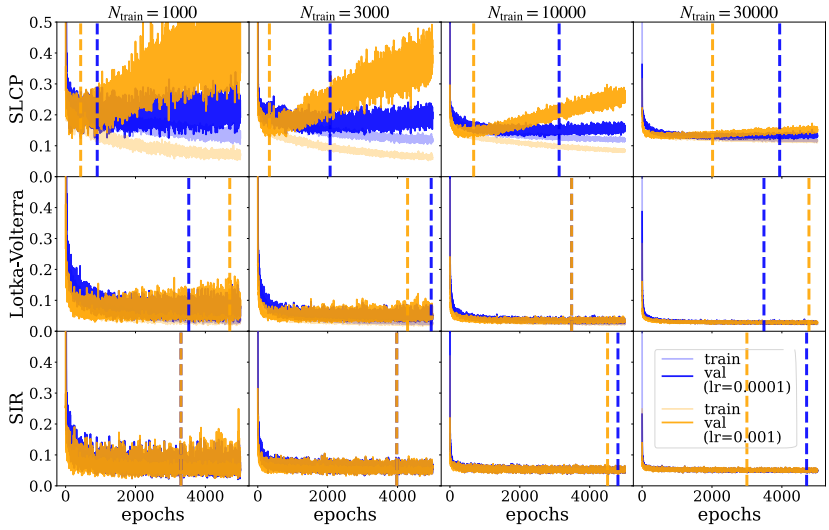
Evaluation metrics. To evaluate the accuracy of the sampling algorithms, we compute the sliced Wasserstein (sW) distance using the implementation form `POT` with 1 000 projections and the Maximum Mean Discreapancy (MMD) from the `sbi` package. We also evaluate the distance to the true parameters $\theta^\star$ used to simulated the observations $x^\star_{1:n}$ by computing the MMD to the Dirac function $\delta_{\theta^\star}$ (MMD to Dirac).

[10] This applies to the prior for both, the `Lotka Volterra` and `SIR` examples, and to the simulator for `Lotka Volterra` only.

F. Loss functions, training strategy and model selection



(a) Batch size 64.



(b) Batch size 256.

Figure 6. Train and validation losses for the SBI benchmark examples obtained for score networks trained over 5 000 epochs with the Adam optimzer and learning rates $\text{lr} = 1e^{-3}$ (orange) and $\text{lr} = 1e^{-4}$ (blue). The two Figures respectively corresponds to a batch size of 64 and 256. For small N_{train}, over fitting behavior can be detected (`SLCP` and `Lotka Volterra`). In these cases, a higher batch size of 256 is chosen, since it yields more stable loss functions, with delayed over fitting and thus also delayed early stopping. Since the best validation loss in each setting is similar (with small variations due to the stochasticity of the loss function), the learning rate can then be chosen to correspond to the latest early stopping epoch.

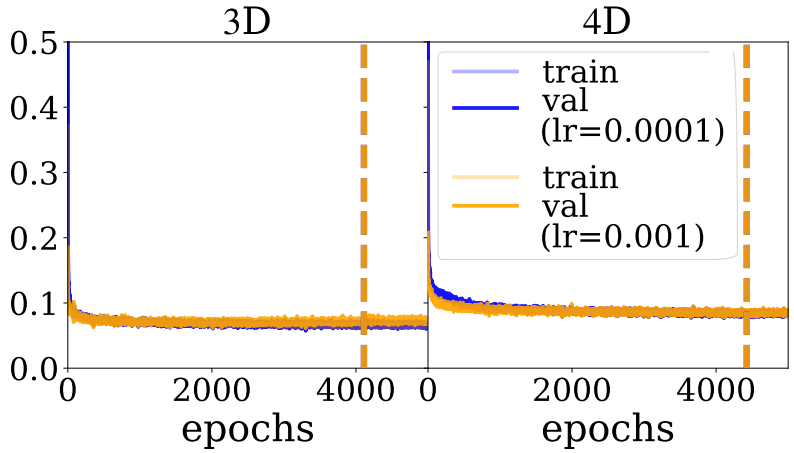


Figure 7. Loss functions for the Jansen & Rit Neural Mass Model. The validation loss was computed on a held-out validation set of 20% of the training dataset of size $N_{\text{train}} = 50\,000$. Dotted lines indicate the epoch at early stopping time.

G. Additional results for the toy models

m	Speed up GAUSS	Speed up JAC
2	0.39 ± 0.01	0.56 ± 0.00
4	0.39 ± 0.01	0.55 ± 0.00
8	0.39 ± 0.01	0.56 ± 0.00
10	0.38 ± 0.01	0.52 ± 0.00
16	0.38 ± 0.01	0.54 ± 0.00
32	0.37 ± 0.01	0.52 ± 0.00

Table 2. Confidence intervals for the ratio between elapsed time for each algorithm (GAUSS, JAC) and the equivalent Langevin sampler in the Gaussian case for each tested m. The values are averaged over number of sampling steps, ϵ and n.

Algorithm	N steps	Δt (s)	sW
GAUSS	50	0.99 +/- 0.00	0.30 +/- 0.05
JAC	50	0.89 +/- 0.00	82.73 +/- 67.41
Langevin	50	1.77 +/- 0.01	0.24 +/- 0.01
GAUSS	150	1.83 +/- 0.01	0.31 +/- 0.04
JAC	150	2.66 +/- 0.00	5.89 +/- 1.07
Langevin	150	5.28 +/- 0.01	0.35 +/- 0.02
GAUSS	400	3.91 +/- 0.01	0.31 +/- 0.04
JAC	400	7.13 +/- 0.02	3.41 +/- 1.04
Langevin	400	14.06 +/- 0.04	0.43 +/- 0.02
GAUSS	1000	8.85 +/- 0.03	0.34 +/- 0.03
JAC	1000	17.69 +/- 0.07	7.33 +/- 5.41
Langevin	1000	35.08 +/- 0.15	0.47 +/- 0.02

Table 3. Sliced Wasserstein (sW) and total elapsed time for the Gaussian Mixture toy problem with $d = 10$, $n = 32$ and $\epsilon = 10^{-2}$ per algorithm and number of generation steps. Mean and std over 5 different seeds.

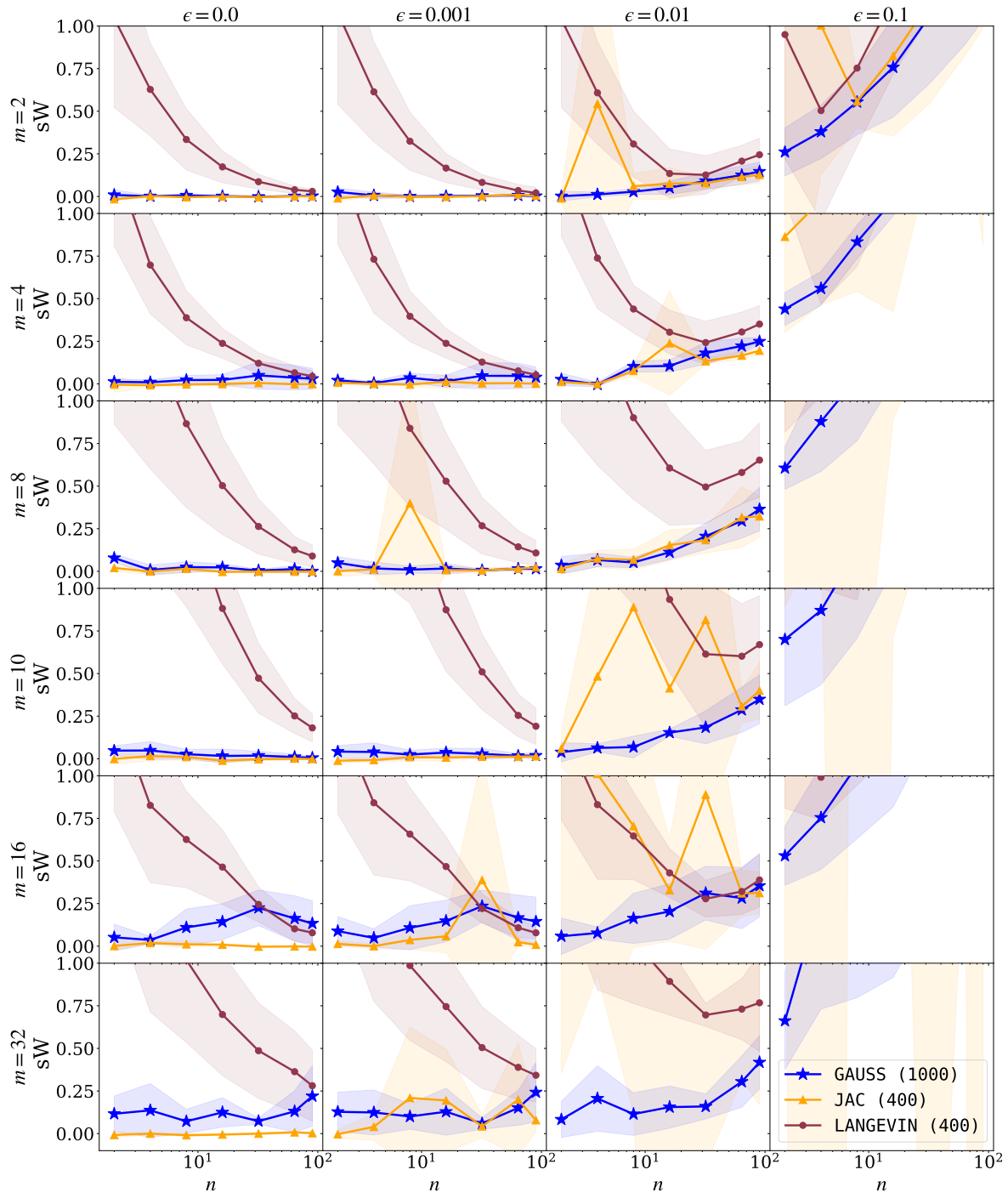


Figure 8. Normalized sW distance as a function of the number of observations n for **GAUSS**, **JAC** and **LANGEVIN** with different levels of ϵ. Results are shown for the `Gaussian` example in several dimensions $m \in [2, 4, 8, 10, 16, 32]$. Mean and std over 5 different seeds.

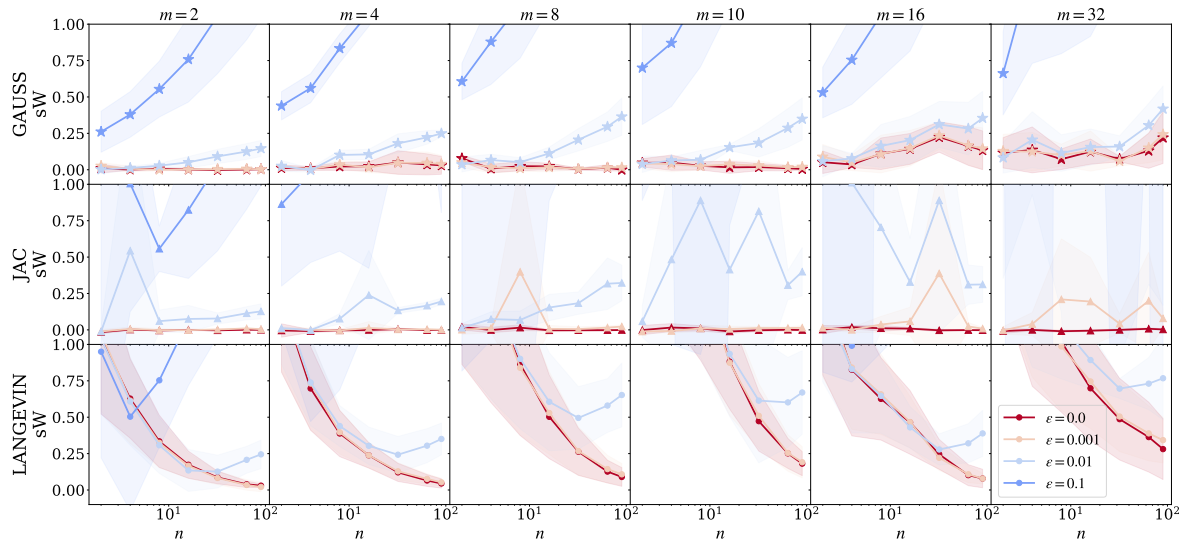


Figure 9. Sliced Wassersteiin (sW) distance as a function of the number of observations n for the different methods with different levels of ϵ. Results are shown for the `Gaussian` example in several dimensions $m \in [2, 4, 8, 10, 16, 32]$. Mean and std over 5 different seeds.

H. Additional results for SBI benchmarks

H.1. Sliced Wasserstein (sW) distance)

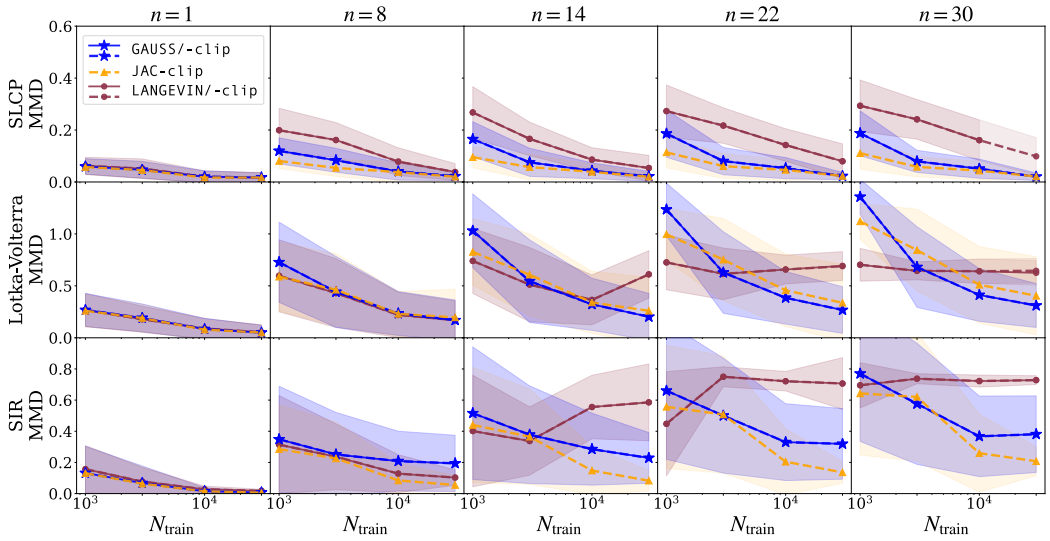


Figure 10. Sliced Wasserstein (sW) as a function of $N_{\text{train}} \in [10^3, 3.10^3, 10^4, 3.10^4]$ between the samples obtained by each algorithm and the true tall posterior distribution $p(\theta \mid x^\star_{1,n})$ (for $n \in [1, 8, 14, 22, 30]$). Mean and std over 25 different parameters $\theta^\star \sim \lambda(\theta)$.

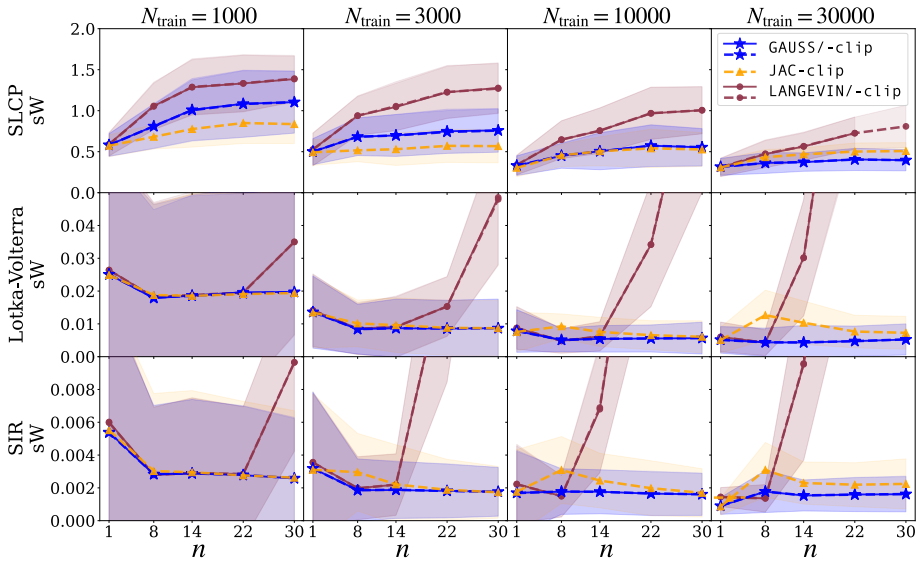


Figure 11. Sliced Wasserstein (sW) a function of $n \in [1, 8, 14, 22, 30]$ between the samples obtained by each algorithm and the true tall posterior distribution $p(\theta \mid x^\star_{1,n})$ (for $N_{\text{train}} \in [10^3, 3.10^3, 10^4, 3.10^4]$). Mean and std over 25 different parameters $\theta^\star \sim \lambda(\theta)$.

H.2. Maximum Mean Discrepancy (MMD)

As a complementary result to the one reported with the sliced Wassersetin distance, the MMD metric highlights an interesting point: Figures 12 and 13 show a general trend of *increasing MMD values for higher* n. This can be explained by the accumulation of approximation errors, as we sum over n evaluations of the score model to obtained the tall posterior score conditioned on n observations. We refer the reader to Appendix K, which investigates a possible solution to this issue: partially factoriized score-based posterior sampling algorithms (such as PF-NPSE proposed by (Geffner et al., 2023)).

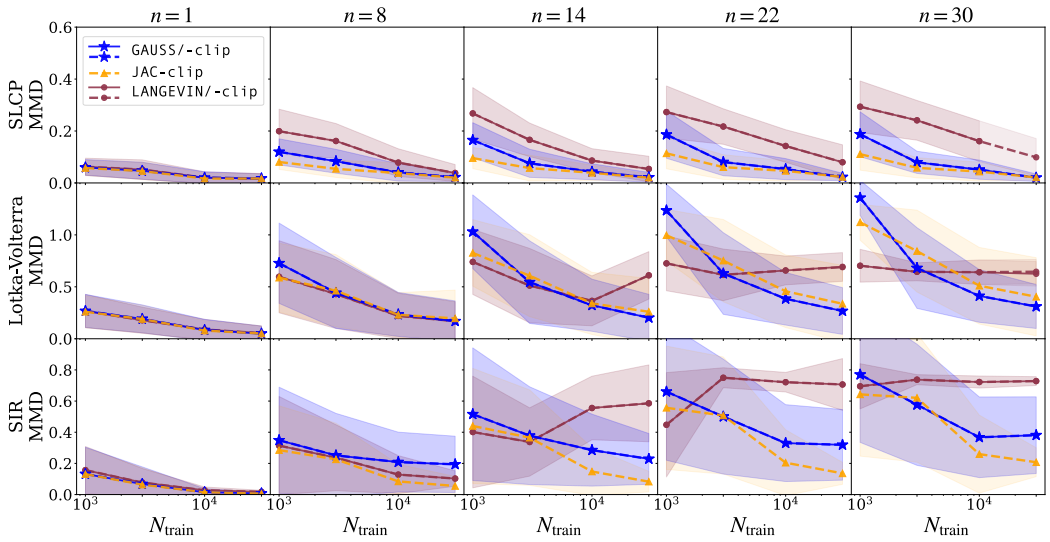


Figure 12. MMD as a function of $N_{\text{train}} \in [10^3, 3.10^3, 10^4, 3.10^4]$ between the samples obtained by each algorithm and the true tall posterior distribution $p(\theta \mid x^\star_{1,n})$ (for $n \in [1, 8, 14, 22, 30]$). Mean and std over 25 different parameters $\theta^\star \sim \lambda(\theta)$.

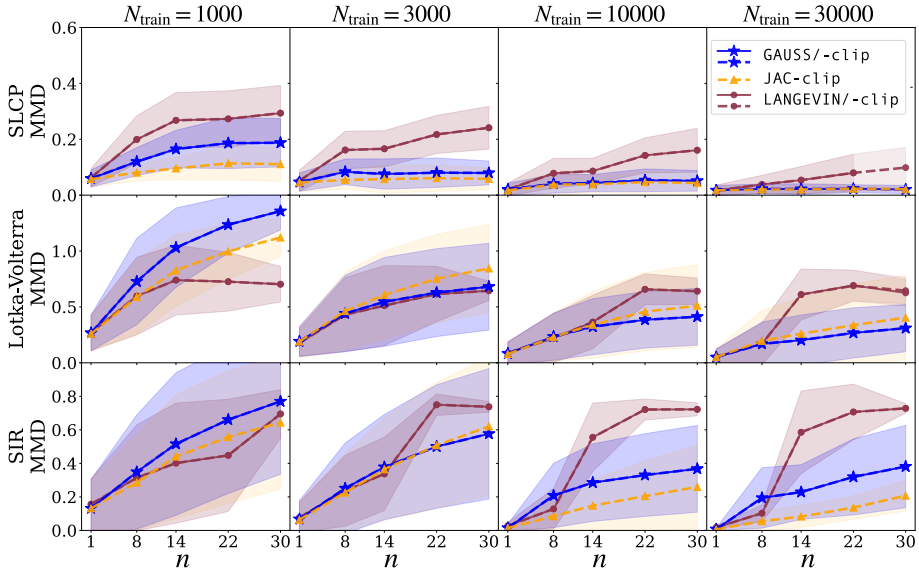


Figure 13. MMD as a function of $n \in [1, 8, 14, 22, 30]$ between the samples obtained by each algorithm and the true tall posterior distribution $p(\theta \mid x^\star_{1,n})$ (for $N_{\text{train}} \in [10^3, 3.10^3, 10^4, 3.10^4]$). Mean and std over 25 different parameters $\theta^\star \sim \lambda(\theta)$.

H.3. Distance to the Dirac of the true parameters (MMD to Dirac)

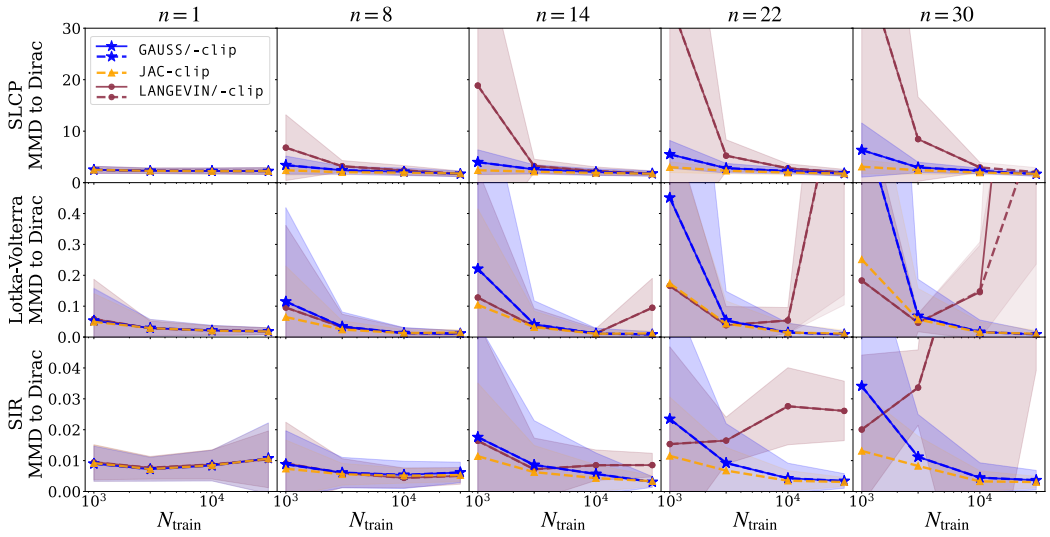


Figure 14. MMD as a function of $N_{\text{train}} \in [10^3, 3.10^3, 10^4, 3.10^4]$ between the marginals of the approximate tall posterior distribution $p(\theta \mid x^\star_{1,n})$ and the Dirac of the true parameters $\theta^\star$ used to simulate the observations $x^\star_{1,n}$ (for $n \in [1, 8, 14, 22, 30]$). Mean and std over 25 different parameters $\theta^\star \sim \lambda(\theta)$.

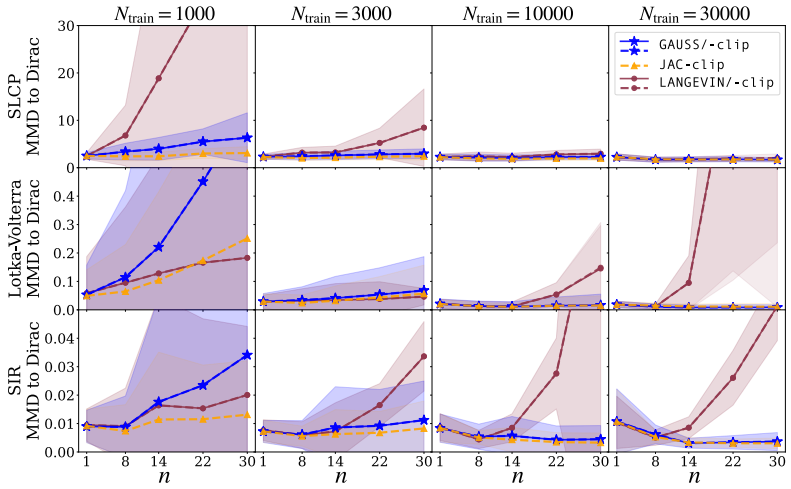


Figure 15. MMD as a function of $n \in [1, 8, 14, 22, 30]$ between the marginals of the approximate tall posterior $p(\theta \mid x^\star_{1,n})$ (for $N_{\text{train}} \in [10^3, 3.10^3, 10^4, 3.10^4]$) and the Dirac of the true parameters $\theta^\star$ used to simulate the observations $x^\star_{1,n}$. Mean and std over 25 different parameters $\theta^\star \sim \lambda(\theta)$.

I. Results for additional SBI Benchmark examples

To complete our empirical study, we added results for additional examples from the SBI benchmark (Lueckmann et al., 2021b): `Gaussian Linear`, `GMM`, `GMM (uniform)`[11], `B-GLM/ (raw)`[12] and `Two Moons`. These new results allows us to compare the performance of our proposal on other challenging situations, such as when scaling to highly structured (e.g. multimodal) posteriors and high-dimensional observation spaces. Note that these examples go a step further as compared to the experiments carried out by Geffner et al. (2023), including non-Gaussian priors[13]. Figures 16 and 17 respectively report the sW and MMD as a function of N_{train}.

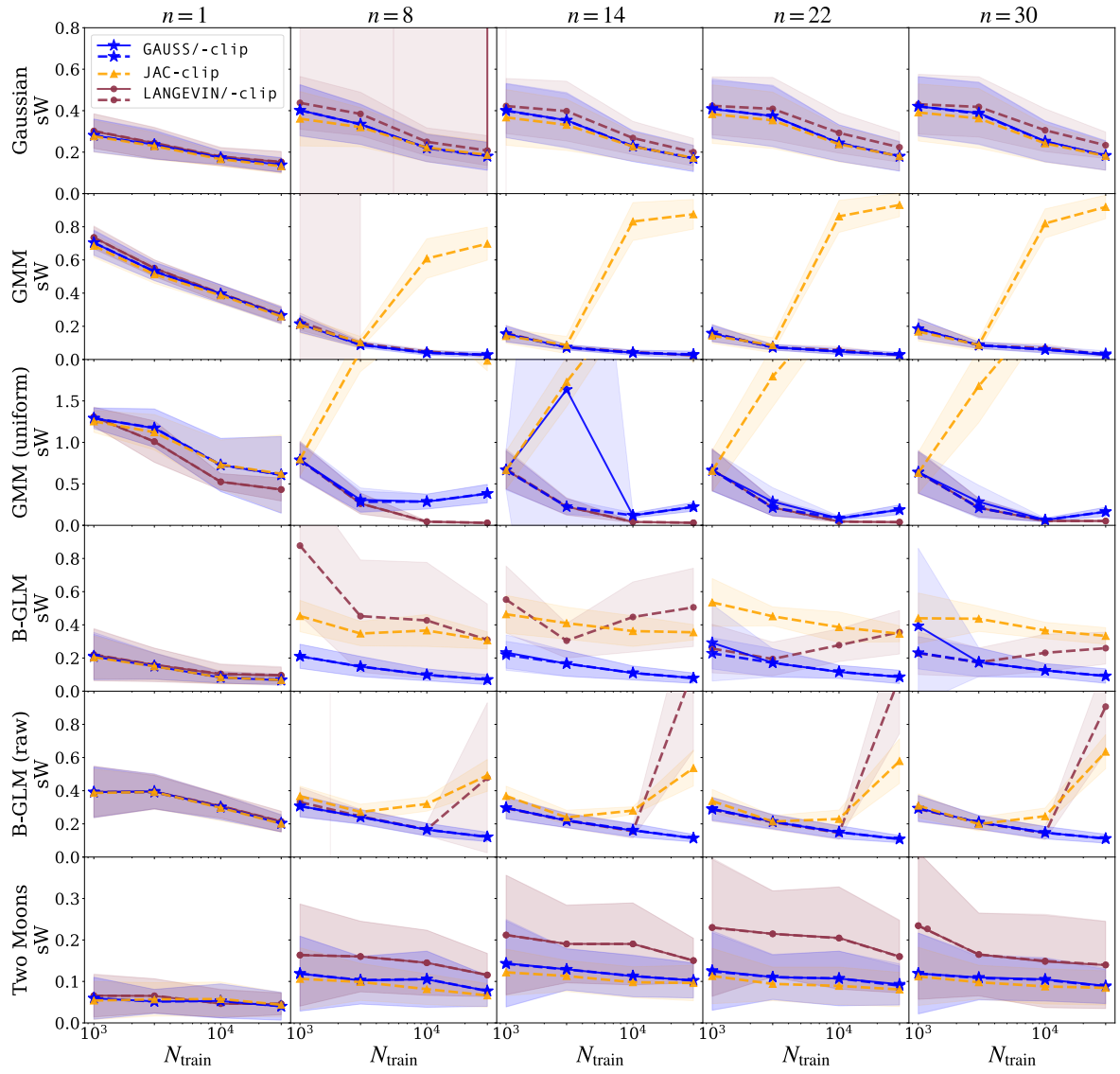


Figure 16. Results for additional benchmark examples (sW as a function of N_{train}).

[11] same as `GMM` but with a Uniform prior.
[12] Bernoulli GLM with summary statics / high dimentional raw data.
[13] Note that this was already the case for `SLCP` with Uniform prior.

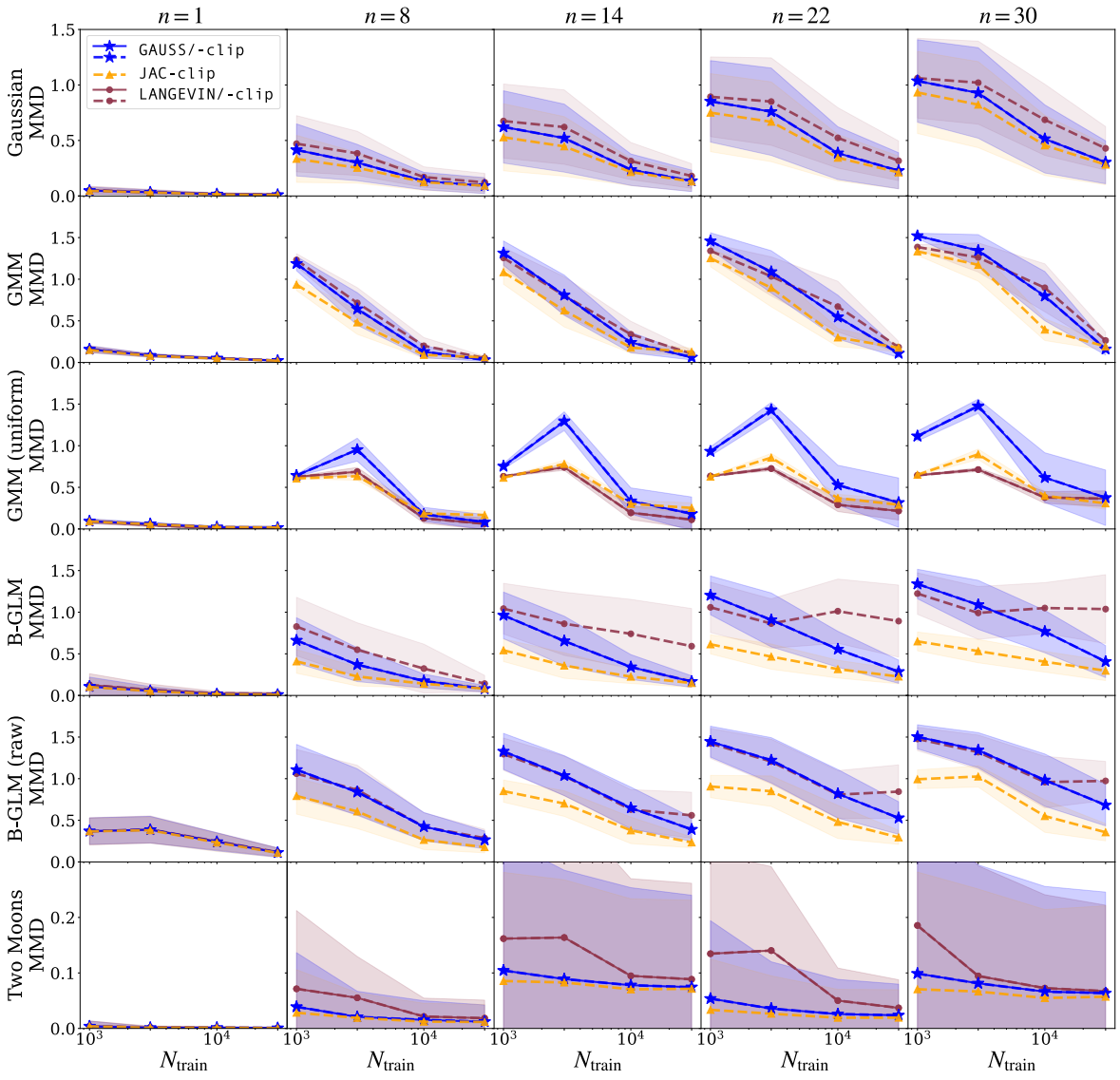


Figure 17. Results for additional benchmark examples (MMD as a function of N_{train}).

J. Limitations of Langevin sampling

Sensibility to the quality of the score model. Our results in section 4.1 of the main paper analyze the robustness of the different sampling algorithms and essentially show that the Langevin algorithm is very sensible to noisy score networks.

Step-size choice. We found that different step sizes can generate very different results for a given score model. We have added in Figure J a comparison between the original ULA (Unadjusted Langevin algorithm) used in (Geffner et al., 2023) and an additional implementation with "tamed" step sizes from (Brosse et al., 2017), which is known to stabilize ULA. We can see that the same choice of hyper-parameters for the "Geffner setting" leads to different behaviour with increasing n. Namely, the learning rate in the ULA algorithm seems to be to large (i.e. not enough steps are done) in settings with big n. We can see that the stabilization tools from the tamed version yield a more stable ULA algorithm but that does not provide a completely satisfying solution. Fundamentally, there exists a setting where the Langevin algorithm will work (taking a small enough learning rate for a long enough time), but this setting is extremely dependent of the problem at hand. This is precisely the strength of our algorithm when compared to ULA: we do not need to sample several times for each marginal p_t at each time step t. Note that unfortunately, the code for (Geffner et al., 2023) is not available, so our results are based in a best-effort attempt to reproduce the proposed algorithm.

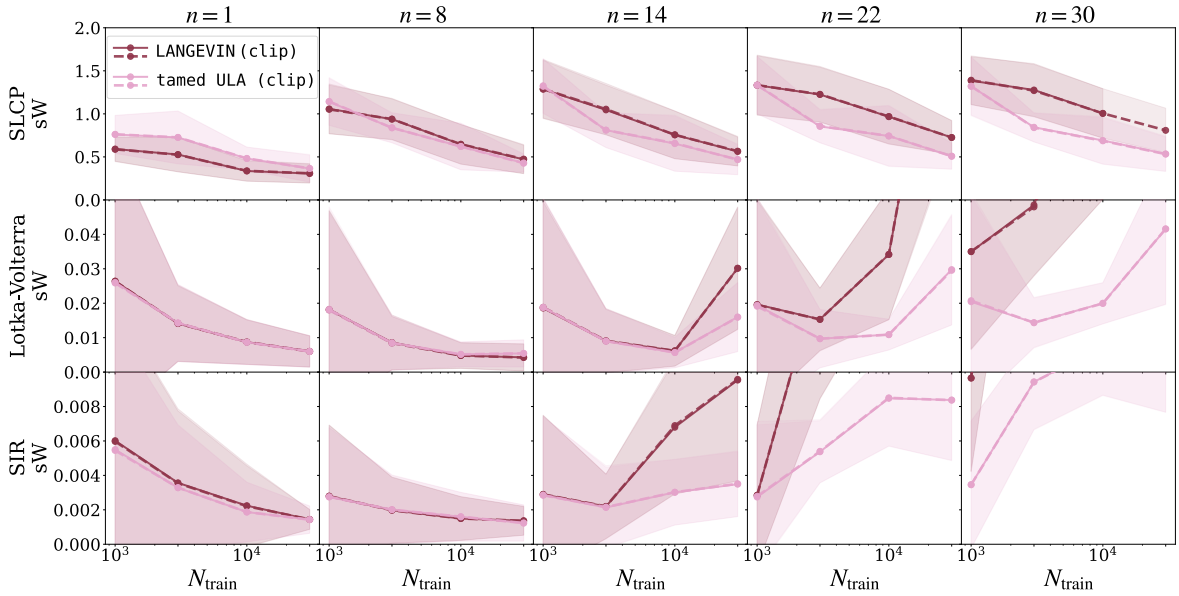
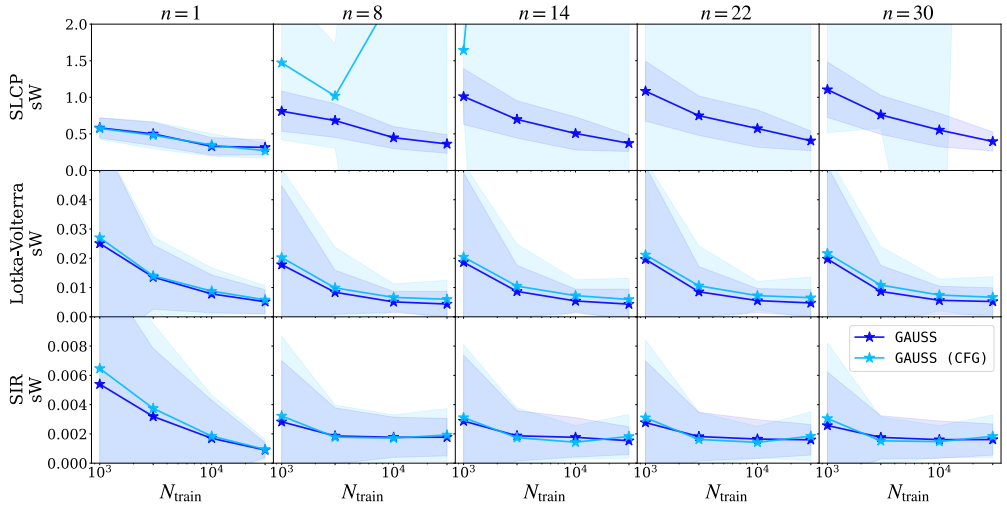


Figure 18. Comparison between the `LANGEVIN` algorithm from (Geffner et al., 2023) (used in all our experiments) and a more stable `tamed ULA` version with "tamed" step size from (Brosse et al., 2017). The plots show the sliced Wasserstein (sW) w.r.t. the true tall posterior $p(\theta \mid x^\star_{1:n})$ as a function of $N_{\text{train}} \in \{10^3, 3.10^3, 10^4, 3.10^4\}$ and for $n \in \{1, 8, 14, 22, 30\}$.

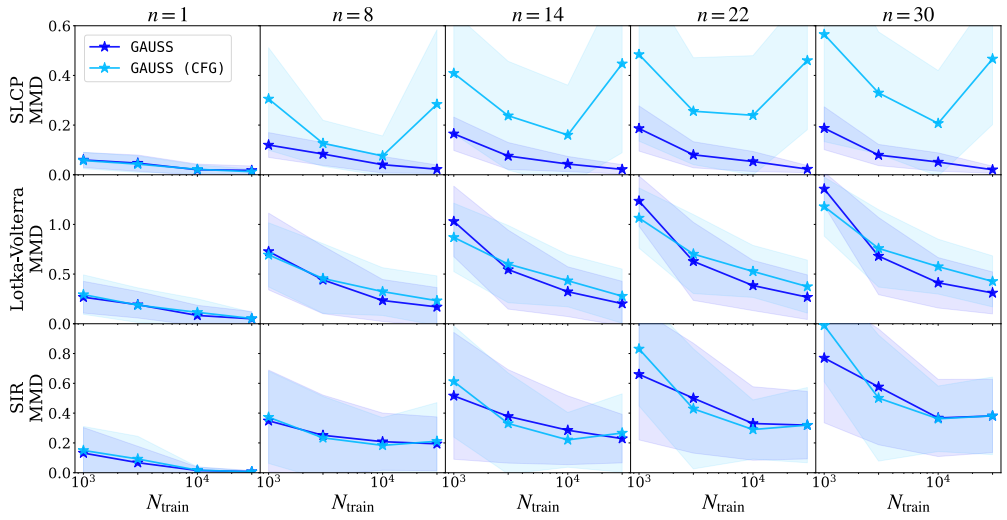
K. Beyond the proposal: Classifier-free guidance and Partial Factorization

The implementation of both of these extension can be found in our Code repository[14]. Their performance was investigated on the three benchmarks `Lotka-Volterra`, `SLCP` and `SIR`.

Classifier-free guidance (CFG). It is possible to implicitly learn the prior score via the classifier-free guidance approach (Ho & Salimans, 2021), which essentially consists in randomly dropping the context variables when training the posterior score model (e.g. 20% of the time). This is useful in cases where the diffused prior score cannot be computed analytically. Figure 19 displays the sliced Wasserstein and MMD as a function of N_{train} and compares the results obtained when using our proposition with the learned vs. the analytical prior score (resp. `GAUSS (CFG)` vs. `GAUSS`). The results are highly accurate for the LogNormal priors of `Lotka-Volterra` and `SIR`, but less satisfying for the Unifrom prior in `SLCP`. We think that this is caused by the discontiuities of the Uniform distribution. In summary, it seems that (under some smoothness assumptions) it is indeed possible to learn the prior score via the classifier-free guidance approach.



(a) Sliced Wasserstein (sW).



(b) Maximum Mean Discreapancy (MMD).

Figure 19. Comparison between the results obtained with our algorithm when combined with the analytical prior score and the one *learned via classifier-free guidance* (resp. `GAUSS` and `GAUSS (CFG)`). The plots show the sW and MMD w.r.t. the true tall posterior $p(\theta \mid x^\star_{1:n})$ as a function of $N_{\text{train}} \in \{10^3, 3.10^3, 10^4, 3.10^4\}$ and for $n \in \{1, 8, 14, 22, 30\}$.

[14] https://anonymous.4open.science/r/diffusions-for-sbi-7675

Partial factorization (PF-NPSE). In the same way as in (Geffner et al., 2023), our proposed algorithm can naturally be extended to a partially factorized version. Specifically, it consists in approximating the tall posterior by factorizing it over batches of context observations (instead of a single x). To do so, the score model is modified to take as input context sets with variable sizes (between 1 and $n_{\max}$). The given sampling algorithm (e.g. `GAUSS`, `LANGEVIN`) is then modified to split the context observations $x_1^\star, \dots, x_n^\star$ into subsets of smaller size $k < n_{\max} < n$, before passing them to the trained score model. This approach should allow for a good trade-off between the accumulation of approximation errors due to multiple evaluations of the score model ($n/n_{\max}$ times) and the increased simulation budget ($\times n_{\max}$). This approach should allow for a good trade-off between the accumulation of approximation errors due to multiple evaluations of the score model ($n/n_{\max}$ times) and the increased simulation budget ($\times n_{\max}$).

We investigated the performance of PF-NPSE on the SBI benchmarks (`Lotka-Volterra`, `SIR` and `SLCP`). For each of the three examples we trained a PF-NPSE model targeting the score models for the law of θ given $x_{1:n_{\max}}$ for $n_{\max} \in \{1, 3, 6, 30\}$. Figure 20 displays the Sliced Wasserstein (sW) for $n = 30$ observations as a function of the N_{train} for samples obtained with the partially factorized `LANGEVIN` and `GAUSS` samplers and for all $n_{\max}$. The extreme case $n_{\max} = 1$ corresponds to the original "fully" factorized version of the samplers. $n_{\max} = n = 30$ correspond to the other extreme case with no factorization, but maximum simulation budget. We can see that the optimal sW values lie in the middle of the spectrum (i.e. for $n_{\max} = 3, 6$), which corresponds to what was concluded in (Geffner et al., 2023). Note that the performance of `LANGEVIN` is drastically improved for $n_{\max} > 1$, while `GAUSS` all results are close. In any case, the results suggests that a practitioner will gain in choosing (a small enough) $n_{\max} > 1$.

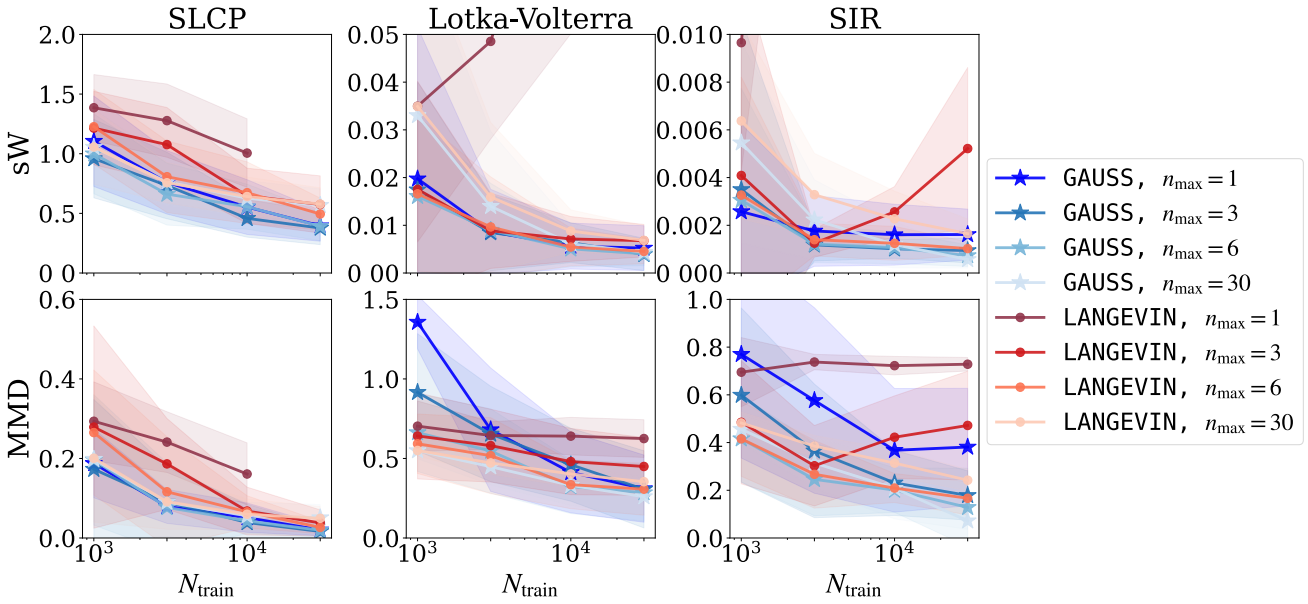


Figure 20. Results obtained with the *partially factorized* `LANGEVIN` and `GAUSS` samplers to infer the tall posterior conditioned on a total number of observations $n = 30$, for $n_{\max} = 1, 3, 6, 30$. We report the sliced Wasserstein (sW) and MMD w.r.t. the true tall posterior $p(\theta \mid x_{1:n}^\star)$ as a function of $N_{\text{train}} \in \{10^3, 3.10^3, 10^4, 3.10^4\}$. The extreme case $n_{\max} = 1$ corresponds to the original "fully" factorized version of the samplers. $n_{\max} = n = 30$ correspond to the other extreme case with no factorization, but maximum simulation budget. We can see that the optimal distance values lie in the middle of this spectrum (i.e. for $n_{\max} = 3, 6$).