Data assimilation as simulation-based inference

Auteur : Andry, Gérôme
Promoteur(s) : Louppe, Gilles
Faculté : Faculté des Sciences appliquées
Diplôme : Master : ingénieur civil électricien, à finalité spécialisée en "signal processing and intelligent robotics"
Année académique : 2022-2023
URI/URL : http://hdl.handle.net/2268.2/18255

Master's thesis

completed in order to obtain the degree of Master of Science in Electrical Engineering : signal processing and intelligent robotics

Data assimilation as Simulation-based inference

Gérôme Andry

Academic year 2022-2023
at the University of Liège
School of Engineering and Computer Science

Under the supervision of
Pr. Gilles Louppe

Abstract

Complex dynamical systems are found across various scientific disciplines, representing phenomena like atmospheric and oceanic behavior, brain activity, robot state in its environment, among many others. Due to the challenges that those systems may address, it is often impractical to observe their complete state, leading to the collection of partial observations. For instance, weather stations can only measure a limited number of variables like temperature and pressure, but not the entire state of the atmosphere. However, despite the limited nature of those observations, we can still use them to infer and deduce states that are consistent with the gathered data. By leveraging advanced inference methods, we can make predictions about the complete state of complex dynamical systems based on these information.

In this thesis, we delve into the realm of simulation-based inference methods applied to inverse problems in high-dimensional dynamical systems. We discuss how classical methods can be adapted to our problem and investigate how existing evaluation techniques can be used to assess our estimator's performances. Unlike classical simulation-based inference problems, our focus extends to incorporating the temporal dimension of such systems and scaling consistently existing inference methods with the size of the problem. Our goal is to infer the posterior density of states in dynamic systems, using observations to condition the inference process. By accounting for the temporal aspect, we can extend our understanding of the system's behavior and make informed predictions about its future states.

Eventually, we show that existing estimation methods can adapt to our problem by incorporating consistently available information related to both system dynamics and observation process. We argue that convolutional estimators are needed to allow good scaling without increasing excessively computational costs. By leveraging system's structure, we found diffusion-based estimators being promising to solve our problem. We also highlight the need of new evaluation techniques that scales correctly and propose a classifier-based posterior check that fill the lacks of other classical evaluations at the cost of harder interpretation.

"He believed in the primacy of doubt, not as a blemish upon our ability to know, but as the essence of knowing. The alternative to uncertainty is authority, against which science had fought for centuries."[1]

James Gleick, *Genius: The Life and Science of Richard Feynman*

[1]Quote extracted from the book of Tim Palmer, *The Primacy of Doubt*, 2022

Acknowledgements

I am first thankful to my advisor Pr. Gilles Louppe for his strong following, invaluable advice and for introducing me to research field. Your guidance during my thesis has been crucial in shaping my understanding, my passion and my rigor. I also thank François Rozet for his assistance with the mathematical background of score-based models and his continuous support throughout the development phase. Our discussions have greatly enhanced my work.

I am grateful to Alice for her unwavering support, belief in me, and inspiration. I thank my family for their presence at every stage of my academic journey. I am grateful for your love, encouragement, and understanding.

Last but not least, to all my friends who have experienced the challenges and triumphs of this year with me. I am thankful for our shared hopes, concerns, and the wonderful years of study we have experienced together. Within this work, I would like to mention Jérôme for the enjoyable and relaxing coffee breaks we have shared.

To each of you, I extend my sincere gratitude for your contributions, support, love and friendship.

Table of Contents

1. Introduction

1.1. Problem statement

Complex dynamical systems are pervasive across scientific domains, encompassing phenomena such as fluid dynamics, atmospheric evolution, neuronal activity, and more. Models of those systems offer the ability to make predictions about their behavior, provided they accurately capture the underlying dynamics. However, due to the inherent chaotic nature of complex systems, making accurate predictions, especially far in the future, becomes a hard challenge.

In such systems, predictions about their evolution depends on how strong are the beliefs we have concerning the current state. The only way we can strengthen those beliefs is by reducing their uncertainty incorporating at best past and present system's related information. Once those beliefs are strong, we can propagate them through the model to estimate evolution of the system, which is relevant as long as this model is well-specified. The lower the uncertainty, the longer the prediction remains valuable.

To address this problem, data assimilation techniques play an important role. Data assimilation (DA) combines observations with simulations to estimate and improve our knowledge of the system's current state and its evolution. By assimilating observed data into dynamical models, we can correct for the state's trajectory estimate and update our beliefs. Under a probabilistic point of view, the most natural way of formulating this problem is through the Baye's rule

$$p(state \mid observations) = \frac{p(observations \mid state)p(state)}{p(observations)} \tag{1.1}$$

which describes how to update beliefs modelled by a density $p(state)$ given evidences in the form of observations. By accounting how likely are those observations given our beliefs using the likelihood $p(observations \mid state)$, we can update them, leading to what we call the posterior density $p(state \mid observations)$. This relation highlights the importance of having confident beliefs about the state in order to refine them consistently.

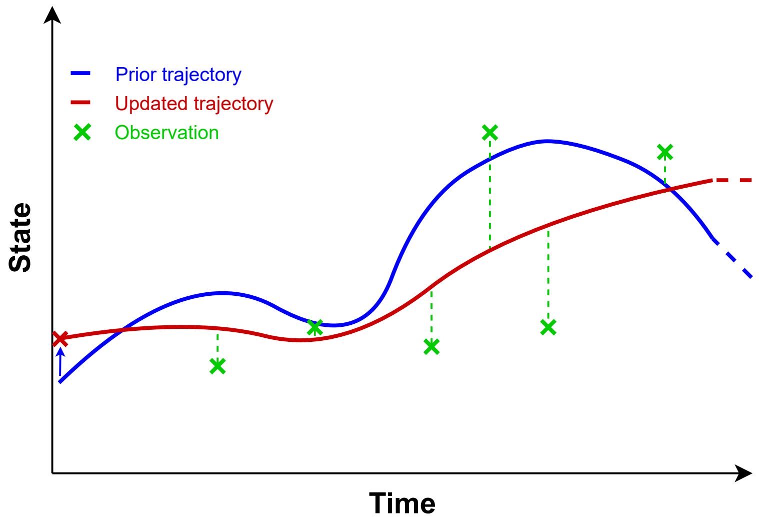


Figure 1.1.: Classical point-estimate DA pipeline. A prior forecast is performed based on previous observations. When fresh data is collected, the trajectory is corrected to better fit the received information.

By integrating observations and simulations through data assimilation, we bridge the gap between limited observations and the hard to predict chaotic dynamics, extracting the best of both. This approach acknowledges the inherent chaotic nature of complex systems via simulation and trades off forward prediction's accuracy with the periodic assimilation of fresh observations. It enables us to refine our estimators and improve predictions value for potential related decision making.

This thesis addresses the challenges of adapting the classical simulation-based inference framework to state-space models, considering issues of scalability and incorporation of temporal information. Our objective is to explore how existing methods can be modified and tailored to effectively handle state-space models. Benefiting from the advent of deep learning, we focus on modern density estimation techniques using artificial neural networks.

Through empirical evaluations and comprehensive analysis, this work aims to provide valuable insights into the effectiveness and limitations of adapting simulation-based inference techniques to DA. These insights can have significant implications for practical applications, where the integration of temporal information is crucial for accurate inference. The proposed method pipeline involves the generation of simulated data, which serves as ground truth, and the subsequent learning of a posterior model that is coherent with the underlying physics and assimilated observations. Eventually, the formulation of this problem as a posterior estimation (1.1) allows uncertainty quantification which is of high interest for practical decision making tasks related to the evolution of such physical systems.

2. Background

2.1. Simulation-based inference

This work focuses on the simulation-based inference [1] framework. Specifically, we use methods called neural posterior, ratio, and score estimation. It is important to note that these techniques are not exhaustive, and the field of simulation-based inference continues to evolve rapidly in various directions [2, 3]. It has found applications in a wide range of domains, including robotics [4], astrophysics [5], and healthcare [6], among others.

2.1.1. Bayesian inference

Bayesian inference is a statistical methodology that enables the updating of beliefs regarding the likelihood of parameters θ based on new observations x. Unlike other statistical approaches, Bayesian techniques consider all parameters as random variables. By incorporating prior assumptions $p(\theta)$ about the distribution of these parameters, one can update our beliefs through the likelihood $p(x \mid \theta)$ defined by a simulator, leading to the posterior $p(\theta|x)$. This belief's update resorts to the Bayes' rule

$$p(\theta \mid x^*) = \frac{p(x^* \mid \theta)p(\theta)}{p(x^*)}. \tag{2.1}$$

Here, x^* is a specific observation drawn from the evidence density $p(x)$. An important aspect of this rule is its ability to incorporate uncertainty into tests and analyses. This allows us to directly infer a probability density function over the parameters, which is often necessary for practical decision-making problems and various other applications. By considering the underlying uncertainty, we can ensure that decisions are made aware.

Traditional Bayesian methods, such as Markov Chain Monte Carlo (MCMC) and Approximate Bayesian Computation (ABC), have been widely used for estimating posterior densities [7–12]. However, these methods often lack amortization for observations. This means that the whole pipeline must be executed for each new observations resulting in high computational costs, especially when the system is expensive to simulate. With the advent of deep learning, neural density estimation has emerged as a promising approach that has the potential to circumvent many issues of classical density estimation (DE) methods such as the curse of dimensionality and the lack of amortization. The integration of these techniques alongside simulation has led to the field of simulation-based inference (SBI) [1, 13].

The field of SBI regroups tools for bayesian inference when the likelihood function is either known (e.g. SLCP [14]), unknown or intractable (likelihood-free inference). Simulators are only models of real-world systems whose internal dynamics are complex and themselves modelled by unobserved latent variables z. Simulators only implicitly define the path from parameters θ to observations x through latent space $\mathcal{Z}$. In many real-world scenarios, obtaining a closed-form expression for the likelihood function is impractical or impossible as it would require to integrate over all possible trajectories across this latent space as

$$p(x \mid \theta) = \int_{\mathcal{Z}} p(x, z \mid \theta) dz \tag{2.2}$$

which becomes intractable when the latent space is large, as in most cases. Consequently, traditional inference methods like Maximum Likelihood Estimation (MLE) [15] or MCMC may not be directly applicable nor efficient. Furthermore, in scenarios where the evidence $p(x)$ is not accessible, the latter could be computed using

$$p(x) = \int_{\Theta} p(x \mid \theta) p(\theta) d\theta. \tag{2.3}$$

This makes the problem even harder to solve as it would require to compute a double integral on arbitrarily low to high dimensional spaces $\mathcal{Z}$ and Θ. The simulation-based inference field provides a toolbox of likelihood-free inference (LFI) methods that can handle such situations and have diverse applications.

2.1.2. Variational inference

Variational inference [16] (VI) is a methodology that approximates a posterior distribution $p(\theta \mid x)$ by a density estimator $q_\phi(\theta \mid x)$ parameterized by ϕ. The approximation takes the form of an optimization problem for which the objective is to fit ϕ in order to minimize a dissimilarity measure $d(p, q_\phi)$ between densities. The most common choice for such measure is the reverse Kullback-Leibler [17] (KL) divergence

$$\mathrm{KL}\big(q_\phi(\theta \mid x) \| p(\theta \mid x)\big) = \mathbb{E}_{q_\phi(\theta|x)}\left[\log \frac{q_\phi(\theta \mid x)}{p(\theta \mid x)}\right]. \tag{2.4}$$

In our context, neural networks are used to parameterize density estimators $q_\phi(\theta|x)$. Mathematically, the optimal parameters ϕ^* are the ones that satisfy

$$\begin{aligned}\phi^* &= \arg\min_\phi \mathrm{KL}\big(q_\phi(\theta \mid x) \| p(\theta \mid x)\big) \\ &= \arg\min_\phi \mathbb{E}_{q_\phi(\theta|x)}\left[\log \frac{q_\phi(\theta \mid x)}{p(\theta \mid x)}\right] \\ &= \arg\min_\phi \mathbb{E}_{q_\phi(\theta|x)}\big[\log q_\phi(\theta \mid x) - \log p(\theta, x) + \log p(x)\big] \\ &= \arg\max_\phi \mathbb{E}_{q_\phi(\theta|x)}\big[\log p(\theta, x) - \log q_\phi(\theta \mid x)\big] \end{aligned} \tag{2.5}$$

where we leverage the fact that $p(x)$ is constant with respect to ϕ. The problem finally turns to the maximization of a quantity (2.5) called the evidence lower bound objective (ELBO).

2.1.3. Estimation methods

Direct posterior estimation

Despite the difficulty that we can face while computing the posterior, there remains specific cases in which the latter can be properly computed. However, those techniques are rooted on strong assumptions that are hardly often encountered in real-world scenarios. The latter being mainly limited to low-dimensional systems for the reasons explained previously.

In this work, we implement a method that we call the direct posterior estimation (DPE). It allows to estimate the posterior for a fixed observation x^* by using (2.1) under the following assumptions:

1. We have access to the likelihood $p(x \mid \theta)$.
2. We have access to the evidence $p(x)$.
3. We can sample from $p(\theta)$.

Hence, given a sufficiently large number of samples $\theta_i \sim p(\theta)$, we can estimate the posterior using histograms [18]. Indeed, since the posterior is an update of the prior, it is expected to be contained in the support of the latter. Ensuring that we can easily cover the prior domain with sufficient number of samples (third assumption), we can approximate the posterior by weighting the contribution of each samples with their associated likelihood-to-evidence (LTE) ratio

$$r(x, \theta) = \frac{p(x \mid \theta)}{p(x)}. \tag{2.6}$$

Following the two first assumptions, we have access to the exact value of this ratio for any pair (x, θ) as it depends only on known densities. However using histograms limits the inference capacity of this technique as it suffers from the curse of dimensionality, meaning that the number of required samples to produce a consistent estimation grows exponentially with the dimension of the problem. Moreover, computing the LTE ratio needs to evaluate both the likelihood and evidence densities for each pair, leading to further computational costs if their evaluation is expensive.

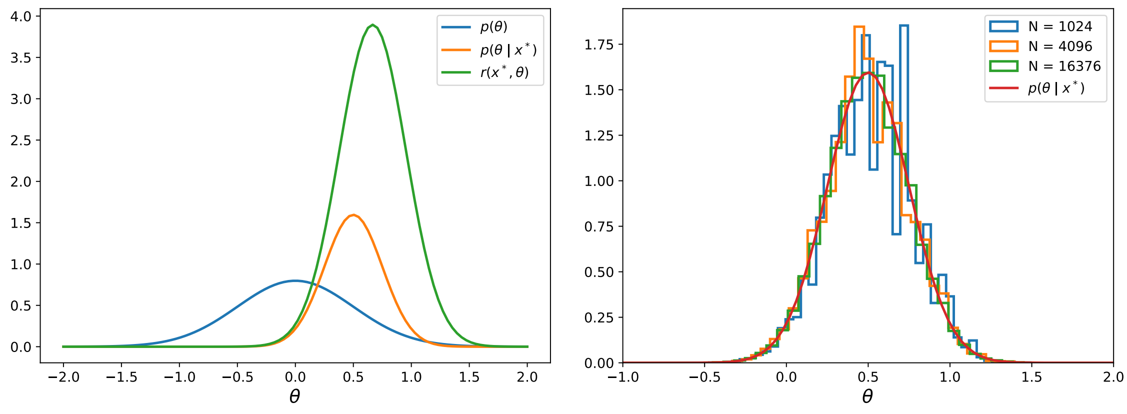


Figure 2.1.: DPE for $p(\theta) = \mathcal{N}(0, 0.5^2)$ and $p(\theta|x^*) = \mathcal{N}(0.5, 0.25^2)$. On the left, prior and posterior densities are depicted with the associated LTE ratio. On the right, posterior is estimated using DPE based on samples from the prior and LTE ratio weights. The latter plot illustrates the importance of sampling (N denotes the number of samples) as much as possible from the prior to consistently cover the whole support of the posterior.

The use of this method will be further discussed as it applies only to very restrictive setups.

Neural posterior estimation

Under the variational inference perspective, neural posterior estimation (NPE) aims to estimate the posterior directly using neural density estimators, denoted as $q_\phi(\theta|x)$.

We notice that (2.5) involves the computation of the expectation of the joint $p(\theta, x)$ w.r.t. the estimated density $q_\phi(\theta|x)$. In the context of SBI, the joint is typically unknown analytically. However, it remains obvious to sample from it as

$$p(x, \theta) = p(x \mid \theta)p(\theta). \tag{2.7}$$

Samples from the joint are drawn by sampling from the prior and then from the likelihood implicitly defined by the simulator.

To bypass the computation of the joint expectation, we aim at minimizing the forward KL divergence $\mathrm{KL}\big(p(\theta \mid x) \| q_\phi(\theta \mid x)\big)$. The latter is still minimal when both densities are identical but, since the KL is not symmetrical, it does not correspond to (2.4). In addition, we want our estimator to be amortized meaning that we minimize the expected KL over the observations. This allows to infer posterior for any observation x^* without the need of retraining the density estimator. The final objective is the minimization of

$$\begin{aligned} \mathbb{E}_{p(x)}\Big[\mathrm{KL}\big(p(\theta \mid x) \| q_\phi(\theta \mid x)\big)\Big] &= \mathbb{E}_{p(x)}\mathbb{E}_{p(\theta|x)}\left[\log \frac{p(\theta \mid x)}{q_\phi(\theta \mid x)}\right] \\ &= \mathbb{E}_{p(x,\theta)}\Big[\log p(\theta \mid x) - \log q_\phi(\theta \mid x)\Big] \end{aligned} \tag{2.8}$$

for which optimal parameters are

$$\phi^* = \underset{\phi}{\operatorname{argmin}}\ \mathbb{E}_{p(x,\theta)}\big[-\log q_\phi(\theta|x)\big]. \tag{2.9}$$

The density estimator thus minimizes its expected negative log-density (NLD) w.r.t. simulated data. The log posterior density is omitted from the objective as it is independent of the parameters ϕ. The objective is lower bounded but the minimum value is not known beforehand since it would require direct access to the posterior, which is the unknown we are estimating.

Neural ratio estimation

This method takes roots on the estimation of the LTE ratio (2.6) previously defined in the direct posterior estimation (DPE) section. Whereas DPE computes directly the LTE ratio, neural ratio estimation (NRE) [19] replace its computation by a neural estimator $r_\phi(x, \theta)$. NRE only requires to be able to sample from the joint $p(x, \theta)$ and the marginals $p(x), p(\theta)$ for training and only requires access to an explicit prior density $p(\theta)$ for sampling.

The ratio $r(x,\theta)$ can be estimated by training a neural classifier $d_\phi(x,\theta)$ to discriminate between pairs (x,θ) drawn from the joint $p(x,\theta)$ and the product of the marginals $p(x)p(\theta)$. The optimization problem can be formulated using a suitable loss function, such as the cross-entropy. In this context, the optimization problem takes the form of

$$\phi^* = \arg\min_\phi \mathbb{E}_{p(x,\theta)}\big[-\log(d_\phi(x,\theta)\big] + \mathbb{E}_{p(x)p(\theta)}\big[-\log(1-d_\phi(x,\theta))\big]. \tag{2.10}$$

Here, $d_\phi(x,\theta)$ is the classifier that outputs the probability of a given pair (x,θ) to belong to the joint distribution. As proven by Hermans, Begy, and Louppe [20], the Bayes optimal classifier for this task, denoted as $d_{\phi^*}(x,\theta)$, is

$$d_{\phi^*}(x,\theta) = \frac{p(x,\theta)}{p(x,\theta)+p(x)p(\theta)}. \tag{2.11}$$

Combining (2.6) and (2.11), we have

$$r(x,\theta) = \frac{p(x \mid \theta)}{p(x)} = \frac{p(x,\theta)}{p(x)p(\theta)} = \frac{d_{\phi^*}(x,\theta)}{1-d_{\phi^*}(x,\theta)} = r_{\phi^*}(x,\theta). \tag{2.12}$$

Therefore, the Bayes optimal classifier gives access to the true LTE ratio $r(x,\theta)$. Using MCMC sampling procedures like Metropolis-Hastings [21], one can sample from the estimated posterior distribution $p(\theta)r_\phi(x,\theta)$. In addition, since the classifier gives access to an estimate of the LTE ratio, the density can be approximated in a similar fashion than the DPE method, by weighting samples of the prior.

The neural ratio estimation method simplifies the problem of posterior approximation to a binary classification task without imposing constraints on the neural network architecture. However, it has some limitations, such as the requirement of MCMC sampling procedures and the need for access to the prior density $p(\theta)$

Neural score estimation

Neural Score Estimation (NSE) [22] is a recent approach grounded in the theory of Score-Based Models (SBM) [23]. While the mathematical foundations of diffusion models are not latest discovery [24], their popularity has surged due to recent advancements, demonstrating their remarkable performance on various tasks [25–29].

Diffusion model[1]. The objective of diffusion models is to learn a generative process that can produce samples from a given distribution using stochastic differential equations (SDE) as depicted in Figure 2.2. Those models are linked to variational models through the concept of hierarchical variational auto-encoders [30] (HVAE). Especially, continuous diffusion model is a degenerate case of HVAE with an infinite number of hierarchical latents [31].

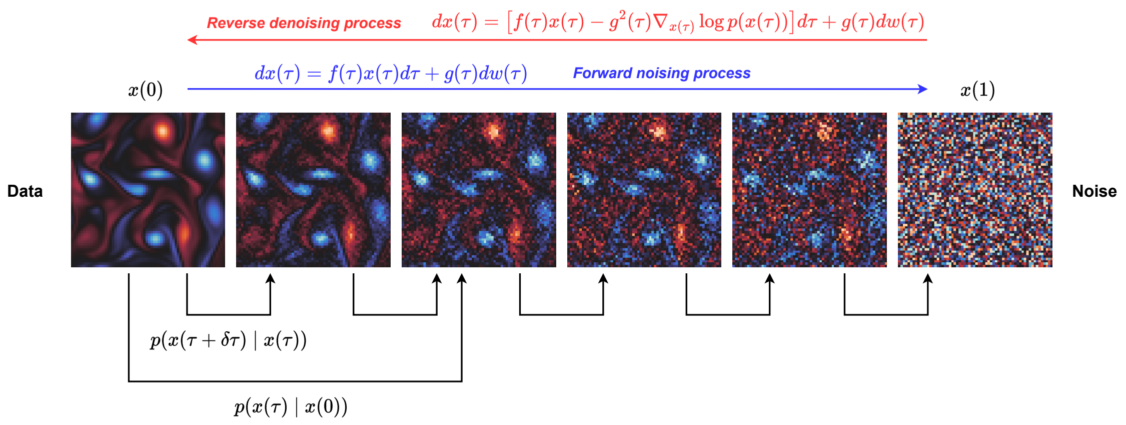

Reverse denoising process
$dx(\tau) = [f(\tau)x(\tau) - g^2(\tau)\nabla_{x(\tau)} \log p(x(\tau))]d\tau + g(\tau)dw(\tau)$
$dx(\tau) = f(\tau)x(\tau)d\tau + g(\tau)dw(\tau)$
Forward noising process
$x(0)$
$x(1)$
Data
Noise
$p(x(\tau + \delta\tau) \mid x(\tau))$
$p(x(\tau) \mid x(0))$

Figure 2.2.: Forward continuous diffusion process (blue) and its associated generative reverse process (red).

Formally, the forward process

$$dx(\tau) = f(\tau)x(\tau)d\tau + g(\tau)dw(\tau)$$

defines a continuous mapping between a data distribution $p(x) \approx p(x(0))$ and a reference distribution $p(x(1))$ (corresponding to pure isotropic Gaussian noise) through an SDE of parameters f and g. Under the HVAE perspective, the process follows fixed Gaussian transitions

$$x(\tau) = \sqrt{\alpha_\tau} x(\tau - \delta\tau) + \sqrt{1 - \alpha_\tau} \mathcal{N}(0, I) \tag{2.13}$$

scheduled by the diffusion time τ and assuming infinitesimal step $\delta\tau$. The hyperparameter α_τ is fixed and characterizes the type of diffusion process. Following [31], the latter is related to the parameter functions $f(\tau)$ and $g(\tau)$.

[1]Much of the theoretical background draws from the Deep Learning course of the University of Liège given by Pr. Gilles Louppe. **Link** : `https://github.com/glouppe/info8010-deep-learning`

Developing recursively (2.13) down to $\tau = 0$ leads to

$$\begin{aligned} x(\tau) &= \sqrt{\alpha_\tau} x(\tau - \delta\tau) + \sqrt{1 - \alpha_\tau}\mathcal{N}(0, I) \\ &= \sqrt{\alpha_\tau \alpha_{\tau-\delta\tau}} x(\tau - 2\delta\tau) + \sqrt{1 - \alpha_\tau \alpha_{\tau-\delta\tau}}\mathcal{N}(0, I) \\ &= \dots \\ &= \sqrt{\beta_\tau} x(0) + \sqrt{1 - \beta_\tau}\mathcal{N}(0, I) \end{aligned} \tag{2.14}$$

with $\beta_\tau = \prod_{i=0}^{\frac{\tau}{\delta\tau}} \alpha_{i\delta\tau}$. In the continuous limit $\delta\tau \to 0$, (2.14) defines the diffusion kernel

$$p(x(\tau) \mid x(0)) = \mathcal{N}(\mu(\tau)x, \sigma^2(\tau)I) \tag{2.15}$$

where $\mu(\tau)$ and $\sigma(\tau)$ are generalized continuous diffusion scheduling hyper-parameters related to the previously defined continuous forward process.

Under the variational inference paradigm, diffusion models aim at estimating the tractable denoising posteriors $p(x(\tau - \delta\tau)|x(\tau), x)$. The latter is known analytically since the whole process follows Gaussian transitions. In this study, we adopt the score matching interpretation of diffusion models as outlined in [31], which provides a unified perspective on SBM and diffusion models.

Under the score matching interpretation, the goal is to learn the denoising process through the reverse SDE, using a neural network $s_\phi(x(\tau), \tau)$ as a score estimator. The score of $p(x(\tau) \mid x)$ is formally defined as $\nabla_{x(\tau)} \log p(x(\tau) \mid x)$. As depicted in the aforementioned figure, if one has access to this score, given the forward SDE parameters $f(\tau)$ and $g(\tau)$, it becomes possible to simulate the reverse SDE and sample from an approximate of the true data generating process. Such an estimator can be optimized by minimizing the denoising score matching loss

$$\mathbb{E}_{\tau \sim \mathcal{U}[0,1]} \mathbb{E}_{p(x(\tau), x)} \left[\lambda(\tau) \| s_\phi(x(\tau), \tau) - \nabla_{x(\tau)} \log p(x(\tau) \mid x) \|_2^2 \right] \tag{2.16}$$

This loss function represents a continuous weighted combination of Fisher divergences. The latter derives directly from the KL objective between the tractable posterior and the estimate. Due to the Gaussian nature of the process, the KL is known analytically and resumes to simple mean matching. Complete derivations can be found in [31].

Developing for the score of the diffusion kernel, we have

$$\begin{aligned}
\nabla_{x(\tau)} \log p(x(\tau) \mid x) &= \nabla_{x(\tau)} \log \mathcal{N}(\mu(\tau)x, \sigma^2(\tau)I) \\
&= \nabla_{x(\tau)} \log \left[\frac{1}{\sigma(\tau)\sqrt{2\pi}} exp\left(\frac{-(x(\tau) - \mu(\tau)x)^2}{2\sigma^2(\tau)} \right) \right] \\
&= -\nabla_{x(\tau)} \log \sigma(\tau)\sqrt{2\pi} - \nabla_{x(\tau)} \left(\frac{(x(\tau) - \mu(\tau)x)^2}{2\sigma^2(\tau)} \right) \\
&= \frac{\mu(\tau)x - x(\tau)}{\sigma^2(\tau)}. \qquad (2.17)
\end{aligned}$$

To degrade the data smoothly, we choose a consistent scheduling for μ and σ that transitions from $(\mu(0), \sigma(0)) = (1, 0)$ to $(\mu(1), \sigma(1)) = (0, 1)$. The specific choice we made is detailed in **Appendix A.1.**.

A commonly used weighting scheme is $\lambda(\tau) = \sigma^2(\tau)$. By incorporating it into (2.16) and substituting the expression of the target score (2.17), we obtain the following rescaled score loss

$$\mathbb{E}_{\tau \sim \mathcal{U}[0,1]} \mathbb{E}_{p(x(\tau),x)} \left\| \sigma(\tau) s_\phi(x(\tau), \tau) - \frac{\mu(\tau)x - x(\tau)}{\sigma(\tau)} \right\|_2^2. \qquad (2.18)$$

To improve optimization stability, an alternative network parameterization is proposed, introducing rescaled score matching [31]. This involves defining the rescaled score function as $\epsilon_\phi(x(\tau), \tau) = -\sigma(\tau) s_\phi(x(\tau), \tau)$. Notably, the rescaled score of the diffusion kernel follows a standard Gaussian distribution. Thus, the final form of (2.18) is

$$\mathbb{E}_{\tau \sim \mathcal{U}[0,1]} \mathbb{E}_{p(x)} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)} \left\| \epsilon_\phi(\mu(\tau)x + \sigma(\tau)\epsilon, \tau) - \epsilon \right\|_2^2. \qquad (2.19)$$

Sampling via the reverse SDE. Once the parameterized score function ϵ_ϕ is learned, we can sample from the approximate data distribution $p(x(0))$ by solving the reverse SDE. This can be achieved using either a classical SDE solver or by solving the associated probability flow ODE [23]. In this work, we adopt the exponential-integrator (EI) scheme proposed by Zhang and Chen [32], inspired by the deterministic DDIM sampling procedure [33]. Specifically, when using the variance-preserving (VP) SDE, the sampling procedure is given by

$$x(\tau - \delta\tau) \leftarrow \frac{\mu(\tau - \delta\tau)}{\mu(\tau)} - \left(\frac{\mu(\tau - \delta\tau)}{\mu(\tau)} - \frac{\sigma(\tau - \delta\tau)}{\sigma(\tau)} \right) \sigma(\tau) \epsilon_\phi(x(\tau), \tau). \qquad (2.20)$$

Diffusion models for SBI. Diffusion models have been introduced generally for any distribution of data denoted x. When we use them under the simulation-based inference framework, the density we are interested in is the posterior $p(\theta \mid x)$. What we called data in diffusion models corresponds to θ here. As a first approach, diffusion models can be adapted to directly approximate the posterior by incorporating context into the score network as

$$s_\phi(\theta(\tau), \tau, x) \approx \nabla_{\theta(\tau)} \log p(\theta(\tau) \mid x). \tag{2.21}$$

However, according to the Bayes' rule (2.1)

$$\begin{aligned}\nabla_\theta \log p(\theta \mid x) &= \nabla_\theta \log \frac{p(x \mid \theta)p(\theta)}{p(x)} \\ &= \nabla_\theta \log p(x \mid \theta) + \nabla_\theta \log p(\theta)\end{aligned} \tag{2.22}$$

where we leverage the fact that the evidence score w.r.t. θ is zero. This leads to an alternate method allowing for learning marginal score and likelihood score independently. In practice, we will see that the latter could be approximated for our problem, simplifying the training only to the marginal score estimation but trading-off between samples quality and sampling speed.

2.2. Data assimilation

Inverse problems [34] involve identifying likely variables or parameters of a system given observed data. Belonging to this class of problems, data assimilation [35, 36] aims to estimate the states of a system based on partial observations. Data assimilation (DA) is commonly used in fields like numerical weather prediction (NWP) where recent observations are assimilated to refine state estimates and improve the accuracy of inferred trajectories. In this context, posterior density estimation could be used to quantify uncertainty and provide a more comprehensive understanding of the variability of system's states, unlike classical point estimates. The goal would then not only to improve accuracy of simulated trajectories but also to reduce uncertainty we have about them.

2.2.1. State-space models

State-space models[2] (SSM) are a powerful framework for modeling and analyzing dynamic systems. They consist of two key components: a transition process that represents the latent states dynamics of the system, and an observation process that links the latent states to the observed data. We introduce the following notations

- t is the simulation time, indexing the states and observation through the system's dynamics.
- $x_t \in \mathbb{R}^N$ is the state vector of the system at time t.
- $y_t \in \mathbb{R}^M$ is the observation vector related to the state at time t.
- $\mathcal{M}(.) : \mathbb{R}^N \mapsto \mathbb{R}^N$ represents $p(x_{t+1} \mid x_t)$ known as the transition model.
- $\mathcal{O}(.) : \mathbb{R}^N \mapsto \mathbb{R}^M$ is the observation process defining $p(y_t \mid x_t)$.

In this work, we focus on deterministic transition models

$$x_{t+1} = \mathcal{M}(x_t)$$

governed by a system of ordinary differential equations (ODE). On the other hand, we assume linear Gaussian observation process

$$\mathcal{O}(x_t) = \mathcal{N}(A_t x_t, \sigma_y^2 I)$$

that models uncertainty in the measures. A_t stands for a linear operator which maps states to observations whereas σ_y is a presumed known and shared noise intensity among observations. To further simplify the problem, we use a fixed observation process, simplifying the notation of the operator A_t to A as it will not be indexed by time. The observation process being time invariant, we will denote it as

$$p(y_t \mid x_t) = p(y \mid x, t) = p(y \mid x) = \mathcal{N}(Ax, \sigma_y^2 I). \tag{2.23}$$

2.2.2. Posterior estimation formulation

In this work, we shift our focus from traditional maximum a posteriori (MAP) estimation

$$MAP(y_t) = \arg\max_{x_t} p(x_t \mid y_t) \tag{2.24}$$

[2]This notation is used in the literature mainly for systems with linear dynamics. In this work, we use it to denote any dynamical system whose dynamics are governed by a set of differential equations.

to the direct estimation of the posterior density. This approach enables us to make probabilistic predictions that quantify the uncertainty associated with state estimates and avoids being trapped in local maxima corresponding to specific modes of the posterior distribution.

The classical data assimilation framework aims to infer a posterior trajectory based on a prior assumption on the actual trajectory and new sequence of observations. In this context, the target posterior is given by

$$p(x_{t-T:t} \mid y_{t-T:t}) = \frac{p(y_{t-T:t} \mid x_{t-T:t})p(x_{t-T:t})}{p(y_{t-T:t})} \tag{2.25}$$

with $T-1$ equal to the assimilation window width. Equation (2.25) resorts to the Bayes rule (2.1). The likelihood is implicitly defined by the observation process $\mathcal{O}(.)$ whereas $p(x_{t-T:t})$ is defined by the simulator whose transitions are modelled by $\mathcal{M}(.)$.

In this work, we focus on the filtering task. The latter is defined as the assimilation of previous observations $y_{t-T:t}$ to update our belief of the current state x_t. Therefore, our objective is to approximate the posterior density $p(x_t|y_{t-T:t})$ with an estimator $q_\phi(x_t|y_{t-T:t})$. This objective is not always suitable for practical data assimilation problems since it does not explicitly take into account the evolution of the system. The posterior approximation only targets a single time step rather than a sub-trajectory. In this work, single step assimilation ($T = 0$) will act as a baseline, which can be viewed as learning a stochastic inverse observer indexed by time. On the other hand, the same problem with a window of size 10 ($T = 9$) will be studied. It is expected that the second problem will yield sharper posterior, as the wider context incorporates information about the underlying dynamics of the states.

2.2.3. Related work

One widely used approach in data assimilation is the Kalman filter [37], which provides an optimal estimate of the system state by recursively updating the state prediction based on observations and dynamical model. The optimality holds for linear dynamics and observations perturbed by Gaussian noise. However, many real-world systems exhibit nonlinear and non-Gaussian behaviors, which motivates the use of more advanced techniques. As a consequence, non-linear methods such as the extended Kalman filter and particle filter [38] have emerged as powerful tools for data assimilation in more general settings.

Another class of data assimilation techniques is based on variational methods [39], such as the 4D-Var and 3D-Var [40, 41] methods. Variational methods provide a computationally efficient approach to estimate the state of the system and can handle nonlinearity and non-Gaussianity to some extent.

4D-Var (Four-Dimensional Variational Data Assimilation) is a widely used technique that aims to estimate the system state by finding the best fit (maximum a posteriori (MAP)) between model predictions and observations over a specific time window, typically referred to as the assimilation window. It is particularly suited for systems with strong temporal dependencies, such as weather and ocean models. Such method is the keystone of current used techniques at the European Centre for Medium-Range Weather Forecasts (ECMWF) for 25 years [42, 43]. The main idea behind 4D-Var is to formulate data assimilation as an optimization problem. The goal is to minimize the cost function

$$J(x_0) = (x_0 - x^b)^T B^{-1}(x_0 - x^b) + \sum_{t=0}^{T} (Ax_t - y_t)^T \Sigma_y^{-1} (Ax_t - y_t) \tag{2.26}$$

to fit the initial state x_0 to x^b (called the background, with its associated covariance matrix B) and the collected observations y_t over a time window of width T. By simulating forward via the transition model $\mathcal{M}$ starting from x_0, the objective aims at reducing the assumed Gaussian error between reconstructed observations Ax_t and corresponding collected y_t. Indeed, (2.26) is equivalent to a maximum likelihood estimation under Gaussian assumptions, leading to a sum of squared errors.

While such method provides unique estimate of the state, often targeting the one that maximizes the posterior density, it is not always sufficient. Conversely, estimating the whole posterior allows for criticizing the inference by quantifying how uncertain we might be about any point estimate. The former sometimes leads to easier estimation process but the latter is way more valuable for practical decision making. It is even more crucial when the decision can have dramatic consequences as in autonomous driving to name just one among many.

Nowadays, machine learning techniques have also been applied to data assimilation problems, leveraging the power of neural networks to learn complex relationships between observations and system states. These data-driven approaches have shown promising results in improving the accuracy and efficiency of data assimilation. Frerix et al. [44] reformulates (2.26) in the state space by learning an inverse observation process that maps y_t to an estimate of the corresponding state x_t. Mack et al. [45] and Amendola et al. [46] suggest to formulate the optimization problem in the latent space of a learned variational auto-encoder allowing for faster optimization at the cost of loss of physical guarantee for the reconstructed states. Rozet et al. [47] address the data assimilation problem under the formalism of score-based models. By leveraging the Markovian structure of dynamical systems, they propose a method that allows for zero-shot trajectory posterior sampling applicable to any known differentiable observation process $\mathcal{A}$. As data assimilation techniques are primarily used for forecasting purposes, with meteorology being a prominent field of application, it is worth mentioning the work of Ravuri et al. [48] and Lam et al. [49].

Overall, estimation techniques in the data assimilation field play a crucial role in combining observational data with models to estimate the state of complex systems. The choice of the appropriate method depends on the system characteristics, availability of data, and computational resources. Advancements in this field continue to enhance our ability to make accurate predictions in various scientific disciplines.

2.3. Estimators

2.3.1. Conditional normalizing flows

Normalizing flows [50] (NF) are a family of neural density estimators that leverage the change of variable formula in probability theory

$$\begin{aligned} z &= f(x) \\ p(x) &= p(z)\left|\det \frac{\partial f(x)}{\partial x}\right|. \end{aligned} \tag{2.27}$$

By selecting a simple base distribution $p(z)$ that is easy to sample from, and choosing a suitable learnable transformation function $f : \mathcal{X} \mapsto \mathcal{Z}$ parameterized by a network of parameters ϕ, one can estimate any distribution with sufficient capacity in the neural network. Normalizing flows are recognized as universal density estimators [51] provided that their transforms are not uniquely affine. However, the main challenge lies in determining the appropriate form of the transformation function f. To enable sampling an estimate of $p(x)$, it is necessary to define a mapping $g : \mathcal{Z} \mapsto \mathcal{X}$. Since NF uses transformations throughout which dimension is preserved, a straightforward loss-less mapping is given by

$$x = g(z) = g(f(x)) = f^{-1}(f(x)) = x$$

which implies that g is the inverse of f.

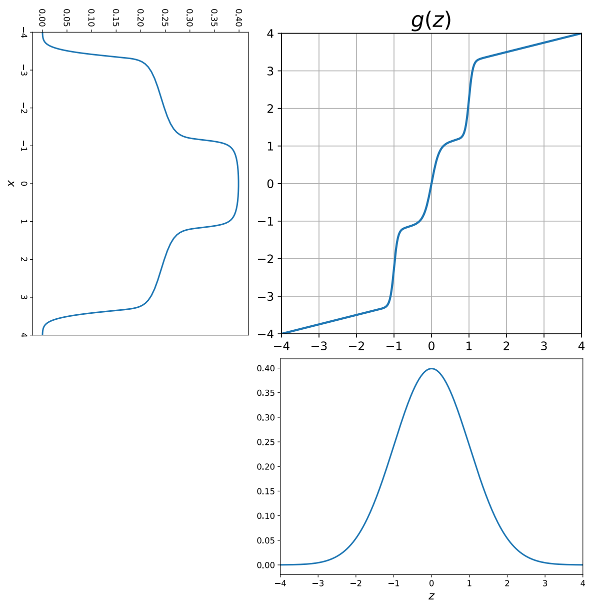


Figure 2.3.: Example of bijective transform from a base distribution $p(z) = \mathcal{N}(0, 1)$.

The function f can be any complex bijective function. With neural networks, we can construct invertible layers that allow f to be expressed as a composition of simple bijective functions f_i

$$f = f_N \circ f_{N-1} \circ ... \circ f_2 \circ f_1.$$

Each f_i represents a transformation that maps latents z_{i-1} to z_i, where $z_0 = x$ and $z_i = f_i(z_{i-1})$. The Jacobian $\frac{\partial f}{\partial x}$ can be computed by the chain rule as the product of the Jacobian of each individual transforms

$$\frac{\partial f}{\partial x} = \prod_{i=1}^{N} \frac{\partial f_i(z_{i-1})}{\partial z_{i-1}}. \tag{2.28}$$

The determinant of the Jacobian of this composed transform is obtained by taking the product of the determinants of the Jacobians of the successive sub-transforms f_i. This formulation allows us to express the log density of our estimator, using

(2.27)

$$\log p(x) \approx \log q_\phi(x) = \log p(z) + \sum_{i=1}^{N} \log \left| \det \frac{\partial f_i(z_{i-1}, \phi)}{\partial z_{i-1}} \right|. \qquad (2.29)$$

In summary, NF offer several advantages and drawbacks. On one hand, they provide direct access to the estimated density, they allow sampling in a single inverse pass (starting from latent $z \sim p(z)$ and reconstructing through g) and they are recognized as universal density estimators. On the other hand, they require tailored bijective transforms with restrictive constraints, those transforms are all defined in the data dimension which makes scaling hard and the NLD optimum is not known beforehand.

Eventually, in order to implement a conditional normalizing flow to estimate conditional densities, one must adapt (2.29) by making the transforms context-dependent. This is easily achieved by introducing the context as an input of the neural network that parameterize each transform. While it is also possible to make the base density $p(z)$ context-dependent using, for example, $z \sim \mathcal{N}(\mu_\phi(y_{t-T:t}, t), \Sigma_\phi(y_{t-T:t}, t))$, we will not incorporate this in our work.

Masked auto-regressive flows

The baseline model that we will consider is a Masked Auto-regressive Flow (MAF) with an embedding network. Masked Autoregressive Flows were introduced by Papamakarios, Pavlakou, and Murray [52] and belong to the category of autoregressive flows. Autoregressive refers to a transformation from an input vector x to a latent representation z of the form

$$z_i = f(x_i, x_{<i})$$

with $x_{<i}$ denoting every element of x that have a smaller index than i. This transformation is based on the concept that any joint distribution of variables $x \in \mathbb{R}^N$ can be expressed as $p(x) = \prod_{i=1}^{N} p(x_i | x_{1:i-1})$. In practice, a masked auto-regressive transform is an affine transformation of the form

$$f(x_i, x_{<i}) = \frac{x_i - \mu_i(x_{<i})}{\sigma_i(x_{<i})}$$

where μ_i and σ_i are unconstrained parameterized functions of $x_{<i}$. The functions μ_i and σ_i are implemented by masking a simple feed-forward network, which involves setting the connections between μ_i, σ_i and $x_{\geq i}$ to 0.

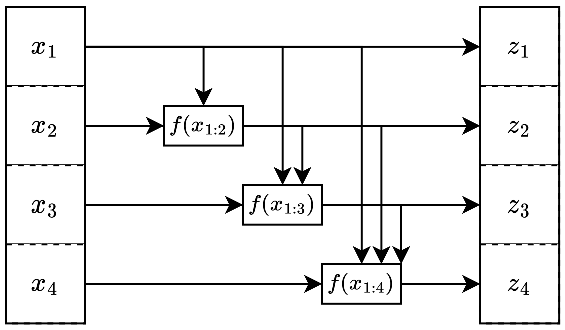


Figure 2.4.: Example of auto-regressive transform for a state of size 4. Since $z_1 = x_1$, one can easily inverse the transform by reverting the horizontal arrows direction since $f(x_i, x_{<i})$ is affine w.r.t x_i.

To enhance the expressivity of the flow, multiple autoregressive transforms are stacked together, and the inner latent representations z^k are internally permuted before each subsequent layer to allow for variable mixing along the transforms.

In practice, we need to condition our transforms in order to approximate the posterior distribution. In order to estimate the posterior, we condition each transform by directly incorporating an embedding of the context into the parameterized functions μ_i and σ_i, making the transforms both auto-regressive and context-dependent.

Convolutional flows

Our convolutional flow architecture is inspired by classical convolutional normalizing flows [53–55], where the main component is a series of blocks with convolutional transforms. As discussed in the theoretical part, the transforms within our flow should be easily invertible, with simple log absolute Jacobian determinant computations. To achieve this, we combine ideas from Glow [54], Emerging [55], and CInC flows [56][3]. However, designing invertible convolutions presents its challenges.

Convolution layers differ from traditional linear layers as they involve weight sharing between features. However, a single convolution kernel can still be unrolled into a matrix multiplication with the flattened input. The resulting convolution matrix corresponds to a sparse Toeplitz circulant matrix [58] characterized by the associated kernel. Hence, we can express a convolution as

$$y = Ux \tag{2.30}$$

[3]In this work, we have not implemented the recently proposed faster-to-invert convolutional flows [57].

where y is the flattened layer's output, U the convolution matrix and x the flattened input.

Starting from (2.30), our goal is to design a learnable kernel K such that the corresponding matrix U is invertible. Additionally, computing the determinant of U should be straightforward since it will be equal to the Jacobian determinant of the convolutional transform. In normalizing flows, it is crucial to preserve the dimension of the state throughout the transforms. For convolutional transforms, this necessitates padding the input to maintain the dimension. To address this, [56] proposed padding the input x in a way that U is always triangular (or at least block-triangular), as shown in Figure 2.5.

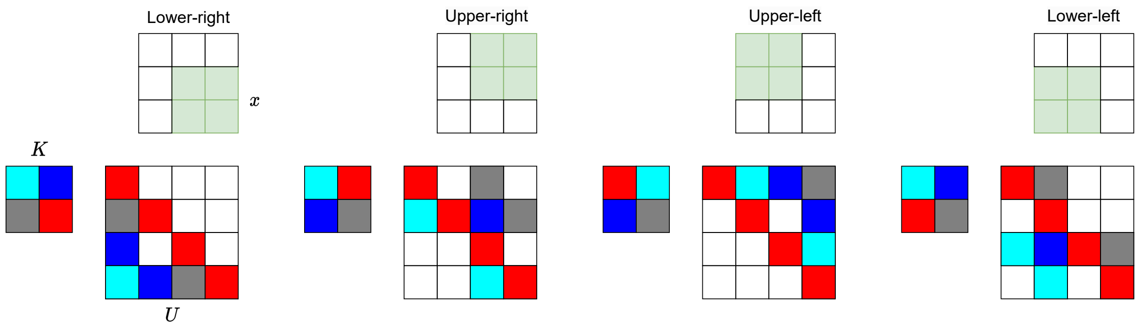


Figure 2.5.: Example of convolution matrix for the four padding orientations proposed by [56]. Convolution kernel K is applied to padded state x with a stride of 1.

Triangular Jacobian transforms are commonly used in normalizing flows because their determinant is easy to compute and is equal to the product of the diagonal elements. The padding procedure described above allows us to have a block-triangular matrix. The determinant of a block-triangular matrix is equal to the product of the determinants of the diagonal blocks which are themselves triangular. Therefore, we have a transform with an easy-to-compute Jacobian determinant, regardless of the selected padding orientation. However, ensuring the invertibility of our transform requires the existence of U^{-1}. A square matrix is not singular if and only if its determinant is non-zero. From Figure 2.5, we observe that

- When the matrices are triangular, the diagonal elements are all the same. Thus, the determinant is the red kernel element r to the power N (N being the size of the vector x).
- When the matrices are block-triangular, the diagonal blocks are themselves triangular, with the diagonal elements consisting solely of the red kernel element.

Hence, the determinant is also r^N.

We've intentionally placed the red kernel element in the corresponding location of the state in the padded input. Therefore, if we ensure that the red element corresponding to the selected padding mode is non-zero, the determinant of U will be non-zero, U^{-1} will exist, and the determinant of the associated Jacobian will be r^N. We subsequently mask our kernels such that $r = 1$, resulting in

$$\log |\det(J_f)| = \log |\det(U)| = N \log r = 0.$$

In conclusion, we have a suitable convolutional transform for normalizing flows with the only constraint being the presence of a fixed element in the kernel, while the remaining elements are learned. During training, we compute the forward pass using the standard convolution module with the appropriate masked kernel. For sampling, we compute the inverse by deducing the convolutional matrix U from the learned masked kernel, as depicted in Figure 2.5. Our convolutional flow architecture consists of a main block called the multi-scale convolutional transform, as shown in 2.6. The transform is composed of separate blocks, which are detailed in **Appendix A.4.**.

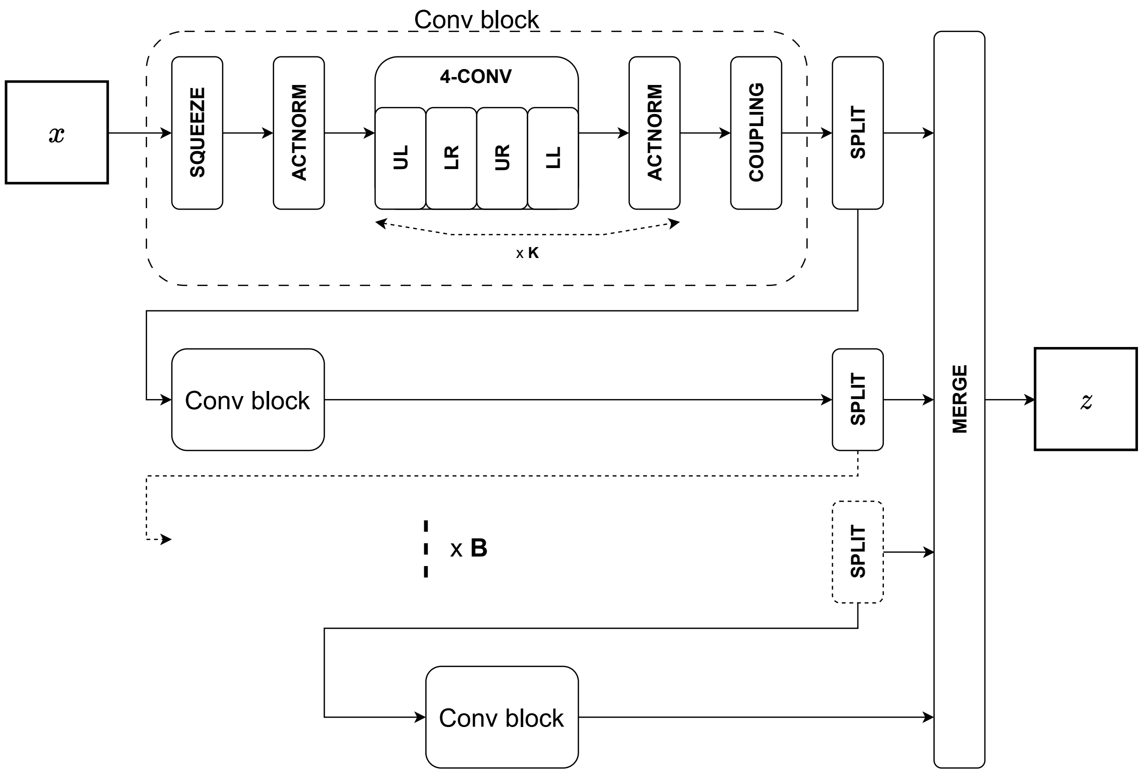


Figure 2.6.: Full multi-scale convolutional transform.

Similar to MAF, these blocks are stacked, and the latent variable z is permuted after each block. The context is incorporated into the convolutional blocks through the 4-conv and coupling transforms. The 4-conv block uses context-dependent convolution kernels, while the coupling functions take the context as input.

2.3.2. Conditional score estimators

Score-based models provide flexibility in choosing the network architecture for estimating the score. While UNet-like backbones [59] are commonly used, recent advancements, such as Diffusion transformers (DiT) [60], based on the transformer architecture [61], have also shown promising results.

In our work, we adopt an Attention UNet [62] architecture, which incorporates an additive attention mechanism into the skip connections (Figure 2.8) of the classical UNet. To ensure the score-based models are indexed by the diffusion time τ, we incorporate time encoding at each scale of the extraction branch. This encoding aligns the extracted features with the desired diffusion time. The overall architecture is illustrated in Figure 2.7.

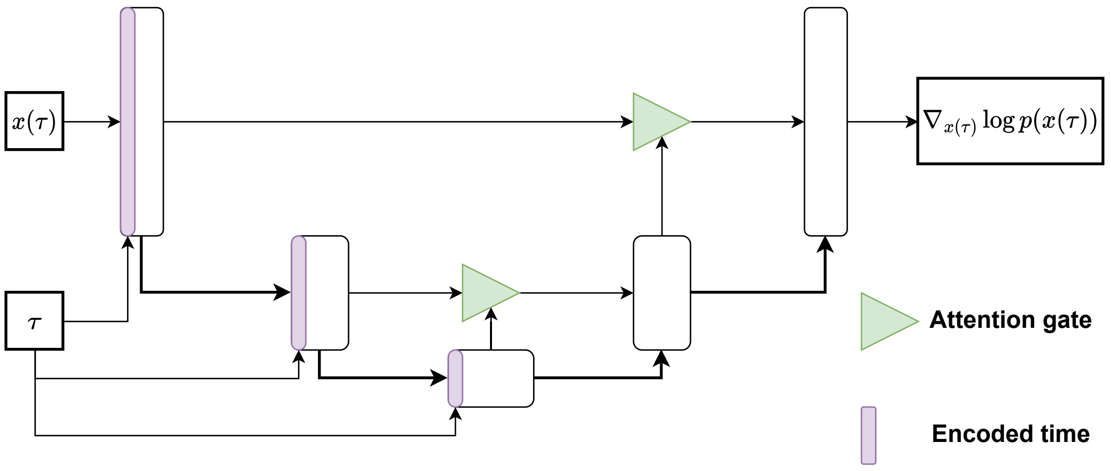


Figure 2.7.: Abstract schematic of the Attention-UNet architecture with diffusion time encoding

It is important to note that Figure 2.7 does not exhibit the context. However, this schematic is applicable for marginal score estimation. In the case of learning the posterior score, the embedded context is concatenated with the input of the network, allowing for the use of the same base architecture for both PS and CS where CS is a degenerated case where the context consists only of t.

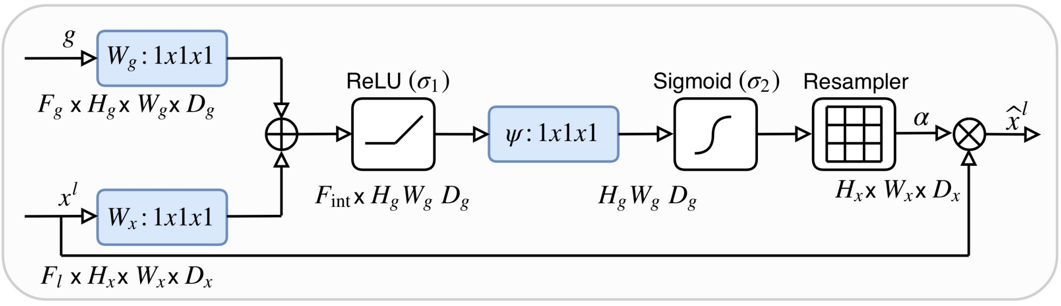


Figure 2.8.: Attention mechanism implemented in skip-connections and proposed by Oktay et al. [62]. The attention mask, applied to current scale data x^l is computed using signal g at coarser scale (previous stage in the UNet). Both are embedded to fit dimensions and then added. Computed importance map is passed through a sigmoïd, ensuring that each pixel of the original data is weighted by an individual attention gate $\in [0, 1]$.

3. Inference in state-space models

Simulation-based inference (SBI) methods have demonstrated their effectiveness in various scientific domains. However, their application to dynamical systems has not been explored yet. Classical SBI focuses on inferring fixed parameters of a simulator, whereas data assimilation deals with continuously evolving states in a state-space model. To establish a connection between classical SBI and state-space model problems, we can consider the task of filtering as a posterior estimation. Table 3.1 illustrates the mapping between the naming conventions in SBI and data assimilation for state-space models (SSM).

SBI	DA in SSM	Meaning
x	y	Observation(s)
θ	x	Parameter(s)/State(s)
\	t	Time

Table 3.1.: Correspondence between naming conventions in SBI and DA

In classical SBI, the goal is to approximate the posterior distribution of parameters given observations, denoted as $p(\theta|x)$ whereas the filtering objective is to estimate the time-varying posterior distribution of states given observations $p(x_t|y_{t-T:t})$ that we will denote as $p(x|y_{t-T:t}, t)$.

In the following sections, we address three key differences between classical SBI and SBI for state-space models: the time-varying posterior, problem scaling, and evaluation methods. Our adaptations aim to enhance the applicability and effectiveness of SBI techniques for SSM.

Time-varying prior

In the context of inferring states from dynamical systems, it is essential to consider the time-varying nature of the state marginal density, denoted as $p(x_t)$ or $p(x \mid t)$.

Since

$$p(x_t|y_t) = \frac{p(x_t, y_t)}{p(y_t)} = \frac{p(y_t|x_t)p(x_t)}{\int p(y_t|x_t)p(x_t)dx_t},$$

the posterior is therefore likewise time-dependent. Unlike classical SBI, which assumes fixed parameters throughout the simulation, our estimator need to be indexed by time to accurately estimate the posterior distribution at each time step. Our posterior estimators are denoted $q_\phi(x \mid y_{t-T:t}, t)$.

Problem scaling

Traditional SBI methods are often not well-suited for high-dimensional posterior estimation. We have already mentioned that NPE relies on normalizing flows that model a series of transforms in the dimension which could become computationally expensive at high dimension. State-space models, with their inherent structural complexity, can quickly lead to high-dimensional states. To address this challenge, we make use of specific architectures such as convolutional networks. We want to extract meaningful features from the state structure. By incorporating structure in our networks, we ensure that our estimator can efficiently handle the increasing dimensionality of the problem while maintaining expressive and scalable representations.

Evaluation

Endowed with a multitude of evaluation and diagnostic methods, the field of SBI already offers valuable tools [63]. However, given the challenges posed by time-varying posteriors and high-dimensional state-space models, conventional evaluation techniques may not always be appropriate. Therefore, we propose a classifier-based evaluation method that takes into account the specification of the problem and the observation process. This approach allows us to accurately assess the performance of our generative models within the context of SSM.

3.1. Direct posterior estimation

Adapting previously introduced DPE technique to our problem, we target the posterior

$$p(x \mid y_{t-T:t}, t) = \frac{p(y_{t-T:t} \mid x, t)}{p(y_{t-T:t} \mid t)} p(x \mid t) = r(x, y_{t-T:t}, t)p(x \mid t) \tag{3.1}$$

As a reminder, we aim at approximating the posterior by weighting the contribution of samples from the marginal $p(x \mid t)$ with the corresponding LTE ratio we assumed known. The hypotheses we made to compute this ratio must adapt to our problem.

1. *We have access to the likelihood.* In our setup, the likelihood is $p(y_{t-T:t} \mid x, t)$. The latter is known analytically (2.23) for $T = 0$ which corresponds to the one-step assimilation problem.
2. *We have access to the evidence.* In the DA posterior formulation, the evidence is $p(y_{t-T:t} \mid t)$ that we can approximate with histograms when the problem is not too big and when we can easily sample window of observations.
3. *We can sample from the prior.* The latter corresponds to $p(x \mid t)$ from which it is straightforward to sample via simulation.

In a nutshell, this technique will only be used for small problems where the evidence approximation is faithful. In addition, we will limit to single step assimilation since the likelihood is well defined in this case.

3.2. Neural posterior estimation

Adapting the NPE objective (2.8) to our targeted posterior, we have the optimal parameters of the estimator $q_\phi(x \mid y_{t-T:t}, t)$

$$\phi^* = \underset{\phi}{\operatorname{argmin}} \ \mathbb{E}_{p(x, y_{t-T:t} \mid t)} \big[-\log q_\phi(x \mid y_{t-T:t}, t) \big] \tag{3.2}$$

such that $q_{\phi^*}(.) \approx p(x \mid y_{t-T:t}, t)$ as defined in (3.1). The expectation over the joint is estimated by Monte-Carlo leveraging

$$p(x, y_{t-T:t} \mid t) = p(y_{t-T:t} \mid x, t) p(x \mid t).$$

This corresponds to sampling from the simulated marginal $p(x \mid t)$ and then the associated observation process. In practice, commonly used density estimators q_ϕ are normalizing flows. To incorporate the context $(y_{t-T:t}, t)$ into the normalizing flow, we construct a conditional normalizing flow [64]. This is achieved by making the transforms context-dependent. The final formulation of the objective (2.29) for state posterior estimation with conditional normalizing flows is

$$\begin{aligned} \log p(x \mid y_{t-T:t}, t) &\approx \log q_\phi(x \mid y_{t-T:t}, t) \\ &= \log p(z) + \sum_{i=1}^{N} \log \left| \det \frac{\partial f_i(z_{i-1}, \phi, y_{t-T:t}, t)}{\partial z_{i-1}} \right|. \end{aligned} \tag{3.3}$$

3.3. Neural ratio estimation

Adapting (2.12) to our problem, we have

$$r(x, y_{t-T:t}, t) = \frac{p(y_{t-T:t} \mid x, t)}{p(y_{t-T:t} \mid t)} \approx r_\phi(x, y_{t-T:t}, t) \tag{3.4}$$

as in DPE (3.1). Then, we have to learn a classifier network that will provide an estimate of the LTE ratio via (2.11).

One condition for sampling from the estimated posterior is to have access to the prior $p(x \mid t)$. Nevertheless, the latter is not known analytically in our setup since it would require to propagate the initial state prior $p(x \mid t = 0)$ along the simulation which depends on complex non-linear dynamics. In practice, we can use a density estimator for the marginal (with histograms for example) and use NRE for posterior estimation. Nevertheless, this would mean that we depend on another approximation that could make the posterior estimation worse. Moreover, as for the evidence in DPE, histograms struggle at estimating higher dimensional densities as it would require an exorbitant number of samples to cover sufficiently the state space. Regarding those limitations, we decided to not use this technique in our work.

3.4. Neural score estimation

Returning to (2.21) and (2.22), our objective is to estimate the perturbed posterior score in order to sample from the estimated posterior $p(x(0) \mid y_{t-T:t}, t)$ that we generally denote as $q_\phi(x \mid y_{t-T:t}, t)$. In this thesis, our focus is on learning the score function

$$s_\phi(t, y_{t-T:t}, x(\tau), \tau) \approx \nabla_{x(\tau)} \log p(x(\tau) | y_{t-T:t}, t). \tag{3.5}$$

This allows us to sample from the approximated posterior $q_\phi(x|y_{t-T:t}, t)$ using the reverse SDE sampling procedure described in (2.20). Developing (3.5), we have

$$\begin{aligned}
\nabla_{x(\tau)} \log p(x(\tau)|y_{t-T:t}, t) &= \nabla_{x(\tau)} \log \frac{p(x(\tau), y_{t-T:t}, t)}{p(y_{t-T:t}, t)} \\
&= \nabla_{x(\tau)} \log \frac{p(x(\tau)|t)}{p(x(\tau)|t)} \frac{p(x(\tau), y_{t-T:t}, t)}{p(y_{t-T:t}, t)} \\
&= \nabla_{x(\tau)} \log p(x(\tau)|t) \frac{p(t)p(x(\tau), y_{t-T:t}, t)}{p(x(\tau), t)p(y_{t-T:t}, t)} \\
&= \nabla_{x(\tau)} \log p(x(\tau)|t) \frac{p(y_{t-T:t}|x(\tau), t)}{p(y_{t-T:t}|t)} \\
&= \nabla_{x(\tau)} \log p(x(\tau)|t) + \nabla_{x(\tau)} \log p(y_{t-T:t}|x(\tau), t). \qquad (3.6)
\end{aligned}$$

Equations (3.5) and (3.6) are adaptation of (2.21) and (2.22) to our assimilation problem. We therefore have two methods to approximate the perturbed posterior score, each with their advantages and drawbacks.

3.4.1. Posterior score learning

To learn the conditional score of the perturbed posterior (3.5), we follow a similar approach as in [65]. The estimator network $s_\phi(.)$ takes inputs composed of the perturbed state $x(\tau)$, the diffusion time τ, the context window $y_{t-T:t}$, and the context time t. The estimated scores at each diffusion time $\tau \in [0, 1]$ are conditioned on the complete posterior context $(y_{t-T:t}, t)$.

Once the conditional score estimator is learned, we can sample from the approximate posterior using the reverse SDE sampling procedure (2.20). However, this method relies on the observation process and lacks flexibility in incorporating new observations. The contribution of the context is expected to decrease as τ increases, reflecting the diffusion of the perturbed state into independent noise.

3.4.2. Composed score learning

The second method explores the decomposition (3.6) of the conditional score into the marginal score and the perturbed likelihood score (PLS). By leveraging the Bayes' rule and the fact that the evidence score with respect to $x(\tau)$ is zero, the posterior score can be expressed as the sum of the marginal score and the PLS. We then face a two-fold task. Firstly, we need to learn the marginal score, which can be done using score matching theory and a suitable network. Secondly, we need to address the estimation or approximation of the PLS.

Learning the PLS directly using score matching does not offer significant advantages over learning the posterior score directly. It requires training a new score estimator network when the observation process changes, which can introduce instability in the estimation. Alternatively, some have proposed guided score learning [66, 67], but those techniques are not well-suited for our problem as they may assume categorical likelihood density. Hence, we explore other approaches to address the challenge of learning the PLS.

To approximate the PLS without relying on an additional estimator network, we adopt the approach proposed in [68]. This approach assumes knowledge of the observation process $p(y_{t-T:t}|x,t)$, which takes the form of

$$y_{t-T:t} = \mathcal{A}_t(x) + \eta. \tag{3.7}$$

Here, $\mathcal{A}_t$ represents a differentiable operator mapping the state x to the observation window $y_{t-T:t}$ at time t, and η is an independent additive Gaussian noise whose density is $\mathcal{N}(0, \sigma_y^2 I)$. As explained in (2.23), the observation process for a single step is known to be linear and time invariant. We then get rid of time indexing and use the notation $\mathcal{A}$.

Since $\mathcal{A}$ is defined for a single step, we do not have access to the mapping between a state and a window of previous observations as it would require to simulate backward in time. Therefore, the following technique is only used for the single-step assimilation problem ($T = 0$), where the observation process is well-defined. In future developments related to this method, we will use $y_{t-0:t} = y_t = y$. Combining the assumptions made previously, we obtain the likelihood function

$$p(y|x) = \mathcal{N}(\mathcal{A}(x), \sigma_y^2 I). \tag{3.8}$$

To approximate the PLS, Chung et al. [68] proposed the expression

$$p(y|x(\tau)) \approx \mathcal{N}(\mathcal{A}(\hat{x}(x(\tau))), \sigma_y^2 I) \tag{3.9}$$

for the perturbed likelihood. Here, $\hat{x}(x(\tau))$ represents the denoising posterior mean, given by the Tweedie's formula

$$\hat{x}(x(\tau)) = \frac{x(\tau) + \sigma^2(\tau)\nabla_{x(\tau)} \log p(x(\tau))}{\mu(\tau)}. \tag{3.10}$$

The derivation and explanation of the Tweedie's formula can be found in **Appendix A.2.**.

Using the analytical expression for the score of a Gaussian distribution, the approximation for the PLS is expressed as

$$\begin{aligned}\nabla_{x(\tau)} \log p(y|x(\tau)) &\approx \nabla_{x(\tau)} \log \mathcal{N}(\mathcal{A}(\hat{x}(x(\tau))), \sigma_y^2 I) \\ &= -\frac{1}{2\sigma_y^2} \nabla_{x(\tau)} \big(y - \mathcal{A}(\hat{x}(x(\tau)))\big)^2. \end{aligned} \tag{3.11}$$

Since the approximation (3.10) can be obtained by replacing the noisy marginal score with our score network, the entire PLS can be approximated by following (3.11) and backpropagating through the network.

Nevertheless, there are some remaining issues with the previous approximations. When the ratio $\sigma(\tau)/\mu(\tau)$ is large, the stability of the approximations (3.10) and (3.11) is compromised. Rozet and Louppe [47] argue that this instability arises due to the significant variance of $p(x|x(\tau))$, which should be taken into account in (3.9). The likelihood should have a decreasing contribution as τ approaches 1, indicating the transformation of the perturbed data into pure noise. To address this, they propose an adaptation of (3.9) as

$$p(y|x(\tau)) \approx \mathcal{N}\big(\mathcal{A}(\hat{x}(x(\tau))), (\sigma_y^2 + \gamma \frac{\sigma^2(\tau)}{\mu^2(\tau)})I\big)$$

In this revised approximation, the variance of the perturbed likelihood changes over diffusion time. The parameter γ controls the importance of the noising process in the likelihood score and is set to 10^{-2} in this work. The effect of this additional term along diffusion time is discussed in **Appendix A.1.**.

These approximations may generate low-quality samples if the estimated score deviates significantly from the true score, leading the approximated reverse SDE (2.20) to follow a trajectory that deviates from the desired distribution. To address this, a correction inspired by Predictor-Corrector (PC) sampling is applied between each prediction step, using one step of Langevin Monte Carlo (LMC) [23]. The LMC correction step is given by

$$\begin{aligned}
\varepsilon &\sim \mathcal{N}(0, I) \\
\lambda &= \|s_\phi(x(\tau), \tau)\|_2^2 \\
x'(\tau) &= x(\tau) + \frac{\kappa}{\lambda} s_\phi(x(\tau), \tau) + \sqrt{2\frac{\kappa}{\lambda}}\varepsilon
\end{aligned} \tag{3.12}$$

The parameter κ and the number of predictions and corrections are not discussed in this work, but their values are set to $\kappa = 0.5$, 100 predictions, and 1 correction, respectively.

4. Experiments

We conduct a series of experiments that can be categorized into different groups, as depicted in Figure 4.1. We consider two distinct problems that are detailed in the **Simulators** and **Observers** sections. The code used to perform those experiments is available at `https://github.com/gerome-andry/dasbi.git`.

For each problem, we train estimators for two tasks. The one-step task with $T = 0$ serves as a baseline where temporal information is not available. The window assimilation task with $T = 9$, which involves ingesting observations that provide spatio-temporal information about the state is more akin to a classical assimilation problem. It is expected to yield sharper posteriors due to the additional contextual information it ingests compared to the baseline task.

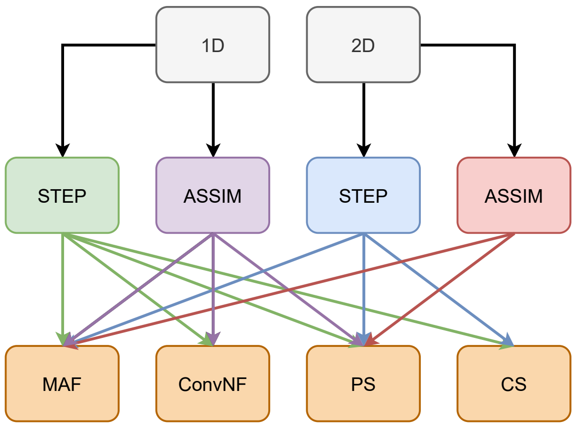


Figure 4.1.: Implemented estimators for each problem and method. The STEP task stands for $T = 0$ and the ASSIM one for $T = 9$.

We consider four types of estimators, each with its own specific applications.

- Normalizing flows
  - **MAF** : Masked autoregressive flow
  - **ConvNF** : Convolutional normalizing flow

- Score-based estimators
  - **PS** : Posterior score estimator
  - **CS** : Composed score estimator

4.1. Simulators

We consider two models in this work, the one-dimensional Lorenz 96 [69] and two-dimensional turbulent flow models. The choice of these SSM have been inspired by [44, 47] works. We have chosen these systems for their chaotic dynamics, which make them suitable for studying methods on complex problems. They offer a cheaper simulation cost compared to global climate models [70] (GCMs) and are commonly used for validating data assimilation techniques.

4.1.1. Lorenz-96

The one-dimensional Lorenz 96 model is described by a system of ODEs given by:

$$\frac{dx^i}{dt} = -x^{i-1}(x^{i-2} - x^{i+1}) - x^i + F \qquad i \in \{1, 2, ..., N\}$$

This system represents an advection-diffusion process with external forcing F. The state x consists of N points arranged on a circular domain as illustrated in Figure 4.2. The value of the forcing term F is set to 8, which is a commonly used value known to induce chaotic behavior. Further details about the experiments and problem sizes can be found in the **Appendix B.1.**.

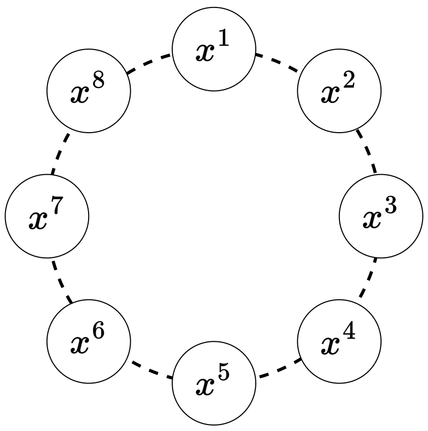


Figure 4.2.: Lorenz96 problem's domain for $N = 8$.

Since the Lorenz 96 system is one-dimensional, directly looking at a state x can be difficult to interpret. In addition to the circular representation shown in Figure 4.2, we can also use Hovmöller diagrams [71], which are widely deployed in meteorology to visualize the evolution of a state over time (see Figure 4.3).

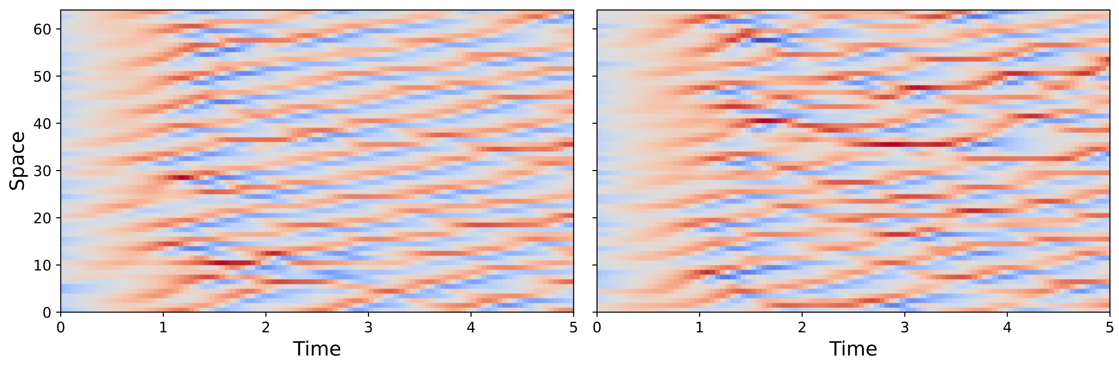


Figure 4.3.: Hovmöller diagrams for two simulations over 100 steps for $t \in [0, 5]$, $N = 64$, $x_0 \sim \mathcal{N}(0, I^{N \times N})$.

Data for training are generated on-the-fly using the `torchode` package [72] for ODE solver since the Lorenz 96 system is computationally inexpensive to simulate.

4.1.2. Two-dimensional turbulent flows

The two-dimensional Kolmogorov turbulent flow system is governed by the following equations

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} &= -(\mathbf{u}.\nabla)\mathbf{u} + \nu\nabla^2\mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{F} \\ 0 &= \nabla.\mathbf{u} \ . \end{aligned}$$

Here, $\mathbf{u}$ represents the velocity field, with components $\mathbf{u}_x$ and $\mathbf{u}_y$. Other variables include ν (fluid viscosity), ρ (fluid density), p (pressure field), and $\mathbf{F}$ (Kolmogorov forcing with linear damping).

Visualizing the velocity field directly is not suitable, so we focus on visualizing the vorticity of the velocity field instead, as shown in Figure 4.4. The vorticity, denoted by ω, is computed as the curl of the velocity field

$$\omega \triangleq \left(\frac{\partial \mathbf{u}_y}{\partial x} - \frac{\partial \mathbf{u}_x}{\partial y}\right). \tag{4.1}$$

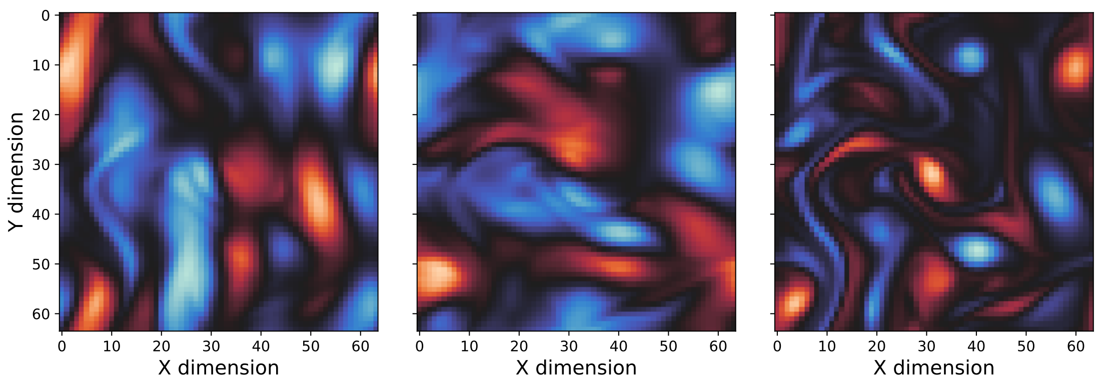


Figure 4.4.: Velocity field projections $\mathbf{u}_x$, $\mathbf{u}_y$, and corresponding vorticity (from left to right) for state's domain of size 256x256 coarsened to 64x64.

For this system, we gather data from [47], which corresponds to simulations of spatial size 256x256. After collection, we coarsen the states spatially to a size of 32x32 to lighten the problem in terms of computational expenses at the cost of a loss of resolution in the simulation.

4.2. Observation process

In this work, we build what we call a 2D-stations observer (2SO) to generate observations for assimilation. The 2SO is inspired by meteorology and represents a network of stations placed within the state's domain. Each station is randomly located in sub-regions of size $m \times n$, with axial minimal distances of d_x and d_y between each of them. The sensing capability of each station is modeled using a two-dimensional Gaussian kernel centered at the station's location. This kernel is truncated in both directions (by s_x and s_y units of the domain), limiting the region of the state that the station can sense. The power of sensing in each direction is controlled by parameters l_x and l_y, which correspond to the directed standard deviation of the Gaussian kernel.

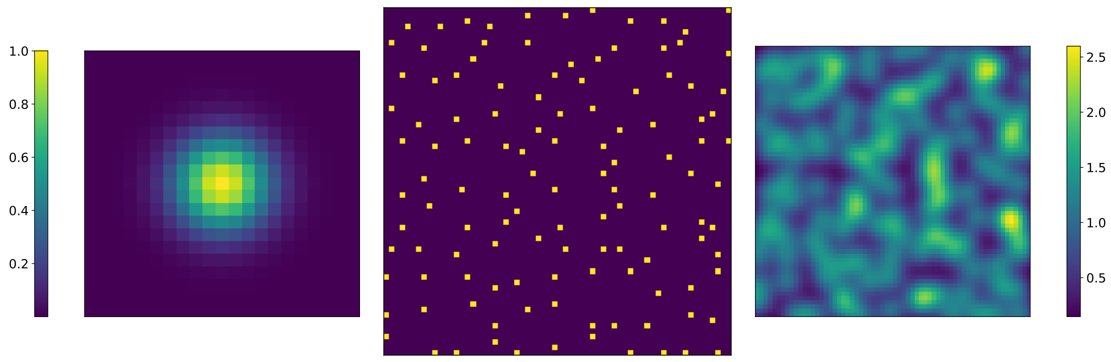


Figure 4.5.: Gaussian kernel (left) with power of sensing $l_x = l_y = 2.5$ and field of view $s_x = s_y = 10$. Random positioned stations (center) in a domain of size 64x64 in sub-regions of size 5x5 with a minimal separating distance $d_x = d_y = 1$. Corresponding importance map (right) of each state's points in the observation when the kernel is applied to each station.

The application of multiple Gaussian kernels to the state can be viewed as a linear operator A applied to the flattened state vector x. In practice, Gaussian noise is independently added to each measurement to account for uncertainty of the stations. This results in observations $y \sim \mathcal{N}(Ax, \sigma_o I)$, where σ_o represents the intensity of the measurement noise, which is shared across all stations. It is worth noting that the 2SO is differentiable with respect to the state x, and the operator A is a specific case of operator $\mathcal{A}$ in Equation (3.9). Furthermore, we have $\sigma_y \equiv \sigma_o$.

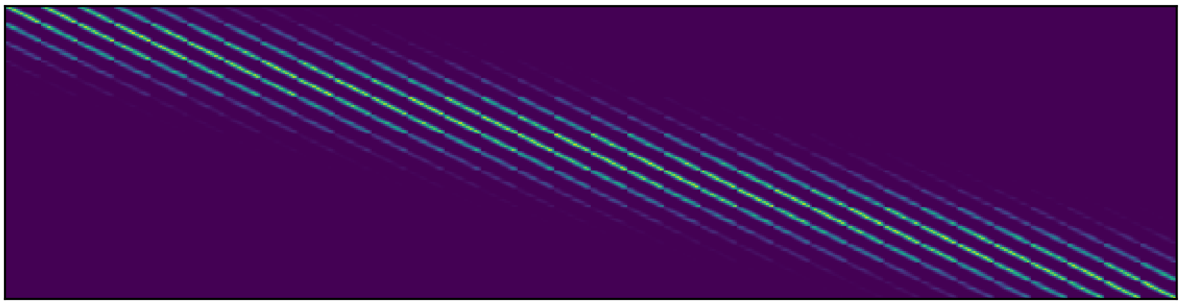


Figure 4.6.: Example of a station observer linear operator A^T, mapping a flattened state of 1024 points to an observation of size 256.

4.3. Architectures

This section aims at reviewing used architectures for each method we investigate in this work. The implementation of normalizing flows are greatly inspired from the `zuko`, `lampe` and `nflows` [73] packages [1]. For score-based methods, we adapted the implementation of Rozet and Louppe [47] to our task.

The main approach is to initially use a masked auto-regressive flow (MAF) as a baseline. In this naive implementation, the inputs are flattened and processed through masked feed-forward networks, resulting in a large number of parameters to learn. This serves as a reference for comparison. To further explore NF and SBM methods, we consider convolutional architectures [74], taking advantage of the structural properties of the SSM. Indeed, it is worth noting that SSM often exhibit local and multi-scale interactions between different parts of the state. Conversely to the baseline flow, we aim to limit the number of parameters in those architectures to maintain reasonable computational and training times while still generating challenging results. Hopefully, convolutional networks allow for weight sharing across spatial dimensions, reducing the number of parameters while preserving expressiveness.

In general, we use an embedding network for the context (y_T, t) which is structured similarly across all methods. In our problem, it is of interest to discriminate between nodes in a state. So it is crucial to encode the positions within the state, as convolutions are translation equivariant. Spatial encoding is incorporated to enable the networks to preserve spatial information, and a station mask related to the observation process is also included. The embedding network is fully convolutional and maps the context to an embedding with multiple channels, matching the spatial shape of the state.

Additionally, we take advantage of the periodic nature of the state's domains by using circular padding where applicable. This approach ensures that the convolutions operate seamlessly across the boundaries of the state, preserving their inherent periodicity.

By incorporating these design choices, we introduce inductive biases that align with the underlying structure of the selected models, enabling our estimators to better capture the relationships and patterns within the data, which supposedly lead to improved generalization and performances.

[1]`zuko` : https://zuko.readthedocs.io/en/stable/
`lampe` : https://lampe.readthedocs.io/en/stable/

Classifier architecture

Implementing a classifier in our context can be useful for NRE, if we had used the method, but also for the proposed evaluation technique detailed in the following section. The classifier architecture we will use for the posterior evaluation draws inspiration from classical convolutional architectures such as LeNet [75], AlexNet [76], and VGG [77], among others. The architecture, depicted in Figure 4.7, is made of

1. A double CNN block that extracts features from the sample and its context separately.
2. A shared CNN that combines the extracted features from the previous stage and reduces the dimension using pooling layers[2].
3. A final feed-forward block that recombines the low-scale features to output the class probability vector for the discriminator.

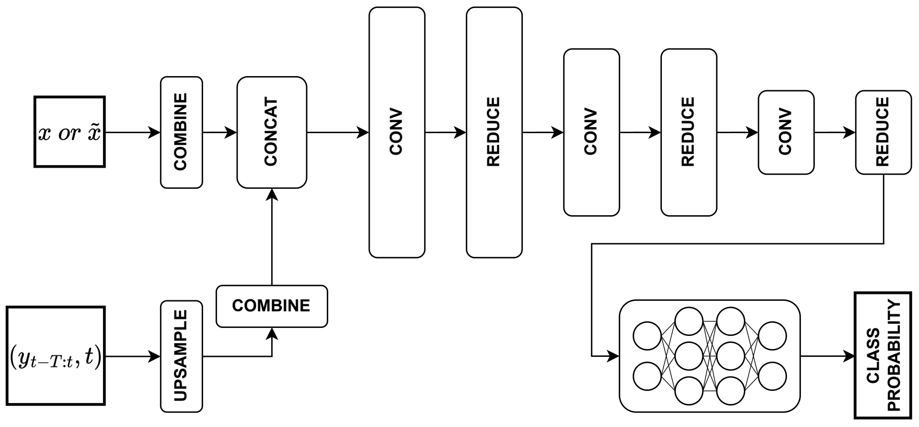


Figure 4.7.: Classifier abstract architecture for posterior check. Activation functions as well as encoding layers for time t are omitted for simplicity

4.4. Protocol

4.4.1. Training process

In our experimental setup, we use the 1D system as a toy problem to facilitate the comparison of different techniques. The size of the system scales geometrically from 8 to 256 nodes by powers of 2. On the other hand, the 2D system is designed to

[2]In practice, to maximize expressivity of our networks, pooling layers are implemented via strided convolutional layers.

challenge the limits of certain techniques and assess their performance at higher dimensions. In this bigger system, the states are of size 32×32 and consist of 2 channels, resulting in states with a dimension of 2048 points.

During the training process, the capacity of each model is adjusted proportionally to the size of the problem. To ensure fair comparisons between methods, we aim to maintain a relatively similar number of parameters for each model, except for the MAF baseline (see **Appendix B.2.**). However, it is important to note that the number of parameters alone is not sufficient for conducting fair comparisons between models. While we adhere to this criterion in our experiments, we acknowledge the need to assess models based on their complexity in time, sampling efficiency, training speed, and other relevant factors.

In practice, we exercise `Weights and Biases`[3] for managing the runs. Hyperparameters are not extensively fine-tuned for each method due to the computational cost of performing numerous runs. Each model is optimized based on its respective loss function and validation set, which is independent from the training set to prevent overfitting. The model selected at the end of training is the one that achieves the lowest validation loss. Additionally, it is worth mentioning that we conduct 5 independent training runs for each method and problem to assess the variability of their performance. Consequently, the presented results are averaged over these runs and accompanied by their respective standard deviation.

4.4.2. Evaluation methods

This section explores evaluation methods of simulation-based inference in the context of data assimilation and proposes a new diagnostic approach. We address the limitations of traditional evaluation procedures due to state size and temporal considerations. We present the advantages and drawbacks of each method.

Loss function

The loss function for flows estimators is the negative log-density (NLD) (3.2), while for the score-based models, it is the rescaled denoising score matching (RDSM) loss (2.18). It is not possible to directly compare the loss values between different models, as they are based on different objectives. The NLD loss is task-dependent and cannot be compared across different problem scales. The RDSM loss can potentially

[3]**Link** : `https://wandb.ai/gandry/dasbi?workspace=user-gandry`

reach a minimum of 0, no matter the considered problem.

In brief, the NLD loss will be comparable for a given problem (fixed scale, simulator, and observer) between NF methods (MAF and ConvNF). The RDSM loss will be comparable only for score-related methods (PS and CS), but it is expected to be larger for composed score estimation as it is related to the marginal state density for which wee have less information. Despite this lack of comparison between methods, it is important to mention that we can compute the log-likelihood of the samples generated from score-based models. While this does not directly apply to our methods, as we use a discretized sampling procedure, an approach is presented in **Appendix A.3.** to bridge the gap between normalizing flows and diffusion models.

Corner plot

A corner plot is a common visualization tool used in SBI to display the marginal density along paired joint distributions of each state's node. It is often used to compare approximated densities with MCMC estimations or known targets. However, in the context of state-space models, corner plots have limitations.

1. **SSM scale and dimension** : As the scale of the state increases, generating and interpreting histograms in the corner plot becomes computationally expensive and challenging to read. Furthermore those plots are designed for 1D data, while data assimilation problems generally involves higher-dimensional data. Using this tool for SSM may lack spatial expressiveness since states are considered as flat vectors.
2. **Time-varying posterior** : The posterior in SSM problems is varying over simulation time. Analyzing consistency requires evaluating a corner plot at every simulation time which is impractical.
3. **Sampling requirements** : Corner plots rely on histograms and require a large number of samples to show approximated densities. With the non-negligible state size in our setup, generating enough samples can be time-consuming.

Since we lack access to the target posterior, we will analyze our estimators along DPE results for the simplest problem. Indeed, both corner plots and DPE technique have limitations regarding our setup which makes the analysis hard and costly. As a consequence, corner plots will only be displayed for the 1D problem of smallest size $N = 8$ at a random picked simulation time.

Qualitative inspection

The qualitative inspection involves visually comparing samples from our generative models with ground truth data to assess coherence and physical consistency. While it does not provide quantitative metrics, it is a crucial step in evaluating the samples.

For the **1D problem**, samples at given simulation time are compared to the ground truth that led to the conditioning context, focusing on problem of size 32. This allows us to check the coherence between the ground truth and the samples. We delegate inspection of larger problems to other evaluation techniques.

For the **2D problem**, a few samples are examined alongside the ground truth. Although the limited number of samples may not cover a sufficient region of the posterior, this visual examination provides insights into physical consistency. Mean and standard deviation of the samples could also be considered to understand the variation of the estimated states.

Qualitative inspection provides valuable insights into the coherence of samples with physics and observations. However, further evaluation techniques are necessary for quantifying performances of our generative models.

Discriminator network

The discriminator network evaluation is inspired by GANs [78]. It consists in training a classifier to distinguish between real data and artificial states sampled from our estimators. The underlying idea is that, the better the classifier discriminates, the less plausible are the generator samples.

In this approach, the task is reduced to a binary classification problem. To evaluate the performance of the classifier, we use the area under the curve (AUC) metric, which represents the area under the receiver-operating curve [79] (ROC) of the classifier. The AUC is interpreted as the probability that a positive sample (simulated data) will be assigned by the classifier a higher probability of being positive than negative (sample from the estimated posterior). The AUC metric provides a quantitative measure of the classifier's ability to distinguish between real and artificial data.

The ROC curve (shown in Figure 4.8) opposes the true positive rate (TPR) against the false positive rate (FPR) for the decision function modeled by our classifier d_ψ. It is constructed by evaluating the classifier's confusion matrix (Table 4.1) at different decision thresholds.

	Predicted positive (PP)	Predicted negative (PN)
Positive (P)	True positive (TP)	False negative (FN)
Negative (N)	False positive (FP)	True negative (TN)

Table 4.1.: Confusion matrix

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{TN + FP}.$$

The TPR represents the proportion of correctly classified positive samples (true positives) over all actual positive samples, while the FPR represents the proportion of incorrectly classified negative samples (false positives) out of all actual negative samples.

To construct the ROC curve, we assign predicted labels to the samples based on a given threshold $p_{th} \in [0, 1]$. If the decision function value for a provided sample exceeds the threshold, it is labeled as positive (P), otherwise as negative (N). By varying the threshold, we obtain different TPR and FPR values, which are used to plot the ROC curve. In our evaluation, we approximate the ROC curve using the trapezoidal rule, integrating over 1000 points corresponding to uniformly spaced thresholds in the range [0,1].

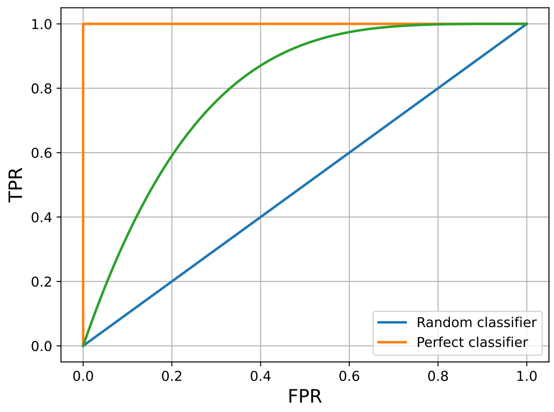


Figure 4.8.: ROC curve for different types of classifier. The orange ROC characterizes a perfect classifier (AUC = 1) whereas the blue curve represents the ROC of a random classifier (AUC = 0.5). Any classifier than performs better than a random classification is expected to have a ROC comprised between those two extreme cases, as for the green ROC.

As explained, the AUC is interpreted as a probability value that ranges from 0 to 1. A perfect classifier would have an AUC of 1, indicating that it can accurately distinguish between real data and samples from the estimated posterior. On the other hand, a random classifier would have an AUC of 0.5, implying that the classification is not better than random labelling as depicted in Figure 4.8.

In this work, we aim for the classifier to achieve an AUC close to 0.5. This indicates that the classifier cannot distinguish between the real data and the samples from the estimated posterior, which is a desirable outcome. To evaluate the performance of our models, we will conduct two types of posterior checks.

1 - Posterior check

The posterior check involves sampling pairs $(x, y_{t-T:t}) \mid t$ from both the simulated and estimated state-observations joint.

- **Positive samples (P)** : $(x, y_{t-T:t})|t \sim p(x, y_{t-T:t}|t) = p(y_{t-T:t}|x, t)p(x|t)$.
- **Negative samples (N)** : $(\tilde{x}, y_{t-T:t})|t \sim \tilde{p}(x, y_{t-T:t}|t) = q_\phi(x|y_{t-T:t}, t)p(y_{t-T:t}|t)$.

Positive samples are generated by simulating the state trajectory up to time t and sampling from the observation process conditioned on the generated state x. On the other hand, negative samples are obtained by sampling from the marginal distribution of the observations and the conditioned estimated posterior. This posterior check allows us to assess the physical consistency of the generated states and their plausibility with respect to the true observations $y_{t-T:t}$.

2 - Posterior predictive check

The idea is similar, we sample pairs $(y'_0, y_{t-T:t})|t$ from the simulated and estimated predictive-observations joint.

- $\mathbf{P}$: $(y'_0, y_{t-T:t})|t \sim p(y'_0, y_{t-T:t}|t) = \int p(y'_0|x, y_{t-T:t}, t)p(y_{t-T:t}|x, t)p(x|t)dx$.
- $\mathbf{N}$: $(\tilde{y}'_0, y_{t-T:t})|t \sim \tilde{p}(y'_0, y_{t-T:t}|t) = \int p(y'_0|x, y_{t-T:t}, t)q_\phi(x|y_{t-T:t}, t)p(y_{t-T:t}|t)dx$.

In practice, the integral is approximated by averaging over multiple samples drawn from $p(x \mid t)$. Here, y'_0 represents a one-step observation drawn from the observation process after sampling a state x from either the simulator or the generative model. Posterior predictive check allows us to evaluate the coherence of the generated states when passed through the observation process and their plausibility with respect to the received observations $y_{t-T:t}$.

Both of the presented check methods are comprehensive as they assess the quality of the generated samples based on the system's dynamics and the observation process. However, we can relax the analysis to focus either on the consistency of the states or the predictive capacity by marginalizing over the observations $y_{t-T:t}$ to perform checks on $p(x \mid t)$ (or $p(y'_0 \mid t)$). In this work, we've decided to perform only posterior checks, as they provide more penalizing evaluations.

This method is powerful as it allows for quantifying the quality of the estimated posterior over time, observations, and states. However, it does not provide visual inspection of the success or failure of our estimators. We cannot locate nor characterize where/how the estimators fails at estimating the state. Moreover, it requires sampling from the generative models, training an additional neural network for the classifier, and access to additional simulated data that are independent from the generator's training to avoid bias. Fortunately, this method scales well, as it only requires increasing the capacity of the classifier, which is easy to do if we use convolutional architectures. While simulating from the forward model may not be costly, the sampling time of our generative models could become a limiting factor.

4.5. Results

4.5.1. Remarks

During the training and evaluation process, we encountered several challenges that may impact the quality of our results.

1. All evaluation procedures were conducted on 1D problems. As previously mentioned, the evaluation of 2D problems was limited to qualitative inspection only.
2. Qualitative displayed results are samples of a unique model for each considered methods. Variability in those models are taken into account only for losses reporting and AUC measures.
3. When attempting to apply convolutional normalizing flows to the 2D problem, we were unable to generate samples. The issue arose from certain aspects of the designed flow lacking sufficient restrictiveness, resulting in the loss function exploding. Initially reconstructed samples exhibited extremely low likelihoods, causing the log probability to explode and resulting in extremely large Jacobian determinant values for some transformation steps.
4. Sampling time for score-based models is not insignificant and increases with the size of the problem. Consequently, training classifiers on samples became prohibitively time-consuming for larger problems (1D of size 128 and 256). Therefore, the training of classifiers for these larger problems was occasionally terminated prematurely. Hopefully, the loss was in most cases on a plateau.
5. The available dataset for the 2D problem was quite limited. As a consequence, our models tended to overfit the training set relatively quickly. We selected the best-performing model based on minimizing the validation loss prior to overfitting. However, score-based models typically require longer training periods to generate smooth results across the entire manifold of samples. This limitation could potentially impact the quality of our generated samples. Moreover, the validation and testing sets were also quite limited, further hindering the evaluation of posterior checks.

4.5.2. Performances

Losses

Detailed training and validation loss can be found on `W&B`[4]. In this section, we will report the best reached validation loss on every considered problems for each methods. As a reminder, the loss for flows is the Negative log-density (3.3) and the one of score-based models is the rescaled denoising score matching loss (2.19). Both must be minimized and cannot be compared between each other.

	1D					
Size N	**8**	**16**	**32**	**64**	**128**	**256**
MAF $(T = 0)$	3.266 ± 0.056	8.596 ± 0.789	17.866 ± 0.399	39.137 ± 0.536	85.970 ± 1.574	183.770 ± 5.956
MAF $(T = 9)$	-4.444 ± 0.188	-4.465 ± 0.240	-3.841 ± 0.416	-0.575 ± 0.275	21.318 ± 0.380	69.634 ± 2.505
ConvNF $(T = 0)$	2.491 ± 0.066	8.717 ± 0.109	18.123 ± 0.331	36.275 ± 2.587	70.333 ± 3.334	135.685 ± 2.453
ConvNF $(T = 9)$	-5.469 ± 0.081	-2.257 ± 0.115	-1.836 ± 1.246	-11.931 ± 0.933	-20.719 ± 1.285	-33.674 ± 1.479
PS $(T = 0)$	0.226 ± 0.002	0.231 ± 0.003	0.228 ± 0.001	0.222 ± 0.002	0.218 ± 0.002	0.216 ± 0.001
PS $(T = 9)$	0.107 ± 0.001	0.126 ± 0.002	0.120 ± 0.001	0.103 ± 0.003	0.098 ± 0.001	0.094 ± 0.002
CS $(T = 0)$	0.300 ± 0.002	0.302 ± 0.003	0.301 ± 0.002	0.297 ± 0.001	0.296 ± 0.002	0.293 ± 0.002

Table 4.2.: Best validation losses for 1D problems for each method.

	2D
Size N	**2048**
MAF $(T = 0)$	1095.204 ± 53.896
MAF $(T = 9)$	1184.298 ± 29.542
PS $(T = 0)$	0.044 ± 0.002
PS $(T = 9)$	0.042 ± 0.002
CS $(T = 0)$	0.050 ± 0.002

Table 4.3.: Best validation losses for 2D problem for each method.

As mentioned, it is hard to draw conclusion based on reached validation loss. Indeed, all problems are different (varying size and observation process). Nevertheless, we can compare flow-based methods between each other and score-based as well.

In the 1D problem, we clearly observe that additional information always lead to a loss decrease (from $T = 0$ to $T = 9$) which is expected since posterior are sharper. For NF, convolutional estimators seem to better decrease the loss while being less impacted by problem scale increment. Indeed, MAF loss increases significantly with the problem size, even for $T = 9$. We cannot say that our estimators are better when the assimilation window is higher just because of the lower loss since the NLD depends on the posterior density which depends itself on the problem. About SBM,

[4] `https://wandb.ai/gandry/dasbi`

we observe that CS is always higher than PS no matter the assimilation window. We argue this is due to the fact that CS leverages an approximation for the likelihood score and learns only the marginal score which is less constrained than the posterior one in PS. It then could be harder to learn the score related to a wider density. In addition, we observe that models perform a little bit better when the scale increases. This justifies even more the use of convolutional architectures and highlights the ability of those methods to learn spatial correlation between state's nodes even at higher dimension.

For the 2D problem, MAF terribly performs as the NLD remains very high and no more decreases when the window is wider. On the other side, SBM shine while maintaining very low RDSM losses. One can compare the scale of the RDSM for 1D and 2D problem. The latter is about one order of magnitude lower. We explain this huge difference by the underlying spatial information of the system. Indeed, flat Lorenz96 states have less local/global correlation between nodes than 2D Kolmogorov flows, making the task harder for convolutional architectures. Finally, we can observe that the gap between PS and CS method decreases when the problem size increases. This suggests that bigger problems could be harder to handle and CS is a solution to alleviate this issue by approximating part of the posterior score independently from learning phase.

Corner plots

In this section, we present the corner plots for each of the considered methods. It is important to note that we only focus on the smallest problem size to ensure readability of the plots. The following figures depict the generated corner plots for the estimated posterior of a Lorenz 96 system with 8 nodes. The nodes are arranged from top to bottom and left to right. Plots represents estimated state's posterior for both the single-step and assimilation tasks. However, it should be noted that the CS method can only be applied to the single-step setup.

The reported posteriors correspond to a simulation time of $t = 2.5$ seconds. The corresponding predictive posteriors can be found in **Appendix C.1.**. In the corner plots, the marked points represent the simulated state that generated the observation on which the estimators are conditioned. It is expected that this ground truth state belongs to the estimated posterior distribution. DPE is supposed to be a consistent estimate of the true posterior for this problem size. The computed posterior with this method is displayed in order to assess the coherence of our estimators in this setup. However, DPE performances already decreases significantly for bigger assimilation task. We then have chosen to compare estimated posterior still with the one-step case, showing that we discriminate the posterior of the simpler task. For DPE, we draw 64 000 samples from the prior that we weight with the corresponding estimate LTE ratio. Regarding estimator's samples, we drawn them in a quantity of 2048 in order to cover a sufficient region of the posterior support.

Figures 4.9 and 4.10 oppose flow architectures results with score-based ones. We clearly observe that generated posteriors are coherent with the DPE. However, the shape of the marginals is not very complex as marginals are mainly unimodal. It could be interesting to study by how much the posterior discriminates over the prior as well as the posterior allure at other simulation times.

In general, all methods seem consistent and able to capture more complex correlations (as between x^1 and x^2). The estimation seems successful, especially for the assimilation with $T = 9$ where sharper posteriors are totally in accordance with the ground-truth state despite the very narrow region it covers.

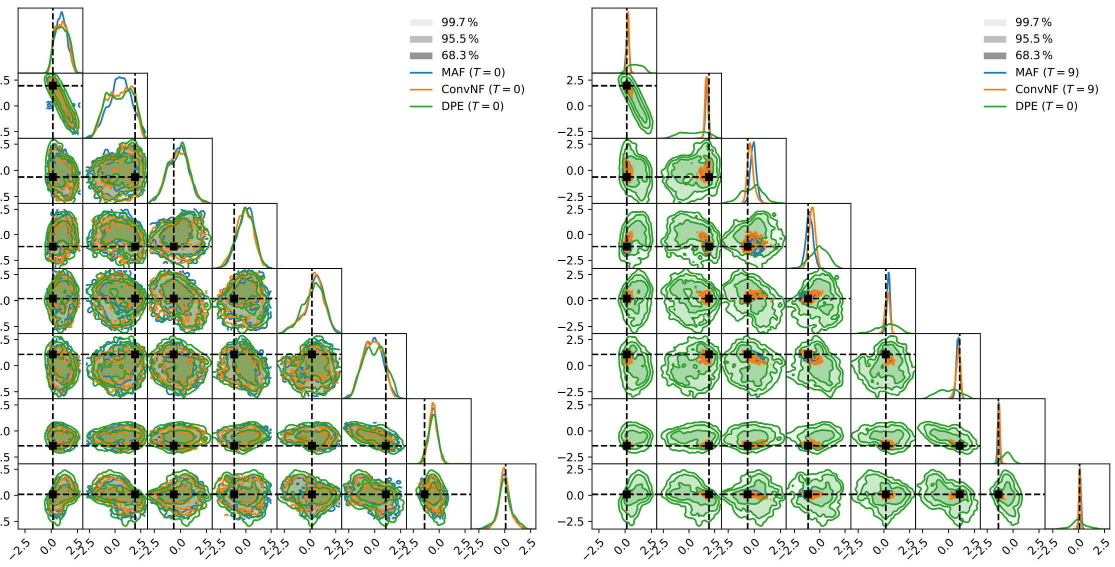
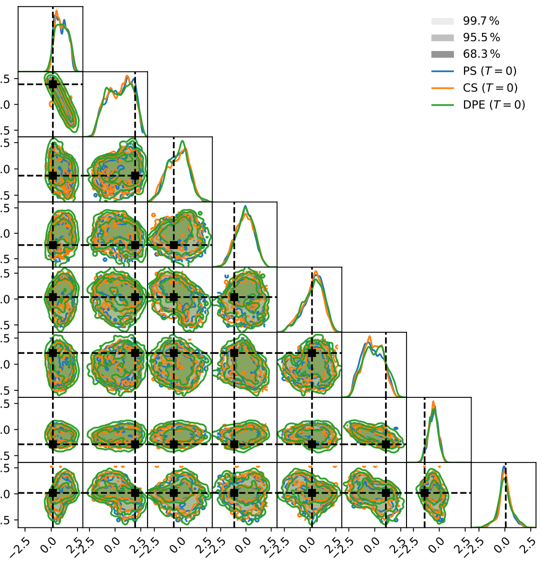


Figure 4.9.: Flows estimated posteriors along single-step DPE.



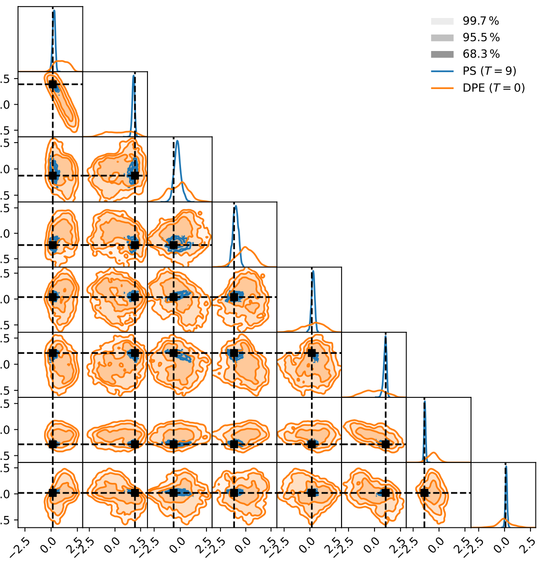


Figure 4.10.: Score-based estimated posteriors along single-step DPE.

Qualitative - Lorenz-96

For qualitative analysis, we are in the same setup than the previous evaluation method except that we consider a problem with 32 nodes. The latter is supposed harder than the previous smaller one. Since the size is already notable, DPE is not considered as a good approximation. Moreover, the latter method allow to estimate the posterior shape (suitable for corner plots) but not to sample from the posterior without relying on additional techniques as for NRE.

Considering a bigger problem allows to draw posterior predictives along the posterior states samples to check both coherence with conditioning ground-truth and observation process. Optimizing in the observation space is a problem of lower dimension which is expected to be much more easier than the posterior state task. Indeed, it is easy to identify a set of samples that produce coherent observations, but the latter must also be physically consistent.

In order to properly read the following plots, here are further details. Each node of the state is displayed in a polar coordinate plot. For readability, the origin ($x^i = 0$) corresponds to the orange dashed line. Nodes inside this circle corresponds to negative values and the other way around outside. Green dashed lines represents the nodes at which stations are placed. Eventually, blue sample is the conditioning ground-truth whereas grey thin lines represents estimate posterior's samples. We already perceive the limitation of such an evaluation method as the readability is poor regarding the samples. In practice, only 512 samples of our estimators are displayed.

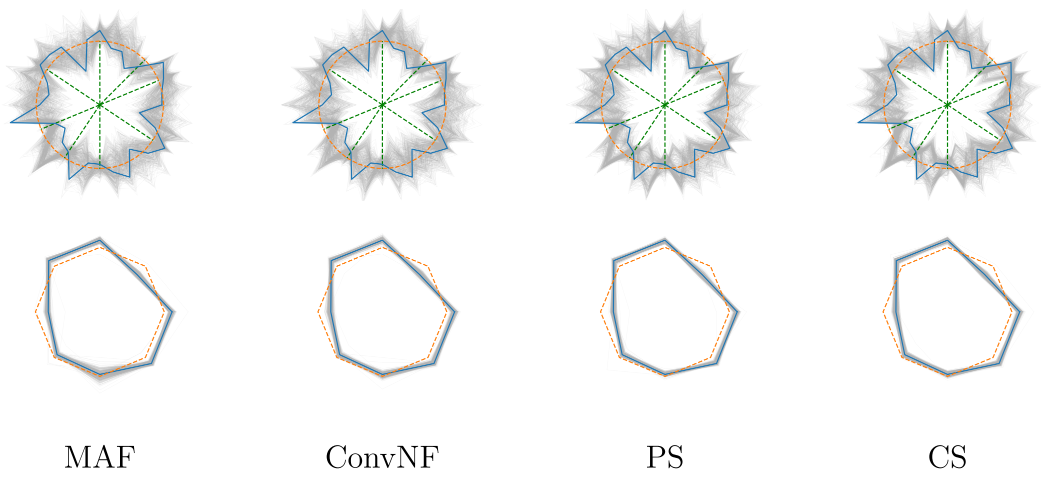


Figure 4.11.: One step assimilation estimated posterior samples for considered methods. States (UP) along predictives (DOWN).

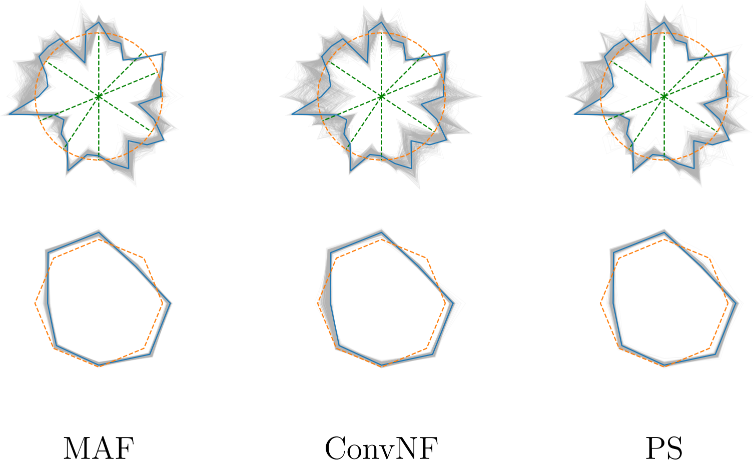


Figure 4.12.: Ten step assimilation estimated posterior samples for considered methods.

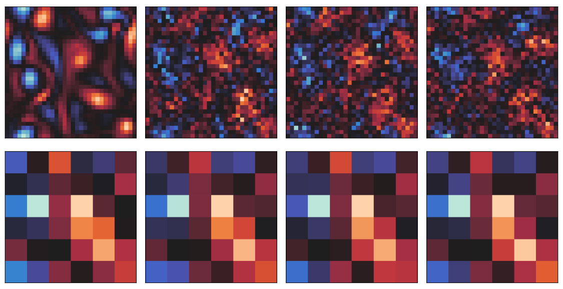
Since we lack access to the targeted posterior or an estimate of it, the relevance of drawn samples is very hard to assess. However, we can better analyze physical and observation coherence and compare between methods.

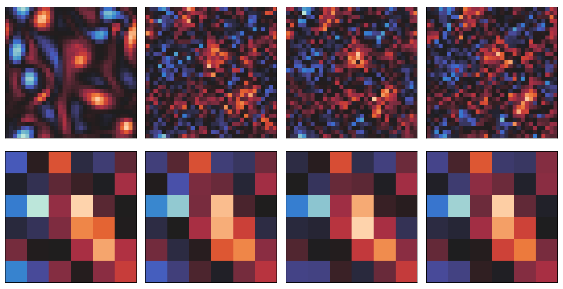
Mass of samples seems coherent with conditioning ground-truth and predictives are good for each considered estimators. This first observation bolster us about the relevance of our estimators. As before, bigger assimilation window still leads to sharper posteriors which implies a more concentrated mass of samples. However, we see some artifacts for the convolutional flow architecture that is even more visible in predictive plot. We argue that this is inherent to the choice we made about permutations between transforms and splitting inside some transforms as we observed that big artifacts were mainly present at the first and the middle node. Globally, samples of every methods are in accordance with each other. Finally, we can note a general sharper posterior around more observed zones (conversely, more uncertainty in regions where observations are more scattered). This behavior is even more visible for the ten steps assimilation task.

Qualitative - Kolmogorov flows

For this problem, we sample from our models at a time index of 50 over 64 time steps. As explained, states are of size 2x32x32 and corresponding observations 2x6x6. In the following, vorticity (4.1) of both states and observations are shown. For each estimator, we display (from left to right) the conditioning ground-truth and then 3 samples of the estimated posterior along one corresponding posterior predictive.

In contrast with the 1D problem, only few samples are displayed as sampling in high dimension is much more costly and a lot of samples is hard to visualize. In order to give a glimpse of the variability of the estimated posterior, we have depicted mean and standard deviation over 128 samples of each method in **Appendix C.2.**.


MAF ($T = 0$)


MAF ($T = 9$)

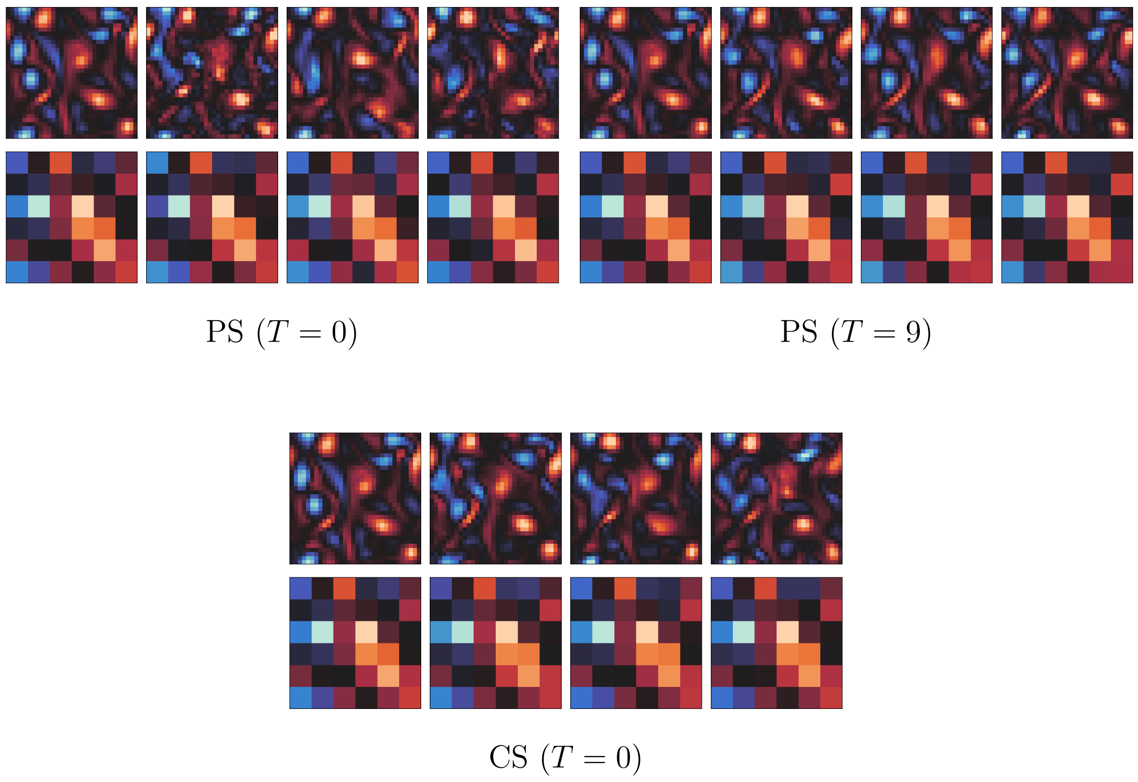


Figure 4.13.: Turbulent flows conditioning ground truth (left) along three samples of a posterior estimation for each method (right). Vorticity of the state (up) is displayed with a corresponding posterior predictive (down).

Samples from the MAF architecture seems to be poorly physically coherent although corresponding predictives are plausible regarding the ground-truth. We observe that some blobs are roughly recovered by the estimator (light blue and red regions) but the latter struggles at producing smooth spatially correlated samples. Unfortunately, we lack the ConvNF results for comparison but we reasonably assign the failure of the MAF method to the non-convolutional nature of the underlying architecture.

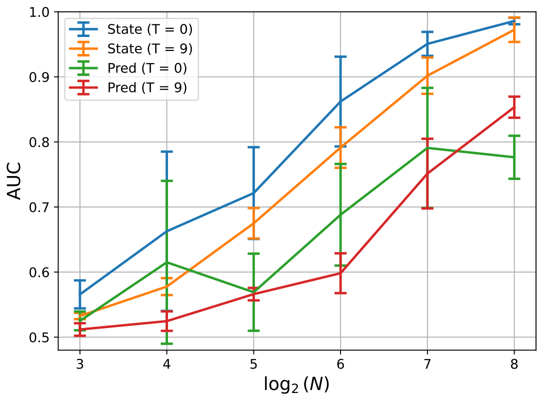
Score models are without debate much more coherent. We observe that one step PS sometimes struggles a bit to generate smooth sample but additional information (when $T = 9$) highly solves this problem. Though samples are physically coherent. On the other side, CS is surprisingly very good even if it applies only for the one-step task. Learning only the score related to underlying physics of the system by learning the marginal state's score and leveraging the likelihood approximation seems to orient very well sample generation at the cost of a decrease in time efficiency.

We have to be cautious with those observations. The dataset used for this task was very limited and our estimators quickly overfitted. We argue that performances comparison must be tested on bigger set that unlock longer training for both methods in order to compare them at the top of their performances. It remains valuable to notice that, despite this limitation, SBM produce satisfactory results probably due to their inherent architecture and their other strengths.

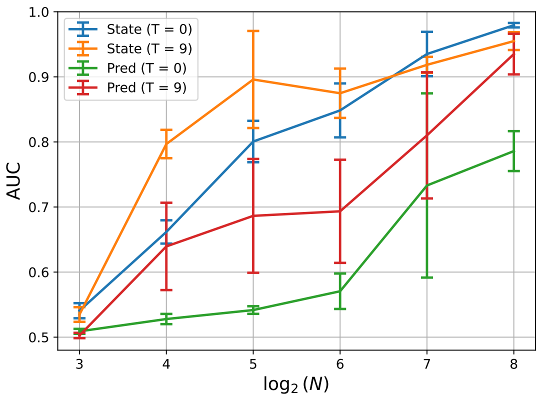
Posterior discriminator

As introduced earlier, this method provides a metric of estimator's quality that allows for comparison between methods and problems no matter their scale. Moreover this method assess the overall performances of the estimators through the whole simulation timeline. As bigger problem could be harder, we expect our estimators to struggle more with the task when the state's size increases even if we managed to scale our network capacity accordingly. We saw in the previous section that MAF method works poorly on a big problem whereas SBM were still visually consistent. We then expect to recover this disparity in our results. We also expect the posterior predictive check to be better than the posterior check as the latter concerns higher dimension.

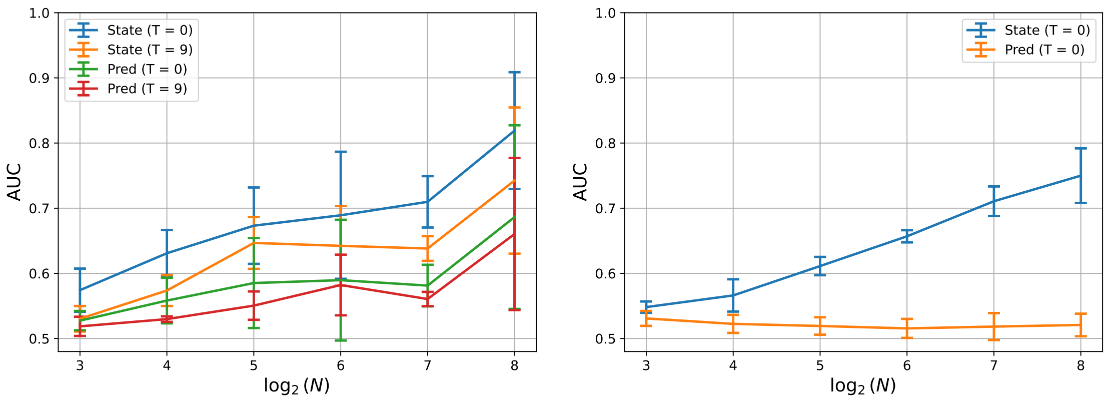
Even tough bigger assimilation window showed better performances in previous evaluations, we argue that this will be more challenging with this evaluation criteria. Indeed, as detailed in Figure 4.7, the whole context is provided to the classifier which discriminates between simulated and artificial samples. As a result, the latter is also able to capture more structure in the context and better discriminate samples. This method seems to discriminate fairly between problems, taking into account data that was available for the estimator. Following figures display the classifier AUC for different scales considered in the 1D problem.



MAF



ConvNF



PS CS

Figure 4.14.: AUC measure for each scale of the 1D problem considered and each estimation technique. AUC for posterior check are displayed along posterior predictive check.

Figure 4.14 confirms our expectations. Predictive check are better than direct posterior checks. Furthermore, score models clearly outperform flows estimators. Score-based methods seems more consistent with different test we performed whereas flows estimator's performances are entangled. We expected our estimator with wider assimilation window to behave slightly better (if they correctly make use of the given context, as explained) and the predictive check to be easier.

MAF seems coherent with this assumption but performances finally drop with the problem size. Moreover, the AUC presents considerable variation, which could depend on the considered architecture. ConvNF, on the other hand, struggles much more when $T = 9$. Indeed, our expectations are not encountered as bigger window seems to worsen the estimation. More care must be taken in the flow architecture concerning the transforms conditioning. MAF reasonable performances indicate that context embedding can be sufficient at first but the transforms of ConvNF must be better conditioned in order to achieve similar performances. Despite that, both flow-based methods struggle at handling bigger problems.

Concerning PS and CS, results are much more satisfying. We still have an overall decrease in performances with the problem size. However, the latter is less critical than for NF methods. Leveraging the likelihood score estimation in CS greatly ensures predictive coherence. We can see that posterior predictive performances are maintained no matter the problem scale which suggest that the PLS approximation is very strong to ensure state generation that lead to coherent observations.

Unfortunately, posterior check still worsen with problem size which indicates that our estimators produce samples that stay consistent with the observation process but which become less and less physically coherent with the SSM dynamics. As for flows, we suggest that increasing the network capacity would lead to challenging results.

5. Discussion

5.1. Limitations

Our study of adapting the SBI framework to assimilation problems as well as the methods that we used have some limitations. Many of them have been discussed or mentioned in appropriate sections. We will highlight in this section the major limitations of this work.

We have limited our training procedure to simulations bounded in time. We have not analyzed how our methods generalize outside of the distribution, at further simulation time. This could have been relaxed if considered SSM were statistically stationary as we could get rid of the time dependence of our estimated posterior similar to [47].

The networks we used are not fine tuned for our problems. Comparing methods at different scales is even stronger when networks scales correctly to have sufficient capacity for the task. We get use of convolutional architectures to avoid exponential growth of network's parameters. However, we did not ensure that capacity of each method at each scale is fairly comparable in every way.

The proposed classifier-based diagnostic method is very powerful at the cost of sampling requirement. As explained, to train the classifier, we need to generate a big dataset composed of half simulated and artificial (estimator's samples) data. Best methods we found in this work are SBM which still needs improvement on sampling speed despite their astonishing sample's quality. In addition, PS and CS sampling parameters are fixed. Varying them could have led to better but longer to obtain samples.

Finally, we faced several technical issues such as the failure at training ConvNF for the 2D problem, the need of early training shutdown of classifier for CS method, ...

5.2. Conclusion and future work

We showed that the SBI framework can adapt to data assimilation problems when care is taken regarding both domain differences. Especially, incorporating time into the estimation is theoretically mandatory to amortize inference for time and observations. For the latter, we saw that it is crucial to extract as much information as we can to increase estimator performances. It is nothing new in the DA field, but incorporating bigger window of observation helps a lot as it means estimating sharper posterior. Furthermore, importance must be carried on embedding of all this context.

Considered systems often have inherent structure that we must take into account for neural networks design. Along that, we often have important information about the state and the observation process. This bunch of information must be incorporated to the estimator as much as possible (e.g. state's node positions, station's location mask, ...). Convolutional architectures are well-suited for general SSM. Inference is very challenging for high dimensional data and systems. Most of the time, they require estimators with high capacity that could lead to slow sampling. This observation further emphasizes the need of appropriate architectures that trade-off the number of parameters with forward speed by leveraging the structure of the data.

Evaluating SBI methods adapted to SSM problems is challenging. Classical evaluation provides valuable insights for estimator quality but are very limited. Aggregating observations of different evaluations allow to identify an estimator to be able to learn correctly the underlying process or not. However, stronger quality assessment must be conducted to ensure consistent performances through time at the cost of training an additional classifier network in our case.

To tackle some of our limitations, we could have conducted further experiments. Time embedding effect in estimator networks could be studied and one must analyze if the latter is sufficient to handle time consistently and maybe generalize correctly outside the training domain. However, the time embedding is not expected to bring so much importance in chaotic systems as the variability of the states is very high, no matter the time (exception near the initial state, where chaos gradually takes places and states are less diverse).

Score-based models have shown promising results to adapt SBI to assimilation. In spite of their sampling speed being the main limiting factor, the latter are very powerful. CS has the main advantage of enabling zero-shot inverse problem by only specifying the differentiable observation process $\mathcal{A}$ at inference time. Sample quality could be further tuned by conducting longer training and tuning reverse SDE solver parameters. It is of interest to investigate novel methods that further increase the sampling speed such as consistency models [80] but this method still needs more investigation about the trade-off it imposes between inference speed and quality.

In further work, it could also be valuable to analyze the impact of the observation window width on the assimilation quality as well as the embedding network effect. We could push our estimators to their limits. A good experiment to conduct would be to push each method at their maximum for a big problem (such as 2D turbulent flows). And after reaching desirable level of performances, criticize about the number of parameters of each method, their inference speed and their application field.

Eventually, it would be very useful to study the adaptation of those techniques in practice. After getting valuable information about the impact of each parameters in simulations, the next challenge would be to tailor those methods to real world scenarios. How could we define the observation operator in practice ? How do we model the uncertainty in measurement process ? What are the clues we have access to, that can be incorporated to our estimators ? A lot of questions that need answers for practical deployment of those methods.

Bibliography

[1] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. “The frontier of simulation-based inference”. In: *Proceedings of the National Academy of Sciences* 117.48 (Dec. 2020), pp. 30055–30062.

[2] Arnaud Delaunoy et al. *Towards Reliable Simulation-Based Inference with Balanced Neural Ratio Estimation.* Aug. 2022.

[3] François Rozet and Gilles Louppe. *Arbitrary Marginal Neural Ratio Estimation for Simulation-based Inference.* Nov. 2021.

[4] Norman Marlier, Olivier Brüls, and Gilles Louppe. *Simulation-based Bayesian inference for multi-fingered robotic grasping.* Sept. 2021.

[5] Malavika Vasist et al. “Neural posterior estimation for exoplanetary atmospheric retrieval”. In: *Astronomy & Astrophysics* 672 (Apr. 2023), A147.

[6] Christian Schroeder de Witt et al. *Simulation-Based Inference for Global Health Decisions.* May 2020.

[7] Christoph Leuenberger Daniel Wegmann Laurent Excoffier. *Bayesian Computation and Model Selection in Population Genetics.* Jan. 2009.

[8] Roger Frigola, Fredrik Lindsten, Thomas B. Schön, and Carl E. Rasmussen. *Bayesian Inference and Learning in Gaussian Process State-Space Models with Particle MCMC.* Dec. 2013.

[9] Donald B. Rubin. “Bayesianly Justifiable and Relevant Frequency Calculations for the Applied Statistician”. In: *The Annals of Statistics* 12.4 (Dec. 1984), pp. 1151–1172.

[10] Joshua S. Speagle. *A Conceptual Introduction to Markov Chain Monte Carlo Methods.* Mar. 2020.

[11] Gael M. Martin, Brendan P. M. McCabe, Worapree Maneesoonthorn, and Christian P. Robert. *Approximate Bayesian Computation in State Space Models.* Sept. 2014.

[12] S. A. Sisson, Y. Fan, and M. A. Beaumont. *Overview of Approximate Bayesian Computation.* Feb. 2018.

[13] Jan-Matthis Lueckmann et al. *Benchmarking Simulation-Based Inference.* Apr. 2021.

[14] George Papamakarios, David C. Sterratt, and Iain Murray. *Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows.* 2019.

[15] Johanna Bertl, Gregory Ewing, Carolin Kosiol, and Andreas Futschik. *Approximate Maximum Likelihood Estimation.* July 2015.

[16] Jianlin Su. *Variational Inference: A Unified Framework of Generative Models and Some Revelations.* July 2018.

[17] Huaiyu Zhu. "On Information and Sufficiency". In: (Apr. 1997).

[18] Hideaki Kim and Hiroshi Sawada. *Histogram Meets Topic Model: Density Estimation by Mixture of Histograms.* Dec. 2015.

[19] Kyle Cranmer, Juan Pavez, and Gilles Louppe. *Approximating Likelihood Ratios with Calibrated Discriminative Classifiers.* Mar. 2016.

[20] Joeri Hermans, Volodimir Begy, and Gilles Louppe. *Likelihood-free MCMC with Amortized Approximate Ratio Estimators.* June 2020.

[21] Christian P. Robert. *The Metropolis-Hastings algorithm.* Jan. 2016.

[22] Louis Sharrock, Jack Simons, Song Liu, and Mark Beaumont. *Sequential Neural Score Estimation: Likelihood-Free Inference with Conditional Score Based Diffusion Models.* en. Nov. 2022.

[23] Yang Song et al. *Score-Based Generative Modeling through Stochastic Differential Equations.* Feb. 2021.

[24] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics.* Nov. 2015.

[25] Aditya Ramesh et al. *Hierarchical Text-Conditional Image Generation with CLIP Latents.* 2022.

[26] Prashanth Gurunath Shivakumar et al. *Distillation Strategies for Discriminative Speech Recognition Rescoring.* 2023.

[27] Paul Hagemann et al. *Multilevel Diffusion: Infinite Dimensional Score-Based Diffusion Models for Image Generation.* 2023.

[28] Fang Wu and Stan Z. Li. *DiffMD: A Geometric Diffusion Model for Molecular Dynamics Simulations.* 2023.

[29] Hengyuan Ma et al. *Accelerating Score-based Generative Models for High-Resolution Image Synthesis.* 2022.

[30] Casper Kaae Sønderby et al. *Ladder Variational Autoencoders.* 2016.

[31] Calvin Luo. *Understanding Diffusion Models: A Unified Perspective.* Aug. 2022.

[32] Qinsheng Zhang and Yongxin Chen. *Fast Sampling of Diffusion Models with Exponential Integrator.* Feb. 2023.

[33] Jiaming Song, Chenlin Meng, and Stefano Ermon. *Denoising Diffusion Implicit Models.* Oct. 2022.

[34] Daniel Sanz-Alonso, Andrew M. Stuart, and Armeen Taeb. *Inverse Problems and Data Assimilation.* Feb. 2023.

[35] Amit Apte. “An Introduction to Data Assimilation”. en. In: *Applied Mathematics.* Ed. by Susmita Sarkar, Uma Basu, and Soumen De. Springer Proceedings in Mathematics & Statistics. New Delhi: Springer India, 2015, pp. 31–42.

[36] Alberto Carrassi, Marc Bocquet, Laurent Bertino, and Geir Evensen. “Data assimilation in the geosciences: An overview of methods, issues, and perspectives”. en. In: *WIREs Climate Change* 9.5 (2018), e535.

[37] Eviatar Bach and Michael Ghil. “A multi-model ensemble Kalman filter for data assimilation and forecasting”. In: *Journal of Advances in Modeling Earth Systems* 15.1 (Jan. 2023).

[38] Aishah Albarakati et al. “Model and Data Reduction for Data Assimilation: Particle Filters Employing Projected Forecasts and Data with Application to a Shallow Water Model”. In: *Computers & Mathematics with Applications* 116 (June 2022), pp. 194–211.

[39] Florence Rabier and Zhiquan Liu. “Variational Data Assimilation: Theory and Overview”. en. In: ().

[40] R. N. Bannister. “A review of operational methods of variational and ensemble-variational data assimilation”. en. In: *Quarterly Journal of the Royal Meteorological Society* 143.703 (Jan. 2017), pp. 607–633.

[41] Nicole Aretz-Nellesen, Martin A. Grepl, and Karen Veroy. *3D-VAR for Parametrized Partial Differential Equations: A Certified Reduced Basis Approach.* May 2019.

[42] Georg Lentze. *20 years of 4D-Var: better forecasts through a better use of observations.* en. Text. Nov. 2017.

[43] Georg Lentze. *25 years of 4D-Var: how machine learning can improve the use of observations.* en. Text. Dec. 2022.

[44] Thomas Frerix et al. *Variational Data Assimilation with a Learned Inverse Observation Operator.* May 2021.

[45] Julian Mack, Rossella Arcucci, Miguel Molina-Solana, and Yi-Ke Guo. "Attention-based Convolutional Autoencoders for 3D-Variational Data Assimilation". In: *Computer Methods in Applied Mechanics and Engineering* 372 (Dec. 2020), p. 113291.

[46] Maddalena Amendola et al. *Data Assimilation in the Latent Space of a Neural Network*. Dec. 2020.

[47] François Rozet and Gilles Louppe. *Score-based Data Assimilation*. 2023.

[48] Suman Ravuri et al. "Skillful precipitation nowcasting using deep generative models of radar". en. In: *Nature* 597.7878 (Sept. 2021), pp. 672–677.

[49] Remi Lam et al. *GraphCast: Learning skillful medium-range global weather forecasting*. Dec. 2022.

[50] Danilo Jimenez Rezende and Shakir Mohamed. *Variational Inference with Normalizing Flows*. June 2016.

[51] Antoine Wehenkel and Gilles Louppe. *You say Normalizing Flows I see Bayesian Networks*. June 2020.

[52] George Papamakarios, Theo Pavlakou, and Iain Murray. *Masked Autoregressive Flow for Density Estimation*. June 2018.

[53] Guoqing Zheng, Yiming Yang, and Jaime Carbonell. *Convolutional Normalizing Flows*. July 2018.

[54] Diederik P. Kingma and Prafulla Dhariwal. *Glow: Generative Flow with Invertible 1x1 Convolutions*. July 2018.

[55] Emiel Hoogeboom, Rianne van den Berg, and Max Welling. *Emerging Convolutions for Generative Normalizing Flows*. May 2019.

[56] Sandeep Nagar, Marius Dufraisse, and Girish Varma. *CInC Flow: Characterizable Invertible 3x3 Convolution*. July 2021.

[57] Aditya Kallappa, Sandeep Nagar, and Girish Varma. *FInC Flow: Fast and Invertible $k \times k$ Convolutions for Normalizing Flows*. Jan. 2023.

[58] Alexandre Araujo, Benjamin Negrevergne, Yann Chevaleyre, and Jamal Atif. *On Lipschitz Regularization of Convolutional Layers using Toeplitz Matrix Theory*. 2020.

[59] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. May 2015.

[60] William Peebles and Saining Xie. *Scalable Diffusion Models with Transformers*. Mar. 2023.

[61] Ashish Vaswani et al. *Attention Is All You Need.* Dec. 2017.

[62] Ozan Oktay et al. *Attention U-Net: Learning Where to Look for the Pancreas.* May 2018.

[63] Joeri Hermans et al. *A Trust Crisis In Simulation-Based Inference? Your Posterior Approximations Can Be Unfaithful.* Dec. 2022.

[64] Christina Winkler, Daniel Worrall, Emiel Hoogeboom, and Max Welling. *Learning Likelihoods with Conditional Normalizing Flows.* Nov. 2019.

[65] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. *CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation.* Oct. 2021.

[66] Prafulla Dhariwal and Alex Nichol. *Diffusion Models Beat GANs on Image Synthesis.* June 2021.

[67] Jonathan Ho and Tim Salimans. *Classifier-Free Diffusion Guidance.* July 2022.

[68] Hyungjin Chung et al. *Diffusion Posterior Sampling for General Noisy Inverse Problems.* Feb. 2023.

[69] E.N. Lorenz. "Predictability: a problem partly solved". eng. PhD thesis. Shinfield Park, Reading, 1995 1995, pp. 1–18.

[70] Martin Stute, Amy Clement, and G. Lohmann. "Global climate models: Past, present, and future". In: *Proceedings of the National Academy of Sciences of the United States of America* 98 (Oct. 2001), pp. 10529–30.

[71] Ernest HovmÖller. "The Trough-and-Ridge diagram". In: *Tellus* 1.2 (1949), pp. 62–66.

[72] Marten Lienen and Stephan Günnemann. *torchode: A Parallel ODE Solver for PyTorch.* Jan. 2023.

[73] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. *nflows: normalizing flows in PyTorch.* Version v0.14. Nov. 2020.

[74] Zhuang Liu et al. *A ConvNet for the 2020s.* Mar. 2022.

[75] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (Nov. 1998), pp. 2278–2324.

[76] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems.* Vol. 25. Curran Associates, Inc., 2012.

[77] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition.* Apr. 2015.

[78] Ian J. Goodfellow et al. *Generative Adversarial Networks.* June 2014.

[79] Tom Fawcett. “An introduction to ROC analysis”. en. In: *Pattern Recognition Letters.* ROC Analysis in Pattern Recognition 27.8 (June 2006), pp. 861–874.

[80] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. *Consistency Models.* 2023.

[81] Yaron Lipman et al. *Flow Matching for Generative Modeling.* 2023.

[82] Will Grathwohl et al. *FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models.* Oct. 2018.

Appendix

A. Mathematical background

A.1. On the diffusion scheduling and its effect on the PLS

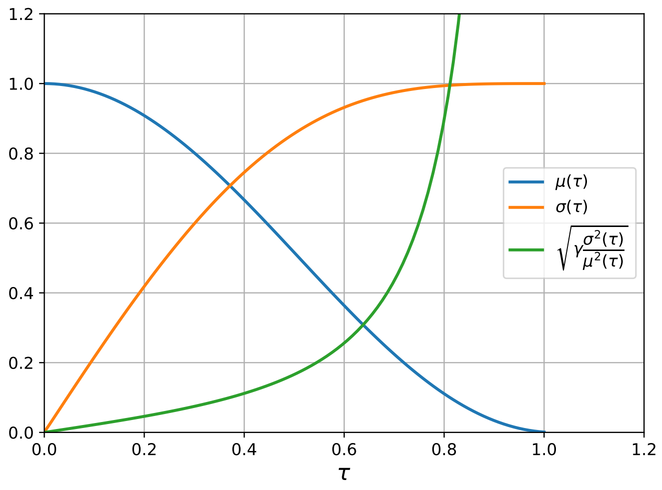


Figure A.1.: Scheduling of the diffusion kernel mean (μ), standard deviation (σ) and their effect on the PLS variance along diffusion time τ

For diffusion kernel scheduling, we followed parameters choice of [47]. This involves

$$\mu(\tau) = \cos^2(\arccos(\sqrt{\chi})\tau)$$
$$\sigma(\tau) = \sqrt{1 - \mu^2(\tau) + \chi^2}$$

Here, $\chi = 10^{-3}$ and prevent instability at $\tau = 0$ for the score estimator and at $\tau = 1$ for the PLS. This choice alter the scheduling conditions detailed earlier. We have $(\mu(0), \sigma(0)) = (1, \chi)$ and $(\mu(1), \sigma(1)) = (\chi, 1)$.

This choice allows to prevent from division by 0. However, it is accompanied by necessary counterparts.

1. Data distribution estimate $p(x(0)) \neq p(x)$. Indeed, this distribution represents hardly perturbed states. If χ is chosen sufficiently small, $p(x(0)) \approx p(x)$ is reasonable.
2. When sampling, we have to start from $x(1) \sim \mathcal{N}(\chi, I)$ and not $\mathcal{N}(0, I)$. But this is not a problem if χ is small enough and fixed.

Eventually, this scheduling choice impacts the PLS as it affects the variance of the perturbed likelihood. We observe in Figure A.1 that the closer to $\tau = 1$, the bigger the variance increase. As a result, PLS has initially few importance on the sampling guidance since an huge variance implies a negligible PLS (3.11). As we approach $\tau = 0$, PLS importance will start to increase smoothly as its variance decreases.

A.2. Tweedie's formula for the approximated likelihood

This section is mainly based on developments made in [31]. For a Gaussian variable $x \sim \mathcal{N}(\mu, \Sigma)$, Tweedie's formula states that

$$\mathbb{E}_{p(\mu|x)}(\mu) = x + \Sigma \nabla_x \log p(x) \tag{A.1}$$

With regards to the defined diffusion kernel (2.15) and (A.1). The expected mean of the noisy state $x(\tau)$ is given by:

$$\mu_{x(\tau)} = x(\tau) + \sigma^2(\tau) \nabla_{x(\tau)} \log p(x(\tau)) = \mu(\tau) x \tag{A.2}$$

Hence, the expected state x denoted $\hat{x}(x(\tau))$ from which the noisy state is generated is

$$\hat{x}(x(\tau)) = \frac{x(\tau) + \sigma^2(\tau) \nabla_{x(\tau)} \log p(x(\tau))}{\mu(\tau)} \tag{A.3}$$

A.3. Bridge the gap between normalizing flows and score-based models

SBM could be assimilated to continuous normalizing flows [81]. As detailed in [23], one can express a probability flow ODE that shares the same marginal densities than the reverse SDE based only on the diffusion parameters $\mu(\tau)$, $\sigma(\tau)$ and the estimated score $s_\phi(.)$. We denote this ODE $f_\phi(.)$. Similarly to (2.27), we can express the diffusion process as a transformation from the base density $p(x(1)) = \mathcal{N}(0, I)$ to the estimated data density $p(x(0))$ via the instantaneous change of variable formula

$$\log p(x) \approx \log p(x(0)) = \log p(x(1)) + \int_0^1 \nabla . f_\phi(.) d\tau. \tag{A.4}$$

This links diffusion models to continuous normalizing flows. However, computing $\nabla . f_\phi(.)$ is often cumbersome. An alternate way to compute this quantity is by using, as in [82] the Hutchinson unbiased estimate

$$\nabla . f_\phi(.) = \mathbb{E}_{p(\epsilon)} \left[\epsilon^T \nabla f_\phi \epsilon \right] \tag{A.5}$$

which holds as long as $\mathbb{E}[\epsilon] = 0$ and $\mathbb{C}ov[\epsilon] = I$.

As a result, one can approximate the log-density of the estimator as in NF. This estimation can be arbitrarily close to the exact log-density by averaging over a large number of samples. This allows then to compare directly SBM and NF performances.

A.4. Convolutional normalizing flow transforms

- **Squeeze**
  This transform consists in splitting in four spatially the input. This means that the input of size $C \times H \times W$ is reshaped in an output of size $4C \times \frac{H}{2} \times \frac{W}{2}$. Since this transform can be seen as a permutation of the input (if we flatten back both input and outputs as in Figure A.2), the log absolute determinant of the Jacobian is zero. Indeed, the determinant of a permutation matrix is equal to 1.

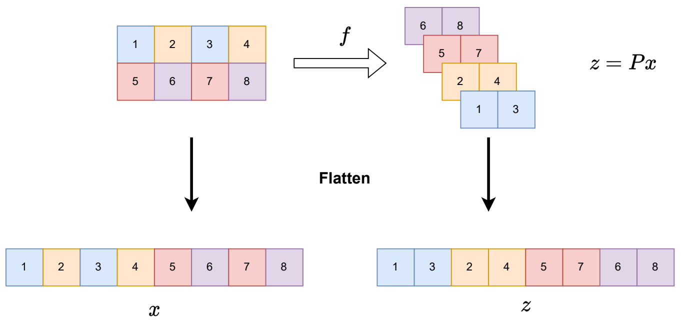


Figure A.2.: Example of Squeeze transform.

In the above example, the permutation matrix is given by

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The inverse transform is straightforward to compute by applying the inverse permutation.

- **Actnorm**
  This transform, similarly to MAF, consists in an affine transformation of the input x

  $$z = \frac{x - \alpha}{\beta}$$

  where α and β are learnable vectors of parameters. Contrary to MAF, those parameters are not functions of the input. Furthermore, they are initialized at first forward pass such that z corresponds to x normalized.

  The Jacobian of this transform is

  $$J_f = diag\left(\frac{1}{\beta}\right)$$

  for which the determinant is trivial to compute.

  The inverse transform is given by

  $$x = \beta z + \alpha.$$

- **4-Conv**
  All the theory about this block of transforms has been introduced when we defined the convolutional invertible transform. This block only consists in 4 consecutive convolution transforms, each implementing one of the four padding orientations possible.

  Computing the Jacobian is done by composition of transforms and inverting this block is done sequentially by inverting each convolution in reverse order. For visualisation of those blocks, one can go back to Figure 2.5.

- **Coupling**
  Since this transform is putted after the Squeeze transform, one ensures that the number of channels of the input is a multiple of 4. By leveraging that, we implement quad-coupling transform as in [56]. The latter follows the same principle than in MAF. We transform autoregressively the input by splitting it along in four equivalent patches along the channels. One could refer to Figure 2.4 for visual illustration, replacing x_i by each fourth of the input.

  For inversion and Jacobian computation, this transform follows the same principles than classical auto-regressive transforms. They leverage the lower triangular form of the Jacobian to compute its determinant as the sum of the diagonal elements.

- **Split**
  This transform is nothing more complex than an identity transform except that output is split to pass into other sub-transforms of the flow independently. In implemented ConvNF, we ensure that the input given to this transform has 4n channels (because of a previous Squeeze transform in the flow). We then output the n first channels to the following Merge transform and the other 3n channels are fed to another whole block of transforms as depicted in Figure 2.6.

  Determinant of the Jacobian is simply 0 as the latter is the identity matrix. Inverting the transform consists in concatenating both input along the channel dimension.

- **Merge**
  The latter is opposed to the split transform. Once every split block outputs their corresponding data, the merge block concatenate them into a single block and reshape the resulting tensor consistently in order for z to have the same dimension than the input of the whole multi-scale convolution block. Once again, this corresponds to a permutation transform for which the Jacobian and inverse is easy to compute.

B. Networks and training parameters

B.1. Training parameters and datasets

As explained earlier, simulations are performed with several parameters fixed for each simulator. First of all, Lorenz96 forcing F is fixed to 8 to ensure chaotic dynamics of the system. Then, Kolmogorov flows parameters $(\rho, \nu, ...)$ follow the specifications of [47] as data has been borrowed from this paper.

Regarding the observation process, stations are placed such that the observation size is a fourth of the state's size for the 1D problem. For the 2D one, stations are placed such that the resulting observation is of size 2x6x6. Additional noise follow a normal density $\mathcal{N}(0, 0.5^2)$ for both problems.

Concerning the training, every network is optimized with the AdamW optimizer with a linear scheduling for the learning rate and small weight decay of 10^{-4}. The number of batch per epochs and the size of those batches for the 1D (resp. 2D) problem are 128 (resp. 512) and 512 (resp. 32). The initial learning rates are 10^{-3} for every methods excepts for SBM applied to the 2D problem where it has been set to 10^{-4}.

For Lorenz96, we simulated 1024 (resp. 256) trajectories of 1024 times steps uniformly spaced $\in [0, 50]$ for training (resp. validation). For the 2D problem, once again, following the original paper, we have access to 819 (resp. 102) trajectories of 64 time steps for training (resp. validation) which is very limited and caused the overfitting of our networks.

B.2. Architectures parameters

The following section presents architectures parameters that changes from default setup for each considered methods. The list of parameters is not exhaustive as architecture are very modular. For a complete overview of each method composition, the attentive reader is invited to have a look at the source code. The following tables display parameters for both tasks $T = 0$ and $T = 9$. However, final number of parameters is related to $T = 0$. For the bigger assimilation window, the number of parameters increases a little bit due to the embedding network's first layer but reported values are good indicators.

MAF	**Hidden layers**	**Transforms**	**Embedding C.**	**Embedding lay.**	**Number param.**
$N = 8$	(64-64-64-64)	3	11	3	128,469
$N = 16$	(128-128-128-128)	3	11	3	304,389
$N = 32$	(160-160-160-160)	3	11	3	516,837
$N = 64$	(256-256-256-256)	3	11	3	1,350,693
$N = 128$	(352-352-352-352)	3	11	3	3,081,765
$N = 256$	(512-512-512-512)	4	11	3	10,565,285
$N = 2048$	(1024-2048-2048-1024)	4	20	3	142,716,894

ConvNF	**N ms-modules**	**Kernel**	**N conv.**	**Width**	**Embedding C.**	**Embedding lay.**	**Number param.**
$N = 8$	2	2	2	1	11	3	207,225
$N = 16$	2	2	2	1	11	3	213,053
$N = 32$	2	2	2	1	11	3	222,717
$N = 64$	3	2	2	2	11	3	589,794
$N = 128$	3	2	2	3	11	3	1,054,053
$N = 256$	4	2	2	4	11	3	2,860,761
$N = 2048$	/	/	/	/	/	/	/

PS	**Depth**	**Input C.**	**N conv.**	**Embedding C.**	**Embedding lay.**	**Number param.**
$N = 8$	2	48	3	11	3	203,324
$N = 16$	2	49	3	11	3	208,810
$N = 32$	2	50	3	11	3	214,458
$N = 64$	3	51	3	11	3	734,347
$N = 128$	3	52	3	11	3	760,492
$N = 256$	4	53	3	11	3	2,998,458
$N = 2048$	3	64	4	20	3	3,956,284

The CS parameters are not reported as they match with the PS method except for the input block channels (Input C.) that are adapted in order to roughly match the final number of parameters of PS as CS does not have embedded context comprised in the input.

One can notice that number of parameters of convolutional methods are tuned in order to be as more comparable as possible while maintaining good scaling with the problem size. On the opposite, MAF parameters number increases exponentially as the problem size. This was required in order to maintain good expressivity of the network due to its linear structure (compared with convolutional ones). We tried to allocate the same parameter budget than other methods for MAF, bu the latter failed as expressivity was too poor because layers were too small regarding the input.

C. Additional figures

C.1. Corner plots for posterior predictive

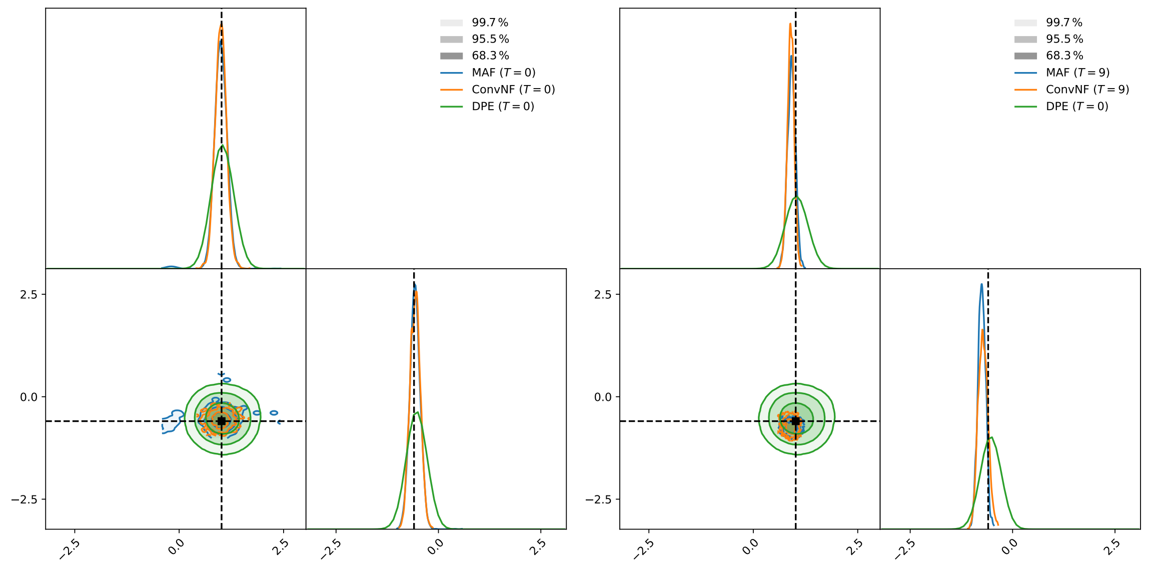


Figure C.1.: Flows estimated posteriors predictives along single-step DPE.

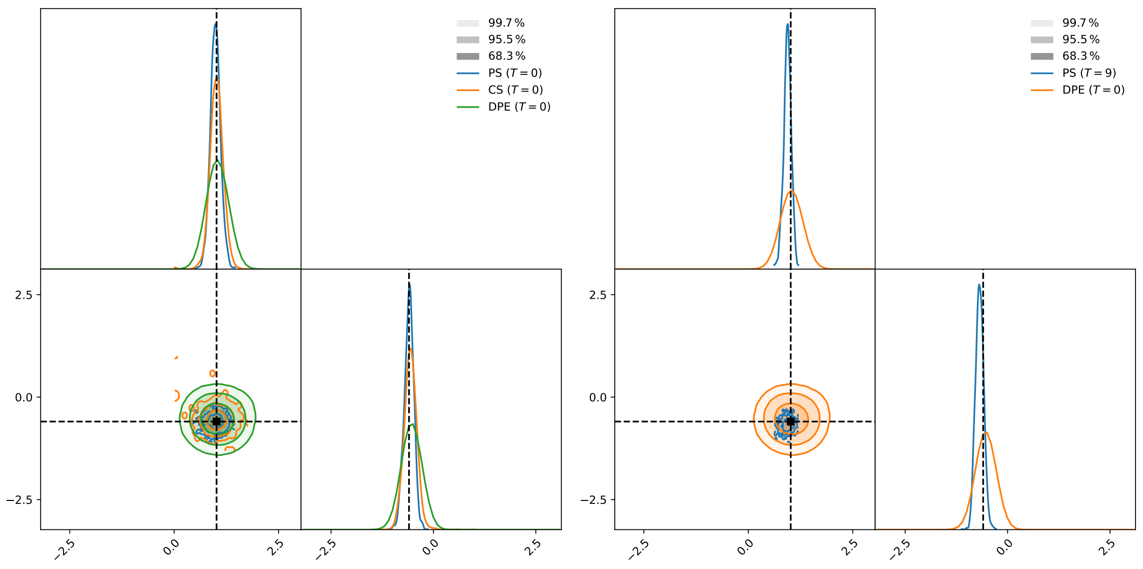


Figure C.2.: Score-based estimated posteriors predictives along single-step DPE.

C.2. Variability of 2D problem estimated posterior

Following figures are ordered by methods (rows), window size ($T = 0$ left and $T = 9$ right except for CS) and conditioning sample (first of three sub-images), estimator's mean over 128 samples (middle) and standard deviation (right).

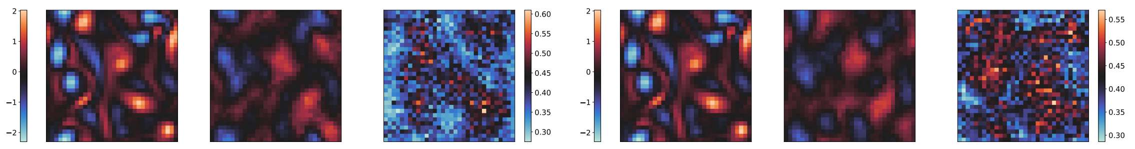


Figure C.3.: MAF

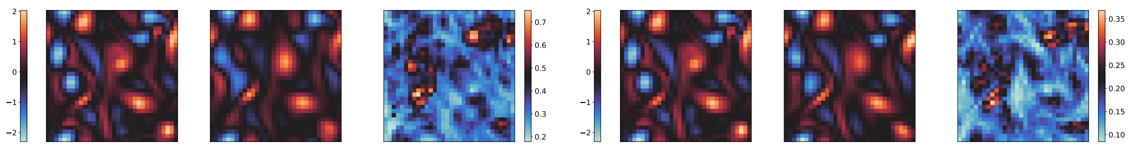


Figure C.4.: PS

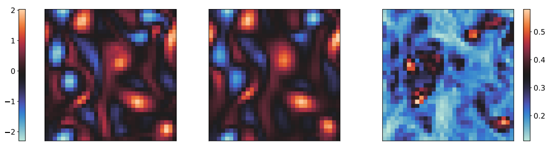


Figure C.5.: CS