FLOW MATCHING FOR ROBUST SIMULATION-BASED INFERENCE UNDER MODEL MISSPECIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Simulation-based inference (SBI) is transforming experimental sciences by enabling parameter estimation in complex non-linear models from simulated data. A persistent challenge, however, is model misspecification: simulators are only approximations of reality, and mismatches between simulated and real data can yield biased or overconfident posteriors. We address this issue by introducing Flow Matching Corrected Posterior Estimation (`FMCPE`), a framework that leverages the flow matching paradigm to refine simulation-trained posterior estimators using a small set of real calibration samples. Our approach proceeds in two stages: first, a posterior approximator is trained on abundant simulated data; second, flow matching transports its predictions toward the true posterior supported by real observations, without requiring explicit knowledge of the misspecification. This design enables `FMCPE` to combine the scalability of SBI with robustness to distributional shift. Across synthetic benchmarks and real-world datasets, we show that our proposal consistently mitigates the effects of misspecification, delivering improved inference accuracy and uncertainty calibration compared to standard SBI baselines, while remaining computationally efficient.

1 INTRODUCTION

Many fields of science and engineering describe complex phenomena with stochastic models, which capture inherent sources of randomness, such as measurement noise, probabilistic dynamics, etc. While such simulators provide a convenient way to produce synthetic data $\boldsymbol{x} \in \mathbb{R}^d$ given parameters $\boldsymbol{\theta} \in \mathbb{R}^p$, they rarely yield tractable likelihoods, making classical statistical inference methods such as Markov chain Monte Carlo (MCMC) (Robert & Casella, 2005) or variational inference (Rezende & Mohamed, 2015) inapplicable. Simulation-Based Inference (SBI) (Deistler et al., 2025; Cranmer et al., 2020) addresses this limitation by performing Bayesian parameter inference directly from simulated datasets, bypassing the need for an explicit likelihood $p_{\boldsymbol{X}|\boldsymbol{\Theta}}$.

Given a prior $p_{\boldsymbol{\Theta}}$ over parameters and assuming the existence of an unknown real data-generating process $p_{\boldsymbol{Y}|\boldsymbol{\Theta}}$ for observations $\boldsymbol{y} \in \mathbb{R}^d$, SBI algorithms provide various approaches (Papamakarios & Murray, 2016; Lueckmann et al., 2017; Boelts et al., 2022; Hermans et al., 2020) to obtain approximate samples from the posterior distribution $p_{\boldsymbol{\Theta}|\boldsymbol{Y}}$ with the aid of deep generative models. In this paper, we focus on neural-based approaches that directly approximate the posterior distribution, often called neural posterior estimation (NPE). This corresponds to the following form of posterior estimates, where the simulator's likelihood $p_{\boldsymbol{X}|\boldsymbol{\Theta}}$ is used in place of the real data likelihood $p_{\boldsymbol{Y}|\boldsymbol{\Theta}}$,

$$\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}(\boldsymbol{\theta}|\boldsymbol{y}) \propto p_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) p_{\boldsymbol{X}|\boldsymbol{\Theta}}(\boldsymbol{y}|\boldsymbol{\theta}) \,. \tag{1}$$

However, discrepancies between $p_{\boldsymbol{Y}|\boldsymbol{\Theta}}$ and $p_{\boldsymbol{X}|\boldsymbol{\Theta}}$, can severely degrade parameter inference using the above $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}$. In the Bayesian literature, this problem is referred to as *model misspecification* (Walker, 2013). Misspecification reflects limitations of the simulator or surrogate forward model—whether due to mathematical approximations, simplified dynamics, unmodeled noise, or insufficient computational resources to run more realistic simulations. Such mismatches can lead to biased or overconfident posterior distributions (Frazier et al., 2020; Schmitt et al., 2024), ultimately undermining inference reliability. In particular, modern SBI methods built on deep generative models are especially vulnerable to misspecification. They often perform poorly when faced with out-of-distribution data (Nalisnick et al., 2019) and can fail dramatically across a wide range of real-world problems (Cannon et al., 2022).

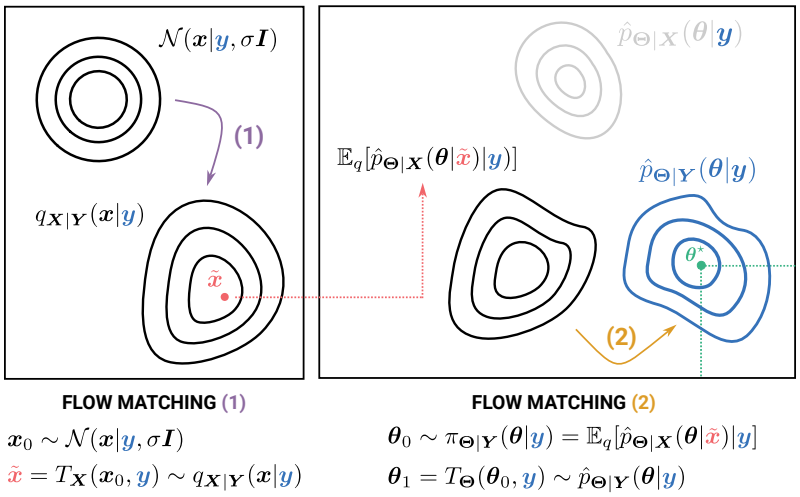


Figure 1: Overview of FMCPE. The method combines two complementary flow matching steps to correct simulation-based posterior distributions under model misspecification (represented by $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}(\boldsymbol{\theta}|\boldsymbol{y})$ in grey level sets). (1) Scarce calibration data $(\boldsymbol{\theta}, \boldsymbol{y})$ are used to learn a transport map $T_{\boldsymbol{X}}$ that couples real observations $\boldsymbol{y}$ with surrogate counterparts $\tilde{\boldsymbol{x}}$ lying in the simulator's domain. (2) We then learn $T_{\boldsymbol{\Theta}}$ to transport samples from a $q_{\boldsymbol{X}|\boldsymbol{Y}}$-weighted version of the simulation-based posterior $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}(\boldsymbol{\theta}|\tilde{\boldsymbol{x}})$ toward the final corrected posterior $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{Y}}$. Note that both transports are required: $T_{\boldsymbol{X}}$ addresses the mismatches between simulated data and real observations, while $T_{\boldsymbol{\Theta}}$ refines parameter inference to align with the true posterior.

A natural way to address imperfections in $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}$ is to act to reduce its deviation from an estimator built on high-fidelity data—accurate representations of the phenomenon obtained either from costly high-quality simulations or from ground-truth observations. Since such data are typically scarce due to their prohibitive cost, the central challenge is to design strategies that maximally exploit the limited information they provide in order to correct $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}$.

In this work, we propose a correction method that leverages the flow matching paradigm of Lipman et al. (2023) to refine posterior estimators trained on simulations using only a small set of high-fidelity calibration samples. The principled ability of flows to efficiently model paths between distributions makes them good candidates to design corrective procedures agnostic to the type of misspecifications. Moreover, flows have demonstrated high-scalability with state-of-the-art performance at large-scale image generation (Esser et al., 2024) and data-efficient performance in SBI (Wildberger et al., 2023). Our approach proceeds in two stages. First, a posterior approximator $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}$ is trained on abundant simulated data via NPE. Second, a reweighted version of this estimator is used to define a proposal distribution $\pi_{\boldsymbol{\Theta}|\boldsymbol{Y}}$, which serves as the source in a flow matching model that learns a transport map to the well-specified posterior $p_{\boldsymbol{\Theta}|\boldsymbol{Y}}$. Crucially, this transport does not require explicit knowledge of the misspecification form: it aligns the simulation-based posterior with the true posterior supported by real observations. The resulting procedure enables accurate estimation of $p_{\boldsymbol{\Theta}|\boldsymbol{Y}}$ despite the limited availability of calibration data. Figure 1 gives an overview of our method, which we call "Flow Matching for Corrected Posterior Estimation" (FMCPE). Experiments on synthetic and real-world tasks show that FMCPE is more robust to misspecification than standard SBI baselines, while being computationally efficient, amortized and applicable as a post-processing to any SBI model.

The remainder of the paper is organized as follows. We begin with an overview of SBI under model misspecification and outline how it relates to our contribution. Next, we motivate and detail our methodology, introducing flow matching concepts and notation as needed. Finally, we present numerical experiments and conclude with a discussion of the results.

2 Related Work

Model misspecification in SBI has been studied both in the framework of Approximate Bayesian Computation (ABC) (Frazier et al., 2020; Bharti et al., 2022; Fujisawa et al., 2021) and with modern

neural-based approaches (Kelly et al., 2024; Ward et al., 2022; Huang et al., 2023; Schmitt et al., 2024). These works, however, focus on settings distinct from ours. For instance, Ward et al. (2022) assume a known form of misspecification, while Huang et al. (2023) penalize the NPE loss according to the distributional shift between summary statistics of $\boldsymbol{x}$ and $\boldsymbol{y}$. Other contributions, such as Schmitt et al. (2024), restrict attention to detecting misspecification at inference time by imposing a Gaussian prior on the space of summary statistics.

An alternative perspective is to interpret misspecification as a noisy channel between simulated and real-world observations. In this view (Wehenkel et al., 2025; Ward et al., 2022), one assumes that once simulated data $\boldsymbol{x}$ are known, the real observation $\boldsymbol{y}$ does not carry additional information about the parameters $\boldsymbol{\theta}$. Formally, this conditional independence reads

$$\boldsymbol{y} \perp \boldsymbol{\theta} \mid \boldsymbol{x} \,. \tag{2}$$

Under this hypothesis, the true posterior distribution can be expressed as

$$p_{\boldsymbol{\Theta}|\boldsymbol{Y}}(\boldsymbol{\theta}|\boldsymbol{y}) = \int p_{\boldsymbol{\Theta}|\boldsymbol{X}}(\boldsymbol{\theta}|\boldsymbol{x}) p_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y}) \, \mathrm{d}\boldsymbol{x} \,, \tag{3}$$

which states that the true posterior distribution is a mixture of simulation-based posteriors, weighted by the probability of each simulated outcome $\boldsymbol{x}$ given $\boldsymbol{y}$.

Wehenkel et al. (2025) exploit this idea by assuming access to a calibration dataset of parameter–observation pairs $\{(\boldsymbol{\theta}_i, \boldsymbol{y}_i)\}_{1 \le i \le N_{\text{cal}}}$ collected from costly or time-consuming experiments. They approximate $p_{\boldsymbol{\Theta}|\boldsymbol{Y}}$ by replacing $p_{\boldsymbol{X}|\boldsymbol{Y}}$ in Equation (3) with a coupling $q_{\boldsymbol{X}|\boldsymbol{Y}}$ estimated via optimal transport (Peyré & Cuturi, 2019), which links real observations $\boldsymbol{y}$ and simulated data $\boldsymbol{x}$. However, a key limitation is that their method requires access to the full test set at inference time, preventing its use in online or sequential prediction scenarios. Moreover, the conditional independence assumption (2), while convenient, does not always hold. For example, if the model misspecification stems from the fact that the low-fidelity model neglects some parameters—say, for $\boldsymbol{\theta} = (\theta_1, \theta_2)$, $p_{\boldsymbol{X}|\boldsymbol{\Theta}}(\boldsymbol{x}|\boldsymbol{\theta}) = f(\theta_1)$ while $p_{\boldsymbol{Y}|\boldsymbol{\Theta}}(\boldsymbol{y}|\boldsymbol{\theta}) = g(\theta_1, \theta_2)$—then $\boldsymbol{y}$ will still depend on the full parameter vector, violating conditional independence.

In contrast, the method `MFNPE` proposed by Krouglova et al. (2025) does not rely on conditional independence nor require access to the full test set. Their approach first trains a posterior approximator $p_{\text{LF}}(\boldsymbol{\theta}|\boldsymbol{x})$ on low-fidelity simulations using standard NPE, then refines it with high-fidelity data. Our method follows the same spirit of leveraging low-fidelity approximations but uses them differently: the low-fidelity estimator plays a distinct role, and the subsequent corrections it undergoes are of a different nature, as detailed in the next section.

3 Flow Matching Corrected Posterior Estimation

We consider the general problem of sampling from a posterior distribution $p_{\boldsymbol{\Theta}|\boldsymbol{Y}}$ resulting from an *unknown* likelihood $p_{\boldsymbol{Y}|\boldsymbol{\Theta}}$ over observation $\boldsymbol{y} \in \mathbb{R}^d$, and known prior $p_{\boldsymbol{\Theta}}$ over a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^p$. If the likelihood were known, the posterior would be given by the Bayes rule as $p_{\boldsymbol{\Theta}|\boldsymbol{Y}}(\boldsymbol{\theta}|\boldsymbol{y}) \propto p_{\boldsymbol{Y}|\boldsymbol{\Theta}}(\boldsymbol{y}|\boldsymbol{\theta}) p_{\boldsymbol{\Theta}}(\boldsymbol{\theta})$. Instead, we assume access to an *imperfect* stochastic simulator $S : (\boldsymbol{\theta}, \epsilon) \mapsto \boldsymbol{x}$, generating low cost simulations $\boldsymbol{x} \in \mathbb{R}^d$, with ϵ a random noise accounting for randomness in the generative process. The simulator implicitly defines a generative model $p_{\boldsymbol{X}|\boldsymbol{\Theta}}$ whose posterior distribution can be approximated by an easy-to-sample model $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}$, for instance, using techniques from the SBI literature such as NPE. Unfortunately, since the simulator is inaccurate, the SBI model $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}$ is likely not to provide an accurate approximation to the true posterior $p_{\boldsymbol{\Theta}|\boldsymbol{Y}}$, in general, no matter how accurately it approximates the simulator's posterior $p_{\boldsymbol{\Theta}|\boldsymbol{X}}$. However, one can reasonably expect such learned model to be informative about the true posterior. We propose to leverage the simulator S and posterior model $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}$ in addition to a small set of calibration pairs of high-quality data $\mathcal{D}_{\text{cal}} = \{(\boldsymbol{\theta}_j, \boldsymbol{y}_j)\}_{1 \le j \le N_{\text{cal}}}$ from the joint distribution $p_{\boldsymbol{\Theta},\boldsymbol{Y}}$ to provide an accurate and efficient model $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{Y}}$ for the true posterior $p_{\boldsymbol{\Theta}|\boldsymbol{Y}}$. Such a dataset typically represents scarce ground-truth measurements or high-fidelity simulations that are costly to obtain. Consequently, we assume that N_{cal} is not large enough to provide an accurate posterior estimate from this dataset alone.

We propose to use the flow matching paradigm to learn a dynamic transport map, a vector field, from a carefully designed source distribution $\pi_{\boldsymbol{\Theta}|\boldsymbol{Y}}$ towards the target posterior $p_{\boldsymbol{\Theta}|\boldsymbol{Y}}$ using the

calibration dataset $\mathcal{D}_{\text{cal}}$. Increasing the proximity of the source and target distributions reduces the complexity of the flow and makes it easier to learn from a small number of samples. Intuitively, smaller distributional gaps are likely to require fewer steps and improve sample complexity (Cui et al., 2024; Lin et al., 2025; Kong et al., 2025; Wang et al., 2025). Following this principle, we employ the simulator S and the posterior approximation $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}$, to design and train an informative source distribution that facilitates the learning of the vector field from few calibration samples. The resulting algorithm (Algorithm 1 in Section 3.3) combines simultaneous learning of both the source distribution and the vector field, ensuring the source is continually updated as the vector field becomes more accurate. Next, we detail the approaches for learning the vector field given an informed source distribution, constructing the source itself and training them jointly.

3.1 Data-efficient posterior flow matching

The goal is to use the calibration dataset $\mathcal{D}_{\text{cal}}$ to learn a transport map $T_{\boldsymbol{\Theta}} : \mathbb{R}^p \times \mathbb{R}^d \to \mathbb{R}^p$ from an easy-to-sample base (or source) distribution $\pi_{\boldsymbol{\Theta}}$ towards the posterior distribution $p_{\boldsymbol{\Theta}|\boldsymbol{Y}}$:

$$\boldsymbol{\theta}_0 \sim \pi_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) \quad \Rightarrow \quad \boldsymbol{\theta}_1 = T_{\boldsymbol{\Theta}}(\boldsymbol{\theta}_0, \boldsymbol{y}) \sim p_{\boldsymbol{\Theta}|\boldsymbol{Y}}(\boldsymbol{\theta}|\boldsymbol{y}) \,. \tag{4}$$

To learn $T_{\boldsymbol{\Theta}}$, we use the flow matching framework of Lipman et al. (2023), an approach that has proven effective for modeling complex target distributions, such as images (Esser et al., 2024), and has recently been applied successfully in simulation-based inference (Wildberger et al., 2023). Flow matching consists of learning a time-dependent vector field $u_{\boldsymbol{\Theta}}$ capable of transporting a sample $\boldsymbol{\theta}$ from $\pi_{\boldsymbol{\Theta}}$ along a trajectory $(\boldsymbol{\theta}_t)_{t\in[0,1]}$ starting from $\boldsymbol{\theta}_0 = \boldsymbol{\theta}$, so that $\boldsymbol{\theta}_1$ is distributed according to the target $p_{\boldsymbol{\Theta}|\boldsymbol{Y}}$. The trajectory is obtained as a solution of the following ODE:

$$\frac{\mathrm{d}\boldsymbol{\theta}_t}{\mathrm{d}t} = u_{\boldsymbol{\Theta}}\left(t, \boldsymbol{\theta}_t, \boldsymbol{y}\right), \quad \forall t \in [0,1], \qquad \text{with } \boldsymbol{\theta}_0 = \boldsymbol{\theta}.$$

The time-dependent flow $\psi_{\boldsymbol{\Theta}}$ associated to the above ODE is simply given by samples along the path starting from the initial condition $\boldsymbol{\theta}$, i.e. $\psi_{\boldsymbol{\Theta}}(t, \boldsymbol{\theta}, \boldsymbol{y}) := \boldsymbol{\theta}_t$. Such a flow allows defining the transport map from $\pi_{\boldsymbol{\Theta}}$ to $p_{\boldsymbol{\Theta}|\boldsymbol{Y}}$ as $T_{\boldsymbol{\Theta}}(\boldsymbol{\theta}, \boldsymbol{y}) := \psi_{\boldsymbol{\Theta}}(1, \boldsymbol{\theta}, \boldsymbol{y})$. The vector field is approximated with a deep neural network $\hat{u}_{\boldsymbol{\Theta}}$ trained on a *guided* version of the conditional flow-matching loss from Lipman et al. (2023) with a linear interpolation strategy:

$$\mathcal{L}_{\boldsymbol{\Theta}}(\hat{u}_{\boldsymbol{\Theta}}) = \mathbb{E}\left[\int_0^1 \left\|\hat{u}_{\boldsymbol{\Theta}}(t, \boldsymbol{\theta}_t, \boldsymbol{y}) - (\boldsymbol{\theta}_1 - \boldsymbol{\theta}_0)\right\|^2 \mathrm{d}t\right], \quad \boldsymbol{\theta}_t := (1-t)\boldsymbol{\theta}_0 + t\boldsymbol{\theta}_1 \; \forall t \in [0,1] \,, \tag{5}$$

where the expectation is taken over $(\boldsymbol{\theta}_1, \boldsymbol{y}, \boldsymbol{\theta}_0) \sim p_{\boldsymbol{\Theta},\boldsymbol{Y}}(\boldsymbol{\theta}_1, \boldsymbol{y})\pi_{\boldsymbol{\Theta}}(\boldsymbol{\theta}_0)$. Here, the high-fidelity dataset $\mathcal{D}_{\text{cal}}$ can be used to provide joint samples $(\boldsymbol{\theta}_1, \boldsymbol{y})$ from $p_{\boldsymbol{\Theta},\boldsymbol{Y}}(\boldsymbol{\theta}_1, \boldsymbol{y})$, thus providing an empirical version of the above objective.

In most flow matching instances, the base distribution $\pi_{\boldsymbol{\Theta}}$ is set to a simple standardized Gaussian distribution, i.e. $\pi_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta} \mid \mathbf{0}, \boldsymbol{I}_p)$. However, for settings such as the one we consider here, where $\mathcal{D}_{\text{cal}}$ is small, this choice may result in very poor posterior approximators with high variance. Instead, we use the simulator to construct a more informative data-driven source distribution $\pi_{\boldsymbol{\Theta}} = \pi_{\boldsymbol{\Theta}|\boldsymbol{Y}}$ that acts as a possibly low-quality surrogate of the true $p_{\boldsymbol{\Theta}|\boldsymbol{Y}}$. Training itself remains consistent with standard guided flow matching; the learned flow then serves to refine this source distribution using only the limited high-fidelity data.

3.2 Simulation-informed source distribution

The source distribution $\pi_{\boldsymbol{\Theta}|\boldsymbol{Y}}$ plays a central role in our framework. Ideally, it should be very close to the true posterior distribution, so that the flow $\psi_{\boldsymbol{\Theta}}$ induces minimal corrections to $\boldsymbol{\theta}_0 \sim \pi_{\boldsymbol{\Theta}|\boldsymbol{Y}}$. Making use of the availability of the simulator S, a first natural possibility is to set $\pi_{\boldsymbol{\Theta}|\boldsymbol{Y}}(\boldsymbol{\theta}|\boldsymbol{y}) = \hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}(\boldsymbol{\theta}|\boldsymbol{y})$ where $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}$ is a posterior approximation obtained via SBI. This provides a reasonable approximation of the true $p_{\boldsymbol{\Theta}|\boldsymbol{Y}}$ in the absence of misspecification but is likely to be a poor one in less favorable settings. A natural alternative is to plug into $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}$ a conditioning sample from a distribution $q_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y})$ with the same support as $p_{\boldsymbol{X}}(\boldsymbol{x})$ and informative of $\boldsymbol{y}$. This corresponds to considering a source distribution of the form

$$\pi_{\boldsymbol{\Theta}|\boldsymbol{Y}}(\boldsymbol{\theta}|\boldsymbol{y}) = \int \hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}(\boldsymbol{\theta}|\boldsymbol{x})\, q_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y})\, d\boldsymbol{x}. \tag{6}$$

The most straightforward choice for $q_{\boldsymbol{X}|\boldsymbol{Y}}$ is $q_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y}) := p_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y})$, which would require, in addition to $\mathcal{D}_{\text{cal}}$, a large number of ground truth samples $(\boldsymbol{y}, \boldsymbol{x})$ for training an approximator. Thus, we consider instead,

$$q_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y}) := \int p_{\boldsymbol{X}|\boldsymbol{\Theta}}(\boldsymbol{x}|\boldsymbol{\theta})\, p_{\boldsymbol{\Theta}|\boldsymbol{Y}}(\boldsymbol{\theta}|\boldsymbol{y})\, d\boldsymbol{\theta}, \tag{7}$$

that can be seen as a mixture and approximated using the simulator S and joint calibration pairs $(\boldsymbol{\theta}_j, \boldsymbol{y}_j) \sim p_{\boldsymbol{\Theta},\boldsymbol{Y}}(\boldsymbol{\theta}, \boldsymbol{y}) = p_{\boldsymbol{\Theta}|\boldsymbol{Y}}(\boldsymbol{\theta}|\boldsymbol{y})p_{\boldsymbol{Y}}(\boldsymbol{y})$. Indeed, note that for each $\boldsymbol{y}_j \in \mathcal{D}_{\text{cal}}$, its associated $\boldsymbol{\theta}_j$ is a sample from $p_{\boldsymbol{\Theta}|\boldsymbol{Y}}(\boldsymbol{\theta}|\boldsymbol{y}_j)$ that can be plugged in the simulator to generate $\boldsymbol{x}_j \sim p_{\boldsymbol{X}|\boldsymbol{\Theta}}(\boldsymbol{x}|\boldsymbol{\theta}_j)$, leading to $\boldsymbol{x}_j \sim q_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y_j})$. This is typically used in lines 2 and 3 of Algorithm 2.

In practice, a sample $\boldsymbol{\theta}$ from the source distribution (6), can be obtained by first drawing a sample $\tilde{\boldsymbol{x}}$ from $q_{\boldsymbol{X}|\boldsymbol{Y}}(\tilde{\boldsymbol{x}}|\boldsymbol{y})$ in (7), then setting $\boldsymbol{\theta}$ to a sample from $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}(\boldsymbol{\theta}|\tilde{\boldsymbol{x}})$. As we can already easily sample from $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}$, it only remains to provide a way to sample from $q_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y})$ for any $\boldsymbol{y}$. To this end, we use the flow matching framework to construct a conditional transport map $T_{\boldsymbol{X}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$, from a Gaussian distribution centered around $\boldsymbol{y}$ with isotropic variance σ^2, towards $q_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y})$ i.e.:

$$\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{y}, \sigma^2 \boldsymbol{I}_d) \to \tilde{\boldsymbol{x}} = T_{\boldsymbol{X}}(\boldsymbol{x}_0, \boldsymbol{y}) \approx q_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y}). \tag{8}$$

More specifically, we define $T_{\boldsymbol{X}}(\boldsymbol{x}_0, \boldsymbol{y}) = \psi_{\boldsymbol{X}}(1, \boldsymbol{x}_0, \boldsymbol{y})$, where $\psi_{\boldsymbol{X}}$ is the flow associated to the velocity field neural approximator $\hat{u}_{\boldsymbol{X}}$ learned by minimizing the following objective:

$$\mathcal{L}_{\boldsymbol{X}}(\hat{u}_{\boldsymbol{X}}) = \mathbb{E}\left[\int_0^1 \left\|\hat{u}_{\boldsymbol{X}}(t, \boldsymbol{x}_t, \boldsymbol{y}) - (\boldsymbol{x}_1 - \boldsymbol{x}_0)\right\|^2 \mathrm{d}t\right],\ \boldsymbol{x}_t := (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1\ \forall t \in [0,1]\,, \tag{9}$$

where the expectation is taken over $(\boldsymbol{y}, \boldsymbol{x}_1, \boldsymbol{x}_0) \sim q_{\boldsymbol{Y},\boldsymbol{X}}(\boldsymbol{y}, \boldsymbol{x}_1)\mathcal{N}(\boldsymbol{x}_0|\boldsymbol{y}, \sigma^2\boldsymbol{I}_d)$. Here, the dataset $\mathcal{D}_{\text{cal}}$ and simulator S can be used to provide joint samples $(\boldsymbol{y}, \boldsymbol{x}_1)$ from $q_{\boldsymbol{Y},\boldsymbol{X}}(\boldsymbol{y}, \boldsymbol{x}_1)$ as discussed later in Section 3.3. We took inspiration from Albergo et al. (2024) to define a source distribution that induces a coupling between the base and target distributions through the conditioning variable, which greatly helps the training process under limited data.

When the conditional independence hypothesis in Equation (2) is valid and chosing $q_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y}) = p_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y})$, we can use Equation (3) to see that $p_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y})$ is indeed the optimal choice for transporting data from $\boldsymbol{y}$-space to $\boldsymbol{x}$-space, since

$$p_{\boldsymbol{\Theta}|\boldsymbol{Y}}(\boldsymbol{\theta}|\boldsymbol{y}) = \mathbb{E}_{\boldsymbol{X}|\boldsymbol{Y}}\left[p_{\boldsymbol{\Theta}|\boldsymbol{X}}(\boldsymbol{\theta}|\boldsymbol{x})|\boldsymbol{y}\right],$$
$$\tilde{\boldsymbol{x}} \sim p_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y}) \text{ and } \boldsymbol{\theta} \sim p_{\boldsymbol{\Theta}|\boldsymbol{X}}(\boldsymbol{\theta}|\tilde{\boldsymbol{x}}) \Longrightarrow \boldsymbol{\theta} \sim p_{\boldsymbol{\Theta}|\boldsymbol{Y}}(\boldsymbol{\theta}|\boldsymbol{y}),.$$

In that case, learning only $\hat{u}_{\boldsymbol{X}}$ could be seen as sufficient, since the proposal $\pi_{\boldsymbol{\Theta}|\boldsymbol{Y}}$ is already a good approximation to the true posterior. However, if the conditional independence is no longer valid, if $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}$ is poorly trained, or $\hat{u}_{\boldsymbol{X}}$ is not optimal, the proposal would not have the flexibility to compensate for errors from each of its different parts. This is no longer an issue when using $\hat{u}_{\boldsymbol{X}}$ to define a source distribution for learning the vector field $\hat{u}_{\boldsymbol{\Theta}}$, as proposed in this work, since it does not rely on the validity of Equation (2).

3.3 Joint training of posterior and source distributions by flow matching

We now face two optimization tasks: minimizing the loss in (9) to train the simulation-space vector field $\hat{u}_{\boldsymbol{X}}$ and minimizing (5) to train the parameter-space vector field $\hat{u}_{\boldsymbol{\Theta}}$. Instead of solving them separately, we propose to optimize the following *joint objective* for improved practical performance:

$$\mathcal{L}_{\boldsymbol{\Theta},\boldsymbol{X}}(\hat{u}_{\boldsymbol{X}}, \hat{u}_{\boldsymbol{\Theta}}) = \mathbb{E}\left[\int_0^1 \left\|\hat{u}_{\boldsymbol{\Theta}}(t, \boldsymbol{\theta}_t, \boldsymbol{y}) - (\boldsymbol{\theta}_1 - \boldsymbol{\theta}_0)\right\|^2 + \left\|\hat{u}_{\boldsymbol{X}}(t, \boldsymbol{x}_t, \boldsymbol{y}) - (\boldsymbol{x}_1 - \boldsymbol{x}_0)\right\|^2 \mathrm{d}t\right],$$

with $(\boldsymbol{x}_1, \boldsymbol{\theta}_1, \boldsymbol{y}, \boldsymbol{x}_0, \boldsymbol{\theta}_0) \sim q_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}_1|\boldsymbol{y})p_{\boldsymbol{\Theta},\boldsymbol{Y}}(\boldsymbol{\theta}_1, \boldsymbol{y})\pi_{\boldsymbol{\Theta}|\boldsymbol{Y}}(\boldsymbol{\theta}_0|\boldsymbol{y})\mathcal{N}(\boldsymbol{x}_0, \boldsymbol{y}, \sigma^2 I_d)$ and where $\boldsymbol{\theta}_t$ and $\boldsymbol{x}_t$ are convex combinations as in Equations (5) and (9). An important specificity of the above objective compared to standard flow matching is that the source sample $\boldsymbol{\theta}_0$ depends on $\hat{u}_{\boldsymbol{X}}$, by definition of the source $\pi_{\boldsymbol{\Theta}|\boldsymbol{Y}}$, which is evolving during training. Intuitively, this joint formulation forces $\hat{u}_{\boldsymbol{\Theta}}$ to be robust to noisy or inaccurate samples from the source distribution: during early training stages, $\pi_{\boldsymbol{\Theta}|\boldsymbol{Y}}$ may yield poor candidates $\boldsymbol{\theta}_0$, yet $\hat{u}_{\boldsymbol{\Theta}}$ must learn to accommodate them.

This robustness is desirable at test time, where the source distribution may also generate imperfect samples for previously unseen observations $\boldsymbol{y}$.

The joint learning of vector fields $\hat{u}_{\boldsymbol{X}}$ and $\hat{u}_{\boldsymbol{\Theta}}$ is summarized in Algorithm 1, which specifies how to optimize the objective $\mathcal{L}_{\boldsymbol{\Theta},\boldsymbol{X}}(\hat{u}_{\boldsymbol{X}}, \hat{u}_{\boldsymbol{\Theta}})$ using the sampling procedure from Algorithm 2 referred to as function SAMPLETRAININGTUPLE. In practice, variable t is sampled i.i.d. for each of the terms $\boldsymbol{X}$ and $\boldsymbol{\Theta}$ from $\mathcal{L}_{\boldsymbol{X},\boldsymbol{\Theta}}$ (lines 5 and 6) to reduce bias. Also, because $\hat{u}_{\boldsymbol{X}}$ affects the distribution of $\boldsymbol{\theta}_0$ (via $\tilde{\boldsymbol{x}}$), the effective source distribution for $\hat{u}_{\boldsymbol{\Theta}}$ is non-stationary; to mitigate instability we use small learning rates and gradient clipping to reduce training instability in the early stages.

Algorithm 1 Joint flow training for FMCPE

Require: SAMPLETRAININGTUPLE function, trainable flows $\hat{u}_{\boldsymbol{X}}, \hat{u}_{\boldsymbol{\Theta}}$ and minibatch size B.

1: **repeat**
2: $\quad\mathcal{L} \leftarrow 0$
3: $\quad$**for** $i = 1$ **to** B **do**
4: $\quad\quad(\boldsymbol{y}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_0, \boldsymbol{x}_1, \boldsymbol{x}_0) \leftarrow$ SAMPLETRAININGTUPLE$(\hat{u}_{\boldsymbol{X}})$ ▷ see Algorithm 2
5: $\quad\quad$Draw $t \sim \mathcal{U}[0,1]$ and set $\boldsymbol{x}_t = (1-t)\boldsymbol{x}_0 + t\,\boldsymbol{x}_1$
6: $\quad\quad$Draw *another independent* $\tau \sim \mathcal{U}[0,1]$ and set $\boldsymbol{\theta}_\tau = (1-\tau)\boldsymbol{\theta}_0 + \tau\,\boldsymbol{\theta}_1$
7: $\quad\quad\ell_{\boldsymbol{X}} \leftarrow \|\hat{u}_{\boldsymbol{X}}(t, \boldsymbol{x}_t, \boldsymbol{y}) - (\boldsymbol{x}_1 - \boldsymbol{x}_0)\|^2$
8: $\quad\quad\ell_{\boldsymbol{\Theta}} \leftarrow \|\hat{u}_{\boldsymbol{\Theta}}(\tau, \boldsymbol{\theta}_\tau, \boldsymbol{y}) - (\boldsymbol{\theta}_1 - \boldsymbol{\theta}_0)\|^2$
9: $\quad\quad\mathcal{L} \leftarrow \mathcal{L} + (\ell_{\boldsymbol{X}} + \ell_{\boldsymbol{\Theta}})$
10: $\quad$**end for**
11: $\quad\mathcal{L} \leftarrow \mathcal{L}/B$
12: $\quad$Take one optimizer step on trainable parameters of $\hat{u}_{\boldsymbol{X}}$ and $\hat{u}_{\boldsymbol{\Theta}}$ using $\mathcal{L}$
13: **until** convergence

Algorithm 2 first samples $(\boldsymbol{\theta}_1, \boldsymbol{y})$ from the calibration dataset, where $\boldsymbol{\theta}_1$ serves as target sample to the vector field $\hat{u}_{\boldsymbol{\Theta}}$. We then generate $\boldsymbol{x}_1 \sim p_{\boldsymbol{X}|\boldsymbol{\Theta}}(\boldsymbol{x}|\boldsymbol{\theta}_1)$ which serves as target for $\hat{u}_{\boldsymbol{X}}$ and is approximately distributed according to $q_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y})$ (see Equation (7)). The source sample $\boldsymbol{x}_0$ is obtained from a Gaussian distribution centered at $\boldsymbol{y}$. To obtain a source sample $\boldsymbol{\theta}_0$ for the vector field $\hat{u}_{\boldsymbol{\Theta}}$, an intermediate sample $\tilde{\boldsymbol{x}}$ is computed by solving an ODE driven by the current estimated vector field $\hat{u}_{\boldsymbol{X}}$ and starting from $\boldsymbol{x}_0$. This sample is simply provided to the approximate posterior to get $\boldsymbol{\theta}_0 \sim \hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}(\boldsymbol{\theta}|\tilde{\boldsymbol{x}})$ as discussed in Section 3.2. Note that one could alternatively generate $\tilde{\boldsymbol{x}} = S(\boldsymbol{\theta}_1, \epsilon)$ directly from the simulator instead of solving the ODE induced by $\hat{u}_{\boldsymbol{X}}$. However, this strategy cannot be applied at inference time, where only the observation $\boldsymbol{y}$ is available and not the pair $(\boldsymbol{\theta}, \boldsymbol{y})$. For consistency between training and inference, we therefore sample $\tilde{\boldsymbol{x}}$ using the ODE. A crucial aspect in the above sampling procedure is to prevent gradients from propagating through the intermediate sample $\tilde{\boldsymbol{x}}$ when optimizing the vector field $\hat{u}_{\boldsymbol{X}}$ which would bias its training.

Algorithm 2 SAMPLETRAININGTUPLE$(\hat{u}_{\boldsymbol{X}})$

Require: Calibration set $\mathcal{D}_{\text{cal}}$, simulator S, pretrained SBI model $\hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}(\boldsymbol{\theta} \mid \tilde{\boldsymbol{x}})$ (frozen) and velocity field $\hat{u}_{\boldsymbol{X}}$

function SAMPLETRAININGTUPLE$(\hat{u}_{\boldsymbol{X}})$
$\quad$Sample $(\boldsymbol{\theta}_1, \boldsymbol{y}) \sim \mathcal{D}_{\text{cal}}$ ▷ calibration samples
$\quad$Sample $\boldsymbol{x}_1$ using simulator S evaluated at $\boldsymbol{\theta}_1$ ▷ see Equation (7)
$\quad$Draw base sample $\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{x}_0, \boldsymbol{y}, \sigma^2 I)$ ▷ base for $\boldsymbol{X}$-flow
$\quad$Solve ODE: $\frac{d\bar{\boldsymbol{x}}_t}{dt} = \hat{u}_{\boldsymbol{X}}(t, \bar{\boldsymbol{x}}_t, y)$ with $\bar{\boldsymbol{x}}_0 = \boldsymbol{x}_0$.
$\quad$Set $\tilde{\boldsymbol{x}} \leftarrow T_{\boldsymbol{X}}(\boldsymbol{x}_0; \boldsymbol{y}) := \bar{\boldsymbol{x}}_1$ ▷ map base → simulator-space (uses current $\hat{u}_{\boldsymbol{X}}$)
$\quad$Set $\tilde{\boldsymbol{x}} \leftarrow$ StopGradient$(\tilde{\boldsymbol{x}})$ ▷ do not propagate gradient through sample
$\quad$Sample $\boldsymbol{\theta}_0 \sim \hat{p}_{\boldsymbol{\Theta}|\boldsymbol{X}}(\boldsymbol{\theta}|\tilde{\boldsymbol{x}})$ ▷ sample from source
$\quad$**return** $(\boldsymbol{y}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_0, \boldsymbol{x}_1, \boldsymbol{x}_0)$
end function

4 EXPERIMENTS

We benchmark our method against two baselines, NPE and `MFNPE`. Note that we do not include `RoPE` (Wehenkel et al., 2025) because it is a method that requires access to the full test set at inference time and is not directly comparable to our approach, as explained in Section 2. The comparison is carried out on four tasks described in Section 4.1, using various evaluation metrics

specified in Section 4.2. We follow the implementation from Wehenkel et al. (2025) for NPE and train it using only the limited calibration data $\mathcal{D}_{\text{cal}}$. Note that since the simulator is not used, we expect this baseline NPE to fail as the complexity of the generating process increases. For each experiment, all evaluations are performed on a test set $\mathcal{D}_{\text{test}} = \{(\boldsymbol{\theta}_j, \boldsymbol{y}_j)\}_{1 \leq j \leq N_{\text{test}}}$ with $N_{\text{test}} = 2000$ unless otherwise specified, and each metric is reported for different sizes N_{cal} of the calibration set. Calibration sets are constructed in an expanding manner, gradually adding new samples to an initial set, to limit the sources of variability in the comparison; see Appendix A for details.

All experiments described below are implemented in Python using `pytorch` (Paszke et al., 2019) and `mlxp` (Arbel & Zouaoui, 2024). We also use `nflows` (Durkan et al., 2020) and the `dingo` (Dax et al., 2021) packages for the implementation of continuous normalizing flows. Our code is provided as a zip file in the supplementary materials.

4.1 Tasks

Our experiments consist of two synthetic and two real-world tasks. Setup and implementation details can be found in Appendix A.

`Gaussian`: A multivariate Gaussian model is considered, with $\boldsymbol{\theta} \in \mathbb{R}^3$ and $\boldsymbol{\theta} \sim \mathcal{N}(\mu_\theta, \Sigma_\theta)$. Both $\boldsymbol{x} \in \mathbb{R}^{10}$ and $\boldsymbol{y} \in \mathbb{R}^{10}$ follow multivariate Gaussian distributions centered on different linear combinations of $\boldsymbol{\theta}$.

`Pendulum`: The damped pendulum (Takeishi & Kalousis, 2021) models the oscillations of a mass around a fixed attachment point. Parameters $\boldsymbol{\theta} = [A, \omega_0]$ are the oscillation amplitude A and the natural frequency ω_0. Simulations are generated from a simplified model that omits friction forces, thus creating a systematic misspecification relative to the real dynamics. Both observations and simulations are real-valued time-series of length 200.

`WindTunnel`: This task from Gamella et al. (2025) consists of measuring the air pressure inside an horizontal tube where air is being pushed through by two controllable fans at both ends. The goal is to infer the opening angle (in degrees) of a hatch $H \in [0, 45]$ on the side, given the pressure values inside the tunnel after applying a short power impulse to the intake fan. For the simulator, we use the model A2C3 from (Gamella et al., 2025, Appendix IV).

`LightTunnel`: In this task from Gamella et al. (2025), a camera is capturing light passing through two linear polarizers inside an elongated chamber. The goal is to predict RGB values of the light source and the polarizer effect $\alpha \in [0, 1]$, which is a function of the polarizer angle, so that $\boldsymbol{\theta} := [R, G, B, \alpha] \in [0, 255]^3 \times [0, 1]$. The simulator is a simplification of the real world process and described in (Gamella et al., 2025, Appendix IV) (Model F1). Observations are RGB images of size $(W, H, C) = (64, 64, 3)$ produced by either the simulator or the real apparatus; model misspecification arises because the simulator omits certain physical effects present in the real measurements.

4.2 Evaluation Metrics

We assess the performance of our method performance based on three metrics.

Joint Classifier Two-Sample Test (jC2ST). The C2ST (Lopez-Paz & Oquab, 2017) measures the discrepancy between two distributions by training a binary classifier to distinguish samples drawn from them. The test statistic is the classifier's accuracy on samples from a test set, which is equal to chance-level (usually 0.5) for indistinguishable distributions and approaches 1 as the distributions diverge. As our reference $\mathcal{D}_{\text{test}}$ provides a single ground truth value $\boldsymbol{\theta}_j$ for each $\boldsymbol{y}_j$ in the test set, standard C2ST cannot be used, as it would require a full sample from the true posterior $p(\boldsymbol{\theta}|\boldsymbol{y}_j)$ for each $\boldsymbol{y}_j$. Instead, for each $\boldsymbol{y}_j$ we generate one $\tilde{\boldsymbol{\theta}}_j$ from the learned posterior and compare $\mathcal{D}_{\text{test}}$ to the set of pairs $\{(\tilde{\boldsymbol{\theta}}_j, \boldsymbol{y}_j)\}_{1 \leq j \leq N_{\text{test}}}$ as samples from a joint distribution, justifying the name jC2ST.

Wasserstein Distance (W_2). Similarly, the Wasserstein distance for the L_2 cost is computed between samples from the true and approximate joint distributions,

$$W_2 = \min_{\gamma \in \mathbb{R}_+^{N_{\text{test}} \times N_{\text{test}}}} \left(\sum_{i,j} \gamma_{i,j} \|(\boldsymbol{\theta}_i, \boldsymbol{y}_i) - (\tilde{\boldsymbol{\theta}}_j, \boldsymbol{y}_j)\|_2^2 \right)^{\frac{1}{2}} \text{ such that } \gamma.\mathbf{1} = \tfrac{1}{N_{\text{test}}} \text{ and } \gamma^T.\mathbf{1} = \tfrac{1}{N_{\text{test}}} .$$

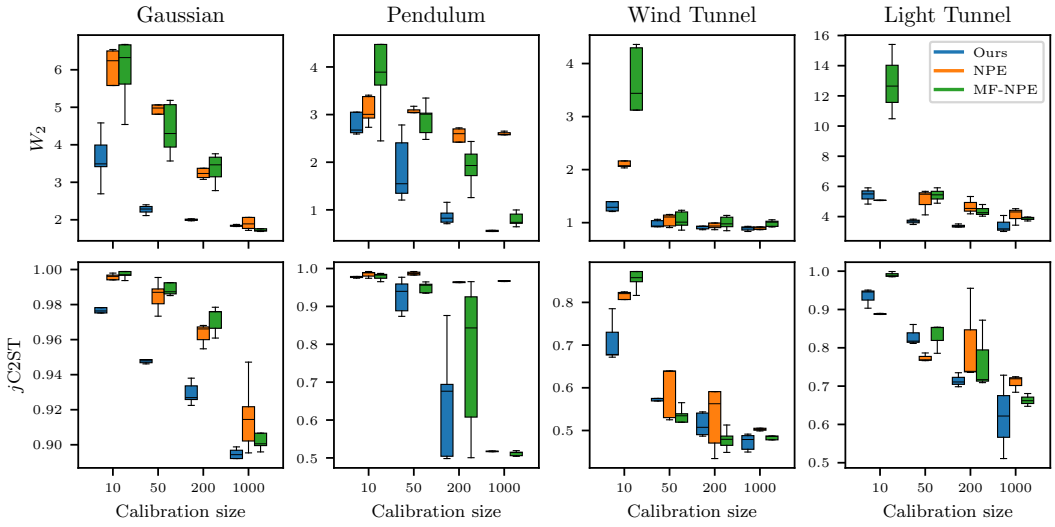


Figure 2: Wasserstein distance (top row, ↓ is better) and jC2ST (bottom row, ↓ is better) with respect to an increasing calibration set size $N_{\text{cal}} \in \{10, 50, 200, 1000\}$. Each boxplot shows the distribution of metric values across five independent runs, each using a different randomly chosen calibration set.

We compute W_2 between samples from the approximate and true joint distribution $p(\boldsymbol{\theta}, \boldsymbol{y})$ to provide a geometry-aware notion of distributional similarity.

Mean Squared Error (MSE). We also report the average mean squared error between M generated samples $\{\tilde{\boldsymbol{\theta}}_j^i\}_{1 \le i \le M}$ and the ground truth parameters $\{\boldsymbol{\theta}_j\}_{1 \le j \le N_{\text{test}}}$ for each observation $\boldsymbol{y}_j$ in $\mathcal{D}_{\text{test}}$.

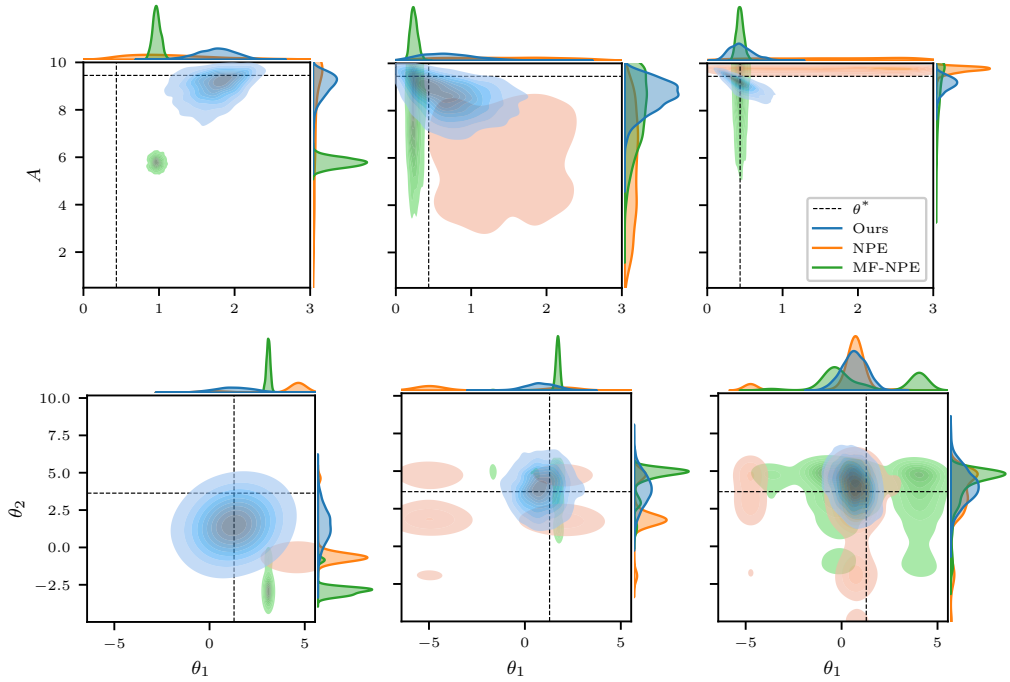
$$\text{MSE} = \frac{1}{N_{\text{test}}} \frac{1}{M} \sum_{j}^{N_{\text{test}}} \sum_{i}^{M} \|\tilde{\boldsymbol{\theta}}_j^i - \boldsymbol{\theta}_i\|_2^2 \, .$$

MSE is a good accuracy measure for unimodal posteriors, which is often the case in our experiments.

4.3 Results and Discussion

Figure 2 and Appendix Figure 4, first illustrate the impact of N_{cal} on posterior estimation. All metrics decrease as N_{cal} increases, highlighting the importance of having enough calibration data to correct misspecification. Note that for `WindTunnel` we sometimes observe jC2ST values below 0.5. This is due to an insufficient test set with respect to the dimensions of the problem. For this task, N_{test} was set to 5000, the maximal number of validation data points provided by the `causal-chamber` package. For a given ground truth couple $(\boldsymbol{\theta}^*, \boldsymbol{y}^*)$, Figure 3 illustrates the posterior density of the first two components of $\boldsymbol{\theta}$ for each method, for `Gaussian` and `Pendulum`. Similar plots for the other tasks are provided in the Appendix in Figures 5 (`WindTunnel`) and 6 (`LightTunnel`).

All experiments demonstrate that our method consistently outperforms both baselines, quantitatively and qualitatively. In Figure 2, we observe that even on the simple `Gaussian` task, our approach achieves better performance than plain NPE and fine-tuned `MFNPE`. This gap becomes more striking in Figure 3 (second row), where both NPE and `MFNPE` produce multimodal posteriors even when the true posterior is unimodal. In contrast, our method yields unimodal posteriors that are better calibrated and centered on the ground-truth parameter $\boldsymbol{\theta}^*$, and this even for a low calibration size. For more complex tasks, such as `Pendulum` and `LightTunnel`, NPE fails to capture the intricate dependencies between parameters and observations—even when $N_{\text{cal}} = 1000$. In these cases, our method achieves superior performance in both jC2ST and W_2, while also exhibiting lower variance across calibration seeds compared to `MFNPE` (Figure 2). Figure 3 further illustrates that `MFNPE` often produces overly sharp posterior distributions in one or more dimensions, yet fails to recover the true parameter $\boldsymbol{\theta}^*$. In contrast, our method consistently recovers $\boldsymbol{\theta}^*$ with higher accuracy. We hypothesize that this behavior stems from the `MFNPE` training procedure. `MFNPE` learns a neural encoder $h_\omega(\boldsymbol{x})$ to extract latent representations from simulated data $\boldsymbol{x}$. During fine-tuning, however,



(a) $N_{\text{cal}} = 10$ (b) $N_{\text{cal}} = 50$ (c) $N_{\text{cal}} = 200$

Figure 3: Kernel density estimates of joint and marginal samples for Tasks B (first row) and A (second row). For a given $\boldsymbol{y}^* \in \mathcal{D}_{\text{test}}$, we draw $\{\tilde{\boldsymbol{\theta}}_i\}_{1 \le i \le 2000}$, for each method and 3 calibration sizes $N_{\text{cal}} \in \{10, 50, 200\}$. Dotted black lines indicate the true parameter $\boldsymbol{\theta}^*$ that generated $\boldsymbol{y}^*$.

this encoder is evaluated on real observations $\boldsymbol{y}$, whose distribution differs from that of $\boldsymbol{x}$. This distributional shift leads to erroneous latent representations $h_\omega(\boldsymbol{y})$, and consequently to biased posteriors. While this issue diminishes as N_{cal} increases, it remains pronounced at small calibration sizes ($N_{\text{cal}} = 10$ or 50). The `Pendulum` task provides further evidence supporting this analysis. The misspecification arises from exponential damping, which predominantly affects the estimation of the oscillation amplitude A, as seen in Figure 3 (first row). In contrast, the estimation of the frequency ω_0 remains accurate, since it is less sensitive to the damping mismatch. Finally, in the real-world tasks (`WindTunnel` and `LightTunnel`), our approach yields substantially better results at small calibration sizes, particularly in terms of W_2, and remains competitive or superior for larger calibration sets. We provide additional visualizations of the posterior estimates for these tasks in Appendix B.

5 Conclusion

We tackled model misspecification in SBI by combining scarce real calibration data with abundant simulations. Our method builds a proposal posterior from both sources and refines it via flow matching, producing posterior estimates that are more accurate and better calibrated as compared to standard SBI baselines. Importantly, our proposal is also computationally efficient, as it can leverage off-the-shelf SBI posterior distributions as proposals, requiring only lightweight refinement with calibration data.

While severe misspecifications in high dimensions may still require larger calibration sets, our results show that even small amounts of real data can substantially improve inference quality. This highlights the promise of our framework as a practical and scalable way to bring SBI closer to real-world scientific applications, and opens exciting opportunities for richer proposal architectures, domain adaptation techniques, and deployment on large-scale simulators where misspecification is inevitable.

REFERENCES

Michael S. Albergo, Mark Goldstein, Nicholas M. Boffi, Rajesh Ranganath, and Eric Vanden-Eijnden. Stochastic interpolants with data-dependent couplings. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, 2024.

Michael Arbel and Alexander Zouaoui. Mlxp: A framework for conducting replicable experiments in python. In *Proceedings of the 2nd ACM Conference on Reproducibility and Replicability*, ACM REP '24, pp. 134–144, New York, NY, USA, 2024. Association for Computing Machinery.

Ayush Bharti, Louis Filstroff, and Samuel Kaski. Approximate Bayesian Computation with Domain Expert in the Loop. In *Proceedings of the 39th International Conference on Machine Learning*, ICML'22, 2022.

Jan Boelts, Jan-Matthis Lueckmann, Richard Gao, and Jakob H Macke. Flexible and efficient simulation-based inference for models of decision-making. *eLife*, 11:e77220, July 2022. Publisher: eLife Sciences Publications, Ltd.

Patrick Cannon, Daniel Ward, and Sebastian M. Schmon. Investigating the impact of model misspecification in neural simulation-based inference, 2022. URL https://arxiv.org/abs/2209.01845.

Edward Collett. *Field Guide to Polarization*, volume 5 of *SPIE Field Guides*. SPIE Press, The International Society for Optical Engineering, Bellingham, Washington, USA, 2005. ISBN 0-8194-5868-6.

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, December 2020.

Hugo Cui, Florent Krzakala, Eric Vanden-Eijnden, and Lenka Zdeborova. Analysis of learning a flow-based generative model from limited sample complexity. In *Proceedings of the 12th International Conference on Learning Representations, ICLR'24*, 2024.

Maximilian Dax, Stephen R. Green, Jonathan Gair, Jakob H. Macke, Alessandra Buonanno, and Bernhard Schölkopf. Real-Time Gravitational Wave Science with Neural Posterior Estimation. *Phys. Rev. Lett.*, 127(24):241103, 2021.

Michael Deistler, Jan Boelts, Peter Steinbach, Guy Moss, Thomas Moreau, Manuel Gloeckler, Pedro L. C. Rodrigues, Julia Linhart, Janne K. Lappalainen, Benjamin Kurt Miller, Pedro J. Gonçalves, Jan-Matthis Lueckmann, Cornelius Schröder, and Jakob H. Macke. Simulation-Based Inference: A Practical Guide, 2025. URL https://arxiv.org/abs/2508.12939.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. nflows: normalizing flows in PyTorch, November 2020. URL https://doi.org/10.5281/zenodo.4296287.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, 2024.

David Frazier, Christian Robert, and Judith Rousseau. Model Misspecification in Approximate Bayesian Computation: Consequences and Diagnostics. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82, 01 2020.

Masahiro Fujisawa, Takeshi Teshima, Issei Sato, and Masashi Sugiyama. γ-ABC: Outlier-Robust Approximate Bayesian Computation Based on a Robust Divergence Estimator . In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130, pp. 1783–1791, 2021.

Juan L. Gamella, Jonas Peters, and Peter Bühlmann. Causal chambers as a real-world physical testbed for AI methodology. *Nature Machine Intelligence*, 7(1):107–118, 2025.

Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free MCMC with amortized approximate ratio estimators. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 4239–4248, 2020.

Daolang Huang, Ayush Bharti, Amauri Souza, Luigi Acerbi, and Samuel Kaski. Learning robust statistics for simulation-based inference under model misspecification. *Advances in Neural Information Processing Systems*, 36:7289–7310, 2023.

Ryan Kelly, David J Nott, David T Frazier, David Warne, and Chris Drovandi. Misspecification-robust sequential neural likelihood for simulation-based inference. *Transactions on Machine Learning Research*, 2025(June), 2024.

Lingkai Kong, Haichuan Wang, Tonghan Wang, Guojun Xiong, and Milind Tambe. Composite flow matching for reinforcement learning with shifted-dynamics data, 2025. URL https://arxiv.org/abs/2505.23062.

Anastasia N. Krouglova, Hayden R. Johnson, Basile Confavreux, Michael Deistler, and Pedro J. Gonçalves. Multifidelity simulation-based inference for computationally expensive simulators, 2025. URL https://arxiv.org/abs/2502.08416.

Yexiong Lin, Yu Yao, and Tongliang Liu. Beyond optimal transport: Model-aligned coupling for flow matching, 2025. URL https://arxiv.org/abs/2505.23346.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *Proceedings of the 11th International Conference on Learning Representations, ICLR'23*, 2023.

David Lopez-Paz and Maxime Oquab. Revisiting Classifier Two-Sample Tests. In *Proceedings of the International Conference on Learning Representations, ICLR'17*, 2017.

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

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *Proceedings of the International Conference on Learning Representations, ICLR'19*, 2019.

George Papamakarios and Iain Murray. Fast ϵ -free Inference of Simulation Models with Bayesian Conditional Density Estimation. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Gabriel Peyré and Marco Cuturi. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Danilo Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning, ICML'15*, volume 37, pp. 1530–1538, 2015.

Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2005. ISBN 0387212396.

Marvin Schmitt, Paul-Christian Bürkner, Ullrich Köthe, and Stefan T. Radev. Detecting Model Misspecification in Amortized Bayesian Inference with Neural Networks. In *Pattern Recognition*, pp. 541–557. Springer Nature Switzerland, 2024.

Naoya Takeishi and Alexandros Kalousis. Physics-integrated variational autoencoders for robust and interpretable generative modeling. In *Advances in Neural Information Processing Systems*, volume 34, pp. 14809–14821. Curran Associates, Inc., 2021.

Stephen G. Walker. Bayesian inference with misspecified models. *Journal of Statistical Planning and Inference*, 143(10):1621–1633, 2013.

Zifan Wang, Alice Harting, Matthieu Barreau, Michael M. Zavlanos, and Karl H. Johansson. Source-guided flow matching, 2025. URL `https://arxiv.org/abs/2508.14807`.

Daniel Ward, Patrick Cannon, Mark Beaumont, Matteo Fasiolo, and Sebastian M Schmon. Robust neural posterior estimation and statistical model criticism. In *Advances in Neural Information Processing Systems*, 2022.

Antoine Wehenkel, Juan L. Gamella, Ozan Sener, Jens Behrmann, Guillermo Sapiro, Joern-Henrik Jacobsen, and marco cuturi. Addressing misspecification in simulation-based inference through data-driven calibration. In *Proceedings of the 42th International Conference on Machine Learning, ICML'25*, 2025.

Jonas Wildberger, Maximilian Dax, Simon Buchholz, Stephen Green, Jakob H Macke, and Bernhard Schölkopf. Flow matching for scalable simulation-based inference. In *Advances in Neural Information Processing Systems*, volume 36, pp. 16837–16864, 2023.

A Experimental Setup

We provide additional details on the training setup used across all experiments. For every task, we allocate a simulation budget of $N_{\text{sim}} = 5 \times 10^4$ samples and evaluate four calibration set sizes, $N_{\text{cal}} \in \{10, 50, 200, 1000\}$, which we denote by N_1, N_2, N_3, N_4.

During training, 20% of each calibration set is reserved for validation and the remaining 80% is used for training. We did not optimize over random seeds. All models were trained on a Nvidia RTX3060 GPU under 3 hours.

Data preprocessing. For tasks `Pendulum`, `WindTunnel`, and `LightTunnel` (uniform priors), we apply a logit transformation to map prior samples into $\mathbb{R}^p$. All datasets are standardized (z-scored) prior to training.

Calibration sets. For each N_{cal}, we generate 5 calibration sets by subsampling from a larger pool of calibration data. To reduce variance across runs, the sets are constructed in a nested fashion: denoting by $\mathcal{D}^i_{N_r}$ the i-th calibration set of size N_r, we enforce $\mathcal{D}^i_{N_r} \subset \mathcal{D}^i_{N_s}$ whenever $r < s$.

Neural Posterior Estimation (NPE). NPE is implemented with two components: a neural statistic estimator (NSE), $h_\omega(\mathbf{x})$, that encodes data into a low-dimensional representation, and a normalizing flow (NF) that maps a base distribution to the posterior. For task `Gaussian`, we use a standard neural spline flow (Durkan et al., 2019) and omit the NSE. For tasks `Pendulum`, `WindTunnel` and `LightTunnel`, we reuse the architectures and hyperparameters from Wehenkel et al. (2025).

Flow Matching. We use the architecture of Wildberger et al. (2023) as a backbone. For the $\boldsymbol{\theta}$-space flow $u_{\boldsymbol{\Theta}}$, conditioning on $\mathbf{x}$ is implemented through a task-specific embedding network. For the data-space flow $u_{\boldsymbol{X}}$, we employ the same architecture but with a separate embedding head for $(\mathbf{x}_t, t)$, equipped with positional encoding.

Evaluation details. The Wasserstein distance is computed using the `POT` package with default settings. For the C2ST, we train a classifier based on an MLP backbone, augmented with an embedding network for $\mathbf{y}$; the embedding architecture matches the one used for the normalizing flow. We apply 3-fold cross-validation and report the average validation accuracy across folds. By default, we balance the two classes - $C = 0$ (true samples) and $C = 1$ (generated samples)—but note that stratified K-fold can also be used to handle class imbalance.

A.1 Gaussian

The Gaussian task is defined as

$$p_{\boldsymbol{\Theta}} = \mathcal{N}(\mu_\theta, \Sigma_\theta), \quad p_{\boldsymbol{X}|\boldsymbol{\Theta}} = \mathcal{N}(A\boldsymbol{\theta} + b, \Sigma_x), \quad p_{\boldsymbol{Y}|\boldsymbol{\Theta}} = \mathcal{N}(C\boldsymbol{\theta} + d, \Sigma_y), \tag{10}$$

where

$\mu_\theta \in \mathbb{R}^3,\ \Sigma_\theta \in \mathbb{R}^{3\times3},\ A \in \mathbb{R}^{10\times3},\ b \in \mathbb{R}^{10},\ \Sigma_x \in \mathbb{R}^{10\times10},\ C \in \mathbb{R}^{10\times3},\ d \in \mathbb{R}^{10},\ \Sigma_y \in \mathbb{R}^{10\times10}.$

All parameters above are drawn randomly at the start of the experiment.

A.2 Pendulum

We follow the setup of Wehenkel et al. (2025, Appendix I.2). We sample $N = 200$ timesteps $t_i \sim \mathcal{U}[0, 10]$ and define the simulator

$$S: \begin{array}{rcl} \boldsymbol{\theta}, \epsilon & \longmapsto & [x_1, \dots, x_N]^T \\ \text{with } x_i & = & A\cos(\omega_0 t_i + \varphi) + \epsilon_i, \quad \varphi \sim \mathcal{U}[0, 2\pi],\ \ \epsilon_i \sim \mathcal{N}(0, \sigma^2), \end{array} \tag{11}$$

where $\boldsymbol{\theta} = [A, \omega_0]$ are the parameters of interest. The high-fidelity data-generating process (DGP) includes damping:

$$\text{DGP}: \begin{array}{rcl} \boldsymbol{\theta}, \epsilon & \longmapsto & [y_1, \dots, y_N]^T \\ \text{with } y_i & = & e^{-\alpha t_i} A\cos(\omega_0 t_i + \varphi) + \epsilon_i, \quad \alpha \sim \mathcal{U}[0, 1], \end{array} \tag{12}$$

where α encodes the friction coefficient. The prior is uniform: $p_{\boldsymbol{\Theta}} = \mathcal{U}[0, 3] \times \mathcal{U}[0.5, 10]$.

A.3 Wind Tunnel

We use the `load_out_0.5_osr_downwind_4` experiment from the `wt_intake_impulse_v1` dataset (Gamella et al., 2025). The data consist of 50-step time series measuring air pressure in the chamber after an impulse applied to the input fan. A hatch on the side controls an additional opening, which can be controlled with precision. The inference task is to predict its position $H \in [0, 45]$. We adopt model A2C3 from the `causalchamber` package as the simulator.

A.4 Light Tunnel

We use the light tunnel experiment `uniform_ap_1.8_iso_500.0_ss_0.005` from the `lt_camera_v1` dataset (Gamella et al., 2025). A camera at the rear-end of an elongated chamber captures a light source emitted from the other end passing through two linear polarizers. We refer the reader to (Gamella et al., 2025) for more details about the mechanistic model of the tunnel. The inference task consists in inferring the color of the light source ($(R, G, B) \in [0, 255]^3$) as well as the Malus law (Collett, 2005) coefficient $\alpha \in [0, 1]$. The prior over these variables is uniform. The coefficient is a function of the polarizer angle $\alpha = \cos^2(\phi_1 - \phi_2)$, which are given in the dataset. The misspecification is introduced by omitting some physical aspects, which are detailed in (Gamella et al., 2025, Appendix D.IV.2.2).

B Additional Plots & Metrics

We display here the MSE metric in Figure 4 and the KDE plots for tasks `WindTunnel` and `LightTunnel`, respectively in Figures 5 and 6.

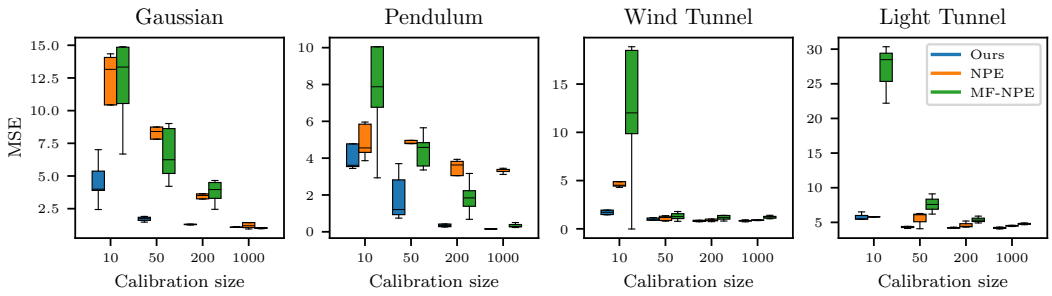
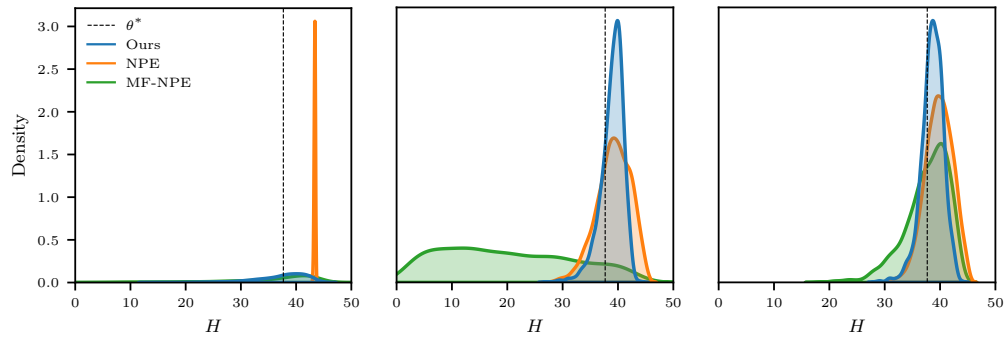
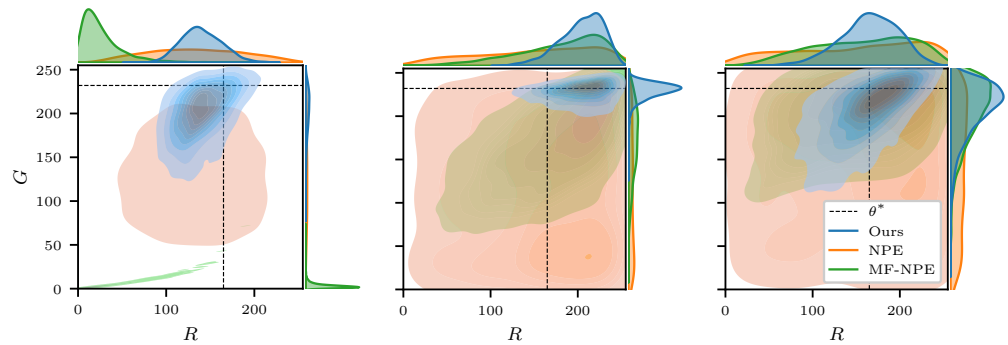


Figure 4: MSE with respect to an increasing calibration set size $N_{\text{cal}} \in \{10, 50, 200, 1000\}$. Each boxplot shows the distribution of MSE values across five independent runs, each using a different randomly chosen calibration set.



(a) $N_{\text{cal}} = 10$ (b) $N_{\text{cal}} = 50$ (c) $N_{\text{cal}} = 200$

Figure 5: Kernel density estimates of the learned posteriors for task `WindTunnel`. For a given $\boldsymbol{y}^* \in \mathcal{D}_{\text{test}}$, we draw $\{\tilde{\boldsymbol{\theta}}_i\}_{1 \le i \le 2000}$, for each method and 3 calibration sizes $N_{\text{cal}} \in \{10, 50, 200\}$. The dotted black line indicates the true parameter $\boldsymbol{\theta}^*$ that generated $\boldsymbol{y}^*$.



(a) $N_{\text{cal}} = 10$ (b) $N_{\text{cal}} = 50$ (c) $N_{\text{cal}} = 200$

Figure 6: Kernel density estimates of joint and marginal samples for task `LightTunnel`. We report the posterior densities for the first two coordinates of the parameter $[\theta_1, \theta_2] = (R, G)$. For a given $\boldsymbol{y}^* \in \mathcal{D}_{\text{test}}$, we draw $\{\tilde{\boldsymbol{\theta}}_i\}_{1 \le i \le 2000}$, for each method and 3 calibration sizes $N_{\text{cal}} \in \{10, 50, 200\}$. Dotted black lines indicate the true parameter $\boldsymbol{\theta}^*$ that generated $\boldsymbol{y}^*$.