ConDiSim: Conditional Diffusion Models for Simulation-Based Inference

Mayank Nautiyal
Science for Life Laboratory,
Dept. of Information Technology,
Uppsala University

Andreas Hellander
Dept. of Information Technology,
Uppsala University

Prashant Singh
Science for Life Laboratory,
Dept. of Information Technology,
Uppsala University

Abstract

We present ConDiSim, a conditional diffusion model for simulation-based inference in complex systems with intractable likelihoods. ConDiSim leverages denoising diffusion probabilistic models to approximate posterior distributions, consisting of a forward process that adds Gaussian noise to parameters, and a reverse process learning to denoise, conditioned on observed data. This approach effectively captures complex dependencies and multimodalities within posteriors. ConDiSim is evaluated across ten benchmark problems and two real-world test problems, where it demonstrates effective posterior approximation accuracy while maintaining computational efficiency and stability in model training. ConDiSim provides a robust and extensible framework for simulation-based inference, well suited to parameter estimation tasks that demand fast methods for handling noisy, time series observations.

1 INTRODUCTION

Statistical inference of model parameters from empirical observations is a fundamental challenge in scientific research, enabling researchers to derive meaningful insights from complex simulation models. These parameters govern the behavior of simulators that replicate real-world phenomena, providing a bridge between theoretical constructs and empirical observations (Lavin et al., 2021). Calibrating these parameters to ensure that simulator outputs align with observed data constitutes an inverse problem, formally defined within the framework of simulation-based inference (SBI) (Cranmer et al., 2020). Such inverse problems are often ill-posed, meaning they may lack unique or stable solutions, especially when the simulator is stochastic or when different parameter configurations produce similar observations (Sisson et al., 2018). To address these challenges, the Bayesian SBI places a prior distribution over the parameters and infers a posterior distribution conditioned on observed data. This posterior captures the full set of plausible solutions and enables principled uncertainty quantification, offering a more robust alternative to relying on single-point estimates.

Recent advances in machine learning have introduced powerful generative models capable of approximating complex data distributions without explicit likelihood computations. Among these, diffusion models have emerged as a particularly promising class (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020). They model data by learning to reverse a fixed noising process, gradually denoising samples from pure noise. Moreover, diffusion models naturally support amortized inference (Kingma and Welling, 2014): a single trained model approximates posteriors for any observation without per-instance optimization, enabling faster inference for new data.

This work introduces ConDiSim, a specialized posterior solver for SBI based on conditional DDPMs (Ho et al., 2020), with a lightweight FiLM-based architecture (Perez et al., 2018) that explicitly separates parameter denoising from observation conditioning. ConDiSim addresses common SBI challenges: limited simulation budgets from expensive simulators (particularly in systems biology and epidemiology with stiff ODE/PDE solvers), noisy and stochastic outputs, multimodal posteriors from nonlinearities and identifiability issues, and the need for specialized lightweight architectures to model the reverse diffusion process that integrates parameters, timesteps, and observations to recover diverse posterior structures. ConDiSim is particularly well-suited for SBI problems involving

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

learning low-dimensional posteriors from noisy, high-dimensional discrete-time observations.

Contributions: (i) We adapt conditional Denoising Diffusion Probabilistic Models (DDPMs) for SBI to learn complex posterior distributions under limited simulation budgets;

(ii) ConDiSim introduces a novel reverse diffusion architecture for SBI that employs time-dependent Feature-wise Linear Modulation (FiLM) to jointly condition on noisy parameters, diffusion timesteps, and observations for posterior estimation;

(iii) We empirically evaluate ConDiSim across benchmarks and real-world SBI tasks, analyzing performance under limited simulation budgets, noisy outputs, and multimodal posteriors. The source code implementing the approach is publicly available in our project GitHub repository.

Outline: The paper is organized as follows. Section 2 outlines the related work. Section 3 formally defines the SBI problem. Section 3.1 introduces the ConDiSim model, and Section 3.2 details its conditioning mechanism and sampling procedure, including the proposed architecture. Section 4 presents a comprehensive evaluation on benchmark and real-world problems, with additional results provided in the Appendix. Section 5 presents a discussion of the results, while Section 6 concludes the paper.

2 RELATED WORK

Simulation-based inference has advanced significantly in recent years, evolving from traditional statistical likelihood-free methods to contemporary deep learning frameworks. Classical techniques like Approximate Bayesian Computation (ABC) (Beaumont et al., 2002) and Sequential Monte Carlo-ABC (SMC-ABC) (Toni et al., 2009) offer theoretical convergence guarantees but may scale poorly to high dimensional problems (Sisson et al., 2018). While neural summary statistics (Jiang et al., 2017; Wrede et al., 2022) can reduce dimensionality, they introduce trade-offs: additional computational burden and the possibility of losing information necessary for accurate posterior estimation.

Neural Posterior Estimation (NPE) (Papamakarios and Murray, 2016) pioneered the use of deep generative models for amortized SBI, introducing conditional normalizing flows to model the posterior distribution, substantially improving accuracy and scalability over traditional methods. Sequential extensions like SNPE-A (Papamakarios and Murray, 2016), SNPE-B (Lueckmann et al., 2017), and SNPE-C (also known as Automatic Posterior Transformation or APT) (Greenberg et al., 2019) enhanced accuracy by iteratively refining the model to focus on relevant parameter regions. However, these sequential methods require multiple rounds of training and are constrained by the invertible transformations required in flow-based models. Recently, Flow Matching Posterior Estimation (FMPE) (Wildberger et al., 2023) introduced continuous normalizing flows based on conditional vector fields, overcoming architectural constraints and efficiently scaling to high-dimensional inference tasks. However, FMPE requires ODE solvers at inference time, introducing additional computational costs, and its performance depends critically on the choice of flow interpolant, which can be challenging to optimize under tight simulation budgets.

Beyond flow-based methods, other deep generative approaches to SBI include adversarial models. Generative Adversarial Training for SBI (GATSBI) (Ramesh et al., 2022) uses adversarial networks to model posteriors but suffers from training instability, mode collapse (Arjovsky and Bottou, 2017; Arjovsky et al., 2017), and poor performance in low-dimensional settings (Ramesh et al., 2022). More recently, Simformer (Gloeckler et al., 2024) was proposed as a general-purpose inference solver that approximates multiple conditional distributions (including likelihoods, posteriors, and mixed conditionals) within a single model, by combining score-based diffusion models with a transformer backbone. While this flexibility enables high-fidelity posterior estimation, it comes at the expense of quadratic computational cost in the observation size and treats parameters and observations as equivalent tokens, making it computationally heavy and harder to tune for specialized tasks.

In contrast, ConDiSim is a specialized posterior estimator based on conditional DDPMs, which routes all observation information through a dedicated condition encoder and a lightweight FiLM module, explicitly separating what is being inferred from what conditions the inference. This design avoids the quadratic attention cost, scales more favorably with observation size, and yields a lighter and easier-to-tune model that achieves competitive posterior accuracy with significantly lower training and inference overhead. This makes ConDiSim particularly suitable for settings like iterative model exploration, where many inference queries must be resolved quickly (Wrede and Hellander, 2019).

In this study, we evaluate ConDiSim against five distinct SBI approaches: NPE (Papamakarios and Murray, 2016), APT (Greenberg et al., 2019), GATSBI (Ramesh et al., 2022), FMPE (Wildberger et al., 2023), and Simformer (posterior inference mode only) (Gloeckler et al., 2024). Together, these methods span a wide spectrum of generative modeling paradigms, including discrete flow-based methods in both one-shot and sequential forms, continuous flow-based approaches, ad-

versarial models, and diffusion models in both discrete and continuous formulations. This selection enables a balanced evaluation that emphasizes efficiency and scalability in addition to posterior accuracy. For a broader overview of deep learning approaches to SBI, see (Zammit-Mangion et al., 2024; Radev et al., 2023a).

3 SIMULATION-BASED INFERENCE

Simulation-based inference (SBI) aims to infer the posterior distribution $p(\boldsymbol{\theta} \mid \mathbf{y})$ of model parameters $\boldsymbol{\theta}$, given observed data $\mathbf{y}$. From Bayes' theorem:

$$p(\boldsymbol{\theta} \mid \mathbf{y}) \propto p(\mathbf{y} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta}), \tag{1}$$

where $p(\mathbf{y} \mid \boldsymbol{\theta})$ is the likelihood and $p(\boldsymbol{\theta})$ is the prior distribution. In many scientific models, particularly those involving stochastic simulators, the likelihood function $p(\mathbf{y} \mid \boldsymbol{\theta})$ is either intractable or computationally expensive to evaluate, rendering likelihood-based methods such as Markov Chain Monte Carlo (MCMC) infeasible (Andrieu et al., 2003).

When the likelihood is inaccessible, a common strategy is to sample parameters from a prior, $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$, where $p(\boldsymbol{\theta})$ may be informed by domain knowledge or chosen to be noninformative, such as a broad uniform prior. For each sampled parameter, a corresponding observation is generated via the simulator, $\mathbf{y} \sim p(\mathbf{y} \mid \boldsymbol{\theta})$. This yields joint samples $(\boldsymbol{\theta}, \mathbf{y}) \sim p(\boldsymbol{\theta}, \mathbf{y})$, which can then be used to train generative models that approximate the posterior $p(\boldsymbol{\theta} \mid \mathbf{y})$ without requiring explicit likelihood evaluation.

3.1 ConDiSim: Conditional Diffusion Framework for SBI

We adapt conditional denoising diffusion probabilistic models (Ho et al., 2020) to SBI, where the goal is to approximate the posterior $p(\boldsymbol{\theta} \mid \mathbf{y})$ in likelihood-free settings using joint samples $(\boldsymbol{\theta}, \mathbf{y}) \sim p(\boldsymbol{\theta}, \mathbf{y})$. The model employs a forward diffusion process that gradually corrupts the parameters $\boldsymbol{\theta}$ with Gaussian noise until they follow a tractable prior distribution. The reverse process then learns to denoise these corrupted parameters, conditioned on observations $\mathbf{y}$, thereby recovering samples from the posterior distribution $p(\boldsymbol{\theta} \mid \mathbf{y})$.

In the *forward diffusion process*, we construct a discrete-time Markov chain $\{\boldsymbol{\theta}_t\}_{t=0}^{T}$, initialized at $\boldsymbol{\theta}_0 = \boldsymbol{\theta}$. At each step t, the new state $\boldsymbol{\theta}_t$ is derived from the old state $\boldsymbol{\theta}_{t-1}$ by perturbing it with Gaussian noise according to the transition kernel:

$$q(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1}) = \mathcal{N}\big(\boldsymbol{\theta}_t;\, \sqrt{1-\beta_t}\,\boldsymbol{\theta}_{t-1},\, \beta_t \mathbf{I}\big), \tag{2}$$

where $\beta_t \in (0,1)$ determines the noise level at each step, and $\mathbf{I}$ is the identity matrix. This step-wise transformation progressively corrupts $\boldsymbol{\theta}_0$, moving its mean towards zero and increasing its variance to unity, such that as $t \to T$, the state converges to $q(\boldsymbol{\theta}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Alternatively, $\boldsymbol{\theta}_t$ can be directly computed from $\boldsymbol{\theta}_0$ using the marginal distribution:

$$q(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_0) = \mathcal{N}\big(\boldsymbol{\theta}_t;\, \sqrt{\bar{\alpha}_t}\,\boldsymbol{\theta}_0,\, (1-\bar{\alpha}_t)\mathbf{I}\big), \tag{3}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ is the cumulative product of the noise scaling factors. This formulation allows efficient computation of $\boldsymbol{\theta}_t$ at any step t, directly linking $\boldsymbol{\theta}_t$ to $\boldsymbol{\theta}_0$. The joint forward distribution is given by:

$$q(\boldsymbol{\theta}_{1:T} \mid \boldsymbol{\theta}_0) = q(\boldsymbol{\theta}_T \mid \boldsymbol{\theta}_0) \prod_{t=2}^{T} q(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{\theta}_t, \boldsymbol{\theta}_0). \tag{4}$$

In the *reverse diffusion process*, latent states $\{\boldsymbol{\theta}_t\}_{t=T}^{0}$ are denoised from $\boldsymbol{\theta}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ to recover $\boldsymbol{\theta}_0$, with every step conditioned on observed data $\mathbf{y}$ (Saharia et al., 2023). The joint reverse distribution is formulated as:

$$p(\boldsymbol{\theta}_{0:T} \mid \mathbf{y}) = p(\boldsymbol{\theta}_T) \prod_{t=1}^{T} p(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{\theta}_t, \mathbf{y}). \tag{5}$$

Training the reverse model involves maximizing the log-posterior of the latent variables, given the observations:

$$\mathcal{L}_{\text{ConDiSim}} = \max \log p(\boldsymbol{\theta}_0 \mid \mathbf{y}), \tag{6}$$

$$\text{where,} \quad \log p(\boldsymbol{\theta}_0 \mid \mathbf{y}) = \log \int p(\boldsymbol{\theta}_{0:T} \mid \mathbf{y})\, d\boldsymbol{\theta}_{1:T}. \tag{7}$$

However, evaluating $\log p(\boldsymbol{\theta}_0 \mid \mathbf{y})$ directly is intractable, as it entails integrating over the full trajectory of intermediate latent states. Instead, we employ a variational approach by introducing the joint forward distribution $q(\boldsymbol{\theta}_{1:T} \mid \boldsymbol{\theta}_0)$. Using Jensen's inequality, the lower bound on the log-posterior is:

$$\log p(\boldsymbol{\theta}_0 \mid \mathbf{y}) \;\geq\; \mathbb{E}_{q(\boldsymbol{\theta}_{1:T} \mid \boldsymbol{\theta}_0)}\left[\log \frac{p(\boldsymbol{\theta}_{0:T} \mid \mathbf{y})}{q(\boldsymbol{\theta}_{1:T} \mid \boldsymbol{\theta}_0)}\right]. \tag{8}$$

Applying Eqs. (4) and (5) to Eq. (8) leads to the following decomposition:

$$\mathcal{L}_{\text{ConDiSim}} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{denoise}}, \tag{9}$$

where, the reconstruction term, $\mathcal{L}_{\text{recon}} = \mathbb{E}_q[\log p(\boldsymbol{\theta}_0 \mid \boldsymbol{\theta}_1, \mathbf{y})]$, evaluates how well the original parameters are recovered from the final denoised state. The prior term, $\mathcal{L}_{\text{prior}} = -D_{\text{KL}}(q(\boldsymbol{\theta}_T \mid \boldsymbol{\theta}_0) \,\|\, p(\boldsymbol{\theta}_T))$, enforces consistency with the prior at the final noised step. The denoising term, $\mathcal{L}_{\text{denoise}} = -\sum_{t=2}^{T} \mathbb{E}_q[D_{\text{KL}}(q(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{\theta}_t, \boldsymbol{\theta}_0) \,\|\, p(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{\theta}_t, \mathbf{y}))]$, aligns reverse transitions with

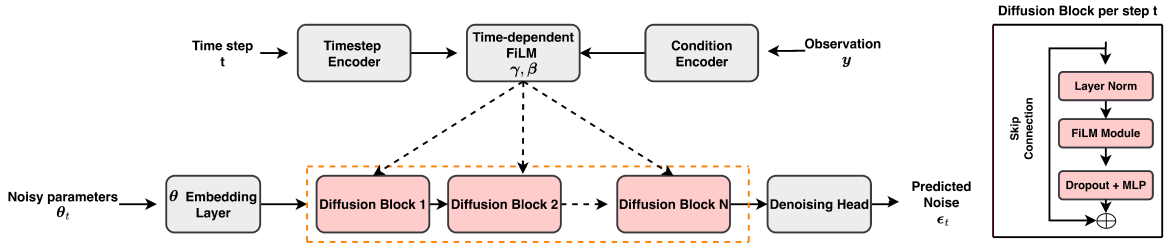


Figure 1: **ConDiSim Architecture**: The reverse diffusion process is implemented via a stack of FiLM-modulated diffusion blocks, each conditioned on a shared context vector derived from the observation $\mathbf{y}$ and diffusion timestep t, enabling the noise predictor $\boldsymbol{\epsilon}_\phi(\boldsymbol{\theta}_t, \mathbf{y}, t)$ to remain observation-aware and time-aware at every denoising step.

the forward process. For sufficiently large T, the reconstruction and prior terms vanish (Ho et al., 2020), leaving the denoising KL as the dominant training objective (i.e., $\mathcal{L} \approx \mathcal{L}_{\text{denoise}}$):

$$\mathcal{L} = \sum_{t=2}^{T} \mathbb{E}_q \big[D_{\text{KL}} \big(q(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{\theta}_t, \boldsymbol{\theta}_0) \parallel p_\phi(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{\theta}_t, \mathbf{y}) \big) \big]. \tag{10}$$

The first term in the KL divergence, $q(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{\theta}_t, \boldsymbol{\theta}_0)$, can be expressed using Bayes' theorem as $q(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{\theta}_t, \boldsymbol{\theta}_0) = \frac{q(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_0)\, q(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{\theta}_0)}{q(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_0)}$. From Eq. (2) and Eq. (3), each term is Gaussian, so the resulting distribution is also Gaussian: $q(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{\theta}_t, \boldsymbol{\theta}_0) = \mathcal{N}(\boldsymbol{\theta}_{t-1}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$, where:

$$\begin{aligned} \boldsymbol{\mu}_q &= \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\, \boldsymbol{\theta}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\, \boldsymbol{\theta}_t, \\ \boldsymbol{\Sigma}_q &= \frac{(1-\bar{\alpha}_{t-1})\alpha_t}{1-\bar{\alpha}_t}\, \mathbf{I}. \end{aligned} \tag{11}$$

To eliminate the explicit dependence on $\boldsymbol{\theta}_0$, we substitute $\boldsymbol{\theta}_0 = (\boldsymbol{\theta}_t - \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon})/\sqrt{\bar{\alpha}_t}$ (with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$) into Eq. (3), reparameterizing $\boldsymbol{\mu}_q$ to yield a noise-parameterized expression for the mean:

$$\boldsymbol{\mu}_q(\boldsymbol{\theta}_t, t) = \tfrac{1}{\sqrt{\alpha_t}} \Big(\boldsymbol{\theta}_t - \tfrac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\, \boldsymbol{\epsilon} \Big). \tag{12}$$

Assuming the learned reverse process preserves the covariance structure ($\boldsymbol{\Sigma}_p = \boldsymbol{\Sigma}_q$) (Ho et al., 2020), the KL divergence in Eq. (10) reduces to the squared Euclidean distance between the means:

$$\mathcal{L} = \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{\theta}_{1:T} \mid \boldsymbol{\theta}_0)} \Big[\big\| \boldsymbol{\mu}_q(\boldsymbol{\theta}_t, \boldsymbol{\theta}_0) \;-\; \boldsymbol{\mu}_p(\boldsymbol{\theta}_t, \mathbf{y}, t) \big\|^2 \Big]. \tag{13}$$

From Eq. (12), the mean of learned reverse process can be expressed as:

$$\boldsymbol{\mu}_p(\boldsymbol{\theta}_t, \mathbf{y}, t) = \tfrac{1}{\sqrt{\alpha_t}} \Big(\boldsymbol{\theta}_t - \tfrac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\, \boldsymbol{\epsilon}_\phi(\boldsymbol{\theta}_t, \mathbf{y}, t) \Big), \tag{14}$$

where, $\boldsymbol{\epsilon}_\phi(\boldsymbol{\theta}_t, \mathbf{y}, t)$ is a neural network that, given the noisy latent $\boldsymbol{\theta}_t$, conditioning $\mathbf{y}$, and time step t, predicts the noise. Substituting the mean parameterizations from Eqs.(12) and (14) into the loss of Eq.(13) and discarding all constant factors, yields the compact noise prediction objective:

$$\mathcal{L}_{\text{ConDiSim}} \approx \mathbb{E}_{t,\, \boldsymbol{\epsilon},\, (\boldsymbol{\theta}, \mathbf{y}) \sim p(\boldsymbol{\theta}, \mathbf{y})} \Big[\big\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi(\boldsymbol{\theta}_t, \mathbf{y}, t) \big\|^2 \Big]. \tag{15}$$

3.2 Conditioning and Sampling

The objective in Eq. (15) requires a denoising network that predicts the noise $\boldsymbol{\epsilon}$ from the noisy state $\boldsymbol{\theta}_t$, conditioned on the observation $\mathbf{y}$ and the diffusion timestep t. Preserving this dual conditioning throughout the denoising trajectory is essential to align with $p(\boldsymbol{\theta} \mid \mathbf{y})$. A naive scheme that feeds $[\boldsymbol{\theta}_t; \mathbf{y}; t]$ into a standard network often performs poorly in SBI, where $\dim(\mathbf{y}) \gg \dim(\boldsymbol{\theta})$, input-level concatenation forces the network to disentangle $\mathbf{y}$ and t from $\boldsymbol{\theta}_t$ in a single step, yielding an ill-conditioned coupling that dilutes the posterior signal and destabilizes training.

ConDiSim instead treats $\boldsymbol{\theta}_t$ as the primary state and routes all observation and timestep information through dedicated encoders $f_y(\mathbf{y})$ and $f_t(t)$, whose embeddings are concatenated into a context vector $\mathbf{c} = [\, f_y(\mathbf{y});\; f_t(t)\,]$. A lightweight MLP maps $\mathbf{c}$ to per-block FiLM parameters $(\boldsymbol{\gamma}_\ell(\mathbf{c}), \boldsymbol{\beta}_\ell(\mathbf{c}))$, which modulate each diffusion block via $\text{FiLM}_\ell(\mathbf{h}_\ell) = \boldsymbol{\gamma}_\ell(\mathbf{c}) \odot \mathbf{h}_\ell + \boldsymbol{\beta}_\ell(\mathbf{c})$ (Perez et al., 2018). This explicitly separates what is being inferred (parameters) from what conditions the inference (observations and timestep), ensuring the noise predictor $\boldsymbol{\epsilon}_\phi(\boldsymbol{\theta}_t, \mathbf{y}, t)$ remains observation-aware and time-aware at every denoising step, while scaling more favorably with observation size than attention-based alternatives (Gloeckler et al., 2024). A schematic of the full architecture is shown in Fig. 1.

To perform amortized posterior inference, ConDiSim draws N samples $\boldsymbol{\theta}_T^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and iteratively denoises them from $t = T$ down to $t = 1$ using the

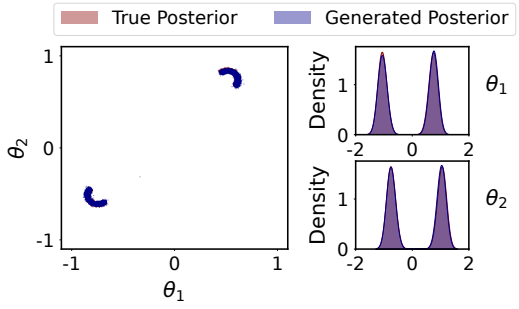
learned denoiser $\hat{\boldsymbol{\epsilon}}_\phi$:

$$\boldsymbol{\theta}_{t-1}^{(i)} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{\theta}_t^{(i)} - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\hat{\boldsymbol{\epsilon}}_\phi(\boldsymbol{\theta}_t^{(i)}, \mathbf{y}, t)\right) + \sigma_t\,\mathbf{z}_t, \tag{16}$$

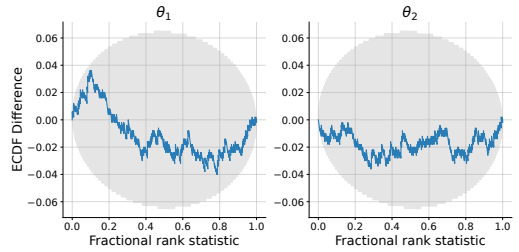
where $\mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\sigma_t = \sqrt{\beta_t}$. To balance fidelity to $\mathbf{y}$ with sample diversity, classifier-free guidance (Ho and Salimans, 2021) is applied by interpolating between conditional and unconditional noise predictions:

$$\hat{\boldsymbol{\epsilon}}_\phi = (1+\lambda)\,\boldsymbol{\epsilon}_\phi(\boldsymbol{\theta}_t, \mathbf{y}, t) - \lambda\,\boldsymbol{\epsilon}_\phi(\boldsymbol{\theta}_t, t), \tag{17}$$

where $\lambda \geq 0$ controls the guidance strength. The resulting set $\{\boldsymbol{\theta}_0^{(i)}\}_{i=1}^N$ provides approximate samples from the posterior $p(\boldsymbol{\theta} \mid \mathbf{y})$.



(a) Posterior Plot.

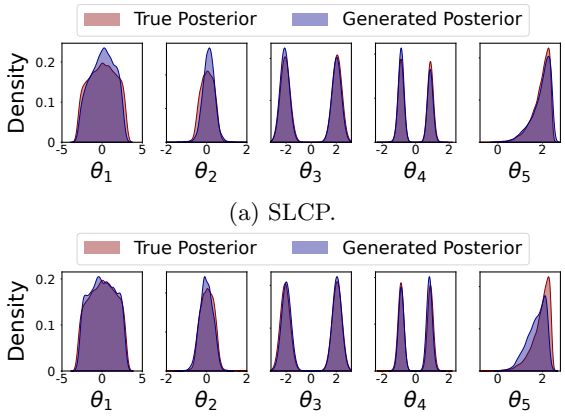


(b) ECDF Plot.

Figure 2: **Two Moons:** Comparison of generated and reference posterior distributions, with the ECDF plot representing the cumulative distribution of fractional rank statistics from posterior draws.

4 EXPERIMENTS

We evaluate ConDiSim on 10 benchmark problems from the *sbibm* suite (Lueckmann et al., 2021), and two real-world problems. For each benchmark, we constrain the training to simulation budgets of 10,000, 20,000, and 30,000 simulations, and infer the posterior by drawing 10,000 samples from the learned model. We assess posterior quality using Maximum Mean Discrepancy (MMD), the Classifier Two-Sample Test (C2ST) (Friedman, 2003), and Empirical Cumulative Distribution Function (ECDF) diagnostics from the *BayesFlow* framework (Radev et al., 2023b). C2ST results are



(a) SLCP.

(b) SLCP Distractors.

Figure 3: **SLCP:** Comparison of generated vs. reference posteriors for SLCP and SLCP Distractors.

reported in the main text; MMD scores and full ECDF plots for all benchmarks are provided in Appendix G. Complete hyperparameter configurations for all methods are detailed in Appendix C.

4.1 Benchmark Posterior Analysis

In this section, we examine three representative benchmarks from the *sbibm* suite: *Two Moons*, *SLCP* (Simple Likelihood Complex Posterior), and *SLCP Distractors* (SLCP with added high-dimensional noise), chosen for their distinct and complementary challenges in posterior inference. The Two Moons benchmark poses the challenge of capturing a bi-modal posterior with a distinctive crescent geometry. As depicted in Figure 2a, ConDiSim accurately recovers both modes and the complex posterior structure. The ECDF plots for θ_1 and θ_2 (Figure 2b) demonstrate that the inferred distributions are well-calibrated, with the majority of rank statistics falling within the 90% confidence bands; a minor deviation for θ_1 suggests a slight degree of overconfidence.

The SLCP benchmark is particularly challenging as the posterior exhibits a range of complex behaviours, including uniform, Gaussian, multimodal, and skewed distributions across its parameters. Its variant, SLCP Distractors, further tests the model's robustness by incorporating uninformative noise dimensions, thereby simulating scenarios with heavily corrupted data. Figures 3a and 3b illustrate that ConDiSim is able to capture these complex posterior landscapes with high fidelity, and also effectively isolate the informative parameters from the distractors. The rank ECDF plots for both SLCP variants (Figures 14a and 14b in the Appendix) also remain centered within the confidence intervals, confirming that the model reliably captures

parameter variability even under noisy conditions. Detailed descriptions of these benchmarks, are provided in Appendix B.

Table 1 reports C2ST scores across the 10 benchmarks, with complementary posterior plots provided in Appendix E. On tasks such as Two Moons, ConDiSim achieves performance comparable to Simformer, APT, and NPE, while requiring substantially shorter training times (Table 2). Although ConDiSim does not always attain the top C2ST score, the method offers a favorable trade-off between efficiency and posterior quality, which is crucial under limited simulation budgets. Notably, the SLCP task remains challenging for all methods, highlighting the inherent difficulty of accurately recovering high-dimensional, multi-modal posteriors. Moreover, C2ST can be overly sensitive to small errors in individual dimensions, making it less reliable in high-dimensional settings. For example, as shown in Figure 3b, ConDiSim closely matches the true posterior in most dimensions, with deviations concentrated in the final parameter. To provide additional insight, we report the MMD metric for each benchmark problem in the Supplementary Table 4.

4.2 Application to Real-World Scenarios

To test the real-world adaptability of ConDiSim, we evaluate it on two real-world simulation models: (i) the *Hodgkin–Huxley* (HH) model (Gloeckler et al., 2024), which simulates neuronal action potentials by modeling membrane voltage dynamics governed by sodium, potassium, and leak currents; and (ii) a *Genetic Oscillator* model (Vilar et al., 2002), a relatively high-dimensional gene regulatory network exhibiting oscillatory behavior driven by a positive–negative feedback loop mimicking a circadian clock, encompassing 9 species undergoing 18 reactions parameterized by 15 reaction rate constants, with inference on three key species: the activator protein A, the repressor protein R, and their complex C, which is formed through the interaction of A and R. Full descriptions of both models are provided in Appendix B.

Unlike the SBI benchmarking problems, the true posterior distributions for these real-world models are unknown. To assess ConDiSim, we use well motivated parameter combinations from literature to generate corresponding synthetic observations (e.g., parameter combination where the genetic oscillator results in stable oscillations). Performing inference based on these observations enables us to evaluate the inferred posterior by comparing its concentration around the true parameter value. We also perform a posterior predictive check to validate the model. In this procedure, five posterior samples are randomly selected and used to simulate corresponding observations, which are then compared with the observed data. This check provides an additional measure of the consistency and reliability of the inferred posteriors.

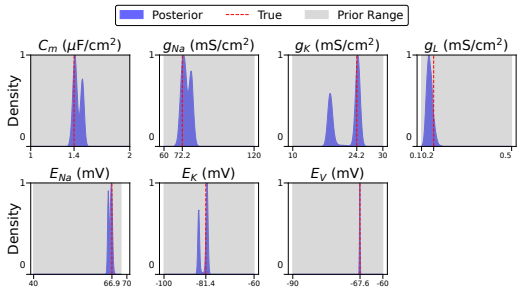
For the HH model, Figure 4a shows that ConDiSim effectively infers a posterior distribution concentrated around the true parameter values, while Figure 4b confirms that the simulated voltage trajectories closely match the reference signals. The parameter estimation task for the genetic oscillator is more challenging due to broad and varied parameter priors. Figure 5a indicates that the peak of the posterior distribution is near the true parameter values. Figure 5b illustrates both the true and generated trajectories for three selected species, obtained by randomly sampling three posterior sets and plotting the corresponding simulations. The trajectories align well with the true data, with slight deviations emerging over time due to the inherent stochastic nature of the model, as expected. Figure 6 (Appendix) depicts multiple trajectories simulated using the true parameter combination, which closely match the inferred results. In both cases, the trajectories are very close to each other during initial time steps, and develop small deviations owing to model stochasticity as time steps evolve. The plots highlight the system's inherent variability, and also validate the inferred solution. Comparing Figs. 4 and 5 with Figs. 8 and 9, ConDiSim infers tighter, more accurate posteriors than Simformer, underscoring its suitability for inference problems involving stochastic, discrete-time observations.

5 DISCUSSION

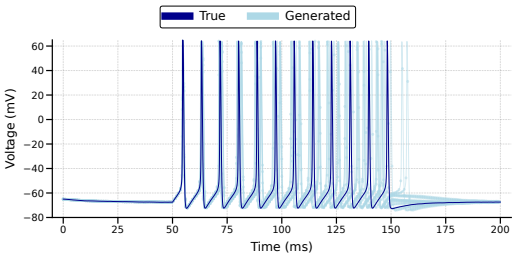
In this section, we compare ConDiSim against established SBI methods (Section 5.1) and analyze its behavior along three practical axes: *scalability* with problem dimensionality (Section 5.2), *robustness* to simulator noise or misspecification (Section 5.3), and *reliability* in capturing posterior uncertainty (Section 5.4). This analysis provides practical guidance for deploying diffusion-based inference methods by identifying regimes where ConDiSim excels and scenarios where alternative approaches may be more suitable.

5.1 ConDiSim vs. Other SBI Methods

Tables 1 and 4 show that diffusion-based methods (Simformer, ConDiSim) often achieve higher posterior accuracy than flow-based (NPE, APT, FMPE) and GAN-based (GATSBI) approaches, particularly on problems with multimodal or highly nonlinear posteriors. Simformer attains the best overall performance on several benchmarks, owing to its transformer-based diffusion architecture that tokenizes parameters and observations into a joint sequence and applies multi-head

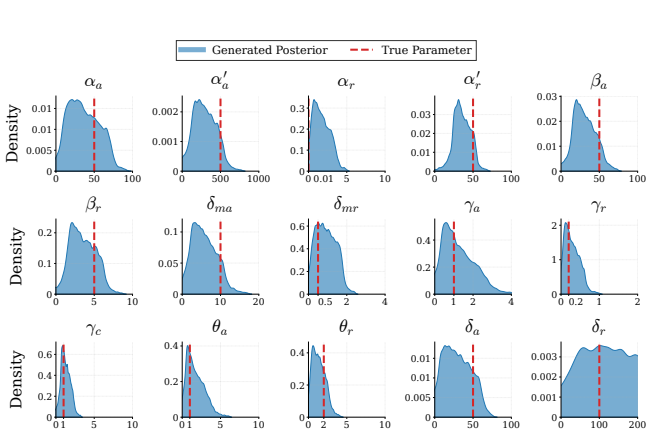


(a) Approximated Posterior distribution.

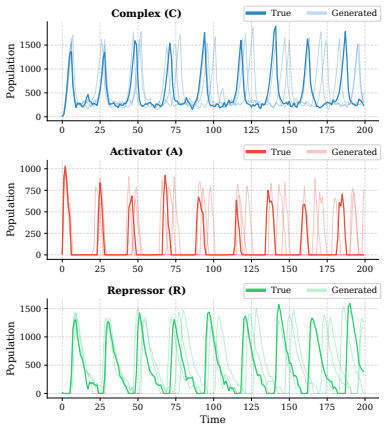


(b) Posterior Predictive Voltage Traces.

Figure 4: Posterior inference for the Hodgkin–Huxley model.



(a) Approximated Posterior distribution.



(b) Posterior Predictive Trajectories.

Figure 5: Posterior inference for the Genetic Oscillator model.

Table 1: C2ST scores across SBI benchmarks (mean ± std.) at varying simulation budgets; best in **bold**, second-best underlined.

Method	SLCP Distractors			Bernoulli GLM Raw			Lotka Volterra		
Budget	10,000	20,000	30,000	10,000	20,000	30,000	10,000	20,000	30,000
NPE	0.9714 ±0.0144	0.9468 ±0.0157	0.9359 ±0.0127	0.9354 ±0.0093	0.9114 ±0.0082	0.9065 ±0.0105	0.9999 ±0.0001	0.9999 ±0.0000	0.9999 ±0.0000
APT	0.9823 ±0.0055	0.9451 ±0.0158	0.9130 ±0.0301	0.8455 ±0.0116	0.8596 ±0.0152	0.8461 ±0.0135	0.9999 ±0.0001	0.9999 ±0.0000	0.9999 ±0.0001
GATSBI	0.9995 ±0.0007	0.9978 ±0.0016	0.9994 ±0.0004	0.9998 ±0.0001	0.9999 ±0.0001	0.9999 ±0.0000	0.9999 ±0.0001	0.9999 ±0.0000	1.0 ±0.0000
Simformer	**0.9198** ±0.0451	**0.8814** ±0.0433	**0.8717** ±0.0453	0.9814 ±0.0106	0.8600 ±0.0452	0.7750 ±0.0473	0.9999 ±0.0003	0.9999 ±0.0012	1.0000 ±0.0002
FMPE	0.9713 ± 0.0117	0.9779 ± 0.0095	0.9387 ± 0.0377	0.9742 ± 0.0113	0.6661 ± 0.0307	0.6653 ± 0.0377	0.9999 ± 0.0178	0.9999 ± 0.0345	0.9999 ± 0.0018
ConDiSim	0.9456 ±0.0245	0.9222 ±0.02609	0.8832 ±0.08519	**0.6673** ±0.0668	**0.6099** ±0.0272	**0.6041** ±0.0237	0.9995 ±0.0001	0.9995 ±0.0005	0.9995 ±0.0003

Method	Gaussian Mixture			Gaussian Linear			Gaussian Linear Uniform			SLCP		
Budget	10,000	20,000	30,000	10,000	20,000	30,000	10,000	20,000	30,000	10,000	20,000	30,000
NPE	0.6118 ±0.0155	0.5830 ±0.0240	0.5818 ±0.0158	0.5472 ±0.0076	**0.5238** ±0.0112	0.5266 ±0.0037	0.5562 ±0.0147	**0.5292** ±0.0068	**0.5256** ±0.0066	0.8996 ±0.0206	0.8502 ±0.0174	0.8546 ±0.0213
APT	0.5457 ±0.0058	0.5425 ±0.0124	0.5467 ±0.0050	**0.5367** ±0.0038	0.5293 ±0.0052	**0.5258** ±0.0047	**0.5401** ±0.0115	0.5302 ±0.0069	0.5264 ±0.0075	0.8164 ±0.0295	0.6882 ±0.0342	**0.6473** ±0.0343
GATSBI	0.7474 ±0.0364	0.6977 ±0.0260	0.7234 ±0.0599	0.9930 ±0.0038	0.9916 ±0.0046	0.9929 ±0.0059	0.9989 ±0.0004	0.9976 ±0.0016	0.9988 ±0.0010	0.9687 ±0.0286	0.9436 ±0.0203	0.9723 ±0.0140
Simformer	**0.5166** ±0.0210	**0.5100** ±0.0090	**0.5126** ±0.0070	0.6130 ±0.0281	0.6226 ±0.0398	0.5866 ±0.0430	0.6501 ±0.0519	0.6207 ±0.0519	0.5780 ±0.0211	**0.7661** ±0.0685	**0.6201** ±0.0490	0.7192 ±0.0721
FMPE	0.7894 ± 0.0254	0.7872 ± 0.0049	0.5821 ± 0.0292	0.9605± 0.0019	0.9716 ± 0.0023	0.9708 ± 0.0025	0.9218 ± 0.0122	0.9100 ± 0.0174	0.9082 ± 0.0178	0.9698 ± 0.0132	0.9686 ± 0.0139	0.9686 ± 0.0139
ConDiSim	0.6776 ±0.0283	0.6420 ±0.0247	0.6087 ±0.0262	0.5970 ±0.0228	0.5733 ±0.0250	0.5655 ±0.0117	0.7030 ±0.0469	0.6641±0.0498	0.5990±0.0501	0.92995 ±0.00680	0.87111 ±0.07559	0.72778±0.03867

Method	Two Moons			Bernoulli GLM			SIR		
Budget	10,000	20,000	30,000	10,000	20,000	30,000	10,000	20,000	30,000
NPE	0.5749 ±0.0331	0.5931 ±0.0329	0.5411 ±0.0195	0.9121 ±0.0338	0.8988 ±0.0306	0.8898 ±0.0565	0.8378 ±0.0396	0.9122 ±0.0207	0.9405 ±0.0125
APT	**0.5189** ±0.0080	0.5202 ±0.0114	0.5147 ±0.0125	0.8404 ±0.0107	0.8328 ±0.0156	0.8353 ±0.0164	0.9433 ±0.0076	0.9434 ±0.0061	0.9457 ±0.0068
GATSBI	0.7770 ±0.0630	0.6634 ±0.0503	0.7467 ±0.1036	0.9999 ±0.0001	0.9999 ±0.0000	0.9997 ±0.0005	0.9990 ±0.0015	0.9912 ±0.0161	0.9838 ±0.0157
Simformer	0.5273 ±0.0229	**0.5190** ±0.0174	**0.5098** ±0.0084	0.6584 ±0.0381	**0.6131** ±0.0225	0.6154 ±0.0265	0.9378 ±0.0053	0.9520 ±0.0064	0.9478 ±0.0049
FMPE	0.8051 ± 0.0506	0.7745 ± 0.0261	0.5706 ± 0.0209	0.9266 ± 0.0288	0.9118 ± 0.0376	0.8431 ± 0.0393	0.9573 ± 0.0013	0.8454 ± 0.0014	**0.8101** ± 0.0020
ConDiSim	0.5716 ± 0.1088	0.5495 ± 0.0802	0.5226 ± 0.0158	**0.6276** ± 0.0432	0.6228 ± 0.0339	**0.5884** ± 0.0398	**0.7614**± 0.0457	**0.8142** ± 0.0397	0.8271 ± 0.0426

Table 2: Training time (in minutes) for different methods across simulation budgets (mean ± std. dev.); best in **bold**, second-best underlined.

Method	Two Moons			Bernoulli GLM			SIR		
Budget	10,000	20,000	30,000	10,000	20,000	30,000	10,000	20,000	30,000
NPE	7.04 ±0.21	14.21 ±0.04	20.95 ±0.37	12.17 ±0.17	24.96 ±0.64	37.43 ±0.22	20.10 ±5.45	29.62 ±3.74	44.92 ±9.24
APT	90.31 ±8.99	186.17 ±13.79	268.28 ±32.88	101.39 ±7.89	197.30 ±9.32	267.38 ±21.93	40.15 ±3.90	73.74 ±6.25	112.51 ±2.51
GATSBI	117.67 ±2.43	164.90 ±57.10	249.50 ±105.05	118.62 ±0.18	179.69 ±49.96	253.48 ±113.38	227.16 ±1.44	241.24 ±7.58	250.47 ±2.22
Simformer	3.96 ±0.04	7.36 ±0.11	10.73 ±0.25	17.87 ±0.18	35.05 ±0.52	52.81 ±0.91	4.24 ±0.29	7.67 ±0.38	10.68 ±0.45
FMPE	2.63 ± 0.05	4.73± 1.11	10.65 ± 0.04	2.65 ± 0.05	5.75 ± 1.43	7.64 ± 1.38	**2.78 ± 0.15**	**6.39 ± 0.19**	**8.94 ± 2.94**
ConDiSim	**2.17** ±0.68	**4.22** ±1.09	**7.02** ±0.30	**3.518** ±0.19	**6.65** ±0.93	**9.56** ±2.99	2.15 ±0.39	4.47 ±0.62	6.18 ±1.27

Method	SLCP Distractors			Bernoulli GLM Raw			Lotka Volterra		
Budget	10,000	20,000	30,000	10,000	20,000	30,000	10,000	20,000	30,000
NPE	21.92 ±2.41	43.92 ±2.58	64.38 ±7.13	25.39 ±2.09	61.43 ±2.61	85.94 ±6.55	80.67 ±14.61	160.92 ±42.82	188.04 ±29.46
APT	26.30 ±0.36	49.96 ±2.73	74.31 ±2.54	21.87 ±1.28	40.00 ±1.37	60.94 ±3.00	45.52 ±4.85	96.15 ±13.77	144.33 ±18.33
GATSBI	74.37 ±2.59	100.59 ±2.39	102.30 ±0.51	114.76 ±1.19	115.22 ±1.43	144.10 ±24.15	78.91 ±6.32	75.73 ±0.70	74.99 ±0.20
Simformer	15.98 ±1.22	31.98 ±2.34	47.85 ±1.86	16.96 ±0.51	32.46 ±1.89	48.01 ±1.44	5.0 ±0.34	9.23 ±0.54	13.64 ±0.21
FMPE	2.76 ± 0.08	**3.08** ± 1.76	12.97 ± 0.87	**2.44** ± 0.44	**4.42** ± 1.37	**8.05** ± 1.38	3.63 ± 0.42	**6.97** ± 0.03	**9.55** ± 0.72
ConDiSim	**2.02** ±0.27	5.16 ±0.84	**8.44** ±1.02	3.22 ±0.72	7.22 ±0.50	10.67 ±0.69	**2.42** ±0.21	7.19 ±0.97	11.19 ±2.31

Method	Gaussian Mixture			Gaussian Linear			Gaussian Linear Uniform			SLCP		
Budget	10,000	20,000	30,000	10,000	20,000	30,000	10,000	20,000	30,000	10,000	20,000	30,000
NPE	19.56 ±2.47	42.88 ±4.54	64.03 ±6.51	12.03 ±0.80	24.69 ±1.63	38.79 ±1.23	16.55 ±0.85	30.90 ±2.77	49.42 ±1.49	42.97 ±6.35	97.70 ±6.66	117.61 ±4.12
APT	76.27 ±7.02	167.88 ±15.99	240.13 ±22.74	70.47 ±0.88	135.38 ±5.75	203.27 ±5.97	78.61 ±2.11	158.57 ±9.32	219.44 ±11.83	93.40 ±1.26	150.29 ±11.10	216.30 ±21.38
GATSBI	114.07 ±2.11	111.30 ±0.28	137.46 ±19.16	48.19 ±0.24	48.70 ±0.84	49.12 ±0.49	43.27 ±0.36	66.43 ±5.70	69.10 ±0.15	162.72 ±14.73	196.19 ±119.42	99.39 ±5.63
Simformer	3.84 ±0.04	7.15 ±0.09	10.48 ±0.21	18.18 ±0.13	35.46 ±0.17	55.46 ±1.13	17.89 ±0.23	33.47 ±0.40	51.24 ±0.89	4.23 ±0.01	7.67 ±0.13	11.32 ±0.40
FMPE	**2.16**± 0.08	5.36 ± 0.78	8.31 ± 1.32	1.78 ± 0.05	**4.40** ± 0.78	**5.94** ± 4.31	**2.15** ± 0.22	**4.30** ± 3.86	**8.71** ± 2.12	**1.36** ± 0.14	**3.71** ± 2.57	**6.34** ± 2.33
ConDiSim	2.22 ±0.41	**4.64** ±0.42	**7.02** ±0.98	**1.59** ±0.23	4.75 ±1.04	7.58 ±1.84	2.26 ±0.92	4.68 ±1.36	8.06 ±2.43	1.70 ±0.58	4.10 ±1.68	8.23 ±0.66

self-attention, allowing every variable to attend to every other variable to capture complex, high-dimensional dependencies; however, this comes at the cost of substantially longer training times (Table 2). In contrast, ConDiSim achieves competitive accuracy with significantly lower training overhead by replacing costly attention computations with a lightweight FiLM-based conditioning mechanism, while its iterative denoising incurs no practical slowdown in inference. NPE and APT remain strong baselines on simpler or lower-dimensional tasks, where their one-shot or sequential flow-based strategies are sufficient, while GATSBI is hampered by the well-known instabilities of adversarial training.

Notably, Table 1 shows that ConDiSim outperforms all competing methods on problems involving noisy, discrete observations such as Bernoulli GLM, Bernoulli GLM Raw, and SIR, where the FiLM conditioning mechanism proves particularly effective. This makes ConDiSim especially well-suited for SBI problems in computational biology, where parameter inference of stochastic simulators with time-series observations is often encountered (e.g., the genetic oscillator test problem, as noted before).

5.2 Scalability

We evaluate ConDiSim across a spectrum of SBI benchmarks, ranging from low-dimensional problems (Two Moons, GMM, SIR; 2 parameters) to higher-dimensional settings (Gaussian Linear, Bernoulli GLM; 10 parameters, HH Model; 7 parameters, Genetic Oscillator; 15 parameters). C2ST scores in Table 1 and posterior visualizations (Figs. 2a–13) consistently show that ConDiSim maintains high posterior fidelity as dimensionality and model complexity increase, while training times in Table 2 confirm favorable computational scaling across problem sizes. The 15-dimensional Genetic Oscillator is a strong test case: despite its large parameter space and the noisy, stochastic nature of the simulator, ConDiSim preserves inference accuracy without substantial degradation, highlighting the effectiveness of its lightweight FiLM-based architecture.

5.3 Robustness

In many real-world problems, observational data are inherently imperfect and often contaminated by measurement noise, confounded by irrelevant features, or available only in raw, unprocessed form. Consequently, it is essential to evaluate how inference methods perform under such adverse conditions. We assess ConDiSim's robustness on two challenging benchmarks: SLCP Distractors, where 92 uninformative variables are added to the observation vector to simulate high-dimensional nuisance features, and Bernoulli GLM Raw, which operates directly on raw binary observations rather than relying on handcrafted summary statistics, emulating realistic scenarios with noisy, unprocessed measurements. The genetic oscillator test problem is also noisy, simulated using the Stochastic Simulation Algorithm (SSA) (Gillespie, 2007). As shown in Table 1 and Figures 3b–13b, ConDiSim maintains accurate posterior recovery across both settings, demonstrating resilience to observation noise and feature redundancy that demonstrates its suitability for practical scientific

applications where data are high-dimensional, unprocessed, or corrupted by measurement uncertainty.

5.4 Reliability

Beyond predictive accuracy, it is crucial to verify that the learned posteriors are statistically well-calibrated, meaning that the reported uncertainty faithfully captures the true variability of the inference problem. To assess calibration, we use Simulation-Based Calibration (SBC) (Talts et al., 2018), which is a standard diagnostic that evaluates whether the rank statistics of posterior samples follow their expected uniform distribution. The empirical cumulative distribution function (ECDF) differences shown in Figures 2b–14 remain within the expected bounds, with fluctuations centered around zero, indicating that ConDiSim produces unbiased and well-calibrated posteriors across benchmarks. Such calibration is essential for trustworthy inference in scientific applications, where both accurate point estimates and reliable uncertainty quantification are required.

6 CONCLUSIONS

The paper introduces ConDiSim, a DDPM-based conditional diffusion model for simulation-based inference. Across ten benchmark tasks and two real-world problems, ConDiSim effectively handles ill-posed inference, multimodal and high-dimensional posteriors, and demonstrates robustness with respect to raw, unprocessed observations, and in the presence of noise and distractors. It also adapts naturally to diverse data modalities, while offering substantially faster training time than competing methods. We demonstrate that ConDiSim is particularly suited for inference of stochastic simulators with time series responses, where it outperforms other methods and delivers accuracy, well-calibrated posterior estimates, and fast amortized inference.

Acknowledgements

Computations and data handling were enabled by the Berzelius resource provided by the Knut and Alice Wallenberg Foundation at the National Supercomputer Centre, and by the Alvis resource provided by Chalmers e-Commons at Chalmers. Andreas Hellander and Prashant Singh acknowledge funding from the Swedish Research Council under grant agreements no. 2023-05167 and no. 2023-05593, respectively. Prashant Singh also acknowledges support from the Knut and Alice Wallenberg Foundation through the Wallenberg AI, Autonomous Systems and Software Program (WASP) and the Data-Driven Life Science (DDLS) NEST project "TIMED". The authors thank Li Ju and Aleksandr Karakulev for their valuable feedback on early drafts of the paper.

References

Akesson, M., Singh, P., Wrede, F., and Hellander, A. (2022). Convolutional neural networks as summary statistics for approximate bayesian computation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(6):3353–3365.

Anderson, B. D. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326.

Andrieu, C., Freitas, N., Doucet, A., and Jordan, M. (2003). An introduction to mcmc for machine learning. *Machine Learning*, 50:5–43.

Arjovsky, M. and Bottou, L. (2017). Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR.

Beaumont, M. A., Zhang, W., and Balding, D. J. (2002). Approximate bayesian computation in population genetics. *Genetics*, 162(4):2025–2035.

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

Friedman, J. H. (2003). On multivariate goodness-of-fit and two-sample testing. *Statistical Problems in Particle Physics, Astrophysics, and Cosmology*, 1:311.

Gillespie, D. T. (2007). Stochastic simulation of chemical kinetics. *Annu. Rev. Phys. Chem.*, 58(1):35–55.

Gloeckler, M., Deistler, M., Weilbach, C. D., Wood, F., and Macke, J. H. (2024). All-in-one simulation-based inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 15735–15766. PMLR.

Greenberg, D., Nonnenmacher, M., and Macke, J. (2019). Automatic posterior transformation for likelihood-free inference. In *International Conference on Machine Learning*, pages 2404–2414. PMLR.

Hang, T., Gu, S., Li, C., Bao, J., Chen, D., Hu, H., Geng, X., and Guo, B. (2023). Efficient diffusion training via min-snr weighting strategy. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7407–7417.

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

Ho, J. and Salimans, T. (2021). Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

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

Jiang, B., Wu, T.-y., Zheng, C., and Wong, W. H. (2017). Learning summary statistic for approximate bayesian computation via deep neural network. *Statistica Sinica*, pages 1595–1618.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014*.

Lavin, A., Zenil, H., Paige, B., Krakauer, D. C., Gottschlich, J. E., Mattson, T. G., Anandkumar, A., Choudry, S., Rocki, K., Baydin, A. G., Prunkl, C. E. A., Isayev, O., Peterson, E. J., McMahon, P. L., Macke, J. H., Cranmer, K., Zhang, J., Wainwright, H. M., Hanuka, A., Veloso, M. M., Assefa, S. A., Zheng, S., and Pfeffer, A. (2021). Simulation intelligence: Towards a new generation of scientific methods. *ArXiv*, abs/2112.03235.

Lopez-Paz, D. and Oquab, M. (2017). Revisiting classifier two-sample tests. In *International Conference on Learning Representations*.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Lueckmann, J.-M., Boelts, J., Greenberg, D., Goncalves, P., and Macke, J. (2021). Benchmarking simulation-based inference. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 343–351. PMLR.

Lueckmann, J.-M., Goncalves, P. J., Bassetto, G., Öcal, K., Nonnenmacher, M., and Macke, J. H. (2017). Flexible statistical inference for mechanistic models of neural dynamics. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Papamakarios, G. and Murray, I. (2016). Fast ε-free inference of simulation models with bayesian conditional density estimation. *Advances in Neural Information Processing Systems*, 29.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. (2018). Film: visual reasoning with a general conditioning layer. AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Radev, S. T., Schmitt, M., Pratz, V., Picchini, U., Köthe, U., and Bürkner, P.-C. (2023a). Jana: Jointly amortized neural approximation of complex bayesian models. In *Uncertainty in Artificial Intelligence*, pages 1695–1706. PMLR.

Radev, S. T., Schmitt, M., Schumacher, L., Elsemüller, L., Pratz, V., Schälte, Y., Köthe, U., and Bürkner, P.-C. (2023b). BayesFlow: Amortized Bayesian workflows with neural networks. *Journal of Open Source Software*, 8(89):5702.

Ramesh, P., Lueckmann, J.-M., Boelts, J., Tejero-Cantero, Á., Greenberg, D. S., Goncalves, P. J., and Macke, J. H. (2022). GATSBI: Generative adversarial training for simulation-based inference. In *International Conference on Learning Representations*.

Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., and Norouzi, M. (2023). Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4713–4726.

Sisson, S. A., Fan, Y., and Beaumont, M. (2018). *Handbook of approximate Bayesian computation*. CRC press.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France. PMLR.

Song, Y., Garg, S., Shi, J., and Ermon, S. (2020). Sliced score matching: A scalable approach to density and score estimation. In Adams, R. P. and Gogate, V., editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 574–584. PMLR.

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

Säilynoja, T., Bürkner, P.-C., and Vehtari, A. (2022). Graphical test for discrete uniformity and its applications in goodness-of-fit evaluation and multiple sample comparison. *Statistics and Computing*, 32(2).

Talts, S., Betancourt, M., Simpson, D., Vehtari, A., and Gelman, A. (2018). Validating bayesian inference algorithms with simulation-based calibration. *arXiv preprint arXiv:1804.06788*.

Tejero-Cantero, A., Boelts, J., Deistler, M., Lueckmann, J.-M., Durkan, C., Gonçalves, P. J., Green-

berg, D. S., and Macke, J. H. (2020). sbi: A toolkit for simulation-based inference. *Journal of Open Source Software*, 5(52):2505.

Toni, T., Welch, D., Strelkowa, N., Ipsen, A., and Stumpf, M. P. (2009). Approximate bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31):187–202.

Vilar, J. M. G., Kueh, H. Y., Barkai, N., and Leibler, S. (2002). Mechanisms of noise-resistance in genetic oscillators. *Proceedings of the National Academy of Sciences*, 99(9):5988–5992.

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

Wildberger, J., Dax, M., Buchholz, S., Green, S., Macke, J. H., and Schölkopf, B. (2023). Flow matching for scalable simulation-based inference. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 16837–16864. Curran Associates, Inc.

Wrede, F., Eriksson, R., Jiang, R., Petzold, L., Engblom, S., Hellander, A., and Singh, P. (2022). Robust and integrative bayesian neural networks for likelihood-free parameter inference. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE.

Wrede, F. and Hellander, A. (2019). Smart computational exploration of stochastic gene regulatory network models using human-in-the-loop semi-supervised learning. *Bioinformatics*, 35(24):5199–5206.

Zammit-Mangion, A., Sainsbury-Dale, M., and Huser, R. (2024). Neural methods for amortized inference. *Annual Review of Statistics and Its Application*, 12.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary Material: ConDiSim-Conditional Diffusion Models for Simulation-Based Inference

A CONNECTION BETWEEN CONDISIM AND SCORE-BASED DIFFUSION MODELS

In this section, we establish a formal theoretical connection between ConDiSim, a DDPM based diffusion model, and score based diffusion models. We demonstrate its equivalence to a variance preserving stochastic differential equation (SDE) by deriving the continuous time limit of ConDiSim's discrete forward process, providing a unified perspective on diffusion based generative modeling.

A.1 Forward Process and Continuous-Time Limit

ConDiSim employs a discrete time forward process, defined as:

$$\boldsymbol{\theta}_t = \sqrt{1-\beta_t}\,\boldsymbol{\theta}_{t-1} + \sqrt{\beta_t}\,\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), \tag{18}$$

where $\boldsymbol{\theta}_t$ represents the state at time step t, β_t is the noise schedule, and $\boldsymbol{\epsilon}$ is standard Gaussian noise. To convert this discrete time process to continuous time, we consider the limit as the number of time steps $T \to \infty$ and define a small time step $\Delta t = \frac{1}{T} \to 0$. We introduce a continuous noise schedule $\beta(t)$ such that $\beta_t \approx \beta(t)\Delta t$. Using a first-order Taylor expansion for small β_t, we approximate $\sqrt{1-\beta_t} \approx 1 - \frac{1}{2}\beta(t)\Delta t$. Substituting into the forward process:

$$\boldsymbol{\theta}_t = \left(1 - \frac{1}{2}\beta(t)\Delta t\right)\boldsymbol{\theta}_{t-1} + \sqrt{\beta(t)\Delta t}\,\boldsymbol{\epsilon}. \tag{19}$$

Defining the increment $\Delta\boldsymbol{\theta}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}$, and noting that the noise term $\sqrt{\beta(t)\Delta t}\,\boldsymbol{\epsilon}$ corresponds to a Wiener process increment $\Delta\mathbf{W} \sim \mathcal{N}(\mathbf{0}, \Delta t\mathbf{I})$, we rewrite the equation as:

$$\Delta\boldsymbol{\theta}_t = -\frac{1}{2}\beta(t)\boldsymbol{\theta}_{t-1}\Delta t + \sqrt{\beta(t)}\,\Delta\mathbf{W}. \tag{20}$$

In the continuous-time limit ($\Delta t \to 0$), we replace $\Delta\boldsymbol{\theta}_t \to d\boldsymbol{\theta}_t$, $\Delta t \to dt$, $\Delta\mathbf{W} \to d\mathbf{w}_t$, and $\boldsymbol{\theta}_{t-1} \simeq \boldsymbol{\theta}_t$, yielding the following stochastic differential equation (SDE):

$$d\boldsymbol{\theta}_t = -\frac{1}{2}\beta(t)\boldsymbol{\theta}_t\,dt + \sqrt{\beta(t)}\,d\mathbf{w}_t, \tag{21}$$

where $\mathbf{w}_t$ denotes standard Brownian motion and $d\mathbf{w}_t \sim \mathcal{N}(0, dt\,\mathbf{I})$. Comparing Equation 21 with the general form of the stochastic differential equations (SDEs):

$$d\boldsymbol{\theta}_t = \mathbf{f}(\boldsymbol{\theta}_t, t)\,dt + g(t)\,d\mathbf{w}_t, \tag{22}$$

where, $\mathbf{f}(\boldsymbol{\theta}_t, t)$ is the deterministic drift term and $g(t)$ is the scalar valued diffusion coefficient. Equation (21) is a special case of the general SDE, with $\mathbf{f}(\boldsymbol{\theta}_t, t) = -\frac{1}{2}\beta(t)\boldsymbol{\theta}_t$ and $g(t) = \sqrt{\beta(t)}$; this is known as the Variance-Preserving SDE (VP-SDE) (Song et al., 2021) and corresponds to an Ornstein–Uhlenbeck (OU) process. The solution to the VP-SDE can be derived using Itô calculus and is given by:

$$\boldsymbol{\theta}_t = \sqrt{\alpha_t}\,\boldsymbol{\theta}_0 + \sqrt{1-\alpha_t}\,\boldsymbol{\epsilon}, \quad \text{where } \alpha_t = \exp\left(-\int_0^t \beta(s)\,ds\right), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}). \tag{23}$$

This expression represents the marginal distribution $q(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_0)$ under the continuous diffusion process and serves as the continuous time analog of the discrete DDPM forward process:

$$\boldsymbol{\theta}_t = \sqrt{\alpha_t}\,\boldsymbol{\theta}_0 + \sqrt{1-\alpha_t}\,\boldsymbol{\epsilon}, \quad \alpha_t = \prod_{i=1}^{t}(1-\beta_i). \tag{24}$$

Thus, the VP-SDE generalizes the discrete Markov chain of ConDiSim to continuous time.

A.2 Reverse Diffusion Process and Score Matching

Given a general forward SDE as in Equation 22, the associated *reverse-time SDE* describes the dynamics when running the process backward in time (from $t = T$ to $t = 0$) and corresponds to the reverse diffusion process of ConDiSim. Following (Anderson, 1982), the general form of the reverse SDE is:

$$d\boldsymbol{\theta}_t = \left[\mathbf{f}(\boldsymbol{\theta}_t, t) - g^2(t)\,\nabla_{\boldsymbol{\theta}_t} \log q_t(\boldsymbol{\theta}_t)\right] dt + g(t)\, d\mathbf{w}_t, \tag{25}$$

where $q_t(\boldsymbol{\theta}_t)$ is the marginal probability density of $\boldsymbol{\theta}_t$ at time t and $d\mathbf{w}_t$ is Brownian motion in reverse time. For the VP-SDE (Equation (21)), the reverse SDE becomes:

$$d\boldsymbol{\theta}_t = \left[-\frac{1}{2}\beta(t)\,\boldsymbol{\theta}_t - \beta(t)\,\nabla_{\boldsymbol{\theta}_t} \log q_t(\boldsymbol{\theta}_t)\right] dt + \sqrt{\beta(t)}\, d\mathbf{w}_t. \tag{26}$$

In this formulation, all terms are known except for the score function $s(\boldsymbol{\theta}_t, t) = \nabla_{\boldsymbol{\theta}_t} \log q_t(\boldsymbol{\theta}_t)$, which is crucial for simulating the reverse process. Approximating this score is a standard optimization problem known as score matching, which is well-studied in probabilistic sampling, especially in the context of Langevin dynamics. However, since the true score depends on the intractable marginal $q_t(\boldsymbol{\theta}_t)$, we instead introduce a parameterized approximation $s_\phi(\boldsymbol{\theta}_t, t)$ and learn it by minimizing the Fisher divergence (explicit score matching loss):

$$\phi^* = \arg\min_{\phi}\ \mathbb{E}_t\, \mathbb{E}_{q_t(\boldsymbol{\theta}_t)} \left[\left\| s_\phi(\boldsymbol{\theta}_t, t) - \nabla_{\boldsymbol{\theta}_t} \log q_t(\boldsymbol{\theta}_t)\right\|^2\right]. \tag{27}$$

To address the intractability of the true score, several methods have been developed, including implicit score matching (Hyvärinen and Dayan, 2005), sliced score matching (Song et al., 2020), and denoising score matching (Vincent, 2011). To align the training objective with ConDiSim, we adopt denoising score matching (DSM), which leverages the tractable conditional score $\nabla_{\boldsymbol{\theta}_t} \log q_t(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_0)$. The corresponding objective is given by:

$$\phi^* = \arg\min_{\phi}\ \mathbb{E}_t\, \mathbb{E}_{q_0(\boldsymbol{\theta}_0)}\, \mathbb{E}_{q_t(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_0)} \left[\left\| s_\phi(\boldsymbol{\theta}_t, t) - \nabla_{\boldsymbol{\theta}_t} \log q_t(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_0)\right\|^2\right]. \tag{28}$$

Given the closed-form solution for the forward process in Equation (23), the score function $\nabla_{\boldsymbol{\theta}_t} \log q_t(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_0)$ becomes tractable and can be written as:

$$\nabla_{\boldsymbol{\theta}_t} \log q_t(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_0) = -\frac{\boldsymbol{\theta}_t - \sqrt{\alpha(t)}\,\boldsymbol{\theta}_0}{1-\alpha(t)}. \tag{29}$$

Reparameterizing $\boldsymbol{\theta}_t = \sqrt{\alpha_t}\,\boldsymbol{\theta}_0 + \sqrt{1-\alpha_t}\,\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$, the DSM objective becomes:

$$\phi^* = \arg\min_{\phi}\ \mathbb{E}_t\, \mathbb{E}_{q_0(\boldsymbol{\theta}_0)}\, \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})} \left[\left\| s_\phi(\boldsymbol{\theta}_t, t) + \frac{1}{\sqrt{1-\alpha_t}}\,\boldsymbol{\epsilon}\right\|^2\right]. \tag{30}$$

This objective aligns closely with the ConDiSim training objective, where the score is approximated as:

$$s_\phi(\boldsymbol{\theta}_t, t) \approx -\frac{1}{\sqrt{1-\alpha_t}}\,\hat{\boldsymbol{\epsilon}}_\phi(\boldsymbol{\theta}_t, t), \tag{31}$$

with $\hat{\boldsymbol{\epsilon}}_\phi(\boldsymbol{\theta}_t, t)$ being the predicted noise. This equivalence demonstrates that ConDiSim's training objective is a rescaled variant of the DSM objective, unifying the discrete DDPM framework with continuous-time score-based diffusion models. This analysis reveals that, although ConDiSim is formulated as a discrete DDPM, it is fundamentally equivalent to a score-based diffusion model in the continuous-time limit. The VP-SDE provides a principled framework for analyzing ConDiSim's dynamics: the forward diffusion process can be understood through the Fokker–Planck equation, which characterizes convergence properties of the marginal distributions, while the reverse diffusion process connects naturally to annealed Langevin dynamics.

B SBI PROBLEM DESCRIPTION

In this section, we describe the general setup for simulation-based inference (SBI) used across all benchmarking problems, following the framework of Lueckmann et al. (2021). The goal in each case is to estimate the posterior distribution $p(\boldsymbol{\theta} \mid \mathbf{y})$ over parameters $\boldsymbol{\theta}$ given observed data $\mathbf{y}$, and each problem is defined by the following components:

(i) **Prior Distribution** $p(\boldsymbol{\theta})$: encodes initial beliefs over the parameters before any data is observed.

(ii) **Simulation Model** $\mathcal{M}$: a stochastic simulator that generates synthetic observations $\mathbf{y}$ given parameters $\boldsymbol{\theta}$, providing samples from the joint distribution $p(\boldsymbol{\theta}, \mathbf{y})$ without requiring explicit likelihood evaluation.

(iii) **Dimensionality**: the parameter space $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$ and observation space $\mathbf{y} \in \mathbb{R}^{d_y}$ vary across problems, directly impacting the complexity and computational cost of posterior estimation.

B.1 Two Moons

This parameter inference task is designed to test the performance of inference methods under multi-modal conditions, featuring a posterior distribution with both global (bi-modal) and local (crescent-shaped) characteristics.

(i) **Prior:** $\boldsymbol{\theta} \sim \mathcal{U}(-1, 1)$.

(ii) **Simulator:** $\mathbf{y}|\boldsymbol{\theta} = r \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \end{bmatrix} + 0.25 \begin{bmatrix} -|\theta_1 + \theta_2|/\sqrt{2} \\ (-\theta_1 + \theta_2)/\sqrt{2} \end{bmatrix}$, where $\alpha \sim \mathcal{U}(-\pi/2, \pi/2)$ and $r \sim \mathcal{N}(0.1, 0.01^2)$.

(iii) **Dimensionality:** $\boldsymbol{\theta} \in \mathbb{R}^2, \mathbf{y} \in \mathbb{R}^2$.

B.2 Gaussian Mixture

This task challenges inference methods to estimate the shared mean of a mixture of Gaussian distributions with distinct covariances, thereby assessing their performance in handling multi-modal and overlapping data distributions.

(i) **Prior:** $\boldsymbol{\theta} \sim \mathcal{U}(-10, 10)$.

(ii) **Simulator:** $\mathbf{y}|\boldsymbol{\theta} = 0.5\,(\mathcal{N}(\boldsymbol{\theta}, \mathbf{I}) + \mathcal{N}(\boldsymbol{\theta}, 0.01\mathbf{I}))$.

(iii) **Dimensionality:** $\boldsymbol{\theta} \in \mathbb{R}^2, \mathbf{y} \in \mathbb{R}^2$.

B.3 Gaussian Linear

This task involves estimating the mean vector $\boldsymbol{\theta}$ of a multivariate Gaussian distribution with a fixed and known covariance matrix. Since both the prior and the likelihood are Gaussian, they form a conjugate pair, resulting in a Gaussian posterior distribution. This conjugacy makes the task an ideal benchmark for evaluating inference methods, as it allows for a clear assessment of their ability to accurately capture known posterior distributions.

(i) **Prior:** $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, 0.1\mathbf{I})$.

(ii) **Simulator:** Generates data $\mathbf{y}$ using $\mathbf{y}|\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}, 0.1\mathbf{I})$.

(iii) **Dimensionality:** $\boldsymbol{\theta} \in \mathbb{R}^{10}, \mathbf{y} \in \mathbb{R}^{10}$.

B.4 Gaussian Linear Uniform

This task is a variant of the Gaussian Linear problem, where the mean parameter $\boldsymbol{\theta}$ is assigned a uniform prior distribution $\boldsymbol{\theta} \sim \mathcal{U}(-1, 1)$ instead of a Gaussian prior. The introduction of a uniform prior imposes bounded support on the parameters, altering the posterior distribution from a Gaussian to a truncated Gaussian. This modification tests the inference method's ability to handle non-Gaussian priors and to accurately capture the resulting posterior distributions.

B.5 SLCP

The SLCP task poses a demanding inference scenario in which the likelihood is simple by construction, yet the resulting posterior exhibits complex, non-linear dependencies among the five-dimensional parameter vector $\boldsymbol{\theta}$. The simulator applies non-linear transformations to $\boldsymbol{\theta}$ to generate 8-dimensional observations, producing intricate posterior structures that serve as a strong test of an inference method's ability to capture non-linear parameter relationships.

(i) **Prior:** $\boldsymbol{\theta} \sim \mathcal{U}(-3, 3)$.

(ii) **Simulator:** Outputs data $\mathbf{y}|\boldsymbol{\theta} = (y_1, \ldots, y_4)$, where:

$$y_i \sim \mathcal{N}\left(\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}, \begin{bmatrix}\theta_3^2 & \tanh(\theta_5)\theta_3\theta_4\\ \tanh(\theta_5)\theta_3\theta_4 & \theta_4^2\end{bmatrix}\right).$$

(iii) **Dimensionality:** $\boldsymbol{\theta} \in \mathbb{R}^5, \mathbf{y} \in \mathbb{R}^8$.

B.6 SLCP with Distractors

An extension of the SLCP task that incorporates additional non-informative dimensions (distractors) into the observed data. This setup evaluates the robustness of inference methods when faced with high-dimensional data containing redundant information.

(i) **Prior:** $\boldsymbol{\theta} \sim \mathcal{U}(-3, 3)$.

(ii) **Simulator:** Outputs data $y|\theta = (y_1, \ldots, y_{100})$, where a permutation function $p(y)$ reorders y with a fixed random permutation.

(iii) **Dimensionality:** $\boldsymbol{\theta} \in \mathbb{R}^5, \mathbf{y} \in \mathbb{R}^{100}$.

B.7 Bernoulli GLM

This task involves performing inference in a Generalized Linear Model (GLM) with Bernoulli-distributed observations and a 10-dimensional parameter vector $\boldsymbol{\theta}$. In this model, each binary observation is linked to the parameters through a logistic function, modeling the probability of success in a Bernoulli trial. Gaussian priors are imposed on the coefficients β and the function f, promoting smoothness in the parameter estimates. This setup evaluates the inference method's ability to handle discrete data and accurately capture the posterior distribution in models with latent smoothness constraints.

(i) **Prior:** $\beta \sim \mathcal{N}(\mathbf{0}, 2)$, $f \sim \mathcal{N}(\mathbf{0}, (\mathbf{F}^\top \mathbf{F})^{-1})$.

(ii) **Simulator:** $\mathbf{y}|\boldsymbol{\theta} = (y_1, \ldots, y_{100})$, where $y_i \sim \text{Bern}(\eta(\mathbf{V}_i^\top f + \beta))$, with $\eta(z) = \frac{\exp(z)}{1+\exp(z)}$.

(iii) **Dimensionality:** $\boldsymbol{\theta} \in \mathbb{R}^{10}, \mathbf{y} \in \mathbb{R}^{10}$.

(iv) **Fixed Parameters:** $T = 100$.

B.8 Bernoulli GLM Raw

This task is similar to the Bernoulli GLM described above but utilizes raw binary observations instead of sufficient statistics. Without the simplifying benefit of sufficient statistics, the inference method must directly model the relationship between the parameters and the raw data, increasing the complexity of the inference task. This scenario tests the method's capability to accurately infer parameters in the presence of high-dimensional, discrete, and unprocessed data, challenging its robustness and flexibility.

(i) **Prior:** Same as Bernoulli GLM.

(ii) **Simulator:** $\mathbf{y}|\boldsymbol{\theta} = (y_1, \ldots, y_{100})$, where $y_i \sim \text{Bern}(\eta(\mathbf{V}_i^\top f + \beta))$.

(iii) **Dimensionality:** $\boldsymbol{\theta} \in \mathbb{R}^{10}, \mathbf{y} \in \mathbb{R}^{100}$.

(iv) **Fixed Parameters:** $T = 100$.

B.9 SIR Model

The SIR model is an epidemiological framework that describes the number of individuals in three compartments: Susceptible (S), Infectious (I), and Recovered (R). The dynamics of the system are governed by the following differential equations:

$$\frac{d}{dt}\begin{pmatrix} S \\ I \\ R \end{pmatrix} = \begin{pmatrix} -\beta S \\ \beta S - \gamma I \\ \gamma I \end{pmatrix} \frac{I}{N}, \tag{32}$$

where N represents the total population size. The parameter vector $\boldsymbol{\theta}$ is defined as:

$$\boldsymbol{\theta} = \begin{pmatrix} \beta \\ \gamma \end{pmatrix} \sim \text{LogNormal}\left(\begin{pmatrix} \log(0.4) \\ \log(1/8) \end{pmatrix}, \begin{pmatrix} 0.5 \\ 0.2 \end{pmatrix} \right).$$

(i) **Simulator:** $\mathbf{y}|\boldsymbol{\theta} = (y_1, \ldots, y_{10})$, where $y_i \sim \mathcal{B}(1000, \frac{I}{N})$.

(ii) **Dimensionality:** $\boldsymbol{\theta} \in \mathbb{R}^2, \mathbf{y} \in \mathbb{R}^{10}$.

(iii) **Fixed Parameters:** Population size $N = 1,000,000$, task duration $T = 160$.

B.10 Lotka-Volterra

The Lotka-Volterra (LV) model is a classical ecological framework that describes the interaction between prey (X) and predator (Y) populations. The dynamics are expressed as:

$$\frac{d}{dt}\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} \alpha X - \beta XY \\ -\gamma Y + \delta XY \end{pmatrix}, \tag{33}$$

with the parameter vector defined as:

$$\boldsymbol{\theta} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} \sim \text{LogNormal}\left(\begin{pmatrix} -0.125 \\ -3 \\ -0.125 \\ -3 \end{pmatrix}, \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix} \right).$$

(i) **Simulator:** $\mathbf{y}|\boldsymbol{\theta} = (y_1, \ldots, y_{10})$, where $y_i \sim \text{LogNormal}(\log(X), 0.1)$ and $y_{2i} \sim \text{LogNormal}(\log(Y), 0.1)$.

(ii) **Dimensionality:** $\boldsymbol{\theta} \in \mathbb{R}^4, \mathbf{y} \in \mathbb{R}^{20}$.

(iii) **Fixed Parameters:** Task duration $T = 20$, initial conditions $(X(0), Y(0)) = (30, 1)$.

B.11 Hodgkin-Huxley Model

The Hodgkin-Huxley (HH) model describes the generation and propagation of action potentials in neurons by modeling the membrane voltage $V(t)$ as influenced by ionic currents through voltage-gated sodium (Na^+), potassium (K^+), and leak channels. We adopt the formulation from (Gloeckler et al., 2024). The model is given by the following set of stochastic differential equations:

$$\frac{dV}{dt} = \frac{I_{\text{inj}}(t) - g_{\text{Na}} m^3 h (V - E_{\text{Na}}) - g_{\text{K}} n^4 (V - E_{\text{K}}) - g_{\text{L}} (V - E_{\text{L}})}{C_m} + 0.05\, dw_t, \tag{34}$$

where dw_t represents a Gaussian white noise term that captures stochastic fluctuations in the membrane potential.

The gating variables $m(t), h(t), n(t)$ follow first-order kinetics:

$$\frac{dm}{dt} = \alpha_m(V)(1 - m) - \beta_m(V)m, \tag{35}$$

$$\frac{dh}{dt} = \alpha_h(V)(1 - h) - \beta_h(V)h, \tag{36}$$

$$\frac{dn}{dt} = \alpha_n(V)(1 - n) - \beta_n(V)n, \tag{37}$$

The voltage-dependent rate functions $\alpha_x(V)$ and $\beta_x(V)(x \in \{m, h, n\})$ defined as follows:

$$\alpha_m(V) = 0.32 \times \frac{\text{efun}(-0.25(V - V_0 - 13.0))}{0.25}, \quad \beta_m(V) = 0.28 \times \frac{\text{efun}(0.2(V - V_0 - 40.0))}{0.2},$$

$$\alpha_h(V) = 0.128 \times \exp\left(\frac{-(V - V_0 - 17.0)}{18.0}\right), \quad \beta_h(V) = \frac{4.0}{1 + \exp\left(\frac{-(V - V_0 - 40.0)}{5.0}\right)},$$

$$\alpha_n(V) = 0.032 \times \frac{\text{efun}(-0.2(V - V_0 - 15.0))}{0.2}, \quad \beta_n(V) = 0.5 \times \exp\left(\frac{-(V - V_0 - 10.0)}{40.0}\right).$$

Here, efun(x) is defined to ensure numerical stability when evaluating small arguments:

$$\text{efun}(x) = \begin{cases} 1 - \frac{x}{2} & \text{if } x < \text{1e-4}, \\ \frac{x}{\exp(x) - 1.0} & \text{otherwise.} \end{cases}$$

The metabolic cost $H(t)$, representing the energy consumption due to sodium channel activity, is modeled as:

$$\frac{dH}{dt} = g_{\text{Na}} m^3 h (V - E_{\text{Na}}). \tag{38}$$

(i) **Parameters:** The parameter vector is $\boldsymbol{\theta} = (C_m, g_{\text{Na}}, g_{\text{K}}, g_{\text{L}}, E_{\text{Na}}, E_{\text{K}}, E_{\text{L}})$, where C_m is the membrane capacitance, $g_{\text{Na}}, g_{\text{K}}, g_{\text{L}}$ are the conductances for sodium, potassium, and leak currents, and $E_{\text{Na}}, E_{\text{K}}, E_{\text{L}}$ are the corresponding reversal potentials.

(ii) **Prior:** The parameters $\boldsymbol{\theta}$ are sampled from a uniform distribution:

$$\boldsymbol{\theta} \sim \mathcal{U}\left([1.0, 60, 10, 0.1, 40, -100, -90], [2.0, 120, 30, 0.5, 70, -60, -60]\right).$$

(iii) **Simulator:** The HH model is simulated for 200 ms. An input current of $I_{\text{inj}}(t) = 4\,\text{mA}$ is applied between 50 ms and 150 ms. The voltage equation is integrated with added Gaussian white noise $(0.05\,dw_t)$. To reduce dimensionality, the raw voltage trace is summarized into a set of statistics:

- Spike count,
- Mean and standard deviation of the resting potential,
- Mean voltage during spiking,
- Higher-order moments (normalized skewness and kurtosis) of the voltage response,
- Total energy consumption due to sodium current.

(iv) **Dimensionality:** The parameter space is $\boldsymbol{\theta} \in \mathbb{R}^7$, and the output summary statistics $\mathbf{y} \in \mathbb{R}^8$.

(v) **Fixed Parameters:** The initial membrane voltage is $V_0 = -65.0\,\text{mV}$, with gating variables initialized to their steady-state values at V_0. Simulations use a time resolution of 5000 steps over 200 ms simulation window. For inference, we fix the true parameter vector $\boldsymbol{\theta}^* = (1.437, 72.177, 24.209, 0.154, 66.87, -81.435, -67.606)$ and generate the corresponding synthetic observations, against which the inferred posterior is evaluated.

B.12 Genetic Oscillator

The genetic oscillator model describes a biochemical reaction network involving 9 species governed by 18 reactions, parameterized by 15 reaction rate constants. This network is based on a positive–negative feedback loop, where the activator protein A upregulates transcription of both its gene and the repressor protein R. The repressor R sequesters A by forming a complex C, thereby inhibiting further activation (Vilar et al., 2002; Akesson et al., 2022). The species in the network are:

- D_A, D_A^*: Activator gene in unbound (D_A) and bound states (D_A^*).
- D_R, D_R^*: Repressor gene in unbound (D_R) and bound states (D_R^*).
- M_A, M_R: mRNA transcribed from activator (M_A) and repressor genes (M_R).

- A: Activator protein.
- R: Repressor protein.
- C: Complex formed by A and R.

The network includes the following reactions:

$$
\begin{aligned}
D_A, A &\xrightarrow{\gamma_A} D_A^*, & D_A^* &\xrightarrow{\theta_A} D_A,\\
D_A^* &\xrightarrow{\theta_A} D_A,\ A, & D_A^* &\xrightarrow{\alpha_A^*} D_A^*,\ M_A,\\
D_A &\xrightarrow{\alpha_A} D_A,\ M_A, & &\\
D_R, A &\xrightarrow{\gamma_R} D_R^*, & D_R^* &\xrightarrow{\theta_R} D_R,\\
D_R^* &\xrightarrow{\theta_A} D_R^*,\ A, & D_R^* &\xrightarrow{\alpha_R^*} D_R^*,\ M_R,\\
D_R &\xrightarrow{\alpha_R} D_R,\ M_R, & &\\
M_A &\xrightarrow{\beta_A} M_A + A, & M_R &\xrightarrow{\beta_R} M_R + R,\\
M_A &\xrightarrow{\delta_{M_A}} \phi, & M_R &\xrightarrow{\delta_{M_R}} \phi,\\
A, R &\xrightarrow{\gamma_C} C, & C &\xrightarrow{\delta_A} R,\\
A &\xrightarrow{\delta_A} \phi, & R &\xrightarrow{\delta_R} \phi.
\end{aligned}
$$

The reaction rates include:

$$
\begin{aligned}
\theta_A, \theta_R &: \text{Gene unbinding rates}, \quad \gamma_A, \gamma_R, \gamma_C : \text{Binding rates},\\
\alpha_a, \alpha_a', \alpha_r, \alpha_r' &: \text{Transcription rates}, \quad \beta_a, \beta_r : \text{Translation rates},\\
\delta_{ma}, \delta_{mr} &: \text{mRNA degradation rates}, \quad \delta_a, \delta_r : \text{Protein degradation rates}.
\end{aligned}
$$

Prior: The parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{15}$ is defined with prior bounds:

$$\boldsymbol{\theta} \sim \mathcal{U}([0, 100, 0, 20, 10, 1, 1, 0, 0, 0, 0.5, 0, 0, 0, 0], [80, 600, 4, 60, 60, 7, 12, 2, 3, 0.7, 2.5, 4, 3, 70, 300]).$$

Simulator: The model is simulated using Gillespie's stochastic simulation algorithm (SSA) over 200 time steps. The key observables are the copy numbers of the complex C, activator A, and repressor R.

Dimensionality: The parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{15}$, and the output time series data has the shape $N \times S \times T$, where N is the number of samples, $S = 3$ is the number of observed species, and $T = 200$ is the number of time steps. To reduce the dimensionality and make the data more manageable, we train a simple autoencoder. The autoencoder compresses the data from $N \times 3 \times 200$ to $N \times 15$, effectively replacing the time series with a set of summary statistics.

Fixed Parameters: The model is initialized with the following parameter values and species concentrations:

- **True parameters:**

$$
\begin{aligned}
&\alpha_a = 50, \quad \alpha_a' = 500, \quad \alpha_r = 0.01, \quad \alpha_r' = 50,\\
&\beta_a = 50, \quad \beta_r = 5, \quad \delta_{ma} = 10, \quad \delta_{mr} = 0.5,\\
&\delta_a = 1, \quad \delta_r = 0.2, \quad \gamma_a = 1, \quad \gamma_r = 1,\\
&\gamma_c = 2, \quad \theta_a = 50, \quad \theta_r = 100.
\end{aligned}
$$

- **Initial Species Values:**

$$D_A = 1,\ D_A^* = 0,\ M_A = 0,\ D_R = 1,\ D_R^* = 0,\ M_R = 0,\ C = 10,\ A = 10,\ R = 10.$$

- **Simulation Time:** The system is simulated over 200 time steps uniformly distributed across 200 ms.

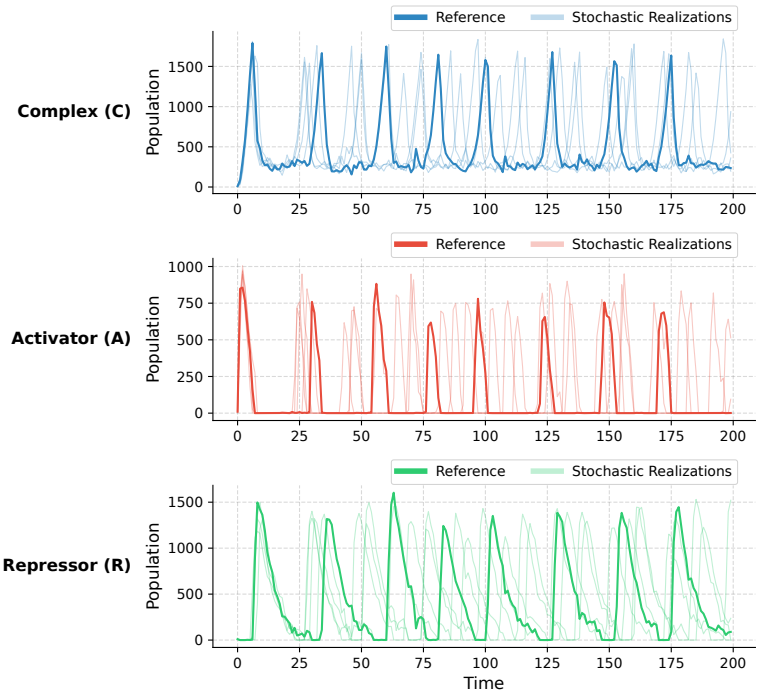


Figure 6: Stochastic trajectories of three key species (Complex (C), Activator (A), and Repressor (R)) in the *Genetic Oscillator* system. The dark line shows the reference trajectory at the true parameter values, while the lighter lines show four independent stochastic realizations under identical conditions, illustrating the system's intrinsic variability while preserving its consistent oscillatory behavior.

C HYPERPARAMETERS

In this section, we describe the hyperparameters and simulation environments used across all methods. Training data for the SBI benchmark problems were generated using *sbibm* (Lueckmann et al., 2021), except for the Lotka–Volterra (LV) and SIR models, which were simulated with *BayesFlow* (Radev et al., 2023b). For SimFormer, we adopt the hyperparameters reported in (Gloeckler et al., 2024) for both the general SBI setting and the Hodgkin–Huxley (HH) model. For the Genetic Oscillator model, we follow the simulation and preprocessing setup of (Akesson et al., 2022). For GATSBI, we start from the task-specific hyperparameters in (Ramesh et al., 2022) and adjust them as needed based on problem structure and computational constraints. All experiments were run on a system with dual 16-core Intel Xeon Gold 6226R CPUs, 576 GB RAM, and an NVIDIA Tesla T4 GPU (16 GB).

For NPE, we adopt the setup of (Lueckmann et al., 2021), who evaluate NPE across ten *sbibm* benchmark tasks. Appendix H of their work shows that Neural Spline Flows (NSF) consistently outperform Masked Autoregressive Flows (MAF), which were originally proposed as the density estimator for NPE. We therefore use the NSF implementation from the *sbi* package (Tejero-Cantero et al., 2020) in all experiments. We also use *sbi*'s implementation of Automatic Posterior Transformation (APT, or SNPE-C). As a sequential method, APT allocates the total simulation budget uniformly across rounds. For FMPE, we follow the experimental setup recommended by the authors in (Wildberger et al., 2023).

For the Genetic Oscillator model, Simformer is trained on a simulation budget of 30,000 using an 8-layer Transformer with 8 attention heads, 64-dimensional head projections, a 512-unit MLP, and dropout rate 0.1, optimized with Adam (learning rate 10^{-3}) for 100,000 iterations with batch size 128. Posterior samples are generated via Euler–Maruyama integration over 1,000 diffusion steps.

For *ConDiSim*, we tuned hyperparameters individually for each task in the *sbibm* benchmark suite (Lueckmann et al., 2021) while keeping the training pipeline fixed. We optimized with AdamW (Loshchilov and Hutter, 2019) using a YOLOX-inspired learning rate schedule: a quadratic warmup over the first 10% of steps from 0.2η to

the base learning rate η, followed by cosine annealing down to 10^{-6}, and a constant phase at 10^{-6} for the final 5% of steps; weight decay was set to $\lambda = 10^{-4}$. We use scaled linear and scaled quadratic noise schedule (Ho et al., 2020) with $\beta_{\text{start}} = 10^{-4}s$ and $\beta_{\text{end}} = 0.02s$ with $s = 1000/T$ to maintain consistent noise levels across varying diffusion steps T. Inputs were standardized to zero mean and unit variance, and a 70/30 train–validation split was employed to handle the high variance of posterior estimation. Training stability was ensured via gradient clipping (max norm 5.0) and SNR-based loss weighting (Hang et al., 2023) with $\gamma = 5.0$, where timestep weights $w_t = \min(\text{SNR}_t, \gamma)/\text{SNR}_t$ and $\text{SNR}_t = \bar{\alpha}_t/(1 - \bar{\alpha}_t)$ emphasize learning at noisier timesteps. Early stopping (patience 30 after epoch 50) and SiLU activations throughout the network were also used. Task-specific hyperparameters are provided in Table 3.

Table 3: ConDiSim hyperparameters.

Benchmark	# Diffusion Blocks	Hidden Dim	Scheduler	# Timesteps	Batch Size	Learning Rate
SLCP	6	128	linear	200	32	1e-3
SLCP with Distractors	6	128	quadratic	1000	50	1e-3
Two Moons	4	64	cosine	160	32	1e-3
Gaussian Mixture	4	64	cosine	160	32	1e-3
Gaussian Linear	6	64	linear	100	50	2e-4
Gaussian Linear Uniform	6	64	cosine	200	50	2e-4
SIR	4	64	linear	100	32	1e-4
Bernoulli GLM	6	128	linear	200	32	1e-3
Bernoulli GLM Raw	6	128	linear	200	32	1e-3
Lotka-Volterra	6	128	cosine	200	50	2e-4
HH Model	6	128	quadratic	1000	50	1e-3
Genetic Oscillator	6	128	cosine	300	64	1e-4

D ADDITIONAL RESULTS

D.1 Posterior Evolution for Two Moons

Fig.7. illustrates the evolution of the approximated posterior for the Two Moons benchmark across selected reverse diffusion timesteps. Starting from pure Gaussian noise at $t = T$, the reverse diffusion process progressively refines the sample distribution, gradually recovering the characteristic bi-modal crescent structure of the true posterior as $t \to 0$. This visualization demonstrates that ConDiSim's learned denoiser successfully guides the denoising trajectory towards the target posterior geometry, confirming that the FiLM-based conditioning effectively communicates the observational information throughout the entire reverse diffusion process.

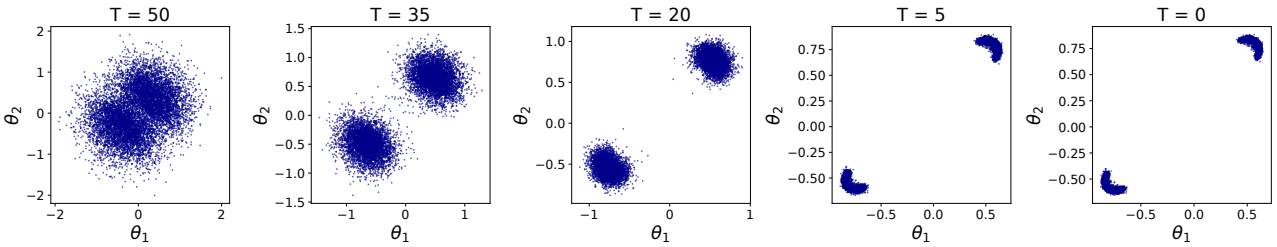
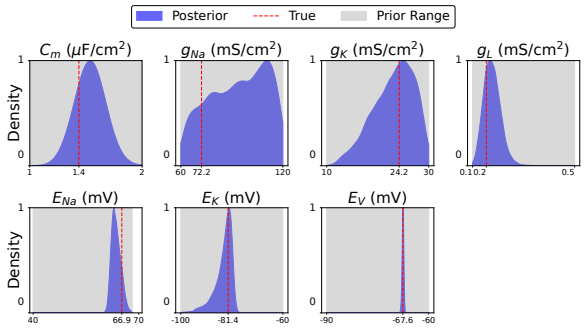


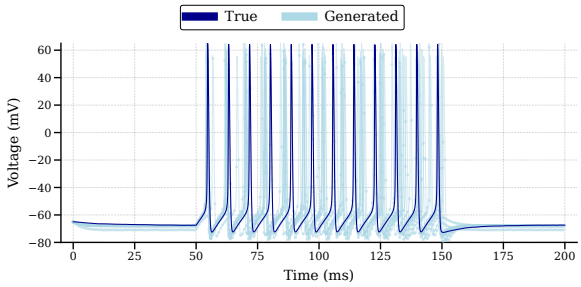
Figure 7: Evolution of the posterior distribution for the Two Moons task, showing the transition from a broad, noisy distribution to a well-defined bi-modal structure as $T \to 0$.

D.2 Comparison with Simformer on Hodgkin–Huxley and Genetic Oscillator Models

Unlike standard SBI benchmarks with analytical posteriors, the Hodgkin–Huxley and Genetic Oscillator models lack closed-form solutions, preventing direct evaluation via C2ST or MMD; we therefore compare ConDiSim only against Simformer as the closest competing diffusion-based method on these real-world problems. For the Hodgkin–Huxley model, posterior samples and predictive checks (Figs. 4 and 8) show that both methods recover plausible voltage traces, with ConDiSim yielding tighter posterior support across several parameters, while for the Genetic Oscillator (Figs. 5a and 9), ConDiSim achieves sharper concentration around the true parameter



(a) Hodgkin–Huxley: Approximated Posterior.



(b) Posterior Predictive Voltage Traces.

Figure 8: Simformer: Posterior Inference for Hodgkin–Huxley model.

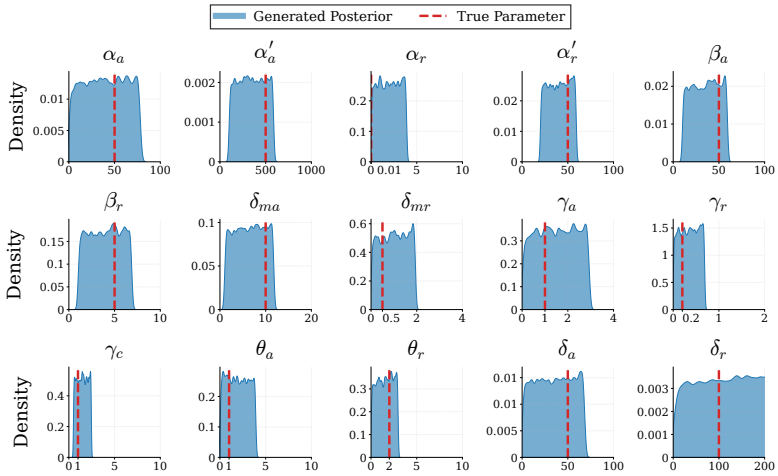


Figure 9: Simformer: Posterior Distributions for the Genetic Oscillator problem.

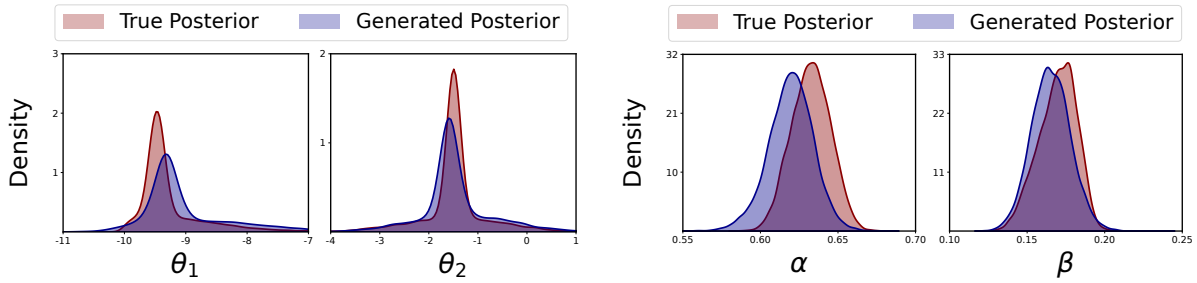
values for several key rates (e.g., $\alpha_r, \beta_r, \gamma_c, \theta_a$) compared to the broader supports of Simformer, demonstrating more precise uncertainty quantification without sacrificing coverage.

E Posterior Plots for SBI benchmarking problems

In this section, we present posterior plots for the remaining SBI benchmarks as a complementary evaluation measure, offering visual insight into the quality of the inferred distributions that is often overlooked by scalar metrics alone.

In the *Gaussian Mixture* (GMM) problem, the main challenge is identifying the shared mean between two Gaussian distributions with different variances. Figure 10a demonstrates that our model effectively captures the primary modes and shapes of these distributions. For *Gaussian Linear* and *Gaussian Linear Uniform*, the challenge is accurately modeling the posterior under both conjugate and non-conjugate settings. The Gaussian Linear Uniform variant complicates posterior inference with a uniform prior. Figure 12a and Figure 12b show that our model effectively handles these differences.

In *Bernoulli GLM* and *Bernoulli GLM Raw*, the challenge involves estimating posteriors from Bernoulli observations with Gaussian priors. The "Raw" variant introduces complexity by using raw data instead of sufficient statistics. Figures 13a and 13b indicate accurate posterior modeling for both cases. The *SIR* model aims to infer contact and recovery rates from noisy infection data over time. Figure 10b illustrates that our model robustly estimates the parameters despite the high variability in the data. Lastly, the *Lotka-Volterra* model is challenging due to its nonlinear dynamics and potential chaotic behavior. Figure 11 shows that our model successfully infers the underlying parameters, effectively navigating the model's complexity.



(a) GMM. (b) SIR Model.

Figure 10: **Posterior Plots:** Comparison of generated vs. reference posterior distributions for the GMM and SIR models.

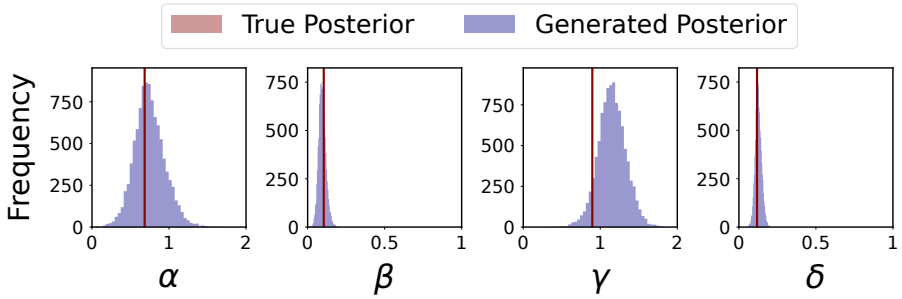
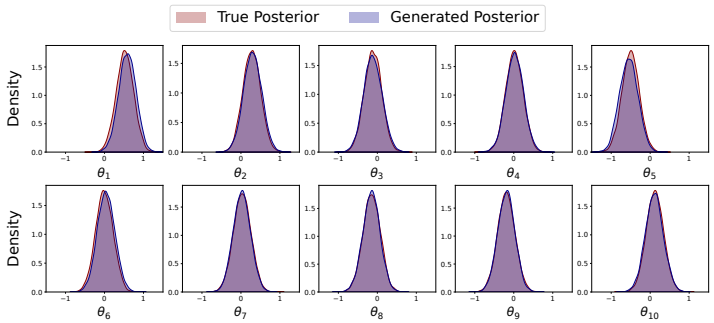
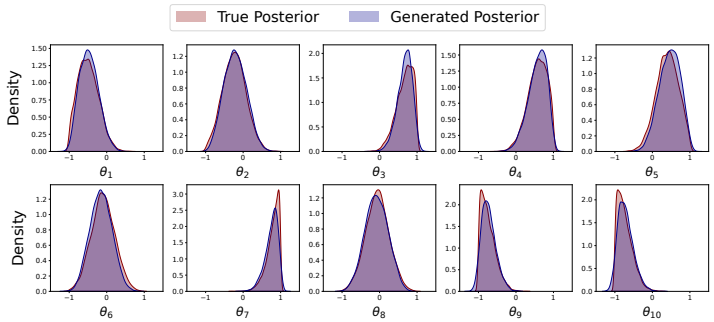


Figure 11: **Lotka Volterra Posterior:** Visualization of the posterior for the Lotka–Volterra model. The true posterior mean is indicated by a vertical line (due to low variance).

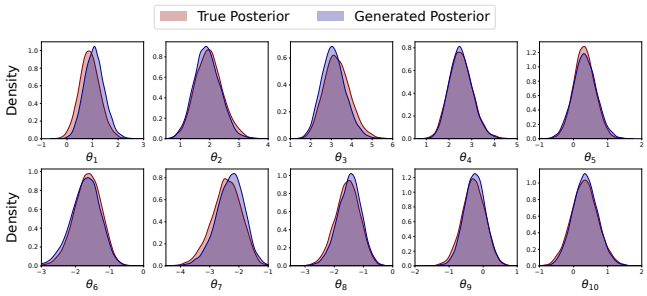


(a) Gaussian Linear Model.

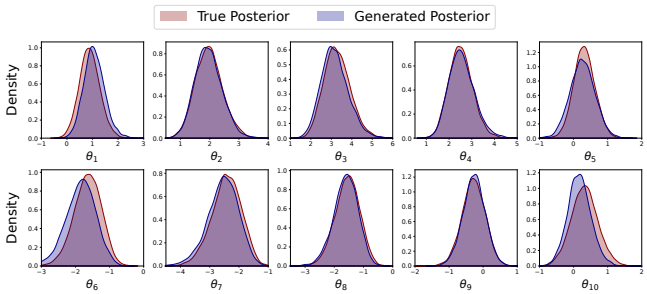


(b) Gaussian Linear Uniform Model.

Figure 12: Posterior Distributions for Gaussian Linear and Gaussian Linear Uniform Models: Generated vs. Reference Posterior.



(a) Bernoulli GLM.



(b) Bernoulli GLM Raw.

Figure 13: Posterior Distributions for Bernoulli GLM and Bernoulli GLM Raw: Generated vs. Reference Posterior.

F SBI METRICS

F.1 Limitations of the C2ST Metric

The Classifier Two-Sample Test (C2ST) evaluates the similarity between a generated posterior and the true posterior by training a classifier to differentiate their samples. A score near 0.5 indicates that the two distributions are similar, while higher values indicate discrepancies. Although widely used in SBI literature for its straightforward interpretability providing a clear score between 0.5 and 1, the metric can be overly sensitive, as a single poorly estimated dimension may distort the results. Furthermore, its reliability depends significantly on the design and training of the classifier (Friedman, 2003; Lopez-Paz and Oquab, 2017). However, C2ST remains the most widely used metric in SBI, as it is both interpretable and well established for evaluating benchmark problems. For completeness, we also report MMD results in the following section, although MMD is kernel dependent, less commonly used, and often difficult to interpret solely from numerical values.

F.2 Maximum Mean Discrepancy (MMD)

The Maximum Mean Discrepancy (MMD) measures the distance between the mean embeddings of the generated posterior P and the approximated posterior Q in a reproducing kernel Hilbert space (RKHS) (Lueckmann et al., 2021). MMD is defined as,

$$\mathrm{MMD}^2(P, Q; \mathcal{H}) = \|\mathbb{E}_{x\sim P}[\phi(x)] - \mathbb{E}_{y\sim Q}[\phi(y)]\|^2_{\mathcal{H}}\,, \tag{39}$$

where ϕ is a feature mapping into the RKHS $\mathcal{H}$. A low MMD value indicates that the generated posterior closely approximates the true posterior. MMD therefore provides a holistic measure of similarity between distributions without being overly influenced by discrepancies in a single dimension.

Table 4: MMD metric for different methods across simulation budgets (mean ± std. dev.).

(a) Two Moons, Bernoulli GLM, and SIR.

Method	Two Moons			Bernoulli GLM			SIR		
Budget	10,000	20,000	30,000	10,000	20,000	30,000	10,000	20,000	30,000
NPE	0.0254 ±0.0025	0.0241 ±0.0099	0.0125 ±0.0045	1.0371 ±0.1830	0.9379 ±0.1001	0.8675 ±0.1945	0.6611 ±0.2236	1.0312 ±0.1255	1.1903 ±0.1007
APT	0.0073 ±0.0030	0.0069 ±0.0030	0.0037 ±0.0012	0.7383 ±0.0853	0.4944 ±0.0663	0.5808 ±0.0539	1.1003 ±0.0712	1.1093 ±0.0311	1.1291 ±0.0481
GATSBI	0.1213 ±0.0650	0.0582 ±0.0122	0.1432 ±0.1161	3.6381 ±0.5428	4.8299 ±0.6825	3.9965 ±1.8510	2.9470 ±0.4891	2.7502 ±0.7268	2.1466 ±0.5625
Simformer	**0.0005** ±0.0003	**0.0001** ±0.0001	**0.0003** ±0.0004	**0.0191** ±0.0098	**0.0156** ±0.0087	**0.0116** ±0.0061	**0.0018** ±0.0028	**0.0016** ±0.0042	**0.0016** ±0.0085
FMPE	0.0133 ± 0.0197	0.0039 ± 0.0060	0.019 ± 0.0014	0.1076 ± 0.0283	0.0826 ± 0.0138	0.0746 ± 0.0247	0.2058 ± 0.0024	0.3618 ± 0.0048	0.1819 ± 0.0044
ConDiSim	0.0172 ±0.0226	0.0091 ±0.0099	0.0052 ±0.0018	0.0393 ±0.0095	0.0525 ±0.0547	0.0190 ±0.0134	0.2413 ±0.0681	0.3853 ±0.1195	0.4860 ±0.1198

(b) SLCP Distractors, Bernoulli GLM Raw, and Lotka Volterra.

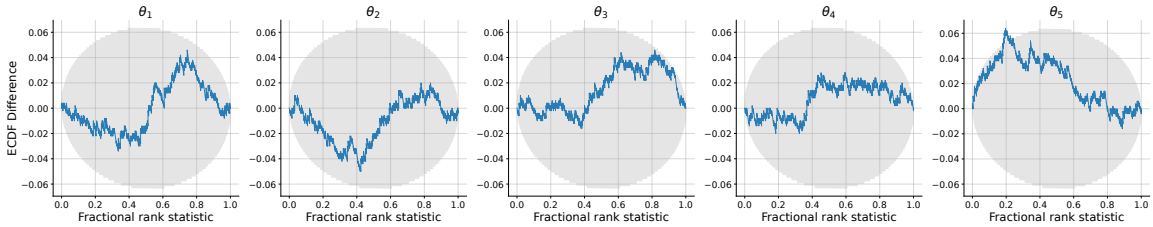
Method	SLCP Distractors			Bernoulli GLM Raw			Lotka Volterra		
Budget	10,000	20,000	30,000	10,000	20,000	30,000	10,000	20,000	30,000
NPE	1.2167 ±0.2397	0.7834 ±0.3453	0.6428 ±0.1779	1.1227 ±0.0987	1.1389 ±0.1878	1.1448 ±0.1287	2.6357 ±0.3468	3.8635 ±0.2770	4.3310 ±0.0709
APT	1.8380 ±0.4448	0.8278 ±0.2373	0.4422 ±0.1428	0.7233 ±0.2131	0.6968 ±0.0826	0.5437 ±0.1333	3.1921 ±0.8198	3.2722 ±0.8169	4.4739 ±0.9642
GATSBI	5.1993 ±1.2320	7.2478 ±1.2260	3.0631 ±0.3244	3.4369 ±0.8517	4.3479 ±0.5433	3.5950 ±0.8613	7.2684 ±0.5733	5.5183 ±1.7126	4.8858 ±1.2146
Simformer	**0.0540** ±0.0173	**0.0541** ±0.0161	0.0552 ±0.0195	0.2056 ±0.0486	0.0926 ±0.0493	0.0731 ±0.0238	**0.9696** ±0.4434	**0.8342** ±0.4742	**0.9634** ±0.2588
FMPE	0.1598 ± 0.0501	0.1403 ± 0.0471	0.0930 ± 0.0561	**0.0859** ± 0.0167	0.0831 ± 0.0181	0.0781 ± 0.0102	1.0537 ± 0.6277	1.0858 ± 0.5639	1.1696 ± 0.4336
ConDiSim	0.1378 ±0.0404	0.0580 ±0.0205	**0.0275** ±0.0182	0.0968 ±0.1012	**0.0359** ±0.0212	**0.0305** ±0.0279	2.3226 ±0.1486	2.1965 ±0.3856	2.1180 ±0.2584

(c) Gaussian Mixture, Gaussian Linear, Gaussian Linear Uniform, and SLCP.

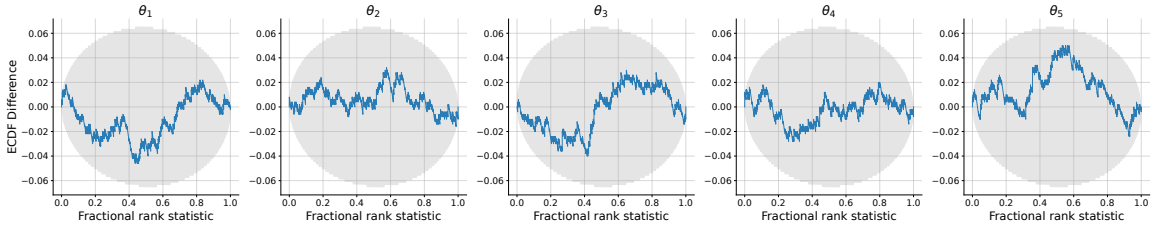
Method	Gaussian Mixture			Gaussian Linear			Gaussian Linear Uniform			SLCP		
Budget	10,000	20,000	30,000	10,000	20,000	30,000	10,000	20,000	30,000	10,000	20,000	30,000
NPE	0.0604 ±0.0260	0.0418 ±0.0211	0.0554 ±0.0347	0.0105 ±0.0036	0.0074 ±0.0018	0.0082 ±0.0011	0.0258 ±0.0098	0.0117 ±0.0041	0.0121 ±0.0051	0.4021 ±0.0551	0.1632 ±0.0684	0.2333 ±0.0935
APT	0.0137 ±0.0031	0.0108 ±0.0088	0.0110 ±0.0034	**0.0075** ±0.0013	0.0070 ±0.0018	0.0080 ±0.0008	0.0166 ±0.0087	0.0138 ±0.0053	0.0113 ±0.0039	0.1405 ±0.0347	0.0487 ±0.0296	0.0287 ±0.0166
GATSBI	0.2242 ±0.0296	0.1149 ±0.0549	0.1733 ±0.1082	0.4999 ±0.1595	0.6476 ±0.3050	0.7182 ±0.2839	0.9044 ±0.2353	0.7712 ±0.1483	1.1197 ±0.4795	0.7197 ±0.5598	0.3375 ±0.1473	0.7948 ±0.5992
Simformer	**0.0009** ±0.0007	**0.0009** ±0.0006	**0.0007** ±0.0010	0.0092 ±0.0116	**0.0037** ±0.0035	**0.0037** ±0.0028	**0.0014** ±0.0006	**0.0015** ±0.0011	**0.0008** ±0.0003	**0.0323** ±0.0203	**0.0025** ±0.0034	**0.0021** ±0.0030
FMPE	0.1893 ± 0.0985	0.0208 ± 0.0078	0.0013 ± 0.0011	0.0511 ± 0.0053	0.0500 ± 0.0041	0.0497 ± 0.0034	0.1425 ± 0.0219	0.1261 ± 0.0206	0.1236 ± 0.0247	0.1573 ± 0.0543	0.1489 ± 0.0528	0.1489 ± 0.0528
ConDiSim	0.1451 ±0.0269	0.1047 ±0.0314	0.0686 ±0.0228	0.0591 ±0.0249	0.0165 ±0.0083	0.0116 ±0.0032	0.0462±0.0321	0.0297 ±0.0252	0.0291 ±0.0459	0.0996 ±0.0287	0.0510 ±0.0345	0.0267 ±0.0178

G SIMULATION BASED CALIBRATION

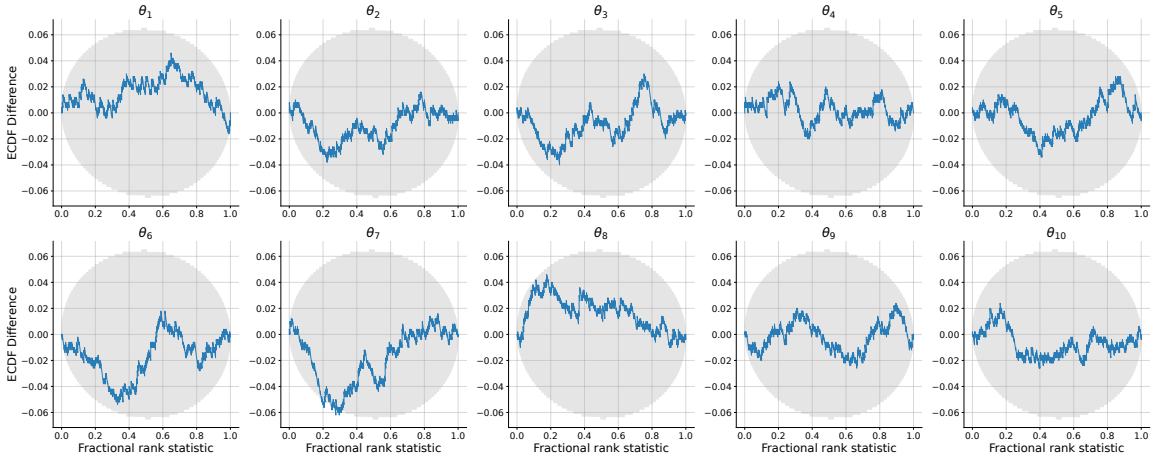
Simulation-Based Calibration (SBC) is an important metric for generative models in SBI, as it verifies alignment between inferred posterior distributions and true parameter values, assisting in detecting potential biases in model-generated samples. For SBC, we generate $M = 500$ prior samples, simulate corresponding observations, and draw $L = 250$ posterior samples for each observation using the learned ConDiSim model. Calibration is assessed through rank statistics of true parameters within their posterior distributions, with empirical cumulative distribution functions (ECDFs) providing a robust, bin-free visualization of these ranks. Following (Säilynoja et al., 2022), ECDF-based SBC transforms posterior draws into fractional rank statistics, avoiding arbitrary binning and facilitating interpretation. A uniform ECDF indicates well-calibrated posteriors, while deviations suggest miscalibration: curves above the diagonal indicate overconfidence, and those below indicate underconfidence. Using *BayesFlow* (Radev et al., 2023b), we generate ECDF plots with 90% confidence interval bands as calibration benchmarks. Solid blue lines represent rank ECDFs of posterior draws, with shaded gray areas denoting the 90% confidence intervals. The SBC results for SBI benchmarking problems are presented below:



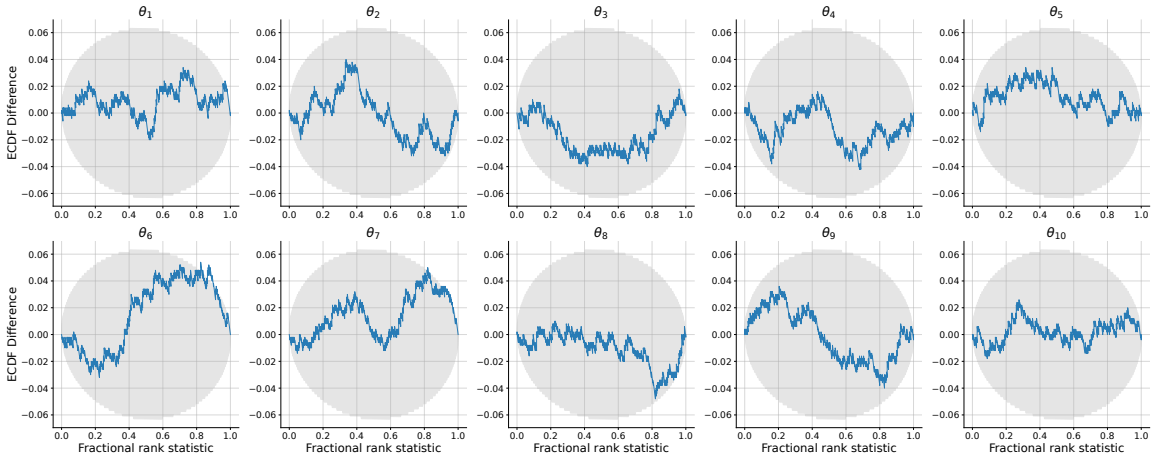
(a) SLCP.



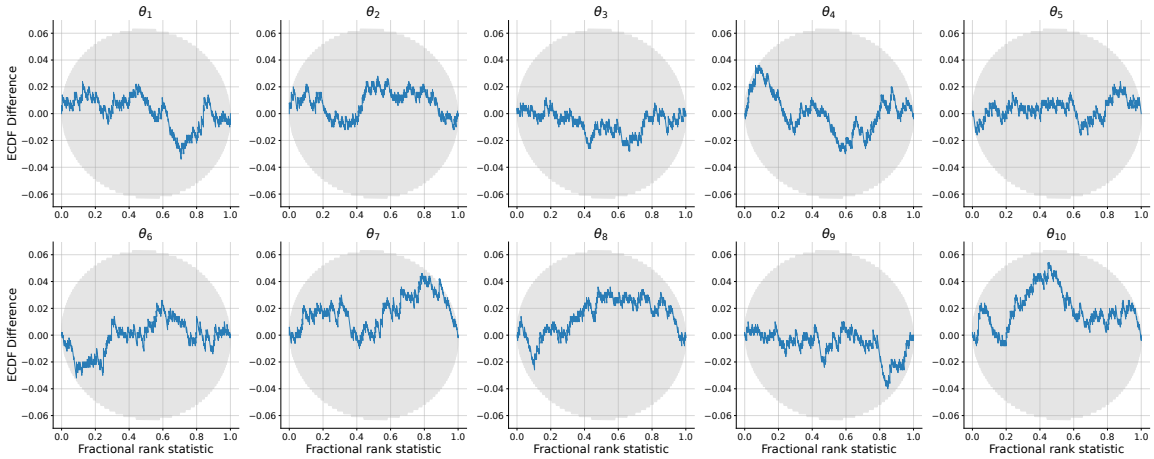
(b) SLCP Distractors.



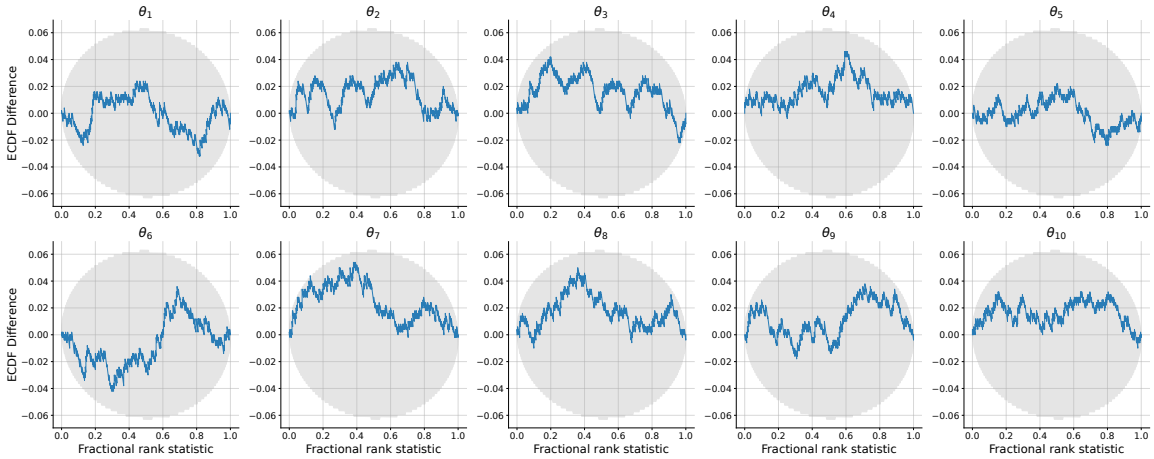
(c) Gaussian Linear.



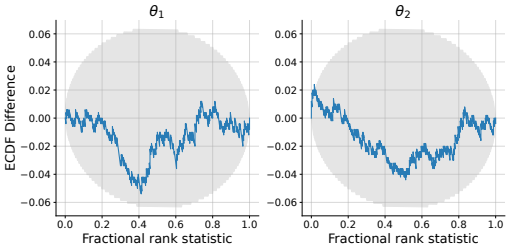
(d) Gaussian Linear Uniform.



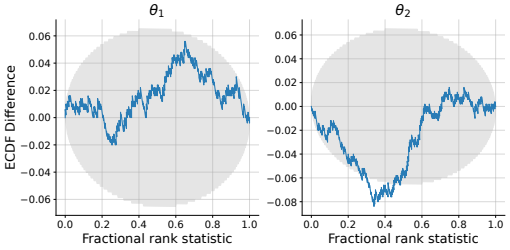
(e) Bernoulli GLM.



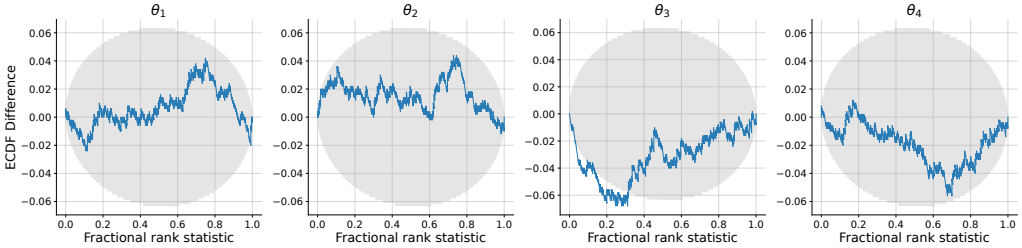
(f) Bernoulli GLM Raw.



(g) SIR.



(h) Gaussian Mixture.



(i) Lotka Volterra.

Figure 14: ECDF results for the 30,000 simulation budget and the first run for each problem.