CAUSAL POSTERIOR ESTIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We present Causal Posterior Estimation (CPE), a novel method for Bayesian inference in simulator models, where evaluating the likelihood function is intractable or computationally expensive, but generating outputs given parameter values is straightforward. CPE approximates the posterior distribution using flow matching while directly incorporating the conditional dependence structure induced by the model's graphical representation into the neural network architecture. Across extensive experiments, we demonstrate that hard-coding these conditional dependencies into the network, rather than requiring them to be learned from data, enables CPE to achieve highly accurate posterior inference that matches or outperforms state-of-the-art baselines.

1 INTRODUCTION

Consider the problem of inferring the Bayesian posterior distribution $\pi(\theta|x) \propto \pi(x|\theta)\pi(\theta)$ when the likelihood function $\pi(x|\theta)$ is intractable to evaluate, but simulating from the joint distribution $\pi(x, \theta)$ is feasible. *Simulation-based inference* (SBI) methods address this problem by training neural networks on synthetic data sets generated from the forward model to approximate the posterior.

Recently, continuous neural density estimators based on flow matching and diffusion models (Song et al., 2021; Lipman et al., 2023; Liu et al., 2023) have become the state of the art in SBI. Both classes of methods transport a simple base distribution q_0, e.g., a standard Gaussian, to an approximation of the posterior by integrating an ordinary differential equation

$$\frac{\mathrm{d}\theta^{(t)}}{\mathrm{d}t} = v_t\big(\theta^{(t)}, x\big), \quad \theta^{(0)} \sim q_0, \tag{1}$$

Flow matching regresses v_t directly onto a target vector field (Lipman et al., 2023; Wildberger et al., 2023), whereas score-based diffusion models learn the score $\nabla_\theta \log \pi_t(\theta|x)$ of the noised posterior and obtain v_t, up to a reversal of time, through the probability flow ODE (Song et al., 2021; Sharrock et al., 2024; Gloeckler et al., 2024),

$$v_t(\theta, x) = f(\theta, t) - \tfrac{1}{2} g(t)^2 \, \nabla_\theta \log \pi_t(\theta|x), \tag{2}$$

where f and g are the drift and diffusion coefficients of the noising process. In comparison to ABC algorithms (Sisson et al., 2007; Beaumont et al., 2009; Sisson et al., 2018; Albert et al., 2015; 2025), neural SBI methods based on flow- and score-matching scale gracefully to high-dimensional spaces achieving high posterior accuracy on a wide range of tasks (Gloeckler et al., 2024; Sharrock et al., 2024; Wildberger et al., 2023).

However, a critical limitation of current continuous SBI methods lies in their architectural design. Most approaches parameterize the vector field with dense architectures, such as multi-layer perceptrons (MLPs), which must learn the dependence structure between parameters and observations entirely from data. They thereby ignore the conditional independencies implied by the Bayesian graphical model, i.e., the prior and posterior programs (van de Meent et al., 2018), which leads to large networks and reduced sample efficiency. Transformer-based approaches can encode this structure through attention masks (Gloeckler et al., 2024), but require a comparatively large number of parameters.

To address this, we introduce *Causal Posterior Estimation* (CPE), a novel SBI method that restricts the hypothesis space of the neural density estimator by encoding the causal relationships of the Bayesian model directly into a continuous normalizing flow. Instead of relying on a network to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

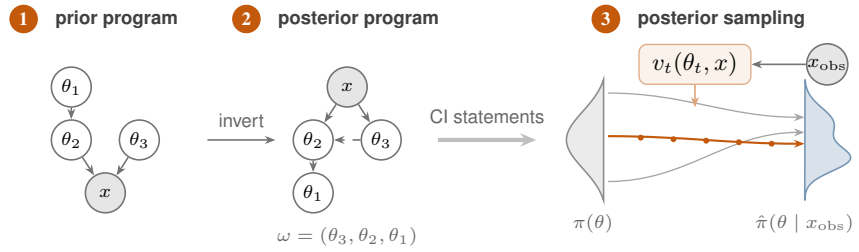


Figure 1: **Causal Posterior Estimation**: **(1)** The prior program, i.e., the generative model whose parameters θ we aim to infer. **(2)** The prior program is inverted into the posterior program. From the posterior program, conditional dependencies are computed and incorporated into a NN architecture. **(3)** Posterior inference is conducted by drawing a sample from the prior $\pi(\theta)$ and transporting it to the posterior $\pi(\theta|x_{\text{obs}})$ by integrating the vector field $v_t(\theta^{(t)}, x)$.

discover the dependence structure from data, CPE parameterizes the flow's vector field with a causal block-autoregressive neural network whose dependencies structurally follow the posterior program (Figure 1). Specifically, our contributions are as follows:

- **Causal block neural network architecture**: We propose a novel continuous normalizing flow whose vector field is parameterized by a structural masking mechanism that explicitly respects the graphical model of the posterior.
- **Prior-guided vector fields**: We parameterize the vector field as a convex combination of the current state and the network output, using the prior directly as the base distribution.
- **Accurate and efficient posterior inference**: Empirically, CPE achieves on average higher accuracy than state-of-the-art baselines across multiple benchmark tasks, while requiring considerably fewer trainable parameters.

2 Preliminaries

In conventional Bayesian inference, a tractable likelihood enables posterior inference via Markov chain Monte Carlo or variational inference (Brooks et al., 2011; Blei et al., 2017). Neural SBI methods avoid likelihood evaluations altogether and instead use a data set of N simulated pairs $\{(x_n, \theta_n)\}_{n=1}^N \sim \pi(x, \theta)$ to fit neural approximations of the likelihood, the posterior, or the likelihood-to-evidence ratio. Below, we introduce the continuous generative models that underpin our approach.

2.1 Continuous Normalizing Flows and Flow Matching

Continuous normalizing flows (CNFs) (Chen et al., 2018) are neural density estimators based on pushforward measures. A neural network with weights ϕ defines a time-dependent vector field $v_t : [0, 1] \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_\theta}$ (we omit the subscript ϕ for brevity), which induces the ODE

$$\frac{\mathrm{d}}{\mathrm{d}t}\Lambda_t(\theta) = v_t\big(\Lambda_t(\theta)\big), \quad \Lambda_0(\theta) = \theta, \tag{3}$$

where the flow Λ_t is a time-dependent diffeomorphism. We write $\theta^{(t)} = \Lambda_t(\theta^{(0)})$ for the trajectory starting at $\theta^{(0)}$. Unlike discrete normalizing flows, CNFs impose no architectural constraints, such as explicit invertibility, on the network. Given a base distribution q_0 and a sample $\theta^{(0)} \sim q_0$, integrating Equation (3) yields a sample $\theta^{(1)}$ from the pushforward distribution, whose density follows from the instantaneous change of variables formula

$$q_1\big(\theta^{(1)}\big) = q_0\big(\theta^{(0)}\big) \exp\left(-\int_0^1 \operatorname{div} v_t\big(\theta^{(t)}\big)\,\mathrm{d}t\right). \tag{4}$$

Training a CNF by maximizing the likelihood in Equation (4) is computationally expensive, since every gradient step requires solving the ODE. *Flow matching* instead regresses v_t onto a known

conditional target vector field u_t, yielding a simple simulation-free least-squares objective (Lipman et al., 2023). The target vector field is commonly obtained by drawing $\theta^{(0)}$ from the base distribution and $\theta^{(1)}$ independently from the data distribution, and moving from $\theta^{(0)}$ to $\theta^{(1)}$ along a straight line at constant velocity $\theta^{(1)} - \theta^{(0)}$ (Liu et al., 2023)

$$\theta^{(t)} = t\,\theta^{(1)} + (1-t)\,\theta^{(0)}, \qquad u_t\big(\theta^{(t)} | \theta^{(0)}, \theta^{(1)}\big) = \theta^{(1)} - \theta^{(0)}, \tag{5}$$

which, for a standard Gaussian base distribution, recovers the optimal transport path of Lipman et al. (2023) with $\sigma_{\text{min}} = 0$. Flow matching is closely related to denoising score matching (Song et al., 2021); the two differ primarily in the choice of interpolation and regression target, and both allow deterministic sampling with ODE solvers.

2.2 Neural Posterior Estimation (NPE)

Modern NPE methods (Wildberger et al., 2023; Sharrock et al., 2024; Gloeckler et al., 2024) approximate the posterior with a conditional vector field $v_t : [0,1] \times \mathbb{R}^{d_\theta} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_\theta}$. For flow matching (Wildberger et al., 2023), v_t is trained by regressing it onto the target u_t in Equation (5) while conditioning on simulated observations $x \sim \pi(x|\theta^{(1)})$:

$$\hat{\phi} = \arg\min_{\phi} \mathbb{E}_{\substack{t\sim\mathcal{U}(0,1),\ \theta^{(0)}\sim q_0 \\ \theta^{(1)}\sim\pi(\theta),\ x\sim\pi(x|\theta^{(1)})}} \left[\left\| v_t\big(\theta^{(t)}, x\big) - \big(\theta^{(1)} - \theta^{(0)}\big) \right\|^2 \right]. \tag{6}$$

The resulting conditional density $q_1(\theta|x)$ serves as an *amortized* posterior approximation, i.e., $q_1(\theta|x) \approx \pi(\theta|x)$. Given an arbitrary observation x_{obs}, a posterior sample is obtained by drawing $\theta^{(0)} \sim q_0$ and integrating the ODE with vector field $v_t(\cdot, x_{\text{obs}})$ from $t = 0$ to $t = 1$.

3 Causal Posterior Estimation

We introduce Causal Posterior Estimation (CPE), an SBI method that approximates the posterior $\pi(\theta|x)$ with a CNF whose vector field v_t embeds the conditional dependence structure of the posterior program. Since this structure is known from the model specification, encoding it restricts the hypothesis space of the estimator rather than leaving the network to learn it from data. At its core, the vector field is a composition of L blocks (Figure 2),

$$v_t\big(\theta^{(t)}, x\big) = \big(g_{t,x}^{(L)} \circ \cdots \circ g_{t,x}^{(1)}\big)\big(\theta^{(t)}\big), \tag{7}$$

where each block $g_{t,x}^{(l)}$ is conditioned on the time t and the observation x. In the remainder of this section, we describe CPE in three parts: the prior-guided blocks $g^{(l)}$ (Section 3.1), the causal block-autoregressive layers inside each block (Section 3.2), and the training and sampling procedure (Section 3.3). Architectural details can be found in Appendix A.

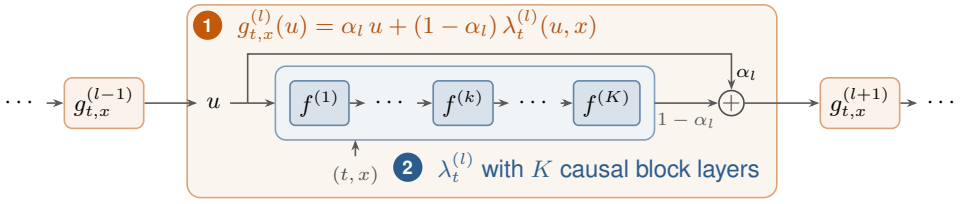


Figure 2: Overview of CPE. The vector field $v_t(\theta^{(t)}, x)$ is a composition of L prior-guided blocks (Section 3.1). Each block forms a trainable convex combination of its input u, with $u = \theta^{(t)}$ for the first block, and a causal block-autoregressive network $\lambda_t^{(l)}$ with K causal block layers (Section 3.2).

3.1 Prior-Guided Vector Fields

Inspired by the fact that, for conjugate exponential family models, the posterior mean is a convex combination of the prior mean and a data-dependent estimate (Diaconis & Ylvisaker, 1979), we use the prior $\pi(\theta)$ directly as the base distribution, $q_0(\theta) = \pi(\theta)$, instead of starting the flow, for instance,

from a standard Gaussian. We then form each block as a convex combination of its input and the output of a neural network:

$$g_{t,x}^{(l)}(u) = \alpha_l \, u + (1 - \alpha_l) \, \lambda_t^{(l)}(u, x), \tag{8}$$

where u is the output of the previous block, with $u = \theta^{(t)}$ for the first block, $\alpha_l \in (0, 1)$ are trainable scalars, and each $\lambda_t^{(l)}$ is a time-dependent neural network with K layers (Section 3.2). The convex combinations act as learnable skip connections between the state of the flow and the network outputs, which, together with the prior as base distribution, provide an inductive bias that can simplify optimization and improve the accuracy of the estimator.

3.2 Causal Block-Autoregressive Layers

Consider a hierarchical model with four variables that forms a v-structure

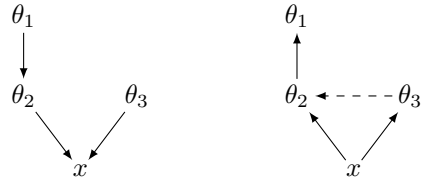
$$\pi(\theta_1, \theta_2, \theta_3, x) = \pi(\theta_1) \, \pi(\theta_2|\theta_1) \, \pi(\theta_3) \, \pi(x|\theta_2, \theta_3), \tag{9}$$

where the objective is to infer the joint posterior distribution $\pi(\theta_1, \theta_2, \theta_3|x) \propto \pi(\theta_1, \theta_2, \theta_3, x)$ (Figure 3a). Dense architectures leave the dependence structure between parameters and data unrestricted and must learn it from data. Here, we aim to instead restrict the vector field to the conditional dependencies of the inverted graphical model, the *posterior program* (Figure 3b).[1] In this example, conditioning on the collider x renders θ_2 and θ_3 dependent, so the posterior program contains the edge $\theta_3 \to \theta_2$ in addition to the reversed edges $x \to \theta_2$, $x \to \theta_3$ and $\theta_2 \to \theta_1$. At the same time, the posterior satisfies $\theta_1 \perp\!\!\!\perp \theta_3 | \theta_2, x$, which we aim to encode by letting the vector field of θ_1 depend on θ_2 but not on θ_3.

Posterior Factorization To enforce the conditional dependence structure of the posterior program, we carefully parameterize the mapping λ_t. Assuming the posterior program forms a directed acyclic graph (DAG),[2] we define a topological ordering over its parameter nodes, denoted by $\omega = \{\omega_1, \ldots, \omega_{d_\theta}\}$. The mapping λ_t is constructed to respect the factorization:

$$\pi(\theta|x) = \prod_{i=1}^{d_\theta} \pi\big(\theta_{\omega_i} | \mathrm{pa}(\theta_{\omega_i}), x\big), \tag{10}$$

where $\mathrm{pa}(\theta_{\omega_i})$ denotes the parents of θ_{ω_i} in the posterior program.



(a) Prior program. (b) Posterior program.

Figure 3: Graphical models illustrating prior and posterior dependencies.

Structural Masking We implement Equation (10) by applying a lower-triangular structural mask $W \in \mathbb{R}^{d_\theta \times d_\theta}$ over features ordered by ω. To avoid modeling spurious correlations, we strictly set $W_{ij} = 0$ wherever $\theta_i \perp\!\!\!\perp \theta_j | \theta_{\setminus\{i,j\}}, x$ holds in the posterior program. For the posterior program in Figure 3b with topological ordering $\omega = \{3, 2, 1\}$, the structural matrix takes the form

$$W = \begin{bmatrix} g(W_{1,1}) & 0 & 0 \\ W_{2,1} & g(W_{2,2}) & 0 \\ 0 & W_{3,2} & g(W_{3,3}) \end{bmatrix}, \tag{11}$$

where $g : \mathbb{R} \to \mathbb{R}$ is an element-wise activation function[3].

[1] Several approaches exist for inverting graphical models (Stuhlmüller et al., 2013; Webb et al., 2018). We adopt the formulation of Ambrogioni et al. (2021) for its computational simplicity. The algorithm of Webb et al. (2018) yields faithful inverses but denser graphs, and did not measurably improve performance in our preliminary experiments.

[2] Generative models with temporal loops, such as dynamic Bayesian networks, can be unrolled into finite acyclic structures for inference.

[3] Using g is not strictly necessary: we introduce it to render the model invertible for the discrete NF case in Appendix A.

Block-Matrix Augmentation To increase expressivity, we expand the masking strategy into a causal block neural autoregressive flow (cf. Huang et al. (2018); De Cao et al. (2020); Ambrogioni et al. (2021)) by replacing scalar entries W_{ij} with block matrices $B_{ij} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ (see Figure 4). The mapping λ_t is then composed as a sequence of transformations $\lambda_t = f^{(K)} \circ \cdots \circ f^{(1)}$, where each layer computes

$$f^{(k)}(\theta) = \text{act}\left(B^{(k)}\theta + b^{(k)}\right), \tag{12}$$

with bias vector $b^{(k)}$ and activation function $\text{act}(\cdot)$. The block sizes are configured with $d_{\text{in}}^{(1)} = d_{\text{out}}^{(K)} = 1$.

Conditioning Mechanism The vector field v_t is conditioned on the time t and the observation x. We embed t with sinusoidal features, project the time embedding and the data x through separate MLPs, and concatenate the resulting embeddings. A linear projection of this vector is added to the hidden units of all nodes after the first layer $f^{(1)}$ (cf. the conditioning in Germain et al. (2015); Papamakarios et al. (2017)).

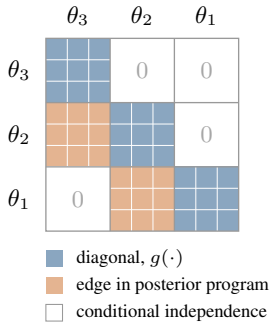


Figure 4: Masked weight matrix $B^{(k)}$ for the posterior program in Figure 3b.

3.3 Training and Sampling

We train CPE with the simulation-free rectified flow objective (Liu et al., 2023), i.e., Equation (6) with the prior as base distribution, $q_0 = \pi(\theta)$, and the vector field in Equation (7). Training therefore requires no ODE integration, but merely a synthetic data set of pairs $(\theta^{(1)}, x)$ drawn from the prior and the simulator.

To obtain posterior samples given an observation x_{obs}, we draw $\theta^{(0)} \sim \pi(\theta)$ from the prior and integrate the ODE with vector field $v_t(\cdot, x_{\text{obs}})$ from $t = 0$ to $t = 1$. We consider two solvers: an adaptive Runge–Kutta 5(4) solver (CPE-RK) and an Euler solver with 20 fixed steps (CPE-Euler).Because the vector field is trained on convex interpolations between prior and posterior samples (Equation (5)) we expect its trajectories to be close to straight, such that a coarse Euler discretization suffices. Also, since the ODE is not constrained to the support of the prior, individual samples can fall outside it and thus have zero prior density. As in rejection sampling, we discard such samples and draw new ones until the desired number of posterior samples is reached.

4 Related Work

Flow and denoising score matching Flow and denoising score matching have been applied with great success to generative modeling (Ho et al., 2020; Song et al., 2021; Karras et al., 2022; Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023), Bayesian and variational inference (Zhang & Chen, 2022; Vargas et al., 2023; Berner et al., 2024; Vargas et al., 2024), and inverse problems (Kawar et al., 2022; Chung et al., 2023; Dou & Song, 2024).

Simulation-based inference Early neural SBI methods approximate the posterior with mixture density networks or discrete normalizing flows (Papamakarios & Murray, 2016; Lueckmann et al., 2017; Greenberg et al., 2019; Deistler et al., 2022; Radev et al., 2023). More recent methods use continuous-time models trained with flow matching or score matching objectives (Geffner et al., 2023; Wildberger et al., 2023; Sharrock et al., 2024; Gloeckler et al., 2024; Schmitt et al., 2024; Tong et al., 2024; Gao et al., 2025), which have since been extended to, e.g., hierarchical models (Charles et al., 2026), mixed discrete-continuous or constrained parameter spaces (Boelts et al., 2026; Carrasco-Pollo et al., 2026), and tall data (Linhart et al., 2024; Touron et al., 2026). We refer to Deistler et al. (2025); Dax et al. (2026); Arruda et al. (2025) for recent overviews.

Structured inference architectures Encoding the conditional dependence structure of a probabilistic model into the inference network is not a novel approach. Webb et al. (2018) introduced

faithful inversion of generative models for amortized inference, which Weilbach et al. (2020; 2022) used to sparsify continuous normalizing flows and diffusion models. Masked weight matrices, which restrict each output of a network to a subset of its inputs, form the core of masked autoregressive flows (Germain et al., 2015; Papamakarios et al., 2017) and have been extended to encode graphical structure (Wehenkel & Louppe, 2021; Silvestri et al., 2021; Chen et al., 2023). Most related to our work, Gloeckler et al. (2024) use attention masks in a transformer-based score network to encode the dependence structure of the simulator. CPE differs from these approaches in three respects: it encodes the posterior program directly in the block-masked weight matrices of a comparatively small network rather than in the attention masks of a transformer, it combines this architecture with the prior as base distribution, and its layers are invertible, such that CPE can also be used as a discrete normalizing flow, whereas the networks of Gloeckler et al. (2024); Weilbach et al. (2022) are not.

Further related work Our use of the term *causal* refers to the conditional independence structure of the prior and posterior programs and should not be confused with causal normalizing flows, which reflect a structural causal model and enable interventional and counterfactual queries (Javaloy et al., 2023; Visentin & Cheridito, 2025). Robust SBI methods, which account for model misspecification (Frazier et al., 2020; Ward et al., 2022; Huang et al., 2023; Kelly et al., 2024), are largely agnostic to the architecture of the inference network and thus complementary to CPE. We discuss both lines of work in more detail in Appendix B.

5 Experiments

We evaluate CPE extensively on several SBI benchmark models, the susceptible–infected–recovered (SIR) epidemiological model, and a realistic deep hierarchical model which is in similar forms often found in computational natural sciences.

We compare CPE against three state-of-the-art baselines: Flow Matching Posterior Estimation (FMPE; Wildberger et al., 2023), Neural Posterior Score Estimation (PSE; Sharrock et al., 2024), and All-in-One Posterior Estimation (AIO; Gloeckler et al., 2024). For AIO, we report results both with an undirected graph mask (AIO) and without a mask (AIO (no mask)) to be able to separate the effect of the mask from that of the transformer architecture. For FMPE, PSE, and AIO, we use the default configurations recommended in their original publications. To isolate the contribution of the causal masking mechanism (Section 3.2), we additionally compare against an ablation of CPE that uses a block-autoregressive network without causal masks, termed BNPE. All methods draw samples from the posterior using an adaptive Runge–Kutta 5(4) solver. For CPE, we additionally report results with an Euler solver with $T = 20$ steps.

For each benchmark task, we draw five pairs $(x_{\text{obs}}, \theta) \sim \pi(x, \theta)$ from the generative model and approximate the posterior $\pi(\theta|x_{\text{obs}})$ for each observation. Since all benchmark tasks have a tractable likelihood, which is not the case in real-world applications, we can obtain "ground-truth" posteriors with Markov chain Monte Carlo. We compare the approximate posteriors to these ground-truth posteriors for different simulation budgets N using the H-min divergence (Zhao et al., 2022; Dirmeier et al., 2025) and classifier two-sample tests (C2ST; Lopez-Paz & Oquab, 2017).64

See Appendices C and D for details on the benchmark models and experimental setup, including computation of reference posterior distributions, training, and exact neural network architectures.

5.1 SBI benchmarks

We first evaluate CPE in an experimental setting using eight common SBI benchmark models (see Appendix C for details).

CPE outperforms or matches all baselines on all eight benchmark models under both H-min and C2ST (Figure 5; C2ST results in Appendix E). Compared to BNPE, the non-causal variant of our architecture, CPE performs better on the majority of benchmark models and on par on the rest, indicating that the causal masking is critical to its accuracy.

CPE-Euler, which uses 20 Euler steps, performs on par with CPE-RK, which uses an adaptive Runge–Kutta solver, and in some cases even outperforms it. This is consistent with the learned

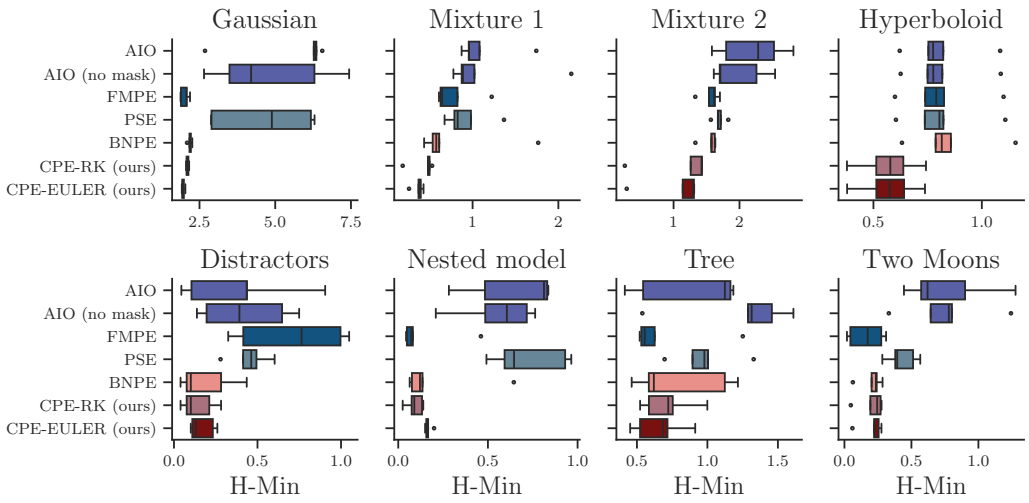


Figure 5: Performance of CPE and all baselines on eight common SBI benchmark models, measured by the H-min metric (lower is better).

trajectories having low curvature (Section 3.3), such that posterior samples can be drawn with few solver steps without loss in accuracy.

In addition, CPE reaches this performance with generally fewer trainable parameters than the baselines. On the mixture model with distractors, CPE uses 47k parameters, against 172k, 191k and 191k for AIO, FMPE and NPSE, respectively. On the hyperboloid, two moons and Gaussian mixture models, CPE uses 71k, against 172k, 190k and 190k for the same baselines.

5.2 SIR MODEL

We next evaluate CPE on a compartmental model from epidemiology. The model describes how the numbers of susceptible (S), infected (I) and recovered (R) individuals change as an infectious disease spreads through a population. It has only two parameters, the contact rate θ_0 and the recovery rate θ_1, but integrating its ODE produces a time series whose length is determined only by the time grid. Following Lueckmann et al. (2021), we integrate the ODE for a population of 10^6 individuals at 160 equally spaced time points in $[0, 160]$. At each time point t, we observe $Y_t \sim \text{Binomial}(1\,000, I_t/10^6)$. Instead of conditioning on the full series, we reduce it to a five-dimensional vector of summary statistics

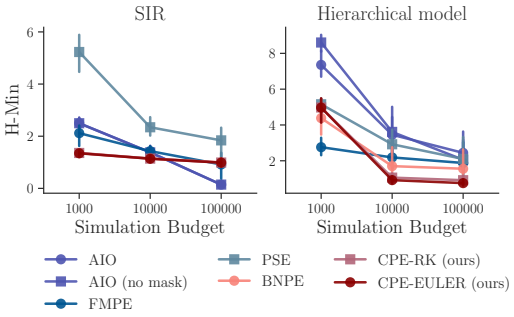


Figure 6: Evaluations for the hierarchical and SIR models, measured by the H-min metric (lower is better).

$$s(Y) = \left[\max_t Y_t,\ \arg\max_t Y_t,\ \sum_t Y_t,\ \bar{Y},\ \widehat{\text{Var}}(Y) \right],$$

where $\bar{Y}$ and $\widehat{\text{Var}}(Y)$ denote the empirical mean and variance of the observed series.[4]

Here, both CPE variants outperform all baselines, except at the largest simulation budget of $N = 100\,000$, where AIO performs best, consistent with the results reported by Gloeckler et al. (2024)

[4]CPE can also be applied to the raw time series directly. This, however, requires an appropriate summary network in addition to the architecture described in Appendix 8.

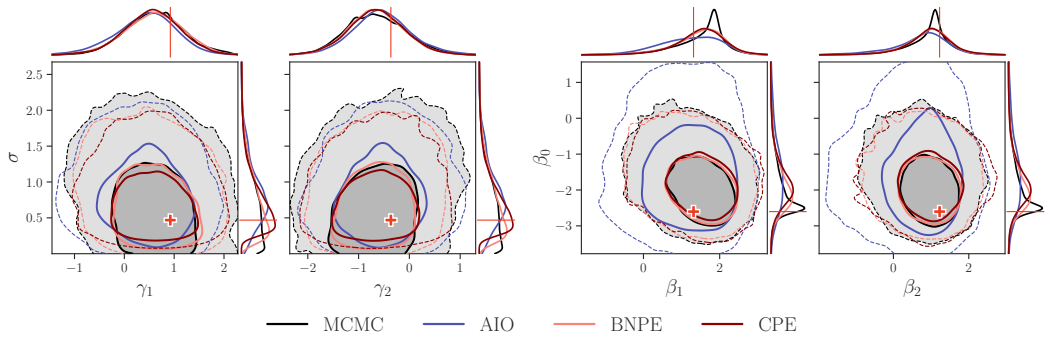


Figure 7: Posterior approximations of the compared methods for the hierarchical model (Section 5.3). Contours enclose 68% and 95% of the posterior mass.

(Figure 6, left panel). At this budget, the posterior predictive distributions of AIO also reproduce the observed data closely (Appendix 13). When the summary statistics are not sufficient for the parameters (but merely informative, as in our case), posterior predictive distributions of models trained on them typically show considerably higher variance (cf. posterior and posterior predictive distributions in Appendix E.2). This suggests that the transformer-based network of AIO is (as expected) more expressive than the MLP-based networks of FMPE, PSE, and CPE, and hence better suited to models using non-sufficient summary statistics. Conversely, CPE already yields convincing results at small simulation budgets, whereas AIO and FMPE require a large number of simulations.

5.3 Hierarchical model

To show that CPE is particularly effective when the simulator has a hierarchical or nested structure, we evaluate it on the following generative model:

$$\begin{aligned} \sigma &\sim \text{HalfNormal}(1) \\ \gamma &\sim \mathcal{N}_2(0, I) \\ \beta_k \mid \gamma &\sim \mathcal{N}_2(\gamma, I), \quad k = 1, 2, 3 \\ x \mid \beta, \sigma &\sim \mathcal{N}\left([\beta_1, \beta_2, \beta_3]^\top, \sigma^2 I\right) \end{aligned}$$

The model has nine parameters, including a positive scale parameter σ that governs the observation noise. Hierarchical models of this kind can be difficult to infer even for gradient-based MCMC samplers, since dependencies between levels of the hierarchy, and between location and scale parameters induce degenerate posterior geometries. At the same time, such models are ubiquitous in the natural sciences, for instance in computational biology and neuroscience, which underlines their practical relevance.

We evaluate CPE with simulation budgets of $N \in \{1\,000, 10\,000, 100\,000\}$. At the smallest budget of $N = 1\,000$, FMPE and BNPE, the non-causal variant of our method, outperform CPE (Figure 6, right panel). For $N \geq 10\,000$, CPE outperforms all baselines. Qualitatively, the posterior inferred by CPE matches the reference MCMC posterior considerably better than that of AIO, which, like CPE, accounts for the statistical dependencies induced by the graphical model (Figure 7). While both methods identify the top-level parameters γ well, AIO overestimates the scale σ, which in turn widens its posteriors over the lower-level parameters β (see Appendix E.3 for posterior and posterior predictive distributions of all methods).

6 Limitations

Our method has several limitations. In particular, the block matrix structure scales poorly with the number of parameters. While this is negligible for models with a small number of parameters (e.g., on the order of 10), it can become inefficient when hundreds of parameters need to be inferred. Transformer-based methods avoid this issue by embedding the data and parameters in a vector space

of fixed dimension (Gloeckler et al., 2024; Patacchiola et al., 2024). Future research could investigate how the expressivity of our approach can be combined with such efficient representations.

Furthermore, we invert the generative model by naively reversing its edges. While this was sufficient for the models in Section 5, it may fail to capture all conditional dependencies correctly (Stuhlmüller et al., 2013; Webb et al., 2018). In future work, we aim to account for these dependencies explicitly to further improve the accuracy of our method.

Finally, the block matrices in our approach are dense within the sparsity pattern induced by the graphical model, so their memory and computational costs grow quadratically with the number of parameters. Several structured parameterizations could reduce this cost. For dense or weakly structured posterior programs, the blocks could be restricted to low-rank factorizations $B_{ij} = \xi_{ij}\zeta_{ij}^\top$ with trainable vectors ξ_{ij} and ζ_{ij}. For sparse posterior programs, the block matrices resemble structured semiseparable or block-banded matrices (Fu et al., 2023; Dao & Gu, 2024). Dao & Gu (2024) show that semiseparable matrices, i.e., matrices whose submatrices below the diagonal have low rank, admit matrix–vector products in time linear in the sequence length. They use this structure to relate state space models to masked attention. Enforcing such structure would allow CPE to use these algorithms and could thereby address the scaling limitation discussed above. We have neither enforced such structures nor made use of these algorithms in this work. Investigating how structured low-rank matrices can be incorporated into SBI methods, and how much accuracy they sacrifice in exchange for computational efficiency, is an interesting direction for future research.

7 Conclusion

We presented a simple but powerful approach to SBI, called *Causal Posterior Estimation* (CPE), which incorporates the conditional dependence structure of the prior and posterior programs into a continuous normalizing flow. CPE parameterizes the flow's vector field with block-masked layers that follow the inverted graphical model, uses the prior as base distribution, and formulates the vector field as a convex combination of the current state and the network output. Across several benchmark tasks, CPE achieves state-of-the-art performance while generally requiring considerably fewer trainable parameters than dense and transformer-based baselines such as FMPE, PSE and AIO. The benefit of encoding the model structure is most apparent for hierarchical models, where the posterior inferred by CPE closely matches the reference MCMC posterior.

While we focus on a continuous normalizing flow in this work, CPE can equally be implemented as a discrete normalizing flow (see Appendix A, where we derive the log-determinants of the network for the discrete case). More generally, the idea of restricting the hypothesis space of a neural density estimator to the conditional dependencies of the underlying model is not tied to a particular generative framework and could be transferred to score-based or transformer-based SBI methods. Promising directions for future work include faithful inversions of the graphical model and structured low-rank parameterizations of the block matrices for models with many parameters.

AI Use Statement

In this work, we used a generative AI tool for writing assistance and literature search. In the main text, we used it to improve the clarity and conciseness of text drafted by the authors. In the supplementary material, we additionally used it to double-check the descriptions of the experimental models and settings against our code. We further used it to search for related work; all suggested references were verified and read by the authors before being cited. We have **not** used generative AI tools for research ideation, the design of the method and experiments, writing the code, or running and analyzing the experiments. We have reviewed all AI-assisted work: all AI-assisted text was checked and edited by the authors, and all reported experimental settings were confirmed against the code and results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics Statement

This work proposes a methodological contribution to simulation-based inference. It raises no ethical considerations.

Reproducibility statement

The supplementary material contains the complete source code, configurations, and a Snakemake workflow that runs all experiments automatically. With pinned dependencies and fixed random seeds, all results are fully reproducible on the same hardware.

References

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023.

Carlo Albert, Hans R Künsch, and Andreas Scheidegger. A simulated annealing approach to approximate Bayes computations. *Statistics and Computing*, 25:1217–1232, 2015.

Carlo Albert, Simone Ulzega, Simon Dirmeier, Andreas Scheidegger, Alberto Bassi, and Antonietta Mira. Simulated annealing ABC with multiple summary statistics. *arXiv e-prints*, pp. arXiv–2505, 2025.

Luca Ambrogioni, Gianluigi Silvestri, and Marcel van Gerven. Automatic variational inference with cascading flows. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Jonas Arruda, Niels Bracher, Ullrich Köthe, Jan Hasenauer, and Stefan T. Radev. Diffusion models in simulation-based inference: A tutorial review. *arXiv preprint arXiv:2512.20685*, 2025.

Igor Babuschkin, Kate Baumli, et al. The DeepMind JAX Ecosystem, 2020. URL http://github.com/deepmind.

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

Julius Berner, Lorenz Richter, and Karen Ullrich. An optimal control perspective on diffusion-based generative modeling. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Ayush Bharti, Charita Dellaporta, Yuga Hikida, and François-Xavier Briol. Amortised and provably-robust simulation-based inference. *arXiv preprint arXiv:2602.11325*, 2026.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

Jan Boelts, Cornelius Schröder, Jonas Beck, Jakob H. Macke, Michael Deistler, and Daniel Gedon. Mixed neural posterior estimation for simulators with discrete and continuous parameters. *arXiv preprint arXiv:2605.13551*, 2026.

Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

Jorge Carrasco-Pollo, Floor Eijkelboom, and Jan-Willem van de Meent. Pawsterior: Variational flow matching for structured simulation-based inference. *arXiv preprint arXiv:2602.13813*, 2026.

Giovanni Charles, Cosmo Santoni, Seth Flaxman, and Elizaveta Semenova. Tokenised flow matching for hierarchical simulation based inference. *arXiv preprint arXiv:2604.20723*, 2026.

Asic Q. Chen, Ruian Shi, Xiang Gao, Ricardo Baptista, and Rahul G. Krishnan. Structured neural networks for density estimation and causal inference. *arXiv preprint arXiv:2311.02221*, 2023.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018.

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning*, 2024.

Maximilian Dax, Theo Heimel, and Gilles Louppe. An introduction to Bayesian and frequentist simulation-based inference with machine learning. *arXiv preprint arXiv:2607.21702*, 2026.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Block neural autoregressive flow. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, 2020.

Michael Deistler, Pedro J. Goncalves, and Jakob H. Macke. Truncated proposals for scalable and hassle-free simulation-based inference. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Michael Deistler, Jan Boelts, Peter Steinbach, Guy Moss, and Thomas Moreau. Simulation-based inference: A practical guide. *arXiv preprint arXiv:2508.12939*, 2025.

Charita Dellaporta, Jeremias Knoblauch, Theodoros Damoulas, and Francois-Xavier Briol. Robust Bayesian inference for simulator-based models via the MMD posterior bootstrap. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, 2022.

Persi Diaconis and Donald Ylvisaker. Conjugate priors for exponential families. *The Annals of Statistics*, pp. 269–281, 1979.

Simon Dirmeier, Antonietta Mira, and Carlo Albert. Simulation-based inference with the Python package sbijax. *arXiv preprint arXiv:2409.19435*, 2024.

Simon Dirmeier, Carlo Albert, and Fernando Perez-Cruz. Simulation-based inference for high-dimensional data using surjective sequential neural likelihood estimation. In *Proceedings of the Forty-First Conference on Uncertainty in Artificial Intelligence*, 2025.

Zehao Dou and Yang Song. Diffusion posterior sampling for linear inverse problem solving: A filtering perspective. In *The Twelfth International Conference on Learning Representations*, 2024.

Florence Forbes, Hien Duy Nguyen, TrungTin Nguyen, and Julyan Arbel. Summary statistics and discrepancy measures for approximate Bayesian computation via surrogate posteriors. *Statistics and Computing*, 32(5):85, 2022.

David T Frazier, Christian P Robert, and Judith Rousseau. Model misspecification in approximate Bayesian computation: consequences and diagnostics. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(2):421–444, 2020.

Daniel Y Fu, Simran Arora, Jessica Grogan, Isys Johnson, Sabri Eyuboglu, Armin W Thomas, Benjamin Frederick Spector, Michael Poli, Atri Rudra, and Christopher Re. Monarch Mixer: A simple sub-quadratic GEMM-based architecture. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Ruiqi Gao, Emiel Hoogeboom, Jonathan Heek, Valentin De Bortoli, Kevin Patrick Murphy, and Tim Salimans. Diffusion models and Gaussian flow matching: Two sides of the same coin. In *The Fourth Blogpost Track at ICLR 2025*, 2025.

Tomas Geffner, George Papamakarios, and Andriy Mnih. Compositional score modeling for simulation-based inference. In *International Conference on Machine Learning*, 2023.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: Masked autoencoder for distribution estimation. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Manuel Gloeckler, Michael Deistler, and Jakob H. Macke. Adversarial robustness of amortized Bayesian inference. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Manuel Gloeckler, Michael Deistler, Christian Dietrich Weilbach, Frank Wood, and Jakob H. Macke. All-in-one simulation-based inference. In *Forty-first International Conference on Machine Learning*, 2024.

David Greenberg, Marcel Nonnenmacher, and Jakob Macke. Automatic posterior transformation for likelihood-free inference. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Joeri Hermans, Arnaud Delaunoy, François Rozet, Antoine Wehenkel, Volodimir Begy, and Gilles Louppe. A crisis in simulation-based inference? beware, your posterior approximations can be unfaithful. *Transactions on Machine Learning Research*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

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

Adrián Javaloy, Pablo Sánchez-Martín, and Isabel Valera. Causal normalizing flows: from theory to practice. *arXiv preprint arXiv:2306.05415*, 2023.

Haoyu Jiang, Yuexi Wang, and Yun Yang. Likelihood-free inference via structured score matching. *arXiv preprint arXiv:2603.29054*, 2026.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, 2022.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. In *Advances in Neural Information Processing Systems*, 2022.

Ryan P. Kelly, David J Nott, David Tyler Frazier, David J Warne, and Christopher Drovandi. Misspecification-robust sequential neural likelihood for simulation-based inference. *Transactions on Machine Learning Research*, 2024.

Ryan P. Kelly, David T. Frazier, David J. Warne, and Christopher C. Drovandi. Preconditioned robust neural posterior estimation for misspecified simulators. *arXiv preprint arXiv:2602.18004*, 2026.

Sherman Khoo, Dennis Prangle, Song Liu, and Mark Beaumont. Minimum distance summaries for robust neural posterior estimation. *arXiv preprint arXiv:2602.09161*, 2026.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Joohwan Ko and Tomas Geffner. Latent target score matching, with an application to simulation-based inference. *arXiv preprint arXiv:2602.07189*, 2026.

Johannes Köster and Sven Rahmann. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522, 2012.

Ravin Kumar, Colin Carroll, Ari Hartikainen, and Osvaldo Martin. ArviZ a unified library for exploratory analysis of Bayesian models in Python. *Journal of Open Source Software*, 4(33):1143, 2019.

William Laplante, Yuga Hikida, Charita Dellaporta, François-Xavier Briol, and Ayush Bharti. Conservative neural posterior estimation via distributionally robust training. *arXiv preprint arXiv:2605.28516*, 2026.

Julia Linhart, Gabriel Victorino Cardoso, Alexandre Gramfort, Sylvain Le Corff, and Pedro L. C. Rodrigues. Diffusion posterior sampling for simulation-based inference in tall data settings. *arXiv preprint arXiv:2404.07593*, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *International Conference on Learning Representations*, 2017.

Jan-Matthis Lueckmann, Pedro J Goncalves, Giacomo Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H Macke. Flexible statistical inference for mechanistic models of neural dynamics. In *Advances in Neural Information Processing Systems*, 2017.

Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking simulation-based inference. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, 2021.

Mayank Nautiyal, Li Ju, Melker Ernfors, Klara Hagland, Ville Holma, and Maximilian Werkö Söderholm. OneFlowSBI: One model, many queries for simulation-based inference. *arXiv preprint arXiv:2601.22951*, 2026.

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

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, 2017.

George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, 2019.

Massimiliano Patacchiola, Aliaksandra Shysheya, Katja Hofmann, and Richard E. Turner. Transformer neural autoregressive flows. In *ICML 2024 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2024.

Stefan T. Radev, Marvin Schmitt, Valentin Pratz, Umberto Picchini, Ullrich Köthe, and Paul-Christian Bürkner. JANA: Jointly amortized neural approximation of complex Bayesian models. In *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, 2023.

Pierre-Louis Ruhlmann, Michael Arbel, Florence Forbes, and Pedro L. C. Rodrigues. Flow matching calibration for simulation-based inference under model misspecification. *arXiv preprint arXiv:2509.23385*, 2025.

Marvin Schmitt, Valentin Pratz, Ullrich Köthe, Paul-Christian Bürkner, and Stefan T. Radev. Consistency models for scalable and fast simulation-based inference. In *Advances in Neural Information Processing Systems*, 2024.

Louis Sharrock, Jack Simons, Song Liu, and Mark Beaumont. Sequential neural score estimation: Likelihood-free inference with conditional score based diffusion models. In *Forty-first International Conference on Machine Learning*, 2024.

Gianluigi Silvestri, Emily Fertig, Dave Moore, and Luca Ambrogioni. Embedded-model flows: Combining the inductive biases of model-free deep learning and explicit probabilistic modeling. *arXiv preprint arXiv:2110.06021*, 2021.

Scott A Sisson, Yanan Fan, and Mark M Tanaka. Sequential Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 104(6):1760–1765, 2007.

Scott A Sisson, Yanan Fan, and Mark Beaumont. *Handbook of Approximate Bayesian Computation*. CRC Press, 2018.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Andreas Stuhlmüller, Jacob Taylor, and Noah Goodman. Learning stochastic inverses. In *Advances in Neural Information Processing Systems*, 2013.

Lorenzo Tomaselli, Valérie Ventura, and Larry Wasserman. Robust simulation based inference. *arXiv preprint arXiv:2508.02404*, 2025.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024.

Camille Touron, Gabriel V. Cardoso, Julyan Arbel, and Pedro L. C. Rodrigues. Theoretical guidelines for annealed Langevin dynamics in compositional simulation-based inference. *arXiv preprint arXiv:2605.21253*, 2026.

Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An introduction to probabilistic programming. *arXiv preprint arXiv:1809.10756*, 2018.

Francisco Vargas, Will Sussman Grathwohl, and Arnaud Doucet. Denoising diffusion samplers. In *The Eleventh International Conference on Learning Representations*, 2023.

Francisco Vargas, Shreyas Padhy, Denis Blessing, and Nikolas Nüsken. Transport meets variational inference: Controlled Monte Carlo diffusions. In *The Twelfth International Conference on Learning Representations*, 2024.

Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. Rank-normalization, folding, and localization: An improved $\widehat{R}$ for assessing convergence of MCMC. *Bayesian Analysis*, 16(2):667–718, 2021.

Yogesh Verma, Ayush Bharti, and Vikas Garg. Robust simulation-based inference under missing data via neural processes. In *The Thirteenth International Conference on Learning Representations*, 2025.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

Gabriele Visentin and Patrick Cheridito. Robust optimization in causal models and G-Causal normalizing flows. *arXiv preprint arXiv:2510.15458*, 2025.

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

Stefan Webb, Adam Golinski, Rob Zinkov, Tom Rainforth, Yee Whye Teh, Frank Wood, et al. Faithful inversion of generative models for effective amortized inference. In *Advances in Neural Information Processing Systems*, 2018.

Antoine Wehenkel and Gilles Louppe. Graphical normalizing flows. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, 2021.

Christian Weilbach, Boyan Beronov, Frank Wood, and William Harvey. Structured conditional continuous normalizing flows for efficient amortized inference in graphical models. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, 2020.

Christian Weilbach, William Harvey, and Frank Wood. Graphically structured diffusion models. *arXiv preprint arXiv:2210.11633*, 2022.

Jonas Bernhard Wildberger, Maximilian Dax, Simon Buchholz, Stephen R Green, Jakob H. Macke, and Bernhard Schölkopf. Flow matching for scalable simulation-based inference. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Wang Yuyan, Michael Evans, and David J Nott. Robust Bayesian methods using amortized simulation-based inference. *arXiv preprint arXiv:2504.09475*, 2025.

Qinsheng Zhang and Yongxin Chen. Path integral sampler: A stochastic control approach for sampling. In *International Conference on Learning Representations*, 2022.

Shengjia Zhao, Abhishek Sinha, Yutong He, Aidan Perreault, Jiaming Song, and Stefano Ermon. Comparing distributions by measuring differences that affect decision making. In *International Conference on Learning Representations*, 2022.

A Model details

CPE is a normalizing-flow-based neural posterior estimation method. In this section, we derive the continuous normalizing flow (CNF) variant of CPE introduced in the main text from first principles (Appendix A.1). Since the block transforms of CPE can be made invertible, we additionally describe a variant based on discrete normalizing flows (Appendix A.2). Throughout, q_0 denotes the base distribution on $\mathbb{R}^{d_\theta}$, $\nabla_\theta v$ the Jacobian of a vector field $v : \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_\theta}$, and

$$\operatorname{div} v(\theta) = \sum_{i=1}^{d_\theta} \frac{\partial v_i(\theta)}{\partial \theta_i} = \operatorname{tr}\big(\nabla_\theta v(\theta)\big)$$

its divergence. As in the main text, we omit the dependence of the vector field on the network weights ϕ.

A.1 CPE with continuous normalizing flows

A.1.1 Continuous normalizing flows

Let $v : [0,1] \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_\theta}$, $(t, \theta) \mapsto v_t(\theta)$, be a time-dependent vector field that is continuous in t and continuously differentiable in θ with bounded Jacobian. By the Picard–Lindelöf theorem, the ODE

$$\frac{\mathrm{d}}{\mathrm{d}t}\Lambda_t(\theta) = v_t\big(\Lambda_t(\theta)\big), \qquad \Lambda_0(\theta) = \theta, \tag{13}$$

has a unique global solution for every initial value θ, and the resulting flow $\Lambda_t : \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_\theta}$ is a diffeomorphism for every $t \in [0,1]$, whose inverse is obtained by integrating Equation (13) backwards in time. A CNF (Chen et al., 2018) parameterizes v_t with a neural network and defines the family of distributions

$$q_t = (\Lambda_t)_{\#}\, q_0, \tag{14}$$

i.e., q_t is the distribution of $\theta^{(t)} = \Lambda_t(\theta^{(0)})$ for $\theta^{(0)} \sim q_0$. The following two results characterize how q_t evolves in time.

Proposition 1 (Continuity equation)**.** *The densities q_t in Equation (14) satisfy*

$$\frac{\partial}{\partial t} q_t(\theta) = -\operatorname{div}\big(q_t(\theta)\, v_t(\theta)\big). \tag{15}$$

Proof. Let $\varphi : \mathbb{R}^{d_\theta} \to \mathbb{R}$ be a smooth test function with compact support. By definition of the pushforward, $\mathbb{E}_{q_t}[\varphi(\theta)] = \mathbb{E}_{q_0}[\varphi(\Lambda_t(\theta))]$. Differentiating both sides with respect to t and using Equation (13) yields

$$\int \varphi(\theta)\, \frac{\partial q_t(\theta)}{\partial t}\, \mathrm{d}\theta = \mathbb{E}_{q_0}\Big[\nabla\varphi\big(\Lambda_t(\theta)\big)^\top v_t\big(\Lambda_t(\theta)\big)\Big] = \int \nabla\varphi(\theta)^\top v_t(\theta)\, q_t(\theta)\, \mathrm{d}\theta$$
$$= -\int \varphi(\theta)\, \mathrm{div}\big(q_t(\theta)\, v_t(\theta)\big)\, \mathrm{d}\theta,$$

where the last step is integration by parts, whose boundary term vanishes since φ has compact support. Since the identity holds for all test functions φ, Equation (15) follows. □

Proposition 2 (Instantaneous change of variables)**.** *Along a trajectory $\theta^{(t)} = \Lambda_t(\theta^{(0)})$, the log-density of the CNF evolves as*

$$\frac{\mathrm{d}}{\mathrm{d}t} \log q_t\big(\theta^{(t)}\big) = -\mathrm{div}\, v_t\big(\theta^{(t)}\big), \tag{16}$$

such that

$$\log q_1\big(\theta^{(1)}\big) = \log q_0\big(\theta^{(0)}\big) - \int_0^1 \mathrm{div}\, v_t\big(\theta^{(t)}\big)\, \mathrm{d}t, \tag{17}$$

which is Equation (4) of the main text in log-space.

Proof. By the chain rule and Equation (13),

$$\frac{\mathrm{d}}{\mathrm{d}t} \log q_t\big(\theta^{(t)}\big) = \frac{\partial \log q_t}{\partial t}\big(\theta^{(t)}\big) + \nabla \log q_t\big(\theta^{(t)}\big)^\top v_t\big(\theta^{(t)}\big).$$

Expanding the divergence in Equation (15) as $\mathrm{div}(q_t v_t) = q_t\, \mathrm{div}\, v_t + \nabla q_t^\top v_t$ and dividing by q_t gives $\partial_t \log q_t = -\mathrm{div}\, v_t - \nabla \log q_t^\top v_t$. Inserting this into the display above, the two gradient terms cancel, which yields Equation (16). Integrating from $t = 0$ to $t = 1$ yields Equation (17). □

Equation (17) is the continuous-time analogue of the change-of-variables formula for discrete flows (Equation (26) below). To see this, let $J_t(\theta) = \partial\Lambda_t(\theta)/\partial\theta$. Differentiating Equation (13) with respect to θ gives the variational equation

$$\frac{\mathrm{d}}{\mathrm{d}t} J_t(\theta) = \nabla_\theta v_t\big(\Lambda_t(\theta)\big)\, J_t(\theta), \qquad J_0(\theta) = I.$$

By Jacobi's formula,

$$\frac{\mathrm{d}}{\mathrm{d}t} \log\big|\det J_t(\theta)\big| = \mathrm{tr}\left(J_t(\theta)^{-1}\, \frac{\mathrm{d}}{\mathrm{d}t} J_t(\theta)\right) = \mathrm{tr}\left(\nabla_\theta v_t\big(\Lambda_t(\theta)\big)\right) = \mathrm{div}\, v_t\big(\Lambda_t(\theta)\big),$$

such that $\log|\det J_1(\theta^{(0)})| = \int_0^1 \mathrm{div}\, v_t(\theta^{(t)})\, \mathrm{d}t$, and Equation (17) is exactly $q_1(\theta^{(1)}) = q_0(\theta^{(0)})\,|\det J_1(\theta^{(0)})|^{-1}$.

Maximum likelihood training of a CNF requires evaluating Equation (17) for every training sample in every gradient step, i.e., solving the augmented ODE for $(\theta^{(t)}, \log q_t(\theta^{(t)}))$ and computing $\mathrm{div}\, v_t$, which costs d_θ vector–Jacobian products or a stochastic trace estimate per function evaluation. Flow matching avoids this entirely.

A.1.2 Flow matching

Let π_{target} denote the distribution we want to sample from, and let $(\varrho_t)_{t\in[0,1]}$ be a probability path with $\varrho_0 = q_0$ and $\varrho_1 = \pi_{\mathrm{target}}$, generated by a vector field u_t, i.e., (ϱ_t, u_t) satisfy the continuity equation (15). If $v_t = u_t$, then $q_t = \varrho_t$ for all t, since the continuity equation has a unique solution for Lipschitz vector fields. *Flow matching* (Lipman et al., 2023) therefore regresses the neural network v_t onto u_t:

$$\hat{\phi} = \arg\min_\phi\, \mathcal{L}_{\mathrm{FM}}(\phi), \qquad \mathcal{L}_{\mathrm{FM}}(\phi) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\, \theta\sim\varrho_t}\Big[\big\|v_t(\theta) - u_t(\theta)\big\|^2\Big]. \tag{18}$$

For any nontrivial target, neither ϱ_t nor u_t is available in closed form. Lipman et al. (2023) instead construct both as mixtures of simple *conditional* paths. Let $z \sim \pi(z)$ be a conditioning variable, and

let $\varrho_t(\theta|z)$ be conditional probability paths generated by conditional vector fields $u_t(\theta|z)$. We define the marginal path and the marginal vector field as

$$\varrho_t(\theta) = \int \varrho_t(\theta|z)\,\pi(z)\,\mathrm{d}z, \tag{19}$$

$$u_t(\theta) = \int u_t(\theta|z)\,\frac{\varrho_t(\theta|z)\,\pi(z)}{\varrho_t(\theta)}\,\mathrm{d}z = \mathbb{E}\big[u_t(\theta|z)\,\big|\,\theta\big], \tag{20}$$

where the conditional expectation is taken with respect to the posterior $\varrho_t(z|\theta) \propto \varrho_t(\theta|z)\,\pi(z)$.

Theorem 3 (Lipman et al., 2023)**.** *If the conditional vector fields $u_t(\theta|z)$ generate the conditional probability paths $\varrho_t(\theta|z)$, then the marginal vector field u_t in Equation (20) generates the marginal probability path ϱ_t in Equation (19), for any distribution $\pi(z)$.*

Proof. It suffices to show that (ϱ_t, u_t) satisfy the continuity equation. Using that each conditional pair satisfies it,

$$\begin{aligned}\frac{\partial}{\partial t}\varrho_t(\theta) &= \int \frac{\partial}{\partial t}\varrho_t(\theta|z)\,\pi(z)\,\mathrm{d}z = -\int \mathrm{div}\big(u_t(\theta|z)\,\varrho_t(\theta|z)\big)\,\pi(z)\,\mathrm{d}z \\ &= -\mathrm{div}\Big(\int u_t(\theta|z)\,\varrho_t(\theta|z)\,\pi(z)\,\mathrm{d}z\Big) = -\mathrm{div}\big(u_t(\theta)\,\varrho_t(\theta)\big),\end{aligned}$$

where the last equality is Equation (20) multiplied by $\varrho_t(\theta)$. □

Theorem 3 shows that u_t is the right regression target, but evaluating it still requires the intractable integral in Equation (20). The key observation of Lipman et al. (2023) is that we can regress onto the conditional vector field instead:

$$\mathcal{L}_{\mathrm{CFM}}(\phi) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\, z\sim\pi(z),\, \theta\sim\varrho_t(\theta|z)}\Big[\big\|v_t(\theta) - u_t(\theta|z)\big\|^2\Big], \tag{21}$$

which has the same gradients with respect to ϕ as Equation (18) (Lipman et al., 2023, Theorem 2). The objective in Equation (21) is simulation-free: it only requires samples from $\pi(z)$ and $\varrho_t(\theta|z)$, and evaluations of $u_t(\theta|z)$.

Linear interpolation paths. Following Liu et al. (2023), we condition on both endpoints, $z = (\theta^{(0)}, \theta^{(1)}) \sim q_0 \otimes \pi_{\mathrm{target}}$, and connect them by a straight line:

$$\begin{aligned}&\varrho_t(\theta|\theta^{(0)},\theta^{(1)}) = \delta\big(\theta - \psi_t(\theta^{(0)},\theta^{(1)})\big), \qquad \psi_t(\theta^{(0)},\theta^{(1)}) = t\,\theta^{(1)} + (1-t)\,\theta^{(0)}, \\ &u_t(\theta|\theta^{(0)},\theta^{(1)}) = \theta^{(1)} - \theta^{(0)}.\end{aligned} \tag{22}$$

Here, $u_t(\cdot|\theta^{(0)},\theta^{(1)})$ generates $\varrho_t(\cdot|\theta^{(0)},\theta^{(1)})$ in the sense that $\frac{\mathrm{d}}{\mathrm{d}t}\psi_t = \theta^{(1)} - \theta^{(0)}$. As in Equation (5) of the main text, we write $\theta^{(t)} = \psi_t(\theta^{(0)},\theta^{(1)})$ for the interpolant from here on; note that it is in general not the trajectory $\Lambda_t(\theta^{(0)})$ of the learned flow.

Proposition 4. *Let q_0 have a density. The linear interpolation paths in Equation (22) satisfy:*

1. *$\varrho_0 = q_0$ and $\varrho_1 = \pi_{\mathrm{target}}$ exactly, for an arbitrary base distribution q_0.*
2. *For $t \in [0,1)$, the marginal vector field is*

$$u_t(\theta) = \mathbb{E}\big[\theta^{(1)} - \theta^{(0)}\,\big|\,\theta^{(t)} = \theta\big] = \frac{\mathbb{E}\big[\theta^{(1)}\,\big|\,\theta^{(t)} = \theta\big] - \theta}{1-t}. \tag{23}$$

3. *If $q_0 = \mathcal{N}(0, I)$, the marginal path ϱ_t coincides with the optimal transport path of Lipman et al. (2023) with $\sigma_{\min} = 0$.*

Proof. (i) follows from $\psi_0(\theta^{(0)},\theta^{(1)}) = \theta^{(0)}$ and $\psi_1(\theta^{(0)},\theta^{(1)}) = \theta^{(1)}$. For (ii), the first equality is Equation (20). On the event $\theta^{(t)} = \theta$, we have $\theta^{(0)} = (\theta - t\,\theta^{(1)})/(1-t)$, hence $\theta^{(1)} - \theta^{(0)} = (\theta^{(1)} - \theta)/(1-t)$, and taking the conditional expectation gives the second equality. For (iii), conditionally on $\theta^{(1)}$, the interpolant $\theta^{(t)} = t\,\theta^{(1)} + (1-t)\,\theta^{(0)}$ with $\theta^{(0)} \sim \mathcal{N}(0, I)$ is distributed as $\mathcal{N}(t\,\theta^{(1)}, (1-t)^2 I)$, which is the conditional path $\mathcal{N}(t\,\theta^{(1)}, (1-(1-\sigma_{\min})t)^2 I)$ of Lipman et al. (2023) with $\sigma_{\min} = 0$. Marginalizing over the same π_{target} yields the same ϱ_t. □

Property (i) is what allows CPE to use the prior as base distribution (Appendix A.1.3). While every conditional path is straight, the marginal flow generated by Equation (23) is in general curved, since conditional paths belonging to different endpoint pairs intersect. Its trajectories nevertheless never cross, by uniqueness of ODE solutions (Liu et al., 2023). Combining Equations (21) and (22) yields the objective

$$\mathcal{L}(\phi) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\,\theta^{(0)}\sim q_0,\,\theta^{(1)}\sim\pi_{\text{target}}}\Big[\big\|v_t\big(\theta^{(t)}\big) - \big(\theta^{(1)} - \theta^{(0)}\big)\big\|^2\Big]. \tag{24}$$

A.1.3 Neural posterior estimation

For neural posterior estimation, the target distribution is the posterior, $\pi_{\text{target}} = \pi(\theta|x)$, for every observation x. We therefore make the vector field conditional on the data, $v : [0,1] \times \mathbb{R}^{d_\theta} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_\theta}$, $(t,\theta,x) \mapsto v_t(\theta,x)$. For fixed x, Proposition 4 gives the target vector field

$$u_t(\theta|x) = \mathbb{E}\big[\theta^{(1)} - \theta^{(0)} \,\big|\, \theta^{(t)} = \theta,\, x\big], \qquad \theta^{(0)} \sim q_0,\ \theta^{(1)} \sim \pi(\theta|x),$$

whose flow transports q_0 to $\pi(\theta|x)$. To amortize over observations, we average the objective in Equation (24) over the evidence $\pi(x)$,

$$\mathcal{L}_{\text{NPE}}(\phi) = \mathbb{E}_{\pi(x)}\,\mathbb{E}_{t\sim\mathcal{U}(0,1),\,\theta^{(0)}\sim q_0,\,\theta^{(1)}\sim\pi(\theta|x)}\Big[\big\|v_t\big(\theta^{(t)},x\big) - \big(\theta^{(1)} - \theta^{(0)}\big)\big\|^2\Big].$$

Neither $\pi(x)$ nor $\pi(\theta|x)$ can be sampled directly, but by Bayes' theorem $\pi(x)\,\pi(\theta|x) = \pi(\theta)\,\pi(x|\theta)$, such that

$$\mathcal{L}_{\text{NPE}}(\phi) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\,\theta^{(0)}\sim q_0,\,\theta^{(1)}\sim\pi(\theta),\,x\sim\pi(x|\theta^{(1)})}\Big[\big\|v_t\big(\theta^{(t)},x\big) - \big(\theta^{(1)} - \theta^{(0)}\big)\big\|^2\Big], \tag{25}$$

which only requires draws from the prior and the simulator (Wildberger et al., 2023). This is the objective in Equation (6) of the main text.

Prior as base distribution. Since Proposition 4(i) holds for arbitrary base distributions, CPE uses the prior itself as base, $q_0 = \pi(\theta)$ (Section 3.1). Both endpoints $\theta^{(0)}$ and $\theta^{(1)}$ are then prior draws, and the learned flow transports the prior to the posterior. In practice, we sample $t \sim \mathcal{U}(\varepsilon, 1)$ with $\varepsilon = 10^{-3}$ for numerical stability, and integrate the ODE from $t = \varepsilon$ to $t = 1$. Since the ODE is not constrained to the support of the prior, a sample can leave it, in which case we discard it (Section 3.3).

A.1.4 Architecture

In the continuous case, the vector field is a function of time, parameters and data, $v : [0,1] \times \mathbb{R}^{d_\theta} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_\theta}$. Since a CNF does not require the network to be invertible, the activation functions act and g (Section 3.2) can be arbitrary; we use $\mathrm{act} = \tanh$ and the identity for g. See Figure 8 for an overview. The structural masks enforce the dependency structure of the posterior program on the vector field, which we formalize below.

Recall the topological ordering $\omega = \{\omega_1, \ldots, \omega_n\}$ of the n parameter nodes of the posterior program (Section 3.2), where a node θ_{ω_i} may be multi-dimensional, and denote by $\mathrm{pa}(\theta_{\omega_i})$ and $\mathrm{an}(\theta_{\omega_i})$ its parents and ancestors in the posterior program.

Lemma 5. *For all t, x and $i, j \in \{1, \ldots, n\}$,*

$$\frac{\partial\, v_t(\theta,x)_{\omega_i}}{\partial\, \theta_{\omega_j}} = 0 \qquad \textit{unless } j = i \textit{ or } \theta_{\omega_j} \in \mathrm{an}(\theta_{\omega_i}).$$

In particular, $\nabla_\theta v_t(\theta,x)$ is block lower-triangular with respect to ω.

Proof. Consider a single block $g^{(l)}_{t,x}$ and denote by $h^{(k)}_{\omega_i}$ the hidden units of node θ_{ω_i} after the k-th layer $f^{(k)}$. The structural mask of $f^{(k)}$ connects the output units of θ_{ω_i} only to the input units of θ_{ω_i} and its parents $\mathrm{pa}(\theta_{\omega_i})$. The activation acts elementwise, the reversal of hidden units permutes units within a node, and the linear projection of the conditioning vector does not depend on θ. By induction over k, $h^{(k)}_{\omega_i}$ depends on θ_{ω_j} only if $j = i$ or θ_{ω_j} is an ancestor of θ_{ω_i} at distance at most k. The convex combination in Equation (8) adds α_l times the identity, which preserves this structure, and so does the composition of L blocks in Equation (7), since the ancestor relation is transitive. □

Within a node with more than one dimension, the masks are additionally autoregressive over the node's dimensions. In the continuous variant, we reverse the order of the hidden units of each node after every intermediate layer $f^{(2)}, \ldots, f^{(K-1)}$, which alternates the autoregressive order and lets all dimensions of a node depend on each other. By Lemma 5, this does not affect the dependencies between nodes.

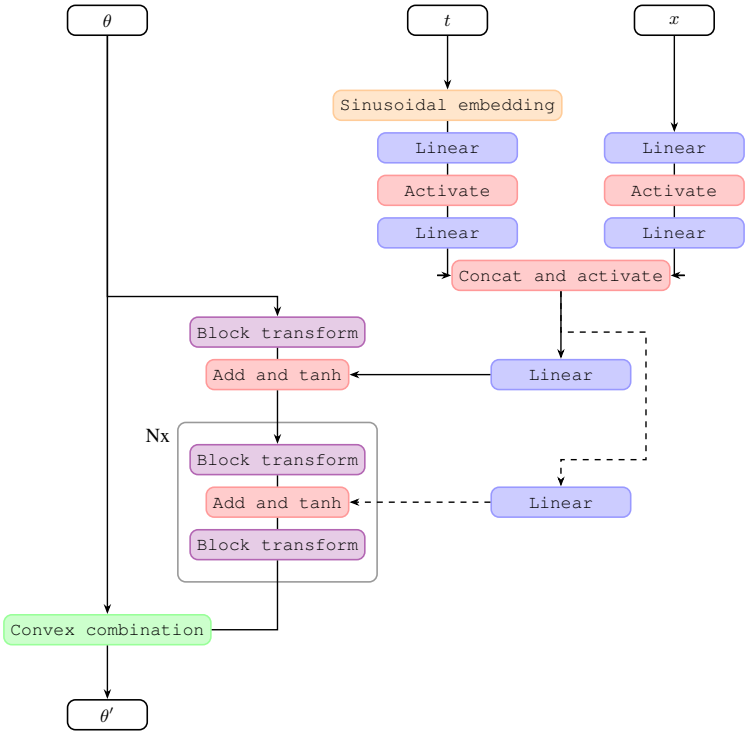


Figure 8: Continuous-time CPE architecture. The data x and the time t are embedded using an MLP and a sinusoidal embedding followed by an MLP, respectively, and concatenated to form the conditioning vector c. We add a linear projection of c after the first block transform, before applying the activation function (Equation (12)). While it is possible to condition after every block transform (dashed lines), we only do so after the first. Finally, the output is combined with the input through the convex combination in Equation (8).

A.2 CPE WITH DISCRETE NORMALIZING FLOWS

A.2.1 DISCRETE NORMALIZING FLOWS

Like CNFs, discrete normalizing flows define a distribution as the pushforward $q = \Lambda_{\#}\, q_0$ of the base distribution under a diffeomorphism $\Lambda : \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_\theta}$, but parameterize Λ directly instead of through an ODE. For a measurable set $\mathcal{A}$, substituting $\theta^{(0)} = \Lambda^{-1}(\theta)$ gives

$$\mathbb{P}\big(\Lambda(\theta^{(0)}) \in \mathcal{A}\big) = \int_{\Lambda^{-1}(\mathcal{A})} q_0\big(\theta^{(0)}\big)\, \mathrm{d}\theta^{(0)} = \int_{\mathcal{A}} q_0\big(\Lambda^{-1}(\theta)\big) \left|\det \frac{\partial \Lambda^{-1}(\theta)}{\partial \theta}\right| \mathrm{d}\theta,$$

and hence the change-of-variables formula

$$q(\theta) = q_0\big(\Lambda^{-1}(\theta)\big) \left|\det \frac{\partial \Lambda^{-1}(\theta)}{\partial \theta}\right|. \tag{26}$$

If $\Lambda^{-1} = T_M \circ \cdots \circ T_1$ is a composition, the chain rule and the multiplicativity of the determinant give

$$\log \left|\det \frac{\partial \Lambda^{-1}(\theta)}{\partial \theta}\right| = \sum_{m=1}^{M} \log \left|\det \frac{\partial T_m(\eta_{m-1})}{\partial \eta_{m-1}}\right|, \qquad \eta_0 = \theta,\ \eta_m = T_m(\eta_{m-1}).$$

Evaluating a general determinant costs $\mathcal{O}(d_\theta^3)$. If the Jacobian is triangular, however, the determinant is the product of its diagonal entries and costs $\mathcal{O}(d_\theta)$. Autoregressive flows, to which the block transforms of CPE belong, have triangular Jacobians by construction.

A.2.2 Invertible block transforms

We now make the block transforms of Section 3.2 invertible, following the block neural autoregressive flow of De Cao et al. (2020). Order the coordinates of θ by ω, let $h^{(0)} = \theta$, and for $k = 1, \dots, K$ let

$$\tilde{h}^{(k)} = B^{(k)} h^{(k-1)} + b^{(k)}, \qquad h^{(k)} = \begin{cases} \mathrm{act}\big(\tilde{h}^{(k)}\big), & k < K, \\ \tilde{h}^{(k)}, & k = K, \end{cases}$$

i.e., Equation (12) without the activation in the last layer. In the conditional case, the linear projection $\texttt{Linear}(c)$ of the conditioning vector c is added to $\tilde{h}^{(1)}$. Each coordinate θ_i owns a block of $d_{\mathrm{out}}^{(k)}$ hidden units in layer k, with $d_{\mathrm{in}}^{(k)} = d_{\mathrm{out}}^{(k-1)}$ and $d_{\mathrm{in}}^{(1)} = d_{\mathrm{out}}^{(K)} = 1$, such that $B^{(k)}$ is a $d_\theta \times d_\theta$ grid of blocks $B_{ij}^{(k)} \in \mathbb{R}^{d_{\mathrm{out}}^{(k)} \times d_{\mathrm{in}}^{(k)}}$. The structural mask makes $B^{(k)}$ block lower-triangular (Figure 4), and the diagonal blocks are $B_{ii}^{(k)} = g(\bar{B}_{ii}^{(k)})$, where $g = \exp$ is applied elementwise to unconstrained weights $\bar{B}_{ii}^{(k)}$.

Proposition 6. *Let* act *be strictly increasing and differentiable. Then $\lambda = h^{(K)}$ has a lower-triangular Jacobian with strictly positive diagonal entries*

$$\frac{\partial \lambda_i}{\partial \theta_i} = B_{ii}^{(K)}\, D_i^{(K-1)}\, B_{ii}^{(K-1)} \cdots D_i^{(1)}\, B_{ii}^{(1)}, \qquad D_i^{(k)} = \mathrm{diag}\big(\mathrm{act}'(\tilde{h}_i^{(k)})\big), \tag{27}$$

where $\tilde{h}_i^{(k)} \in \mathbb{R}^{d_{\mathrm{out}}^{(k)}}$ are the pre-activations of the block of coordinate θ_i. In particular, $\theta_i \mapsto \lambda_i(\theta)$ is strictly increasing for fixed preceding coordinates.

Proof. By the chain rule, $\nabla_\theta \lambda = B^{(K)} D^{(K-1)} B^{(K-1)} \cdots D^{(1)} B^{(1)}$, where $D^{(k)} = \mathrm{diag}(\mathrm{act}'(\tilde{h}^{(k)}))$ is block-diagonal and each $B^{(k)}$ is block lower-triangular. Products of block lower-triangular matrices are block lower-triangular, and the diagonal blocks of the product are the products of the diagonal blocks, which gives Equation (27). Since $d_{\mathrm{in}}^{(1)} = d_{\mathrm{out}}^{(K)} = 1$, the right-hand side is a scalar. All entries of $B_{ii}^{(k)} = \exp(\bar{B}_{ii}^{(k)})$ and $D_i^{(k)}$ are strictly positive, so the product is strictly positive. □

Since the conditioning term does not depend on θ, it does not contribute to the Jacobian. To evaluate Equation (27) stably, we compute it in log-space. Let

$$\big[\,\mathrm{LME}(P, Q)\big]_{jl} = \log \sum_m \exp\big(P_{jm} + Q_{ml}\big),$$

such that $\log(PQ) = \mathrm{LME}(\log P, \log Q)$ for matrices with positive entries. With $\log B_{ii}^{(k)} = \bar{B}_{ii}^{(k)}$, the recursion

$$\begin{aligned} \ell_i^{(1)} &= \bar{B}_{ii}^{(1)} + \log \mathrm{act}'\big(\tilde{h}_i^{(1)}\big), \\ \ell_i^{(k)} &= \mathrm{LME}\big(\bar{B}_{ii}^{(k)},\, \ell_i^{(k-1)}\big) + \log \mathrm{act}'\big(\tilde{h}_i^{(k)}\big), \qquad k = 2, \dots, K-1, \\ \log \frac{\partial \lambda_i}{\partial \theta_i} &= \mathrm{LME}\big(\bar{B}_{ii}^{(K)},\, \ell_i^{(K-1)}\big), \end{aligned}$$

where $\log \mathrm{act}'(\tilde{h}_i^{(k)})$ is added to every column, yields $\log|\det \nabla_\theta \lambda| = \sum_{i=1}^{d_\theta} \log \partial \lambda_i / \partial \theta_i$. For $\mathrm{act} = \tanh$, the log-derivative can be computed without cancellation as

$$\log \tanh'(s) = \log\big(1 - \tanh^2(s)\big) = 2\big(\log 2 - s - \mathrm{softplus}(-2s)\big).$$

Convex combination. The prior-guided block $g_x^{(l)}(u) = \alpha_l\, u + (1-\alpha_l)\, \lambda^{(l)}(u, x)$ from Equation (8) has Jacobian $\alpha_l I + (1-\alpha_l)\, \nabla_u \lambda^{(l)}$, which is again lower-triangular with diagonal entries $\alpha_l + (1-\alpha_l)\, \partial \lambda_i^{(l)} / \partial u_i > 0$. Parameterizing $\alpha_l = 1 - \sigma(\kappa_l) = 1/(1+e^{\kappa_l})$ with an unconstrained $\kappa_l \in \mathbb{R}$ and writing $\ell_i = \log \partial \lambda_i^{(l)} / \partial u_i$,

$$\log \frac{\partial g_{x,i}^{(l)}}{\partial u_i} = \log \frac{1 + e^{\kappa_l + \ell_i}}{1 + e^{\kappa_l}} = \mathrm{softplus}(\kappa_l + \ell_i) - \mathrm{softplus}(\kappa_l).$$

The convex combination also makes the block surjective. Since $\tanh$ is bounded and the last layer is affine, each $\lambda_i^{(l)}$ is bounded, say $|\lambda_i^{(l)}| \le M_i$. Hence $u_i \mapsto g_{x,i}^{(l)}(u)$ is strictly increasing and tends to $\pm\infty$ as $u_i \to \pm\infty$, and by induction over the coordinates, $g_x^{(l)}$ is a bijection of $\mathbb{R}^{d_\theta}$. Without the convex combination ($\alpha_l = 0$), the image of $\lambda^{(l)}$ would be bounded, and Equation (26) would not define a normalized density for base distributions with unbounded support.

No reversal. Reversing the hidden units of a multi-dimensional node, as in the continuous variant, would break the triangular structure within that node, and Proposition 6 would no longer apply. The discrete variant therefore omits the reversal.

A.2.3 Neural posterior estimation

For posterior estimation, we condition the flow on the data and compose L prior-guided blocks, $\Lambda_x^{-1} = g_x^{(L)} \circ \cdots \circ g_x^{(1)}$, which defines

$$q(\theta|x) = q_0\big(\Lambda_x^{-1}(\theta)\big) \left| \det \frac{\partial \Lambda_x^{-1}(\theta)}{\partial \theta} \right|. \tag{28}$$

In contrast to the continuous case, evaluating $q(\theta|x)$ does not require solving an ODE, so we can train by maximum likelihood, following the discrete NPE literature (e.g., Greenberg et al., 2019):

$$\hat{\phi} = \arg\max_{\phi} \mathbb{E}_{\theta \sim \pi(\theta),\, x \sim \pi(x|\theta)} \big[\log q(\theta|x) \big]. \tag{29}$$

Using $\pi(\theta)\, \pi(x|\theta) = \pi(x)\, \pi(\theta|x)$,

$$\mathbb{E}_{\pi(\theta,x)} \big[\log q(\theta|x) \big] = -\mathbb{E}_{\pi(x)} \Big[\mathrm{KL}\big(\pi(\theta|x) \,\big\|\, q(\theta|x) \big) \Big] + \mathbb{E}_{\pi(\theta,x)} \big[\log \pi(\theta|x) \big],$$

where the second term does not depend on ϕ. Maximum likelihood training therefore minimizes the expected forward KL divergence to the posterior.

Sampling. To sample, we draw $\theta^{(0)} \sim q_0$ and compute $\theta = \Lambda_x(\theta^{(0)})$ by inverting the blocks $g_x^{(L)}, \ldots, g_x^{(1)}$ in turn. A block has no analytical inverse, but its triangular structure allows inverting it one coordinate at a time, in the order ω: to solve $g_x^{(l)}(u) = w$ for u, given $u_1, \ldots, u_{i-1}$, the coordinate u_i solves $g_{x,i}^{(l)}(u_1, \ldots, u_i) = w_i$, where the left-hand side is strictly increasing in u_i. The bound $|\lambda_i^{(l)}| \le M_i$ brackets the solution,

$$u_i \in \left[\frac{w_i - (1-\alpha_l) M_i}{\alpha_l},\ \frac{w_i + (1-\alpha_l) M_i}{\alpha_l} \right],$$

such that bisection converges to precision ϵ in $\mathcal{O}\big(\log_2(M_i/(\alpha_l\, \epsilon))\big)$ iterations per coordinate.

A.2.4 Architecture

In the discrete case, we drop the time argument, and each block $g_x^{(l)}$ is a transform $\mathbb{R}^{d_\theta} \to \mathbb{R}^{d_\theta}$ that is invertible for every x. In Figure 8, this corresponds to removing the t-branch, such that the conditioning vector c consists of the data embedding only. We use $g = \exp$ and $\mathrm{act} = \tanh$, and omit the reversal of hidden units.

B Extended related work

Simulation-based inference Beyond the methods discussed in Section 4, continuous-time SBI methods have been extended to lower-variance score matching objectives (Ko & Geffner, 2026; Jiang et al., 2026). Nautiyal et al. (2026) train a single flow matching model over the joint distribution with query-aware masking, which supports posterior sampling, likelihood estimation and arbitrary conditionals with one model.

Causal flows A parallel line of work studies normalizing flows whose architecture reflects a structural causal model. Javaloy et al. (2023) establish a correspondence between autoregressive flows and identifiable causal models via triangular monotonically increasing maps, which enables interventional and counterfactual queries, and Visentin & Cheridito (2025) enforce causal constraints through hypercoupling flows that transform each variable conditioned on its parents in a given DAG. We emphasize that our use of the term *causal* refers to the conditional independence structure induced by the prior and posterior programs, and that we make no identifiability or interventional claims. Adapting CPE to answer counterfactual queries is an interesting direction for future work.

Robust SBI An important but orthogonal line of research develops *robust* SBI methods that account for model misspecification and poor calibration (Frazier et al., 2020; Hermans et al., 2022; Ward et al., 2022; Dellaporta et al., 2022; Huang et al., 2023; Gloeckler et al., 2023; Kelly et al., 2024; Verma et al., 2025; Yuyan et al., 2025; Tomaselli et al., 2025; Ruhlmann et al., 2025; Bharti et al., 2026; Khoo et al., 2026; Kelly et al., 2026; Laplante et al., 2026). These methods modify the training objective (Laplante et al., 2026; Bharti et al., 2026), learn robust summary statistics (Huang et al., 2023; Khoo et al., 2026), or apply a post-hoc correction to an already trained estimator (Ruhlmann et al., 2025; Kelly et al., 2026), and are for the most part agnostic to the architecture of the inference network. CPE instead changes the inductive bias of that architecture under a correctly specified simulator. The two are therefore complementary, and applying robust objectives to CPE is a natural extension.

C Experimental models

We evaluate CPE on benchmark models from the recent literature (Papamakarios et al., 2019; Greenberg et al., 2019; Lueckmann et al., 2021; Forbes et al., 2022; Vargas et al., 2024; Gloeckler et al., 2024) and develop new benchmark tasks. We describe the different models below.

C.1 Linear Gaussian

The linear Gaussian benchmark task is defined by the following generative process:

$$\theta \sim \mathcal{N}_{10}(0, \sigma^2 I) \tag{30}$$

$$x \mid \theta \sim \mathcal{N}_{10}(\theta, \sigma^2 I) \tag{31}$$

where $\sigma^2 = 0.1$, I is a unit matrix of appropriate dimensionality, and both $\theta \in \mathbb{R}^{10}$ and $x \in \mathbb{R}^{10}$ are ten-dimensional random variables. The linear Gaussian follows the representation in Lueckmann et al. (2021).

C.2 Gaussian mixture 1

The Gaussian mixture 1 has the following generative process:

$$\theta \sim \mathcal{U}_2(-10, 10)$$

$$x \mid \theta \sim \frac{1}{2}\mathcal{N}_2(\theta, I) + \frac{1}{2}\mathcal{N}_2(\theta, \sigma^2 I)$$

where $\sigma^2 = 0.01$, I is a unit matrix, and both $\theta \in \mathbb{R}^2$ and $x \in \mathbb{R}^2$ are two-dimensional random variables. The Gaussian mixture follows the representation in Lueckmann et al. (2021).

C.3 GAUSSIAN MIXTURE 2

The Gaussian mixture 2 has been proposed in Vargas et al. (2024). It is a 3-component mixture that we adapted for SBI. It has the following generative process:

$$\theta \sim \mathcal{N}_6\left([-1,-1,0,0,1,1]^T, I\right)$$
$$x \mid \theta \sim \frac{1}{3}\mathcal{N}_2\left(\theta_{1,2}, \begin{bmatrix}0.7 & 0\\ 0 & 0.05\end{bmatrix}\right) + \frac{1}{3}\mathcal{N}_2\left(\theta_{3,4}, \begin{bmatrix}0.7 & 0\\ 0 & 0.05\end{bmatrix}\right) + \frac{1}{3}\mathcal{N}_2\left(\theta_{5,6}, \begin{bmatrix}1 & 0.95\\ 0.95 & 1\end{bmatrix}\right)$$

where I is a unit matrix, and $\theta \in \mathbb{R}^6$ and $x \in \mathbb{R}^2$ are six- and two-dimensional random variables, respectively.

C.4 NESTED MODEL

We propose a nested model for benchmarks in SBI. The model has the following generative process:

$$\begin{aligned}\gamma &\sim \mathcal{N}(0,1)\\ \beta_k \mid \gamma &\sim \mathcal{N}(\gamma,1), \quad k=1,2,3\\ x_k \mid \beta_k &\sim \mathcal{N}(\beta_k, 0.1^2), \quad k=1,2,3\end{aligned}$$

where $\theta = (\gamma, \beta_1, \beta_2, \beta_3)$ is a four-dimensional random variable for which the posterior should be inferred, and $x = (x_1, x_2, x_3) \in \mathbb{R}^3$.

C.5 HYPERBOLOID

The hyperboloid model (Forbes et al., 2022) is a 2-component mixture of Student's t-distributions of the form

$$\theta \sim \mathcal{U}_2(-2,2) \tag{32}$$
$$x \mid \theta \sim \frac{1}{2}t_{10}(\nu, F(\theta; a_1, a_2)\mathbb{I}, \sigma^2 I) + \frac{1}{2}t_{10}(\nu, F(\theta; b_1, b_2)\mathbb{I}, \sigma^2 I) \tag{33}$$

which are parameterized by degrees of freedom ν, mean

$$F(\theta; m_1, m_2) = \big|\|\theta - m_1\|_2 - \|\theta - m_2\|_2\big|,$$

and scale matrix $\sigma^2 I$. $\mathbb{I}$ is a ten-dimensional vector of ones. We follow Forbes et al. (2022) and set $a_1 = [-0.5, 0.0]^T$, $a_2 = [0.5, 0.0]^T$, $b_1 = [0.0, -0.5]^T$, $b_2 = [0.0, 0.5]^T$, $\nu = 3$ and $\sigma^2 = 0.01$ for our experiments.

C.6 MIXTURE MODEL WITH DISTRACTORS

Similarly to the SLCP task (Lueckmann et al., 2021), we evaluate a benchmark task that uses distractor dimensions appending the data with dimensions that are not informative of the parameters. The generative process of the model is taken from Albert et al. (2025) as follows:

$$\begin{aligned}\theta &\sim \mathcal{U}(-10,10)\\ x_j \mid \theta &\sim \alpha\mathcal{N}(\theta,1) + (1-\alpha)\mathcal{N}(-\theta,\sigma^2), \quad j=1,2\\ x_j &\sim \mathcal{N}(0,1), \quad j=3,\dots,10\end{aligned}$$

where we set $\alpha = \sigma = 0.3$. In addition to the two informative data dimensions x_1, x_2, the model adds 8 samples from a standard Gaussian that do not carry information of the parameters. If both $x_{1,obs}$ and $x_{2,obs}$ are from the same mode (for one of the observations, we set $x_{1,obs} = x_{2,obs} = 5$), then the posterior for θ is bimodal with very uneven mass distribution.

C.7 TREE

The tree model is a recently introduced SBI benchmark model (Gloeckler et al., 2024). It has the following generative process:

$$\begin{aligned}
\theta_1 &\sim \mathcal{N}(0, 1) \\
\theta_2 \mid \theta_1 &\sim \mathcal{N}(\theta_1, 1) \\
\theta_3 \mid \theta_1 &\sim \mathcal{N}(\theta_1, 1) \\
x_1 \mid \theta_2 &\sim \mathcal{N}\left(\sin(\theta_2)^2, 0.2^2\right) \\
x_2 \mid \theta_2 &\sim \mathcal{N}\left(0.1\theta_2^2, 0.2^2\right) \\
x_3 \mid \theta_3 &\sim \mathcal{N}\left(0.1\theta_3^2, 0.6^2\right) \\
x_4 \mid \theta_3 &\sim \mathcal{N}\left(\cos(\theta_3)^2, 0.1^2\right)
\end{aligned}$$

C.8 TWO MOONS

Two moons is a common benchmark task in the SBI literature (Greenberg et al., 2019). Its generative process is defined as:

$$\begin{aligned}
\theta &\sim \mathcal{U}_2(-1, 1) \\
\alpha &\sim \mathcal{U}(-\pi/2, \pi/2) \\
r &\sim \mathcal{N}(0.1, 0.01^2) \\
x \mid \theta &= \begin{pmatrix} r\cos\alpha + 0.25 \\ r\sin\alpha \end{pmatrix} + \begin{pmatrix} -|\theta_1 + \theta_2|/\sqrt{2} \\ (-\theta_1 + \theta_2)/\sqrt{2} \end{pmatrix},
\end{aligned}$$

where we treat α and r as nuisance parameters.

C.9 SIR MODEL

The SIR model (Lueckmann et al., 2021) describes the spread of an infectious disease in a population of $N_{\text{pop}} = 10^6$ individuals, divided into susceptible (S), infected (I), and recovered (R) compartments, with dynamics

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\theta_0 \frac{SI}{N_{\text{pop}}}, \qquad \frac{\mathrm{d}I}{\mathrm{d}t} = \theta_0 \frac{SI}{N_{\text{pop}}} - \theta_1 I, \qquad \frac{\mathrm{d}R}{\mathrm{d}t} = \theta_1 I.$$

We infer the contact rate $\theta_0 \sim \log\mathcal{N}(\log 0.4, 0.5^2)$ and the recovery rate $\theta_1 \sim \log\mathcal{N}(\log 1/8, 0.2^2)$. Starting from a single infected individual, we solve the ODE on $t \in [0, 160]$ and, at each of 160 equally spaced time points, observe the number of infected individuals in a sample of 1 000 via a binomial distribution. Instead of the full time series, all methods are trained on five summary statistics: the peak number of infections, the time of the peak, and the sum, mean, and variance of the infection counts. These summaries are informative but not sufficient for the parameters. The reference posterior is obtained by slice sampling on the full time series.

C.10 HIERARCHICAL MODEL

The hierarchical model draws a two-dimensional top-level location γ, three lower-level locations β_k centered at γ, and an observation scale σ, and generates each observation x_k from a Gaussian centered at β_k with scale σ. In total, it has nine parameters and six-dimensional observations.

$$\begin{aligned}
\sigma &\sim \text{HalfNormal}(1) \\
\gamma &\sim \mathcal{N}_2(0, I) \\
\beta_k \mid \gamma &\sim \mathcal{N}_2(\gamma, I), \quad k = 1, 2, 3 \\
x \mid \beta, \sigma &\sim \mathcal{N}_6\big([\beta_1^\top, \beta_2^\top, \beta_3^\top]^\top, \sigma^2 I\big)
\end{aligned}$$

D EXPERIMENTAL DETAILS

In the following, we describe the experimental details for each benchmark task and inferential algorithm. All source code to reproduce our results can be found in the supplemental material and on GitHub (link published after review period).

D.1 REFERENCE POSTERIOR DISTRIBUTIONS

We use the Python package `sbijax` (Dirmeier et al., 2024) to draw reference, i.e., "ground-truth" posterior samples using Markov chain Monte Carlo. For all experimental models, we use a hit-and-run slice sampler with a step size of 1 and at most 5 doubling steps, and run 10 independent chains. For each chain, we discard the first 5 000 iterations as warmup and then keep every second of the following 10 000 iterations to reduce autocorrelation, yielding 5 000 samples per chain. Hence, the reference posterior sample consists of 50 000 particles. We monitored convergence using the typical MCMC diagnostic tools, i.e., the rank-normalized split $\widehat{R}$ (Vehtari et al., 2021) and the effective sample size as implemented in the Python package ArviZ (Kumar et al., 2019).

D.2 COMPARING REFERENCE POSTERIORS TO INFERRED POSTERIORS

We compare the inferred posterior distributions to the reference posterior distributions using the classifier two-sample test statistic (C2ST; Lopez-Paz & Oquab, 2017) and the recently introduced H-Min divergence (Zhao et al., 2022), which we, like Dirmeier et al. (2025), found to be easier to tune than C2ST and other divergences. To compute the metrics, we first subsample both reference and inferred posteriors to 10 000 samples without replacement, and then compute both metrics on these subsamples.

D.3 NEURAL NETWORK ARCHITECTURES

For AIO, FMPE and PSE, we use implementations from the Python package `sbijax` (Dirmeier et al., 2024). The neural network architectures of CPE and all baselines are detailed below.

CPE CPE is structured as a block neural autoregressive flow. We first embed the time t using a sinusoidal embedding of dimension 64 and then project the result through a two-layer MLP with 64 nodes each. We project the data x through a two-layer MLP with 128 nodes each. Both embedding networks use SiLU activations. We then concatenate the embeddings of time t and data x before projecting the resulting vector and the parameter values θ through the transformations $f = (f^{(1)}, \ldots, f^{(K)})$. In our experiments, we set $K = 4$ and use a block size of 64, i.e.,

$$
\begin{aligned}
d_{\text{in}}^{(1)} &= 1, & d_{\text{out}}^{(1)} &= 64, \\
d_{\text{in}}^{(2)} &= 64, & d_{\text{out}}^{(2)} &= 64, \\
d_{\text{in}}^{(3)} &= 64, & d_{\text{out}}^{(3)} &= 64, \\
d_{\text{in}}^{(4)} &= 64, & d_{\text{out}}^{(4)} &= 1.
\end{aligned}
$$

Each $f^{(k)}$ is designed as described in Equation (12), where we use the identity function for g and apply $\text{act} := \tanh$ after every layer except the last. After each intermediate layer $f^{(2)}, \ldots, f^{(K-1)}$, we additionally reverse the order of the hidden units belonging to each parameter, such that all dimensions of a multi-dimensional parameter can depend on each other.We condition on data and time by adding the concatenation of the data and time embeddings after the first projection $f^{(1)}$, i.e.,

$$
\begin{aligned}
\theta' &= f^{(1)}(\theta) \\
\theta'' &= \theta' + \texttt{Linear}(c)
\end{aligned}
$$

where we denote with `Linear` a linear neural network layer (with bias) and with c the concatenation of time and data embeddings. We denote with λ_t the composition of the embeddings and the projection through f. Its output is combined with the input through the convex combination in Equation (8).

BNPE BNPE uses the same architecture as CPE, but without causal masks, such that the weight matrices of all transformations $f^{(1)}, \ldots, f^{(K)}$ are dense.

FMPE FMPE uses a two-layer MLP with 128 nodes each to embed parameter values θ and a two-layer MLP with 128 nodes each to embed data values x. We first project time values t through a sinusoidal embedding of dimension 64, before embedding the result using a two-layer MLP with 64 nodes each. We then concatenate the embedded parameter, data and time values and project them through an MLP with two hidden layers of 256 nodes each. FMPE uses the forward process and vector field as described in Lipman et al. (2023), and we set $\sigma_{\min} = 0.001$. All activation functions are GELUs (Hendrycks & Gimpel, 2016).

AIO Following the original publication (Gloeckler et al., 2024), AIO uses a transformer backbone as a score model with a token dimension of 50, an attention size of 10, and a widening factor of 3. It uses six transformer layers and four attention heads for all experiments. We use embedding dimensions of 32, 10 and 8 for values, ids and conditioning variables, respectively, and embed time using a sinusoidal embedding followed by a two-layer MLP with 64 nodes each. We use a dropout rate of 0.0. All activation functions are GELUs (Hendrycks & Gimpel, 2016). AIO uses the variance-preserving SDE of Song et al. (2021) with $\beta_{\min} = 0.1$ and $\beta_{\max} = 10$. For AIO, we use the undirected graph of the model as attention mask. A directed graph would require recomputing the attention mask whenever the conditioning changes, e.g., at every training batch when the condition mask is sampled randomly (Gloeckler et al., 2024, Section 3.2), which makes it impractical to use with AIO. For AIO (no mask), we use no attention mask.

For the SIR model, we use AIO without an attention mask: since every summary statistic is computed from the same simulated time series, which depends on both parameters, the graph of the model is fully connected and the mask would not remove any dependencies.

PSE To be able to compare PSE fairly to FMPE and AIO, we use the same neural network architecture as for FMPE (since the mathematical frameworks are related) and the same variance-preserving SDE as for AIO with $\beta_{\min} = 0.1$ and $\beta_{\max} = 10$. All activation functions are GELUs (Hendrycks & Gimpel, 2016).

D.4 Training

All methods, i.e., CPE, BNPE, and all baselines, are trained in exactly the same way. Given a simulation budget of N pairs (x_n, θ_n), we hold out 10% of the pairs as a validation set and train on the remaining 90%. We use the Adam optimizer (Kingma & Ba, 2015) with learning rate 10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$, and a batch size of 200. After each epoch, i.e., one pass over the training set, we compute the loss on the validation set. An epoch counts as an improvement if its validation loss is lower than that of the last improving epoch by more than $\delta = 0.01$. Training stops as soon as more than 20 consecutive epochs pass without improvement, or after a maximum of 2 000 epochs, whichever occurs first. We then use the parameters that achieved the lowest validation loss during training. We use `Optax` for gradient-based optimization (Babuschkin et al., 2020).

D.5 Sampling

For each method, we draw 20 000 posterior samples. CPE-RK and all baselines integrate the ODE with the adaptive Runge–Kutta 5(4) solver of `SciPy` (Virtanen et al., 2020) with relative and absolute tolerances of 10^{-5}. For the baselines, we use the implementations of `sbijax` (Dirmeier et al., 2024). CPE integrates the ODE from $t = 10^{-3}$ to $t = 1$. For the CPE variant CPE-Euler, we use a custom Euler solver with $T = 20$ steps and step size $\mathrm{d}t = 1/20$.

D.6 Computational resources

All MCMC sampling, model training and model evaluations were run on AMD EPYC™ 7742 processors with 64 cores and 256 GB of RAM. Runtimes of all methods were between 2 and 6 h. In total, we conducted 950 experiments (5 seeds $\times$ 10 experimental models $\times$ (6 methods $\times$ 3 simulation budgets + 1 MCMC reference)), yielding a total runtime of approximately $950 \times 4 \times 64 = 243\,200$

CPU hours. We use the workflow manager Snakemake (Köster & Rahmann, 2012) to run all experiments automatically on a Slurm cluster.

E ADDITIONAL RESULTS

E.1 BENCHMARK TASKS

In this section, we report additional results for the benchmark models. Figure 9 shows the C2ST of all methods for a simulation budget of $N = 10\,000$, complementing the H-min results in Figure 5 of the main text. Figure 10 shows the posterior approximations of all methods for the linear Gaussian model, on which the two metrics disagree.

Gaussian The H-min and C2ST results are generally consistent across benchmark models, except for the linear Gaussian model (Gaussian in Figure 5 of the main text), where AIO performs considerably worse under H-min than under C2ST (Figure 9). Like Dirmeier et al. (2025), we find that the H-min divergence, although harder to compare across benchmark models, is the more reliable metric in this case, since it is more sensitive to differences in the geometry of the approximate and reference posteriors. Indeed, the posterior plots in Figure 10 show that the AIO posterior is degenerate even though it achieves low C2ST values.

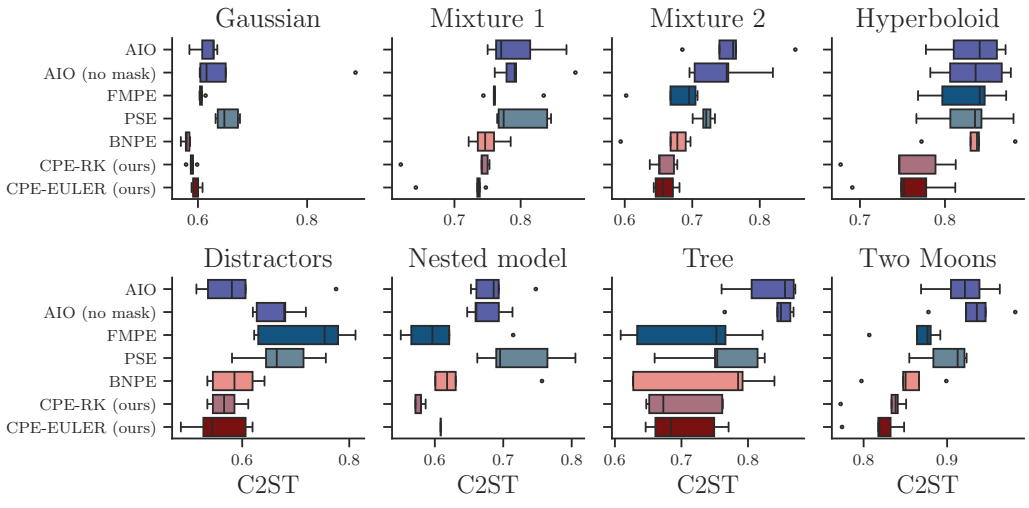
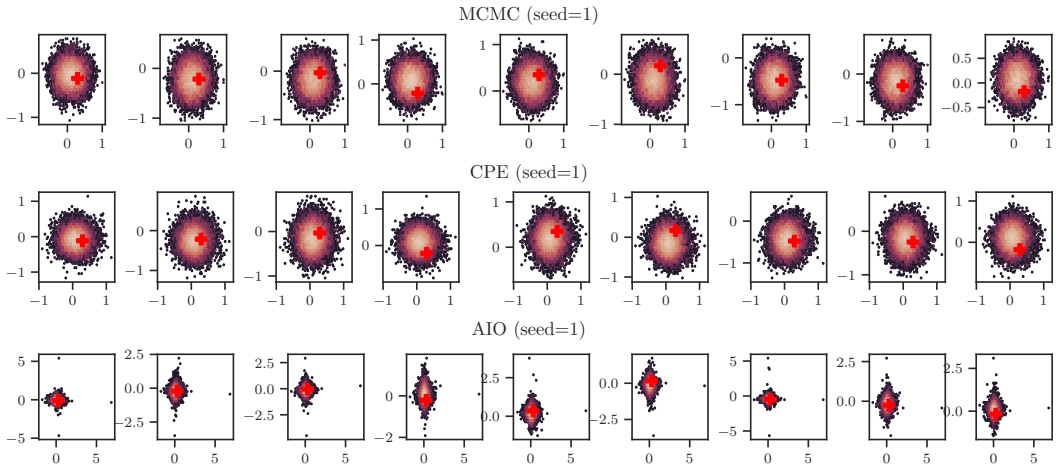
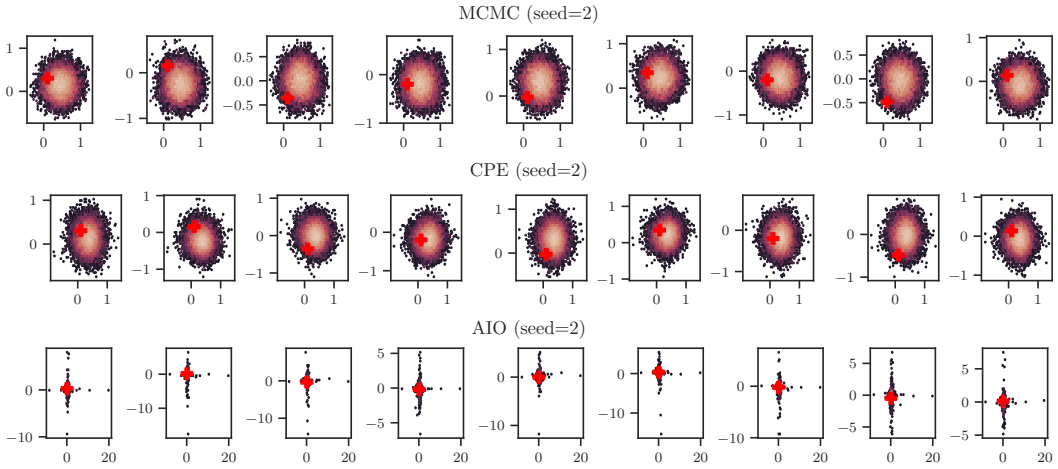


Figure 9: C2ST of CPE and the baselines on the eight benchmark models for a simulation budget of $N = 10\,000$. Values closer to 0.5 are better. CPE-RK and CPE-Euler denote CPE with an adaptive Runge–Kutta 5(4) solver and a 20-step Euler solver, respectively.



(a) Seed 1

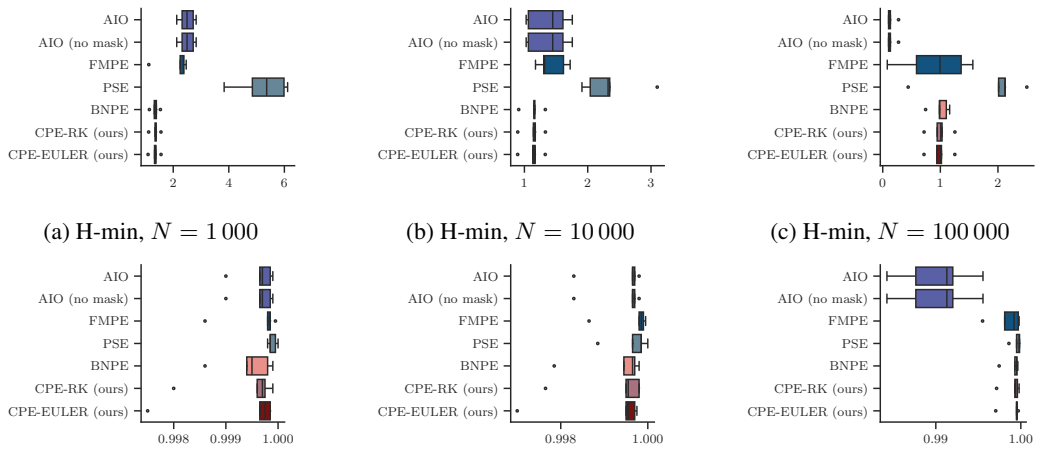


(b) Seed 2

Figure 10: Posterior distribtuions of CPE and AIO for the linear Gaussian model. The AIO posterior is degenerate for all five seeds (only two out of five seeds shown), although it achieves low C2ST values (Figure 9).

E.2 SIR MODEL

We report the H-min and C2ST of all methods on the SIR model for all three simulation budgets (Figure 11), complementing the H-min results in the main text. In addition, we show the posterior approximations of all methods (Figure 12) and their posterior predictive distributions, obtained by simulating data from posterior samples (Figure 13). Since the summary statistics of the SIR model are informative but not sufficient for the parameters, the posterior predictive distributions indicate how closely each method recovers the observed data.



(a) H-min, $N = 1\,000$ (b) H-min, $N = 10\,000$ (c) H-min, $N = 100\,000$

(d) C2ST, $N = 1\,000$ (e) C2ST, $N = 10\,000$ (f) C2ST, $N = 100\,000$

Figure 11: H-min (top row) and C2ST (bottom row) of all methods on the SIR model for simulation budgets $N \in \{1\,000, 10\,000, 100\,000\}$. Lower H-min values and C2ST values closer to 0.5 are better.

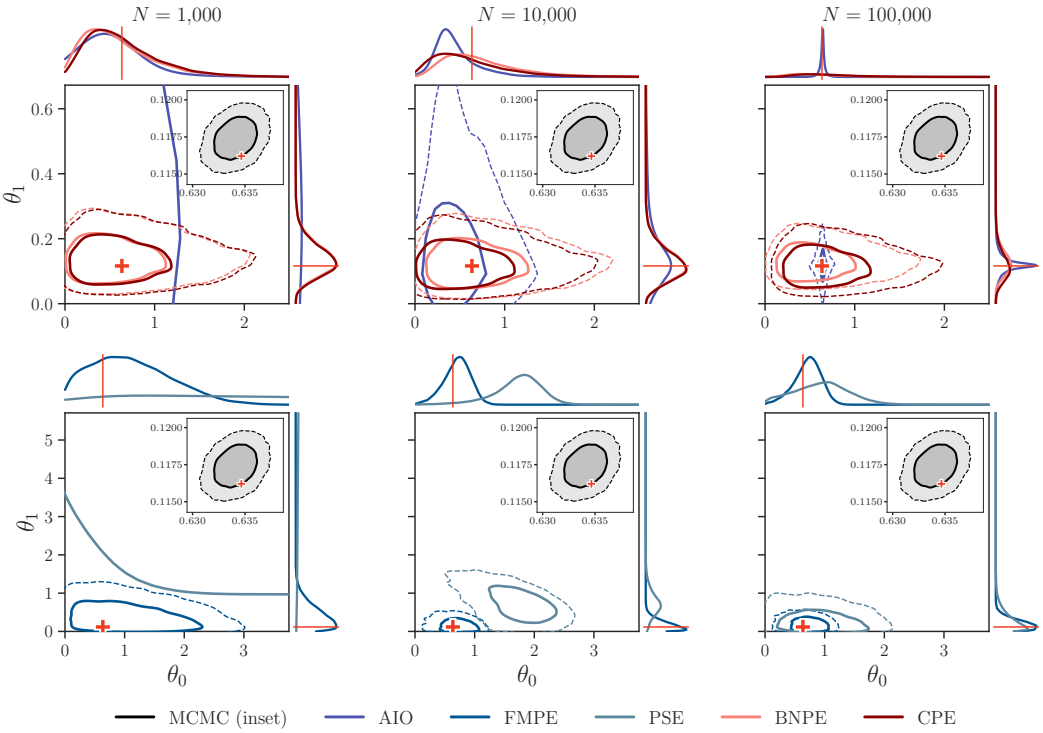


Figure 12: Posterior distributions of all methods for the SIR model, where θ_0 denotes the contact rate and θ_1 the recovery rate.

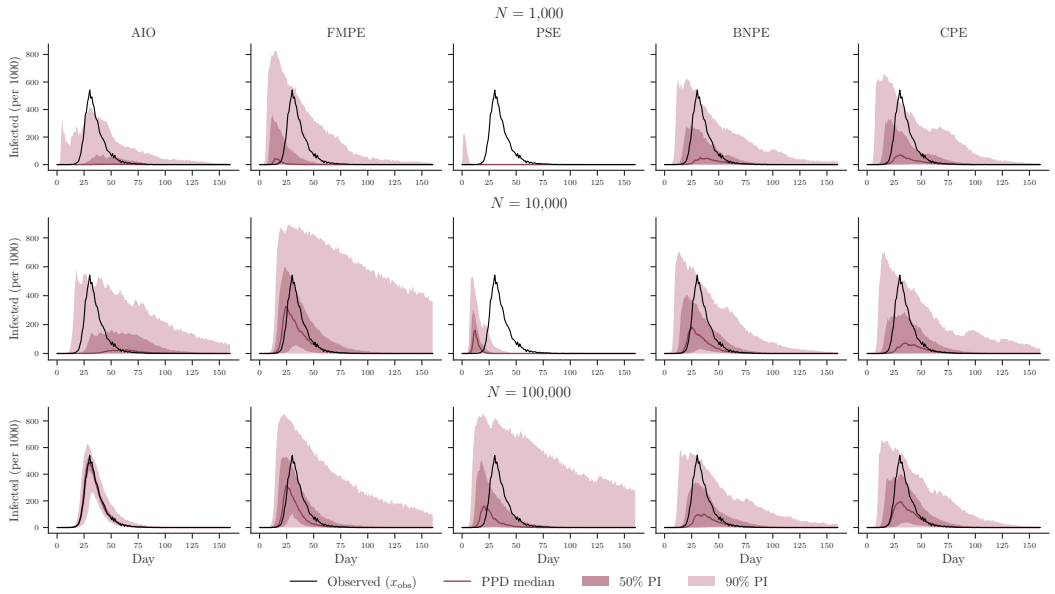
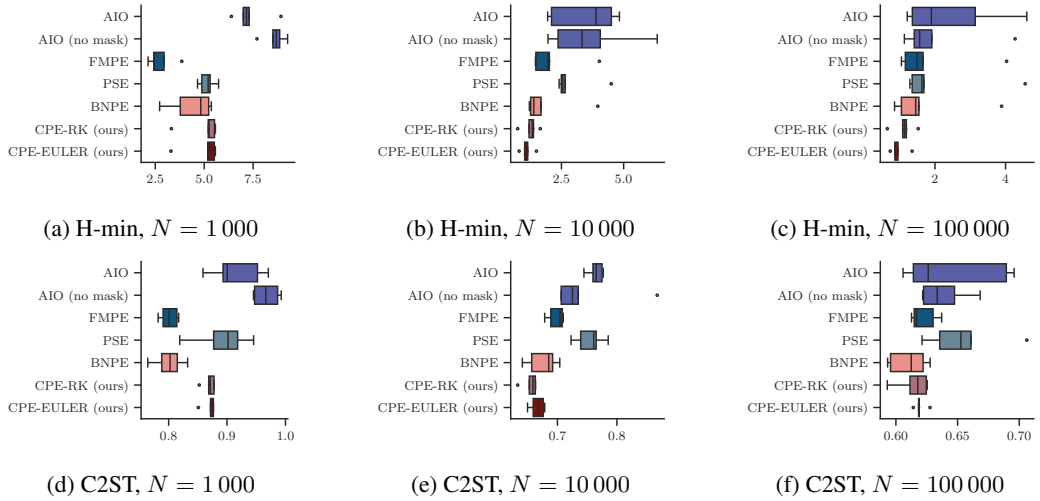


Figure 13: Posterior predictive distributions of all methods for the SIR model at a simulation budget of $N = 100\,000$. The posterior predictive distributions of AIO closely reproduce the observed data, whereas those of the other methods have considerably higher variance, as expected for summary statistics that are not sufficient for the parameters.

E.3 HIERARCHICAL MODEL

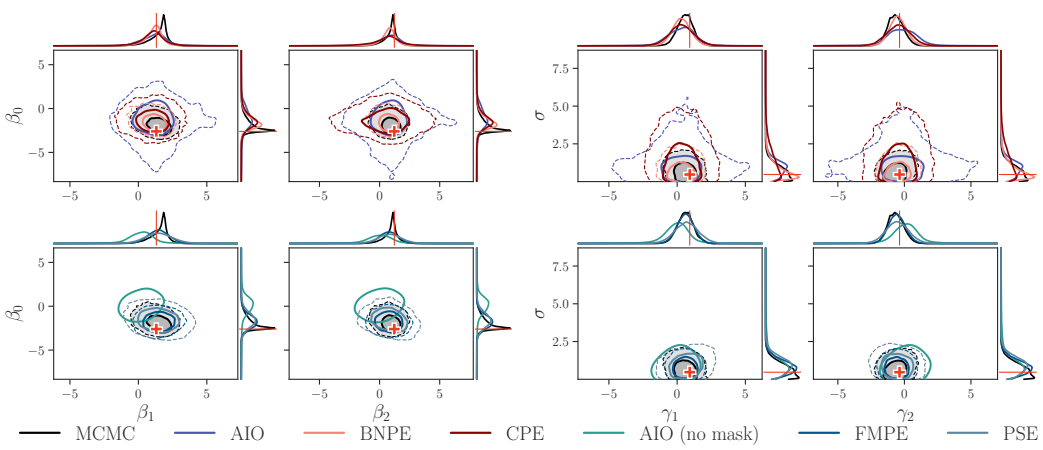
We report the H-min and C2ST of all methods on the hierarchical model for all three simulation budgets (Figure 14), complementing the H-min results in the main text. In addition, we show the posterior approximations of all methods for each simulation budget (Figure 15) and their posterior predictive distributions, obtained by simulating data from posterior samples (Figure 16).



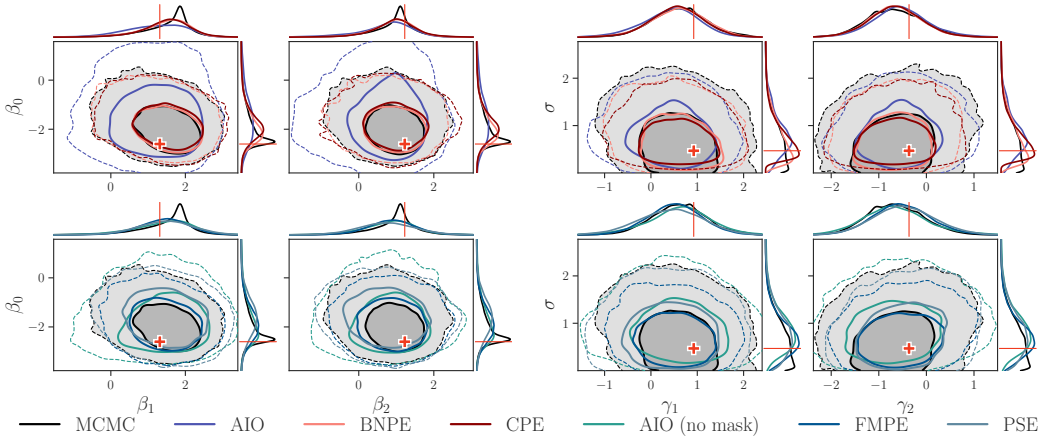
(a) H-min, $N = 1\,000$ (b) H-min, $N = 10\,000$ (c) H-min, $N = 100\,000$

(d) C2ST, $N = 1\,000$ (e) C2ST, $N = 10\,000$ (f) C2ST, $N = 100\,000$

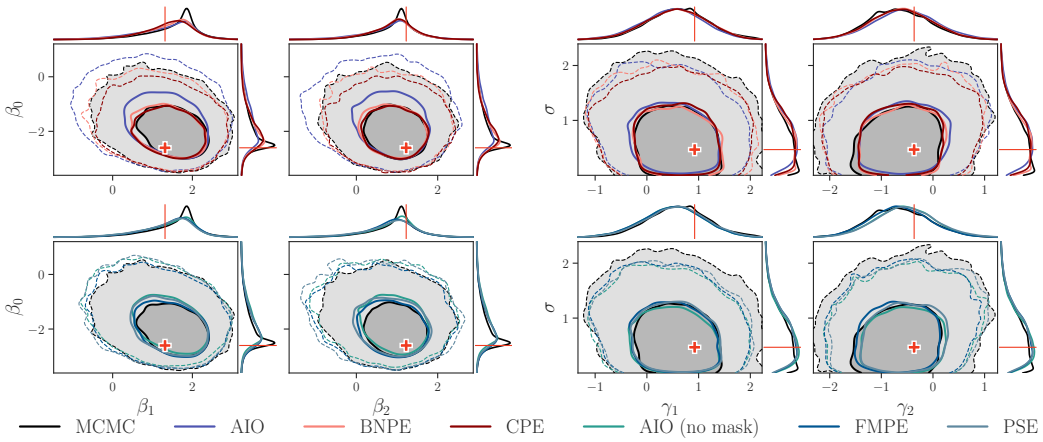
Figure 14: H-min (top row) and C2ST (bottom row) of all methods on the hierarchical model for simulation budgets $N \in \{1\,000, 10\,000, 100\,000\}$. Lower H-min values and C2ST values closer to 0.5 are better.



(a) $N = 1\,000$



(b) $N = 10\,000$



(c) $N = 100\,000$

Figure 15: Posterior distributions of all methods for the hierarchical model for simulation budgets $N \in \{1\,000, 10\,000, 100\,000\}$, showing the lower-level locations β, the top-level location γ and the observation scale σ.

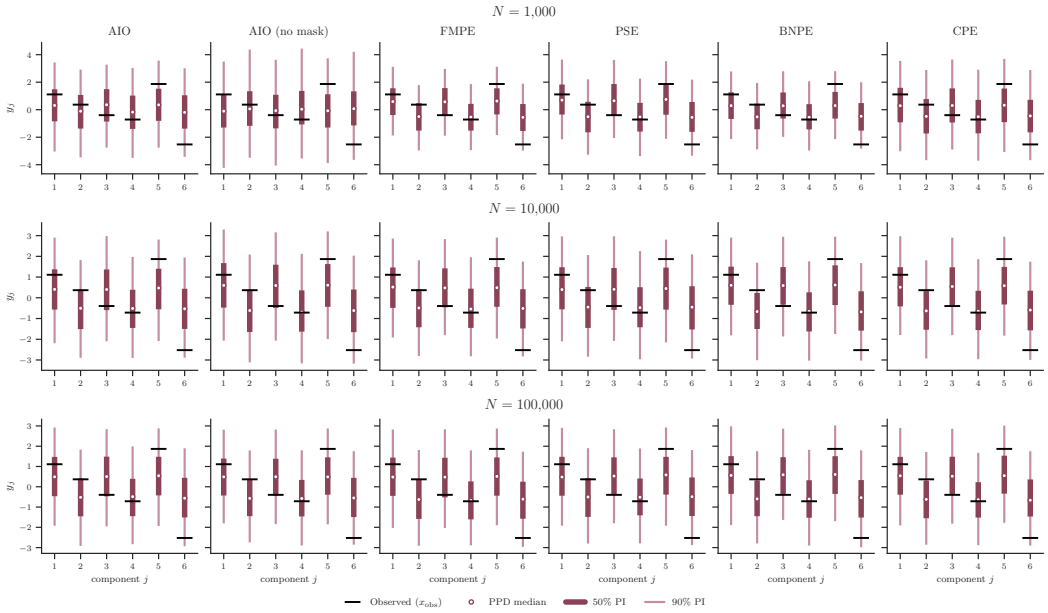


Figure 16: Posterior predictive distributions of all methods for the hierarchical model.