Tokenised Flow Matching for Hierarchical Simulation Based Inference

Giovanni Charles[1,2], **Seth Flaxman**[2], **Oliver Watson**[1], **Elizaveta Semenova**[1]
[1]Imperial College London, [2]University of Oxford
`{gc1610,o.watson15,e.semenova}@ic.ac.uk`
`{giovanni.charles,seth.flaxman}@cs.ox.ac.uk`

Abstract

Large simulation costs are a persistent challenge in Simulation Based Inference (SBI). Thus we strive for methods which are more sample efficient, requiring fewer simulations to achieve accurate parameter estimates. Across many scientific domains, SBI can be made more sample efficient by exploiting hierarchical structure. Rather than directly learning the full posterior, we learn single-site likelihoods, or posterior factors, that can be combined into a full hierarchical posterior. Current approaches assume conditional independence of local posteriors given the global parameters, which is often not suitable, and require training separate estimators for each level of the hierarchy which adds complexity to training. We present a tokenised flow matching estimator for posterior estimation (TFMPE), along with a sample efficient algorithm (Bottom-up sampling) for hierarchical parameter inference which makes no structural assumptions on the dependence of local and global parameters. We find that our method exhibits stable inference with improved sample efficiency compared to non-hierarchical methods for hierarchical inference tasks. We also examine posterior estimates for an infectious disease model and find that they are as reliable as MCMC approaches despite reduced computational demands.

1 Introduction

In a hierarchical SBI, the number of parameters to estimate and the number of simulations required for inference increase dramatically. With hierarchical models, a simulator is applied to many local settings and is parameterised with global parameters, which govern dynamics which are common to all settings, and local parameters, which condition local dynamics. For example, fitting a disease transmission model to data from multiple countries requires global parameters for pathogenic or immunological properties and local parameters for region-specific contact rates, seasonal effects and intervention policies. However, the computational cost of dataset generation and the complexity of posterior estimation grows with the size of the parameters space.

Notation. In Simulation Based Inference, we we would like to estimate a posterior distribution over some parameters θ given observational data y: $p(\theta|y) \propto p(y|\theta)p(\theta)$. There is no closed form solution for $p(y|\theta)$, however we do have a simulator at our disposal a which can generate y given θ.

In the hierarchical case, we consider a two-level hierarchical model where θ is partitioned into global parameters $\theta_g \in \mathbb{R}^{d_g}$ and local parameters $\theta_l = (\theta_{l[1]}, \dots, \theta_{l[n_s]})$ where each $\theta_{l[s]} \in \mathbb{R}^{d_l}$ parameterises a site $s \in \{1, \dots, n_s\}$. The global parameters act as hyperpriors for the local parameters, giving the prior distribution: $p(\theta_g, \theta_l) = p(\theta_g) \prod_{s=1}^{n_s} p(\theta_{l[s]}|\theta_g)$.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Frontiers in Probabilistic Inference: Sampling Meets Learning.

The simulator generates observations $y_{[s]} \in \mathbb{R}^{n_e \times d_y}$ for each site $s \in \{1, \dots, n_s\}$. We denote the complete set of observations across all sites as $y = (y_{[1]}, \dots, y_{[n_s]})$. The complete hierarchical posterior takes the form: $p(\theta_l, \theta_g|y) \propto p(\theta_g) \prod_{s=1}^{n_s} p(\theta_{l[s]}|\theta_g)p(y_{[s]}|\theta_{l[s]})$. Note that this is the same hierarchical posterior proposed by Gelman et al. [2021].

Factorised Posteriors. In cases where observational data can be assumed i.i.d, the posterior can be factorised as, $p(\theta|y) \propto p(\theta)^{1-n} \prod_s p(\theta|y_{[s]})$, by applying the Bayes rule twice. Many posterior estimation workflows have exploited this factorisation to train estimators for arbitrarily sized sets of observational data using using datasets containing pairs of parameters and single observations $(\theta, y_{[s]})$ [Radev et al., 2023, Boelts et al., 2024, Geffner et al., 2023]. Some of these methods rely on "summary networks", such as permutation invariant networks such as Deep Sets [Zaheer et al., 2017], in order to estimate posteriors from variable sized observations.

To account for local parameters in factorisations, recent methods have introduced separate estimators for global and local parameters to learn the posterior from datasets with fewer simulations [Arruda et al., 2025, Heinrich et al., 2024, Habermann et al., 2024, Rodrigues et al., 2021]. These methods extend the previous factorisation by separating global and local estimation through conditional independence, $p(\theta_g, \theta_l|y) \propto p(\theta_g, \theta_l)^{1-n}p(\theta_g|y) \prod_s p(\theta_{l[s]}|\theta_g, y_{[s]}$. In this approach, two estimators are required, a global estimator for $\hat{\theta_g} \sim p(\theta_g|y)$ trained from dataset wide observations, and a local estimator for $\hat{\theta_{l[s]}} \sim p(\theta_{l[s]}|\hat{\theta_g}, y[s])$.

Conditional Independence in hierarchical posteriors. However the above factorisation assumes that local posteriors are conditionally independent given θ_g, i.e. θ_l can be ignored when estimating θ_g, which is not always an appropriate treatment. Note that is distinct from the widely held assumption in hierarchical modelling that θ_l is conditionally independent from θ_g, which describes the data generating/forward process *not* the inverse problem of parameter inference after observing data. To illustrate this distinction with the epidemiological example from the introduction, while the pathogenic and intervention parameters for an infectious disease model can be safely assumed to have independently influential effects on the observed incidence data, the same assumption does not hold for posterior estimates. When we condition on observed incidence data, the posterior estimates for both parameters become dependent.

We alternatively propose to learn the posterior by factorising the likelihood function, similarly to Neural Likelihood Estimation, described in Section A.1. This allows us to estimate the posterior in its entirety and relax any structural assumptions with respect to the parameters and simply assume conditional independence of observations given parameters.

Contributions. In this work, we combine likelihood factorisation, tokenisation and flow matching to address the limitations of existing hierarchical SBI methods. Likelihood factorisation avoids the conditional independence assumptions of prior work on hierarchical SBI, while still reducing simulations for training. Tokenisation (Appendix A.3) mitigates error propagation from multiple estimators and enables amortised posterior estimation. Flow matching (Appendix A.4) provides a more stable training objective compared to score-based approaches which have been widely used in hierarchical SBI. Together, these contributions form TFMPE, a unified tokenised flow matching posterior estimator that learns factorised likelihoods and posteriors jointly.

2 Method

2.1 Tokenisation

We adopt a tokenisation approach that embeds arbitrarily-sized parameter and observation sets, enabling a flexible handling of variable-sized local parameters and function-valued observations. Each token encodes three components: (1) parameter or observation values embedded as fixed-length vectors in $\mathbb{R}^d$, (2) variable identifiers enumerated and embedded in $\mathbb{R}^i$, and (3) functional inputs encoded using Gaussian Fourier embeddings in $\mathbb{R}^f$ to map continuous function inputs (e.g., time or spatial coordinates) into bounded fixed-length vectors via sinusoidal features. Tokens are processed through an encoder-decoder transformer architecture where the encoder embeds context and the decoder generates vector fields for flow matching.

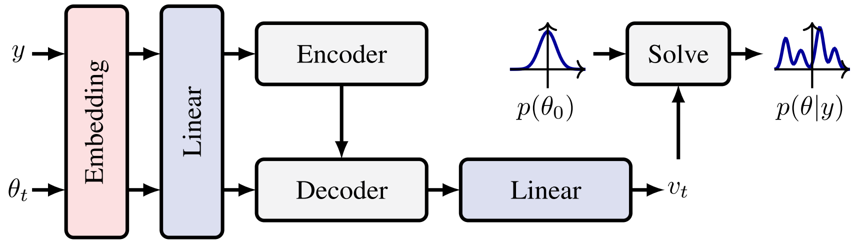


Figure 1: Method overview: Encoder-decoder transformer architecture for tokenised flow matching. The encoder processes context tokens y, while the decoder processes parameter tokens θ_t. Masking at attention blocks models conditional independence structure, and the final linear layer outputs vector field v_t optimised using the FMPE objective.

For hierarchical models, we embed the conditional independence of observations with local parameters, i.e. $p(y|\theta_l) = \prod_s p(y_{[s]}|\theta_{l[s]})$, using structured cross-attention masking. When training the factorised likelihood, the decoder masks attention from non-influential local parameter tokens. The full implementation details are provided in Section A.5.1 of the appendix.

2.2 Bottom-up Sampling for Hierarchical Models

We propose a two-stage sampling strategy, which we refer to as "Bottom-up sampling" which creates training data for amortised hierarchical posterior estimation. In stage one, we learn a factorised likelihood by generating training data $D_s = \{(\theta_g^{(i)}, \theta_{l[s]}^{(i)}, y_{[s]}^{(i)})\}_{i=1}^N$ from single-site simulations, optimising: $L_1(\phi) = \mathbb{E}_{\theta_g,\theta_{l[s]},y_{[s]}}\left[L_{\text{FMPE}}\left(\phi; y_{[s]}|\theta_g, \theta_{l[s]}\right)\right]$

In stage two, we use the estimated likelihood to generate full multi-site training data $D_m = \{(\theta_g^{(i)}, \theta_l^{(i)}, y^{(i)})\}_{i=1}^N$ where observations are sampled from $q_\phi(y|\theta_g^{(i)}, \theta_l^{(i)})$, and optimise: $L_2(\phi) = \mathbb{E}_{\theta_g,\theta_l,y}\left[L_{\text{FMPE}}\left(\phi; \theta_g, \theta_l|y\right)\right]$

An aggregate dataset $D = D_s \cup D_m$ is maintained during training, jointly optimising both objectives with appropriate loss functions applied per sample. This avoids the conditional independence assumptions of prior hierarchical approaches while enabling amortisation through a unified estimator. The strategy can optionally be extended as sequential refinement by resampling parameters from the latest posterior to generate more concentrated training data. Detailed data generation procedures and padding strategies for variable-sized parameters are described in Section A.5.2 of the appendix.

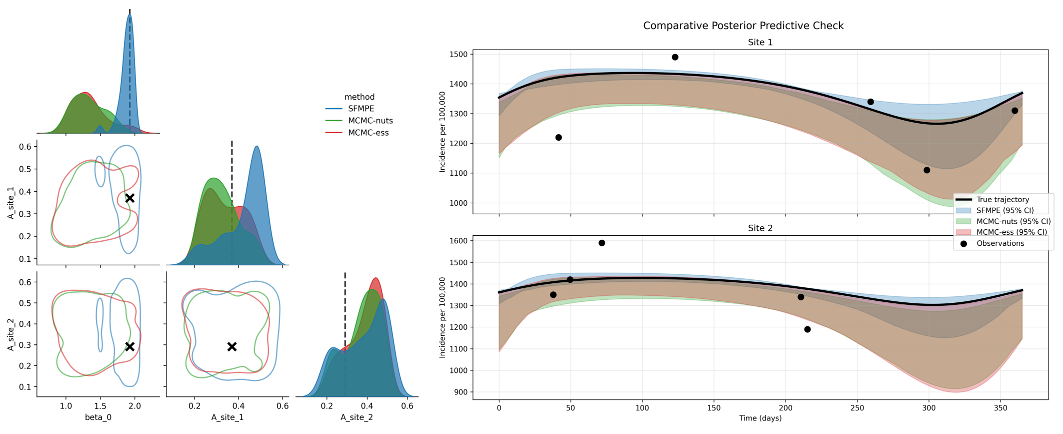
3 Results

We evaluated TFMPE with bottom up sampling on two toy hierarchical SBI tasks as well as a more realistic infectious disease model. Through the toy models we could investigate the scaling behaviour of TFMPE compared to FMPE with respect to different sampling budgets and the number of local parameters. For the infectious disease model, we provided a more qualitative assessment of posterior estimates compared to MCMC.

We are developing a benchmark for hierarchical inference tasks. A description of the development strategy and formulation of the models is listed in Section A.9.

3.1 Toy hierarchical inference tasks

We compared TFMPE to a baseline FMPE approach. In the baseline, parameters and observations were vectorised, concatenated, and provided to a multi-layer perceptron for vector field estimation. For Brownian motion, fixed observation times were used during training. FMPE was trained on multi-simulation datasets with n_s simulations per sample, consuming the budget more rapidly than tokenised TFMPE. The toy examples used for comparison are listed in Table 2 We found that tokenised FMPE consistently exhibited a preferable sample efficiency when increasing simulation budget, shown in Figure 4a and Figure 4c, and increasing number of local parameters, shown in



(a) (b)

Figure 2: SEIR posterior comparison: a Pairplot of posterior estimates for global parameter β_0 and local parameters A_1, A_2 comparing TFMPE (tokenised flow matching), NUTS, and ESS estimates. True parameter values shown as red crosshairs. b Posterior predictive check showing observed data (red crosshairs) against posterior predictive samples for each method across both sites.

Figure 4b and Figure 4d. The biggest advantage over traditional FMPE is realised when hierarchical inference is scaled over large amounts of local parameters, making FMPE quickly untenable for $n_s > 20$.

Table 1: LC2ST results for varying simulation budget (n_theta = 50, m = 5 observations).

n_simulations	Gaussian Task		Brownian Task	
	FMPE	TFMPE(prior)	FMPE	TFMPE(prior)
1,000	0.216 [0.181, 0.252]	**0.045 [-0.008, 0.098]**	0.215 [0.201, 0.228]	**0.003 [-0.002, 0.007]**
5,000	0.176 [0.146, 0.206]	**0.038 [0.015, 0.060]**	0.222 [0.213, 0.230]	**0.006 [0.001, 0.011]**
10,000	0.140 [0.072, 0.209]	**0.012 [0.001, 0.023]**	0.227 [0.218, 0.236]	**0.000 [-0.000, 0.001]**

3.2 Comparison to MCMC estimates for an Infectious Disease Model

MCMC is widely considered the gold standard for Bayesian parameter estimation and so we compared TFMPE posterior estimates to MCMC estimates for a simple Infectious Disease model. We used a Susceptible, Exposed, Infected, Recovered (SEIR) compartmental model as the simulator. Details of the SEIR model specification and parameterisation are provided in the appendix, see Section A.7. For the MCMC inference procedure, details on the sampling kernels and convergence criteria are provided in Section A.8. We found that TFMPE estimates were well calibrated in the posterior predictive and showed a strong update towards the latent true parameters, see Figure 2. The MCMC estimates were more dominated by the priors and showed more conservative updates towards the true parameters and posterior predictive.

4 Discussion

We plan to evaluate TFMPE with a more extensive benchmark, outlined in Section A.9, include alternative hierarchical estimators as baselines and apply our method to more complex and realistic inference problems to test sample efficiency in practice. We would also like to thoroughly investigate the effects of accumulated error in our two-step estimation and the stability of sequential inference for large and complex parameter and observation spaces.

References

Jonas Arruda, Vikas Pandey, Catherine Sherry, Margarida Barroso, Xavier Intes, Jan Hasenauer, and Stefan T. Radev. Compositional amortized inference for large-scale hierarchical bayesian models. 5 2025. doi: 10.48550/arXiv.2505.14429. URL `http://arxiv.org/abs/2505.14429`.

Jan Boelts, Michael Deistler, Manuel Gloeckler, Álvaro Tejero-Cantero, Jan-Matthis Lueckmann, Guy Moss, Peter Steinbach, Thomas Moreau, Fabio Muratore, Julia Linhart, Conor Durkan, Julius Vetter, Benjamin Kurt Miller, Maternus Herold, Abolfazl Ziaeemehr, Matthijs Pals, Theo Gruner, Sebastian Bischoff, Nastya Krouglova, Richard Gao, Janne K. Lappalainen, Bálint Mucsányi, Felix Pei, Auguste Schulz, Zinovia Stefanidi, Pedro Rodrigues, Cornelius Schröder, Faried Abu Zaid, Jonas Beck, Jaivardhan Kapoor, David S. Greenberg, Pedro J. Gonçalves, and Jakob H. Macke. sbi reloaded: a toolkit for simulation-based inference workflows. 11 2024. doi: 10.48550/arXiv.2411.17337. URL `https://arxiv.org/abs/2411.17337v1`.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv:1806.07366 [cs, stat]*, 12 2019. URL `http://arxiv.org/abs/1806.07366`.

Maximilian Dax, Jonas Wildberger, Simon Buchholz, Stephen R. Green, Jakob H. Macke, and Bernhard Schölkopf. Flow matching for scalable simulation-based inference. 10 2023. URL `http://arxiv.org/abs/2305.17161`.

Tomas Geffner, George Papamakarios, and Andriy Mnih. Compositional score modeling for simulation-based inference. 2023.

Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. Bayesian data analysis third edition (with errors fixed as of 15 february 2021). 2021.

Manuel Glöckler, Michael Deistler, and Jakob H. Macke. Variational methods for simulation-based inference. 10 2022. URL `http://arxiv.org/abs/2203.04176`.

Manuel Gloeckler, Michael Deistler, Christian Weilbach, Frank Wood, and Jakob H. Macke. All-in-one simulation-based inference. 7 2024. doi: 10.48550/arXiv.2404.09636. URL `http://arxiv.org/abs/2404.09636`.

David S. Greenberg, Marcel Nonnenmacher, and Jakob H. Macke. Automatic posterior transformation for likelihood-free inference. 5 2019. URL `http://arxiv.org/abs/1905.07488`.

Daniel Habermann, Paul-Christian Bürkner, Stefan T. Radev, Andreas Bulling, Lars Kühmichel, and Marvin Schmitt. Amortized bayesian multilevel models. 8 2024. URL `http://arxiv.org/abs/2408.13230`.

Lukas Heinrich, Siddharth Mishra-Sharma, Chris Pollard, and Philipp Windischhofer. Hierarchical neural simulation-based inference over event ensembles. 2 2024. doi: 10.48550/arXiv.2306.12584. URL `http://arxiv.org/abs/2306.12584`.

Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free mcmc with amortized approximate ratio estimators. 2020. ISSN 2640-3498. URL `https://proceedings.mlr.press/v119/hermans20a.html`.

Matthew D. Hoffman and Andrew Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. 11 2011. URL `http://arxiv.org/abs/1111.4246v1`.

Minas Karamanis and Florian Beutler. Ensemble slice sampling: Parallel, black-box and gradient-free inference for correlated & multimodal distributions. *Stat Comput 31, 61 (2021)*, 2 2020. doi: 10.1007/s11222-021-10038-2. URL `http://arxiv.org/abs/2002.06212v3`.

Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. 10 2022. URL `http://arxiv.org/abs/2210.02747v2`.

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 I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/addfa9b7e234254d26e9c7f2af1005cb-Paper.pdf`.

Jan-Matthis Lueckmann, Jan Boelts, David S Greenberg, Pedro J Gonçalves, and Jakob H Macke. Benchmarking simulation-based inference. 4 2021.

George Papamakarios and Iain Murray. Fast ϵ-free inference of simulation models with bayesian conditional density estimation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper_files/paper/2016/file/6aca97005c68f1206823815f66102863-Paper.pdf`.

George Papamakarios, David C. Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. 1 2019. URL `http://arxiv.org/abs/1805.07226`.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22:1–95, 12 2021. doi: 10.48550/arXiv.1912.02762. URL `http://arxiv.org/abs/1912.02762`.

Du Phan, Neeraj Pradhan, and Martin Jankowiak. Composable effects for flexible and accelerated probabilistic programming in numpyro. 12 2019. URL `http://arxiv.org/abs/1912.11554`.

Stefan T. Radev, Marvin Schmitt, Valentin Pratz, Umberto Picchini, Ullrich Köthe, and Paul-Christian Bürkner. Jana: Jointly amortized neural approximation of complex bayesian models. 6 2023. URL `http://arxiv.org/abs/2302.09125`.

Pedro LC Rodrigues, Thomas Moreau, Gilles Louppe, and Alexandre Gramfort. Hnpe: Leveraging global parameters for neural posterior estimation. 2 2021. URL `http://arxiv.org/abs/2102.06477`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 6 2017. doi: 10.48550/arXiv.1706.03762. URL `http://arxiv.org/abs/1706.03762`.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbhakhsh, Barnabás Poczós, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in Neural Information Processing Systems*, 3 2017. doi: 10.48550/arXiv.1703.06114. URL `http://arxiv.org/abs/1703.06114`.

A Appendix

A.1 Extended Background

Simulation Based Inference (SBI) methods are essential in the sciences for calibrating numerical simulators to real world observations. As simulators become more realistic, requiring complex parameterisation and computational resources, we must scale our inference methods to keep robust and reliable inference feasible. Markov Chain Monte Carlo (MCMC) methods, widely considered the gold standard for parameter inference, require (often intractable) closed form likelihoods, struggle to sample large parameter spaces, and are often slow due to the sequential nature of sampling.

Neural Density Estimation (NDE) methods are strong candidates for scalable SBI. In NDE, one trains neural networks to estimate parameters from a dataset of simulations, often generated in parallel to make use of modern computing resources, and observations without requiring a likelihood function. Many "amortised" estimators are able to generate parameter estimates for observations within wide observational data spaces, for example arbitrary singular observations, sets of independent observations, or function-valued observations without further training or sampling.

In a hierarchical setting, the number of parameters to estimate and the number of simulations required for inference increase dramatically. With hierarchical models, a simulator is applied to many local settings and its parameters can be divided into global parameters, which govern common dynamics, and local parameters, which condition local dynamics. For example, fitting a disease transmission model to data from multiple countries requires global parameters for pathogenic or immunological properties and local parameters for region-specific contact rates, seasonal effects and intervention policies. However, scaling inference with respect to local settings requires increasing the number of simulations and parameters during inference which adds to the computational cost of dataset generation and the complexity of density estimation.

There has been recent work on sample efficient hierarchical SBI [Arruda et al., 2025, Heinrich et al., 2024, Habermann et al., 2024, Rodrigues et al., 2021]. In general, the proposed factorisations of the posterior have assumed conditional independence of local parameters such that one can train a global estimator based on all observations, or a "dataset wide" estimator, and subsequently train a separate local estimator for local parameters given the global estimate. Commonly, permutation invariant architectures, such as Deep Sets [Zaheer et al., 2017] and Transformers [Vaswani et al., 2017] are used to allow for arbitrarily sized sets of observations and to reduce simulation costs during training through sub-set sampling. Arruda et al. [2025] in particular have demonstrated score based estimators which scale to an impressive 750,000 local parameters. However, the structural assumptions are restrictive for many real world hierarchical inference tasks.

In this work, we investigated the sample efficiency of likelihood factorisation (through the "Bottom-up" algorithm) and tokenised flow matching (through TFMPE) for hierarchical SBI. The Bottom-up algorithm estimates global and local parameters jointly, avoiding the structural assumptions of competing methods, without the need for several simulations per parameter draw. We demonstrate that TFMPE can serve as a unified estimator in hierarchical SBI, as it can estimate different conditionals under a joint distribution. We compiled a benchmark of hierarchical inference tasks, adapted from Lueckmann et al. [2021] to compare our contributions to existing hierarchical and non-hierarchical methods. We also applied method for inference to two realistic hierarchical SBI tasks, a global infectious disease model and a haemodynamics model to compare estimates to MCMC.

A.2 Non-hierarchical SBI

Neural Likelihood Estimation and Neural Likelihood Ratio Estimation (NLE/NRE) approaches posterior estimation by first estimating the likelihood $q_\phi(\theta) \approx p(y|\theta)$ [Papamakarios et al., 2019, Hermans et al., 2020, Greenberg et al., 2019] or the likelihood evidence ratio $q_\phi(\theta) \approx p(y|\theta)/p(y)$. The likelihood estimator trained on a dataset of (y, θ) pairs generated by a proposal distribution, $\tilde{p}$, i.e. $(y^{(i)}, \theta^{(i)}) \sim \tilde{p}(\theta)p(y|\theta)$. One can then infer the posterior through traditional means, for example MCMC [Papamakarios et al., 2019] or variational inference [Glöckler et al., 2022]. NLE is convenient for hierarchical settings since the likelihood is trivially factorised per site, i.e. $p(y|\theta_g, \theta_l) = \prod_{s=1}^{n_s} p(y_{[s]}|\theta_{l[s]})$, and can be trained on single simulations with few alterations to the workflow. However, the combination of likelihood estimation and traditional inference introduces multiple layers of approximation error to the inference workflow.

Neural Posterior Estimation (NPE), tackles posterior estimation directly and in an amortised fashion. In NPE, one trains a posterior estimator, $q_\phi(\theta|y^o) \approx p(\theta|y^o)$, for arbitrary observations y^o [Papamakarios and Murray, 2016, Lueckmann et al., 2017, Greenberg et al., 2019]. In hierarchical settings, the dataset $(y^{(i)}, \theta_g^{(i)}, \theta_l^{(i)}) \sim \tilde{p}(\theta)p(y|\theta)$ requires n_s simulations per sample and cannot be trivially decomposed. This problem worsens with large n_s because $\theta_{l[s]}$ depends on θ_g and $y_{[s]}$ depends on $\theta_{l[s]}$, restricting the diversity of training data and requiring even more simulations to adequately cover the parameter-observation space. To mitigate this, there has been significant research on factorising the posterior to benefit from efficient single simulation datasets without sacrificing amortisation.

A.3 Tokenised SBI

Recently, Gloeckler et al. [2024] (SimFormer) introduced a general abstraction for simulation based inference in which the structure of the model is embedded into tokens, which we refer to as tokenised SBI. Alongside the sampled value for each parameter, which is used by traditional neural estimators, a tokenised estimator uses tokens which include parameter identifiers, functional inputs, and dependence and conditioning information. This comes with several benefits: *(1)* parameters and observations could be treated symmetrically so that any conditional in the joint distribution can be estimated, *(2)* structured attention masks can be parameterised based on known dependencies, *(3)* functional parameters or observations could be modelled to flexibly handle missing and/or unstructured data.

Their proposed SimFormer used a score estimation approach for estimating conditionals. Score matching is a valuable alternative to the widespread normalising flow-based estimators, avoiding mode-seeking biases of the reverse KL objective and the computational demands of invertability and Jacobian computation [Papamakarios et al., 2021]. However, Flow Matching provides a more principled alternative, combining the architectural flexibility of score-based methods with improved training stability and efficiency.

A.4 Flow Matching for Posterior Estimation

Flow matching, a method for training continuous normalising flows, has shown promise for posterior estimation [Dax et al., 2023]. Lipman et al. [2022] provided a framework for defining vector fields, $u_t(\theta, \theta_1)$, which generate desired probability paths between a base and desired distribution. The vector field is used in the Conditional Flow Matching objective, a tractable objective for learning continuous time flows over continuous data. Dax et al. [2023] then adapted the objective to posterior estimation for parameters θ and observations y, where the goal is to learn a flow from a base distribution to the posterior, $p(\theta|y)$ using a learned vector field $v_t(.)$:

$$L_{\text{FMPE}}(\phi; \theta|y) = \mathbb{E}_{t,p(\theta),p(y|\theta),p_t(\theta|\theta_1)} ||v_t(\phi, \theta_t, y) - u_t(\theta_t|\theta_1)||^2 \tag{1}$$

There are several benefits to estimating the posterior this way. Learning a continuous flow places fewer restrictions on the neural architecture. The only constraint is that, for uniqueness, the vector field must be Lipshitz continuous in θ and continuous in t [Chen et al., 2019]. However, flow matching provides similar guarantees to discrete normalising flows for density estimation in the target space and bijection with the base distribution, through the continuity equation [Lipman et al., 2022]. Compared to score matching, flow matching can learn simpler paths, e.g. conditional optimal transport paths [Lipman et al., 2022], which lead to faster and more stable training.

A popular objective is to learn Gaussian probability paths, $p_t(\theta|\theta_1) = N(\mu_t(\theta_1), \sigma_t(\theta_1)^2 I)$. And the Optimal Transport map is learned through the vector field: $u_t(\theta_t|\theta_1) = \frac{\theta_1 - (1-\sigma_{\min})\theta_t}{1-(1-\sigma_{\min})t}$, see Lipman et al. [2022] for details.

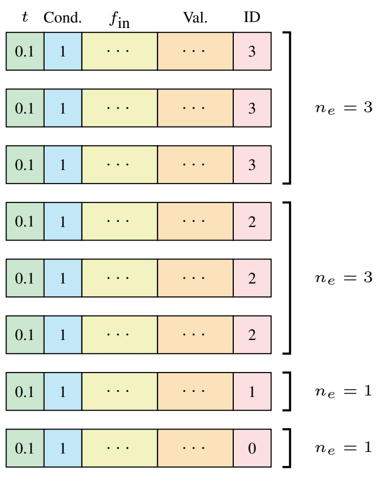
A.5 Additional method details

A.5.1 Tokenisation

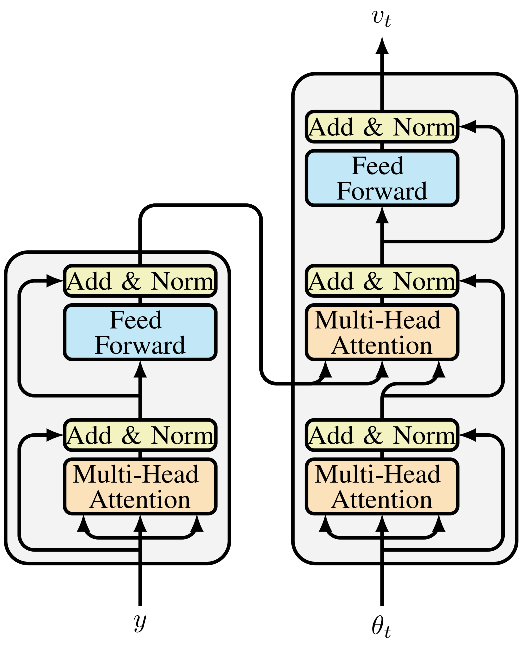
We defined a tokenisation scheme which embeds arbitrarily sized, functional or non-functional parameter and observation sets. This provides the flexibility for estimating posteriors or factorised likelihoods with different numbers of local parameters and embedding spatial or temporal correlates.

Let parameter-observation samples be represented by (θ_g, θ_l, y) where $\theta_l = (\theta_{l[1]}, \ldots, \theta_{l[n_s]})$ and $y = (y_{[1]}, \ldots, y_{[n_s]})$. Local parameters can represent either i.i.d. draws $\theta_{l[s]} \sim p(\theta_l)$ or draws from function-valued random variables $\theta_{l[s]} \sim p(\theta_l|h_{[s]})$ where $h_{[s]}$ are functional inputs to the local prior. Similarly, observations can be i.i.d., $y_{[s]} \sim p(y)$, or function-valued, $y_{[s]} \sim p(y|i_{[s]})$, where $i_{[s]}$ are functional inputs for observations. We will refer to vectorised function inputs for θ_l and y as h and i respectively. Independence can be specified at the token level using attention masking.

We embed tokens by encoding the values, variable name, and function inputs for each token, as is depicted in Figure 3a. All parameter and observation values are embedded as fixed-length vectors in $\mathbb{R}^d$. Vectors with fewer than d dimensions are zero-padded to match this size. Variable names are enumerated and then embedded into a continuous space in $\mathbb{R}^i$. Function inputs are encoded using Gaussian Fourier embeddings in $\mathbb{R}^f$. These map continuous function inputs (e.g., time or spatial coordinates) into bounded fixed-length vectors via sinusoidal features, similarly to SimFormer [Gloeckler et al., 2024]. Since tokens for parameters and observations are symmetric, each token is also given an identifier in $\{0, 1\}$ to mark it as a parameter or an observation.



(a)



(b)

Figure 3: Detailed method figures from method overview: a Tokenisation scheme for embedding parameters and observations with structural information. Each column represents a token with five components: ID (variable identifier), Val. (variable values in $\mathbb{R}^d$), f_{in} (functional inputs using Gaussian Fourier embeddings), Cond. (conditioning indicator), and t (flow time). Parameters or observations are grouped by brackets showing variable event sizes n_e. b Detailed encoder-decoder layers with attention mechanisms and masking for conditional independence structure.

We used an encoder-decoder transformer architecture, shown in Figure 3b, to estimate vector fields for flow matching. The encoder embeds the context, while the decoder generates a vector field for flow estimation.

For hierarchical models, we embedded the conditional independence of observations with local parameters, i.e $p(y|\theta_l) = \prod_s p(y_{[s]}|\theta_{l[s]})$, using cross attention masking. When training or sampling from the factorised likelihood, described in Section A.5.2, the decoder masked attention from encoded local parameter tokens which were not influential.

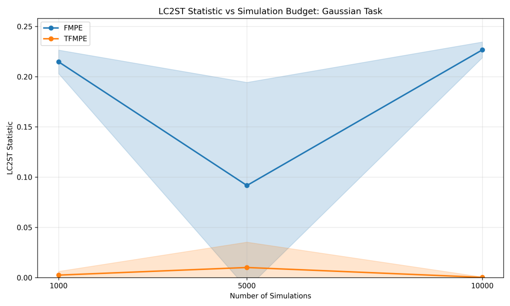
A.5.2 Bottom-up sampling for hierarchical models

The bottom up sampling strategy is designed to create training data for amortised hierarchical posterior estimation by first learning a factorised likelihood function. Our estimator is initialised with parameters ϕ and the sampling strategy proceeds in two stages with distinct data generation:

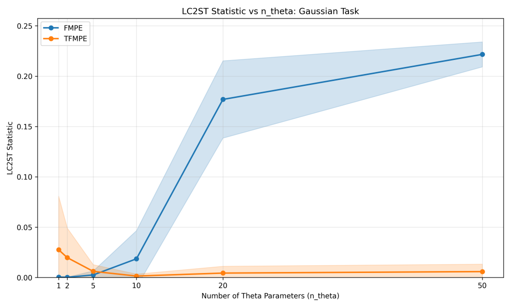
(1) Factorised Likelihood Learning: We first generate factorised likelihood training data $D_s = \{(\theta_g^{(i)}, \theta_{l[s]}^{(i)}, y_{[s]}^{(i)}, h_{[s]}^{(i)}, i_{[s]}^{(i)})\}_{i=1}^N$, where for each sample i we select a locale $s \in \{1, \ldots, n_s\}$, sample parameters from the prior $(\theta_g, \theta_{l[s]}) \sim p(\theta_g)p(\theta_{l[s]}|\theta_g)$, sample function inputs from $p(h_{[s]}, i_{[s]})$, and generate observations from the simulator.

D_s samples are then used to learn the factorised likelihood:

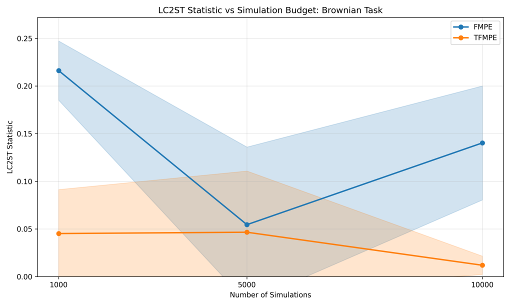
$$L_1(\phi) = \mathbb{E}_{\theta_g, \theta_{l[s]}, y_{[s]}, h_{[s]}, i_{[s]}} \left[L_{\text{FMPE}} \left(\phi; y_{[s]} | \theta_g, \theta_{l[s]}, h_{[s]}, i_{[s]} \right) \right]$$



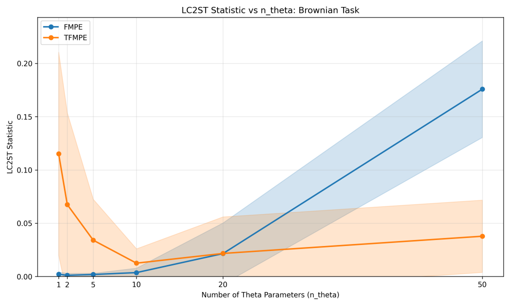
(a)



(b)



(c)



(d)

Figure 4: Sample efficiency results for Gaussian and Brownian motion tasks (detailed plots). Top row shows Gaussian task results: a Sample efficiency with simulation budget and b sample efficiency with number of local variables. Bottom row shows Brownian motion task results: c Sample efficiency with simulation budget and d sample efficiency with number of local variables. For all experiments: $m = 5$ observations per variable, $n_{\text{round}} = 1$. Simulation budget experiments used $n = 50$ local variables; local variable scaling experiments used $n_{\text{sim}} = 5,000$. All posteriors evaluated using the ℓ-c2st metric (lower is better). Each experiment was repeated over 6 random number generation seeds to capture mean and standard deviation performance. Quantitative results for the parameter scaling are summarized in Table 3.

(2) Full Posterior Learning: In this stage we generate posterior training data $D_m = \{(\theta_g^{(i)}, \theta_l^{(i)}, y^{(i)}, h^{(i)}, i^{(i)})\}_{i=1}^N$, where $\theta_l^{(i)} = (\theta_{l[1]}^{(i)}, \ldots, \theta_{l[n_s]}^{(i)})$ and observations $y^{(i)}$ are simulated from the estimated likelihood: $y^{(i)} \sim q_\phi(y|\theta_g^{(i)}, \theta_l^{(i)}, h^{(i)}, i^{(i)})$.

D_m samples are then used to learn the full posterior, $q_\phi(\theta_g, \theta_l|y, h, i)$, by optimising:

$$L_2(\phi) = \mathbb{E}_{\theta_g, \theta_l, y, h, i}\left[L_{\text{FMPE}}\left(\phi; \theta_g, \theta_l|y, h, i\right)\right]$$

At each training step, an aggregate dataset $D = D_s \cup D_m$ is maintained to train the estimator, applying the corresponding loss functions for each sample to jointly optimise each objective. The variable sizes of conditioning and target variables are addressed by padding the training data with empty tokens for fast batching, applying the appropriate attention masks for decoding and padding masks when computing the loss function.

Bottom-up sampling can also be extended as a sequential refinement. To refine the posterior estimate, in subsequent rounds one can resample (θ_g, θ_l) from the latest $q_\phi(\theta_g, \theta_l|y, h, i)$ to generate single-site training data which are more concentrated around the desired posterior. Sequential schemes have been shown to improve the quality of posterior estimates, see [Papamakarios et al., 2019, Greenberg et al., 2019].

```
**Input:** Prior functions $p(\theta_g), p(\theta_{l[s]}|\theta_g, h_{[s]})$, function input distributions $p(h_{[s]}), p(i_{[s]})$,
        simulator function $\text{sim}(\theta_g, \theta_{l[s]}, i_{[s]})$, observed data $y^{\text{obs}} = (y^{\text{obs}}_{[1]}, \ldots, y^{\text{obs}}_{[n_s]})$ with
        corresponding function inputs $h^{\text{obs}}$, $i^{\text{obs}}$, number of rounds $R$, number of samples per
        round $N$
**Output:** Trained tokenised estimator parameters $\phi$
// Initialization
Randomly initialize estimator parameters $\phi$ ;
Initialize empty single-simulation dataset $D_s = \{\}$ ;
Initialize empty multi-simulation dataset $D_m = \{\}$ ;
**for** $r = 0, 1, \ldots, R-1$ **do**
    // Generate single-simulation training data
    **if** $r = 0$ **then**
        Sample $\theta_g^{(i)} \sim p(\theta_g)$ for $i = 1, \ldots, N$ ;
        Sample function inputs $h_{[s]}^{(i)} \sim p(h_{[s]}), i_{[s]}^{(i)} \sim p(i_{[s]})$ for $i = 1, \ldots, N$ ;
        Sample $\theta_{l[s]}^{(i)} \sim p(\theta_{l[s]}|\theta_g^{(i)}, h_{[s]}^{(i)})$ for $i = 1, \ldots, N$ ;
        Simulate $y_{[s]}^{(i)} \sim \text{sim}(\theta_g^{(i)}, \theta_{l[s]}^{(i)}, i_{[s]}^{(i)})$ for $i = 1, \ldots, N$ ;
        $D_s = \{(\theta_g^{(i)}, \theta_{l[s]}^{(i)}, y_{[s]}^{(i)}, h_{[s]}^{(i)}, i_{[s]}^{(i)})\}_{i=1}^N$ ;
    **else**
        Sample $\theta_g^{(i)} \sim p(\theta_g)$ for $i = 1, \ldots, N$ ;
        Sample $\theta_l^{(i)} = (\theta_{l[1]}^{(i)}, \ldots, \theta_{l[n_s]}^{(i)}) \sim q_\phi(\theta_l|\theta_g^{(i)}, y^{\text{obs}}, h^{\text{obs}}, i^{\text{obs}})$ from previous round ;
        Simulate $y_{[s]}^{(i)} \sim \text{sim}(\theta_g^{(i)}, \theta_{l[s]}^{(i)}, i_{[s]}^{\text{obs}})$ for $i = 1, \ldots, N$ ;
        $D_s := D_s \cup \{(\theta_g^{(i)}, \theta_{l[s]}^{(i)}, y_{[s]}^{(i)}, h_{[s]}^{\text{obs}}, i_{[s]}^{\text{obs}})\}_{i=1}^N$ ;
    **end**
    // Stage 1: Learn local likelihood
    Optimize $\phi$ using $L_1(\phi) = L_{\text{FMPE}}(\phi; y_{[s]}|\theta_g, \theta_{l[s]}, h_{[s]}, i_{[s]})$ on $D_s$ ;
    // Generate multi-simulation training data using learned local
       likelihood
    **for** $i = 1, \ldots, N$ **do**
        Sample $\theta_g^{(i)} \sim p(\theta_g)$ ;
        Sample function inputs $h^{(i)} \sim p(h), i^{(i)} \sim p(i)$ ;
        Sample $\theta_{l[s]}^{(i)} \sim p(\theta_{l[s]}|\theta_g^{(i)}, h_{[s]}^{(i)})$ for $s = 1, \ldots, n_s$ ;
        Generate $y_{[s]}^{(i)} \sim q_\phi(y_{[s]}|\theta_g^{(i)}, \theta_{l[s]}^{(i)}, i_{[s]}^{(i)})$ for $s = 1, \ldots, n_s$ ;
    **end**
    $D_m := D_m \cup \{(\theta_g^{(i)}, \theta_l^{(i)}, y^{(i)}, h^{(i)}, i^{(i)})\}_{i=1}^N$ ;
    // Stage 2: Learn full posterior
    Optimize $\phi$ using $L_2(\phi) = L_{\text{FMPE}}(\phi; \theta_g, \theta_l|y, h, i)$ on $D_m$ ;
**end**
**return** $\phi$ *such that* $\theta_l, \theta_g \sim q_\phi(\theta_g, \theta_l|y^{obs}, h^{obs}, i^{obs})$
```

Algorithm 1: Bottom-up Sampling for Tokenised FMPE

A.6 Toy hierarchical inference experiments

A.7 The SEIR Infectious Disease Model

We use a Susceptible, Exposed, Infected, Recovered (SEIR) compartmental model as a simulator for comparing TFMPE and MCMC posterior estimation. An SEIR model is specified as Ordinary Differential Equations (ODE) which track the proportions of the population in each disease state. The infectiousness is parameterised by β_0, which controls the base transmission rate. The infectiousness

Task	**Global Parameters**	**Local Parameters**	**Observations**	**Functional Input**
Gaussian	$\mu \sim N(\mu_0, \sigma_0^2)$	$\nu_i \sim N(\mu, \sigma_\mu^2)$ $i = 1, \ldots, n_s$	$y_{i,j} \sim N(\nu_i, \sigma_y^2)$ $j = 1, \ldots, m$	—
Brownian	$\mu \sim N(\mu_0, \sigma_0^2)$	$\nu_i \sim N(\mu, \sigma_\mu^2)$ $i = 1, \ldots, n_s$	$y_{i,j} \sim N(\nu_i, t)$ $j = 1, \ldots, m$	$t \sim \text{Exp}(1)$

Table 2: Symbolic definition of toy hierarchical inference tasks. Global parameters govern the distribution of local parameters (partial pooling). A single simulation generates all observations $\{y_{i,j}\}$ for one local parameter ν_i. The Brownian task includes function-valued observations through the functional input t.

Table 3: LC2ST results for varying number of local variables (n_simulations = 5000, m = 5 observations).

n_theta	**Gaussian Task**		**Brownian Task**	
	FMPE	TFMPE(prior)	FMPE	TFMPE(prior)
1	**0.002 [-0.000, 0.005]**	0.115 [0.006, 0.225]	**0.000 [0.000, 0.001]**	0.028 [-0.034, 0.089]
2	**0.001 [-0.001, 0.003]**	0.067 [-0.032, 0.166]	**0.000 [0.000, 0.001]**	0.020 [-0.014, 0.053]
5	**0.002 [0.001, 0.003]**	0.034 [-0.010, 0.078]	**0.003 [-0.002, 0.007]**	0.006 [-0.002, 0.014]
10	**0.004 [-0.001, 0.008]**	0.013 [-0.003, 0.028]	0.019 [-0.014, 0.051]	**0.001 [-0.001, 0.004]**
20	**0.021 [-0.012, 0.055]**	0.022 [-0.018, 0.061]	0.177 [0.133, 0.221]	**0.004 [-0.003, 0.012]**
50	0.176 [0.146, 0.206]	**0.038 [0.015, 0.060]**	0.222 [0.213, 0.230]	**0.006 [0.001, 0.011]**

in any specific site is assumed to follow sinusoidal seasonal dynamics with amplitude A_s. The Kolmogorov forward equations for the model are:

$$\begin{aligned}
\frac{dS}{dt} &= -\beta_s SI \\
\frac{dE}{dt} &= \beta_s SI - \sigma E \\
\frac{dI}{dt} &= \sigma E - \gamma I \\
\frac{dR}{dt} &= \gamma I \\
\text{where} \quad \beta_s &= \beta_0(1 + A_s \cos(2\pi T^1 - \phi))
\end{aligned} \tag{2}$$

A.8 SEIR Model Inference with MCMC

As MCMC kernels we used both the No-U-Turn Sampler (NUTS) [Hoffman and Gelman, 2011] and Ensemble Slice Sampler (ESS) [Karamanis and Beutler, 2020] from the numpyro package [Phan et al., 2019]. Each sampler was run until convergence, with derivatives for NUTS provided by forward-mode autodifferentiation as implemented in the diffrax package [Kidger, 2021]. To make MCMC estimation tractable, we selected $n_s = 2$, and estimated only β and A_s, assuming other parameters were known. However for global infectious disease studies, it is not uncommon to estimate parameters using $n_s > 1000$ sites.

A.9 Hierarchical SBI Benchmark

We adapted a benchmark for Simulation Based Inference [Lueckmann et al., 2021] to evaluate the sample efficiency on hierarchical inference tasks. Similarly to the original benchmark, the posteriors exhibit a range of posterior geometries, dimensionality and dependence structures. However, we adapted the tasks in one of two ways: (1) All parameters were assumed conditionally independent given observations and replicated across n_s sites. New global parameters were introduced to govern the distribution of the replicated parameters (i.e. a partial pooling model). (2) Parameters were partitioned into global and local parameters with only the local parameters being replicated across n_s sites. The local parameters retained their original priors, however they were sampled independently across n_s sites.

Task	**Description**	**Parameter**	**Prior Distribution**
Gaussian Linear	Shared noise scale with site-specific means	*Global*	
		$\sigma \in \mathbb{R}^+$	$\mathrm{HalfNormal}(0.1)$
		Local	
		$\mu_s \in \mathbb{R}^5$	$\mathcal{N}(0, 0.1I_5)$, $s \in [n_s]$
Gaussian Linear Uniform	Shared noise scale with site-specific bounded means	*Global*	
		$\sigma \in \mathbb{R}^+$	$\mathrm{HalfNormal}(0.1)$
		Local	
		$\mu_s \in [-10, 10]^5$	$\mathrm{Unif}(-10, 10)$ per dim, $s \in [n_s]$
Gaussian Mixture	Global pooling controls local mixture parameters	*Global*	
		$\mu_g \in [-10, 10]^2$	$\mathrm{Unif}(-10, 10)$ per dim
		$\sigma_g \in \mathbb{R}^2_+$	$\mathrm{HalfNormal}(1.0)$ per dim
		Local	
		$\theta_{l[s]} \in [-10, 10]^2$	$\mathrm{TruncNorm}(\mu_g, \sigma_g, -10, 10)$, $s \in [n_s]$
Lotka-Volterra	Partial pooling of all LV parameters	*Global*	
		$\mu_\alpha, \mu_\beta, \mu_\gamma, \mu_\delta$	$\mathcal{N}(0, 1)$
		$\sigma_\alpha, \sigma_\beta, \sigma_\gamma, \sigma_\delta$	$\mathrm{HalfNormal}(0.5)$
		Local	
		$\alpha_s, \beta_s, \gamma_s, \delta_s \in \mathbb{R}^+$	$\mathrm{LogNormal}(\mu_*, \sigma_*)$, $s \in [n_s]$
SIR	Shared recovery rate with site-specific transmission rates	*Global*	
		$\gamma \in \mathbb{R}^+$	$\mathrm{LogNormal}(\log 0.125, 0.2)$
		Local	
		$\beta_s \in \mathbb{R}^+$	$\mathrm{LogNormal}(\log 0.4, 0.5)$, $s \in [n_s]$
SLCP	Shared covariance structure with site-specific means	*Global*	
		$s_1, s_2 \in [0.5, 3.0]$	$\mathrm{Unif}(0.5, 3.0)$ each
		$\rho \in [-3, 3]$	$\mathrm{Unif}(-3, 3)$
		Local	
		$m_{0,s}, m_{1,s} \in [-3, 3]$	$\mathrm{Unif}(-3, 3)$ each, $s \in [n_s]$
Two Moons	Global pooling controls local two-moon parameters	*Global*	
		$\mu_{g,0}, \mu_{g,1} \in [-1, 1]$	$\mathrm{Unif}(-1, 1)$ each
		$\sigma_{g,0}, \sigma_{g,1} \in \mathbb{R}^+$	$\mathrm{HalfNormal}(0.5)$ each
		Local	
		$\theta_{l[s]} \in [-1, 1]^2$	$\mathrm{TruncNorm}(\mu_g, \sigma_g, -1, 1)$, $s \in [n_s]$

Table 4: Hierarchical SBIBM benchmark tasks with parameter specifications and prior distributions. Strategy 1 tasks (Gaussian Linear, Gaussian Linear Uniform, SIR, SLCP) have naturally separated global and local parameters. Strategy 2 tasks (Gaussian Mixture, Lotka-Volterra, Two Moons) have global parameters controlling the distribution of local parameters. All local parameters are replicated independently across n_s sites.