Flow-Matching Generative Models and Conditional Models

Firas Darwish
firas.darwish@worc.ox.ac.uk

Bevis Drury
bevis.drury25@imperial.ac.uk

Susanna Paoli
s.paoli25@imperial.ac.uk

Anqi Qu
anqi.qu@ccc.ox.ac.uk

1 Introduction

Flow matching is a relatively novel method that can be employed to recover a target data distribution. It finds great applicability in generative models for complex data, particularly images, and overcomes the long training and ad hoc sampling procedures typical of diffusion-based models. More specifically, flow matching builds upon continuous normalising flows by parameterising with a vector field the ordinary differential equation describing the flow state at time t, where the vector field is approximated via neural network regression. Beyond its success in generative modelling, flow matching can also be adapted to posterior estimation in Bayesian inference, offering tractable posterior densities and architectural flexibility in the neural parameterisation. In the simulation-based inference (SBI) setting, the flow matching objective can be reformulated in terms of the prior over parameters and the likelihood of simulated data, providing a new approach to amortised Bayesian inference.

In this project, we investigate the use of flow matching for Bayesian posterior estimation, following recent developments by Wildberger et al. [2023] in Flow Matching Posterior Estimation (FMPE). We begin by reviewing the theoretical foundations underpinning this approach, including discrete and continuous normalising flows and neural posterior estimation (NPE), both of which have been widely adopted for amortised inference in complex simulators. Building upon this foundation, we discuss flow matching for generative modelling - including the formulation of the loss function via conditional vector fields - and then detail the necessary adaptations that allow flow matching to be applied in the simulation-based inference context.

We develop and test our own FMPE implementation for posterior estimation. We first explore the probability mass coverage properties of flow matching and demonstrate empirically that, like the forward Kullback–Leibler (KL) objective, the FMPE loss encourages mass-covering posterior estimates. We then evaluate the neural network architecture and conditioning strategies - including both gated linear unit (GLU) and naive concatenation conditioning - to identify the most effective design for amortised posterior inference.

Our experimental investigation focuses on both theoretical validation and empirical performance. We first test FMPE on a controlled synthetic problem: the bimodal "*two moons*" distribution, which provides a clear diagnostic for assessing posterior coverage. By comparing FMPE against models trained with forward and reverse KL objectives, we show that the FMPE loss, like the forward KL, effectively captures multimodality in the posterior, unlike the reverse KL, which tends to collapse to a single mode.

We then conduct an evaluation on the Two Moons SBI benchmark, reproducing the setup used in Wildberger et al. [2023]. Through an extensive hyperparameter search, we assess the sensitivity of FMPE to simulation budget, network width and depth, and conditioning method. Performance is quantified using Classifier Two-Sample Tests (C2ST), Maximum Mean Discrepancy (MMD), and mass coverage histograms, providing a robust, multi-metric comparison between FMPE and established SBI methods.

Our results confirm that flow matching posterior estimation yields competitive, and in some cases superior, posterior reconstructions relative to classical NPE approaches. In particular, FMPE demonstrates strong probability mass coverage and scalability with increasing network capacity and simulation budget. Moreover, we find that FMPE inherits several desirable properties from flow matching in generative modelling - namely, efficient training dynamics and flexible vector field parameterisation - while retaining tractable density evaluation required for Bayesian inference.

2 Background

2.1 Normalising flows

Normalising flows represent a flexible framework for defining expressive probability distributions through invertible transformations. Let $\mathbf{x} = T(\mathbf{u})$ where $\mathbf{u}$ is sampled from a simple *base distribution* $p_u(\mathbf{u})$. If the transformation T is a *diffeomorphism* – that is, both T and its inverse T^{-1} are differentiable – then it is possible to recover the distribution of $\mathbf{x}$ by a change of variables:

$$p_x(\mathbf{x}) = p_u(\mathbf{u}) |\det J_T(\mathbf{u})|^{-1}$$

In this setup, $J_T(\mathbf{u})$ denotes the Jacobian matrix of T with respect to $\mathbf{u}$. In practice, T is usually a neural network, and $p_u(\mathbf{u})$ is chosen to be a simple distribution such as a standard Normal.

Papamakarios et al. [2021] show that even if p_u is chosen to be a Uniform distribution over $\mathbb{R}^D$, any target distribution $p_x(\mathbf{x})$ satisfying certain conditions (e.g., non-zero everywhere) can be approximated by a normalising flow.

The parameters of a flow-based model are obtained by minimizing a loss function, or divergence, between the model $p_x(\mathbf{x}; \theta)$ and a target distribution $p_x^*(\mathbf{x})$, where $\theta = (\phi, \psi)$ denotes the parameters of the transformation T_ϕ and the base density $p_{u,\psi}$. Papamakarios et al. [2021] indicate the Kullback-Leibler (KL) divergence as one of the most popular choices, differentiating between forward KL divergence and reverse KL divergence. The *forward* KL divergence is defined as:

$$\mathcal{L}(\theta)_F = D_{KL}[p_x^*(\mathbf{x}) || p_x(\mathbf{x}; \theta)] = -\mathbb{E}_{p_x^*(\mathbf{x})}[\log p_x(\mathbf{x}; \theta)] + \text{const}$$

Using the change of variables formula, $\mathbf{x} = T_\phi(\mathbf{u})$, the divergence given by the expression above can be approximated from samples of the target distribution via Monte Carlo and minimised with respect to the parameters with stochastic gradient descent. Alternatively, if we can evaluate the target distribution but do not have samples from it, the divergence to fit the flow-based model can be obtained using the reverse KL divergence, defined as

$$\mathcal{L}(\theta)_R = D_{KL}[p_x(\mathbf{x}; \theta) || p_x^*(\mathbf{x})] = \mathbb{E}_{p_x(\mathbf{x};\theta)}[\log p_x(\mathbf{x}; \theta) - \log p_x^*(\mathbf{x})]$$

Such loss is usually minimised via stochastic gradient-based methods and is often employed in variational inference, where the target distribution is the posterior in a Bayesian model. Other divergences, such as f-divergences or integral probability metrics (IPMs), can also be employed depending on the learning objective.

Flow-based models can be categorised into two main types: discrete and continuous normalising flows.

Discrete normalising flows: In discrete flows, a complex expression T can be expressed as a finite composition of simpler bijections T_k, for $k \in \{1, ..., K\}$. As highlighted in Papamakarios et al. [2021]:

$$T = T_K \circ ... \circ T_1$$

with intermediate variables $\mathbf{z}_0 = \mathbf{u}$ and $\mathbf{z}_K = \mathbf{x}$.

Each sub-transformation T_k has parameters ϕ_k and is designed to be both invertible and to have a tractable Jacobian determinant. This compositional structure allows for high expressivity while retaining exact likelihood computation.

A widely used family of discrete flows are autoregressive flows, where the transformation of each dimension depends only on the preceding ones:

$$z'_i = \tau(z_i; \mathbf{h}_i), \quad \mathbf{h}_i = c_i(\mathbf{z}_{<i})$$

with a *transformer* τ and a *conditioner* c. This structure ensures a triangular Jacobian, making the log-determinant computation efficient:

$$\log|\det J_T| = \sum_i \log\left|\frac{\partial\tau(z_i; \mathbf{h}_i)}{\partial z_i}\right|$$

Autoregressive flows are universal density approximators, and the Jacobian is triangular; hence, it is easy to compute and use the determinant in the change of variables expression. Moreover, as thoroughly explained in Papamakarios et al. [2021], varying the choice of the transformer and conditioner allows for achieving different model expressiveness. In particular, transformers can be implemented using a variety of methods, including affine transformations, combination-based transformations, integration-based transformations, or splines. Instead, recurrent neural networks are popular choices for the conditioners, which determine the parameters of the transformer. Besides autoregressive flows, discrete flow-based methods can also be implemented using linear flows and residual flows.

Continuous normalising flows: In contrast to discrete normalizing flows, continuous-time transformations express T as the integration of infinitesimal steps, which is obtained using an ordinary differential equation (ODE). In particular, let $\mathbf{z}t$ denote the state of the flow at continuous time t, with $\mathbf{z}t_0 = \mathbf{u}$ and $\mathbf{z}t_1 = \mathbf{x}$. The flow dynamics are governed by a vector field $g\phi(t, \mathbf{z}_t)$:

$$\frac{d\mathbf{z}_t}{dt} = g_\phi(t, \mathbf{z}_t)$$

Integrating this ODE yields the transformation:

$$\mathbf{x} = \mathbf{u} + \int_{t=t_0}^{t_1} g_\phi(t, \mathbf{z}_t)dt.$$

Training such continuous-time models requires evaluating the change in log-density along the trajectory, which can be done using many different methods, including the change-of-variables formula and solved via numerical integrators such as Euler or adjoint methods.

Continuous normalizing flows thus provide a natural bridge to recent advances such as diffusion models and flow matching, where the transformation is learned by fitting vector fields directly rather than by explicit invertible layers.

2.2 Neural posterior estimation (NPE)

Many scientific and engineering problems require inferring parameters $\boldsymbol{\theta}$ of a simulator or generative model given observed data $\mathbf{x}$. When the likelihood $p(\mathbf{x}|\boldsymbol{\theta})$ is intractable or computationally expensive to evaluate, traditional Bayesian inference methods become infeasible. *Simulation-based inference* (SBI) circumvents this by relying solely on the ability to generate samples $\mathbf{x} \sim p(\mathbf{x}|\boldsymbol{\theta})$.

Wildberger et al. [2023] discusses how **Neural posterior estimation (NPE)** tackles SBI by learning a neural conditional density estimator $q_\phi(\boldsymbol{\theta}|\mathbf{x})$ that approximates the true posterior $p(\boldsymbol{\theta}|\mathbf{x})$. The model is trained on synthetic pairs $(\boldsymbol{\theta}, \mathbf{x})$ generated from the joint prior–simulator distribution $p(\boldsymbol{\theta})p(\mathbf{x} \mid \boldsymbol{\theta})$ using the objective:

$$\mathcal{L}(\phi)_{NPE} = -\mathbb{E}_{p(\boldsymbol{\theta})p(\mathbf{x}|\boldsymbol{\theta})}[\log q_\phi(\boldsymbol{\theta}|\mathbf{x})]$$

which is equivalent to minimizing the forward Kullback–Leibler divergence.

In practice, the expectation is approximated by Monte Carlo using a finite set of simulated samples. Once trained, the estimator can produce approximate posteriors for new observations $\mathbf{x}_{obs}$ without re-running the simulator, which effectively amortises the inference cost across multiple observations.

A key ingredient in NPE is the use of expressive normalizing flows as the conditional density estimator $q_\phi(\boldsymbol{\theta}|\mathbf{x})$. For each conditioning value $\mathbf{x}$, the flow defines an invertible mapping:

$$\boldsymbol{\theta} = T_\phi(\mathbf{u}; \mathbf{x}), \quad \mathbf{u} \sim p_u(\mathbf{u})$$

allowing both efficient sampling and exact evaluation of the approximate posterior density:

$$q_\phi(\boldsymbol{\theta}|\mathbf{x}) = p_u(\mathbf{u}) \left|\det J_{T_\phi}(\mathbf{u};\mathbf{x})\right|^{-1}$$

By conditioning the flow on $\mathbf{x}$, NPE captures complex, multimodal, and heteroscedastic posterior structures that arise in nonlinear simulators.

From a methodological standpoint, NPE connects naturally to continuous and flow-based models. The conditional flows used in NPE are discrete normalizing flows trained by maximum likelihood, while recent developments - such as continuous normalizing flows and flow matching - extend this framework to vector-field-based formulations.

2.3 Flow matching for generative modeling

Building upon continuous normalizing flows, Lipman et al. [2022] propose a methodology, namely *flow matching*, that enables the recovery of a data distribution $q(x_1)$, assuming samples from $q(x_1)$ are available. Flow matching allows the model to transform the base distribution p_0 into a target distribution $p_1 \approx q$. In this framework, the parametrization of the ODE is obtained through a vector field. In particular, the loss minimised in flow matching is given by

$$\mathcal{L}(\theta)_{FM} = \mathbb{E}_{t,p_t(x)}||v_t(x) - u_t(x)||^2$$

where t is sampled from a Uniform distribution, $u_t(x)$ is the vector field generating the probability path $p_t(x)$, and v_t is a neural network with parameters θ that learns to approximate u_t.

Notably, Lipman et al. [2022] show that probability paths can be constructed by marginalizing conditional probability paths. Specifically, after setting $p_0(x|x_1) = p(x)$ and $p_1(x|x_1)$ to be concentrated around x_1 (with $x_1 \sim q(x_1)$), the marginal probability paths are obtained by integration:

$$p_t(x) = \int p_t(x|x_1)q(x_1)dx_1.$$

Following this construction, at the end of the flow, the target distribution $q(x_1)$ is approximated via a mixture distribution of the form:

$$q(x) \approx \int p_1(x|x_1)q(x_1)x_1$$

Similarly, vector fields u_t can be obtained by marginalizing the conditional vector fields, which in turn generate the marginal probability path. Since these integrals are generally intractable, Lipman et al. [2022] prove the equivalence between the flow matching objective defined above and the *conditional flow matching* objective:

$$\mathcal{L}(\theta)_{CFM} = \mathbb{E}_{t,p_t(x_1),p_t(x|x_1)}||v_t(x) - u_t(x|x_1)||^2$$

By minimizing this loss, it is possible to obtain unbiased estimates for θ. The construction works in general for any choice of conditional probability path and conditional vector fields.

In Lipman et al. [2022], an extensive analysis of Gaussian probability paths is presented. In particular, $p_t(x|x_1) = \mathcal{N}(x|\mu_t(x_1), \sigma_t(x_1)^2 I)$, with $\mu_1(x_1)$ and $\sigma_1(x_1)$ set such that $p_1(x|x_1)$ is a Gaussian distribution centered around x_1. Using a simple transformation between Gaussian distributions, $\psi_t(x) = \sigma_t(x_1)x + \mu_t(x_1)$, the vector field that transports $p_0(x|x_1)$ to $p_t(x|x_1)$ is defined as:

$$\frac{d}{dt}\psi_t(x) = u_t(\psi_t(x)|x_1).$$

An interesting connection to Optimal Transport can be noticed by making the mean and standard deviation of the Gaussian distribution family change linearly with time. In particular, the conditional flow is the Optimal Transport displacement map between the starting point $p_0(x|x_1)$ and the arriving distribution $p_1(x|x_1)$.

Lipman et al. [2022] also investigate the performance of flow matching with other image generation baselines, i.e., U-Net architecture with different diffusion-based losses. Their experimental results show that flow matching converges the fastest and results in better sampling efficiency when implemented via the Optimal Transport construction.

2.4 Flow matching for posterior distributions

Wildberger et al. [2023] adapt the flow matching construction described above to posterior densities in Bayesian inference, as an alternative to discrete normalizing flows. The method falls within the neural posterior estimation methodology, which allows for fitting a density estimator $q(\theta|x)$ to the posterior distribution $p(\theta|x)$.

To adapt flow matching to Bayesian inference, Wildberger et al. [2023] use Bayes' theorem to rewrite the flow matching objective with respect to the prior and the likelihood, obtaining the following expression

$$\mathcal{L}(\theta)_{FMPE} = \mathbb{E}_{t\sim p(t),\theta_1\sim p(\theta),x\sim p(x|\theta_1),\theta_t\sim p_t(\theta_t|\theta_1)}||v_{t,x}(\theta_t) - u_t(\theta_t|\theta_1)||^2$$

The data to minimise this loss is obtained by sampling θ from the prior distribution and simulating the x from $p(x|\theta)$ given the sampled parameter. In terms of theoretical guarantees, Wildberger et al. [2023] show that flow matching for posterior distributions is promising in terms of probability mass coverage. In particular, the mean squared error of the vector fields in the flow matching objective bounds the KL divergence employed in neural posterior estimation.

For the network architecture used to parametrise v_t, Wildberger et al. [2023] start by extracting features from the data x and fixing the dimension to $dim(\theta)$. Subsequently, they condition on θ and t using gated linear units in the hidden layers of the net.

When examined with other simulation-based inference (SBI) methods, flow matching compares to and, in some cases, outperforms SBI benchmarks, and empirical investigation also provides evidence for mass-covering estimates.

3 Methods

3.1 Probability mass coverage

Trained Flow Matching Posterior Posterior Estimation (FMPE) models have been shown to approximate the true posterior distribution with high accuracy. However, their accuracy is, in general, limited by the model's expressiveness and training limitations. To study the extent of these limitations, we investigated how FMPE performs when using a model that is incapable of capturing the full posterior distribution. Using a bimodal distribution as the true posterior, we trained a Gaussian model with three separate training objectives to see how well the model captures both modes of the posterior in each case. The first two training objectives were the minimisation of the forward and reverse KL divergences between the true posterior and the predicted posterior. The divergences were minimised using automatic differentiation and stochastic gradient descent on two learnable parameters $\hat{\mu}$ and $\hat{\sigma}$, such that the posterior approximation is distributed $q(\theta|x) = \mathcal{N}(\hat{\mu}, \hat{\sigma})$. For the final training objective, we trained with FMPE. Wildberger et al. [2023] showed that the following vector field $v_t(\theta_t)$ produces a continuous flow to a one dimensional Gaussian with mean $\hat{\mu}$ and variance $\hat{\sigma}^2$

$$v_t(\theta_t) = \frac{(\sigma_t^2 + (t\hat{\sigma})^2 - \sigma_t)\theta_t + t\hat{\mu}\hat{\sigma}}{t \cdot (\sigma_t^2 + (t\hat{\sigma})^2)}$$

By sampling from the true posterior $\theta_1 \sim p(\theta|x)$ we train this parameterised vector field by minimising the mean-squared-error loss between $v_t(\theta_t)$ and the sample-conditional vector field $u_t(\theta_t|\theta_1)$

$$u_t(\theta_t|\theta_1) = \frac{\theta_1 - (1 - \sigma_{min})\theta_t}{1 - (1 - \sigma_{min}) \cdot t}$$

For the reverse KL divergence objective, the estimate $q(\theta|x)$ exhibited probability-mass coverage on only one of the modes of the true posterior. However, for both the forward KL divergence and the

FMPE objectives, the predicted posterior showed coverage of both modes, indicating that the flow matching and forward KL divergence objectives produce similar results.

(a) $\underset{\hat{\mu},\,\hat{\sigma}}{\text{argmin}}\, \mathcal{L}_{FM}$ (b) $\underset{\hat{\mu},\,\hat{\sigma}}{\text{argmin}}\, D_{KL}(p||q)$ (c) $\underset{\hat{\mu},\,\hat{\sigma}}{\text{argmin}}\, D_{KL}(q||p)$

Figure 1: A Gaussian (blue) fit to a bimodal distribution (grey) with three different minimisation objectives: (a) flow matching loss, (b) forward KL divergence and (c) reverse KL divergence

3.2 Network Architecture

We use the same architecture proposed in Wildberger et al. [2023] and condition on t and θ in the same way. The backbone of the network is a dense residual network composed of multiple hidden layers (an explanation of the number and width of hidden layers is described in 3.4.2.

This network is a direct mapping from x (input) to $v(x)$ as opposed to a direct mapping from θ_t to a vector field $v(\theta_t)$ due to the higher dimensionality of the observations compared the dimensionality of θ Wildberger et al. [2023]. There are two principal ways we can condition on t and θ and the choice between these two methods depends on the task that is being evaluated Wildberger et al. [2023]:

1. **GLU Conditioning**: In this approach, we condition on (t, θ) using gated linear units connected to the hidden layers of the dense residual network.

2. **Naive Concatenation**: In this approach, we simply concatenate t, θ, and x into a single vector (t, θ, x) that we pass into the network as input.

3.3 Evaluation

To assess the quality of FMPE in estimating posterior distributions, we use the same set of metrics utilised by Wildberger et al. [2023]. We describe these in detail below as well as offer an argument as to why they are each ideal for evaluating FMPE:

- **C2ST Score**: To extract this score, we train a classifier on samples from $\theta \sim q(\theta|x)$ which we acquire from FMPE and reference samples from $\theta \sim p(\theta|x)$ Wildberger et al. [2023]. We then test this classifier on a held out dataset and use its accuracy as an indicator of how different the distributions $q(\theta|x)$ and $p(\theta|x)$ are. An accuracy of 0.5 indicates that the two distributions are highly similar (and the classifier struggles with distinguishing samples from them). An accuracy of 1.0 indicates that the two distributions are different (and the classifier can easily discriminate samples from them).

- **Maximum Mean Discrepancy (MMD)**: This metric measures the divergence between the two probability distributions $q(\theta|x)$ and $p(\theta|x)$ based on their mean embeddings in a reproducing kernel Hilbert space. An MMD score of 0 indicates that the two distributions are identical under the defined kernel. One critical limitation of MMD that is expressed by Wildberger et al. [2023] is that it can be sensitive to hyperparameters, which is why we it is not our only evaluation metric.

- **Mass Coverage**: Following from Wildberger et al. [2023], we empirically assess FMPE's mass coverage. We provide a histogram of the density log $q(\theta|x)$ based on our FMPE model of the reference samples $\theta \sim p(\theta|x)$. We plot this with the another histogram of the density log $q(\theta|x)$ but this time evalauted on samples from our FMPE model $\theta \sim q(\theta|x)$. This allows us to empirically assess whether the samples from $p(\theta|x)$ fall under the support of $q(\theta|x)$ Wildberger et al. [2023].

3.4 Experiments

3.4.1 Tasks

Wildberger et al. [2023] evaluates FMPE on ten different simulations. These ten tasks are listed in 1 and offer a great variety of problems with varying dimensionality of observations $dim(x) \in [2, 100]$ and dimensionality of parameters $dim(\theta) \in [2, 10]$. Despite the vast computational resources and GPUs used in training (see 3.4.3), it would take around 2 weeks to effectively train on all 10 of these tasks, particularly because it would involve conducting an extensive hyperparameter search (a total of 18,000 models would have to be trained).

Therefore, to ensure that we test and evaluate the method proposed fairly and conduct a hyperparameter search space akin to Wildberger et al. [2023], we only reproduce results on the **Two Moons** task. We simulate the Two Moons task with two different simulation budgets $N = \{10^3, 10^5\}$.

Table 1: Overview of the ten simulation-based inference benchmark (SBIBM) tasks from Lueckmann et al. [2021]. Each task defines a generative model $p(x \mid \theta)$ with specified data and parameter dimensionalities.

Task Name	$\dim(x)$	$\dim(\theta)$	**Generative Model / Description**
Gaussian Linear	10	10	Linear Gaussian model with Gaussian prior; θ is the mean of a Gaussian likelihood with fixed covariance.
Gaussian Linear Uniform	10	10	As with the Gaussian Linear task, but with a uniform prior on θ instead of Gaussian.
Gaussian Mixture	2	2	Mixture of two 2D Gaussians (one broad, one narrow), where θ parameterises the means.
Two Moons	2	2	Bimodal "two moons" likelihood; posterior exhibits crescent-shaped and globally bimodal structure.
SLCP	8	5	Simple Likelihood, Complex Posterior (SLCP): Gaussian likelihood whose mean and variance are nonlinear functions of θ.
SLCP Distractors	100	5	SLCP with 92 additional non-informative (distractor) dimensions in x.
Bernoulli GLM	10	10	Bernoulli Generalised Linear Model (GLM) with 10 parameters, using sufficient statistics as data.
Bernoulli GLM Raw	100	10	Same as Bernoulli GLM, but inference performed directly on 100 raw binary observations.
SIR	4	2	Epidemiological SIR model describing Susceptible–Infectious–Recovered dynamics. $\theta = (\beta, \gamma)$ parameterises infection and recovery rates; data are infection counts at 10 time points.
Lotka Volterra	5	4	Predator–prey dynamical system with four parameters governing species interaction and growth; data are population counts at 10 time points.

3.4.2 Hyperparameter Search

We conduct an extensive hyperparameter search to fairly evaluate the FMPE method. As suggested by Wildberger et al. [2023], the hyperparameter space we search depends on the simulation budget and is described in 2.

This means we train 60 different models for each simulation budget and select the best model (and hyperparameter configuration) for evaluation based on the lowest validation loss acquired. The network architecture is defined by the number of blocks and maximum width of the network, which

combine to form a diamond-shaped architecture. The width of the layers grows to reach the defined maximum width of the network and then decreases to the output dimension. Each block in the network is two fully-connected residual layers.

Following from Wildberger et al. [2023], several factors in training remain constant throughout our grid search of the hyperparameters. We use an Adam optimiser in all of our experiments and a `ReduceLROnPlateau` learning rate scheduler with a factor of 0.2 and patience of 1. We train for 100 epochs and hold out 5% of our simulation for validation througout training. As we are evaluating on the Two Moons task, we use the **Naive Concatenation** approach for conditioning on t and θ as discussed in 3.2. Furthermore, we use the same ODE solver, Dormand–Prince 5(4) Runge–Kutta solver (`dopri5`), throughout.

Table 2: Hyperparameter search spaces for 10^3 and 10^5 simulation budgets following from Wildberger et al. [2023]

Parameter	**Budget** 10^3	**Budget** 10^5
Batch Size	[8]	[1024]
Maximum Width (Power of 2)	[5, 6]	[9, 10]
Number of Blocks in Network	[10, 12]	[16, 18]
Learning Rate	[1e-3, 5e-4, 2e-4]	[5e-4, 2e-4, 1e-4]
α **(for time prior)**	[-0.25, -0.5, 0, 1, 4]	[-0.25, -0.5, 0, 1, 4]

3.4.3 Implementation Details

We use 4 V100 GPUs simultaneously to accelerate our training using the Python framework PyTorch, which under the conditions we set (described in the following sections) took around 6 hours to complete. We evaluate our methods on 6 CPUs. We build on several code bases and repositories, which we describe below:

- **SBIBM**[1]: the simulation benchmark from Lueckmann et al. [2021] that includes the code for the **Two Moons** task we are evaluating FMPE on.
- **Dingo**[2]: a library that provides code for efficiently training neural networks for posterior estimation.
- **Flow-Matching-Posterior-Estimation**[3]: the companion repository for Lueckmann et al. [2021], which provides a basic training and evaluation script for FMPE.

All of our code is published and available for use and can be found here. Our code parallelises training on GPUs using `DataParallel` from the PyTorch library. It also offers a new training script that automatically searches a preset hyperparameter space, stores all models trained (along with their weights, training logs, and best model), and saves the best performing model.

4 Results & Discussion

Despite the limited simulation environment undertaken in our experiments, our results reflect the success of Wildberger et al. [2023] and show that FMPE is a promising direction for estimating the posterior distribution. 2 empirically showcases both how FMPE approximates the posterior distribution well (based on the samples shown) but also how the estimation drastically improves when the simulation budget is increased from 10^3 to 10^5.

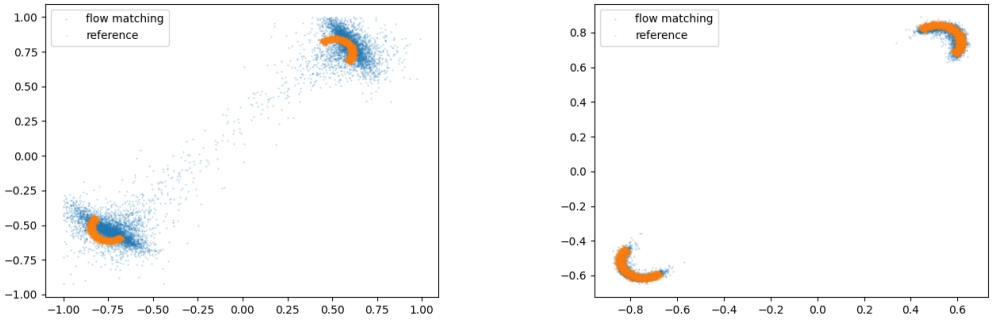
When both FMPE models (of different budgets) are tested on the held-out datasets from Lueckmann et al. [2021], we can see from 3 and 4 that the C2ST score is much closer to 0.5 and the MMD score (with the exception of set 2) is much closer to 0 when we train on 10^5 samples–indicating an improvement in the posterior estimation when the simulation budget is higher.

[1] `https://github.com/sbi-benchmark/sbibm`

[2] `https://github.com/dingo-gw/dingo`

[3] `https://github.com/dingo-gw/flow-matching-posterior-estimation`

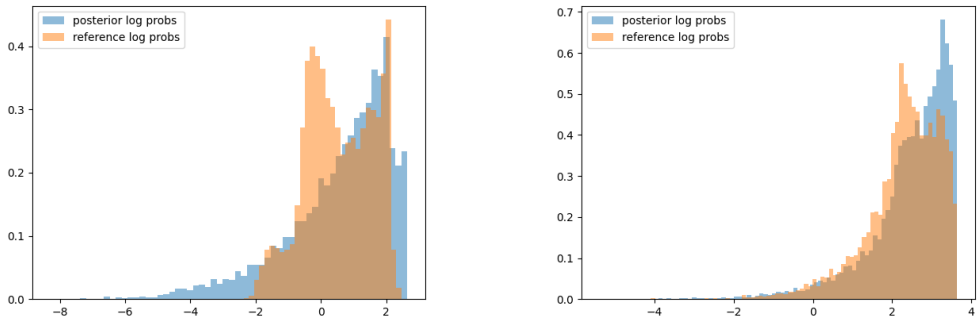
We also find in 3 that the reference samples $\theta \sim p(\theta|x)$ fall in the support of the learned FMPE model $q(\theta|x)$, and we find that this remains true across both simulation budgets (reaffirming the mass covering property of FMPE discussed in Wildberger et al. [2023]). 3 also shows how the posterior estimation improves with a higher simulation budget (the densities match better when the simulation budget is 10^5–where the testing loss is lower).



(a) Simulation budget 10^3.

(b) Simulation budget 10^5.

Figure 2: Comparison of posterior samples on the Two Moons simulation for different simulation budgets with a small budget (10^3) (left) and a larger budget (10^5) (right).



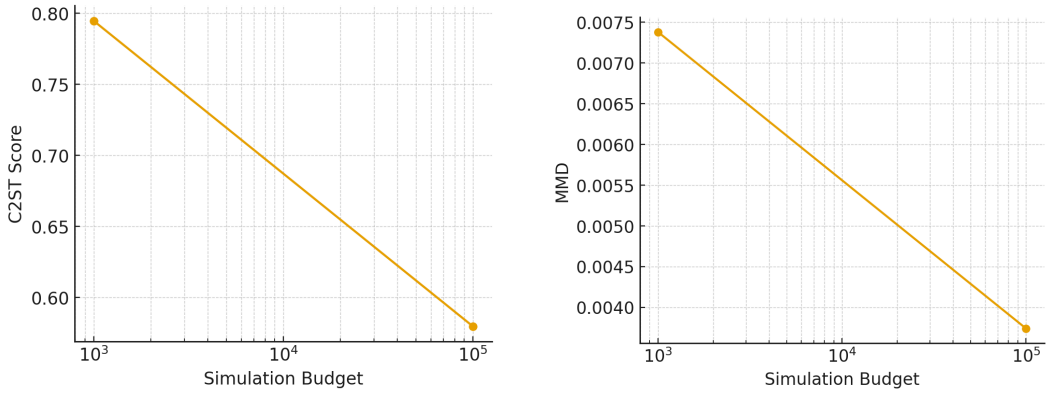
(a) Simulation budget 10^3.

(b) Simulation budget 10^5.

Figure 3: Histogram of FMPE densities log $q(\theta|x)$ for samples from FMPE model $\theta \sim q(\theta|x)$ and reference samples $\theta \sim p(\theta|x)$ for different simulation budgets with a small budget (10^3) (left) and a larger budget (10^5) (right).

Table 3: FMPE performance on nine held-out sets from the SBI benchmark Lueckmann et al. [2021], with metrics (C2ST, MMD) reported for simulation budgets 10^3 and 10^5. Lower MMD and C2ST values closer to 0.5 indicate better posterior estimation. **Bolded** results indicate better performance

	C2ST		MMD	
Simulation Budget	10^3	10^5	10^3	10^5
Set 1	0.7946	**0.5796**	0.00738	**0.00374**
Set 2	0.7786	**0.5610**	**0.00178**	0.00285
Set 3	0.9433	**0.5729**	0.07561	**0.00705**
Set 4	0.8989	**0.5623**	0.05191	**0.00568**
Set 5	0.9071	**0.5637**	0.06272	**0.00056**
Set 6	0.9165	**0.5608**	0.02854	**0.00355**
Set 7	0.9152	**0.5870**	0.03827	**0.00946**
Set 8	0.9322	**0.5764**	0.01924	**0.01220**
Set 9	0.8817	**0.5942**	0.02320	**0.00404**



(a) C2ST score when varying simulation budget.

(b) MMD score when varying simulation budget.

Figure 4: Comparison of C2ST (left) and MMD (right) on held-out set 1 from the SBI benchmark Lueckmann et al. [2021] when varying the simulation budget

5 Conclusion

In conclusion, in this project, we investigated the usage of flow matching for posterior estimation in Bayesian inference. Flow matching is a powerful technique that allows for approximating data distributions when the true density is unavailable. We reviewed normalising flows, which provide the basis for flow-matching generative models, discussing the choice of objective function and differentiating between discrete and continuous normalising flows. Since flow matching offers an alternative within the context of simulation-based inference methods, we also analysed neural posterior estimation, which operated as a state-of-the-art methodology before the novel flow-matching approach. After recalling the background on flow matching for image generation and simulation-based inference, we implemented practical experiments to analyse the probability mass-covering property and test the approximation of the bimodal "two moons" distribution. We found that flow matching satisfies the mass-covering property and performs similarly to the KL divergence. It is also able to approximate the bimodal "two moons" distribution properly, tested with two different simulation budgets, hence substantiating the results obtained in Wildberger et al. [2023].

References

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow Matching for Generative Modeling. September 2022. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking simulation-based inference. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 343–351. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/lueckmann21a.html.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22(1):57:2617–57:2680, January 2021. ISSN 1532-4435.

Jonas Bernhard Wildberger, Maximilian Dax, Simon Buchholz, Stephen R. Green, Jakob H. Macke, and Bernhard Schölkopf. Flow Matching for Scalable Simulation-Based Inference. November 2023. URL https://openreview.net/forum?id=D2cS6SoYlP&noteId=aatXK1ish.