Bayesian Inverse Problems Meet Flow Matching: Efficient and Flexible Inference via Transformers

The 10th International Conference on Stochastic Methods

Daniil Sherki
Skoltech
AI4Science, Sber

June 4, 2025

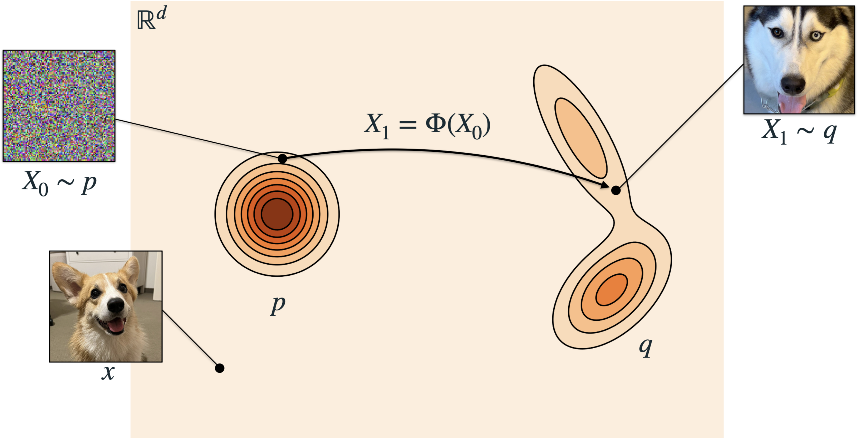
Overview

1. Flow Matching Basics

2. Why it works?

3. Combining Flow Matching and Transformers for Efficient Solution of Bayesian Inverse Problems

The Generative Modeling Problem [1]



[1]Some images are taken from the paper: [Lipman et al., 2024]

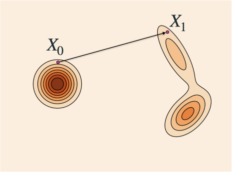
Flow Matching definition

Flow Matching is a generative modeling method that is a simulation-free approach to training continuous normalizing flow models via regression of vector fields of fixed conditional probability trajectories between noise and data, providing training scalability, robustness, and compatibility with different families of probabilistic paths (including optimal transport). [Lipman et al., 2023]

Generative Modeling Approaches

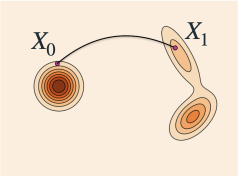
Direct Map

$$X_1 = \Phi(X_0)$$

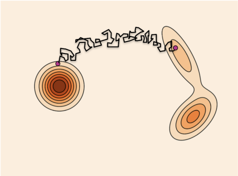


Continuous-time Markov process $(X_t)_{0 \le t \le 1}$

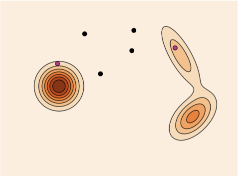
$$X_{t+h} \leftarrow \Phi_{t+h|t}(X_t)$$



Flow



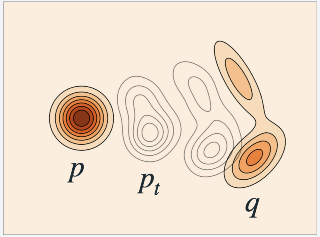
Diffusion

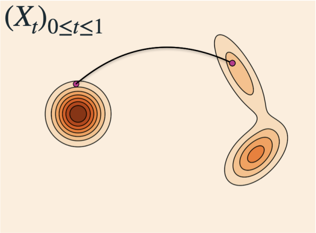


Jump

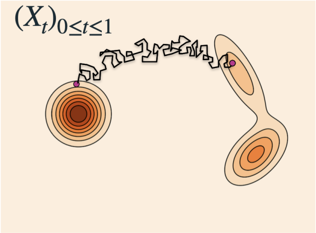
Marginal Probability Path

$$X_t \sim p_t$$

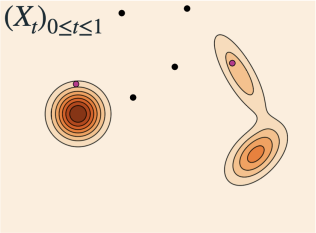




Flow

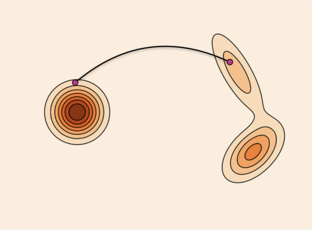


Diffusion



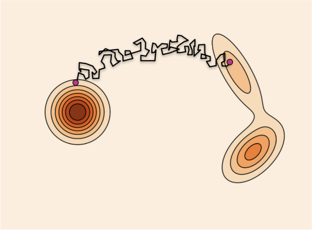
Jump

Why Flow Matching?

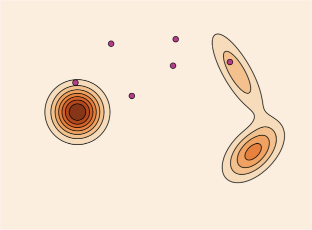


Flow

- Simple
- Faster sampling
- Exact likelihood estimator
- Flexible, easier to build



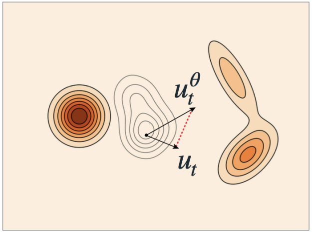
Diffusion



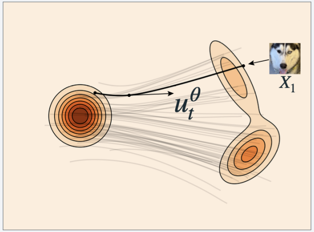
Jump

- Larger design space
- Slower sampling
- ELBO

Flow Matching: Train and Sampling

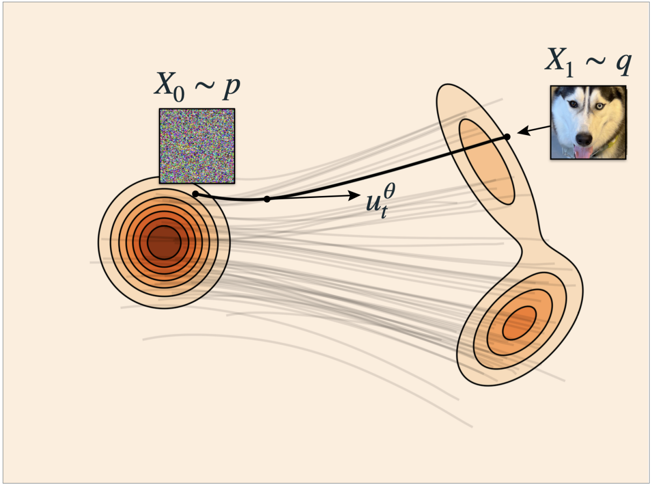


Train a velocity generating p_t with $p_0 = p$ and $p_1 = q$



Sample from $X_0 \sim p$

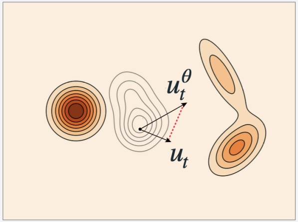
Flow Matching: Sampling



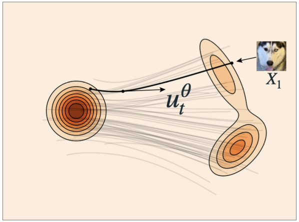
$$\frac{d}{dt} X_t = u_t^\theta(X_t)$$

Use any ODE numerical solver.
One that works well: Midpoint

Flow Matching: Train and Sampling

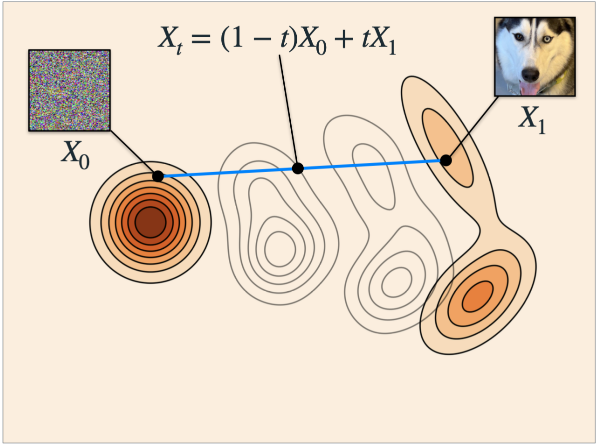


Train a velocity generating p_t with $p_0 = p$ and $p_1 = q$



Sample from $X_0 \sim p$

Flow Matching: Train



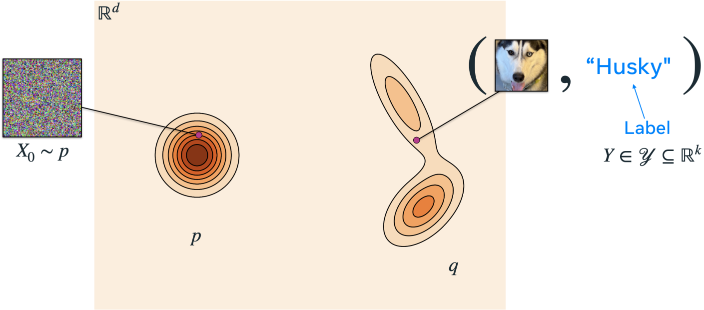
Training process:

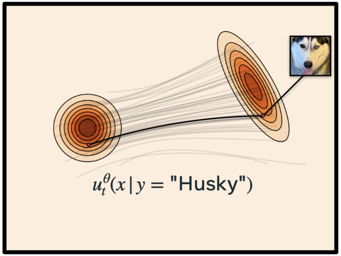
$$\arg\min_{\theta} \mathcal{L}(\theta)$$

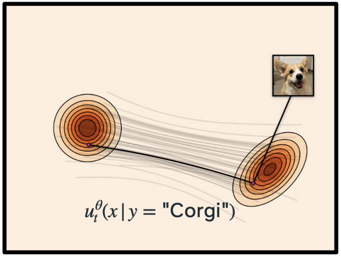
using Stochastic gradient descent $u_t^\theta(\cdot)$ is Flow Matching neural network

$$\mathcal{L}(\theta) = \mathbf{E}_{t,X_0,X_1} \| u_t^\theta(X_t) - (X_1 - X_0) \|^2$$

Flow Matching for Conditional generation







$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t,(X_0,X_1,Y)\sim\pi_{0,1,Y}} \left\| (X_1 - X_0) - u_t^\theta(X_t \mid Y) \right\|^2$$

where $\quad u_t^\theta(x \mid y) : [0,1] \times \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}^d$

Flow Matching: Vector Fields

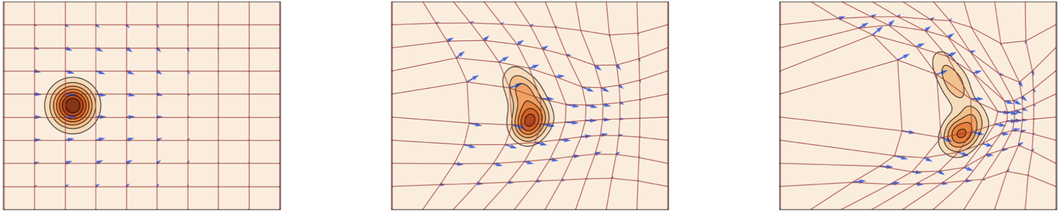


Figure: A velocity field u_t (in blue) *generates* a probability path p_t (PDFs shown as contours) if the flow defined by u_t (square grid) reshapes p (left) to p_t at all times $t \in [0, 1]$.

Problem Statement

- Let $p_0(x)$ be a simple initial density (e.g., $\mathcal{N}(0, I)$), and $p_1(x) = p_{\text{data}}(x)$ the target density.
- Define a family of densities $\{p_t(x)\}_{t\in[0,1]}$, where

$$p_0(x), \quad p_1(x), \quad \forall\, t \in (0, 1)\ p_t(x) \text{ is given beforehand (linear interpolation).}$$

- We seek a vector field $v(x, t)$ such that, when solving

$$\frac{dX_t}{dt} = u(X_t, t), \qquad X_0 \sim p_0,$$

  it holds that $X_1 \sim p_1$ exactly.
- The key condition: the *continuity equation* links $u_t(x)$ and the evolution of the density p_t.

Kolmogorov (Fokker–Planck) Equation

- For a stochastic process

$$dX_t = f(X_t, t)\,dt + g(t)\,dW_t,$$

the density $p_t(x)$ satisfies

$$\frac{\partial p_t(x)}{\partial t} = -\nabla_x \cdot \big(f(x,t)\,p_t(x)\big) + \frac{1}{2}\sum_{i,j} \frac{\partial^2}{\partial x_i\,\partial x_j}\big([g(t)g(t)^\top]_{ij}\,p_t(x)\big). \tag{Kolmogorov–Fokker–Planck}$$

- In the deterministic case ($g(t) \equiv 0$, $f(x,t) = u(x,t)$), the second term vanishes, reducing to the continuity equation:

$$\partial_t p_t + \nabla \cdot (p_t\,u) = 0.$$

- Thus, the Continuity Equation is a special case of the Kolmogorov forward equation when there is no noise.

Continuity Equation

- For a deterministic flow $\dot{X}_t = u(X_t, t)$, the density $p_t(x)$ evolves according to

$$\frac{\partial p_t(x)}{\partial t} + \nabla_x \cdot \big(p_t(x)\, u(x, t)\big) = 0. \qquad \text{(Continuity Equation)}$$

- Interpretation: no "mass" is lost or created; all particles move according to the field $u(x, t)$.
- If we predefine $p_t(x)$ (e.g., as a mixture), the unique flow preserving mass must satisfy the continuity equation.

Loss Function for Training

- We can sample:
  For $t \sim U[0,1]$, $X_0 \sim p_0$, $X_1 \sim p_1$, compute

$$X_t = (1-t)\,X_0 + t\,X_1, \qquad r_{\text{sample}} = X_1 - X_0.$$

- Train a neural network $u_\theta(x,t)$ by minimizing:

$$\mathcal{L}(\theta) = \mathbb{E}_{t,X_0,X_1}\left[\left\|\, v_\theta(X_t,t) \;-\; (X_1 - X_0)\right\|^2\right]. \qquad \text{(Velocity Regression)}$$

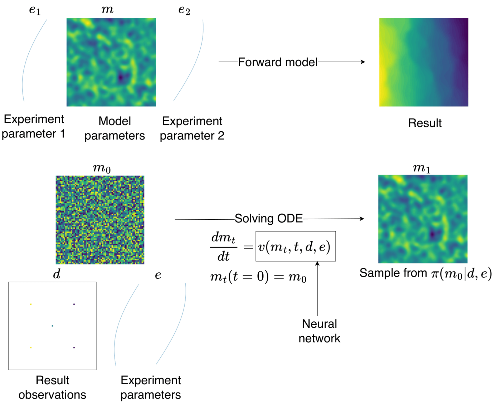
- Ideally: $u_\theta^*(x,t) = \mathbb{E}[X_1 - X_0 \mid X_t = x]$.
- Then γ_t solves the continuity equation, and the ODE $\dot{X}_t = v_\theta(X_t,t)$ yields $X_1 \sim p_1$.

Methodology [Sherki et al., 2025]

$$d(e, m) = \mathcal{F}(e, m) + \eta$$

- m unknown/unobservable parameters
- e known experiment parameters
- d observations from the forward model

Our goal to estimate conditional distribution $\pi(m|d, e) = \frac{\pi(d|m,e)}{\pi(m)}$ using conditional flow matching



Algorithm

Algorithm 1: Conditional Flow Matching Training Algorithm

Input: Dataset of paired samples (m_1, e, d), neural network model $\mathbf{v}_\theta(t, m, e, d)$, conditioning data e and d, time $t \sim \text{Uniform}(0,1)$, number of epochs N_{epoch}

Output: Trained conditional flow model $\mathbf{v}_\theta(t, m, e, d)$

```
for 1 to N_epoch do
    for each minibatch of samples (m_0, m_1) do
        t ~ U(0,1)                                                    // Sample t
        m_0 ~ prior distribution
        m_t ← t · m_1 + (1 − t) · m_0
        Compute the target velocity: u_t ← m_1 − m_0
        Predict the velocity: v_t ← v(t, m_t, e, d)
        Compute the loss: L(θ) ← E[(v_t − u_t)^2]
        Compute gradients: ∇_θ L(θ)
        Update θ using the optimizer and ∇_θ L(θ)
    end
end
```

for 1 **to** N_{epoch} **do**

  for *each minibatch of samples* (m_0, m_1) **do**

    $t \sim \mathcal{U}(0,1)$ `// Sample t`

    $m_0 \sim$ prior distribution

    $m_t \leftarrow t \cdot m_1 + (1 - t) \cdot m_0$

    Compute the target velocity: $u_t \leftarrow m_1 - m_0$

    Predict the velocity: $v_t \leftarrow \mathbf{v}(t, m_t, e, d)$

    Compute the loss: $\mathcal{L}(\theta) \leftarrow \mathbb{E}\left[(v_t - u_t)^2\right]$

    Compute gradients: $\nabla_\theta \mathcal{L}(\theta)$

    Update θ using the optimizer and $\nabla_\theta \mathcal{L}(\theta)$

  end

end

Simple non-linear model

$$d(e, m) = e^2 m^3 + m \exp(-|0.2 - e|) + \eta$$

- $m \sim \mathcal{U}[0, 1]$
- $e \sim \mathcal{U}[0, 1]$
- $\eta \sim \mathcal{N}(0, 10^{-4})$
- $d = f(m, e) + \eta$

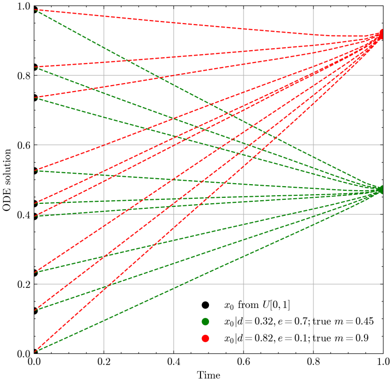


Figure: Generation paths of variable m conditional on different d, e from prior uniform distribution

Existing Methods comparison

Method	Base model	Exact likelihood estimation	No middle-man Training	Arbitrary number of observations
MDGM	VAE based on CNN	×	✓	×
MCGAN	MCMC + GAN	×	×	×
PI-INN	PI + flow-based model	✓	✓	×
CFM-Tr (ours)	CFM + Transformer	✓	✓	✓

Table: Comparison of methods for solving Bayeisan Inverse problems. *MDGM use the PDE solution as a holistic observation; the problem was not formulated as the recovery of the forward model from a small number of observations

SEIR disease model

The SEIR (Susceptible-Exposed-Infected-Removed) model is a mathematical model used to simulate the spread of infectious diseases [Koval et al., 2024]
We simulate a realistic scenario where we measure the several number of infected I and deceased R individuals at random times a and use this information to recover the control parameters of the ODE system m.

$$\frac{dS}{dt} = -\beta(t)SI, \frac{dE}{dt} = \beta(t)SI - \alpha E$$
$$\frac{dI}{dt} = \alpha E - \gamma(t)I, \frac{dR}{dt} = \gamma(t)I$$

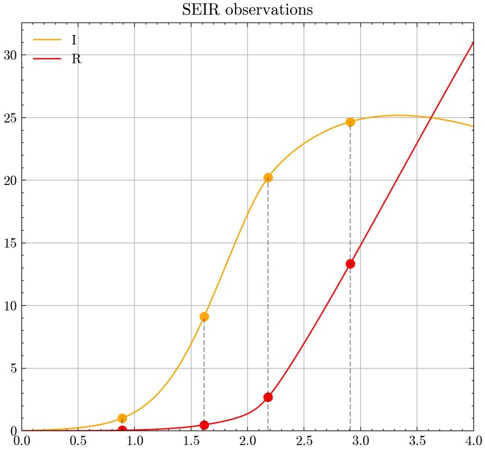
$$S(0) = 99,\ E(0) = 1,\ I(0) = R(0) = 0.$$

$$\beta(t) = \beta_1 + \frac{\tanh(7(t-\tau))}{2}(\beta_2 - \beta_1)$$
$$\gamma(t) = \gamma^r + \gamma^d(t)$$
$$\gamma^d(t) = \gamma_1^d + \frac{\tanh(7(t-\tau))}{2}(\gamma_2^d - \gamma_1^d)$$

SEIR disease model

We fix $\tau = 2.1$ over a time interval of $[0, 4]$.

- The experiment consists of choosing four time points $e = [a_1, a_2, a_3, a_4] \sim \mathcal{U}[1, 3]$
- $d_i = [I_{e_i}, R_{e_i}]$ for $i \in [1, 4]$ ($d \in \mathbf{R}^{2\times 4}$) is the number of infected and deceased individuals
- $\mathbf{m} = [\beta_1, \alpha, \gamma^r, \gamma_1^d, \beta_2, \gamma_2^d]$.



Method validation: SEIR disease model

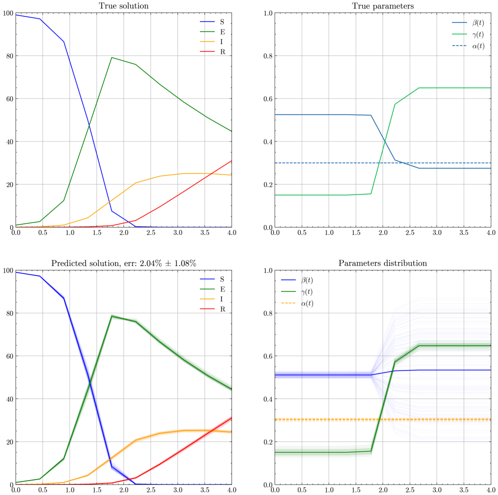


Figure: Probabilistic solutions to the SEIR inverse problem

Table: The relative inference error of the trained model for SEIR model

N	Relative Error
2	$10.88\% \pm 2.39\%$
3	$3.31\% \pm 1.47\%$
4	$2.80\% \pm 1.37\%$
5	$2.15\% \pm 0.99\%$
6	$1.97\% \pm 0.91\%$
7	$1.59\% \pm 0.75\%$
8	$1.48\% \pm 0.71\%$

Permeability field inversion

$$-\nabla \cdot (\kappa \nabla u) = 0$$

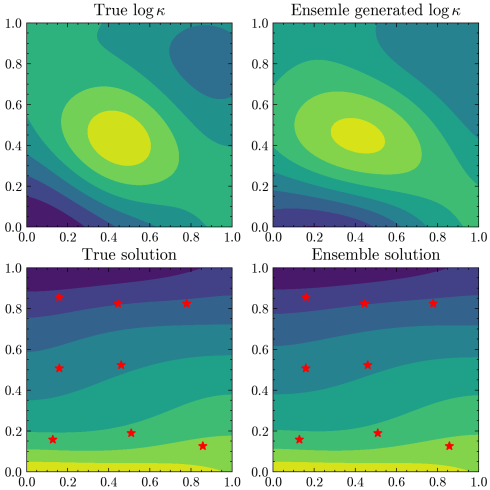
$$u(x=0,y) = \exp\left(-\frac{1}{2\sigma_w}(y-e_1)^2\right)$$

$$u(x=1,y) = -\exp\left(-\frac{1}{2\sigma_w}(y-e_2)^2\right)$$

$$m = \log(\kappa) \sim N(0, C_{pr})$$

$$m(x, \mathbf{m}) \approx \sum_{i=1}^{n_m} m_i \sqrt{\lambda_i}\phi_i(x)$$

e is a boundary condition parameters, d is a value of PDE solution with coordinates information, m is the KL expansion 16 modes.



Results

Table: The relative inference error of the trained model for two numerical experiments

N	SEIR Problem	Permeability Field
2	$10.88\% \pm 2.39\%$	$28.84\% \pm 3.43\%$
3	$3.31\% \pm 1.47\%$	$16.23\% \pm 1.53\%$
4	$2.80\% \pm 1.37\%$	$17.80\% \pm 1.99\%$
5	$2.15\% \pm 0.99\%$	$16.86\% \pm 1.76\%$
6	$1.97\% \pm 0.91\%$	$7.21\% \pm 1.26\%$
7	$1.59\% \pm 0.75\%$	$7.48\% \pm 1.23\%$
8	$1.48\% \pm 0.71\%$	$2.75\% \pm 0.60\%$

References

Koval, K., Herzog, R., and Scheichl, R. (2024).
Tractable optimal experimental design using transport maps.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. (2023).
Flow matching for generative modeling.

Lipman, Y., Havasi, M., Holderrieth, P., Shaul, N., Le, M., Karrer, B., Chen, R. T. Q., Lopez-Paz, D., Ben-Hamu, H., and Gat, I. (2024).
Flow matching guide and code.

Sherki, D., Oseledets, I., and Muravleva, E. (2025).
Bayesian inverse problems meet flow matching: Efficient and flexible inference via transformers.

Thx