STOCHASTIC FEW-STEP MODELS

Romeo Passaro[*1,2], **Zander W. Blasingame**[2], **Michael M. Bronstein**[1,2], **Alexander Tong**[2]
[1]Department of Computer Science, University of Oxford [2]AITHYRA

ABSTRACT

Reward alignment for flow and diffusion models, via test-time steering or fine-tuning, often relies on sampling from *conditional* distributions $p_{\tau|t}(\cdot \mid x_t)$ induced by stochastic dynamics. In practice, this creates a computational bottleneck requiring expensive SDE simulation. Meanwhile, recent few-step accelerations for generative flows largely target *deterministic* dynamics and therefore do not directly address stochastic conditional sampling. We introduce **stochastic few-step models**, a framework for fast sampling from SDE-defined conditional distributions by mapping the conditional SDE to a *deterministic ODE* with matching marginals. Building on this formulation, we show that the resulting conditional ODEs can be effectively *distilled* into a single few-step model, enabling efficient conditional rollouts. Experiments on Gaussian-mixture and MNIST steering show that the resulting sampler provides accurate conditional samples to improve reward steering, outperforming standard denoiser heuristics.

1 INTRODUCTION

Flow matching and diffusion models (Liu et al., 2022; Albergo et al., 2025; Song et al., 2021; Lipman et al., 2023; Ho et al., 2020) have revolutionized generative modelling, enabling high-quality sampling from complex data distributions. These methods transport samples, through an ODE or SDE, from an easy-to-sample distribution to the data distribution.

A powerful tool to improve the performance and controllability of a generative model is **reward alignment**. A successful example is classifier-free guidance (Ho & Salimans, 2022). A large body of recent work studies reward alignment for generative models (Singhal et al., 2025; Wallace et al., 2023; Domingo-Enrich et al., 2025). Informally, given a reward function $R : \mathbb{R}^d \to \mathbb{R}$, where d is the data dimension, we seek to adjust the model so that generated samples achieve higher reward without drifting too far from the base distribution. Importantly, the mathematical frameworks underlying such *tilting* are inherently stochastic, which typically necessitates expensive SDE simulation.

There has been strong interest in accelerating sampling for deterministic flow-based models (Lu & Song, 2025; Boffi et al., 2025; Geng et al., 2025). The idea is to train a neural network to approximate the solution map of an ODE, either via distillation from a teacher or by training it from scratch. However, these frameworks are developed for deterministic dynamics and therefore do not obviously apply to the stochastic conditional distributions required for reward alignment.

To improve diffusion sampling, Holderrieth et al. (2025) consider two time points $(\tau, t) \in [0,1]^2$ and construct a *virtual* flow-matching model whose terminal density matches the conditional $p_{\tau|t}(\cdot \mid \boldsymbol{x}_t)$, where the joint law of $(\boldsymbol{x}_\tau, \boldsymbol{x}_t)$ is induced by a diffusion model or, more generally, by non-Markovian trajectories in the style of Song et al. (2022). Building on this, Potaptchik et al. (2026) show that the collection of such conditional ODEs for $\tau = 0$ can be *distilled* into a single few-step model, and demonstrate applications to reward-aligned generation.

Notably, in Holderrieth et al. (2025) the conditional ODE is defined *separately* for each time pair (τ, t), and the resulting intermediate trajectories arise from *virtual* dynamics that are not tied to the underlying conditional diffusion dynamics. Therefore, the intermediate *virtual* states along these trajectories can lack a direct interpretation; moreover, a model that aims to distil the entirety of the conditional distributions would require distilling an ODE for every (τ, t).

[*]Corresponding author: `romeo.passaro@stcatz.ox.ac.uk`

In contrast, we propose a formulation that works directly with the conditional dynamics and aims to capture the full family of conditionals indexed by $(\boldsymbol{x}_t, t)$ within a single dynamic, enabling more systematic exploration and distillation. Concretely, our contribution in this work is twofold:

- We propose a new theoretically grounded way to translate from SDE to ODE sampling, without necessitating *virtual* flow-matching models.
- We demonstrate empirically that flow maps can distil this new family of conditional ODEs, and that this improves **reward alignment**.

This matters for two independent reasons: (i) distilled conditional rollouts lower the computational cost of reward-aligned sampling; and (ii) using the conditional dynamics avoids auxiliary *virtual* processes and reduces the number of separate ODEs that must be learned.

2 BACKGROUND

In this section, we briefly review flow matching and diffusion models through the lens of stochastic interpolants (Albergo et al., 2025), focusing on a special case relevant to our setting and the translation between the two formalisms. We also summarize MeanFlows (Geng et al., 2025), a representative few-step generative model that we use in our experiments.

2.1 ODE/SDE

Given the base distribution $p_1 := \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and a target distribution p_0, we couple endpoints

$$\boldsymbol{x}_0 \sim p_0, \qquad \boldsymbol{x}_1 \sim p_1, \qquad \boldsymbol{x}_0 \perp \boldsymbol{x}_1, \tag{1}$$

and define the interpolating path

$$I_\tau \;:=\; (1-\tau)\boldsymbol{x}_0 \;+\; \tau \boldsymbol{x}_1, \qquad \tau \in [0,1]. \tag{2}$$

We denote by p_t the law of I_t. Consider the flow-matching ODE

$$\frac{d\boldsymbol{x}_\tau}{d\tau} = \mathbb{E}_{(x_0,x_1)\sim p_0\otimes p_1}\left[\boldsymbol{x}_1 - \boldsymbol{x}_0 \mid I_\tau\right] := \boldsymbol{v}_\tau(\boldsymbol{x}_\tau), \quad \boldsymbol{x}_0 \sim p_0 \tag{3}$$

whose solution satisfies $\boldsymbol{x}_\tau \sim p_\tau$. In particular, by simulating the reverse-time ODE starting from $\boldsymbol{x}_1$, we can generate samples from $\boldsymbol{x}_0$. Moreover, starting from the continuity equation associated with eq. (3), one can derive an SDE that shares the same marginal distributions as eq. (3):

$$d\boldsymbol{x}_\tau = -\frac{\boldsymbol{x}_\tau}{1-\tau}d\tau + \sqrt{\frac{2\tau}{1-\tau}}d\boldsymbol{w}_\tau, \quad \boldsymbol{x}_0 \sim p_0, \tag{4}$$

where $\{\boldsymbol{w}_\tau\}_{\tau\in[0,1]}$ is the standard Brownian motion. Note that eq. (4) satisfies $\boldsymbol{x}_1 \mid \boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ (Domingo-Enrich et al., 2025). The reverse SDE is

$$d\boldsymbol{x}_\tau = \left(-\frac{\boldsymbol{x}_\tau}{1-\tau} - \frac{2\tau}{1-\tau}\nabla_x \log p_\tau(\boldsymbol{x}_\tau)\right)d\tau + \sqrt{\frac{2\tau}{1-\tau}}d\boldsymbol{w}_\tau, \quad \boldsymbol{x}_1 \sim p_1, \tag{5}$$

where, with a slight abuse of notation, $\{\boldsymbol{w}_\tau\}_{\tau\in[0,1]}$ denotes a standard Brownian motion *reverse-time*. Therefore, integrating eq. (5) from $t = 1$ to $t = 0$ yields samples $\boldsymbol{x}_0 \sim p_0$.

The score $\nabla_x \log p_\tau$ admits a convenient expression via Tweedie's identity:

$$\nabla_x \log p_\tau(\boldsymbol{x}) = \frac{(1-\tau)\,\mathbb{E}[\boldsymbol{x}_0 \mid I_\tau = \boldsymbol{x}] - \boldsymbol{x}}{\tau^2}. \tag{6}$$

In particular, since $\boldsymbol{v}_\tau(\boldsymbol{x}) := \mathbb{E}[\boldsymbol{x}_1 - \boldsymbol{x}_0 \mid I_\tau = \boldsymbol{x}]$, the score and the flow-matching velocity are related by

$$\nabla_x \log p_\tau(\boldsymbol{x}) = -\frac{\boldsymbol{x} + (1-\tau)\,\boldsymbol{v}_\tau(\boldsymbol{x})}{\tau}. \tag{7}$$

Hence, it suffices to learn either the velocity field $\boldsymbol{v}_\tau$ or the score $\nabla_x \log p_\tau$: the other is obtained by a simple closed-form linear transformation, and can be plugged directly into the corresponding ODE/SDE drift.

For example, to learn $\boldsymbol{v}_\tau(\boldsymbol{x})$ with a neural network $f(\boldsymbol{x}, \tau; \theta)$, we can minimize

$$\mathcal{L}(\theta) = \mathbb{E}_{(x_0,x_1)\sim p_0\otimes p_1,\ \tau\sim U[0,1]}\left[\|(\boldsymbol{x}_1 - \boldsymbol{x}_0) - f(I_\tau, \tau; \theta)\|_2^2\right]. \tag{8}$$

Analogously, one may train a denoiser/score model directly using a standard denoising objective.

2.2 Few-step models

The acceleration of few-step generative models has largely been driven by the concept of *consistency*. Given the ODE we ultimately want to learn for sampling, we can avoid directly parameterizing the instantaneous velocity field and instead learn an affine quantity, such as the flow map (Boffi et al., 2025) or the average velocity field along a trajectory (Geng et al., 2025). In the absence of direct estimators for these functions, we learn them by imposing an implicit constraint on the neural network, namely *consistency*. Many of these methods are reminiscent of physics-informed neural networks (Raissi et al., 2017), with the key difference that, in a purely temporal setting, the direction of the signal is clear.

For example, MeanFlow (Geng et al., 2025) targets the average velocity field of eq. (3) between two time steps (r, t) along the trajectory. The mean velocity is defined as

$$(t-r)\,\boldsymbol{u}(\boldsymbol{x}_t, t, r) := \int_r^t \boldsymbol{v}_\tau(\boldsymbol{x}_\tau)\,d\tau, \qquad d\boldsymbol{x}_\tau = \boldsymbol{v}_\tau(\boldsymbol{x}_\tau)\,d\tau. \tag{9}$$

Differentiating eq. (9) yields

$$\boldsymbol{u}(\boldsymbol{x}_t, t, r) = \boldsymbol{v}_t(\boldsymbol{x}_t) - (t-r)\,\frac{d\,\boldsymbol{u}(\boldsymbol{x}_t, t, r)}{dt}. \tag{10}$$

We enforce this relation with the loss

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim U[0,1],\,\boldsymbol{x}_t\sim p_t}\left[\left(\boldsymbol{u}(\boldsymbol{x}_t, t, r;\theta) - \mathrm{sg}(\boldsymbol{u}_{\mathrm{tgt}})\right)^2\right], \tag{11}$$

$$\boldsymbol{u}_{\mathrm{tgt}} = \boldsymbol{v}_t(\boldsymbol{x}_t, t) - (t-r)\,\frac{d\,\boldsymbol{u}(\boldsymbol{x}_t, t, r;\theta)}{dt}, \tag{12}$$

where $\mathrm{sg}(\cdot)$ denotes *stop-gradient*. We use $\mathrm{sg}(\cdot)$ so the learning signal comes from $\boldsymbol{v}_t$ and updates $\boldsymbol{u}_\theta$. In practice, $\boldsymbol{v}_t$ is provided by a pre-trained flow-matching model, or by an unbiased estimator of the velocity field, such as $\boldsymbol{x}_1 - \boldsymbol{x}_0$. In practice, we pick a fraction of training instances for which we set $t = r$, thereby enforcing a boundary condition on the average velocity.

3 Methods

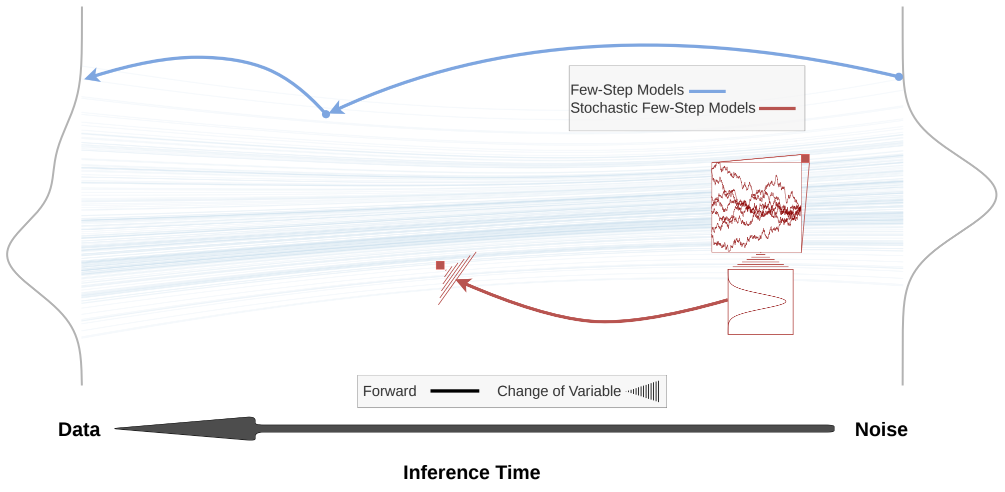


Figure 1: Schematic comparison of transition mechanisms. Flow maps (Blue): jump between arbitrary time steps. Ours (Red): reparameterize the *conditional ODEs* and distil them into a few-step model to sample conditional marginals at **any** intermediate time.

3.1 Conditional ODE

Let us consider eq. (5) initialized at $(\boldsymbol{x}, t)$:

$$d\boldsymbol{x}_\tau = \left(-\frac{\boldsymbol{x}_\tau}{1-\tau} - \frac{2\tau}{1-\tau}\nabla_x \log p_\tau(\boldsymbol{x}_\tau)\right) d\tau + \sqrt{\frac{2\tau}{1-\tau}} d\boldsymbol{w}_\tau, \quad \boldsymbol{x}_t = \boldsymbol{x}. \tag{13}$$

In particular this SDE has marginals $p_{\tau|t}\left(\cdot \mid \boldsymbol{x}\right)$. Unfortunately, the dynamics are stochastic, making them computationally expensive to simulate. Surprisingly, the score(s) of eq. (13) can be written in closed form via Bayes' theorem:

$$\nabla_x \log p_{\tau|t}\left(\boldsymbol{x} \mid \boldsymbol{x}_t\right) = \nabla_x \log p_\tau\left(\boldsymbol{x}\right) + \nabla_x \log p_{t|\tau}\left(\boldsymbol{x}_t \mid \boldsymbol{x}\right). \tag{14}$$

The latter term is expressible in terms of the known corruption kernel of eq. (4). More generally, for linear SDEs the transition kernel is available analytically; for example, in our case it corresponds to the DDPM SDE (Ho et al., 2020). Following the same route for bridging SDEs and ODEs as in Albergo et al. (2025), we obtain the following result.

Proposition 1. *Let $\boldsymbol{x}_t \mid \boldsymbol{x}_\tau \sim \mathcal{N}\left(\mu(t,\tau)\boldsymbol{x}_\tau, \sigma^2\left(t,\tau\right)\boldsymbol{I}\right)$ denote the transition kernel of eq.* (4) *with $t \geq \tau$ (see section A.1). Then, for any $\delta > 0$, the ODE*

$$d\boldsymbol{x}_\tau = \left(\boldsymbol{v}_\tau\left(\boldsymbol{x}_\tau\right) + \frac{\tau}{1-\tau}\mu(t,\tau)\frac{\boldsymbol{x} - \mu(t,\tau)\boldsymbol{x}_\tau}{\sigma^2(t,\tau)}\right) d\tau, \qquad \boldsymbol{x}_{t-\delta} \sim p_{t-\delta|t}\left(\cdot \mid \boldsymbol{x}\right) \tag{15}$$

has marginals $p_{\tau|t}(\cdot \mid \boldsymbol{x})$ for all $\tau \leq t-\delta$ (see section A.2).

Remarkably, there exists a *deterministic* dynamics whose marginals match $p_{\tau|t}(\cdot \mid \boldsymbol{x})$, which is much cheaper to simulate. Since the dynamics are deterministic, they cannot be initialized exactly at $(\boldsymbol{x}, t)$; instead, as stated in **Proposition** 1, they must start from $\boldsymbol{x}_{t-\delta} \sim p_{t-\delta|t}(\cdot \mid \boldsymbol{x})$. This leads to the efficient **Algorithm** 1 for sampling from $p_{\tau|t}(\cdot \mid \boldsymbol{x})$.

Algorithm 1: Conditional transition sampling via conditional ODE

```
def Init($\boldsymbol{x}, t, \delta, \hat{\boldsymbol{s}}$):
    $\xi \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
    $\sigma \leftarrow \sqrt{\frac{2t}{1-t}}$
    $\boldsymbol{b} \leftarrow -\frac{\boldsymbol{x}}{1-t} - \frac{2t}{1-t}\,\hat{\boldsymbol{s}}_t(\boldsymbol{x})$
    return $\boldsymbol{x} - \delta\,\boldsymbol{b} + \sqrt{\delta}\,\sigma\,\xi$
def $\boldsymbol{v}\,(\boldsymbol{x}_\tau, \tau \mid \boldsymbol{x}, t)$:
    $\boldsymbol{v} \leftarrow \hat{\boldsymbol{v}}_\tau(\boldsymbol{x}_\tau)$
    $\boldsymbol{c} \leftarrow \frac{\tau}{1-\tau}\,\mu(t,\tau)\,\frac{\boldsymbol{x} - \mu(t,\tau)\,\boldsymbol{x}_\tau}{\sigma^2(t,\tau)}$
    return $\boldsymbol{v} + \boldsymbol{c}$

Input: $t, r, \boldsymbol{x}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{s}}, \mu, \sigma, M, \delta$
Output: $\boldsymbol{x}_r \sim p_{r|t}(\cdot \mid \boldsymbol{x})$
$\tau \leftarrow t - \delta$
$\boldsymbol{x}_\tau \leftarrow \mathsf{Init}(\boldsymbol{x}, t, \delta, \hat{\boldsymbol{s}})$
$h \leftarrow \frac{(t-\delta)-r}{M}$
for $m = 0, \ldots, M-1$ do
    $\boldsymbol{v} \leftarrow \boldsymbol{v}(\boldsymbol{x}_\tau, \tau \mid \boldsymbol{x}, t)$
    $\boldsymbol{x}_\tau \leftarrow \boldsymbol{x}_\tau - h\,\boldsymbol{v}$
    $\tau \leftarrow \tau - h$
end
return $\boldsymbol{x}_\tau$
```

In words, we obtain a sample from $p_{t-\delta|t}$ by simulating the SDE for ϵ time (one step), and then integrating the deterministic dynamics.

While this may appear unusual, it is inevitable: conditioning on a fixed starting point removes randomness, so some stochasticity must be reintroduced to sample from the conditional distribution. This is analogous to Holderrieth et al. (2025), where transition sampling is realized by introducing a new Gaussian base distribution and transporting it through a virtual flow.

We can further generalize **Algorithm** 1 by modularizing it. To sample from $p_{r|t}(\cdot \mid \boldsymbol{x})$, choose (adaptively) an intermediate time grid

$$r = r_0 < r_1 < \cdots < r_n = t, \tag{16}$$

and apply **Algorithm** 1 successively on each interval (r_k, r_{k+1}). In this formulation, one can explicitly control how much noise is injected in total.

Moreover, we can freely alternate between **Algorithm** 1 and restarting it. This is possible because each step of **Algorithm** 1 produces a sample from the correct conditional distribution $p_{\tau|t}(\cdot \mid \boldsymbol{x})$.

3.2 Reparameterization trick

A closer look at eq. (15) shows that the velocity field blows up as $\tau \to t$, reflecting the fact that the dynamics are forced to hit the boundary condition at time t. Motivated by the corresponding SDE, we introduce the rescaled variable

$$\boldsymbol{z}_\tau = \frac{\boldsymbol{x} - \boldsymbol{x}_\tau}{\sigma(t, \tau)}. \tag{17}$$

Unlike $\boldsymbol{x}_\tau$, this scaled variable does **not** converge to a deterministic limit as $\tau \to t$; instead, it converges in distribution to $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Moreover, $\boldsymbol{z}_\tau$ satisfies the ODE

$$d\boldsymbol{z}_\tau = -\frac{1}{\sigma(t,\tau)} \left(\boldsymbol{v}_\tau \left(\boldsymbol{x} - \sigma(t,\tau) \boldsymbol{z}_\tau \right) + \frac{\tau}{1-\tau} \frac{\mu(t,\tau)\left(1-\mu(t,\tau)\right)}{\sigma^2(t,\tau)} \boldsymbol{x} \right). \tag{18}$$

Although the drift in eq. (18) is singular as $\tau \to t$, the singularity is integrable.[1] This observation is formalized in the following proposition.

Proposition 2. *The scaled variable $\boldsymbol{z}_\tau \Rightarrow \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ as $\tau \to t < 1$. Under mild regularity conditions, eq.* (18) *is well-defined for $0 \leq \tau \leq t < 1$, and the inverse transformation $\boldsymbol{x} - \sigma(t,\tau)\boldsymbol{z}_\tau$ (from eq.* (17)*) is distributed according to $p_{\tau|t}(\cdot \mid \boldsymbol{x})$ (see sections A.3 and B.2).*

As a consequence, we can sample from the conditional marginals of eq. (13) by integrating eq. (18) and then mapping back via eq. (17), as summarized in **Algorithm 2**.

Algorithm 2: Conditional transition sampling via reparameterized ODE

```
def z2x($\boldsymbol{z}_\tau, \tau \mid \boldsymbol{x}_t, t$):
    $\boldsymbol{x}_\tau \leftarrow \boldsymbol{x}_t - \sigma(t,\tau)\, \boldsymbol{z}_\tau$
    return $\boldsymbol{x}_\tau$

def DriftZ($\boldsymbol{z}_\tau, \tau \mid \boldsymbol{x}_t, t$):
    $\boldsymbol{x}_\tau \leftarrow$ z2x($\boldsymbol{z}_\tau, \tau \mid \boldsymbol{x}_t, t$)
    $\boldsymbol{v} \leftarrow \hat{\boldsymbol{v}}_\tau(\boldsymbol{x}_\tau)$
    $\boldsymbol{c} \leftarrow \frac{\tau}{1-\tau} \frac{\mu(t,\tau)(1-\mu(t,\tau))}{\sigma^2(t,\tau)} \boldsymbol{x}_t$
    return $-\frac{1}{\sigma(t,\tau)} (\boldsymbol{v} + \boldsymbol{c})$

Input: $t, r, \boldsymbol{x}_t, \hat{\boldsymbol{v}}, \mu, \sigma, M$
Output: $\boldsymbol{x}_r \sim p_{r|t}(\cdot \mid \boldsymbol{x}_t)$
$\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
$h \leftarrow \frac{(t-r)}{M}$
$\tau \leftarrow t$
for $m = 0, \ldots, M-1$ do
    $\boldsymbol{z}_\tau \leftarrow \boldsymbol{z}_\tau - h$ DriftZ($\boldsymbol{z}_\tau, \tau \mid \boldsymbol{x}_t, t$)
    $\tau \leftarrow \tau - h$
end
$\boldsymbol{x}_r \leftarrow$ z2x($\boldsymbol{z}_r, r \mid \boldsymbol{x}_t, t$)
return $\boldsymbol{x}_r$
```

In our reparameterization, randomness is *intrinsic* to the conditioning point, indeed for a fixed conditioning time t, the same deterministic dynamics yields samples from $p_{\tau|t}(\cdot \mid \boldsymbol{x}_t)$ for *any* intermediate $\tau < t$ by simply stopping at τ. Note that **Algorithm** 2 can be modularized as **Algorithm** 1.

By contrast, Holderrieth et al. (2025) sample $p_{t'|t}(\cdot \mid \boldsymbol{x}_t)$ by constructing an *inner* flow matching model for the desired transition kernel, which has no direct connection with the *conditional* underlying dynamic.

Note that the particular reparameterization we use is arbitrary. Alternative parameterizations of eq. (15) can be chosen so that the resulting boundary condition is stochastic instead of deterministic. In this sense, Holderrieth et al. (2025) may be interpreted as adopting the parameterization induced by a sufficient statistic.

3.3 Stochastic Few-step Models

Our reparameterization produces a tractable deterministic dynamics; the resulting ODE (eq. (18)) is amenable to few-step distillation. In our experiments, we adopt the MeanFlow formulation (Geng et al., 2025). Specifically, we learn a conditional few-step model

$$f(\boldsymbol{z}_\tau, \tau \to r \mid \boldsymbol{x}_t, t; \theta) \tag{19}$$

[1] In particular, the ODE can be simulated via a change of variables such as $u^2 = t - \tau$.

which takes as input the conditioning pair $(\boldsymbol{x}_t, t)$ and the current state $(\boldsymbol{z}_\tau, \tau)$, and predicts the *average* velocity of eq. (18) over the interval $[\tau, r]$. This can be viewed as a standard MeanFlow model augmented with conditioning information that specifies the particular conditional ODE. Importantly, this conditioning mechanism is agnostic to the choice of few-step objective: f_θ can be trained with any consistency-style loss, either from scratch or via distillation.

Since $\tau < t$, in the from scratch setting we first sample $\boldsymbol{x}_\tau$ and then $\boldsymbol{x}_t$. As a result, the usual flow matching construction yields an unbiased estimator of the unconditional velocity field at $\boldsymbol{x}_\tau$, as shown in **Algorithm** 3.

Algorithm 3: Training (From Data)

```
Input: initial $\theta$, $f_\theta$
Output: trained $\theta$
while not converged do
    $t, \tau, r \sim \texttt{Sample}(t, \tau, r)$
    $\boldsymbol{x}_0 \sim p_0$
    $\xi \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
    $\boldsymbol{x}_\tau \leftarrow (1-\tau)\,\boldsymbol{x}_0 + \tau\xi$
    $\eta \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
    $\boldsymbol{x}_t \leftarrow \mu(t,\tau)\,\boldsymbol{x}_\tau + \sigma(t,\tau)\,\eta$
    $\boldsymbol{z}_\tau \leftarrow \frac{\boldsymbol{x}_t - \boldsymbol{x}_\tau}{\sigma(t,\tau)}$
    $\hat{\boldsymbol{v}} \leftarrow \texttt{ConditionalV}(\xi - \boldsymbol{x}_0, \boldsymbol{x}_t, \tau, t)$
    $\mathcal{L} \leftarrow \texttt{MFLoss}(\hat{\boldsymbol{v}}, f_\theta(\boldsymbol{z}_\tau, \tau \to r \mid \boldsymbol{x}_t, t))$
    $\theta \leftarrow \theta - \nabla_\theta \mathcal{L}$
end
return $\theta$
```

Algorithm 4: Sampling

```
Input: Outer steps $M$, Inner steps $N$, model
    $f_\theta$, start time $s_{\text{start}}$, $s_{\text{end}}$, start point $\boldsymbol{x}_{s_{\text{start}}}$,
    inner and outer schedules $s_i, t_{i,j}$'s
Output: $\boldsymbol{x}_{s_{\text{end}}}$
for $i = 0, \ldots, M-1$ do
    $s \leftarrow s_i$
    $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
    for $j = 0, \ldots, N-1$ do
        $t \leftarrow t_{i,j}$
        $r \leftarrow t_{i,j+1}$
        $\boldsymbol{z} \leftarrow \boldsymbol{z} - (t-r)\, f_\theta(\boldsymbol{z}, t, r \mid \boldsymbol{x}_s, s)$
    end
    $\boldsymbol{x}_{s_{i+1}} \leftarrow \text{z2x}(\boldsymbol{z}, s_{i+1} \mid \boldsymbol{x}_s, s)$
end
return $\boldsymbol{x}_{s_{\text{end}}}$
```

4 Reward Alignment

In this section, we briefly review how reward alignment is performed in flow models at test-time, and how it can be improved with stochastic few-step samplers. We start by stating an equivalence between several standard formulations of reward-tilted path measures.

Theorem 3 (Reward equivalence)**.** *Consider the path measure $\mathbb{P}$ of eq.* (5)*, so that $\mathbb{P}_0 \overset{d}{=} p_1 \overset{d}{=} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $\mathbb{P}_1 \overset{d}{=} p_0$. Let $(\boldsymbol{z}_\tau)_{\tau\in[0,1]}$ denote the corresponding process. Then the path measure $\mathbb{Q}$*[2] *can be described in the following equivalent ways:*

Tilted: $\dfrac{d\mathbb{Q}}{d\mathbb{P}} \propto \exp(R(\boldsymbol{z}_1))$*, where R is a general reward function. In particular* $\dfrac{d\mathbb{Q}_\tau}{d\mathbb{P}_\tau} \propto \mathbb{E}[\exp R(\boldsymbol{z}_1) \mid \boldsymbol{z}_\tau]$.

RL/SOC: $\mathbb{Q} = \arg\max_{\mathbb{S}} \mathbb{E}_{\boldsymbol{z}_1 \sim \mathbb{S}_1}[R(\boldsymbol{z}_1)] - D_{\mathrm{KL}}(\mathbb{S} \| \mathbb{P})$.

Schrödinger problem: $\mathbb{Q} = \arg\min_{\mathbb{S}} D_{\mathrm{KL}}(\mathbb{S}\|\mathbb{P})$ *such that* $\mathbb{S}_0 = \mathbb{P}_0$ *and* $\dfrac{d\mathbb{S}_1}{d\mathbb{P}_1} \propto \exp(R(\boldsymbol{z}_1))$

Double Reverse: *Consider the path measure induced by eq.* (4)*, where the starting distribution is reweighted as $\propto \exp(R(\boldsymbol{x}_0))$. Then the time-reversal of this path measure is $\mathbb{Q}$.*

[2]In general this equivalence can fail when $\mathbb{P}$ couples $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$. Here $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ are independent under $\mathbb{P}$, so the equivalence is valid.

We refer the reader to Domingo-Enrich et al. (2025); Denker et al. (2025) for a proof. Using the **Double Reverse** characterization, we can represent $\mathbb{Q}$, the tilted measure, as

$$d\boldsymbol{x}_\tau = \Big(-\frac{\boldsymbol{x}_\tau}{1-\tau} - \frac{2\tau}{1-\tau}\nabla_x \log p_\tau\left(\boldsymbol{x}_\tau\right) - \frac{2\tau}{1-\tau}\nabla_x V_\tau\left(\boldsymbol{x}_\tau\right)\Big) d\tau + \sqrt{\frac{2\tau}{1-\tau}} d\boldsymbol{w}_\tau, \tag{20}$$

$$\text{where} \quad V_\tau(\boldsymbol{x}) \coloneqq \log \mathbb{E}\left[\exp\left(R(\boldsymbol{x}_0)\right) \mid \boldsymbol{x}_\tau = \boldsymbol{x}\right]. \tag{21}$$

We will refer to V_t as the value function.

4.1 Test-time

To sample from the terminally reward-tilted distribution, we can employ sequential Monte Carlo (SMC) techniques (Singhal et al., 2025). In practice, the intermediate potentials implied by theorem 3 are not available in closed form. Alternatively, one may sample by directly simulating the reward-tilted diffusion in eq. (20). In either case, the missing term is an estimate of $V_t(\boldsymbol{x})$ or $\nabla_x V_t(\boldsymbol{x})$: existing approaches typically rely on heuristics (Chung et al., 2024; Wu et al., 2024). In contrast, our *stochastic few-step* sampler enables repeated cheap rollouts $\boldsymbol{x}_t \to \boldsymbol{x}_0$, yielding approximate samples $\hat{\boldsymbol{x}}_0^{(i)} \sim p_{0|t}(\cdot \mid \boldsymbol{x}_t = \boldsymbol{x})$ that we use to estimate $V_t(\boldsymbol{x})$ and its gradient (Potaptchik et al., 2026) (see section A.4 for derivation)

$$V_t \approx \log \frac{1}{N}\sum_{i=1}^{N} \exp\left(R\left(\hat{\boldsymbol{x}}_0^{(i)}\right)\right) \tag{22}$$

$$\nabla_x V_t \approx \nabla_x \log \frac{1}{N}\sum_{i=1}^{N} \exp(R\left(\hat{\boldsymbol{x}}_0^{(i)}\right)) \tag{23}$$

$$\nabla_x V_t\left(\boldsymbol{x}\right) \approx \frac{1-t}{t^2}\left(\frac{\sum_{i=1}^N \exp\left(R\left(\hat{\boldsymbol{x}}_0^{(i)}\right)\right)\hat{\boldsymbol{x}}_0^{(i)}}{\sum_{i=1}^N \exp\left(R\left(\hat{\boldsymbol{x}}_0^{(i)}\right)\right)} - \frac{1}{N}\sum_{i=1}^{N}\hat{\boldsymbol{x}}_0^{(i)}\right) \tag{24}$$

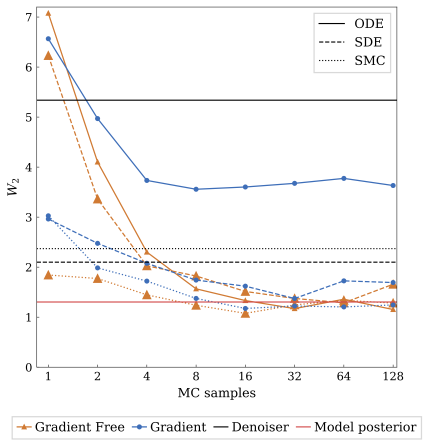
5 Experiments

In the following experiments, we evaluate our model's ability to approximate the posterior distribution and improve the reward-related estimators eqs. (22) to (24). Moreover, our stochastic few-step model yields an estimator of the *unconditional* velocity field (see section B.1), which we use to simulate the ODE and SDE dynamics in our experiments. Unless stated otherwise, the stochastic few-step model is run with 2 NFE, while the ODE/SDE/SMC baselines use Euler discretizations with 100 steps.

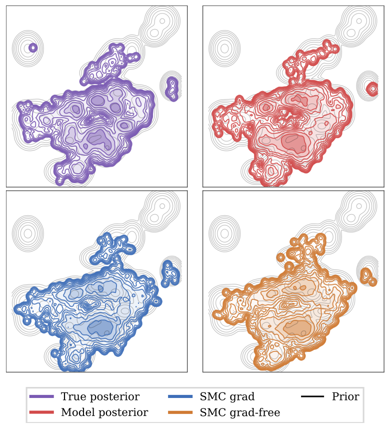
5.1 GMM

We train, using **Algorithm** 3, a small model on a mixture of 40 Gaussians. The task is posterior sampling from $p(\boldsymbol{x}_0 \mid \boldsymbol{x}_t)$ given a fixed noised observation at $t = 0.4$ (where $t = 0.4$ has been chosen randomly); the reward is the corresponding likelihood $r(\boldsymbol{x}_0) = \log p(\boldsymbol{x}_\tau \mid \boldsymbol{x}_0)$, which is available in closed form. We adapt Chung et al. (2024) and Wu et al. (2024) by replacing denoiser-based quantities with our estimators eqs. (22) to (24). This is a fully controlled setting in which the reward is exact and the posterior is tractable, allowing us to identify the quality of the posterior samples generate by our model.

Increasing the Monte Carlo budget in eqs. (22) to (24) consistently improves posterior-tilting performance (lower W_2) across ODE/SDE and SMC variants, and outperforms the denoiser-based estimate (see fig. 2a). Even 2–4 samples yield noticeable gains. The *direct* model-implied posterior is already a strong baseline; only after 16 Monte Carlo samples, the steered SDE/SMC methods match its performance. Qualitatively, the learned model captures the posterior geometry and modes well (see fig. 2b). Additional results are reported in section C.1.



(a) W_2 of model(s) against true posterior



(b) Posterior contours

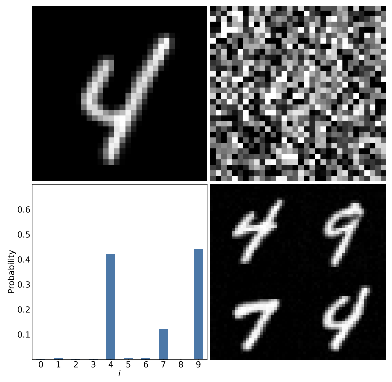
Figure 2

5.2 MNIST

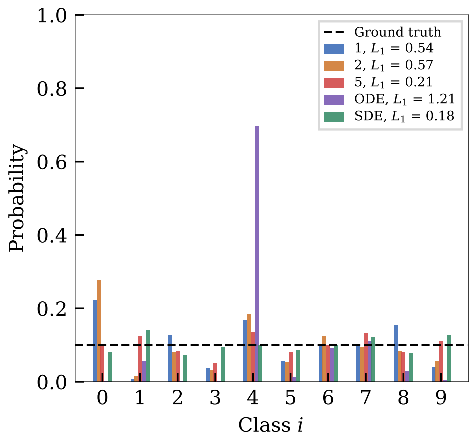
We train a 5M parameter unconditional DiT on MNIST using **Algorithm** 3. All the probabilities are computed using a small CNN classifier.

5.2.1 DIVERSITY

The number of *outer* steps controls the amount of stochasticity in the rollout. In the reference process eq. (5), we have $\boldsymbol{x}_1 \perp \boldsymbol{x}_0$, so rollouts started from a fixed $\boldsymbol{x}_1$ should, in principle, reproduce the unconditional class frequencies. Empirically, with few outer steps the outcome still depends strongly on the $\boldsymbol{x}_1$, because the boundary condition at $\tau = t$ is hard-coded (the map returns $\boldsymbol{x}_t$). As shown in fig. 3b, increasing the outer-step budget reduces this dependence and moves the sampler toward the correct equilibrium. SDE rollouts mix across classes, in contrast, the ODE rollout *should* be deterministic given $\boldsymbol{x}_1$; the small spread we observe is due to the stochastic denoiser used in our implementation (see section B.1).

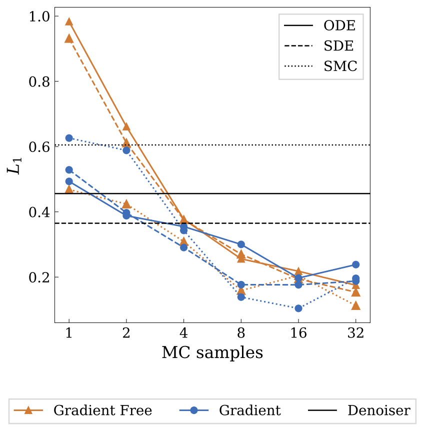


(a) Clean image (top-left) and its noisy version at $t = 0.75$ (top-right), with unconditional rollouts (bottom-right) and mean class probabilities over 1000 samples (bottom-left).

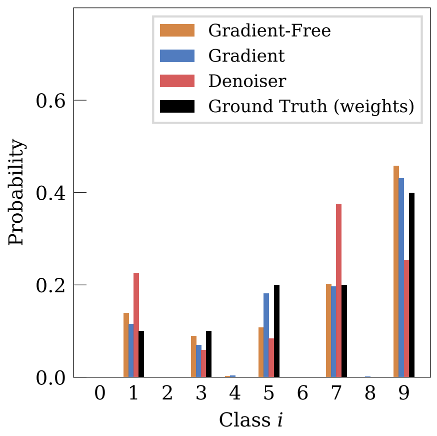


(b) Empirical class probabilities from 1000 rollouts started from the same initial noise seed $\boldsymbol{x}_1$.

Figure 3



(a) L_1 distance between the empirical class distribution and the ground-truth distribution.



(b) Empirical class probabilities under the sampler with 32 MC samples.

Figure 4

5.2.2 STEERING

We now repeat the same posterior-tilting setup of GMM on MNIST, following Chung et al. (2024); Wu et al. (2024). We use the CNN, with logits $p_\phi(c \mid \boldsymbol{x})$, to define a reward via a mixture of class probabilities,

$$\exp\big(r(\boldsymbol{x})\big) = p_\phi(c_{\text{mix}} \mid \boldsymbol{x}) \coloneqq \sum_{c=0}^{9} w_i \, p_\phi(i \mid \boldsymbol{x}), \qquad \sum_{i=0}^{9} w_i = 1, \quad w_i \geq 0. \tag{25}$$

This choice corresponds to standard classifier guidance As shown in fig. 4, increasing the number of Monte Carlo samples in our guidance estimator consistently steers the sampler toward the desired class distribution (lower L_1 error), whereas the standard one-step denoiser guidance is miscalibrated under our strong tilt and oversteer in this regime.

6 CONCLUSION

In this work we introduced **stochastic few-step models**, a framework for few-step sampling from *SDE-defined conditional distributions* by distilling a family of conditional ODEs that matches the respective distributions. By working directly with the underlying conditional process, rather than introducing separate *virtual* dynamics, we obtain a unified distillation target across conditioning times. In our experiments, the resulting samplers produce accurate conditional samples and improve **reward steering**, substantially reducing the cost of conditional rollouts and making reward-aligned sampling practical in regimes where repeated conditional simulations would otherwise be the dominant computational bottleneck.

Future work will evaluate the method on more complex datasets. We also plan to integrate it into a **self-distillation** pipeline, where reward-steered rollouts provide targets for training a single few-step tilted stochastic model end-to-end. Finally, we aim to explore alternative parameterizations of eq. (15) and the corresponding network parameterizations, with the goal of identifying formulations that are most stable for distillation; we believe that choosing the right parameterization will be key to scaling this setup.

REFERENCES

Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions, 2025. URL https://arxiv.org/abs/2303.08797.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models, 2025. URL https://arxiv.org/abs/2406.07507.

Hyungjin Chung, Jeongsol Kim, Michael T. Mccann, Marc L. Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems, 2024. URL https://arxiv.org/abs/2209.14687.

Alexander Denker, Francisco Vargas, Shreyas Padhy, Kieran Didi, Simon Mathis, Vincent Dutordoir, Riccardo Barbano, Emile Mathieu, Urszula Julia Komorowska, and Pietro Lio. Deft: Efficient fine-tuning of diffusion models by learning the generalised h-transform, 2025. URL https://arxiv.org/abs/2406.01781.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control, 2025. URL https://arxiv.org/abs/2409.08861.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling, 2025. URL https://arxiv.org/abs/2505.13447.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Peter Holderrieth, Uriel Singer, Tommi Jaakkola, Ricky T. Q. Chen, Yaron Lipman, and Brian Karrer. Glass flows: Transition sampling for alignment of flow and diffusion models, 2025. URL https://arxiv.org/abs/2509.25170.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022. URL https://arxiv.org/abs/2209.03003.

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models, 2025. URL https://arxiv.org/abs/2410.11081.

Peter Potaptchik, Adhi Saravanan, Abbas Mammadov, Alvaro Prat, Michael S. Albergo, and Yee Whye Teh. Meta flow maps enable scalable reward alignment, 2026. URL https://arxiv.org/abs/2601.14430.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, 2017. URL https://arxiv.org/abs/1711.10561.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models, 2025. URL https://arxiv.org/abs/2501.06848.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021. URL https://arxiv.org/abs/2011.13456.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization, 2023. URL `https://arxiv.org/abs/2311.12908`.

Luhuan Wu, Brian L. Trippe, Christian A. Naesseth, David M. Blei, and John P. Cunningham. Practical and asymptotically exact conditional sampling in diffusion models, 2024. URL `https://arxiv.org/abs/2306.17775`.

APPENDICES

A Derivations

A.1 Corruption Kernel

Consider the SDE in eq. (4) starting from $(\boldsymbol{x}_r, r)$. Let's consider the substitution $\boldsymbol{y}_r = \frac{\boldsymbol{x}_r}{1-r}$, then by Ito's:

$$d\boldsymbol{y}_\tau = \left(\frac{1}{(1-\tau)^2} - \frac{1}{(1-\tau)^2}\right) d\tau + \sqrt{\frac{2\tau}{(1-\tau)^3}} d\boldsymbol{w}_\tau. \tag{26}$$

Thus,

$$\boldsymbol{x}_t \overset{d}{=} (1-t)\left(\frac{\boldsymbol{x}_r}{1-r} + \sqrt{\int_r^t \frac{2\tau}{(1-\tau)^3} d\tau \boldsymbol{z}}\right) \tag{27}$$

where $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Therefore,

$$\mu(t,r) = \frac{1-t}{1-r}, \qquad \sigma^2(t,r) = t^2 - \left(r\frac{1-t}{1-r}\right)^2 \tag{28}$$

A.2 Proposition 1

We will drop $\boldsymbol{x}$ to ease the exposition. Let's write down the Fokker-Planck of eq. (13) for $\tau' := t - \delta - \tau$:

$$\partial_{\tau'} p_{\tau'|t}(\boldsymbol{x}_{\tau'}) = \tag{29}$$

$$\nabla_x \left(\left(-\frac{\boldsymbol{x}_{\tau'}}{1-\tau(\tau')} - \frac{2\tau(\tau')}{1-\tau(\tau')}\nabla_x \log p_{\tau'}(\boldsymbol{x}_{\tau'})\right)\cdot p_{\tau'|t}(\boldsymbol{x}_{\tau'})\right) + \frac{\tau(\tau')}{1-\tau(\tau')}\Delta_x p_{\tau'|t}(\boldsymbol{x}_{\tau'}) \tag{30}$$

$$= \nabla_x \left(\left(-\frac{\boldsymbol{x}_{\tau'}}{1-\tau(\tau')} - \frac{2\tau(\tau')}{1-\tau(\tau')}\nabla_x \log p_{\tau'}(\boldsymbol{x}_{\tau'}) + \frac{\tau(\tau')}{1-\tau(\tau')}\nabla_x \log p_{\tau'|t}(\boldsymbol{x}_{\tau'})\right)\cdot p_{\tau'|t}(\boldsymbol{x}_{\tau'})\right) \tag{31}$$

$$= \nabla_x \left(\left(-\frac{\boldsymbol{x}_{\tau'}}{1-\tau(\tau')} - \frac{\tau(\tau')}{1-\tau(\tau')}\nabla_x \log p_{\tau'|t}(\boldsymbol{x}_{\tau'}) + \frac{\tau(\tau')}{1-\tau(\tau')}\nabla_x \log p_{t|\tau'}(\boldsymbol{x} \mid \boldsymbol{x}_{\tau'})\right)\cdot p_{\tau'|t}(\boldsymbol{x}_{\tau'})\right) \tag{32}$$

which is exactly the continuity equation of eq. (15) in reverse time. Therefore, prop. 1 holds.

A.3 Proposition 2

Let's start with a derivation of eq. (18). Let's recall the change of variable and eq. (15):

$$\boldsymbol{z}_\tau = \frac{\boldsymbol{x} - \boldsymbol{x}_\tau}{\sigma(t,\tau)}, \qquad \boldsymbol{x}_\tau = \boldsymbol{x} - \sigma(t,\tau)\boldsymbol{z}_\tau, \tag{33}$$

$$\frac{d\boldsymbol{x}_\tau}{d\tau} = \boldsymbol{v}_\tau(\boldsymbol{x}_\tau) + \frac{\tau}{1-\tau}\mu(t,\tau)\,\frac{\boldsymbol{x} - \mu(t,\tau)\boldsymbol{x}_\tau}{\sigma^2(t,\tau)} \tag{34}$$

Now differentiating $\boldsymbol{x}_\tau = \boldsymbol{x} - \sigma(t,\tau)\boldsymbol{z}_\tau$ with respect to τ we get:

$$\frac{d\boldsymbol{x}_\tau}{d\tau} = -\partial_\tau \sigma(t,\tau)\,\boldsymbol{z}_\tau - \sigma(t,\tau)\frac{d\boldsymbol{z}_\tau}{d\tau} \tag{35}$$

Substituting eq. (33) and eq. (15) (we suppress (t,τ) from μ, σ to ease the exposition):

$$\sigma\frac{d\boldsymbol{z}_\tau}{d\tau} = -\boldsymbol{v}_\tau(\boldsymbol{x} - \sigma\boldsymbol{z}_\tau) - \frac{\tau}{1-\tau}\,\frac{\mu(1-\mu)}{\sigma^2}\boldsymbol{x} - \frac{\tau}{1-\tau}\,\frac{\mu^2}{\sigma}\boldsymbol{z}_\tau - \partial_\tau\sigma\,\boldsymbol{z}_\tau. \tag{36}$$

Simple algebra shows that:

$$\partial_\tau \sigma^2 = 2\sigma\partial_\tau\sigma = -\left(1-t\right)^2 \frac{2\tau}{(1-\tau)^3} = -2\frac{\tau}{1-\tau}\mu^2 \tag{37}$$

does we have the cancellation between the $\boldsymbol{z}_\tau$ terms and thus eq. (18) from eq. (36). Now let's move to prove the actual proposition. First note that $\frac{\sigma(t,\tau)}{\sqrt{t-\tau}} \to \sqrt{\frac{2t}{1-t}}$ as $\tau \to t$. Now let us consider the eq. (13), we have that:

$$\operatorname{Var}\left(\boldsymbol{x}_\tau \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right) = \frac{\tau^2}{t^2}\sigma^2(t,\tau). \tag{38}$$

And we have as well $\boldsymbol{x}_\tau \mid \boldsymbol{x}_0, \boldsymbol{x}_t$ Gaussian, with $\mathbb{E}\left[\boldsymbol{x}_\tau \mid \boldsymbol{x}_0, \boldsymbol{x}_t = \boldsymbol{x}\right] \to \boldsymbol{x}$ as $\tau \to t$. Therefore, $\boldsymbol{z}_\tau \mid \boldsymbol{x}_0 \Rightarrow \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{I}\right) \overset{d}{=} \boldsymbol{z}$. We will drop $\boldsymbol{x}_t$ conditioning and assume is fixed to $\boldsymbol{x}$. Thus, for any continuos bounded function ϕ:

$$\mathbb{E}\left[\phi\left(\boldsymbol{z}_\tau\right) \mid \boldsymbol{x}_0\right] \to \mathbb{E}\left[\phi\left(\boldsymbol{z}\right)\right] \tag{39}$$

as $\tau \to t$, then by the bounded convergence theorem:

$$\lim_{\tau\to t}\mathbb{E}\left[\phi\left(\boldsymbol{z}_\tau\right)\right] = \mathbb{E}\left[\lim_{\tau\to t}\mathbb{E}\left[\phi\left(\boldsymbol{z}_\tau\right) \mid \boldsymbol{x}_0\right]\right] = \mathbb{E}\left[\phi\left(\boldsymbol{z}\right)\right]. \tag{40}$$

Hence, $\boldsymbol{z}_\tau \Rightarrow \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{I}\right)$ as $\tau \to t$.

Now, substituting in eq. (15) $\boldsymbol{z}_\tau = \frac{\boldsymbol{x} - \boldsymbol{y}_\tau}{\sigma\left(t,\tau\right)}$ yields eq. (18). Furthermore, if we substitute $u = \sigma\left(t,\tau\right)$:

$$\frac{du}{d\tau} = -\frac{\tau}{1-\tau}\frac{\mu(t,\tau)^2}{u} \tag{41}$$

$$\frac{d\boldsymbol{z}}{du} = \frac{1-\tau}{\tau\mu(t,\tau)^2}\boldsymbol{v}_\tau\left(\boldsymbol{x} - u\boldsymbol{z}\right) + \frac{1-\mu\left(t,\tau\right)}{\mu(t,\tau)}\frac{\boldsymbol{x}}{u^2} \tag{42}$$

Now, note that $1 - \mu\left(t,\tau\right) \to \mathcal{O}\left(u^2\right) = \mathcal{O}\left(t-\tau\right)$, thus this ode is well-defined in $0 < \tau \le t$ and using eq. (18) we get it is well defined at $0 \le \tau < t$. Furthermore, if we assume Lipschitz continuity of $\boldsymbol{v}_\tau$ and moment boundness of $\boldsymbol{z}_0$, using eq. (18) we get the existence and uniqueness in $0 \le \tau < t$ and using eq. (42) in $0 < \tau \le t$. Therefore, the distribution limit of $\boldsymbol{z}_\tau$ as $\tau \to t$ corresponds to the solution of the ODE, i.e. $\boldsymbol{z}_t \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{I}\right)$. Hence, running the ODE backward from $\tau = t$ gives us the solution of the ODE, which we know satisfies the last claim of proposition 2, because $\boldsymbol{z}_\tau \leftrightarrow \boldsymbol{y}_\tau$ is an invertible transformation. Therefore proposition 2 holds.

A.4 Value Function

We have that eqs. (22) and (23) are obvious estimator for the value function and its gradient (based on the reparameterization trick), and we refer to the latter one as the gradient estimator. For eq. (24), the gradient-free estimator, let $\hat{\boldsymbol{x}}_0^{(i)} \sim p_{0|t}(\cdot \mid \boldsymbol{x}_t = \boldsymbol{x})$ and $p^*_{0|t}(\boldsymbol{x}_0 \mid \boldsymbol{x}) \propto \exp\bigl(R(\boldsymbol{x}_0)\bigr)\, p_{0|t}(\boldsymbol{x}_0 \mid \boldsymbol{x})$ we have then:

$$\nabla_x V_t(\boldsymbol{x}) = \frac{\mathbb{E}_{\boldsymbol{x}_0 \sim p_{0|t}(\cdot|\boldsymbol{x})}\left[\exp(R(\boldsymbol{x}_0))\ \nabla_x \log p_{0|t}(\boldsymbol{x}_0 \mid \boldsymbol{x})\right]}{\mathbb{E}_{\boldsymbol{x}_0 \sim p_{0|t}(\cdot|\boldsymbol{x})}\left[\exp\bigl(R(\boldsymbol{x}_0)\bigr)\right]} \tag{43}$$

$$\nabla_x V_t(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{x}_0 \sim p^*_{0|t}(\cdot|\boldsymbol{x})}\left[\nabla_x \log p_{0|t}(\boldsymbol{x}_0 \mid \boldsymbol{x})\right] \tag{44}$$

$$\nabla_x \log p_{0|t}(\boldsymbol{x}_0 \mid \boldsymbol{x}) = \frac{1-t}{t^2}\Bigl(\boldsymbol{x}_0 - \mathbb{E}_{p_{0|t}}[\boldsymbol{x}_0 \mid \boldsymbol{x}]\Bigr) \tag{45}$$

$$\nabla_x V_t(\boldsymbol{x}) = \frac{1-t}{t^2}\Bigl(\mathbb{E}_{p^*_{0|t}}[\boldsymbol{x}_0 \mid \boldsymbol{x}] - \mathbb{E}_{p_{0|t}}[\boldsymbol{x}_0 \mid \boldsymbol{x}]\Bigr) \tag{46}$$

Substituting SNIS estimator gives us eq. (24).

B PARAMETERIZATION

B.1 UNCONDITIONAL DRIFT

If we consider 18 for $t = 1$ then it reduces to:

$$d\boldsymbol{z}_\tau = -\boldsymbol{v}_\tau\left(\boldsymbol{x} - \boldsymbol{z}_\tau\right), \quad \boldsymbol{x}_\tau = \boldsymbol{x} - \boldsymbol{z}_\tau \tag{47}$$

Thus, our *stochastic few-step* model $f_\theta\left(\boldsymbol{x} - \boldsymbol{x}_\tau, \tau \to \tau \mid \boldsymbol{x}, 1; \theta\right)$ should approximate the *unconditional* velocity field. In the above expression, there is a degree of freedom, namely $\boldsymbol{x}$: in principle, any choice of $\boldsymbol{x}$ should yield the correct velocity field at $(\boldsymbol{x}_\tau, \tau)$. In practice every time we use the unconditional drift we call a new random $\boldsymbol{x}$. Therefore, as shown in 3b, this leads to non-deterministic dynamics.

B.2 SINGULARITY

As noted in Proposition 2, the ODE is still not well-defined at the endpoint $\tau = t = 1$. In particular, the first claim of Proposition 2 fails at $t = 1$: under the dynamics eq. (4), $\boldsymbol{x}_1$ is independent of $\boldsymbol{x}_{t<1}$. In practice, we therefore evaluate the neural network slightly away from the endpoint, using $\tau = 1 - \varepsilon, t = 1$ to initialize the sampling procedure. We emphasize, however, that the dynamics remain well-defined for $t = 1$ and any $\tau < 1$.

C Experimental Details

C.1 GMM

Table 1: Additional evaluation metrics for the GMM40 experiment. We report MMD and W_2 (lower is better) together with the Monte Carlo budget (MC) used by each method.

Method	**MMD** ↓	W_2 ↓	**MC**
ODE + ∇	0.1035	3.557	4 / 8
ODE + ∇-free	0.009823	1.153	128 / 128
ODE + denoiser	0.2865	5.338	—
SDE + ∇	0.0153	1.373	32 / 32
SDE + ∇-free	0.01436	1.287	32 / 64
SDE + denoiser	0.02718	2.099	—
SMC + ∇	0.01199	1.174	64 / 16
SMC + ∇-free	0.01124	1.076	16 / 16
SMC + denoiser	0.03625	2.37	—
Model posterior	0.01281	1.303	—

D Comparison with other methods

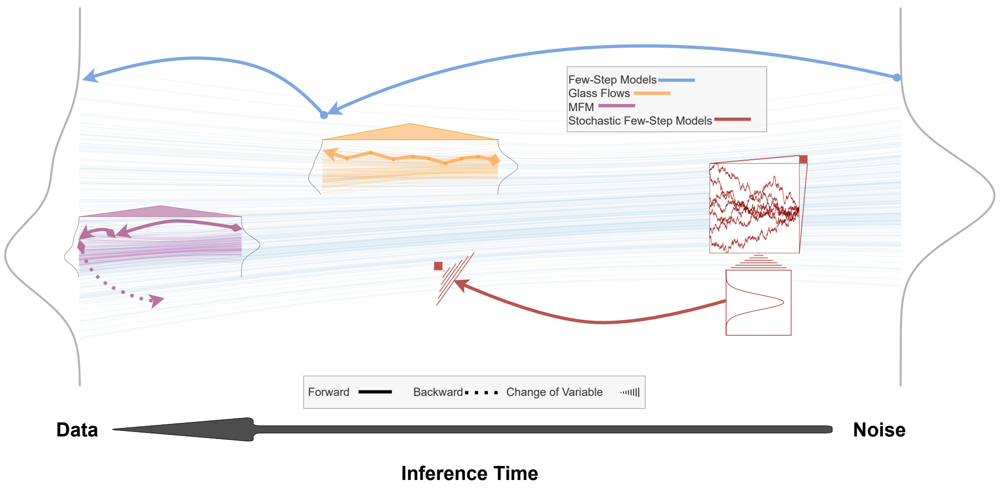


Figure 5: Schematic comparison of transition mechanisms. Flow maps (Blue): jump between arbitrary time steps. GLASS Flows (Orange): They integrate a *virtual* ODE along a path to draw a sample from the desired conditional distribution. MetaFlowMaps (Violet): distil the *virtual* ODE trajectories of GLASS into a single learned flow map. Ours (Red): reparameterize the *conditional ODEs* and distil them into a few-step model to sample conditional marginals at **any** intermediate time.