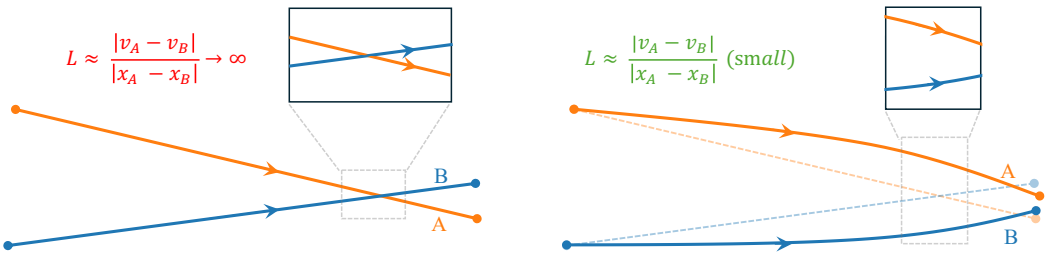
SMOOTHER FLOW MATCHING VIA CONTRASTIVE TRAJECTORY REPULSION

Anonymous authors
Paper under double-blind review



(a) Standard FM (b) CoFlow (Ours)

Figure 1: **Illustration of our motivation.** (a) In standard FM, straight trajectories inevitably cross. At the intersection, spatially adjacent points have conflicting velocity targets ($v_A \neq v_B$ while $x_A \approx x_B$), leading to an extremely high local Lipschitz constant L. (b) CoFlow explicitly repels trajectories to alleviate crossing, keeping x_A and x_B separated. This significantly reduces the local Lipschitz constant, yielding a smoother target velocity field that is easier to learn and induces lower numerical integration errors during few-step sampling.

ABSTRACT

Trajectory crossing remains a critical bottleneck in Flow Matching (FM), and previous works typically view these crossings from a theoretical optimization perspective causing velocity averaging. They attempt to address it indirectly by post-hoc distillation or endpoint coupling, without explicitly regulating the intermediate trajectories. In this paper, we introduce a new network learning perspective: crossing points inherently induce large local Lipschitz constants in the target velocity field, leading to two drawbacks. First, high Lipschitz constants correspond to high-frequency signals in the velocity field that neural networks struggle to fit due to spectral bias. Second, they also imply drastic velocity variations, leading to severe numerical integration errors in few-step inference. To alleviate this, we propose CoFlow, a framework that introduces the contrastive learning paradigm into FM to explicitly repel trajectories during training, thereby lowering the local Lipschitz constants of the velocity field. Specifically, we formulate CoFlow from a Stochastic Differential Equation (SDE) perspective by injecting a repulsive drift term. This drift actively guides the forward process of "positive samples" away from "negative trajectories", effectively reducing the local Lipschitz constant. Furthermore, we derive an equivalent stochastic interpolant formulation from this SDE, providing a simple and tractable design space to control the influence of negative samples. Extensive experiments on ImageNet 256×256 demonstrate that CoFlow significantly reduces FID compared to standard FM in few-step inference (e.g., 20 steps), with no added training overhead. The code can be accessed from: https://anonymous.4open.science/r/CoFlow-A0A0

1 INTRODUCTION

Continuous-time generative models, formulated as stochastic differential equations (SDEs) or ordinary differential equations (ODEs), have driven remarkable breakthroughs in high-quality visual synthesis. Specifically, diffusion models (Ho et al., 2020; Song et al., 2020) first gradually corrupt data with

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

noise according to a predefined forward SDE and then learn a score function to reverse this process. However, probability paths induced by these forward SDEs are highly curved, thus requiring hundreds of sampling steps during inference. To overcome this, recent flow matching (FM) (Lipman et al., 2022; Liu et al., 2022; Albergo & Vanden-Eijnden, 2022) enables the construction of straight probability paths and directly learns a velocity field, substantially reducing the required sampling steps.

Although designed to straighten paths for rapid sampling, standard flow matching still requires numerous steps (e.g., 50–100) in practice to generate high-fidelity samples. From a theoretical optimization perspective, this gap stems from trajectory crossings (Liu et al., 2022): when trajectories cross[1], the marginal velocity at intersections inherently becomes the expectation of conflicting velocities, yielding abrupt changes in the velocity field. During inference, these drastic velocity variations will amplify the truncation errors of numerical ODE solvers, thus degrading few-step generation performance. To alleviate this issue, prevailing approaches primarily operate on the boundaries: they either rely on post-hoc distillation to bypass intermediate integration (Yin et al., 2024; Song et al., 2023; Salimans & Ho, 2022; Liu et al., 2022), or optimize endpoint couplings to reduce crossings (Tong et al., 2023; Davtyan et al., 2026). While both approaches are effective, a more straightforward direction seems to remain unexplored: *why not fix the simple couplings and directly adjust the intermediate trajectories during training to explicitly mitigate crossings?*

To bridge this gap, we first examine trajectory crossing from a neural network learning perspective, and use the local Lipschitz constant to quantify the **smoothness** of the velocity field. As illustrated in Fig. 1(a), when trajectories cross, spatially adjacent points have drastically different velocity targets, causing the Lipschitz constant of the target velocity field to be extremely large. This large Lipschitz constant leads to two critical drawbacks: 1) It corresponds to high-frequency signals in the velocity field, which neural networks struggle to fit due to spectral bias (Rahaman et al., 2019). This will make the velocity field difficult to learn accurately. 2) Theoretically, a high Lipschitz constant directly amplifies the truncation error of ODE solvers, degrading few-step sampling quality.

Building upon this perspective, a compelling direction emerges: Explicitly repelling these conflicting trajectories during training to suppress the Lipschitz constant should enable the network to learn a smoother (*i.e.*, with a lower local Lipschitz constant, see Fig. 1(b)) velocity field. Motivated by this, we introduce **CoFlow**, a novel framework that systematically rectifies training trajectories via negative samples. Drawing inspiration from contrastive learning (Chen et al., 2020; He et al., 2020), CoFlow explicitly forces the trajectory of a target sample (positive) to repel adjacent negative paths (*e.g.*, paths targeting other data points in a mini-batch). Theoretically, we demonstrate that this dynamic is mathematically equivalent to injecting a repulsive drift into the forward stochastic process, closely related to Doob's h-transform (Zhou et al., 2023). From this SDE we can further derive a closed-form, stochastic interpolant (SI) formulation (Albergo & Vanden-Eijnden, 2022), which provides a flexible design space to control the influence of negative samples. We further prove that, when the repulsive drift vanishes at both endpoints, CoFlow still maps the source distribution to the target distribution, as in standard flow matching. A 2D experiment (Fig. 3) provides some initial evidence for our motivation, showing that CoFlow can learn a velocity field with lower Lipschitz and decrease the truncation errors during infernce.

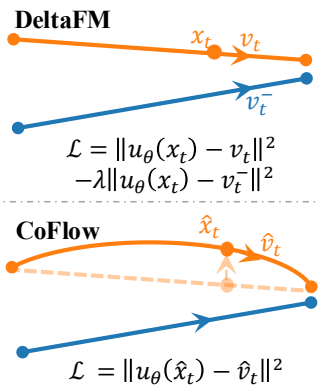


Figure 2: Comparison between DeltaFM and CoFlow.

A closely related contemporary work is DeltaFM (Stoica et al., 2025), which also explores negative samples by introducing an auxiliary contrastive loss to the standard objective to penalize velocity similarity with negative samples. Despite sharing a similar motivation, DeltaFM focuses on velocity-level alignment rather than altering intermediate trajectories, as shown in Fig. 2. Notably, another theoretical contribution of our work is proving that the optimal velocity field learned by DeltaFM is mathematically equivalent to an inference-time affine transformation of the standard FM velocity field. Leveraging this insight, we can seamlessly incorporate this transformation into CoFlow at inference time, achieving further performance gains without extra training overhead.

[1]While exact intersections are rare in high-dimensional spaces, we use "crossing" in this paper to describe scenarios where distinct trajectories are close to each other but have divergent target velocities.

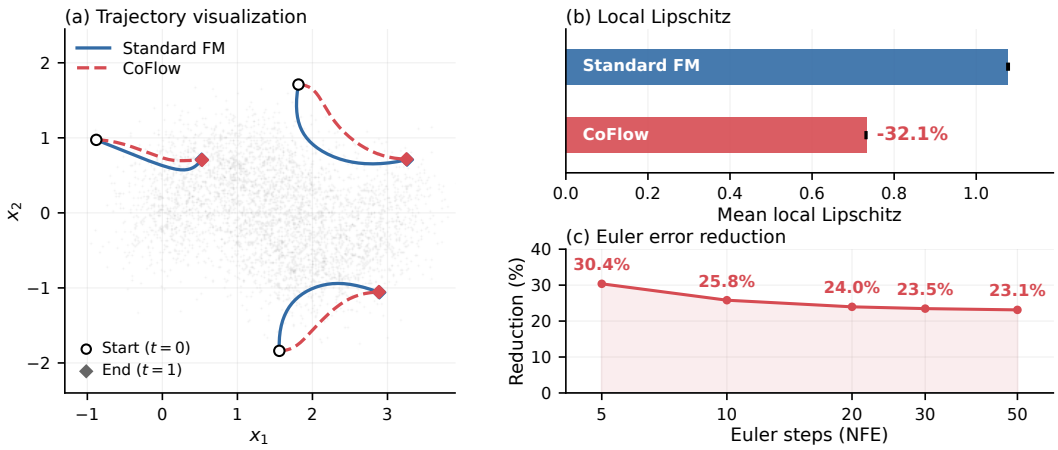


Figure 3: **2D Example: Gaussian to Two Moons.** (a) Visualization of generated trajectories: CoFlow yields paths with reduced curvature compared to standard FM. (b) Local Lipschitz constant of the learned velocity field. (c) Relative Truncation Error reduced compared with standard FM.

To further validate the effectiveness of CoFlow, we conduct extensive experiments on the ImageNet-256 generation benchmark (Deng et al., 2009). Our evaluations demonstrate that CoFlow significantly enhances the fidelity of few-step(e.g., 20 steps) image generation. Furthermore, we illustrate its compatibility with complementary techniques, such as REPA (Yu et al., 2024), OT-CFM (Tong et al., 2023), and DeltaFM (Stoica et al., 2025). In summary, our main contributions are as follows:

1) **A Lipschitz perspective on trajectory crossing.** We quantify trajectory crossing via the local Lipschitz constant, showing that its explosion amplifies numerical truncation errors and thereby degrades few-step sampling quality.
2) **CoFlow framework inspired by Doob-*h* transform.** We propose CoFlow, which explicitly repels conflicting trajectories via negative samples during training to suppress local Lipschitz constants, thus learning a smoother velocity field. We derive an exact closed-form formulation that enables efficient training without costly numerical simulation.
3) **Theoretical insight into DeltaFM.** We theoretically prove that DeltaFM is mathematically equivalent to an inference-time affine transformation, revealing its machanism and allowing it to be combined with CoFlow for further performance gains.

2 Related Works

Few Step Generation. Improving generation quality with few sampling steps is an important goal in diffusion and flow-based models. Existing efforts to achieve this can be broadly categorized into three paradigms. The first paradigm focuses on optimizing noise-data coupling during training to explicitly reduce trajectory crossing, such as Minibatch OT-CFM (Tong et al., 2023). The second paradigm relies on post-hoc rectification or distillation. For example, ReFlow (Liu et al., 2022) and InstaFlow (Liu et al., 2023b) generate new synthetic data-noise pairs to retrain the model, while distillation methods like DMD (Yin et al., 2024; Salimans & Ho, 2022; Song et al., 2023) attempt to directly distill a pre-trained model for one-step generation. Recently, a third paradigm has emerged, which introduces the concept of mean velocity by incorporating extra conditioning to naturally enable single-step sampling, like Shortcut Model (Frans et al., 2024) or MeanFlow (Geng et al., 2025).CoFlow explores a complementary approach: directly adjusting intermediate training trajectories using negative samples to encourage a smoother velocity field with lower local Lipschitz constants, thereby supporting more accurate few-step sampling.

Contrastive Learning in Generative Models. Contrastive Learning (He et al., 2020; Chen et al., 2020) is a powerful learning paradigm in deep learning, fundamentally relying on negative samples to benefit training. While extensively utilized in models like GANs (Goodfellow et al., 2014), its usage into the training of continuous-time generative models has remained relatively underexplored. Interestingly, at inference time, Classifier-Free Guidance (CFG) (Ho & Salimans, 2022; Dhariwal & Nichol, 2021) acts as a *de facto* contrastive mechanism, pushing the target conditional generation

trajectories away from unconditional ones. The remarkable effectiveness of CFG strongly implies that explicitly incorporating contrastive learning into the training phase holds great potential. Pioneering work includes Drifting Models (Deng et al., 2026) and DeltaFM (Stoica et al., 2025).

3 METHOD

In this section, we detail the formulation of CoFlow, a framework that incorporates negative samples into continuous-time generative modeling. Typically, continuous-time generative models can be described from two complementary perspectives: the stochastic diffusion perspective (e.g., score SDEs (Song et al., 2020)) and the stochastic interpolant(SI) perspective (Albergo & Vanden-Eijnden, 2022), which directly constructs the conditional probability path. Currently, SI has emerged as the prevailing paradigm (Ma et al., 2024), as it provides a more flexible design space. However, it often obscures the underlying physical intuition behind the stochastic diffusion dynamics.

Therefore, to rigorously introduce our method, we leverage the connection between these two paradigms. Specifically, we first introduce the mechanism of CoFlow from the SDE perspective, providing an intuitive motivation for injecting a repulsive drift via negative samples. Subsequently, we derive its equivalent formulation within the stochastic interpolant framework. Finally, we demonstrate how to train a velocity field in our framework.

3.1 PRELIMINARIES

Continuous-Time Generative Modeling with SDE. From the stochastic dynamics perspective, continuous-time generative models define a forward process that gradually corrupts the data distribution p_0 to a prior noise distribution p_1 over time $t \in [0, 1]$, governed by the following SDE:

$$\mathrm{d}\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{w}_t, \tag{1}$$

where $\boldsymbol{x}_t \in \mathbb{R}^d$ represents the state at time t, $\boldsymbol{w}_t$ denotes a standard d-dimensional Wiener process, $\boldsymbol{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ and $g(t) : [0, 1] \to \mathbb{R}$ are the drift and diffusion coefficients, respectively.

In mainstream diffusion frameworks (Ho et al., 2020; Song & Ermon, 2019), the drift term is designed to be linear with respect to $\boldsymbol{x}_t$, i.e., $\boldsymbol{f}(x_t, t) = \alpha(t)\boldsymbol{x}_t$. This linearity guarantees that the intermediate state $\boldsymbol{x}_t$ can be sampled in closed form by solving Eq. (1) :

$$\boldsymbol{x}_t = s(t)\boldsymbol{x}_0 + s(t)\sigma(t)\boldsymbol{x}_1, \tag{2}$$

where $\boldsymbol{x}_0 \sim p_0$, $\boldsymbol{x}_1 \sim p_1 = \mathcal{N}(0, \boldsymbol{I})$, and $s(t)$, $\sigma(t)$ are related schedule functions in EDM (Karras et al., 2022)(More details in Appendix B). This makes it feasible to sample $\boldsymbol{x}_t$ in the training of the score function $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$, with the ground truth as $\frac{-\boldsymbol{x}_1}{s(t)\sigma(t)}$. Once a score function network is learned, generation is performed by solving the corresponding reverse-time SDE from $t = 1$ to $t = 0$.

$$\mathrm{d}\boldsymbol{x}_t = \left[\boldsymbol{f}(\boldsymbol{x}_t, t) - g^2(t)\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)\right]\,\mathrm{d}t + g(t)\,\mathrm{d}\bar{\boldsymbol{w}}_t, \tag{3}$$

where $\mathrm{d}\bar{\boldsymbol{w}}_t$ is a Wiener process running backwards in time. Because directly constructing the conditional probability paths (as in Eq. (2)) offers greater engineering efficiency, subsequent frameworks like FM opt to define the generative process directly via stochastic interpolant (Albergo et al., 2023).

Special Case. To promote straighter transport trajectories, we consider a specific stochastic-interpolant configuration where the drift and diffusion coefficients are given by $\boldsymbol{f}(\boldsymbol{x}_t, t) = \frac{-1}{1-t}\boldsymbol{x}_t$ and the diffusion coefficient is $g(t) = \sqrt{\frac{2t}{1-t}}$. Under this specific SDE, the stochastic interpolant formulation of $\boldsymbol{x}_t$ essentially becomes a simple linear interpolation between the data point and the prior noise: $\boldsymbol{x}_t = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$, where $s(t) = 1-t$, $\sigma(t) = \frac{t}{1-t}$. In this configuration, the learned flows are straighter than others.

Lipschitz Constant and Truncation Error. The spatial smoothness of a velocity field $\boldsymbol{v}(\boldsymbol{x}, t)$ is characterized by its Lipschitz constant L, where $\|\boldsymbol{v}(\boldsymbol{x}_A, t) - \boldsymbol{v}(\boldsymbol{x}_B, t)\| \le L\|\boldsymbol{x}_A - \boldsymbol{x}_B\|$. When simulating the ODE via an Euler step with step size Δt, the single-step truncation error is defined as the gap between the true trajectory and the numerical update:

$$\mathcal{E}(\boldsymbol{x}_t, \Delta t, \boldsymbol{v}_t) = \|\boldsymbol{x}(t + \Delta t) - (\boldsymbol{x}(t) + \Delta t\,\boldsymbol{v}(\boldsymbol{x}(t), t))\| \le (LM + B)\Delta t^2 \tag{4}$$

Where M, B are constants bounding the velocity and time variable (See Appendix F for detailed proof). Consequently, for a fixed step size Δt, suppressing the local Lipschitz constant L effectively decreases this error, thus improving few-step sampling performance.

3.2 FIXED ENDPOINTS VIA DOOB'S h-TRANSFORM

The standard forward diffusion trajectory described in Eq. (1) transports a source point $\boldsymbol{x}_0$ towards a random destination in the prior p_1. However, in many practical scenarios, such as image-to-image translation (Zhou et al., 2023; Chung et al., 2022; Liu et al., 2023a; Kawar et al., 2022), the source-target pairs are explicitly defined. Thus, it becomes necessary to modify the forward trajectory such that it deterministically terminates at a specific target state $\boldsymbol{x}_1$.

This can be achieved via Doob's h-transform (Zhou et al., 2023). Specifically, conditioning the SDE on a fixed terminal endpoint $\boldsymbol{x}_1$ at $t = 1$ introduces an additional drift term to Eq. (1), yielding the conditional bridge SDE that terminates at $\boldsymbol{x}_1$:

$$\mathrm{d}\boldsymbol{x}_t = \left[\boldsymbol{f}(\boldsymbol{x}_t, t) + g^2(t) \cdot \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_1|\boldsymbol{x}_t)\right] \mathrm{d}t + g(t)\, \mathrm{d}\boldsymbol{w}_t. \tag{5}$$

In this formulation, the additional drift, $g^2(t)\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_1|\boldsymbol{x}_t)$, acts as an **attractive force** guiding the particle's trajectory towards the target $\boldsymbol{x}_1$. For instance, under the specific linear configuration discussed in Sec. 3.1, this attractive force simplifies to $\frac{\boldsymbol{x}_1 - \boldsymbol{x}_t}{1-t}$. As $t \to 1$, the magnitude of this force increases to infinity, mathematically guaranteeing that the trajectory terminates exactly at $\boldsymbol{x}_1$.

This mechanism inspires our core motivation: *if injecting a specific drift can exert an attractive force towards a target point, it is a natural extension to introduce a repulsive force that pushes the trajectory away from a negative sample.* This physical intuition forms the foundational motivation for our following CoFlow framework.

3.3 COFLOW

Consider a target trajectory that transports a source point $\boldsymbol{x}_0$ towards a destination $\boldsymbol{x}_1$ alongside a reference pair $(\mathbf{x}_0^r, \mathbf{x}_1^r)$. We regard the reference pair as a negative sample when its induced trajectory lies in close spatial proximity to the positive trajectory, as such cases can lead to a more challenging optimization landscape, characterized by a large local Lipschitz constant. The core motivation of CoFlow is to inject a repulsive force $\boldsymbol{b}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r)$ into the drift term of Eq. (1) to push the target particle away from the negative trajectory. Formally, this repulsive SDE is defined as follows:

$$\mathrm{d}\hat{\boldsymbol{x}}_t = \left[\boldsymbol{f}(\boldsymbol{x}_t, t) - \boldsymbol{b}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r)\right] \mathrm{d}t + g(t)\, \mathrm{d}\boldsymbol{w}_t \tag{6}$$

Mathematically, the corresponding stochastic interpolant for this modified SDE can be expressed in the following equation (See Appendix C for more details):

$$\hat{\boldsymbol{x}}_t = s(t)\boldsymbol{x}_0 + s(t)\sigma(t)\boldsymbol{x}_1 - s(t) \int_0^t \frac{\boldsymbol{b}(\boldsymbol{x}_\xi, \xi, \boldsymbol{x}_\xi^r)}{s(\xi)} \mathrm{d}\xi, \tag{7}$$

where $\hat{\boldsymbol{x}}_t$ represents the new trajectory that dynamically bypasses the negative sample. *Eq. (7) demonstrates that injecting a repulsive drift into the SDE is fundamentally equivalent to modifying the standard stochastic interpolant.*

Simulation-Free Reparameterization. The integral in Eq. (7) renders the sampling of $\hat{\boldsymbol{x}}_t$ computationally intractable during training. To circumvent this, we propose a simple reparameterization trick to make it simulation-free. Specifically, instead of defining the repulsive drift $\boldsymbol{b}$ and computing the integral, we directly define a tractable modified trajectory $\hat{\boldsymbol{x}}_t$:

$$\hat{\boldsymbol{x}}_t = s(t)\boldsymbol{x}_0 + s(t)\sigma(t)\boldsymbol{x}_1 - \hat{\boldsymbol{b}}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r), \tag{8}$$

Here, $\boldsymbol{x}_t = s(t)\boldsymbol{x}_0 + s(t)\sigma(t)\boldsymbol{x}_1$ denotes the standard baseline trajectory. Under this formulation, the repulsive drift in Eq. (6) can be analytically calculated through

$$\boldsymbol{b}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r) = s(t) \frac{\mathrm{d}}{\mathrm{d}t} \left[\hat{\boldsymbol{b}}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r)/s(t)\right] \tag{9}$$

Crucially, CoFlow provides the flexibility to explicitly design $\hat{\boldsymbol{b}}$. For example, by constraining $\hat{\boldsymbol{b}}$ to vanish at the boundaries (i.e., $\hat{\boldsymbol{b}}(\cdot, 0, \cdot) = \hat{\boldsymbol{b}}(\cdot, 1, \cdot) = 0$), CoFlow mathematically guarantees a strictly unbiased transformation between the prior noise and the target data distribution (See proof in Appendix D). In this paper, we primarily focus on this unbiased regime to ensure a fair comparison with standard FM. However, the empirical success of CFG indicates that introducing specific distributional biases can sometimes yield superior generative fidelity. This suggests a broader

design space for $\hat{\boldsymbol{b}}$, potentially allowing for intentional shifts at the boundary distributions. And we leave this for future work. After deciding the drift $\hat{\boldsymbol{b}}$, a velocity field $\boldsymbol{u}_t^\theta(\hat{\boldsymbol{x}}_t)$ can be learned by regressing against the closed-form target conditional velocity field:

$$\mathcal{L}_{CoFlow} = \mathbb{E}\left[||\boldsymbol{u}_t^\theta(\hat{\boldsymbol{x}}_t) - \boldsymbol{v}_t(\hat{\boldsymbol{x}}_t)||^2\right] \tag{10}$$

where the input $\hat{x}_t$ is sampled from Eq. (8), the target $v_t(\hat{x}_t) = \frac{\mathrm{d}\hat{x}_t}{\mathrm{d}t}$ is calculated by taking the time derivative of $\hat{x}_t$. The inference stage is the same as that in standard flow matching.

3.4 Design Space

The adjustment term $\hat{\boldsymbol{b}}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r)$ and scheduling functions $s(t), \sigma(t)$ constitute the design space of CoFlow. While $\hat{\boldsymbol{b}}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r)$ can theoretically take any form, to systematically explore this space, we decouple it into three distinct components:

$$\hat{\boldsymbol{b}}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r) = \lambda \cdot \beta(t) \cdot \boldsymbol{x}^-. \tag{11}$$

Strength Factor λ. This hyperparameter controls the overall strength of the repulsive adjustment.

Time Schedule $\beta(t)$. This function dictates how the repulsive adjustment varies across time, providing fine-grained dynamic control. As discussed in Sec 3.3, to ensure an unbiased transformation, we enforce the boundary conditions, i.e., $\beta(0) = \beta(1) = 0$. For simplicity, we fix this as a symmetric schedule $\beta(t) = t(1-t)$.

Negative Signal $\boldsymbol{x}^-$. This term defines the specific directional signal used to repel the current state $\boldsymbol{x}_t$ away from the reference trajectory $\boldsymbol{x}_t^r$. In this paper, we investigate three distinct formulations for $\boldsymbol{x}^-$: 1) *State Residual* $(\boldsymbol{x}_t^r - \boldsymbol{x}_t)$, the most intuitive spatial signal. Pushing the trajectory in the exact opposite direction of the concurrent negative state at timestep t directly reduces the local Lipschitz constant. 2) *Endpoint Residual* $(\boldsymbol{x}_1^r - \boldsymbol{x}_1)$. Because the state residual might introduce high variance dynamically across different timesteps, this strategy repels the trajectories at an instance level, thereby reducing temporal variance. 3) *Velocity Residual* $(\boldsymbol{v}_t^r - \boldsymbol{v}_t)$. Inspired by DeltaFM, this strategy leverages the difference in the target velocities as the directional signal.

Algorithm 1: CoFlow: Training

```
# fn(x, t): velocity network
# x0: training batch of data
t, x1 = rand_t(),randn_like(x0)

# Negative sampling
idx = randperm(x0.shape[0])
x0_r, x1_r = x0[idx], x1[idx]

# Base interpolants and velocities
xt = (1 - t) * x0 + t * x1
xt_r = (1 - t) * x0_r + t * x1_r
vt, vt_r = x1 - x0, x1_r - x0_r

# CoFlow trajectory adjustment
x_hat = xt- lam* t* (1- t)* (xt_r
    - xt)
v_hat = vt- lam* (1- 2* t)* (xt_r
    - xt)
      - lam* t* (1- t)* (vt_r- vt)

u = fn(x_hat, t)
loss = mse_loss(u, v_hat)
```

Sampling Policy. This dictates how we obtain the reference trajectory $(\boldsymbol{x}_0^r, \boldsymbol{x}_1^r)$ within a mini-batch. Specifically, we explore three distinct policies: 1) *Random*: We uniformly sample another trajectory from the current mini-batch to serve as the negative reference. 2) *Nearest*. We search for the most spatially adjacent sample in the batch as the reference. 3) *Largest Lipschitz*. We select the sample that yields the highest local Lipschitz constant relative to the target.

For the remainder of this paper, we default to the linear interpolation configuration, where $s(t) = 1-t$ and $\sigma(t) = \frac{t}{1-t}$. By default, we adopt the random sampling policy and utilize the state residual $(\boldsymbol{x}_t^r - \boldsymbol{x}_t)$ as the negative signal. Under this specific setting, the explicit CoFlow stochastic interpolant and its corresponding target velocity field become:

$$\hat{\boldsymbol{x}}_t = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1 - \lambda t(1-t)(\boldsymbol{x}_t^r - \boldsymbol{x}_t) \tag{12}$$

$$\boldsymbol{v}_t(\hat{\boldsymbol{x}}_t) = \boldsymbol{x}_1 - \boldsymbol{x}_0 - \lambda\big[(1-2t)(\boldsymbol{x}_t^r - \boldsymbol{x}_t) + t(1-t)(\boldsymbol{v}_t^r - \boldsymbol{v}_t)\big] \tag{13}$$

Here, $\boldsymbol{v}_t^r = \boldsymbol{x}_1^r - \boldsymbol{x}_0^r$ and $\boldsymbol{v}_t = \boldsymbol{x}_1 - \boldsymbol{x}_0$ represent the velocities for the reference and target trajectories, respectively. The complete training pipeline is summarized in Algorithm 1.



Figure 4: **Qualitative comparison**. All images are generated using a 20-step EM solver.

4 Experiments

Experimental Setup. We evaluated CoFlow on class-conditional ImageNet 256×256 using SiT (Ma et al., 2024) backbones. Images were encoded into the latent space using a pre-trained SD-VAE (Rombach et al., 2022) before training. Unless otherwise specified, all models were trained for 400K steps with a global batch size of 256, using AdamW (Loshchilov & Hutter, 2017) with a constant learning rate of 1×10^{-4} and no weight decay. For evaluation, we generated 50K images using a 20-step Euler–Maruyama (EM) sampler by default. We reported FID (Heusel et al., 2017), sFID (Nash et al., 2021), Inception Score (IS) (Salimans et al., 2016), Recall (Sajjadi et al., 2018), and Precision (Sajjadi et al., 2018). All remaining settings followed the SiT configuration.

4.1 Comparison with Standard Flow Matching

Setting. We compared CoFlow with standard FM using SiT-B/2 and SiT-XL/2 using the training protocol described above. For each method, we trained with and without REPA. For CoFlow, it was trained using the state residual as the negative signal, $\lambda = 0.2$, and $\beta(t) = t(1-t)$, as described in Sec. 3.4. The results in Table 1 are obtained without classifier-free guidance (CFG).

Few-Step Generation. Table 1 reports generation quality at NEF$=20$. Without REPA, CoFlow reduces FID from 61.43 to 52.79 on SiT-B/2 and from 38.50 to 29.65 on SiT-XL/2. The other four metrics also improve and the improvements are retained when using REPA. These results demonstrate the effectiveness of CoFlow in improving few-step generation. Fig. 4 provides a qualitative comparison using the 20-step EM sampler.

Effect of Sampling Budget. Fig. 5 reports FID-10K across different numbers of evaluation (NFE). CoFlow substantially outperforms standard FM at low NFEs, and this gap generally narrows as NFE increases, with two methods achieving similar FID at 250 NFEs. This trend is consistent with our motivation: learning a smoother velocity field with lower local Lipschitz constants can decrease numerical truncation errors under few sampling steps. As the sampling budget increases, finer discretization reduces these errors for both methods, diminishing the advantage of CoFlow. These results suggest that CoFlow primarily improves sampling efficiency in the few-step setting.

4.2 Compatibility with Other Methods

We investigated CoFlow's compatibility with two closely related methods: DeltaFM, which also exploits negative samples, and OT-CFM, which seeks to mitigate trajectory crossing.

DeltaFM. DeltaFM incorporates negative samples through modifying the training objective, whereas CoFlow adjusts the intermediate trajectories. Although DeltaFM was originally introduced as a training-based method, we show that its optimal velocity field can be obtained by an affine

Table 1: **Class-Conditional ImageNet 256×256.** Training for 400k steps. We sample 50K images using the EMSolver, with the number of function evaluations (NFE) set to 20. No CFG involved.

Model	Method	FID↓	sFID↓	IS↑	Precision↑	Recall↑
SiT-B/2	FM	61.43	35.18	27.46	0.43	0.54
	CoFlow	**52.79**	**21.71**	**35.00**	**0.45**	**0.58**
	FM + REPA	47.64	34.33	40.75	0.48	0.59
	CoFlow + REPA	**37.76**	**20.23**	**52.92**	**0.51**	**0.60**
SiT-XL/2	FM	38.50	29.27	50.14	0.54	0.58
	CoFlow	**29.65**	**17.41**	**64.12**	**0.57**	**0.59**
	FM + REPA	35.35	27.98	63.13	0.52	**0.55**
	CoFlow+ REPA	**27.78**	**17.88**	**80.52**	**0.55**	0.53

Table 2: **Compatibility with DeltaFM.** Trained on SiT-B/2 + REPA for 400K steps.

Method	FID↓	sFID↓	IS↑
FM	47.64	34.33	40.75
DeltaFM	32.32	15.73	56.33
CoFlow	37.76	20.23	52.92
CoFlow+VC	**27.74**	**8.68**	**63.94**

Table 3: **Compatibility with OT-CFM.** Trained on SiT-B/2 without REPA.

Method	FID↓	sFID↓	IS↑
OT-CFM	69.90	38.87	23.47
+CoFlow	**63.92**	**25.96**	**28.75**

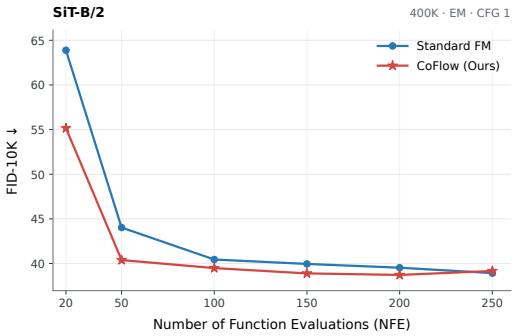


Figure 5: **FID-10K versus NFE for Standard FM and CoFlow.** EM solver was used to sample on SiT-B/2 without REPA.

transformation of the optimal standard FM field. We therefore apply this transformation to a trained standard FM model to approximate the DeltaFM solution without additional training:

$$\boldsymbol{u}_\theta^{\Delta\mathrm{FM}}(\boldsymbol{x}_t) = (\boldsymbol{u}_\theta^{\mathrm{FM}}(\boldsymbol{x}_t) - \lambda \boldsymbol{m}_t)/(1-\lambda), \tag{14}$$

where $\boldsymbol{m}_t$ is calculated by averaging the latents across the entire training set. We refer to this equivalent training-free operation as velocity centering (VC) and provide the derivation in Appendix E. As shown in Table 5, applying VC to trained FM models yields performance comparable to models trained with the DeltaFM objective.

We next evaluated whether VC could further improve CoFlow, using SiT-B/2 trained for 400K steps with REPA. As shown in Table 2, applying VC reduces CoFlow's FID from 37.76 to 27.74 and outperforms DeltaFM on all three metrics. These results demonstrate that CoFlow's trajectory adjustment can be effectively combined with DeltaFM for further improvement.

OT-CFM. OT-CFM attempts to reduce trajectory crossing by optimizing the coupling between noise and data, whereas CoFlow directly adjusts the intermediate trajectories. To evaluate whether they are compatible, we combined CoFlow with OT-CFM using SiT-B/2 trained for 400K steps without REPA. Table 3 shows that incorporating CoFlow with OT-CFM reduces FID from 69.90 to 63.92 and sFID from 38.87 to 25.96. These improvements indicate that trajectory adjustment remains beneficial when combined with optimized endpoint coupling.

4.3 Component-Wise Ablation

We examined the negative signal, sampling policy, and repulsive strength using SiT-S/2 trained for 400K steps without REPA. Table 4 reports the results of varying one component at a time. The default configuration uses $\lambda = 0.2$, the state residual $(\boldsymbol{x}_t^r - \boldsymbol{x}_t)$, and random negative sampling.

Table 4: **Component-wise ablation on ImageNet 256×256.** All models were SiT-S/2 trained for 400K iterations without REPA. We evaluated 50K images using the EM sampler with NFE = 20 and fixed $\beta(t) = t(1-t)$. Colored cells indicate the factor varied in each block.

λ	Negative signal x^-	Sampling policy	FID↓	sFID↓	IS↑
Standard FM			86.99	43.47	16.25
0.2	ϵ	Random	86.49	41.77	16.49
0.2	$x_t^r - x_t$	Random	79.55	26.63	20.36
0.2	$x_1^r - x_1$	Random	92.24	47.22	16.25
0.2	$v_t^r - v_t$	Random	122.19	87.04	9.41
0.2	$x_t^r - x_t$	Random	79.55	26.63	20.36
0.2	$x_t^r - x_t$	Lipschitz	78.12	26.16	20.66
0.2	$x_t^r - x_t$	Nearest	79.40	24.64	20.27
0.05	$x_t^r - x_t$	Random	85.66	39.77	17.04
0.2	$x_t^r - x_t$	Random	79.55	26.63	20.36
0.5	$x_t^r - x_t$	Random	75.06	14.82	22.45
1.0	$x_t^r - x_t$	Random	87.43	56.73	14.15

Negative signal. We first used random noise as signal to test whether random perturbations alone could explain CoFlow's gains. This control changes FID only slightly, from 86.99 to 86.49, suggesting that simply perturbing the trajectories cannot bring the observed improvement. We then compared three signals derived from negative trajectories: the state, endpoint, and velocity residuals. The state residual $(\boldsymbol{x}_t^r - \boldsymbol{x}_t)$ performs best across all three metrics, achieving an FID of 79.55. We hypothesize that the state residual is more suitable to reduce local Lipschitz constants, as it guides repulsion using the relative positions of trajectories.

Sampling policy. We compared the Random, Nearest, and Lipschitz policies described in Sec. 3.4. Although negatives can be selected using states at time t, the selected negatives may change as t varies, introducing discontinuities in the stochastic interpolant in Eq. (12). To avoid this, we used endpoint states $\boldsymbol{x}_0$ to select negatives for the Nearest and Lipschitz policies as a practical compromise. It turns out all three policies improve upon standard FM, with Lipschitz achieving the lowest FID of 78.12. This is consistent with our motivation to improve few-step generation by reducing local Lipschitz constants. Because random sampling remains competitive with an FID of 79.55, we adopt it as the default policy for its simplicity.

Strength factor. We varied $\lambda \in \{0.05, 0.2, 0.5, 1.0\}$, with $\lambda = 0$ corresponding to standard FM. Increasing λ from 0.05 to 0.5 improves FID from 85.66 to 75.06. However, increasing λ further to 1.0 degrades FID to 87.43, slightly worse than standard FM. These results show that moderate repulsion improves generation quality, while an excessively strong adjustment can be detrimental.

5 Conclusion

This paper presents CoFlow, a framework that utilizes negative samples to repel crossing training trajectories, which is inspired by Doob-*h* transform. Motivated by the relationship between trajectory crossing and local Lipschitze constants, CoFlow adjusts intermediate to encourage a smoother velocity field. We derived a closed-form stochastic interpolant that preserves the endpoint distributions and enables simulation-free training. Experiments on ImageNet 256×256 demonstrate improved few-step generation and compatibility with methods like REPA and OT-CFM. We also developed an inference-time velocity adjustment that is equivalent to DeltaFM, which can further improves CoFlow without additional training. Our findings suggest that contrastive learning can serve as a mechanism for adjusting training trajectories to improve few-step generation in flow matching.

AI USE STATEMENT

In this work, we used generative AI tools (specifically, ChatGPT) solely for assistance with language polishing and grammatical improvements. We have not used generative AI tools for research ideation, data collection, mathematical proofs, experimental implementation, or writing the scientific claims of this paper. All AI-assisted revisions were thoroughly reviewed and verified by the authors. We take full responsibility for the final content of this work, including all text, claims, and artifacts.

REPRODUCIBILITY STATEMENT

We provide the CoFlow training procedure in Algorithm 1 and describe its design choices and default configuration in Sec. 3.4. The training and evaluation settings, together with component-wise ablations, are reported in Sec. 4. Supporting derivations and assumptions are provided in Appendices B–E, including the formulation and implementation of velocity centering. Our anonymous implementation is available at https://anonymous.4open.science/r/CoFlow-A0A0.

REFERENCES

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Michael S Albergo, Mark Goldstein, Nicholas M Boffi, Rajesh Ranganath, and Eric Vanden-Eijnden. Stochastic interpolants with data-dependent couplings. *arXiv preprint arXiv:2310.03725*, 2023.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PmLR, 2020.

Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022.

Aram Davtyan, Leello Tadesse Dadi, Volkan Cevher, and Paolo Favaro. Faster inference of flow-based generative models via improved data-noise coupling. *arXiv preprint arXiv:2603.15279*, 2026.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. *arXiv preprint arXiv:2602.04770*, 2026.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *Advances in neural information processing systems*, 35:23593–23606, 2022.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Guan-Horng Liu, Arash Vahdat, De-An Huang, Evangelos A Theodorou, Weili Nie, and Anima Anandkumar. I^2SB: Image-to-image schrödinger bridge. *arXiv preprint arXiv:2302.05872*, 2023a.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, et al. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *The Twelfth International Conference on Learning Representations*, 2023b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pp. 23–40. Springer, 2024.

Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W Battaglia. Generating images with sparse representations. *arXiv preprint arXiv:2103.03841*, 2021.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pp. 5301–5310. PMLR, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. *Advances in neural information processing systems*, 31, 2018.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International conference on machine learning*, 2023.

George Stoica, Vivek Ramanujan, Xiang Fan, Ali Farhadi, Ranjay Krishna, and Judy Hoffman. Contrastive flow matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1185–1194, 2025.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6613–6623, 2024.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

Linqi Zhou, Aaron Lou, Samar Khanna, and Stefano Ermon. Denoising diffusion bridge models. *arXiv preprint arXiv:2309.16948*, 2023.

A LIMITATIONS

Our image-generation experiments focus on class-conditional ImageNet 256 $\times$256 with SiT backbones, due to limited computational resources. The effectiveness of CoFlow on other datasets, higher resolutions, and different generation tasks remains to be evaluated. In addition, our current implementation uses a fixed repulsion schedule and selects negative samples independently of time to retain a simple closed-form target velocity. Future work could explore adaptive schedules and time-dependent negative selection while preserving smooth interpolants and tractable velocity targets.

B EQUIVALENCE BETWEEN SDE AND SI

In this section, we give a self-contained proof of related conclusions in Sec. 3.1. Formally, we need to prove the solution of the following SDE:

$$\mathrm{d}\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{w}_t, \tag{15}$$

is the form of a stochastic interpolant:

$$\boldsymbol{x}_t = s(t)\boldsymbol{x}_0 + s(t)\sigma(t)\boldsymbol{x}_1, \tag{16}$$

where $\boldsymbol{f}(\boldsymbol{x}_t, t) = \alpha(t)\boldsymbol{x}_t$, as mentioned before. A typical method is to introduce an integrating factor as follow:

$$s(t) = \exp\left(\int_0^t \alpha(\xi)d\xi\right) \tag{17}$$

Take the derivative with respect to time t of $s(t)$, we have:

$$\frac{\mathrm{d}s(t)}{\mathrm{d}t} = \alpha(t)\cdot\exp\left(\int_0^t \alpha(\xi)d\xi\right) = \alpha(t)s(t) \tag{18}$$

Which means:

$$\mathrm{d}s(t) = \alpha(t)s(t)\mathrm{d}t \tag{19}$$

Then we consider the differential of $\frac{\boldsymbol{x}_t}{s(t)}$:

$$\mathrm{d}\left(\frac{\boldsymbol{x}_t}{s(t)}\right) = \frac{\mathrm{d}\boldsymbol{x}_t}{s(t)} - \frac{\boldsymbol{x}_t\mathrm{d}s(t)}{s(t)^2} \tag{20}$$

$$\mathrm{d}\left(\frac{\boldsymbol{x}_t}{s(t)}\right) = \frac{\alpha(t)\boldsymbol{x}_t\mathrm{d}t + g(t)\mathrm{d}\boldsymbol{w}_t}{s(t)} - \frac{\boldsymbol{x}_t(\alpha(t)s(t)\mathrm{d}t)}{s(t)^2} \tag{21}$$

$$\mathrm{d}\left(\frac{\boldsymbol{x}_t}{s(t)}\right) = \left[\frac{\alpha(t)\boldsymbol{x}_t}{s(t)}\mathrm{d}t + \frac{g(t)}{s(t)}\mathrm{d}\boldsymbol{w}_t\right] - \frac{\alpha(t)\boldsymbol{x}_t}{s(t)}\mathrm{d}t \tag{22}$$

$$\mathrm{d}\left(\frac{\boldsymbol{x}_t}{s(t)}\right) = \frac{g(t)}{s(t)}\mathrm{d}\boldsymbol{w}_t \tag{23}$$

Then we can integrate both sides of the equation from 0 to t to solve x_t:

$$\int_0^t \mathrm{d}\left(\frac{\boldsymbol{x}_\xi}{s(\xi)}\right) = \int_0^t \frac{g(\xi)}{s(\xi)}\mathrm{d}\boldsymbol{w}_\xi \tag{24}$$

$$\frac{\boldsymbol{x}(t)}{s(t)} - \frac{\boldsymbol{x}(0)}{s(0)} = \int_0^t \frac{g(\xi)}{s(\xi)}\mathrm{d}\boldsymbol{w}_\xi \tag{25}$$

According to the definition, $s(0) = \exp\left(\int_0^0 \alpha(\xi)d\xi\right) = \exp(0) = 1$, therefore we have:

$$\boldsymbol{x}(t) = s(t)\boldsymbol{x}(0) + s(t)\int_0^t \frac{g(\xi)}{s(\xi)}\mathrm{d}\boldsymbol{w}_\xi \tag{26}$$

According to the properties of stochastic calculus, the integral of a deterministic function $\frac{g(\xi)}{s(\xi)}$ with respect to Brownian motion $d\boldsymbol{w}_\xi$ follows a normal distribution with a mean of 0 and a variance equal to the integral of the square of that function (known as the Itô Isometry):

$$\int_0^t \frac{g(\xi)}{s(\xi)}d\boldsymbol{w}_\xi \sim \mathcal{N}\left(0, \left(\int_0^t \frac{g(\xi)^2}{s(\xi)^2}d\xi\right)\boldsymbol{I}\right). \tag{27}$$

Therefore, let $\sigma(t)^2 = \int_0^t \frac{g(\xi)^2}{s(\xi)^2} d\xi$, and $\boldsymbol{x}_1 \sim \mathcal{N}(0, \boldsymbol{I})$, the solution can be reparameterized as Eq (2):

$$\boldsymbol{x}_t = s(t)\boldsymbol{x}_0 + s(t)\sigma(t)\boldsymbol{x}_1 \tag{28}$$

$$s(t) = \exp\left(\int_0^t \alpha(\xi) d\xi\right) \tag{29}$$

$$\sigma(t)^2 = \int_0^t \frac{g(\xi)^2}{s(\xi)^2} d\xi \tag{30}$$

The results are consistent with those in EDM (Karras et al., 2022). For the special configuration, where $\alpha(t) = \frac{-1}{1-t}$ and $g(t) = \sqrt{\frac{2t}{1-t}}$, substitute them into the definition of $s(t), \sigma(t)$:

$$s(t) = \exp\left(\int_0^t \frac{-1}{1-\xi} d\xi\right) = 1 - t. \tag{31}$$

$$\sigma(t)^2 = \int_0^t \frac{2\xi}{(1-\xi)^3} d\xi = \frac{t^2}{(1-t)^2} \tag{32}$$

Therefore, $\boldsymbol{x}_t = s(t)\boldsymbol{x}_0 + s(t)\sigma(t)\boldsymbol{x}_1 = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$, which is linear.

C EQUIVALENCE OF COFLOW

In this section, we prove the equivalence between Eq (6) and Eq(7). Formally, for the following SDE:

$$\mathrm{d}\boldsymbol{x}_t = [\boldsymbol{f}(\boldsymbol{x}_t, t) - \boldsymbol{b}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r)]\ \mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{w}_t \tag{33}$$

Its solution should be in the form as follows:

$$\hat{\boldsymbol{x}}_t = s(t)\boldsymbol{x}_0 + s(t)\sigma(t)\boldsymbol{x}_1 - s(t)\int_0^t \frac{\boldsymbol{b}(\boldsymbol{x}_\xi, \xi, \boldsymbol{x}_\xi^r)}{s(\xi)}\mathrm{d}\xi, \tag{34}$$

The proof is similar to that in Appendix B, with the same definition of $s(t)$ and $\sigma(t)$. Specifically, while adding an additional drift $\boldsymbol{b}(x_t, t, x_t^r)$ to Eq (1), the differential of $\frac{\boldsymbol{x}_t}{s(t)}$ becomes:

$$\mathrm{d}\left(\frac{\boldsymbol{x}_t}{s(t)}\right) = \frac{\mathrm{d}\boldsymbol{x}_t}{s(t)} - \frac{\boldsymbol{x}_t \mathrm{d}s(t)}{s(t)^2} \tag{35}$$

$$\mathrm{d}\left(\frac{\boldsymbol{x}_t}{s(t)}\right) = \frac{\alpha(t)\boldsymbol{x}_t\mathrm{d}t - \boldsymbol{b}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r)\mathrm{d}t + g(t)\mathrm{d}\boldsymbol{w}_t}{s(t)} - \frac{\boldsymbol{x}_t(\alpha(t)s(t)\mathrm{d}t)}{s(t)^2} \tag{36}$$

$$\mathrm{d}\left(\frac{\boldsymbol{x}_t}{s(t)}\right) = \left[\frac{\alpha(t)\boldsymbol{x}_t}{s(t)}\mathrm{d}t - \frac{\boldsymbol{b}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r)}{s(t)}\mathrm{d}t + \frac{g(t)}{s(t)}\mathrm{d}\boldsymbol{w}_t\right] - \frac{\alpha(t)\boldsymbol{x}_t}{s(t)}\mathrm{d}t \tag{37}$$

$$\mathrm{d}\left(\frac{\boldsymbol{x}_t}{s(t)}\right) = \frac{g(t)}{s(t)}\mathrm{d}\boldsymbol{w}_t - \frac{\boldsymbol{b}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r)}{s(t)}\mathrm{d}t \tag{38}$$

Similarly, we integrate both sides of the equation from 0 to t to solve x_t:

$$\int_0^t \mathrm{d}\left(\frac{\boldsymbol{x}_\xi}{s(\xi)}\right) = \int_0^t \frac{g(\xi)}{s(\xi)}\mathrm{d}\boldsymbol{w}_\xi - \int_0^t \frac{\boldsymbol{b}(\boldsymbol{x}_\xi, \xi, \boldsymbol{x}_\xi^r)}{s(\xi)}\mathrm{d}\xi \tag{39}$$

$$\frac{\boldsymbol{x}(t)}{s(t)} - \frac{\boldsymbol{x}(0)}{s(0)} = \int_0^t \frac{g(\xi)}{s(\xi)}\mathrm{d}\boldsymbol{w}_\xi - \int_0^t \frac{\boldsymbol{b}(\boldsymbol{x}_\xi, \xi, \boldsymbol{x}_\xi^r)}{s(\xi)}\mathrm{d}\xi \tag{40}$$

Therefore, we can solve x_t:

$$\boldsymbol{x}(t) = s(t)\boldsymbol{x}(0) + s(t)\int_0^t \frac{g(\xi)}{s(\xi)}\mathrm{d}\boldsymbol{w}_\xi - s(t)\int_0^t \frac{\boldsymbol{b}(\boldsymbol{x}_\xi, \xi, \boldsymbol{x}_\xi^r)}{s(\xi)}\mathrm{d}\xi \tag{41}$$

Because $\boldsymbol{f}(\boldsymbol{x}_t, t), s(t), \sigma(t), g(t)$ are identical in the two SDEs(Eq (1) and Eq (6))), $\boldsymbol{x}_t$ can be simplified into:

$$\hat{\boldsymbol{x}}_t = s(t)\boldsymbol{x}_0 + s(t)\sigma(t)\boldsymbol{x}_1 - s(t)\int_0^t \frac{\boldsymbol{b}(\boldsymbol{x}_\xi, \xi, \boldsymbol{x}_\xi^r)}{s(\xi)}\mathrm{d}\xi \tag{42}$$

Additionally, for the **Simulation-Free Reparameterization**, we define $\hat{\boldsymbol{b}}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r)$ as:

$$\hat{\boldsymbol{b}}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r) = s(t) \int_0^t \frac{\boldsymbol{b}(\boldsymbol{x}_\xi, \xi, \boldsymbol{x}_\xi^r)}{s(\xi)} \mathrm{d}\xi \tag{43}$$

$$\frac{\hat{\boldsymbol{b}}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r)}{s(t)} = \int_0^t \frac{\boldsymbol{b}(\boldsymbol{x}_\xi, \xi, \boldsymbol{x}_\xi^r)}{s(\xi)} \mathrm{d}\xi \tag{44}$$

Take the time derivative of both sides:

$$\frac{\mathrm{d}}{\mathrm{d}t}\big[\hat{\boldsymbol{b}}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r)/s(t)\big] = \frac{\boldsymbol{b}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r)}{s(t)} \tag{45}$$

$$\boldsymbol{b}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r) = s(t) \frac{\mathrm{d}}{\mathrm{d}t}\big[\hat{\boldsymbol{b}}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r)/s(t)\big] \tag{46}$$

This completes the proof.

D UNBIASEDNESS OF COFLOW

In this section, we provide a rigorous mathematical proof that the proposed CoFlow interpolant strictly guarantees an unbiased transformation from the prior noise distribution p_1 to the target data distribution p_0, provided that the repulsive adjustment term respects the boundary conditions.

Let $\boldsymbol{x}_0 \sim p_0$ be the data distribution and $\boldsymbol{x}_1 \sim p_1$ be the prior noise distribution. Let $\boldsymbol{z} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_0^r, \boldsymbol{x}_1^r)$ denote the joint set of base and reference samples. We define the CoFlow stochastic interpolant path as:

$$\hat{\boldsymbol{x}}_t = I(t, z) = s(t)\boldsymbol{x}_0 + s(t)\sigma(t)\boldsymbol{x}_1 - \hat{\boldsymbol{b}}_t(\boldsymbol{z})$$

where $\hat{\boldsymbol{b}}_t(\boldsymbol{z})$ is an abbreviation for the repulsive adjustment term $\hat{\boldsymbol{b}}(\boldsymbol{x}_t, t, \boldsymbol{x}_t^r)$. For the CoFlow adjustment term, we strictly enforce $\hat{\boldsymbol{b}}_0(\boldsymbol{z}) = 0$ and $\hat{\boldsymbol{b}}_1(\boldsymbol{z}) = 0$ for all $\boldsymbol{z}$.Let $\rho_t(x)$ denote the time-dependent marginal probability density function of $\hat{\boldsymbol{x}}_t$.

First, we check the boundary situation when $t = 0$ and $t = 1$:

$$\hat{\boldsymbol{x}}_0 = 1 \cdot \boldsymbol{x}_0 + 0 \cdot \boldsymbol{x}_1 - \hat{\boldsymbol{b}}_0(\boldsymbol{z}) = \boldsymbol{x}_0 \tag{47}$$

Since $\boldsymbol{x}_0 \sim p_0$, the marginal density $\rho_0(\boldsymbol{x}) = p_0(\boldsymbol{x})$.

$$\hat{\boldsymbol{x}}_1 = 0 \cdot \boldsymbol{x}_0 + 1 \cdot \boldsymbol{x}_1 - \hat{\boldsymbol{b}}_1(\boldsymbol{z}) = \boldsymbol{x}_1 \tag{48}$$

Since $\boldsymbol{x}_1 \sim p_1$, the marginal density $\rho_1(\boldsymbol{x}) = p_1(\boldsymbol{x})$.

About the flow from ρ_1 to ρ_0, the continuty equation always holds:

$$\frac{\partial \rho_t(\boldsymbol{x})}{\partial t} + \nabla \cdot (\rho_t(\boldsymbol{x})\boldsymbol{v}_t(\boldsymbol{x})) = 0 \tag{49}$$

Therefore, once $\boldsymbol{v}_t(x)$ is obtained, we can use it to transfer between the two distributions. To solve $\boldsymbol{v}_t(x)$, we use the standard test function approach. For any test function $\phi(\boldsymbol{x})$, we consider the time derivative of its expectation with respect to the marginal density $\rho_t(x)$:

$$\frac{d}{dt}\mathbb{E}_{\hat{\boldsymbol{x}}_t \sim \rho_t}[\phi(\hat{\boldsymbol{x}}_t)] = \frac{d}{dt}\int \phi(\boldsymbol{x})\rho_t(\boldsymbol{x})d\boldsymbol{x} = \int \phi(\boldsymbol{x})\frac{\partial \rho_t(\boldsymbol{x})}{\partial t}d\boldsymbol{x} \tag{50}$$

Table 5: C omparison with DeltaFM. Both models are trained with REPA involved and NEF=50 for inference. "VC" denotes the inference-time Velocity Centering operation.

Model	Method	FID ↓	sFID ↓	IS ↑
SiT-B/2	FM	27.64	11.76	61.07
	DeltaFM	20.40	5.48	70.41
	FM + VC	19.97	5.45	71.44
SiT-XL/2	FM	11.40	8.69	114.69
	DeltaFM	7.29	4.93	129.89
	FM + VC	7.38	4.84	130.25

Alternatively, by taking the derivative inside the expectation over the path generation process and applying the chain rule:

$$\frac{d}{dt}\mathbb{E}_{\boldsymbol{x}_t}[\phi(\hat{\boldsymbol{x}}_t)] = \mathbb{E}_{\boldsymbol{x}_t}\left[\nabla\phi(\hat{\boldsymbol{x}}_t)\cdot\frac{d\hat{\boldsymbol{x}}_t}{dt}\right] \tag{51}$$

$$= \mathbb{E}_{\hat{\boldsymbol{x}}_t\sim\rho_t}\left[\mathbb{E}_{\boldsymbol{z}}\left[\nabla\phi(\hat{\boldsymbol{x}}_t)\cdot\frac{d\hat{\boldsymbol{x}}_t}{dt}\;\middle|\;\hat{\boldsymbol{x}}_t\right]\right] \tag{52}$$

$$= \mathbb{E}_{\hat{\boldsymbol{x}}_t\sim\rho_t}\left[\nabla\phi(\hat{\boldsymbol{x}}_t)\cdot\mathbb{E}_{\boldsymbol{z}}\left[\frac{d\hat{\boldsymbol{x}}_t}{dt}\;\middle|\;\hat{\boldsymbol{x}}_t\right]\right] \tag{53}$$

$$= \int \rho_t(\boldsymbol{x})\nabla\phi(\boldsymbol{x})\cdot\boldsymbol{v}_t(\boldsymbol{x})d\boldsymbol{x} \tag{54}$$

$$= -\int \phi(\boldsymbol{x})\nabla\cdot(\rho_t(\boldsymbol{x})\boldsymbol{v}_t(\boldsymbol{x}))d\boldsymbol{x} \tag{55}$$

Here, $\boldsymbol{z}$ means all possible $(\boldsymbol{x}_0, \boldsymbol{x}_1)$ pairings that pass given $\boldsymbol{x}_t$, and $\boldsymbol{v}_t(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}}\left[d\hat{\boldsymbol{x}}_t/dt \;\middle|\; \hat{\boldsymbol{x}}_t = \boldsymbol{x}\right]$. This means:

$$\int \phi(\boldsymbol{x})\frac{\partial\rho_t(\boldsymbol{x})}{\partial t}d\boldsymbol{x} = -\int \phi(\boldsymbol{x})\nabla\cdot(\rho_t(\boldsymbol{x})\boldsymbol{v}_t(\boldsymbol{x}))d\boldsymbol{x} \tag{56}$$

This perfectly matches the continuity equation, which means $\boldsymbol{v}_t(\boldsymbol{x})$ defined as $\mathbb{E}_{\boldsymbol{z}}\left[d\hat{\boldsymbol{x}}_t/dt \;\middle|\; \hat{\boldsymbol{x}}_t = \boldsymbol{x}\right]$ is the velocity field that can achieve the distribution transformation.

According to Theorem 7 in Stochastic Interpolant (Albergo et al., 2023), to learn this marginal velocity field, a neural network $\boldsymbol{u}_\theta(\hat{\boldsymbol{x}}_t, t)$ can be trained to minimize the mean squared error against the conditional velocity target:

$$\mathcal{L}_{CoFlow}(\theta) = \mathbb{E}_{t,\boldsymbol{z}}\left[||\boldsymbol{u}_\theta(\hat{\boldsymbol{x}}_t, t) - \frac{d\hat{\boldsymbol{x}}_t}{dt}||^2\right]$$

Therefore, we prove that the training of CoFlow is the same as that in standard FM.

E Training-free Equivalence of DeltaFM

In this section, we provide the detailed derivation for the training-free formulation of DeltaFM and discuss the empirical results presented in Table (5). We adapt the derivations in the DeltaFM appendix (Stoica et al., 2025) to our notation. The objective of DeltaFM is defined as follows:

$$\mathcal{L}^{(\Delta FM)}(\theta) = \mathbb{E}\begin{bmatrix}||\boldsymbol{v}_\theta(\boldsymbol{x}_t, t) - (\dot{\alpha}_t\hat{\boldsymbol{x}} + \dot{\sigma}_t\epsilon)||^2 \\ -\lambda||\boldsymbol{v}_\theta(\boldsymbol{x}_t, t) - (\dot{\alpha}_t\tilde{\boldsymbol{x}} + \dot{\sigma}_t\tilde{\epsilon})||^2\end{bmatrix}$$

Let $(\hat{\boldsymbol{x}}, \epsilon)$ denote the positive data-noise pair used to construct the model's input, $\boldsymbol{x}_t = \alpha_t\hat{\boldsymbol{x}} + \sigma_t\epsilon$. Similarly, let $(\tilde{\boldsymbol{x}}, \tilde{\epsilon})$ represent another data-noise pair independently sampled from the current batch, serving as the negative sample. The interpolant coefficients α_t and σ_t correspond to $s(t)$ and $s(t)\sigma(t)$ in our CoFlow framework, respectively.

For brevity, write $\boldsymbol{v}(\theta) = \boldsymbol{v}_\theta(\boldsymbol{x}_t, t)$. Expanding the squared norms and completing the square gives

$$\begin{aligned}\mathcal{L}^{(\Delta\mathrm{FM})}(\theta) = \mathbb{E}\Big[&(1-\lambda)\|\boldsymbol{v}(\theta)\|^2 \\ &- 2\boldsymbol{v}(\theta)^\top\Big(\dot\alpha_t\hat{\boldsymbol{x}} + \dot\sigma_t\epsilon - \lambda(\dot\alpha_t\tilde{\boldsymbol{x}} + \dot\sigma_t\tilde\epsilon)\Big)\Big] + C_1,\end{aligned} \tag{57}$$

$$= \mathbb{E}\left[\left\|\sqrt{1-\lambda}\,\boldsymbol{v}(\theta) - \frac{\dot\alpha_t\hat{\boldsymbol{x}} + \dot\sigma_t\epsilon - \lambda(\dot\alpha_t\tilde{\boldsymbol{x}} + \dot\sigma_t\tilde\epsilon)}{\sqrt{1-\lambda}}\right\|^2\right] + C_2, \tag{58}$$

where C_1 and C_2 do not depend on θ and therefore do not affect the minimizer.

For a fixed input $(\boldsymbol{x}, t)$, the conditional objective is strictly convex in the predicted velocity because $1-\lambda > 0$. Setting its gradient to zero yields

$$\begin{aligned}(1-\lambda)\boldsymbol{v}_t^{\Delta\mathrm{FM},*}(\boldsymbol{x}) = &\,\mathbb{E}\left[\dot\alpha_t\hat{\boldsymbol{x}} + \dot\sigma_t\epsilon \mid \boldsymbol{x}_t = \boldsymbol{x}, t\right] \\ &- \lambda\mathbb{E}\left[\dot\alpha_t\tilde{\boldsymbol{x}} + \dot\sigma_t\tilde\epsilon \mid \boldsymbol{x}_t = \boldsymbol{x}, t\right].\end{aligned} \tag{59}$$

The first conditional expectation is the optimal standard FM velocity field:

$$\boldsymbol{v}_t^{\mathrm{FM},*}(\boldsymbol{x}) = \mathbb{E}\left[\dot\alpha_t\hat{\boldsymbol{x}} + \dot\sigma_t\epsilon \mid \boldsymbol{x}_t = \boldsymbol{x}, t\right]. \tag{60}$$

Because the negative pair is independent of the positive pair and the sampled time, the second conditional expectation simplifies to

$$\begin{aligned}&\mathbb{E}\left[\dot\alpha_t\tilde{\boldsymbol{x}} + \dot\sigma_t\tilde\epsilon \mid \boldsymbol{x}_t = \boldsymbol{x}, t\right] \\ &\quad= \mathbb{E}\left[\dot\alpha_t\tilde{\boldsymbol{x}} + \dot\sigma_t\tilde\epsilon \mid t\right] = \boldsymbol{m}_t.\end{aligned} \tag{61}$$

We therefore obtain the following relation between the optimal velocity fields:

$$\boldsymbol{v}_t^{\Delta\mathrm{FM},*}(\boldsymbol{x}) = \frac{\boldsymbol{v}_t^{\mathrm{FM},*}(\boldsymbol{x}) - \lambda\boldsymbol{m}_t}{1-\lambda}. \tag{62}$$

Since the Gaussian prior has zero mean, $\boldsymbol{m}_t$ simplifies to

$$\boldsymbol{m}_t = \dot\alpha_t\mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_0}[\tilde{\boldsymbol{x}}] + \dot\sigma_t\mathbb{E}_{\tilde\epsilon\sim\mathcal{N}(0,\boldsymbol{I})}[\tilde\epsilon] \tag{63}$$

$$= \dot\alpha_t\boldsymbol{\mu}_{\mathrm{data}}, \tag{64}$$

where $\boldsymbol{\mu}_{\mathrm{data}}$ is the mean of the data distribution in VAE latent space. Eq. (62) shows that the optimal DeltaFM velocity field is an affine transformation of the optimal standard FM field. Applying this transformation to a learned standard FM velocity field gives an approximation to the optimal DeltaFM field:

$$\boldsymbol{u}_\theta^{\Delta\mathrm{FM}}(\boldsymbol{x}_t) = \frac{\boldsymbol{u}_\theta^{\mathrm{FM}}(\boldsymbol{x}_t) - \lambda\boldsymbol{m}_t}{1-\lambda}, \tag{65}$$

where $\boldsymbol{u}_\theta^{\Delta\mathrm{FM}}$ denotes the transformed prediction and $\boldsymbol{m}_t$ is estimated from the training data. We call this operation velocity centering (VC), which requires no additional model training. To verify its effectiveness, we conducted the experiments in Table 5. Sepcifically, we estimated the data mean by averaging the SD-VAE latents across the entire training set.

To evaluate VC, we estimated the data mean by averaging the SD-VAE latents across the entire training set and applied the transformation to trained FM models during inference. As shown in Table 5, FM+VC achieves performance comparable to models explicitly trained with the DeltaFM objective, supporting the effectiveness of this training-free formulation.

F Lipschitz Constants and Euler Discretization Error

Consider the ODE $\dot{\boldsymbol{x}}(s) = \boldsymbol{v}(\boldsymbol{x}(s), s)$. Assume that, in a neighborhood of the trajectory over $[t, t+h]$, the velocity is bounded by M and is Lipschitz continuous in space and time with constants L and B,

respectively. Starting from the definition of the single-step Euler error, we obtain

$$\begin{aligned}
\mathcal{E} &= \|\boldsymbol{x}(t+h) - \boldsymbol{x}(t) - h\boldsymbol{v}(\boldsymbol{x}(t), t)\| \\
&= \left\| \int_t^{t+h} \big[\boldsymbol{v}(\boldsymbol{x}(s), s) - \boldsymbol{v}(\boldsymbol{x}(t), t)\big] \, \mathrm{d}s \right\| \\
&\leq \int_t^{t+h} \Big[\|\boldsymbol{v}(\boldsymbol{x}(s), s) - \boldsymbol{v}(\boldsymbol{x}(t), s)\| + \|\boldsymbol{v}(\boldsymbol{x}(t), s) - \boldsymbol{v}(\boldsymbol{x}(t), t)\| \Big] \, \mathrm{d}s \\
&\leq \int_t^{t+h} \big[L\|\boldsymbol{x}(s) - \boldsymbol{x}(t)\| + B(s-t) \big] \, \mathrm{d}s \\
&\leq \int_t^{t+h} \left[L \int_t^s \|\boldsymbol{v}(\boldsymbol{x}(r), r)\| \, \mathrm{d}r + B(s-t) \right] \, \mathrm{d}s \\
&\leq \int_t^{t+h} (LM + B)(s-t) \, \mathrm{d}s \\
&= \frac{1}{2}(LM + B)h^2.
\end{aligned} \tag{66}$$

For fixed M and B, reducing the spatial Lipschitz constant L tightens this error bound.