FLOW MAP LEARNING VIA NONGRADIENT VECTOR FLOW

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion and flow-based models benefit from simple regression losses, but inference (i.e, producing samples) incurs significant computational overhead because it requires integration. Consistency models address this overhead by directly learning the flow maps along the ODE trajectory, revealing a design space *for the learning problem* between one-step and many-step approaches. However, existing consistency training methods feature computational challenges such as requiring model inverses or backpropagation through iterated model calls, and do not always prove that the desired ODE flow map is a solution to the loss. We introduce CurlFlow, an approach for learning flow maps that bypasses explicit invertibility constraints and expensive differentiation through model iteration. CurlFlow trains a model to compute both the ODE solutions and the implied velocity from scratch by following non-conservative dynamics (i.e., those with *curl*) with stationary point at the desired flow map. On the CIFAR image benchmark, CurlFlow attains a favorable relationship of FID to step count, relative to flow matching, MeanFlow, and several other flow map learning methods.

1 INTRODUCTION

Diffusion and flow models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020; Kingma et al., 2021; Albergo and Vanden-Eijnden, 2022; Singhal et al., 2023; Pandey and Mandt, 2023; Bartosh et al., 2024; Singhal et al., 2024; Albergo et al., 2023; Lipman et al., 2022; Liu et al., 2022) have improved generation in domains such as proteins (Abramson et al., 2024) and images (Peebles and Xie, 2023; Esser et al., 2024). Sampling from these models typically requires numerically integrating an ordinary or stochastic differential equation. Numerical integration requires multiple forward passes of a neural network, leading to increased sampling latency and cost.

To ameliorate this generation cost by changing the training, recent approaches for consistency modeling and map matching (Song et al., 2023; Song and Dhariwal, 2023; Kim et al., 2023; Lu and Song, 2024; Boffi et al., 2024; 2025) aim to learn direct mappings from noise to intermediate or final data points along trajectories defined by probability flow ODEs, thereby avoiding costly integration. However, the methods have their respective complexities. For example, flow map matching requires model invertibility, while consistency models need either to map in one step or introduce extra steps that leave the target ODE trajectory.

We introduce CurlFlow, an approach that builds on flows and map matching methods and

- Has true flow map as optimum
- Does not require integration for generation
- Does not restrict the class of neural networks used (e.g., to invertible functions)
- Does not require auxiliary losses involving invertibility or adversarial optimization
- Does not require optimizing through nested calls to the model or through products of model outputs (which lead to large autodifferentiation graphs)
- Allows for generation along the ODE trajectory with any number of steps

Existing methods for learning flow maps fall into a few categories in terms of their challenges; all challenges relate to the idea that a flow map is characterized by certain derivative properties and

Methods	Multistep	Follows ODE	Sim. Free	Regression Loss	Inverse Required	Prove Optimum	Model Nesting/Product
Consistency Distillation (Song et al., 2023)	✗	✓	✗	✓	✗	✓	✗
Consistency Training (Song et al., 2023)	✗	✓	✓	✓	✗	✓	✗
Consistency Trajectory Models (Kim et al., 2023)	✓	✓	✗	✗	✗	✓	✗
L-FMM (Boffi et al., 2024)	✓	✓	✓	✓	✓	✓	✗
LSD (Boffi et al., 2025)	✓	✓	✓	✓	✗	✓	✓
ESD (Boffi et al., 2025)	✓	✓	✓	✓	✗	✓	✓
PSD (Boffi et al., 2025)	✓	✓	✓	✓	✗	✓	✓
MeanFlow (Geng et al., 2025)	✓	✓	✓	✓	✗	✗	✗
CurlFlow (this work)	✓	✓	✓	✓	✗	✓	✗

Table 1: **Comparison to prior works**. We categorize consistency modeling (flow map learning) techniques and our proposed CurlFlow method according to: (1) ability to adjust sampling steps post-training, (2) whether they follow the PF-ODE (Song et al., 2020), (3) whether they allow simulation-free training, (4) whether their objectives use regression, (5) whether model inversion is required during training, (6) whether the true flow map is shown to be optimal or stationary, and (7) whether differentiation passes through nested model calls or products of evaluations. See Section 5 for details.

that losses minimize squared error to make these properties hold. Flow map matching and related methods rely on a fundamental relationship between invertible mappings and ordinary differential equations (ODEs). This relationship typically requires explicitly computing both the forward map (the model being trained) and its inverse during training, complicating training, and requires explicitly materializing a large derivative matrix in the forward pass. Other methods, such as MeanFlow (Geng et al., 2025), do not explicitly enforce the model inverse identities, but also do not prove that their loss is minimized at the true flow map. Moreover, the only model derivative terms that MeanFlow uses to enforce the properties of a flow map, are subject to the stopgrad operator, meaning that it is unclear whether optimization can lead to a function satisfying the flow map derivative properties.

CurlFlow avoids the complexity of tracking a model and its inverse, as well as materializing derivatives, by exploiting an alternate identity involving only Jacobian-vector products (JVPs) without inverse functions. This identity allows us to formulate the objective purely in terms of the forward map, without needing explicit access to its inverse. Since solutions to ODEs naturally produce invertible mappings, the CurlFlow objective implicitly encourages invertibility without explicitly enforcing it. Thus, at optimality, CurlFlow yields a continuously differentiable function that precisely integrates the velocity field, directly generating the desired data distribution. We summarize the trade-offs among recent methods in Table 1 and in Section 5.

Experimentally, we keep things simple. We do not explore any generalizations of classifier-free guidance (Ho and Salimans, 2022) for conditional sampling in flow maps. We ask, for a basic training setup using the same common architecture, how do flow matching, MeanFlow, CurlFlow, Lagrangian Map Matching, Eulerian Map Matching, and Progressive Map Matching compare in moderate dimensions (CIFAR-10) on unconditional metrics (FID) when decreasing the number of sampling steps? We additionally compare GPU memory usage for these methods. Finally, we provide reasoning and numerical evidence that MeanFlow does not preserve the true flow map as an optimum.

2 Background

Stochastic interpolants (Lipman et al., 2022; Albergo et al., 2023), and more generally most diffusion and flow methods, hereafter just *flows* pose generative modeling as transport of a simple base density to a target density. Interpolants tackle the problem as follows. For $t \in [0, 1]$:

1. Choose (α_t, σ_t) where $\alpha_0 = \sigma_1 = 1$ and $\alpha_1 = \sigma_0 = 0$. Commonly, $\alpha_t = 1 - t$ and $\sigma_t = t$.
2. Define $X_t = \alpha_t X_0 + \sigma_t X_1$ for base density $X_0 \sim q_0$ and data $X_1 \sim q_1$ (or vice versa).
3. Learn to produce new samples along the trajectory of densities.

For a function f, let $\dot{f}_t := \frac{d}{dt} f_t$. Thus $\dot{X}_t := \dot{\alpha}_t X_0 + \dot{\sigma}_t X_1$. It follows that X_t has density q_t satisfying:

$$\partial_t q_t(x) = -\nabla_x \cdot (q_t(x) v(t, x)), \qquad v(t, x) := \mathbb{E}[\dot{X}_t \mid X_t = x], \tag{1}$$

where v is called the velocity. The PDE in Equation (1) is derived in the above works and we provide a self-contained derivation in Section A.1. To accomplish step 3, one starts by making the observation that a density satisfies Equation (1) if and only if it is the density of the solution to the *probability flow ODE* $dx = vdt$ integrated forward from $X_0 \sim q_0$ or in reverse from $X_1 \sim q_1$ (Albergo and Vanden-Eijnden, 2024). One then proceeds by first approximating v using the following (simulation-free) loss:

$$\mathcal{L}_v(v_\theta) = \mathbb{E}\Big[\|v_\theta(t, X_t) - (\dot{\alpha}_t X_0 + \dot{\sigma}_t X_1)\|^2\Big]_{X_t = \alpha_t X_0 + \sigma_t X_1}, \tag{2}$$

which has minimizer $v_\theta = v$ and then solving $dx = v_\theta dt$.

Background on Consistency Methods. Sampling from flows requires integration where each integration step evaluates a neural network v_θ modeling a score, velocity, or similar. Therefore, knowing integrals of v or similar quantities directly could, in principle, speed up sampling. The goal of consistency and map matching methods is to learn to map along the trajectory implied by the optimal v. We review an example here, with others described in Section 5. Song et al. (2023); Song and Dhariwal (2023) seek to learn a mapping $\hat{g}$ that maps interpolant samples $X_t \sim q_t$ to $\widehat{X}_0$, the $t = 0$ solution to $dx = vdt$ when starting at X_t (note that $\widehat{X}_0$ usually differs from the independent endpoint sample X_0 used to draw X_t under the interpolant). The loss measures the distance between modeled outputs when evaluated at two different nearby points. Let $\text{sg}[\hat{g}]$ indicate stopgrad (i.e., bookkeeping a term as a constant when computing loss gradients). Then the loss is:

$$\text{Consistency}(\hat{g}) := \mathbb{E}_{q(X_t)}[\text{dist}(\hat{g}(t, X_t), \text{sg}[\hat{g}](t - \Delta t, \widehat{X}_{t-\Delta t}))]. \tag{3}$$

The sample $\widehat{X}_{t-\Delta t}$ used in the target should optimally come from integrating the true velocity or an approximation v_θ a small step Δt from X_t, where v_θ either comes from a pretrained diffusion model, or $(v_\theta, \hat{g})$ are derived from one another. In practice, further approximations are used to compute $x_{t-\Delta t}$. Approximations are introduced because $\widehat{X}_{t-\Delta t}$ is not simply defined by drawing a second interpolant sample at a smaller noise level, but instead corresponds to integrating the velocity one step from X_t; the velocity is unknown and thus may come from a pre-trained model, which increases training cost and may not be a good approximation in the first place.

It is challenging to directly learn solutions in just one step. While this allows multistep sampling, the re-noising step necessarily takes the trajectory off the probability-flow ODE and the resulting updates no longer correspond to integrating the PF-ODE solution. Highlighting the issue with methods featuring 1 time argument, Kim et al. (2023) note that this CM multistep sampling approach "exhibits degrading sample quality with increasing NFE, lacking a clear trade-off between computational budget (NFE) and sample fidelity". In practice, a number of training-time or inference-time changes are made to this setup to try to break apart the problem into somewhere between 1 step and the hundreds of steps used by diffusions (Song et al., 2023; Lu and Song, 2024; Kim et al., 2023; Boffi et al., 2024; 2025; Sabour et al., 2025; Geng et al., 2025; Zhou et al., 2025). We discuss the various solutions and their trade-offs in Section 5.

3 METHOD

We present CurlFlow, a method for learning to solve the probability flow ODE without adversarial training, without model inverse during training, without representing explicit derivative matrices, without costly simulations from pretrained models, and without adversarial training. CurlFlow trains a model to compute both the ODE solutions and the implied velocity from scratch by following non-conservative dynamics .

Consider a two-time map f that for $t \leq u$ brings X_t up to x_u by solving the probability flow ODE $dx = vdt$. Such an f that integrates v can be defined as follows:

$$f(t, u, x) = x + \int_t^u v(s, X_s)ds = x + \int_t^u v(s, f(t, s, x))ds \tag{4}$$

Differentiating the recursive form on the RHS w.r.t. t using the total derivative (see **??**) yields:

$$\partial_t f + (\partial_x f)v(t, x) = 0, \quad f(u, u, x) = x \tag{5}$$

This is uniquely solved at the true flow map f . We can square the LHS for a parameterized f_θ and take an expectation over X_t

$$L = \mathbb{E}_{X_t}[\|\partial_t f_\theta + (\partial_x f_\theta)\mathbb{E}[\dot{X}_t|X_t]\|^2] \tag{6}$$

The true map f is the unique minimizer of this loss. Using $v(t,x) = \mathbb{E}[\dot{X}_t|X_t]$, we can expand,

$$L = \mathbb{E}_{X_t}\Big[\|\partial_t f_\theta + (\partial_x f_\theta)\dot{X}_t\|^2 - \|(\partial_x f_\theta)\big(\dot{X}_t - \mathbb{E}[\dot{X}_t|X_t]\big)\|^2\Big] \tag{7}$$

We can then use the parameterization $f_\theta = x + (u-t)\tilde{f}_\theta(t,u,x)$ for an underlying model $\tilde{f}_\theta$. The parameterization yields two useful properties:

- time derivative: $\partial_t f_\theta(t,t,x) = -\tilde{f}_\theta(t,t,x)$
- Jacobian: $\partial_x f_\theta(t,t,x) = I$

Using these properties and evaluating at $t = u$, we see that the minimization of eq. (7) reduces flow matching where $\tilde{f}_\theta(t,t,x)$ is trained to match the velocity:

$$L\Big|_{t=u} = \mathbb{E}_{X_t}[\|\tilde{f}_\theta(t,t,x) - \dot{X}_t\|^2], \tag{8}$$

which reveals that for the true f, we have that

$$-\partial_t f(t,t,\cdot) = \tilde{f}(t,t,\cdot) = v(t,x) = \mathbb{E}[\dot{X}_t|X_t = x] \tag{9}$$

This motivates replacing the unknown v in eq. (7) with stopgrad$[\tilde{f}_\theta(t,t,\cdot)]$. The stopgrad is used under the principle that because the original v did not provide gradient updates for f, so neither should a term that approximates it. The **CurlFlow** method follows the negative gradient with respect to θ of:

$$L_{\text{sg}} := \mathbb{E}\Big[\|(\partial_t f_\theta)_{(t,u,X_t)} + (\partial_x f_\theta)_{(t,u,X_t)}\dot{X}_t\|^2 - \|(\partial_x f_\theta)_{(t,u,X_t)}(\dot{X}_t - \text{sg}[\tilde{f}_\theta]_{(t,t,X_t)})\|^2\Big], \tag{10}$$

where sg() means stopgrad() and $f_\theta(t,u,x) := x + (u-t)\tilde{f}_\theta(t,u,x)$. The expectation is taken over X_t sampled by drawing data X_1, noise X_0, and computing $X_t = \alpha_t X_0 + \sigma_t X_1$ and $\dot{X}_t = \dot{\alpha}_t X_0 + \dot{\sigma}_t X_1$.

In practice, the PDE must hold for all pairs $t \le u$, Let $q(t,u)$ be a joint distribution with support over $t \le u$ and with positive probability on $t = u$. Take expectations over time and define $\mathcal{L} = \mathbb{E}_{q(t,u)}[L]$ and $\mathcal{L}_{\text{sg}} = \mathbb{E}_{q(t,u)}[L_{\text{sg}}]$. We now connect $\mathcal{L}$ and $\mathcal{L}_{\text{sg}}$ formally. Theorem 1 shows that optimization of $\mathcal{L}$ and $\mathcal{L}_{\text{sg}}$ stop at the same solutions.

Theorem 1. *Let $q(t,u)$ be a joint distribution with support over $t \le u$ and with positive probability on $t = u$. Let the family $\tilde{\mathcal{F}}$ include functions $\tilde{f}$ that are continuously differentiable in all arguments. Let $X_t = \alpha_t X_0 + \sigma_t X_1$ and $\dot{X}_t = \alpha_t X_0 + \sigma_t X_1$. Evaluate f at $f(t,u,x) + (u-t)\tilde{f}(t,u,x)$. Take expectations over $q(X_0)q(X_1)$. Let sg stand for stop-gradient. Define $\mathcal{L} = \mathbb{E}_{q(t,u)}[L]$ and $\mathcal{L}_{sg} = \mathbb{E}_{q(t,u)}[L_{sg}]$. Then $\tilde{f}^*$ is a stationary point of $\mathcal{L}_{sg}$ w.r.t. $\tilde{\mathcal{F}}$ if and only if $\tilde{f}^*$ is a stationary point of $\mathcal{L}$ w.r.t. $\tilde{\mathcal{F}}$.*

This is shown in Section B.5.

Intuition. The main point of the theorem is to establish that $\mathcal{L}_{\text{sg}}$ has the same set of solutions as L despite not having access to v. The intuition is that, despite the stopgrad, when $t = u$, $\mathcal{L}_{\text{sg}}$ tries to match the velocity. We show that $\mathcal{L}_{\text{sg}}$ is not at a stationary point when this velocity estimate is inaccurate, so the optimization continues moving and does not become stuck at functions that distill an incorrect velocity. As this match improves so does the match between the parameter updates from $\mathcal{L}_{\text{sg}}$ and $\mathcal{L}$ at $t \neq u$. The main reason this works is that $\tilde{f}(t,t,\cdot)$ appears in other terms outside of the stopgrad, and those terms tell it where to go. This is crucial and not all stopgrad optimizations benefit from this property.

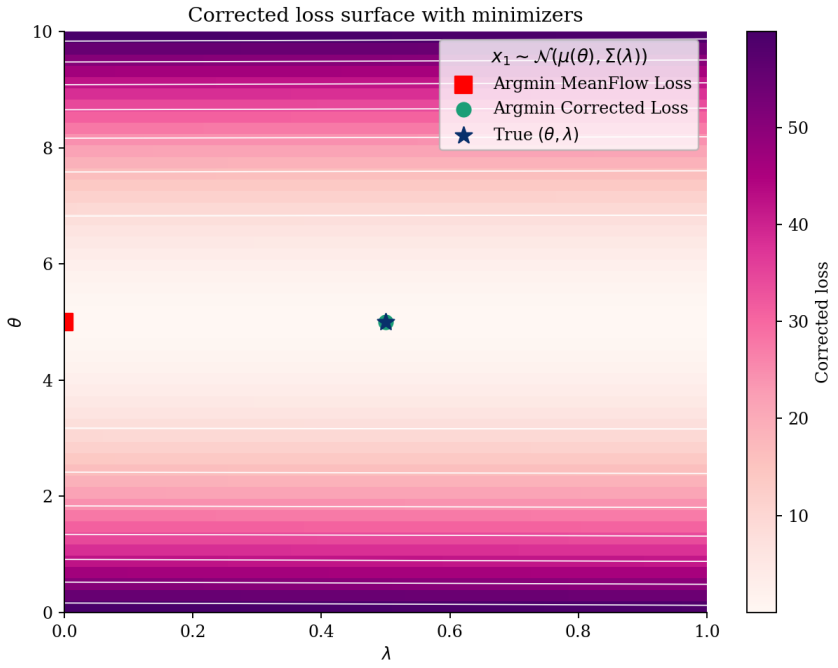

Figure 1: **Minimizers of MeanFlow.** We compute the loss surface of the corrected loss (what MeanFlow starts with before replacing v) and show that MeanFlow (using $\dot{X}_t$ instead of v) does not have its optimum at the true data-generating parameters for a 2D Gaussian example.

Computation. Both terms in $\mathcal{L}_{\text{sg}}$ can be computed as expected squared norms of Jacobian-vector products (JVPs), which use forward-mode autodifferentiation to avoid explicitly materializing Jacobians, saving memory. Using PyTorch notation,

$$\text{JVP}[f, (t, u, x), (a, b, c)] := (\partial_t f) \cdot a + (\partial_u f) \cdot b + (\partial_x f) \cdot c$$

for $(\partial_t f, \partial_u f, \partial_x f)$ evaluated at (t, u, x). For the first loss term, $a = 1$, $b = 0$ and $c = \dot{X}_t$ and for the second loss term, $a = 0$, $b = 0$, and $c = \dot{X}_t + \text{sg}[\partial_t f(t, t, X_t)] = \dot{X}_t - \text{sg}[\tilde{f}(t, t, X_t)]$. Though we have two distinct JVPs, we can split the batch and randomly assign either pair of (a, c) values to each batch element.

Nongradient Flow Following the update rules of $\mathcal{L}_{\text{sg}}$ does not correspond to following the gradients of any one scalar objective J (section C.1). This is because the optimization dynamics are in general *non-conservative*. The stopgrad structure breaks the symmetry required for the updates to be the gradient of a single scalar function. In this sense, CurlFlow is formally a (two-player) *game* rather than standard gradient descent on one potential function, albeit a trivial one where the main player controls all parameters except those in the stopgrad, and the stopgrad player keeps a virtual copy of the parameters that simply equal the first players parameters. Put another way, the optimization dynamics if taken in the limit of small step size correspond to non-conservative / non-gradient vector flow.

4 Experiments

4.1 Multivariate Gaussian and Wrong MeanFlow Optimum

To motivate CurlFlow, we first study a simple 2D Gaussian experiment that shows what can go wrong in flow map learning methods. The losses that are derivable from PDEs such as $\partial_t f + (\partial_x f) \cdot v = 0$, depend on the velocity $v(t, x)$. Because v is unknown apriori, some methods work with pretrained velocity models (Sabour et al., 2025; Boffi et al., 2025). For from-scratch methods, the velocity must be replaced, somehow. MeanFlow (Geng et al., 2025) is trained with JVPs like CurlFlow, but replaces v with $\dot{X}_t$. Crucially, this swap implicitly drops a Trace Covariance Jacobian term (which is tracked in the above CurlFlow derivation), no longer preserving the true flow map as the minimizer

of the loss. That is, the MeanFlow loss using v does not differ from the loss using $\dot{X}_t$ only by a model-independent constant.

The data $X_1 \sim \mathcal{N}(\mu_1, \Sigma_1)$ are generated with parameters $\theta \in \mathbb{R}$ and $\lambda \in [0, 1]$ via

$$\mu_1 = \theta \cdot [1, 1], \quad \Sigma_1 = \lambda\, \Sigma_a + (1 - \lambda)\, \Sigma_b \tag{11}$$

where

$$\Sigma_a := \begin{pmatrix} 1.0 & -0.3 \\ -0.3 & 1.5 \end{pmatrix}, \quad \Sigma_b := \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}, \quad \Sigma_1 = \lambda\Sigma_a + (1 - \lambda)\Sigma_b \tag{12}$$

Over a grid of (θ, λ) values that includes the true parameters $(\theta = 5.0, \lambda = 0.5)$, we compute two loss functions: the *MeanFlow loss*, and a *corrected loss* that instead uses the exact velocity (available in closed form for Gaussians, though generally unknown in practice). The corrected loss is thus equal to $\mathcal{L}$ in Theorem 1. In Figure 1, we visualize the corrected loss surface together with the true parameter point and the minimizers of both objectives. The results show that while MeanFlow correctly recovers the mean parameter θ, it fails to identify the correct covariance parameter λ. Specifically, MeanFlow sets $\lambda = 0$ which chooses $\Sigma_1(\lambda) = \Sigma_b$ where Σ_b is diagonal while Σ_a features correlation.

CurlFlow deals differently with the unknown velocity, instead substituting it with the flow map model's implied velocity, while also making sure that another loss term trains this implied velocity with a flow matching loss (see Theorem 1). When the velocity is trained enough, the CurlFlow loss becomes equal to this corrected loss.

4.2 Image Modeling on CIFAR-10

Architecture. We modify the existing diffusion U-Net from Dhariwal and Nichol (2021) by embedding both t and u with the usual Fourier embeddings and then concatenate on input to small MLP that maps the two times to a hidden representation for use in the network. We use 128 channels and channel multipliers set to (1,2,2,2) with `attention` set to (False, False, True, False).

Training Settings. We use dropout 0.1. We do not condition on the class label We use $\alpha_t = 1 - t$ and $\sigma_t = t$, with noise at X_0 and data at X_1. We train for 200,000 steps at learning rate 2e-4.

Losses. We train with Flow Matching (Lipman et al., 2022), MeanFlow (Geng et al., 2025), and the proposed CurlFlow method (Equation (10)). Beyond the swap of v with $\dot{X}_t$, MeanFlow also `StopGrad`'s all model derivatives in the loss to avoid backpropogation through differentiation. We also train with three losses proposed in (Boffi et al., 2025): the Lagrangian loss, Eulerian loss, and Progressive loss. These latter three losses are derived under similar principles as CurlFlow and are discussed in Section 5.

Method	**10 steps**	**50 steps**	**100 steps**	theory
Flow Matching	24.87	3.53	3.05	yes
Lagrange	248.76	230.43	221.22	yes
Euler	77.19	66.99	38.95	yes
Progressive	337.36	235.20	206.18	yes
Meanflow	37.32	4.54	4.23	no
CurlFlow	**12.26**	**2.88**	**2.81**	yes

Table 2: FID scores versus sampling steps on CIFAR-10 computed from 50,000 EMA samples after 200,000 training steps. The "theory" column means whether the stationary points of the optimization have been proven to exist if and only if the function is the flow map that integrates the ODE.

Results. We report the Frechet Inception Distance (FID) (Heusel et al., 2017) in Table 2. We find that CurlFlow produces better FID than MeanFlow at each choice of sampling steps for the given (rather common for CIFAR) training configuration. The Lagrangian, Euler, and Progressive losses seem not to train well with the standard hyperparameters; Boffi et al. (2025) make remarks after presenting the losses that `StopGrad` may be used for some of the featured nested-model or product-of-model terms to stabilize training for certain datasets, and seem to benefit from additional adaptive loss reweighting, which are not explored in this work.

4.3 Memory Usage

In the last column of Table 1, we note whether a method requires differentiation through an iterated model call or a product of model evaluations. As discussed in Section 5, for a flow map model $f_\theta = x + (u-t)\tilde{f}_\theta$, the Lagrangian loss involves a nested evaluation $\tilde{f}_\theta(u, u, f_\theta(t, u, x))$, with a similar nesting for the Progressive loss. The Eulerian loss requires computing $\nabla_x f_\theta(t, u, x)\ \tilde{f}_\theta(t, t, X_t)$, which entails a product-rule expansion. Here we empirically compare the *peak GPU memory usage during the backward pass* for different losses, holding the architecture and data size fixed (the U-Net with a batch of CIFAR images).

Flow Matching	**MeanFlow**	CurlFlow	**Lagrange**	**Euler**	**Progressive**
16.8 Gb	14.2 Gb	43.2 Gb	69.8 Gb	69.8b G	54.3 Gb

Table 3: **Peak GPU memory usage during backward pass (in Gb).** Values reflect the maximum allocated memory measured across the training step's backward pass (i.e., during gradient computation). CurlFlow strikes a balance in GPU memory usage, preserves the true flow map as an optimum, and optimizes through all model derivatives without detaching.

Results. We demonstrate the peak GPU memory usage during backward pass in Table 3. As expected, the Lagrange, Euler, and Progressive losses are the most memory-intensive, reflecting the need to backpropagate through nested model evaluations or product-rule terms. On the other end, MeanFlow, which computes Jacobian–vector products but detaches the full JVP, has the lowest memory usage, but this detachment may limit its ability to fully optimize toward satisfying the derivative conditions. As discussed, MeanFlow also does not preserve the true flow map as an optimum in theory (though it may empirically demonstrate good performance for certain datasets). CurlFlow falls between thesee extremes, striking a balance between memory usage, preserving the optimum, and benefiting from optimizing through the model derivatives in the loss.

5 Related Work

Sampling from continuous-time generative models such as diffusion and flow models requires numerical integration. Each integration step requires a forward pass of a neural network, leading to computational costs and slow sampling. Current approaches to address this cost can be broadly categorized into two types (1) distilling a pretrained diffusion or flow model into a few-step solver (Salimans and Ho, 2022; Kim et al., 2023; Liu et al., 2023), and (2) learning a few-step solver (Zhou et al., 2025). Some approaches in this area allow for distillation as well as training from scratch (Song et al., 2023; Boffi et al., 2024; Boffi and Vanden-Eijnden, 2023).

Consistency Models (CMs) (Song et al., 2023; Song and Dhariwal, 2023; Lu and Song, 2024) learn a one-step map from noise to data, either by distilling a pretrained model or by learning from scratch. Distillation requires sampling trajectories from the teacher model. To allow for more steps after either training approach, CMs iteratively re-noise the one-step solution back to successively smaller time under the interpolant and then denoise, but this can take the solver off the probability flow.

Consistency trajectory models (CTMs) (Kim et al., 2023) extend CMs to learn two-time maps using a combination of consistency and adversarial objectives, which requires training an additional discriminator model (Goodfellow et al., 2014). CTM and Gameflow both target the same mathematical object, the probability–flow ODE flow map (i.e., the integral of the ODE), but they learn this map through different means. CTM learns the map by distilling a teacher solver, and the losses for teacher and student involve several nested model evaluations (with data at x_0, for $0 \le s \le u \le t \le 1$, the teacher integrates from t to u, then jumps from u to s, then from s to 0; and the student jumps from t to s and then to 0). The objective depends on a chosen feature-space distance and, in practice, includes DSM and GAN terms that further influence the optimum. Consequently, the CTM loss is sensitive to the quality of the ODE discretization used by the teacher (in practice CTM finds the need to use a 2nd order solver during training) and necessitates the presence of the GAN. GANs can, in principle, be used to augment any generative model or even solve the problem itself. In an idealized limit with perfect teacher solves of the velocity field, unlimited model capacity, perfect min–max optimization, no auxiliary GAN loss, annd no optimization issues due to stopgrad, the true flow map is a solution of the CTM objective because the student matches the teacher everywhere, which

matches the ODE. CTM, however, does not show that this solution is unique or that its stationary points during optimization coincide with those of the underlying PDE. CurlFlow learns a two-time map from scratch directly from the flow map PDE, uses neither discriminators nor pretrained models and proves that the stop-gradient optimization has stationary points if and only if the model equals the true flow map.

Inductive Moment Matching (IMM) (Zhou et al., 2025) learns a few-step model via an implicit generative model trained with MMD (Smola et al., 2006; Gretton et al., 2012), with the MMD which is estimated biasedly within subsets of data. In practice, the authors must use time-weighting schedules and specific curriculum/inductive procedure to stabilize optimization. While IMM produces high quality image samples, solves the problem of marginally sampling the data distribution rather than sampling along a probability flow, where the latter is the task studied in this work.

MeanFlow (Geng et al., 2025) derives a JVP-based objective for flow maps. They train $\tilde{f}$ to bring x_u down to X_t via the parameterization $f_\theta = x + (u-t)\tilde{f}_\theta$, and train $\tilde{f}$ via the following loss:

$$\mathcal{L}^{\text{meanflow}}_{t,u} := \mathbb{E}[\|\tilde{f}_\theta(t,u,X_t) - \dot{x}_u + (u-t)(\text{sg}[\partial_x \tilde{f}_\theta]_{(t,u,X_t)} \cdot \dot{x}_u + \text{sg}[\partial_u \tilde{f}_\theta]_{(t,u,X_t)})\|^2]. \quad (13)$$

Applying the stopgrad sg to all model derivatives improves efficiency, but there are no differentiated loss terms that encourage the model derivatives $\partial_t \hat{f}$ and $\partial_x \hat{f}$ to move toward the true flow map derivatives. This contrasts CurlFlow where $\text{sg}[\tilde{f}_\theta(t,t,x)]$ is used in place of $\mathbb{E}[\dot{X}_t|X_t]$, but where another term in the loss trains these two quantities to match. Finally, and importantly, between equations (10, 11) in Geng et al. (2025), v is replaced with $\dot{X}_t$ where it appears quadratically, thereby pulling an expectation through a square and missing a resulting trace covariance term. This does not preserve the true flow map as an optimum (this is mentioned as well in Boffi et al. (2025)).

Flow Map Matching (Boffi et al., 2024) learns a two-time flow map. This allows for mapping along the probability flow in either direction, without adversarial training. Their Lagrangian Flow Map Matching loss requires only time derivatives, but an additional invertibility loss encouraging invertibility via time-swapping so that $\hat{f}(t,u,\hat{f}(u,t,x)) \approx x$. While straightforward to compute, gradient steps require evaluating the model and its inverse at each step of training.

Boffi et al. (2025) build on Boffi et al. (2024) and optimize velocity matching along with one of the three following losses. For the parameterization $f_\theta := x + (u-t)\tilde{f}_\theta$, the first one is:

$$\text{LSD}: \mathcal{L}^{\text{lagrange}}_{t,u} := \mathbb{E}[\|\partial_u f_\theta(t,u,X_t) - \tilde{f}_\theta(u,u,f_\theta(t,u,X_t))\|^2]$$

LSD comes from condition that $\partial_t f(t,u,x) = v(u,x_u) = v(u,f(t,u,X_t))$ uses $\partial_t f(t,t,\cdot) = -v(t,\cdot) \implies \tilde{f}(t,t,\cdot) = v(t,\cdot)$. It is a variant of the Lagrangian loss from Boffi et al. (2024). The next is:

$$\text{ESD}: \mathcal{L}^{\text{euler}}_{t,u} := \mathbb{E}[\|\partial_t f_\theta(t,u,X_t) + \nabla_x f_\theta(t,u,X_t)\tilde{f}_\theta(t,t,X_t)\|^2].$$

ESD comes from condition $\partial_t f(t,u,x) + \nabla_x f(t,t,x)v(t,x) = 0$ where in the loss, v is replaced with $\tilde{f}(t,t,x)$. The last one, for an intermediate time m, is:

$$\text{PSD}: \mathcal{L}^{\text{progress}}_{t,u} := \mathbb{E}[\|f_\theta(t,u,X_t) - f_\theta(m,u,f_\theta(t,m,X_t))\|^2]$$

PSD comes from the composition property: $f(t,u,x) = f(m,u,f(t,m,x))$ for $m \in (t,u)$. CurlFlow, LSD, ESD, and PSD all aim to enforce a similar set of flow map properties. $\mathcal{L}^{\text{lagrange}}$ and $\mathcal{L}^{\text{progress}}$ must optimize through nested model evaluations, doubling the computational graph for backprop. $\mathcal{L}^{\text{euler}}$ optimizes through $\nabla_x f_\theta \cdot \tilde{f}_\theta$, which causes reverse-mode autodifferentiation to invoke the full product rule term, $\nabla_\theta[\nabla_x f_\theta \cdot \tilde{f}_\theta] = [\nabla_\theta \nabla_x f_\theta] \cdot \tilde{f}_\theta + [\nabla_x f_\theta] \cdot \nabla_\theta \tilde{f}_\theta$; In contrast, CurlFlow's stopgrad operation means that the second product rule term is not featured in the computational graph, saving memory and compute. But, unlike MeanFlow, the term that is subject to stopgrad receives supervision from elsewhere in the loss.

Distillation methods. A complementary line of work approaches flow map learning by *distilling* the outputs of pretrained flow matching velocity models into few-step solvers. Specifically the unknown $v()$ in the flow map identities is taken to be a pretrained network. By contrast, we emphasize training from scratch, avoiding dependence on a teacher model and ensuring that all components of the flow map are learned end-to-end. That said, distillation can be attractive in practice when a pretrained model is already trusted (when v_θ corresponds to the endpoint distributions and the chosen α_t and σ_t, or when the objective is weaker—for example, to marginally sample from the approximated data distribution without explicitly solving the probability flow ODE).

6 Discussion and Limitations

Identifying functions through PDEs *Given a PDE solved by a sought-after mapping f,* featuring a combination terms such as $\partial_t f, \partial_x f, v$ *that should be set to 0, which terms should be parameterized by the model and which should be approximated as part of a ground-truth loss target?* If the equations can be rewritten in several ways, which yield easier or more challenging objectives? Answering this is applicable to improving training objectives for generative models as well as more generally solving PDEs with machine learning.

Invertibility The loss targets an invertible function at optimum. To simplify training, we explicitly give up knowing the inverse, meaning that we only learn maps in one direction. Luckily, this is the usual scenario for generative modeling. For likelihoods, one can still substitute $-\partial_t \hat{f}$ for v_θ in the probability flow ODE (Song et al., 2021; Boffi and Vanden-Eijnden, 2023). Thus this method can be seen from the perspective of training a normalizing flow (Tabak and Vanden-Eijnden, 2010; Tabak and Turner, 2013; Rezende and Mohamed, 2015; Papamakarios et al., 2021) without requiring the invertible architecture or inverse-dependent loss.

Architectures. CurlFlow, Flow Map Matching, Simplified Consistency Models, and MeanFlow all specify models whose *time-derivative* equal the target of diffusion model training, but directly adapt architectures meant for diffusion models themselves. Example architectures used in these works are the UNet from Dhariwal and Nichol (2021), the diffusion transformer from Peebles and Xie (2023); Ma et al. (2024), and the EDM architecture from Karras et al. (2022; 2024). These architectures may thus be suboptimal for the problem at hand, precisely because the target of interest is defined as a function often computed in many diffusion model forward passes (an integral). In this work, compute limitations did not allow for the thorough exploration of architectures, but the authors believe that rethinking architectures is a convincing direction to improve the quality and training-efficiency of learned flow maps.

Reproducability Statement. We will be glad to open source the complete code during or after review of the manuscript. For the proofs in the appendix, we have made a genuine attempt to be thorough and pedagogical.

REFERENCES

Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., Ronneberger, O., Willmore, L., Ballard, A. J., Bambrick, J., et al. (2024). Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500.

Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. (2023). Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*.

Albergo, M. S. and Vanden-Eijnden, E. (2022). Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*.

Albergo, M. S. and Vanden-Eijnden, E. (2024). Learning to sample better. *Journal of Statistical Mechanics: Theory and Experiment*, 2024(10):104014.

Bartosh, G., Vetrov, D. P., and Andersson Naesseth, C. (2024). Neural flow diffusion models: Learnable forward process for improved diffusion modelling. *Advances in Neural Information Processing Systems*, 37:73952–73985.

Boffi, N. M., Albergo, M. S., and Vanden-Eijnden, E. (2024). Flow map matching. *arXiv preprint arXiv:2406.07507*.

Boffi, N. M., Albergo, M. S., and Vanden-Eijnden, E. (2025). How to build a consistency model: Learning flow maps via self-distillation. *arXiv preprint arXiv:2505.18825*.

Boffi, N. M. and Vanden-Eijnden, E. (2023). Probability flow solution of the fokker–planck equation. *Machine Learning: Science and Technology*, 4(3):035012.

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

Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., et al. (2024). Scaling rectified flow transformers for high-resolution image synthesis, 2024. *URL https://arxiv.org/abs/2403.03206*, 2.

Geng, Z., Deng, M., Bai, X., Kolter, J. Z., and He, K. (2025). Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Ho, J. and Salimans, T. (2022). Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.

Karras, T., Aittala, M., Aila, T., and Laine, S. (2022). Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577.

Karras, T., Aittala, M., Lehtinen, J., Hellsten, J., Aila, T., and Laine, S. (2024). Analyzing and improving the training dynamics of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24174–24184.

Kim, D., Lai, C.-H., Liao, W.-H., Murata, N., Takida, Y., Uesaka, T., He, Y., Mitsufuji, Y., and Ermon, S. (2023). Consistency trajectory models: Learning probability flow ode trajectory of diffusion. *arXiv preprint arXiv:2310.02279*.

Kingma, D. P., Salimans, T., Poole, B., and Ho, J. (2021). Variational diffusion models. *arXiv preprint arXiv:2107.00630*.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. (2022). Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*.

Liu, X., Gong, C., and Liu, Q. (2022). Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*.

Liu, X., Zhang, X., Ma, J., Peng, J., et al. (2023). Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *The Twelfth International Conference on Learning Representations*.

Lu, C. and Song, Y. (2024). Simplifying, stabilizing and scaling continuous-time consistency models. *arXiv preprint arXiv:2410.11081*.

Ma, N., Goldstein, M., Albergo, M. S., Boffi, N. M., Vanden-Eijnden, E., and Xie, S. (2024). Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pages 23–40. Springer.

Pandey, K. and Mandt, S. (2023). A complete recipe for diffusion generative models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4261–4272.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64.

Peebles, W. and Xie, S. (2023). Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR.

Sabour, A., Fidler, S., and Kreis, K. (2025). Align your flow: Scaling continuous-time flow map distillation. *arXiv preprint arXiv:2506.14603*.

Salimans, T. and Ho, J. (2022). Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*.

Singhal, R., Goldstein, M., and Ranganath, R. (2023). Where to diffuse, how to diffuse, and how to get back: Automated learning for multivariate diffusions. *arXiv preprint arXiv:2302.07261*.

Singhal, R., Goldstein, M., and Ranganath, R. (2024). What's the score? automated denoising score matching for nonlinear diffusions. In *International Conference on Machine Learning*, pages 45734–45758. PMLR.

Smola, A. J., Gretton, A., and Borgwardt, K. (2006). Maximum mean discrepancy. In *13th international conference, ICONIP*, volume 6.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr.

Song, Y. and Dhariwal, P. (2023). Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*.

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. (2023). Consistency models. *arXiv preprint arXiv:2303.01469*.

Song, Y., Durkan, C., Murray, I., and Ermon, S. (2021). Maximum likelihood training of score-based diffusion models. *Advances in neural information processing systems*, 34:1415–1428.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Tabak, E. G. and Turner, C. V. (2013). A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164.

Tabak, E. G. and Vanden-Eijnden, E. (2010). Density estimation by dual ascent of the log-likelihood. *Commun. Math. Sci. 8 (1) 217 - 233, March 2010.*

Zhou, L., Ermon, S., and Song, J. (2025). Inductive moment matching. *arXiv preprint arXiv:2503.07565*.

A BACKGROUND

A.1 CONTINUITY EQUATION AND DERIVATION OF EXPRESSION FOR VELOCITY

Theorem 2 (Continuity Equation for Interpolants)**.** *Let $x_0 \sim q_0$ and $x_1 \sim q_1$. Let $x_t = \alpha_t x_0 + \sigma_t x_1$ for $\alpha_0 = \sigma_1 = 1$ and $\alpha_1 = \sigma_0 = 0$. Let $\dot{\alpha}_t$ and $\dot{\sigma}_t$ denote $\frac{d}{dt}\alpha_t$ and $\frac{d}{dt}\sigma_t$, respectively. Then $x_t \sim q_t$ where q_t satisfies*

$$\partial_t q_t(x) = -\nabla_x \cdot (q_t(x) v(t,x)), \qquad v(t,x) := \mathbb{E}[\dot{\alpha}_t x_0 + \dot{\sigma}_t x_1 \mid x_t = x] \tag{14}$$

Proof. In this derivation, we use * for scalar-vector multiplication and $\cdot$ for dot product. In the following, $x \in \mathbb{R}^d$ and we write some functions evaluated at an arbitrary frequency $k \in \mathbb{R}^d$. We derive "Fourier transform of time derivative of density equals Fourier transform of something" and then invert the Fourier on both sides.

Let $\mathcal{F}[f](k) := \int \exp(ik \cdot x) f(x) dx$. Then, for any density $p(x)$ we have

$$\mathcal{F}[p](k) = \mathbb{E}[\exp(ik \cdot x)] \tag{15}$$

Let us reveal x_t as a function $x_t(x_0, x_1)$ so that $v(x,t) = \mathbb{E}[\dot{x}_t(x_0,x_1)|x_t = x]$. Then,

$$\begin{aligned}
\partial_t \mathcal{F}[q_t(x_t)](k) &= \partial_t \, \mathbb{E}\left[e^{ik \cdot x_t}\right] && (16)\\
&= \partial_t \int_{x_0} \int_{x_1} e^{ik \cdot x_t(x_0,x_1)} dq(x_0,x_1) && (17)\\
&= \int_{x_0} \int_{x_1} \partial_t e^{ik \cdot x_t(x_0,x_1)} dq(x_0,x_1) && (18)\\
&\left[\partial_t e^{ik \cdot x_t(x_0,x_1)} = ik \cdot \left(e^{ik \cdot x_t(x_0,x_1)} \dot{x}_t\right)\right] && (19)\\
&= ik \cdot \int_{x_0} \int_{x_1} \left(e^{ik \cdot x_t(x_0,x_1)} \dot{x}_t(x_0,x_1)\right) dq(x_0,x_1) && (20)\\
&= ik \cdot \int_{x_0} \int_{x_1} \left(e^{ik \cdot x_t(x_0,x_1)} \dot{x}_t(x_0,x_1)\right) dq(x_0,x_1) && (21)\\
&= ik \cdot \mathop{\mathbb{E}}_{q(x_t)} \mathop{\mathbb{E}}_{q(x_1,x_0|x_t)} \left[e^{ik \cdot x_t} \dot{x}_t(x_0,x_1)\right] && (22)\\
&= ik \cdot \mathop{\mathbb{E}}_{q(x_t)} \left[e^{ik \cdot x_t} \, \mathbb{E}[\dot{x}_t(x_0,x_1)|x_t]\right] && (23)\\
&= ik \cdot \mathop{\mathbb{E}}_{q(x_t)} \left[e^{ik \cdot x_t} v(x_t,t)\right] && (24)\\
&= ik \cdot \int \left[e^{ik \cdot x} v(x,t)\right] q_t(x) dx && (25)\\
&\left[\text{integration by parts, boundary } \int q_t dS = 0 \text{ for normalized densities}\right] && (26)\\
&= \int \left(v(x,t) q_t(x)\right) \cdot \left(\nabla_x e^{ik \cdot x}\right) dx && (27)\\
&= -\int e^{ik \cdot x} * \nabla \cdot (v * q_t) dx && (28)\\
&= -\mathcal{F}[\nabla \cdot (v * q_t)](k) && (29)
\end{aligned}$$

We thus have

$$\partial_t \mathcal{F}[q_t](k) = -\mathcal{F}[\nabla \cdot (v q_t)](k) \tag{30}$$

But, for the same LHS we can suppress x_t's dependence on (t, x_0, x_1) and instead use its marginal distribution:

$$\partial_t \mathcal{F}[q_t](k) = \partial_t \Big(\int e^{ik \cdot x} q_t(x) dx \Big) = \int e^{ik \cdot x} \Big(\partial_t q_t(x)\Big) dx = \mathcal{F}[\partial_t p_t](k) \tag{31}$$

Putting the two right hand sides next to each other:

$$\mathcal{F}[\partial_t q_t](k) = -\mathcal{F}[\nabla \cdot (v q_t)](k) \tag{32}$$

Taking an inverse Fourier transform,

$$\partial_t q_t = -\nabla \cdot (v q_t) \tag{33}$$

This establishes the expression for $\partial_t q_t$ and the expression for v. □

A.2 Replacing velocity with interpolant time derivative in squared error losses

In diffusions, flow matching, and in this setting, we deal with objectives of the form

$$\min_{\theta} \mathbb{E}_{q(x_t)} \left[\|m_\theta(t, x_t) - v(t, x_t)\|^2, \right] \tag{34}$$

where $x_t \sim q_t$ is drawn by the interpolant $x_t = \alpha_t x_0 + \sigma_t x_1$, v is the velocity $v(t, x) = \mathbb{E}[\dot{\alpha}_t x_0 + \dot{\sigma}_t x_1]$, and m_θ is some model (or its derivative, etc) evaluated at (t, x_t). Here we show the common set of steps to replace the intractable v in the loss but preserve the right minimizer for m_θ, which we use in our loss derivations.

$$\mathbb{E}_{q(x_t)} \left[\|m_\theta(t, x_t) - \mathbb{E}[\dot{\alpha}_t x_0 + \dot{\sigma}_t x_1 | x_t = x]\|^2 \right] \tag{35}$$

$$= \mathbb{E}_{q(x_t)} \left[\|m_\theta(t, x_t)\|^2 + \|\mathbb{E}[\dot{\alpha}_t x_0 + \dot{\sigma}_t x_1 | x_t = x]\|^2 - 2m_\theta(t, x_t)^\top \mathbb{E}[\dot{\alpha}_t x_0 + \dot{\sigma}_t x_1 | x_t = x] \right] \tag{36}$$

$$= \mathbb{E}_{q(x_t)} \Bigg[\|m_\theta(t, x_t)\|^2 + \mathbb{E}[\|\dot{\alpha}_t x_0 + \dot{\sigma}_t x_1\|^2 \mid x_t = x] \tag{37}$$

$$\underbrace{- \sum_j \text{Var}([\dot{\alpha}_t x_0 + \dot{\sigma}_t x_1]_j \mid x_t = x])}_{=:\text{constant } C} - 2m_\theta(t, x_t)^\top \mathbb{E}[\dot{\alpha}_t x_0 + \dot{\sigma}_t x_1 | x_t = x] \Bigg] \tag{38}$$

$$= \mathbb{E}_{q(x_t) q(x_0, x_1 | x_t)} \left[\|m_\theta(t, x_t)\|^2 + \|\dot{\alpha}_t x_0 + \dot{\sigma}_t x_1\|^2 - 2m_\theta(t, x_t)^\top \left(\dot{\alpha}_t x_0 + \dot{\sigma}_t x_1 \right) \right] + C \tag{39}$$

$$= \mathbb{E}_{q(x_t) q(x_0, x_1 | x_t)} \left[\|m_\theta(t, x_t) - (\dot{\alpha}_t x_0 + \dot{\sigma}_t x_1)\|^2 \right] + C \tag{40}$$

$$= \mathbb{E}_{q(x_0, x_1, x_t)} \left[\|m_\theta(t, x_t) - (\dot{\alpha}_t x_0 + \dot{\sigma}_t x_1)\|^2 \right] + C \tag{41}$$

$$= \mathbb{E}_{q(x_0) q(x_1)} \left[\|m_\theta(t, x_t) - (\dot{\alpha}_t x_0 + \dot{\sigma}_t x_1)\|^2 \right]_{x_t = \alpha_t x_0 + \sigma_t x_1} + C \tag{42}$$

Dropping the constant preserves the minimizer:

$$\min_{\theta} \mathbb{E}_{q(x_0) q(x_1)} \left[\|m_\theta(t, x_t) - (\dot{\alpha}_t x_0 + \dot{\sigma}_t x_1)\|^2 \right]_{x_t = \alpha_t x_0 + \sigma_t x_1} \tag{43}$$

B PROOFS FOR STATIONARY POINTS

B.1 FIRST VARIATION DEFINITIONS

We consider scalar-valued loss functions $\mathcal{L} : \mathcal{F} \to \mathbb{R}$ that map a function $f \in \mathcal{F}$ to a real value.

Define the tangent space $\mathcal{T}_f(\mathcal{F})$ at f. This space contains functions $h \in \mathcal{T}_f(\mathcal{F})$ such that there exists a curve indexed by scalar ϵ such that for each ϵ, $f_\epsilon \in \mathcal{F}$, and we have that $f_0 = f$ and $(\frac{d}{d\epsilon} f_\epsilon)|_{\epsilon=0} = h$.

The first variation $\delta\mathcal{L}$ of such a functional $\mathcal{L}$ evaluated at $f \in \mathcal{F}$ in direction $h \in \mathcal{T}_f(\mathcal{F})$ is defined as:

$$\delta\mathcal{L}[f; h] := \Big(\frac{d}{d\epsilon}\mathcal{L}[f + \epsilon h]\Big)_{\epsilon=0} \tag{44}$$

We then have that f^* is a stationary point w.r.t. $\mathcal{F}$ if $\delta[f; h] = 0$ for all $h \in \mathcal{T}_f(\mathcal{F})$.

B.2 STOPGRAD FOR FUNCTIONALS

We define the stopgrad symbol sg for a functional as follows. Let $\mathcal{O}$ be a functional that maps two functions f, g to a real value. Let $\mathcal{L}[f]$ be a functional that is written in terms of $\mathcal{O}$ with symbol sg as $\mathcal{L}[f] := \mathcal{O}[f, \text{sg}[f]]$, then we evaluate the following two quantities as follows

$$\mathcal{L}[f] = \mathcal{O}[f, f] \tag{45}$$

$$\delta\mathcal{L}[f; h] = \delta\mathcal{O}[f, f; h, 0] \tag{46}$$

That is, the functional evaluates as usual but in a first variation, we do not perturb terms in sg. This corresponds to the stopgrad or detach() operation used in machine learning code with autodifferentiation.

B.3 First Variation of Original Loss Functional

Our functional $\mathcal{L}[\tilde{f}]$ acts on functions $\tilde{f}$. According to the definitions in Section B.1, we need to compute $\delta\mathcal{L}[\tilde{f}; h] = (\frac{d}{d\epsilon}\mathcal{L}[\tilde{f} + \epsilon h])_{\epsilon=0}$. The functional is:

$$
\begin{aligned}
T_1[\tilde{f}] &:= \|\|(\partial_t f)_{(t,u,x_t)} + (\partial_x f)_{(t,u,x_t)}\dot{x}_t\|^2_{f=x+(u-t)\tilde{f}} \\
T_2[\tilde{f}] &:= \|(\partial_x f)_{(t,u,x_t)}(\dot{x}_t - \mathbb{E}[\dot{x}_t|x_t])\|^2_{f=x+(u-t)\tilde{f}} \\
A[\tilde{f}] &= T_1[\tilde{f}] - T_2[\tilde{f}] \\
\mathcal{L}[\tilde{f}] &= \mathbb{E}_{q(t,u),q(x_0),q(x_1)}\Big[A[\tilde{f}]\Big]
\end{aligned}
$$

Lets define the path f_ϵ by replacing $\tilde{f}$ with $\tilde{f}_\epsilon := \tilde{f} + \epsilon h$. Then:

$$
f_\epsilon := x + (u-t)\tilde{f}_\epsilon = x + (u-t)(\tilde{f} + \epsilon h) = x + (u-t)\tilde{f} + \epsilon(u-t)h \tag{47}
$$

Then

$$
\frac{d}{d\epsilon}\mathcal{L}[\tilde{f} + \epsilon h] = \mathbb{E}\Big[\frac{d}{d\epsilon}A[\tilde{f}_\epsilon]\Big] = \mathbb{E}\Big[\frac{d}{d\epsilon}T_1[\tilde{f} + \epsilon h] - \frac{d}{d\epsilon}T_2[\tilde{f} + \epsilon h]\Big] \tag{48}
$$

We first compute this derivative and then evaluate it at $\epsilon = 0$.

So

$$
\begin{aligned}
\partial_t f_\epsilon &= \partial_t\Big[x + (u-t)\tilde{f} + \epsilon(u-t)h\Big] && (49)\\
&= \partial_t x + \partial_t\Big[(u-t)\tilde{f}\Big] + \epsilon\partial_t\Big[(u-t)h\Big] && (50)\\
&= (u-t)\partial_t\tilde{f} - \tilde{f} + \epsilon\Big((u-t)\partial_t h - h\Big) && (51)\\
&= (u-t)(\partial_t\tilde{f} + \epsilon\partial_t h) - (\tilde{f} + \epsilon h) && (52)
\end{aligned}
$$

and

$$
\partial_x f_\epsilon = \partial_x\Big[x + (u-t)\tilde{f} + \epsilon(u-t)h\Big] = I + (u-t)\partial_x(\tilde{f} + \epsilon h) \tag{53}
$$

and

$$
\begin{aligned}
\frac{d}{d\epsilon}\partial_t f_\epsilon &= \frac{d}{d\epsilon}\Big[(u-t)(\partial_t\tilde{f} + \epsilon\partial_t h) - (\tilde{f} + \epsilon h)\Big] && (54)\\
&= \frac{d}{d\epsilon}\Big[(u-t)\partial_t\tilde{f} + \epsilon(u-t)\partial_t h - \tilde{f} - \epsilon h\Big] && (55)\\
&= \frac{d}{d\epsilon}\Big[\epsilon(u-t)\partial_t h - \epsilon h\Big] && (56)\\
&= (u-t)\partial_t h - h && (57)
\end{aligned}
$$

and

$$
\begin{aligned}
\frac{d}{d\epsilon}\partial_x f_\epsilon &= \frac{d}{d\epsilon}\Big[I + (u-t)\partial_x(\tilde{f} + \epsilon h)\Big] && (58)\\
&= \frac{d}{d\epsilon}I + \frac{d}{d\epsilon}(u-t)\partial_x\tilde{f} + \frac{d}{d\epsilon}(u-t)\partial_x\epsilon h && (59)\\
&= (u-t)\partial_x h && (60)
\end{aligned}
$$

For the first term,

$$
T_1[\tilde{f} + \epsilon h] = \|\partial_t f_\epsilon + (\partial_x f_\epsilon)\dot{x}_t\|^2
$$

Differentiating

$$
\begin{aligned}
\frac{d}{d\epsilon}T_1[\tilde{f} + \epsilon h] &= 2\Big(\partial_t f_\epsilon + (\partial_x f_\epsilon)\dot{x}_t\Big)^\top \frac{d}{d\epsilon}\Big(\partial_t f_\epsilon + (\partial_x f_\epsilon)\dot{x}_t\Big) && (61)\\
&= 2\Big(\partial_t f_\epsilon + (\partial_x f_\epsilon)\dot{x}_t\Big)^\top \Big(\underbrace{(u-t)\partial_t h - h} + \underbrace{(u-t)\partial_x h}\,\dot{x}_t\Big) && (62)
\end{aligned}
$$

$$= 2\Big(\underbrace{(u-t)(\partial_t \tilde{f} + \epsilon \partial_t h) - (\tilde{f} + \epsilon h)}_{} + \underbrace{(I + (u-t)\partial_x(\tilde{f} + \epsilon h))}_{} \dot{x}_t \Big)^\top \tag{63}$$

$$\Big(\underbrace{(u-t)\partial_t h - h}_{} + \underbrace{(u-t)\partial_x h}_{} \, \dot{x}_t \Big) \tag{64}$$

So

$$\frac{d}{d\epsilon} T_1[\tilde{f} + \epsilon h]\Big|_{\epsilon=0} = 2\Big((u-t)(\partial_t \tilde{f} + \partial_x \tilde{f} \dot{x}_t) - \tilde{f} + \dot{x}_t \Big)^\top \Big((u-t)(\partial_t h + \partial_x h \dot{x}_t) - h \Big) \tag{65}$$

For the second term,

$$T_2[\tilde{f} + \epsilon h] = \|\partial_x f_\epsilon (\dot{x}_t - v)\|^2 \tag{66}$$

and

$$\frac{d}{d\epsilon} T_2[\tilde{f} + \epsilon h] = 2\Big[\partial_x f_\epsilon (\dot{x}_t - v) \Big]^\top \frac{d}{d\epsilon} \Big[\partial_x f_\epsilon (\dot{x}_t - v) \Big] \tag{67}$$

$$= 2\Big[\partial_x f_\epsilon (\dot{x}_t - v) \Big]^\top \frac{d}{d\epsilon} (\partial_x f_\epsilon)(\dot{x}_t - v) \tag{68}$$

$$= 2\Big[\Big(I + (u-t)\partial_x(\tilde{f} + \epsilon h) \Big)(\dot{x}_t - v) \Big]^\top (u-t)\partial_x h (\dot{x}_t - v) \tag{69}$$

So

$$\frac{d}{d\epsilon} T_2[\tilde{f} + \epsilon h]\Big|_{\epsilon=0} = 2\Big[\Big(I + (u-t)\partial_x \tilde{f} \Big)(\dot{x}_t - v) \Big]^\top (u-t)\partial_x h (\dot{x}_t - v) \tag{70}$$

So combining

$$\frac{d}{d\epsilon} A\Big|_{\epsilon=0} = 2\Big((u-t)(\partial_t \tilde{f} + \partial_x \tilde{f} \dot{x}_t) - \tilde{f} + \dot{x}_t \Big)^\top \Big((u-t)(\partial_t h + \partial_x h \dot{x}_t) - h \Big) \tag{71}$$

$$- 2\Big[\Big(I + (u-t)\partial_x \tilde{f} \Big)(\dot{x}_t - v) \Big]^\top (u-t)\partial_x h (\dot{x}_t - v) \tag{72}$$

At $t = u$, this simplifies

$$1[t = u] \frac{d}{d\epsilon} A[\tilde{f}_\epsilon]\Big|_{\epsilon=0} = 2(\dot{x}_t - \tilde{f})^\top (-h) \tag{73}$$

which is the first variation for regression that makes $\tilde{f}$ equal to $E[\dot{x}_t | x_t]$. Summarizing,

$$\delta \mathcal{L}[\tilde{f}; h] = \mathbb{E}\Big[2\Big((u-t)(\partial_t \tilde{f} + \partial_x \tilde{f} \dot{x}_t) - \tilde{f} + \dot{x}_t \Big)^\top \Big((u-t)(\partial_t h + \partial_x h \dot{x}_t) - h \Big) \tag{74}$$

$$- 2\big[(I + (u-t)\partial_x \tilde{f})(\dot{x}_t - v) \big]^\top (u-t)\partial_x h (\dot{x}_t - v) \Big] \tag{75}$$

B.4 LEMMA: VELOCITY MATCHES AT A STATIONARY POINT OF ORIGINAL FUNCTIONAL

Lemma 1. *Let $\tilde{f}^*$ be a stationary point of $\mathcal{L}$. Assume that $\tilde{f}^*$ is bounded. Assume that $\tilde{f}^*, v \in C^1$ in arguments (t, u, x) and that all expectations of terms featured in the integrand of $\mathcal{L}$ (i.e., $v, \tilde{f}, \partial_t \tilde{f}, \partial_u \tilde{f}, \partial_x \tilde{f}, \ldots$) are finite. Then we have that $\tilde{f}^*(t, t, \cdot) = v(t, \cdot)$ where the velocity $v(t, x) = \mathbb{E}[\dot{x}_t | x_t]$.*

Proof. We proceed by contradiction. By the premise, we are at a stationary point $\tilde{f}^*$. Let $f^* := x + (u - t)\tilde{f}^*$. By the definition of stationary point in section B.1, we have that $\delta\mathcal{L}[\tilde{f}^*; h] = 0$ for all admissible h. Suppose for the sake of contradiction that at this stationary point, the velocity does not match, meaning

$$-\partial_t f^*(t, t, \cdot) = \tilde{f}^*(t, t, \cdot) \underbrace{\neq}_{\text{suppose for contradiction}} v(t, \cdot) \tag{76}$$

The proof proceeds by picking a direction for which the first variation is nonzero, providing a contradiction to being at a stationary point. The contradiction (the direction for which the first variation is nonzero) is constructed to arise from assuming that the velocity does not match, meaning that by contradiction the velocity does not match. Specific care is taken to ensure that this direction is admissible, in this case meaning it is a continuous function.

We name a sequence of functions g_η such there exists η^* such that g_{η^*} is continuous but yields the nonzero variation when chosen as a direction. To establish this is existence under continuity, the dominated convergence theorem is used.

Let us define $g(t, u, x) = 1[t = u]\Big(\tilde{f}^*(t, u, x) - v(t, x)\Big)$ and evaluate it at $t = u$ so that $g(t, t, x) = \tilde{f}^*(t, t, x) - v(t, x)$. We then define the soft indicator $I_\eta(t, u)$ that goes to $1[t = u]$ as $\eta \to 0$ and define it as:

$$I_\eta(t, u) = 1[\eta > 0]2\Big(1 - \frac{1}{1 + \exp(-\frac{1}{\eta^2}(t - u)^2)}\Big) + 1[\eta = 0]1[t = u] \tag{77}$$

Using the soft indicator, we define a sequence of functions g_η so that as $\eta \to 0$ we will have pointwise convergence of $g_\eta(t, t, x) \to g(t, t, x)$ which also means that $g_\eta(t, u, x)$ for $t \neq u$ goes to 0. We pick

$$g_\eta(t, u, x) = I_\eta(t, u)\Big(\tilde{f}^*(t, t, x) - v(t, x)\Big) \tag{78}$$

Pointwise convergence of g in eta. We first establish pointwise convergence of g_η to g for all arguments (t, u, x) as $\eta \to 0$ from the right.

$$\forall \hat{\eta} \geq 0, \quad \lim_{\eta \to (\hat{\eta})^+} g_\eta(t, u, x) = g_{\hat{\eta}}(t, u, x) \tag{79}$$

To show this, for any $\hat{\eta} > 0$, use continuity of $g_{\hat{\eta}}(t, u, x)$ in $\hat{\eta}$ (product of function without η times the soft indicator which is continuous). Then to establish for $\hat{\eta} = 0$, we consider two cases $t = u$ and $t \neq u$. For equality:

$$\lim_{\eta \to 0^+} g_\eta(t, t, x) \tag{80}$$

$$= \lim_{\eta \to 0^+} I_\eta(t, t)\Big(\tilde{f}^*(t, t, x) - v(t, x)\Big) \tag{81}$$

$$= \lim_{\eta \to 0^+} \Big[1[\eta > 0]2\Big(1 - \frac{1}{1 + \exp(0)}\Big) + 1[\eta = 0]\Big]\Big(\tilde{f}^*(t, t, x) - v(t, x)\Big) \tag{82}$$

$$= \lim_{\eta \to 0^+} \Big[1[\eta > 0]1 + 1[\eta = 0]\Big]\Big(\tilde{f}^*(t, t, x) - v(t, x)\Big) \tag{83}$$

$$= \lim_{\eta \to 0^+} 1[\eta \geq 0]\Big(\tilde{f}^*(t, t, x) - v(t, x)\Big) \tag{84}$$

$$= \tilde{f}^* - v \tag{85}$$

which equals $g_{\eta=0}(t,t,x)$, establishing continuity. Now for $t \neq u$. $\forall \delta > 0$ we must name an $\eta(\delta)$ such that $g_{\eta(\delta)}$ such that $|g_{\eta(\delta)} - g_0| < \delta$ i.e. $|g_{\eta(\delta)} - 0| < \delta$. Assume $|\tilde{f}^* - v| < k$ uniformly in all input values t,x.

$$\lim_{\eta \to 0^+} g_\eta(t,u,x)$$
$$= \lim_{\eta \to 0^+} \Big[1[\eta > 0]2\Big(1 - \frac{1}{1+\exp(-\frac{1}{\eta^2}(t-u)^2)}\Big) + 1[\eta = 0]1[t=u]\Big]\Big(\tilde{f}^*(t,t,x) - v(t,x)\Big)$$
$$= \lim_{\eta \to 0^+} \Big[1[\eta > 0]2\Big(1 - \frac{1}{1+\exp(-\frac{1}{\eta^2}(t-u)^2)}\Big)\Big]\Big(\tilde{f}^*(t,t,x) - v(t,x)\Big)$$

Since we are finding a δ and $\eta(\delta)$ so that $|g_{\eta(\delta)} - 0| < \delta$ which means $|g_{\eta(\delta)}| < \delta$, this just means we can set δ to an upper bound on the term we are limiting: let the indicator take on 1 as when it is 0 we are done.

$$\delta = \Big|2(1 - \frac{1}{1+\exp(...)})\Big|k \tag{86}$$
$$\iff \frac{\delta}{k} = 2(1 - \frac{1}{1+\exp(-\frac{1}{\eta^2}(t-u)^2)}) \tag{87}$$
$$\iff \frac{\delta}{2k} = 1 - \frac{1}{1+\exp(-\frac{1}{\eta^2}(t-u)^2)} \tag{88}$$
$$\iff 1 - \frac{\delta}{2k} = \frac{1}{1+\exp(-\frac{1}{\eta^2}(t-u)^2)} \tag{89}$$
$$\iff \frac{2k}{2k} - \frac{\delta}{2k} = \frac{1}{1+\exp(-\frac{1}{\eta^2}(t-u)^2)} \tag{90}$$
$$\iff \frac{2k-\delta}{2k} = \frac{1}{1+\exp(-\frac{1}{\eta^2}(t-u)^2)} \tag{91}$$
$$\iff \frac{2k}{2k-\delta} = 1 + \exp(-\frac{1}{\eta^2}(t-u)^2) \tag{92}$$
$$\iff \frac{2k}{2k-\delta} - 1 = \exp(-\frac{1}{\eta^2}(t-u)^2) \tag{93}$$
$$\iff \frac{2k}{2k-\delta} - \frac{2k-\delta}{2k-\delta} = \exp(-\frac{1}{\eta^2}(t-u)^2) \tag{94}$$
$$\iff \frac{\delta}{2k-\delta} = \exp(-\frac{1}{\eta^2}(t-u)^2) \tag{95}$$
$$\iff \log\frac{\delta}{2k-\delta} = -\frac{1}{\eta^2}(t-u)^2 \tag{96}$$
$$\iff -\log\frac{\delta}{2k-\delta} = \frac{1}{\eta^2}(t-u)^2 \tag{97}$$
$$\iff -\frac{\log\frac{\delta}{2k-\delta}}{(t-u)^2} = \frac{1}{\eta^2} \tag{98}$$
$$\iff -\frac{(t-u)^2}{\log\frac{\delta}{2k-\delta}} = \eta^2 \tag{99}$$
$$\tag{100}$$

Now note that the soft indicator is strictly < 1 and that g_η for fixed (t,u,x) is between $-k$ and 0 or 0 and k depending on the sign of $\tilde{f}^* - v$, but never both. So its magnitude is at most k. So

$$|g_{\eta(\delta)} - g_0| = |g_{\eta(\delta)} - 0| < \delta < k \tag{101}$$

This can help us ascertain that the above square root to solve for η will be well defined:

$$\delta < k \implies 2k - \delta > k \tag{102}$$

$$\implies \frac{1}{2k-\delta} < \frac{1}{k} \tag{103}$$

$$\implies \frac{\delta}{2k-\delta} < \frac{\delta}{k} \tag{104}$$

$$\implies \frac{\delta}{2k-\delta} < 1 \tag{105}$$

$$\implies \log \frac{\delta}{2k-\delta} < \log 1 \tag{106}$$

$$\implies \log \frac{\delta}{2k-\delta} < 0 \tag{107}$$

meaning

$$\eta(\delta) = \sqrt{\frac{(t-u)^2}{|\log \frac{\delta}{2k-\delta}|}} \tag{108}$$

thus establishing convergence of $g_\eta \to g$ as $\eta \to 0^+$ for each (t, u, x) i.e. pointwise convergence.

Now recall the fist variation of $\mathcal{L}$ (section B.3) and consider it as a function of η:

$$s(\eta) := \delta\mathcal{L}[\tilde{f}^*; g_\eta] = \mathbb{E}\Bigg[2\Big((u-t)(\partial_t \tilde{f}^*) - \tilde{f}^* + (I + (u-t)(\partial_x \tilde{f}^*))\dot{x}_t\Big)^\top \tag{109}$$

$$\Big((u-t)\partial_t g_\eta - g_\eta + (u-t)(\partial_x g_\eta)\dot{x}_t\Big) \tag{110}$$

$$- 2\Big[\Big(I + (u-t)\partial_x \tilde{f}^*\Big)(\dot{x}_t - v)\Big]^\top (u-t)\partial_x g_\eta(\dot{x}_t - v)\Bigg] \tag{111}$$

Pointwise convergence of integrand in eta. Collect the variables $\omega = (t, u, x_0, x_1)$ and recall that x_t and $\dot{x}_t$ are functions of (x_0, x_1). Define $\phi_\eta(\omega)$ as shorthand for the expectand so that $s(\eta) = \mathbb{E}[\phi_\eta(\omega)]$. Under the boundedness assumptions and noting that ϕ_η only polynomially combines g_η with $(\partial_t \tilde{f}^*, \partial_u \tilde{f}^*, \partial_x \tilde{f}^*, v, \partial_x v, \ldots)$, similar reasoning used to show $g_\eta \to g$ can also be used to establish that $\phi_\eta \to \phi$ pointwise.

Establish upper envelope. In addition to pointwise convergence of $\phi_\eta \to \eta$ as $\eta \to 0$ from the right, we need an upper envelope $G(\omega)$. Beyond the assumptions, the only thing needed to show that an upper envelope exists is to control the term $|(u-t)\partial_t I_\eta(t,u)|$. The derivative of the soft indicator appears since $\partial_t g_\eta = \partial_t(I_\eta g) = (\partial_t I_\eta)g + I_\eta \partial_t g$. At $\eta = 0$ we have $I_\eta(t,u) = 1[t = u]$, so $(u-t)\,\partial_t I_\eta(t,u)$ vanishes identically: it is 0 for $t \neq u$ because I_0 is constant, and for $t = u$ because of the $(u-t)$ prefactor. For $\eta > 0$, define:

$$z := \frac{(t-u)^2}{\eta^2}, \quad \sigma(z) := \frac{1}{1+\exp(-z)}, \quad \sigma'(z) := \frac{\exp(-z)}{[1+\exp(-z)]^2} \tag{112}$$

Note that for $\eta > 0$,

$$\partial_t I_\eta(t,u) = 2\frac{2(t-u)}{\eta^2}\sigma'(\frac{(t-u)^2}{\eta^2}) \tag{113}$$

and so

$$|(u-t)\partial_t I_\eta(t,u)| = 2(u-t)\frac{2(u-t)}{\eta^2}\sigma'(\frac{(t-u)^2}{\eta^2}) = 4z\sigma'(z) \leq \sup_{z\geq 0} 4z\sigma'(z) \leq C_0 < \infty \tag{114}$$

Because $r(z) := 4z\sigma'(z)$ is continuous and satisfies $r(0) = 0$ and $r(z) \to 0$ as $z \to \infty$, it attains a finite maximum. **This bound is independent of η.** Thus every term containing $(u-t)\partial_t g_\eta$ is uniformly bounded in η by a product of a constant (from the bound and $I_\eta \in [0,1)$). The other quantities in ϕ are bounded by assumption. Thus such a bounding envelope $G(\omega)$ exists.

Using dominated convergence. First,

$$\lim_{\eta\to 0^+} s(\eta) = \lim_{\eta\to 0^+} \mathbb{E}[\phi_\eta] = \lim_{\eta\to 0^+} p(t=u)\mathbb{E}[\phi_\eta \,|\, t=u] + p(t<u)\mathbb{E}[\phi_\eta \,|\, t<u] \tag{115}$$

By the pointwise convergence of $\phi_\eta \to \phi$ as $\eta \to 0^+$ and by the envelope, we can compute the limit of the first and second terms separately. Expanding the first term (with $p = u$):

$$\begin{aligned}
\lim_{\eta\to 0^+} p(t=u)\mathbb{E}[\phi_\eta \,|\, t=u] &= \lim_{\eta\to 0^+} p(t=u)\mathbb{E}\Bigg[2\Big((u-t)(\partial_t \tilde{f}^*) - \tilde{f}^* + (I + (u-t)(\partial_x \tilde{f}^*))\dot{x}_t\Big)^\top \\
&\quad \Big((u-t)\partial_t g_\eta - g_\eta + (u-t)(\partial_x g_\eta)\dot{x}_t\Big) \\
&\quad - 2\Big[\Big(I + (u-t)\partial_x \tilde{f}^*\Big)(\dot{x}_t - v)\Big]^\top (u-t)\partial_x g_\eta(\dot{x}_t - v) \,|\, t=u\Bigg] \\
&= \lim_{\eta\to 0^+} p(t=u)\mathbb{E}\Bigg[2\Big(-\tilde{f}^* + \dot{x}_t\Big)^\top\Big(-g_\eta\Big) \,|\, t=u\Bigg] \\
&= \lim_{\eta\to 0^+} p(t=u)\mathbb{E}\Bigg[2\Big(-\tilde{f}^* + \mathbb{E}[\dot{x}_t \,|\, x_t]\Big)^\top\Big(-g_\eta\Big) \,|\, t=u\Bigg] \\
&= \lim_{\eta\to 0^+} p(t=u)\mathbb{E}\Bigg[2\Big(-\tilde{f}^* + v\Big)^\top\Big(-g_\eta\Big) \,|\, t=u\Bigg] \\
&= p(t=u)\mathbb{E}\Bigg[\lim_{\eta\to 0^+} 2\Big(-\tilde{f}^* + v\Big)^\top\Big(-g_\eta\Big) \,|\, t=u\Bigg] \\
&= p(t=u)\mathbb{E}\Bigg[\lim_{\eta\to 0^+} 2||-\tilde{f}^* + v||_2^2 \,|\, t=u\Bigg]
\end{aligned}$$

This term is greater than zero by assumption that the velocity does not match at the stationary point and assumption on the positive probabiity on $p(t=u) > 0$.

Expanding the second term (with $p(t<u)$):

$$\begin{aligned}
\lim_{\eta\to 0^+} p(t<u)\mathbb{E}[\phi_\eta \,|\, t<u] &= \lim_{\eta\to 0^+} p(t<u)\mathbb{E}\Bigg[2\Big((u-t)(\partial_t \tilde{f}^*) - \tilde{f}^* + (I + (u-t)(\partial_x \tilde{f}^*))\dot{x}_t\Big)^\top \\
&\quad \Big((u-t)\partial_t g_\eta - g_\eta + (u-t)(\partial_x g_\eta)\dot{x}_t\Big) \\
&\quad - 2\Big[\Big(I + (u-t)\partial_x \tilde{f}^*\Big)(\dot{x}_t - v)\Big]^\top (u-t)\partial_x g_\eta(\dot{x}_t - v) \,|\, t<u\Bigg] \\
&= p(t<u)\mathbb{E}\Bigg[\lim_{\eta\to 0^+} 2\Big((u-t)(\partial_t \tilde{f}^*) - \tilde{f}^* + (I + (u-t)(\partial_x \tilde{f}^*))\dot{x}_t\Big)^\top \\
&\quad \Big((u-t)\partial_t g_\eta - g_\eta + (u-t)(\partial_x g_\eta)\dot{x}_t\Big) \\
&\quad - 2\Big[\Big(I + (u-t)\partial_x \tilde{f}^*\Big)(\dot{x}_t - v)\Big]^\top (u-t)\partial_x g_\eta(\dot{x}_t - v) \,|\, t<u\Bigg]
\end{aligned}$$

There's no η in the first term in each of the two dot products that make up the expectand, so we can focus on the second term in the dot products, where $t < u$ For the second term in the first dot product:

$$\begin{aligned}
&\lim_{\eta\to 0^+} \Big((u-t)\partial_t g_\eta - g_\eta + (u-t)(\partial_x g_\eta)\dot{x}_t\Big) \\
&= \lim_{\eta\to 0^+} \Big((u-t)\partial_t g_\eta - I_\eta(t,u)\Big(\tilde{f}^*(t,t,x) - v(t,x)\Big) + (u-t)(I_\eta(t,u)\partial_x[\Big(\tilde{f}^*(t,t,x) - v(t,x)\Big)])\dot{x}_t\Big) \\
&= \lim_{\eta\to 0^+} (u-t)\partial_t g_\eta \\
&= \lim_{\eta\to 0^+} (u-t)\partial_t[I_\eta(t,u)\Big(\tilde{f}^*(t,t,x) - v(t,x)\Big)] \\
&= \lim_{\eta\to 0^+} (u-t)(\partial_t I_\eta)g + I_\eta \partial_t g
\end{aligned}$$

$$= \lim_{\eta \to 0^+} (u-t)(\partial_t I_\eta) I_\eta(t,u) \Big(\tilde{f}^*(t,t,x) - v(t,x) \Big)$$

$$= \Big(\tilde{f}^*(t,t,x) - v(t,x) \Big)(u-t) \lim_{\eta \to 0^+} (\partial_t I_\eta) I_\eta(t,u) = 0$$

The last equality holds because the function and its time derivative both go to zero.

This means the first product in the expecation goes to zero. By a similar argument the second term goes to zero as well.

Putting it all together

$$L := \lim_{\eta \to 0^+} s(\eta) = \lim_{\eta \to 0^+} \mathbb{E}[\phi_\eta] = \lim_{\eta \to 0^+} p(t=u)\mathbb{E}[\phi_\eta \,|\, t=u] + p(t<u)\mathbb{E}[\phi_\eta \,|\, t<u] > 0$$

Resultingly,

$$\exists \eta_0 > 0 \text{ s.t. } \forall \eta^* \text{ s.t. } 0 < \eta^* < \eta_0 \implies |s(\eta^*) - L| < \epsilon \tag{116}$$

If we pick $\epsilon = 0.5L$ then

$$|s(\eta^*) - L| < .5L \implies s(\eta^*) > L - .5L = .5L > 0 \tag{117}$$

so $s(\eta^*) > 0$. But this contradicts being at a stationary point. It cannot be that $\tilde{f}^*(t,t,\cdot) \neq v(t,\cdot)$ Therefore the velocity must match.

□

B.5 THEOREM 1

We present a proof about the functional stationary points of the GameFlow functional. We use the definitions of first variation and stationary point from section B.1 and the definition of functional stopgrad from section B.2.

Theorem. *Let $q(t,u)$ be a joint distribution with support over $t \le u$ and with positive probability on $t = u$. Let the family $\tilde{\mathcal{F}}$ include functions $\tilde{f}$ that are continuously differentiable in all arguments. Let $x_t = \alpha_t x_0 + \sigma_t x_1$ and $\dot{x}_t = \alpha_t x_0 + \sigma_t x_1$. Evaluate f at $f(t,u,x) + (u-t)\tilde{f}(t,u,x)$. Take expectations over $q(t,u)$ and $q(x_0)q(x_1)$. Let sg stand for stop-gradient. Define:*

$$\mathcal{L}[\tilde{f}] := \mathbb{E}\left[\|(\partial_t f)|_{(t,u,x_t)} + (\partial_x f)|_{(t,u,x_t)}\dot{x}_t\|^2 - \|(\partial_x f)_{(t,u,x_t)}(\dot{x}_t - \mathbb{E}[\dot{x}_t|x_t])\|^2\right]$$

$$\mathcal{L}^{sg}[\tilde{f}] := \mathbb{E}\left[\|(\partial_t f)_{(t,u,x_t)} + (\partial_x f)_{(t,u,x_t)}\dot{x}_t\|^2 - \|(\partial_x f)_{(t,u,x_t)}(\dot{x}_t + sg[(\partial_t f)]_{(t,t,x_t)})\|^2\right]$$

Then $\tilde{f}^*$ is a stationary point of $\mathcal{L}^{sg}$ w.r.t. $\tilde{\mathcal{F}}$ if and only if $\tilde{f}^*$ is a stationary point of $\mathcal{L}$ w.r.t. $\tilde{\mathcal{F}}$.

Proof. **Case 1: If $\tilde{f}^*$ is a stationary point of $\mathcal{L}$, then $\tilde{f}^*$ is a stationary point of $\mathcal{L}^{\text{sg}}$.**

- Since $\tilde{f}^*$ is a stationary point, $\delta\mathcal{L}[\tilde{f}^*;\cdot] = 0$
- by section B.4, we have that $\partial_t f^*(t,t,x) = -\tilde{f}^*(t,t,x) = -\mathbb{E}[\dot{x}_t|x_t]$ where $f^* = x + (u-t)\tilde{f}^*$
- $\mathcal{L}^{\text{sg}} = \mathcal{L}$
- Since $\mathcal{L}^{\text{sg}} = \mathcal{L}$, then $\delta\mathcal{L}^{\text{sg}}[\tilde{f}^*;\cdot] = \delta\mathcal{L}[\tilde{f}^*;\cdot] = 0$

Case 2: If f^* is not a stationary point of $\mathcal{L}$, then f^* is not a stationary point of $\mathcal{L}^{\text{sg}}$.

Since f^* is not a stationary point of $\mathcal{L}$, then $\exists h$ that is admissible (continuous) such that $\delta\mathcal{L}[f^*;h] \neq 0$. Then,

$$\underbrace{\delta\mathcal{L}[f^*;h]}_{\text{LHS}} = \underbrace{\mathbb{E}\Big[1[t=u]\dots\Big]}_{\text{RHS-L}} + \underbrace{\mathbb{E}\Big[1[t\neq u]\dots\Big]}_{\text{RHS-R}} \tag{118}$$

If the LHS is nonzero, then one of RHS-L or RHS-R is nonzero. Consider both cases.

case 2a:The RHS-R is nonzero and RHS-L is zero. RHS-L being zero means that the velocity matches, which means that RHS-R has the same first variation between $\mathcal{L}$ and $\mathcal{L}^{sg}$. So they must coincide regarding stationary points.

case 2b: RHS-L is nonzero and RHS-R is either zero or nonzero.

case2b-i $\delta\mathcal{L}^{sg}[f^*;h] \neq 0$ directly holds. This is all we are trying to ensure anyway, so we are done in this case.

case2b-ii Define the soft indicator, I_η:

$$I_\eta(t,u) = 1[\eta > 0]2\Big(1 - \frac{1}{1+\exp(-\frac{1}{\eta^2}(t-u)^2)}\Big) + 1[\eta = 0]1[t=u]. \tag{119}$$

Define the direction, $\hat{h}_\eta$:

$$\hat{h}_\eta(t,u,x) = I_\eta(t,u)h(t,u,x). \tag{120}$$

$\hat{h}_\eta$ is continuously differentiable for any $\eta > 0$. This is true cause it's a product of two functions that are each continuously differentiable (h is assumed continuously differentiable). Recall the mapping $s(\eta)$ from Section B.4, that maps η to $\delta\mathcal{L}[f;\hat{h}_\eta]$. Under the conditions of dominated convergence established in Section B.4, we know that $\exists\eta^* > 0$ such that $\hat{h}_{\eta^*}$ is a continuously differentiable function for which $\delta\mathcal{L}[f^*;\hat{h}_{\eta^*}] \neq 0$. This must mean that the velocity is not matched. But we know that if the velocity does not match, we are not at a stationary point of $\mathcal{L}^{sg}$ either, since $\mathcal{L}^{sg}$ and $\mathcal{L}$ coincide on penalizing velocity matching on $t = u$.

C Other useful derivations and results

C.1 Gradient updates are not optimization of one scalar objective via gradients

We show here that there exists a data distribution and a model such that the Gameflow updates are not the gradient of any single scalar objective. We illustrate this by considering an simple 1D setting. The key point is that, even in this restricted case, the update field induced by the stopgrad operator has non-zero curl and therefore cannot be written as the gradient of any scalar objective $J(\theta)$. The form to be differentiated is:

$$\begin{aligned} L_{\text{sg}}(\theta) = \mathbb{E}_{X_t}\Big[\big\|\partial_t f_\theta(t,u,X_t) + (\partial_x f_\theta(t,u,X_t))\,\dot{X}_t\big\|^2\Big] \\ - \mathbb{E}_{X_t}\Big[\big\|(\partial_x f_\theta(t,u,X_t))\big(\dot{X}_t - \text{stopgrad}[\tilde{f}_\theta(t,t,X_t)]\big)\big\|^2\Big]. \end{aligned} \tag{121}$$

for $f_\theta(t,u,x) = x + (u-t)\tilde{f}_\theta(t,u,x)$. Let us work in 1D and fix values of $X_t = x$ and $\dot{X}_t = d$, a constant. To accomplish this, we can choose X_1 freely and set $X_0 = X_1 - d$, so that the interpolation satisfies both $X_t = x$ and $\dot{X}_t = d$. In this case the expectations in (121) collapse to evaluation at this point (equivalently, think of us approximating with 1 Monte Carlo sample). Now, consider the parameters $\theta = (\theta_1, \theta_2)^\top$ and a single time pair (t,u) such that at the position $X_t = x$,

$$\partial_t f_\theta(t,u,x) = 0, \qquad \partial_x f_\theta(t,u,x) = \theta_1, \qquad \tilde{f}_\theta(t,t,x) = \theta_2. \tag{122}$$

(For a sufficiently expressive model, such local values can be realized; we only need existence of such a configuration.). Plugging these into (121) and dropping the expectation (single point), the Gameflow functional is:

$$L_{\text{sg}} = (\theta_1 d)^2 - \big(\theta_1\big(d - \text{stopgrad}[\theta_2]\big)\big)^2. \tag{123}$$

Let $\tilde{\nabla}$ denote differentiation with the stopgrad applied to θ_2 in the second term. Differentiating (123) with respect to θ_1 yields

$$\tilde{\nabla}_{\theta_1} L_{\text{sg}} = 2\theta_1 d^2 - 2\theta_1\big(d - \theta_2\big)^2 \tag{124}$$

$$= 2\theta_1\Big(d^2 - (d-\theta_2)^2\Big) \tag{125}$$

$$= 2\theta_1\Big(d^2 - (d^2 - 2d\theta_2 + \theta_2^2)\Big) \tag{126}$$

$$= 2\theta_1\big(2d\theta_2 - \theta_2^2\big) \tag{127}$$

$$= 2\theta_1\theta_2(2d - \theta_2). \tag{128}$$

Here the stopgrad on θ_2 only affects the second term and does not change the first term. For θ_2, all occurrences appear inside a stopgrad and the first term does not depend on θ_2, hence

$$\tilde{\nabla}_{\theta_2} L_{\text{sg}} = 0. \tag{129}$$

Thus the update field induced by CurlFlow in this simple example is

$$g(\theta_1,\theta_2) := \big(g_1(\theta_1,\theta_2), g_2(\theta_1,\theta_2)\big) = \big(2\theta_1\theta_2(2d-\theta_2),\ 0\big). \tag{130}$$

If this update were the gradient of some scalar objective $J(\theta_1,\theta_2)$, then the mixed partial derivatives would commute:

$$g_1 = \partial_{\theta_1} J, \qquad g_2 = \partial_{\theta_2} J \quad \Rightarrow \quad \partial_{\theta_2} g_1 = \partial_{\theta_1} g_2. \tag{131}$$

However,

$$\frac{\partial g_1}{\partial \theta_2} = \frac{\partial}{\partial \theta_2}\Big(2\theta_1\theta_2(2d-\theta_2)\Big) = 2\theta_1\big(2d - \theta_2 - \theta_2\big) = 4\theta_1(d-\theta_2), \tag{132}$$

$$\frac{\partial g_2}{\partial \theta_1} = 0, \tag{133}$$

and hence the curl of the update field is

$$\frac{\partial g_1}{\partial \theta_2} - \frac{\partial g_2}{\partial \theta_1} = 4\theta_1(d-\theta_2), \tag{134}$$

which is non-zero for generic θ (for example, whenever $\theta_1 \neq 0$ and $\theta_2 \neq d$). Therefore g is a smooth vector field with non-zero curl and *cannot* be written as the gradient of any scalar objective $J(\theta_1, \theta_2)$ in general. Put differently, a constraint on the relationship between the parameters of the model and the value of the chosen datapoint is necessary to ensure zero curl.

This 1D example is a specific instantiation of Gameflow (121) with a simple model and a single training point. It shows that, once we introduce the stopgrad on $\tilde{f}_\theta(t, t, x)$, the resulting optimization dynamics are in general *non-conservative*. The stopgrad structure breaks the symmetry required for the updates to be the gradient of a single scalar function. In this sense, CurlFlow is formally a (two-player) *game* rather than standard gradient descent on one potential function. Or one prefers, a nongradient vector flow.

□

C.2 Differentiating w.r.t. t versus u

Picking one of t or u just switches the sign of some terms in the loss, which ultimately only affects whether the learned map goes upward in time or downwards. The PDE can be obtained by differentiating the flow map $f(t, u, x)$ with respect to either endpoint. If X_s solves $\dot{X}_s = v(s, X_s)$, then keeping u fixed and differentiating in t gives a condition that evaluates v at t and x:

$$\partial_t f(t, u, x) + (\partial_x f(t, u, x))\, v(t, x) = 0, \tag{135}$$

while keeping t fixed and differentiating in u gives a condition that evaluates v at u and $x_u = f(t, u, x)$:

$$\partial_u f(t, u, x) + (\partial_x f(t, u, x))\, v\big(u, f(t, u, x)\big) = 0. \tag{136}$$

Along the ODE trajectory $dX = v\, ds$ one has the identity

$$\partial_t f(t, u, x) = -\,\partial_u f(t, u, x),$$

so choosing to differentiate in t versus u only changes the sign convention and which endpoint is held fixed.