Noise, Scores, and Straight Lines:
Diffusion Models and Flow Matching, Slowly

Matthew Willetts
with assistance from Codex and Claude

v0.1 — June 2026

Abstract

Diffusion models and flow matching look, on first contact, like two different religions: one worships stochastic differential equations, denoising, and a mysterious object called the score; the other speaks of velocity fields, optimal transport, and straight paths. These notes develop both from scratch and show that they are one subject. The unifying objects are (i) a *path of distributions* interpolating between data and noise, chosen by us and fully under our control, and (ii) a single, almost embarrassingly simple lemma — the L^2 projection property of conditional expectation — which is used twice, once to make score learning tractable (denoising score matching) and once to make velocity learning tractable (conditional flow matching). Everything else — reverse SDEs, probability flow ODEs, DDPM's ELBO, classifier-free guidance, rectified flow — is bookkeeping on top of those two ideas, and we do the bookkeeping slowly, with derivations, pictures, code, recaps, and problems. Nothing about flows, stochastic calculus, or numerical integration is assumed: a transport toolkit (ODE flows, the continuity equation, change of variables) and an SDE toolkit (Brownian motion, Itô's lemma, linear SDEs, Fokker–Planck, Langevin dynamics) are built from a standing start in Sections 3–4, and a solver's-eye refresher on ODE methods — order, curvature, and why DDIM is Euler in disguise — occupies Section 8. Prior exposure to VAEs is occasionally exploited and never required.

Contents

1 The problem, and the move that unlocks it **2**
1.1 One notation for both religions . . . 2

2 The score: the one function worth knowing **4**
2.1 Tweedie's formula: scores are denoisers . . . 4
2.2 The lemma that powers everything . . . 4

3 Interlude: the transport toolkit **8**
3.1 Flows: existence, uniqueness, and the two gifts . . . 8
3.2 The continuity equation: probability is a conserved fluid . . . 8
3.3 How density changes along your own trajectory . . . 9

4 Interlude: the SDE toolkit, from a standing start **9**
4.1 Brownian motion: $\sqrt{\mathrm{d}t}$ is the whole personality . . . 10
4.2 What an SDE actually says . . . 10
4.3 Itô's lemma is Taylor's theorem with one extra term . . . 11
4.4 Linear SDEs, solved completely — and the forward SDE derived . . . 11
4.5 From paths to densities: the Fokker–Planck equation . . . 12
4.6 Langevin dynamics: the SDE that stands still . . . 12

5 Sampling I: the probability flow ODE **13**
5.1 Sampling II: putting the noise back (and why you might) . . . 14

6 Interlude: DDPM in discrete time — the chain, the ELBO, the VAE **14**
6.1 The discrete chain, and where $\bar{\alpha}$ comes from . . . 14
6.2 The ELBO: a VAE with an incorruptible encoder . . . 15

7 Flow matching: regress on the velocity directly **16**
7.1 Rectified flow: the case for straight lines 17
7.2 The optimal transport vista 17

8 Sampling III: a solver's-eye refresher **19**
8.1 One-step methods, order, and what curvature has to do with it 19
8.2 Exploit the structure: DDIM is Euler in disguise 19
8.3 Likelihoods along the flow, while we are here 20
8.4 The stochastic option, costed 21

9 Steering: guidance as score arithmetic **21**

10 The whole subject on one page **22**

1 The problem, and the move that unlocks it

We want a generative model: a recipe that turns samples of something easy (Gaussian noise) into samples of something hard (images, molecules, audio — call the data distribution p_{data}). Abstractly we seek a map T with $T_{\#}\mathcal{N}(0, I) = p_{\text{data}}$: push noise through T, get data.[1]

You know the classical attempts and their pathologies. Fit T directly as a neural network and train adversarially: GANs, with their instabilities and dropped modes. Make T invertible so you can write the likelihood exactly: normalizing flows, paying for invertibility with restricted architectures. Introduce latent variables and bound the likelihood: VAEs, where the ELBO is beautiful but the learned posterior fights the learned prior and samples come out oversmoothed.[2]

The move that unlocks the modern subject is one of those reframings that sounds too small to matter:

Don't learn the map. Learn the infinitesimal *map — the velocity field of a continuous transformation — and integrate it.*

Why does this help? Because a map taking a Gaussian blob to the manifold of images is a monstrous, discontinuous-looking object, but a *flow* that morphs one into the other over a time interval can be gentle at every instant; and the regression target for "how should points move right now" will turn out to be available in closed form along paths *we design*. The price is that sampling means numerically integrating a differential equation — the network is called many times rather than once. The whole modern literature (DDPM, score SDEs, DDIM, flow matching, rectified flow, consistency distillation) is a sequence of answers to "which path of distributions?," "what regression target?," and "how few integration steps can we get away with?"

1.1 One notation for both religions

Fix it now, once, because the two communities' conventions collide head-on.[3] Let $x_0 \sim p_{\text{data}}$ and $\varepsilon \sim \mathcal{N}(0, I)$, independent, and define the **interpolant**

$$x_t \;=\; \alpha_t\, x_0 \;+\; \sigma_t\, \varepsilon, \qquad t \in [0, 1], \tag{1}$$

where the **schedule** (α_t, σ_t) is smooth, with $\alpha_0 = 1, \sigma_0 = 0$ (pure data) and $\alpha_1 \approx 0, \sigma_1 \approx 1$ (pure noise). Write q_t for the marginal distribution of x_t; since $x_t \mid x_0 \sim \mathcal{N}(\alpha_t x_0, \sigma_t^2 I)$, it has the explicit mixture density

$$q_t(x) \;=\; \int q(x \mid x_0)\, p_{\text{data}}(x_0)\, \mathrm{d}x_0, \qquad q(x \mid x_0) = \mathcal{N}\big(x;\ \alpha_t x_0,\ \sigma_t^2 I\big) \tag{2}$$

— the data distribution shrunk by α_t and blurred by a Gaussian of width σ_t. Both the mixture form and the kernel notation $q(x \mid x_0)$ recur constantly.[4]

[1] Pushforward notation, used exactly twice in these notes: $T_{\#}\mu$ is the distribution of $T(z)$ when $z \sim \mu$ — sample, then map. The displayed equation is just "T(noise) has the data's distribution."

[2] If you spent your PhD on VAEs, hold that experience nearby; in Section 6 diffusion models will turn out to *be* a VAE — one whose encoder is fixed, analytic, and incapable of collapsing, which is a useful lens on why they work where vanilla VAEs strain.

[3] Diffusion papers put data at $t = 0$ and noise at $t = 1$ (or $t = T$); flow matching papers put *noise* at $t = 0$ and data at $t = 1$. Both are fine; mixing them silently is the single most common source of sign errors in this subject. These notes use the diffusion convention throughout: **data lives at $t = 0$, noise at $t = 1$**, and generation runs time *backwards*. To convert any flow-matching paper, send $t \mapsto 1 - t$ and flip the sign of velocities.

[4] Is p_{data} the population law or the empirical one — a set of deltas? The notation suggests a density, but nothing in these notes requires one: read every $\int \cdot\, p_{\text{data}}(x_0)\, \mathrm{d}x_0$ as an expectation over x_0, valid verbatim for the empirical measure $\frac{1}{n}\sum_i \delta_{x^{(i)}}$, since the Gaussian kernel makes q_t smooth for every $t > 0$ regardless. Conceptually we mean the population law; operationally, every training expectation is over the empirical one. The gap between those two readings is important, but easier to discuss after the regression objective exists; Section 2 returns to it under "what training buys."

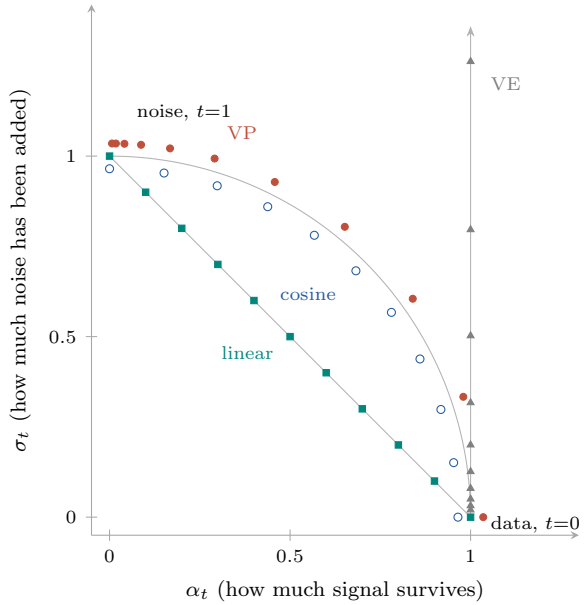

Figure 1: Every schedule is a curve from data $(1, 0)$ toward noise in the (α_t, σ_t) plane; markers are placed at $t = 0, 0.1, \ldots, 1$, so their spacing shows the *clock*. Cosine (open circles, drawn just inside the circle for visibility) moves along the unit circle at uniform speed. VP with the standard linear $\beta \in [0.1, 20]$ (filled circles, just outside) traces the *same* circle but sprints: by $t = 0.5$ it has already fallen to $\alpha \approx 0.28$, and by $t = 0.7$ to $\alpha \approx 0.08$ — nearly all of its time is spent loitering at the noise end, which is why so much DDPM engineering is schedule re-warping. The linear/rectified schedule (squares) cuts the diagonal at uniform speed. VE (triangles, geometric σ_t) never shrinks the signal and exits the unit square upward.

What every $\mathbb{E}$ in these notes means. Stated once, here, because the literature never states it and readers are entitled to wonder. There is exactly one source of randomness throughout: the triple (x_0, ε, t) with $x_0 \sim p_{\text{data}}$, $\varepsilon \sim \mathcal{N}(0, I)$, $t \sim \text{Uniform}[0, 1]$, mutually independent; everything else (x_t, training targets, trajectories) is a deterministic function of these. A bare $\mathbb{E}$ integrates over this one joint law — always, no exceptions. A subscript, as in $\mathbb{E}_{t,x_0,\varepsilon}$, is a *reminder* of which coordinates the integrand involves, not a different operator. A conditioning bar, $\mathbb{E}[\,\cdot \mid x_t = x\,]$, swaps the joint for its conditional given the stated value — and because x_t sits downstream of (x_0, ε), that conditional is automatically a *posterior*, computed by Bayes from (2): Tweedie's $\mathbb{E}[x_0 \mid x_t]$ below is the posterior mean of the clean signal, by these rules and nothing else. Where the conditional law is not obvious at a glance we spell it out on the subscript, as in $\mathbb{E}_{x_0 \sim p(x_0 \mid x_t = x)}$. (Reweighting t away from uniform is a design choice, flagged where it occurs. And the fail-safe is always available: rewrite any $\mathbb{E}$ as an integral against the density just named — the physicist's move, never wrong.)

The schedule is a *design choice*; the main characters:

Name	α_t	σ_t	Personality
Variance-preserving (DDPM)	$e^{-\frac{1}{2}\int_0^t \beta}$	$\sqrt{1-\alpha_t^2}$	$\mathbb{E}\|x_t\|^2$ constant; curved
Variance-exploding (SMLD)	1	growing, large	data never shrinks; noise engulfs it
Linear / rectified flow	$1 - t$	t	straight conditional paths
Cosine-ish	$\cos\frac{\pi t}{2}$	$\sin\frac{\pi t}{2}$	gentle at both ends; $\alpha^2{+}\sigma^2{=}1$

Everything in these notes is derived for general (α_t, σ_t), so each named model appears as a row of a table rather than a separate theory. (One row needs an asterisk: VE breaks the $\alpha_1 \approx 0$ normalization — nothing ever shrinks the data; instead σ_1 is made so much larger than the data scale that $q_1 \approx \mathcal{N}(0, \sigma_1^2 I)$, a Gaussian you can sample after rescaling.)

Geometrically, a schedule is a curve in the (α, σ) plane from $(1, 0)$ to the noise end, traversed at some speed — Figure 1 draws the four rows. Two lessons sit in that picture. First, VP and cosine trace the *same* curve (the unit circle, since both satisfy $\alpha^2 + \sigma^2 = 1$) with different clocks: the schedule is really a curve *plus a time-parameterization*, and the parameterization controls where training effort and integration steps concentrate. Second, "which schedule" is a smaller decision than the literature's notation wars suggest: all of them are gentle monotone curves between the same two endpoints.

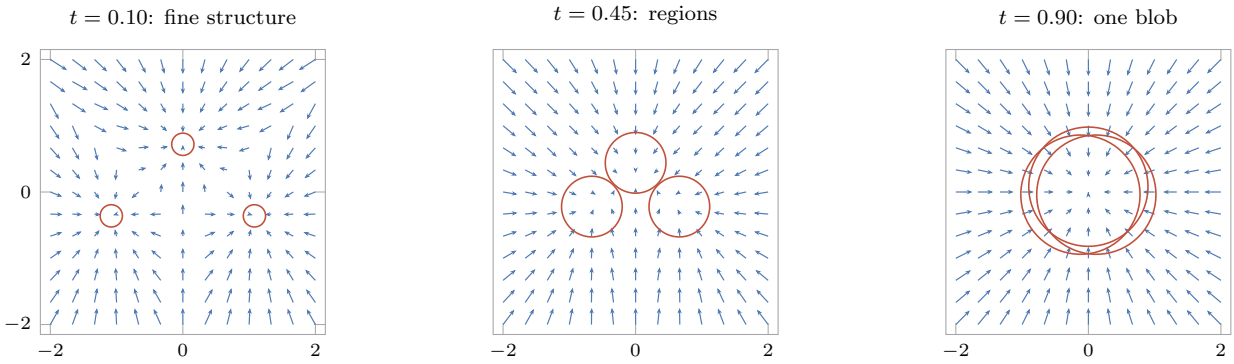

Figure 2: The score family $\{s(\cdot, t)\}_t$ as a multiscale description of the data, computed in closed form for a three-component Gaussian mixture under the linear schedule. Arrows show the *Tweedie displacement* $\sigma_t^2\, s(x,t) = \alpha_t \mathbb{E}[x_0 \mid x_t = x] - x$ of (4) — the step that carries x to the (rescaled) posterior mean of clean data — with lengths square-root compressed for legibility; red circles are the one-standard-deviation level sets of the mixture components of q_t (centers shrink by α_t, widths grow to $\sqrt{\alpha_t^2 s^2 + \sigma_t^2}$). At $t = 0.9$ the field has forgotten everything but "head for the middle": $s \approx -x$. At $t = 0.45$ it resolves regions; at $t = 0.10$ it encodes the exact location of each component. Following these fields from right panel to left is generation.

2 The score: the one function worth knowing

Diffusion models orbit a single object: the **score** of the noisy marginals,

$$s(x,t) \;=\; \nabla_x \log q_t(x). \tag{3}$$

Before any SDEs, build intuition for what this vector field *is*. At each noise level, $s(\cdot, t)$ points from where you are toward where the probability mass is — "uphill on the log-density." For t near 1, q_t is nearly Gaussian and the score is nearly $-x$: everything points blandly at the origin. For t near 0, q_t hugs the data manifold and the score points sharply toward it, encoding fine detail. The family $\{s(\cdot, t)\}_t$ is a multiscale description of the data: coarse structure at high noise, texture at low noise. Generation, we will see, is the act of following these vector fields from coarse to fine. Figure 2 shows the whole family for a toy mixture; the rest of these notes is machinery for learning and integrating these fields.

Two reasons the score, rather than the density, is the right object to learn. First, it is *normalization-free*: $\nabla \log q = \nabla \log(Zq)$ for any constant Z, so the eternal partition-function problem of energy-based models evaporates. Second, it satisfies an identity relating it to *denoising*, on which everything that follows runs.

2.1 Tweedie's formula: scores are denoisers

Condition on a noisy observation x_t and ask for the posterior mean of the clean signal. The answer:

$$\boxed{\mathbb{E}[x_0 \mid x_t] \;=\; \frac{x_t + \sigma_t^2\, s(x_t, t)}{\alpha_t}} \tag{4}$$

— the optimal denoiser *is* the score, rescaled and shifted.[5] Pause on what (4) is saying: "move from x_t a distance σ_t^2 along the score, then undo the shrinkage α_t, and you land on the conditional average of all clean signals that could have produced your observation." Uphill on the log-density *is* denoising. This is why everything in this subject can be phrased either in score language or in denoiser language, and why the networks are called "denoisers" even in papers that never mention denoising.

Exercise 1. *Prove* (4). *Start from the mixture form* (2) *of* q_t, *differentiate in* x *under the integral (the Gaussian kernel gives* $\nabla_x q(x \mid x_0) = q(x \mid x_0)\, \frac{\alpha_t x_0 - x}{\sigma_t^2}$*), divide by* $q_t(x)$, *and recognize a posterior expectation. Three lines.*

2.2 The lemma that powers everything

We would like to learn the score by regression, and at first sight there is nothing to regress on: no dataset has $\nabla \log q_t$ as its response variable. The score is a functional of the very density being modelled, and

[5] Attributed by Robbins (1956) to the actuary Maurice Tweedie; rediscovered roughly once per decade since, most recently by all of us.

learning objects of that kind used to be expensive — trace-of-Jacobian objectives (Hyvärinen's original score matching), MCMC against partition functions (energy-based models), differential equation solves inside the training loop (likelihood-trained CNFs). The escape is to *manufacture* the dataset: corrupt the data yourself, and choose the per-sample prediction target so that its conditional average is provably the score. That takes two ingredients. One is a guarantee that regressing on noisy per-sample targets recovers their conditional mean — the lemma below, which is ordinary. The other is an identity exhibiting the score as exactly such a conditional mean — equation (6) below, which is the actual discovery, and one the field took years to make, twice over.[6] We state the lemma once, in full generality, because the same two ingredients recur for velocities in Section 7.

Lemma 1 (Conditional targets suffice)**.** *Let (A, B) be jointly distributed and suppose we want $f^*(b) = \mathbb{E}[A \mid B = b]$, the L^2-optimal predictor of A from B. Then the regression problem*

$$\min_f \ \mathbb{E}_{(A,B)} \big\| f(B) - A \big\|^2 \tag{5}$$

has minimizer f^* over all functions (over a restricted class, the class's best L^2 approximation to f^*) — even though no training pair ever shows f the target $f^*(b)$ itself, only samples A that average to it.

Proof. Pythagoras in L^2. Add and subtract $f^*(B)$ inside the square:

$$\mathbb{E}\big\| f(B) - A \big\|^2 = \mathbb{E}\big\| f(B) - f^*(B) \big\|^2 + \mathbb{E}\big\| f^*(B) - A \big\|^2 + 2\,\mathbb{E}\big\langle f(B) - f^*(B),\ f^*(B) - A \big\rangle.$$

Condition the cross term on B (the tower property): given B, the first factor is a constant and the second averages to $f^*(B) - \mathbb{E}[A \mid B] = 0$. What remains is a constant plus a nonnegative term that vanishes iff $f = f^*$. □

The proof is worth a moment's recognition: it is the projection property of conditional expectation, the same L^2 Pythagoras behind every orthogonal decomposition in statistics — mean-squared error = approximation error + irreducible noise (this instance); bias^2 + variance (same trick, conditioning on the training set instead of B); the law of total variance; Rao–Blackwell. As that list suggests, the lemma is the daily business of regression, not a discovery: nobody running least squares expects to observe the true regression function, only responses that average to it. It does carry one corollary worth pocketing. At the optimum the loss does *not* vanish — its floor is the irreducible term $\mathbb{E}\,\mathrm{Var}(A \mid B)$, the spread of the per-sample targets about their conditional mean (the vertical width of the scatter in Figure 3). For noise prediction the floor is essentially zero at the noise end, where x_t all but determines ε, and the full d at the data end, where given $x_t \approx x_0$ the noise is anybody's guess — which is why raw loss values are a famously poor progress signal for diffusion training, and per-t loss curves must be read against the floor rather than against zero.

So the lemma supplies the regression, and everything turns on the other ingredient: *an identity writing the quantity you want as a conditional expectation of something computable per sample.* The art is in the target. The two famous instances:

Instance 1: denoising score matching. The score is a conditional expectation of the *conditional* score:

$$\begin{aligned}
\nabla \log q_t(x) = \frac{\nabla q_t(x)}{q_t(x)} &= \frac{\int \nabla_x q(x \mid x_0)\, p_{\text{data}}(x_0)\, \mathrm{d}x_0}{q_t(x)} \\
&= \int \nabla_x \log q(x \mid x_0)\ \underbrace{\frac{q(x \mid x_0)\, p_{\text{data}}(x_0)}{q_t(x)}}_{= p(x_0 \mid x_t = x) \text{ by Bayes}}\ \mathrm{d}x_0 \\
&= \mathbb{E}_{x_0 \sim p(x_0 \mid x_t = x)} \big[\, \nabla_x \log q(x \mid x_0) \,\big],
\end{aligned} \tag{6}$$

differentiating the mixture form (2) under the integral in the first line, then making two moves in the second: the log-derivative trick $\nabla q = q\, \nabla \log q$ manufactures the logarithm and frees a factor $q(x \mid x_0)$,[7] and that

[6] Denoising score matching arrived six years after the trace form; flow matching, four years after continuous normalizing flows. The lemma sat in the textbooks throughout. The hard step was never the regression; it was recognizing the score as a conditional mean at all.

[7] Readers from reinforcement learning will recognize the move: this is the log-derivative trick of policy gradients, applied in the x-slot rather than the θ-slot — and "score" is itself borrowed vocabulary: Fisher's score is $\nabla_\theta \log p_\theta$, and for a location family $q(x - \mu)$ the parameter score and the data-space score coincide up to sign. The companion identity $\mathbb{E}[\nabla \log p] = 0$ (differentiate $\int p = 1$) holds here too: $\mathbb{E}_{x \sim q_t}[s(x,t)] = \int \nabla q_t = 0$, which is what reconciles Tweedie with unconditional means

factor joins $p_{\text{data}}(x_0)/q_t(x)$ to form exactly the posterior — $q_t(x)$ does not disappear, it is spent as the Bayes normalizer, (2) again. The final line writes the measure where it belongs: the average is over x_0 drawn from the *posterior* given the noisy observation, not over p_{data}, and the gradient is in the observation slot, evaluated at the fixed point x. The literature's shorthand for this object is $\mathbb{E}[\nabla_x \log q(x_t \mid x_0) \mid x_t = x]$ — defensible when you already know what it means, treacherous when the integrand mentions the conditioning variable; we use the bare-conditioning form only where the integrand is unambiguous, as in Tweedie (4). And the conditional score is closed-form, because the corruption is Gaussian:

$$\nabla_{x_t} \log q(x_t \mid x_0) = -\frac{x_t - \alpha_t x_0}{\sigma_t^2} = -\frac{\varepsilon}{\sigma_t}. \tag{7}$$

So by Lemma 1, the humble objective

$$\boxed{\mathcal{L}_{\text{DSM}} = \mathbb{E}_{t,\, x_0,\, \varepsilon}\left\| s_\theta(x_t, t) + \frac{\varepsilon}{\sigma_t} \right\|^2} \tag{8}$$

is minimized exactly at $s_\theta = \nabla \log q_t$. Equivalently, reparameterizing $s_\theta = -\varepsilon_\theta/\sigma_t$: *train a network to predict the noise you added* (up to a $1/\sigma_t^2$ reweighting across t; the practical loss below drops it, which changes the t-weighting but not the per-t minimizer). That is the entire DDPM training loop: sample a data point, sample t, sample ε, form x_t, predict ε, take an MSE step. Since "entire" is a strong word, here it is with nothing left to hide (PyTorch; `alpha` and `sigma` implement the schedule). Mixed-precision, EMA, and logging plumbing are suppressed; the point is the objective, not PyTorch boilerplate:

```
def train_eps_model(model, opt, x0):       # x0: [B, ...] data batch
    t   = torch.rand(x0.shape[0], device=x0.device)
    eps = torch.randn_like(x0)
    # reshape [B] schedules to [B, 1, ...] for non-batch broadcasting
    shape = (-1,) + (1,) * (x0.ndim - 1)
    a, s = alpha(t).view(shape), sigma(t).view(shape)
    x_t = a * x0 + s * eps

    loss = (model(x_t, t) - eps).square().mean()
    loss.backward(); opt.step(); opt.zero_grad()
```

For CFM with the linear schedule, the same code trains a velocity model by replacing the target `eps` with `eps - x0`. Everything else in these notes is about what to do with the trained `model`; nothing above changes. The mystery of why predicting noise builds a generative model is dissolved by (6): the noise, on average given x_t, *is* the (negative, scaled) score. Figure 3 makes the lemma concrete: the training set is a cloud of per-sample targets that never lies on the function being learned, and L^2 regression threads the conditional mean through it anyway.

What training buys. One calibration before moving on. The opening claim — nothing to regress on, q_t unknown — deserves its asterisk: for the dataset actually in hand, q_t is not unknown at all. With p_{data} the empirical measure, the mixture (2) is a finite Gaussian mixture whose score is closed-form — softmax responsibilities over the training points times the per-component scores (7) — at the price of n kernel evaluations per query and no learning anywhere. What the trained network buys is therefore not access but two other things. *Amortization*: an $O(n)$-per-query sweep over the dataset becomes a single call of a fixed network — the training set compressed into a function. And *the right estimand*: the lemma certifies that the population minimizer of the regression is the population score, so the same procedure run on samples is an ordinary, consistent estimator of the object we actually want, the finite-n gap being standard generalization rather than anything diffusion-specific. The limit case ties the two together. A network that achieved the *empirical* optimum exactly would be an expensive implementation of the closed-form score of (training set $*$ Gaussian), and sampling it would return, approximately, training points. Everything generative about generalization lives in the daylight between the empirical optimum and the learned function: architecture, optimization, finite data, weighting over t, and early stopping all act as implicit regularizers. The field leans on that fact more than its clean population equations admit.

($\mathbb{E}[x_t] = \alpha_t \mathbb{E}[x_0]$); its conditional version is why baselines work in REINFORCE, and its regression cousin is the freedom to add anything of zero conditional mean to the target in Lemma 1 — the minimizer stays, only the loss floor moves. The roles differ instructively: policy gradients use the trick at every update to build an unbiased gradient *estimator*, then fight its variance; DSM uses it once, offline, to build an exact *identity* for the estimand, and trains by reparameterization ($x_t = \alpha_t x_0 + \sigma_t \varepsilon$) with plain MSE — one reason diffusion training is so much more placid than policy optimization.

Figure 3: Lemma 1 at work. Data is the bimodal mixture $\frac{1}{2}\mathcal{N}(-1, 0.25^2) + \frac{1}{2}\mathcal{N}(+1, 0.25^2)$; each gray dot is one training example (x_t, ε) at fixed $t = 0.6$ under the linear schedule ($x_t = 0.4\, x_0 + 0.6\, \varepsilon$). The dots form two slanted bands — one per mode, because the same x_t can arise as "left mode plus rightward noise" or "right mode plus leftward noise" — and *no individual target is correct*. The L^2 minimizer is the closed-form conditional mean (blue), which interpolates the bands according to their posterior weights; by (6) this curve is $-\sigma_t s(x, t)$. The network only ever sees the dots.

Naming the parameterizations. One quantity, four wardrobes: an epsilon predictor $\hat{\varepsilon}$, a score predictor s, a clean-data predictor $\hat{x}_0$, and a velocity predictor $\hat{v}$. They are interconvertible by (4)–(7); fluency here is what reading this literature mostly consists of:

$$\hat{\varepsilon} = -\sigma_t s, \qquad s = -\frac{\hat{\varepsilon}}{\sigma_t}, \qquad \hat{x}_0 = \frac{x_t - \sigma_t \hat{\varepsilon}}{\alpha_t}, \qquad \hat{v} = \dot{\alpha}_t\, \hat{x}_0 + \dot{\sigma}_t\, \hat{\varepsilon}, \tag{9}$$

where $\hat{v}$, the *velocity*, will take center stage in Section 7. (Numerically the choices differ: ε-prediction is well-conditioned near the data end where $\sigma_t \to 0$ makes the score blow up; x_0-prediction is better near the noise end; v-type parameterizations interpolate. Same function, different conditioning.)[8]

Gaussian sandbox. One closed-form case is worth carrying as a unit test for the notation. Let $p_{\text{data}} = \mathcal{N}(\mu, s^2 I)$ and set $\tau_t^2 = \alpha_t^2 s^2 + \sigma_t^2$. Then

$$q_t = \mathcal{N}(\alpha_t \mu, \tau_t^2 I), \qquad s(x, t) = -\frac{x - \alpha_t \mu}{\tau_t^2}, \tag{10}$$

and Tweedie gives the posterior mean

$$\mathbb{E}[x_0 \mid x_t = x] = \mu + \frac{\alpha_t s^2}{\tau_t^2}\,(x - \alpha_t \mu), \qquad \mathbb{E}[\varepsilon \mid x_t = x] = \frac{\sigma_t}{\tau_t^2}\,(x - \alpha_t \mu). \tag{11}$$

So the $\hat{v}$ wardrobe of (9), assembled from these two posterior means, is just an affine field,

$$v_t(x) = \dot{\alpha}_t \mu + \frac{\dot{\alpha}_t \alpha_t s^2 + \dot{\sigma}_t \sigma_t}{\tau_t^2}\,(x - \alpha_t \mu) \tag{12}$$

(when Section 7 defines the *marginal velocity*, this is what it will be for Gaussian data). This little sandbox is the fastest way to check signs: high-noise scores point back toward the mean, posterior means shrink linearly toward μ, and the velocity is affine because Gaussian families stay Gaussian under the interpolant.

> **The story so far.** *The score $\nabla \log q_t$ is a multiscale, normalization-free description of the data, and Tweedie's formula says it is secretly the optimal denoiser. We can't regress on it directly, but Lemma 1 plus the Gaussian corruption gives a per-sample proxy target — the added noise — whose regression has the true score as its exact minimizer. Training a diffusion model is MSE noise prediction; everything sophisticated happens at sampling time.*

[8] And the name is not a coincidence: for any schedule with $\alpha_t^2 + \sigma_t^2 = 1$, differentiating the constraint gives $(\dot{\alpha}_t, \dot{\sigma}_t) = \lambda_t\,(-\sigma_t, \alpha_t)$ for some $\lambda_t > 0$, so $\hat{v} = \lambda_t\,(\alpha_t \hat{\varepsilon} - \sigma_t \hat{x}_0)$ — up to the time-rescaling λ_t (a constant $\pi/2$ for the cosine schedule), the velocity here *is* the v of Salimans–Ho "v-prediction" from progressive distillation.

A reader's map for the middle of these notes. The spine of the argument is short: this section (a learnable score) → Section 5 (turn it into a sampler) → Section 7 (the same construction, velocity-first). Everything stacked in between and after — deterministic transport (Section 3), stochastic calculus (Section 4), discrete-time DDPM (Section 6), numerical solvers (Section 8) — is an interlude that exists to make one step of the spine honest, and each opens by saying which step it is holding up. If you already own one of those toolkits, skip it on a first pass and let its recap vouch for you; the spine does not branch. On a first read, a defensible fast path is: Section 2, the statement of the probability-flow ODE in Section 5, then Section 7; return to the interludes when a sign, factor of two, or solver claim needs proof.

3 Interlude: the transport toolkit

The move of Section 1 was "don't learn the map, learn the velocity field and integrate it." Two interludes now build everything that move needs. This one is deterministic: what it means to integrate a velocity field (flows, and the two gifts determinism hands us for free), what it means for a *density* to be carried by a velocity field (the continuity equation — the single most load-bearing prerequisite in these notes), and how density changes along the flow. The next interlude is stochastic. Nothing here is deep; all of it will turn out to be holding something up.

3.1 Flows: existence, uniqueness, and the two gifts

An ordinary differential equation $\dot{x} = v(x,t)$ with starting condition $x_{t_0} = x_0$ is an *initial value problem*, and the one theorem we need about it is Picard–Lindelöf, statement only: if v is Lipschitz in x and continuous in t, a solution exists and is *unique*. (Our velocity fields are neural networks — smooth by construction — so we qualify and will not fuss.) Solving from every starting point at once gives the **flow map** $\Phi_{s\to t}$, sending where a particle is at time s to where it will be at time t; flow maps compose, $\Phi_{u\to t} \circ \Phi_{s\to u} = \Phi_{s\to t}$, and $\Phi_{t\to s}$ inverts $\Phi_{s\to t}$. A flow is a smoothly deforming family of bijections of space: nothing tears, nothing overlaps.

Two consequences come free, and both will silently carry weight later.

Gift one: trajectories never cross. If two solutions agree at any single instant, run the initial value problem from the meeting point — in both time directions — and uniqueness forces them to agree always. Distinct particles stay distinct forever. In one dimension this makes the flow *monotone*: particle order is preserved. Innocuous-looking; it is why probability-flow trajectories can split at a ridge without ever touching (Figure 7), and it is the entire content of the non-crossing claim on which reflow rests (Figure 8(b)).

Gift two: reversal is free. If x_t solves $\dot{x} = v(x,t)$ on $[0,1]$, then $y_\tau = x_{1-\tau}$ solves $\dot{y} = -v(y, 1-\tau)$: to run a flow backwards, flip a sign and integrate. Hold this against the stochastic case: a Brownian increment does not "unhappen" when you negate a drift, and reversing an SDE will cost an actual theorem (Anderson's, Section 5) plus possession of the score. This contrast is half the reason the probability-flow ODE, when it appears, is such a gift: determinism buys time-reversibility for nothing, and the entire price of generation gets front-loaded into learning the velocity field.

3.2 The continuity equation: probability is a conserved fluid

Now release a whole population: draw $x_0 \sim q_0$ and let every particle integrate the same field v. Write q_t for the population's density. Which PDE does q_t obey? The answer is bookkeeping, and the bookkeeping is worth doing slowly because the resulting equation is the hinge of the entire subject.

Work in one dimension first. Probability is *conserved*: the mass in an interval $[a,b]$ changes only because particles cross its endpoints. Define the **flux** $J(x,t)$ as the rate at which probability crosses the point x, rightward positive. For particles moving with velocity v, the ones that cross x in the next $\mathrm{d}t$ are exactly those within distance $v\,\mathrm{d}t$ upstream, a mass $q(x,t)\,v(x,t)\,\mathrm{d}t$: so

$$J = q\,v \qquad \text{(flux = density} \times \text{velocity)}. \tag{13}$$

Then, as Figure 4 draws,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_a^b q\,\mathrm{d}x \;=\; J(a,t) - J(b,t) \;=\; -\int_a^b \partial_x J\,\mathrm{d}x, \tag{14}$$

in through the left, out through the right. The interval was arbitrary, so the integrands must agree pointwise: $\partial_t q + \partial_x(qv) = 0$. In d dimensions the same argument over a small box gives the **continuity equation**

$$\boxed{\partial_t q_t + \nabla\cdot\big(q_t\,v\big) = 0,} \tag{15}$$

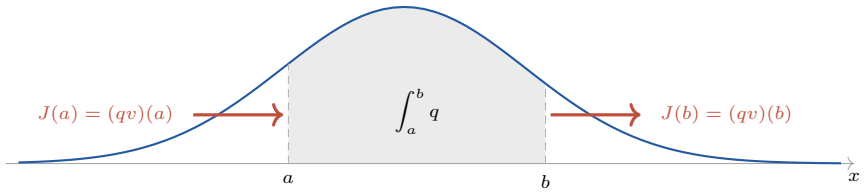

Figure 4: Probability is a conserved fluid: the mass in $[a, b]$ changes only by the flux $J = qv$ through the endpoints — in at a, out at b — and shrinking the interval turns that sentence into $\partial_t q + \partial_x(qv) = 0$. Every density-evolution equation in these notes (continuity, Fokker–Planck, the probability-flow rewriting) is this bookkeeping with a particular flux plugged in.

where the **divergence** $\nabla \cdot J = \sum_i \partial_i J_i$ of a flux field measures the net rate at which stuff flows *out* of an infinitesimal box, per unit volume; summing boxes gives the only integral fact we use, $\int_V \nabla \cdot J =$ (flux out through ∂V).

Two directions of travel through (15), and we use both. *Forward*: given the flow, the transported density satisfies continuity — that is the derivation above. *Converse*, the direction that does the real work later: (15) is the *complete signature* of transport. If a family of densities q_t and a velocity field v_t satisfy (15), then sampling $x_0 \sim q_0$ and integrating $\dot{x} = v_t(x)$ produces $x_t \sim q_t$ at every time — because (for fields as nice as ours) the equation propagates q_0 forward uniquely, and the transported density is one solution, hence the only one. Keep this converse loaded: the central manoeuvre of Section 5 will be to stare at a PDE until it looks like (15) for some velocity field, and then to declare victory — legitimately, because of exactly this paragraph.

3.3 How density changes along your own trajectory

One more deterministic fact, three lines long, with an entire likelihood theory hanging off it. Ride a particle: how does the density *at your own position* evolve? The $\frac{\mathrm{d}}{\mathrm{d}t}$ below is the *total* derivative along the moving particle — both arguments of $\log q_t(x_t)$ carry the t — which fluid mechanics calls the material derivative $\partial_t + v \cdot \nabla$; the Eulerian ∂_t at a fixed point is only its first term. Chain rule plus continuity:

$$\frac{\mathrm{d}}{\mathrm{d}t} \log q_t(x_t) = \partial_t \log q_t + \dot{x} \cdot \nabla \log q_t = \big(-\nabla \cdot v - v \cdot \nabla \log q_t \big) + v \cdot \nabla \log q_t, \tag{16}$$

expanding $\partial_t q = -\nabla \cdot (qv) = -q\, \nabla \cdot v - v \cdot \nabla q$ and dividing by q. The advection terms cancel and what survives is the **instantaneous change-of-variables formula**

$$\boxed{\frac{\mathrm{d}}{\mathrm{d}t} \log q_t(x_t) = -\nabla \cdot v(x_t, t).} \tag{17}$$

Read it physically: $\nabla \cdot v$ is the local rate of volume expansion, and density dilutes exactly as fast as the volume around you stretches — conservation again, now per-particle. (It is also the familiar change-of-variables Jacobian done infinitesimally: $\log \det$ of the flow map's Jacobian grows at rate $\nabla \cdot v$.) This formula is the entire likelihood machinery of continuous normalizing flows, and Section 8.3 will cash it in for diffusion models.

Exercise 2. *Check* (17) *by hand for the scale flow* $\dot{x} = c_t\, x$ *in* $\mathbb{R}^d$ *with* $q_0 = \mathcal{N}(0, s^2 I)$*: solve the flow, write down* q_t*, and verify that both sides equal* $-d\, c_t$ *along every trajectory.*

> **The story so far.** *A Lipschitz velocity field integrates to a flow: a composable family of bijections whose trajectories never cross (uniqueness) and which reverses by flipping a sign — two free gifts the stochastic world will conspicuously lack. A density carried by the flow obeys the continuity equation* $\partial_t q = -\nabla \cdot (qv)$*, which is nothing but "flux = density × velocity" plus conservation; conversely, any* (q_t, v_t) *pair satisfying it is realized by integrating the field. Along your own trajectory,* $\log q$ *changes at rate* $-\nabla \cdot v$*: density dilutes as volume stretches.*

4 Interlude: the SDE toolkit, from a standing start

The sampling sections ahead run on stochastic differential equations and on their density-level shadows; this is the stochastic twin of the toolkit just built, and it is genuinely small: one process (Brownian motion), one rule

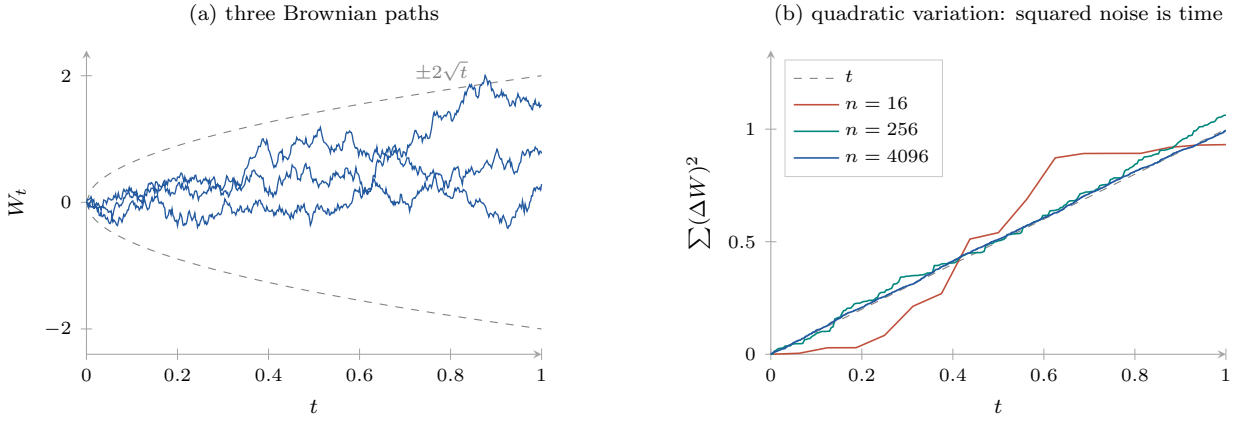

Figure 5: The two faces of $\sqrt{\mathrm{d}t}$. (a) Sample Brownian paths: individually rough beyond repair (zooming in never makes them look smoother — the wiggles are scale-free), collectively filling out the $\pm 2\sqrt{t}$ envelope. (b) The running sum of *squared* increments of one fixed path, chopped into n pieces: as the mesh refines, randomness drains out of the sum and it converges to the diagonal — equation (19), $(\mathrm{d}W)^2 = \mathrm{d}t$, in pictures.

(Itô's lemma, which is Taylor's theorem plus a single new fact), one solvable family (linear SDEs — the only SDEs this subject ever asks us to solve), and one bridge from paths to densities (Fokker–Planck, which will turn out to be the continuity equation with one extra flux). Readers who can solve an Ornstein–Uhlenbeck equation without pausing should read the recap at the end of the section and skip ahead.

4.1 Brownian motion: $\sqrt{\mathrm{d}t}$ is the whole personality

A standard **Brownian motion** (or Wiener process) W_t in $\mathbb{R}^d$ is the process with $W_0 = 0$, continuous paths, independent increments, and

$$W_t - W_s \;\sim\; \mathcal{N}\big(0,\,(t-s)\,I\big), \qquad 0 \le s < t. \tag{18}$$

That such a process exists is a theorem (Wiener, 1923); we take it off the shelf. Everything you need to know about it follows from one number: over a window of length $\mathrm{d}t$, the typical displacement is $\sqrt{\mathrm{d}t}$. Stare at that scaling — $\sqrt{\mathrm{d}t} \gg \mathrm{d}t$ for small $\mathrm{d}t$ — and two consequences fall out.

First: no velocity. The difference quotient $\Delta W/\Delta t \sim \Delta t^{-1/2}$ blows up as the window shrinks: Brownian paths are continuous everywhere and differentiable nowhere. A Brownian particle has a position at every time and a velocity at none. (File this away: it is why the subject will have to work to extract a *deterministic* velocity field from a stochastic process, an extraction Section 5 supplies.)

Second: squared noise is not noise. Take a partition $0 = t_0 < \cdots < t_n = t$ with mesh $\max_k \Delta t_k \to 0$ and sum the *squared* increments of a scalar Brownian motion:

$$\sum_k \big(W_{t_{k+1}} - W_{t_k}\big)^2 \;\longrightarrow\; t \qquad \text{in } L^2. \tag{19}$$

The mean of the left side is $\sum_k \Delta t_k = t$ on the nose, and its variance is $\sum_k 2\,\Delta t_k^2 \le 2t \max_k \Delta t_k \to 0$ (Exercise 3). A sum of squared noises is deterministic: it is elapsed time in disguise. This is the **quadratic variation** of Brownian motion, and we abbreviate it as

$$(\mathrm{d}W)^2 = \mathrm{d}t, \qquad \mathrm{d}t \cdot \mathrm{d}W = o(\mathrm{d}t), \qquad (\mathrm{d}t)^2 = o(\mathrm{d}t). \tag{20}$$

Figure 5 shows both faces: paths that fill out a $\sqrt{t}$ envelope while individually refusing to be smooth, and squared increments whose running sum hugs the diagonal more tightly the finer you chop.

Exercise 3. *Prove* (19)*: using* $\mathbb{E}[\xi^4] = 3$ *for* $\xi \sim \mathcal{N}(0,1)$*, show each squared increment has mean* Δt_k *and variance* $2\Delta t_k^2$*, and conclude by independence.*

4.2 What an SDE actually says

The expression

$$\mathrm{d}x = f(x,t)\,\mathrm{d}t + g_t\,\mathrm{d}W \tag{21}$$

is shorthand for an integral equation, $x_t = x_0 + \int_0^t f(x_s, s)\,\mathrm{d}s + \int_0^t g_s\,\mathrm{d}W_s$, but for our purposes the honest definition is operational: (21) is the small-h limit of the simulation

$$x_{t+h} \;=\; x_t + f(x_t, t)\,h + g_t\,\sqrt{h}\,\xi_t, \qquad \xi_t \sim \mathcal{N}(0, I) \text{ i.i.d.}, \tag{22}$$

the **Euler–Maruyama** scheme — drift times h, noise times $\sqrt{h}$, with the $\sqrt{h}$ now unsurprising. For Lipschitz f the limit exists and is unique; ours will be *linear* in x, the friendliest possible case.[9]

4.3 Itô's lemma is Taylor's theorem with one extra term

How does a smooth function φ of the state evolve? Expand to second order and keep everything of size $\mathrm{d}t$:

$$\mathrm{d}\varphi = \nabla\varphi \cdot \mathrm{d}x + \tfrac{1}{2}\,\mathrm{d}x^\top(\nabla^2\varphi)\,\mathrm{d}x + \cdots, \qquad \mathrm{d}x\,\mathrm{d}x^\top = \big(f\mathrm{d}t + g\,\mathrm{d}W\big)\big(\cdots\big)^\top = g_t^2\,I\,\mathrm{d}t + o(\mathrm{d}t), \tag{23}$$

using (20) to kill every term except the $\mathrm{d}W\mathrm{d}W^\top$ square. The result is **Itô's lemma**:

$$\boxed{\mathrm{d}\varphi = \Big(f\cdot\nabla\varphi + \tfrac{g_t^2}{2}\,\Delta\varphi\Big)\,\mathrm{d}t + g_t\,\nabla\varphi\cdot\mathrm{d}W.} \tag{24}$$

The deterministic chain rule keeps only first-order terms; the stochastic chain rule must keep the second-order one, because the square of noise is time. That $\frac{g^2}{2}\Delta$ is the source of every diffusion term in every PDE of this subject. Sanity check, $\varphi = x^2$ for pure Brownian motion ($f = 0$, $g = 1$, scalar): $\mathrm{d}(W^2) = \mathrm{d}t + 2W\,\mathrm{d}W$, so $\mathbb{E}[W_t^2] = t$ — the variance growth recovered as a calculus statement.

Exercise 4. *Use* (24) *with* $\varphi = x^4$ *to show* $\mathbb{E}[W_t^4] = 3t^2$*, confirming the fourth-moment input to Exercise 3.*

4.4 Linear SDEs, solved completely — and the forward SDE derived

Now the family we actually use: $\mathrm{d}x = f_t\,x\,\mathrm{d}t + g_t\,\mathrm{d}W$ with time-dependent scalars f_t, g_t. Set $\alpha_t = \exp\big(\int_0^t f_s\mathrm{d}s\big)$ and change variables to $y = x/\alpha_t$. Because α_t is deterministic and y is linear in x (so $\nabla^2 y = 0$ and Itô's correction vanishes), plain calculus applies:

$$\mathrm{d}y = \frac{\mathrm{d}x}{\alpha_t} - \frac{x\,\dot\alpha_t}{\alpha_t^2}\,\mathrm{d}t = \frac{g_t}{\alpha_t}\,\mathrm{d}W \quad\Longrightarrow\quad x_t = \alpha_t\,x_0 + \alpha_t\int_0^t \frac{g_s}{\alpha_s}\,\mathrm{d}W_s. \tag{25}$$

The integral is a weighted sum of independent Gaussian increments with deterministic weights, hence Gaussian, with mean zero and variance given by the **Itô isometry** $\mathbb{E}\big[\big(\int h_s\mathrm{d}W_s\big)^2\big] = \int h_s^2\,\mathrm{d}s$ (Exercise 5). Therefore

$$x_t \mid x_0 \;\sim\; \mathcal{N}\big(\alpha_t x_0,\; \sigma_t^2 I\big), \qquad \sigma_t^2 = \alpha_t^2\int_0^t \frac{g_s^2}{\alpha_s^2}\,\mathrm{d}s. \tag{26}$$

This is exactly the interpolant (1)'s conditional law — and now run the logic *backwards*. Given a designer pair (α_t, σ_t), what SDE realizes it? Matching the mean forces $f_t = \dot\alpha_t/\alpha_t$; differentiating the variance relation in (26) gives $\frac{\mathrm{d}}{\mathrm{d}t}\sigma_t^2 = 2f_t\sigma_t^2 + g_t^2$, i.e.

$$g_t^2 = 2\sigma_t\dot\sigma_t - 2\,\frac{\dot\alpha_t}{\alpha_t}\,\sigma_t^2. \tag{27}$$

These are precisely the coefficients that will appear in Section 5: not an ansatz to be checked but the unique linear SDE with the prescribed conditional law.

The constant-coefficient case $f_t \equiv -\theta$, $g_t \equiv g$ is the **Ornstein–Uhlenbeck process**: mean decays like $e^{-\theta t}$, variance saturates at $g^2/2\theta$, and the process forgets its initial condition exponentially fast, equilibrating to $\mathcal{N}(0, \frac{g^2}{2\theta})$. The VP diffusion is an OU process on a deformed clock — which is the path-level picture of "data is shredded into a standard Gaussian." Figure 6 shows it happening.

Exercise 5. *Prove the Itô isometry for piecewise-constant* h*: expand the square of* $\sum_k h_{t_k}(W_{t_{k+1}} - W_{t_k})$ *and use independence of increments. (General* h *follows by approximation; take that on faith.)*

[9] One genuine subtlety is dodged here: if the noise amplitude depended on the state, $g(x_t, t)$, the limit of (22) would depend on *where in the interval* you evaluate g — evaluate at the left endpoint and you get the Itô integral, at the midpoint and you get Stratonovich, and the two differ by a drift term of size $O(g\,\partial_x g)$, because g's fluctuations correlate with $\mathrm{d}W$'s. Every SDE in the diffusion-model literature has g depending on t only, so the ambiguity never bites, and we shall not mention it again.

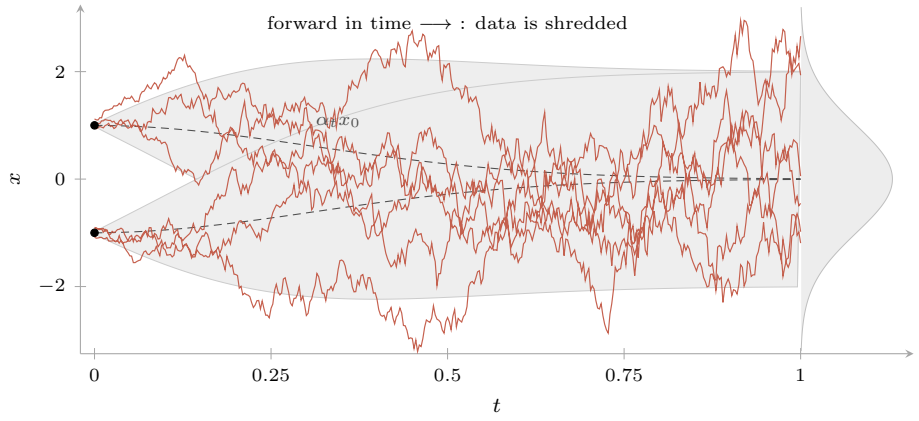

Figure 6: The forward VP diffusion doing its one job, simulated by Euler–Maruyama (22) from the two data modes $x_0 = \pm 1$ (red paths). Dashed: the conditional means $\alpha_t x_0$ from (26), sliding to zero. Gray tubes: the conditional $\pm 2\sigma_t$ bands, inflating until the two tubes merge; by $t = 1$ the marginal is the standard Gaussian (right profile), and nothing about a path's endpoint betrays which mode it came from. Same picture as Figure 7, opposite direction of travel: this is the easy direction, and the entire subject is the art of traversing it backwards.

4.5 From paths to densities: the Fokker–Planck equation

A process is a cloud of paths; call the cloud's density q_t. Section 3.2 taught us how to find a density's PDE: identify the flux. The drift is a velocity field like any other and contributes the advective flux $f(x,t)\,q_t$ of (15). The genuinely new question is what flux the *noise* carries. Isolate it: take pure noise, $\mathrm{d}x = g_t\,\mathrm{d}W$. One step is a Gaussian blur — $x_{t+h} = x_t + g\sqrt{h}\,\xi$ convolves the density with $\mathcal{N}(0, g^2 h I)$ — so Taylor-expand under the convolution:

$$q_{t+h}(x) = \mathbb{E}_\xi\big[q_t(x - g\sqrt{h}\,\xi)\big] = q_t(x) - g\sqrt{h}\,\underbrace{\mathbb{E}[\xi]}_{=0}\cdot\nabla q_t + \tfrac{g^2 h}{2}\,\underbrace{\mathbb{E}\big[\xi^\top(\nabla^2 q_t)\,\xi\big]}_{=\Delta q_t} + O(h^2), \tag{28}$$

using $\mathbb{E}[\xi\xi^\top] = I$ and the vanishing of all odd moments. So pure noise obeys $\partial_t q = \frac{g^2}{2}\Delta q$ — the heat equation — and the heat equation is itself in conservation form: $\frac{g^2}{2}\Delta q = -\nabla\cdot\big(-\frac{g^2}{2}\nabla q\big)$. Diffusion is a flux $-\frac{g^2}{2}\nabla q$ that drives probability *down its own density gradient* — Fick's law, here derived rather than postulated. Drift and noise contribute independently at order h (their cross-terms are $O(h^{3/2})$), so the fluxes add:

$$\boxed{\partial_t q_t = -\nabla\cdot\big(f\,q_t\big) + \tfrac{g_t^2}{2}\,\Delta q_t \;=\; -\nabla\cdot\Big(\underbrace{f\,q_t - \tfrac{g_t^2}{2}\,\nabla q_t}_{\text{total flux}}\Big).} \tag{29}$$

This is **Fokker–Planck** (the physicists' name) or **Kolmogorov forward** (the probabilists'): the continuity equation with the flux enlarged from qv to advection plus down-gradient diffusion — probability still neither created nor destroyed, only carried and spread. And — this is the loaded gun placed on the mantelpiece — the equation depends on the process only through the pair (f, g_t^2) *and constrains only the marginals*: very different processes can share one and the same family q_t. Section 5 fires this gun.

Exercise 6. *Re-derive* (29) *the weak way: take expectations of Itô's lemma* (24) *(the* $\mathrm{d}W$ *term has mean zero and dies), write* $\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[\varphi(x_t)] = \mathbb{E}[f\cdot\nabla\varphi + \frac{g^2}{2}\Delta\varphi]$ *as integrals against* q_t*, and integrate by parts — once on the drift term, twice on the diffusion term — to move the derivatives onto the density. Same equation, different dress.*

4.6 Langevin dynamics: the SDE that stands still

One special case deserves its own subsection. Fix a target density p and consider, for a step-size parameter $\eta > 0$,

$$\mathrm{d}x = \tfrac{\eta}{2}\,\nabla\log p(x)\,\mathrm{d}t + \sqrt{\eta}\,\mathrm{d}W. \tag{30}$$

Plug $f = \frac{\eta}{2}\nabla\log p$, $g^2 = \eta$ into (29) and evaluate at $q = p$:

$$-\tfrac{\eta}{2}\nabla\cdot\big(p\,\nabla\log p\big) + \tfrac{\eta}{2}\,\Delta p = -\tfrac{\eta}{2}\nabla\cdot\big(\nabla p\big) + \tfrac{\eta}{2}\,\Delta p = 0, \tag{31}$$

using $p\,\nabla \log p = \nabla p$. So p is stationary: the score-driven drift uphill exactly balances the diffusive spreading. **Langevin dynamics** is the SDE that goes nowhere, distributionally — and notice what it needs: *only the score*, the normalization-free object of Section 2. Run (30) long enough and (under conditions we will not fuss over) it samples p. Why is that not already a generative model, given a learned score at noise level zero? Because Langevin mixes between well-separated modes at a rate that is exponentially small in the barrier height, and near a low-dimensional data manifold the score is enormous and ill-behaved off-manifold. The historical fix (annealed Langevin, SMLD) and the modern subject are the same idea: maintain the whole family $\{q_t\}$, mix at high noise where mixing is free, and use the low-noise scores only for the final approach. Keep (30) in your pocket: in Section 5 it reappears inside the reverse-time SDE as the "churn" that holds a sampler on the marginal q_t while transport does the real work.

Exercise 7. *Show that adding any vector field u with $\nabla\cdot(p\,u) = 0$ to the drift of* (30) *leaves p stationary. (Moral: the dynamics realizing given marginals are never unique — remember this when the marginal velocity of Section 7 is introduced with a definite article.)*

> **The story so far.** *Brownian increments scale as $\sqrt{\mathrm{d}t}$, so squared increments are deterministic: $(\mathrm{d}W)^2 = \mathrm{d}t$. That single fact upgrades Taylor to Itô* (24)*, solves every linear SDE in closed form — yielding exactly the interpolant's Gaussian conditionals and hence, read backwards, deriving the forward SDE's coefficients — and, in expectation, turns path statements into the density statement* (29)*. Langevin dynamics is the special SDE whose Fokker–Planck flow vanishes: score drift balancing diffusion, the score's normalization-freeness made kinetic.*

5 Sampling I: the probability flow ODE

We have (an estimate of) the scores. How do they generate? The cleanest route — cleaner than the historical one — goes through a small calculation with the Fokker–Planck equation (29), and it is worth doing in full because it is three lines and it manufactures the bridge between diffusions and flows.

The interpolant (1) can be realized as a forward-time SDE that gradually shreds data,

$$\mathrm{d}x = f_t\,x\,\mathrm{d}t + g_t\,\mathrm{d}W, \qquad f_t = \frac{\dot\alpha_t}{\alpha_t}, \quad g_t^2 = 2\big(\sigma_t\dot\sigma_t - \sigma_t^2\tfrac{\dot\alpha_t}{\alpha_t}\big), \tag{32}$$

— these coefficients were *derived* in Section 4.4: match the conditional mean to get f_t, differentiate the conditional variance to get g_t. (Note also $g_t^2 = 2\sigma_t^2\frac{\mathrm{d}}{\mathrm{d}t}\log(\sigma_t/\alpha_t)$, which is nonnegative precisely when the signal-to-noise ratio α_t^2/σ_t^2 is decreasing — a schedule is realizable as a diffusion exactly when it destroys information at every instant.) Its marginals obey Fokker–Planck (29) with linear drift:

$$\partial_t q_t = -\nabla\cdot\big(f_t\,x\,q_t\big) + \tfrac{g_t^2}{2}\,\Delta q_t. \tag{33}$$

Now the move. The diffusion term is also a divergence — of a flux proportional to the score: $\Delta q_t = \nabla\cdot\nabla q_t = \nabla\cdot\big(q_t\,\nabla\log q_t\big)$. Substitute and gather:

$$\partial_t q_t = -\nabla\cdot\Big[\Big(\underbrace{f_t\,x - \tfrac{g_t^2}{2}\,\nabla\log q_t(x)}_{v_t(x)}\Big)\,q_t\Big]. \tag{34}$$

But (34) is precisely the continuity equation (15) for the velocity field v_t — and the converse direction of Section 3.2 now does the work: the same family of marginals q_t is carried by the deterministic ODE $\dot x = v_t(x)$. No noise anywhere. The stochastic shredding process and this **probability flow ODE** are statistically indistinguishable at the level of single-time marginals — and the ODE can be run *backwards*, from $x_1 \sim \mathcal{N}(0, I)$ to a data sample, by any off-the-shelf integrator, with the learned score plugged in (Section 8 examines how few steps the integrator can get away with).

The randomness of the diffusion was an implementation detail. The footprint it leaves on the marginals can be reproduced by a velocity field, and the velocity field is an affine function of the score (9). Deterministic samplers (DDIM is a particular discretization of this ODE), flow likelihoods (the change-of-variables formula (17) applies the moment a flow exists), and the entire flow-matching viewpoint all enter through equation (34).

And since the training loop got its six lines, generation deserves its six — the probability-flow ODE under Euler, run from noise back to data:

```
@torch.no_grad()
def sample_pf_ode_eps_model(model, shape, steps):
    x  = torch.randn(shape)                    # x_1 ~ N(0, I)
    n  = shape[0]
    bshape = (-1,) + (1,) * (x.ndim - 1)
    ts = torch.linspace(1.0, 0.0, steps + 1)
    for t, t_next in zip(ts[:-1], ts[1:]):
        tt = t.expand(n).to(x.device)
        score = -model(x, tt) / sigma(tt).view(bshape)  # epsilon -> score
        v = (f(tt).view(bshape) * x
             - 0.5 * g2(tt).view(bshape) * score)
        x = x + (t_next - t) * v                    # Euler, backwards
    return x
```

Swapping in the reverse SDE (35) adds one line of injected noise; choosing the integrator, the step count, and where the steps go is Section 8's entire business.

5.1 Sampling II: putting the noise back (and why you might)

You can also reverse the SDE itself — though, unlike the ODE, not for free: Section 3.1's sign-flip gift is exclusive to deterministic flows, because a Brownian increment does not unhappen when you negate a drift. Reversal here costs a theorem and a score. The classical result (Anderson, 1982) is that the time-reversal of (32) is again an SDE,

$$\mathrm{d}x = \big[f_t\, x - g_t^2\, \nabla \log q_t(x)\big]\, \mathrm{d}t + g_t\, \mathrm{d}\bar{W} \qquad \text{(run backwards)}, \tag{35}$$

with *twice* the score correction of the ODE plus reinjected noise. Anderson's theorem has a fearsome reputation and a four-line proof-by-Bayes,[10] so rather than dwell on the proof, see the structure: (35) = probability-flow ODE + $\big[\frac{g_t^2}{2} \nabla \log q_t\, \mathrm{d}t + g\, \mathrm{d}\bar{W}\big]$, and that bracketed extra, per unit of *reverse* time, is precisely a step of *Langevin dynamics* (30) targeting q_t — the SDE that stands still (Section 4.6), leaving each marginal invariant. So the reverse SDE is "transport + equilibration": the ODE part moves mass between noise levels; the Langevin part churns within a noise level, correcting accumulated error at the cost of extra stochasticity. Every sampler in the literature — DDPM ancestral sampling, DDIM with its η knob, predictor–corrector schemes, EDM's churn parameter — is a point on this one-parameter family between pure ODE and heavily-churned SDE. Deterministic is fast and reproducible; stochastic forgives a sloppy score model. Figure 7 shows both samplers riding the same path of marginals.

> **The story so far.** *Rewriting the diffusion's Fokker–Planck equation as a continuity equation shows its marginals are equally generated by a deterministic velocity field $v_t = f_t x - \frac{g_t^2}{2} \nabla \log q_t$: the probability flow ODE. The reverse SDE is this ODE plus marginal-preserving Langevin churn. Sampling is numerical integration; the stochastic/deterministic choice is a robustness/speed dial, not a modeling commitment.*

6 Interlude: DDPM in discrete time — the chain, the ELBO, the VAE

Before flows, two reconciliations with the literature's native dialect: the discrete Markov chain that DDPM actually defines (and the $\bar{\alpha}$ notation it spawned, a reliable generator of bugs), and the ELBO that trains it — which will turn out to be a VAE with every classic failure mode engineered away.

6.1 The discrete chain, and where $\bar{\alpha}$ comes from

DDPM as published is not an SDE; it is a Markov chain of N corruption steps,

$$q(x_k \mid x_{k-1}) = \mathcal{N}\big(\sqrt{1-\beta_k}\, x_{k-1},\ \beta_k I\big), \qquad k = 1, \ldots, N, \tag{36}$$

[10] One Euler–Maruyama step of the forward SDE is a Gaussian kernel $q(x_{t+h} \mid x_t)$. Bayes inverts it: $q(x_t \mid x_{t+h}) \propto q(x_{t+h} \mid x_t)\, q_t(x_t)$. Expand $\log q_t(x_t)$ to first order around x_{t+h} and complete the square, and out falls a Gaussian reverse kernel with mean $x_{t+h} - h\big[f_t\, x_{t+h} - g_t^2 \nabla \log q_t(x_{t+h})\big] + O(h^2)$ and covariance $g_t^2 h\, I$ — one Euler–Maruyama step of (35). Problem 4 asks you to push the algebra through and locate where the factor of *two*, relative to the ODE's $\frac{g^2}{2}$, comes from.

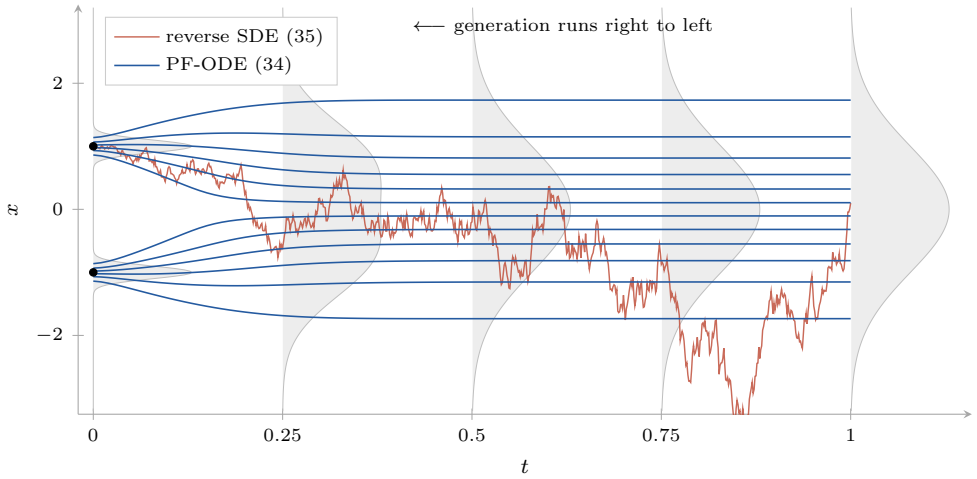

Figure 7: One path of marginals, two ways to traverse it, computed exactly for $p_{\text{data}} = \frac{1}{2}\mathcal{N}(-1, 0.1^2) + \frac{1}{2}\mathcal{N}(+1, 0.1^2)$ under the VP schedule (β linear in $[0.1, 20]$). Gray profiles: the marginal density q_t at $t = 0, 0.25, \ldots, 1$ (widths normalized per slice), morphing from two sharp data modes to a standard Gaussian. Blue: probability-flow ODE trajectories (34) started from twelve quantiles of $\mathcal{N}(0, 1)$ and integrated backwards — smooth, deterministic, never crossing (the uniqueness gift of Section 3.1), splitting at the ridge between basins. Red: one reverse-SDE path (35) from $x_1 = 0.1$ — same drift family, but the Langevin churn keeps it rattling around inside each marginal before it commits to a mode. Both end with the correct statistics; only the red one can change its mind.

with small per-step variances β_k. The $\sqrt{1-\beta_k}$ is the variance-preserving choice in discrete dress: if $\mathbb{E}\|x_{k-1}\|^2 = d$ then $\mathbb{E}\|x_k\|^2 = (1-\beta_k)\,d + \beta_k\,d = d$. Because each step is a linear-Gaussian map, the chain composes in closed form (check by adding variances of independent Gaussians):

$$x_k = \sqrt{\bar\alpha_k}\,x_0 + \sqrt{1-\bar\alpha_k}\,\varepsilon, \qquad \bar\alpha_k = \prod_{j\le k}(1-\beta_j), \qquad \varepsilon \sim \mathcal{N}(0, I). \tag{37}$$

Set this against the interpolant (1) and the dictionary is

$$\alpha_{t_k} = \sqrt{\bar\alpha_k}, \qquad \sigma_{t_k} = \sqrt{1-\bar\alpha_k} \qquad \text{— } \textit{DDPM's } \bar\alpha \textit{ is our } \alpha \textit{ squared.} \tag{38}$$

That single squaring is responsible for a measurable fraction of all sign and scale errors in this literature (an early draft of Figure 1's caption included — the author confidently quoted an $\bar\alpha$ as an α); when opening any paper, the first thing to establish is whether its alphas are ours or their squares.

The continuum limit connects the chain to everything we have built. Put $\beta_k = \beta(t_k)\,h$ with $h = 1/N$ and expand (36): $\sqrt{1-\beta h} = 1 - \frac{1}{2}\beta h + O(h^2)$, so one step reads

$$x_k - x_{k-1} = -\tfrac{1}{2}\,\beta(t_k)\,h\,x_{k-1} + \sqrt{\beta(t_k)\,h}\;\xi_k, \tag{39}$$

which is exactly one Euler–Maruyama step (22) of the VP forward SDE $\mathrm{d}x = -\frac{1}{2}\beta x\,\mathrm{d}t + \sqrt{\beta}\,\mathrm{d}W$. Likewise $\log\bar\alpha(t) = \sum_j \log(1-\beta_j h) \to -\int_0^t \beta$, recovering the table row of Section 1.1, $\alpha_t = e^{-\frac{1}{2}\int_0^t \beta}$. The same correspondence runs through sampling: DDPM's ancestral sampler is the Euler–Maruyama discretization of the reverse SDE (35) (and Problem 4's Bayes-expansion derivation of Anderson is nothing but this discrete chain inverted step by step, then sent to the limit). Discrete DDPM and the continuous-time story are one model at two resolutions; these notes stay continuous because the calculus is lighter, and translate at the boundary with the dictionary above.

6.2 The ELBO: a VAE with an incorruptible encoder

Keep the discretization, $0 = t_0 < t_1 < \cdots < t_N = 1$, and view $x_{t_1}, \ldots, x_{t_N}$ as a *hierarchy of latent variables* above the data x_0. The forward corruption $q(x_{t_k} \mid x_{t_{k-1}})$ is the encoder; the learned reverse model $p_\theta(x_{t_{k-1}} \mid x_{t_k})$ is the decoder; the standard ELBO telescopes into

$$\begin{aligned}\log p_\theta(x_0) \;\ge\; & \mathbb{E}_q\Big[\log p_\theta(x_0 \mid x_{t_1})\Big] - \sum_{k=2}^{N} \mathbb{E}_q\,\mathrm{KL}\Big(q(x_{t_{k-1}} \mid x_{t_k}, x_0)\,\Big\|\,p_\theta(x_{t_{k-1}} \mid x_{t_k})\Big) \\ & - \mathrm{KL}\big(q(x_{t_N} \mid x_0)\,\|\,p(x_{t_N})\big).\end{aligned} \tag{40}$$

Read the three pieces from right to left. The final KL asks whether the last corrupted sample is actually close to the chosen prior; it contains no parameters, and with a long enough schedule it is also near zero. The middle sum is the real training signal: at each noise level, the model's one-step reverse Gaussian must match the analytic posterior one would get if the clean x_0 were known. The first term is the final reconstruction or decoder term at the data end, whose exact treatment depends on the data likelihood convention (Gaussian for continuous data, discretized logistic or categorical variants for pixels), but which plays the same role as the bottom decoder term of an ordinary VAE.

Every middle term is between Gaussians. The forward posterior $q(x_{t_{k-1}} \mid x_{t_k}, x_0)$ is Gaussian with closed-form mean (Problem 3); if the reverse model uses the same variance, its KL is just a constant times the squared error between the true posterior mean and the model's predicted mean. Parameterize that predicted mean by $\hat{\varepsilon}_\theta$ using (37), and the same KL becomes a weighted version of the noise-prediction loss (8). DDPM's celebrated "simple loss" is (40) with the per-level weights thrown away (which empirically helps samples and hurts likelihoods — you are reweighting how much the model cares about coarse versus fine scales).

Now compare with the VAEs of your past life, because every classic failure mode is structurally absent:

- The encoder is *fixed and analytic* — there is nothing to collapse, no posterior to fight the prior, no pressure toward the mean image. The eternal tug-of-war between reconstruction and KL has been settled by fiat: the "inference network" is a noise schedule.
- The latent has the data's dimension and the prior is exactly reached by construction ($q_{t_N} \approx \mathcal{N}$): the aggregate-posterior / prior mismatch — the hole-in-the-prior problem — is engineered away.
- What is given up: a one-shot encoder–decoder pass becomes an N-step decoder, and the rich learned latent semantics of a good VAE are replaced by ... noise. Diffusion buys its stability by making the latent space maximally boring.[11]

7 Flow matching: regress on the velocity directly

Section 5 produced a velocity field by way of scores and Fokker–Planck. Flow matching asks the impertinent question: *why take the detour?* If generation is just integrating $\dot{x} = v_t(x)$, let's learn v_t by regression, directly.

Two obstacles, both instructive. First, which v_t? For a given path of marginals $\{q_t\}$ there are many velocity fields satisfying the continuity equation (add any divergence-free-relative-to-q_t field — this is Exercise 7 again, minus the noise); we should pick a canonical one. Second — the apparent killer — the canonical choice depends on the unknown marginals, exactly as the score did. The resolution is exactly the resolution of Section 2, which is the punchline of these notes: *flow matching is Lemma 1 applied to velocities instead of scores.*

Dictionary before the switch. Four vector fields are now in play, and keeping them separate prevents most mistakes. The **score** $s = \nabla \log q_t$ points up the noisy marginal density. The **probability-flow velocity** $f_t x - \frac{1}{2} g_t^2 s$ is the deterministic field manufactured from that score in Section 5. The **conditional velocity** below is a per-sample derivative of one interpolant path. The **marginal velocity** is the posterior average of those conditional velocities over all sample pairs passing through the same x_t; for the Gaussian interpolants used in these notes, it will equal the probability-flow velocity exactly.

Differentiate the interpolant (1) along its own clock: each pair (x_0, ε) traces the **conditional path** $x_t = \alpha_t x_0 + \sigma_t \varepsilon$ with **conditional velocity**

$$\dot{x}_t = \dot{\alpha}_t \, x_0 + \dot{\sigma}_t \, \varepsilon \qquad \text{— known, per sample, in closed form.} \tag{41}$$

Define the **marginal velocity** as the conditional average of (41) over everything passing through x at time t:

$$v_t(x) \;=\; \mathbb{E}_{(x_0,\varepsilon)\sim p(x_0,\varepsilon \mid x_t = x)} \big[\dot{\alpha}_t x_0 + \dot{\sigma}_t \varepsilon \big], \tag{42}$$

the measure spelled out as in (6): the joint posterior over *pairs*, which is supported on the affine slice $\{(x_0, \varepsilon) : \alpha_t x_0 + \sigma_t \varepsilon = x\}$ — the phrase "everything passing through x" made into a probability distribution, each line through x weighted by how likely it was to be the one you are on. A short continuity-equation computation (Problem 2) confirms that (42) transports the marginals q_t — indeed, applying Tweedie (4)

[11] Latent diffusion gets some semantics back by running the diffusion inside a (frozen) VAE's latent space — the two technologies composed, each doing what it is good at: the VAE compresses, the diffusion generates.

inside (42) reproduces *exactly* the probability-flow velocity of (34). Same object, second derivation, no Fokker–Planck required.

And now Lemma 1, verbatim, with $A = \dot\alpha_t x_0 + \dot\sigma_t \varepsilon$ and $B = (x_t, t)$: the **conditional flow matching** objective

$$\boxed{\mathcal{L}_{\mathrm{CFM}} = \mathbb{E}_{t,\,x_0,\,\varepsilon}\big\|v_\theta(x_t,t) - \big(\dot\alpha_t\, x_0 + \dot\sigma_t\, \varepsilon\big)\big\|^2} \tag{43}$$

is minimized exactly at the marginal velocity (42), even though the network only ever sees the crude per-sample targets (41). The gradients of (43) and of the intractable "regress on v_t directly" objective coincide. (Figure 3 is once more the picture, with $\dot\alpha_t x_0 + \dot\sigma_t \varepsilon$ in place of ε on the vertical axis.)

Two remarks to let this land. First, put (8) and (43) side by side: for this Gaussian-interpolant setting, they have the same shape — sample a pair, form the interpolant, regress on a closed-form per-sample vector — differing only in which member of the wardrobe (9) is worn. In that precise sense, score matching and flow matching are two parameterizations of the same marginal transport field. In the wider flow-matching literature one can choose more general conditional paths and couplings, so the slogan should not be read as a theorem about every object bearing the name. Second, the historical significance for the flows community: continuous normalizing flows were always elegant and always untrainable at scale, because computing the likelihood meant *solving the ODE inside the training loop*. Flow matching is CNF training gone *simulation-free*: no solver until sampling time. That single property is why the method took over.

7.1 Rectified flow: the case for straight lines

Among schedules, the linear one — $\alpha_t = 1 - t$, $\sigma_t = t$ — has a distinguished personality. Its conditional velocity (41) is *constant in time*:

$$\dot x_t = \varepsilon - x_0, \tag{44}$$

each training pair contributing a straight line from data to noise traversed at uniform speed, and the CFM target losing its time dependence entirely: predict the displacement. (In the popular convention this is "rectified flow" or the "stochastic interpolant" with linear path; it is the training objective behind several current large image models.)

Why care about straightness? Because of the cost model of sampling. The expensive primitive is a network evaluation; an ODE whose trajectories are straight is integrated *exactly by one Euler step* (a slogan that Section 8 sharpens into a statement about truncation error). Now, a subtlety that is worth being slow about, because it is the most commonly garbled point in the subject: **straight conditional paths do not make the marginal ODE's trajectories straight**. The marginal velocity (42) is an *average* of crossing straight lines, and averaging straight lines that cross produces curved integral curves (the average direction changes as the mix of lines overhead changes). The linear schedule gives you straight *targets* and a clean objective, not one-step sampling. Figure 8 shows the distinction at a glance, and the resolution. The genuinely clever move — *reflow* (rectification) — is iterative: sample pairs (x_1, x_0') by running your current model's ODE, so that endpoints are *coupled* rather than independent, and retrain on the linear interpolant of these non-crossing pairs. Each round provably straightens trajectories (it can only reduce the transport cost), and a round or two gets close enough to straight that one–four step sampling works. Distillation methods (consistency models, progressive distillation) attack the same step-count problem by other means; the underlying currency is always trajectory curvature versus network calls.[12]

7.2 The optimal transport vista

The abstract promised a community that "speaks of optimal transport," and one debt from the toolkit is still outstanding: Exercise 7 showed that the dynamics realizing a given family of marginals are never unique, and left hanging the question *then which one?* Optimal transport is the principled answer, and — now that the continuity equation is built — it costs one definition. The **Benamou–Brenier** formulation: among all pairs (q_t, v_t) satisfying the continuity equation (15) with prescribed endpoints q_0 and q_1, choose the one of least total kinetic energy,

$$\inf_{(q,v)} \int_0^1 \int \big\|v_t(x)\big\|^2\, q_t(x)\,\mathrm{d}x\,\mathrm{d}t \qquad \text{subject to} \qquad \partial_t q + \nabla\cdot(qv) = 0, \quad q_0, q_1 \text{ fixed.} \tag{45}$$

The character of the minimizer is the one sentence we actually need: *every particle travels in a straight line at constant speed*, and the optimal value is (by definition, if you like) the squared Wasserstein-2 distance

[12] Readers of the previous notes in this series will recognize the shape of the argument: the cost model (network evaluations) selects the geometry (straight paths) — choose the chart in which your expensive primitive is cheap. Table-making lessons transfer.

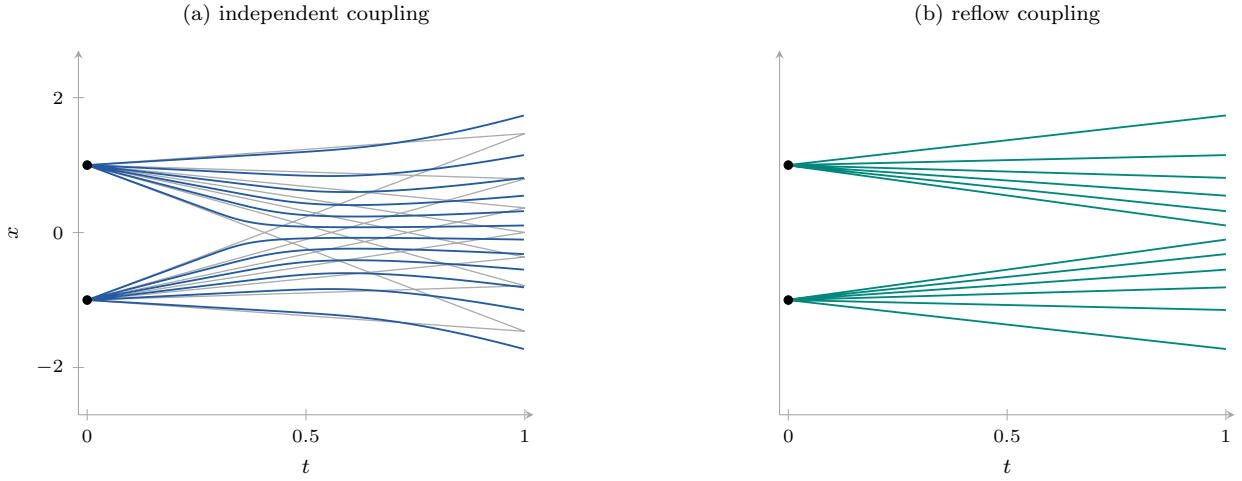

Figure 8: Straight targets versus straight trajectories, exactly computed for data concentrated at $x_0 = \pm 1$ with the linear schedule. (a) Under the independent coupling of training, the conditional paths (gray) are straight lines from each data point to each noise sample — and they cross. The marginal velocity (42) at a crossing point must average the gray directions overhead, so the integral curves of the probability-flow ODE (blue) bend: flat near $t = 1$ where the average is dominated by "head for the origin," then committing to a mode as the posterior sharpens. Euler with one step would follow the tangent at $t = 1$ and miss. (b) Reflow replaces the coupling: each noise point is paired with *its own ODE endpoint*, and the straight interpolants of these coupled pairs (teal) no longer cross (non-crossing again: the uniqueness gift of Section 3.1 makes the one-dimensional flow monotone, so the coupling is order-preserving). Retrained on these pairs, the marginal flow has straight trajectories and one Euler step is exact. The coupling change, not the linear schedule, is the active ingredient.

$W_2^2(q_0, q_1)$ — we need the phrase, not the theory. Why straight lines? The population problem inherits a single-particle fact: for a path γ on $[0, 1]$ with fixed endpoints, $\int_0^1 \|\dot\gamma\|^2 \, \mathrm{d}t \geq \|\gamma(1) - \gamma(0)\|^2$, with equality exactly for constant velocity (Exercise 8) — wiggling and speed variation are pure energetic waste. On top of that, the optimal *pairing* of sources to targets never lets two transport rays cross: if they crossed, swapping targets would lower the combined cost (one leg may lengthen; for quadratic cost the sum cannot fail to drop). The same non-crossing geometry, a third time.

Now place the subject on this map, honestly. *One*: the diffusion/flow-matching path of marginals is **not** the OT geodesic. The Gaussian-corruption path was chosen for trainability — closed-form conditionals, Lemma 1 applicable — not for kinetic economy; the probability-flow field is the canonical transport of a deliberately non-optimal path. *Two*: rectified flow's training pairs are straight rays with the *independent* coupling, and independent rays cross (Figure 8(a)) — precisely what an optimal plan never does. Straight targets alone do not an OT solution make. *Three*: reflow moves toward kinetic economy without arriving. Each round replaces the coupling by the model's own flow map and provably does not increase the transport cost $\mathbb{E}\|x_1 - x_0\|^2$ (it strictly decreases it unless the paths are already straight); fixed points are straight, non-crossing flows. But "straight" is a larger club than "optimal": reflow converges to *a* straight coupling, not necessarily *the* OT one. The cost model of sampling (Section 8: Euler is exact on straight trajectories) prices exactly straightness, not optimality, so reflow buys everything the sampler can spend. Least kinetic energy and fewest network calls are different objectives that happen to share a minimizer-shape; that coincidence is why straight lines keep winning scenes they were never cast in. Everything deeper — duality, Brenier maps, geodesics in Wasserstein space — is real mathematics with real payoff elsewhere; none of it is needed here.

Exercise 8. *Prove the single-particle lemma: for $\gamma : [0, 1] \to \mathbb{R}^d$ with fixed endpoints, $\int_0^1 \|\dot\gamma\|^2 \mathrm{d}t \geq \|\gamma(1) - \gamma(0)\|^2$, with equality iff $\dot\gamma$ is constant. (Jensen, or Cauchy–Schwarz against the constant function.)*

The story so far. *Define a marginal velocity as the conditional average of the per-sample interpolant velocity; Lemma 1 makes it learnable by plain regression (CFM), and Tweedie shows it equals the probability-flow velocity for the Gaussian interpolants used here. In this setting, flow matching and score matching are two parameterizations of the same marginal transport field. The linear schedule gives constant per-sample targets; actual one-step sampling additionally needs the* coupling *fixed by reflow, because averages of crossing straight lines are curved. Benamou–Brenier names the ideal in*

the background: least kinetic energy among all transports means straight lines at constant speed; reflow descends that objective far enough for the sampler's purposes without reaching its minimizer.

8 Sampling III: a solver's-eye refresher

Training, we now know, is one lemma used twice. The remaining cost of the whole enterprise lives at sampling time: integrate $\dot{x} = v_\theta(x, t)$ from $t = 1$ to $t = 0$ using as few *network function evaluations* (NFE) as possible — the network call is the only expensive primitive, so NFE is the only currency. This section is the numerical-analysis refresher that the literature assumes: what "order" buys, what structure the diffusion ODE has that a generic solver wastes, and what the stochastic alternative really costs.

8.1 One-step methods, order, and what curvature has to do with it

A one-step method advances x across a step h using some recipe built from evaluations of v; its **order** is p if the error of a single step is $O(h^{p+1})$, so that the errors of the $\sim 1/h$ steps needed to cross the interval accumulate to a *global* error $O(h^p)$. The catalogue, in the only sizes we need:

Euler ($p = 1$, one NFE per step): $x \leftarrow x + h\, v(x, t)$. Taylor-expand the true solution and subtract: the leading local error is $\frac{1}{2}h^2\ddot{x}$, and along a trajectory of $\dot{x} = v$,

$$\ddot{x}_t \;=\; \frac{\mathrm{d}}{\mathrm{d}t}\, v(x_t, t) \;=\; \big(\partial_t v + (v \cdot \nabla)\, v\big)(x_t, t), \tag{46}$$

the *material acceleration* of the flow — the same total-derivative-along-the-trajectory that produced the change-of-variables formula (17), here applied to v instead of $\log q$. Euler's error *is* curvature, literally: the method is exact precisely when trajectories are straight lines traversed at constant speed (Exercise 9). Section 7's slogan — straightness is wealth, reflow buys it — is a statement about equation (46).

Heun ($p = 2$, two NFE per step): predict with Euler, correct with the trapezoid rule,

$$\tilde{x} = x + h\, v(x, t), \qquad x \leftarrow x + \tfrac{h}{2}\big[v(x, t) + v(\tilde{x}, t{+}h)\big]. \tag{47}$$

Averaging the slopes at both ends cancels the $\ddot{x}$ term; the workhorse of EDM-style samplers.

RK4 ($p = 4$, four NFE per step): the classical four-stage recipe; we used it to manufacture every "exact" trajectory in these notes' figures.

Now the budget arithmetic that the order notation hides. With n NFE total, a method of order p costing s NFE per step takes n/s steps of size $h = s/n$, for a global error $\sim C_p\,(s/n)^p$: doubling the order is worth far more than halving the constant — *eventually*. The constant C_p is built from p-th derivatives of v along the trajectory, and at very low budgets the asymptotic regime simply has not begun: Figure 9(a) shows Heun and RK4 *losing* to Euler at one and two steps — their internal stages extrapolate deep into regions where the field punishes them — then crossing over and winning by orders of magnitude as n grows, each with its promised slope. The practical summary: at generous budgets use a high-order method; at the brutal end (NFE $\sim$ 1–8), no member of the catalogue helps, and you need to change the *trajectories* (reflow, Section 7) or the *task* (distillation), not the solver.

Exercise 9. *Verify* (46)*, and show that the local Euler error* $\frac{1}{2}h^2\ddot{x} + O(h^3)$ *vanishes for all* h *exactly when the trajectory is affine in* t*. Conclude that one Euler step from* $t = 1$ *to* $t = 0$ *is exact for the reflowed field of Figure 8(b).*

8.2 Exploit the structure: DDIM is Euler in disguise

A generic solver treats v_θ as a black box. But the probability-flow ODE is *semi-linear*,

$$\dot{x} = f_t\, x - \tfrac{g_t^2}{2}\, s_\theta(x, t), \tag{48}$$

a known, exactly-solvable linear part plus a learned remainder — and numerical analysis has standing advice for this shape: solve the linear part analytically and spend your budget only on the remainder ("exponential integrators"). Here the whole trick is a two-line change of variables. Let

$$y = \frac{x}{\alpha_t} \qquad \text{and} \qquad \rho_t = \frac{\sigma_t}{\alpha_t} \quad \text{(the noise-to-signal clock)}. \tag{49}$$

Using $s_\theta = -\hat{\varepsilon}_\theta/\sigma_t$, the drift reads $\dot{x} = \frac{\dot{\alpha}_t}{\alpha_t}\, x + \frac{g_t^2}{2\sigma_t}\, \hat{\varepsilon}_\theta$, and differentiating y the linear parts cancel exactly:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \frac{\dot{x}}{\alpha_t} - \frac{x\,\dot{\alpha}_t}{\alpha_t^2} = \underbrace{\Big(\frac{\dot{\alpha}_t}{\alpha_t^2} - \frac{\dot{\alpha}_t}{\alpha_t^2}\Big)x}_{=\,0} + \frac{g_t^2}{2\alpha_t\sigma_t}\,\hat{\varepsilon}_\theta, \tag{50}$$

while the new clock, using (32) for g_t^2, runs at the very same rate:

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} = \frac{\dot{\sigma}_t\alpha_t - \sigma_t\dot{\alpha}_t}{\alpha_t^2} = \frac{g_t^2}{2\alpha_t\sigma_t}. \tag{51}$$

The two rates are *identical* up to the network output, so in the (ρ, y) chart the ODE collapses to

$$\boxed{\frac{\mathrm{d}y}{\mathrm{d}\rho} = \hat{\varepsilon}_\theta(x,t)} \tag{52}$$

— the schedule has been integrated out; the network output is the whole velocity. Now take one Euler step of (52) from t to s and translate back through $x = \alpha y$:

$$x_s = \alpha_s\,\hat{x}_0(x_t) + \sigma_s\,\hat{\varepsilon}_\theta(x_t), \tag{53}$$

which is exactly the **DDIM** update. DDIM is not a different sampler so much as Euler performed in coordinates where the schedule's own bending has been flattened: the error that remains comes only from the variation of $\hat{\varepsilon}_\theta$ along the path — from the *data* — not from the schedule. In code, one DDIM step is just the wardrobe conversion plus (53):

```
@torch.no_grad()
def ddim_step_eps_model(model, x, t, s):
    eps = model(x, t)                              # epsilon wardrobe
    shape = (-1,) + (1,) * (x.ndim - 1)
    at, st = alpha(t).view(shape), sigma(t).view(shape)
    as_, ss = alpha(s).view(shape), sigma(s).view(shape)

    x0_hat = (x - st * eps) / at
    return as_ * x0_hat + ss * eps
```

Two corollaries worth keeping. First, for a point-mass data distribution $\hat{\varepsilon}$ is constant along each trajectory (Exercise 10), so DDIM is *exact* there — it integrates every conditional path perfectly and errs only where modes compete. Second, the final hop to $t = 0$ is $x_0 = \hat{x}_0$, a Tweedie-mean readout, where naive Euler in t would be dividing by $\sigma_t \to 0$.

One decision remains: *where to put the steps.* The error of a step is (change in ρ) $\times$ (variation of $\hat{\varepsilon}$), and Figure 1 already showed that uniform-t spacing is a wildly nonuniform way to spend ρ — under VP, almost all of the ρ-clock elapses in the last few uniform-t steps before $t = 1$. Figure 9(b) shows the consequence: DDIM with uniform-t steps stalls on a plateau (its noise-end steps stay enormous in ρ no matter how many steps you add), while the same method with steps placed uniformly in log-SNR — the standard repair — beats plain Euler by an order of magnitude at the budgets people actually use. Solvers of order 2 and 3 built on (52) in exactly these coordinates are the DPM-Solver family; same idea, more stages.

Exercise 10. *Let $p_{\mathrm{data}} = \delta_{x_0}$. Show from Tweedie (4) that $\hat{\varepsilon}(x,t) = (x - \alpha_t x_0)/\sigma_t$, and that along the probability-flow trajectory through $x_t = \alpha_t x_0 + \sigma_t\varepsilon^*$ this is the constant ε^*. Conclude DDIM is exact for point masses, with any step placement.*

8.3 Likelihoods along the flow, while we are here

Section 5 promised that the ODE viewpoint buys exact likelihoods; the machinery is now all assembled, so let us collect the payment. Along any flow, the density at your own position obeys the change-of-variables formula $\frac{\mathrm{d}}{\mathrm{d}t}\log p_t(x_t) = -\nabla\cdot v(x_t,t)$, derived as (17) in the transport toolkit. Integrate it from data to noise:

$$\log p_0(x) \;=\; \log\mathcal{N}\big(x_1;\,0,I\big) \;+\; \int_0^1 \nabla\cdot v\big(x_t,t\big)\,\mathrm{d}t, \tag{54}$$

where x_t is the probability-flow trajectory launched from $x_0 = x$ — one more ODE solve, with the divergence accumulated as an extra state variable alongside x. The only obstacle is that the exact divergence of a

d-dimensional network field costs d backward passes. **Hutchinson's estimator** removes it: for any random z with $\mathbb{E}[zz^\top] = I$,

$$\nabla\cdot v = \mathbb{E}_z\big[z^\top(\partial v/\partial x)\,z\big], \tag{55}$$

and a single z per trajectory gives an unbiased estimate at the cost of one vector–Jacobian product per solver step. This is FFJORD's trick, inherited whole; it is how diffusion models report bits-per-dim, and the result is exact up to solver tolerance and the Monte Carlo noise of the trace — both knobs you control, neither involving a variational bound. (The same trajectory, run without the divergence bookkeeping, is a deterministic invertible encoder $x \mapsto x_1$ — the latent that DDIM inversion and most image-editing pipelines use.)

8.4 The stochastic option, costed

The reverse SDE (35) is integrated by Euler–Maruyama (22) — Euler plus injected noise — and here numerical analysis bills differently. Two error notions split apart for SDEs: **strong** error (pathwise closeness to a reference solution driven by the same noise; EM has order $\frac{1}{2}$ in general, order 1 for the additive noise used here) and **weak** error (closeness in distribution; EM has order 1). Generation only needs weak — nobody checks your sample against a reference path — but weak order 1 still loses to Heun's deterministic order 2. (Setting a distributional error rate against a pathwise truncation rate is apples-to-oranges on its face, but it is the comparison that matters here: same target marginals, same NFE budget, and the thing a generative model is judged on is its distribution — for the ODE the pathwise rate simply transfers to the distributional one.) So for a fixed budget and a *perfect* score model, the SDE sampler is strictly the worse integrator. What the noise buys, as Section 5 argued structurally, is self-correction: the Langevin component (30) contracts the sampled ensemble back toward q_t, so errors — including the network's own, which no solver sees — decay rather than compound. Predictor–corrector samplers make the splitting explicit: one transport step along the ODE, a few Langevin steps to re-equilibrate, repeat. And one honest engineering note closes the account: with a learned $\hat\varepsilon_\theta$, there is a floor below which solver error is irrelevant — once discretization error drops beneath the network's approximation error, further NFE purchase precision in the wrong place. The marginal value of solver sophistication is bounded by the quality of the thing being integrated.

> **The story so far.** *Solver error is (step size)$^{\text{order}}$ with constants made of the flow's higher derivatives: Euler's error is the material acceleration* (46)*, so straight trajectories integrate exactly — the truncation-error face of reflow. Higher order pays only once steps are small enough; at NFE $\sim$ 1–8 change the trajectories, not the solver. The diffusion ODE's linear part can be removed exactly: in coordinates* $(\sigma/\alpha,\, x/\alpha)$ *the equation reads* $\mathrm{d}y/\mathrm{d}\rho = \hat\varepsilon_\theta$ *and Euler becomes DDIM, exact for point masses, with step* placement *in the ρ-clock mattering as much as step count. SDE sampling has weak order 1 — a worse integrator with a perfect model, a more forgiving one with a real model.*

9 Steering: guidance as score arithmetic

Everything so far models p_{data} unconditionally. Conditioning on a label or prompt c could be done by just feeding c to the network, and is; but the trick that makes modern systems *follow* their conditioning is an afterthought-looking piece of vector arithmetic with deep consequences.

Bayes, differentiated — the normalizer dies under ∇_x:

$$\nabla_x \log q_t(x \mid c) = \nabla_x \log q_t(x) + \nabla_x \log q_t(c \mid x). \tag{56}$$

The conditional score is the unconditional score plus a "which way makes the condition more likely" correction. *Classifier guidance* (the original form) implements the second term with an external classifier trained on noisy inputs and, crucially, scales it by a knob $w > 1$: overweighting the likelihood term, sampling (approximately) from a sharpened $q_t(x)\,q_t(c \mid x)^w$. *Classifier-free guidance* eliminates the classifier by training one network with c randomly dropped, then rearranging (56) to express the classifier term as a difference of scores the network already knows:

$$\tilde s \;=\; s(x,t \mid \varnothing) + w\,\big[s(x,t \mid c) - s(x,t \mid \varnothing)\big] \qquad (w > 1 \text{ extrapolates}). \tag{57}$$

If the network is wearing a different wardrobe, nothing conceptual changes: convert to scores, do (57), and convert back. Because the conversions in (9) are affine at fixed (x,t), this usually appears in code as the same extrapolation on the predicted quantity itself, for example

$$\tilde\varepsilon = \hat\varepsilon(x,t \mid \varnothing) + w\big[\hat\varepsilon(x,t \mid c) - \hat\varepsilon(x,t \mid \varnothing)\big], \tag{58}$$

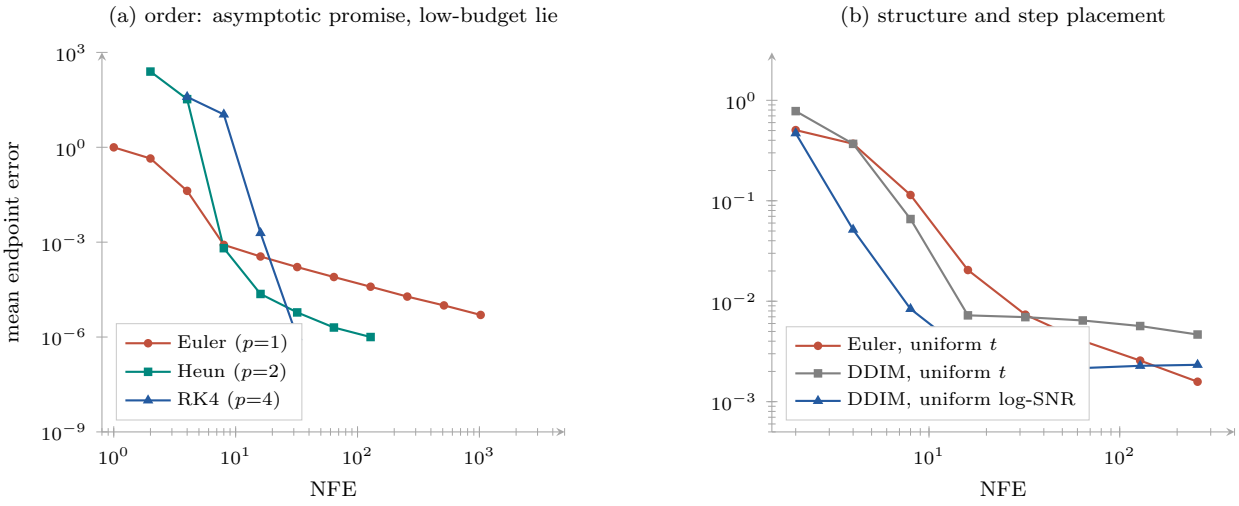

Figure 9: Solver economics on exactly known velocity fields (errors against a 4096–8192-step RK4 reference, averaged over sixteen starting quantiles). (a) Euler, Heun, RK4 on the *curved* rectified-flow ODE of Figure 8(a): at one or two steps the high-order methods are *worse* — their internal stages extrapolate into regions where the field is violent — but past the crossover each converges at its promised slope (h^1, h^2, h^4), and at 64 NFE RK4 leads Euler by five orders of magnitude. (b) The VP probability-flow ODE with sharp data modes (± 1, width 10^{-2} — the realistic regime where data detail lives far below the noise scale). Plain Euler in t converges at order 1. DDIM with uniform-t steps stalls: its noise-end steps remain enormous in the $\rho = \sigma/\alpha$ clock however many steps are taken. DDIM with uniform log-SNR placement is an order of magnitude better than Euler at NFE 4–16; its late floor is the $t_{\min} = 10^{-3}$ cutoff and final denoising hop. Coordinates *and* clock, not either alone.

and similarly for $\hat{x}_0$ or $\hat{v}$ under the corresponding parameterization. The arithmetic is simple; the modeling claim behind it is the approximate part.

Read (57) as language: "head where the conditional model wants to go, *exaggerated* by how it differs from the unconditional model." The honest fine print: applying (57) at every t does *not* exactly sample any single tilted distribution (tilting each noisy marginal is not the same as diffusing a tilted data distribution), and large w visibly trades diversity and naturalism for prompt adherence — the familiar over-saturated, over-committed look of heavily guided samples is this approximation showing. Guidance is best understood as a controlled, useful distortion of the flow, not as exact inference; that it works as well as it does is one of the field's load-bearing pieces of luck. Figure 10 shows both halves: the vector arithmetic at a point, and what the tilt does to a density.

10 The whole subject on one page

Choice	Options	Comment
Path of marginals	VP / VE / linear / cosine schedule (α_t, σ_t)	a design input, not a discovery; all give Gaussian conditionals
Learned wardrobe	$\hat{\varepsilon}$ / $\hat{x}_0$ / s / $\hat{v}$	one function, four parameterizations; convert by (9)
Training identity	DSM (8) or CFM (43)	both use Lemma 1; for Gaussian interpolants, different target dress for the same marginal field
Sampler	PF-ODE $\leftrightarrow$ reverse SDE (+ churn)	ODE = fast, reproducible; SDE = self-correcting; a dial
Likelihood	$\frac{\mathrm{d}}{\mathrm{d}t} \log p = -\nabla \cdot v$ along the ODE	diffusion inherits CNF's flow-likelihood machinery (§8.3)
Step count	solver order & step placement (§8), curvature, reflow, distillation	the cost model: pay per network call; straightness is wealth
Steering	guidance arithmetic (57)	approximate tilting; the w knob trades diversity for adherence

Two sentences to carry away. *A generative model is a path of distributions from data to noise plus a*

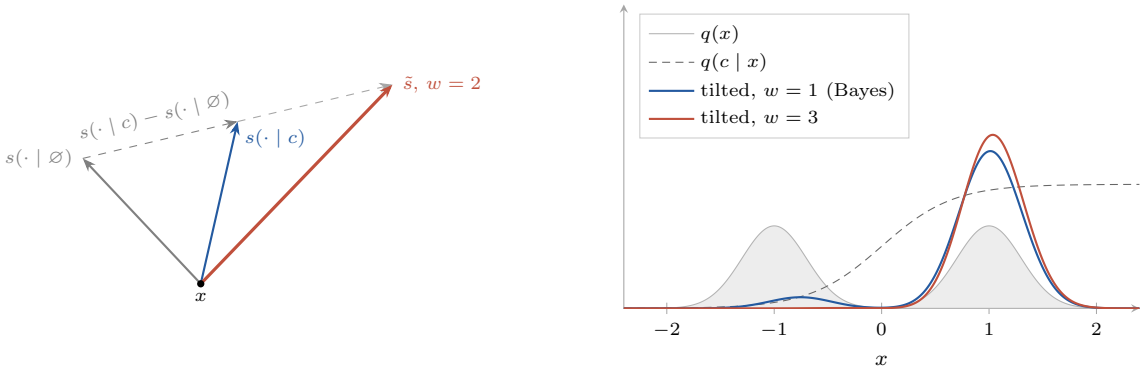

Figure 10: Guidance as score arithmetic. Left: equation (57) at a single point — the guided score $\tilde{s}$ continues past the conditional score along the line from unconditional to conditional; $w = 1$ would stop at $s(\cdot \mid c)$, $w > 1$ extrapolates. Right: what the corresponding tilt $q(x)\, q(c \mid x)^w$ (normalized) does to a bimodal density with a soft classifier for the right-hand class. At $w = 1$ this is exact Bayesian conditioning: the wrong mode is suppressed but survives where the classifier is unsure. At $w = 3$ the surviving mode sharpens and shifts *into* the classifier-confident region — prompt adherence up, diversity down, and the mode is no longer quite where the data was. Now recall the fine print: the sampler applies this tilt at *every* noise level, which is not the same as diffusing the tilted data distribution.

velocity that transports it; both the path and the stochasticity of its traversal are design choices. The path's conditional structure hands you closed-form per-sample regression targets, and one lemma — conditional expectation minimizes L^2 — guarantees that regressing on them learns the true score/velocity: that lemma, used twice, is the entire trainability of the field.

Operationally, almost every training loop in these notes is the same five-line program:

1. sample $x_0 \sim p_{\text{data}}$, $\varepsilon \sim \mathcal{N}(0, I)$, and t;
2. form $x_t = \alpha_t x_0 + \sigma_t \varepsilon$;
3. choose the wardrobe: predict ε, s, x_0, or v;
4. regress on the corresponding closed-form target with MSE;
5. at sampling time, draw x_1 from the noise prior and integrate the learned ODE or SDE backwards to $t = 0$.

Problems

1. Prove Tweedie (4) as outlined in Exercise 1 (if you have not already), then derive the $\mathbb{E}[\varepsilon \mid x_t]$ version and confirm the wardrobe conversions (9) are mutually consistent.
2. Show that (42) satisfies the continuity equation (15) for q_t: write $q_t(x) = \mathbb{E}_{x_0,\varepsilon}\, \delta(x - \alpha_t x_0 - \sigma_t \varepsilon)$, differentiate in t, and integrate by parts. (Read the δ as the $w \to 0$ limit of $\mathcal{N}(0, w^2 I)$ densities; or sidestep it by integrating everything against a smooth test function first, in which case you are running the Exercise-after-(29) argument in reverse.)
3. Derive the DDPM forward posterior $q(x_s \mid x_t, x_0)$ for $s < t$ (Gaussian; find its mean and variance), and verify that the KL terms of (40) reduce to weighted ε-prediction MSEs.
4. Anderson by hand. One Euler–Maruyama step of the forward SDE (32) is a Gaussian kernel $q(x_{t+h} \mid x_t)$. Apply Bayes, $q(x_t \mid x_{t+h}) \propto q(x_{t+h} \mid x_t)\, q_t(x_t)$, expand $\log q_t(x_t) \approx \log q_t(x_{t+h}) + (x_t - x_{t+h}) \cdot \nabla \log q_t(x_{t+h})$, and complete the square. Show the reverse kernel is Gaussian with mean $x_{t+h} - h\big[f_t\, x_{t+h} - g_t^2 \nabla \log q_t(x_{t+h})\big] + O(h^2)$ and covariance $g_t^2 h\, I$ — the reverse SDE (35), derived in four lines with no stochastic analysis. Where did the factor of *two* relative to the probability-flow correction come from?
5. Gaussian sandbox, continued: verify the closed forms (10)–(12) asserted in Section 2 (complete the square; check the velocity lands in both wardrobes). Show that with the linear schedule the marginal ODE trajectories are straight only if the data is a single Gaussian and the coupling is the model's own — then exhibit a two-component mixture whose marginal trajectories visibly curve.

6. Gaussian sandbox, part II (solvers). For the single-Gaussian data of the previous problem the probability-flow ODE is linear with a known solution. Implement Euler, Heun, and DDIM (53) with both uniform-t and uniform log-SNR grids; measure endpoint error against the exact flow and reproduce the qualitative content of Figure 9. Verify numerically that DDIM becomes exact as $s \to 0$ (Exercise 10). Then add the divergence as an extra integrated state, as in (54), and check the recovered $\log p_0$ against the closed-form Gaussian log-density — first with the exact divergence, then with Hutchinson's estimator.

7. Show that one CFG step (57) corresponds to the score of the unnormalized density $q_t(x)\left[q_t(x \mid c)/q_t(x)\right]^w$, and explain in two sentences why doing this at every t does not sample the $t = 0$ tilted distribution exactly.

8. (Open-ended.) Implement CFM with the linear schedule on two-moons in 2D ($\sim$60 lines — the snippets in Sections 2.2 and 5 are most of them). Plot the learned velocity field at several t, sample with 1, 4, and 50 Euler steps, then do one round of reflow and repeat. Watch the trajectories straighten. Finally, implement the *learning-free* competitor: the closed-form empirical score of the mixture (2) over your training points, plugged into the same ODE. Watch it return the training set, then reread the "what training buys" paragraph of Section 2.2.

References

[1] J. Sohl-Dickstein et al., "Deep unsupervised learning using nonequilibrium thermodynamics," ICML 2015. (The origin; reads remarkably well in hindsight.)

[2] J. Ho, A. Jain, P. Abbeel, "Denoising diffusion probabilistic models," NeurIPS 2020.

[3] Y. Song et al., "Score-based generative modeling through stochastic differential equations," ICLR 2021. (The unifying SDE/ODE paper; source of Section 5.)

[4] P. Vincent, "A connection between score matching and denoising autoencoders," Neural Computation 2011. (Lemma 1, instance 1.)

[5] Y. Lipman et al., "Flow matching for generative modeling," ICLR 2023. (Lemma 1, instance 2.)

[6] X. Liu, C. Gong, Q. Liu, "Flow straight and fast: learning to generate and transfer data with rectified flow," ICLR 2023.

[7] M. Albergo, N. Boffi, E. Vanden-Eijnden, "Stochastic interpolants," 2023. (The general framework containing all the schedules.)

[8] T. Karras et al., "Elucidating the design space of diffusion-based generative models," NeurIPS 2022. (The engineering companion: schedules, parameterizations, samplers, ablated.)

[9] B. D. O. Anderson, "Reverse-time diffusion equation models," Stochastic Processes and their Applications, 1982.

[10] J. Song, C. Meng, S. Ermon, "Denoising diffusion implicit models," ICLR 2021. (The update (53), derived there very differently.)

[11] C. Lu et al., "DPM-Solver: a fast ODE solver for diffusion probabilistic model sampling in around 10 steps," NeurIPS 2022. (Higher-order solvers in the coordinates of Section 8.2.)

[12] W. Grathwohl et al., "FFJORD: free-form continuous dynamics for scalable reversible generative models," ICLR 2019. (Hutchinson-traced likelihoods, Section 8.3.)

[13] J.-D. Benamou, Y. Brenier, "A computational fluid mechanics solution to the Monge–Kantorovich mass transfer problem," Numerische Mathematik, 2000. (The kinetic-energy formulation (45) of optimal transport.)

[14] S. Särkkä, A. Solin, "Applied stochastic differential equations," Cambridge University Press, 2019. (Everything in Section 4, at textbook pace and with the analysis done honestly.)

[15] E. Hairer, S. P. Nørsett, G. Wanner, "Solving ordinary differential equations I: nonstiff problems," Springer, 1993. (The solver's-eye view, fully grown.)