Solving Inverse Problems via Diffusion Optimal Control

Authors: Henry Li & Marcus Pereira

Speaker: Ali Vaziri

November 21, 2025

What is this paper about?

- Inverse problems: recover an unknown signal x_0 given measurements

$$y = A(x_0) + \eta,$$

  e.g. super-resolution, inpainting, deblurring.
- Recent trend: use diffusion models as powerful priors for $p(x_0)$.
- Classical approach: **posterior sampling** from $p(x_0 \mid y)$ using score-based diffusion.
- This paper: **reframe** diffusion inverse problem solving as an *optimal control* problem on the reverse diffusion dynamics.

Key limitations of posterior sampling methods

Consider conditional sampling SDE

$$\mathrm{d}x_t = \Big(f(x_t, t) - \tfrac{1}{2} g(t)^2 \nabla_x \log p_t(x_t \mid y)\Big)\mathrm{d}t + g(t)\mathrm{d}w_t.$$

Main obstacles:

- Need $\nabla_x \log p_t(y \mid x_t)$, which is *intractable* for $t > 0$.
- Approximations rely on Tweedie-type formulas and $\hat{x}_0(x_t)$, degrading in the noisy regime.
- Methods are sensitive to time discretization T and heavily depend on score network accuracy.

Limitation: Tweedie-type likelihood approximations

The forward diffusion can be written as

$$x_t = \sqrt{\bar{\alpha}(t)}\, x_0 + \sqrt{1 - \bar{\alpha}(t)}\, z, \quad z \sim \mathcal{N}(0, I).$$

The posterior $p(x_0 \mid x_t)$ has mean (Tweedie / denoising result)

$$\hat{x}_0(x_t) := \mathbb{E}[x_0 \mid x_t] = \frac{1}{\sqrt{\bar{\alpha}(t)}}\Big(x_t + (1 - \bar{\alpha}(t))\nabla_{x_t} \log p_t(x_t)\Big),$$

and in practice

$$\hat{x}_0(x_t) \approx \frac{1}{\sqrt{\bar{\alpha}(t)}}\Big(x_t + (1 - \bar{\alpha}(t))\, s_\theta(x_t, t)\Big).$$

For the likelihood term, diffusion process sampling (DPS) factorizes

$$p(y \mid x_t) = \int p(y \mid x_0)\, p(x_0 \mid x_t)\, \mathrm{d}x_0 = \mathbb{E}_{x_0 \sim p(x_0 \mid x_t)}[p(y \mid x_0)]$$

and then approximates

$$p(y \mid x_t) \;\approx\; p(y \mid \hat{x}_0(x_t)).$$

DPS Algorithm limitation

- In regimes with high measurement noise or multimodal $p(x_0 \mid x_t)$, the gap can be large; the gradient $\nabla_{x_t} \log p(y \mid x_t) \approx \nabla_{x_t} \log p(y \mid \hat{x}_0(x_t))$ can be biased and unstable.
- Computing this gradient requires backpropagating through $\hat{x}_0(x_t)$ and the score network s_θ, which is expensive and error-prone, especially at high noise levels (large t).

$x_N \sim \mathcal{N}(0, I)$

for $i = N-1$ down to 0 **do**

$\quad \hat{s} \leftarrow s_\theta(x_i, i)$

$\quad \hat{x}_0 \leftarrow \dfrac{1}{\sqrt{\bar{\alpha}_i}}\big(x_i + (1-\bar{\alpha}_i)\hat{s}\big)$

$\quad z \sim \mathcal{N}(0, I)$

$\quad x'_{i-1} \leftarrow \sqrt{\dfrac{\alpha_i(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}}\, x_i + \sqrt{\dfrac{\bar{\alpha}_{i-1}\beta_i}{1-\bar{\alpha}_i}}\, \hat{x}_0 + \tilde{\sigma}_i z$

$\quad x_{i-1} \leftarrow x'_{i-1} - \zeta_i \nabla_{x_i} \big\| y - A(\hat{x}_0) \big\|_2^2$

end for

return $\hat{x}_0$

High-level idea of the paper

Core idea

Treat the discretized reverse diffusion sampler as a **nonlinear dynamical system** and solve an *optimal control* problem over the control sequence $\{u_t\}$.

- Use iLQR updates on the diffusion dynamics.
- Terminal cost encodes data fidelity: $\ell_0(x_0) = -\log p(y \mid x_0)$.
- Running cost regularizes control: $\ell_t(x_t, u_t) = \alpha \|u_t\|^2$.
- Show that:
  - Posterior sampling appears as a *special case*.
  - The ideal conditional score emerges as the Jacobian of a value function.

Outline

Reverse-time Diffusion SDE and Probability-Flow ODE

Reverse-time Itô SDE used in diffusion models:

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

where

- $\mathbf{x}_t \in \mathbb{R}^n$ is the state,
- $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is drift,
- $g(t)$ is scalar diffusion,
- $\mathrm{d}\mathbf{w}_t$ is standard Brownian motion,
- $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ is the score of the marginal $p_t(\mathbf{x}_t)$.

Associated probability-flow ODE:

$$\mathrm{d}\mathbf{x}_t = \left[\mathbf{f}(\mathbf{x}_t) - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\right] \mathrm{d}t,$$

which has the same marginals $p_t(\mathbf{x}_t)$ as the SDE.

Euler Discretization of the Probability-Flow ODE

Euler discretization in reverse time:

$$\mathbf{x}_{t-1} = \mathbf{x}_t - \Big[\mathbf{f}(\mathbf{x}_t) - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\Big] \Delta t.$$

We can write this abstractly as a discrete-time dynamical system

$$\mathbf{x}_{t-1} = \mathbf{h}(\mathbf{x}_t), \qquad \mathbf{h} : \mathbb{R}^n \to \mathbb{R}^n,$$

where $\mathbf{h}$ encodes one reverse diffusion step. Later we will treat this as an **uncontrolled** dynamical system and then inject controls $\mathbf{u}_t$.

Goal: Backward Conditional SDE for Inverse Problems

We start from the **marginal** reverse-time SDE (unconditional diffusion):

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

and the inverse problem

$$\mathbf{y} = \mathbf{A}\mathbf{x}_0 + \eta, \qquad \eta \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_d).$$

Goal: Derive the reverse-time SDE for the *posterior* process

$$\mathrm{d}\mathbf{x}_t = \left[\mathbf{f}(\mathbf{x}_t) - g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t \mid \mathbf{y})\right] \mathrm{d}t + g(t)\, \mathrm{d}\mathbf{w}_t,$$

directly from:

1. The marginal reverse SDE of $\mathbf{x}_t$.
2. The static observation model $\mathbf{y} = \mathbf{A}\mathbf{x}_0 + \eta$.

Step 1: Joint Law $p_t(\mathbf{x}, \mathbf{y})$

Consider a **joint random pair** $(\mathbf{x}_t, \mathbf{y})$ constructed as:

- Forward diffusion: $\mathbf{x}_t$ evolves in time via the forward SDE

$$\mathrm{d}\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}_t, \qquad t \in [0, T].$$

- Observation (time-independent):

$$\mathbf{y} = \mathbf{A}\mathbf{x}_0 + \eta, \qquad \eta \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_d),$$

  drawn *only at* $t = 0$ and then held fixed.

This defines a joint density at each time t:

$$p_t(\mathbf{x}, \mathbf{y}) = \int p(\mathbf{y} \mid \mathbf{x}_0)\, p(\mathbf{x}_t = \mathbf{x} \mid \mathbf{x}_0)\, p_0(\mathbf{x}_0)\, \mathrm{d}\mathbf{x}_0.$$

Equivalently, we can view $(\mathbf{x}_t, \mathbf{y})$ as a Markov process where

- $\mathbf{x}_t$ diffuses in time,
- $\mathbf{y}$ is a static random variable coupled to $\mathbf{x}_0$.

Step 2: Joint Forward Process $(\mathbf{x}_t, \mathbf{y}_t)$

Define a joint process $(\mathbf{x}_t, \mathbf{y}_t)$:

- $\mathbf{x}_t$ follows the forward SDE (in forward time s, written as t for simplicity):

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

- $\mathbf{y}_t$ is **constant in time**:

$$\mathrm{d}\mathbf{y}_t = 0.$$

Thus the joint dynamics are

$$\begin{cases} \mathrm{d}\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}_t, \\ \mathrm{d}\mathbf{y}_t = 0, \end{cases}$$

and the joint density of $(\mathbf{x}_t, \mathbf{y}_t)$ at time t is exactly

$$p_t(\mathbf{x}, \mathbf{y}).$$

Step 3: Reverse-Time SDE for the Joint Process

Let

$$\mathbf{z}_t := (\mathbf{x}_t, \mathbf{y}_t) \in \mathbb{R}^{n+d}.$$

The joint forward SDE can be written as

$$\mathrm{d}\mathbf{z}_t = a(\mathbf{z}_t, t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}\mathbf{w}_t,$$

with

$$a(\mathbf{z}, t) = \begin{pmatrix} \mathbf{f}(\mathbf{x}, t) \\ 0 \end{pmatrix}, \qquad \sigma(t) = \begin{pmatrix} g(t)\, I_n \\ 0 \end{pmatrix}, \quad \mathbf{z} = (\mathbf{x}, \mathbf{y}).$$

Hence

$$\sigma(t)\sigma(t)^\top = \begin{pmatrix} g(t)^2 I_n & 0 \\ 0 & 0 \end{pmatrix}.$$

The time-reversal theorem (with the convention of Eq. (1)) gives the reverse drift

$$a_{\mathrm{rev}}(\mathbf{z}, t) = a(\mathbf{z}, t) - \tfrac{1}{2}\sigma(t)\sigma(t)^\top \nabla_{\mathbf{z}} \log p_t(\mathbf{z}),$$

where $p_t(\mathbf{z}) = p_t(\mathbf{x}, \mathbf{y})$.

Step 3: Reverse-Time SDE for the Joint Process (cont.)

Write

$$\nabla_{\mathbf{z}} \log p_t(\mathbf{z}) = \begin{pmatrix} \nabla_{\mathbf{x}} \log p_t(\mathbf{x}, \mathbf{y}) \\ \nabla_{\mathbf{y}} \log p_t(\mathbf{x}, \mathbf{y}) \end{pmatrix}.$$

Then

$$\sigma(t)\sigma(t)^\top \nabla_{\mathbf{z}} \log p_t(\mathbf{z}) = \begin{pmatrix} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}, \mathbf{y}) \\ 0 \end{pmatrix}.$$

Thus the joint reverse drift is

$$\begin{aligned} a_{\text{rev}}(\mathbf{z}, t) &= \begin{pmatrix} \mathbf{f}(\mathbf{x}, t) \\ 0 \end{pmatrix} - \tfrac{1}{2} \begin{pmatrix} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}, \mathbf{y}) \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{f}(\mathbf{x}, t) - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}, \mathbf{y}) \\ 0 \end{pmatrix}. \end{aligned}$$

Therefore, the reverse-time SDE for the joint process $(\mathbf{x}_t, \mathbf{y}_t)$ is

$$\begin{cases} \mathrm{d}\mathbf{x}_t = \left[\mathbf{f}(\mathbf{x}_t, t) - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t, \mathbf{y}_t)\right] \mathrm{d}t + g(t)\, \mathrm{d}\widetilde{\mathbf{w}}_t, \\ \mathrm{d}\mathbf{y}_t = 0. \end{cases}$$

Step 4: From Joint to Conditional SDE

Fix an observed value $\mathbf{y}$ and consider the conditional law

$$p_t(\mathbf{x} \mid \mathbf{y}) = \frac{p_t(\mathbf{x}, \mathbf{y})}{p_t(\mathbf{y})}.$$

When taking gradients w.r.t. $\mathbf{x}$, any factor depending only on $\mathbf{y}$ is constant:

$$\begin{aligned} \nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid \mathbf{y}) &= \nabla_{\mathbf{x}} \big[\log p_t(\mathbf{x}, \mathbf{y}) - \log p_t(\mathbf{y}) \big] \\ &= \nabla_{\mathbf{x}} \log p_t(\mathbf{x}, \mathbf{y}), \end{aligned}$$

because $p_t(\mathbf{y})$ is independent of $\mathbf{x}$.
Hence

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}, \mathbf{y}) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid \mathbf{y}).$$

Substitute this into the reverse drift of $\mathbf{x}_t$:

$$\mathrm{d}\mathbf{x}_t = \Big[\mathbf{f}(\mathbf{x}_t, t) - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t \mid \mathbf{y}_t)\Big] \mathrm{d}t + g(t)\, \mathrm{d}\widetilde{\mathbf{w}}_t.$$

Conditioning on the event $\mathbf{y}_t = \mathbf{y}$ gives

$$\mathrm{d}\mathbf{x}_t = \Big[\mathbf{f}(\mathbf{x}_t, t) - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t \mid \mathbf{y})\Big] \mathrm{d}t + g(t)\, \mathrm{d}\widetilde{\mathbf{w}}_t,$$

which is exactly the **backward conditional SDE** (Eq. (6)).

Step 5: Bayes Decomposition of the Conditional Score

From Bayes' rule at time t,

$$p_t(\mathbf{x}_t \mid \mathbf{y}) = \frac{p_t(\mathbf{x}_t)\, p_t(\mathbf{y} \mid \mathbf{x}_t)}{p_t(\mathbf{y})}.$$

Taking logs,

$$\log p_t(\mathbf{x}_t \mid \mathbf{y}) = \log p_t(\mathbf{x}_t) + \log p_t(\mathbf{y} \mid \mathbf{x}_t) - \log p_t(\mathbf{y}).$$

Differentiate w.r.t. $\mathbf{x}_t$:

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t \mid \mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p_t(\mathbf{y} \mid \mathbf{x}_t),$$

since $\log p_t(\mathbf{y})$ does not depend on $\mathbf{x}_t$. This is Eq. (7). Plugging this into the conditional SDE (Eq. (6)) yields

$$\mathrm{d}\mathbf{x}_t = \Big[\mathbf{f}(\mathbf{x}_t, t) - \tfrac{1}{2} g(t)^2 \big(\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p_t(\mathbf{y} \mid \mathbf{x}_t)\big)\Big]\mathrm{d}t + g(t)\, \mathrm{d}\widetilde{\mathbf{w}}_t.$$

Associated ODE:

$$\mathrm{d}\mathbf{x} = \big[\mathbf{f}(\mathbf{x}) - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid \mathbf{y})\big]\mathrm{d}t.$$

Euler discretization:

$$\mathbf{x}_{t-1} = \mathbf{x}_t + \Big[\mathbf{f}(\mathbf{x}_t) - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t \mid \mathbf{y})\Big]\Delta t.$$

Finite-Horizon Cost and Value Function

Consider a discrete-time system

$$\mathbf{x}_{t-1} = \mathbf{h}(\mathbf{x}_t, \mathbf{u}_t), \quad t = T, \ldots, 1.$$

Define total cost

$$J_T = \sum_{t=T}^{1} \ell_t(\mathbf{x}_t, \mathbf{u}_t) + \ell_0(\mathbf{x}_0),$$

where ℓ_t is running cost and ℓ_0 is terminal cost. Value function:

$$V(\mathbf{x}_t, t) := \min_{\{\mathbf{u}_n\}_{n=t}^{1}} J_t,$$

i.e., optimal cost-to-go starting from state $\mathbf{x}_t$ at time t. Bellman recursion:

$$V(\mathbf{x}_t, t) = \min_{\mathbf{u}_t} \left[\ell_t(\mathbf{x}_t, \mathbf{u}_t) + V(\mathbf{x}_{t-1}, t-1) \right].$$

State-Action Value Function Q

Define the **state-action value**:

$$Q(\mathbf{x}_t, \mathbf{u}_t) := \ell_t(\mathbf{x}_t, \mathbf{u}_t) + V(\mathbf{x}_{t-1}, t-1),$$

and

$$V(\mathbf{x}_t, t) = \min_{\mathbf{u}_t} Q(\mathbf{x}_t, \mathbf{u}_t).$$

iLQR approximates Q and V locally by quadratic expansions around a nominal trajectory

$$\{\bar{\mathbf{x}}_t\}_{t=0}^{T}, \qquad \{\bar{\mathbf{u}}_t\}_{t=1}^{T},$$

and updates the controls via a local Linear-Quadratic approximation.

Derivatives of Q

Let $\mathbf{h}_{\mathbf{x}}$ and $\mathbf{h}_{\mathbf{u}}$ denote Jacobians of $\mathbf{h}(\mathbf{x}_t, \mathbf{u}_t)$. Local quadratic approximation yields

$$
\begin{aligned}
Q_{\mathbf{x}} &= \ell_{\mathbf{x}} + \mathbf{h}_{\mathbf{x}}^{\top} V'_{\mathbf{x}}, \\
Q_{\mathbf{u}} &= \ell_{\mathbf{u}} + \mathbf{h}_{\mathbf{u}}^{\top} V'_{\mathbf{x}}, \\
Q_{\mathbf{xx}} &= \ell_{\mathbf{xx}} + \mathbf{h}_{\mathbf{x}}^{\top} V'_{\mathbf{xx}} \mathbf{h}_{\mathbf{x}}, \\
Q_{\mathbf{ux}} &= \ell_{\mathbf{ux}} + \mathbf{h}_{\mathbf{u}}^{\top} V'_{\mathbf{xx}} \mathbf{h}_{\mathbf{x}}, \\
Q_{\mathbf{xu}} &= \ell_{\mathbf{xu}} + \mathbf{h}_{\mathbf{x}}^{\top} V'_{\mathbf{xx}} \mathbf{h}_{\mathbf{u}}, \\
Q_{\mathbf{uu}} &= \ell_{\mathbf{uu}} + \mathbf{h}_{\mathbf{u}}^{\top} V'_{\mathbf{xx}} \mathbf{h}_{\mathbf{u}},
\end{aligned}
$$

where primes indicate evaluation at time $t - 1$.

Deriving Q_x, Q_u (First Derivatives)

Recall the definition of the state–action value:

$$Q(\mathbf{x}_t, \mathbf{u}_t) = \ell_t(\mathbf{x}_t, \mathbf{u}_t) + V(\mathbf{x}_{t-1}), \qquad \mathbf{x}_{t-1} = \mathbf{h}(\mathbf{x}_t, \mathbf{u}_t).$$

So Q is a scalar function of $(\mathbf{x}_t, \mathbf{u}_t)$ built from:

- current running cost $\ell_t(\mathbf{x}_t, \mathbf{u}_t)$,
- future cost $V(\mathbf{x}_{t-1}) = V(\mathbf{h}(\mathbf{x}_t, \mathbf{u}_t))$.

Use the chain rule for the composite $V(\mathbf{h}(\mathbf{x}_t, \mathbf{u}_t))$. Since V is scalar,

$$V'_x := \frac{\partial V}{\partial \mathbf{x}_{t-1}} \in \mathbb{R}^n, \qquad \mathbf{h}_x := \frac{\partial \mathbf{h}}{\partial \mathbf{x}_t} \in \mathbb{R}^{n \times n},$$

and

$$\frac{\partial}{\partial \mathbf{x}_t} V(\mathbf{h}(\mathbf{x}_t, \mathbf{u}_t)) = \mathbf{h}_x^\top V'_x.$$

Therefore,

$$Q_x = \ell_x + \mathbf{h}_x^\top V'_x, \qquad Q_u = \ell_u + \mathbf{h}_u^\top V'_x.$$

Deriving Q_{xx}, Q_{ux}, Q_{uu} (Second Derivatives)

From the previous slide:

$$Q_x = \ell_x + \mathbf{h}_x^\top V'_x, \qquad Q_u = \ell_u + \mathbf{h}_u^\top V'_x.$$

- Linearize the dynamics $\mathbf{h}(\mathbf{x}, \mathbf{u})$ around the nominal trajectory using first order derivatives: $\mathbf{h}_{xx} \approx 0$, $\mathbf{h}_{xu} \approx 0$, and $\mathbf{h}_{uu} \approx 0$.

Deriving Q_{xx} (other terms also similar):

$$Q_x = \ell_x + \mathbf{h}_x^\top V'_x, \qquad Q_{xx} = \ell_{xx} + \frac{\partial}{\partial \mathbf{x}}\left(\mathbf{h}_x^\top V'_x\right).$$

$$\frac{\partial}{\partial \mathbf{x}}\left(\mathbf{h}_x^\top V'_x\right) = \underbrace{\left(\frac{\partial \mathbf{h}_x}{\partial \mathbf{x}}\right)^\top V'_x}_{\text{depends on } \mathbf{h}_{xx}} + \underbrace{\mathbf{h}_x^\top \frac{\partial V'_x}{\partial \mathbf{x}}}_{\text{via } \mathbf{x}'}.$$

$$\left(\tfrac{\partial \mathbf{h}_x}{\partial \mathbf{x}}\right)^\top V'_x \approx 0 \quad \Rightarrow \quad \frac{\partial}{\partial \mathbf{x}}\left(\mathbf{h}_x^\top V'_x\right) \approx \mathbf{h}_x^\top \frac{\partial V'_x}{\partial \mathbf{x}}.$$

Now apply the chain rule with $\mathbf{x}' = \mathbf{h}(\mathbf{x}, \mathbf{u})$:

$$\frac{\partial V'_x}{\partial \mathbf{x}} = \frac{\partial V'_x}{\partial \mathbf{x}'} \frac{\partial \mathbf{x}'}{\partial \mathbf{x}} = V'_{xx}\, \mathbf{h}_x.$$

Therefore $Q_{xx} \approx \ell_{xx} + \mathbf{h}_x^\top V'_{xx} \mathbf{h}_x$.

Minimizing Q: optimal control update

We now consider Q as a quadratic function of $(\delta\mathbf{x}, \delta\mathbf{u})$:

$$Q(\delta\mathbf{x}, \delta\mathbf{u}) = Q_0 + Q_{\mathbf{x}}^\top \delta\mathbf{x} + Q_{\mathbf{u}}^\top \delta\mathbf{u} + \tfrac{1}{2}\delta\mathbf{x}^\top Q_{\mathbf{xx}} \delta\mathbf{x} + \tfrac{1}{2}\delta\mathbf{u}^\top Q_{\mathbf{uu}} \delta\mathbf{u} + \delta\mathbf{u}^\top Q_{\mathbf{ux}}\, \delta\mathbf{x}.$$

To find the optimal variation $\delta\mathbf{u}^*$, take the derivative w.r.t. $\delta\mathbf{u}$:

$$\frac{\partial Q}{\partial \delta\mathbf{u}} = Q_{\mathbf{u}} + Q_{\mathbf{ux}}\, \delta\mathbf{x} + Q_{\mathbf{uu}}\, \delta\mathbf{u}.$$

Set this gradient to zero for optimality:

$$Q_{\mathbf{uu}}\, \delta\mathbf{u}^* + Q_{\mathbf{ux}}\, \delta\mathbf{x} + Q_{\mathbf{u}} = 0.$$

Solve for $\delta\mathbf{u}^*$:

$$\delta\mathbf{u}^* = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}} - Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}\, \delta\mathbf{x}.$$

This yields the usual iLQR form

$$\delta\mathbf{u}^* = \mathbf{k} + \mathbf{K}\, \delta\mathbf{x},$$

with

$$\mathbf{k} = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}}, \qquad \mathbf{K} = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}.$$

Deriving $V_{\mathbf{x}}$ and $V_{\mathbf{xx}}$ from Q

At the current time t the value function is

$$V(\mathbf{x}_t, t) = \min_{\delta \mathbf{u}} Q(\mathbf{x}_t, \mathbf{u}_t) = Q(\delta \mathbf{x}, \delta \mathbf{u}^*).$$

Plug $\delta \mathbf{u}^* = \mathbf{k} + \mathbf{K}\,\delta \mathbf{x}$ into the quadratic form of Q and collect terms in $\delta \mathbf{x}$:

$$V(\mathbf{x}_t, t) \approx V_0 + V_{\mathbf{x}}^\top \delta \mathbf{x} + \tfrac{1}{2}\,\delta \mathbf{x}^\top V_{\mathbf{xx}}\,\delta \mathbf{x}.$$

Matching coefficients of $\delta \mathbf{x}$ and $\delta \mathbf{x} \delta \mathbf{x}^\top$ gives

$$V_{\mathbf{x}} = Q_{\mathbf{x}} - \mathbf{K}^\top Q_{\mathbf{uu}} \mathbf{k},$$

$$V_{\mathbf{xx}} = Q_{\mathbf{xx}} - \mathbf{K}^\top Q_{\mathbf{uu}} \mathbf{K}.$$

Equivalently, using $\mathbf{k} = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}}, \quad \mathbf{K} = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}$, one can also write

$$V_{\mathbf{x}} = Q_{\mathbf{x}} - Q_{\mathbf{xu}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}},$$
$$V_{\mathbf{xx}} = Q_{\mathbf{xx}} - Q_{\mathbf{xu}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}.$$

Controlling the Reverse Diffusion Dynamics

Uncontrolled reverse diffusion (Euler) step:

$$\mathbf{x}_{t-1} = \mathbf{x}_t - \left[\mathbf{f}(\mathbf{x}_t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\right] \Delta t.$$

We treat this as a dynamical system and inject a control $\mathbf{u}_t$:

- **Input perturbation:**

$$\mathbf{x}_{t-1} = (\mathbf{x}_t + \mathbf{u}_t) - \left[\mathbf{f}(\mathbf{x}_t + \mathbf{u}_t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t + \mathbf{u}_t)\right] \Delta t.$$

- **Output perturbation:**

$$\mathbf{x}_{t-1} = \mathbf{x}_t - \left[\mathbf{f}(\mathbf{x}_t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)\right] \Delta t + \mathbf{u}_t.$$

In both cases, we can write $\mathbf{x}_{t-1} = \mathbf{h}(\mathbf{x}_t, \mathbf{u}_t)$ and apply iLQR.

Diffusion Optimal Control Algorithm

Algorithm: Diffusion Optimal Control

Input: step size λ, horizon T, measurement $\mathbf{y}$, initial state $\mathbf{x}_T$
Initialize:
    $\mathbf{u}_t \leftarrow \mathbf{0},\ \mathbf{k}_t \leftarrow \mathbf{0},\ \mathbf{K}_t \leftarrow \mathbf{0}$ for $t = 1, \ldots, T$
for `iter` $= 1$ **to** `num_iters` **do**
    $V_{\mathbf{x}}, V_{\mathbf{xx}} \leftarrow \nabla_{\mathbf{x}_0} \log p(\mathbf{y} \mid \mathbf{x}_0),\ \nabla^2_{\mathbf{x}_0} \log p(\mathbf{y} \mid \mathbf{x}_0)$ ▷ Initialize $V(\mathbf{x}_t, t)$ at $t = 0$
    for $t = 1$ **to** T **do**
        Compute $\mathbf{k}_t, \mathbf{K}_t, V_{\mathbf{x}}, V_{\mathbf{xx}}$
    end for
    for $t = T$ **down to** 1 **do**
        $\mathbf{u}_t \leftarrow \mathbf{u}_t + \delta \mathbf{u}^\star$
        $\mathbf{x}_{t-1} \leftarrow \mathbf{h}(\mathbf{x}_t, \mathbf{u}_t)$ ▷ One controlled reverse diffusion step
    end for
end for

Initializing $V_{\mathbf{x}}$ and $V_{\mathbf{xx}}$ in the Algorithm

We choose the terminal value function at $t = 0$ as the log-likelihood:

$$V(\mathbf{x}_0, 0) \;=\; \log p(\mathbf{y} \mid \mathbf{x}_0).$$

For the usual linear-Gaussian forward model

$$\mathbf{y} = \mathbf{A}\mathbf{x}_0 + \eta, \qquad \eta \sim \mathcal{N}(0, \sigma^2 \mathbf{I}),$$

we have

$$p(\mathbf{y} \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{y};\, \mathbf{A}\mathbf{x}_0, \sigma^2 \mathbf{I}) \;\propto\; \exp\Big(-\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{A}\mathbf{x}_0\|^2 \Big).$$

So

$$\log p(\mathbf{y} \mid \mathbf{x}_0) = -\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{A}\mathbf{x}_0\|^2 + C.$$

Gradient and Hessian of $\log p(\mathbf{y} \mid \mathbf{x}_0)$

Write the quadratic term explicitly:

$$\|\mathbf{y} - \mathbf{A}\mathbf{x}_0\|^2 = (\mathbf{y} - \mathbf{A}\mathbf{x}_0)^\top (\mathbf{y} - \mathbf{A}\mathbf{x}_0).$$

Gradient:

$$\begin{aligned} \nabla_{\mathbf{x}_0} \|\mathbf{y} - \mathbf{A}\mathbf{x}_0\|^2 &= 2\,\mathbf{A}^\top (\mathbf{A}\mathbf{x}_0 - \mathbf{y}), \\ \Rightarrow \quad \nabla_{\mathbf{x}_0} \log p(\mathbf{y} \mid \mathbf{x}_0) &= -\frac{1}{2\sigma^2}\, 2\,\mathbf{A}^\top (\mathbf{A}\mathbf{x}_0 - \mathbf{y}) \\ &= \frac{1}{\sigma^2}\, \mathbf{A}^\top (\mathbf{y} - \mathbf{A}\mathbf{x}_0). \end{aligned}$$

Hessian:

$$\begin{aligned} \nabla^2_{\mathbf{x}_0} \log p(\mathbf{y} \mid \mathbf{x}_0) &= \nabla_{\mathbf{x}_0} \left[\frac{1}{\sigma^2}\, \mathbf{A}^\top (\mathbf{y} - \mathbf{A}\mathbf{x}_0) \right] \\ &= -\frac{1}{\sigma^2}\, \mathbf{A}^\top \mathbf{A}. \end{aligned}$$

Role of the Diffusion Model in $\mathbf{h}$

$$\mathbf{x}_{t-1} = \mathbf{x}_t - \Big[\mathbf{f}(\mathbf{x}_t) - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\Big] \Delta t.$$

- We never need $p_t(\mathbf{x}_t)$ explicitly.
- We only need its score

  $$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \approx s_\theta(\mathbf{x}_t, t),$$

  provided by the pretrained diffusion model.
- Using s_θ we implement $\mathbf{h}(\mathbf{x}_t)$ and its Jacobian $\mathbf{h}_\mathbf{x}$ (via automatic differentiation).

Thus, the diffusion model defines the **uncontrolled dynamics** and the prior, while optimal control (iLQR) uses $V_\mathbf{x}, V_{\mathbf{x}\mathbf{x}}$ from $\log p(\mathbf{y} \mid \mathbf{x}_0)$ to steer these dynamics.

Theorem 1 (Output Perturbation, Statement)

Consider the discretized reverse diffusion dynamics with **output perturbation**:

$$\mathbf{x}_{t-1} = \mathbf{x}_t - \Big[\mathbf{f}(\mathbf{x}_t) - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\Big]\Delta t + \mathbf{u}_t.$$

Let the terminal cost be

$$\ell_0(\mathbf{x}_0) = -\log p(\mathbf{y} \mid \mathbf{x}_0),$$

and running costs

$$\ell_t(\mathbf{x}_t, \mathbf{u}_t) = 0.$$

Suppose iLQR is used with a Tikhonov-regularized $Q_{\mathbf{uu}}$:

$$Q_{\mathbf{uu}} \leftarrow Q_{\mathbf{uu}} + \alpha \mathbf{I}.$$

Then the resulting control at time t satisfies

$$\mathbf{u}_t = \alpha \, \nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_0),$$

where $\mathbf{x}_0$ is obtained by rolling forward the controlled dynamics from $\mathbf{x}_t$.
Goal: Show that iLQR's control recovers the conditional score.

Theorem 1 (Output Perturbation, Proof) – Part 1

For output perturbation, the dynamics are linear in $\mathbf{u}_t$:

$$\mathbf{x}_{t-1} = \mathbf{h}(\mathbf{x}_t, \mathbf{u}_t) = \big[\text{uncontrolled step from } \mathbf{x}_t\big] + \mathbf{u}_t.$$

Hence

$$\mathbf{h}_{\mathbf{x}} = \frac{\partial \mathbf{x}_{t-1}}{\partial \mathbf{x}_t}, \qquad \mathbf{h}_{\mathbf{u}} = \mathbf{I}.$$

With $\ell_t(\mathbf{x}_t, \mathbf{u}_t) = 0$, the Q-derivatives simplify ($\ell_{\mathbf{u}} = \ell_{\mathbf{uu}} = 0$):

$$\begin{aligned}
Q_{\mathbf{x}} &= \mathbf{h}_{\mathbf{x}}^\top V'_{\mathbf{x}}, \\
Q_{\mathbf{u}} &= \ell_{\mathbf{u}} + \mathbf{h}_{\mathbf{u}}^\top V'_{\mathbf{x}} = V'_{\mathbf{x}}, \\
Q_{\mathbf{xx}} &= \mathbf{h}_{\mathbf{x}}^\top V'_{\mathbf{xx}} \mathbf{h}_{\mathbf{x}}, \\
Q_{\mathbf{ux}} &= \mathbf{h}_{\mathbf{u}}^\top V'_{\mathbf{xx}} \mathbf{h}_{\mathbf{x}} = V'_{\mathbf{xx}} \mathbf{h}_{\mathbf{x}}, \\
Q_{\mathbf{uu}} &= \ell_{\mathbf{uu}} + \mathbf{h}_{\mathbf{u}}^\top V'_{\mathbf{xx}} \mathbf{h}_{\mathbf{u}} = V'_{\mathbf{xx}}.
\end{aligned}$$

Theorem 1 (Output Perturbation, Proof) – Part 2

We add Tikhonov regularization:

$$Q_{\mathbf{uu}} \leftarrow Q_{\mathbf{uu}} + \alpha \mathbf{I} = V'_{\mathbf{xx}} + \alpha \mathbf{I}.$$

We have

$$V_{\mathbf{xx}} = Q_{\mathbf{xx}} - Q_{\mathbf{ux}}^T Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}} = h_{\mathbf{x}}^T V'_{\mathbf{xx}} h_{\mathbf{x}} - h_{\mathbf{x}}^T V'_{\mathbf{xx}} (V'_{\mathbf{xx}})^{-1} V'_{\mathbf{xx}} h_{\mathbf{x}} = \mathbf{0}.$$

Similarly,

$$V_{\mathbf{x}} = Q_{\mathbf{x}} + Q_{\mathbf{ux}}^T Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}} = h_{\mathbf{x}}^T V'_{\mathbf{x}} + h_{\mathbf{x}}^T V'_{\mathbf{xx}} (V'_{\mathbf{xx}})^{-1} V'_{\mathbf{x}} = h_{\mathbf{x}}^T V'_{\mathbf{x}}.$$

Feedforward term

$$\mathbf{k} = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}} = -(h_{\mathbf{x}}^T \underbrace{V_{\mathbf{xx}}}_{\mathbf{0}} h_{\mathbf{x}} + \alpha \mathbf{I})^{-1} Q_{\mathbf{u}} = -(\mathbf{0} + \alpha \mathbf{I})^{-1} Q_{\mathbf{u}} = -\frac{1}{\alpha} V'_{\mathbf{x}}.$$

Feedback term vanishes:

$$\mathbf{K}_t = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}} = 0,$$

since $V'_{\mathbf{xx}} = 0$.

Theorem 1 (Output Perturbation, Proof) – Part 3

We now relate $V_{\mathbf{x}}$ to the likelihood gradient.

At $t = 0$:

$$V(\mathbf{x}_0, 0) = \ell_0(\mathbf{x}_0) = -\log p(\mathbf{y} \mid \mathbf{x}_0),$$

hence

$$V_{\mathbf{x}}^{(0)} = \nabla_{\mathbf{x}_0} V = -\nabla_{\mathbf{x}_0} \log p(\mathbf{y} \mid \mathbf{x}_0).$$

Since we have the recursion

$$V_{\mathbf{x}}^{(1)} = \left(\frac{\partial \mathbf{x}_0}{\partial \mathbf{x}_1}\right)^\top V_{\mathbf{x}}^{(0)}, \quad V_{\mathbf{x}}^{(2)} = \left(\frac{\partial \mathbf{x}_1}{\partial \mathbf{x}_2}\right)^\top V_{\mathbf{x}}^{(1)} = \left(\frac{\partial \mathbf{x}_1}{\partial \mathbf{x}_2}\right)^\top \left(\frac{\partial \mathbf{x}_0}{\partial \mathbf{x}_1}\right)^\top V_{\mathbf{x}}^{(0)},$$

$$V_{\mathbf{x}}^{(t)} = \left(\frac{\partial \mathbf{x}_{t-1}}{\partial \mathbf{x}_t}\right)^\top \cdots \left(\frac{\partial \mathbf{x}_0}{\partial \mathbf{x}_1}\right)^\top V_{\mathbf{x}}^{(0)} = \left(\frac{\partial \mathbf{x}_0}{\partial \mathbf{x}_t}\right)^\top \left(-\nabla_{\mathbf{x}_0} \log p(\mathbf{y} \mid \mathbf{x}_0)\right).$$

$$V_{\mathbf{x}}^{(0)} = \left(-\nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_0(\mathbf{x}_t)\right).$$

Therefore,

$$\mathbf{k}_t = -\tfrac{1}{\alpha} V_{\mathbf{x}}' = \tfrac{1}{\alpha} \nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_0).$$

Lemma (Choosing α Recovers Posterior Sampling)

Lemma.

Consider the deterministic sampler (probability-flow ODE discretization) with **output perturbation** and control

$$\mathbf{u}_t = \frac{1}{\alpha}\,\nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_0).$$

If we choose

$$\alpha = \frac{1}{g(t)^2 \Delta t},$$

then the resulting update recovers the posterior sampling discretization

$$\mathbf{x}_{t-1} = \mathbf{x}_t + \left[\mathbf{f}(\mathbf{x}_t) - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t \mid \mathbf{y})\right]\Delta t.$$

Proof

From Theorem 1 we have

$$\mathbf{u}_t = \frac{1}{\alpha} \nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_0).$$

Substitute $\mathbf{u}_t$:

$$\mathbf{x}_{t-1} = \mathbf{x}_t + \Big[\mathbf{f}(\mathbf{x}_t) - \tfrac{1}{2} g(t)^2 \big(\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_0)\big)\Big] \Delta t.$$

Under the deterministic probability-flow dynamics, $\mathbf{x}_t$ and $\mathbf{x}_0$ are in one-to-one correspondence, so

$$\log p(\mathbf{y} \mid \mathbf{x}_0) = \log p(\mathbf{y} \mid \mathbf{x}_t),$$

with

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(\mathbf{y} \mid \mathbf{x}),$$

we recover

$$\mathbf{x}_{t-1} = \mathbf{x}_t - \Big[\mathbf{f}(\mathbf{x}_t) - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\Big] \Delta t.$$

□

Theorem 2 (Input Perturbation)

Consider the discretized reverse diffusion dynamics with **input perturbation**:

$$\mathbf{x}_{t-1} = (\mathbf{x}_t + \mathbf{u}_t) - \Big[\mathbf{f}(\mathbf{x}_t + \mathbf{u}_t) - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t + \mathbf{u}_t)\Big] \Delta t.$$

Let

$$\ell_0(\mathbf{x}_0) = -\log p(\mathbf{y} \mid \mathbf{x}_0), \qquad \ell_t(\mathbf{x}_t, \mathbf{u}_t) = 0,$$

and use iLQR with Tikhonov regularizer

$$\alpha = \frac{1}{g(t)^2 \Delta t}.$$

Define the shifted variable

$$\tilde{\mathbf{x}}_t := \mathbf{x}_t + \mathbf{u}_t.$$

Then the resulting dynamics can be written as

$$\tilde{\mathbf{x}}_{t-1} = \tilde{\mathbf{x}}_t + \big[\mathbf{f}(\tilde{\mathbf{x}}_t) - \tfrac{1}{2} g(t)^2 \big(\nabla_{\mathbf{x}} \log p_t(\tilde{\mathbf{x}}_t) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}_t)\big)\big] \Delta t.$$

Theorem 2 – Proof

Using $\ell_t \equiv 0$ for $t \geq 1$ and $\mathbf{h_u} = \mathbf{h_x}$, the (first–order) iLQR relations give

$$Q_{\mathbf{x}} = \mathbf{h}_{\mathbf{x}}^{\top} V'_{\mathbf{x}}, \qquad Q_{\mathbf{u}} = \mathbf{h}_{\mathbf{u}}^{\top} V'_{\mathbf{x}} = \mathbf{h}_{\mathbf{x}}^{\top} V'_{\mathbf{x}},$$

$$Q_{\mathbf{xx}} = \mathbf{h}_{\mathbf{x}}^{\top} V'_{\mathbf{xx}} \mathbf{h}_{\mathbf{x}}, \qquad Q_{\mathbf{ux}} = \mathbf{h}_{\mathbf{u}}^{\top} V'_{\mathbf{xx}} \mathbf{h}_{\mathbf{x}} = \mathbf{h}_{\mathbf{x}}^{\top} V'_{\mathbf{xx}} \mathbf{h}_{\mathbf{x}} = Q_{\mathbf{xx}},$$

$$Q_{\mathbf{uu}} = \ell_{\mathbf{uu}} + \mathbf{h}_{\mathbf{u}}^{\top} V'_{\mathbf{xx}} \mathbf{h}_{\mathbf{u}} = \mathbf{h}_{\mathbf{x}}^{\top} V'_{\mathbf{xx}} \mathbf{h}_{\mathbf{x}} = Q_{\mathbf{xx}},$$

where $V'_{\mathbf{x}}$ and $V'_{\mathbf{xx}}$ are evaluated at time $t-1$. We can write

$$V_{\mathbf{xx}} = Q_{\mathbf{xx}} - Q_{\mathbf{ux}}^{\top} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}} = 0.$$

Thus, for $t \geq 1$, the value-function Hessian vanishes: $V_{\mathbf{xx}} = 0$.
Similarly, with $V_{\mathbf{xx}} = 0$ and $\ell_t \equiv 0$, we have

$$V_{\mathbf{x}} = Q_{\mathbf{x}} = \mathbf{h}_{\mathbf{x}}^{\top} V'_{\mathbf{x}}.$$

Theorem 2 – Proof

Since $V'_{\mathbf{xx}} = 0$ implies $Q_{\mathbf{uu}} = 0$, we use the regularized inverse

$$(Q_{\mathbf{uu}} + \alpha I)^{-1} = (\alpha I)^{-1}.$$

Feedforward gain:

$$\mathbf{k} = -(Q_{\mathbf{uu}} + \alpha I)^{-1} Q_{\mathbf{u}} = -(\alpha I)^{-1} \mathbf{h}_{\mathbf{x}}^{\top} V'_{\mathbf{x}} = -\frac{1}{\alpha}\, \mathbf{h}_{\mathbf{x}}^{\top} V'_{\mathbf{x}}.$$

Feedback gain:

$$\mathbf{K} = -(Q_{\mathbf{uu}} + \alpha I)^{-1} Q_{\mathbf{ux}} = -(\alpha I)^{-1} \mathbf{h}_{\mathbf{x}}^{\top} V'_{\mathbf{xx}} \mathbf{h}_{\mathbf{x}} = 0,$$

because $V'_{\mathbf{xx}} = 0$.

Chain rule for $V_{\mathbf{x}}^{(t)}$. Similar to Theorem 1, we obtain

$$V_{\mathbf{x}}^{(t)} = \mathbf{h}_{\mathbf{x}}^{(t)\top} V_{\mathbf{x}}^{(t-1)} = \nabla_{\mathbf{x}_t} \log p\left(\mathbf{y} \mid \mathbf{x}_0(\mathbf{x}_t)\right),$$

where $\mathbf{x}_0$ is the terminal state obtained by rolling out the dynamics backwards from $\mathbf{x}_t$.

Theorem 2 – Proof

Then

$$Q_{\mathbf{u}} = \mathbf{h}_{\mathbf{x}}^{\top} V'_{\mathbf{x}} = V_{\mathbf{x}}^{(t)} = \nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_0),$$

so

$$\mathbf{k}_t = -\frac{1}{\alpha} Q_{\mathbf{u}} = -\frac{1}{\alpha} \nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_0), \qquad \mathbf{K}_t = 0.$$

Thus the optimal control (for the backward pass) is

$$\mathbf{u}_t = \mathbf{k}_t = -\frac{1}{\alpha} \nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_0).$$

Input–perturbation sampler. The deterministic reverse PF–ODE step (Euler) with input perturbation is

$$\mathbf{x}_{t-1} = (\mathbf{x}_t + \mathbf{u}_t) - \Big[f(\mathbf{x}_t + \mathbf{u}_t) - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t + \mathbf{u}_t) \Big] \Delta t.$$

Theorem 2 – Proof

Define the perturbed state

$$\tilde{\mathbf{x}}_t := \mathbf{x}_t + \mathbf{u}_t.$$

Then

$$\tilde{\mathbf{x}}_{t-1} - \mathbf{u}_{t-1} = \tilde{\mathbf{x}}_t - \Big[f(\tilde{\mathbf{x}}_t) - \tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\tilde{\mathbf{x}}_t)\Big] \Delta t.$$

With

$$\mathbf{u}_{t-1} = -\frac{1}{\alpha} \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}_0),$$

and choose

$$\alpha = \frac{1}{g(t)^2 \Delta t}.$$

Interpreting this as an additional guidance term in the reverse flow, the resulting dynamics for the perturbed state can be written as

$$\tilde{\mathbf{x}}_{t-1} = \tilde{\mathbf{x}}_t + \Big[f(\tilde{\mathbf{x}}_t) - \tfrac{1}{2} g(t)^2 \big(\nabla_{\mathbf{x}} \log p_t(\tilde{\mathbf{x}}_t) + \nabla_{\mathbf{x}} \log p_t(\mathbf{y} \mid \tilde{\mathbf{x}}_t)\big)\Big] \Delta t,$$

i.e. an Euler step for the ideal posterior sampler in input-perturbation form, with the conditional score recovered from the iLQR control.

Summary of Theoretical Insights

- The reverse diffusion sampler (probability-flow ODE) can be seen as a discrete-time dynamical system.
- Injecting controls $\mathbf{u}_t$ and optimizing a cost

$$\ell_0(\mathbf{x}_0) = -\log p(\mathbf{y} \mid \mathbf{x}_0)$$

  leads to an optimal control problem.
- Using iLQR with a Tikhonov regularizer:
  - **Output perturbation**: $\mathbf{u}_t$ becomes proportional to the conditional score

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_0),$$

    and with suitable α recovers ideal posterior sampling.
  - **Input perturbation**: yields a predictor-corrector-like sampler combining unconditional and conditional scores.
- This connects probabilistic posterior sampling to deterministic optimal control on the discretized dynamics.

High-Dimensional Control: Challenges

- State and control are image-sized:

$$x_t \in \mathbb{R}^d, \quad u_t \in \mathbb{R}^d, \quad d \approx 256 \times 256 \times 3.$$

- iLQR requires Jacobians and Hessians:

$$h_x,\ h_u,\ V_{xx},\ Q_{xx},\ Q_{ux},\ Q_{xu},\ Q_{uu}.$$

- Naive storage cost:

$$\dim(V_{xx}) = d \times d \approx (256 \cdot 256 \cdot 3)^2 \approx 3.9 \times 10^{10} \text{ entries.}$$

- Direct formation and inversion of these matrices is infeasible in terms of:
  - memory (quadratic in d),
  - runtime (many backprop passes per diffusion step).
- Appendix D.1 introduces two modifications:
  1. Randomized low-rank approximations,
  2. Matrix-free evaluation

Matrix-Free Evaluation: Idea

Goal: avoid materializing any $d \times d$ matrices at all.

- For each large matrix A_i (e.g. Q_{xx}, Q_{ux}, Q_{uu}), store compressed pairs

$$(Q_i, B_i), \quad B_i = Q_i^\top A_i,$$

  with $Q_i \in \mathbb{R}^{d\times\ell}$, $B_i \in \mathbb{R}^{\ell\times d}$.
- Approximate products of the form $A_i A_j$ as

$$A_i A_j \approx Q_i B_i Q_j^\top B_j.$$

- To keep things "matrix-free", we drop the leading Q_i and carry only projected pieces; effectively we propagate new (Q_k, B_k) pairs instead of full matrices.
- This turns all large matrix multiplications into operations on $\ell \times \ell$ and $d \times \ell$ blocks.

Matrix-Free Evaluation: In iLQR

In Appendix D.2, matrix-free is expressed via projected quantities. For a projection matrix $P \in \mathbb{R}^{d \times k}$:

$$\begin{aligned}
Q_x &= h_x^\top V'_x, \\
Q_u &= \ell_u + h_x^\top V'_x, \\
PQ_{xx}P^\top = PQ_{ux}P^\top = PQ_{xu}P^\top &= Ph_x^\top V'_{xx} h_x P^\top, \\
PQ_{uu}P^\top &= P\ell_{uu}P^\top + Ph_x^\top V'_{xx} h_x P^\top.
\end{aligned}$$

- Q_x, Q_u are d-dimensional and kept at full resolution (small enough to store).
- Second-order terms are *only* stored through their projections $PQP^\top \in \mathbb{R}^{k \times k}$.
- All heavy operations (inversions, multiplications) are thus on $k \times k$ blocks.

Tikhonov Regularization and Inversion

- In high dimensions, Q_{uu} can be ill-conditioned or singular.
- Use Tikhonov regularization in iLQR:

$$Q_{uu} \;\mapsto\; Q_{uu} + \alpha I, \qquad \alpha > 0.$$

- In the projected space:

$$PQ_{uu}^{-1}P^\top = P\ell_{uu}^{-1}P^\top + P\ell_{uu}^{-1}P^\top \left(C^{-1} + P^\top \ell_{uu}^{-1} P\right)^{-1} P\ell_{uu}^{-1}P^\top,$$

where

$$C = Ph_x^\top V'_{xx} h_x P.$$

- This is a direct application of the Woodbury matrix identity, reducing inversion to $k \times k$ matrices.

Matrix-Free iLQR Updates

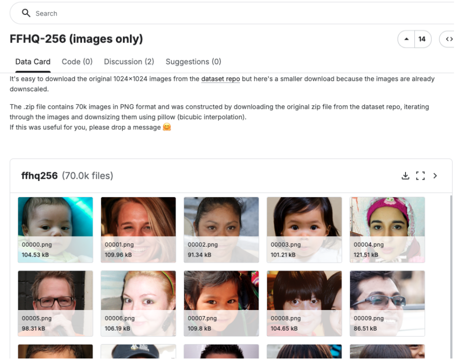
With projections, the iLQR updates become

$$\begin{aligned}
\mathbf{k} &= -P^\top \left(PQ_{uu}^{-1}P^\top\right) PQ_u, \\
V_x &= Q_x - P^\top \left(PK^\top P^\top\right) PQ_{uu}P^\top \left(Pk\right), \\
PKP^\top &= -\left(PQ_{uu}^{-1}P^\top\right) PQ_{ux}P^\top, \\
PV_{xx}P^\top &= PQ_{xx}P^\top - PK^\top P^\top PQ_{uu}P^\top PKP^\top.
\end{aligned}$$

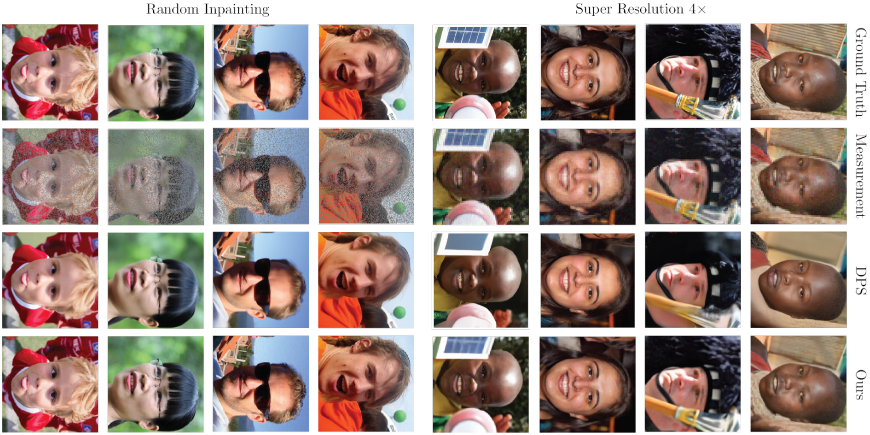
- Only $PV_{xx}P^\top$ is stored; V_{xx} itself is never formed.
- V_x and k remain full d-dimensional vectors (image-sized).
- Second-order structure enters only via low-rank projected quantities.

Experimental Setup

- **Tasks**: Linear inverse problems on FFHQ 256×256.
- **Measurement noise**: add i.i.d. Gaussian noise with $\sigma = 0.05$ to all measurements.
- **Dataset**: 256×256 human faces.
  - Train diffusion on 49K images.
  - Hold out last 1K images for evaluation.



Results on FFHQ



Nonlinear Inverse Problem: MNIST Class-Conditional Generation

- **Task**: Nonlinear inverse problem defined via a classifier

$$A(x) = \text{classifier}(x), \quad p(y \mid x) \text{ from a CNN trained on MNIST.}$$

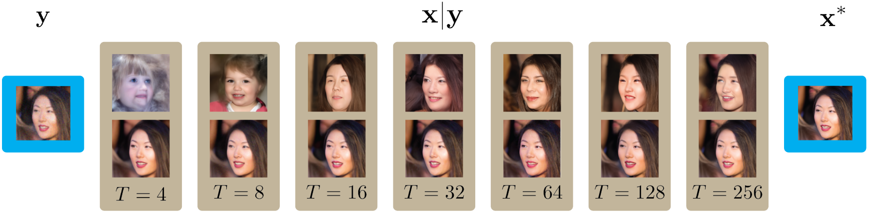
- **Goal**: Generate an MNIST digit x given a label y (class-guided sampling).
- **Model**:
  - Diffusion prior: HuggingFace `1aurent/mnist-28` model.
  - Likelihood term: classifier-based $p(y \mid x)$.
- **Comparison**: Our method vs. DPS.
- **Qualitative findings**:
  - Our samples show stronger alignment with the target label.
  - Higher perceived sample quality (sharper and more class-distinct digits).

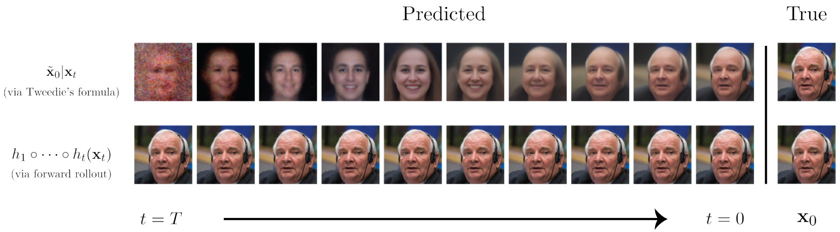
Results on MNIST

(a) DPS

(b) Optimal control based diffusion

Results on sensitivity to T and score





Results on random initialization of DPS

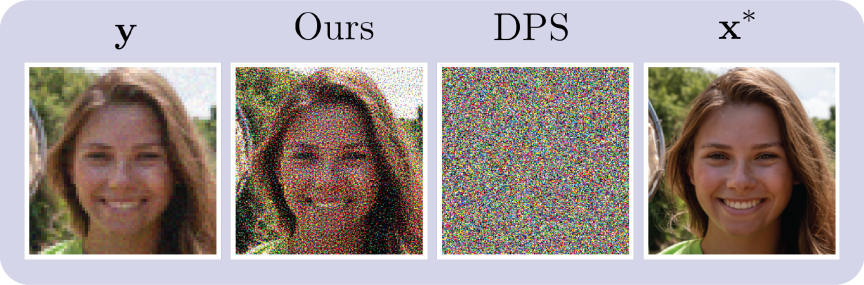


Figure: Robustness to approximation quality of the score function. We consider the $4\times$ super-resolution task with a randomly initialized diffusion model. Since the reverse diffusion process is no longer well approximated, DPS cannot produce a feasible solution, while our method still can.

Q/A

Thanks! Questions?