Predict-Project-Renoise: Sampling Diffusion Models under Hard Constraints

Omer Rochman-Sharabi
University of Liège
o.rochman@uliege.be

Gilles Louppe
University of Liège

Abstract

Diffusion models cannot enforce hard constraints, yet applications in the physical sciences demand exact satisfaction of conservation laws, boundary conditions, and observational consistency. In this work, we identify a corrector kernel whose unique stationary distribution is the constrained marginal at each noise level, and approximate it by iteratively projecting through the denoiser and renoising via the forward kernel. The resulting Predict-Project-Renoise (PPR) algorithm enables sampling from pretrained diffusion models under hard constraints. Its three components are each necessary: projecting through the denoiser keeps samples close to the data manifold, while renoising and iterating drive samples toward the constrained marginal. On 2D distributions, the Kuramoto-Sivashinsky equation, and global weather forecasting with a 10^8-dimensional atmospheric model, PPR simultaneously achieves low constraint violations and high distributional fidelity, a combination that existing methods fail to deliver.

1 Introduction

Diffusion models [39, 16, 43] have emerged as the state-of-the-art framework for deep generative modeling, showing strong performance in approximating high-dimensional data distributions p across modalities. However, in many scientific and engineering domains, such as medical imaging, fluid dynamics, or climate modeling, sampling p is insufficient. Samples x must also satisfy hard constraints $x \in \mathcal{C}$, such as exact consistency with observations or governing equations, zero divergence in incompressible fluids, conservation laws, boundary and initial conditions, or geometric constraints.

Although training-based approaches can incorporate constraints [49, 22, 32], they often lack flexibility, requiring retraining for different constraints, and enforce the constraints only softly. For these reasons, recent efforts focus on *training-free constrained sampling*, where a pre-trained unconditional model is guided to satisfy constraints during inference. Current methods typically rely on treating constraints as observations [10] or implement projection-based corrections [8, 35].

When treating the constraint as an observation, Bayesian decomposition can be used to express the posterior score as a prior score plus a likelihood score. While effective for noisy inverse problems [9], especially in the context of images, these methods are ill-suited for hard constraints: the implicit likelihood approaches a Dirac delta, causing gradients to become unstable and informative only in the immediate vicinity of the feasible set $\mathcal{C}$. Projection methods can enforce feasibility explicitly, but have theoretical and practical pitfalls as they do not correctly sample the true constrained distribution. This manifests as three problems: (i) the constraints are violated, (ii) the constrained distribution is incorrect, and (iii) the sample lies outside of the support $\mathcal{S}$ of the data distribution p.

For nonlinear constraints, the projection typically has no closed-form solution, so it must be approximated with an iterative solver. Moreover, it is often unclear what exactly should be projected. Directly projecting the noisy state onto the constraint set can be ill-posed or numerically unstable, since many constraints are only meaningful for clean data. Projecting an estimate of the clean sample is not a fix

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

either. Sample consistency with p is often broken, for example, by introducing discontinuities into otherwise continuous samples. Finally, repeated projection steps can systematically bias the sampler. In the extreme, this behaves like an attractor toward "nearest" feasible points, causing mode collapse rather than sampling from the constrained distribution.

Contributions. In this work, we address these limitations by arguing that correct constrained sampling requires approximating the constrained marginal distributions $p_t^{\mathcal{C}}$ induced by a forward diffusion SDE starting from the constrained set $\mathcal{C}$. We define the constrained forward process and derive the associated constrained marginals and backward process. Based on the backward process, we identify a corrector kernel that has the constrained marginals as its unique stationary distributions, design the Predict-Project-Renoise (PPR) algorithm that approximates that kernel, and show that each of its components is necessary. PPR achieves higher feasibility rates, lower constraint violations, and better mass distribution compared to other methods.

2 Score-based generative modeling

Forward process. Generative modeling seeks to approximate a data distribution $p(x)$ of $x \in \mathbb{R}^D$. In score-based diffusion [43], this is done by learning to reverse a forward corruption process. Following Lai et al. [25], we define the *forward process* as the stochastic differential equation (SDE)

$$\mathrm{d}x_t = f(x_t, t)\mathrm{d}t + g(t)\mathrm{d}w_t, \tag{1}$$

where $f : \mathbb{R}^{D+1} \to \mathbb{R}^D$ is a drift term, $g : \mathbb{R} \to \mathbb{R}$ is a scalar diffusion coefficient, and w_t is a standard Wiener process. Starting from $p_0 = p$, this SDE defines a series of marginal densities $\{p_t(x_t)\}_{t\in[0,T]}$ which interpolate between the data distribution p_0 and p_T. Choosing f and g defines the perturbation or *forward kernels* $p(x_t|x_0)$, inducing the marginals

$$p_t(x_t) = \int p(x_t \mid x_0)\, p(x_0)\, \mathrm{d}x_0. \tag{2}$$

If f and g are chosen appropriately, then $p(x_t|x_0)$ and p_T become Gaussian [44]. In particular, p_T can become $\mathcal{N}(0, \sigma_T I)$.

Backward process. Anderson [2] shows that given a forward process (Eq. 1), there exists a *reverse* or *backward process* from T to 0 transporting p_T back to p_0 given by

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

where $x_T \sim p_T(x_T)$ and $\tilde{w}_t$ is the reverse-time Wiener process. In this reverse process, the marginal distributions of x_t are identical to those of the forward process. This allows for sampling from p by generating a sample from p_T and then simulating the reverse process.

Training and sampling. Since the score $s(x_t) = \nabla_{x_t} \log p_t(x_t)$ is unknown, it has to be learned from data. We follow Karras et al. [21] and parameterize the score using Tweedie's formula $s_\theta(x_t) = (d_\theta(x_t) - x_t)/\sigma_t^2$ [45, 11]. The resulting denoising objective minimizes

$$\mathbb{E}_{x_0\sim p(x_0)}\mathbb{E}_{x_t\sim p(x_t|x_0)}\|d_\theta(x_t) - x_0\|^2. \tag{4}$$

The denoiser $d_\theta(x_t) \approx \mathbb{E}[x_0|x_t]$ predicts the posterior mean of the clean sample given the noisy input. Once it is available, a way to sample from the reverse process is to use the predictor-corrector (PC) sampler (Song et al. [43], Algorithm 1). This algorithm alternates between a PREDICT step (e.g. DDPM [16], DDIM [41]), which moves the samples from p_{t+1} to p_t, and a number of CORRECT steps (e.g. Langevin steps based on the score). The CORRECT steps reduce the discretization error by pulling samples toward higher-density regions of p_t and re-injecting noise to increase diversity. PREDICT can be any valid transition. In our experiments it is DDIM, which takes the denoised estimate $\hat{x}_0 = d_\theta(x_{t+1})$ as input.

3 Method

The reverse process (Eq. 3) allows us to approximate the data distribution p, but in many domains we are interested in imposing constraints on the samples. Let $\mathcal{S} \subseteq \mathbb{R}^D$ be the support of $p(x)$ and

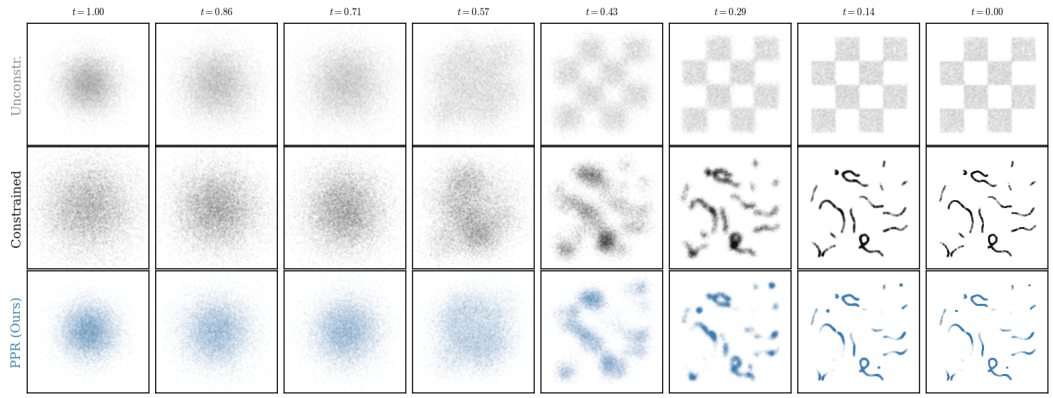


Figure 1: Visualization of the constrained process on DATA2D. The top row shows the unconstrained marginals p_t. The middle row shows the target constrained marginals $p_t^{\mathcal{C}}$, obtained by applying the diffusion transition kernel to $p_0^{\mathcal{C}}$, which was sampled by rejection sampling. The bottom row shows the marginals induced by PPR. The columns progress in diffusion time, with the last column corresponding to $t = 0$ (clean data).

$\mathcal{C} \subseteq \mathbb{R}^D$ be a (constrained or feasible) set corresponding to samples with certain characteristics, such as conservation laws, adherence to given dynamics, or a desired decay in the energy spectrum. $\mathcal{C}$ can be represented using a nonnegative constraint function $c(x) \geq 0$ as $\mathcal{C} \coloneqq \{x \in \mathbb{R}^D : c(x) = 0\}$. The intersection $\mathcal{S} \cap \mathcal{C}$ strictly restricts the prior to the feasible set, inducing the density $p^{\mathcal{C}} \propto p(x_0)\mathbb{1}_{\mathcal{C}}(x_0)$, where $\mathbb{1}_{\mathcal{C}}(x)$ is the indicator function of $\mathcal{C}$. We say a sample $x \in \mathcal{S} \cap \mathcal{C}$ is *feasible*.

3.1 Constrained diffusion

Constrained forward process. We define the *constrained forward process* similarly to Eq. 1, but with $x_0 \sim p^{\mathcal{C}}(x_0)$. This constrained process induces a sequence of marginals $\{p_t^{\mathcal{C}}\}_{t\in[0,T]}$, given by

$$p_t^{\mathcal{C}}(x_t) \propto \int p(x_t \mid x_0)\, p(x_0)\, \mathbb{1}_{\mathcal{C}}(x_0)\, \mathrm{d}x_0. \tag{5}$$

The unconstrained and constrained forward processes share the same dynamics and transition kernels but their different initial densities lead to distinct marginals and reverse processes.

Constrained backward process. Mirroring the unconstrained backward process in Eq. 3, the *constrained reverse process* is

$$\mathrm{d}x_t = \left[f(x_t, t) - g^2(t)\nabla_{x_t} \log p_t^{\mathcal{C}}(x_t)\right] \mathrm{d}t + g(t)\mathrm{d}\tilde{w}_t. \tag{6}$$

This process involves the constrained score function $\nabla_{x_t} \log p_t^{\mathcal{C}}(x_t)$. It can be computed by taking the log and then the gradient of the constrained marginals (Eq. 5)

$$\nabla_{x_t} \log \int_{\mathcal{C}} p(x_t|x_0)p(x_0)\mathrm{d}x_0, \tag{7}$$

dropping the normalization constant because it vanishes under the derivative and absorbing the indicator function into the integral. Then, from Bayes' theorem, $p(x_t|x_0)p(x_0) = p(x_0|x_t)p(x_t)$, we get

$$\nabla_{x_t} \log p_t^{\mathcal{C}}(x_t) = \nabla_{x_t} \log p_t(x_t) + \nabla_{x_t} \log \int_{\mathcal{C}} p(x_0|x_t)\mathrm{d}x_0. \tag{8}$$

The constrained score is thus the unconstrained score plus the gradient of the log-feasibility $\nabla_{x_t} \log \int_{\mathcal{C}} p(x_0|x_t)\mathrm{d}x_0 = \nabla_{x_t} \log \mathbb{P}(x_0 \in \mathcal{C}|x_t)$, a term pointing in the direction (in x_t-space) of increasing probability of x_0 being in $\mathcal{C}$. Figure 1 gives visual intuition on these processes. No additional training is required. Eqs. 5 and 6 define the target processes analytically, and PPR imposes the constraint at inference time on a pretrained denoiser, which can be reused for different constraints.

3.2 Predict-Project-Renoise (PPR) sampling

At each reverse step, the unconstrained predictor (Algorithm 1) provides $x_t \sim p_t$, whose denoised estimates are unlikely to satisfy the constraints exactly. Based on the structure of the constrained backward process (Section 3.1), we identify the following corrector:

Proposition 3.1. *Let* $K_t(x'_t|x_t) = \int_{\mathcal{C}} p(x'_t|x_0)\, p^{\mathcal{C}}(x_0|x_t)\, dx_0$ *with* $p^{\mathcal{C}}(x_0|x_t) = \frac{p(x_0|x_t)\,\mathbb{1}_{\mathcal{C}}(x_0)}{\mathbb{P}(x_0 \in \mathcal{C}|x_t)}$. *Under this kernel,* $p_t^{\mathcal{C}}$ *is the unique stationary distribution and* $(K_t)^M q \to p_t^{\mathcal{C}}$ *as* $M \to \infty$ *for any initial distribution* q. *See Appendix C for the proof.*

That is, iterating the kernel K_t at a fixed noise level acts as a corrector that drives any sample, including $x_t \sim p_t$ from the predictor, toward the constrained marginal $p_t^{\mathcal{C}}$, just as Langevin correction drives samples toward p_t. Its structure is a two-step conditional resampling: sample a feasible $x_0 \in \mathcal{C}$ from the constrained posterior $p^{\mathcal{C}}(x_0|x_t)$, then renoise via the forward kernel to obtain x'_t. Renoising is essential as it restores the correct noise level and injects the stochasticity needed for mixing.

Of course, sampling from the constrained posterior $p^{\mathcal{C}}(x_0|x_t)$ is itself intractable. Instead, PPR approximates it with a point estimate $p^{\mathcal{C}}(x_0|x_t) \approx \delta(x_0 - d_\theta(\textsc{Project}(x_t)))$, where $\textsc{Project}(x_t) \coloneqq \arg\min_x c(d_\theta(x))$ is solved via an off-the-shelf optimizer initialized at x_t. Plugging this approximation into K_t yields

$$K_t^{\mathrm{PPR}}(x'_t|x_t) \coloneqq p\left(x'_t \mid d_\theta(\textsc{Project}(x_t))\right), \tag{9}$$

where the integral over $\mathcal{C}$ in K_t collapses to a single forward kernel evaluation at the projected point estimate. Again, the renoising step is critical: without it, K_t^{PPR} reduces to a deterministic map that collapses to fixed points of the projection; with it, the kernel retains the stochasticity needed for mixing. Following Proposition 3.1, we iterate K_t^{PPR} M times to drive samples toward $p_t^{\mathcal{C}}$. In summary, the proposed *Predict-Project-Renoise* sampler (Algorithm 2) alternates at each reverse step between a predictor update, a projection through the denoiser, and a renoising step, with the last two repeated M times.

Approximation error. Replacing the constrained posterior with a point estimate introduces an error that we characterize in Appendix C. The error decomposes into two terms: a mixing term that decays exponentially with M, justifying the repetition of project-renoise steps, and an irreducible bias from the point estimate approximation, which is small when the posterior concentrates around a single feasible x_0. Proposition 3.1 concerns the ideal kernel. Practical PPR replaces the conditional draw with a projected point estimate, and a finite optimizer does not solve every nonconvex c exactly. Feasibility is therefore evaluated at a stated tolerance.

Projecting through the denoiser. There are three natural ways to enforce feasibility during reverse sampling. The first is to minimize $c(x_t)$ with respect to x_t directly. However, this is problematic because constraints are defined on clean data, while $x_t \notin \mathcal{S}$ almost surely, making $c(x_t)$ ill-conditioned or undefined. The second is to compute $x_0 = d_\theta(x_t)$ and minimize $c(x_0)$ with respect to x_0. This evaluates the constraint where it is meaningful, but the optimized x_0 is free to leave $\mathcal{S}$, producing artifacts such as discontinuities (Section 5). PPR takes a third route: minimize $c(d_\theta(x_t))$ with respect to x_t. Because the denoiser is trained to output elements of $\mathcal{S}$, the projected estimate $d_\theta(x_t^*)$ is encouraged to remain on the data manifold. Backpropagation through d_θ further propagates local constraint information through learned correlations, helping maintain $x_0 \in \mathcal{C} \cap \mathcal{S}$ globally. The cost is higher due to differentiation through d_θ, but can be negligible when the constraint itself dominates computation, as in latent-space models with large decoders. In practice, PROJECT is solved with off-the-shelf optimizers, L-BFGS [27] for DATA2D and KS and Adam [23] for WEATHER, with a fixed maximum of 8 iterations and no convergence-based stopping. We observed no NaNs, failed solves, or need for restarts. Renoising perturbs the state after a possibly imperfect solve, so later PROJECT applications do not repeat the same optimization trajectory. The projection budget provides a direct tradeoff between compute and sample feasibility.

4 Related Work

Previous work on constrained sampling in diffusion models is summarized in Table 1. First, we focus on projection-based constrained samplers and then discuss a related body of work on posterior sampling under noisy observation processes. To help understand how previous methods relate to each

other, we annotate each projection-based method according to the space in which the projection is done. If done with respect to x_t, we say the projection is in x_t-space and use red. If done with respect to x_0, we call it x_0-space and use blue. We will also add "through the denoiser" if the x_t-space projection is done using $c(d_\theta(x_t))$ and "renoise" if any noise is added after the projection.

Projection-based constrained sampling. A number of methods enforce constraints by modifying reverse-time sampling with projections. Christopher et al. [5] (x_t-space) and Liu et al. [28] focus on convex constraints, while Jayaram et al. [19, 20] study linear constraints. Both are x_t-space and the latter project through the denoiser.

Zampini et al. [48] (x_t-space) use proximal Langevin dynamics within the latent x_t space, propagating gradients through the decoder. Their approach has the side effect of mitigating potential deviations from $\mathcal{S}$ by leveraging a structured latent space where the decoder can correct samples. In contrast, Narasimhan et al. [33] (x_0-space) operate solely in x_0-space, which can cause samples to be inconsistent with the data distribution, i.e. $x_0 \notin \mathcal{S}$. For example, discontinuities can appear in continuous data (see, in their appendix, Figures 6 and 10). Adding more information via constraints can mask this issue. Huang et al. [18] (x_t-space through the denoiser with renoising) propose a heuristic trust region method based on the magnitude of the predicted noise $\epsilon_\theta \propto x_t - d_\theta(x_t)$. We refer to this method as TS for Trust Sampling. Christopher et al. [6] (x_0-space) uses domain knowledge, global constraints, and the forward kernel for protein design. In the flow matching context, Cheng et al. [4] (x_0-space) use extrapolation, a known exact correction, and interpolation to generate the constrained samples.

Table 1: Summary of related works, showing in which space they operate, if they reapply the kernel, and whether they propagate information through d_θ.

Method	Space	Kernel reapplications	Prop. d_θ
Constraints as observations			
DPS [7]	x_t	✗	✓
MMPS [36]	x_t	✗	✓
ReSample/HDC [40]	x_0	✓	✗
Constrained sampling			
CPS [33]	x_0	✗	✗
Zampini et al. [48]	x_t	✗	✗
TS [18]	x_t	✓	✓
PDM [5]	x_t	✗	✗
CDIM [20]	x_t	✗	✗
PPR (Ours)	x_t	✓	✓

Posterior sampling under observation processes. A distinct yet related body of work focuses on sampling from posteriors derived from observation processes. While constrained sampling enforces strict adherence to a feasible set, observation-based methods typically condition generation on external measurements, often assuming an underlying Gaussian likelihood $p(y|x) = \mathcal{N}(y \mid A(x), \sigma^2)$, where A is the observation process and y is the observation value, as proposed in [30, 42, 7, 36, 13]. Theoretically, hard constraints can be framed as a special case of inverse problems by modeling the observation likelihood as a Dirac delta distribution $\mathcal{N}(y \mid A(x), \sigma^2) \xrightarrow{\sigma^2 \to 0} \delta(y - A(x))$. However, applying standard posterior sampling algorithms [9, 47] in this regime presents challenges, as these methods are generally designed for more informative likelihoods rather than the narrow distributions required for strict constraint satisfaction. Notably among these, Song et al. [40] (x_0-space) focus on constraints and do renoising with a hand-designed kernel. Huang et al. [17] (DiffusionPDE) guide an

Algorithm 1 Predictor-corrector sampling.

Input: denoiser d_θ, predictor PREDICT, corrector CORRECT, and N.
Initialize: sample $x_T \sim p_T(x_T)$
for decreasing $t \in \{T-1, \dots, 0\}$ **do**
  $\hat{x}_0 \leftarrow d_\theta(x_{t+1})$
  $x_t \leftarrow \text{PREDICT}(x_{t+1}, \hat{x}_0, t+1)$
  for $i = 1, \dots, N$ **do**
    $\hat{x}_0 \leftarrow d_\theta(x_t)$
    $x_t \leftarrow \text{CORRECT}(x_t, \hat{x}_0, t)$
  end for
end for
Output: x_0

Algorithm 2 Predict-Project-Renoise (PPR).

Input: denoiser d_θ, predictor PREDICT, projection Π_d, steps M.
Initialize: sample $x_T \sim p_T(x_T)$
for decreasing $t \in \{T-1, \dots, 0\}$ **do**
  $\hat{x}_0 \leftarrow d_\theta(x_{t+1})$
  $x_t \leftarrow \text{PREDICT}(x_{t+1}, \hat{x}_0, t+1)$
  for $i = 1, \dots, M$ **do**
    $x_t^* \leftarrow \text{PROJECT}(x_t)$
    $x_t \sim p(x_t \mid x_0 = d_\theta(x_t^*))$ {RENOISE}
  end for
end for
Output: x_0

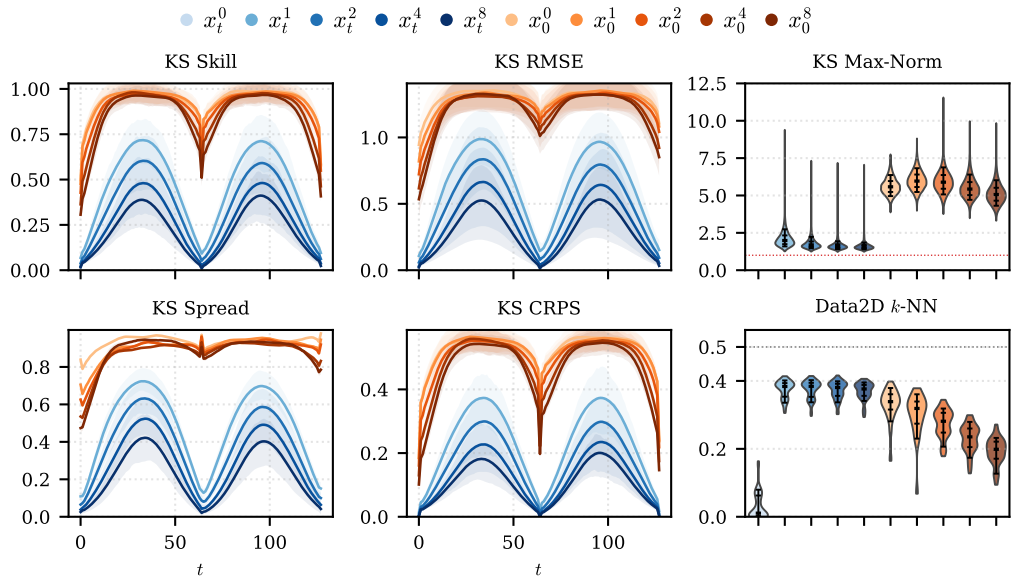


Figure 2: KS and DATA2D PPR ablation. We vary the number of kernel applications M and whether information is propagated through the denoiser. Projecting in x_0-space rather than x_t-space worsens the constrained distribution for both datasets and in KS, x_0-space samples are not in $\mathcal{S}$ because they are discontinuous. When projecting through the denoiser in x_t-space, larger M improves the constrained distributions in both settings. For KS, skill, RMSE, and CRPS measure prediction accuracy; for DATA2D, k-NN measures constrained-distribution quality. Max-Norm is better when closer to 1 (dotted red line). In the KS metrics, the x-axis t denotes system time, not diffusion time.

unconditional prior with observation and PDE-residual gradients. Yao et al. [46] (FunDPS) train a function-space neural-operator diffusion prior and then apply likelihood-gradient guidance.

5 Experiments

Protocol. We consider three experimental settings: DATA2D, Kuramoto-Sivashinsky (KS), and WEATHER. DATA2D comprises 2D problems where the ground truth constrained distribution can be approximated. KS is based on the 1D Kuramoto-Sivashinsky equation [24] and assesses our method and the baselines in a high-dimensional setting. Finally, WEATHER uses Appa, a global latent weather model [3], to test PPR in a large-scale, realistic setting. The full numerical results, additional experiments, and qualitative samples are shown in the Appendices D, E, F, and G. The code is available at `https://github.com/OmerRochman/ppr`. Numerically, we consider $x \in \mathcal{C}$ if $c(x) \leq 4 \times 10^{-4}$ for KS and $c(x) \leq 4 \times 10^{-6}$ for DATA2D.

DATA2D. We first train diffusion models on two distinct 2D distributions, a uniform checkerboard (Fig. 1, top), and a mixture of Gaussians, which serve as priors. The term "prior" refers to these learned distributions $p(x_0)$, not to the terminal distribution p_T, which remains Gaussian. We then generate 12 random functions $f(x)$ by sampling cosine series. For each function, we define the constraint $c(x) = 1 - e^{-f(x)^2}$, so that $\mathcal{C}$ is the zero-level set of f, and approximate the ground-truth (GT) constrained distribution by rejection sampling from the diffusion model prior with a small tolerance ϵ, that is, $\text{GT} = \{x \in \mathcal{S} : c(x) \leq \epsilon\}$. The accepted samples are then refined with a few deterministic gradient steps to better satisfy the constraint. This results in 12 constraints for each of the 2 distributions, giving 24 total experimental settings. For each of these, we generated ensembles with PPR and the baselines to approximate the GT constrained distribution. More details can be found in Appendix B. To assess the quality of the resulting constrained distributions, we use the k-nearest-neighbor (k-NN) cross-edge rate [37]. The k-NN cross-edge rate constructs a graph on the union of GT and model samples, and counts the number of edges that connect a GT sample to a model sample. The cross-edge rate (cross edges divided by total edges) is expected to be 0.5 when the two distributions match. The results are averaged across all prior distributions and all 12 constraints.

KURAMOTO-SIVASHINSKY. We train a diffusion prior to generate trajectories of the 1D KS equation $u_t = -u_{xx} - u_{xxxx} - uu_x$. To avoid confusion with diffusion time, we denote the dynamical system time with a superscript and generated samples with a subscript u_0, which is omitted for the ground truth. We generate a dataset by sampling cosine series initial conditions $u^0 \in \mathbb{R}^{512}$ and solving the equation with periodic boundary conditions on $x \in [0, L]$, resulting in trajectories $u^{1:1024}$. We then select a subset of each trajectory $u^{512:1024} \in \mathbb{R}^{512\times512}$. This is done to ensure that the system has reached the statistically stationary regime. To form the dataset, we downsize each trajectory to $\mathbb{R}^{128\times128}$. As evaluating the quality of approximations of conditional distributions for high-dimensional systems is an open problem [26], we rely on proxy quantities. To test the distribution of samples, we frame the perfect nonlinear observation $c(u_0) = \sum_i \|\sin(u_0^i) - \sin(u^i)\|^2$, $i \in \{0, 64, 127\}$ as a constraint. This is not a PDE-forecasting task but constrained trajectory generation, or perfect nonlinear data assimilation. In Appendix F, we test other constraints, including conservation of the spatial mean. We observe each of the 256 test trajectories and for each method we generate an ensemble of 32 trajectories. The system is deterministic, so we expect to observe close agreement with the trajectory the observation was obtained from, i.e., a delta distribution. To judge the quality of the constrained distribution, we present 4 metrics that are best considered together. The root-mean-squared-error (RMSE) measures how far members of the ensemble are from the ground truth, on average. The skill measures how far the ensemble mean is from the ground truth, that is, the bias of the ensemble. The spread measures the deviation of the ensemble from said mean, i.e., how close it is to the delta. Lastly, the continuous ranked probability score (CRPS) penalizes both bias and misrepresentation of uncertainty (too much or too little spread). See Appendix A for their definitions.

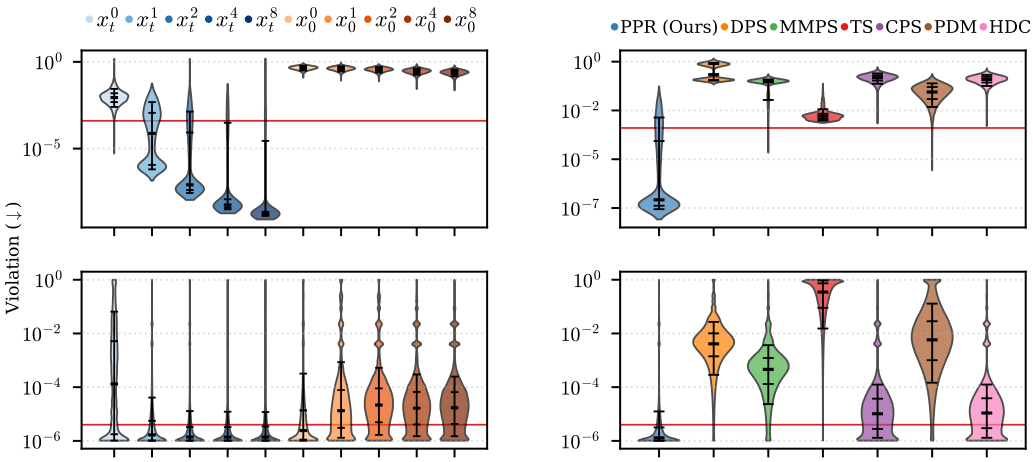
PPR ablation. Proposition 3.1 indicates that renoising and repeated kernel application are fundamental requirements in approximating the constrained marginals and we argue that using the denoiser in the projection loop has key benefits. We justify the claims on KS and DATA2D by carefully ablating each component and showing that without any one of them performance degrades dramatically. To that end, we introduce a PPR variant that does not use the denoiser in the projection. In the variant, labeled x_0^M in the figures, we project x_0 directly (x_0-space), without the denoiser, which is only used to get the initial $x_0 = d_\theta(x_t)$. That makes the projection $\Pi(x_0) = \arg\min_x c(x)$ and the transition kernel $K_t^{\text{var}}(x_t'|x_t) = p\big(x_t'|\Pi(x_0)\big)$. For PPR and the variant, we study no renoising ($M = 0$), only renoising ($M = 1$) and renoising plus reapplication of the K_t^{PPR} kernel ($M > 1$).

We run the ablations on KS and DATA2D, confirming the necessity of the projection through the denoiser, renoising, and multiple kernel applications. Figures 2 and 3a show that not using the denoiser (in orange) results in samples that score worse on all metrics and have much higher constraint violation ($x_0 \notin \mathcal{C}$), with the number of kernel applications doing little to improve the situation. The Max-Norm shows that the KS samples are not in $\mathcal{S}$ as they present sharp discontinuities. The advantage of the denoiser (in blue) is clear, the samples remain in $\mathcal{S}$ while improving all metrics and the constraint violation, achieving $x_0 \in \mathcal{C}$. Increasing the number of kernel applications M consistently improves the samples at a linear computational cost. No renoising, i.e. $M = 0$ is omitted from the KS subfigures as it diverged, supporting the necessity of $M > 0$. These conclusions are confirmed on DATA2D. x_t-space variants achieve k-NN cross-edge rates closer to 0.5 and lower constraint violation. $M = 0$ fails catastrophically but the constraint violation improves as M increases. On this low dimensional distribution, increasing $M > 1$ quickly saturates (see Appendix C), resulting in diminishing returns that do not appear in the higher dimensional settings.

Baselines comparison. We now compare PPR to other constrained sampling methods. The baselines are four explicit constrained samplers (TS, CPS, PDM, and HDC) and two general posterior samplers (DPS and MMPS), plus DiffusionPDE on KS. All are training-free and, within each experiment, all use the same learned model, isolating the effect of the sampler. Figures 3b and 4, and Tables 2 and 3 show that PPR has the highest feasibility rate, the lowest constraint violation, and produces a

Table 2: Data2D comparison. The feasibility threshold is 4×10^{-6}. The k-NN rate is better closer to 0.5.

Method	Time (s)	Feas. ↑	k-NN
PPR (Ours)	14.9 ± 0.6	**79.0%**	$\mathbf{0.375 \pm 0.026}$
DPS	2.6 ± 0.2	1.0%	0.279 ± 0.046
MMPS	11.7 ± 0.4	3.6%	0.340 ± 0.034
Trust	3.8 ± 0.2	0.1%	0.124 ± 0.027
CPS	5.9 ± 0.4	31.6%	0.373 ± 0.030
PDM	10.8 ± 0.1	1.4%	0.002 ± 0.000
HDC	3.5 ± 0.1	30.3%	$\mathbf{0.375 \pm 0.030}$



(a) We vary the number of kernel applications M (x^M) and whether information is propagated through the denoiser (x_t^M), or not (x_0^M). Projecting in x_0-space instead of x_t-space increases constraint violation; on KS, no x_0-space sample lies in $\mathcal{C}$. When projecting through the denoiser in x_t-space, larger M lowers violation.

(b) PPR obtains the lowest constraint violation and highest feasibility %. The % of feasible samples is over 82% for PPR and 8% for the next best method on KS and 79% vs 31% on DATA2D.

Figure 3: Constraint violation for KS (top) and DATA2D (bottom) for the ablation (left) and the baselines comparison (right). Red lines show the thresholds, 4×10^{-4} for KS and 4×10^{-6} for DATA2D. Samples above are considered not in $\mathcal{C}$.

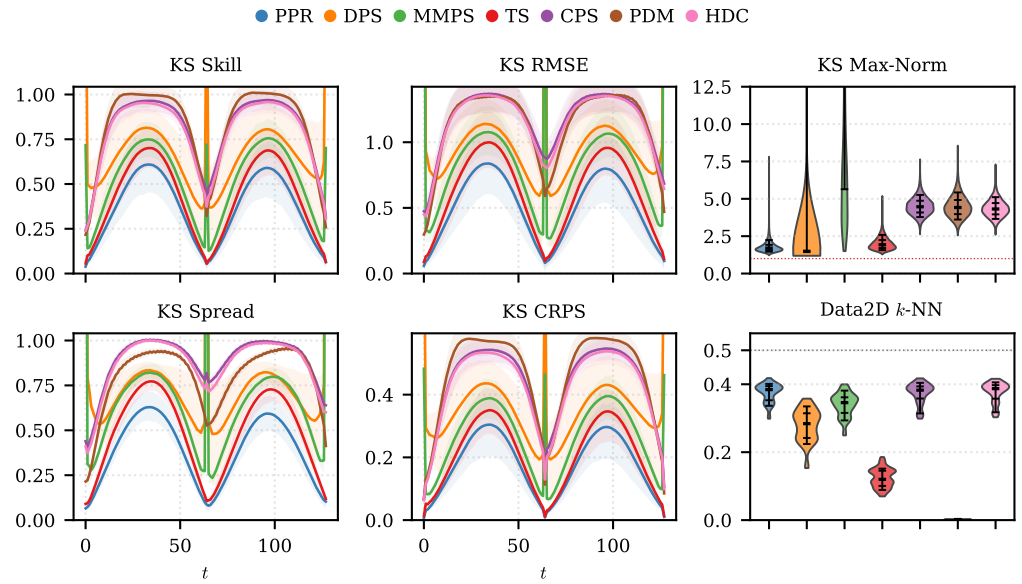


Figure 4: Baseline comparison on KS and DATA2D (bottom right). PPR gives the best constrained distribution in both cases. Only PPR and TS produce samples in $\mathcal{S}$; all other methods yield discontinuous samples. For KS, skill, RMSE, and CRPS measure prediction accuracy; for DATA2D, the k-NN cross-edge rate measures constrained-distribution quality. Max-Norm is better when closer to 1 (dotted red line). In the KS metrics, the x-axis t denotes system time, not diffusion time.

Table 3: Comparison on KS. Only PPR produces feasible samples $x \in \mathcal{C} \cap \mathcal{S}$ at a meaningful rate. The feasibility threshold was set at $c(x) < 4 \times 10^{-4}$. Max-Norm is better the closer it is to 1.

Method	Feas. (%) ↑	RMSE ↓	CRPS ↓	Skill ↓	Spread ↓	Max-Norm	Time (s) ↓
PPR (Ours)	**82.7**	**0.503**	**0.177**	**0.365**	**0.385**	**1.80**	80.2
DPS	0.0	1.089	0.352	0.666	0.865	17.11	25.5
MMPS	1.0	0.830	0.265	0.519	0.672	19.28	307
TS	0.0	0.614	0.206	0.427	0.487	2.01	398
CPS	0.0	1.179	0.436	0.805	0.890	4.50	50.8
PDM	0.8	1.137	0.463	0.831	0.798	4.48	296
HDC	0.0	1.152	0.425	0.787	0.870	4.36	49.8
DiffusionPDE	8.2	1.035	0.374	0.707	0.788	2.80	91.5

closer approximation to the constrained distribution as measured by the Skill, RMSE, and the k-NN cross-edge rate on DATA2D and CRPS on KS. PDM, CPS, and HDC, working in x_0-space, fail to produce $x_0 \in \mathcal{S}$ as the KS samples are discontinuous. The percentage of feasible samples is over 82% for PPR vs 8% for the next best method on KS and 79% vs 31% on DATA2D.

Cost is a secondary objective: a sample that violates the constraint or leaves the data distribution is useless. The cost is tunable through the number of diffusion steps, project-renoise applications, and inner projection iterations. All methods used the same computational budget. However, many baselines do not offer an effective way to trade compute for improvements. For DPS and MMPS, the main lever is the number of diffusion steps, which quickly hit diminishing returns. CPS, PDM, and HDC also expose the number of iterations in their projection routines, but these x_0-space projections converge quickly, often to points outside $\mathcal{S}$, so additional iterations bring little benefit. In TS, the trust-region mechanism can terminate projection prematurely and even with a large region its normalized projection is less effective than the off-the-shelf solvers used by PPR, leading to wasted computation.

Large-scale latent weather. In a setting similar to KS, we treat a perfect nonlinear observation as a constraint. We use Appa [3], a score-based model that generates global atmospheric trajectories at $0.25°$ resolution and 1-hour intervals trained on ERA5 [15]. One atmospheric state has shape $1440 \times 721 \times 71$, which is $\mathcal{O}(10^8)$ elements $\approx$ 300MB. The decoder $\mathcal{D}$, through which we define the constraint, has 170M parameters. Appa's denoiser generates 24-hour trajectories, from which we observe 12:00 PM through the decoder. This results in the constraint $c(x_0) = \|\mathcal{D}(x_0^{12}) - y\|_2^2$, where y is the value of the observation, extracted from ERA5. Figure 5 shows the results of the constrained sampling across 4 test dates. For each method, an ensemble of size 32 was sampled under the same computational budget and wall runtime. The computational budget is the same as [3], and was not optimized. PPR outperforms all baselines: it obtains lower constraint violation and its ensemble trajectories remain closer to the true trajectory for longer. See Appendix D and G for more results.

6 Conclusion

To address the problem of constrained sampling in score-based models, we defined a constrained forward process with its corresponding reverse process, identified a transition kernel whose unique stationary distribution is that constrained marginal, and introduced PPR as an approximation of that kernel. Over a series of experiments, we showed that PPR achieves a higher feasibility rate, lower constraint violation, and closer approximation to the constrained distribution compared to the state of the art and that each of the components of PPR, the projection through the denoiser, the renoising, and the reapplication of the kernel, is fundamental to its performance. Additionally, PPR offers a meaningful way of trading compute for sample and constrained distribution quality.

Limitations. We rely on approximations to sample $p_t^{\mathcal{C}}(x_0|x_t)$ (see Appendix C for the error bound). PPR relies on the differentiability of the constraint function $c(x)$ to propagate information from x_0 to x_t. PPR applies to differentiable constraints whose projection is numerically tractable, and does not assume a closed-form or domain-specific projection. Highly discontinuous, discrete, combinatorial, or black-box constraints are outside the current scope. To extend PPR to another domain, one needs a pretrained denoiser and a differentiable c that measures violation of the desired property, for example engineering design requirements or geometric constraints, although we have not evaluated

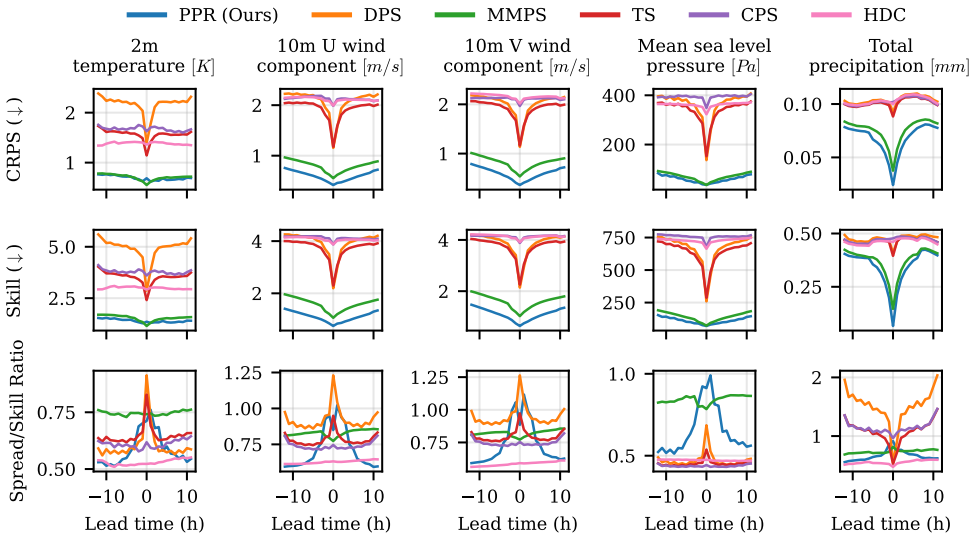


Figure 5: Comparison on Appa across 4 test dates. PPR outperforms all baselines on all variables. ERA5 surface variables shown. The metrics were computed with WeatherBench2 [34].

these applications. PPR also has a computational overhead, as each reverse step requires repeated projections through the denoiser and renoising. This cost grows linearly with M and can be traded against feasibility by tuning M and the number of projection steps.

Future work. A possible way of reducing the computational cost of PPR is to only project at a subset of diffusion steps. While we preliminarily found that it gives poorer constraint satisfaction and constrained distribution approximation, it is an additional lever to be tuned by the user. Further, extending beyond equality constraints to mixed and inequality constraints is a promising direction. Inequality constraints could be handled by a projection with momentum or Langevin projection that stops once inside $\mathcal{C}$. Extending the framework to explicitly handle discrete constraints and domains such as inverse folding in constrained protein design also remains open for further research.

Acknowledgments and Disclosure of Funding

This work was partially supported by ServicePublic de Wallonie Recherche under grant n° 2010235 – ARIAC by DIGITALWALLONIA4.AI. Computational resources have been provided by the Consortium des Équipements de Calcul Intensif (CÉCI), funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11 and by the Walloon Region. The present research benefited from computational resources made available on Lucia, the Tier-1 supercomputer of the Walloon Region, infrastructure funded by the Walloon Region under the grant agreement n°1910247.

References

[1] Robert J. Adler and Jonathan E. Taylor. Gaussian fields. In *Random Fields and Geometry*, Springer Monographs in Mathematics. Springer, New York, NY, 2007. ISBN 978-0-387-48112-8.

[2] Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982. ISSN 0304-4149.

[3] Gérôme Andry, Sacha Lewin, François Rozet, Omer Rochman, Victor Mangeleer, Matthias Pirlet, Elise Faulx, Marilaure Grégoire, and Gilles Louppe. Appa: Bending weather dynamics with latent diffusion models for global data assimilation, 2025.

[4] Chaoran Cheng, Boran Han, Danielle C. Maddix, Abdul Fatir Ansari, Andrew Stuart, Michael W. Mahoney, and Yuyang Wang. Gradient-free generation for hard-constrained systems, 2025.

[5] Jacob K Christopher, Stephen Baek, and Ferdinando Fioretto. Constrained synthesis with projected diffusion models, 2024.

[6] Jacob K. Christopher, Austin Seamann, Jingyi Cui, Sagar Khare, and Ferdinando Fioretto. Constrained diffusion for protein design with hard structural constraints, 2026.

[7] Hyungjin Chung, Jeongsol Kim, Michael T. Mccann, Marc L. Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems, 2024.

[8] Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints, 2024.

[9] Giannis Daras, Hyungjin Chung, Chieh-Hsin Lai, Yuki Mitsufuji, Jong Chul Ye, Peyman Milanfar, Alexandros G. Dimakis, and Mauricio Delbracio. A survey on diffusion models for inverse problems, 2024.

[10] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

[11] Bradley Efron. Tweedie's Formula and Selection Bias. *Journal of the American Statistical Association*, 2011.

[12] Vincent Fortin, Mabrouk Abaza, Francois Anctil, and Raphael Turcotte. Why should ensemble spread match the rmse of the ensemble mean? *Journal of Hydrometeorology*, 15(4):1708–1713, 2014.

[13] Alexandre Galashov, Ashwini Pokle, Arnaud Doucet, Arthur Gretton, Mauricio Delbracio, and Valentin De Bortoli. Learn to guide your diffusion model, 2025.

[14] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

[15] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, Adrian Simmons, Cornel Soci, Saleh Abdalla, Xavier Abellan, Gianpaolo Balsamo, Peter Bechtold, Gionata Biavati, Jean Bidlot, Massimo Bonavita, Giovanna De Chiara, Per Dahlgren, Dick Dee, Michail Diamantakis, Rossana Dragani, Johannes Flemming, Richard Forbes, Manuel Fuentes, Alan Geer, Leo Haimberger, Sean Healy, Robin J. Hogan, Elías Hólm, Marta Janisková, Sarah Keeley, Patrick Laloyaux, Philippe Lopez, Cristina Lupu, Gabor Radnoti, Patricia de Rosnay, Iryna Rozum, Freja Vamborg, Sebastien Villaume, and Jean-Noël Thépaut. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020.

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[17] Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. DiffusionPDE: Generative PDE-Solving Under Partial Observation, 2024.

[18] William Huang, Yifeng Jiang, Tom Van Wouwe, and C. Karen Liu. Constrained diffusion with trust sampling, 2024.

[19] Vivek Jayaram, Ira Kemelmacher-Shlizerman, Steven M. Seitz, and John Thickstun. Constrained diffusion implicit models, 2024.

[20] Vivek Jayaram, Ira Kemelmacher-Shlizerman, Steven M. Seitz, and John Thickstun. Linearly constrained diffusion implicit models, 2025.

[21] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022.

[22] Shervin Khalafi, Dongsheng Ding, and Alejandro Ribeiro. Constrained diffusion models via dual training, 2024.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[24] Yoshiki Kuramoto. Diffusion-induced chaos in reaction systems. *Progress of Theoretical Physics Supplement*, 64:346–367, 02 1978. ISSN 0375-9687.

[25] Chieh-Hsin Lai, Yang Song, Dongjun Kim, Yuki Mitsufuji, and Stefano Ermon. The principles of diffusion models, 2025.

[26] Julia Linhart, Alexandre Gramfort, and Pedro L. C. Rodrigues. L-c2st: Local diagnostics for posterior approximations in simulation-based inference. In *Advances in Neural Information Processing Systems (NeurIPS 2023)*, volume 36, 2023.

[27] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(1–3):503–528, August 1989. ISSN 0025-5610.

[28] Guan-Horng Liu, Tianrong Chen, Evangelos A. Theodorou, and Molei Tao. Mirror diffusion models for constrained and watermarked generation, 2024.

[29] Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for llm training, 2025.

[30] Xiangming Meng and Yoshiyuki Kabashima. Diffusion model based posterior sampling for noisy linear inverse problems, 2024.

[31] Sean Meyn, Richard L. Tweedie, and Peter W. Glynn. *Markov Chains and Stochastic Stability*. Cambridge Mathematical Library. Cambridge University Press, 2 edition, 2009.

[32] Saeid Naderiparizi, Xiaoxuan Liang, Berend Zwartsenberg, and Frank Wood. Constrained generative modeling with manually bridged diffusion models, 2025.

[33] Sai Shankar Narasimhan, Shubhankar Agarwal, Litu Rout, Sanjay Shakkottai, and Sandeep P. Chinchali. Constrained posterior sampling: Time series generation with hard constraints, 2025.

[34] Stephan Rasp, Stephan Hoyer, Alexander Merose, Ian Langmore, Peter Battaglia, Tyler Russel, Alvaro Sanchez-Gonzalez, Vivian Yang, Rob Carver, Shreya Agrawal, Matthew Chantry, Zied Ben Bouallegue, Peter Dueben, Carla Bromberg, Jared Sisk, Luke Barrington, Aaron Bell, and Fei Sha. Weatherbench 2: A benchmark for the next generation of data-driven global weather models, 2024.

[35] Omer Rochman and Gilles Louppe. Enforcing governing equation constraints in neural pde solvers via training-free projections, 2025.

[36] François Rozet, Gérôme Andry, François Lanusse, and Gilles Louppe. Learning diffusion priors from observations by expectation maximization, 2025.

[37] Mark F. Schilling. Multivariate two-sample tests based on nearest neighbors. *Journal of the American Statistical Association*, 81(395):799–806, 1986.

[38] G.I. Sivashinsky. Nonlinear analysis of hydrodynamic instability in laminar flames—i. derivation of basic equations. *Acta Astronautica*, 4(11):1177–1206, 1977. ISSN 0094-5765.

[39] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[40] Bowen Song, Soo Min Kwon, Zecheng Zhang, Xinyu Hu, Qing Qu, and Liyue Shen. Solving inverse problems with latent diffusion models via hard data consistency, 2024.

[41] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

[42] Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2023.

[43] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021.

[44] Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Cambridge University Press, 2019.

[45] M. C. K. Tweedie. Functions of a statistical variate with given means, with special reference to laplacian distributions. *Mathematical Proceedings of the Cambridge Philosophical Society*, 1947.

[46] Jiachen Yao, Abbas Mammadov, Julius Berner, Gavin Kerrigan, Jong Chul Ye, Kamyar Azizzadenesheli, and Anima Anandkumar. Guided Diffusion Sampling on Function Spaces with Applications to PDEs, 2025.

[47] Haotian Ye, Haowei Lin, Jiaqi Han, Minkai Xu, Sheng Liu, Yitao Liang, Jianzhu Ma, James Zou, and Stefano Ermon. Tfg: Unified training-free guidance for diffusion models, 2024.

[48] Stefano Zampini, Jacob K. Christopher, Luca Oneto, Davide Anguita, and Ferdinando Fioretto. Training-free constrained generation with stable diffusion models, 2025.

[49] Zihan Zhou, Xiaoxue Wang, and Tianshu Yu. Generating Physical Dynamics under Priors, 2025.

A Metrics

We use the same metrics as in Andry et al. [3], and partially reproduce their text for the definitions here. The evaluation is performed using WeatherBench2 [34] for Appa data. For the KS experiments, the metrics are the same, but with one spatial dimension removed.

Skill. Skill is computed as the root mean square error of the posterior mean of an ensemble compared to the ground-truth trajectory. For K ensembles each consisting of M predicted states $\hat{x}$ of resolution $H \times W$, ground truth x, the skill of a single time step is computed as

$$\text{Skill} = \sqrt{\frac{1}{KHW} \sum_{k=1}^{K} \sum_{i=1}^{H} \sum_{j=1}^{W} \left(x_{i,j}^{k} - \frac{1}{M} \sum_{m=1}^{M} \hat{x}_{i,j}^{k,m} \right)^2}.$$

Spread. Ensemble spread is computed as the square root of the ensemble variance [12]:

$$\text{Spread} = \sqrt{\frac{1}{KHW} \sum_{k=1}^{K} \sum_{i=1}^{H} \sum_{j=1}^{W} \frac{1}{M-1} \sum_{m=1}^{M} \left(\hat{x}_{i,j}^{k,m} - \frac{1}{M} \sum_{n=1}^{M} \hat{x}_{i,j}^{k,n} \right)^2}.$$

Spread-skill ratio. A well-calibrated forecast should have a (corrected for ensemble size) spread-skill ratio of 1, which is a necessary but not sufficient condition. Ratios below one indicate overconfident estimations. The correct ratio is defined as

$$\text{Ratio} = \sqrt{\frac{M+1}{M}} \frac{\text{Spread}}{\text{Skill}}.$$

Continuous ranked probability score (CRPS). The CRPS [14] is defined as

$$\text{CRPS} = \frac{1}{K} \sum_{k=1}^{K} \left(\frac{1}{M} \sum_{m=1}^{M} ||\hat{x}^{k,m} - x^{k}||_{L_1} - \frac{1}{2M(M-1)} \sum_{m=1}^{M} \sum_{n=1}^{M} ||\hat{x}^{k,m} - \hat{x}^{k,n}||_{L_1} \right).$$

The first term penalizes the average divergence from the ground truth, while the second term encourages spread. Therefore, the CRPS is lowest when the distribution of the ensemble matches the ground-truth distribution.

B Data

B.1 DATA2D

We generated i.i.d. samples in $\mathbb{R}^2$ from two parametric distributions: a uniform checkerboard distribution and a "banana" Gaussian mixture. Unless otherwise stated, reported samples are standardized by using an empirically estimated mean $\mu \in \mathbb{R}^2$ and standard deviation $\sigma \in \mathbb{R}^2$, i.e.,

$$\tilde{x} \;=\; \frac{(x-\mu)}{\sigma}.$$

The normalization statistics are estimated from Monte Carlo samples drawn from the *non-standardized* distribution.

Checkerboard. Fix an integer grid size m and a jitter scale $\eta \geq 0$. We draw $i \sim \text{Unif}\{0, \ldots, m-1\}$ and then draw j uniformly from $\{0, \ldots, m-1\}$ subject to the parity constraint $(i+j) \bmod 2 = 0$ (equivalently, points occupy alternating squares). Given (i, j), we draw $u_1, u_2 \sim \mathcal{U}(0,1)$ independently and set

$$x \;=\; \begin{bmatrix} i+u_1 \\ j+u_2 \end{bmatrix}.$$

If $\eta > 0$, we add isotropic Gaussian jitter, $x \leftarrow x + \eta\,\xi$ with $\xi \sim \mathcal{N}(0, I_2)$.

Banana Gaussian mixture model (Banana-GMM). Let $\{w_k\}_{k=1}^K$ be nonnegative mixture weights with $\sum_k w_k = 1$, and let $\mu_k \in \mathbb{R}^2$ and $s_k \in \mathbb{R}^2_+$ be component means and diagonal standard deviations. We first sample a component index $k \sim \mathrm{Categorical}(w_1, \ldots, w_K)$ and then draw a Gaussian sample

$$z \sim \mathcal{N}\big(\mu_k,\ \mathrm{diag}(s_k^2)\big), \qquad z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}.$$

We then apply a "banana" shear in the second coordinate,

$$x_1 = z_1, \qquad x_2 = z_2 + b\big(z_1^2 - \mathbb{E}[z_1^2]\big),$$

where $b \in \mathbb{R}$ controls the curvature and the centering term $\mathbb{E}[z_1^2]$ is computed analytically under the mixture,

$$\mathbb{E}[z_1^2] \ = \ \sum_{k=1}^K w_k \big((\mu_k)_1^2 + (s_k)_1^2\big).$$

GRF Constraints. We generated a collection of 12 independent scalar constraint functions on $\mathbb{R}^2$ as random draws from an approximate Gaussian Random Field (GRF) [1] with an RBF kernel, using random Fourier features. All constraints are defined by the same hyperparameters. Fix the number of features M (here $M = 64$), lengthscale $\ell > 0$ (here $\ell = 0.25$ in each input dimension), kernel variance σ_f^2 (here $\sigma_f^2 = 1$), and an optional additive bias. For each constraint, we sample frequencies and phases

$$\omega_m \sim \mathcal{N}\big(0, \ell^{-2} I\big), \qquad \phi_m \sim \mathcal{U}(0, 2\pi), \qquad m = 1, \ldots, M,$$

independently, and we sample coefficients

$$a_m \sim \mathcal{N}(0, 1), \qquad m = 1, \ldots, M.$$

We define the random feature map for $x \in \mathbb{R}^2$ as

$$\varphi_m(x) \ = \ \cos(\omega_m^\top x + \phi_m),$$

and the corresponding random function

$$f(x) \ = \ \sqrt{\frac{2\sigma_f^2}{M}} \sum_{m=1}^M a_m \, \varphi_m(x) \ + \ \beta,$$

where the bias term is sampled as

$$\beta \sim \mathcal{N}\big(0,\ \sigma_b^2\, \sigma_f^2\big),$$

with $\sigma_b = 0.05$. In the limit $M \to \infty$, this construction converges (in distribution) to a zero-mean GP with an RBF kernel of lengthscale ℓ and marginal variance σ_f^2. Each constraint is obtained by smoothing $f(x)$,

$$c(x) \ = \ 1 - \ e^{-f(x)^2}.$$

We formed the 12 constraints $\{c_i\}_{i=1}^{12}$ by sampling $(\{\omega_m, \phi_m, a_m\}_{m=1}^M, \beta)$ independently for each i. We used a tolerance $\epsilon = 10^{-2}$ for the creation of the constrained distributions.

B.2 KURAMOTO-SIVASHINSKY

The 1-D Kuramoto–Sivashinsky Eq. [24, 38]

$$\partial_t u + \partial_{xx} u + \partial_{xxxx} u + u\, \partial_x u = 0$$

is solved on the periodic domain $x \in [0, 64]$ with a pseudospectral solver. The spatial derivatives use the FFT on the grid of size $N = 512$ and no de-aliasing. Time stepping was done using the first-order backward-difference formula (BDF1) with fixed step $\mathrm{d}t = 10^{-1}$ for 1024 steps, giving $t_{\max} = 102.4$. During training, we used a subset of each solution, starting at step 512 and ending at step 1024, corresponding to $t \in [51.2, 102.4]$. Each implicit step is solved by Newton iterations. Initial data are random sums of ten cosine modes,

$$u_0(x) = \sum_{k=1}^{10} a_k \cos\big(2\pi\omega_k x/64 + \phi_k\big),$$

where amplitudes $a_k \sim \mathcal{U}(0, 1)$, frequencies $\omega_k \sim \mathcal{U}(1, 5)$ and phases $\phi_k \sim \mathcal{U}(0, 2\pi)$ are sampled independently. The dataset comprises 65,536 training and 128 validation trajectories, each generated with an independent initial condition. The test trajectories for Section 5 were generated with the unconditional diffusion model.

B.3 Architecture

Both architectures use SiLU activations, sinusoidal noise-level embeddings, and EDM-style preconditioning.

Data2D uses a Modulated MLP: a 3-hidden-layer MLP (256 units per layer) with residual connections. The noise level σ and the input coordinates $x \in \mathbb{R}^2$ and their norm are each encoded by separate sinusoidal embeddings and injected into every hidden layer via FiLM modulation.

Kuramoto-Sivashinsky (KS) uses a UNet with channel progression $[32, 64, 128, 256]$. Each scale has one FiLM-modulated residual block (3×3 convolutions) and a strided convolution (encoder) or transposed convolution (decoder) for downsampling and upsampling. A small MLP lifts the 1-channel input to the base feature width and projects back at the output. Circular padding is applied in the spatial direction; asymmetric zero-padding handles the (non-periodic) time dimension.

The noise schedule is given by

$$\sigma_t = \exp\left(s \ \mathrm{logit}\left(t(1-2\varepsilon)+\varepsilon\right) + \log \sigma_{\mathrm{med}}\right), \tag{10}$$

$$\varepsilon = \left(\sqrt{\frac{\sigma_{\min}}{\sigma_{\max}}}\right)^{1/\alpha}, \quad \sigma_{\mathrm{med}} = \sqrt{\sigma_{\min}\sigma_{\max}}, \tag{11}$$

and a constant $\alpha_t = 1$ scale schedule. This schedule was inspired by Andry et al. [3].

Table 4: Architecture and noise-schedule parameters for KS and Data2D.

	Data2D	**KS**
Architecture	Mod MLP	UNet
Depth / channels	3 layers, 256 units	[32, 64, 128, 256]
Activation	SiLU	SiLU
Time conditioning	FiLM (per layer)	FiLM (per layer)
Kernel size	—	3×3
Spatial padding	—	circular
Noise schedule	log logit	
$\sigma_{\min}$	10^{-3}	10^{-4}
$\sigma_{\max}$	10^{2}	10^{4}
Spread s	2.0	4.0

B.4 Training

DATA2D and KS were trained for 300 epochs using the Muon optimizer [29] with no weight decay and $k = 1$ optimizer step per update. DATA2D used a batch size of 1024, a peak learning rate of 10^{-3}, no warmup, cosine decay over 101×5000 steps, and a final learning rate of 10^{-6}. KS used a batch size of 64, a peak learning rate of 10^{-4}, 100 warmup steps, cosine decay over 100,000 steps, and the same final learning rate of 10^{-6}. All models were trained on a single NVIDIA A5000-24GB. The KS model trained overnight while the DATA2D model trained for 4 h.

B.5 Appa

Appa was trained on ERA5 [15]. We refer the reader to Andry et al. [3] for details, including training and compute.

C Theoretical analysis and approximation error bound

In this section we restate Proposition 3.1 and prove it. We also bound the error of the PPR approximation, and justify the assumptions in that bound. Both propositions assume bounded second

moment for all distributions, a nondegenerate constrained set $0 < \mathbb{P}(x_0 \in \mathcal{C}) < 1$, and well-defined conditional constrained posteriors $\mathbb{P}(x_0 \in \mathcal{C} \mid x_t) > 0$ on the relevant support.

Proposition C.1. *Let* $K_t(x_t'|x_t) = \int_{\mathcal{C}} p(x_t'|x_0)\, p^{\mathcal{C}}(x_0|x_t)\, dx_0$ *with* $p^{\mathcal{C}}(x_0|x_t) = \frac{p(x_0|x_t)\, \mathbb{1}_{\mathcal{C}}(x_0)}{\mathbb{P}(x_0 \in \mathcal{C}|x_t)}$. *This kernel connects the unconstrained* $p_t(x_t)$ *and constrained* $p_t^{\mathcal{C}}(x_t)$ *priors at time* t. *Under this kernel,* $p_t^{\mathcal{C}}$ *is the unique stationary distribution and* $(K_t)^M q \to p_t^{\mathcal{C}}$ *as* $M \to \infty$ *for any initial distribution* q.

Sketch. The kernel K_t can be viewed as a two-step resampling procedure: starting from the current noisy variable x_t, it first samples a clean point x_0 from the posterior restricted to the constraint set $\mathcal{C}$, and then renoises this x_0 to obtain a new x_t'. If the current x_t is already distributed according to the constrained noisy marginal $p_t^{\mathcal{C}}$, then this posterior-resampling step recovers exactly the constrained clean distribution, and the renoising step maps it back to $p_t^{\mathcal{C}}$. This shows stationarity. Since the renoising distribution is Gaussian, the chain can move between any regions of the state space with positive probability, which rules out multiple closed classes or periodic behavior. Therefore the stationary distribution is unique, and repeated applications of the kernel converge to $p_t^{\mathcal{C}}$.

Proof. Fix $t > 0$ and write

$$p_t^{\mathcal{C}}(x_t) = \frac{1}{Z^{\mathcal{C}}} \int_{\mathcal{C}} p(x_t|x_0) p(x_0)\, dx_0, \qquad Z^{\mathcal{C}} = \mathbb{P}(x_0 \in \mathcal{C}).$$

We first show that $p_t^{\mathcal{C}}$ is stationary for K_t, namely

$$\int K_t(x_t'|x_t)\, p_t^{\mathcal{C}}(x_t)\, dx_t = p_t^{\mathcal{C}}(x_t').$$

By Bayes' rule,

$$p(x_0|x_t) = \frac{p(x_t|x_0)p(x_0)}{p_t(x_t)}.$$

Moreover,

$$\mathbb{P}(x_0 \in \mathcal{C}|x_t) = \int_{\mathcal{C}} p(x_0|x_t)\, dx_0 = \frac{\int_{\mathcal{C}} p(x_t|x_0)p(x_0)\, dx_0}{p_t(x_t)} = \frac{Z^{\mathcal{C}} p_t^{\mathcal{C}}(x_t)}{p_t(x_t)}.$$

Hence, for $x_0 \in \mathcal{C}$,

$$p^{\mathcal{C}}(x_0|x_t) = \frac{p(x_0|x_t)\mathbb{1}_{\mathcal{C}}(x_0)}{\mathbb{P}(x_0 \in \mathcal{C}|x_t)} = \frac{p(x_t|x_0)p(x_0)}{Z^{\mathcal{C}} p_t^{\mathcal{C}}(x_t)}.$$

Substituting this expression into the stationarity equation gives

$$\begin{aligned}
\int K_t(x_t'|x_t) p_t^{\mathcal{C}}(x_t)\, dx_t &= \int \int_{\mathcal{C}} p(x_t'|x_0) p^{\mathcal{C}}(x_0|x_t)\, dx_0\, p_t^{\mathcal{C}}(x_t)\, dx_t \\
&= \int \int_{\mathcal{C}} p(x_t'|x_0) \frac{p(x_t|x_0)p(x_0)}{Z^{\mathcal{C}} p_t^{\mathcal{C}}(x_t)}\, dx_0\, p_t^{\mathcal{C}}(x_t)\, dx_t \\
&= \frac{1}{Z^{\mathcal{C}}} \int \int_{\mathcal{C}} p(x_t'|x_0) p(x_t|x_0) p(x_0)\, dx_0\, dx_t \\
&= \frac{1}{Z^{\mathcal{C}}} \int_{\mathcal{C}} p(x_t'|x_0) p(x_0) \left(\int p(x_t|x_0)\, dx_t \right) dx_0 \\
&= \frac{1}{Z^{\mathcal{C}}} \int_{\mathcal{C}} p(x_t'|x_0) p(x_0)\, dx_0 \\
&= p_t^{\mathcal{C}}(x_t').
\end{aligned}$$

Thus $p_t^{\mathcal{C}}$ is stationary under K_t.

It remains to justify uniqueness and convergence. Since $\sigma_t > 0$, the renoising density

$$p(x_t' \mid x_0) = \mathcal{N}(x_t'; \alpha_t x_0, \sigma_t^2 I)$$

is strictly positive for all x'_t. Hence $K_t(x'_t \mid x_t) > 0$, for all x_t, x'_t. Therefore K_t is Lebesgue-irreducible and aperiodic. Since $p_t^{\mathcal{C}}$ is an invariant probability measure, standard general-state-space Markov chain theory [31] implies that K_t is positive Harris recurrent, that $p_t^{\mathcal{C}}$ is the unique stationary distribution, and that for every initial distribution q,

$$(K_t)^M q \to p_t^{\mathcal{C}} \qquad \text{as } M \to \infty$$

for any initial distribution q in the total variation sense. This proves the claim. □

Proposition C.2. *Assume $x_t \mapsto d_\theta(\Pi_d(x_t))$ is L_t-Lipschitz and that $\alpha_t L_t < 1$. Then*

$$W_2\Big(q_t^{(M)}, p_t^{\mathcal{C}}\Big) \le (\alpha_t L_t)^M \, W_2\Big(q_t^{(0)}, \tilde{p}_t^{\mathcal{C}}\Big) + \frac{\Delta_t}{1 - \alpha_t L_t},$$

where

$$\Delta_t = \sup_{x_t} W_2\big(K_t^{\mathrm{PPR}}(\,\cdot \mid x_t), K_t^*(\,\cdot \mid x_t)\big)\,.$$

Sketch. The error is split into two effects: the finite number of PPR iterations and the fact that the PPR kernel is only an approximation of the ideal constrained kernel. The first effect is controlled by showing that the PPR update is contractive: two nearby noisy inputs are mapped to Gaussian transitions whose means remain closer by a factor at most $\alpha_t L_t < 1$. Hence repeated PPR steps forget the initialization geometrically fast. The second effect is the limiting bias caused by replacing the ideal kernel with the approximate one. This bias is controlled by the largest one-step discrepancy Δ_t between the two kernels. Because the approximate kernel is contractive, these one-step errors do not accumulate without bound; instead they are amplified by at most the geometric factor $1/(1 - \alpha_t L_t)$. Combining the geometric mixing error with this accumulated approximation bias gives the stated bound.

Proof. Let

$$\rho_t := \alpha_t L_t.$$

By assumption, $\rho_t < 1$. We decompose

$$W_2\Big(q_t^{(M)}, p_t^{\mathcal{C}}\Big) \le W_2\Big(q_t^{(M)}, \tilde{p}_t^{\mathcal{C}}\Big) + W_2\big(\tilde{p}_t^{\mathcal{C}}, p_t^{\mathcal{C}}\big)\,.$$

We first bound the mixing term. The PPR kernel is

$$K_t^{\mathrm{PPR}}(x'_t|x_t) = \mathcal{N}\big(x'_t; \alpha_t d_\theta(\Pi_d(x_t)), \sigma_t^2 I\big)\,.$$

For any $x_t, \bar{x}_t$, the two transition distributions have the same covariance. Hence

$$\begin{aligned} W_2\big(K_t^{\mathrm{PPR}}(\cdot|x_t), K_t^{\mathrm{PPR}}(\cdot|\bar{x}_t)\big) &= \alpha_t \, \|d_\theta(\Pi_d(x_t)) - d_\theta(\Pi_d(\bar{x}_t))\| \\ &\le \alpha_t L_t \|x_t - \bar{x}_t\| \\ &= \rho_t \|x_t - \bar{x}_t\|. \end{aligned}$$

Therefore, for any probability measures q_1, q_2,

$$W_2\big(K_t^{\mathrm{PPR}} q_1, K_t^{\mathrm{PPR}} q_2\big) \le \rho_t W_2(q_1, q_2).$$

That is, K_t^{PPR} is a contraction.

Define

$$q_t^{(M)} = \big(K_t^{\mathrm{PPR}}\big)^M \, q_t^{(0)}$$

and let $\tilde{p}_t^{\mathcal{C}}$ be stationary for K_t^{PPR}. It exists since K_t^{PPR} is a contraction. Then repeated application gives

$$W_2\Big(q_t^{(M)}, \tilde{p}_t^{\mathcal{C}}\Big) \le \rho_t^M W_2\Big(q_t^{(0)}, \tilde{p}_t^{\mathcal{C}}\Big)\,.$$

It remains to bound the bias term. Since $\tilde{p}_t^{\mathcal{C}}$ is stationary for K_t^{PPR} and $p_t^{\mathcal{C}}$ is stationary for K_t^*,

$$W_2\big(\tilde{p}_t^{\mathcal{C}}, p_t^{\mathcal{C}}\big) = W_2\big(K_t^{\mathrm{PPR}} \tilde{p}_t^{\mathcal{C}}, K_t^* p_t^{\mathcal{C}}\big)\,.$$

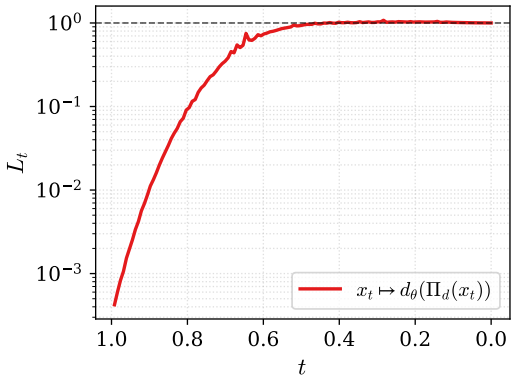


Figure 6: Lipschitz constant of $x_t \mapsto d_\theta(\Pi_d(x_t))$. It was measured numerically by evaluating the Lipschitz constant for each of the 12 constraints across 1024 samples at every timestep, then taking the maximum value.

By the triangle inequality,

$$W_2\big(K_t^{\mathrm{PPR}}\tilde{p}_t^{\mathcal{C}}, K_t^* p_t^{\mathcal{C}}\big) \le W_2\big(K_t^{\mathrm{PPR}}\tilde{p}_t^{\mathcal{C}}, K_t^{\mathrm{PPR}} p_t^{\mathcal{C}}\big) + W_2\big(K_t^{\mathrm{PPR}} p_t^{\mathcal{C}}, K_t^* p_t^{\mathcal{C}}\big)\,.$$

The first term is bounded by the contraction estimate:

$$W_2\big(K_t^{\mathrm{PPR}}\tilde{p}_t^{\mathcal{C}}, K_t^{\mathrm{PPR}} p_t^{\mathcal{C}}\big) \le \rho_t W_2\big(\tilde{p}_t^{\mathcal{C}}, p_t^{\mathcal{C}}\big)\,.$$

For the second term, coupling the same $x_t \sim p_t^{\mathcal{C}}$ and then using an optimal conditional coupling of the two kernels gives

$$\begin{aligned} W_2\big(K_t^{\mathrm{PPR}} p_t^{\mathcal{C}}, K_t^* p_t^{\mathcal{C}}\big) &\le \sup_{x_t} W_2\big(K_t^{\mathrm{PPR}}(\cdot|x_t), K_t^*(\cdot|x_t)\big) \\ &= \Delta_t. \end{aligned}$$

Thus

$$W_2\big(\tilde{p}_t^{\mathcal{C}}, p_t^{\mathcal{C}}\big) \le \rho_t W_2\big(\tilde{p}_t^{\mathcal{C}}, p_t^{\mathcal{C}}\big) + \Delta_t.$$

Since $\rho_t < 1$, rearranging yields

$$W_2\big(\tilde{p}_t^{\mathcal{C}}, p_t^{\mathcal{C}}\big) \le \frac{\Delta_t}{1-\rho_t} = \frac{\Delta_t}{1-\alpha_t L_t}.$$

Combining the mixing and bias bounds,

$$\begin{aligned} W_2\Big(q_t^{(M)}, p_t^{\mathcal{C}}\Big) &\le W_2\Big(q_t^{(M)}, \tilde{p}_t^{\mathcal{C}}\Big) + W_2\big(\tilde{p}_t^{\mathcal{C}}, p_t^{\mathcal{C}}\big) \\ &\le (\alpha_t L_t)^M W_2\Big(q_t^{(0)}, \tilde{p}_t^{\mathcal{C}}\Big) + \frac{\Delta_t}{1-\alpha_t L_t}. \end{aligned}$$

□

Figure 6 shows how the Lipschitz constant L_t evolves over diffusion time, measured by taking the maximum constant from all the constraints and sampling trajectories of DATA2D. The figure supports the assumption that $x_t \mapsto d_\theta(\Pi_d(x_t))$ is L_t-Lipschitz and that $\alpha_t L_t < 1$ as a sensible one. Here $\alpha_t = 1$.

D Detailed results

In this section we provide the detailed numerical results of the experiments from Section 5 in the main text. Table 5 extends Table 3 from the main text with the number of calls to the denoiser and includes other simpler constraints (Appendix F) in the statistics. Tables 6, 7, 8, 9, and 10 give the average over time of the values in Figure 5.

Table 5: KS results. T: diffusion steps. fwd is the number of network call. bwd is the number of gradient calls (adds one to fwd too). The feasibility shows the % of samples under a threshold (0 is worse, 100 is the best). Max-Norm is better close to 1. Note that the feasibility is averaged over different constraints. For the most challenging constraints, the feasibility of PPR is over 84% compared to 8% for the next best method (main text).

Method	T	fwd	bwd	Time (s)	Feas. ↑	Skill ↓	RMSE ↓	CRPS ↓	Spread ↓	MaxNorm
PPR (Ours)	64	1216	1024	84.8 ± 8.1	**86.9%**	$\mathbf{0.648 \pm 0.212}$	$\mathbf{0.891 \pm 0.285}$	$\mathbf{0.351 \pm 0.130}$	$\mathbf{0.644 \pm 0.189}$	$\mathbf{1.65 \pm 0.17}$
DPS	1024	2048	1024	25.6 ± 0.3	11.0%	0.817 ± 0.153	1.240 ± 0.165	0.450 ± 0.099	0.956 ± 0.129	16.64 ± 22.82
MMPS	1024	5120	20480	306 ± 3	42.1%	0.735 ± 0.161	1.069 ± 0.179	0.399 ± 0.101	0.801 ± 0.101	9.08 ± 10.30
Trust	192	24960	12288	405 ± 11	0.0%	0.667 ± 0.181	0.929 ± 0.233	0.358 ± 0.114	0.682 ± 0.144	1.71 ± 0.25
CPS	1024	1024	0	52.6 ± 3.1	36.0%	0.873 ± 0.062	1.242 ± 0.055	0.483 ± 0.042	0.911 ± 0.022	2.82 ± 1.22
PDM	384	24576	0	296 ± 3	56.9%	0.897 ± 0.064	1.211 ± 0.066	0.508 ± 0.043	0.837 ± 0.045	2.97 ± 1.10
HDC	1024	1024	0	49.5 ± 0.9	36.3%	0.865 ± 0.068	1.229 ± 0.063	0.478 ± 0.046	0.902 ± 0.027	2.76 ± 1.16

Table 6: Appa results for 2m temperature [K]. Time is the wallclock to produce 32 samples sequentially. All values are mean ± std across the 24 timesteps.

Method	Time	Skill ↓	CRPS ↓	Spread ↓	Spread/Skill
PPR (Ours)	2h 01m	$\mathbf{1.399 \pm 0.080}$	$\mathbf{0.693 \pm 0.045}$	$\mathbf{0.807 \pm 0.069}$	0.589 ± 0.072
MMPS	2h 05m	1.536 ± 0.134	0.708 ± 0.055	1.129 ± 0.102	$\mathbf{0.746 \pm 0.009}$
TS	1h 53m	3.525 ± 0.295	1.564 ± 0.105	2.238 ± 0.109	0.648 ± 0.041
CPS	1h 26m	3.786 ± 0.104	1.675 ± 0.040	2.274 ± 0.077	0.610 ± 0.017
HDC	1h 26m	2.996 ± 0.053	1.385 ± 0.024	1.562 ± 0.019	0.530 ± 0.012
DPS	2h 15m	4.918 ± 0.542	2.165 ± 0.196	2.892 ± 0.126	0.605 ± 0.072

Table 7: Appa results for 10m U-wind component [m/s]. Time is the wallclock to produce 32 samples sequentially. All values are mean ± std across the 24 timesteps.

Method	Time	Skill ↓	CRPS ↓	Spread ↓	Spread/Skill
PPR (Ours)	2h 01m	$\mathbf{1.104 \pm 0.181}$	$\mathbf{0.599 \pm 0.095}$	$\mathbf{0.755 \pm 0.053}$	0.715 ± 0.136
MMPS	2h 05m	1.581 ± 0.236	0.796 ± 0.108	1.293 ± 0.203	0.829 ± 0.021
TS	1h 53m	3.718 ± 0.374	1.923 ± 0.194	2.858 ± 0.222	0.784 ± 0.044
CPS	1h 26m	4.122 ± 0.067	2.129 ± 0.041	3.002 ± 0.108	0.740 ± 0.026
HDC	1h 26m	4.075 ± 0.065	2.117 ± 0.039	2.518 ± 0.033	0.628 ± 0.011
DPS	2h 15m	3.923 ± 0.461	2.072 ± 0.244	3.537 ± 0.281	$\mathbf{0.924 \pm 0.077}$

Table 8: Appa results for 10m V-wind component [m/s]. Time is the wallclock to produce 32 samples sequentially. All values are mean ± std across the 24 timesteps.

Method	Time	Skill ↓	CRPS ↓	Spread ↓	Spread/Skill
PPR (Ours)	2h 01m	$\mathbf{1.098 \pm 0.224}$	$\mathbf{0.594 \pm 0.117}$	$\mathbf{0.770 \pm 0.061}$	0.741 ± 0.160
MMPS	2h 05m	1.599 ± 0.250	0.815 ± 0.122	1.297 ± 0.219	0.822 ± 0.021
TS	1h 53m	3.722 ± 0.391	1.907 ± 0.196	2.910 ± 0.230	0.798 ± 0.046
CPS	1h 26m	4.137 ± 0.058	2.114 ± 0.033	3.042 ± 0.110	0.747 ± 0.025
HDC	1h 26m	4.140 ± 0.066	2.159 ± 0.045	2.389 ± 0.055	0.586 ± 0.015
DPS	2h 15m	3.854 ± 0.460	2.006 ± 0.233	3.570 ± 0.286	$\mathbf{0.949 \pm 0.079}$

Table 9: Appa results for Mean sea level pressure [Pa]. Time is the wallclock to produce 32 samples sequentially. All values are mean ± std across the 24 timesteps.

Method	Time	Skill ↓	CRPS ↓	Spread ↓	Spread/Skill
PPR (Ours)	2h 01m	$\mathbf{111.186 \pm 23.199}$	$\mathbf{59.473 \pm 13.391}$	$\mathbf{68.678 \pm 5.978}$	0.653 ± 0.141
MMPS	2h 05m	141.364 ± 32.115	69.818 ± 15.414	117.677 ± 27.700	$\mathbf{0.843 \pm 0.024}$
TS	1h 53m	636.298 ± 92.946	337.614 ± 47.335	281.827 ± 36.586	0.452 ± 0.020
CPS	1h 26m	756.719 ± 17.679	393.572 ± 10.043	325.945 ± 8.827	0.437 ± 0.006
HDC	1h 26m	727.634 ± 16.358	361.569 ± 9.251	338.532 ± 7.344	0.472 ± 0.003
DPS	2h 15m	667.376 ± 108.510	361.394 ± 57.688	306.990 ± 38.747	0.474 ± 0.048

Table 10: Appa results for Total precipitation [mm]. Time is the wallclock to produce 32 samples sequentially. All values are mean ± std across the 24 timesteps.

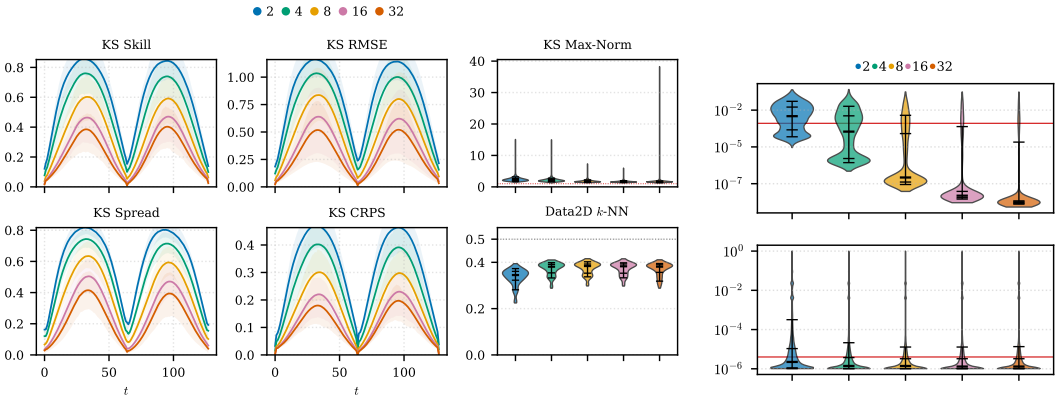
Method	Time	Skill ↓	CRPS ↓	Spread ↓	Spread/Skill
PPR (Ours)	2h 01m	$\mathbf{0.345 \pm 0.084}$	$\mathbf{0.068 \pm 0.014}$	$\mathbf{0.220 \pm 0.042}$	0.666 ± 0.086
MMPS	2h 05m	0.371 ± 0.066	0.075 ± 0.011	0.269 ± 0.047	0.737 ± 0.029
TS	1h 53m	0.464 ± 0.018	0.101 ± 0.004	0.495 ± 0.092	$\mathbf{1.079 \pm 0.190}$
CPS	1h 26m	0.470 ± 0.012	0.102 ± 0.003	0.532 ± 0.047	1.149 ± 0.104
HDC	1h 26m	0.462 ± 0.012	0.103 ± 0.003	0.251 ± 0.017	0.552 ± 0.031
DPS	2h 15m	0.476 ± 0.018	0.104 ± 0.004	0.708 ± 0.150	1.503 ± 0.299

E Additional Ablations

This appendix reports three additional experiments that complement the main-text ablation and baselines comparison: the number of projection steps $\Pi_d(x_t) \equiv \text{PROJECT} = \arg\min_x c(d_\theta(x))$ (Figure 7 and Table 11), the optimizer used for this inner projection problem (Figure 8), and a DPS predictor-corrector variant (DPS-PC) (Figure 9).

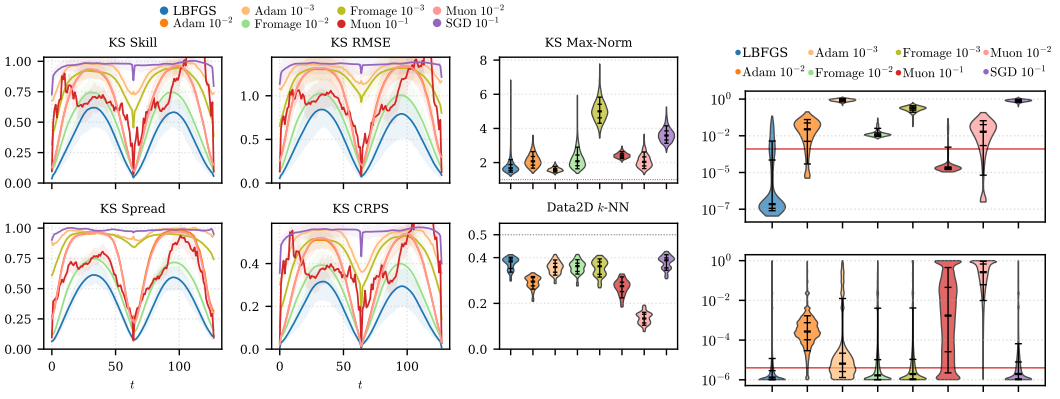
Table 11: Effect of the number of projection steps on PPR performance. Columns as in the main text.

	Data2D			KS					
Projection steps	Feas. (%) ↑	KNN	Time (s) ↓	Feas. (%) ↑	RMSE ↓	CRPS ↓	Skill ↓	Max-Norm	Time (s) ↓
2	62.0	0.336 ± 0.036	9.4 ± 0.4	33.3	1.140 ± 0.171	0.445 ± 0.093	0.825 ± 0.133	1.91 ± 0.44	39.1 ± 2.1
4	76.0	0.371 ± 0.027	10.4 ± 0.5	73.2	1.069 ± 0.218	0.403 ± 0.111	0.770 ± 0.167	1.79 ± 0.31	53.0 ± 2.2
8	**78.5**	$\mathbf{0.374 \pm 0.027}$	14.1 ± 0.8	87.2	1.012 ± 0.605	0.361 ± 0.134	0.711 ± 0.229	1.65 ± 0.29	83.7 ± 6.6
16	78.2	0.373 ± 0.027	21.3 ± 1.2	94.0	0.925 ± 0.340	0.337 ± 0.150	0.666 ± 0.256	$\mathbf{1.60 \pm 0.21}$	150 ± 25
32	78.0	0.370 ± 0.029	35.3 ± 2.0	**95.9**	$\mathbf{0.904 \pm 0.440}$	$\mathbf{0.330 \pm 0.162}$	$\mathbf{0.652 \pm 0.285}$	1.61 ± 0.41	305 ± 97



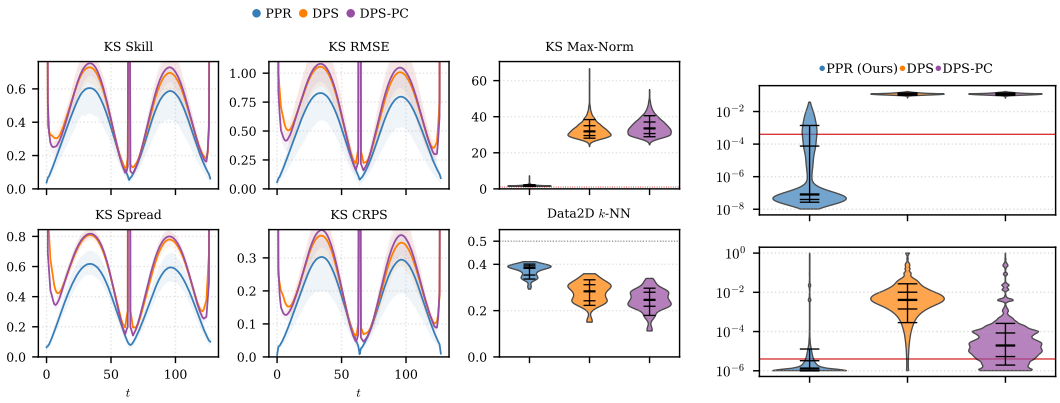
(a) Metrics. (b) Constraint violation.

Figure 7: Projection-step ablation on KS and DATA2D. The labels $2, 4, 8, 16, 32$ denote the number of projection iterations used inside Π_d. More projection iterations improve the local projection solve and reduce violation in both settings. The gains are clearest on KS; on DATA2D, the k-NN cross-edge rate improves quickly and then starts to saturate, consistent with the lower-dimensional projection problem reaching a good local minimum after relatively few iterations.



(a) Metrics. (b) Constraint violation.

Figure 8: Optimizer ablation for the denoiser-space projection problem. We compare L-BFGS, Adam, Fromage, Muon, and SGD with representative learning rates; divergent learning rates have been omitted. L-BFGS is the strongest and most stable optimizer across metrics and violation, while the first-order methods are more sensitive to step size and generally give worse constrained distributions or less regular KS trajectories.



(a) Metrics.

(b) Constraint violation.

Figure 9: DPS-PC ablation. DPS-PC adds DPS-style correction steps in a predictor-corrector scheme. Although this can improve some feasibility-related behavior relative to DPS, it does not close the gap to PPR. The KS panels show that DPS-PC does not recover the accuracy and regularity obtained by projecting through the denoiser and renoising, and PPR remains the most reliable method for low-violation samples that preserve the data distribution.

F Additional KS Constraints

This section repeats the KS experiments on three additional constraints: two easier observation constraints and a conservation law. The main text uses the multi-time sine constraint

$$c(u_0) = \sum_i \|\sin(u_0^i) - \sin(u^i)\|^2, \qquad i \in \{0, 64, 127\},$$

which constrains nonlinear observations at three system times. Here we also consider the single-time sine constraint $c(u_0) = \|\sin(u_0^0) - \sin(u^0)\|^2$ and the linear observation $c(u_0) = \|u_0^0 - u^0\|^2$. These alternatives are substantially less challenging, especially the linear observation. Consequently, the baselines can appear more competitive on the metrics for these constraints. The relevant distinction is whether a method also maintains feasibility and continuity; PPR remains the most reliable method under that stricter criterion. The linear observation is forecasting from an initial condition, but it is not the main benchmark and we do not claim parity with specialized PDE forecasters.

Each figure below aggregates the two additional observation constraints, with the single-time sine observation on the left and the linear observation on the right. Within each block, the figure reports constraint violation, Max-Norm, spread, skill, RMSE, and CRPS. In all metric panels, the x-axis is the dynamical system time, not diffusion time.

Conservation of the spatial mean. We also enforce a conservation law. The spatial mean $m^i = \frac{1}{L}\int_0^L u^i(x)\,dx$, with $L = 64$, is zero for our initial conditions and should remain zero (numerically $\sim 10^{-7}$) at all system times. We frame $m_0^i = 0$ for all i as a constraint. For ease of comparison, we set the feasibility threshold to the largest violation observed among PPR samples. Therefore, even the worst PPR sample has a smaller violation than $(100 - \text{feasibility})\%$ of the baseline samples. The unconstrained prior has 0% feasibility (Table 12), showing that the pretrained model does not conserve the spatial mean on its own. PPR also achieves the lowest residual among all samplers, making it the closest to the fine-resolution GT and, under this numerical criterion, more feasible than the coarse GT the prior was trained on.

DiffusionPDE. We run DiffusionPDE [17] with the same pretrained prior and select its two guidance weights per constraint with a grid search on feasibility at 1024 diffusion steps. The row in Table 3 uses 4096 steps and takes 91.5 s, comparable to PPR's 80.2 s, evaluated on the first 32 test trajectories of the main constraint. At 1024 steps (25.8 s) on all 256 test trajectories, feasibility is 9.1% with RMSE 1.036.

Table 12: Conservation of the spatial mean on KS. The feasibility threshold was set at $\max_i |m^i| < 4 \times 10^{-5}$. GT (fine) and GT (coarse) are the test trajectories at resolutions 512 and 128.

Method	Time (s) ↓	$\max_i \lvert m^i \rvert$ ↓	Feas. (%) ↑
GT (fine)	—	$(2.67 \pm 1.40) \times 10^{-7}$	100.0
GT (coarse)	—	$(2.14 \pm 0.45) \times 10^{-4}$	0.4
Prior	—	$(8.96 \pm 3.81) \times 10^{-2}$	0.0
PPR (Ours)	127 ± 2	$(2.99 \pm 0.59) \times 10^{-5}$	**100.0**
DPS	25.6 ± 0.1	$(3.61 \pm 11.80) \times 10^{-2}$	0.0
MMPS	305 ± 0	$(3.85 \pm 0.53) \times 10^{-5}$	68.8
TS	408 ± 0	$(4.02 \pm 0.43) \times 10^{-5}$	43.8
CPS	79.1 ± 0.5	$(3.68 \pm 0.39) \times 10^{-5}$	75.0
PDM	295 ± 0	$(1.22 \pm 0.28) \times 10^{-3}$	0.0
HDC	113 ± 2	$(3.78 \pm 0.41) \times 10^{-5}$	68.8
DiffusionPDE	27.0 ± 2.3	$(5.09 \pm 2.17) \times 10^{-3}$	0.0

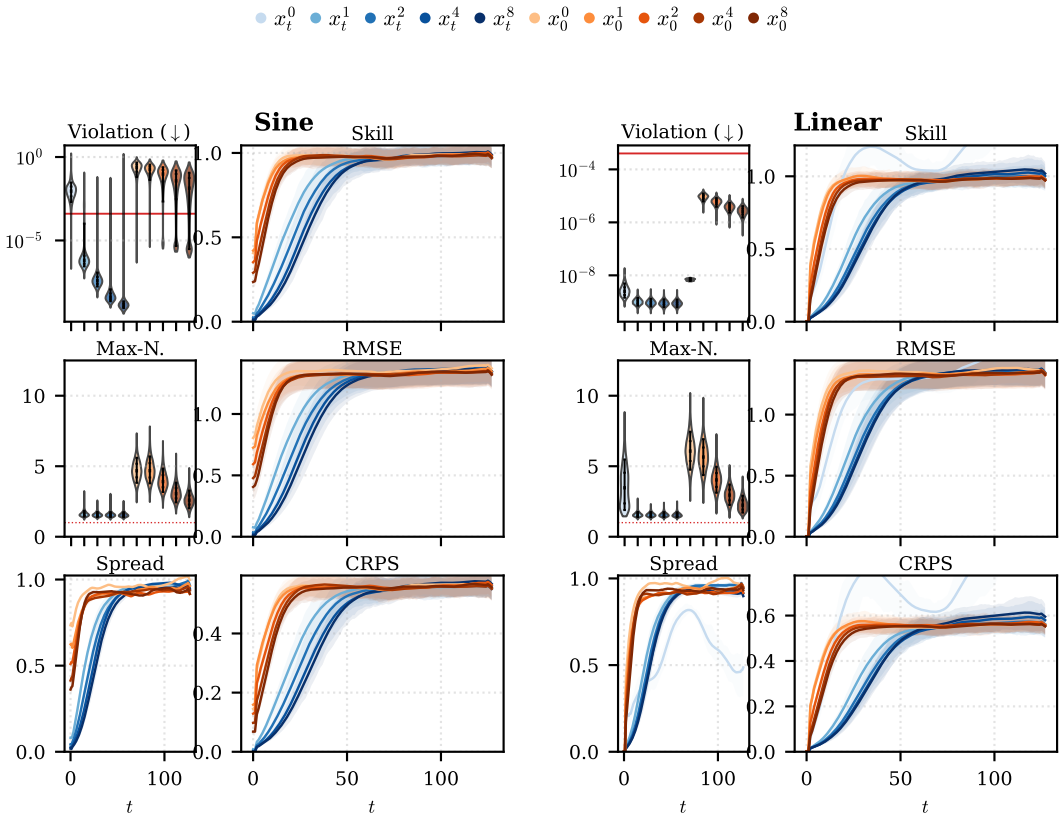


Figure 10: KS PPR ablation on additional constraints. Despite the easier single-time sine and linear observations, the same conclusion holds: projecting through the denoiser and renoising are necessary to preserve regular trajectories while reducing constraint violation. These easier constraints are less discriminating than the main-text multi-time sine constraint, but they still show the same failure mode for projections that do not preserve the data manifold.

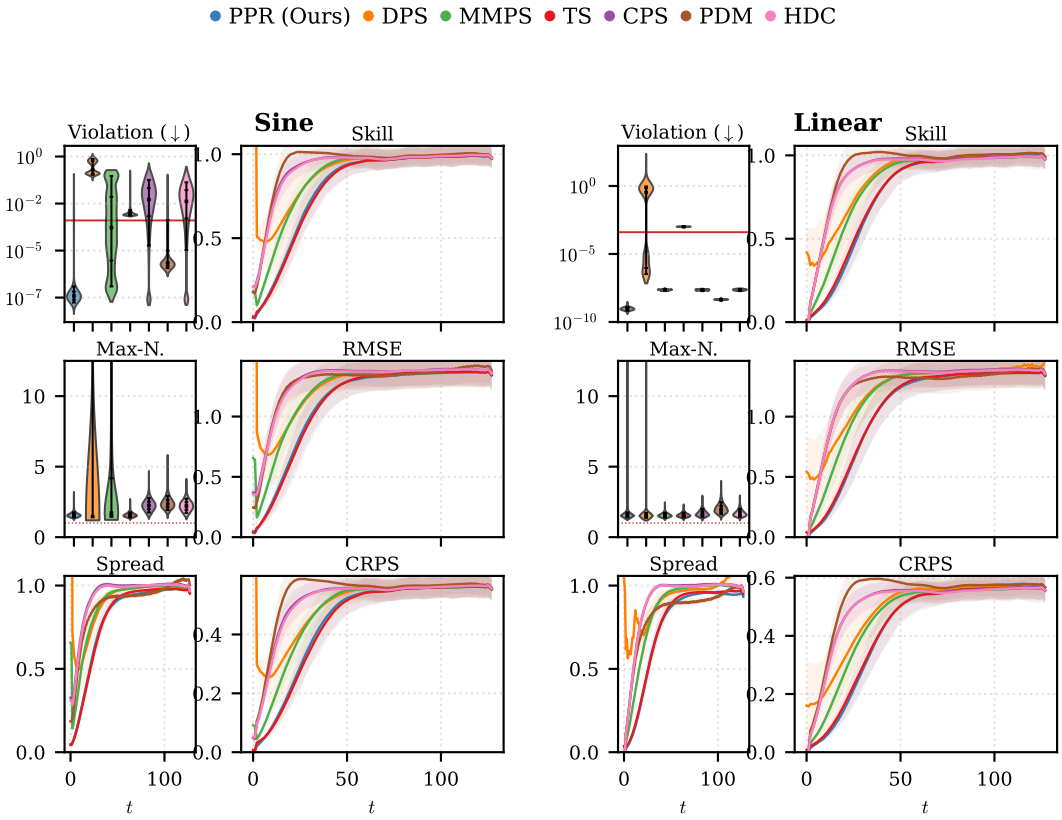


Figure 11: KS baseline comparison on additional constraints. Because the single-time sine and linear observations are easier, several baselines become more competitive on the trajectory proxy metrics. However, PPR remains the most consistent method when metrics are considered together with the constraint violation.

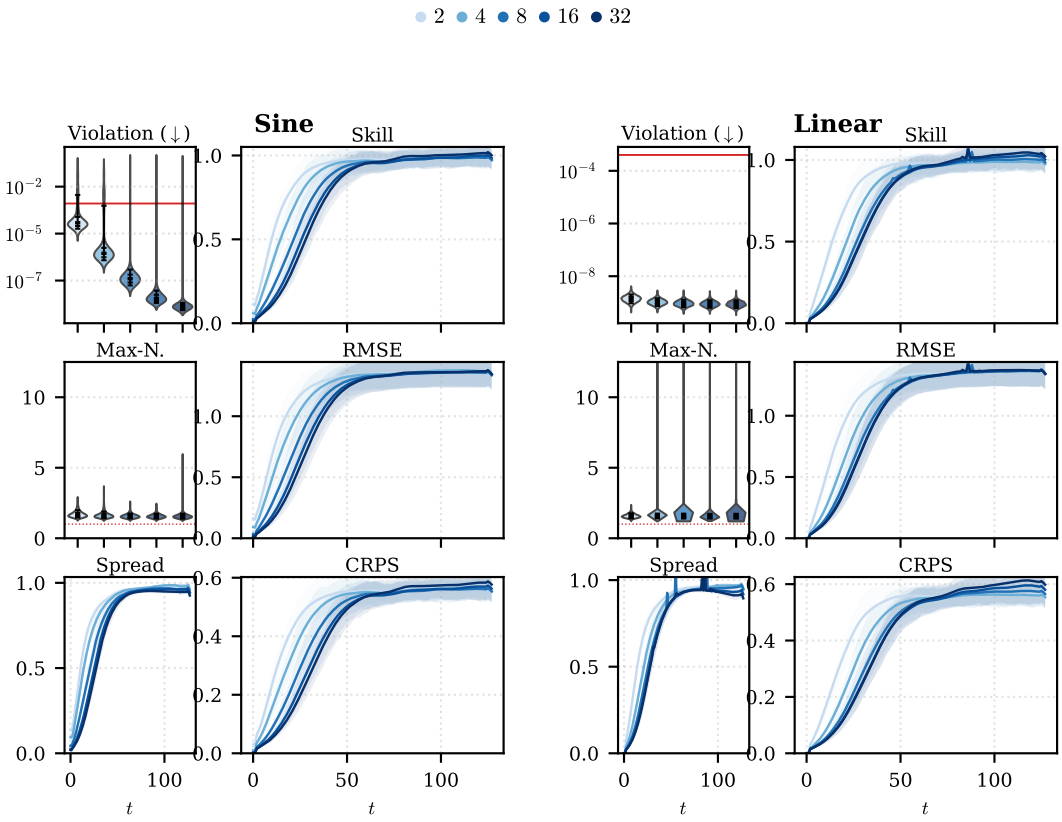


Figure 12: KS projection-step ablation on additional constraints. Increasing the number of projection iterations improves the local solution of Π_d and generally lowers violation.

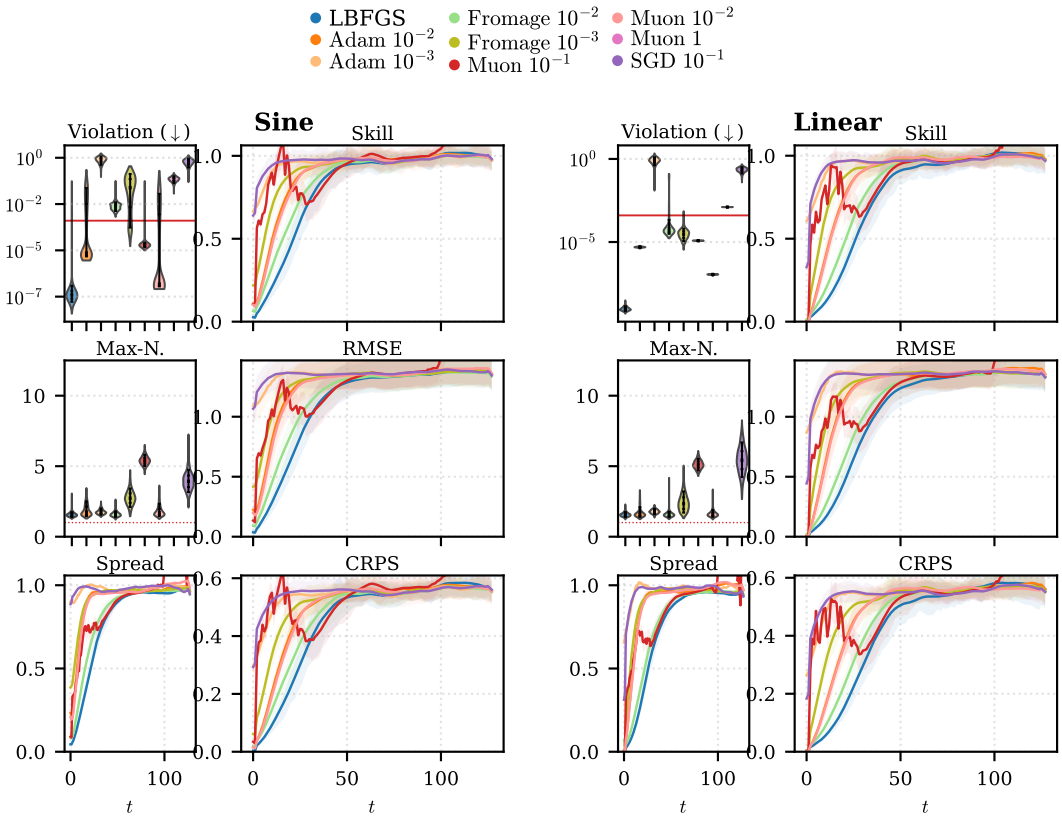


Figure 13: KS optimizer ablation on additional constraints. L-BFGS remains the most stable option across constraints, while first-order optimizers are more sensitive to their learning rates and less reliable in maintaining both feasibility and trajectory regularity.

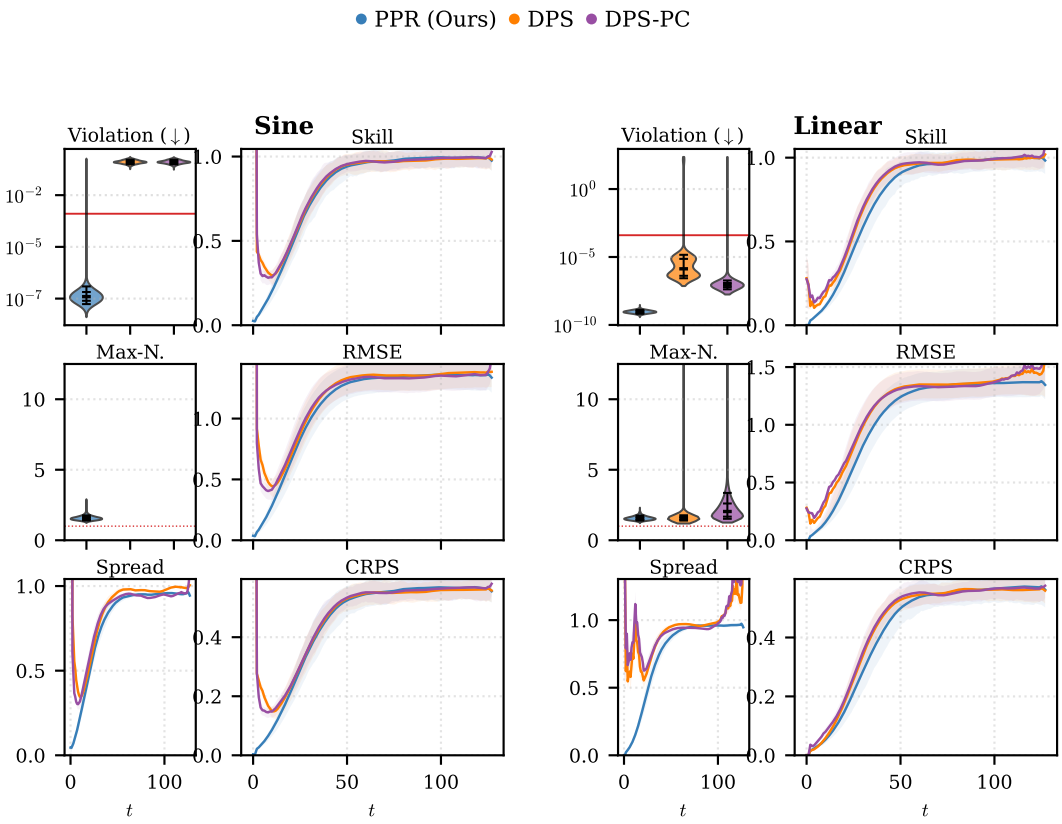


Figure 14: KS DPS-PC ablation on additional constraints. DPS-PC improves over DPS in some easier cases, but this does not close the gap to PPR.

G Example samples from all experiments

We provide below some representative samples from each of the experiments. For KS and WEATHER, we simply picked a random ensemble member from each method to plot. Higher resolution versions are available in the supplementary material.

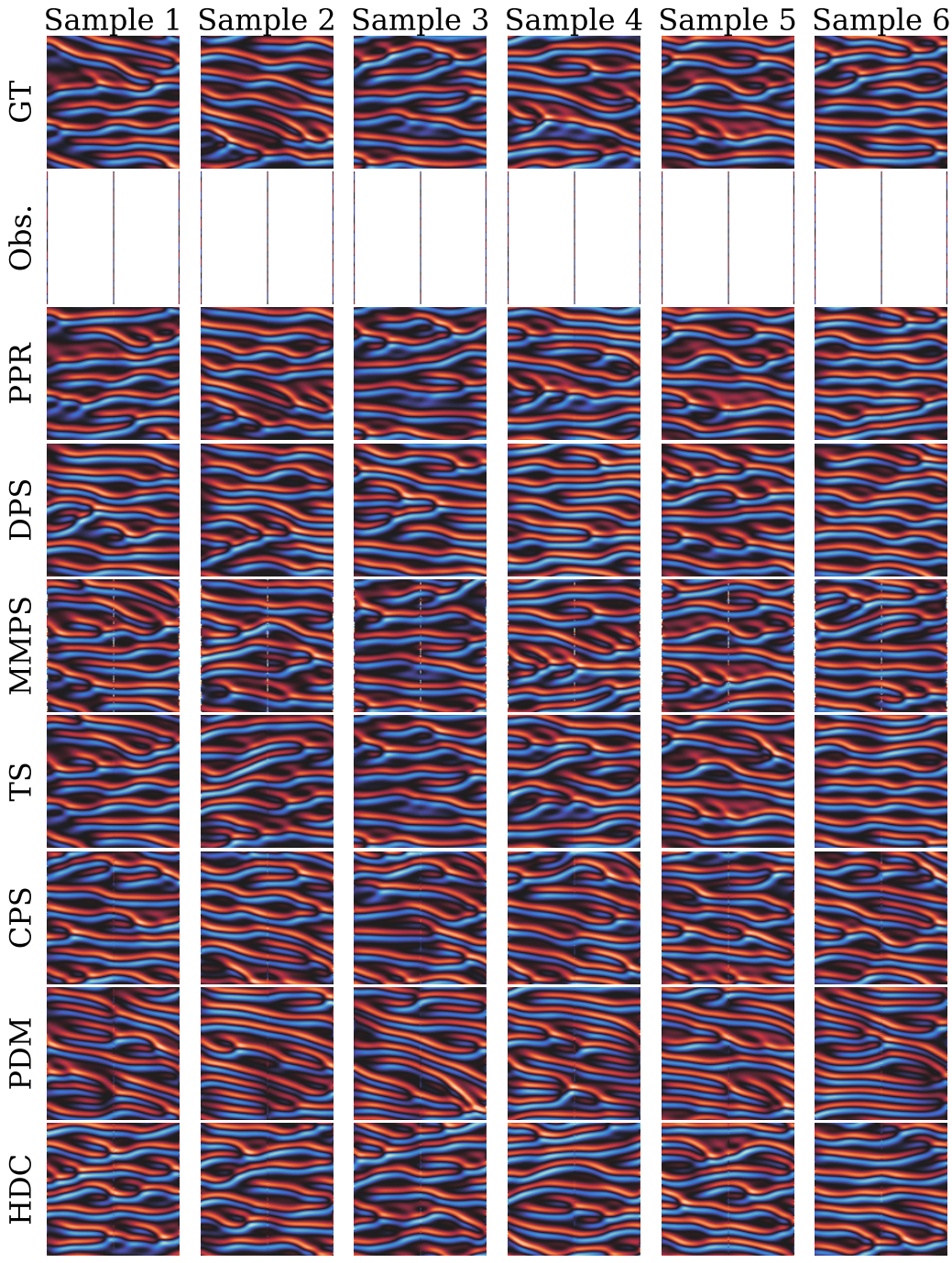


Figure 15: Samples from the sine observation at 3 different times, corresponding to the experiment in Section 5. For each sample from the ground truth set, we show one member of the ensemble produced by each method. The y-axis is the spatial domain, while the x-axis is the temporal domain.

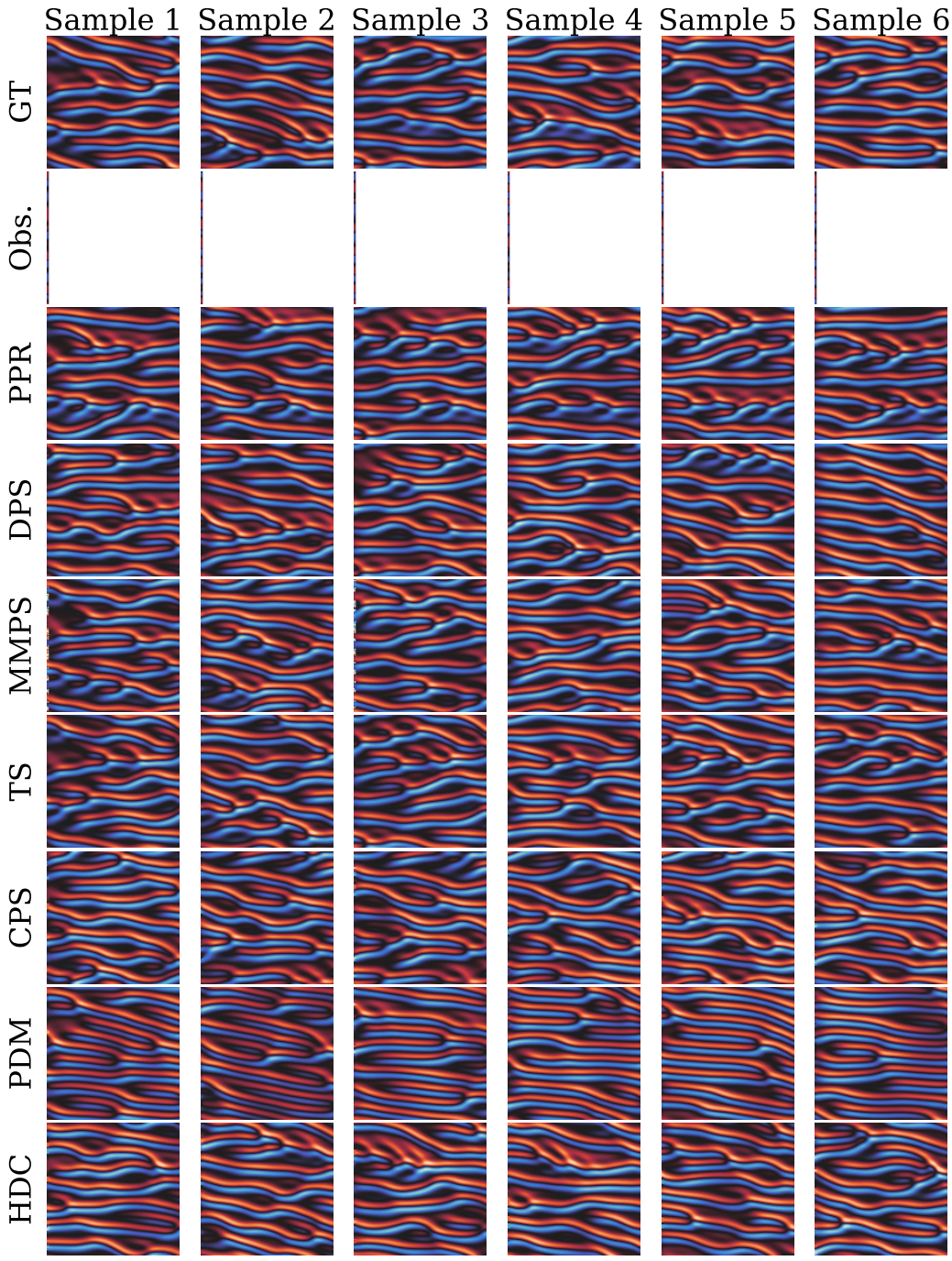


Figure 16: Samples from the sine observation at $t = 0$. For each sample from the ground truth set, we show one member of the ensemble produced by each method. The y-axis is the spatial domain, while the x-axis is the temporal domain.

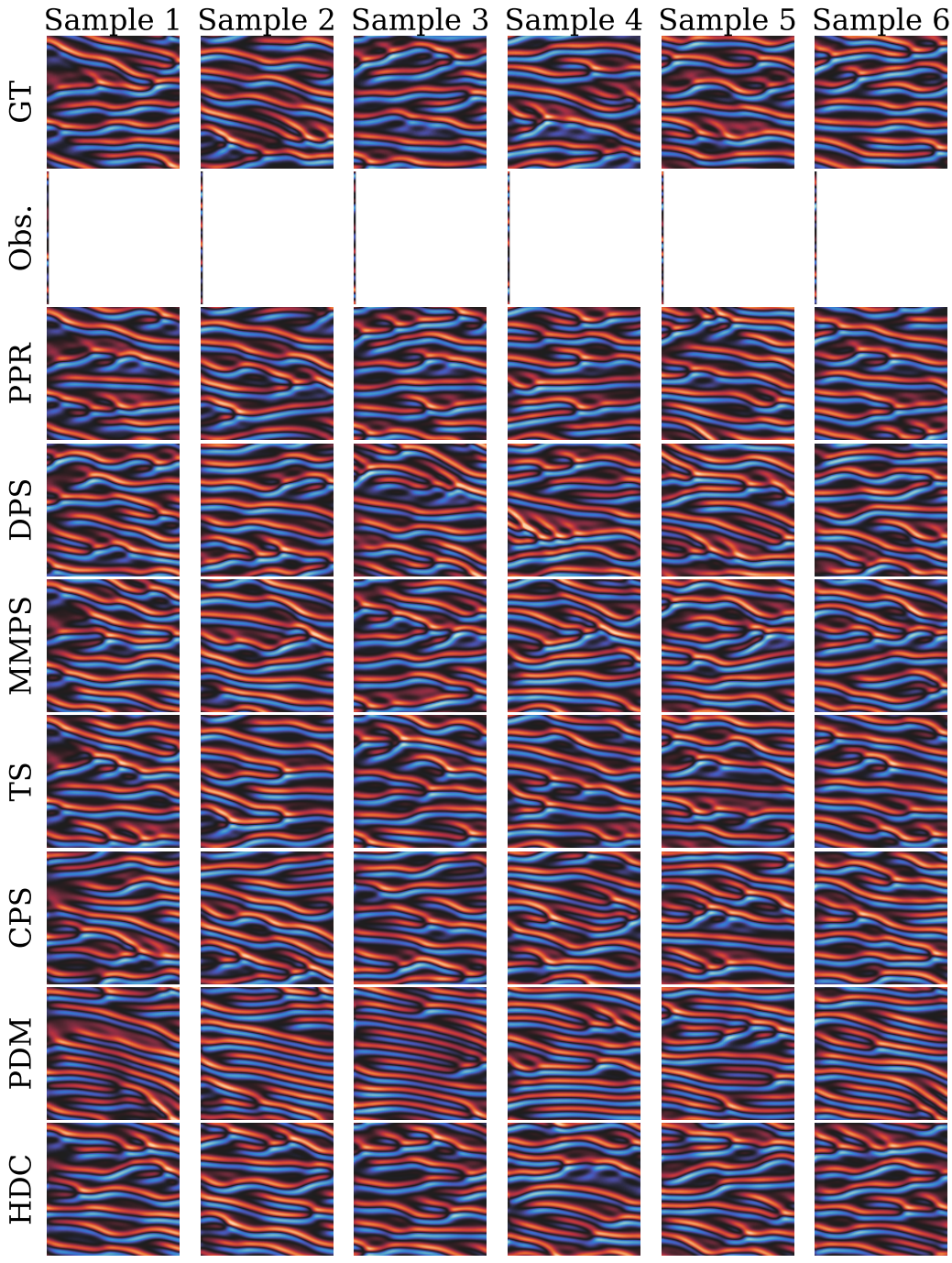


Figure 17: Samples from the linear observation at $t = 0$. For each sample from the ground truth set, we show one member of the ensemble produced by each method. The y-axis is the spatial domain, while the x-axis is the temporal domain.

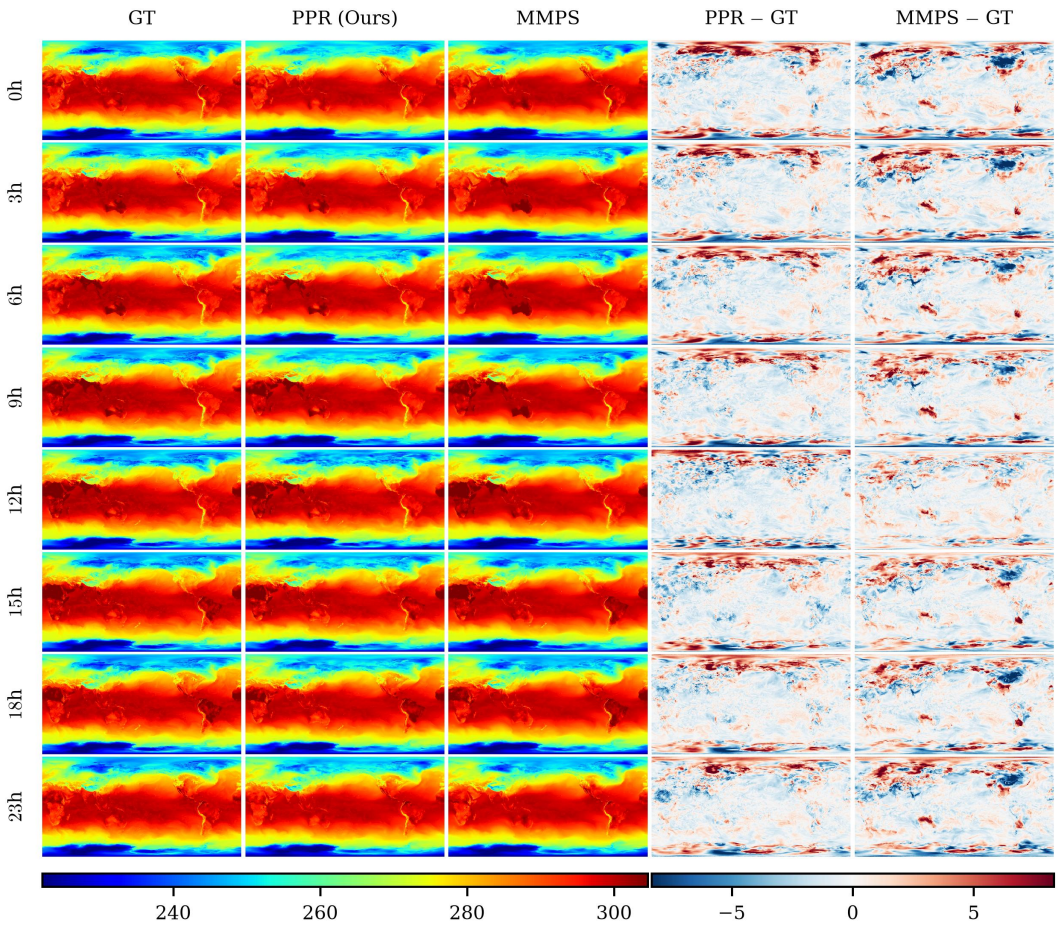


Figure 18: WEATHER sample showing the 2m temperature (K) for the ground truth, PPR, and MMPS, and the errors of PPR and MMPS against the ground truth. The y-axis is the hour in 24 h format.

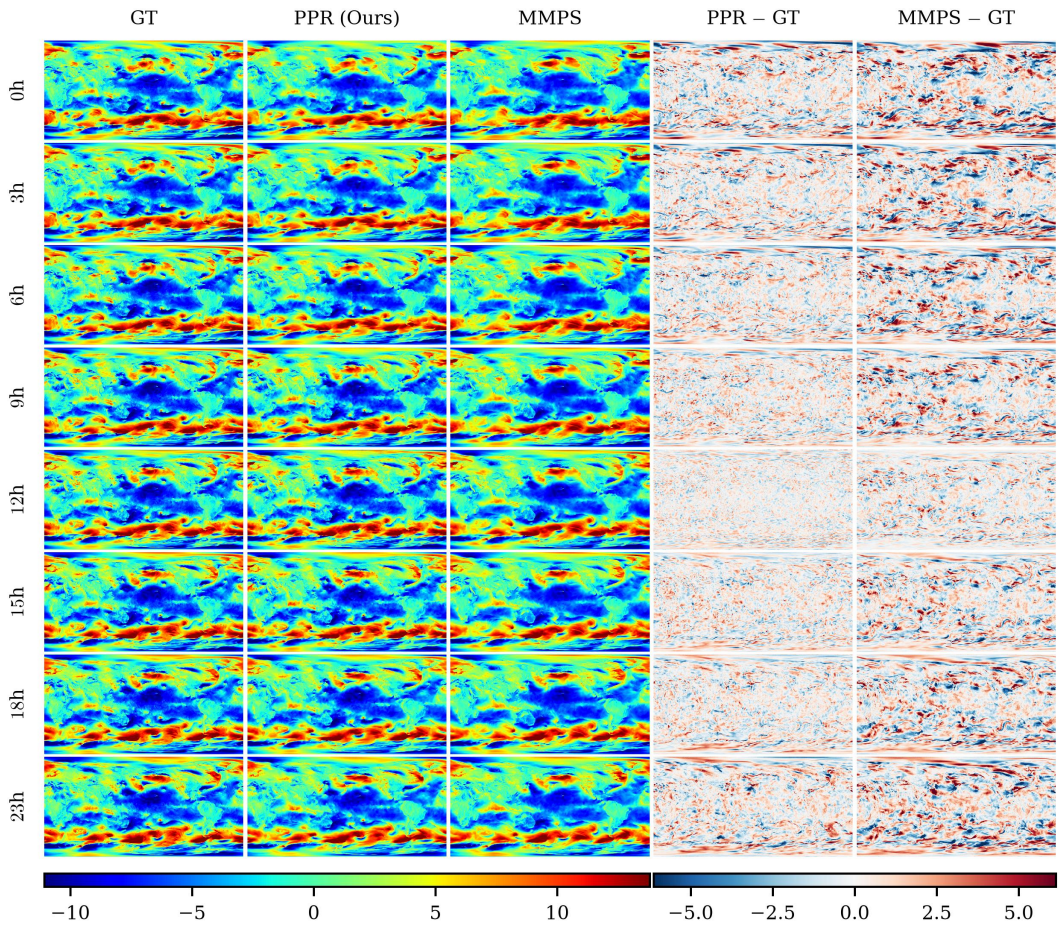


Figure 19: WEATHER sample showing the 10m zonal wind component u ($\frac{m}{s}$) for the ground truth, PPR, and MMPS, and the errors of PPR and MMPS against the ground truth. The y-axis is the hour in 24 h format.

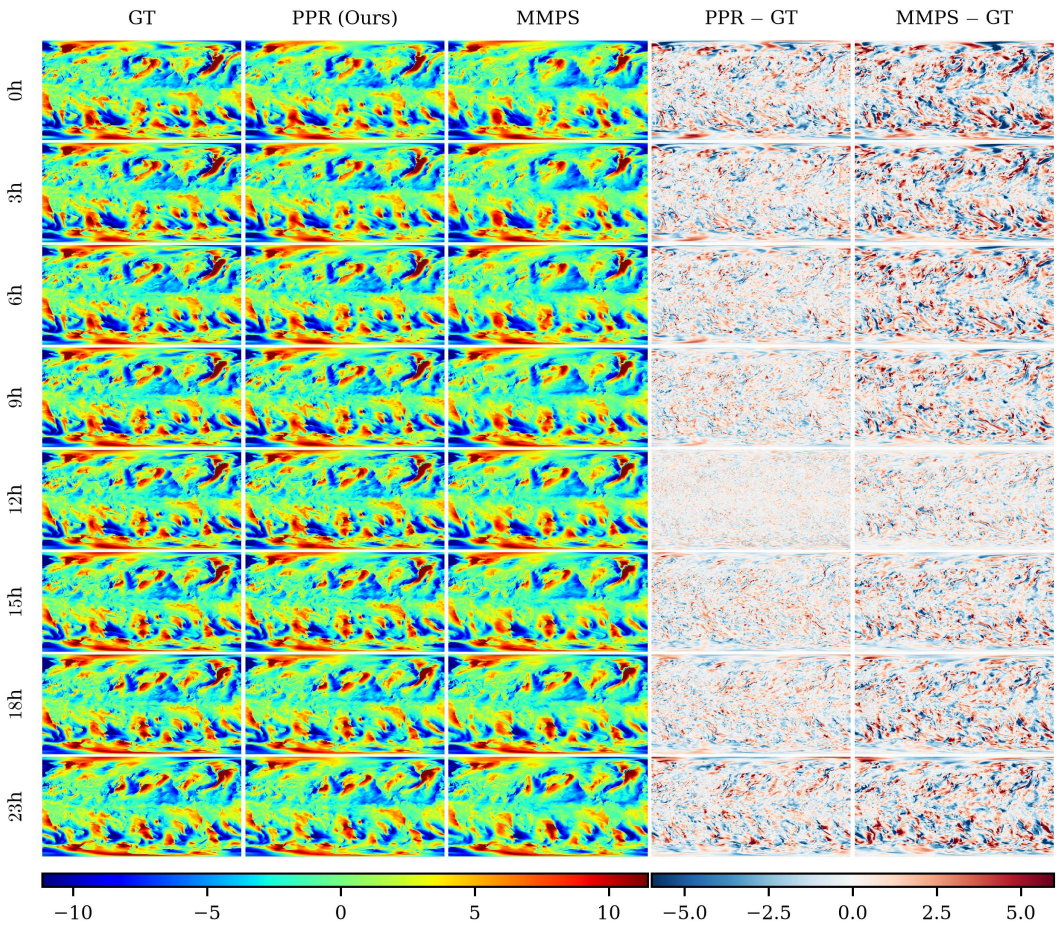


Figure 20: WEATHER sample showing the 10m meridional wind component v ($\frac{m}{s}$) for the ground truth, PPR, and MMPS, and the errors of PPR and MMPS against the ground truth. The y-axis is the hour in 24 h format.

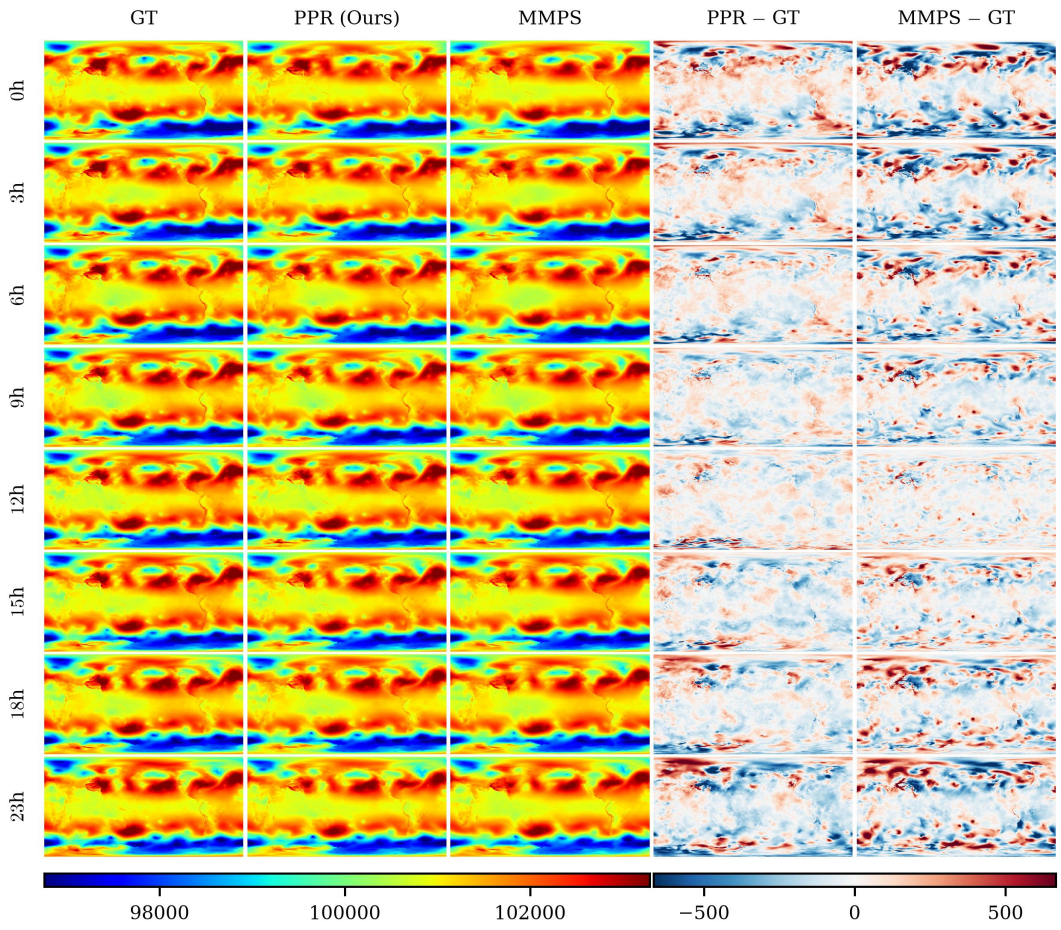


Figure 21: WEATHER sample showing the mean sea level pressure (Pa) for the ground truth, PPR, and MMPS, and the errors of PPR and MMPS against the ground truth. The y-axis is the hour in 24 h format.

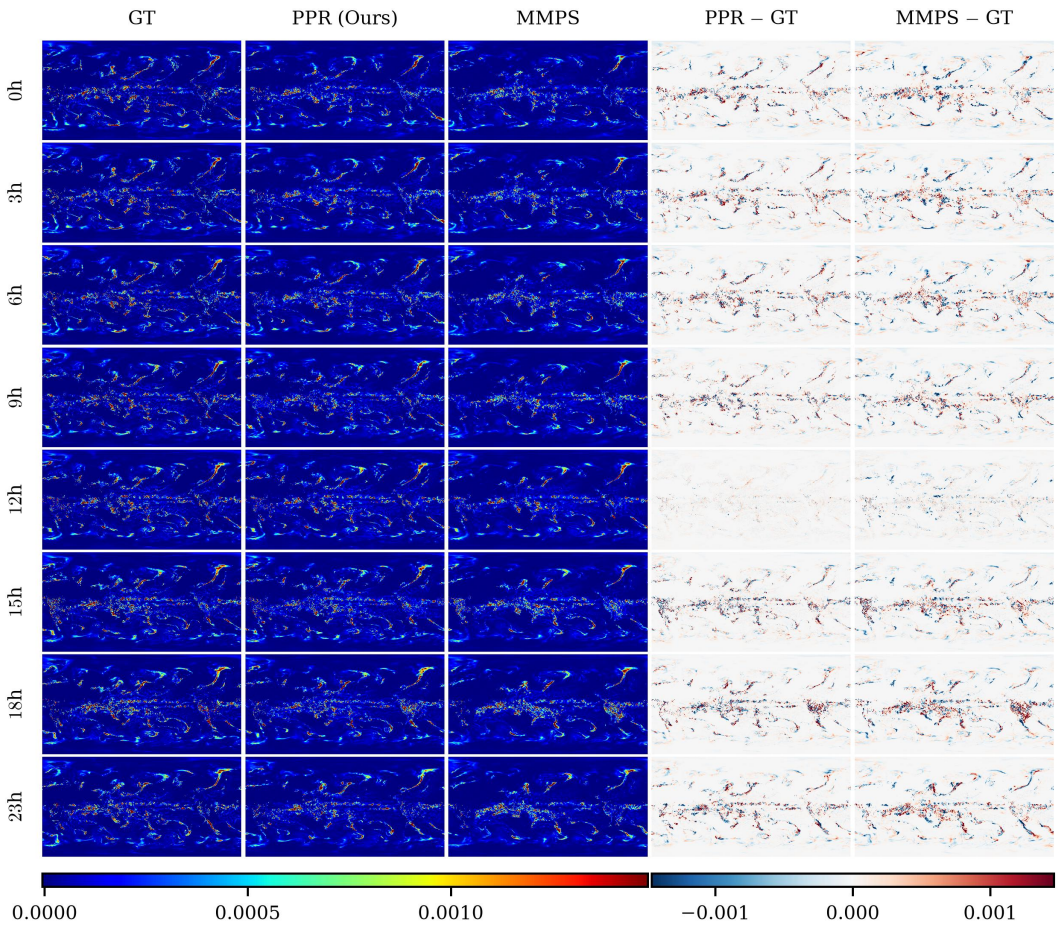


Figure 22: WEATHER sample showing the total precipitation (m) for the ground truth, PPR, and MMPS, and the errors of PPR and MMPS against the ground truth. The y-axis is the hour in 24 h format.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction reflect the core contribution of the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have included a discussion on the limitations of our work in the Conclusion section of the paper.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The proofs to all propositions are given in the appendix and the assumptions are stated.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have provided the data details, experimental details, and model architectures. Further, the code is available at `https://github.com/OmerRochman/ppr`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is available at `https://github.com/OmerRochman/ppr`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We have specified the details in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The results are givens statistically, with means and standard deviations, violin plots, and quantiles plots.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provided it in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the code.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This is foundational research into constrained sampling in the physical sciences. It does not have societal impacts beyond the standard impact of machine learning in the physical science research.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We have not used risky data or models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: We generated our own models and data.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We have not released new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.