SEQ-FLOW: EFFICIENT PROBABILISTIC FORECASTING WITH SELF-ROLLOUT ERROR CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Many scientific forecasting tasks require updating a distribution over future trajectories as new observations arrive. Conventional diffusion and flow models generate each forecast from Gaussian noise, often at the cost of many sampling steps. Warm-start methods reuse earlier predictions to reduce this cost, but their models are not trained to perform the forecast update itself, which can compromise quality under few-step sampling. In this work, we introduce Seq-Flow, a conditional flow model whose ODE transports samples from the previous forecast distribution to the updated one. Because successive forecasts often differ only modestly, this transport starts from an informative distribution and can produce accurate updates with few flow evaluations. Recursive reuse also creates a challenge: errors in one forecast become errors in the initial states of subsequent flows. We address this with self-rollout training, in which a moving average copy of the model generates forecasts that initialize later training updates. Unlike self-forcing methods, which reuse generated outputs as conditioning context, Seq-Flow reuses them as the source of the next flow. Experiments On particle-accelerator beam spill forecasting show Seq-Flow reduces CRPS by 65% under a few-NFE sampling budget, while remaining competitive with strong baselines on fluid-dynamics forecasting tasks. Although trained on self-rollouts of at most four updates, Seq-Flow remains accurate over more than 400 consecutive updates.

1 INTRODUCTION

Faithfully and efficiently modeling the evolving distribution of future trajectories is crucial for scientific forecasting, monitoring, and control. In applications such as weather forecasting (Kalnay, 2003; Lorenz, 2017), fluid dynamics (Bewley, 2001), and magnetic plasma control (Degrave et al., 2022), noisy observations and partial observability make the future inherently uncertain, requiring models to continually update this distribution as new observations arrive. Canonical filtering methods, such as the Kalman filter and its nonlinear extensions (Kalman, 1960; Julier & Uhlmann, 1997), recursively propagate predictive distributions but require a specified dynamical model, limiting their ability to capture complex and multimodal futures. Recurrent neural models (Shi et al., 2015; Chung et al., 2015; Salinas et al., 2020) learn more expressive dynamics directly from data, but often produce point forecasts or rely on restrictive parametric output distributions, limiting the predictive uncertainty they can represent.

Diffusion models (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Ho et al., 2020) and flow matching (Lipman et al., 2023; Liu et al., 2023) provide expressive ways to model complex, multimodal distributions, motivating their growing use in forecasting, planning, and other sequential tasks (Chen et al., 2024; Gao et al., 2023; Janner et al., 2022; Wei et al., 2024). In conventional approaches, however, each forecast is generated anew from an uninformative Gaussian source. Reaching a high-fidelity sample from this source can require many denoising or ODE solver steps, while using only a few function evaluations often degrades sample quality (Salimans & Ho, 2022; Liu et al., 2023). In rapidly evolving systems, this quality–compute trade-off directly limits how frequently predictive distributions can be updated as new observations arrive.

To reduce the cost of repeatedly sampling a predictive distribution from Gaussian noise, we propose Seq-Flow, a flow-matching approach that updates predictive samples as new observations arrive. Unlike a deterministic point forecaster, Seq-Flow maintains multiple samples representing plausible

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

future trajectories. At each observation time, it uses samples from the previous predictive distribution as the source and transports them toward the updated predictive distribution conditioned on the newly observed history. Because these samples already reflect past observations, they offer an informative starting point that may allow accurate updates with fewer flow evaluations than sampling from Gaussian noise anew. This recursive transport parallels classical filtering, which updates predictive beliefs as observations arrive (Kalman, 1960; Gordon et al., 1993).

However, recursively reusing forecasts creates a train–test shift in the flow's source distribution, a failure mode absent when each generation starts from a fixed Gaussian prior. If training uses ground-truth trajectories as sources, the flow is not exposed to the errors produced by its own recursive updates. At inference, however, each output feeds into the next flow, so these errors can propagate despite conditioning on new observations. This mechanism differs from the autoregressive exposure bias studied in prior work (Bengio et al., 2015; Huang et al., 2026): autoregressive models feed generated outputs into the conditioning history, whereas Seq-Flow uses generated forecasts to construct the source distribution of the next flow. This source-level feedback is particularly risky under few-step sampling. To address this mismatch, we train Seq-Flow through self-rollouts generated by an exponential moving average (EMA) copy of the update model. Each generated forecast is used to construct the source for the next training update, allowing the flow to learn from imperfect forecasts of the kind encountered during recursive deployment.

We evaluate Seq-Flow on particle-accelerator Beam Spill forecasting (Whitbeck et al., 2025), simulated Burgers' dynamics (Hwang et al., 2022), and real-world fluid–structure interaction data from RealPDEBench (Hu et al., 2026). On Beam Spill, Seq-Flow reduces CRPS by 65% relative to a standard diffusion baseline under a matched 3-NFE sampling budget, while remaining competitive with strong baselines on both fluid-dynamics tasks. We further show that Seq-Flow remains stable over long forecast runs: despite training on self-rollouts of at most four updates, it maintains forecast accuracy through more than 400 consecutive updates without substantial error accumulation.

2 Methodology

2.1 Problem Formulation and Flow Matching Preliminary

Problem Formulation. We consider an online probabilistic forecasting problem for a dynamical system evolving over time $t = 1, 2, \ldots, T$. Let x_t denote the system state at time t, and let $x_{\leq t} := x_{1:t}$ denote the observed state history up to time t. At each time step, the goal is to characterize the distribution of the future H-step trajectory $x_{t+1:t+H}$ conditioned on the current history $x_{\leq t}$, where H denotes the prediction horizon. We use $\mathbf{X}_t := x_{t+1:t+H}$ to denote the H-step future trajectory. We therefore formulate online probabilistic forecasting as tracking the time-evolving forecasting distribution $p_t := p(\mathbf{X}_t | x_{\leq t})$ as the system evolves and new observations become available.

Flow Matching. Flow matching learns a velocity field v_θ whose ODE $\frac{d}{d\tau}\mathbf{Z}(\tau) = v_\theta(\mathbf{Z}(\tau), \tau)$ transports samples from a source distribution $\mathbf{Z}(0) \sim p_{\text{source}}$ to a target distribution $\mathbf{Z}(1) \sim p_{\text{target}}$ (Lipman et al., 2023; Liu et al., 2023). Using the linear interpolation $\mathbf{Z}(\tau) = (1-\tau)\mathbf{Z}(0) + \tau\mathbf{Z}(1)$, the velocity field is trained by regressing to the conditional displacement $\mathbf{Z}(1) - \mathbf{Z}(0)$, with $\tau \sim \text{Uniform}(0, 1)$. At inference, samples from the target distribution are obtained by integrating the learned ODE starting from the source distribution. Throughout the paper, we use τ to denote flow time and t to denote the physical time of the dynamical system. In our forecasting setting, $\mathbf{X}_t$ denotes the future trajectory at physical time t, whereas $\mathbf{Z}_t(\tau)$ denotes the auxiliary variable evolving along the flow used to generate a forecast sample of $\mathbf{X}_t$.

2.2 Existing Practices for Online Forecasting with Flow Models

A common approach to probabilistic forecasting with diffusion or flow models is to formulate each forecast as a conditional generation problem (Rasul et al., 2021; Kollovieh et al., 2025). Specifically, at physical time t, the model transports samples from a fixed source distribution, typically Gaussian noise, to the predictive distribution $p_t = p(\mathbf{X}_t | x_{\leq t})$, conditioned on the observed history $x_{\leq t}$. This gives the flow ODE

$$\frac{d}{d\tau}\mathbf{Z}_t(\tau) = v_\theta(\mathbf{Z}_t(\tau), \tau; x_{\leq t}), \qquad \mathbf{Z}_t(0) \sim \mathcal{N}(0, I), \quad \mathbf{Z}_t(1) \sim p_t, \tag{1}$$

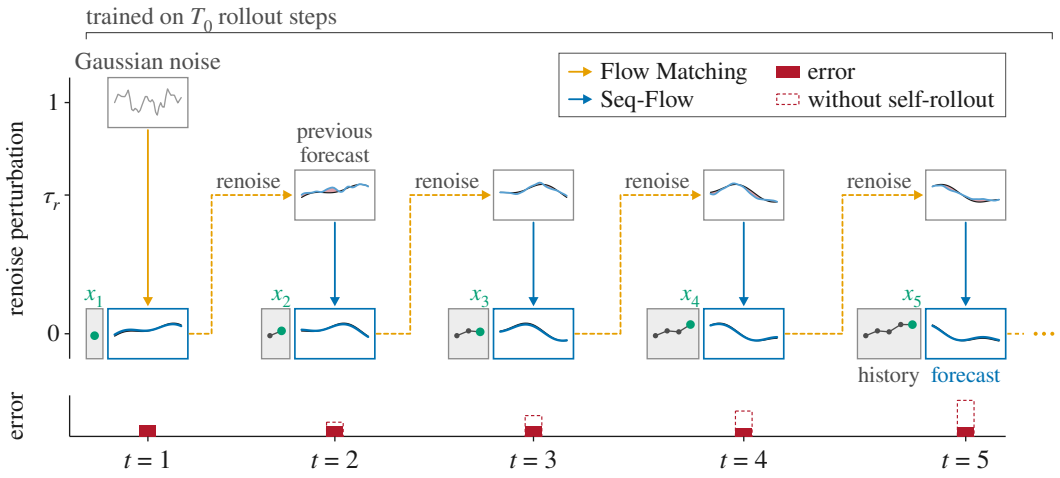


Figure 1: **Seq-Flow with self-rollout training.** Seq-Flow tracks the forecast distribution by modeling the forecast update. It uses a renoised sample from the previous forecast as the source distribution for the next flow, enabling efficient few-step updates. Self-rollout training exposes the model to its own forecast sources, teaching it to correct errors that would otherwise compound across updates.

where $\tau \in [0, 1]$ denotes the flow time and t denotes the physical time step.

Equation 1 requires repeatedly transporting samples from Gaussian noise to p_t. Such noise-to-target transport typically requires multiple neural function evaluations (NFEs), while reducing the integration steps can degrade sample quality (Salimans & Ho, 2022). It motivates recent work on few-step flow models via flow map distillation (Song et al., 2023; Geng et al., 2025). In contrast, we aim to exploit the temporal continuity of online forecasting to reduce the transport burden itself rather than distillating the flow sampling path.

2.3 Seq-Flow: Aligning Flow Transport with Forecast Update

Our key idea is to formulate the distributional change in the forecasting update itself as the flow transport problem. We propose Seq-Flow, which aligns the flow transport with the temporal evolution of the forecasting distribution. Instead of transporting samples from a fixed Gaussian source to p_t at every physical time step, Seq-Flow directly transports the previous forecasting distribution p_{t-1} to the updated distribution p_t, via the following flow ODE:

$$\frac{d}{d\tau}\mathbf{Z}_t(\tau) = v_\theta(\mathbf{Z}_t(\tau), \tau; x_{\leq t}), \qquad \mathbf{Z}_t(0) \sim p_{t-1}, \quad \mathbf{Z}_t(1) \sim p_t. \tag{2}$$

Thus, Seq-Flow learns an update operator between successive forecast distributions by incorporating new observed history. Recursively applying these transports tracks the evolution $p_1 \to p_2 \to \cdots$. At the initial step $t = 1$, we define $p_0 := \mathcal{N}(0, I)$ and generate the first forecast from Gaussian noise. Figure 1 illustrates the recursive forecast update process of Seq-Flow.

Temporal continuity enables efficient sampling. For temporally smooth dynamical systems, successive predictive distributions p_{t-1} and p_t typically share substantial structure. Using samples from p_{t-1} as the source therefore gives the flow a more informative starting distribution than Gaussian noise, so each update only needs to model the change in the forecast distribution rather than generating from scratch. Prior work has shown that replacing an uninformed Gaussian source with a more informative prior can improve generation under limited sampling budgets (Ren et al., 2025; Scholz & Turner, 2025; Park et al., 2024). Instead, Seq-Flow obtains an informative source from the sequential forecasting process itself, using the previous predictive distribution and directly learning the recursive transport $p_{t-1} \to p_t$.

Comparison to physical-time flow formulations. Recent approaches such as Streaming Flow (Jiang et al., 2025) and ODEWorld (Liu et al., 2026) also align flow ODE dynamics with physical-time evolution, but in a different manner. First, advancing along their flow ODE step corresponds to advancing the system in one physical time step. In contrast, Seq-Flow uses an entire flow

transport integration to model one physical-time forecast update $p_{t-1} \to p_t$, allowing multiple flow evaluations for each update. Second, these methods model an open-loop temporal evolution under a fixed observation context. Newly arriving observations are incorporated by initiating or replanning a subsequent rollout. Seq-Flow instead tracks the evolving forecast distribution in a closed-loop manner, where each newly observed state directly induces the next transport.

2.4 Controlling Recursive Source-distribution Error by Self-Rollout

Leveraging the flow ODE to model the forecast update $p_{t-1} \to p_t$ enables efficient sampling but carries sampling errors directly into subsequent updates. At inference, Seq-Flow reuses its previous generated forecast $\hat{p}_{t-1}$ as the source distribution of the next flow. Errors in one forecast therefore directly perturb the initialization of the subsequent transport. If the model is trained only with ground-truth sources p_{t-1}, it is never exposed to these model-induced source deviations, which can compound over successive forecast updates.

This error accumulation problem differs from exposure bias (Bengio et al., 2015; Huang et al., 2026) in autoregressive noise-to-target generation. There, each prediction is generated from a fixed noise source and past errors enter through the conditioning history. In our setting, the observed history $x_{\leq t}$ remains unchanged, while the model-generated forecast $\hat{\mathbf{X}}_{t-1}$ initializes the next flow, resulting train–test mismatch in the flow source distribution. We next develop training strategies to mitigate this recursive source-distribution error.

Self-rollout training with robust source distributions. We propose to explicitly unroll the flow model to follow the inference process of forecast update during training. To our knowledge, this is the first training framework that uses a flow model's own recursive forecast rollouts as *source samples* for subsequent flow updates. Given a ground-truth trajectory $x^*_{1:T}$ and starting from $p_0 = \mathcal{N}(0, I)$, we recursively generate model-induced forecasts $p_0 \to \hat{p}_1 \to \cdots \to \hat{p}_{T_0}$ using equation 2 (the unrolling step T_0 can be far less than the task episode, as we will discuss in Section 2.5). At each step t, a generated forecast $\hat{\mathbf{X}}_{t-1} \sim \hat{p}_{t-1}$ provides the source for the next forecast update. To improve robustness to deviations in these recursively generated sources, we partially renoise each rollout sample as $\tilde{\mathbf{X}}_{t-1} = (1 - \tau_r)\hat{\mathbf{X}}_{t-1} + \tau_r \cdot \mathcal{N}(0, I)$, where $\tau_r \in [0, 1]$ is a renoise level hyperparameter. After collecting the rollouts $\{\tilde{\mathbf{X}}_t\}_{t=1}^{T_0}$, the model is then trained to transport the resulting source $\tilde{\mathbf{X}}_{t-1}$ to the ground-truth future trajectory $\mathbf{X}^*_t = x^*_{t+1:t+H}$ using the flow-matching objective

$$\mathcal{L}(\theta) = \sum_t \mathbb{E}_{\tau \sim \mathrm{Uniform}(0,1)} \left\| v_\theta\Big((1-\tau)\tilde{\mathbf{X}}_{t-1} + \tau \mathbf{X}^*_t, \tau; x^*_{\leq t}\Big) - \Big(\mathbf{X}^*_t - \tilde{\mathbf{X}}_{t-1}\Big) \right\|^2. \tag{3}$$

This training procedure exposes the model to the source-distribution errors that arise during recursive deployment, while renoising further introduces robustness by perturbing the source distributions.

The renoising level τ_r controls the trade-off between preserving information from the previous forecast and robustness to accumulated source errors. At $\tau_r = 1$, the source reduces to Gaussian noise, recovering the conventional noise-to-target generation paradigm. It eliminates recursive source errors, but meanwhile requires substantially larger sampling steps. At $\tau_r = 0$, the model directly reuses its raw previous forecast, which may maximize the information encoded in the historical generation but also making subsequent updates most sensitive to errors. We treat τ_r as a hyperparameter to tune in practice.

Self-rollout and renoising provide complementary roles that are both critical for stable long-horizon deployment: self-rollout exposes the model to the source-distribution errors induced by its own recursive generation, while renoising improves robustness around these model-induced sources. In Section 4.3, we empirically show that removing either component leads to clear error accumulation over forecasting time.

Comparison to self-forcing diffusion. Although self-forcing (Huang et al., 2026) also exposes the model to its own generations during training, they consider autoregressive noise-to-target diffusion and feeds generated outputs back to models' conditioning history, and the diffusion's source distribution remains Gaussian noise. Empirically, we find that this noise-to-target formulation yields suboptimal performance and requires a large number of sampling steps when modeling a long-horizon $\mathbf{X}_t = x_{t+1:t+H}$. In contrast, self-rollout Seq-Flow unrolls generated forecasts as source samples

for subsequent flow transports, and we find this forecast-to-forecast flow yields better performance-efficiency trade-offs.

2.5 Stable and Efficient Self-rollout Training

The self-rollout training directly mitigates the recursive source-distribution mismatch, but naively optimizing it is computationally expensive and unstable. A full rollout over a long forecast episode T requires repeatedly solving the flow ODE, while backpropagating through both the flow ODE steps and the entire rollout chain leads to high memory cost and unstable gradients. We discuss several practical strategies to make self-rollout training stable and efficient.

Firstly, we apply stop gradient to all model rollouts $\hat{\mathbf{X}}_t$, so that gradients are not propagated through earlier flow ODE and rollout steps. This substantially reduces the backpropagation depth and memory cost. Stop-gradient techniques are also adopted in previous self-forcing training and other flow matching optimization (Huang et al., 2026; Geng et al., 2025).

Self-rollout also introduces a moving source distribution in flow matching loss, since the source samples $\hat{\mathbf{X}}_t$ used for flow matching training are generated by the model itself whose parameters θ are continuously updated. To reduce the resulting optimization instability, we maintain an exponential moving average of the model parameters, denoted by θ_{EMA}, and use the EMA model to generate rollout sources. Since θ_{EMA} evolves more smoothly than the online parameters θ, the corresponding source distributions vary more slowly during training. The EMA model is also used for inference after training. We find this greatly stabilizes the training and makes optimization easier.

Prior work also adopts EMA models to stabilize self-distillation training when model-generated predictions directly define training targets along the generative trajectory (Song et al., 2023; Frans et al., 2025). Self-forcing (Huang et al., 2026), by contrast, feeds model generations back as conditioning context and does not rely on an EMA rollout model. In Seq-Flow, generated forecasts directly determine the source distribution of subsequent transports, motivating EMA to stabilize this evolving source distribution.

Lastly, unrolling over the full trajectory of length T remains computationally expensive even with stopped gradients. Instead, we randomly slice a contiguous trajectory chunk of length $T_0 \ll T$ and perform self-rollout only within this T_0-step chunk. This truncated rollout substantially reduces training cost while still exposing the model to multi-step error propagation. As shown in Section 4.3, training with short rollout chunks ($T_0 \leq 4$) is sufficient for stable deployment over forecasting steps far beyond T_0.

With these strategies, self-rollout Seq-Flow can be trained efficiently and stably. The complete training and inference procedures are described in Algorithms 1 and 2.

3 Related Works

Diffusion models for temporal data. Diffusion and flow-based models have been increasingly applied to temporal and sequential data, with different designs for incorporating temporal structure into the generative process. One line of work develops asynchronous denoising schedules, where tokens at different physical times are assigned different noise levels (Ruhe et al., 2024; Wu et al., 2023; Chen et al., 2024). Compared to full-sequence denoising (Li et al., 2022; Ho et al., 2022) and autoregressive denoising (Hoogeboom et al., 2021; Rasul et al., 2021), asynchronous denoising allows different parts of the sequence to represent different levels of uncertainty during generation. Another line of work incorporates temporal structure into the source distribution itself. Rather than using isotropic Gaussian noise, these methods adopt structured priors, such as Gaussian processes, to introduce temporal correlations that better match the data (Biloš et al., 2023; Kollovieh et al., 2025). However, the deployment of these models in online forecasting is still typically re-initialized from a pre-determined source distribution and requires many denoising steps to reach the target distribution. Achieving high-fidelity generation with only one or a few sampling steps is still generally a challenging problem (Liu et al., 2023; Song et al., 2023; Geng et al., 2025).

Temporal reuse for efficient diffusion models. A growing line of work exploits temporal continuity by reusing previous predictions to accelerate subsequent generation. Warm-start methods perturb

Algorithm 1 Self-rollout Seq-Flow Training

Require: A ground-truth trajectory slice $x^*_{1:T}$ and its corresponding $\mathbf{X}^*_t := x_{t+1:t+H}$, self-rollout step T_0, model weights θ and its EMA weights θ_{EMA}, EMA parameter $\beta \in [0, 1]$

Randomly sample $t_0 \sim \text{Uniform}(1, T - T_0 - H + 1)$
for $t = t_0$ to $t_0 + T_0 - 1$ **do**
  if $t = t_0$ **then**
    $\tilde{\mathbf{X}}_0 \sim \mathcal{N}(0, I)$
  else
    $\tilde{\mathbf{X}}_{t-1} = (1 - \tau_r)\hat{\mathbf{X}}_{t-1} + \tau_r \cdot \mathcal{N}(0, I)$
  end if
  Sample a random flow interpolation time $\tau \in [0, 1]$
  Flow matching loss $\mathcal{L}_t(\theta) = \left\| v_\theta((1-\tau)\tilde{\mathbf{X}}_{t-1} + \tau \mathbf{X}^*_t, \tau; x^*_{\leq t}) - (\mathbf{X}^*_t - \tilde{\mathbf{X}}_{t-1}) \right\|^2$
  Rollout by solving ODE $\frac{d}{d\tau}\mathbf{Z}_t(\tau) = v_{\theta_{\text{EMA}}}(\mathbf{Z}_t(\tau), \tau; x_{\leq t})$ starting by $\mathbf{Z}_t(0) = \tilde{\mathbf{X}}_{t-1}$, and obtain $\mathbf{Z}_t(1)$.
  Collect new forecast $\hat{\mathbf{X}}_t = \text{sg}(\mathbf{Z}_t(1))$
end for
return updated weights $\theta \leftarrow \theta - \nabla_\theta \sum_t \mathcal{L}_t(\theta)$ and EMA weights $\theta_{\text{EMA}} \leftarrow (1 - \beta)\theta_{\text{EMA}} + \beta\theta$.

Algorithm 2 Seq-Flow Inference

Require: Initial state x_1, renoise level τ_r, EMA Seq-Flow model $v_{\theta_{\text{EMA}}}(\mathbf{Z}_t(\tau), \tau; x_{\leq t})$, task episode T

Initialize $\mathbf{Z}_1(0) \sim \mathcal{N}(0, I)$ and solve $\frac{d}{d\tau}\mathbf{Z}_1(\tau) = v_\theta(\mathbf{Z}_1(\tau), \tau; x_1)$ to generate the initial forecast $\mathbf{X}_1 = \mathbf{Z}_1(1)$
for $t = 2$ to T **do**
  Receive new observation x_t
  $\tilde{\mathbf{X}}_{t-1} = (1 - \tau_r) \cdot \mathbf{X}_{t-1} + \tau_r \cdot \mathcal{N}(0, I)$
  Solve ODE $\frac{d}{d\tau}\mathbf{Z}_t(\tau) = v_{\theta_{\text{EMA}}}(\mathbf{Z}_t(\tau), \tau; x_{\leq t})$ starting from $\mathbf{Z}_t(0) = \tilde{\mathbf{X}}_{t-1}$, and obtain $\mathbf{Z}_t(1)$
  Obtain new forecast $\mathbf{X}_t = \mathbf{Z}_t(1)$
end for

predictions from the previous time step and denoise them under the updated observations (Janner et al., 2022; Duan et al., 2025; Li et al., 2026). However, the diffusion model is still trained for noise-to-target generation rather than for the forecast-update transport itself. Asynchronous denoising methods instead distribute denoising computation across physical time by maintaining partially denoised future predictions and progressively updating them as the system evolves (Wei et al., 2025; Høeg et al., 2025; Guo et al., 2026). This pipeline improves sampling efficiency, but the future predictions remain intentionally partially denoised and therefore less accurate at future time steps. Streaming Flow (Jiang et al., 2025) and ODEWorld (Liu et al., 2026) are more closely related in aligning flow or ODE dynamics with temporal evolution. Their formulations, however, differ from Seq-Flow in how flow dynamics correspond to physical-time evolution and how new observations are incorporated. We discussed these distinctions in Section 2.3.

Error accumulation in recursive generation. Prior work studies train–test mismatch from conditioning on ground-truth histories during training but model-generated histories at inference. Seq-Flow instead faces a flow-matching source-distribution mismatch, as each generated forecast initializes the next flow. We discuss this distinction in Section 2.4.

4 Experiments

We aim to study the following questions for Seq-Flow. **Q1** (*Temporal alignment*): Does aligning the flow ODE with the temporal evolution of forecast distribution improve forecasting in fidelity and efficiency? **Q2** (*Long-horizon robustness*): Does self-rollout Seq-Flow control distributional error accumulation over a long run?

Table 1: Results on the synthetic dataset, Beam Spill, Burgers' Equation, and RealPDE-FSI. Prediction horizon $H = 5$ for the synthetic task and 10 for the remaining tasks.

Method	NFE	Synthetic	Beam Spill		Burgers' Equation		RealPDE-FSI	
		$\mathcal{W}_1$ dist.↓	CRPS ↓	RMSE ↓	CRPS ↓	RMSE ↓	CRPS ↓	RMSE ↓
AR Diffusion	$H \times 3$	1.270	0.363	1.001	0.0589	0.1183	0.0150	0.0260
Self-forced AR Diffusion	$H \times 3$	2.765	0.055	0.209	0.0143	0.0395	0.0031	0.0062
Full-trajectory Diffusion	3	2.260	0.043	0.179	0.0139	0.0366	0.0030	0.0063
AR Flow	$H \times 3$	1.203	0.446	1.092	0.0822	0.1401	0.0130	0.0208
Self-forced AR Flow	$H \times 3$	2.872	0.059	0.230	0.0198	0.0356	0.0039	0.0081
Full-trajectory Flow	3	2.221	0.036	0.139	0.0068	0.0175	0.0029	0.0056
Full-trajectory Flow	10	**0.905**	0.035	0.151	0.0066	0.0178	0.0028	0.0062
Diffusion Forcing	3	1.533	0.050	0.240	0.0137	0.0357	0.0040	0.0080
Flow Forcing	3	1.526	0.040	0.167	0.0100	0.0205	0.0030	**0.0055**
MeanFlow	3	50.081	0.103	0.631	0.7253	2.7046	0.0028	0.0059
StreamingFlow	$H \times 1$	2.381	0.469	2.485	0.0706	0.3114	0.0029	0.0083
Warm-start Flow	3	304.403	1.349	6.051	0.0191	0.0391	0.0027	0.0059
Asynchronous Flow	3	20.766	0.522	0.912	0.0572	0.0935	0.0067	0.0202
Seq-Flow	3	1.069	**0.015**	**0.080**	**0.0060**	**0.0165**	**0.0026**	**0.0055**

Baselines. We compare Seq-Flow against several classes of baselines: (1) *Generic diffusion/flow models for sequential data*, including full-trajectory diffusion, autoregressive (AR) diffusion (Rasul et al., 2021), and Diffusion Forcing (Chen et al., 2024). These models all use Gaussian noises as the source distribution for each forecast. We adapt the idea of Diffusion Forcing (training with independently sampled noise levels for each token) to flow model, resulting Flow Forcing. (2) *Diffusion methods with generation reuse*, which exploit previously generated states to facilitate subsequent generation. We implement warm-start diffusion (Janner et al., 2022; Duan et al., 2025), asynchronous diffusion following Wei et al. (2025); Høeg et al. (2025); Guo et al. (2026) and StreamingFlow (Jiang et al., 2025); (3) *Self-forcing diffusion* (Huang et al., 2026), which uses model-generated rollouts during training to reduce the train–test distribution gap in autoregressive diffusion; (4) *Few-step diffusion*: we compare to Meanflow (Geng et al., 2025), which is specifically designed to achieve high-quality generation with one or only a few flow sampling steps.

Trajectory alignment. Since consecutive H-step forecasts are offset by one physical time step, when initializing for the next update, we drop the first state of the previous forecast and pad its final state, $\mathrm{shift}(\mathbf{X}_{t-1}) = [x_{t+1}, \ldots, x_{t+H-1}, x_{t+H-1}]$, so that the overlapping forecast coordinates are aligned in physical time. Other details of the model implementation are deferred to Appendix A.2.

4.1 Scientific Dynamic Systems

To answer **Q1**, we first evaluate Seq-Flow on a synthetic and several scientific forecasting tasks. We primarily consider the beam spill dynamic in the Mu2e experiment at Fermilab, a particle-physics experiment designed to search for new physics Whitbeck et al. (2025). We further evaluate Seq-Flow on a simulated and an experimentally-measured real fluid system (Hwang et al., 2022; Hu et al., 2026). We present the dataset and task setup below, and the full details is deferred to Appendix A.1.

Synthetic system. We consider a mixture-of-Gaussian random walk $p(x_{t+1}|x_t) = \frac{1}{2}\mathcal{N}(x_t+1, \delta^2) + \frac{1}{2}\mathcal{N}(x_t - 1, \delta^2)$, with $\delta = 10^{-3}$ and $x_1 \sim \mathcal{N}(0, 10^{-4})$. The task is to infer the $H = 5$ future trajectory distribution $p(x_{t+1:t+5}|x_t)$ over $T = 100$ forecast updates. Since the ground-truth distribution is trackable, this setting provides a controlled testbed for analyzing distributional fidelity.

Beam spill forecasting. The Mu2e experiment at Fermilab is a particle-physics experiment designed to search for charged-lepton flavor violation (Whitbeck et al., 2025). During beam extraction, charged particles are gradually released from the storage ring, producing a beam intensity that evolves over time. Maintaining a uniform spill is important for detector live time and for avoiding the loss of useful physics data (Narayanan, 2022). The spill-regulation system operates at 10 kHz, with a feedback loop on the order of 1 ms, making low-latency prediction and continual updating of the evolving beam state particularly relevant. We use simulated beam-intensity trajectories of length 430 and predict the next $H = 10$ states conditioned on the observed history and control inputs.

Fluid system forecasting. We further evaluate on both simulated and experimentally measured fluid dynamics. (1) For **Burgers' equation**, we simulate one-dimensional velocity fields on a grid

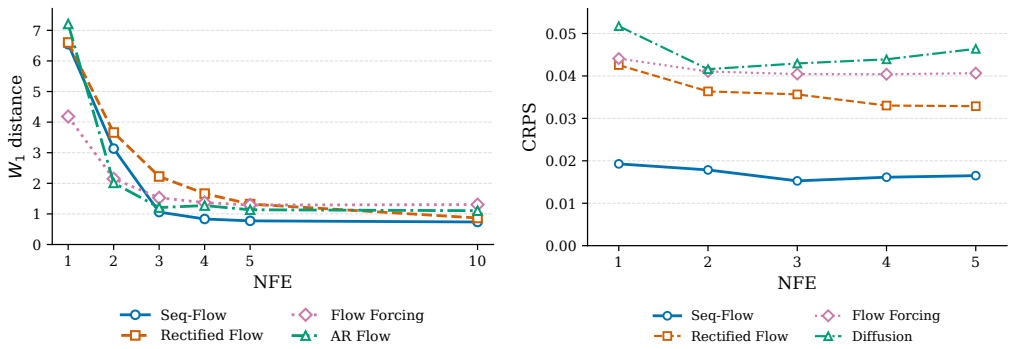


Figure 2: **Performance-efficiency trade-offs**. X axis: sampling steps (NFE), Y axis: task performance. For clarity, each figure shows only a subset of the top-performing baselines. Left: synthetic task. Right: beam spill forecasting.

of 64 spatial points (Hwang et al., 2022). The model observes only the first 32 points and does not observe the randomly generated external forcing, making the future evolution uncertain given the available observations. The task is to forecast future velocities over the observed half of the domain. (2) For **RealPDE-FSI**, we use experimentally measured fluid–structure interaction data from RealPDEBench (Hu et al., 2026). Each state contains two velocity components, and we downsample the original 128×128 fields to 32×32, withholding fine-scale spatial information and thereby introducing additional forecast uncertainty.

Distribution evaluation metrics. To evaluate distributional quality of forecasts, we report the Continuous Ranked Probability Score (CRPS), a proper scoring rule widely used in probabilistic forecasting (Gneiting & Raftery, 2007). Given a ground-truth future trajectory $\mathbf{X}_t$ and predictive distribution $\hat{p}_t$, we compute $\mathrm{CRPS}(\hat{p}_t, \mathbf{X}_t) = \mathbb{E}_{\hat{\mathbf{X}}_t \sim \hat{p}_t} \|\hat{\mathbf{X}}_t - \mathbf{X}_t\|_1 - \frac{1}{2}\mathbb{E}_{\hat{\mathbf{X}}_t, \hat{\mathbf{X}}'_t \sim \hat{p}_t} \|\hat{\mathbf{X}}_t - \hat{\mathbf{X}}'_t\|_1$. The first term measures forecast accuracy, while the second rewards appropriate predictive dispersion. As a complementary pointwise metric, we also report root mean squared error (RMSE). For the synthetic task, where the ground-truth predictive distribution p_t is available, we directly evaluate distributional fidelity using the average 1-Wasserstein distance $\frac{1}{T}\sum_{t=1}^{T} \mathcal{W}_1(p_t, \hat{p}_t)$.

Shared setup and evaluation. We use Transformer backbones for Beam Spill and Burgers, and spatiotemporal U-Nets for FSI. The forecast horizon is $H = 5$ for synthetic and $H = 10$ for Beam Spill, Burgers, and FSI.

Results. Table 1 shows that, using three function evaluations per forecast update, Seq-Flow achieves the lowest $\mathcal{W}_1$ distance among three-NFE methods on the synthetic task (1.069, a 29.9% reduction over Flow Forcing), the lowest CRPS and RMSE on Fermilab beam-spill forecasting (0.015 and 0.080, improving over three-NFE full-trajectory flow by 58.3% and 42.4%, respectively, while also outperforming its ten-NFE variant), and the lowest CRPS with the lowest or joint-lowest RMSE on Burgers' equation and RealPDE-FSI.

4.2 Performance-Efficiency Trade-offs

To further **Q1** in terms of performance-efficiency trade-offs, we vary the number of flow sampling steps (NFE) used consistently during both training rollouts and inference for Seq-Flow, and compare against the other baselines under the same sampling budgets. Figure 2 shows that Seq-Flow achieves better performance than the Gaussian-source flow matching and other baselines. Particularly, on synthetic task, a 3-NFE Seq-Flow attains competitive performance to the full-step rectified flow sampling from Gaussian noise. On beam spill forecasting, Seq-Flow consistently outperform full-step rectified flow for all sampling steps.

4.3 Ablation Study and Error Accumulation Analysis

To answer **Q2**, we study how Seq-Flow's key design choices affect distributional error over recursive forecast updates. We aim to answer: (1) whether self-rollout training and renoising are necessary

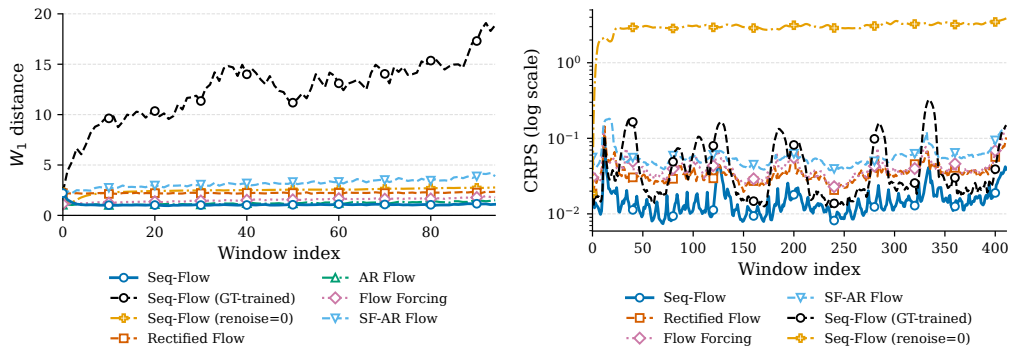


Figure 3: **Seq-Flow with controlled error accumulation over forecast steps**. Each figure shows performance over forecast step (window id). Left: synthetic task. Right: beam spill forecasting.

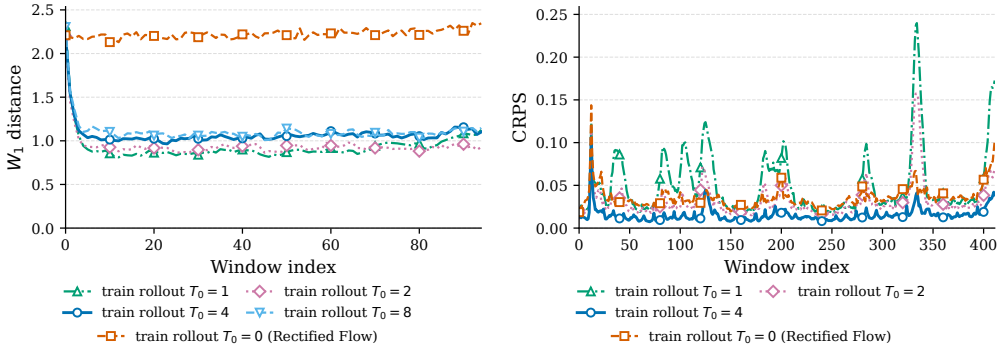


Figure 4: **Short self-rollout steps T_0 leads to robust long-run forecasting**. Each figure shows performance per forecast step with different T_0. Left: synthetic task. Right: beam spill forecasting.

for controlling error accumulation; (2) whether a model trained with a short self-rollout horizon T_0 can remain stable over substantially longer deployment horizons.

Ablation of self-rollout and renoising. We compare Seq-Flow against two ablated variants: one trained with ground-truth trajectories instead of model-generated self-rollouts, and one trained with $\tau_r = 0$. Figure 3 shows that either training with ground-truth sources or removing renoising can lead to rapidly accumulating error over successive forecast updates, consistent with the recursive source-distribution mismatch described in Section 4.3. These results show that both self-rollout exposure and source perturbation are important for robust recursive flow transports.

Effect of self-rollout horizon T_0. We next vary T_0 in Algorithm 1, which determines the number of consecutive forecast updates used during self-rollout training. Figure 4 demonstrates that, although the training rollout step is much shorter than the full task episode, Seq-Flow remains stable over substantially longer forecasting runs. In particular, using $T_0 = 1$ and $T_0 = 4$ is sufficient to control error over episodes $T = 100$ on synthetic and $T = 430$ on Beam Spill respectively, demonstrating that long-horizon robustness does not require unrolling the full deployment horizon during training.

5 Conclusion and Limitations

We introduce Seq-Flow, a flow matching approach that tracks online forecast distributions by transporting the previous forecast to the updated one conditioned on new observations, enabling accurate forecasting with few flow evaluations. To control the source-distribution error induced by recursive forecast reuse, we proposed self-rollout training with EMA-based rollout generation. Across several scientific forecasting tasks, Seq-Flow achieves strong few-step performance and remains stable over long recursive deployments. A limitation of our approach is that self-rollout training introduces extra training cost.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript editing, including improving grammar, clarity, and readability. We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, provide ingredients for or assist in proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean or reformat datasets, analyze or interpret results, or draw scientific conclusions. Translation and qualitative or thematic data analysis were not applicable to this work. All AI-assisted text was manually reviewed and revised by the authors for technical accuracy, consistency with the underlying experiments and methods, and appropriate attribution. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide code with instruction in the supplementary materials that can be used to reproduce experimental results present in the manuscript. Details of dataset construction is provided in the code.

REFERENCES

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

Thomas R Bewley. Flow control: new challenges for a new renaissance. *Progress in Aerospace sciences*, 37(1):21–58, 2001.

Marin Biloš, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, and Stephan Günnemann. Modeling temporal data as continuous functions with stochastic process diffusion. In *international conference on machine learning*, pp. 2452–2470. PMLR, 2023.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *Advances in neural information processing systems*, 28, 2015.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

Yufei Duan, Hang Yin, and Danica Kragic. Real-time iteration scheme for diffusion policy. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11758–11764. IEEE, 2025.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, volume 2025, pp. 34668–34684, 2025.

Zhihan Gao, Xingjian Shi, Boran Han, Hao Wang, Xiaoyong Jin, Danielle Maddix, Yi Zhu, Mu Li, and Yuyang Bernie Wang. Prediff: Precipitation nowcasting with latent diffusion models. *Advances in Neural Information Processing Systems*, 36:78621–78656, 2023.

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

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

Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pp. 107–113. IET, 1993.

Jiaming Guo, Rui Zhang, Zerun Li, Yunkai Gao, Shaohui Peng, Siming Lan, Xing Hu, Zidong Du, Xishan Zhang, and Ling Li. Efficient diffusion planning with temporal diffusion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 21450–21458, 2026.

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

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

Sigmund H Høeg, Yilun Du, and Olav Egeland. Fast policy synthesis with variable noise diffusion models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4821–4828. IEEE, 2025.

Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037*, 2021.

Peiyan Hu, Haodong Feng, Hongyuan Liu, Tongtong Yan, Wenhao Deng, Tianrun Gao, Rong Zheng, Haoren Zheng, Chenglei Yu, Chuanrui Wang, et al. Realpdebench: A benchmark for complex physical systems with real-world data. *arXiv preprint arXiv:2601.01829*, 2026.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *Advances in Neural Information Processing Systems*, 38:167283–167308, 2026.

Rakhoon Hwang, Jae Yong Lee, Jin Young Shin, and Hyung Ju Hwang. Solving pde-constrained control problems using operator learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 4504–4512, 2022.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pp. 9902–9915. PMLR, 2022.

Sunshine Jiang, Xiaolin Fang, Nicholas Roy, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Siddharth Ancha. Streaming flow policy: Simplifying diffusion / flow-matching policies by treating action trajectories as flow trajectories. *arXiv preprint arXiv:2505.21851*, 2025.

Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pp. 182–193. Spie, 1997.

RE Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, D*, 82:35–44, 1960.

Eugenia Kalnay. *Atmospheric modeling, data assimilation and predictability*. Cambridge university press, 2003.

Marcel Kollovieh, Marten Lienen, David Lüdke, Leo Schwinn, and Stephan Günnemann. Flow matching with gaussian process priors for probabilistic time series forecasting. In *International Conference on Learning Representations*, volume 2025, pp. 95435–95457, 2025.

Jinhao Li, Yuxuan Cong, Yingqiao Wang, Hao Xia, Shan Huang, Yijia Zhang, Ningyi Xu, and Guohao Dai. Step: Warm-started visuomotor policies with spatiotemporal consistency prediction. *arXiv preprint arXiv:2602.08245*, 2026.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35: 4328–4343, 2022.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *11th International Conference on Learning Representations, ICLR 2023*, 2023.

Dongxiu Liu, Haoyi Niu, Peng Cheng, Yuan Gao, Xirui Kang, Sangli Teng, Koushil Sreenath, and Xianyuan Zhan. Odeworld: A continuous predictive architecture via physical-time flow. *arXiv preprint arXiv:2607.27924*, 2026.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Edward N Lorenz. Deterministic nonperiodic flow 1. In *Universality in Chaos, 2nd edition*, pp. 367–378. Routledge, 2017.

Aakaash Narayanan. Machine learning for slow spill regulation in the fermilab delivery ring for mu2e. Technical report, Fermi National Accelerator Laboratory (FNAL), Batavia, IL (United States), 2022.

Jinseong Park, Seungyun Lee, Woojin Jeong, Yujin Choi, and Jaewook Lee. Leveraging priors via diffusion bridge for time series generation. *arXiv preprint arXiv:2408.06672*, 2, 2024.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International conference on machine learning*, pp. 8857–8868. PMLR, 2021.

Hao Ren, Yiming Zeng, Zetong Bi, Zhaoliang Wan, Junlong Huang, and Hui Cheng. Prior does matter: Visual navigation via denoising diffusion bridge models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12100–12110, 2025.

David Ruhe, Jonathan Heek, Tim Salimans, and Emiel Hoogeboom. Rolling diffusion models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 42818–42835. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/ruhe24a.html.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3):1181–1191, 2020.

Jonas Scholz and Richard E Turner. Warm starts accelerate conditional diffusion. *arXiv preprint arXiv:2507.09212*, 2025.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

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

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 32211–32252, 2023.

Long Wei, Peiyan Hu, Ruiqi Feng, Haodong Feng, Yixuan Du, Tao Zhang, Rui Wang, Yue Wang, Zhi-Ming Ma, and Tailin Wu. Diffphycon: A generative approach to control complex physical systems. *Advances in Neural Information Processing Systems*, 37:4090–4147, 2024.

Long Wei, Haodong Feng, Yuchen Yang, Ruiqi Feng, Peiyan Hu, Xiang Zheng, Tao Zhang, Dixia Fan, and Tailin Wu. CL-diffphycon: Closed-loop diffusion control of complex physical systems. In *The Thirteenth International Conference on Learning Representations*, 2025.

A Whitbeck, J Berlioz, K Danison-Fieldhouse, K Hazelwood, M Khan, J Mitrevski, A Narayanan, J St John, N Tran, J Ji, et al. Fast adaptive neural control of resonant extraction at fermilab. *arXiv preprint arXiv:2511.19099*, 2025.

Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, Jian Jiao, Juntao Li, Jian Guo, Nan Duan, Weizhu Chen, et al. Ar-diffusion: Auto-regressive diffusion model for text generation. *Advances in Neural Information Processing Systems*, 36:39957–39974, 2023.

A Experimental Details

A.1 Datasets and Preprocessing

Table 2 summarizes the data. Training uses contiguous clips; evaluation updates the conditioning history with observed, not generated, states.

Synthetic random walk. We use a scalar random walk with initial state $x_1 \sim \mathcal{N}(0, 10^{-4})$ and transition law

$$p(x_{t+1} \mid x_t) = \tfrac{1}{2}\mathcal{N}(x_{t+1}; x_t + 1, \delta^2) + \tfrac{1}{2}\mathcal{N}(x_{t+1}; x_t - 1, \delta^2), \qquad \delta = 10^{-3}.$$

Each increment independently chooses $+1$ or -1 with equal probability and adds Gaussian noise. No normalization is applied.

The task is to forecast the next five states given the observed history. Since the process is Markov, the exact forecast depends only on the latest state x_t. For lead $h = 1, \ldots, 5$, its marginal is

$$p(x_{t+h} \mid x_t) = \sum_{k=0}^{h} 2^{-h} \binom{h}{k} \mathcal{N}(x_{t+h}; x_t + 2k - h, h\delta^2),$$

where k counts positive increments. These analytic marginals provide a ground-truth reference for evaluating predictive distributions.

Beam Spill. The Mu2e simulator (Narayanan, 2022) produces trajectories of scalar beam intensity and control signals. Samples are recorded every $0.1\,\mathrm{ms}$, with 430 samples covering a $43\,\mathrm{ms}$ spill. Temporally correlated disturbances make future intensity uncertain given the observed history.

Our task is conditional forecasting, not controller learning. At each update, models receive observed intensities and the supplied controls over the forecast interval, but not future intensities. They predict the next 10 intensity values, corresponding to $1\,\mathrm{ms}$. We train on 40,000 trajectories and evaluate on 5,000 held-out trajectories, using the supplied intensity and control normalization statistics.

Burgers' equation. Following Wei et al. (2025), we simulate a scalar velocity field $u(x, t)$ governed by the forced one-dimensional Burgers' equation on $x \in [0, 1]$ and $t \in [0, 1.98]$:

$$\begin{aligned} \partial_t u &= -u\,\partial_x u + \nu\,\partial_{xx} u + f, \qquad \nu = 0.01, \\ u(0, t) &= u(1, t) = 0, \qquad u(x, 0) = u_0(x). \end{aligned}$$

Here u_0 is the initial velocity field and f is external forcing. We use 64 interior spatial points and an integration step of 10^{-4}, recording states every 0.02 time units. Initial fields combine two Gaussian profiles with random locations, widths, and amplitudes; forcing combines eight localized space–time Gaussians.

Models observe only the first 32 spatial points and forecast the next 10 velocity fields on that same observed region. Velocities are scaled by $1/10$. The remaining spatial points and the forcing are withheld from the model. Thus, although the full dynamics are deterministic, the observed history does not specify the hidden state and forcing, making the forecast uncertain.

RealPDE-FSI. We use measured (u, v) velocity fields from the fluid–structure interaction dataset in RealPDEBench (Hu et al., 2026). Each state contains two velocity components on a spatial grid. We retain every fourth point of the original 128×128 grid, giving two 32×32 channels without changing the temporal sampling. This coarsening withholds fine-scale spatial information from the model.

The task is to forecast the next 10 velocity fields from the observed history, with both velocity components included in the prediction. Channels are standardized using training statistics. We train on 39 trajectories and evaluate on six held-out trajectories.

Table 2: Dataset settings. L: episode length; H: forecast horizon. Counts are trajectories; field shapes are channels $\times$ grid.

Dataset	State shape	Training	Held-out	L	H
Synthetic	1	50,000	5,000	100	5
Beam Spill	1	40,000	5,000	430	10
Burgers	32	90,000	10,000	100	10
RealPDE-FSI	$2 \times 32 \times 32$	39	6	100	10

A.2 Model and Optimization Settings

Backbones. Synthetic and Beam Spill have scalar states, while Burgers has 32-dimensional observed states; these three tasks use a Transformer backbone. RealPDE-FSI uses a spatiotemporal U-Net for its two-channel spatial velocity fields. Table 3 groups the settings by architecture. Input and output shapes follow Table 2.

Table 3: Backbone configurations. U-Net channel multipliers scale the base width.

Transformer Synthetic, Beam Spill, Burgers		**Spatiotemporal U-Net** RealPDE-FSI	
Hidden width	128	Base width	64
Layers	12	Channel multipliers	$(1, 2, 4)$
Feed-forward width	512	Attention heads	4
Attention heads	4	Head dimension	32

Optimization and training. All four Seq-Flow configurations use AdamW with weight decay 10^{-4} and $(\beta_1, \beta_2) = (0.9, 0.999)$. Table 4 reports the dataset-specific learning rates, batch sizes, warmup steps, and checkpoint steps. We use an exponential moving average (EMA) with decay 0.999. Training combines Gaussian-source and self-rollout losses with equal weights of 0.5, and each training rollout includes four forecast updates.

Table 4: Seq-Flow settings. $C_0/C_{\max}$: initial/maximum context length. Ablations vary NFE or rollout length.

Setting	Synthetic	Beam Spill	Burgers	RealPDE-FSI
Learning rate	5×10^{-4}	5×10^{-4}	5×10^{-4}	2×10^{-4}
Effective batch size	4096	512	64	32
Weight decay	10^{-4}	10^{-4}	10^{-4}	10^{-4}
AdamW (β_1, β_2)	$(0.9, 0.999)$	$(0.9, 0.999)$	$(0.9, 0.999)$	$(0.9, 0.999)$
Warmup steps	10,000	10,000	10,000	1,000
Checkpoint step	12k	36k	36k	10k
Initial/update NFE	$3/3$	$3/3$	$3/3$	$5/3$
Renoising level	0.4	0.4	0.4	0.4
$C_0/C_{\max}$	$1/2$	$9/10$	$9/10$	$9/10$
Forecast windows per episode	95	412	82	82

Sequential evaluation. Initial/update NFE in Table 4 denotes the number of function evaluations for the first forecast and subsequent forecast updates, respectively. The context lengths C_0 and $C_{\max}$ specify the initial and maximum observed history. The four-update training rollout does not limit the evaluation episode: each episode has $L - C_0 - H + 1$ forecast windows, including initialization.