Equilibrium Forcing: Adaptive Video Generation Without Noise Conditioning

Hansen Jin Lillemark[1]* **Alex Rojas**[1]* **Zachary Novack**[1] **Runqian Wang**[2]
Yilun Du[3] **Yian Ma**[1] **Taylor Berg-Kirkpatrick**[1] **Rose Yu**[1]
[1]UC San Diego [2]MIT [3]Harvard University

Abstract

Standard autoregressive video generation algorithms based on Diffusion and Flow Matching rely on rigid training objectives and static sampling schedules, limiting inference procedures from adapting to the data. We introduce Equilibrium Forcing (EQF), a simplified framework for video denoising generative models *without* noise level conditioning. EQF pioneers modular training- and inference-time designs for noise-unconditional generation that decouple learning the denoising field from sampling. This flexibility allows for inference-time algorithms that operate in a closed loop by adapting to feedback from the sample, improving video quality and consistency on challenging autoregressive video generation benchmarks. Extensive analysis elucidates exactly how removing the noise level conditioning enables EQF's data-dependent inference properties to surpass the performance of standard noise level-conditional denoising video methods. Project page: https://equilibriumforcing.github.io/

1 Introduction

Video generation has emerged as a central problem in world modeling, simulation, and interactive content creation, where models must produce temporally coherent video rollouts under limited inference budgets. Algorithms for autoregressive inference generate future frames conditional on previously generated segments, leading to accumulation of local errors over time (Huang et al., 2026). We posit that effective samplers should operate in a closed loop with feedback from the state of the generation procedure, adjusting the sampling procedure to balance video refinement and the remaining budget.

Denoising generative models have become the standard approach for autoregressive video generation, but they currently rely on rigid inference schedules for noise level conditioning that preclude adaptive sampling (Song et al., 2025). Thought to be necessary for stabilizing autoregressive video generation, these inference schedules predetermine how much noise is removed from the sample at each step through noise conditions passed directly to the model. Because the schedule is followed open loop, errors can accumulate between the scheduled noise conditions and the true noise level of the sample. In fact, we find that existing models trained only on pairs of noisy data and their exact noise levels suffer a systematic train-inference mismatch due to this divergence that compounds over autoregressive rollouts and degrades quality.

In this work, we challenge the prevailing assumption that explicit noise conditioning is necessary for stable, high quality autoregressive video generation. We introduce Equilibrium Forcing (EQF), a framework for denoising generative video model training and inference unrestricted by noise level conditioning. EQF improves sampling robustness by learning an *equilibrium* velocity field unified across all noise levels for sampling video data. This contrasts with the brittleness of standard noise-conditional video generation, which uses a chain of nonequilibrium velocity fields calibrated only for samples whose true noise level matches an externally provided condition.

An equilibrium sampling landscape naturally supports data-adaptive inference. Rather than relying on external schedules to determine sample update, EQF uses closed loop feedback from the current

*Equal Contribution. Correspondence to hlillemark@ucsd.edu, a5rojas@ucsd.edu
Preprint.

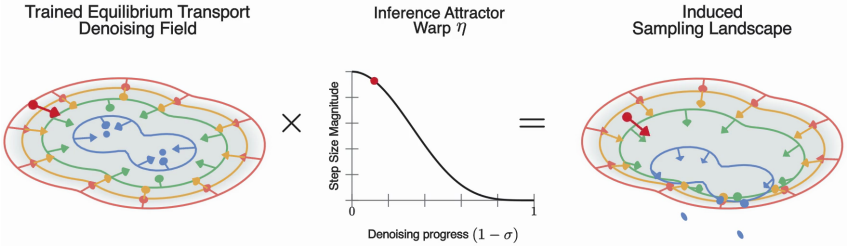

Figure 1: **Equilibrium Forcing Overview.** EQF is trained with a simple unmodulated equilibrium denoising objective (left), which can be shaped at inference time through the η warp (middle). The sampling landscape here is shaped to be an attractor, with the magnitude decreasing as the sample descends and converges (right).

sample to estimate the progress of denoising and modulate the sampling dynamics accordingly. This feedback mechanism can be leveraged for budget-adaptive inference, where the state of generation is used to reallocate the entire remaining sampling trajectory. By connecting transport fields with fixed-point attractor fields at inference time, EQF enables gradient-based samplers to adaptively accelerate inference and converge to higher quality samples (see Figure 1) (Nesterov, 1983).

Prior work on noise-unconditional generation is limited to image generation and suggests that multiple training-time hyperparameters are necessary to remove the reliance on noise level conditions (Wang & Du, 2025). Our approach instead proposes to unify and simplify the design space of denoising models by training with a simpler equilibrium objective, and inducing favorable sampling landscapes at inference time. In addition to removing costly training-time hyperparameter search, we find that prior noise-unconditional methods introduce an unintended loss weighting that is suboptimal in practice. Furthermore, the inference-time decisions for video and image generation are disparate; we study the flexibility uniquely afforded to autoregressive video generation by EQF.

Despite having a simpler training objective and strictly less information than noise-conditional models, EQF improves generation quality and consistency on difficult autoregressive video generation datasets. Extensive experiments attribute these gains to the proposed data-adaptive inference mechanisms, including closed loop feedback, gradient-based samplers, and budget adaptivity. We additionally introduce analysis and experiments that elucidate how noise-unconditional models can indeed learn the same denoising field as noise-conditional models. Our modular framework scales to realistic video generation by finetuning pretrained Flow Matching models at 1.3B and 5B parameters, demonstrating that the advantages of equilibrium samplers can be elicited from existing Flow Matching models through the EQF objective with only a fraction of the pretraining compute (Wan et al., 2025).

2 Background

We begin with an overview of standard denoising generative models for video generation, using Flow Matching as a representative method. We denote sequence variables that have a time dimension with boldface, e.g. the full video $\mathbf{x} = [x_1, x_2, \ldots, x_T]$ is a concatenation of T individual video frames $x_t \in \mathbb{R}^{C\times H\times W}$.

2.1 Flow Matching with Noise Level Conditioning for Video Generation

Flow Matching (Lipman et al., 2023; Liu et al., 2023) models a data distribution $p(\mathbf{x})$ by learning a noise level-dependent velocity field that transports samples from a simple prior distribution $q(\boldsymbol{\epsilon})$ to the data distribution. In this paper, we operate within the Diffusion Forcing framework for video denoising, where each video frame has its own noise level $\boldsymbol{\sigma} := [\sigma_1, \sigma_2, \ldots, \sigma_T] \in [0,1]^T$ (Chen et al., 2024). Samples are generated by solving the probability flow ODE $d\mathbf{x}^{\boldsymbol{\sigma}}/d\boldsymbol{\sigma} = f^{\star}_{\mathrm{FM}}(\mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma})$, with the velocity field approximated with the neural network f_{FM}. The noising process is a linear interpolation of Gaussian noise $\boldsymbol{\epsilon} \sim q(\boldsymbol{\epsilon})$ and clean data $\mathbf{x} \sim p(\mathbf{x})$ applied framewise (denoted $\odot$):

$$\mathbf{x}^{\boldsymbol{\sigma}} = (\mathbf{1} - \boldsymbol{\sigma}) \odot \mathbf{x} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \qquad \epsilon_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I), \qquad \sigma_t \overset{\text{i.i.d.}}{\sim} \mathcal{U}[0,1]. \tag{1}$$

The loss restricts the training distribution of the learned field to pairs of noisy data and the exact corresponding noise level. The network f_{FM} is conditioned on the noise level of the data to directly predict a conditional velocity $\mathbf{v}$, providing the standard Flow Matching loss:

$$\mathbf{v} := \boldsymbol{\epsilon} - \mathbf{x}, \qquad \mathcal{L}_{\mathrm{FM}} = \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon},\boldsymbol{\sigma}} \left[\| f_{\mathrm{FM}}(\mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma}) - \mathbf{v} \|_2^2 \right]. \tag{2}$$

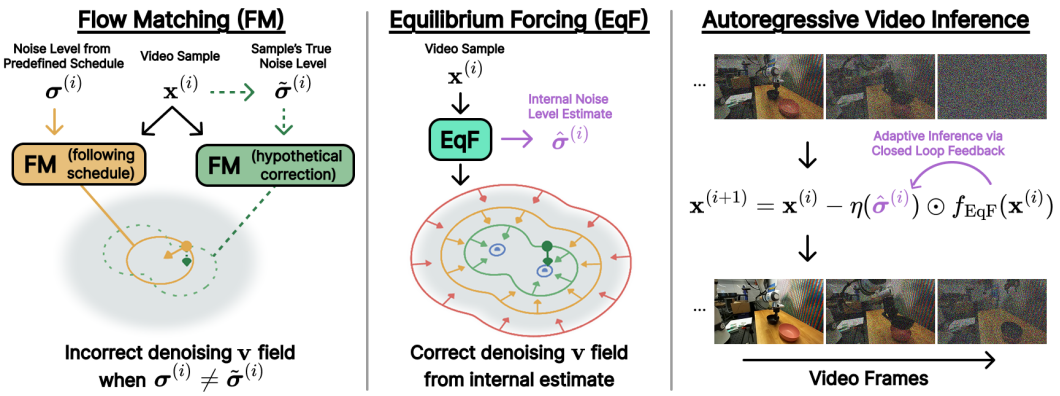

Figure 2: **Left:** FM follows externally-defined noise schedule conditioning, which diverges from the true denoising field. **Center:** By training without explicit noise level conditioning, EQF learns a correct denoising field everywhere by estimating the noise level internally. **Right:** EQF leverages the internal noise level estimate for adaptive autoregressive video generation inference.

For simplicity, we explain inference for non-autoregressive video generation where all frames have the same noise level. Inference initializes from pure noise $\mathbf{x}^{(0)} \sim \mathcal{N}(0, I)$ at $\boldsymbol{\sigma}^{(0)} = \mathbf{1}$ and integrates along a schedule $\{\boldsymbol{\sigma}^{(i)}\}_{i=0,\dots,N}$ until termination at $\boldsymbol{\sigma}^{(N)} = \mathbf{0}$. ODE solvers determine the step size through the difference between noise levels in the schedule (Karras et al., 2022; Zhao et al., 2023). Data at the i-th iteration and noise level $\boldsymbol{\sigma}^{(i)}$ are updated to subsequent noise levels as:

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - (\boldsymbol{\sigma}^{(i)} - \boldsymbol{\sigma}^{(i+1)}) \odot f_{\text{FM}}(\mathbf{x}^{(i)}, \boldsymbol{\sigma}^{(i)}). \tag{3}$$

2.2 Autoregressive Video Denoising

Given ground-truth context or previously generated frames $\mathbf{c}$, autoregressive video generation utilizes a sliding window approach by repeatedly sampling from the distribution $p(\mathbf{x}|\mathbf{c})$. Early attempts relied on bespoke fixed-length conditioning schemes to inject $\mathbf{c}$ as a separate signal (Yang et al., 2025). Instead, Diffusion Forcing-style training covers the case where $\mathbf{x}^{\boldsymbol{\sigma}}$ contains noisy and clean frames simultaneously. This enables context-conditional sampling from $p(\mathbf{x}|\mathbf{c})$ where the clean $\mathbf{c}$ and noisy $\mathbf{x}$ are processed within a unified self-attention window (Song et al., 2025).

Autoregressive inference generalizes the non-autoregressive schedule by assigning each sampling iteration i and temporal index t its own noise level, forming a scheduling matrix over sampling iterations and time (Chen et al., 2024; Ruhe et al., 2024). To support temporal causality, these schedules typically place earlier frames at lower noise levels than later frames, as in Figure 2 (right). Once a frame reaches zero noise, it is committed to the context $\mathbf{c}$, the active window slides forward, and fresh noise is appended for future frames. A full algorithm is given in Appendix I. While such predefined schedules stabilize autoregressive generation, they force the model to follow a fixed denoising path that cannot adapt to the state of the inference procedure (Chen et al., 2024).

3 Divergent Noise Level Conditions in Flow Matching

In this section, we establish that the current design of noise-conditional video denoising models suffers from a lack of data adaptivity, attributable to the open-loop nature of strict noise conditioning schedules. We find that the true noise level diverges from the scheduled noise level during inference, compounding error over autoregressive rollouts. These results suggest that *removing noise conditioning altogether* could unlock data adaptivity that stabilizes inference (see Figure 2).

Scheduled Noise Levels Diverge During Flow Matching Inference. Standard noise-conditional inference algorithms are inherently open loop, unable to respond if the true noise level of the sample $\tilde{\boldsymbol{\sigma}}^{(i)}$ diverges from the scheduled noise level $\boldsymbol{\sigma}^{(i)}$ over an autoregressive rollout. We measure this divergence by training a small framewise noise level predictor $g_\phi : \mathbb{R}^{C\times H\times W} \mapsto [0, 1]$ on the task of regressing the noise level σ from an individually noised frame x^σ. The prediction $\tilde{\sigma} \approx g_\phi(x^\sigma)$ estimates the mode of a sample's true noise level, which we find to estimate the correct value in practice due to posterior concentration of the noise level in high dimensions (see Appendix F).

Under our standard baseline Flow Matching setting on the Minecraft dataset (detailed further in Section 5), we generate by autoregressively sliding a 50-frame window consisting of 25 actively

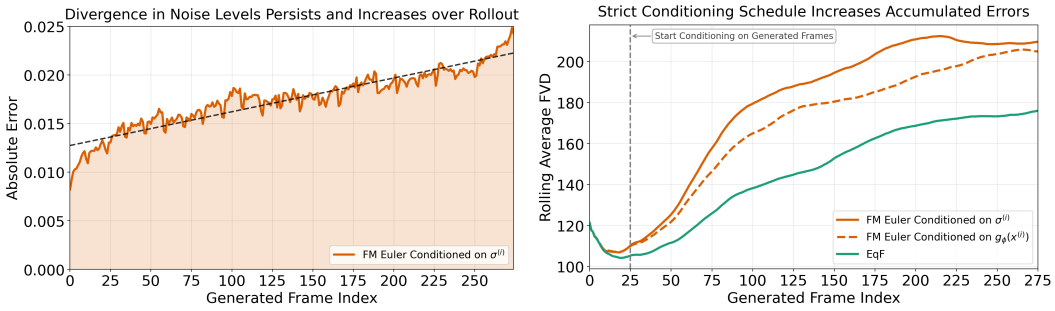

Figure 3: **Left**: Absolute noise conditioning error between scheduled $\sigma^{(i)}$ and the predicted $\tilde{\sigma}^{(i)} \approx g_\phi(x^{(i)})$ increases over 256 autoregressive rollouts. **Right**: FVD worsens over 256 autoregressive rollouts with FM due to externally prescribed noise conditions. Conditioning on estimated noise levels (dotted) helps slightly. Removing the noise condition altogether (EQF) performs best.

predicted frames and 25 clean context frames. Starting with 25 ground truth context frames, we generate 275 frames and compare the scheduled noise level with the predictor-estimated noise level, plotting the absolute residual between the values in Figure 3 (left). If the noise condition is accurate, the divergence should be 0 across the trajectory; we find it is nonzero and increasing over time.

Noise Level Divergence Causes Compounding Error Over Autoregressive Video Rollouts. For static data, an integration error is confined to one sample. However, for sequence data such as videos, accumulated errors can cause the model to be further out of distribution for the next round of generation. We measure the Frechet Video Distance (FVD, Unterthiner et al. (2018)) over 25-frame chunks of the same 275-frame rollouts with respect to 25-frame clips from ground-truth videos. Figure 3 (right) demonstrates that autoregressive errors accumulate steeply in noise conditional inference, especially as generated frames become context. We elaborate on the causal relationship between noise level divergence and compounding error, in addition to further analysis, in Appendix B. We further experiment with attempting to close the loop for Flow Matching with feedback from the current sample during noise-conditional inference. To do so, we condition the Flow Matching model on the predictor-estimated noise level $\tilde{\sigma}$ during inference, with results represented by the dotted line. The error is only partially mitigated, suggesting that fully data-adaptive inference may require removing the reliance on *external* noise conditioning altogether.

4 Equilibrium Forcing

The preceding analysis suggests that inference tied to externally prescribed noise conditioning leads to noise level divergence, resulting in degraded performance over long autoregressive rollouts. EQF addresses this by removing noise conditioning entirely, allowing the denoising process to be adaptively driven by the current sample rather than by a fixed schedule. In this section, we unify recent theory of noise-unconditional denoising with training and inference recipes designed to leverage the unique benefits of an equilibrium sampler for autoregressive video generation.

4.1 Equilibrium Denoising Objective

We introduce EQF, a framework for modeling probability distributions $p(\mathbf{x})$. Instead of learning a separate velocity field for each value of $\boldsymbol{\sigma}$, EQF learns a single field over the data itself, mitigating the effect of the noise conditioning-based error demonstrated in Section 3. Samples are generated by integrating the noise-autonomous ODE $d\mathbf{x}^{\boldsymbol{\sigma}}/d\boldsymbol{\sigma} = f^\star_{\text{EQF}}(\mathbf{x}^{\boldsymbol{\sigma}})$, with the field approximated by a neural network f_{EQF}. Training examples are generated from the same noising process as Equation 1 with independent noise levels per frame. The loss simply drops the noise level conditioning for video data and regresses against the conditional velocity target $\mathbf{v} = \boldsymbol{\epsilon} - \mathbf{x}$:

$$\mathcal{L}_{\text{EQF}} = \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon},\boldsymbol{\sigma}} \left[||f_{\text{EQF}}(\mathbf{x}^{\boldsymbol{\sigma}}) - \mathbf{v}||_2^2 \right]. \tag{4}$$

4.2 Bayes Optimal Denoisers Coincide Under Noise Level Concentration

We next introduce a loss decomposition that establishes how EQF can learn the same denoising field as Flow Matching without conditioning on the noise level, ameliorating the off-manifold integration error established in Section 3. We denote the minimizer of the Flow Matching loss as $f^\star_{\text{FM}}$ and the minimizer of the EQF loss as $f^\star_{\text{EQF}}$.

Algorithm 1: EqF Budget-Adaptive Inference. GD and no autoregressive state for simplicity

```
# f_eqf(x): trained eqf v field
# h_omega(z): noise level estimator
# N: number of sampling steps
sigma_hat = 1
x = randn(sample_shape)
for i in range(N):
    v_hat, activations = f_eqf(x)
    sigma_hat = h_omega(activations)
    # dynamic rescaling
    step_size = sigma_hat / (N - i)
    x = x - step_size * v_hat
return x
```

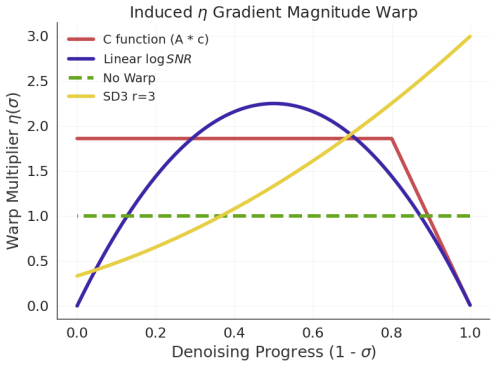

Figure 4: η warp and induced gradient multiplier. η functions that vanish as $\sigma \to 0$ induce an attraction field.

Proposition 4.1. *The optimal* EQF *risk decomposes as*

$$\mathcal{L}^{\star}_{\mathrm{EQF}} = \mathcal{L}^{\star}_{\mathrm{FM}} + \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon},\boldsymbol{\sigma}} \left[\left\| f^{\star}_{\mathrm{FM}}(\mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma}) - f^{\star}_{\mathrm{EQF}}(\mathbf{x}^{\boldsymbol{\sigma}}) \right\|_2^2 \right]. \tag{5}$$

The proof is given in Appendix A.3. The second term in the decomposition measures the excess risk incurred by replacing the noise-conditional Flow Matching Bayes-optimal target $f^{\star}_{\mathrm{FM}}(\mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma})$ with the noise-unconditional EQF Bayes-optimal target $f^{\star}_{\mathrm{EQF}}(\mathbf{x}^{\boldsymbol{\sigma}}) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\mathbb{E}_{\mathbf{x},\boldsymbol{\epsilon}}\left[\mathbf{v} \mid \mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma}\right] \mid \mathbf{x}^{\boldsymbol{\sigma}}\right]$. Since EQF does not receive $\boldsymbol{\sigma}$ explicitly, its Bayes-optimal denoiser averages the fixed-noise targets over the posterior $p(\boldsymbol{\sigma} \mid \mathbf{x}^{\boldsymbol{\sigma}})$.

Recent theoretical work shows that, in high dimensions, this posterior concentrates around a single noise level (Kadkhodaie et al., 2026; Sahraee-Ardakan et al., 2026). Under this concentration, the posterior average collapses to the target corresponding to the identified noise level, making the squared discrepancy in the decomposition small. This suggests that EQF can approximate the FM target pointwise for data in high dimensions, which we provide empirical evidence for in Section 5.5.

4.3 Closed Loop Inference with Noise Level Estimation and η Warp-Induced Dynamics

As minimizing the denoising loss in high dimensions requires the EQF model to internally represent an input's current noise level, an estimate of that noise level provides a natural proxy for sampling progress toward the data manifold. Specifically, we train a lightweight framewise readout model h_ω to predict the mode of the noise level posterior from the EQF model's activations $\mathbf{z}$. This mirrors the prediction task in Section 3, but uses internal model features rather than the raw noisy input $\mathbf{x}^{\boldsymbol{\sigma}}$, which we find to be more performant in closing the loop (see Appendices F,B.3). Driving integration with step sizes modulated by h_ω's estimate of the noise level allows EQF sampling procedures to operate in a closed loop with respect to the status of the sample. We adjust the inference procedure by using a warp function η, defined as a framewise map from the noise level estimate to a step size:

$$(\widehat{\mathbf{v}}^{(i)}, \mathbf{z}^{(i)}) = f_\theta(\mathbf{x}^{(i)}), \qquad \widehat{\boldsymbol{\sigma}}^{(i)} = h_\omega(\mathbf{z}^{(i)}), \tag{6}$$

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta(\widehat{\boldsymbol{\sigma}}^{(i)}) \odot \widehat{\mathbf{v}}^{(i)}. \tag{7}$$

η includes a rescaling factor inversely proportional to the total number of sampling steps N, while the shape of η determines the sampling dynamics. Choosing $\eta(\boldsymbol{\sigma}) \to \mathbf{0}$ as $\boldsymbol{\sigma} \to \mathbf{0}$ warps the sampling problem into a fixed-point **attractor** search for $\eta(\boldsymbol{\sigma}) \odot f_{EqF}(\mathbf{x}) = \mathbf{0}$. This contrasts with **transport-style** sampling (such as standard FM Euler integration), where the general nonzero constraint $\eta(\boldsymbol{\sigma}) \odot f_{\mathrm{EQF}}(\mathbf{x}) = -\eta(\mathbf{0})\mathbf{x}$ holds as $\boldsymbol{\sigma} \to \mathbf{0}$.

4.4 Choice of η Warp Function

Applying attractor-style η warp functions on top of EQF's trained transport field is powerful for unlocking the toolkit of gradient-based optimization solvers that converge to a fixed point. In particular, we implement Nesterov Accelerated Gradient (NAG) within this closed loop sampling framework, with the full algorithm in Appendix H. The usage of these solvers under open-loop, noise-conditional sampling was prohibitive: steps with adaptive, unscheduled step sizes would make the sample jump to unknown noise levels, leading to significantly divergent noise-level conditioning and off-manifold errors.

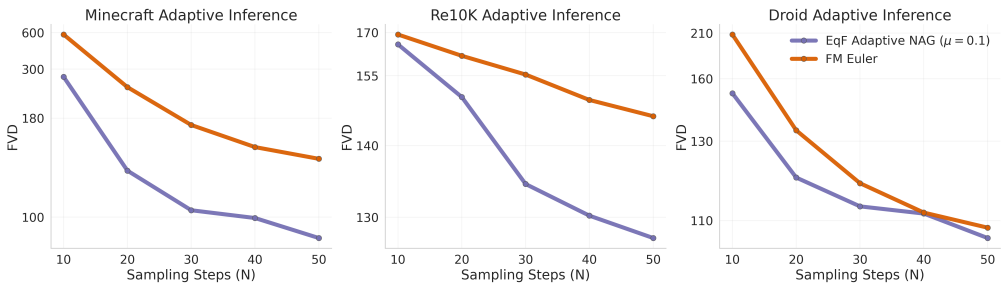

Figure 5: **Budget-adaptive inference results.** EQF with budget-adaptive sampling achieves the lowest FVD across Minecraft, Re10K, and Droid.

DPM-Solver's linear log-SNR schedule and the c-function of EqM are examples of η warps that induce attractor-style dynamics, while the SD3 reparameterization and identity map (using evenly spaced steps) are transport warps (Lu et al., 2022; Esser et al., 2024; Wang & Du, 2025). We visualize these warp function options in Figure 4 and evaluate these warps experimentally in Section 5. The η warp function is closely related to established noise level-reparameterizations of sampling ODEs for denoising generative models, differing in that η explicitly specifies the derivative of the reparameterization rather than a grid of noise levels; further interpretation is in Appendix G.

4.5 Budget-Adaptive Sampling Algorithm

We additionally propose Algorithm 1, a sampler designed to leverage EQF's equilibrium field and noise level estimation capabilities in a budget-aware manner. Standard samplers specify a schedule before inference begins, meaning the update at step i is determined by the precomputed grid in η-warp space. While EQF closed loop-only inference (Equation 7) has the ability to reindex into the predefined η-function with the noise level estimate, it retains the rescaling implied by the total original budget N. Instead, Algorithm 1 incorporates both the estimated noise level and the remaining sampling budget $N - i$ to reallocate the denoising trajectory at each iteration. This budget-adaptivity can be seen as a version of the η function that defines a simple update rule including the number of steps remaining. It avoids overshooting as the sample approaches the data manifold when accelerated with methods like NAG. We present this algorithm as just one instantiation of the myriad possible adaptive algorithms enabled by EQF; we describe an additional adaptive early stopping algorithm that leverages the noise level to reduce spurious additional denoising steps in Appendix C.1.

5 Experiments

Here, we present results on autoregressive video generation experiments, demonstrating how EQF's data adaptivity leads to performance gains over noise-conditional and noise-unconditional baselines.

5.1 Experimental Setup and Baselines

For all video generation experiments, we follow standard practice and process the video through a VAE to retrieve compressed video latents, which are then noised independently per-frame (see Rombach et al. (2021) and Section 2.2). We consider a set of baselines representative of current methods for video denoising generative models: **1) Flow Matching (FM)** is implemented as explained in Section 2.1. We use standard Euler integration with warped noise levels. **2) Diffusion** uses the DDPM training objective from Ho et al. (2020) with the DDIM sampler (Song et al., 2022). **3) Equilibrium Matching (EqM)** drops noise conditioning and trains with a modulated velocity magnitude to bake in a warped sampling landscape during training (Wang & Du, 2025). **4) Noise Unconditional-Flow Matching (NU-FM)** follows Sun et al. (2025)'s training and inference recipe without noise level conditioning, adapted to video.

All baseline models have matched compute and data budgets. We evaluate with EMA weights and autoregressive context conditioning (see Section 2.2 and Appendix I). We report FVD (Unterthiner et al., 2018) to measure generated video quality and diversity, and include VBench scores (Huang et al., 2024) that provide several metrics for video consistency. Extended information on the experimental setting and evaluation is provided in Appendix D.

5.2 Training Equilibrium Video Denoising Models From Scratch

We train models from scratch on clips of 50 frames from the Minecraft dataset, a video dataset with per-frame action conditions (Yan et al., 2023). All models use the same CogVideoX-based backbone

Table 1: Autoregressive Minecraft 300-frame rollout results across 5 seeds. EQF with NAG sampling achieves the best FVD and VBench scores.

Method	Sampler	Inference Warp	FVD↓	VBench↑
EQF (Ours)	NAG, $\mu = 0.3$	c-function	$\mathbf{64.34 \pm 1.78}$	$\mathbf{0.7733 \pm 0.0008}$
EqM (Wang & Du, 2025)	NAG $\mu = 0.1$	–	77.02 ± 5.21	0.7679 ± 0.0009
EQF	GD	c-function	77.63 ± 3.31	0.7721 ± 0.0005
FM	Euler	c-function	104.13 ± 4.33	0.7703 ± 0.0012
NU-FM (Sun et al., 2025)	GD	–	101.16 ± 3.57	0.7698 ± 0.0011
Diffusion	DDIM	–	106.00 ± 1.22	0.7681 ± 0.0008

Figure 6: Visualizations of autoregressively generated RealEstate10K videos using EQF-FT and FM-FT finetuned models from Wan 2.1.

denoising model architecture with an active generation window of 50 frames but with different training objectives (Yang et al., 2025). We drive closed loop inference for EQF with a readout predictor h_ω as explained in Section 4.3. All training details are elaborated upon in Appendix D. Each experiment evaluates on action-conditioned video generation of 256 videos, with 275 future frames generated given 25 ground-truth context frames; strong performance on this long autoregressive rollout requires generations to remain stable and consistent over many shifts of the active window.

The main results on Minecraft are presented in Table 1 with 250 sampling steps per frame and a sliding window of 25 active sampling frames and 25 context frames. Qualitative results are visualized in Figure 11 and on the project page here. EQF achieves the best FVD and VBench scores, with the best setting combining closed-loop inference, a vanishing η function, and NAG.

Across various computational budgets with the standard sampling algorithm, EQF sets a frontier in performance on the same experimental setting in Minecraft as shown by Figure 8. We find that larger inference budgets permit increased acceleration through a higher setting of the momentum parameter μ. We hypothesize that with excessively large updates under a coarser discretization, higher momentum carries stale gradient information to low noise levels and risks overshooting the data manifold. Additional inference-time decisions, including the use of the budget-adaptive sampling algorithm and the effect of closed-loop acceleration with NAG are studied further below.

5.3 Finetuning Large-Scale Flow Matching Models to EQF

Beyond training EQF models from scratch, we investigate whether equilibrium denoising fields are already present inside large pretrained Flow Matching models. As established in Section 4.2, noise level posteriors concentrate in high dimensions, implying that the Bayes-optimal EQF and Flow Matching denoisers coincide pointwise. This suggests that finetuning with the EQF objective can induce an internal estimator of the noise level, transforming the noise-conditional field into an equilibrium field. We test this hypothesis through *equilibrium finetuning* on two datasets.

Equilibrium Finetuning on the RealEstate10K Dataset. Starting from Wan 2.1 T2V 1.3B (Wan et al., 2025), a large scale video model trained with the Flow Matching objective, we remove the external noise-conditioning pathway by zeroing the noise level condition and continue training with the EQF objective. Specifically, we finetune on 49-frame clips from RealEstate10K (Re10K), a realistic video dataset of house tours with camera annotations (Zhou et al., 2018). Afterwards, we train a simple σ-readout predictor following a similar approach to Minecraft. Models converge within 100k steps; we denote the EQF-finetuned model as EQF-FT, and the FM finetuned model as FM-FT. Further finetuning information can be found in Appendices D, E.

We evaluate on camera-conditioned autoregressive generation of 256 videos, where 152 future frames are generated from 37 ground-truth context frames. Sampling uses 50 steps per frame with a sliding window of 12 active frames and 37 context frames, with the stride also set to 12. Quantitative results

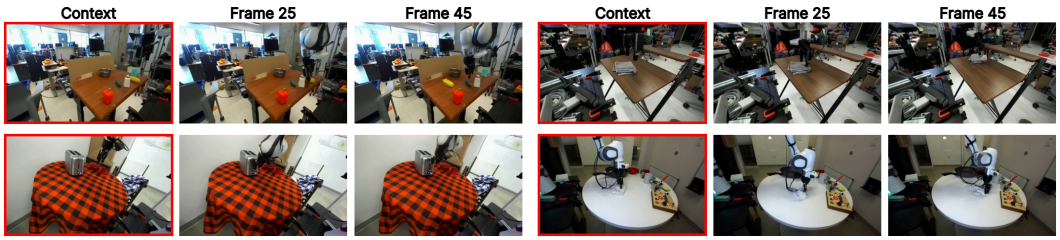

Figure 7: Generated videos from the Droid dataset using EQF-FT finetuned from Wan 2.2.

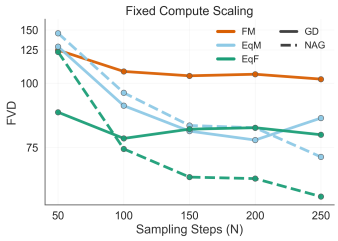

Figure 8: EQF is optimal across compute scales on Minecraft compared with noise-conditional and -unconditional baselines.

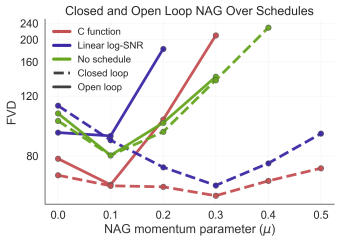

Figure 9: NAG is optimal with closed-loop and attractor (c-function) η warp, compared to open-loop and transport warps.

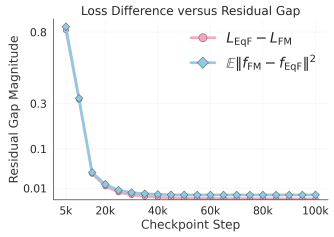

Figure 10: The FM/EQF loss difference and the expected residual gap between the denoising field matches across Re10K finetuning.

are reported in Table 4 and qualitative results are visualized both in Figure 6 and on the project page here. EQF-FT with closed loop sampling and NAG improves upon FM, corroborating our hypothesis about the limiting nature of noise conditioning, and demonstrating that equilibrium finetuning can unlock data-adaptive sampling even at scale. Results on an additional setting where less initial context is given (29 frames) using the budget-adaptive algorithm are reported in the next section.

Equilibrium Finetuning on the Droid Dataset. We perform the same equilibrium finetuning procedure on an additional robot manipulation dataset at higher resolution with a larger scale model to demonstrate these results are not unique to a single model/dataset pair. Specifically, we finetune the Wan 2.2 TI2V 5B model on 49-frame clips from the Droid dataset, a language-conditioned video dataset of robots performing diverse tasks (Khazatsky et al., 2024). The Droid model represents approximately a $4\times$ increase in both model parameters and data resolution over the Re10K model. We evaluate on prompt-conditioned generation of 256 videos, where 36 video frames are generated from 13 video frames of context. Results are discussed in the next subsection.

5.4 Inference-Time Decisions for Equilibrium Forcing

Budget-Adaptive Inference. We evaluate the budget-adaptive sampling algorithm presented in Section 4.5 on the three datasets, with the inference setting described in the previous sections. Standard EQF is run with NAG and $\mu = 0.1$; budget-adaptive EQF uses an identity map η function, and is also run with NAG and $\mu = 0.1$. Results are available in Table 3 and Figure 5. Qualitative results for the Droid dataset under this setting are available in Figure 7. Under a low computational budget of 50 sampling steps or fewer, we find that the adaptive algorithm with NAG consistently sets a lower frontier of performance compared to FM and standard EQF with NAG, demonstrating the practical utility of EQF's data-adaptive flexibility across model sizes and datasets.

Closed-Loop Acceleration with NAG. We study the interplay between inference acceleration, η function properties, and closed loop noise level feedback with h_ω on the same Minecraft dataset setup in Table 2. We test three choices for η with differing properties in addition to an identity map. Our results confirm that shaping the landscape into an attractor at inference time with $\eta(\boldsymbol{\sigma}) \odot f_{\text{EQF}}(\mathbf{x}) = \mathbf{0}$ at the data manifold is necessary for NAG to converge due to its minimum-seeking property. The best setting combines attractor-like schedules (either linear log-SNR or c-function), closed-loop prediction, and the higher acceleration parameter μ that closed-loop prediction enables, displayed in Figure 9. Conversely, the SD3 warp is a non-attractor landscape with $\eta(\boldsymbol{\sigma}) \odot f_{EqF}(\mathbf{x}) = -r\mathbf{x}$ at the data manifold (see Appendix G); as such, NAG-based sampling with SD3 is prone to overshooting the fixed-point, exhibiting unreliable performance and lagging behind other schedules.

Online Replanning Inference. We design a novel experimental setting to evaluate the robustness of EQF and the baselines to reutilize previous predictions when new ground truth context is observed,

Table 2: Ablations of the EQF model on 300-frame Minecraft generation across warp schedules, samplers, and μ for one seed. The reported run for each row corresponds to the best setting of μ. Closed loop inference is necessary for higher settings of μ.

η Warp	Feedback	Sampler	μ	FVD↓
c Function	Closed Loop	NAG	0.3	**64.12**
	Open Loop	NAG	0.1	67.94
	Open Loop	GD	0	78.69
Linear log-SNR	Closed Loop	NAG	0.3	67.70
	Open Loop	NAG	0.1	90.59
	Open Loop	GD	0	92.54
SD3 $r = 3$	Closed Loop	NAG	0.1	203.73
	Open Loop	NAG	0.1	114.11
	Open Loop	GD	0	137.85
No Schedule	Closed Loop	NAG	0.1	80.25
	Open Loop	NAG	0.1	80.25
	Open Loop	GD	0	105.44

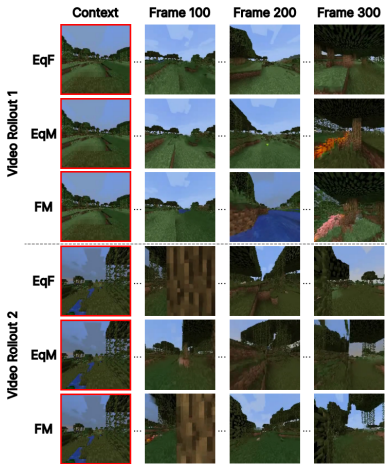

Figure 11: 300-frame autoregressively generated Minecraft videos.

mirroring a realistic world modeling setting. We find that the flexibility of EQF allows it to better utilize previous predictions without needing to fully recompute the entire future prediction. Specifically, to reach the same FVD as Adaptive EQF for each computation threshold, Flow Matching requires between 13% and 29% more compute. The full experimental results and details are in Appendix J.

5.5 Empirical Evidence for Vanishing Gap in Proposition 4.1

Proposition 4.1 predicts that for EQF to achieve loss parity with FM, it must reproduce FM's field from $\mathbf{x}^\sigma$ alone. As evidence that the gap closes, we compare the predictions made by FM and EQF models on identical held-out noisy inputs. We find that across datasets, the predictions are highly correlated (> 0.98) and the residual gap accounts for as little as 2.4% of the denoiser output magnitude. Figure 10 further shows that over Re10K finetuning, the gap between the FM and EQF losses closely tracks the empirical difference in the learned denoising field. Thus EQF approximately recovers the FM velocity field, meaning any performance gains are from differences in the inference procedure. By doing so without noise level conditioning, this substantiates our hypothesis that EQF implicitly identifies the relevant noise level from $\mathbf{x}^\sigma$. See Appendix A.4 for more details.

6 Related Work

6.1 Autoregressive Video Generation

Denoising-based autoregressive generative models have already shown tremendous potential for video generation (Brooks et al., 2024). The predominant challenge in autoregressive video generation lies in producing stable rollouts that adhere to the context, currently addressed by introducing strict inference schedules (Feng et al., 2024; Song et al., 2025). Our results suggest that in fact, these schedules may not be necessary for stable autoregressive generation, and that the noise conditioning they rely upon may be a key hindrance to the model at inference time. Another approach for stabilizing autoregressive rollouts conditioned on generated frames post-trains directly on these generated frames to ameliorate a general train-inference mismatch (Huang et al., 2026). As analyzed in Section 3 and Appendix B.2, we have addressed one aspect of train-inference mismatch by removing the noise level condition; we see other post-training approaches as coming from a different, but synergistic angle.

6.2 Generative Modeling without Noise Level Conditioning

While a few prior works have studied generative modeling without noise level conditioning, they have primarily focused on image generation as the setting. In addition to being the first work to our knowledge to explore the implications of removing noise level conditioning for temporal data such as video, we clarify the contributions we make over these prior works here.

Noise-Unconditional Flow Matching. Sun et al. (2025) introduced a training recipe for noise-unconditional Flow Matching (NU-FM), though their experiments did not improve over Flow Matching on large-scale image datasets, nor did they explore data-adaptive sampling. NU-FM's training recipe, when adapted to video generation through the framework presented in Section 2.2, can be instantiated as a base inference setting of EQF without any data-adaptive sampling applied on top.

Table 3: Comparison of FVD and VBench at 50 sampling steps across Minecraft, RealEstate10K, and Droid datasets using the budget-adaptive EQF algorithm. NAG $\mu = 0.1$.

	Minecraft		Re10K		Droid	
Method	FVD ↓	VBench ↑	FVD ↓	VBench ↑	FVD ↓	VBench ↑
EQF Adaptive NAG	$\mathbf{91.45 \pm 0.92}$	0.771 ± 0.000	$\mathbf{127.69 \pm 1.57}$	$\mathbf{0.761 \pm 0.000}$	$\mathbf{107.38 \pm 0.82}$	$\mathbf{0.812 \pm 0.002}$
EQF NAG	92.08 ± 0.30	$\mathbf{0.774 \pm 0.000}$	166.73 ± 2.98	0.759 ± 0.000	113.58 ± 3.51	0.810 ± 0.002
FM	135.37 ± 1.74	0.769 ± 0.000	145.42 ± 5.93	$\mathbf{0.761 \pm 0.000}$	108.69 ± 3.71	0.809 ± 0.000

Table 4: Autoregressive rollout results on RealEstate10K, finetuned from pretrained Flow Matching model Wan 2.1 1.3B (denoted with -FT). NAG is run with $\mu = 0.1$.

Method	**Sampler**	**FVD↓**	**VBench↑**
EQF-FT	Adaptive, NAG	$\mathbf{139.5 \pm 3.93}$	$\mathbf{0.7472 \pm 0.001}$
EQF-FT	NAG	142.0 ± 5.80	$\mathbf{0.7472 \pm 0.001}$
EQF-FT	GD	149.5 ± 6.06	0.7470 ± 0.000
FM-FT	Euler	174.3 ± 2.17	0.7429 ± 0.001

Table 5: ImageNet with EQF's framework (c Function η)

Method	**Sampler**	**FID↓**
EQF	NAG, $\mu = 0.3$	$\mathbf{15.15 \pm 0.09}$
EQF	GD	15.39 ± 0.09
EqM	NAG, $\mu = 0.3$	15.74 ± 0.10
EqM	GD	15.91 ± 0.08
FM	Euler	18.91 ± 0.10

Our results with NU-FM substantiate the hypothesis that data-adaptive sampling is the key aspect that allows noise-unconditional models to improve over traditional models.

Equilibrium Matching. Equilibrium Matching (EqM) removes explicit noise conditioning from Flow Matching and introduces a modulated training target designed to make the magnitude of the learned field vanish at clean data, enabling attractor-style sampling for image generation (Wang & Du, 2025). However, EqM requires a training-time hyperparameter search over both the shape c and magnitude λ of the field. In this work, we demonstrate that separating the noise-unconditional denoising objective from the sampling dynamics is advantageous: EQF learns an unmodified velocity field and only induces an attractor landscape at inference time through the η warp. In addition to offering the simplification of separating training and inference-time decisions, this enables stronger performance via data-adaptive sampling. Our framework further removes an unintended loss weighting introduced by EqM's target modulation, which we find to be suboptimal in practice. In fact, we analyze further how our model subsumes all possible EqM training-time hyperparameters with inference-time decisions in Appendices A.1, A.2.

Comparison on ImageNet. Though EQF is designed for video data, our insights can be applied on ImageNet to compare against prior noise-unconditional image models (Deng et al., 2009). We train an EQF model using the same settings as the EqM paper, following the SiT architecture, and train the XL/2 size model to 80 epochs (Ma et al., 2024; Wang & Du, 2025). The results in Table 5, using the inference-time c-function η warp across 3 seeds, demonstrate that image generation also benefits from data-driven adaptivity and modular training- and inference-time decisions.

7 Conclusion

By training on a simplified noise-unconditional objective, EQF leverages unique adaptivity and closed-loop inference capabilities to improve quality and consistency on challenging autoregressive video generation tasks. It further elucidates the advantages that come from a modular decoupling of noise-unconditional training and attractor-like inference, surpassing the performance of prior noise-unconditional frameworks. More broadly, our results challenge the assumption that explicit noise conditioning and strict inference schedules are necessary for modern video denoising models. As video becomes an increasingly central modality for world modeling, simulation, and interactive content generation, it is important to distinguish the components that are necessary for stable generation from those adding unnecessary complexity that ultimately constrain performance.

EQF as presented in this paper carries some limitations. Though EQF's framework extends to general spatiotemporal denoising generative models, we only consider video in the paper's primary experiments. We establish that the error in the noise level readout model is small in Appendix F, but it could still contribute a source of error, such as if the partially noised sample is off the training distribution. Finally, while the datasets considered here are large, they are not on the general video domain. Future work pretraining EQF at a larger scale could support more general conclusions; the purpose of this present work is to compare the performance of EQF and the baselines under a matched training budget. Training for longer or using data mixtures as is done in Chen et al. (2025) could yield additional improvements in quality and instruction following on our datasets.

Acknowledgments and Disclosure of Funding

This work is supported in part by the U.S. Army Research Office under Army-ECASE award W911NF-07-R-0003-03; the U.S. Department of Energy, Office of Science, ARPA-H-SOL-24-101 program; IARPA HAYSTAC Program; DARPA YFA; NSF Grants #2146151, #2205093, #2146343, #2134274, #2441832, and CCF-2112665; and CDC-RFA-FT-23-0069. This work has been made possible in part by a gift from the Chan Zuckerberg Initiative Foundation to establish the Kempner Institute for the Study of Natural and Artificial Intelligence at Harvard University.

References

Brooks, T., Peebles, B., Holmes, C., DePue, W., Guo, Y., Jing, L., Schnurr, D., Taylor, J., Luhman, T., Luhman, E., Ng, C., Wang, R., and Ramesh, A. Video generation models as world simulators, 2024. URL `https://openai.com/research/video-generation-models-as-world-simulators`. OpenAI technical report, February 15, 2024.

Cachay, S. R., Aittala, M., Kreis, K., Brenowitz, N. D., Vahdat, A., Mardani, M., and Yu, R. Elucidated rolling diffusion models for probabilistic forecasting of complex dynamics. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL `https://openreview.net/forum?id=NnRQOPzL9P`.

Carreira, J. and Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

Chen, B., Martí Monsó, D., Du, Y., Simchowitz, M., Tedrake, R., and Sitzmann, V. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024. URL `https://arxiv.org/abs/2407.01392`.

Chen, B., Zhang, T., Geng, H., Zhang, C., Li, P., Song, K., Freeman, W. T., Malik, J., Abbeel, P., Tedrake, R., et al. Large video planner enables generalizable robot control. *arXiv preprint arXiv:2512.15840*, 2025.

Decart, Etched, McIntyre, Q., Campbell, S., Chen, X., and Wachen, R. Oasis: A universe in a transformer. *Decart Blog*, 2024. URL `https://oasis-model.github.io`.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Du, Y., Yang, S., Dai, B., Dai, H., Nachum, O., Tenenbaum, J., Schuurmans, D., and Abbeel, P. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Feng, R., Zhang, H., Yang, Z., Xiao, J., Shu, Z., Liu, Z., Zheng, A., Huang, Y., Liu, Y., and Zhang, H. The matrix: Infinite-horizon world generation with real-time moving control, 2024. URL `https://arxiv.org/abs/2412.03568`.

Hafner, D., Yan, W., and Lillicrap, T. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.

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

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

Huang, X., Li, Z., He, G., Zhou, M., and Shechtman, E. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *Advances in Neural Information Processing Systems*, 38: 167283–167308, 2026. URL https://arxiv.org/abs/2506.08009.

Huang, Z., He, Y., Yu, J., Zhang, F., Si, C., Jiang, Y., Zhang, Y., Wu, T., Jin, Q., Chanpaisit, N., et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21807–21818, 2024.

Kadkhodaie, Z., Pooladian, A.-A., Chewi, S., and Simoncelli, E. Blind denoising diffusion models and the blessings of dimensionality. *arXiv preprint arXiv:2602.09639*, 2026.

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

Khazatsky, A., Pertsch, K., Nair, S., Balakrishna, A., Dasari, S., Karamcheti, S., Nasiriany, S., Srirama, M. K., Chen, L. Y., Ellis, K., et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Kingma, D. and Gao, R. Understanding diffusion objectives as the elbo with simple data augmentation. *Advances in Neural Information Processing Systems*, 36:65484–65516, 2023. URL https://arxiv.org/abs/2303.00848.

Lillemark, H., Huang, B., Zhan, F., Du, Y., and Keller, T. A. Flow equivariant world models: Structured memory for dynamic environments. In *Forty-third International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2601.01075.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

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

Nesterov, Y. A method for unconstrained convex minimization problem with the rate of convergence o (1/k2). In *Dokl. Akad. Nauk. SSSR*, volume 269, pp. 543, 1983.

Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023. URL https://arxiv.org/abs/2212.09748.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685, 2021. URL https://arxiv.org/abs/2112.10752.

Ruhe, D., Heek, J., Salimans, T., and Hoogeboom, E. Rolling diffusion models. *arXiv preprint arXiv:2402.09470*, 2024.

Sahraee-Ardakan, M., Delbracio, M., and Milanfar, P. The geometry of noise: Why diffusion models don't need noise conditioning. *arXiv preprint arXiv:2602.18428*, 2026.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2256–2265. PMLR, 2015.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

Song, K., Chen, B., Simchowitz, M., Du, Y., Tedrake, R., and Sitzmann, V. History-guided video diffusion. In *Forty-second International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2502.06764.

Sun, Q., Jiang, Z., Zhao, H., and He, K. Is noise conditioning necessary for denoising generative models? In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=pTSWi6RTtJ.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In Dasgupta, S. and McAllester, D. (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/sutskever13.html.

Unterthiner, T., Van Steenkiste, S., Kurach, K., Marinier, R., Michalski, M., and Gelly, S. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

Wan, T., Wang, A., Ai, B., Wen, B., Mao, C., Xie, C.-W., Chen, D., Yu, F., Zhao, H., Yang, J., et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Wang, R. and Du, Y. Equilibrium matching: Generative modeling with implicit energy-based models. *arXiv preprint arXiv:2510.02300*, 2025.

Yan, W., Hafner, D., James, S., and Abbeel, P. Temporally consistent transformers for video generation. In *International Conference on Machine Learning*, pp. 39062–39098. PMLR, 2023.

Yang, Z., Teng, J., Zheng, W., Ding, M., Huang, S., Xu, J., Yang, Y., Hong, W., Zhang, X., Feng, G., et al. Cogvideox: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations*, volume 2025, pp. 83048–83077, 2025. URL https://arxiv.org/abs/2408.06072.

Zhao, W., Bai, L., Rao, Y., Zhou, J., and Lu, J. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36: 49842–49869, 2023.

Zhou, S., Du, Y., Zhang, S., Xu, M., Shen, Y., Xiao, W., Yeung, D.-Y., and Gan, C. Adaptive online replanning with diffusion models. *Advances in Neural Information Processing Systems*, 36: 44000–44016, 2023.

Zhou, T., Tucker, R., Flynn, J., Fyffe, G., and Snavely, N. Stereo magnification: learning view synthesis using multiplane images. *ACM Trans. Graph.*, 37(4), 2018.

Appendix Contents

A Analysis on Noise Unconditional Denoising

In this section, we elaborate on how EQF subsumes EqM as a training and inference time framework for noise unconditional denoising.

A.1 Effect of EqM Modulation on Target

In the denoising generative modeling literature, the objective used during training under finite compute has been observed to impact the perceptual quality of samples generated by the learned denoiser (Karras et al., 2022; Esser et al., 2024). We demonstrate here that the EqM objective explored in Wang & Du (2025) induces a weighting over noise levels in the corresponding clean-data prediction objective due to the extra modulation term. For $c(\sigma)$ considered in EqM, the weighting places comparatively more emphasis on high-noise samples than the EQF objective.

Proposition A.1 (Induced x-prediction loss weightings)**.** *The EqM and* EQF *velocity losses are equivalent to the following weighted clean data prediction losses:*

$$\mathcal{L}_{\mathrm{EqM}} = \mathop{\mathbb{E}}_{x,\sigma,\epsilon}\left[\frac{c(\sigma)^2}{\sigma^2}\left\|x_{\mathrm{EqM}}(x^\sigma,\sigma)-x\right\|_2^2\right], \tag{8}$$

$$\mathcal{L}_{\mathrm{EQF}} = \mathop{\mathbb{E}}_{x,\sigma,\epsilon}\left[\frac{1}{\sigma^2}\left\|x_{\mathrm{EQF}}(x^\sigma,\sigma)-x\right\|_2^2\right]. \tag{9}$$

Proof. For simplicity, we analyze the case where there is only a single noise level in the sample, σ, though the argument generalizes to when $\boldsymbol{\sigma}$ is T-dimensional. Let

$$v := \epsilon - x, \qquad x^\sigma := (1-\sigma)x + \sigma\epsilon. \tag{10}$$

Using the definition of v, the noised sample can equivalently be written as

$$\begin{aligned} x^\sigma &= (1-\sigma)x + \sigma\epsilon \\ &= x + \sigma(\epsilon - x) \\ &= x + \sigma v. \end{aligned} \tag{11}$$

Therefore, the clean data can be recovered from the noised sample and the velocity target:

$$x = x^\sigma - \sigma v. \tag{12}$$

EqM trains a noise-unconditional network to predict a modulated velocity target $c(\sigma)v$:

$$\mathcal{L}_{\mathrm{EqM}} = \mathop{\mathbb{E}}_{x,\sigma,\epsilon}\left[\left\|f_{\mathrm{EqM}}(x^\sigma) - c(\sigma)v\right\|_2^2\right]. \tag{13}$$

To understand the effect that $c(\sigma)$ has on the loss weighting, we can convert this objective to x-prediction. Since $f_{\mathrm{EqM}}(x^\sigma)$ is trained to estimate $c(\sigma)v$, the corresponding estimate of v is $f_{\mathrm{EqM}}(x^\sigma)/c(\sigma)$. We can then define the clean-data prediction induced by EqM by substituting back into Equation 12. To convert, x_{EqM} becomes reliant on σ as well as the noised data; x_{EqM} is an induced x-prediction model based on the trained v-prediction function.

$$x_{\mathrm{EqM}}(x^\sigma,\sigma) := x^\sigma - \frac{\sigma}{c(\sigma)} f_{\mathrm{EqM}}(x^\sigma). \tag{14}$$

Using Equations 12 and 14, we can now rewrite the EqM loss in terms of the error between the induced x-prediction and the true x:

$$\begin{aligned} x_{\mathrm{EqM}}(x^\sigma,\sigma) - x &= \left(x^\sigma - \frac{\sigma}{c(\sigma)} f_{\mathrm{EqM}}(x^\sigma)\right) - (x^\sigma - \sigma v) \\ &= \sigma v - \frac{\sigma}{c(\sigma)} f_{\mathrm{EqM}}(x^\sigma) \\ &= -\frac{\sigma}{c(\sigma)}\left(f_{\mathrm{EqM}}(x^\sigma) - c(\sigma)v\right). \end{aligned} \tag{15}$$

Rearranging gives

$$f_{\mathrm{EqM}}(x^\sigma) - c(\sigma)v = -\frac{c(\sigma)}{\sigma}\left(x_{\mathrm{EqM}}(x^\sigma;\sigma) - x\right). \tag{16}$$

The original loss for the modulated target prediction becomes the following reweighted x-loss

$$\begin{aligned}\mathcal{L}_{\mathrm{EqM}} &= \mathop{\mathbb{E}}_{x,\sigma,\epsilon}\left[\left\|f_{\mathrm{EqM}}(x^\sigma) - c(\sigma)v\right\|_2^2\right] \\ &= \mathop{\mathbb{E}}_{x,\sigma,\epsilon}\left[\frac{c(\sigma)^2}{\sigma^2}\left\|x_{\mathrm{EqM}}(x^\sigma,\sigma) - x\right\|_2^2\right].\end{aligned} \tag{17}$$

The noise-level weight is given by $w_{\mathrm{EqM}} = \frac{c(\sigma)^2}{\sigma^2}$. For example, if $c(\sigma) = \sigma$, then the weighting term becomes $w_{\mathrm{EqM}} = 1$, yielding an unweighted clean-data prediction objective:

$$\mathcal{L}_{\mathrm{EqM}} = \mathop{\mathbb{E}}_{x,\sigma,\epsilon}\left[\left\|x_{\mathrm{EqM}}(x^\sigma) - x\right\|_2^2\right]. \tag{18}$$

In contrast, EQF trains directly on the unmodulated velocity target:

$$\mathcal{L}_{\mathrm{EQF}} = \mathop{\mathbb{E}}_{x,\sigma,\epsilon}\left[\left\|f_{\mathrm{EQF}}(x^\sigma) - v\right\|_2^2\right]. \tag{19}$$

The corresponding clean-data estimate is

$$x_{\mathrm{EQF}}(x^\sigma,\sigma) := x^\sigma - \sigma f_{\mathrm{EQF}}(x^\sigma). \tag{20}$$

Repeating the same calculation,

$$\begin{aligned}x_{\mathrm{EQF}}(x^\sigma,\sigma) - x &= (x^\sigma - \sigma f_{\mathrm{EQF}}(x^\sigma)) - (x^\sigma - \sigma v) \\ &= -\sigma\left(f_{\mathrm{EQF}}(x^\sigma) - v\right),\end{aligned} \tag{21}$$

which implies

$$\begin{aligned}\mathcal{L}_{\mathrm{EQF}} &= \mathop{\mathbb{E}}_{x,\sigma,\epsilon}\left[\left\|f_{\mathrm{EQF}}(x^\sigma) - v\right\|_2^2\right] \\ &= \mathop{\mathbb{E}}_{x,\sigma,\epsilon}\left[\frac{1}{\sigma^2}\left\|x_{\mathrm{EQF}}(x^\sigma,\sigma) - x\right\|_2^2\right].\end{aligned} \tag{22}$$

□

Comparing Equations 17 and 22, EqM and EQF differ by the factor $c(\sigma)^2$ in their induced x-prediction weighting. EQF's loss weighting more heavily weights low-noise regions, similar to Flow Matching.

Theoretically, all such weightings across noise levels σ do not change the pointwise Bayes-optimal denoising target for noise conditional models (Kingma & Gao, 2023). The same theoretical conclusion holds for noise-unconditional models under the assumption that the noise level posterior concentrates in high dimensions and is thus directly identifiable (studied more in Appendix A.2). However, in practice, finite capacity models are sensitive to how the training signal is distributed across noise levels; state-of-the-art denoising generative models for images have found weightings with increased emphasis on high noise levels (such as the one in EqM) to be less performant (Karras et al., 2022; Esser et al., 2024). Similar results have been found under close examination of noise-sampling distributions in autoregressive denoising generative modeling for complex dynamical systems (Cachay et al., 2026). Our experimental results in Section 5 comparing samples from EqM to EQF corroborate these findings: the EQF loss places less relative emphasis on the highest-noise regions compared to EqM and achieves better results.

A.2 EqM c-Function Inference Relationship to EQF

Here, we continue the discussion from Section 4.3 on how EQF modularly decouples training and inference. We show that, under concentration of the noise-level posterior established in Sahraee-Ardakan et al. (2026); Kadkhodaie et al. (2026), an EqM model trained with target modulation $c(\boldsymbol{\sigma})$ is equivalent to applying the same c-function as an inference-time warp to the EQF velocity field. Thus, in the Bayes-optimal and posterior concentration limit, a trained EQF model can subsume EqM-style target modulation by moving the c-function to the sampler via the warp η, as in Equation 7.

Proposition A.2 (EqM training-time modulation can be rewritten as an inference-time EQF warp)**.** *The EqM Bayes-optimal denoiser satisfies*

$$\begin{aligned}f^\star_{\mathrm{EqM}}(\mathbf{x}^{\boldsymbol{\sigma}}) &\approx c(\widehat{\boldsymbol{\sigma}}) \odot f^\star_{\mathrm{EQF}}\left(\mathbf{x}^{\boldsymbol{\sigma}}\right) \\ &\propto \eta(\widehat{\boldsymbol{\sigma}}) \odot f^\star_{\mathrm{EQF}}\left(\mathbf{x}^{\boldsymbol{\sigma}}\right),\end{aligned} \tag{23}$$

where the final proportionality holds when the inference-time warp η is chosen to have the same shape as c, up to normalization.

Proof. Recall the noising process and velocity target from the main text:

$$\mathbf{v} := \boldsymbol{\epsilon} - \mathbf{x}, \qquad \mathbf{x}^{\boldsymbol{\sigma}} := (1 - \boldsymbol{\sigma}) \odot \mathbf{x} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}.$$

For a squared loss regression objective, the Bayes-optimal denoiser is the conditional expectation of the target given the model input. For EQF, the model input is only $\mathbf{x}^{\boldsymbol{\sigma}}$, and the training target is the unmodulated velocity $\mathbf{v}$, yielding the following Bayes-optimal denoiser:

$$\mathcal{L}_{\text{EQF}} = \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon},\boldsymbol{\sigma}} \left[||f_{\text{EQF}}(\mathbf{x}^{\boldsymbol{\sigma}}) - \mathbf{v}||_2^2 \right], \qquad f^{\star}_{\text{EQF}}(\mathbf{x}^{\boldsymbol{\sigma}}) = \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon},\boldsymbol{\sigma}} \left[\mathbf{v} \mid \mathbf{x}^{\boldsymbol{\sigma}} \right]. \tag{24}$$

Applying the law of total expectation by first conditioning on $\boldsymbol{\sigma}$ gives an outer expectation over possible noise levels under $p(\boldsymbol{\sigma}|\mathbf{x}^{\boldsymbol{\sigma}})$, and an inner expectation over possible clean data and noise pairs under $p(\mathbf{x}, \boldsymbol{\epsilon}|\mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma})$:

$$f^{\star}_{\text{EQF}}(\mathbf{x}^{\boldsymbol{\sigma}}) = \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon}} \left[\mathbf{v} \mid \mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma} \right] \mid \mathbf{x}^{\boldsymbol{\sigma}} \right]. \tag{25}$$

EqM has the same model input, but the training target is the modulated velocity $c(\boldsymbol{\sigma}) \odot \mathbf{v}$.

$$\begin{aligned} f^{\star}_{\text{EqM}}(\mathbf{x}^{\boldsymbol{\sigma}}) &= \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon},\boldsymbol{\sigma}} \left[c(\boldsymbol{\sigma}) \odot \mathbf{v} \mid \mathbf{x}^{\boldsymbol{\sigma}} \right] \\ &= \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[c(\boldsymbol{\sigma}) \odot \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon}} \left[\mathbf{v} \mid \mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma} \right] \mid \mathbf{x}^{\boldsymbol{\sigma}} \right]. \end{aligned} \tag{26}$$

Under concentration in high dimensions (Sahraee-Ardakan et al., 2026; Kadkhodaie et al., 2026), the posterior estimate of the noise level given noisy data collapses to a Dirac delta function at a point mass at $\widehat{\boldsymbol{\sigma}}$. Thus, any outer expectation over $\boldsymbol{\sigma}$ for any function g when conditioned on $\mathbf{x}^{\boldsymbol{\sigma}}$ collapses to evaluation at $\widehat{\boldsymbol{\sigma}}$:

$$p(\boldsymbol{\sigma} \mid \mathbf{x}^{\boldsymbol{\sigma}}) \approx \delta(\boldsymbol{\sigma} - \widehat{\boldsymbol{\sigma}}), \qquad \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[g(\boldsymbol{\sigma}) \mid \mathbf{x}^{\boldsymbol{\sigma}} \right] \approx g(\widehat{\boldsymbol{\sigma}}). \tag{27}$$

Applying this to Equations 25 and 26 and combining, with a final assumption of the EQF sampler using an inference-time warp η with the same shape as c, gives

$$f^{\star}_{\text{EQF}}(\mathbf{x}^{\boldsymbol{\sigma}}) \approx \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon}} \left[\mathbf{v} \mid \mathbf{x}^{\boldsymbol{\sigma}}, \widehat{\boldsymbol{\sigma}} \right], \tag{28}$$

$$\begin{aligned} f^{\star}_{\text{EqM}}(\mathbf{x}^{\boldsymbol{\sigma}}) &\approx c(\widehat{\boldsymbol{\sigma}}) \odot \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon}} \left[\mathbf{v} \mid \mathbf{x}^{\boldsymbol{\sigma}}, \widehat{\boldsymbol{\sigma}} \right] \\ &\approx c(\widehat{\boldsymbol{\sigma}}) \odot f^{\star}_{\text{EQF}}\left(\mathbf{x}^{\boldsymbol{\sigma}}\right) \\ &\propto \eta(\widehat{\boldsymbol{\sigma}}) \odot f^{\star}_{\text{EQF}}\left(\mathbf{x}^{\boldsymbol{\sigma}}\right). \end{aligned} \tag{29}$$

□

As described in Appendix G, a valid transport warp must integrate to 1, so the proportionality between c and η should not be interpreted as saying that every EqM c-function directly defines a valid probability-flow transport. The EqM paper chose to define the c-function generally such that $c(\sigma) \to 0$ as $\sigma \to 0$, which does not in general define a valid transport. EqM introduced another training-time hyperparameter λ to modulate the magnitude of the c-function. When taken together with the inference-time step size, EqM approximates this constraint empirically, falling short of identifying the theoretically correct value and requiring large, expensive training-time sweeps over both λ and the c-function that are unnecessary in our framework.

Together with Appendix A.1, this demonstrates that EqM's training-time target modulation is *theoretically unnecessary* under the stated noise-level concentration and introduces an undesirable weighting over noise levels during training, as we confirm experimentally in Section 5. In high dimensions, this suggests that an equilibrium landscape together with attractor-style sampling are better treated as modular inference time decisions rather than being baked into the training objective.

A.3 Proof of Proposition 4.1

In this section we expand the loss decomposition in Proposition 4.1. This decomposition demonstrates that the loss for a denoiser without noise level conditioning is lower bounded by the one with noise level conditioning, and that if the EQF denoiser can minimize the pointwise denoiser error for any particular noise level, it can achieve the same optima.

Proof. The Bayes-optimal denoiser for Flow Matching is the conditional expectation of the target given the model input:

$$f^\star_{\text{FM}}(\mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma}) = \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon}} \left[\mathbf{v} \mid \mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma} \right].$$

For brevity, throughout this proof we write

$$f^\star_{\text{FM}} := f^\star_{\text{FM}}(\mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma}), \qquad f^\star_{\text{EQF}} := f^\star_{\text{EQF}}(\mathbf{x}^{\boldsymbol{\sigma}}).$$

By the definition of $f^\star_{\text{FM}}$, the Flow Matching residual has zero conditional mean:

$$\mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon}} \left[\mathbf{v} - f^\star_{\text{FM}} \mid \mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma} \right] = \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon}} \left[\mathbf{v} \mid \mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma} \right] - f^\star_{\text{FM}} = \mathbf{0} \tag{30}$$

We now expand the optimal EQF risk by adding and subtracting $f^\star_{\text{FM}}$:

$$\begin{aligned}
\mathcal{L}^\star_{\text{EQF}} &= \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon},\boldsymbol{\sigma}} \left[\left\| \mathbf{v} - f^\star_{\text{EQF}} \right\|_2^2 \right] \\
&= \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon},\boldsymbol{\sigma}} \left[\left\| (\mathbf{v} - f^\star_{\text{FM}}) + \left(f^\star_{\text{FM}} - f^\star_{\text{EQF}} \right) \right\|_2^2 \right] \\
&= \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon},\boldsymbol{\sigma}} \left[\left\| \mathbf{v} - f^\star_{\text{FM}} \right\|_2^2 \right] + \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon},\boldsymbol{\sigma}} \left[\left\| f^\star_{\text{FM}} - f^\star_{\text{EQF}} \right\|_2^2 \right] \\
&\quad + 2 \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon},\boldsymbol{\sigma}} \left[\left\langle \mathbf{v} - f^\star_{\text{FM}}, f^\star_{\text{FM}} - f^\star_{\text{EQF}} \right\rangle \right] \\
&= \mathcal{L}^\star_{\text{FM}} + \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon},\boldsymbol{\sigma}} \left[\left\| f^\star_{\text{FM}} - f^\star_{\text{EQF}} \right\|_2^2 \right] \\
&\quad + 2 \mathop{\mathbb{E}}_{\mathbf{x}^{\boldsymbol{\sigma}},\boldsymbol{\sigma}} \left[\mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon}} \left[\left\langle \mathbf{v} - f^\star_{\text{FM}}, f^\star_{\text{FM}} - f^\star_{\text{EQF}} \right\rangle \mid \mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma} \right] \right] \\
&= \mathcal{L}^\star_{\text{FM}} + \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon},\boldsymbol{\sigma}} \left[\left\| f^\star_{\text{FM}} - f^\star_{\text{EQF}} \right\|_2^2 \right] \\
&\quad + 2 \mathop{\mathbb{E}}_{\mathbf{x}^{\boldsymbol{\sigma}},\boldsymbol{\sigma}} \left[\left\langle \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon}} \left[\mathbf{v} - f^\star_{\text{FM}} \mid \mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma} \right], f^\star_{\text{FM}} - f^\star_{\text{EQF}} \right\rangle \right] \\
&= \mathcal{L}^\star_{\text{FM}} + \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon},\boldsymbol{\sigma}} \left[\left\| f^\star_{\text{FM}} - f^\star_{\text{EQF}} \right\|_2^2 \right] + 2 \mathop{\mathbb{E}}_{\mathbf{x}^{\boldsymbol{\sigma}},\boldsymbol{\sigma}} \left[\left\langle \mathbf{0}, f^\star_{\text{FM}} - f^\star_{\text{EQF}} \right\rangle \right] \\
&= \mathcal{L}^\star_{\text{FM}} + \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon},\boldsymbol{\sigma}} \left[\left\| f^\star_{\text{FM}}(\mathbf{x}^{\boldsymbol{\sigma}}, \boldsymbol{\sigma}) - f^\star_{\text{EQF}}(\mathbf{x}^{\boldsymbol{\sigma}}) \right\|_2^2 \right].
\end{aligned} \tag{31}$$

□

Considering the form of the Bayes-optimal denoisers for Flow Matching and EQF (Equation 25), recall that EQF marginalizes Flow Matching's Bayes-optimal denoisers over $p(\boldsymbol{\sigma} \mid \mathbf{x}^{\boldsymbol{\sigma}})$. As in Equation 27 if this posterior concentrates (Sahraee-Ardakan et al., 2026; Kadkhodaie et al., 2026), the Bayes-optimal denoiser is approximately:

$$p(\boldsymbol{\sigma} \mid \mathbf{x}^{\boldsymbol{\sigma}}) \approx \delta(\boldsymbol{\sigma} - \widehat{\boldsymbol{\sigma}}) \implies f^\star_{\text{EQF}}(\mathbf{x}^{\boldsymbol{\sigma}}) \approx \mathop{\mathbb{E}}_{\mathbf{x},\boldsymbol{\epsilon}} \left[\mathbf{v} \mid \mathbf{x}^{\boldsymbol{\sigma}}, \widehat{\boldsymbol{\sigma}} \right]$$

Taking into account that the gap $\mathcal{L}^\star_{\text{EQF}} - \mathcal{L}^\star_{\text{FM}}$ is the expected difference in regression targets, under concentration the EQF model must drive the gap to zero to minimize the objective, suggesting that to do so it must learn to implicitly estimate the noise level posterior.

A.4 FM-to-EQF Residual Gap

Here, we provide the full table for the experimental analysis provided in Section 5.5. We evaluate the paired FM and EQF predictions on identical held-out noisy inputs. Table 6 shows that EQF obtains a denoising loss only slightly larger than FM, matching the lower bound prediction from the main text. The comparatively small FM/EQF residual explains the gap in the loss, especially so for Re10K and Droid. Their predicted velocity fields obtain low normalized discrepancy and high Pearson correlations (> 0.98) across all three datasets. We additionally show the convergence of the output denoising field Pearson correlation across training of the Re10K model in Figure 12. Droid is evaluated on only 1,417 samples since that is the size of the validation dataset. These results support the conclusion that EQF approximately recovers the FM velocity field without explicit noise conditioning, meaning the gains in performance come from getting rid of the drifted noise conditions and from the application of adaptive algorithms.

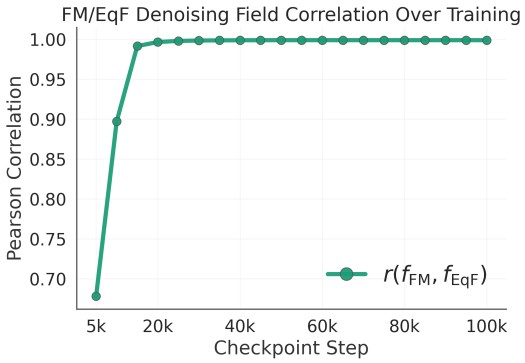

Figure 12: Pearson Correlation convergence across training of the Re10K FM and EQF models.

Table 6: **Empirical agreement between trained FM and EQF denoisers.** All quantities are evaluated on matched held-out samples using the same clean inputs, noise levels, and Gaussian noise. Relative field MSE denotes the magnitude of $\mathbb{E}\|f_{\text{FM}} - f_{\text{EQF}}\|_2^2$ normalized by $\frac{1}{2}\big(\mathbb{E}\|f_{\text{FM}}\|_2^2 + \mathbb{E}\|f_{\text{EQF}}\|_2^2\big)$. Across all three domains, the trained fields have small normalized discrepancy and high Pearson correlation despite EQF receiving no explicit noise-level condition.

Dataset	**# Samples**	**FM/EQF Field MSE**	**Validation v-Loss**		**Relative Field MSE** (%)	**Pearson Correlation**
		$\mathbb{E}\|f_{\text{FM}} - f_{\text{EQF}}\|_2^2$	$\mathbb{E}\|f_{\text{FM}} - \mathbf{v}\|_2^2$	$\mathbb{E}\|f_{\text{EQF}} - \mathbf{v}\|_2^2$		
Minecraft	2,084	0.0425 ± 0.00505	0.576 ± 0.00270	0.576 ± 0.00271	7.378	0.984
Re10K	4,096	0.00347 ± 0.00003	0.142 ± 0.00146	0.143 ± 0.00147	2.432	0.999
Droid	1,417	0.0135 ± 0.000385	0.307 ± 0.00972	0.318 ± 0.00993	4.325	0.996

B Additional Analysis on the Effect of Divergent Noise Levels in Flow Matching

In this section, we provide further analysis on the effect of noise conditioning divergence during autoregressive inference in Flow Matching. Our analysis suggests a causal relationship between this noise level divergence and resulting degraded performance at inference time.

B.1 Flow Matching Predictions Frequently Terminate Scheduled Sampling at Nonzero Noise

In Section 3, we observe a persistent residual between the estimated true noise level of the sample and the scheduled noise level during Flow Matching inference. We use the same experimental setup over 300-frame Minecraft autoregressive rollouts here and visualize the progress of the predicted noise level of the sample during inference with a fixed number of denoising steps. The solid line in Figure 13 represents the mean of the distribution of framewise noise levels over inference, and the dotted line is the scheduled noise level at a particular solver step using the c-function as η. Any gap between the lines at end of the figure demonstrates that sampling with Flow Matching ends with a certain amount of noise remaining, and the miss rates plotted indicate the frequency with which a sample terminates below a specified noise level. Zooming into the end of the denoising process, we see that scheduled inference consistently terminates with a nonzero noise level, indicating that insufficient denoising is a source of error during Flow Matching inference. This further impacts autoregressive inference in that the conditioning signal for *generated context* is also inaccurate, not just over the horizon as analyzed in Section 3.

B.2 Noise Conditioning Divergence Directly Causes Increased Denoising Loss

To measure the direct effect of divergent noise level conditions per inference step on Flow Matching, we noise data to a particular level using the forward process, and then condition the model on an artificially perturbed noise level that is not the same as the true noise level of the data point. Matching our autoregressive inference setting (Section 3 and Section 5), we consider denoising windows of 50 frames from the Minecraft dataset. We apply the forward process with noise level σ_{stable} to the latter 25 frames, constituting the prediction horizon, while retaining a clean context of length 25 frames. We study the effect of incorrect noise levels on the denoising loss when the noise level is correct on different combinations of the 25 active horizon frames and the 25 context frames.

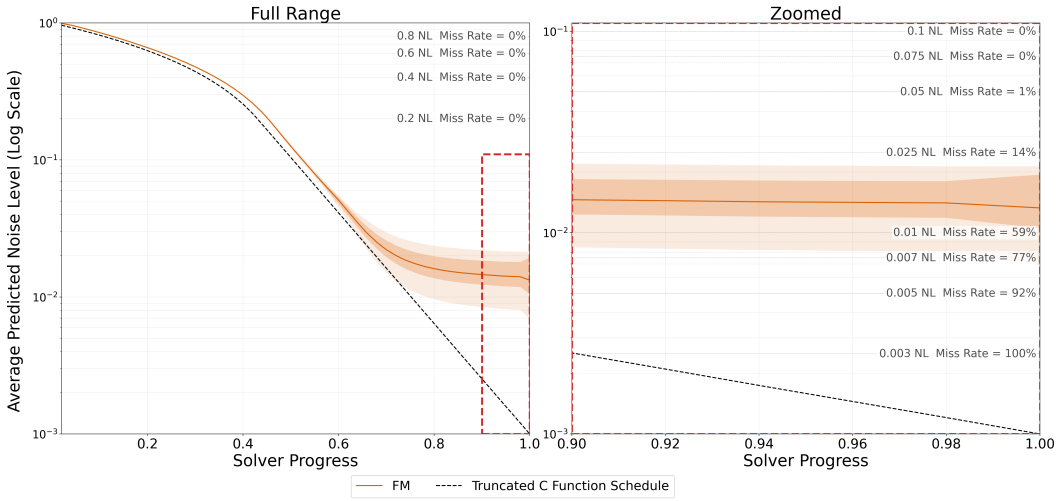

Figure 13: Noise level readout across sampling steps. Ideally, sampling ends at noise level 0, but Flow Matching samples terminate at a significant, nonzero noise level, which is a potential contributing factor to autoregressive error accumulation. The right shows a zoomed view of the red box from the left plot.

Noise Level Divergence on the Active Horizon Causes Denoising Error. We first simulate the noise level divergence only over the prediction horizon. The presence of such errors, where the scheduled noise level is different than the estimated true noise level during inference, is established by Section 3. We give the correct noise level of $\mathbf{0}$ for the 25 context frames, and we give a simulated diverged noise level condition $(\sigma_{\text{stable}} + \delta) \cdot \mathbf{1}$ for the prediction horizon. Specifically, the true noise level of the data is $\tilde{\boldsymbol{\sigma}} = [\mathbf{0}, \sigma_{\text{stable}} \cdot \mathbf{1}]$, and the actual conditioning signal is $\boldsymbol{\sigma} = [\mathbf{0}, (\sigma_{\text{stable}} + \delta) \cdot \mathbf{1}]$.

The discrepancy between $\tilde{\boldsymbol{\sigma}}$ and $\boldsymbol{\sigma}$ causes a significant increase in the denoising v-loss for the Flow Matching model, as shown in Figure 14 (left). At the bottom of each 'bowl' is the denoising loss when the condition matches the data, and the surrounding perturbed points are normalized as the increase in the loss caused by the mismatch. We thus see that as the conditioned noise level diverges further from the truth, the error monotonically increases. This discrepancy, attributable to rigid noise conditioning during Flow Matching training, is one reason for the decreased downstream performance once noise levels diverge.

Inaccurate Context Noise Level Causes Further Denoising Loss. In Subsection B.1, we observe that when following a scheduled path of noise levels with Flow Matching, a sample often terminates at nonzero noise. During autoregressive rollouts, the inference procedure relies on the conditioning signal of generated frames to be accurate. This suggests that any further inaccuracy in the context conditioning signal may further harm performance.

To simulate the effect of generated context terminating at nonzero noise, we measure the v-loss when the noise level condition is corrupted for both the context and the prediction horizon. We measure the loss of a Flow Matching model when paired with the incorrect noise level of $\delta \cdot \mathbf{1}$ for the context and $(\sigma_{\text{stable}} + \delta) \cdot \mathbf{1}$ for the prediction horizon. Specifically, the true noise level of the noisy data is $\tilde{\boldsymbol{\sigma}} = [\mathbf{0}, \sigma_{\text{stable}} \cdot \mathbf{1}]$, and the actual conditioning signal is $\boldsymbol{\sigma} = [\delta \cdot \mathbf{1}, (\sigma_{\text{stable}} + \delta) \cdot \mathbf{1}]$.

When comparing the loss from this experiment to the previous case when only the prediction horizon's signal is corrupted, we find that the loss with mismatched noise level conditions for both the context and prediction horizon is strictly higher than the case where only the horizon is corrupted, as visualized in Figure 14 (right).

B.3 Simulated Closed-Loop Flow Matching Sampling Improves Sample Quality

Here, we demonstrate that using the predicted noise level from $\tilde{\sigma} \approx g_\phi(x^\sigma)$ for Flow Matching inference can improve performance of Flow Matching, but that directly conditioning on noise levels still produces worse quality compared to EQF. More details on the noise level estimator g_ϕ can be found in Appendix F. The evaluation setting is identical to Sections 3 and 5, with 300-frame rollouts on Minecraft.

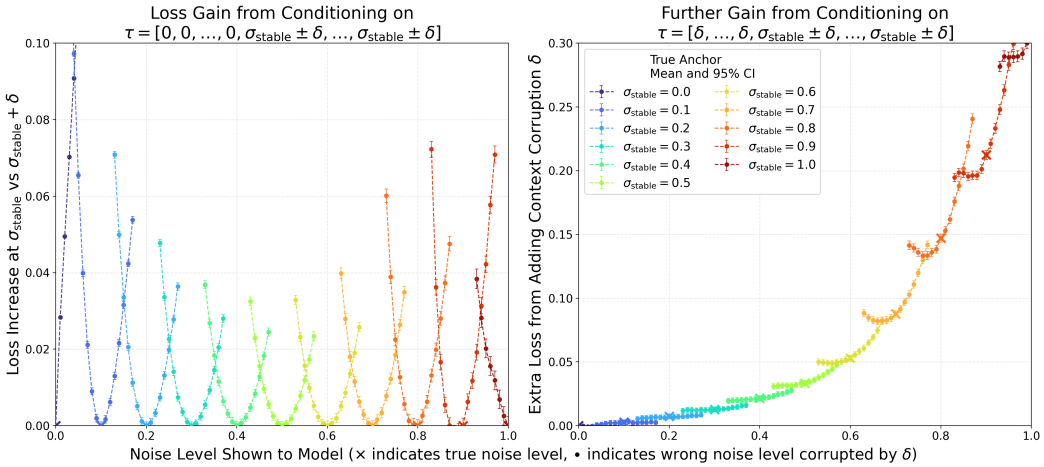

Figure 14: Noise conditioning error introduced artificially, with the loss calculated only on the active sampling horizon. Left: horizon only. Right: gain from adding context corruption. Noise conditions are clamped to $[0, 1]$, and errors are standardized with respect to $\delta \in [-0.07, 0.07]$.

Table 7: Flow Matching autoregressive generation results under different inference schedules and conditioning choices. Conditioning on the inferred noise level $g_\phi(x^{(i)})$ substantially improves FVD, especially when combined with NAG under the c Function inference schedule.

Method	Sampler	Inference Schedule	Conditioning	FVD↓
Flow Matching	Euler	Identity	$\sigma^{(i)}$	126.633
Flow Matching	Euler	Identity	$\tilde{\sigma}^{(i)} \approx g_\phi(x^{(i)})$	121.145
Flow Matching	Euler	c-Function	$\sigma^{(i)}$	103.610
Flow Matching	Euler	c-Function	$\tilde{\sigma}^{(i)} \approx g_\phi(x^{(i)})$	94.488
Flow Matching	NAG, $\mu = 0.2$	c-Function	$\sigma^{(i)}$	1045.633
Flow Matching	NAG, $\mu = 0.2$	c-Function	$\tilde{\sigma}^{(i)} \approx g_\phi(x^{(i)})$	**81.193**

Accelerated gradient-based samplers such as NAG typically do not work for noise-conditional models, since the sample would step to unknown noise levels, causing severe conditioning divergence and highly inaccurate denoising. As shown in Table 7, using the predicted noise levels to close the sampling loop leads to significant improvements. This is especially the case for NAG, where the sample diverges completely without the closed-loop correction. While these results for closed-loop sampling with Flow Matching demonstrate improvements over naive open-loop sampling schedules, we find that these fixes are just extra steps that can be removed when using EQF's framework. Specifically, closing the loop by *removing noise conditioning altogether* and relying on the model's internal estimates of progress produces much stronger results, as our main results in Section 5 demonstrate EQF's improvements in the same evaluation setting.

B.4 Closed Loop Sampling with EQF Produces the Lowest Final Noise Level

In Appendix B.1, we observe that Flow Matching samples consistently terminate at nonzero noise. Here, we provide additional comparisons between the noise level termination values of samples from closed-loop EQF sampling, and of samples from Flow Matching and from NU-FM (Sun et al., 2025). In Figure 15, the black dashed line represents the scheduled noise level, and the result for each sampler is shown by different colors.

The closed-loop samples with EQF terminate at lower noise levels (almost an order of magnitude lower than Flow Matching), exhibiting the robustness of closing the loop with a noise-unconditional sampler via its internal estimator. Furthermore, we can see that sampling with NAG lowers the noise level that the denoised sample reaches, as well as accelerates the denoising progress overall (the noise level readout curve falls significantly below the schedule). This analysis provides further intuition for how closed-loop sampling with NAG produces higher quality samples by converging better to $\eta(\widehat{\boldsymbol{\sigma}}) f_{\text{EQF}}(\mathbf{x}) = \mathbf{0}$, as demonstrated experimentally in Section 5.

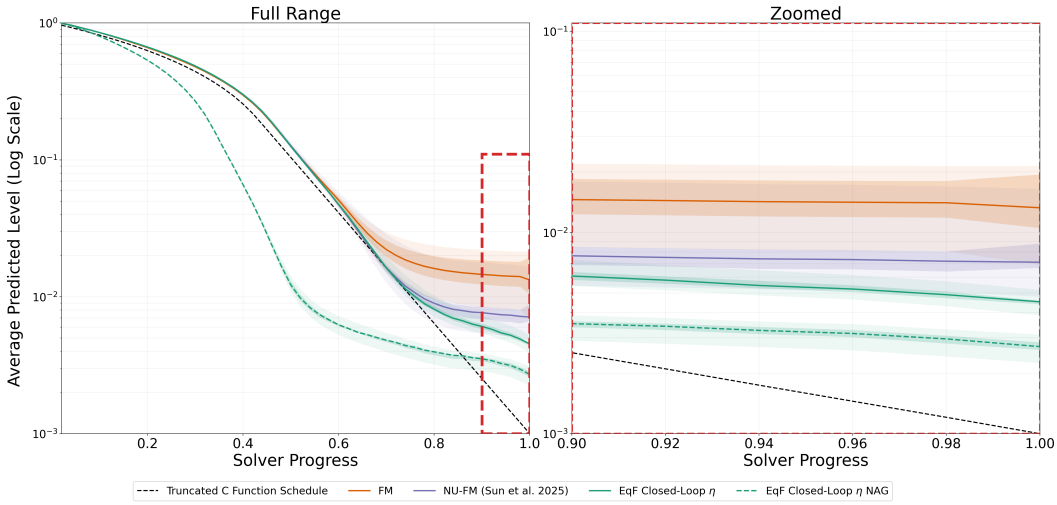

Figure 15: Comparison of the noise level that sampling terminates at across FM, NU-FM, EQF, and EQF with NAG. Ideally, sampling ends at a noise level of 0. Closed-loop sampling with EQF terminates at a lower noise level, and NAG accelerates the schedule and ends at an even lower noise level. The right shows a zoomed view of the red box from the left plot.

C Additional Results

In this section, we describe an additional data-adaptive early stopping algorithm and results on Minecraft. Further, we present full tables for the budget-adaptive results in the main text, and an additional inpainting task on the Re10K dataset.

C.1 Adaptive Early Stopping Algorithm

The noise level estimation capabilities of EQF provide a natural convergence criterion for a data-adaptive early stopping algorithm, presented in Algorithm 2. Once the estimated noise level falls below a threshold, $\widehat{\sigma} < \sigma_{\text{thresh}}$, the sampler enters a short refinement period with a decaying step size before completing denoising for those indices. This refinement period implements the budget-adaptive algorithm presented in Section 4.5 for a few additional 'cleanup' steps, removing the final amount of noise. Noise-driven adaptive early stopping contrasts with how prior work defines stopping criteria for noise-unconditional inference based on the gradient magnitude falling below a threshold (Wang & Du, 2025). Previously, it was unclear how this threshold should be chosen with relation to the convergence of the data; the magnitude of the update itself is relatively arbitrary, and would have to be tuned for each dataset.

On the Minecraft dataset, we demonstrate that the adaptive early stopping algorithm effectively reduces the number of steps taken under different values of the stopping criteria σ_{thresh}, and in some cases also improves the performance over constant-budget samples. The results are presented in Figure 16 and Table 8. These results highlight the flexibility of EQF's inference framework: compute requirements can be derived from the sample itself rather than from a fixed hyperparameter of the sampler. We hypothesize that it can perform better than standard EQF algorithm in some cases due to the ability to "finish" the denoising based on the noise estimate and the number of steps left, whereas the standard EQF can leave too much noise in the sample, especially when taking a lower number of steps.

The inference setting is described in Appendix D.2; it is different than the other experiments, so the FVD numbers are not directly comparable. We use a number of cleanup steps $C_s = 8$, and the value of σ_{thresh} ranges from 0 (no early stopping) to 0.7. The detection of σ_{thresh} triggers based on the highest noise level of a frame within the next chunk of frames to be emitted, and the number of cleanup steps is run for that chunk of frames; multiple chunks of frames can be in 'cleanup mode' simultaneously.

C.2 Budget-Adaptive Full Results

The full results for the budget-adaptive experiments from Section 5 on the Minecraft, Re10K, and Droid datasets are detailed in Table 9.

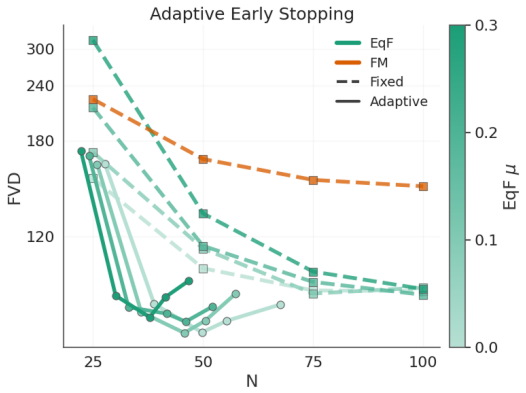

Figure 16: **Early Stopping on Minecraft with EQF.** The early stopping algorithm using noise level readouts optimizes the compute/performance tradeoff. Each value of μ is evaluated with stopping thresholds $\sigma_{\text{thresh}} = 0.1, 0.2, 0.3, 0.5, 0.7$

Algorithm 2: EQF Adaptive Early Stopping Inference

```
# f_eqf(x): trained equilibrium v field
# h_omega(z): noise level readout estimator
# sigma_thresh: stopping threshold
# C_s: number of cleanup steps
# eta(sigma): sampling warp
sigma_hat = ones(num_frames)
x = randn(sample_shape)

# adaptive early stopping
while max(sigma_hat) > sigma_thresh:
    v_hat, activations = f_eqf(x)
    sigma_hat = h_omega(activations)
    x = x - eta(sigma_hat) * v_hat

# cleanup refinement
for j in range(C_s):
    v_hat, activations = f_eqf(x)
    sigma_hat = h_omega(activations)
    x = x - sigma_hat / (C_s - j) * v_hat

return x
```

C.3 Inpainting Results on RealEstate10K

We present an additional set of results on an inpainting task on the RealEstate10K dataset. The model is given frame 1 and the final 4 frames (corresponding to the first and last latent frames). It must denoise the intermediate frames, constituting 49 total frames. The given frames are treated as context in the same context-conditioned style, where they have zero noise and the remaining frames do have noise. The task is not autoregressive, and the model has all the information about the end of the denoising window, making it much easier for the denoising model. We find that on this easier task, the inpainting is essentially perfect, and the models have comparable metric scores (PSNR, FVD, VBench) over 256 generated videos per seed with 5 seeds. This is in contrast to the more difficult open-ended and autoregressive task presented in the main text. The results are presented in Table 10.

D Experiment and Dataset Details

In this section, we provide details on the experimental testbeds for the three video datasets Minecraft, RealEstate10K, and Droid. All experiment hyperparameters are provided in Table 12. We also provide additional experiment details for the ImageNet experiments. All model variants are evaluated on comparable training budgets; FLOP and runtime analysis are included at the end of this section, Subsection D.6. For more information on the geometry of autoregressive generation schedules (including stride length described below), see Appendix I.

Table 8: **Adaptive early stopping experiments on Minecraft with EQF.** For adaptive runs, the stopping noise level induces an empirical number of function evaluations N. Higher setting of μ for NAG makes the adaptive stopping condition trigger sooner. All runs use c-function as the η warp. These are the values plotted in Figure 16.

Inference Mode	**Sampler**	**Stopping Noise Level σ_{thresh}**	N	**FVD↓**
Adaptive Early Stopping	EQF GD	0.7	27.7	161.67
		0.5	38.9	94.87
		0.3	49.8	**86.70**
		0.2	55.4	89.88
		0.1	67.6	94.75
Adaptive Early Stopping	EQF NAG μ=0.1	0.7	26.0	161.59
		0.5	36.0	92.41
		0.3	45.7	**86.55**
		0.2	50.5	89.94
		0.1	57.4	98.12
Adaptive Early Stopping	EQF NAG μ=0.2	0.7	24.1	167.80
		0.5	33.1	93.92
		0.3	41.8	91.95
		0.2	46.1	**89.65**
		0.1	52.1	94.04
Adaptive Early Stopping	EQF NAG μ=0.3	0.7	22.3	171.65
		0.5	30.1	97.53
		0.3	37.9	**90.90**
		0.2	41.5	97.11
		0.1	46.7	102.44
Fixed	EQF GD	0	25	151.78
		0	50	107.05
		0	75	99.22
		0	100	98.99
Fixed	EQF NAG μ=0.1	0	25	170.55
		0	50	114.91
		0	75	98.20
		0	100	100.16
Fixed	EQF NAG μ=0.2	0	25	212.75
		0	50	116.03
		0	75	102.12
		0	100	97.79
Fixed	EQF NAG μ=0.3	0	25	318.11
		0	50	131.33
		0	75	105.81
		0	100	99.70
Fixed	Flow Matching	0	25	222.67
		0	50	165.41
		0	75	150.82
		0	100	146.77

Table 9: Full results table for FVD across different numbers of sampling steps, using the budget-adaptive sampling algorithm for EQF on Minecraft, Re10K, and Droid. EQF Adaptive NAG and EQF NAG are run with $\mu = 0.1$. FM is run with Euler sampling. Runs are across 3 seeds.

Minecraft: FVD ↓

Sampling steps	10	20	30	40	50
EQF Adaptive NAG	$\mathbf{271.22 \pm 2.19}$	$\mathbf{126.29 \pm 4.23}$	$\mathbf{103.16 \pm 6.33}$	$\mathbf{99.59 \pm 1.92}$	$\mathbf{91.45 \pm 0.92}$
EQF NAG	952.46 ± 11.91	246.89 ± 6.65	144.55 ± 4.64	109.50 ± 5.19	92.08 ± 0.30
FM	570.30 ± 3.72	240.81 ± 11.47	170.53 ± 6.55	145.77 ± 5.74	135.37 ± 1.74

Minecraft: VBench ↑

Sampling steps	10	20	30	40	50
EQF Adaptive NAG	$\mathbf{0.770 \pm 0.001}$	0.772 ± 0.002	0.772 ± 0.002	0.772 ± 0.001	0.771 ± 0.000
EQF NAG	0.750 ± 0.001	$\mathbf{0.773 \pm 0.001}$	$\mathbf{0.774 \pm 0.001}$	$\mathbf{0.774 \pm 0.001}$	$\mathbf{0.774 \pm 0.000}$
FM	0.755 ± 0.001	0.766 ± 0.000	0.767 ± 0.001	0.768 ± 0.001	0.769 ± 0.000

Re10K: FVD ↓

Sampling steps	10	20	30	40	50
EQF-FT Adaptive NAG	$\mathbf{165.15 \pm 0.53}$	$\mathbf{149.58 \pm 3.54}$	$\mathbf{134.20 \pm 2.20}$	$\mathbf{130.19 \pm 1.23}$	$\mathbf{127.69 \pm 1.57}$
EQF-FT NAG	244.41 ± 2.58	184.11 ± 1.19	172.56 ± 1.54	168.67 ± 2.62	166.73 ± 2.98
FM-FT	169.13 ± 4.47	161.09 ± 3.16	155.34 ± 3.96	148.92 ± 5.56	145.42 ± 5.93

Re10K: VBench ↑

Sampling steps	10	20	30	40	50
EQF-FT Adaptive NAG	$\mathbf{0.757 \pm 0.000}$	$\mathbf{0.759 \pm 0.000}$	$\mathbf{0.760 \pm 0.000}$	$\mathbf{0.761 \pm 0.001}$	$\mathbf{0.761 \pm 0.000}$
EQF-FT NAG	0.748 ± 0.000	0.757 ± 0.000	0.758 ± 0.001	0.758 ± 0.001	0.759 ± 0.000
FM-FT	$\mathbf{0.757 \pm 0.000}$	$\mathbf{0.759 \pm 0.000}$	$\mathbf{0.760 \pm 0.000}$	0.760 ± 0.000	$\mathbf{0.761 \pm 0.000}$

Droid: FVD ↓

Sampling steps	10	20	30	40	50
EQF-FT Adaptive NAG	$\mathbf{150.89 \pm 6.32}$	$\mathbf{119.40 \pm 3.20}$	$\mathbf{112.82 \pm 3.70}$	$\mathbf{111.36 \pm 4.74}$	$\mathbf{107.38 \pm 0.82}$
EQF-FT NAG	223.89 ± 4.50	145.92 ± 4.96	125.47 ± 4.26	116.96 ± 2.72	113.58 ± 3.51
FM-FT	207.59 ± 6.70	133.86 ± 4.66	117.99 ± 6.09	111.56 ± 5.25	108.69 ± 3.71

Droid: VBench ↑

Sampling steps	10	20	30	40	50
EQF-FT Adaptive NAG	$\mathbf{0.805 \pm 0.001}$	$\mathbf{0.810 \pm 0.001}$	$\mathbf{0.810 \pm 0.003}$	$\mathbf{0.811 \pm 0.003}$	$\mathbf{0.812 \pm 0.002}$
EQF-FT NAG	0.791 ± 0.002	0.805 ± 0.000	0.807 ± 0.001	0.810 ± 0.002	0.810 ± 0.002
FM-FT	0.789 ± 0.003	0.805 ± 0.002	0.808 ± 0.000	0.809 ± 0.001	0.809 ± 0.000

Table 10: Inpainting Results with EQF on Re10K

Method	Sampler	PSNR↑	FVD↓	VBench↑
EQF-FT	NAG, $\mu = 0.05$	$\mathbf{20.34 \pm 0.06}$	37.28 ± 1.14	$\mathbf{0.7738 \pm 0.0002}$
EQF-FT	GD	20.32 ± 0.06	37.82 ± 1.05	0.7737 ± 0.0003
FM-FT	Euler	20.30 ± 0.04	$\mathbf{35.29 \pm 0.65}$	0.7736 ± 0.0004

Table 11: Equilibrium Finetuning on Minecraft. Autoregressive Minecraft generation results across Equilibrium Finetuning settings compared to a Flow Matching model with continued training. All share an equivalent budget of 100k extra steps. All are run with 250 sampling steps.

Method	η Warp	Sampler	μ	FVD↓
EQF	c Function	NAG	0.2	$\mathbf{74.54}$
EqM	c Function	NAG	0.1	75.98
FM	c Function	Euler	–	100.48

Table 12: Training settings for Minecraft, Re10K, and Droid across models. The Rectified Flow schedule is from Liu et al. (2023) and the Cosine schedule is from Nichol & Dhariwal (2021). Minecraft is trained from scratch. For Re10K, we finetune Wan T2V 1.3B (Wan et al., 2025). For Droid, we finetune Wan TI2V 5B.

Config	Minecraft	Re10K	Droid
Training			
Effective Batch Size	8	8	16
Learning Rate	2e-4 (Linear Warmup)	1e-4 (Linear Warmup)	1e-5 (Linear Warmup)
Warmup Steps	2,000	2,000	2,000
Weight Decay	1e-3	1e-3	1e-3
Training Steps	580k	100k	60k
GPU Usage	1×H200	2×H200	4×H200
Optimizer	Adam, betas=(0.9, 0.99)	Adam, betas=(0.9, 0.99)	Adam, betas=(0.9, 0.99)
Training Strategy	Distributed Data Parallel	Distributed Data Parallel	Distributed Data Parallel
Precision	Bfloat16	Bfloat16	Bfloat16
EMA for Eval	0.9999	0.9999	0.9996
Noise/Context Strategy	Random Independent	Random Independent	Random Length Context
Equilibrium Forcing (EQF)			
Objective Target	$\mathbf{v}$	$\mathbf{v}$	$\mathbf{v}$
Noise Schedule	Rectified Flow	Rectified Flow	Rectified Flow
Flow Matching (Liu et al., 2023)			
Objective Target	$\mathbf{v}$	$\mathbf{v}$	$\mathbf{v}$
Noise Schedule	Rectified Flow	Rectified Flow	Rectified Flow
Equilibrium Matching (Wang & Du, 2025)			
Objective Target	$c(\boldsymbol{\sigma}) \odot \mathbf{v}$	–	–
Noise Schedule	Rectified Flow	–	–
Equilibrium Multiplier λ	4	–	–
Diffusion (Sohl-Dickstein et al., 2015)			
Objective Target	$\mathbf{v}$	–	–
Noise Schedule	Cosine	–	–
Loss Weighting	Sigmoid	–	–
VAE Details			
Input Dimension	$50 \times 256 \times 256 \times 3$	$49 \times 256 \times 256 \times 3$	$49 \times 384 \times 640 \times 3$
Latent Dimension	$50 \times 32 \times 32 \times 4$	$13 \times 32 \times 32 \times 16$	$13 \times 24 \times 40 \times 48$
Temporal Downsampling	1	4	4
Spatial Downsampling	8	8	16
VAE Type	Framewise	Causal	Causal
VAE Parameters	83 M	126 M	705 M
Model			
Total Parameters	95.3 M	1.3 B	5 B
# Attention Heads	12	12	24
Head Dimension	64	128	128
# Layers	10	30	30
Time Embed Dimension	256	256	256
Condition Embed Type	Action	Camera Pose	UMT5 Language Prompt
Condition Embed Dimension	768	–	4096 (UMT5) / 3072 (Model)

D.1 Evaluation Metrics

Frechet Video Distance (FVD). FVD is the primary metric that we report to measure the quality of autoregressively generated video, with low scores being better (Unterthiner et al., 2018). FVD first extracts features from generated and true samples through an image feature extractor I3D (Carreira & Zisserman, 2017), and then calculates the Frechet distance in the latent space between the sets of samples (Unterthiner et al., 2018). Autoregressive video generation does not have a ground truth to compare to once the video is long enough, so FVD is particularly helpful for assessing long horizon consistency and stability over generation.

VBench. We report VBench to evaluate the semantic quality of generated video on axes such as frame quality, temporal consistency, and nontrivial motion (Huang et al., 2024). VBench provides a granular, human-aligned metric to assess the tradeoff of consistency and video dynamics, allowing us to quantify whether EQF improves visual quality and long range consistency while still capturing the motion of the video. We utilize the same mix of VBench sub-metrics as Song et al. (2025). The overall metric we report is computed as the weighted average of the normalized VBench submetrics: aesthetic quality, imaging quality, subject consistency, background consistency, temporal flickering, motion smoothness, and dynamic degree, where dynamic degree has weight 0.5 and all other submetrics have weight 1.

Frechet Inception Distance (FID). FID (Heusel et al., 2017) is a distributional metric similar to FVD, widely adopted to measure generation quality for images. We use this metric for the additional ImageNet experiments.

D.2 Minecraft

Dataset Details. The Minecraft Dataset (Yan et al., 2023) is a collection of 200k videos collected from the Minecraft video game, each with a length of 300 frames. We employ a similar setup to Song et al. (2025), where each frame is upsampled to 256×256 pixels. Every video is coupled with an action sequence, making this an action-conditioned generation task. Each frame's action is one of three choices: forward, turn left, or turn right. All videos were collected in the same biome, so the data is relatively homogeneous with respect to the scene, compared to all possible variants of video from the Minecraft game. The egocentric controller rotates along with the turning actions, causing parts of the scene to come in and out of view, necessitating a strong generative model to adhere to the changes and changing context.

Following standard practice, all denoising models are trained in the VAE latent space (Rombach et al., 2021). We use the pretrained VAE on the Minecraft dataset from Song et al. (2025), which is a framewise VAE without temporal downsampling.

Standard Inference Setting. The standard inference setting is a 300 frame rollout. We use 25 ground truth frames to provide context for the generation of 275 future frames. The context window and active sampling horizon are both 25 frames long, forming a 50-frame sliding window. We report FVD and VBench scores over 256 generated videos per seed with 5 seeds for runs in our primary experiments (Section 5). We use 250 sampling steps and an emission stride of 1 frame at a time. The inference-time variant experiments in Section 5.4 use 1 seed.

Budget-Adaptive Inference Setting. The adaptive inference setting uses a 300 frame rollout with a sliding window length of 50, with 25 context frames and 25 active sampling horizon frames. The emission stride is 5, and the number of sampling steps ranges from 10 to 50 with increments of 10. The results are averaged across 3 seeds.

Adaptive Early Stopping Inference Setting. The adaptive early stopping inference setting length is also a 300-frame rollout. The sliding window is still 50 frames, but there are 40 context frames and 10 in the active horizon. The maximum possible number of sampling steps is 100, and the emission stride is 10 frames. We report FVD scores over 256 generated videos per seed with 1 seed per run. Due to the different inference setting, the metric scores are not easily comparable with the other inference settings.

Online Replanning Inference Setting. The online replanning inference setting uses a 300 frame rollout with a sliding window length of 50, with 25 context frames and 25 active sampling horizon frames. The emission stride is 5 frames at a time and we vary the threshold for replanning τ, the range of which was calibrated through hyperparameter search. We report the FVD between the evaluation tensor and true videos. We use a maximum of 50 inference steps per inner denoising loop, though the

algorithm may take fewer steps if it decides to not renoise or to renoise existing predictions to an intermediary noise level.

D.3 RealEstate10K

Dataset Details. The Re10K dataset consists of curated real estate property tour video clips introduced by Zhou et al. (2018). The camera moves through 3D space, exposed to the model through camera pose annotations. We employ the dataset as a testbed for finetuning a large open source Flow Matching model Wan 2.1 T2V 1.3B (Wan et al., 2025) into an EQF model, dropping the noise conditioning as we finetune. Details on equilibrium finetuning are available in Appendix E. This dataset serves as an ideal finetuning benchmark for Wan because the benchmark demands more realism and heterogeneity than Minecraft over an autoregressive rollout. The dataset does not contain captions, so we use a null caption for every clip as input to the T2V model. The dataset contains 58k videos after filtering out videos shorter than 49 frames; the dataset is downsampled to 256×256 resolution. We train on clips by choosing a starting frame in the video, resulting in 5.9 million clips for training.

We directly use the VAE from the Wan 2.1 model, which has temporal padding to handle the first frame, meaning the conversion turns N video frames into $\lfloor (N-1)/4 \rfloor + 1$ latent frames.

Standard Inference Setting. The standard inference setting is a 189-frame rollout, using 37 frames as the context to generate 152 frames. Due to the temporal downsampling of the VAE, we clarify the number of latent frames here in addition to video frames. The sliding window is 49 video frames (13 latent frames), and there are 37 video frames (10 latent frames) in the context and 12 video frames (3 latent frames) in the active window at any given time. We report FVD and VBench scores over 256 generated videos per seed with 5 seeds for runs in our primary experiments. We follow the default setting of Wan 2.1 and use 50 sampling steps per generation round (Wan et al., 2025). The emission stride is 12 video frames (3 latent frames) at a time.

Budget-Adaptive Inference Setting with Less Context. The adaptive inference setting is a 189-frame rollout, using 29 context frames (8 latent frames) and 20 frames (5 latent frames) in the active horizon. The sliding window is 49 video frames (13 latent frames) and the emission stride is 4 video frames (1 latent frame) at a time. The number of sampling steps varies from 10 to 50 with increments of 10.

D.4 Droid Robotics Dataset

Dataset Details. The Droid Robotics dataset is a large scale video dataset of robots executing language-specified manipulation of objects in diverse settings (Khazatsky et al., 2024). See the original paper for more details on how the video dataset was collected. We use the text prompts from the Large Video Planner paper, which are preprocessed with Gemini Flash to describe the motion of the robot in the video with more detail than the original prompts (Chen et al., 2025). This dataset provides us with a realistic benchmark where spatial consistency and object interactions must be predicted with accurate robot dynamics, in contrast with the other two datasets where the environment is static and the camera is dynamic. The resolution is resampled to 384×640, and we resample video to be 49 frames long so that the text prompt corresponds to a completed action. We use the set of videos that were already filtered by Chen et al. (2025), resulting in 140k training clips of length 49 each.

We finetune the large scale open source Flow Matching model Wan 2.2 TI2V 5B into an EQF model on the Droid dataset, following the same finetuning strategy as for the RealEstate10K dataset (details on equilibrium finetuning are available in Appendix E). We directly utilize the UMT5 encoder for text prompt encoding and the VAE from Wan 2.2. The TI2V model has a native image conditioning pathway, but we condition on context using the unified self-attention window. To train without the image conditioning pathway, we turn off the image cross attention layers during training and inference. We also note that the 5B model is not a mixture-of-experts for different noise levels, unlike the Wan 2.2 14B model. Following Chen et al. (2025), instead of random independent noise levels per frame, we train with random length clean context, to match the inference setting better. A random context length from 0 to 6 temporal latent tokens is sampled. Two noise levels are sampled, one for the context length and one for the rest of the tokens. With probability 0.5, the context is set to noise level 0 instead of the sampled noise level; if it is set to 0, then there is no loss applied on the context tokens.

Table 13: Backbone and noise level predictor computational cost per forward pass on Minecraft.

Model	GFLOPs/forward
FM backbone	6895.50
EQF backbone	6895.50
Noise level predictor g_ϕ	61.39
Noise level readout h_ω	0.30

Table 14: Wall clock time per sampling step on Minecraft, 1xH200 with 50 frames.

Method	Runtime (ms/step)
FM Euler	34.41
EQF GD	34.46
EQF NAG	34.46
EQF NAG + h_ω	34.86
FM Euler + g_ϕ	35.10

Adaptive Inference Setting. The inference setting for Droid uses a single generation round with 49 frames in the window. The context is 13 video frames (4 latent frames), and the active window is 36 video frames (9 latent frames). The number of sampling steps varies from 10 to 50 with increments of 10. The emission stride is 36 video frames, since there is just one round. For visualization we use 21 video frames as context (6 latent frames) and an active window of 28 video frames (7 latent frames).

D.5 ImageNet

We train on the ImageNet-1K dataset for 80 epochs with the XL/2 size with center crop and random horizontal flip (Deng et al., 2009). The model is trained with the SD VAE (Rombach et al., 2021), with v-prediction, 10% class label dropout, rectified flow path, total batch size 256, AdamW with learning rate 1e-4 and betas (0.9, 0.999), weight decay of 0, EMA weight of 0.9999, and with tf32 precision. The model is trained to directly predict the unmodulated velocity. The EqM and FM models are trained with identical settings.

We use an identical inference and evaluation setup to the Equilibrium Matching paper (Wang & Du, 2025). Different from EqM, we apply EQF's inference-time η warp. We report FID on 50k generated images with 250 sampling steps and no classifier free guidance.

D.6 Denoising Model and Noise Level Predictor Compute Comparison

In this section, we provide wall clock and GFLOP compute comparisons across models on the Minecraft dataset. Table 13 shows that FM and EQF backbones have identical GFLOP requirements, since EQF is implemented as the same model as FM with the noise level condition always set to zero. The readout h_ω requires only 0.30 GFLOPs, less than $1/10{,}000$ of a denoiser forward pass, while g_ϕ requires approximately 0.89% of the backbone compute.

We additionally measure wall clock sampling time on Minecraft in Table 14. The parameter count ratio between h_ω and the denoiser is largest on Minecraft compared with Re10K and Droid, and thus provides the most conservative setting. Inference procedures like NAG do not require additional model forward passes, so EQF GD and EQF NAG are identical. The noise level prediction models add only negligible additional runtime requirements.

E Details on Equilibrium Finetuning

In this section, we discuss details for finetuning pretrained Flow Matching models into models such as EQF that do not condition on the noise level. To the best of our knowledge, equilibrium models have only been trained from scratch in prior work. This makes equilibrium finetuning a novel approach, also applicable to other frameworks like EqM, and thus it is relevant to further expose implementation details. We report equilibrium finetuning across multiple model scales and on each of the datasets. Details of training noise-level predictors on top of finetuned models are available in Appendix F.

E.1 Minecraft Equilibrium Finetuning

We use the Minecraft dataset as a proof of concept for testing the ability to finetune a noise-conditioned model into an equilibrium model. As the initialization for this task, we use the Flow Matching checkpoint trained for 580k steps that served as the baseline for Minecraft experiments. Our results suggest that the EQF objective alone is sufficient for finetuning, not requiring engineering tricks, curricula, or EqM-style decomposing the target based on the noise level to incentivize the equilibrium landscape.

Equilibrium Finetuning Proof of Concept. We perform equilibrium finetuning by continuing training on a Minecraft Flow Matching checkpoint for 100k more steps. We induce an equilibrium landscape by simply zeroing out the noise conditioning signal $\boldsymbol{\sigma} = \mathbf{0}$ and using the EQF objective from Equation 4. We further experiment with equilibrium finetuning by using the EqM objective

for 100k steps, decomposing the target as $c(\boldsymbol{\sigma})\mathbf{v}$ using the c-function used by the EqM paper and explained in Appendix G. Note the simplicity of the EQF objective in the Minecraft task: the finetuned model only needs to learn the same denoising function without noise level conditioning, while the EqM objective induces a new loss weighting and requires the model to modulate its output magnitude based on the noise level. For fair comparison, we continue training the Flow Matching model for an additional 100k steps as well. After continued training ends, for EQF we repeat the process of training a readout predictor h_ω on top of the finetuned models' activations in order to enable closed loop sampling.

Results on Minecraft. We report inference results on the identical experimental setting from Section 5 on Minecraft in Table 11. We find that both NAG (reported with the best setting of μ) and closed loop sampling improve results, consistent with our results training from scratch.

Other Finetuning Strategies. While we found that simply dropping out the noise level as a finetuning method to be the most effective, we experimented with other strategies as well. We document these other strategies for the sake of the community.

1. The simple dropout method explained above has noise level dropout probability $p = 1$. We experiment with dropout probabilities $p < 1$, where p decides whether we condition on the noise level as in Flow Matching, or pass in zero. We found that in this case, the model took longer to converge and was further unable to fully "forget" the noise conditioning signal. Inference carried out by conditioning the model on $\boldsymbol{\sigma} = \mathbf{0}$, (as is standard in EQF and EqM implementations), produced divergent samples because the model failed to learn an equilibrium field. When running inference by passing in scheduled noise levels (as in standard Flow Matching inference), sample quality was comparatively better, which is evidence of the failure to forget the conditioning signal. Since standard Flow Matching models rely deeply on the noise conditioning, these results suggest that the loss landscape basin of Flow Matching models is shaped by their noise conditioning, relying on substantial changes at finetuning time to escape the local minimum of the new EQF objective. We further experimented with curricula for the dropout probability, noticing similar results where the model is unable to forget the nonequilibrium landscape of its original training objective.
2. We also experimented with annealing an auxiliary distribution over a noise conditioning signal $\boldsymbol{\sigma}'$ such that at the first training iteration $\boldsymbol{\sigma}' = \boldsymbol{\sigma}$ with probability 1 and at the end of training they are fully independent. At the end of training, since the condition and the true noise level are independent, the model may learn to ignore the noise condition. We implement this by resampling the condition signal $\boldsymbol{\sigma}'$ from a uniform distribution centered on the true noise level $\boldsymbol{\sigma}$ such that the uniform anneals into the standard on the entire interval. We experimented with similar annealing schedules for the Beta distribution to encourage resampling concentration on $\boldsymbol{\sigma} = 0$. While more effective than partial dropout, the results were inferior to simple full dropout training, not justifying the added complexity of the curriculum.

E.2 Re10K Equilibrium Finetuning

In this section, we discuss details of finetuning Wan 2.1 T2V 1.3B into a noise unconditional model using the EQF objective. The dataset is camera-conditioned and we adopt pose processing from Song et al. (2025), introducing the conditioning signal into the AdaLN stream (Peebles & Xie, 2023).

With insights from Minecraft finetuning in Appendix E.1 we proceed to employ simple $p = 1$ dropout of the noise conditioning signal to finetune Wan into EQF on the Re10K dataset; we refer to this as the EQF-FT model. We also fine tune Flow Matching as a noise-conditional baseline with the same computational budget, denoted FM-FT.

We elucidate training dynamics by plotting the training and validation loss (validating an EMA model with an EMA weight of 0.9999) against training iteration in Figure 17. The loss decreases rapidly for the baseline Flow Matching model (FM-FT), while generative quality with standard inference-time noise conditioning lags behind (partially due to the EMA). On the other hand, the EQF-FT model must forget the noise conditioning signal at the beginning of training, leading to the model incurring a higher training and validation loss at the start of training. The generation quality of EQF-FT after only 10k steps is substantially worse because we run inference on the model with zero noise conditioning, and the model has not yet learned the equilibrium landscape. However, by 20k steps, the model has

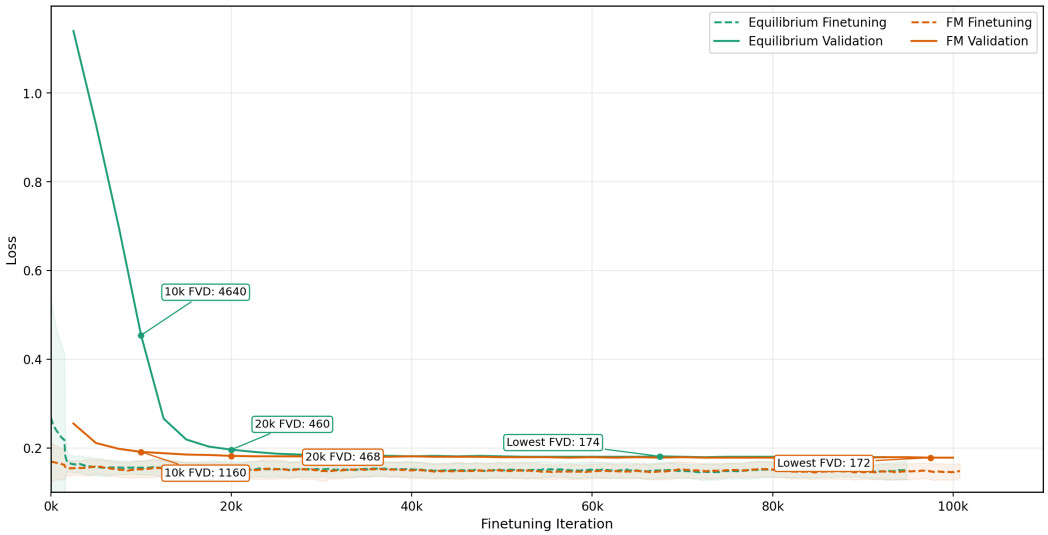

Figure 17: Re10K Validation Loss and FVD across equilibrium finetuning steps.

learned to effectively drop its reliance on noise conditioning and catches up to the Flow Matching baseline, after which they have similar training dynamics. Training is carried out for 100k steps, over which validation FVD continues to improve. Note that the inference results in Figure 17, run for the purpose of validation during training, are strictly open loop and not adaptive.

E.3 Droid Equilibrium Finetuning

To finetune Wan 2.2 TI2V 5B into a noise unconditional model using the EQF objective, we follow the same procedure as with Minecraft and Re10K. Namely, we employ simple $p = 1$ dropout of the noise conditioning signal to create the EQF-FT model, and finetune the Flow Matching model with the same settings to yield the FM-FT model. Models are trained for 60k steps; it has been observed that denoising quality continues to improve with continued training for denoising models even after the model loss converges, and it is likely that further continuing training would yield qualitative and quantitative improvements in this setting as well.

F Details on Estimating Noise Levels from Data or EQF Activations

In this section, we provide neural network architecture and training details for models that predict noise levels from the data or from EQF activations on the Minecraft dataset. We refer to auxiliary networks g_ϕ as *noise level predictors* when they estimate the noise level directly from a noisy frame, and we refer to *readout predictors* h_ω when they bootstrap off an EQF model's activations when evaluated on a noisy frame. Noise level predictors g_ϕ are used in our noise level divergence analyses as well as in an effort to close the loop on noise level divergence for standard Flow Matching inference. On the other hand, readout predictors h_ω are primarily used by EQF for closed-loop inference via the warp η.

F.1 Architecture Design

Prediction of Noise Level Directly from Noisy Data. The noise level predictor g_ϕ on the Minecraft dataset is a framewise CNN with 600k parameters trained to predict the noise level of a frame that has been noised under the Flow Matching noising process. We employ framewise training because autoregressive inference algorithms typically use a rolling denoising window in which contiguous chunks of frames appear at different noise levels. To train, we draw a single frame $x_t \in \mathbb{R}^{C\times H\times W}$ from the data, sample $\epsilon \sim \mathcal{N}(0, I)$ and $\sigma \sim \mathcal{U}[0, 1]$, and construct the noised frame

$$x_t^\sigma = (1 - \sigma)x_t + \sigma\epsilon. \tag{32}$$

The network predicts the logit of the noise level, with objective

$$\mathcal{L}_g = \mathbb{E}_{x_t,\epsilon,\sigma}\left[\|g_\phi(x_t^\sigma) - \text{logit}(\sigma)\|_2^2\right]. \tag{33}$$

We apply the loss in logit space because it provides a more balanced supervision signal near $\sigma \approx 0$ and $\sigma \approx 1$, compared with directly applying squared error on $[0, 1]$. We convert the logit prediction

to a valid noise level by applying a sigmoid,

$$\hat{\sigma}_\phi(x_t^\sigma) = \text{sigmoid}(g_\phi(x_t^\sigma)) \,. \tag{34}$$

In Section 3, we slightly abuse notation by writing the estimated true noise level as $\tilde{\sigma} \approx g_\phi(x_t^\sigma)$, where the sigmoid transformation is implicit. We evaluate the model compared to the readout predictor h_ω below.

The model is able to train accurately on the noising process because the noise level is identifiable from the noisy data, as Gaussian shells theoretically concentrate in high dimensions (Kadkhodaie et al., 2026). In other words, this concentration removes the ambiguity in the data distribution $p(\sigma|x^\sigma)$ that the model was trained on; though that distribution for different values of σ technically has support everywhere, the model is trained to predict the mode which matches the distribution in practice. Further, though intermediate samples at a particular noise level during inference do not necessarily match the marginal distribution from the forward noising process on which g_ϕ is trained, they overlap sufficiently when sampling with a well-trained flow matching backbone such that the noise level predictor remains accurate during the inference process. Evidence of this is established by the result that closed loop sampling using the g_ϕ-estimated noise level leads to performance benefits for Flow Matching inference in Section 3 and Appendix B.3. Finally, since g_ϕ's training prediction error is low (as explained in the next subsection), we may assume that the error reported in Section 3 represents a consequential difference in the noise level condition, rather than the error of g_ϕ. Training dynamics of the noise level prediction model are elaborated upon in Appendix F.2.

Prediction of the Noise Level from EQF Activations. The readout predictor h_ω is a framewise CNN with 260k parameters. Like the noise level predictor, it is trained in a framewise manner to predict the noise levels under a Flow Matching noising process. However, the readout predictor relies on a pretrained EQF model's ability to implicitly estimate $p(\boldsymbol{\sigma}|\mathbf{x}^{\boldsymbol{\sigma}})$ to minimize the training objective (Section 4.3). Given a pretrained f_{EQF} denoiser, we train the readout predictor by extracting f_{EQF}'s activations z_t^σ on noisy data constructed as in Equation 32 and bootstrapping from model activations as follows:

$$\mathcal{L}_h = \mathbb{E}_{x_t,\epsilon,\sigma}\left[\|h_\omega(z_t^\sigma) - \text{logit}(\sigma)\|_2^2\right], \qquad \text{where } (\hat{v}_t^\sigma, z_t^\sigma) = f_{\text{EQF}}(x_t^\sigma). \tag{35}$$

The readout prediction is also converted to a valid noise level by applying the sigmoid.

F.2 Noise Level Predictor and Readout Training Dynamics on Minecraft

We compare the training dynamics of noise level predictors with readout predictors. The results support the hypothesis that EQF models must internally estimate the noise level of a sample to minimize the objective, as we find it far easier to train an estimator of the noise level of data by bootstrapping a readout predictor h_ω from EQF activations compared to training a noise level predictor g_ϕ from the raw data directly.

On the Minecraft dataset, readout predictor h_ω converges in only 10k steps, whereas the noise level predictor g_ϕ is trained for 100k steps due to significantly slower convergence and doesn't reach as low of a loss. The training loss is displayed in Figure 18. We further found that g_ϕ requires larger model capacity to adequately solve the prediction task, using 600k parameters, while h_ω is less than half the size. These results, especially the fast convergence of the noise level estimator, support our hypothesis that the activations of the EQF model must already contain a latent estimate of the noise level in order to minimize the objective (Section 4.2).

The order of magnitude difference in loss is primarily attributable to accuracy near the low noise boundary $\sigma \approx 0$. Since the predictors are trained in logit space, small absolute errors in σ near the boundary are amplified by the logit transform. The corresponding error in noise-level space still means they are quite accurate. This effect is visible in the marginalized residuals in Figure 19: both predictors are accurate enough to serve their intended roles in our analysis, with g_ϕ providing a reliable external estimate of the noise level directly from noisy data and h_ω providing a reliable estimate from EQF activations. The remaining difference is concentrated near $\sigma \approx 0$, where the readout predictor has noticeably smaller residuals and where logit-space errors are most heavily magnified. This improved accuracy at the boundary is especially relevant for our use of h_ω in closed-loop sampling and adaptive stopping, since those mechanisms depend on accurately detecting proximity to the clean-data manifold.

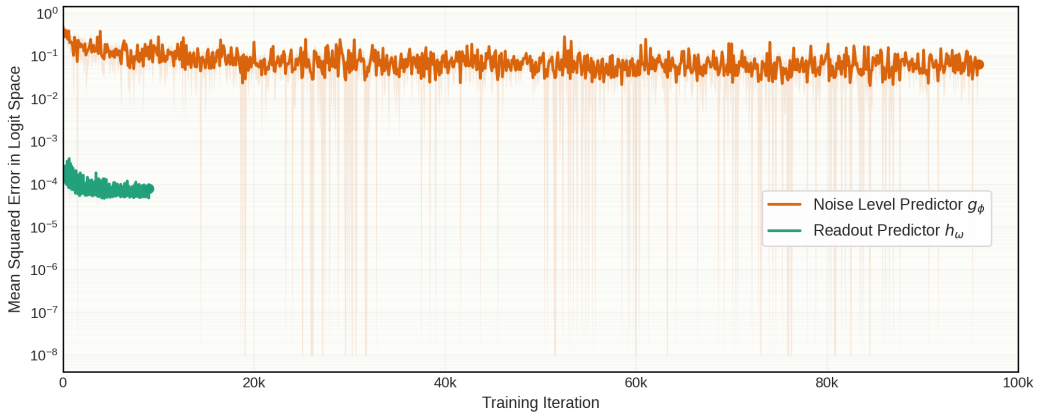

Figure 18: Training loss for noise level estimation. The readout predictor h_ω, trained on EQF activations, converges substantially faster than the framewise predictor g_ϕ trained directly from noisy data.

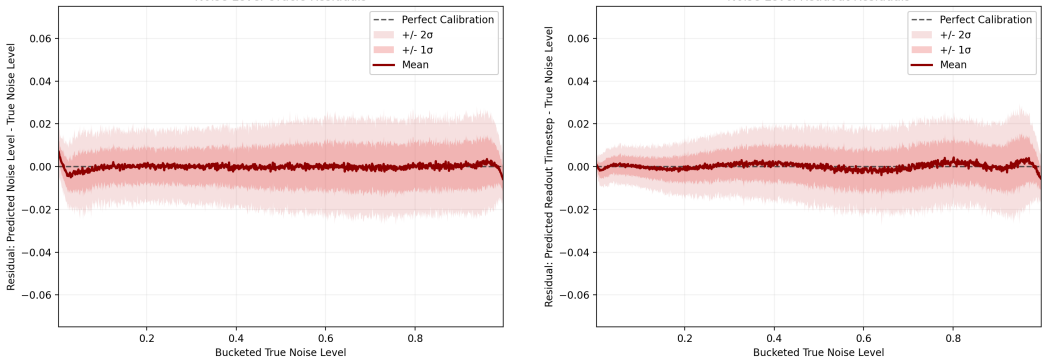

Figure 19: Marginalized residuals of noise level estimates from the framewise noise level predictor g_ϕ trained directly on noisy data (left) and the readout predictor h_ω trained on EQF activations (right). Both predictors are accurate, but h_ω is especially accurate near the low-noise boundary.

F.3 Training Details of Noise Level Predictor on Re10K and Droid Datasets

Here, we elaborate on training details for the noise level predictor h_ω for equilibrium finetuned models. On the 1.3B Re10K model, we train a 460k parameter readout predictor for an additional 5k steps. On the 5B Droid model, we train a 860k parameter readout predictor for an additional 10k steps. Both architectures are the same: the denoising backbone is frozen, we mean pool over the spatial dimensions of the model activations, apply a shared LayerNorm, apply a linear projection down to 256 dimensions, combine the layers using learned softmax mixing weights, and then apply a framewise MLP. The target is the logit of the noise level for each frame, trained with an MSE loss. Similar to the training dynamics on the Minecraft dataset, we find that the model converges quickly on the noise level readout task.

G Details on Sampler Warp Schedules

In this section, we describe the inference-time warp functions used to shape the sampling dynamics of EQF. The warp in EQF is used to modulate the magnitude of the learned equilibrium velocity field during sampling without changing the training objective, providing a unified way to compare transport-style ODE reparameterization functions common in the literature with functions that induce attractor-style dynamics that vanish near the data manifold (Esser et al., 2024). We first define the relationship between ρ, a function that reparameterizes solver time s to noise levels σ, and a warp η for mapping from noise levels to step sizes. The different function definitions are necessary to unify understanding between what is proposed here with other ODE reparameterizations in the literature, as well as to understand normalization factors necessary for making the inference procedure principled. We then define the tradeoffs between warps that are transports and those that induce attractor landscapes. Finally, we give the exact warp definitions used in our experiments.

G.1 Sampling Schedules and Warp Functions

We first define $s \in [0, 1]$ as solver time, where $s = 0$ corresponds to the beginning of sampling at pure noise, and $s = 1$ to the end of sampling at clean data. Note this is the opposite direction of how the noise level σ is defined: $\sigma = 1$ is full noise, and $\sigma = 0$ is clean data. A time-reparameterized noise schedule is a monotonically decreasing function

$$\rho : [0, 1] \to [0, 1], \qquad \rho(0) = 1, \qquad \rho(1) = 0, \tag{36}$$

where $\rho(s)$ denotes the scheduled noise level σ at solver time s. Evenly spaced solver steps in s can therefore correspond to nonuniform steps in noise level $\rho(s)$, as is common in ODE samplers for Flow Matching (Esser et al., 2024).

The corresponding warp function η specifies the local update multiplier applied to the denoising field at each noise level. Along a trajectory $\rho(s)$, we define η as the positive denoising speed through noise-level space:

$$\eta : [0, 1] \to \mathbb{R}, \qquad \eta(\sigma) = \eta(\rho(s)) := -\frac{d\rho(s)}{ds}. \tag{37}$$

For ρ to define a valid transport, it must bring the probability mass from the initial noise level to the final noise level, satisfying

$$\int_0^1 \eta(\rho(s))\, ds = \int_0^1 \left(-\frac{d\rho(s)}{ds}\right) ds \tag{38}$$

$$= \rho(0) - \rho(1) = 1. \tag{39}$$

If we instead define η directly as a function of σ, we can recover the induced trajectory over solver time ρ by solving the ODE

$$\frac{d\rho(s)}{ds} = -\eta(\rho(s)), \qquad \rho(0) = 1. \tag{40}$$

Separating variables in Equation 40 gives $ds = -d\rho/\eta(\rho)$. Therefore, the solver time required to travel from noise level $\rho = 1$ to a terminal noise level σ_{end} is

$$T_\eta(\sigma_{\text{end}}) = \int_{\sigma_{\text{end}}}^1 \frac{du}{\eta(u)}. \tag{41}$$

When specifying a warp directly, we separate its shape from its overall scale by writing

$$\eta(\sigma) = A\tilde{\eta}(\sigma), \tag{42}$$

where $\tilde{\eta}$ is an unnormalized warp shape and $A > 0$ is a scalar normalization factor. This normalization factor is important to make sure that when we define the shape of a warp $\tilde{\eta}$, that it also defines a valid sampling process. Imposing the unit-time traversal condition $T_\eta(\sigma_{\text{end}}) = 1$ gives

$$1 = \int_{\sigma_{\text{end}}}^1 \frac{du}{A\tilde{\eta}(u)} = \frac{1}{A}\int_{\sigma_{\text{end}}}^1 \frac{du}{\tilde{\eta}(u)}, \qquad \Longrightarrow \qquad A = \int_{\sigma_{\text{end}}}^1 \frac{du}{\tilde{\eta}(u)}. \tag{43}$$

Thus, $\tilde{\eta}$ determines the relative allocation of computation across noise levels, while A fixes the total traversal time, together defining η. In the main text, namely Equation 7, we slightly abuse notation and also include a factor inversely proportional to the number of sampling steps N in the η function for simplicity of exposition. In this section, it is expressed as the normalized warp without the scaling by the number of steps.

G.2 Transport and Attractor Warps

The behavior of η near the data boundary determines whether the induced dynamics are transport-style or attractor-style. Transport-style warps remain nonzero as $\sigma \to 0$, so the sampler reaches the data manifold by following a prescribed trajectory whose endpoint is $\sigma = 0$. In this case, the dynamics need not slow down near clean data; successful sampling depends on the solver intersecting the data manifold at the scheduled endpoint.

Attractor-style warps instead satisfy $\eta(\sigma) \to 0$ as $\sigma \to 0$. When using an attractor-style warp with EQF sampling, the warped field $\eta(\sigma) f_{\text{EQF}}(x)$ vanishes near clean data, turning sampling into a fixed-point search. This attractor interpretation further motivates the use of gradient-based optimization tools such as Nesterov Accelerated Gradient, which are designed for convergent dynamics. We empirically demonstrate NAG works better with attractor warps in Table 2. Since attractor warps

vanish at the boundary, reaching exactly $\sigma = 0$ requires infinite continuous time; in practice, we normalize the trajectory down to a finite terminal noise level $\sigma_{\text{end}} > 0$.

Figure 20 illustrates the three equivalent views of a warp schedule. Subfigure 20(a) shows the warp $\eta(\sigma)$ as a noise-dependent update multiplier. Subfigure 20(b) shows the induced noise trajectory $\rho(s)$ over solver time. Subfigure 20(c) shows the composed update multiplier $\eta(\rho(s))$ actually applied over solver time. In open-loop sampling this composition determines the predetermined step-size sequence, whereas in closed-loop EQF sampling we evaluate $\eta(\hat{\sigma})$ using the estimated current noise level.

G.3 Detailed Instantiations of Warp Functions

In this section we explain the instantiations of the warp functions used in our experiments. The identity warp and the EqM c-function are specified directly as noise-dependent warp functions $\eta(\sigma)$. In contrast, SD3 and linear log-SNR are naturally specified as solver-time trajectories $\rho(s)$ and converted into warp functions using $\eta(\rho(s)) = -d\rho(s)/ds$.

Transport Warp: Identity. The simplest transport warp is the identity warp,

$$\eta_{\text{id}}(\sigma) = 1. \tag{44}$$

This assigns constant denoising speed at every noise level. Using the induced trajectory conversion in Equation 40, we obtain

$$\frac{d\rho(s)}{ds} = -1, \qquad \rho_{\text{id}}(s) = 1 - s. \tag{45}$$

Since $\eta_{\text{id}}(0) = 1$, the update multiplier does not vanish near the data boundary, making this a transport-style warp.

Transport Warp: SD3. We also consider the time-reparameterization used in SD3 (Esser et al., 2024). SD3's schedule denotes a shift parameter $r > 0$. The SD3 trajectory (where $s = 0$ is noise and $s = 1$ is data) is

$$\rho_{\text{SD3}}(s) = \frac{r(1-s)}{r - (r-1)s}. \tag{46}$$

This already satisfies the endpoint constraints

$$\rho_{\text{SD3}}(0) = 1, \qquad \rho_{\text{SD3}}(1) = 0, \tag{47}$$

so it defines a valid unit-time transport schedule without additional normalization. To express the induced warp, we compute the positive denoising speed through noise-level space:

$$\eta_{\text{SD3}}(\rho_{\text{SD3}}(s)) := -\frac{d\rho_{\text{SD3}}(s)}{ds} = \frac{r}{(r - (r-1)s)^2}. \tag{48}$$

This expression is currently written as a function of solver time s. To rewrite it as a function of the current noise level σ, we invert the schedule:

$$\sigma = \frac{r(1-s)}{r - (r-1)s} \qquad \Longrightarrow \qquad s = 1 - \frac{\sigma}{r - (r-1)\sigma}. \tag{49}$$

Substituting this inverse into the denoising speed gives

$$\eta_{\text{SD3}}(\sigma) = \frac{(r - (r-1)\sigma)^2}{r}. \tag{50}$$

For $r > 1$, the boundary values are

$$\eta_{\text{SD3}}(1) = \frac{1}{r}, \qquad \eta_{\text{SD3}}(0) = r. \tag{51}$$

Thus, SD3 slows down near pure noise and increases the update multiplier near the data manifold. Since $\eta_{\text{SD3}}(0) \neq 0$, it is a transport-style warp. Consistent with this transport boundary behavior, our experiments find that NAG is less stable with SD3 than with attractor-style warps.

Attractor Warp: EqM c-Function. We next consider the c-function shape introduced by Equilibrium Matching (Wang & Du, 2025). EqM uses this function as a training-time target modulation, whereas in the EQF framework this is an option for the shape of the inference-time warp. EqM

defines

$$c_{\lambda,\alpha}(\sigma) = \lambda \min\left\{1, \frac{\sigma}{1-\alpha}\right\} = \begin{cases} \lambda \dfrac{\sigma}{1-\alpha}, & 0 \le \sigma \le 1-\alpha, \\ \lambda, & 1-\alpha < \sigma \le 1. \end{cases} \tag{52}$$

In EQF, we use the unscaled shape $c_{1,\alpha}$ as an unnormalized warp shape, with $\alpha = 0.8$:

$$\tilde{\eta}_{c,\alpha}(\sigma) = c_{1,\alpha}(\sigma). \tag{53}$$

Applying the unit-time normalization derived above to ensure that the induced trajectory finishes in 1 unit of solver time gives

$$\eta_{c,\alpha}(\sigma) = A_\alpha c_{1,\alpha}(\sigma), \qquad A_\alpha = \int_{\sigma_{\text{end}}}^{1} \frac{du}{c_{1,\alpha}(u)}. \tag{54}$$

Since $c_{1,\alpha}(\sigma) \to 0$ as $\sigma \to 0$, this warp induces attractor-style dynamics and is normalized over a finite terminal noise level $\sigma_{\text{end}} > 0$. In EqM, this normalization was not considered, and instead the magnitude scale λ was swept over as an empirical training-time hyperparameter. The empirically optimal λ and inference-time learning rate scale (called η in their paper) combination that EqM ends up with (equivalent to setting $A_\alpha = 1.7$) is in fact close to the analytical value of $A_\alpha = 1.85$, when the final noise level is taken to be $\sigma_{\text{end}} = 0.001$.

Attractor Warp: Linear log-SNR. Finally, we consider a warp induced by linear motion in log-SNR space (Lu et al., 2022). This schedule is naturally specified by first defining a trajectory in log-SNR and then converting that trajectory back to noise-level space. For the Rectified Flow interpolation, the log signal-to-noise ratio is

$$\lambda(\sigma) = \log \mathrm{SNR}(\sigma) = 2 \log \frac{1-\sigma}{\sigma}. \tag{55}$$

Solving for σ in terms of a log-SNR value ℓ gives

$$\lambda^{-1}(\ell) = \frac{1}{1+\exp(\ell/2)}. \tag{56}$$

Moving from noise to data corresponds to increasing log-SNR: as $\sigma \to 1$, $\lambda(\sigma) \to -\infty$, while as $\sigma \to 0$, $\lambda(\sigma) \to +\infty$. We therefore define a linear trajectory in log-SNR space between finite endpoints $\lambda_{\min}$ and $\lambda_{\max}$:

$$\bar{\lambda}(s) = \lambda_{\min} + s(\lambda_{\max} - \lambda_{\min}), \qquad s \in [0,1], \tag{57}$$

$$\rho_{\log \mathrm{SNR}}(s) = \lambda^{-1}(\bar{\lambda}(s)) = \frac{1}{1+\exp(\bar{\lambda}(s)/2)}. \tag{58}$$

The corresponding positive denoising speed is

$$\eta_{\log \mathrm{SNR}}(\rho_{\log \mathrm{SNR}}(s)) := -\frac{d\rho_{\log \mathrm{SNR}}(s)}{ds} = \frac{\lambda_{\max} - \lambda_{\min}}{2} \rho_{\log \mathrm{SNR}}(s) \left(1 - \rho_{\log \mathrm{SNR}}(s)\right). \tag{59}$$

Equivalently, as a direct function of the current noise level σ,

$$\eta_{\log \mathrm{SNR}}(\sigma) = \frac{\lambda_{\max} - \lambda_{\min}}{2} \sigma(1-\sigma). \tag{60}$$

This warp vanishes as $\sigma \to 0$, so it induces attractor-style dynamics near the data manifold. It also vanishes as $\sigma \to 1$, allocating smaller updates near the pure-noise endpoint.

G.4 Relation Between η Warp Function and Budget-Adaptive Inference.

The budget-adaptive inference algorithm presented in Algorithm 1 defines an update rule with η taken to be the identity warp. Budget-adaptivity calibrates the update scaling factor so that inference can expect to finish at the data manifold based on the most recent noise level estimate. This achieves a similar effect to an attractor warp, helping prevent inference with acceleration techniques like NAG from overshooting. Further exploration of parameterizations that incorporate the number of remaining steps $N - i$ into more general η functions is an interesting direction for future work.

H Details on Gradient Based Inference with Denoising Video Generative Models

Nesterov Accelerated Gradient (NAG) is a method for accelerated function optimization. It is a classical algorithm that leverages momentum and a lookahead step with theoretical guarantees

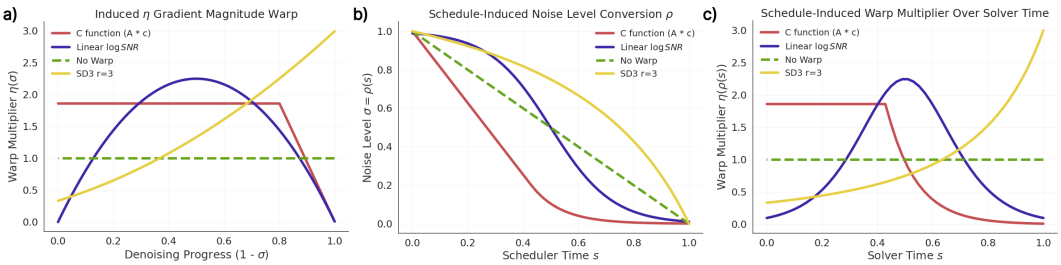

Figure 20: **Conversions between noise level, discretized solver time, and induced warp. a)** Conversion between σ and update multiplier under different choices of the warp function η. **b)** Induced conversion between solver time s and noise level σ through the trajectory ρ. Equally sized steps in solver time can correspond to different changes in noise level. **c)** Warp multiplier as a function of solver time through the composition $\eta(\rho(s))$. Due to the vanishing update multiplier for attractor-style choices of η, equally sized steps in solver time take increasingly smaller steps as the sampler approaches data.

Algorithm 3: Nesterov Accelerated Gradient Sampling Step with EQF

Input: model f_{EQF}, readout h_ω, sample $\mathbf{x}^{(i)}$, momentum $\mathbf{m}^{(i)}$, warp $\eta(\cdot)$, momentum coefficient μ

$\mathbf{x}^{(i)}_{\text{look}} \leftarrow \mathbf{x}^{(i)} - \mu\mathbf{m}^{(i)}$;

$\widehat{\mathbf{v}}^{(i)}, \mathbf{z}^{(i)} \leftarrow f_{\text{EQF}}(\mathbf{x}^{(i)}_{\text{look}})$;

$\widehat{\boldsymbol{\sigma}}^{(i)} \leftarrow h_\omega(\mathbf{z}^{(i)})$;

$\mathbf{m}^{(i+1)} \leftarrow \mu\mathbf{m}^{(i)} + \eta(\widehat{\boldsymbol{\sigma}}^{(i)}) \odot \widehat{\mathbf{v}}^{(i)}$;

$\mathbf{x}^{(i+1)} \leftarrow \mathbf{x}^{(i)} - \mathbf{m}^{(i+1)}$;

return $\mathbf{x}^{(i+1)}, \mathbf{m}^{(i+1)}$;

over gradient descent in general smooth convex function optimization (Nesterov, 1983; Sutskever et al., 2013). The algorithm is for fixed-point iteration, making it a suitable equilibrium landscape optimizer when using an inference warp η such that $\eta(\widehat{\boldsymbol{\sigma}})f_{\text{EQF}}(\mathbf{x}) = \mathbf{0}$ for $\mathbf{x}$ on the data manifold. We empirically confirm in Section 5 that when the warp is chosen to not lead to such an attractor-style field at the data manifold, NAG produces subpar samples as expected, because the algorithm is prevented from converging. We provide a simple NAG sampling algorithm with EQF for one sample step in Algorithm 3; note that the signs of gradient terms align with the convention for the training vector $\mathbf{v} = \boldsymbol{\epsilon} - \mathbf{x}$. Future work may consider accelerated sampling with other fixed point iteration methods.

I Autoregressive Inference Procedures for Standard Denoising Video Generative Models

In this section, we elaborate on the static inference algorithms used by standard Flow Matching and Diffusion video models during inference, such as those used by Chen et al. (2024); Ruhe et al. (2024); Song et al. (2025); Hafner et al. (2025). EQF instead relies on its own estimate of progress to the data manifold within a closed loop, leveraging gradient-based solvers to propose data-adaptive steps and possessing unique early stopping capabilities (Section 4.3, Algorithm 2). However, some aspects, such as the bake-in period, are also shared by EQF. Though the autoregressive inference style we utilize here has fundamental weaknesses in preserving information across repeated inference rounds, we isolate our study to the denoising and inference procedure, and leave the memory issue to future work (Lillemark et al., 2026).

Framewise Noise Level Training Enables Stable Autoregressive Rollouts. Rolling inference algorithms correlate the noise level of a frame with its index in the video, with lower noise levels mapping to earlier frames and higher noise levels to later frames. This capability is enabled by training with different noise levels per frame, instead of having just one noise level for all frames. Diffusion Forcing training assigns independent noise levels per frame (Chen et al., 2024), generally preferred by practitioners since it enables a variety of inference schedules to be 'within training distribution' (Decart et al., 2024; Hafner et al., 2025). Other methods like Rolling Diffusion bake in a

specific inference schedule during training, such as one resembling a staircase with increasing noise levels per index (Ruhe et al., 2024; Cachay et al., 2026).

Inference Schedule Parameter Definitions. These inference schedules specify a global geometry that is predetermined prior to inference, and we expand upon this here in order to explain the baseline inference algorithm for easier comparison. It is static, fully defining the number of steps per frame and the noise levels that will be visited. An inference schedule is defined by three fundamental quantities: the length of the prediction horizon at each denoising step H; the stride length of a group of frames that will be emitted together, denoted S; and the number of denoising steps the model will take for a frame, denoted as the depth D. The algorithm, as presented, can be run in an infinite loop so that any amount of frames can be generated.

Global and Local Constraints of Inference Schedules. The inference algorithm alternates between an outer loop operation of sliding forward the denoising horizon and the inner loop refinement of the active window. The outer loop can run indefinitely for infinite autoregressive generation. The stride breaks the prediction horizon into $G = H/S$ groups of frames, where each group remains at the same noise level during inference and has S frames. Each iteration of the outer loop symmetrically moves forward by S frames, popping S clean frames that have finished denoising, appending S noisy frames at the end, and shifting the active window forward by S frames to continue the rollout. In between these pop-shifts, the inner loop of the algorithm takes exactly $K = D/G$ denoising steps in order to satisfy the global consistency of D steps per frame; intuitively, splitting the horizon into G groups means that we need to take K steps for each pop-shift.

Noise Level Schedule Matrix. The number of denoising steps D determines a set of noise levels that each frame will visit during inference, $\{\sigma^{(d)}\}_{d=0}^{D}$ where $\sigma^{(0)} = 0$ and $\sigma^{(D)} = 1$. These noise levels may be spaced out over the interval $[0, 1]$ in essentially any manner. Common approaches include spacing them out evenly, and extending the EDM inference schedule from static image generation to each chunk of video (Cachay et al., 2026). We refer to choices of these visited noise levels as warps of the solver time s (see Appendix G). In our exposition of the algorithm, for simplicity we can assume the noise levels are evenly spaced. At a given iteration during the steady state of the algorithm (after the bake-in period), the noise level transitions for K inner steps for the active horizon are

$$[\sigma^{(K)} \cdot \mathbf{1}, \ldots, \sigma^{((G-1)K)} \cdot \mathbf{1}, \sigma^{(D)} \cdot \mathbf{1}] \quad \rightarrow \quad [\sigma^{(0)} \cdot \mathbf{1}, \ldots, \sigma^{((G-2)K)} \cdot \mathbf{1}, \sigma^{((G-1)K)} \cdot \mathbf{1}], \quad (61)$$

where each multiplication by $\mathbf{1}$ yields a group of S frames that all have the same noise level. After that transition, the first group of S frames have reached the clean boundary and are 'emitted.' The horizon is then pop-shifted forward and a new group of fully noisy frames with noise level $\sigma^{(D)}$ is appended to recover the same steady-state noise geometry.

Bake-In Period for Initial Conditions. The rolling schedule described above is only well-defined once the local horizon has reached its steady-state triangular noise geometry. However, at initialization all H frames in the horizon are sampled from pure noise, so the noise schedule begins as

$$[\sigma^{(D)} \cdot \mathbf{1}, \sigma^{(D)} \cdot \mathbf{1}, \ldots, \sigma^{(D)} \cdot \mathbf{1}], \quad (62)$$

The bake-in period constructs this steady-state geometry through $G - 1$ masked bake-in iterations. At bake-in iteration $g \in \{1, \ldots, G-1\}$, the model evaluates the full context and prediction horizon, but the denoising update is applied only to the first g groups of the horizon. Thus, the first group receives $(G-1)K$ bake-in steps, the second group receives $(G-2)K$ steps, and so on, while the final group remains at pure noise. After bake-in, the horizon therefore has noise geometry

$$[\sigma^{(K)} \cdot \mathbf{1}, \sigma^{(2K)} \cdot \mathbf{1}, \ldots, \sigma^{((G-1)K)} \cdot \mathbf{1}, \sigma^{(D)} \cdot \mathbf{1}]. \quad (63)$$

The first ordinary inference iteration then applies K steps to the entire horizon, bringing the first group to the clean boundary before it is emitted.

Autoregressive Context Conditioning and Full Algorithm. The final specification of the autoregressive inference algorithm determines how context $\mathbf{c}$ of length C guides the generation of the active prediction horizon $\mathbf{x}$. Similar to History Guidance (Song et al., 2025), we treat the conditioning frames to be unified within the same self-attention window as $\mathbf{x}$. The noise level conditioning is injected into the token space through AdaLN on a per frame basis (Peebles & Xie, 2023; Song et al., 2025).

The schedules explained in Equation 61 define the transitions of noise levels in the active horizon. Denoting the first token in the active window to have index 1, the full input to the model is formed from concatenating the context (denoted with indices for clarity) $\mathbf{c}_{-C+1:0}$ with the active horizon

Algorithm 4: Rolling Inference with Masked Bake-In

Input: Model f_{FM}, context length C, horizon H, stride S, denoising steps D, inner steps K, initial context $\mathbf{c}_{-C+1:0}$

$M \leftarrow H/S$;
$\mathbf{x}_{1:H} \sim \mathcal{N}(0, I)$;
$\boldsymbol{\sigma}_{1:H} \leftarrow \mathbf{1}$;
`// Construct the initial denoising staircase, bake-in period`
for $b \leftarrow 1$ **to** $M - 1$ **do**
  $\mathbf{m}_{1:H} \leftarrow [\mathbf{1}_{bS}, \mathbf{0}_{H-bS}]$;
  $(\mathbf{x}, \boldsymbol{\sigma}) \leftarrow \textsc{MaskedInnerDenoise}(f_{\mathrm{FM}}, \mathbf{c}, \mathbf{x}, \boldsymbol{\sigma}, D, K, \mathbf{m})$;
`// Ordinary rolling generation`
while NotDone **do**
  $(\mathbf{x}, \boldsymbol{\sigma}) \leftarrow \textsc{MaskedInnerDenoise}(f_{\mathrm{FM}}, \mathbf{c}, \mathbf{x}, \boldsymbol{\sigma}, D, K, \mathbf{1}_H)$;
  $\textsc{Emit}(\mathbf{x}_{1:S})$;
  $\mathbf{c} \leftarrow [\mathbf{c}_{S+1:C}, \mathbf{x}_{1:S}]$;
  $\mathbf{x}_{1:H-S} \leftarrow \mathbf{x}_{S+1:H}$;
  $\boldsymbol{\sigma}_{1:H-S} \leftarrow \boldsymbol{\sigma}_{S+1:H}$;
  $\mathbf{x}_{H-S+1:H} \sim \mathcal{N}(0, I)$;
  $\boldsymbol{\sigma}_{H-S+1:H} \leftarrow \mathbf{1}$;

Algorithm 5: Masked Inner Denoising

Input: Model f_{FM}, context $\mathbf{c}$, horizon $\mathbf{x}$, noise levels $\boldsymbol{\sigma}$, denoising steps D, inner steps K, update mask $\mathbf{m}$

for $k \leftarrow 1$ **to** K **do**
  $\widehat{\mathbf{v}} \leftarrow f_{\mathrm{FM}}([\mathbf{c}, \mathbf{x}], [\mathbf{0}_C, \boldsymbol{\sigma}])$;
  $\Delta\boldsymbol{\sigma} \leftarrow \frac{1}{D}\mathbf{m}$;
  $\mathbf{x} \leftarrow \mathbf{x} - \Delta\boldsymbol{\sigma} \odot \widehat{\mathbf{v}}_{1:H}$;
  $\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma} - \Delta\boldsymbol{\sigma}$;
return $(\mathbf{x}, \boldsymbol{\sigma})$;

$\mathbf{x}_{1:H}$. The unified prediction horizon is $[\mathbf{c}_{-C+1:0}, \mathbf{x}_{1:H}]$, with the noise levels fixed to $\mathbf{0}_{-C+1:0}$ for the context, and to $\boldsymbol{\sigma}_{1:H}$ for the active horizon.

The outer loop and masked inner loop can be found in Algorithms 4 and 5, respectively. In the algorithm, we define the function EMIT to slide the S most recent cleanly generated frames to the context. During the bake-in period, only the prefix selected by the bake-in mask is updated. After the steady state is reached, each clean group is emitted and appended to the context, and the window slides forward by S context frames, thereby replacing the oldest S context frames.

J Online Replanning Inference Details

We introduce an online replanning experiment intended to more closely reflect downstream applications of video generation models, such as world modeling in a closed loop environment. In such settings, an agent predicts a future video trajectory using a video generation model as a world model, turning the imagined future into an executable action (Du et al., 2023; Chen et al., 2025). After execution, the agent receives a new observation from the environment (of a length less than the full predicted trajectory). The remaining prediction was generated before this new observation was made, and may therefore no longer be consistent with the new full ground truth state. An efficient planner should try to optimize the tradeoff between quality and cost for replanning. Reusing the previous plan reduces generation cost, but risks propagating predictions that were generated under stale observations. Regenerating the entire horizon is more responsive to new information, but requires additional sampling compute to regenerate all those frames.

We study this tradeoff with a new inference setting using a denoising-based compatibility score inspired by Adaptive Online Replanning with Diffusion Models, a paradigm in which a diffusion-based state-action plan can be re-evaluated as observations are streamed (Zhou et al., 2023). In our setting, we isolate the video generation component, treating actions as fixed from the dataset. The compatibility score evaluates whether the existing prediction remains consistent with the newly

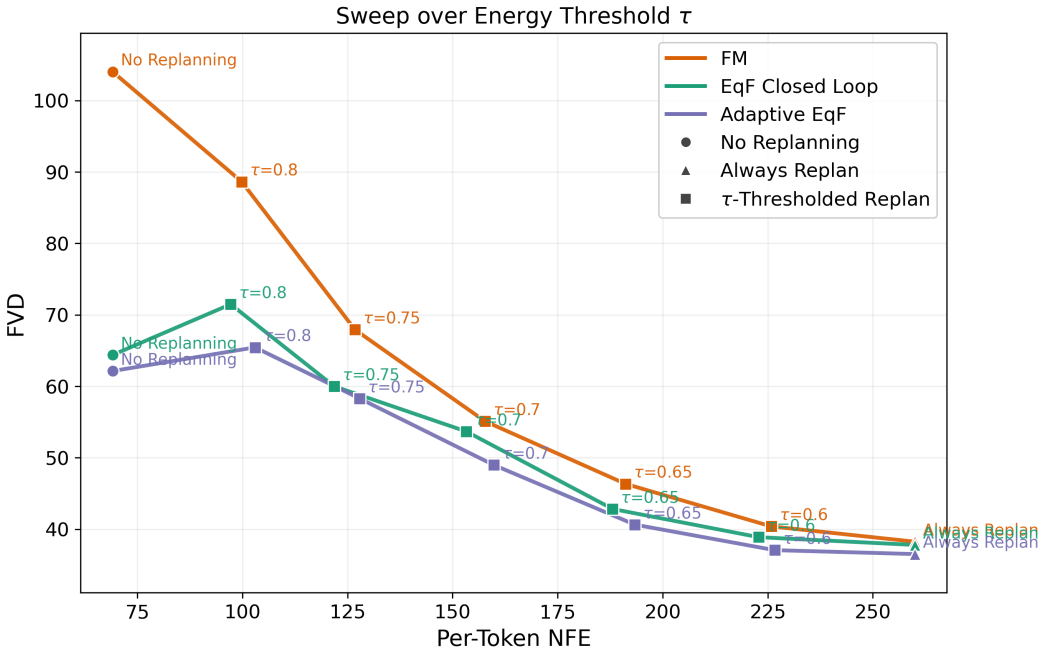

Figure 21: Online replanning on Minecraft. The replanning threshold τ controls when the retained future horizon is discarded and regenerated. $\tau = \infty$ never replans, while $\tau = 0$ always replans. Intermediate thresholds trade off reuse of the current plan against responsiveness to new observations.

observed context. When the score indicates that the prediction remains compatible, the planner retains the existing future and generates only the newly appended portion of the horizon. Otherwise, it discards the previous prediction and regenerates the entire future. We find that EQF provides a stronger quality/compute tradeoff than noise-conditional Flow Matching across replanning thresholds.

J.1 Clean Buffer, Evaluation Buffer, and Compatibility Score

The replanning setting differs from the standard rolling inference setting we use in other experiments. Typically, generated frames are both evaluated as predictions, and reused as context for subsequent generation. Here, the model maintains a predicted future video plan, commits a short prefix of that plan to the evaluation buffer, and then receives the corresponding ground-truth observations to use as context for subsequent generation. Below, we describe how the algorithm maintains the state, and how the compatibility score is defined and used to decide whether to regenerate.

Planning State and Observation Feedback. We use the notation of Appendix I. For a generation round, let C denote the context length, H the prediction horizon, and S the number of frames observed at each replanning step. S also denotes the number of frames committed to the evaluation buffer at each step. For each outer replanning step ℓ, let $r_\ell = \ell \cdot S$ denote the number of frames committed before step ℓ. The algorithm maintains the evaluation buffer $\mathbf{x}^{\text{eval}}$, which stores the predictions used for downstream evaluation, and a working buffer of clean frames $\mathbf{b}^{(\ell)}_{-C+1:H}$. The context $\mathbf{b}^{(\ell)}_{-C+1:0}$ contains the most recent C ground-truth observations, while $\mathbf{b}^{(\ell)}_{1:H}$ contains the current predicted future plan of fully denoised frames. Here, clean means that they are not noisy frames; only the context is ground truth, while the future plan consists of denoised predictions. The conditioning sequence, which for our experiments is the ground truth action sequence, is denoted as $\mathbf{y}^{\text{data}}$.

At each step ℓ, the first S frames of the current working buffer are committed to the evaluation buffer,

$$\mathbf{x}^{\text{eval}}_{r_\ell+1:r_\ell+S} = \mathbf{b}^{(\ell)}_{1:S}. \tag{64}$$

The corresponding ground-truth observations are then revealed and written into the working buffer,

$$\mathbf{b}^{(\ell)}_{1:S} = \mathbf{x}^{\text{gt}}_{r_\ell+1:r_\ell+S}. \tag{65}$$

After this update, $\mathbf{b}^{(\ell)}_{-C+1:S}$ forms an observed prefix of length $C+S$, while $\mathbf{b}^{(\ell)}_{S+1:H}$ is the existing future plan generated before the new observations were available. The replanning decision evaluates whether this existing future prediction remains compatible with the updated observed prefix.

Compatibility Score. We measure whether the retained future remains consistent with the updated observed context. Specifically, we temporarily noise the existing prediction and evaluate the model's denoising error while conditioning on the newly observed prefix. At replanning step ℓ, we evaluate at M probe noise levels, $\sigma_m \in [0,1]$. For each probe level, the observed prefix remains clean and only the existing future is noised:

$$\boldsymbol{\sigma}^{(m)}_{-C+1:H} = [\mathbf{0}_{C+S}, \sigma_m \mathbf{1}_{H-S}]\,. \tag{66}$$

The clean working buffer $\mathbf{b}^{(\ell)}$ is used to construct both the temporary noised value and its corresponding velocity target. In the same denoising style as Equation 1, for sampled Gaussian noise $\boldsymbol{\epsilon}^{(m)}$, we construct

$$\mathbf{b}^{(\ell),\boldsymbol{\sigma}^{(m)}}_{-C+1:H} = (\mathbf{1} - \boldsymbol{\sigma}^{(m)}_{-C+1:H}) \odot \mathbf{b}^{(\ell)}_{-C+1:H} + \boldsymbol{\sigma}^{(m)}_{-C+1:H} \odot \boldsymbol{\epsilon}^{(m)}, \tag{67}$$

$$\mathbf{v}^{(\ell,m)} = \boldsymbol{\epsilon}^{(m)} - \mathbf{b}^{(\ell)}. \tag{68}$$

We define the compatibility score as

$$\mathcal{S}^{(\ell)} = \frac{1}{M}\sum_{m=1}^{M} \mathrm{MSE}_{S+1:H}\Big(f_\theta\Big(\mathbf{b}^{(\ell),\boldsymbol{\sigma}^{(m)}}_{-C+1:H}, \boldsymbol{\sigma}^{(m)}_{-C+1:H}\,;\, \mathbf{y}^{\mathrm{data}}_{r_\ell - C+1:r_\ell+H}\Big), \mathbf{v}^{(\ell,m)}\Big), \tag{69}$$

where $\mathrm{MSE}_{S+1:H}$ averages over all elements of the existing future frames. Lower values of $\mathcal{S}^{(\ell)}$ indicate that the existing prediction is more compatible with the newly observed prefix. In other words, when the denoising loss is low even with the new observed frames, it is a proxy for the model estimating that the frames are consistent with each other. The noise level condition is included in this equation for clarity, as used by noise-conditional baselines like Flow Matching; for EQF it is ignored.

Plan Reuse and Regeneration. A threshold τ converts the compatibility score into a replanning decision.

Plan reuse. If $\mathcal{S}^{(\ell)} < \tau$, the existing future prediction is considered compatible with the new observations. The existing predicted $H-S$ frames are shifted forward, S fresh noisy frames are appended to restore the full prediction horizon, and only these newly appended frames are denoised. The existing prediction therefore remains unchanged and its next S frames are committed at the next replanning round.

Plan regeneration. If $\mathcal{S}^{(\ell)} \geq \tau$, the existing prediction is considered incompatible with the new observations. The retained $H-S$ frames are discarded and replaced by full noise, S additional noisy frames are appended, and the complete H-frame future horizon is regenerated.

The compatibility probes are temporary and do not modify the clean working buffer. The complete procedure is provided in Algorithms 6,7,8. Note that the function MASKEDDENOISE in Algorithm 8 is replaced by a specialized EQF sampler when appropriate, restricted to the frames indicated by the mask $\mathbf{m}_{1:H}$.

J.2 Planning on Minecraft

Experimental Setup. We evaluate the online replanning procedure on Minecraft using the dataset and model settings described in Appendix D.2. We compute the compatibility score using $M=5$ evenly spaced probe noise levels, which we found to provide stable estimates comparable to larger values of M. We sweep the replanning threshold τ over a range calibrated to the observed compatibility scores and include the two boundary values of τ. Setting $\tau = 0$ causes the model to replan after every newly observed stride, while $\tau = \infty$ always retains the previous prediction and generates only the newly appended frames. Intermediate thresholds adaptively trade off between these two behaviors.

Quality/Compute Tradeoff. Figure 21 reports prediction quality as a function of replanning cost. Specifically, we measure the average per-frame number of function evaluations (NFE) to achieve an overall FVD. Always replanning produces the strongest prediction quality, as the complete future horizon is regenerated after every new observation, but requires the most computation. Never replanning minimizes regeneration cost but can retain predictions that are no longer consistent with the observed trajectory. Intermediate thresholds provide a smooth tradeoff by regenerating the horizon

only when its compatibility score exceeds τ. Across this sweep, Budget-Adaptive EQF achieves the strongest quality/compute tradeoff, while the Flow Matching baseline exhibits the weakest tradeoff.

Mixed-Context Prediction. The retained horizon can contain frames generated before the newest observations were available, while newly appended frames are generated after incorporating those observations. The resulting prediction may therefore combine frames produced under different context states and can become internally inconsistent. This represents an out-of-distribution denoising setting because both Flow Matching and EQF are trained on noisy versions of temporally consistent videos. We hypothesize that EQF is more robust to this mismatch because it does not require an externally supplied noise level to accurately describe the effective state of each frame. In contrast, a supplied noise level that does not match the noisy video state may compound the extent to which the input is out-of-distribution, making Flow Matching's denoising error higher. This interpretation is consistent with the noise condition mismatch analyzed in Appendix B.2.

Alternative Renoising Policies. Following prior work on adaptive online replanning (Zhou et al., 2023), we also evaluated policies that partially renoise the existing prediction rather than either preserving it or fully restarting the horizon. These included pyramidal profiles, in which the noise level increases across the retained horizon, and flat profiles, in which all retained frames are renoised to the same intermediate level. We additionally considered hybrid policies interpolating between pyramidal renoising and full restart. These variants did not consistently improve over full restart and were generally less stable across thresholds. We therefore report the simpler restart-based replanning procedure, which provides the clearest quality/compute tradeoff.

We would like to emphasize here that due to the action condition coming from the ground truth data instead of being predicted, the discrepancy between the predicted future and the ground truth observation may be less than what it would be in a more realistic full planning setting. We hope this work can further inspire future work that explores these quality/cost tradeoffs in more realistic world modeling settings.

Algorithm 6: Closed-Loop Planning Inference with Observation Feedback. COMPATIBILITYSCORE is implemented in Algorithm 7. MASKEDDENOISE is implemented in Algorithm 8.

Input: Model f_θ, ground-truth video $\mathbf{x}^{\mathrm{gt}}_{-C+1:R}$, conditioning sequence $\mathbf{y}^{\mathrm{data}}$, context length C, horizon H, observation stride S, rollout length R (divisible by S), denoising steps D, compatibility threshold τ, probe noise levels $\{\sigma^{(m)}_{\mathrm{probe}}\}_{m=1}^{M}$

Output: Evaluation buffer $\mathbf{x}^{\mathrm{eval}}_{1:R}$

$r \leftarrow 0$;
$\mathbf{x}^{\mathrm{eval}}_{1:R} \leftarrow \emptyset$;
// Initialize the working buffer with ground truth context and a noisy future
$\mathbf{b}_{-C+1:0} \leftarrow \mathbf{x}^{\mathrm{gt}}_{-C+1:0}$;
$\mathbf{b}_{1:H} \sim \mathcal{N}(0, I)$;
// Bootstrap an initial clean plan over the full horizon, initialize active mask
$\mathbf{m}_{1:H} \leftarrow \mathbf{1}$;
$\mathbf{b}_{-C+1:H} \leftarrow \textsc{MaskedDenoise}(f_\theta, \mathbf{b}_{-C+1:H}, \mathbf{m}_{1:H}, D, \mathbf{y}^{\mathrm{data}}_{r-C+1:r+H})$;
while $r < R$ **do**
  // Commit generated frames to the evaluation buffer
  $\mathbf{x}^{\mathrm{eval}}_{r+1:r+S} \leftarrow \mathbf{b}_{1:S}$;
  // Reveal ground-truth observations and overwrite the clean buffer
  $\mathbf{b}_{1:S} \leftarrow \mathbf{x}^{\mathrm{gt}}_{r+1:r+S}$;
  // Score whether the retained future plan is compatible with the new observations
  $\mathcal{E} \leftarrow \textsc{CompatibilityScore}(f_\theta, \mathbf{b}_{-C+1:H}, S, \{\sigma^{(m)}_{\mathrm{probe}}\}_{m=1}^{M}, \mathbf{y}^{\mathrm{data}}_{r-C+1:r+H})$;
  // Slide the clean ground-truth context
  $\mathbf{b}_{-C+1:0} \leftarrow [\mathbf{b}_{-C+1+S:0}, \mathbf{x}^{\mathrm{gt}}_{r+1:r+S}]$;
  // Slide the existing future prediction
  $\mathbf{b}_{1:H-S} \leftarrow \mathbf{b}_{S+1:H}$;
  $r \leftarrow r + S$;
  if $\mathcal{E} < \tau$ **then**
    // Reuse the existing prediction; only generate the newly appended stride
    $\mathbf{b}_{H-S+1:H} \sim \mathcal{N}(0, I)$;
    $\mathbf{m}_{1:H} \leftarrow [\mathbf{0}_{H-S}, \mathbf{1}_S]$;
  else
    // Discard and regenerate the complete future horizon
    $\mathbf{b}_{1:H} \sim \mathcal{N}(0, I)$;
    $\mathbf{m}_{1:H} \leftarrow \mathbf{1}$;
  // Denoise only the frames indicated by the mask
  $\mathbf{b}_{-C+1:H} \leftarrow \textsc{MaskedDenoise}(f_\theta, \mathbf{b}_{-C+1:H}, \mathbf{m}_{1:H}, D, \mathbf{y}^{\mathrm{data}}_{r-C+1:r+H})$;
return $\mathbf{x}^{\mathrm{eval}}_{1:R}$;

Algorithm 7: Compatibility Score from a Clean Working Buffer

Input: Model f_θ, clean buffer $\mathbf{b}_{-C+1:H}$, observation stride S, probe noise levels $\{\sigma_{\text{probe}}^{(m)}\}_{m=1}^{M}$, conditions for the active window $\mathbf{y}_{-C+1:H}$
Output: Compatibility score $\mathcal{E}$

$\mathcal{E} \leftarrow 0$;
for $m \leftarrow 1$ **to** M **do**
  `// The clean buffer is not modified; this is only a temporary noised probe`
  $\boldsymbol{\epsilon}_{S+1:H}^{(m)} \sim \mathcal{N}(0, I)$;
  `// Observed prefix stays clean; only retained predictions are noised`
  $\boldsymbol{\sigma}_{-C+1:S}^{(m)} \leftarrow \mathbf{0}_{C+S}$;
  $\boldsymbol{\sigma}_{S+1:H}^{(m)} \leftarrow \sigma_{\text{probe}}^{(m)} \mathbf{1}_{H-S}$;
  $(\mathbf{b}^{\boldsymbol{\sigma}^{(m)}})_{-C+1:S} \leftarrow \mathbf{b}_{-C+1:S}$;
  $(\mathbf{b}^{\boldsymbol{\sigma}^{(m)}})_{S+1:H} \leftarrow (\mathbf{1} - \boldsymbol{\sigma}_{S+1:H}^{(m)}) \odot \mathbf{b}_{S+1:H} + \boldsymbol{\sigma}_{S+1:H}^{(m)} \odot \boldsymbol{\epsilon}_{S+1:H}^{(m)}$;
  $\mathbf{v}_{S+1:H}^{(m)} \leftarrow \boldsymbol{\epsilon}_{S+1:H}^{(m)} - \mathbf{b}_{S+1:H}$;
  $\mathbf{u}_{-C+1:H}^{(m)} \leftarrow f_\theta(\mathbf{b}_{-C+1:H}^{\boldsymbol{\sigma}^{(m)}}, \boldsymbol{\sigma}_{-C+1:H}^{(m)}\,;\, \mathbf{y}_{-C+1:H})$;
  `// MSE averaged over all temporal, spatial, and channel dimensions in the existing future`
  $\mathcal{E} \leftarrow \mathcal{E} + \text{MSE}\Big(\mathbf{u}_{S+1:H}^{(m)}, \mathbf{v}_{S+1:H}^{(m)}\Big)$;
$\mathcal{E} \leftarrow \frac{1}{M}\mathcal{E}$;
return $\mathcal{E}$;

Algorithm 8: Masked future denoising for planning. Initialized as fixed-step Euler schedule for simplicity.

Input: Model f_θ, buffer $\mathbf{b}_{-C+1:H}$, binary active mask $\mathbf{m}_{1:H}$, denoising steps D, conditioning window $\mathbf{y}_{-C+1:H}$

`// Clean context and inactive planned frames have zero noise`
$\boldsymbol{\sigma}_{-C+1:0} \leftarrow \mathbf{0}_C$;
$\boldsymbol{\sigma}_{1:H} \leftarrow \mathbf{m}_{1:H}$;
$\Delta\boldsymbol{\sigma}_{1:H} \leftarrow \frac{1}{D}\mathbf{m}_{1:H}$;
for $d \leftarrow 1$ **to** D **do**
  $\mathbf{u}_{-C+1:H} \leftarrow f_\theta(\mathbf{b}_{-C+1:H}, \boldsymbol{\sigma}_{-C+1:H}\,;\, \mathbf{y}_{-C+1:H})$;
  `// Only noised future frames are updated`
  $\mathbf{b}_{1:H} \leftarrow \mathbf{b}_{1:H} - \Delta\boldsymbol{\sigma}_{1:H} \odot \mathbf{u}_{1:H}$;
  $\boldsymbol{\sigma}_{1:H} \leftarrow \boldsymbol{\sigma}_{1:H} - \Delta\boldsymbol{\sigma}_{1:H}$;
return $\mathbf{b}_{-C+1:H}$;