GROUND YOUR STEPS: OPTIMIZING SAMPLING SCHEDULES IN FLOW MATCHING ROBOT POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

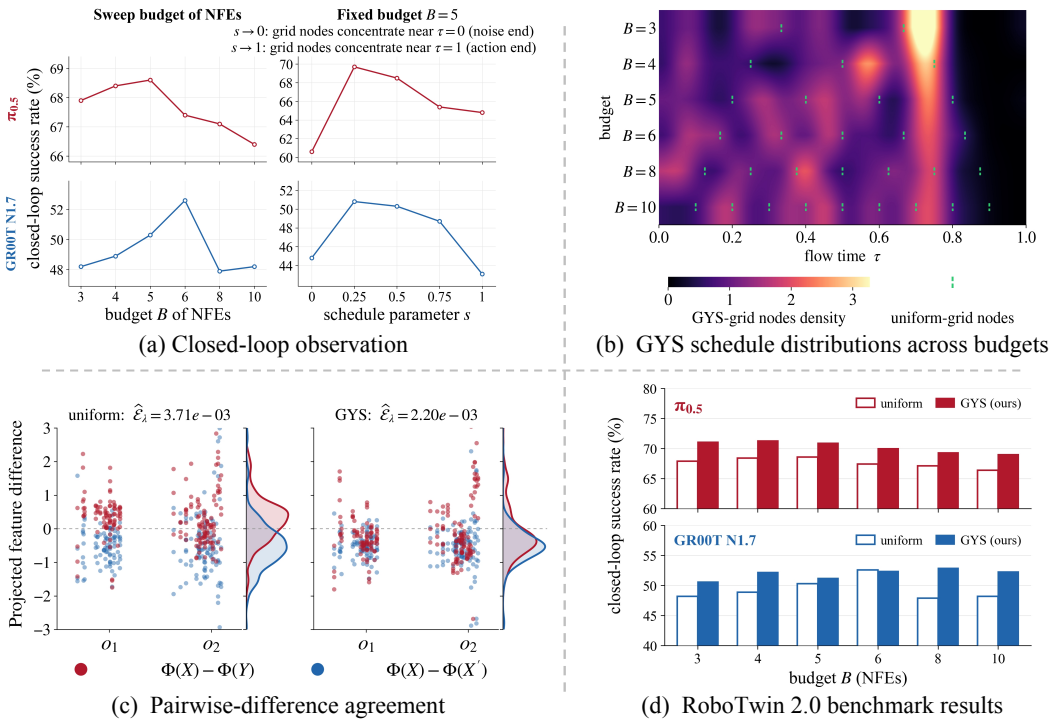
Flow matching robot policies generate action chunks by integrating a learned velocity field at inference. However, under a fixed budget of neural function evaluations (NFEs), the integration result depends on the sampling schedule. This motivates optimizing the schedule to bring the generated action distribution closer to the demonstrations and improve task success. We propose Ground Your Steps (GYS), a training-free method that optimizes the sampling schedule by matching the generated and demonstrated chunk distributions at each observation. GYS applies truncated fractional differencing along each arm joint's action sequence and compares two types of pairwise differences: generated–demonstrated and generated–generated, using independent policy samples at each observation. After fractional differencing, the distributions of generated–demonstrated and generated–generated pairwise differences coincide exactly when the distributions of generated and demonstrated action chunks match. GYS minimizes the energy distance between these distributions to obtain an improved schedule for each task and NFE budget. Compared with the uniform grid, GYS brings the generated chunk distribution closer to the demonstrated distribution and raises the success rate through the sampling schedule alone. On RoboTwin 2.0 benchmarks, GYS improves success rates at nearly every budget, with gains of 2.7 percentage points for $\pi_{0.5}$ and 2.6 percentage points for GR00T N1.7. The gains also transfer to three real-world manipulation tasks, reaching up to 3.3 percentage points.

1 INTRODUCTION

Vision–language–action (VLA) models translate visual observations and language instructions into robot actions (Ma et al., 2026b; Kawaharazuka et al., 2025). Flow-based VLAs couple pretrained vision–language backbones with action experts (Black et al., 2025b). Recent models use flow matching (Black et al., 2025a; NVIDIA et al., 2025) to generate these actions in chunks (Zhao et al., 2023). During training, the action expert predicts the velocities of straight paths from Gaussian noise to demonstrated action chunks. Under squared loss, the optimal prediction is the conditional average of these velocities at a given state, flow time, and observation, called the marginal velocity field (Lipman et al., 2023; Liu et al., 2023). Exact integration of this field transports Gaussian noise to the demonstrated action distribution (Albergo & Vanden-Eijnden, 2023). In practice, however, the policy integrates a learned approximation to the marginal field with finite neural function evaluations (NFEs), so the generated action distribution can deviate from the demonstrated distribution.

Two mechanisms shape the generated action distribution under this finite-step integration. (i) *Trajectory curvature*—averaging differently directed path velocities can produce curved marginal trajectories, which coarse integration steps may fail to resolve (Ma et al., 2026a; Kornilov et al., 2024). (ii) *Lipschitz window*—near the action endpoint, the marginal field can become highly sensitive to the action state, allowing small deviations from the marginal flow to be amplified (Gao et al., 2024; Chen et al., 2026b) (Appendix A.3). Since curvature and the amplification of small deviations vary along flow time, evaluations at different times need not influence the endpoint distribution equally. Choosing these evaluation times, the sampling schedule, therefore provides a way to change the generated distribution. Our goal is to optimize the sampling schedule to bring the generated distribution into agreement with the demonstrations and improve task success for each fixed budget B.

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



(a) Closed-loop observation

(b) GYS schedule distributions across budgets

(c) Pairwise-difference agreement

(d) RoboTwin 2.0 benchmark results

Figure 1: Motivation and effects of GYS. (a) Closed-loop success varies non-monotonically with the NFE budget B under uniform sampling and changes with the grid at B=5 (Table 6), consistent with the proposed curvature and late-time sensitivity mechanisms. (b) Across budgets, GYS places fewer interior sampling nodes near the action endpoint, where deviations may be amplified; green ticks mark uniform-grid nodes. (c) For two observation groups o_1 and o_2 pooled from the calibration set, GYS lowers the energy distance between the two transformed pairwise-difference distributions, the objective used to match transformed generated and demonstrated action laws. (d) Over 50 RoboTwin 2.0 tasks, GYS improves mean closed-loop success at nearly every fixed budget.

To see whether these distributional effects matter in closed loop, we vary the NFE budget and sampling schedule for $\pi_{0.5}$ and GR00T N1.7 across 50 RoboTwin 2.0 tasks (Figure 1, panel (a)). Under uniform sampling, success peaks at $B = 5$ for $\pi_{0.5}$ and $B = 6$ for GR00T N1.7, then falls at larger budgets. At fixed $B = 5$, a moderate bias toward the noise endpoint outperforms the uniform grid, whereas concentrating nodes near either endpoint lowers success. The non-monotonic budget trend and fixed-budget schedule sensitivity are consistent with the combined effects of trajectory curvature and late-time amplification. Taken with the mechanisms above, these observations identify the sampling schedule as a lever on both the generated action distribution and task success. They motivate a demonstration-grounded criterion for selecting evaluation times at each fixed budget.

We propose Ground Your Steps (GYS), a training-free method that grounds the sampling schedule in the demonstrations, motivated by Drifting (Deng et al., 2026), which trains a generator by driving its generated distribution into agreement with the data distribution. GYS's target is that the chunks the frozen policy generates be distributed as the demonstrated ones at each observation. GYS compares candidate sampling schedules using a *truncated fractional-difference operator*: it applies finite differences of a calibrated, possibly non-integer order along each arm joint's action sequence. This generalizes first-order differences between consecutive actions and second-order differences between those changes, using a fixed window of action steps. One demonstrated chunk per observation still cannot estimate the demonstrated distribution, so GYS instead compares two types of *pairwise differences*: generated–demonstrated differences and generated–generated differences, formed at the same observation using independent noise draws. Both are transformed by the fractional-difference operator. Under the stated identifiability assumptions, their distributions coincide exactly when the transformed action distributions match, so driving them into agreement realizes this target (Section 4.2). GYS optimizes the sampling schedule offline for each task by minimizing an energy distance between the two transformed pairwise-difference distributions (Figure 1,

panel (c)), and the optimized schedule replaces the uniform one. At the same budget, GYS brings the generated distribution closer to the demonstrated one and raises the task success rate.

We validate GYS on a simulation benchmark and on a real robot. On 50 RoboTwin 2.0 tasks (Chen et al., 2026a) at 3 to 10 NFEs, GYS raises the mean closed-loop success rate from 67.6% to 70.3% for $\pi_{0.5}$ and from 49.3% to 51.9% for GR00T N1.7 (Figure 1, panel (d)). On three real-world manipulation tasks at 3 and 4 NFEs, it raises the success rate from 67.7% to 69.5% for $\pi_{0.5}$ and from 63.5% to 66.5% for GR00T N1.7. Across nearly every fixed budget, GYS raises the task success rate, and the gain comes from the optimized sampling schedule alone, with the weights and the inference cost untouched.

2 Related Work

Flow matching in robot policies. Flow matching has become an increasingly common approach to action generation in robot policies, appearing in generalist VLAs (Black et al., 2025b; Shukor et al., 2025; Yu et al., 2026; Wu et al., 2026b) and task-specific visuomotor policies such as Flow-Policy (Zhang et al., 2025) and CoLA-Flow Policy (Wu et al., 2026a). These flow-matching approaches generate actions by transporting noise through learned flows. This shared flow-matching formulation makes the distribution mismatch discussed in Section 1 a concern across the family of flow-matching robot policies. We therefore study sampling schedule optimization to answer the following question: Where along flow time should a flow-matching sampler spend its evaluations to reduce this distribution mismatch and improve the task success rate?

Training-based correction of distribution mismatch in flow matching. Curvature in the marginal flow can lead to distribution mismatch when its trajectories are approximated by finite integration steps. Training-based methods address this source of mismatch by changing the velocity quantity predicted by the model. One line explicitly straightens the flow. Rectified flow retrains on the pairs its own flow produces, and each round leaves the flow straighter (Liu et al., 2023). Finding that recursion costly, consistency flow matching straightens within a single round, constraining the velocities along a trajectory to a single predicted endpoint (Yang et al., 2024). A second line regresses the average velocity over an interval, accounting for displacement along curved trajectories (Geng et al., 2025). These methods modify the learned dynamics to ease numerical integration. We instead keep the trained policy fixed and reduce distribution mismatch through the sampling schedule alone. To account for both trajectory curvature and late-time sensitivity, we select the schedule by distributional agreement with demonstrations, thereby improving the task success rate.

Optimizing the sampling schedule at inference. Trained VLAs deploy the default uniform grid as their sampling schedule at inference (Black et al., 2025b; NVIDIA et al., 2025). To address trajectory curvature, several image-generation methods optimize sampling schedules without retraining the generative model, using a finely resolved reference trajectory or an error surrogate to guide step placement. These include LD3 (Tong et al., 2025), Align Your Steps (Sabour et al., 2024), and optimized time steps (Xue et al., 2024) for diffusion models, and SharpEuler (Gupta et al., 2026) for flow models. For a frozen robot policy, however, this objective is insufficient because small deviations from the marginal flow can be amplified within the Lipschitz window. Most related to our work, Dense-Jump (Chen et al., 2026b) traces the same late-time degradation to a velocity field whose Lipschitz constant diverges near the action end. It therefore brings a non-uniform sampling schedule to state-based locomotion and dexterous control, stepping densely up to a jump point and then crossing the remaining flow times in a single step. That one schedule serves every task and requires retraining the policy under a U-shaped time distribution. We instead optimize a sampling schedule for each task and budget with the policy frozen, using a demonstration-grounded energy-distance objective for distribution matching. GYS raises the task success rate at nearly every fixed NFE budget. However, our approach only reduces the effect of amplified small deviations from the marginal flow without eliminating the Lipschitz window.

3 Background

Conditional flow matching. Given an observation condition o, a flow matching policy (Lipman et al., 2023) transforms Gaussian noise into an action chunk of horizon H and action dimension d,

represented as an element of $\mathbb{R}^{H\times d}$. For a VLA policy, the observation is encoded by its vision–language backbone to condition the action expert. Section 4 introduces a summary $\xi(o)$ of this representation for distribution comparison. Expert demonstrations induce a conditional distribution q_o over such chunks. For a demonstrated chunk $Y \sim q_o$ and an independent noise sample $Z \sim \mathcal{N}(0, I)$ of the same shape, rectified flow (Liu et al., 2023) defines the linear interpolation

$$X_\tau = (1-\tau)\, Z + \tau\, Y,$$

which moves from Z at $\tau = 0$ to Y at $\tau = 1$ with constant velocity $Y - Z$. A neural velocity field $v_\theta(x, \tau, o)$ is trained to predict this velocity along the interpolation by minimizing

$$\mathcal{L}(\theta) = \mathbb{E}_o\, \mathbb{E}_{Y\sim q_o,\ Z\sim\mathcal{N}(0,I),\ \tau} \big\| v_\theta(X_\tau, \tau, o) - (Y - Z) \big\|^2. \tag{1}$$

Here, o is drawn from the demonstrations, while τ is drawn independently of (Y, Z) from the policy's training-time distribution on $[0, 1]$. At a fixed (x, τ, o), different pairs (Y, Z) may induce interpolation paths that pass through the same state x, with different velocities $Y - Z$. Under squared loss, the population-optimal prediction is therefore their conditional mean, known as the marginal velocity field,

$$v^*(x, \tau, o) = \mathbb{E}\big[\, Y - Z \;\big|\; X_\tau = x,\ \tau,\ o \,\big]. \tag{2}$$

Equivalently, Equation (1) differs from $\mathbb{E}\|v_\theta - v^*\|^2$ only by a term independent of θ, so training on the sampled velocities $Y - Z$ fits the marginal field v^*.

Under standard regularity conditions, the exact marginal flow follows the marginal distributions of the linear interpolant (Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023). Let $X^*(\tau)$ denote this flow initialized at $X^*(0) \sim \mathcal{N}(0, I)$. For almost every o,

$$\mathrm{Law}\big(X^*(\tau) \mid o\big) = \mathrm{Law}\big(X_\tau \mid o\big), \qquad \tau \in [0, 1).$$

As τ approaches 1, its distribution therefore converges to the demonstrated action distribution q_o.

Inference as numerical integration. At inference, the policy draws a fresh noise sample and integrates the learned field according to

$$\frac{\mathrm{d}X(\tau)}{\mathrm{d}\tau} = v_\theta\big(X(\tau), \tau, o\big), \qquad X(0) = Z \sim \mathcal{N}(0, I), \tag{3}$$

and executes the endpoint $X(1)$ as an action chunk. In practice, the ODE is solved under a budget of B neural function evaluations. We represent a sampling schedule by an integration grid $\boldsymbol{\tau} = (\tau_0, \tau_1, \ldots, \tau_B)$ with $0 = \tau_0 < \tau_1 < \cdots < \tau_B = 1$, and let $\mathcal{T}_B$ denote the set of such schedules. On any $\boldsymbol{\tau} \in \mathcal{T}_B$, explicit Euler discretizes Equation (3) using numerical states $X(\tau_k)$ satisfying

$$X(\tau_k) = X(\tau_{k-1}) + (\tau_k - \tau_{k-1})\, v_\theta\big(X(\tau_{k-1}), \tau_{k-1}, o\big), \qquad k = 1, \ldots, B, \tag{4}$$

thereby defining a generator $\Psi_\theta^{\boldsymbol{\tau}}(Z; o) = X(\tau_B)$ and its conditional output law $p_o^{\boldsymbol{\tau}}$. Trained VLAs default to the uniform schedule $\tau_k = k/B$.[1] For fixed parameters θ and budget B, schedules in $\mathcal{T}_B$ therefore define a family of generators with the same NFE budget, $\{\Psi_\theta^{\boldsymbol{\tau}} : \boldsymbol{\tau} \in \mathcal{T}_B\}$, which differ only in their placement of integration steps along flow time.

4 METHOD

GYS selects a sampling schedule from demonstrations with the policy weights and NFE budget held fixed. We first define a distribution-matching target, then express it through an energy-distance objective, and finally optimize an empirical version of this objective offline.

4.1 THE DISTRIBUTION-MATCHING TARGET

Let $\{(o_m, y_m)\}_{m=1}^M$ be a calibration set of demonstration pairs, where o_m is an observation and y_m its recorded action chunk. The flow-matching action expert receives a representation of o_m from the vision–language backbone: a prefix key–value cache in $\pi_{0.5}$ and backbone features in GR00T N1.7.

[1] The released $\pi_{0.5}$ sampler integrates from $\tau = 1$ (noise) to $\tau = 0$ (action); we write every grid from 0 (noise) to 1 (action). By default $\pi_{0.5}$ uses $B = 10$ and GR00T N1.7 uses $B = 4$.

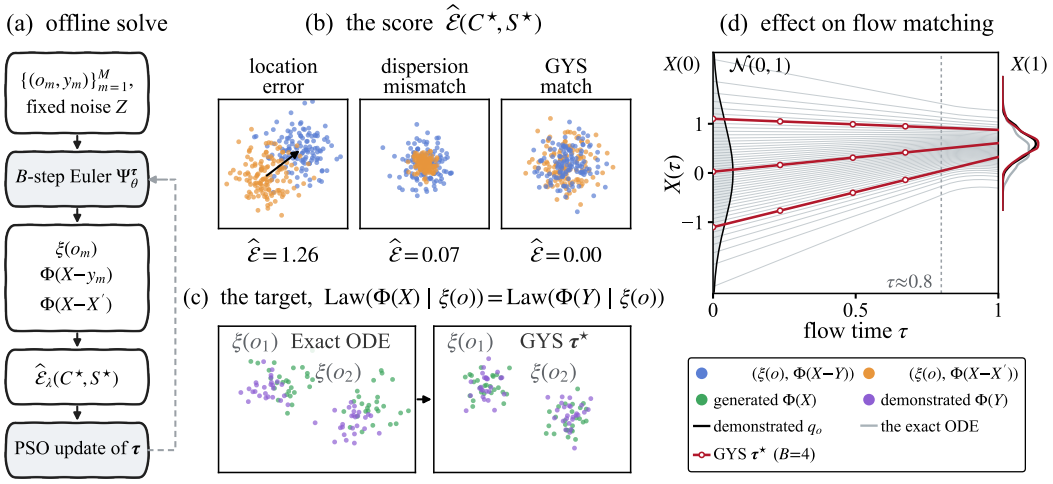


Figure 2: Distributional schedule selection in GYS. *(a)* With policy weights and the NFE budget fixed, GYS searches grids offline using calibration observations and shared noise draws. *(b)* An energy score compares both types of transformed pairwise differences paired with observation summaries, revealing location and dispersion mismatch. *(c)* Aligning these joint laws targets conditional agreement between transformed generated and demonstrated action distributions, illustrated for two observation groups. *(d)* The schematic shows how stepping across a late-time sensitive region can reduce endpoint mismatch relative to exact integration of the learned ODE.

For distribution comparison, we summarize the backbone hidden states by their mean over non-padding tokens and denote this summary by $\xi(o) \in \mathbb{R}^{D_h}$ (Equation (28)). The policy continues to use its full conditioning representation. For the analysis below, we assume that the generated and demonstrated conditional laws depend on o only through $\xi(o)$. This is a sufficiency assumption on the summary, not a property guaranteed by mean pooling.

For fixed policy weights θ and NFE budget B, each schedule $\boldsymbol{\tau} \in \mathcal{T}_B$ defines a generator $\Psi_\theta^{\boldsymbol{\tau}}$. Let $p_\xi^{\boldsymbol{\tau}}$ and q_ξ denote the generated and demonstrated action-chunk laws conditional on ξ. Our ideal target is equality of these laws,

$$p_\xi^{\boldsymbol{\tau}} = q_\xi \qquad \text{for almost every } \xi. \tag{5}$$

We compare candidate sampling schedules using a linear *truncated fractional-difference operator* $\Phi_\nu : \mathbb{R}^{H \times d} \to \mathbb{R}^F$ (Hosking, 1981; Podlubny, 1999). It selects the arm-joint coordinates, applies fractional differencing of order ν along each joint's action sequence, and vectorizes the valid output positions. Each output uses a window of $K = 10$ action steps. For a joint sequence $u[h]$, order one gives $u[h] - u[h-1]$; order two gives $u[h] - 2u[h-1] + u[h-2]$. Non-integer orders extend these finite differences through generalized binomial weights. Positive orders can reduce the contribution of offsets shared across action steps and emphasize changes within the chunk; negative orders instead perform fractional summation. The operation acts along the action horizon, not along flow time. Appendix B.1 gives the weights and truncation convention.

We compare schedules using the resulting transformed target,

$$\mathrm{Law}\big(\Phi_\nu(X) \mid \xi\big) = \mathrm{Law}\big(\Phi_\nu(Y) \mid \xi\big) \qquad \text{for almost every } \xi, \tag{6}$$

where $X \sim p_\xi^{\boldsymbol{\tau}}$ and $Y \sim q_\xi$. Equality of the full laws implies Equation (6), but the converse need not hold because the operator discards some action coordinates and horizon information. The differencing order ν is calibrated once for each task and budget (Section 4.3). To evaluate this target, we must account for a statistical constraint: the calibration set contains only one demonstrated chunk per observation, so it does not directly provide the conditional demonstrated law.

4.2 An energy-distance objective from pairwise-difference distributions

At a fixed observation, the transformed target makes generated and demonstrated chunks equal in distribution after the fractional-difference operator. Consider the pairwise difference between a

generated chunk and an independent demonstration. Replacing that demonstration with another independent generated chunk therefore leaves the transformed pairwise-difference distribution unchanged. We use this property to compare generated–demonstrated differences with generated–generated differences.

Fix a schedule $\boldsymbol{\tau}$, a differencing order ν, and a condition ξ. Let $Y \sim q_\xi$ and let $X, X' \sim p_\xi^{\boldsymbol{\tau}}$ be mutually independent conditional on ξ. We form the *generated–demonstrated pairwise difference* $X - Y$ and the *generated–generated pairwise difference* $X - X'$. Because the fractional-difference operator is linear, transforming each pairwise difference is equivalent to subtracting the transformed chunks. Define their conditional laws by

$$C_\xi = \mathrm{Law}\big(\Phi_\nu(X - Y) \mid \xi\big), \qquad S_\xi = \mathrm{Law}\big(\Phi_\nu(X - X') \mid \xi\big). \tag{7}$$

The replacement argument gives $C_\xi = S_\xi$ under the transformed target Equation (6).

The calibration set does not provide enough demonstrations to estimate these laws separately at every observation, so we pool samples across the task. Different observations, however, can require different actions. Pooling the pairwise differences alone discards their association with the observations and can conceal differences between the conditional laws. We therefore retain the observation summary in each sample and compare the resulting joint laws,

$$c = \big(\xi(o),\ \Phi_\nu(X - Y)\big), \qquad s = \big(\xi(o),\ \Phi_\nu(X - X')\big), \tag{8}$$

where (o, Y) is a demonstration pair and X, X' are generated at o with independent noise. Let C^* and S^* denote the population laws of c and s. Both use the same distribution of observations, so they share the same ξ-marginal. Consequently, $C^* = S^*$ is equivalent to $C_\xi = S_\xi$ for almost every ξ (Appendix A.2).

For a joint sample $(\xi, \phi) \in \mathbb{R}^{D_h + F}$, where ϕ is a transformed pairwise difference, define the weighted Euclidean metric

$$\mathcal{D}_\lambda\big((\xi, \phi), (\xi', \phi')\big) = \sqrt{\lambda^2 \, \|\xi - \xi'\|^2 + \|\phi - \phi'\|^2}, \tag{9}$$

where $\lambda > 0$ sets the relative scale of the observation coordinates. We use this metric to define the energy distance between C^* and S^* (Székely & Rizzo, 2013). It compares distances between samples from different distributions with distances between samples from the same distribution. For mutually independent draws $c, \tilde{c} \sim C^*$ and $s, \tilde{s} \sim S^*$,

$$\mathcal{E}_\lambda\big(C^*, S^*\big) = 2\,\mathbb{E}\,\mathcal{D}_\lambda(c, s) - \mathbb{E}\,\mathcal{D}_\lambda(c, \tilde{c}) - \mathbb{E}\,\mathcal{D}_\lambda(s, \tilde{s}). \tag{10}$$

Assumption 1. Fix a sampling schedule $\boldsymbol{\tau}$ and a differencing order ν. The conditional generated and demonstrated laws depend on o only through $\xi(o)$. The joint laws C^* and S^* have finite first moments. For almost every value of ξ, the characteristic function of $\mathrm{Law}(\Phi_\nu(X) \mid \xi)$ has no open set of zeros.

The final condition permits recovery of the transformed action law from its pairwise-difference law; it is not guaranteed by Gaussian input noise alone.

Proposition 1. *Under Assumption 1, for every $\lambda > 0$,*

$$\mathcal{E}_\lambda(C^*, S^*) = 0 \quad \Longleftrightarrow \quad \mathrm{Law}\big(\Phi_\nu(X) \mid \xi\big) = \mathrm{Law}\big(\Phi_\nu(Y) \mid \xi\big) \quad \textit{for almost every } \xi.$$

Appendix A gives the proof. The positive observation weight preserves the conditional comparison; setting $\lambda = 0$ retains only the pooled pairwise-difference marginals (Appendix A.2). The proposition concerns equality of population distributions and does not bound distribution error or task success in terms of an empirical score. We use the empirical score to compare sampling schedules and evaluate the selected schedules in closed loop in Section 5.

4.3 Offline schedule optimization

For each task and NFE budget, GYS optimizes the sampling schedule using a fixed calibration set and frozen policy weights, without environment interaction. We parameterize the schedule by its step sizes $h = (h_1, \ldots, h_B)$, with nodes $\tau_k = \sum_{j=1}^{k} h_j$, and search over

$$\mathcal{H}_B = \Big\{ h \in \mathbb{R}^B \;\Big|\; \textstyle\sum_{k=1}^{B} h_k = 1,\ \ h_k \geq \varepsilon_h \Big\}, \qquad \varepsilon_h = 10^{-3}, \tag{11}$$

This leaves $B-1$ free parameters while ensuring positive step sizes.

For each observation o_m, draw an even number $N \geq 4$ of independent Gaussian noise samples and reuse them for every schedule. For a candidate $\boldsymbol{\tau}$, let $X_{m,i} = \Psi_\theta^{\boldsymbol{\tau}}(Z_{m,i}; o_m)$ denote the resulting chunks. Split them into two disjoint groups of size $N_c = N/2$. The first group supplies generated–demonstrated differences against y_m; the second supplies generated–generated differences through cyclic pairing. The joint samples are

$$\begin{aligned} c_{m,i} &= \big(\xi(o_m),\ \Phi_\nu(X_{m,i} - y_m)\big), && i = 1, \ldots, N_c, \\ s_{m,j} &= \big(\xi(o_m),\ \Phi_\nu(X_{m,j} - X_{m,j+1})\big), && j = N_c{+}1, \ldots, N, \end{aligned} \tag{12}$$

with cyclic index $X_{m,N+1} = X_{m,N_c+1}$. Samples from all calibration observations are pooled, retaining the observation summary in each sample.

We use this construction on a set of initial sampling schedules to calibrate the comparison. We select a common differencing order $\nu^\star$ by balancing the spreads of the two pairwise-difference distributions on these schedules, and set λ by matching the typical scales of the transformed pairwise differences and standardized observation summaries. Both parameters and the observation standardization remain fixed throughout the search (Appendix B.2).

We then score each candidate schedule by the energy distance between the empirical joint distributions of Equation (12), using standardized observation coordinates in Equation (9). Including all diagonal terms, the score is

$$\begin{aligned} \widehat{\mathcal{E}}_\lambda(\nu;\, \boldsymbol{\tau}) = \frac{2}{n_c\, n_s} \sum_{m,i} \sum_{m',j} \mathcal{D}_\lambda\big(c_{m,i}, s_{m',j}\big) - \frac{1}{n_c^2} \sum_{m,i} \sum_{m',i'} \mathcal{D}_\lambda\big(c_{m,i}, c_{m',i'}\big) \\ - \frac{1}{n_s^2} \sum_{m,j} \sum_{m',j'} \mathcal{D}_\lambda\big(s_{m,j}, s_{m',j'}\big), \qquad n_c = MN_c, \quad n_s = M(N - N_c). \end{aligned} \tag{13}$$

At the calibrated order $\nu^\star$, we solve

$$L(h) = \widehat{\mathcal{E}}_\lambda\big(\nu^\star; \boldsymbol{\tau}(h)\big), \qquad \min_{h \in \mathcal{H}_B} L(h). \tag{14}$$

Reusing the calibration pairs and noise samples makes this a deterministic comparison of schedules under the same metric.

We search the feasible set with particle swarm optimization (PSO; Clerc & Kennedy, 2002), which does not require differentiating through the policy. The particles start at the initial schedules, and each update is projected onto $\mathcal{H}_B$. We return the lowest-scoring schedule among all candidates evaluated, $\boldsymbol{\tau}^\star$; Appendix B.3 gives the full procedure and computational cost. At deployment, $\boldsymbol{\tau}^\star$ replaces the uniform schedule at the same NFE budget, with no additional policy evaluations.

5 Experiments

Experiment setup. We evaluate GYS on a simulation benchmark and on a real robot. Simulated evaluation runs on RoboTwin 2.0 (Chen et al., 2026a) with its dual-arm Aloha-AgileX embodiment and 50 manipulation tasks. Real-world evaluation runs on a ROKAE AR5 dual-arm platform with three tasks of increasing difficulty, pulling a drawer and picking cubes (PaP), sorting cubes by color (SbC), and stacking cubes and covering them (SaC). Figure 3 shows the three real-world tasks with their full prompts. We fine-tune $\pi_{0.5}$ and GR00T N1.7 by co-training on the official clean datasets for RoboTwin 2.0 and on teleoperated demonstrations for the real robot; Appendix C.2 gives the collection and training details. Each RoboTwin 2.0 task–budget cell is evaluated for 100 episodes, and we report the mean success rate. Within a task every method runs under one inference configuration, sharing the prompt, the list of scene seeds, and the sequence of initial-noise seeds bound to each scene seed, so that corresponding chunks are generated from the same noise as far as possible. We compare GYS at budgets $B \in \{3, 4, 5, 6, 8, 10\}$ against the uniform schedule, the released default of both policies; SharpEuler, whose schedule uses finer steps where the flow trajectory bends fastest, adapted to the action expert; and Dense-Jump, whose schedule uses $B-1$ uniform Euler steps on $[0, 0.5]$ and crosses the rest in a single terminal jump. Dense-Jump is the only method evaluated on its own checkpoints. Dense-Jump fine-tunes each policy with flow time



Figure 3: Visualization of the three real-world manipulation tasks. Each row shows one task as a sequence of frames ordered in time, from the initial scene (left) to task completion (right). The task prompts are, for PaP, "Pull the drawer open and place the yellow cube inside"; for SbC, "Place the green, blue, and yellow cubes into the regions of matching color"; and for SaC, "First stack the blue cube and the green cube in the designated region, then cover them with the upside-down cup".

drawn from a U-shaped Beta distribution instead of the policy's default[2]. Appendix C.3 lists their sampling schedules. On the real robot we compare GYS at $B \in \{3, 4\}$ against uniform only, with gridded boards fixing the object placements so every method runs the same 100-episode benchmark.

Main results. Table 1 addresses our central question: can an offline schedule selected from demonstrations improve the closed-loop behavior of a fixed flow-matching VLA under the same inference budget? Across 50 RoboTwin 2.0 tasks, two VLA policies, and six NFE budgets, GYS consistently shifts performance upward relative to the uniform default. It improves mean success by 2.7 percentage points for $\pi_{0.5}$ and 2.6 percentage points for GR00T N1.7, with gains at almost every fixed budget. The improvement across architectures and NFE regimes shows that the uniform discretization leaves meaningful task performance unrealized, and that demonstration-grounded schedule selection provides a more effective allocation of the available inference steps.

Because GYS changes only the sampling schedule, its gains require neither policy retraining nor additional deployment-time computation at a fixed budget. More importantly, the optimized schedules improve the success–latency trade-off: for $\pi_{0.5}$, GYS at $B{=}4$ and $B{=}6$ outperforms the 10-step uniform sampler while reducing per-chunk latency by 35% and 24%, respectively. The same principle transfers beyond simulation. On the real robot, GYS improves the three-task average for both policies and both tested budgets, with gains of 1.7–3.3 percentage points (Table 2). Thus, schedule optimization improves the deployment-time success–latency frontier of a fixed VLA without modifying its parameters.

The comparison with alternative schedulers further clarifies what the schedule should optimize. SharpEuler allocates steps according to the geometry of the learned flow, whereas Dense-Jump learns a single dense-then-jump schedule shared across tasks. Neither yields comparable closed-loop success, and GYS achieves the highest average success for both policies. This comparison supports our central design choice: for VLA action experts, the sampling schedule should be selected by how well its outputs agree with task demonstrations, rather than solely by ODE integration accuracy or a task-agnostic allocation of flow time.

Ablation study. Table 3 ablates the two components of the objective and examines sensitivity to the number of generated samples N. For each variant, we re-optimize the sampling schedule at $B{=}3$ and $B{=}4$; Full denotes GYS with the default $N{=}8$. We first ablate fractional differencing, the most consequential component. Setting $\nu{=}0$ retains the same valid horizon crop but compares un-

[2]On the flow-time axis of Section 3, both released implementations draw the time from $\mathrm{Beta}(1.0, 1.5)$, which concentrates its mass at the noise end.

Table 1: Simulation success rate (%) at $B \in \{3, 4, 5, 6, 8, 10\}$. Each score is averaged over the 50 RoboTwin 2.0 tasks at 100 episodes per task. The Avg. column is the mean over the six budgets, and its subscript is the standard deviation across them. The Gain row is GYS minus Uniform. Latency (ms) is the mean per-chunk inference time on one NVIDIA L4, shared across methods at each B.

Policy	Schedule	B=3	B=4	B=5	B=6	B=8	B=10	Avg.
$\pi_{0.5}$	Uniform	67.9	68.4	68.5	67.4	67.1	66.4	$67.6_{\pm 0.7}$
	SharpEuler	67.5	67.4	66.8	66.5	65.0	64.4	$66.3_{\pm 1.2}$
	Dense-Jump	59.1	60.1	59.6	59.1	60.1	59.7	$59.6_{\pm 0.4}$
	GYS (ours)	**71.1**	**71.3**	**70.9**	**70.0**	**69.3**	**69.0**	$\mathbf{70.3}_{\pm 0.9}$
	Gain	+3.2	+2.9	+2.4	+2.6	+2.2	+2.6	+2.7
	Latency (ms)	185	196	215	230	269	302	–
GR00T N1.7	Uniform	48.2	48.9	50.3	**52.6**	47.9	48.2	$49.3_{\pm 1.7}$
	SharpEuler	47.7	46.1	47.6	43.6	43.3	42.0	$45.1_{\pm 2.2}$
	Dense-Jump	44.6	44.3	44.7	44.8	44.2	45.0	$44.6_{\pm 0.3}$
	GYS (ours)	**50.6**	**52.2**	**51.2**	52.4	**52.9**	**52.3**	$\mathbf{51.9}_{\pm 0.8}$
	Gain	+2.4	+3.3	+0.9	-0.2	+5.0	+4.1	+2.6
	Latency (ms)	123	148	158	175	206	242	–

Table 2: Real robot success rate (%) at B=3 & 4 on the shared 100-episode benchmark.

		B=3				B=4			
Policy	Schedule	PaP	SbC	SaC	Avg.	PaP	SbC	SaC	Avg.
$\pi_{0.5}$	Uniform	**78**	71	52	67.0	77	73	55	68.3
	GYS (ours)	77	**74**	**55**	**68.7**	**80**	**75**	**56**	**70.3**
GR00T N1.7	Uniform	72	67	47	62.0	75	69	51	65.0
	GYS (ours)	**74**	**70**	**52**	**65.3**	75	**73**	**55**	**67.7**

Table 3: Ablation success rate (%) at B=3 & 4 on RoboTwin 2.0. The Full column uses N=8.

			GYS Ablations					
Policy	NFEs	Uniform	Full	ν=0	λ=0	N=4	N=16	N=32
$\pi_{0.5}$	3	67.9	71.1	68.1	70.2	71.0	**71.4**	70.6
	4	68.4	**71.3**	66.1	70.5	70.8	71.2	70.9
GR00T N1.7	3	48.2	50.6	38.8	51.0	**51.3**	50.5	50.8
	4	48.9	52.2	46.2	51.1	51.7	**52.4**	51.5

filtered arm-joint sequences, reducing success by 3.0–5.2 percentage points on $\pi_{0.5}$ and by 6.0–11.8 percentage points on GR00T N1.7 relative to Full. At B=3, the resulting GR00T N1.7 schedule also falls 9.4 percentage points below uniform (38.8% vs. 48.2%). We next ablate the observation weight λ. Setting λ=0 does not diminish the empirical effectiveness of GYS: the resulting schedules continue to outperform uniform for both policies and budgets. We nevertheless retain $\lambda > 0$ because the observation block preserves the conditional comparison: under the stated assumptions, the population objective vanishes if and only if the transformed conditional distributions match (Appendix A.2 and Proposition 1). Finally, performance is stable across $N \in \{4, 8, 16, 32\}$, with every variant lying within 0.7 percentage points of Full. This robustness indicates that the default N=8 provides an effective accuracy–cost trade-off for the pooled, task-level objective.

6 Conclusion and Limitations

We discuss two mechanisms affecting the generated distribution in flow-matching policies, trajectory curvature and the Lipschitz window, and introduce Ground Your Steps (GYS), a training-free method for optimizing sampling schedules. GYS directly targets agreement between the generated and demonstrated distributions through a fractional-difference operator. Experiments on RoboTwin 2.0 and a real robot with $\pi_{0.5}$ and GR00T N1.7 show that the optimized schedules improve mean task success at nearly every fixed NFE budget, without additional inference overhead.

Limitations. Although GYS improves distributional agreement after truncated fractional differencing along the action horizon and raises task success, it does not eliminate either underlying mechanism. This is the trade-off for remaining training-free: GYS mitigates their effects through sampling schedules while leaving the learned velocity field unchanged.

AI USE STATEMENT

We used generative AI tools solely for language-related assistance: to improve the grammar, clarity, and phrasing of author-written text. We reviewed all AI-assisted text and checked that it preserved the intended technical meaning. Generative AI tools were not used to develop the research ideas or methods, design or conduct experiments, or analyze or interpret results. The authors take responsibility for the final content of this paper.

REPRODUCIBILITY STATEMENT

We provide the source code associated with this work as supplementary material accompanying our OpenReview submission.

REFERENCES

Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations (ICLR)*, 2023.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 2025a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025b.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026a.

Zidong Chen, Zihao Guo, Peng Wang, ThankGod Itua Egbe, Yan Lyu, and Chenghao Qian. Dense-jump flow matching with non-uniform time scheduling for robotic policies: Mitigating multi-step inference degradation. In *Proceedings of the 2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026b.

Maurice Clerc and James Kennedy. The particle swarm—explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting, 2026. arXiv:2602.04770.

Yuan Gao, Jian Huang, Yuling Jiao, and Shurong Zheng. Convergence of continuous normalizing flows for learning probability distributions, 2024. arXiv:2404.00551.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Aditi Gupta, Soon Hoe Lim, Annan Yu, and N. Benjamin Erichson. Sharpen your flow: Sharpness-aware sampling for flow matching, 2026. arXiv:2605.11547.

J. R. M. Hosking. Fractional differencing. *Biometrika*, 68(1):165–176, 1981.

Kento Kawaharazuka, Jihoon Oh, Jun Yamada, Ingmar Posner, and Yuke Zhu. Vision-language-action models for robotics: A review towards real-world applications. *IEEE Access*, 2025.

Nikita Kornilov, Petr Mokrov, Alexander Gasnikov, and Alexander Korotin. Optimal flow matching: Learning straight trajectories in just one step. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

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

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

Chenrui Ma, Xi Xiao, Tianyang Wang, Xiao Wang, and Yanning Shen. Learning straight flows: Variational flow matching for efficient generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026a.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied AI. *IEEE Transactions on Neural Networks and Learning Systems*, 37(7):3031–3051, 2026b.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots, 2025. arXiv:2503.14734.

Igor Podlubny. *Fractional Differential Equations*, volume 198 of *Mathematics in Science and Engineering*. Academic Press, San Diego, 1999.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your steps: Optimizing sampling schedules in diffusion models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, et al. SmolVLA: A vision-language-action model for affordable and efficient robotics, 2025. arXiv:2506.01844.

Gábor J. Székely and Maria L. Rizzo. Energy statistics: A class of statistics based on distances. *Journal of Statistical Planning and Inference*, 143(8):1249–1272, 2013.

Vinh Tong, Trung-Dung Hoang, Anji Liu, Guy Van den Broeck, and Mathias Niepert. Learning to discretize denoising diffusion ODEs. In *International Conference on Learning Representations (ICLR)*, 2025.

Songwei Wu, Zhiduo Jiang, Wandong Sun, Guanghu Xie, Yuteng Xie, Rui Zhao, Yang Liu, and Hong Liu. CoLA-Flow Policy: Temporally coherent imitation learning via continuous latent action flow matching for robotic manipulation. *IEEE Robotics and Automation Letters*, 11(9): 10329–10336, 2026a.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, Yiyu Ren, Kejia Zhang, Hui Yu, Jingmei Zhao, Shuai Zhou, Zhenqi Qiu, Houlong Xiong, Ziyu Wang, Zechen Wang, Ran Cheng, Yong-Lu Li, Yongtao Huang, Xing Zhu, Yujun Shen, and Kecheng Zheng. A pragmatic VLA foundation model, 2026b. arXiv:2601.18692.

Shuchen Xue, Zhaoqiang Liu, Fei Chen, Shifeng Zhang, Tianyang Hu, Enze Xie, and Zhenguo Li. Accelerating diffusion sampling with optimized time steps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Ling Yang, Zixiang Zhang, Zhilong Zhang, Xingchao Liu, Minkai Xu, Wentao Zhang, Chenlin Meng, Stefano Ermon, and Bin Cui. Consistency flow matching: Defining straight flows with velocity consistency. *arXiv preprint arXiv:2407.02398*, 2024.

Ryan Yu, Pushi Zhang, Starrick Liu, Brae Liu, Miracle Kang, Shalfun Li, Lights Shi, Ellie Ma, Ping Yang, Chris Pan, Jerry Chen, Dongxiu Liu, Rain Sun, Miles Guo, Byron Zhang, Hugo Zhou, Zach Xu, Vincent Chen, Harrison Huang, James Wang, Dance Kuzi, Andy Zhai, Hang Su, Roy Gan, Lucy Liang, Hao Wang, and Qian Wang. Wall-OSS-0.5 technical report, 2026. arXiv:2605.30877.

Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. FlowPolicy: Enabling fast and robust 3D flow-based policy via consistency flow matching for robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14754–14762, 2025.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

A Theoretical analysis

Fix the sampling schedule $\boldsymbol{\tau}$ and the linear fractional-difference operator Φ_ν. Under Assumption 1, the population claim in Proposition 1 follows from three equivalences for every fixed $\lambda > 0$:

$$\begin{aligned} &\mathrm{Law}\big(\Phi_\nu(X) \mid \xi\big) = \mathrm{Law}\big(\Phi_\nu(Y) \mid \xi\big) \\ &\quad\Longleftrightarrow\quad C_\xi = S_\xi \quad\Longleftrightarrow\quad C^* = S^* \quad\Longleftrightarrow\quad \mathcal{E}_\lambda(C^*, S^*) = 0, \end{aligned} \tag{15}$$

The first equivalence holds for almost every condition and is proved by conditional deconvolution in Appendix A.1. The second uses the common condition marginal of the two joint laws (Appendix A.2). The last follows from the characterization of equality by Euclidean energy distance for any fixed $\lambda > 0$ (Székely & Rizzo, 2013). These equivalences characterize when the energy-distance objective vanishes.

A.1 Deconvolution of the pairwise-difference laws

We prove the first link of Equation (15): given ξ,

$$\mathrm{Law}\big(\Phi_\nu(X) \mid \xi\big) = \mathrm{Law}\big(\Phi_\nu(Y) \mid \xi\big) \quad\Longleftrightarrow\quad \mathrm{Law}\big(\Phi_\nu(X - Y) \mid \xi\big) = \mathrm{Law}\big(\Phi_\nu(X - X') \mid \xi\big).$$

All statements are conditional on ξ, in the sense of a regular conditional distribution on $\mathbb{R}^{D_h}$, and hold for almost every value of ξ under Assumption 1.

Setup. Generate X, X' at observation o from independent noise draws, also independent of (o, Y). Since their conditional laws and the demonstration law depend on o only through $\xi(o)$ under Assumption 1, X, X', Y remain mutually independent given ξ, with X and X' identically distributed. This independence is used in the characteristic-function factorizations below. For any action-space random element W, write

$$\varphi_W(\omega) = \mathbb{E}\big[\exp(i\langle\omega, \Phi_\nu(W)\rangle) \mid \xi\big], \qquad \omega \in \mathbb{R}^F,$$

for the conditional characteristic function of its transformed value.

Necessity. Suppose $\mathrm{Law}(\Phi_\nu(X) \mid \xi) = \mathrm{Law}(\Phi_\nu(Y) \mid \xi)$. Then $\Phi_\nu(Y)$ and $\Phi_\nu(X')$ share a law given ξ and are both independent of $\Phi_\nu(X)$. By linearity,

$$\begin{aligned} \Phi_\nu(X - Y) &= \Phi_\nu(X) - \Phi_\nu(Y), \\ \Phi_\nu(X - X') &= \Phi_\nu(X) - \Phi_\nu(X'). \end{aligned}$$

The two transformed pairwise differences therefore share a conditional law: $C_\xi = S_\xi$.

Sufficiency. Suppose $C_\xi = S_\xi$. Conditional independence and linearity of the fractional-difference operator give

$$\begin{aligned}\varphi_{X-Y}(\omega) &= \mathbb{E}\big[\exp(i\langle\omega, \Phi_\nu(X-Y)\rangle) \mid \xi\big] \\ &= \varphi_X(\omega)\,\overline{\varphi_Y(\omega)}. \end{aligned} \tag{16}$$

The same factorization with X' in place of Y yields

$$\varphi_{X-X'}(\omega) = \varphi_X(\omega)\,\overline{\varphi_X(\omega)} = \big|\varphi_X(\omega)\big|^2. \tag{17}$$

The assumed equality of pairwise-difference laws now gives $\varphi_X(\omega)\,\overline{\varphi_Y(\omega)} = \big|\varphi_X(\omega)\big|^2$ for every ω, or equivalently

$$\varphi_X(\omega)\Big(\overline{\varphi_Y(\omega)} - \overline{\varphi_X(\omega)}\Big) = 0 \qquad \text{for every } \omega. \tag{18}$$

Under Assumption 1, the zero set $\{\omega : \varphi_X(\omega) = 0\}$ has empty interior. Its complement is dense, and division by φ_X in Equation (18) gives equality there. Continuity then implies $\varphi_X(\omega) = \varphi_Y(\omega)$ for every ω. A characteristic function determines its law, so $\mathrm{Law}(\Phi_\nu(X) \mid \xi) = \mathrm{Law}(\Phi_\nu(Y) \mid \xi)$. The zero-set condition is an identifiability assumption, not a consequence of Gaussian input noise to an arbitrary learned generator.

A.2 From conditional laws to the energy-distance objective

We prove the second link of Equation (15):

$$\begin{aligned}&\mathrm{Law}\big(\Phi_\nu(X-Y) \mid \xi\big) = \mathrm{Law}\big(\Phi_\nu(X-X') \mid \xi\big) \qquad \text{for almost every } \xi \\ \iff\quad &\mathrm{Law}\big(\xi(o),\, \Phi_\nu(X-Y)\big) = \mathrm{Law}\big(\xi(o),\, \Phi_\nu(X-X')\big).\end{aligned}$$

Setup. By Equation (8), C^* and S^* are the laws of the pairs $c = (\xi(o), \Phi_\nu(X-Y))$ and $s = (\xi(o), \Phi_\nu(X-X'))$ on $\mathbb{R}^{D_h} \times \mathbb{R}^F$. Both joint laws have the same ξ marginal $\mu = \mathrm{Law}(\xi(o))$. For almost every ξ, their conditional pairwise-difference laws are C_ξ and S_ξ, respectively, by Equation (7). Since the marginal is shared, disintegration writes each joint law as

$$\begin{aligned}C^*(A \times B) &= \int_A C_\xi(B)\,\mu(d\xi), \\ S^*(A \times B) &= \int_A S_\xi(B)\,\mu(d\xi),\end{aligned}$$

for measurable $A \subseteq \mathbb{R}^{D_h}$ and $B \subseteq \mathbb{R}^F$.

Necessity. If $C_\xi = S_\xi$ for almost every ξ, the two integrals agree for every A and B. In particular, the joint cumulative distribution functions coincide, so $C^* = S^*$.

Sufficiency. If $C^* = S^*$, the same identities give, for every measurable A and B,

$$\int_A \big[C_\xi(B) - S_\xi(B)\big]\,\mu(d\xi) = 0.$$

For each B, equality for all A implies equality of the two conditional probabilities for μ-almost every ξ. Taking B in a countable determining class for $\mathbb{R}^F$ gives one common null set; uniqueness of probability measures then yields $C_\xi = S_\xi$ for μ-almost every ξ.

For $\lambda > 0$, define the invertible linear map $T_\lambda(\xi, \phi) = (\lambda\xi, \phi)$. Then $\mathcal{D}_\lambda(w, w') = \|T_\lambda w - T_\lambda w'\|$. The finite-first-moment assumption is preserved by T_λ, so Euclidean energy distance is nonnegative and vanishes exactly when the transformed joint laws agree (Székely & Rizzo, 2013). Invertibility of T_λ makes this equivalent to $C^* = S^*$, completing the third link of Equation (15).

At $\lambda = 0$, the metric ignores ξ. A zero objective then identifies only the pooled laws of $\Phi_\nu(X-Y)$ and $\Phi_\nu(X-X')$, not their conditional laws given ξ.

These results characterize exact equality of population distributions after the fractional-difference operator. They do not identify components discarded by the operator, nor do they provide a quantitative bound on conditional distribution error or task success from a small empirical score.

A.3 THE LIPSCHITZ WINDOW OF THE MARGINAL FIELD

We derive the spatial Jacobian of the exact marginal field and identify when it has an expanding direction. We use *Lipschitz window* for a late-time region in which small deviations from the marginal flow can be amplified. A large spatial Lipschitz constant alone does not imply amplification: the sign of the Jacobian's largest eigenvalue matters.

Setup. Fix an observation o and identify each action chunk with a vector in $\mathbb{R}^{Hd}$. Assume $\mathbb{E}[\|Y\|^2 \mid o] < \infty$. A demonstrated chunk $Y \sim q_o$ is paired with an independent noise sample $Z \sim \mathcal{N}(0, I)$ of the same shape, and the straight interpolant

$$X_\tau = (1-\tau)\, Z + \tau\, Y, \qquad \tau \in [0,1], \tag{19}$$

runs from the noise at $\tau = 0$ to the chunk at $\tau = 1$ with constant velocity $Y - Z$. The population minimizer of the conditional flow matching loss is the marginal velocity field

$$v^*(x, \tau, o) = \mathbb{E}\big[\, Y - Z \mid X_\tau = x,\, o \,\big], \tag{20}$$

the conditional average velocity of the interpolants that pass through state x at time τ. A perfectly trained policy attains this field and generates a chunk by integrating it from a fresh noise sample,

$$\frac{\mathrm{d}X(\tau)}{\mathrm{d}\tau} = v^*\big(X(\tau), \tau, o\big), \qquad X(0) = Z \sim \mathcal{N}(0, I). \tag{21}$$

The calculations below concern $0 < \tau < 1$; statements about $\tau = 1$ are limits, since the endpoint field can be singular.

The field and its Jacobian. Along the interpolant equation 19, the noise is recovered from the state as $Z = (X_\tau - \tau Y)/(1-\tau)$, so the interpolant velocity $Y - Z$ that the field regresses onto becomes

$$Y - Z = \frac{Y - X_\tau}{1-\tau},$$

and taking the conditional expectation in equation 20 writes the marginal field as a rescaled posterior mean,

$$v^*(x, \tau, o) = \frac{m(x, \tau, o) - x}{1-\tau}, \qquad m(x, \tau, o) = \mathbb{E}\big[\, Y \mid X_\tau = x,\, o \,\big]. \tag{22}$$

For $0 < \tau < 1$, conditioning on $Y = y$ gives the Gaussian likelihood

$$X_\tau \mid Y = y \;\sim\; \mathcal{N}\big(\tau y,\, (1-\tau)^2 I\big),$$

with density

$$p(x \mid y, \tau) = C(\tau)\, \exp\!\Big(-\tfrac{\|x - \tau y\|^2}{2(1-\tau)^2}\Big), \tag{23}$$

where $C(\tau)$ is independent of y. Bayes' rule weights the demonstration law by this likelihood, whether q_o has a density or is discrete. For fixed $0 < \tau < 1$, the Gaussian factor makes the posterior moments finite and permits differentiation under the integral, locally uniformly in x. The logarithmic derivative of the normalized posterior weight with respect to x is $\tau(y - m)/(1-\tau)^2$. Hence, with $\Sigma(x, \tau, o) = \mathrm{Cov}(Y \mid X_\tau = x, o)$,

$$\frac{\partial m}{\partial x} = \frac{\tau}{(1-\tau)^2}\, \Sigma(x, \tau, o). \tag{24}$$

Combining equation 22 and equation 24 gives the spatial Jacobian (Gao et al., 2024),

$$J(x, \tau, o) := \frac{\partial v^*}{\partial x}(x, \tau, o) = \frac{\tau}{(1-\tau)^3}\, \Sigma(x, \tau, o) \;-\; \frac{1}{1-\tau}\, I. \tag{25}$$

Since J is symmetric, a locally expanding direction exists exactly when

$$\lambda_{\max}(J) > 0 \quad \Longleftrightarrow \quad \tau\, \lambda_{\max}(\Sigma(x, \tau, o)) > (1-\tau)^2. \tag{26}$$

For multimodal demonstration laws, uncertainty between modes can keep the posterior covariance large near mode boundaries. Wherever Equation (26) holds, the marginal field has a locally expanding direction.

B IMPLEMENTATION DETAILS

B.1 TRUNCATED FRACTIONAL DIFFERENCING ALONG THE ACTION HORIZON

For an action chunk $X \in \mathbb{R}^{H\times d}$, let $u \in \mathbb{R}^{H\times d_a}$ contain its arm-joint coordinates after omitting the two gripper columns, with $d_a = 12$. The operator in Section 4.1 applies the truncated binomial expansion of the backward-difference operator to each joint (Hosking, 1981; Podlubny, 1999):

$$\Phi_\nu(X) = \mathrm{vec}\big(\Delta_K^\nu u\big), \qquad (\Delta_K^\nu u)[h] = \sum_{k=0}^{K-1} (-1)^k \binom{\nu}{k} u[h-k]. \tag{27}$$

Here $\binom{\nu}{0} = 1$ and $\binom{\nu}{k} = \prod_{j=0}^{k-1}(\nu - j)/k!$ for $k \geq 1$. We use $K = 10$ and retain only the valid positions $h = K, \ldots, H$, so the output dimension is $F = (H - K + 1)d_a$. At $\nu = 0$, the operator returns the unfiltered arm-joint values on these same positions. Thus the zero-order ablation changes the differencing operation while preserving the horizon crop. Negative orders use the same finite sum and correspond to truncated fractional summation. For non-integer orders, the truncated coefficients need not sum to zero, so constant offsets need not be removed exactly. The order is calibrated as described next.

B.2 IMPLEMENTATION OF THE OBJECTIVE

The empirical objective in Section 4.3 requires observation summaries, a common differencing order, and an observation weight. We compute these quantities from the calibration set and initial candidate schedules, then hold them fixed during optimization.

Observation summary. Let $\mathcal{V}(o)$ be the set of non-padding token positions and $h_t^{(L)}(o) \in \mathbb{R}^{D_h}$ the final-layer backbone state at position t. The observation summary is

$$\xi(o) = \frac{1}{|\mathcal{V}(o)|} \sum_{t\in\mathcal{V}(o)} h_t^{(L)}(o). \tag{28}$$

These are the prefix states in $\pi_{0.5}$ and the backbone features attended to by the action head in GR00T N1.7. We standardize each coordinate across the M calibration observations, using the resulting means and scales for every candidate schedule. Write the standardized summaries as $\tilde{\xi}_m$. The policy itself continues to use its full conditioning representation.

Differencing order. Let $h_p^{(0)}$, $p = 1, \ldots, P$, be the initial candidate step-size vectors specified in Appendix B.3. For each candidate, construct the pairwise differences as in Equation (12), pooling them across calibration observations. At order ν, denote their transformed coordinates by $\phi_{p,i}^c(\nu)$ and $\phi_{p,j}^s(\nu)$, with respective sample counts n_c and n_s. To calibrate the relative spread of the two types of differences, compute the difference in their mean pairwise distances:

$$\begin{aligned}\varepsilon_p(\nu) = \frac{1}{n_c^2}\sum_{i=1}^{n_c}\sum_{i'=1}^{n_c}\big\|\phi_{p,i}^c(\nu) - \phi_{p,i'}^c(\nu)\big\| \\ - \frac{1}{n_s^2}\sum_{j=1}^{n_s}\sum_{j'=1}^{n_s}\big\|\phi_{p,j}^s(\nu) - \phi_{p,j'}^s(\nu)\big\|.\end{aligned} \tag{29}$$

For each p, evaluate ε_p at 121 equally spaced orders in $[-1, 2]$, in increasing order. If a grid value is exactly zero, choose the first such order as r_p. Otherwise, obtain r_p by linear interpolation across the first pair of adjacent grid values with opposite signs. If neither case occurs, calibration fails and the search is not run. The common order is the arithmetic mean $\nu^\star = P^{-1}\sum_{p=1}^P r_p$. This aggregation defines one comparison operator for the search; it need not make $\varepsilon_p(\nu^\star)$ zero for every candidate.

Observation weight. Apply the operator at the common order $\nu^\star$ to the cached chunks from all P initial candidates. Let $\{\phi_a\}$ pool both types of transformed pairwise differences across these candidates and observations. We match the scales of the two coordinate blocks by setting

$$\lambda = \frac{\mathrm{median}_{a\neq b}\|\phi_a - \phi_b\|}{\mathrm{median}_{m\neq m'}\|\tilde{\xi}_m - \tilde{\xi}_{m'}\|}. \tag{30}$$

The joint samples used in Equation (13) contain these standardized observation coordinates. The order, weight, and standardization statistics remain fixed for all subsequent objective evaluations.

B.3 Offline schedule optimization and its cost

Initialization. We initialize $P = 24$ candidate schedules with the uniform schedule, four fixed nonuniform schedules, and $P - 5$ random schedules. The four fixed schedules have nodes $\tau_k = g_\kappa(k/B)$, where $g_\kappa(s) = (e^{\kappa s} - 1)/(e^\kappa - 1)$ and $\kappa \in \{-2, -1, 1, 2\}$. Positive κ concentrates nodes near the noise endpoint, and negative κ concentrates them near the action endpoint. For each random schedule, we draw its step-size vector from $\mathrm{Dirichlet}(1, \ldots, 1)$. We project the initial step-size vectors onto $\mathcal{H}_B$. All candidate evaluations use the same MN independent Gaussian noise draws and the same calibration observations; the random seed is 0. Algorithm 1 gives the calibration and search procedure.

Algorithm 1 GYS schedule optimization (one task, budget B)

Require: calibration pairs $\{(o_m, y_m)\}_{m=1}^M$; frozen field v_θ; NFE budget B; seed count P; PSO iterations R

draw and fix N noise samples $\{z_{m,i}\}$ for every o_m
$\xi(o_m) \leftarrow$ mean of backbone hidden states over non-padding tokens ▷ Equation (28)
$\mathcal{S} \leftarrow \{\text{uniform schedule},\ \text{four geometric anchors},\ P{-}5 \text{ Dirichlet schedules}\}$ projected onto $\mathcal{H}_B$ ▷ seed round
for each seed schedule $\boldsymbol{\tau} \in \mathcal{S}$ **do**
    generate chunks via Equation (4); form both types of pairwise differences ▷ Equation (12)
    $r_{\boldsymbol{\tau}} \leftarrow$ root of the mean pairwise-distance gap ▷ Equation (29)
end for
$\nu^\star \leftarrow \mathrm{mean}_{\boldsymbol{\tau}}\, r_{\boldsymbol{\tau}}$; $\lambda \leftarrow$ median heuristic at $\nu^\star$ ▷ Equation (30)
$L(\boldsymbol{\tau}) \leftarrow \widehat{\mathcal{E}}_\lambda(\nu^\star; \boldsymbol{\tau})$ for all $\boldsymbol{\tau} \in \mathcal{S}$ ▷ empirical score, Equation (14)
place one particle at each seed schedule; run PSO for R iterations, projecting each particle onto $\mathcal{H}_B$ ▷ Appendix B.3
return $\boldsymbol{\tau}^\star = \arg\min L$ over all schedules evaluated

Particle swarm. Each initial candidate supplies one particle. We use Clerc–Kennedy constriction with $\chi = 0.7298$ and $c_1 = c_2 = 2.05$, and a ring neighborhood (Clerc & Kennedy, 2002). Each update enforces the minimum-step constraints in $\mathcal{H}_B$ and places the sampling nodes on a 10^{-3} lattice. The order, observation weight, and noise draws are not updated. We run $R = 3$ iterations and return the lowest-scoring candidate among all initial and subsequent evaluations. Thus the search evaluates $P(1 + R) = 96$ candidates, counting repeated schedules. Table 4 reports the empirical objective reductions.

Table 4: Empirical objective reductions (%) on GR00T N1.7, averaged across 50 RoboTwin 2.0 tasks. All reductions are normalized by the uniform-schedule objective.

Budget B	3	4	5	6	8	10
Initial candidates (%)	30.0	57.6	16.9	18.3	55.7	33.4
Additional PSO (%)	9.9	3.2	12.3	5.9	1.4	3.1
Total (%)	39.9	60.7	29.1	24.3	57.1	36.5

Cost and deployment. We use $M = 200$ calibration pairs and $N = 8$ noise samples per pair. Evaluating a previously unseen schedule generates MN chunks at B NFEs each. The 96 candidate evaluations therefore require at most $96MNB$ action-expert evaluations, counted per chunk. Repeated schedules reuse cached chunks, and calibration of $\nu^\star$ and λ uses the initial candidates' cached chunks without additional policy evaluations.

Because the calibration observations are fixed, the vision–language backbone runs once per observation. We cache its prefix key–value states in $\pi_{0.5}$ and its conditioning features in GR00T N1.7, so

subsequent flow steps evaluate only the action expert. The search runs on a single GPU with a micro-batch of 512 on an A100-80 GB or 32 on an L4-23 GB. A micro-batch counts the observation–noise pairs processed in one forward pass. Candidate evaluations and their cached chunks use the same fixed inference configuration.

Table 5 reports offline optimization time for $\pi_{0.5}$. The estimates use the measured time for one Euler step over all MN chunks: approximately 1.5 seconds on the A100 and 5.6 seconds on the L4. This integration cost grows with B. Model loading, observation encoding, and parameter calibration add approximately eight and eleven minutes, respectively.

Table 5: Offline optimization time for $\pi_{0.5}$, in minutes for one task and NFE budget on a single GPU. Entries at $B = 8$ and $B = 10$ are measured medians from 17 and 16 runs on the A100, and 30 and 34 runs on the L4, respectively. Other entries are estimates based on the measured per-step time and initialization overhead.

GPU	Micro-batch	Budget B					
		3	4	5	6	8	10
A100-80 GB	512	15	18	20	23	28	32
L4-23 GB	32	38	47	56	65	83	101

At deployment, only the sampling schedule changes. The policy weights and the NFE budget remain fixed, and no calibration or schedule optimization is performed online.

C Experimental details

C.1 Schedule family for Figure 1(a)

At $B = 5$, Figure 1(a) compares five schedules from the one-parameter family

$$\tau_k = (k/B)^{\gamma(s)}, \qquad \gamma(s) = 3^{\,1-2s}, \qquad k = 0, \dots, B, \tag{31}$$

where $s \in \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$. The choice $s = \frac{1}{2}$ gives the uniform grid; $s = 0$ places more nodes near the noise endpoint, while $s = 1$ places more near the action endpoint.

Table 6: The five sampling grids at $B = 5$ in Figure 1(a).

s	$\gamma(s)$	Nodes $\tau_0, \dots, \tau_5$
0	3	$0,\ 0.008,\ 0.064,\ 0.216,\ 0.512,\ 1$
$1/4$	1.732	$0,\ 0.062,\ 0.205,\ 0.413,\ 0.679,\ 1$
$1/2$	1	$0,\ 0.200,\ 0.400,\ 0.600,\ 0.800,\ 1$
$3/4$	0.577	$0,\ 0.395,\ 0.589,\ 0.745,\ 0.879,\ 1$
1	0.333	$0,\ 0.585,\ 0.737,\ 0.843,\ 0.928,\ 1$

C.2 Setup details

RoboTwin 2.0 configuration. Both policies are fine-tuned on the official RoboTwin 2.0 clean datasets, 50 demonstration episodes for each of the 50 tasks, on the dual-arm Aloha-AgileX embodiment. Observations comprise a head camera and two wrist cameras, and actions live in the 14-dimensional absolute joint space of the two arms.

Real-world configuration. Real-world demonstrations are collected on the ROKAE AR5 dual-arm platform of Figure 4, whose three RGB cameras stream at 30 Hz. An operator teleoperates the two arms through a VR headset and hand controllers, and every episode is recorded under the single fixed prompt shown for its task in Figure 3. In SbC, the three cubes start at arbitrary positions on the two boards, and the operator picks and places them in an unconstrained order, keeping the number

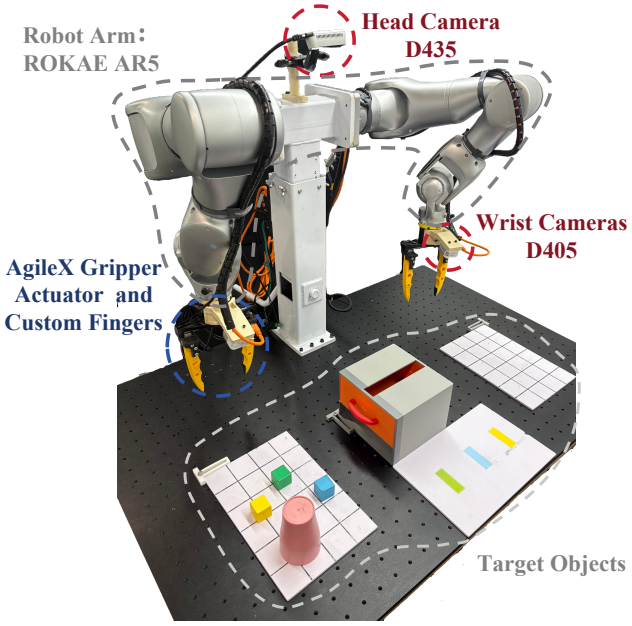


Figure 4: Real-world setup. The platform is a ROKAE AR5 dual-arm system. A RealSense D435 supplies the head view, and a RealSense D405 on each wrist supplies the left and right wrist views. Each arm ends in an AgileX gripper actuator with custom fingers of our own design. The tabletop holds target objects the three real-world tasks manipulate.

of demonstrations balanced across the six possible orderings. In SaC, the blue cube is placed in a grid cell in the upper half of the left board and the green cube in a grid cell in the lower half, while the cup is placed at an arbitrary position on the right board. We record 200 episodes per task, and the resulting 600 episodes form the co-training set.

Policy configuration. Table 7 lists the training configuration of the two policies; all settings outside the table follow the official releases of $\pi_{0.5}$ and GR00T N1.7.

Table 7: Training configuration of the two policies, both fine-tuned on $8\times$ A800 GPUs.

	$\pi_{0.5}$	GR00T N1.7
Image input	$3 \times 224 \times 224$	$3 \times 256 \times 256$
Action space	absolute joint (14-D RoboTwin 2.0, 16-D real robot)	
Chunk length H	32	32
Fine-tuned modules	all	vision tower, projector, action head
Batch size	64 (8 per GPU)	64 (8 per GPU)
Training steps	40,000	40,000

C.3 Baseline definitions and adaptations

The uniform schedule uses time nodes $\tau_k = k/B$. SharpEuler optimizes the sampling schedule using the curvature of the flow trajectory; we apply its rule to each policy's field. Its schedule optimization is training-free and closed-form, requiring no iterative search and using a dense reference ODE trajectory. For each of $S = 50$ calibration chunks per task we integrate the policy's learned field once on a dense uniform grid of $D = 100$ Euler steps at a single fixed noise draw, and record the velocity v_θ at every node, trimmed to the scored (H, d) frame. SharpEuler steps small where the field turns fast, so it places nodes by the field's *material acceleration* along that trajectory. On the

midpoint support $t_k = (\tau_k + \tau_{k+2})/2$ we take the centered finite difference

$$\kappa(t_k) = \Big\| \frac{v_\theta(\tau_{k+1}) - v_\theta(\tau_k)}{(\tau_{k+2} - \tau_k)/2} \Big\|_W, \tag{32}$$

where $\|\cdot\|_W$ is the root mean square over the scored (H, d) frame, the same norm the rest of the method scores with. The per-chunk κ are aggregated episode-balanced—averaged within an episode, then across episodes with equal weight—onto the shared support. The aggregate sets a placement density

$$\rho(t) \propto \big(\kappa(t) + \varepsilon\big)^\gamma, \qquad \gamma = \tfrac{1}{2}, \tag{33}$$

which we smooth with a Gaussian kernel of standard deviation $\sigma = 1$ grid spacing and reflection padding, then floor at $\varepsilon = 10^{-12}$. We extend ρ to both endpoints using its boundary values and obtain the time nodes by evaluating its inverse CDF at $B{+}1$ equally spaced quantiles on $[0, 1]$, with $\tau_0 = 0$ and $\tau_B = 1$. Both policies use these settings and $S = 50$ calibration chunks per task.

Dense-Jump uses $B{-}1$ uniform Euler steps on $[0, 0.5]$ and one terminal step to 1. Its nodes are $\tau_k = k/(2(B{-}1))$ for $k = 0, \ldots, B{-}1$ and $\tau_B = 1$. We re-fine-tune each policy with flow time drawn from $\mathrm{Beta}(0.5, 0.5)$; all other training settings follow Appendix C.2. Table 8 lists the sampling schedules at each evaluated budget.

Table 8: Sampling schedules at every evaluated budget, written as the full node sequence $\tau_0, \ldots, \tau_B$. Uniform and Dense-Jump are task-independent; SharpEuler and GYS are shown on two representative RoboTwin 2.0 tasks, *blocks_ranking_rgb* and *stack_bowls_three*. The percentage after each GYS schedule is its empirical score reduction over uniform on that task, $1 - L/L_{\text{uniform}}$.

B	Method	*blocks_ranking_rgb*	*stack_bowls_three*
3	Uniform	0, 0.333, 0.667, 1	
	Dense-Jump	0, 0.250, 0.500, 1	
	SharpEuler ($\pi_{0.5}$)	0, 0.442, 0.802, 1	0, 0.441, 0.800, 1
	SharpEuler (GR00T N1.7)	0, 0.450, 0.802, 1	0, 0.450, 0.802, 1
	GYS ($\pi_{0.5}$)	0, 0.27, 0.723, 1 (+20.8%)	0, 0.175, 0.685, 1 (+13.3%)
	GYS (GR00T N1.7)	0, 0.218, 0.692, 1 (+38.0%)	0, 0.475, 0.740, 1 (+47.8%)
4	Uniform	0, 0.250, 0.500, 0.750, 1	
	Dense-Jump	0, 0.167, 0.333, 0.500, 1	
	SharpEuler ($\pi_{0.5}$)	0, 0.337, 0.632, 0.874, 1	0, 0.336, 0.630, 0.873, 1
	SharpEuler (GR00T N1.7)	0, 0.349, 0.637, 0.870, 1	0, 0.349, 0.637, 0.870, 1
	GYS ($\pi_{0.5}$)	0, 0.251, 0.413, 0.691, 1 (+16.2%)	0, 0.306, 0.533, 0.774, 1 (+23.8%)
	GYS (GR00T N1.7)	0, 0.583, 0.595, 0.744, 1 (+64.2%)	0, 0.071, 0.554, 0.740, 1 (+56.7%)
5	Uniform	0, 0.200, 0.400, 0.600, 0.800, 1	
	Dense-Jump	0, 0.125, 0.250, 0.375, 0.500, 1	
	SharpEuler ($\pi_{0.5}$)	0, 0.271, 0.518, 0.738, 0.910, 1	0, 0.270, 0.517, 0.736, 0.909, 1
	SharpEuler (GR00T N1.7)	0, 0.282, 0.527, 0.740, 0.903, 1	0, 0.281, 0.527, 0.740, 0.903, 1
	GYS ($\pi_{0.5}$)	0, 0.233, 0.466, 0.658, 0.659, 1 (+22.8%)	0, 0.227, 0.461, 0.545, 0.688, 1 (+28.9%)
	GYS (GR00T N1.7)	0, 0.298, 0.390, 0.583, 0.752, 1 (+17.0%)	0, 0.267, 0.450, 0.451, 0.740, 1 (+25.6%)
6	Uniform	0, 0.167, 0.333, 0.500, 0.667, 0.833, 1	
	Dense-Jump	0, 0.100, 0.200, 0.300, 0.400, 0.500, 1	
	SharpEuler ($\pi_{0.5}$)	0, 0.227, 0.442, 0.632, 0.802, 0.930, 1	0, 0.226, 0.441, 0.630, 0.800, 0.930, 1
	SharpEuler (GR00T N1.7)	0, 0.237, 0.450, 0.637, 0.802, 0.924, 1	0, 0.236, 0.450, 0.637, 0.802, 0.924, 1
	GYS ($\pi_{0.5}$)	0, 0.302, 0.303, 0.622, 0.623, 0.999, 1 (+25.2%)	0, 0.317, 0.399, 0.504, 0.611, 0.863, 1 (+14.9%)
	GYS (GR00T N1.7)	0, 0.082, 0.162, 0.311, 0.465, 0.712, 1 (+15.2%)	0, 0.083, 0.156, 0.300, 0.460, 0.704, 1 (+23.7%)
8	Uniform	0, 0.125, 0.250, 0.375, 0.500, 0.625, 0.750, 0.875, 1	
	Dense-Jump	0, 0.071, 0.143, 0.214, 0.286, 0.357, 0.429, 0.500, 1	
	SharpEuler ($\pi_{0.5}$)	0, 0.170, 0.337, 0.490, 0.632, 0.763, 0.874, 0.953, 1	0, 0.170, 0.336, 0.489, 0.630, 0.761, 0.873, 0.953, 1
	SharpEuler (GR00T N1.7)	0, 0.178, 0.349, 0.499, 0.637, 0.763, 0.870, 0.949, 1	0, 0.177, 0.349, 0.498, 0.637, 0.763, 0.870, 0.949, 1
	GYS ($\pi_{0.5}$)	0, 0.001, 0.002, 0.194, 0.404, 0.405, 0.667, 0.668, 1 (+28.5%)	0, 0.001, 0.24, 0.373, 0.494, 0.495, 0.644, 0.812, 1 (+16.9%)
	GYS (GR00T N1.7)	0, 0.037, 0.069, 0.212, 0.398, 0.399, 0.648, 0.752, 1 (+57.6%)	0, 0.037, 0.069, 0.212, 0.398, 0.399, 0.648, 0.752, 1 (+55.2%)
10	Uniform	0, 0.100, 0.200, 0.300, 0.400, 0.500, 0.600, 0.700, 0.800, 0.900, 1	
	Dense-Jump	0, 0.056, 0.111, 0.167, 0.222, 0.278, 0.333, 0.389, 0.444, 0.500, 1	
	SharpEuler ($\pi_{0.5}$)	0, 0.136, 0.271, 0.401, 0.518, 0.632, 0.738, 0.832, 0.910, 0.965, 1	0, 0.136, 0.270, 0.400, 0.517, 0.630, 0.736, 0.830, 0.909, 0.965, 1
	SharpEuler (GR00T N1.7)	0, 0.143, 0.282, 0.410, 0.527, 0.637, 0.740, 0.831, 0.903, 0.962, 1	0, 0.142, 0.281, 0.409, 0.527, 0.637, 0.740, 0.831, 0.903, 0.962, 1
	GYS ($\pi_{0.5}$)	0, 0.462, 0.546, 0.547, 0.548, 0.549, 0.550, 0.562, 0.606, 0.667, 1 (+32.8%)	0, 0.441, 0.442, 0.443, 0.444, 0.445, 0.446, 0.596, 0.597, 0.612, 1 (+42.3%)
	GYS (GR00T N1.7)	0, 0.160, 0.196, 0.355, 0.444, 0.445, 0.584, 0.705, 0.751, 0.752, 1 (+25.5%)	0, 0.160, 0.196, 0.355, 0.444, 0.445, 0.584, 0.705, 0.751, 0.752, 1 (+34.8%)

C.4 FULL RESULTS ON THE ROBOTWIN 2.0 BENCHMARK

Table 9: Per-task success rate (%) on the 50 RoboTwin 2.0 tasks, under the uniform schedule and the optimized GYS schedule. GR00T N1.7 is reported at $B = 4$ and $B = 8$, $\pi_{0.5}$ at $B = 5$ and $B = 10$. Each GYS entry reports the success rate, with the differencing order $\nu^\star$ used to optimize its schedule in parentheses.

	GR00T N1.7				$\pi_{0.5}$			
	B=4		B=8		B=5		B=10	
Task	Uniform	GYS ($\nu^\star$)	Uniform	GYS ($\nu^\star$)	Uniform	GYS ($\nu^\star$)	Uniform	GYS ($\nu^\star$)
adjust bottle	95	**99** (0.425)	**98**	96 (0.45)	100	100 (1)	100	100 (1.125)
beat block hammer	**67**	64 (0.15)	73	**75** (0.225)	44	**52** (1)	**58**	57 (1.15)
blocks ranking rgb	36	**55** (0.4)	41	**52** (0.45)	79	**88** (1)	81	**88** (1.175)
blocks ranking size	10	**11** (0.475)	**12**	8 (0.55)	40	**44** (0.975)	40	**44** (1.125)
click alarmclock	100	100 (0.175)	98	**100** (0.275)	98	**100** (0.975)	99	**100** (1.175)
click bell	**100**	98 (-0.05)	97	**98** (0.05)	98	**100** (0.65)	98	**100** (0.725)
dump bin bigbin	76	**86** (0.4)	**93**	86 (0.45)	90	**98** (0.9)	**95**	89 (1.025)
grab roller	**100**	97 (0.25)	**99**	97 (0.325)	100	100 (0.85)	100	100 (1.075)
handover block	**43**	40 (0.45)	38	**51** (0.525)	**43**	38 (1)	29	**31** (1.125)
handover mic	64	**84** (0.45)	**87**	78 (0.525)	97	**100** (1.125)	99	**100** (1.325)
hanging mug	8	**17** (0.4)	7	**11** (0.45)	8	**12** (1)	5	**9** (1.15)
lift pot	**66**	55 (0.275)	46	**60** (0.35)	**71**	64 (0.8)	57	**66** (0.925)
move can pot	25	**34** (0.375)	24	**36** (0.475)	53	**56** (0.925)	56	56 (1.05)
move pillbottle pad	29	**34** (0.35)	26	**44** (0.425)	69	**74** (0.975)	67	**70** (1.1)
move playingcard away	**79**	71 (0.225)	**76**	69 (0.3)	90	**93** (1.025)	92	**95** (1.2)
move stapler pad	1	**4** (0.375)	**4**	2 (0.45)	15	**25** (0.925)	13	**18** (1.05)
open laptop	77	**80** (0.575)	**81**	78 (0.625)	86	**88** (0.9)	80	**83** (1)
open microwave	19	**29** (0.45)	7	**33** (0.525)	2	**6** (0.85)	1	**4** (1)
pick diverse bottles	22	**30** (0.35)	**34**	33 (0.35)	63	**64** (1.1)	59	**65** (1.225)
pick dual bottles	**49**	43 (0.325)	47	**54** (0.375)	82	82 (1.025)	73	**78** (1.15)
place a2b left	**31**	29 (0.35)	31	**32** (0.4)	**85**	79 (1)	78	**81** (1.175)
place a2b right	**22**	21 (0.325)	21	**22** (0.375)	75	**82** (1)	70	**76** (1.225)
place bread basket	**51**	50 (0.375)	49	**53** (0.45)	58	**61** (1.075)	54	**57** (1.275)
place bread skillet	52	**56** (0.425)	47	**55** (0.5)	76	**79** (1.125)	76	**79** (1.325)
place burger fries	**81**	70 (0.275)	51	**70** (0.325)	89	**92** (1.075)	**94**	93 (1.3)
place can basket	41	**58** (0.425)	43	**59** (0.475)	41	**50** (0.9)	46	**49** (1.075)
place cans plasticbox	**58**	54 (0.35)	52	**62** (0.4)	**88**	84 (1.025)	**72**	68 (1.175)
place container plate	90	**91** (0.375)	**91**	90 (0.45)	93	**96** (1.1)	94	**97** (1.3)
place dual shoes	**19**	7 (0.375)	**13**	10 (0.4)	51	**54** (0.95)	47	**59** (1.075)
place empty cup	**74**	73 (0.225)	73	**79** (0.275)	83	**89** (0.975)	82	**86** (1.15)
place fan	18	**23** (0.325)	15	**22** (0.4)	62	**65** (0.875)	58	**61** (1.025)
place mouse pad	0	**3** (0.3)	**5**	1 (0.375)	38	**41** (0.95)	32	**35** (1.175)
place object basket	65	**75** (0.45)	**72**	61 (0.5)	70	70 (0.975)	66	**73** (1.125)
place object scale	34	34 (0.325)	**30**	28 (0.375)	72	**77** (0.975)	68	**73** (1.15)
place object stand	66	**75** (0.3)	61	**77** (0.375)	82	**85** (1.075)	**87**	83 (1.3)
place phone stand	**39**	36 (0.25)	33	**35** (0.25)	62	**65** (1.025)	**56**	52 (1.275)
place shoe	35	**38** (0.375)	34	**39** (0.45)	77	**80** (0.875)	**80**	79 (1.075)
press stapler	**93**	91 (0.45)	**93**	91 (0.525)	82	**85** (0.7)	80	**83** (0.75)
put bottles dustbin	16	**42** (0.5)	16	**30** (0.55)	**71**	66 (0.9)	68	**70** (1.025)
put object cabinet	28	28 (0.5)	21	**28** (0.6)	36	**39** (1.225)	33	**36** (1.475)
rotate qrcode	20	**28** (0.425)	21	**25** (0.475)	79	**82** (0.95)	71	**78** (1.15)
scan object	32	**34** (0.55)	23	**32** (0.6)	17	**21** (1.125)	12	**19** (1.225)
shake bottle	95	**98** (0.525)	**98**	97 (0.6)	100	100 (1)	100	100 (1.1)
shake bottle horizontally	96	**97** (0.5)	95	**98** (0.55)	100	100 (1.05)	100	100 (1.15)
stack blocks three	10	**20** (0.425)	10	**19** (0.5)	54	**57** (0.875)	42	**45** (1.025)
stack blocks two	51	**64** (0.35)	44	**67** (0.4)	**95**	91 (0.875)	90	**93** (1)
stack bowls three	38	**52** (0.4)	38	**57** (0.425)	72	**75** (1.075)	70	**76** (1.225)
stack bowls two	**86**	82 (0.35)	75	**87** (0.425)	91	**94** (0.975)	95	**97** (1.125)
stamp seal	14	14 (0.325)	12	**21** (0.425)	56	**59** (1.05)	**55**	54 (1.2)
turn switch	25	**34** (0.35)	**38**	36 (0.375)	**44**	43 (0.675)	42	**45** (0.725)
Mean	48.9	**52.2** (0.366)	47.9	**52.9** (0.428)	68.5	**70.9** (0.969)	66.4	**69.0** (1.125)