SPECFLOW: FAST AND LOSSLESS SPECULATIVE SAMPLING FOR FLOW MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow Matching (FM) has emerged as a leading framework for high-quality image and video generation, yet inference remains expensive because ODE integration requires sequential network evaluations. Existing acceleration methods either require additional training or rely on approximations, generally without exact distributional guarantees. We ask whether FM can instead admit the *lossless* acceleration of speculative decoding, where draft quality affects efficiency but not correctness. A direct extension is challenging because deterministic FM transitions are degenerate (Dirac) and lack the densities required for exact probabilistic verification. We address this by performing speculation on a marginal-preserving stochastic realization of the FM probability flow, whose Gaussian transitions enable exact verification and correction of draft proposals. We obtain the required score directly from the FM velocity, requiring no auxiliary model or additional network evaluations. Building on this, we introduce SpecFlow, a training-free speculative sampler that exactly reproduces its chosen discretized stochastic target sampler for any valid draft; making SpecFlow *lossless*. We further introduce a curvature-adaptive drafting strategy that jointly adapts prediction order and speculative horizon to improve efficiency. Across image generation, image editing, and video generation, SpecFlow achieves $> 2.5\times$ inference speedup while maintaining the target sampler's quality.

1 INTRODUCTION

Flow Matching (FM) (Lipman et al., 2022; Liu et al., 2022) has emerged as a prominent framework for visual generative modeling and underlies large-scale image and video generation models (Deng et al., 2025; Labs et al., 2025; Kong et al., 2024). FM learns a time-dependent velocity field that transports samples from a simple prior to the data distribution along a continuous probability path. Generation then proceeds by numerically integrating the corresponding probability-flow ordinary differential equation (ODE) from noise to data.

Despite its effectiveness, FM inference remains expensive. ODE integration requires repeated evaluations of a large neural network, and these evaluations are inherently sequential because each depends on the state produced by the preceding computation. Inference latency therefore grows with the number of function evaluations (NFEs), creating a significant bottleneck for interactive and high-throughput applications (Davtyan et al., 2025; Liu et al., 2025a; Bajpai et al., 2026a), particularly for computationally demanding tasks such as video generation (Kong et al., 2024; Liu et al., 2025a).

Existing acceleration methods broadly follow two directions: **1) Training-based methods**, including distillation (Yan et al., 2024; Kornilov et al., 2024), consistency models (Yang et al., 2024; Zhang & Zhou, 2025; Dao et al., 2025), and rectification or reflow (Liu et al., 2022), enable sampling with fewer steps but require additional training or modified model weights. **2) Training-free methods** retain the pretrained model and reduce inference cost through techniques such as caching (Liu et al., 2025a), improved ODE solvers (Lu et al., 2022), or adaptive prediction (Bajpai et al., 2026b). While effective, these approaches generally do not guarantee preservation of the target sampling distribution. The accelerated sampler may therefore deviate from the original sampling process, creating a trade-off between computational efficiency and fidelity to the target sampler.

Speculative decoding (SD) (Leviathan et al., 2023) provides a compelling alternative in autoregressive language models. A lightweight draft proposes future tokens, which are verified in parallel by

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.

the target model using a probabilistic acceptance and correction rule (Chen et al., 2023; Xia et al., 2024). This procedure is *lossless*: it exactly preserves the target distribution, so draft quality affects the achievable speedup but not the correctness of the final samples.

Extending this guarantee to FM, however, is fundamentally difficult. FlowCast (Bajpai et al., 2026a) introduces SD for FM by extrapolating and verifying future states, but relies on an approximation-based acceptance criterion and therefore does not provide exact distribution preservation. Fundamentally, deterministic ODE transitions are degenerate (Dirac): unlike token distributions in language models, they provide no non-degenerate transition density for exact probabilistic verification.

Our key insight is that exact speculation need not operate directly on the deterministic ODE. Instead, we construct a stochastic realization of the FM dynamics that shares the same time-marginal distributions as the original probability flow (Song et al., 2020). Its Euler–Maruyama discretization replaces the degenerate ODE with non-degenerate Gaussian transitions, enabling exact probabilistic verification and correction of speculative proposals. Importantly, under the standard linear FM parameterization, the required score can be obtained in closed form from the FM velocity (Albergo et al., 2025; Ma et al., 2024), requiring no auxiliary score model or additional network evaluations. The corresponding marginal-preservation, reverse-time, and score–velocity derivations are provided in detail in Appendices A.1.1, A.1.2, and A.1.3 for completeness.

Building on this stochastic realization, we introduce *Fast and Lossless Speculative Sampling for Flow Matching* (SpecFlow), a training-free framework for speculative acceleration of FM models. Starting from the current committed state, SpecFlow reuses previously computed velocities to fit a low-order polynomial extrapolant of the local velocity field and integrates it in closed form to draft multiple future states without additional network evaluations. The polynomial order and speculative horizon are adapted using divided differences of the velocity history, allowing longer, higher-order drafts in smooth regions and more conservative drafts where the trajectory bends. The drafted states are then verified in parallel against the Gaussian target transitions and, upon the first rejection, corrected exactly before speculation resumes. Consequently, for any valid draft, SpecFlow exactly reproduces the chosen discretized stochastic target sampler, so draft quality affects only the acceptance rate and achievable speedup, not correctness; the complete proof, including the sequential block-level argument, is provided in Appendix A.1.4. Here, *lossless* refers to exact reproduction of this stochastic target, rather than pathwise equivalence with the deterministic FM sampler.

Overall, SpecFlow enables training-free speculative acceleration without modifying the underlying FM model. Across image generation, image editing, and video generation tasks, SpecFlow achieves over $2.5\times$ inference speedup while maintaining the quality of the target sampler.

Contributions: In summary, our main contributions are:

- We propose SpecFlow, a training-free framework for speculative acceleration of pretrained FM models through a marginal-preserving stochastic realization.
- We prove that SpecFlow exactly reproduces its discretized stochastic target sampler for any valid draft, so draft quality affects only acceptance rate and speedup, not correctness.
- We develop a curvature-adaptive strategy that adjusts drafting order and horizon using stored velocities, improving efficiency without affecting exactness.
- Across image generation, image editing, and video generation tasks, SpecFlow achieves over $2.5\times$ inference speedup while maintaining the quality of the target sampler.

2 Related Work

Flow Matching and efficient sampling. Flow Matching (FM) (Lipman et al., 2022; Liu et al., 2022; Dao et al., 2023) learns continuous-time velocity fields that transport samples from a simple prior to the data distribution. FM and rectified-flow formulations underlie large-scale image, editing, and video models, including FLUX (Labs et al., 2025) and HunYuanVideo (Kong et al., 2024), but their sequential ODE sampling remains computationally expensive.

Training-based acceleration. Training-based approaches reduce sampling steps through distillation and consistency (Luhman & Luhman, 2021; Salimans & Ho, 2022; Song et al., 2023; Liu et al., 2023), rectification or reflow (Liu et al., 2022; Yan et al., 2024), and related few-step training ob-

jectives (Schusterbauer et al., 2025). While effective, these methods require additional training or modified model weights.

Training-free acceleration. Training-free methods retain the pretrained model and accelerate inference using improved ODE solvers (Lu et al., 2022; Shaul et al., 2023), caching methods such as TeaCache (Liu et al., 2025a), or adaptive inference-time prediction (Bajpai et al., 2026b). These approaches generally approximate or skip computation and do not guarantee exact preservation of the target sampling distribution.

Speculative sampling. Speculative decoding accelerates autoregressive generation by drafting future tokens and verifying them in parallel while exactly preserving the target distribution through probabilistic acceptance and correction (Leviathan et al., 2023; Chen et al., 2023; Xing et al., 2024). FlowCast (Bajpai et al., 2026a) extends speculation to FM using trajectory extrapolation and MSE-based approximate verification, but does not provide exact distribution preservation. Exact speculative sampling has also been developed for stochastic diffusion processes (De Bortoli et al., 2025), whose non-degenerate transitions permit probabilistic verification; these methods do not directly apply to deterministic FM ODEs.

Our approach. SpecFlow instead performs speculative verification on a marginal-preserving stochastic realization of FM (Song et al., 2020). Its non-degenerate transitions enable exact verification and correction, while the required score is recovered directly from the FM velocity (Ma et al., 2024), yielding training-free speculative acceleration without additional network evaluations.

3 SETUP

We review the two foundations of our approach: speculative decoding (SD), which provides an exact draft-and-verify framework for accelerating sequential generation, and Flow Matching (FM), whose deterministic ODE sampling we seek to accelerate.

3.1 SPECULATIVE DECODING

Speculative decoding (SD) (Xing et al., 2024) accelerates autoregressive generation from an expensive target model using a cheaper draft model, while preserving the target distribution exactly. Its key idea is to generate several future tokens cheaply with the draft model and verify them in parallel with the target model, thereby reducing the number of sequential target-model evaluations.

Let $\mathcal{M}_p$ denote the target model with conditional distribution p, and $\mathcal{M}_q$ a faster draft model with conditional distribution q. Given a prefix $x_{1:i}$, SD proceeds in two stages.

1) Drafting. The draft model $\mathcal{M}_q$ autoregressively proposes K future tokens,

$$\tilde{x}_{i+k} \sim q(\cdot \mid x_{1:i}, \tilde{x}_{i+1:i+k-1}), \qquad k = 1, \dots, K. \tag{1}$$

Since $\mathcal{M}_q$ is substantially cheaper than $\mathcal{M}_p$, these tokens can be generated at relatively low cost.

2) Verification. The target model evaluates the speculative positions in parallel and computes their corresponding conditional distributions. Let p_k and q_k denote the target and draft distributions at the k-th speculative position. A proposal $\tilde{x}$ is accepted with probability

$$\alpha_k(\tilde{x}) = \min\left\{1, \frac{p_k(\tilde{x})}{q_k(\tilde{x})}\right\}. \tag{2}$$

Verification proceeds sequentially through the speculative block until the first rejection. Together with the corresponding correction rule (Leviathan et al., 2023), this acceptance mechanism ensures that the resulting sequence has exactly the same distribution as ordinary sampling from $\mathcal{M}_p$.

The defining property of SD is therefore *losslessness*: the draft affects the acceptance rate, and hence the achievable speedup, but not the output distribution. Crucially, this guarantee relies on draft and target probabilities for each proposed state, making the likelihood ratio in equation 2 well-defined.

This requirement creates the central difficulty in extending SD to FM. Autoregressive language models provide explicit categorical distributions over the next token, whereas deterministic FM sampling maps each state to a unique successor along the probability-flow trajectory. Consequently, FM does not directly provide the non-degenerate next-state distribution required for exact probabilistic verification and residual correction. We formalize this distinction below in detail.

3.2 FLOW MATCHING

Flow Matching (FM) (Lipman et al., 2022; Dao et al., 2023) learns a time-dependent velocity field that transports samples between a simple noise distribution and the data distribution. We consider the linear probability path commonly used in flow models. Given a data sample $x_0 \sim \pi_{\text{data}}$ and independent Gaussian noise $\varepsilon \sim \mathcal{N}(0, I)$, the intermediate state is

$$x_t = (1-t)x_0 + t\varepsilon, \qquad t \in [0,1], \tag{3}$$

where $t = 0$ corresponds to data and $t = 1$ to noise. The velocity along each interpolation path is $\varepsilon - x_0$, and FM learns the corresponding marginal velocity field

$$v^\star(x,t) = \mathbb{E}[\varepsilon - x_0 \mid x_t = x], \tag{4}$$

which is approximated by a neural network $v_\theta(x,t)$.

The learned velocity defines the probability-flow ODE $\frac{dx_t}{dt} = v_\theta(x_t,t)$. Generation starts from Gaussian noise at $t = 1$ and integrates this ODE backward toward data at $t = 0$. Since the learned velocity generally does not admit a closed-form solution, sampling requires numerical integration. For a decreasing schedule $1 = t_0 > t_1 > \cdots > t_N = 0$, explicit Euler gives

$$x_{k+1} = x_k - h_k v_\theta(x_k, t_k), \qquad h_k = t_k - t_{k+1} > 0. \tag{5}$$

Each update requires a network evaluation, and the resulting state is needed before the next evaluation can be performed. Sampling is therefore sequential, making the number of function evaluations (NFEs) a major contributor to inference latency.

Why exact speculation is difficult for FM. Unlike autoregressive sampling, the Euler update in equation 5 is deterministic: conditioned on the current state x_k, the velocity $v_\theta(x_k, t_k)$ determines a unique successor. Its transition kernel is therefore the Dirac measure

$$P_k(dy \mid x_k) = \delta_{x_k - h_k v_\theta(x_k,t_k)}(dy), \tag{6}$$

rather than a non-degenerate distribution over possible next states. If a deterministic draft predicts a different successor, the two transition measures have no overlapping mass; unless the predictions coincide exactly, there is no useful likelihood ratio analogous to the draft–target ratio in SD. Approximate speculative methods can instead accept predictions sufficiently close to the target (Bajpai et al., 2026a), but such proximity-based verification generally introduces approximation error and does not provide the exact distribution-preservation guarantee of SD.

This motivates our approach: rather than speculating directly on the deterministic ODE, we perform speculation on a marginal-preserving stochastic realization of the FM probability path with explicit, non-degenerate transition densities suitable for exact probabilistic verification.

3.3 SPECFLOW: SPECULATIVE SAMPLING FOR FLOW MATCHING

We resolve the degeneracy of deterministic FM transitions by speculating on a marginal-preserving stochastic realization of the same probability flow, yielding non-degenerate transition densities for exact verification while preserving the continuous-time marginals as detailed below.

Stochastic Realization of Flow Matching: We build on the standard probability-flow ODE/SDE equivalence (Song et al., 2020). Let p_t denote the marginal density induced by the ideal FM probability-flow ODE with velocity $v^\star$, and let $s_t(x) = \nabla_x \log p_t(x)$ denote its score. For any diffusion schedule $g(t) \geq 0$, the stochastic process

$$dx_t = \left[v^\star(x_t,t) + \frac{1}{2} g(t)^2 s_t(x_t) \right] dt + g(t)\, dW_t \tag{7}$$

has the same time-marginal distributions p_t as the corresponding probability-flow ODE under standard regularity conditions. Thus, stochasticity changes individual trajectories while preserving the continuous-time marginal evolution of the original FM probability path. The complete marginal-preservation derivation is provided in Appendix A.1.1.

For the linear interpolation in Eq. 3, the score and ideal marginal velocity satisfy (Ma et al., 2024)

$$s_t(x) = -\frac{x + (1-t)v^\star(x,t)}{t}, \qquad t \in (0,1]. \tag{8}$$

In practice, we obtain the corresponding score estimate directly from the learned velocity,

$$\hat{s}_t(x) = -\frac{x + (1-t)v_\theta(x,t)}{t}. \tag{9}$$

Thus, no auxiliary score model or additional network evaluation is required. The identity in equation 8 is exact for the ideal marginal velocity $v^\star$, while $\hat{s}_t$ inherits the approximation error of the learned velocity v_θ. A derivation of the score–velocity relation is provided in Appendix A.1.3; the effect of learned-velocity approximation error is studied empirically in Appendix B.4.

In the noise-to-data direction used for generation, Euler–Maruyama discretization over $1 = t_0 > \cdots > t_N = 0$ yields the Gaussian target transition

$$p_k(\cdot \mid x_{t_k}) = \mathcal{N}(\mu_k, \Sigma_k), \qquad \mu_k = x_{t_k} - h_k v_\theta(x_{t_k}, t_k) + \frac{1}{2} h_k g(t_k)^2 \hat{s}_{t_k}(x_{t_k}), \qquad \Sigma_k = h_k g(t_k)^2 I. \tag{10}$$

Under the ideal-velocity assumption $v_\theta = v^\star$, we have $\hat{s}_t = s_t$, and equation 10 is the Euler–Maruyama discretization of the marginal-preserving stochastic realization in equation 7. In practice, equation 10 is constructed from v_θ and $\hat{s}_t$ and defines the chosen discretized stochastic target that SpecFlow reproduces exactly; this exactness does not require $v_\theta = v^\star$. The reverse-time dynamics and Euler–Maruyama discretization are derived in Appendix A.1.2.

3.3.1 Consistent Speculative Sampling

With an explicit target transition available, SpecFlow follows a draft–verify–correct procedure. Drafting determines efficiency, while verification and correction ensure that every committed transition follows the target stochastic kernel.

Drafting: Sampling begins from $x_{t_0} \sim \mathcal{N}(0, I)$, followed by one target Euler–Maruyama step to initialize the committed state and velocity history. Thereafter, suppose x_{t_m} is the latest committed state. Using only velocities evaluated at previously committed states, the drafter extrapolates the local velocity field and integrates it along the decreasing sampling direction to predict a mean $\bar{x}_{t_{k+1}}$ for each future transition; the precise higher-order construction is given in Section 3.3.2 and Appendix A.1.7. We define the draft transition

$$q_k(\cdot \mid \tilde{x}_{t_k}) = \mathcal{N}\big(\bar{x}_{t_{k+1}}, \Sigma_k\big), \qquad \tilde{x}_{t_{k+1}} \sim q_k(\cdot \mid \tilde{x}_{t_k}), \tag{11}$$

using the same covariance Σ_k as the target transition. Equal covariance is not required for exactness, but makes verification and correction particularly simple.

Because drafting uses only stored velocity evaluations, it requires no additional FM network evaluations. Draft accuracy affects the acceptance probability, and hence efficiency, but not the correctness of the resulting sampler.

Verification: After constructing a speculative block, we evaluate the FM model at all drafted states in a single batched forward pass. The corresponding score estimates $\hat{s}_{t_k}$ are obtained from the same velocity evaluations using equation 9, which determines the target means μ_k in equation 10 without additional network calls.

A proposal $\tilde{x}_{t_{k+1}}$ is accepted with probability

$$\alpha_k = \min\left\{1, \frac{p_k(\tilde{x}_{t_{k+1}} \mid \tilde{x}_{t_k})}{q_k(\tilde{x}_{t_{k+1}} \mid \tilde{x}_{t_k})}\right\}. \tag{12}$$

Because the target and draft transitions have the same covariance,

$$\log \frac{p_k(\tilde{x}_{t_{k+1}} \mid \tilde{x}_{t_k})}{q_k(\tilde{x}_{t_{k+1}} \mid \tilde{x}_{t_k})} = \frac{\|\tilde{x}_{t_{k+1}} - \bar{x}_{t_{k+1}}\|_2^2 - \|\tilde{x}_{t_{k+1}} - \mu_k\|_2^2}{2h_k g(t_k)^2}. \tag{13}$$

Thus, verification requires only the draft and target means after the batched model evaluation, with no Jacobian or divergence computation. Verification proceeds in temporal order until the first rejection; all preceding proposals are committed and all subsequent speculative states are discarded.

Correction: Suppose transition j is the first rejected proposal. To preserve the target transition exactly, the corrected state is sampled from the normalized residual

$$r_j(x \mid x_{t_j}) = \frac{\big[p_j(x \mid x_{t_j}) - q_j(x \mid x_{t_j})\big]_+}{\int \big[p_j(z \mid x_{t_j}) - q_j(z \mid x_{t_j})\big]_+ dz}, \tag{14}$$

Algorithm 1 Consistent Speculative Flow Matching (SpecFlow)

Require: FM velocity model v_θ, schedule $\{t_k\}_{k=0}^{N}$, diffusion schedule g, maximum drafting order $r_{\max}$, maximum speculative horizon $W_{\max}$

Sample $x_{t_0} \sim \mathcal{N}(0, I)$
Take one target Euler–Maruyama step to obtain x_{t_1} and initialize the velocity history
Set $m \leftarrow 1$
while $m < N$ **do**
  Select drafting order r_m and horizon $W_m \le \min(W_{\max}, N - m)$
  Drafting: construct the order-r_m velocity extrapolant from the committed velocity history
  for $k = m, \dots, m + W_m - 1$ **do**
    Compute the predicted mean $\bar{x}_{t_{k+1}}$; Set $q_k \leftarrow \mathcal{N}(\bar{x}_{t_{k+1}}, \Sigma_k)$; Sample $\tilde{x}_{t_{k+1}} \sim q_k(\cdot \mid \tilde{x}_{t_k})$
  end for
  Verification: evaluate $v_\theta(\tilde{x}_{t_k}, t_k)$ for $k = m, \dots, m + W_m - 1$ in one batched forward pass
  Compute $\hat{s}_{t_k}$, μ_k, $p_k = \mathcal{N}(\mu_k, \Sigma_k)$, and α_k using equation 9–equation 12
  Draw $u_k \sim \mathrm{Unif}(0, 1)$ and let j be the first index such that $u_j > \alpha_j$
  if no proposal is rejected **then**
    Commit $x_{t_k} \leftarrow \tilde{x}_{t_k}$ for $k = m + 1, \dots, m + W_m$; Set $m \leftarrow m + W_m$
  else
    Commit $x_{t_k} \leftarrow \tilde{x}_{t_k}$ for $k = m + 1, \dots, j$
    Correction: sample $x_{t_{j+1}} \sim r_j(\cdot \mid x_{t_j})$
    Set $m \leftarrow j + 1$
  end if
  if $m < N$ **then**
    Update the velocity history using the committed target evaluation at t_m; evaluate $v_\theta(x_{t_m}, t_m)$ only if unavailable
  end if
end while
return x_{t_N}

where $[a]_+ = \max(a, 0)$. The corrected state is committed, the remaining speculative states are discarded, and drafting resumes from the corrected state. For the equal-covariance Gaussian transitions used by SpecFlow, the residual in equation 14 admits an exact reflection-based sampler. Lemma 1 establishes that its rejection-branch output has precisely the required residual density; the construction and proof are given in Appendix A.1.6. An ablation isolating the empirical effect of residual correction is provided in Appendix B.3.

3.3.2 Adaptive Drafting

The exactness of SpecFlow does not depend on the particular valid draft transition used in each speculative block (Theorem 1). We therefore optimize the draft solely for acceptance using velocity evaluations already available at committed states.

Higher-order velocity extrapolation: At the latest committed state x_{t_m}, let $v_i = v_\theta(x_{t_i}, t_i)$ denote the stored FM velocities. Using the most recent $r+1$ evaluations, we construct the degree-r Newton interpolation polynomial

$$\widehat{v}_m^{(r)}(t) = \sum_{j=0}^{r} v[t_m, \dots, t_{m-j}] \prod_{\ell=0}^{j-1} (t - t_{m-\ell}), \tag{15}$$

where $v[\cdot]$ denotes divided differences. The predicted future state is obtained by integrating the extrapolated velocity along the decreasing sampling direction,

$$\bar{x}_{t_k} = x_{t_m} - \int_{t_k}^{t_m} \widehat{v}_m^{(r)}(\tau)\, d\tau, \qquad k > m. \tag{16}$$

The integral is available in closed form. For $r = 0$, this reduces to $\bar{x}_{t_k} = x_{t_m} - (t_m - t_k) v_m$, recovering constant-velocity extrapolation. Higher orders capture local changes in the velocity field and can therefore better approximate curved trajectory segments without additional FM evaluations.

Adaptive order and horizon: The reliability of extrapolation varies along the sampling trajectory. We estimate local velocity variation using the divided differences already computed in equation 15. Smooth regions permit higher-order prediction and longer speculative blocks, whereas regions with stronger variation use lower-order predictors and shorter horizons. The complete selection rule for (r_m, W_m), including the normalized divided-difference criterion and thresholds, is specified in Appendix A.1.7. Since the rule uses only stored velocities, adaptation introduces no additional network evaluations. The effects of drafting order and speculative horizon are ablated in Appendix B.8.

3.3.3 Analysis

We now establish the central guarantee of SpecFlow: speculative drafting affects sampling efficiency, but not the distribution of the target Euler–Maruyama sampler.

Theorem 1 (Exactness with respect to the stochastic target)**.** *Let $p_k(\cdot \mid x_k)$ be the target Euler–Maruyama transition and let $q_k(\cdot \mid x_k)$ be any valid draft transition with an evaluable density. Suppose each proposal is accepted according to equation 12 and, upon rejection, the next committed state is sampled from the residual distribution in equation 14. Then every committed transition has conditional distribution $p_k(\cdot \mid x_k)$. Consequently, Algorithm 1 has exactly the same joint distribution as the corresponding target Euler–Maruyama chain, independently of the drafting strategy.*

The full transition- and block-level proof is provided in Appendix A.1.4. Theorem 1 separates correctness from efficiency: draft accuracy affects the acceptance rate and therefore the achievable speedup, but not the distribution of the committed trajectory.

Acceptance rate: For a fixed committed state x_k, the expected acceptance probability is the overlap between the target and draft transitions:

$$\mathbb{E}[\alpha_k \mid x_k] = \int \min\{p_k(x \mid x_k), q_k(x \mid x_k)\}\, dx = 1 - \mathrm{TV}\left(p_k(\cdot \mid x_k), q_k(\cdot \mid x_k)\right). \tag{17}$$

Thus, improving the draft increases acceptance without affecting exactness.

For the equal-covariance Gaussian transitions in equation 10 and equation 11, Proposition 1 gives

$$\mathbb{E}[\alpha_k \mid x_k] = 2\left[1 - \Phi\left(\frac{\Delta_k}{2}\right)\right], \qquad \Delta_k = \frac{\|\bar{x}_{t_{k+1}} - \mu_k\|_2}{\sqrt{h_k}\, g(t_k)}, \tag{18}$$

where Φ is the standard normal CDF. Hence, acceptance decreases monotonically with the normalized discrepancy Δ_k, directly motivating the adaptive drafting strategy in Section 3.3.2. The proof of Proposition 1 is given in Appendix A.1.5, and the effect of the diffusion scale $g(t)$ on target quality, acceptance, and speedup is studied in Appendix B.5.

Scope of exactness: Theorem 1 establishes exactness with respect to the chosen discretized Euler–Maruyama target sampler: SpecFlow introduces no additional distributional error beyond this stochastic target. At the continuous-time level, under the ideal FM velocity assumption, the stochastic realization shares the time-marginal distributions of the FM probability-flow ODE (Song et al., 2020); finite-step discretization and approximation of the ideal velocity by v_θ may introduce their own errors. We separately examine the deterministic–stochastic finite-step discrepancy in Appendix B.2 and learned-velocity approximation error in Appendix B.4. Thus, *lossless* refers specifically to exact speculative acceleration of the discretized stochastic target sampler, rather than pathwise reproduction of the original deterministic FM trajectory.

4 Experiments

We evaluate SpecFlow along three dimensions: **(i)** wall-clock acceleration across FM models and modalities; **(ii)** distributional agreement with its stochastic target relative to approximate accelerators; and **(iii)** the effects of stochasticity and adaptive drafting on acceptance and efficiency. We study image generation, editing, and video generation using pretrained models without additional training.

Datasets and Metrics: For **text-to-image generation**, we evaluate on GenEval (Ghosh et al., 2023), which contains 553 prompts covering compositional attributes including object co-occurrence,

counting, color attribution, and spatial relations. We report the GenEval overall score together with CLIP-IQA for perceptual image quality. For **image editing**, we use GEdit (Liu et al., 2025b), containing 606 real-world editing instructions spanning object, attribute, color, and structural modifications. Following the benchmark protocol, editing quality is evaluated using GPT-4.1 as the judge, and we report semantic consistency (G_{SC}), perceptual quality (G_{PQ}), and the overall score (G_{O}).

For **video generation**, we evaluate on VBench (Zheng et al., 2025). Following the evaluation protocol used by prior FM work, we use a balanced subset of 80 prompts obtained by sampling five prompts from each of the 16 VBench dimensions. We report VBench quality together with frame-level perceptual metrics.

Models: We evaluate several pretrained flow-based generative models. For image generation, we use BAGEL (Deng et al., 2025) and FLUX (Labs et al., 2025). For image editing, we evaluate BAGEL and FLUX-Kontext (Labs et al., 2025). For video generation, we use HunyuanVideo (Kong et al., 2024). All models are evaluated using their released pretrained weights; SpecFlow requires no retraining or modification of model parameters.

Baselines: We compare SpecFlow against both training-based and training-free acceleration methods. As training-based baselines, we include **InstaFlow** (Liu et al., 2023), which distills generation into very few steps, and **PeRFlow** (Yan et al., 2024), which learns piecewise-rectified trajectories for efficient sampling. Since these methods require specialized training and are available with different pretrained backbones, we report results using their released checkpoints and treat them as contextual quality–efficiency comparisons rather than same-model acceleration baselines. Among training-free methods, we compare against **TeaCache** (Liu et al., 2025a), which reuses intermediate computations across timesteps; **FastFlow** (Bajpai et al., 2026b), which uses finite-difference velocity extrapolation with bandit-based adaptive step skipping; and **FlowCast** (Bajpai et al., 2026a), which performs speculative FM inference through trajectory extrapolation and approximate verification. We additionally report the unaccelerated **Full** sampler and direct step reduction to characterize the quality–efficiency trade-off without speculative acceleration.

Evaluation protocol: All experiments are conducted on a single NVIDIA H100 80GB GPU. Speedup is measured as end-to-end wall-clock sampling latency relative to the corresponding full-step sampler under identical hardware, precision, batch size, resolution, conditioning, and timestep schedule. For SpecFlow, speculative states within each block are batched along the model input dimension and evaluated in a single target forward pass on the same GPU; no multi-GPU parallelism is used. Reported latency includes drafting, batched target verification, acceptance testing, rejection handling, and residual correction, and therefore reflects realized end-to-end acceleration rather than an NFE-based estimate. Unless stated otherwise, all quality metrics use identical prompts, seeds, and evaluation protocols across methods. We distinguish batched verification passes from total target states evaluated and report end-to-end wall-clock latency as our primary efficiency measure; detailed timing and implementation settings are provided in Appendices B.6 and B.7.

Importantly, SpecFlow is exact with respect to its discretized stochastic target, rather than the deterministic FM sampler. We therefore separately evaluate deterministic–stochastic and stochastic–SpecFlow agreement on pretrained and controlled analytic models, directly testing both sources of distributional discrepancy; full protocols and results are provided in Appendix A.2.

4.1 Results

Image Generation: Table 1 compares image-generation quality and acceleration on BAGEL and FLUX. Direct step reduction provides large nominal speedups but progressively degrades semantic and perceptual quality. TeaCache improves efficiency while retaining substantially more quality, whereas FlowCast uses speculative execution to obtain larger acceleration at comparable quality. In contrast, SpecFlow accelerates the chosen stochastic target through exact speculative verification and correction. Its quality is therefore evaluated relative to the corresponding stochastic target sampler, while its efficiency is governed by speculative acceptance. Across comparable training-free settings, SpecFlow achieves a favorable quality–speed trade-off while retaining its exactness guarantee with respect to the stochastic target.

Distributional validation: Because SpecFlow is exact with respect to its discretized stochastic target rather than the deterministic finite-step FM sampler, we separately evaluate both the target choice

Method	BAGEL			FLUX		
	GenEval ↑	CLIP-IQA ↑	Speedup ↑	GenEval ↑	CLIP-IQA ↑	Speedup ↑
Full Sampling / Step Reduction						
Full-50	0.78	0.84	1.00×	0.65	0.83	1.00×
Full-25	0.77	0.82	2.00×	0.64	0.80	2.00×
Full-10	0.73	0.75	5.00×	0.57	0.59	5.00×
Full-5	0.57	0.40	10.00×	0.44	0.43	10.00×
Training-Based Acceleration†						
InstaFlow	0.33	0.70	50.0×	–	–	–
PeRFlow	0.58	0.79	5.0×	–	–	–
Training-Free Acceleration						
TeaCache	0.74	0.79	1.80×	0.63	0.80	1.84×
FastFlow	0.77	0.80	2.15×	0.63	0.79	2.21×
FlowCast-50	0.77	0.82	2.35×	0.63	0.80	2.03×
SpecFlow-50	**0.78**	**0.85**	**2.52**×	**0.65**	**0.82**	**2.59**×
SpecFlow-25	**0.77**	**0.82**	**4.45**×	**0.63**	**0.80**	**4.57**×
SpecFlow-10	**0.73**	**0.76**	**7.19**×	**0.58**	**0.60**	**7.11**×
SpecFlow-5	**0.57**	**0.39**	**12.05**×	**0.45**	**0.43**	**12.89**×

Table 1: **Image generation on GenEval.** We report semantic generation quality, perceptual quality, and end-to-end speedup. †InstaFlow and PeRFlow use their released specialized checkpoints and therefore provide contextual quality–efficiency comparisons rather than same-backbone acceleration results. Training-free methods operate without retraining the evaluated pretrained model.

and speculative execution. On BAGEL, Deterministic–Stochastic, Stochastic–SpecFlow, and the independent Stochastic–Stochastic calibration yield only $1/20$ MMD rejections each, indicating no detectable discrepancy beyond the finite-sample calibration level at the evaluated setting. Controlled analytic experiments further examine finite-step target and learned-velocity errors. Full protocols and results are provided in Appendix A.2.

Image Editing: We next evaluate whether acceleration remains effective when generation is constrained by an input image and editing instruction. We evaluate BAGEL and FLUX-Kontext on GEdit, reporting G_{SC}, G_{PQ}, G_{O}, and wall-clock speedup relative to the corresponding full sampler. Because SpecFlow exactly corrects rejected proposals, speculative aggressiveness affects efficiency rather than the transition law of the chosen stochastic target. Full quality–speed curves are shown in Appendix Fig. 1.

Video Generation: Video generation provides a demanding test because each network evaluation operates on a high-dimensional spatiotemporal representation. We evaluate HunyuanVideo on VBench and report semantic/temporal and perceptual quality together with end-to-end latency; full quality–speed curves are shown in Appendix Fig. 2.

5 Conclusion

We introduced SpecFlow, a training-free framework for fast and lossless speculative sampling of Flow Matching models. By moving speculation from the deterministic probability-flow ODE to a marginal-preserving stochastic realization, SpecFlow obtains non-degenerate transition densities that enable exact probabilistic verification and correction without retraining or auxiliary network evaluations. We prove that the resulting sampler exactly reproduces its chosen discretized stochastic target for any valid draft, separating correctness from drafting quality, and exploit this freedom through curvature-adaptive, higher-order drafting. Across image generation, image editing, and video generation, SpecFlow provides substantial end-to-end acceleration while maintaining the quality and distributional behavior of the target sampler. More broadly, our results show that stochastic realizations can provide a principled route for bringing exact speculative acceleration to generative processes whose native dynamics are deterministic.

LIMITATIONS

SpecFlow is exact with respect to its chosen discretized stochastic target, whose fidelity to the ideal continuous-time FM dynamics still depends on the learned velocity, discretization, and diffusion schedule. Moreover, the achievable speedup depends on trajectory predictability: regions with rapidly varying velocities may yield shorter accepted speculative blocks and therefore smaller acceleration. These limitations do not affect the exactness of the speculative mechanism itself, but motivate improved stochastic target design, drafting strategies, and adaptive schedules as promising directions for future work.

AI USE STATEMENT

In this work, we used generative AI tools solely to assist with writing and language refinement, specifically, improving the clarity, grammar, phrasing, and readability of author-drafted text. All technical content, including the problem formulation, methodology, theoretical results and proofs, experimental design, and analysis, was conceived, developed, and written by the authors. AI tools were not used to generate research ideas, derive or verify proofs, produce experimental results, write or debug code, conduct the literature review, or create any figures, tables, or data. Any such tasks requiring disclosure under the policy are therefore not applicable to this work. We have reviewed all AI-assisted text. Every passage edited or rephrased with AI assistance was checked by the authors for factual accuracy, correct representation of our claims, and consistency with the underlying technical content, and was revised as needed. We take full responsibility for the final content of this work, including all text, claims, and artifacts. See the ICLR 2027 AI Policy for Authors for more details.

ETHICS STATEMENT

This work develops an inference-time acceleration method for pretrained generative models and does not introduce new datasets or involve human-subject studies. Our method inherits the potential biases, limitations, and misuse risks of the underlying pretrained generative models. We encourage responsible use in accordance with the licenses and usage policies of the corresponding models and datasets.

REPRODUCIBILITY STATEMENT

We provide the complete algorithm, theoretical assumptions, and proofs in the main paper and Appendix A.1.4. Detailed derivations of the stochastic realization, score–velocity relation, acceptance probability, and exact residual correction are included in the appendix. Experimental protocols, model configurations, sampling hyperparameters, hardware settings, and ablation details are reported in Appendix B.7. Distributional-fidelity experiments include independent target–target controls and repeated trials. These details are provided to enable reproduction of both the theoretical and empirical results.

REFERENCES

Michael Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025.

Divya Jyoti Bajpai, Shubham Agarwal, Apoorv Saxena, Kuldeep Sharad Kulkarni, Subrata Mitra, and Manjesh Kumar Hanawal. Flowcast: Trajectory forecasting for scalable zero-cost speculative flow matching. In *International Conference on Learning Representations*, volume 2026, pp. 91560–91580, 2026a.

Divya Jyoti Bajpai, Dhruv Bhardwaj, Soumya Roy, Tejas Duseja, Harsh Agarwal, Aashay Sandansing, and Manjesh Kumar Hanawal. Fastflow: Accelerating the generative flow matching models with bandit inference. *arXiv preprint arXiv:2602.11105*, 2026b.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023.

Quan Dao, Hao Phung, Trung Tuan Dao, Dimitris N Metaxas, and Anh Tran. Self-corrected flow distillation for consistent one-step and few-step image generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 2654–2662, 2025.

Aram Davtyan, Leello Tadesse Dadi, Volkan Cevher, and Paolo Favaro. Faster inference of flow-based generative models via improved data-noise coupling. In *The Thirteenth International Conference on Learning Representations*, 2025.

Valentin De Bortoli, Alexandre Galashov, Arthur Gretton, and Arnaud Doucet. Accelerated diffusion models via speculative sampling. *arXiv preprint arXiv:2501.05370*, 2025.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36: 52132–52152, 2023.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

Nikita Kornilov, Petr Mokrov, Alexander Gasnikov, and Aleksandr Korotin. Optimal flow matching: Learning straight trajectories in just one step. *Advances in Neural Information Processing Systems*, 37:104180–104204, 2024.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International conference on machine learning*, pp. 19274–19286. PMLR, 2023.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 7353–7363, 2025a.

Shiyu Liu, Yucheng Han, Peng Xing, Fukun Yin, Rui Wang, Wei Cheng, Jiaqi Liao, Yingming Wang, Honghao Fu, Chunrui Han, et al. Step1x-edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025b.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, et al. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *The Twelfth International Conference on Learning Representations*, 2023.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. 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.

Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

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

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Johannes Schusterbauer, Ming Gui, Frank Fundel, and Björn Ommer. Diff2flow: Training flow matching models via diffusion model alignment. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 28347–28357, 2025.

Neta Shaul, Juan Perez, Ricky TQ Chen, Ali Thabet, Albert Pumarola, and Yaron Lipman. Bespoke solvers for generative flow models. *arXiv preprint arXiv:2310.19075*, 2023.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *arXiv preprint arXiv:2401.07851*, 2024.

Zhen Xing, Qijun Feng, Haoran Chen, Qi Dai, Han Hu, Hang Xu, Zuxuan Wu, and Yu-Gang Jiang. A survey on video diffusion models. *ACM Computing Surveys*, 57(2):1–42, 2024.

Hanshu Yan, Xingchao Liu, Jiachun Pan, Jun Hao Liew, Qiang Liu, and Jiashi Feng. Perflow: Piecewise rectified flow as universal plug-and-play accelerator. *Advances in Neural Information Processing Systems*, 37:78630–78652, 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.

Yuchen Zhang and Jian Zhou. Inverse flow and consistency models. In *Forty-second International Conference on Machine Learning*, 2025.

Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yinan He, Fan Zhang, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, Yu Qiao, et al. Vbench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness. *arXiv preprint arXiv:2503.21755*, 2025.

A Appendix

A.1 Proofs and Additional Derivations

This section provides the derivations underlying the stochastic realization, target transition, exact speculative correction, and acceptance analysis used in the main paper. We distinguish throughout between the ideal marginal FM velocity $v^\star$ and the learned velocity v_θ. The continuous-time marginal-preservation statements concern $v^\star$ and its exact score s_t, whereas Algorithm 1 operates on the stochastic target kernel defined by the learned velocity v_θ and the induced score estimate $\hat{s}_t$.

A.1.1 Marginal Preservation of the Stochastic Realization

Our construction builds on the standard probability-flow ODE/SDE correspondence (Song et al., 2020). We include the derivation for completeness; the correspondence itself is not a contribution of our work.

Let the ideal FM probability-flow ODE be

$$dX_t = v^\star(X_t, t)\, dt, \tag{19}$$

and let p_t denote the marginal density of X_t. Assuming sufficient regularity of $v^\star$ and p_t, the density evolves according to the continuity equation

$$\partial_t p_t(x) = -\nabla_x \cdot \left(p_t(x) v^\star(x,t)\right). \tag{20}$$

Let

$$s_t(x) = \nabla_x \log p_t(x)$$

denote the exact marginal score, and let $g : [0,1] \to \mathbb{R}_{\geq 0}$ be a scalar diffusion schedule. Consider the stochastic process

$$dX_t = \left[v^\star(X_t, t) + \frac{1}{2} g(t)^2 s_t(X_t)\right] dt + g(t)\, dW_t. \tag{21}$$

Because the diffusion coefficient depends only on time, its density satisfies the Fokker–Planck equation

$$\partial_t p_t(x) = -\nabla_x \cdot \left[p_t(x) \left(v^\star(x,t) + \frac{1}{2} g(t)^2 s_t(x)\right)\right] + \frac{1}{2} g(t)^2 \Delta_x p_t(x). \tag{22}$$

Using

$$p_t(x) s_t(x) = p_t(x) \nabla_x \log p_t(x) = \nabla_x p_t(x), \tag{23}$$

the score-dependent drift term becomes

$$-\frac{1}{2} g(t)^2 \nabla_x \cdot \left(\nabla_x p_t(x)\right) = -\frac{1}{2} g(t)^2 \Delta_x p_t(x).$$

Substituting this into equation 22 gives

$$\partial_t p_t(x) = -\nabla_x \cdot \left(p_t(x) v^\star(x,t)\right) - \frac{1}{2} g(t)^2 \Delta_x p_t(x) + \frac{1}{2} g(t)^2 \Delta_x p_t(x) \tag{24}$$

$$= -\nabla_x \cdot \left(p_t(x) v^\star(x,t)\right). \tag{25}$$

Equation equation 25 is exactly the continuity equation equation 20. Therefore, if the ODE and SDE are initialized from the same marginal distribution and the corresponding density evolution is unique, they induce the same marginal density p_t for every $t \in [0,1]$.

The equality is distributional rather than pathwise: adding $g(t)dW_t$ changes individual trajectories, but the additional score drift exactly compensates for the diffusion at the level of the Fokker–Planck equation. This is the marginal-preserving property used in Section 3.3.

We emphasize that this is an ideal continuous-time statement. It assumes the exact marginal velocity $v^\star$ and exact score s_t. The implementation uses the learned velocity v_θ and the score estimate $\hat{s}_t$ defined in equation 9; moreover, Euler–Maruyama discretization introduces numerical discretization error. The exactness result of Theorem 1 is therefore stated with respect to the resulting discrete stochastic target sampler, not directly with respect to the continuous-time ideal process.

A.1.2 Reverse-Time Dynamics and the Target Transition

The stochastic realization in equation 21 evolves according to the time convention of equation 3, namely from data at $t = 0$ to noise at $t = 1$. Generation proceeds in the opposite direction, from $t = 1$ to $t = 0$. We therefore derive the reverse-time dynamics used to construct the target transition in equation 10.

Write the forward drift as

$$f(x,t) = v^\star(x,t) + \frac{1}{2} g(t)^2 s_t(x). \tag{26}$$

For a state-independent scalar diffusion coefficient $g(t)$, the standard reverse-time SDE (Song et al., 2020) is

$$dX_t = \left[f(X_t, t) - g(t)^2 s_t(X_t)\right] dt + g(t)\, d\bar{W}_t, \qquad t : 1 \to 0, \tag{27}$$

where the process is integrated with decreasing t. Substituting equation 26 gives

$$dX_t = \left[v^\star(X_t, t) - \frac{1}{2} g(t)^2 s_t(X_t) \right] dt + g(t)\, d\bar{W}_t. \tag{28}$$

Consider the decreasing schedule

$$1 = t_0 > t_1 > \cdots > t_N = 0, \qquad h_k = t_k - t_{k+1} > 0.$$

The increment in the original time variable is

$$t_{k+1} - t_k = -h_k.$$

Applying Euler–Maruyama to equation 28 therefore gives

$$X_{k+1} = X_k - h_k \left[v^\star(X_k, t_k) - \frac{1}{2} g(t_k)^2 s_{t_k}(X_k) \right] + \sqrt{h_k}\, g(t_k) Z_k, \qquad Z_k \sim \mathcal{N}(0, I). \tag{29}$$

The sign of the deterministic term follows directly from the negative time increment $-h_k$.

Conditioned on $X_k = x$, equation 29 is Gaussian:

$$X_{k+1} \mid X_k = x \sim \mathcal{N}\left(\mu_k^\star(x), \Sigma_k\right), \tag{30}$$

with

$$\mu_k^\star(x) = x - h_k v^\star(x, t_k) + \frac{1}{2} h_k g(t_k)^2 s_{t_k}(x), \qquad \Sigma_k = h_k g(t_k)^2 I. \tag{31}$$

Under the ideal-velocity assumption $v_\theta = v^\star$, the score estimate $\hat{s}_t$ in equation 9 equals the exact score s_t, and equation 30 coincides with the target kernel in equation 10.

In practice, Algorithm 1 defines its stochastic target using

$$\mu_k(x) = x - h_k v_\theta(x, t_k) + \frac{1}{2} h_k g(t_k)^2 \hat{s}_{t_k}(x), \qquad \Sigma_k = h_k g(t_k)^2 I, \tag{32}$$

where

$$\hat{s}_t(x) = -\frac{x + (1-t) v_\theta(x, t)}{t}.$$

Thus,

$$p_k(\cdot \mid x) = \mathcal{N}\left(\mu_k(x), \Sigma_k\right) \tag{33}$$

is the discrete target transition that SpecFlow reproduces exactly in Theorem 1. This distinction separates approximation of the ideal continuous-time FM dynamics from the exactness of the speculative mechanism itself.

A.1.3 Recovering the Score from the FM Velocity

We now derive the score–velocity relation used in equation 8. Consider the linear FM interpolation

$$X_t = (1-t) X_0 + t\varepsilon, \qquad X_0 \sim \pi_{\text{data}}, \qquad \varepsilon \sim \mathcal{N}(0, I), \tag{34}$$

with X_0 and ε independent. The ideal marginal FM velocity is

$$v^\star(x, t) = \mathbb{E}\left[\varepsilon - X_0 \mid X_t = x\right]. \tag{35}$$

Conditioned on $X_0 = x_0$, the random variable X_t is Gaussian:

$$X_t \mid X_0 = x_0 \sim \mathcal{N}\left((1-t)x_0, t^2 I\right). \tag{36}$$

Its conditional score is therefore

$$\nabla_x \log p_t(x \mid x_0) = -\frac{x - (1-t)x_0}{t^2}. \tag{37}$$

Using the standard conditional-score identity,

$$\begin{aligned} s_t(x) &= \nabla_x \log p_t(x) && (38)\\ &= \mathbb{E}\left[\nabla_x \log p_t(x \mid X_0) \mid X_t = x\right] && (39)\\ &= -\frac{x - (1-t)\mathbb{E}[X_0 \mid X_t = x]}{t^2}. && (40) \end{aligned}$$

Next, from equation 34,

$$X_t = X_0 + t(\varepsilon - X_0), \tag{41}$$

so

$$\varepsilon - X_0 = \frac{X_t - X_0}{t}. \tag{42}$$

Taking the conditional expectation given $X_t = x$ yields

$$v^\star(x,t) = \frac{x - \mathbb{E}[X_0 \mid X_t = x]}{t}. \tag{43}$$

Therefore,

$$\mathbb{E}[X_0 \mid X_t = x] = x - tv^\star(x,t). \tag{44}$$

Substituting equation 44 into equation 40 gives

$$s_t(x) = -\frac{x - (1-t)\,(x - tv^\star(x,t))}{t^2} \tag{45}$$

$$= -\frac{tx + t(1-t)v^\star(x,t)}{t^2} \tag{46}$$

$$= -\frac{x + (1-t)v^\star(x,t)}{t}, \qquad t \in (0,1]. \tag{47}$$

This is exactly the identity stated in equation 8.

For the learned FM model, we analogously define

$$\hat{s}_t(x) = -\frac{x + (1-t)v_\theta(x,t)}{t}, \tag{48}$$

which is equation 9. Computing $\hat{s}_t$ requires only algebraic operations on the already available network output $v_\theta(x,t)$ and therefore introduces no additional network evaluation.

The identity equation 47 is exact for the ideal marginal velocity $v^\star$. When $v_\theta \neq v^\star$, $\hat{s}_t$ should be understood as the score estimate induced by the learned FM velocity. This modeling approximation is distinct from the speculative approximation: Theorem 1 remains exact with respect to the discrete target kernel equation 33 defined using v_θ and $\hat{s}_t$.

A.1.4 Exactness of SpecFlow (Proof of Theorem 1)

Proof. We prove that, at every sampling step, the next committed state has exactly the target conditional distribution. The proof first establishes the result for a single transition and then shows that the same argument applies sequentially within a speculative block, including when a rejection occurs.

Single-transition correction. Fix a transition k and condition on the information available immediately before its proposal is generated. Let this information be represented by a σ-algebra $\mathcal{F}_k$, which contains the committed history and all drafting information used to construct the proposal distribution at step k. Write

$$p(x) = p_k(x \mid X_k), \qquad q(x) = q_k(x \mid X_k).$$

Conditional on $\mathcal{F}_k$, let $\widetilde{X} \sim q$, and let $U \sim \mathrm{Unif}(0,1)$ be independent of $\widetilde{X}$ and $\mathcal{F}_k$. Define

$$\alpha(x) = \min\left\{1, \frac{p(x)}{q(x)}\right\}.$$

For any measurable set A, the probability mass contributed by accepted proposals is

$$\Pr(\widetilde{X} \in A, \text{accept} \mid \mathcal{F}_k) = \int_A q(x)\alpha(x)\,dx \tag{49}$$

$$= \int_A \min\{p(x), q(x)\}\,dx. \tag{50}$$

Hence the conditional probability of acceptance is

$$a = \int \min\{p(x), q(x)\}\, dx.$$

Because p and q are probability densities,

$$1 - a = \int [p(x) - q(x)]_+\, dx.$$

If $1 - a > 0$, define the residual density

$$r(x) = \frac{[p(x) - q(x)]_+}{\int [p(z) - q(z)]_+\, dz}.$$

If $1 - a = 0$, then $p = q$ almost everywhere and rejection occurs with probability zero.

Let X' denote the next committed state. For every measurable set A,

$$\Pr(X' \in A \mid \mathcal{F}_k) = \Pr(\widetilde{X} \in A, \text{accept} \mid \mathcal{F}_k) + \Pr(\text{reject} \mid \mathcal{F}_k) \Pr(X' \in A \mid \mathcal{F}_k, \text{reject}) \tag{51}$$

$$= \int_A \min\{p(x), q(x)\}\, dx + (1 - a) \int_A r(x)\, dx \tag{52}$$

$$= \int_A \min\{p(x), q(x)\}\, dx + \int_A [p(x) - q(x)]_+\, dx \tag{53}$$

$$= \int_A p(x)\, dx. \tag{54}$$

Therefore,

$$\mathcal{L}(X' \mid \mathcal{F}_k) = p_k(\cdot \mid X_k). \tag{55}$$

Sequential verification within a speculative block. Consider a speculative block beginning from the committed state X_m. The drafter generates

$$\widetilde{X}_{m+1}, \ldots, \widetilde{X}_{m+W}$$

sequentially according to valid draft transitions. Let $U_m, \ldots, U_{m+W-1}$ be independent uniforms used for verification, and let

$$J = \min\left\{j \in \{m, \ldots, m + W - 1\} : U_j > \alpha_j\right\},$$

with $J = \infty$ if no rejection occurs.

The crucial point is that we do not condition on the event that the entire block is accepted. Instead, verification is analyzed one transition at a time.

Suppose the sampler has reached transition j without a previous rejection. Then all proposals before j have been accepted, and therefore

$$X_\ell = \widetilde{X}_\ell, \qquad \ell = m + 1, \ldots, j.$$

In particular, the predecessor used to generate the proposal at transition j is the committed state X_j. Since the draft randomness used to generate the next speculative state is independent of the verification randomness and of the previously committed history conditional on X_j, the conditional proposal law at this transition is precisely

$$\widetilde{X}_{j+1} \mid \mathcal{F}_j \sim q_j(\cdot \mid X_j).$$

Thus the single-transition argument above applies at transition j.

If the proposal is accepted, the state $\widetilde{X}_{j+1}$ is committed. If the proposal is rejected, the residual correction is sampled from $r_j(\cdot \mid X_j)$. In either case,

$$X_{j+1} \mid \mathcal{F}_j \sim p_j(\cdot \mid X_j).$$

This argument applies regardless of what happens at any later speculative position. In particular, after the first rejection at $J = j$, every proposed state $\widetilde{X}_{j+2}, \ldots, \widetilde{X}_{m+W}$ is discarded and cannot

affect the subsequent committed trajectory. The sampler resumes from the corrected state X_{j+1}, to which the same single-transition argument applies for the next block.

If no rejection occurs in the block, the same argument is applied successively to transitions $m, m+1, \ldots, m+W-1$. Importantly, this does not condition on the event $J = \infty$; each transition is analyzed using the information available when that transition becomes the next committed transition.

Consequently, for every committed transition generated by the speculative procedure,

$$\mathcal{L}(X_{k+1} \mid \mathcal{F}_k) = p_k(\cdot \mid X_k). \tag{56}$$

Trajectory-level exactness. It remains to establish the joint law of the complete committed trajectory. The initial state is sampled from the same distribution π_{init} as the target Euler–Maruyama chain. Suppose inductively that the committed history through step k has joint distribution

$$\mathcal{L}(X_0, \ldots, X_k) = \pi_{\text{init}}(dx_0) \prod_{\ell=0}^{k-1} p_\ell(dx_{\ell+1} \mid x_\ell).$$

By equation 56,

$$X_{k+1} \mid \mathcal{F}_k \sim p_k(\cdot \mid X_k).$$

Since the target kernel depends on the committed history only through X_k, this gives

$$X_{k+1} \mid X_k \sim p_k(\cdot \mid X_k).$$

Therefore,

$$\mathcal{L}(X_0, \ldots, X_{k+1}) = \pi_{\text{init}}(dx_0) \prod_{\ell=0}^{k} p_\ell(dx_{\ell+1} \mid x_\ell).$$

Induction over $k = 0, \ldots, N-1$ yields

$$\mathcal{L}(X_0, \ldots, X_N) = \pi_{\text{init}}(dx_0) \prod_{k=0}^{N-1} p_k(dx_{k+1} \mid x_k),$$

which is exactly the joint law of the target Euler–Maruyama chain.

Hence SpecFlow introduces no distributional error relative to the chosen discretized stochastic target, for any valid drafting strategy satisfying the stated measurability and conditional-independence requirements. □

A.1.5 Acceptance Probability for Equal-Covariance Gaussians

We now derive the closed-form acceptance probability in equation 18. Unlike Theorem 1, this result is not required for correctness; rather, it characterizes how draft accuracy determines the efficiency of SpecFlow.

Proposition 1 (Acceptance probability for equal-covariance Gaussians)**.** *Let*

$$p = \mathcal{N}(\mu_p, \Sigma), \qquad q = \mathcal{N}(\mu_q, \Sigma), \qquad \Sigma \succ 0,$$

and define

$$\Delta = \left\| \Sigma^{-1/2}(\mu_p - \mu_q) \right\|_2.$$

Under the exact speculative acceptance rule

$$\alpha(x) = \min\left\{1, \frac{p(x)}{q(x)}\right\}, \qquad X \sim q,$$

the expected acceptance probability is

$$\mathbb{E}_q[\alpha(X)] = 1 - \mathrm{TV}(p, q) = 2\left[1 - \Phi\left(\frac{\Delta}{2}\right)\right], \tag{57}$$

where Φ denotes the standard normal CDF.

Proof. For $X \sim q$,

$$\mathbb{E}_q[\alpha(X)] = \int q(x) \min\left\{1, \frac{p(x)}{q(x)}\right\} dx \tag{58}$$

$$= \int \min\{p(x), q(x)\}\, dx \tag{59}$$

$$= 1 - \mathrm{TV}(p, q). \tag{60}$$

It remains to evaluate the total variation distance. Apply the invertible affine transformation

$$z = \Sigma^{-1/2}\left(x - \frac{\mu_p + \mu_q}{2}\right).$$

Total variation is invariant under invertible transformations. Writing

$$d = \Sigma^{-1/2}(\mu_p - \mu_q), \qquad \|d\|_2 = \Delta,$$

and applying an orthogonal rotation that aligns d with e_1, it suffices to consider

$$P = \mathcal{N}\left(-\frac{\Delta}{2}e_1, I\right), \qquad Q = \mathcal{N}\left(\frac{\Delta}{2}e_1, I\right).$$

Let p_P and p_Q denote their densities. Their log-density ratio is

$$\log \frac{p_P(z)}{p_Q(z)} = -\frac{1}{2}\left\|z + \frac{\Delta}{2}e_1\right\|_2^2 + \frac{1}{2}\left\|z - \frac{\Delta}{2}e_1\right\|_2^2 \tag{61}$$

$$= -\Delta z_1. \tag{62}$$

Hence $p_P(z) > p_Q(z)$ exactly when $z_1 < 0$, and therefore

$$\mathrm{TV}(P, Q) = P(Z_1 < 0) - Q(Z_1 < 0) \tag{63}$$

$$= \Phi\left(\frac{\Delta}{2}\right) - \Phi\left(-\frac{\Delta}{2}\right) \tag{64}$$

$$= 2\Phi\left(\frac{\Delta}{2}\right) - 1. \tag{65}$$

Substitution into equation 60 gives

$$\mathbb{E}_q[\alpha(X)] = 2\left[1 - \Phi\left(\frac{\Delta}{2}\right)\right],$$

which proves the result. □

For SpecFlow,

$$p_k = \mathcal{N}(\mu_k, \Sigma_k), \qquad q_k = \mathcal{N}(\bar{x}_{t_{k+1}}, \Sigma_k), \qquad \Sigma_k = h_k g(t_k)^2 I.$$

Therefore,

$$\Delta_k = \frac{\|\mu_k - \bar{x}_{t_{k+1}}\|_2}{\sqrt{h_k}\, g(t_k)}, \tag{66}$$

and Proposition 1 directly gives equation 18.

Since Φ is monotonically increasing, the expected acceptance probability decreases monotonically with Δ_k. Thus, improving the draft mean increases acceptance and potential speedup, while Theorem 1 guarantees that draft quality does not affect correctness.

A.1.6 Exact Residual Correction for Equal-Covariance Gaussians

Theorem 1 requires an exact draw from the residual density

$$r(x) = \frac{[p(x) - q(x)]_+}{\int [p(z) - q(z)]_+\, dz}$$

when a speculative proposal is rejected. For the isotropic equal-covariance Gaussian transitions used by SpecFlow, this residual can be implemented through a reflection-maximal coupling (De Bortoli et al., 2025). We state the implementation property as a lemma because it directly justifies the correction step in Algorithm 1.

Lemma 1 (Exact reflection correction)**.** *Let*

$$p = \mathcal{N}(\mu_p, \sigma^2 I), \qquad q = \mathcal{N}(\mu_q, \sigma^2 I), \qquad \sigma > 0.$$

Suppose $\widetilde{X} \sim q$ is accepted with probability

$$\min\left\{1, \frac{p(\widetilde{X})}{q(\widetilde{X})}\right\}.$$

When the proposal is rejected, the reflection construction described below produces a corrected state whose conditional density is

$$\frac{[p-q]_+}{\int [p(z)-q(z)]_+ \, dz}.$$

Consequently, the state obtained by either accepting the proposal or applying the reflection correction has marginal distribution p.

Proof. If $\mu_p = \mu_q$, then $p = q$, the acceptance probability is one, and no correction is required. Assume therefore that $\mu_p \neq \mu_q$.

Define

$$\Delta = \frac{\mu_p - \mu_q}{\sigma}, \qquad e = \frac{\Delta}{\|\Delta\|_2}. \tag{67}$$

Generate the proposal as

$$\widetilde{X} = \mu_q + \sigma Z, \qquad Z \sim \mathcal{N}(0, I), \tag{68}$$

and let $U \sim \mathrm{Unif}(0,1)$ be independent of Z. Accept when

$$U \leq \min\left\{1, \frac{p(\widetilde{X})}{q(\widetilde{X})}\right\}. \tag{69}$$

If rejection occurs, define the reflection

$$R_e(z) = z - 2(e^\top z)e \tag{70}$$

and return

$$X = \mu_p + \sigma R_e(Z). \tag{71}$$

Let ϕ denote the density of $\mathcal{N}(0, I)$. Because

$$\widetilde{X} = \mu_q + \sigma Z = \mu_p + \sigma(Z - \Delta),$$

the likelihood ratio is

$$\frac{p(\widetilde{X})}{q(\widetilde{X})} = \frac{\phi(Z-\Delta)}{\phi(Z)}. \tag{72}$$

The unconditional density contributed by accepted proposals is therefore

$$q(x)\min\left\{1, \frac{p(x)}{q(x)}\right\} = \min\{p(x), q(x)\}. \tag{73}$$

It remains to identify the density contributed by the rejection branch. The reflection matrix is

$$R_e = I - 2ee^\top.$$

It satisfies

$$R_e^\top R_e = I, \qquad R_e^{-1} = R_e, \qquad |\det R_e| = 1.$$

Hence it preserves Euclidean norm, volume, and the standard Gaussian density:

$$\phi(R_e z) = \phi(z). \tag{74}$$

Since Δ is parallel to e,

$$R_e \Delta = -\Delta. \tag{75}$$

For an output x on the rejection branch, define

$$w = \frac{x - \mu_p}{\sigma}.$$

By equation 71,

$$w = R_e z, \qquad z = R_e w.$$

The unconditional rejected mass in the proposal coordinate z is

$$\phi(z)\left[1 - \min\left\{1, \frac{\phi(z-\Delta)}{\phi(z)}\right\}\right] = [\phi(z) - \phi(z-\Delta)]_+. \tag{76}$$

Substituting $z = R_e w$ and using equation 74–equation 75,

$$[\phi(R_e w) - \phi(R_e w - \Delta)]_+ = [\phi(w) - \phi(w+\Delta)]_+. \tag{77}$$

For $x = \mu_p + \sigma w$,

$$p(x) = \sigma^{-d}\phi(w). \tag{78}$$

Moreover,

$$x - \mu_q = \mu_p - \mu_q + \sigma w = \sigma(w + \Delta),$$

and hence

$$q(x) = \sigma^{-d}\phi(w + \Delta). \tag{79}$$

Since the affine transformation from w to x contributes the Jacobian σ^{-d}, equation 77 implies that the unconditional density of the rejection output is exactly

$$[p(x) - q(x)]_+. \tag{80}$$

Combining equation 73 and equation 80, the complete output density is

$$\min\{p(x), q(x)\} + [p(x) - q(x)]_+ = p(x).$$

Therefore $X \sim p$.

Finally, the total rejection probability is

$$\int [p(x) - q(x)]_+ \, dx.$$

Since equation 80 is the unconditional density of the rejection output, conditioning on rejection gives

$$\mathcal{L}(X \mid \text{reject}) = \frac{[p - q]_+}{\int [p(z) - q(z)]_+ \, dz}, \tag{81}$$

which is precisely the residual distribution required by equation 14. □

For SpecFlow,

$$\mu_p = \mu_k, \qquad \mu_q = \bar{x}_{t_{k+1}}, \qquad \sigma^2 = h_k g(t_k)^2. \tag{82}$$

Lemma 1 therefore provides the exact correction used after a rejection in Algorithm 1. Accepted proposals are left unchanged. When $\mu_k = \bar{x}_{t_{k+1}}$, the two transition kernels coincide and the acceptance probability is one.

A.1.7 Adaptive Drafting Details

We finally specify the adaptive drafting rule used in Section 3.3.2. The drafter uses only velocity evaluations associated with already committed states and therefore introduces no additional FM network evaluations.

Let

$$v_i = v_\theta(x_{t_i}, t_i)$$

denote the stored velocity at a committed state. At the current committed index m, the degree-r Newton interpolant through the most recent $r+1$ available velocities is

$$\widehat{v}_m^{(r)}(t) = \sum_{j=0}^{r} v[t_m, \ldots, t_{m-j}] \prod_{\ell=0}^{j-1} (t - t_{m-\ell}), \tag{83}$$

where $v[\cdot]$ denotes the corresponding vector-valued divided difference.

For example,

$$v[t_m] = v_m,$$

and

$$v[t_m, t_{m-1}] = \frac{v_m - v_{m-1}}{t_m - t_{m-1}}.$$

Higher-order coefficients are defined recursively by

$$v[t_i, \dots, t_j] = \frac{v[t_i, \dots, t_{j+1}] - v[t_{i-1}, \dots, t_j]}{t_i - t_j}. \tag{84}$$

The corresponding deterministic prediction of a future state is

$$\bar{x}_{t_k} = x_{t_m} - \int_{t_k}^{t_m} \widehat{v}_m^{(r)}(\tau)\, d\tau, \qquad k > m, \tag{85}$$

which matches equation 16. Because $\widehat{v}_m^{(r)}$ is polynomial in time, the integral is available in closed form.

For $r = 0$,

$$\widehat{v}_m^{(0)}(t) = v_m,$$

and therefore

$$\bar{x}_{t_k} = x_{t_m} - (t_m - t_k) v_m. \tag{86}$$

Thus the constant-velocity draft is recovered as a special case. For $r \geq 1$, the higher-order divided differences capture recent temporal variation of the velocity and therefore allow the predictor to account for local trajectory curvature.

To select the order and speculative horizon, we measure the relative magnitude of the highest available divided difference. For $r \geq 1$, define

$$\kappa_m^{(r)} = \frac{\|v[t_m, \dots, t_{m-r}]\|_2}{\epsilon + \|v[t_m, \dots, t_{m-r+1}]\|_2}, \qquad \epsilon = 10^{-8}. \tag{87}$$

A small value indicates that the newly introduced higher-order term is small relative to the lower-order variation, suggesting locally smooth velocity evolution. A larger value indicates stronger high-order variation and therefore less reliable long-range extrapolation.

Using a maximum order $r_{\max} = 2$, the default adaptive rule considered in our implementation is

$$(r_m, W_m) = \begin{cases} (2, W_{\max}), & \kappa_m^{(2)} < \tau_1, \\ (1, \lfloor W_{\max}/2 \rfloor), & \tau_1 \leq \kappa_m^{(2)} < \tau_2, \\ (0, \max\{1, \lfloor W_{\max}/4 \rfloor\}), & \kappa_m^{(2)} \geq \tau_2. \end{cases} \tag{88}$$

The thresholds τ_1 and τ_2 are fixed hyperparameters selected prior to evaluation. During inference, the drafting order and speculative horizon are determined solely from the committed velocity history through the normalized divided-difference statistic $\kappa_m^{(2)}$, requiring no additional model evaluations or online optimization. The threshold values used in our experiments are reported in Appendix B.7. When fewer than three committed velocity evaluations are available, we use the highest order supported by the available history. The speculative horizon is additionally clipped to the number of remaining transitions,

$$W_m \leftarrow \min\{W_m, N - m\}. \tag{89}$$

Crucially, (r_m, W_m) is selected entirely from the committed history before the target evaluation of the current speculative block. The adaptive rule therefore changes only the conditional draft distribution q_k; it does not inspect future target evaluations or alter the target transition p_k. Consequently, adaptive drafting remains within the assumptions of Theorem 1: changing the order or horizon may affect acceptance and speedup, but not the distribution of the committed chain.

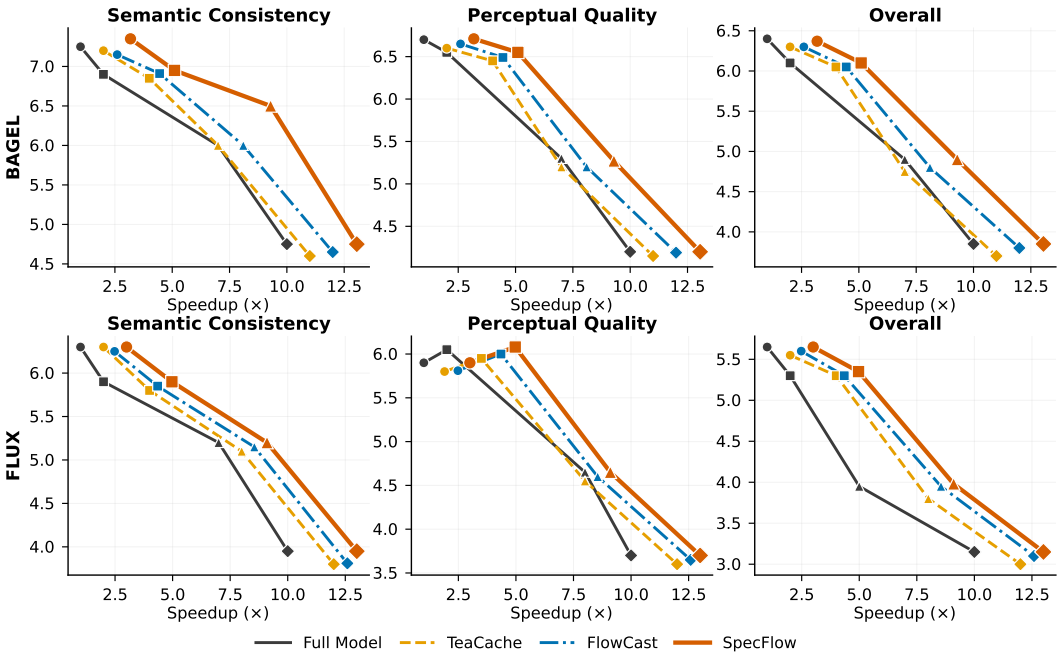

Figure 1: **Image editing on GEdit.** Quality–speedup trade-offs for BAGEL and FLUX-Kontext across semantic consistency (G_{SC}), perceptual quality (G_{PQ}), and overall editing quality (G_{O}).

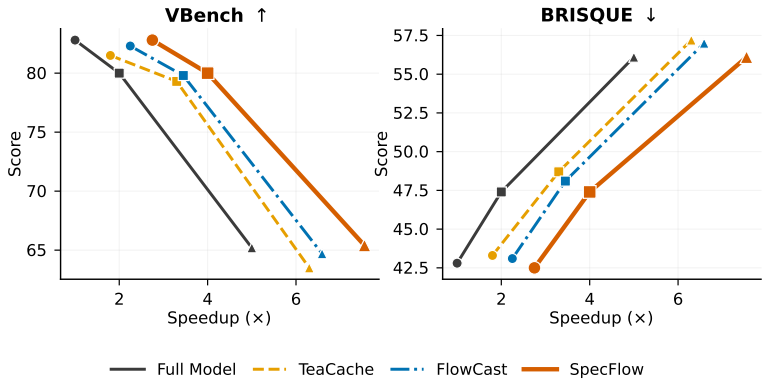

Figure 2: **Video generation on HunyuanVideo.** VBench and BRISQUE quality as a function of end-to-end speedup.

A.2 Distributional Fidelity

The losslessness guarantee of SpecFlow follows from Theorem 1 and is not specific to the analytic model considered below. The theorem applies to the discrete stochastic target implemented by each backbone, provided that the draft density, acceptance rule, and residual correction follow Algorithm 1; it does not depend on the dimensionality of the state space or require access to the ideal velocity $v^\star$. A separate question is whether this stochastic target closely approximates the deterministic finite-step FM sampler used in practice. We therefore evaluate these two properties separately.

Conventional perceptual metrics cannot establish distribution preservation: two samplers may obtain similar perceptual scores while inducing different output distributions. We therefore complement task-level evaluation with direct tests of distributional agreement using an MMD^2 permutation test. Independently sampled target sets provide a finite-sample calibration control.

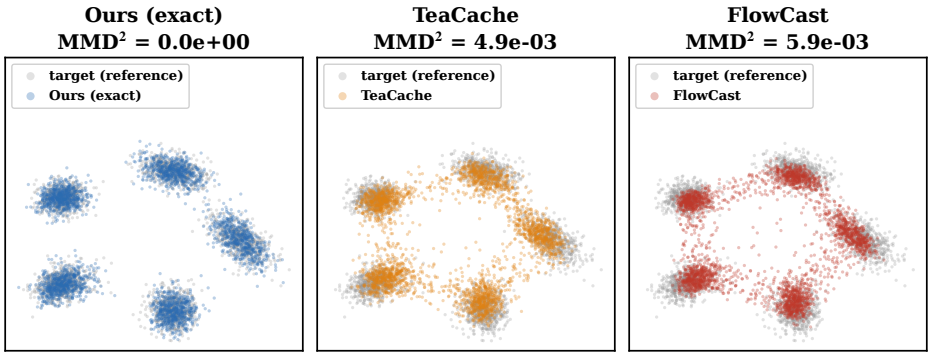

Figure 3: **Distributional fidelity across acceleration methods.** SpecFlow closely matches the target stochastic sampler, with MMD^2 near the finite-sample null level, whereas TeaCache and FlowCast exhibit visible distributional deviations and substantially larger MMD^2. This empirical agreement is consistent with the exactness guarantee of Theorem 1.

Comparison	MMD^2 (mean $\pm$ std)	Rejects / 20
Target A vs. Target B	$-0.2\times10^{-4} \pm 3.1\times10^{-4}$	$1/20$
Deterministic vs. Stochastic	$0.3\times10^{-4} \pm 3.4\times10^{-4}$	$1/20$
Target vs. SpecFlow	$0.1\times10^{-4} \pm 3.3\times10^{-4}$	$1/20$
Target vs. TeaCache	$3.9\times10^{-3} \pm 4.2\times10^{-4}$	$18/20$
Target vs. FlowCast	$4.7\times10^{-3} \pm 4.5\times10^{-4}$	$19/20$

Table 2: **Real-model distributional validation on BAGEL.** The Deterministic–Stochastic comparison evaluates the finite-step target choice, while Target–SpecFlow evaluates speculative execution relative to that stochastic target. Target A vs. Target B provides the finite-sample calibration control. We report mean MMD^2 and the number of 20 independent permutation tests rejecting equality at $\alpha = 0.05$.

Real-model validation. We first evaluate BAGEL under the same checkpoint, prompts, timestep schedule, and sampling budget. The Deterministic–Stochastic comparison measures the finite-step discrepancy introduced by replacing the standard deterministic FM sampler with the serial stochastic target. Target–SpecFlow instead evaluates speculative execution relative to that same stochastic target. Target A vs. Target B compares independent draws from the serial stochastic target and provides the empirical null calibration. For context, we apply the same test to TeaCache and FlowCast.

The Deterministic–Stochastic comparison remains at the empirical null scale: its mean MMD^2 is 0.3×10^{-4}, compared with -0.2×10^{-4} for the Target–Target calibration, and both yield only $1/20$ rejections. Thus, at the evaluated BAGEL discretization, we detect no distributional discrepancy between the deterministic sampler and the chosen stochastic target beyond the finite-sample calibration level. Likewise, Target–SpecFlow yields a near-zero MMD^2 of 0.1×10^{-4} and $1/20$ rejections, matching the target calibration. In contrast, TeaCache and FlowCast exhibit substantially larger discrepancies and are rejected in $18/20$ and $19/20$ trials, respectively.

These comparisons address two distinct questions. Agreement between the deterministic sampler and stochastic target is an empirical property of the chosen finite-step discretization and is not implied by Theorem 1. Exactness of speculative execution with respect to the stochastic target, by contrast, follows from Theorem 1; the Target–SpecFlow comparison provides a real-backbone implementation check of this guarantee. Failure to reject in these finite-sample tests should therefore be interpreted as empirical consistency rather than proof of distributional equality.

Controlled analytic validation. We additionally evaluate an analytic Gaussian-mixture flow model, where the ideal velocity $v^\star$, target density, and sample log-likelihood are available explicitly. This setting allows us to separate finite-step target error, learned-velocity approximation, and speculative error under controlled conditions.

Comparison	MMD^2 (mean $\pm$ std)	Rejects / 20
Target vs. Target	$-1.4\times10^{-5} \pm 3.0\times10^{-4}$	$0/20$
Target vs. TeaCache	$4.8\times10^{-3} \pm 3.1\times10^{-4}$	$20/20$
Target vs. FlowCast	$5.6\times10^{-3} \pm 3.6\times10^{-4}$	$20/20$
Target vs. SpecFlow	$-5.1\times10^{-5} \pm 2.9\times10^{-4}$	$0/20$

Table 3: **Controlled distributional validation on the analytic Gaussian-mixture model.** We report the unbiased MMD^2 estimator over 20 independent trials; negative values can occur under the finite-sample null. The Target–Target comparison provides the empirical null calibration.

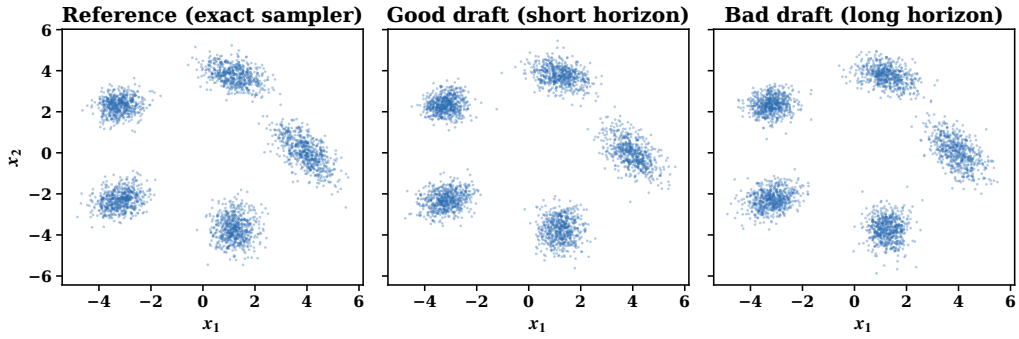

Figure 4: **Distributional invariance to draft quality.** Despite substantially different drafting regimes, both accurate short-horizon and inaccurate long-horizon drafts recover the same target distribution after exact verification and correction. Thus, draft quality affects acceptance and efficiency, but not the committed output distribution, as established by Theorem 1.

The controlled experiment mirrors the real-model speculative comparison: Target–SpecFlow remains at the Target–Target finite-sample null scale, with $0/20$ rejections, whereas TeaCache and FlowCast are rejected in all 20 trials. The analytic setting additionally exposes $v^\star$ and the data density, allowing Appendices B.2 and B.4 to examine finite-step target error and learned-velocity approximation directly. Together, the BAGEL and analytic experiments provide complementary implementation checks, while exact distribution preservation with respect to the chosen stochastic target follows from Theorem 1.

A.3 Acceptance and Speedup

The analysis in Section 3.3.3 predicts that acceptance is governed by the normalized discrepancy

$$\Delta_k = \frac{\|\bar{x}_{t_{k+1}} - \mu_k\|_2}{\sqrt{h_k}\, g(t_k)}.$$

We therefore record the empirical acceptance probability throughout the sampling trajectory and compare it with the theoretical relation in equation 18. This experiment directly connects the theory to observed acceleration.

We report: (i) average acceptance rate, (ii) accepted speculative steps per verification pass, (iii) number of sequential verification passes, and (iv) end-to-end speedup. We additionally visualize acceptance as a function of sampling time to identify regions in which the FM trajectory is particularly easy or difficult to predict.

B Ablation Studies

Drafting order. We compare constant-velocity drafting ($r = 0$) against first- and second-order velocity extrapolation. This isolates the benefit of modeling local trajectory curvature independently of the exact verification mechanism.

Method	SO	TO	CO	CL	CA	PO	Overall ↑	CLIPIQA ↑	Spd. ↑
Full Model									
BAGEL-50	0.99	0.90	0.81	0.85	0.59	0.54	**0.78**	**0.84**	1.0×
BAGEL-25	0.99	0.90	0.78	0.82	0.62	0.51	0.77	0.82	2.0×
BAGEL-10	0.99	0.88	0.68	0.84	0.56	0.48	0.73	0.75	5.0×
BAGEL-5	0.93	0.62	0.55	0.72	0.25	0.36	0.57	0.39	10×
Speculative Decoding (Ours)									
SpecFlow-50	0.99	0.90	0.81	0.85	0.60	0.54	**0.78**	**0.85**	2.52×
SpecFlow-25	0.99	0.91	0.77	0.83	0.60	0.51	0.77	0.82	4.45×
SpecFlow-10	0.99	0.89	0.67	0.84	0.55	0.48	0.73	0.76	7.19×
SpecFlow-5	0.93	0.60	0.57	0.73	0.24	0.36	0.57	0.39	12.05×

Table 4: **Results on the BAGEL model for image generation.** We report GenEval scores (SO: Single Object, TO: Two Objects, CO: Counting, CL: Color, CA: Category, PO: Position), overall score, CLIPIQA, and Speedup (Spd.). Speculative decoding achieves strong speedup while retaining quality close to the full model, outperforming static baselines.

Method	SO	TO	CO	CL	CA	PO	Overall ↑	CLIPIQA ↑	Spd. ↑
Full Model									
FLUX-50	0.97	0.79	0.73	0.78	0.44	0.20	**0.65**	**0.83**	1.0×
FLUX-25	0.97	0.78	0.71	0.75	0.43	0.18	0.64	0.80	2.0×
FLUX-10	0.97	0.65	0.58	0.67	0.41	0.15	0.57	0.59	5.0×
FLUX-5	0.88	0.43	0.47	0.54	0.23	0.08	0.44	0.43	10×
Speculative Decoding (Ours)									
SpecFlow-50	0.97	0.79	0.74	0.77	0.45	0.19	**0.65**	**0.82**	2.59×
SpecFlow-25	0.97	0.78	0.72	0.75	0.41	0.18	0.63	0.80	4.57×
SpecFlow-10	0.97	0.65	0.57	0.67	0.42	0.15	0.58	0.60	7.11×
SpecFlow-5	0.87	0.43	0.48	0.53	0.23	0.08	0.45	0.43	12.89×

Table 5: **Results on the FLUX model for image generation.** We report GenEval scores (SO: Single Object, TO: Two Objects, CO: Counting, CL: Color, CA: Category, PO: Position), overall score, CLIPIQA, and Speedup (Spd.). Speculative decoding achieves substantial speedups while preserving performance close to the full model, surpassing static baselines.

Adaptive horizon. We compare a fixed speculative horizon against the adaptive strategy in Section 3.3.2. We report acceptance rate, verification passes, and wall-clock speedup to determine whether longer drafts are beneficial primarily in locally smooth regions.

Stochasticity schedule. The diffusion schedule $g(t)$ controls the variance of the stochastic target transition and therefore affects the normalized target–draft discrepancy in equation 18. We vary the overall scale of $g(t)$ and measure its effect on acceptance, speedup, and generation quality.

Correction. We compare the complete method against a variant that accepts/rejects proposals but omits exact residual correction. This ablation is not an alternative sampler we advocate; rather, it demonstrates empirically why the correction step is necessary for distributional preservation.

B.1 Implementation Details

All experiments use the released pretrained checkpoints and identical hardware, precision, batch size, resolution, and conditioning across methods. The target and draft evaluations within a speculative block are batched whenever possible. The score estimate is computed algebraically from the already evaluated FM velocity and therefore requires no additional network call. Hyperparameters for the adaptive drafting order, speculative horizon, and diffusion schedule are fixed per model and are reported with the corresponding experimental configuration.

N	**Sampler**	$\log p_0(x) \uparrow$	**Rej. vs. Det.**	**Rej. vs. Stoch.**	**Speedup** $\uparrow$
50	Deterministic FM	-3.040	–	$0/10$	$1.00\times$
	Serial stochastic	-3.040	$0/10$	–	$1.00\times$
	SpecFlow	$\mathbf{-3.022}$	$0/10$	$\mathbf{0/10}$	$\mathbf{2.54}\times$
25	Deterministic FM	-3.062	–	$0/10$	$1.00\times$
	Serial stochastic	-3.054	$0/10$	–	$1.00\times$
	SpecFlow	$\mathbf{-3.053}$	$0/10$	$\mathbf{0/10}$	$\mathbf{2.21}\times$
10	Deterministic FM	-3.187	–	$1/10$	$1.00\times$
	Serial stochastic	-3.097	$1/10$	–	$1.00\times$
	SpecFlow	$\mathbf{-3.114}$	$0/10$	$\mathbf{0/10}$	$\mathbf{2.05}\times$

Table 6: **Deterministic FM, serial stochastic sampling, and SpecFlow.** Results on the analytic Gaussian-mixture flow model over 10 independent seeds for three step budgets, with $g = 1.0$. Quality is measured by mean data log-likelihood $\log p_0(x)$. "Rej." denotes the number of trials (out of 10) for which the MMD^2 permutation test rejects equality at $\alpha = 0.05$. A Det.-vs.-Det.$'$ control yields $0/10$ rejections for every N. The Det.-vs.-Stoch. comparison measures the finite-step effect of adopting the stochastic realization, whereas the Stoch.-vs.-SpecFlow comparison tests whether speculative acceleration introduces additional distributional discrepancy.

B.2 Deterministic FM and Stochastic Target Comparison

The exactness guarantee of SpecFlow is defined with respect to the chosen discretized stochastic target in equation 10, rather than the deterministic finite-step FM sampler. We therefore separately evaluate **(i)** the finite-step effect of replacing deterministic FM integration with the stochastic realization and **(ii)** whether speculative execution introduces any additional detectable distributional discrepancy.

We consider an analytic Gaussian-mixture flow model, for which the data density and hence the sample log-likelihood can be evaluated directly. At each step budget $N \in \{50, 25, 10\}$, we compare three samplers under the same time discretization: *Deterministic FM*, which integrates the probability-flow ODE using the standard deterministic update; *Serial stochastic*, which samples the Euler–Maruyama target in equation 10 sequentially without speculation; and SpecFlow, which accelerates exactly the same stochastic target using draft–verify–correct sampling. We use $g = 1.0$ for the stochastic samplers.

Distributional agreement is evaluated using an MMD^2 permutation test at significance level $\alpha = 0.05$ over 10 independent seeds. We report the number of trials in which equality is rejected. The Deterministic-vs.-Stochastic comparison measures the finite-step effect of introducing the stochastic realization, while the Stochastic-vs.-SpecFlow comparison isolates the effect of speculative acceleration. As a calibration control, two independent deterministic sample sets are compared using the same procedure.

Results. At $N = 50$ and $N = 25$, the MMD test does not distinguish the serial stochastic target from deterministic FM in any of the 10 trials, while their mean log-likelihoods are also closely matched. At the coarser $N = 10$ discretization, the deterministic and stochastic samplers are distinguished in one of the ten trials, indicating that finite-step effects become more visible as the discretization becomes more aggressive.

In contrast, SpecFlow is not distinguished from its serial stochastic target in any trial at any of the three step budgets. This behavior is consistent with Theorem 1: speculative execution changes the computational procedure but not the transition law of the chosen stochastic target. At the same time, SpecFlow provides $2.54\times$, $2.21\times$, and $2.05\times$ end-to-end speedup at $N = 50$, 25, and 10, respectively.

These results separate two effects that are otherwise easy to conflate. The difference between deterministic FM and serial stochastic sampling is attributable to the choice and finite-step discretization of the stochastic target, whereas the comparison between serial stochastic sampling and SpecFlow

Method	MMD^2 vs. Target	Rejects	$\log p_0(x) \uparrow$	Speedup $\uparrow$
Target vs. Target	$-3.3\times10^{-4} \pm 2.1\times10^{-4}$	0/20	-3.055	–
SpecFlow	$\mathbf{-3.9\times10^{-5} \pm 4.3\times10^{-4}}$	**0/20**	$\mathbf{-3.045}$	$\mathbf{2.20\times}$
SpecFlow w/o correction	$2.2\times10^{-4} \pm 5.7\times10^{-4}$	1/20	-5.576	$1.95\times$

Table 7: **Effect of residual correction.** Results on the analytic Gaussian-mixture model with $N = 25$ and $g = 1.0$ over 20 independent seeds. Full SpecFlow remains at the Target-vs.-Target empirical null level. Removing residual correction substantially reduces data log-likelihood, despite producing only a weak signal under the median-bandwidth MMD test, demonstrating that correction is important for preserving the target distribution in this setting.

isolates the additional effect of speculation. Empirically, the latter introduces no detectable distributional discrepancy under this test.

B.3 Effect of Residual Correction

The residual correction in equation 14 restores the target probability mass after a rejected proposal and is therefore essential to the exactness argument in Theorem 1. To isolate its empirical effect, we compare the complete SpecFlow sampler against an otherwise identical variant that removes residual correction. Upon rejection, this variant commits the rejected draft proposal instead of drawing from the residual distribution; all other components, including drafting, target evaluation, and the acceptance rule, are unchanged.

We evaluate both variants on the analytic Gaussian-mixture model with $N = 25$, $g = 1.0$, and 20 independent seeds. Distributional agreement with the serial stochastic target is measured using the MMD^2 permutation test at $\alpha = 0.05$. Since the analytic data density is available, we additionally report mean data log-likelihood.

Full SpecFlow closely matches the stochastic target under both diagnostics: the MMD test yields $0/20$ rejections and the mean log-likelihood is nearly identical to the target control. Removing residual correction produces a substantial decrease in log-likelihood, from approximately -3.05 to -5.58, showing that rejected proposals can move samples into low-density regions when the missing target mass is not restored.

The corresponding MMD test rejects only $1/20$ trials for the uncorrected variant. We report this result without interpreting non-rejection as evidence of equality: in this experiment, the median-bandwidth MMD test is less sensitive to the low-density behavior exposed by the analytic log-likelihood. Together, the results empirically illustrate the role of residual correction, while the exact distribution-preservation guarantee itself follows from Theorem 1.

B.4 Sensitivity to Learned-Velocity Error

The marginal-preservation argument in Section 3.3 is exact for the ideal marginal velocity $v^\star$, whereas the practical target in equation 10 is constructed from the learned velocity v_θ and its induced score estimate $\hat{s}_t$. We therefore isolate the effect of velocity approximation error from the exactness of speculative sampling using the analytic Gaussian-mixture model, where $v^\star$ and s_t are available exactly.

We construct approximate velocity fields with controlled relative error

$$E_v = \frac{\mathbb{E}[\|v_\theta(x,t) - v^\star(x,t)\|_2]}{\mathbb{E}[\|v^\star(x,t)\|_2]}, \tag{90}$$

and obtain the corresponding score estimate from equation 9. We additionally measure the relative score error

$$E_s = \frac{\mathbb{E}[\|\hat{s}_t(x) - s_t(x)\|_2]}{\mathbb{E}[\|s_t(x)\|_2]}. \tag{91}$$

For each error level, we compare the ideal stochastic target constructed from $(v^\star, s_t)$, the approximate stochastic target constructed from $(v_\theta, \hat{s}_t)$, and SpecFlow applied to that same approximate

E_v	E_s	**Ideal–Approx. Rejects**	**Approx.–SpecFlow Rejects**	$\log p_0$ **(Approx.)**	$\log p_0$ **(SpecFlow)**
0.00	0.00	0/12	**0/12**	-3.050	-3.049
0.01	0.10	0/12	**0/12**	-3.059	-3.063
0.05	0.50	0/12	**0/12**	-3.317	-3.333
0.10	1.00	12/12	**0/12**	-4.174	-4.210
0.20	2.00	12/12	**0/12**	-8.194	-8.301

Table 8: **Sensitivity to learned-velocity error.** Controlled perturbations of the ideal velocity on the analytic Gaussian-mixture model. E_v and E_s denote relative velocity and score errors, respectively. "Rejects" is the number of MMD^2 permutation tests (out of 12) rejecting equality at $\alpha = 0.05$. Increasing velocity error eventually changes the induced stochastic target substantially, whereas SpecFlow remains indistinguishable from the corresponding approximate target in all tested trials.

target. Distributional agreement is evaluated over 12 independent trials using the same MMD^2 permutation test as in Section A.2.

The results separate modeling error from speculative error. For small velocity errors ($E_v \leq 0.05$), the MMD test does not distinguish the approximate stochastic target from the ideal target, although its data log-likelihood begins to decrease. At $E_v \geq 0.10$, the ideal and approximate targets are separated in all 12 trials and the log-likelihood deteriorates substantially. Thus, approximation of $v^\star$ by v_θ can alter the stochastic target itself.

Crucially, this effect is distinct from speculative acceleration. Across every tested velocity-error level, including regimes in which the approximate target differs substantially from the ideal target, the Approx.–SpecFlow comparison yields $0/12$ rejections and closely matched log-likelihoods. This is consistent with Theorem 1: SpecFlow exactly targets the transition kernel constructed from the available v_θ and $\hat{s}_t$, independently of their approximation error relative to the ideal flow.

The score error also grows more rapidly than the velocity error in this experiment, consistent with the time-dependent amplification in the score–velocity relation equation 8. This observation further motivates distinguishing approximation of the ideal stochastic realization from the exactness of the speculative sampling procedure.

B.5 Sensitivity to the Diffusion Scale

The diffusion scale $g(t)$ determines the covariance of the stochastic target transition and also enters the normalized draft–target discrepancy in equation 18. We therefore examine whether the exactness and efficiency of SpecFlow depend sensitively on this choice. Using the analytic Gaussian-mixture model with $N = 25$, we vary a constant diffusion scale $g \in \{0.25, 0.5, 1.0, 2.0, 3.0\}$ while keeping the remaining sampler and drafting configuration fixed. Results are averaged over 10 independent seeds.

For each setting, we compare two independently sampled deterministic runs (Det.–Det.$'$), deterministic FM against the serial stochastic target (Det.–Stoch.), and the serial stochastic target against SpecFlow (Stoch.–SpecFlow). Distributional agreement is measured using an MMD^2 permutation test at significance level $\alpha = 0.05$. We additionally report the mean data log-likelihood under the analytic target, speculative acceptance rate, and end-to-end speedup.

Across the tested range, the Stoch.–SpecFlow comparison remains at the empirical null level: no rejection occurs for $g \leq 2$, while the single rejection at $g = 3$ is consistent with the nominal significance level. This supports the predicted independence of the exact speculative correction from the particular diffusion scale.

Interestingly, increasing g does not improve speculative efficiency in this setting. Acceptance changes only moderately from 0.733 to 0.655, and end-to-end speedup remains between $2.03\times$ and $2.21\times$. Thus, the observed acceleration is not obtained by increasing the stochastic variance to make verification artificially easier.

g	Det.–Det.′	Det.–Stoch.	Stoch.–SpecFlow	$\log p_0(x) \uparrow$	Accept. ↑	Speedup ↑
0.25	0/10	1/10	**0/10**	-3.043	0.733	$2.03\times$
0.50	0/10	1/10	**0/10**	-3.037	0.718	$2.18\times$
1.00	0/10	0/10	**0/10**	-3.054	0.690	$\mathbf{2.21}\times$
2.00	0/10	0/10	**0/10**	-3.242	0.693	$2.20\times$
3.00	0/10	0/10	**1/10**	-3.902	0.655	$2.15\times$

Table 9: **Sensitivity to the diffusion scale g.** Results on the analytic Gaussian-mixture model with $N = 25$ over 10 independent seeds. The first three columns report the number of MMD^2 permutation tests (out of 10) rejecting equality at $\alpha = 0.05$. Across a broad range of diffusion scales, SpecFlow remains statistically consistent with its serial stochastic target while acceptance and speedup vary only moderately. Large diffusion scales, however, reduce the likelihood under the analytic data distribution, indicating degradation of the finite-step stochastic target itself.

Method	Latency (s) ↓	Model Evals	Speedup ↑
Full-50	16.57	50.0	$1.00\times$
TeaCache	9.21	27.8	$1.80\times$
FastFlow	7.71	23.2	$2.15\times$
FlowCast-50	7.05	21.2	$2.35\times$
SpecFlow-50	**6.58**	19.8	$\mathbf{2.52}\times$

Table 10: **End-to-end wall-clock speedup on BAGEL.** Results use the $N = 50$ configuration at 1024×1024 resolution on a single NVIDIA H100 80GB GPU. Latency includes the complete inference cost, including batched verification and correction for SpecFlow. Speedup is measured relative to the Full-50 sampler and is not inferred from model-evaluation counts.

The primary effect of excessive stochasticity is instead on the finite-step target itself. Mean data log-likelihood remains nearly unchanged for $g \leq 1$, but deteriorates for $g = 2$ and substantially for $g = 3$. These results motivate using a moderate diffusion scale: SpecFlow remains exact with respect to the chosen stochastic target across the tested range, while overly large values of g can reduce the quality of that target under finite-step discretization.

B.6 Wall-Clock Speedup Analysis

To verify that the acceleration of SpecFlow translates into realized inference speedup rather than only a reduction in nominal model evaluations, we profile end-to-end sampling latency on BAGEL at 1024×1024 resolution. All methods use the same $N = 50$ sampling configuration and are evaluated on a single NVIDIA H100 80GB GPU under identical precision, batch size, and conditioning. We report the mean over 10 runs. Timing includes all method-specific computation; for SpecFlow, this includes drafting, batched verification, acceptance testing, and residual correction.

As shown in Table 10, the reduction in target-model evaluations translates into substantial end-to-end acceleration. SpecFlow reduces the average number of model evaluations from 50.0 to 19.8 and lowers sampling latency from 16.57 s to 6.58 s, corresponding to a measured $2.52\times$ wall-clock speedup. Under the same setup, TeaCache, FastFlow, and FlowCast achieve $1.80\times$, $2.15\times$, and $2.35\times$ speedup, respectively.

Importantly, these speedups are measured directly from end-to-end runtime rather than inferred from model-evaluation counts. The reported latency includes all method-specific computation; for SpecFlow, this includes drafting, batched target verification, acceptance testing, rejection handling, and residual correction.

Compute accounting: We distinguish between *target states evaluated* and *sequential target forward passes*. A speculative block of W states evaluates all W target states in one batched forward pass; a *verification pass* denotes this batched forward pass, whereas target-state evaluations count all states processed within it. Wall-clock latency therefore captures both the computation performed within each batch and the reduction in sequential execution depth.

Configuration	BAGEL-7B-MoT	FLUX.1-dev	FLUX.1-Kontext-dev	HunyuanVideo
Model and task				
Checkpoint	BAGEL-7B-MoT	FLUX.1-dev	FLUX.1-Kontext-dev	HunyuanVideo
Task	Text-to-image / editing	Text-to-image	Image editing	Text-to-video
Benchmark	GenEval / GEdit	GenEval	GEdit	VBench
Evaluation prompts	553 / 606	553	606	80
Seeds / prompt	4	4	1	1
Base sampling configuration				
Sampling steps N	50	50	50	50
Solver	MultiModalEuler	FlowMatchEuler	FlowMatchEuler	Rectified-flow Euler
Timestep schedule	Shifted	Shifted sigmas	Shifted sigmas	Flow-shifted
Schedule shift	3.0	Model default	Model default	7.0
Guidance	4.0	3.5	2.5	6.0
Precision	BF16	BF16	BF16	BF16
Prompt batch	1	1	1	1
Resolution	1024^2	512^2	1024^2	1280×720
Frames	1	1	1	129 (24 fps)
SpecFlow configuration				
Diffusion schedule $g(t)$	1.0	1.0	1.0	1.0
Maximum horizon $W_{\max}$	5	5	5	5
Maximum order $r_{\max}$	2	2	2	2
Adaptive threshold τ_1	0.10	0.10	0.10	0.10
Adaptive threshold τ_2	0.30	0.30	0.30	0.30
Stabilizer ϵ	10^{-8}	10^{-8}	10^{-8}	10^{-8}
Draft covariance	$h_k g(t_k)^2 I$	$h_k g(t_k)^2 I$	$h_k g(t_k)^2 I$	$h_k g(t_k)^2 I$
Score network	None	None	None	None
Additional draft NFE	0	0	0	0
Implementation and hardware				
Network output $\rightarrow v_\theta$	Identity	Identity	Identity	Identity
Hardware	1× H100 80GB	1× H100 80GB	1× H100 80GB	1× H100 80GB
Parallelism	Single GPU	Single GPU	Single GPU	Single GPU
Verification	Batched	Batched	Batched	Batched

Table 11: **Model, inference, and SpecFlow configurations.** We report the backbone, evaluation protocol, base sampling configuration, SpecFlow hyperparameters, numerical precision, and hardware used for each task. N denotes the target sampling budget, $W_{\max}$ the maximum speculative horizon, and $r_{\max}$ the maximum drafting order. Prompt batch denotes the number of independent generation requests; speculative states within a SpecFlow block are internally batched for target verification. All methods compared on a given backbone use the same checkpoint, timestep schedule, conditioning, and evaluation configuration.

B.7 Implementation Details and Hyperparameters

We evaluate SpecFlow using publicly released pretrained flow-based models without retraining or modifying their parameters. Table 11 summarizes the model, sampling, evaluation, and hardware configurations used throughout our experiments. All acceleration methods compared on a given backbone use the same checkpoint, timestep schedule, numerical precision, resolution, conditioning, and evaluation protocol.

Sampling and evaluation. All main experiments use a target sampling budget of $N = 50$. BAGEL uses Euler integration with a shifted schedule (shift $= 3.0$) and is evaluated at 1024×1024 resolution. FLUX.1-dev and FLUX.1-Kontext-dev use their flow-matching Euler samplers with shifted sigma schedules at 512×512 and 1024×1024 resolution, respectively. HunyuanVideo uses rectified-flow Euler sampling with flow shift 7.0 and generates 1280×720 videos of 129 frames. All models are evaluated in BF16 with generation batch size one. We retain identical sampling schedules and conditioning settings across methods when comparing acceleration on the same backbone.

Variant	r	W	Accept. ↑	Acc./Pass ↑	# Passes ↓	Speedup ↑
Constant	0	5	0.319	1.79	10.1	2.21×
First-order	1	5	0.375	1.98	9.5	2.39×
Second-order	2	5	0.441	2.04	8.8	2.60×
Short fixed	2	2	0.717	1.40	13.2	1.76×
Medium fixed	2	3	0.578	1.68	10.5	2.18×
Long fixed	2	5	0.441	2.04	8.8	2.60×
Adaptive	adaptive	adaptive	0.498	1.82	**8.1**	2.79×

Table 12: **Drafting-order and horizon ablation.** Results on the analytic Gaussian-mixture model with $N = 25$, $g = 1.0$, and 20 independent seeds. All variants use identical exact verification and correction and differ only in their drafting strategy. At fixed $W = 5$, increasing the drafting order improves acceptance and reduces the number of sequential verification passes. Shorter horizons increase per-proposal acceptance but require more sequential passes. Adaptive selection achieves the highest measured speedup by balancing drafting order and speculative horizon along the trajectory.

For text-to-image generation, BAGEL and FLUX.1-dev are evaluated on the 553 GenEval prompts using four seeds per prompt. For image editing, BAGEL and FLUX.1-Kontext-dev are evaluated on the 606 GEdit instructions; the corresponding benchmark protocol is used for semantic consistency, perceptual quality, and overall editing quality. For video generation, HunyuanVideo is evaluated on the balanced 80-prompt VBench subset described in Section 4.

SpecFlow configuration. The stochastic target uses the same timestep discretization as the corresponding base sampler, with transition covariance $\Sigma_k = h_k g(t_k)^2 I$. The same covariance is used for the draft transition, enabling the simplified Gaussian verification and exact residual correction described in Section 3.3.1. The score estimate required by the stochastic target is recovered algebraically from the FM velocity using equation 9; consequently, SpecFlow requires no auxiliary score network.

Drafting uses only velocity evaluations associated with previously committed states and therefore requires no additional FM network evaluations. At each committed state, the drafting order r_m and speculative horizon W_m are selected from the committed velocity history using the adaptive rule described in Section 3.3.2 and Appendix A.1.7. The resulting speculative states are evaluated together in a batched target forward pass. Verification proceeds in temporal order, and after the first rejection all subsequent speculative states are discarded. The rejected transition is corrected using the exact residual distribution in equation 14.

Network parameterization. For all evaluated backbones, the network prediction used by the sampler is interpreted directly as the FM velocity v_θ required by our formulation. Accordingly, the score estimate in equation 9 is computed from the same target-model output used by the underlying flow sampler, without an additional network evaluation or parameterization-specific auxiliary model.

Timing protocol. Wall-clock latency is measured end-to-end on a single NVIDIA H100 80GB GPU. Timing includes all method-specific computation; for SpecFlow, this includes draft construction, batched target verification, likelihood-ratio evaluation, acceptance testing, rejection handling, and residual correction. Device execution is synchronized at timing boundaries, and warm-up iterations are excluded from reported measurements. All methods within a timing comparison use identical checkpoint, hardware, precision, batch size, resolution, timestep schedule, and conditioning. Reported speedups are computed directly from measured latency relative to the corresponding full sampler and are not inferred from NFE reduction.

B.8 Drafting Order and Adaptive Horizon

We ablate the drafting order and speculative horizon on the analytic Gaussian-mixture model with $N = 25$ and $g = 1.0$. All variants use the same exact verification and correction procedure and therefore preserve the same stochastic target distribution; only the drafting strategy is changed. Results are averaged over 20 independent seeds.

At fixed $W = 5$, increasing the drafting order from $r = 0$ to $r = 2$ raises acceptance from 0.319 to 0.441, reduces the average number of sequential verification passes from 10.1 to 8.8, and improves speedup from $2.21\times$ to $2.60\times$. This indicates that higher-order velocity extrapolation improves long-horizon drafting in this setting. Varying the horizon reveals a complementary trade-off: shorter blocks achieve higher per-proposal acceptance but require more sequential verification passes, whereas longer blocks reduce sequential depth despite lower acceptance.

The adaptive strategy combines these effects by varying the drafting order and horizon along the trajectory. It reduces the average number of sequential verification passes to 8.1 and achieves highest measured speedup of $2.79\times$. Importantly, these choices affect only efficiency: all variants use the same exact verification and correction mechanism and retain the same target-distribution guarantee.

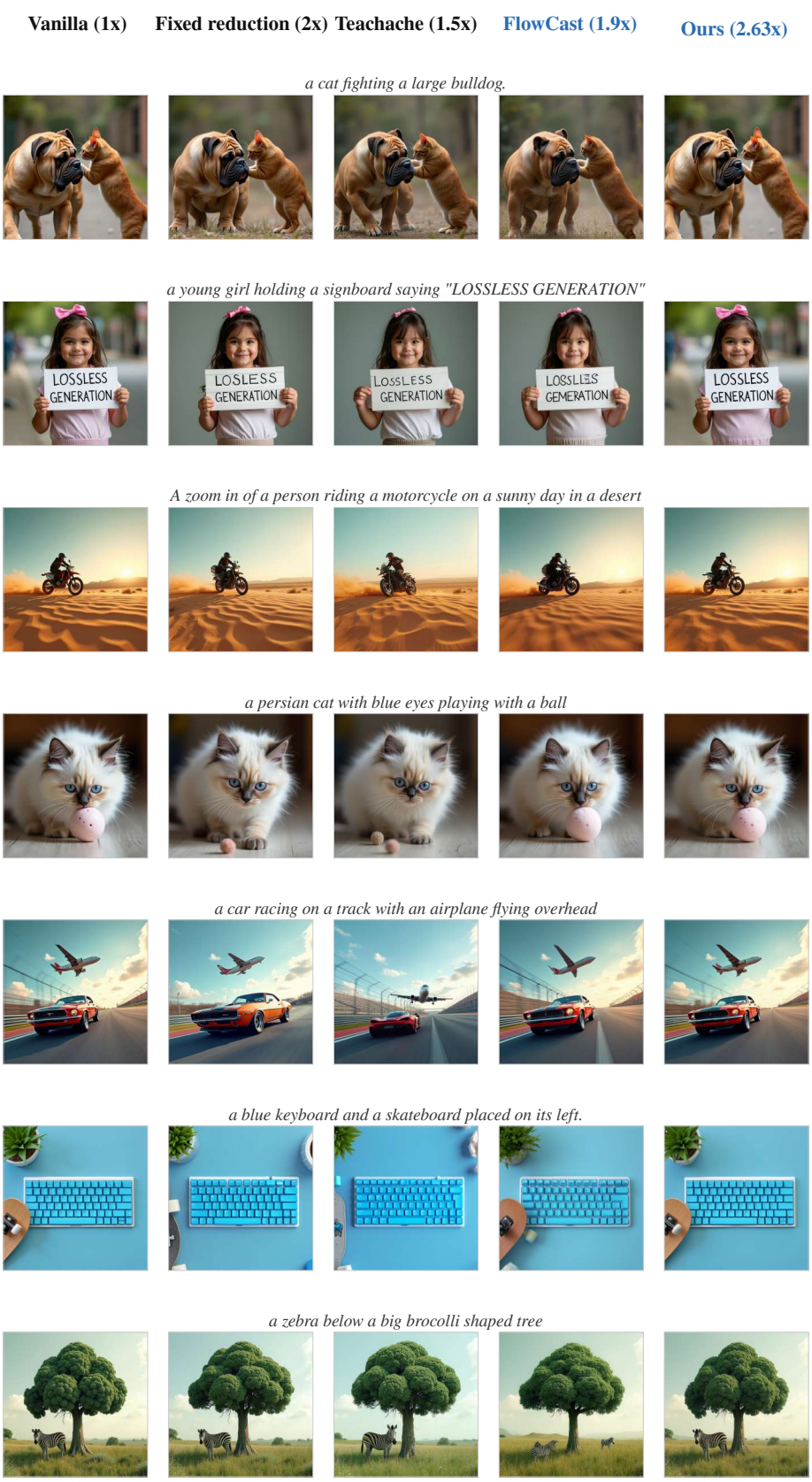

Figure 5: Generation examples for BAGEL model

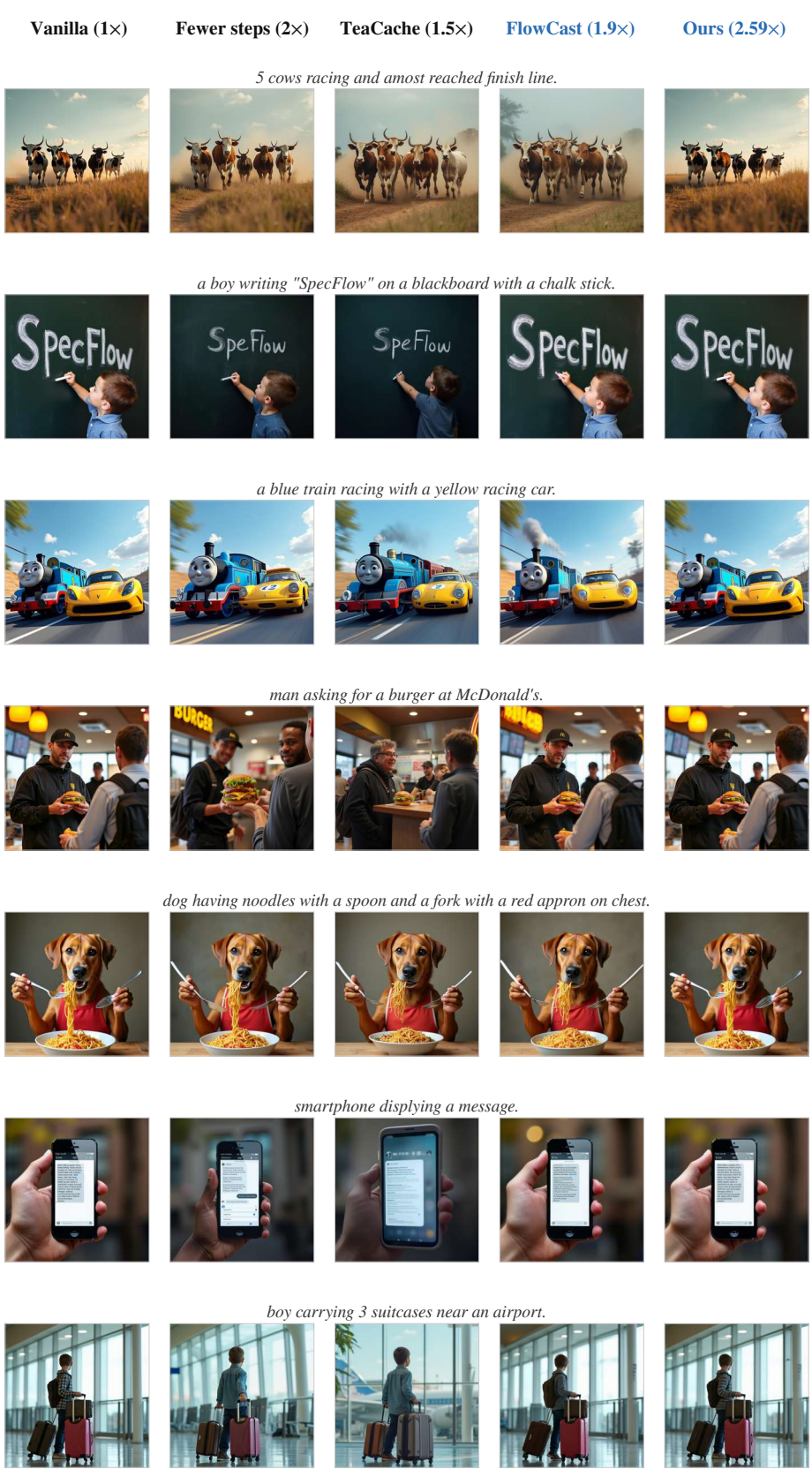

Figure 6: Generation examples for Flux.