FEW STEP VARIABLE LENGTH SAMPLING WITH BRANCHING FLOW MAPS

Anonymous authors
Paper under double-blind review

ABSTRACT

Few-step "flow maps" target the same problem space as diffusion and flow matching, but at vastly lower inference cost, which is becoming increasingly important with the large-scale use of inference-time reward alignment strategies. However, most standard diffusion, flow matching, and flow map models assume the length of the output is known a priori, which limits their usefulness, especially in scientific domains such as conditional small molecule or protein design. While variable-length flow matching approaches exist, they are inefficient at inference time, requiring many neural function evaluations. Few-step flow maps operate on continuous spaces, and it is unclear how to readily extend these to length variation, which is structurally a discrete property. Here we describe Branching Flow Maps, which enable few-step variable length generation on both discrete and continuous state spaces via element splitting and deletion. We demonstrate that our construction can compose with various established flow map approaches, affording efficient variable length generation with dramatically fewer neural function evaluations. We characterize the behavior of Branching Flow Maps on variable-length unconditional and conditional sampling tasks in synthetic and scientific domains, including small molecules and proteins, and we demonstrate accelerated reward-tilted sampling, with a sampling process that is now able to explore length as an additional dimension.

1 INTRODUCTION

Diffusion and flow matching models generate data by simulating from a learned process that transports a simple reference distribution to a complicated target(Ho et al., 2020; Lipman et al., 2023) (Holderrieth et al., 2025). Standardly, the number of elements or tokens in the generated object is fixed before sampling begins. For many scientific objects this is a problematic restriction. The number of atoms in a molecule, or the length of a loop in a protein, is usually not known *a priori* at inference time Billera et al. (2026b). Variable-length models lift this restriction by letting the dimension change during generation, whether through jumps between dimensions (Campbell et al., 2023), edit operations (Havasi et al., 2025), insertions (Ifriqi et al., 2026), or the splitting and deletion of elements (Billera et al., 2026b).

Since sampling requires many network evaluations, these models inherit the primary cost of their fixed-length counterparts. Fixed-length flow maps remove this cost by learning to take larger steps, mapping the state at one time to the state at any later time, so that a sample is produced in a handful of large jumps (Song et al., 2023; Frans et al., 2025; Geng et al., 2025; Boffi et al., 2025). These savings are critical for inference-time search and reward alignment, where every retained sample may require evaluating many candidate continuations (Singhal et al., 2025; Didi et al., 2026).

It is highly non-trivial to combine variable length dynamics with few step models, since traditional ODE flow maps learn the solution operator of an ODE flow defined on fixed dimensional Euclidean space. This can accommodate discrete labels by relaxing them to the simplex or to Euclidean space (Dieleman et al., 2022; Roos et al., 2026; Potaptchik et al., 2026b; Lee et al., 2026; Tang & Chatterjee, 2026), but it is unclear how such a relaxation would work for length, which is an inherently discrete quantity, and it is not in general possible to independently sample expansions or deletions. Many-step models avoid this problem since the probability of more than one event occurring over a small step is of order $o(\Delta)$ where Δ denotes the step size, so each element can be given

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.

its own event rate which is then independently sampled. As an example, a target concentrated on a single length, fixes the total number of children, which independent per-element decisions cannot achieve unless every decision is deterministic (Refer to appendix A for an example of this type of issue). A few-step sampler must therefore produce a correlated collection of discrete events, and a finite-time map of a continuous flow does not directly provide one.

We resolve this by separating the continuous and discrete parts of generation in time rather than in state. Branching Flow Maps divides the unit time interval into a fixed number of stages. Within a stage the process is an ordinary fixed-dimensional flow in which every element carries, alongside its state, a continuous *action* coordinate that encodes how many children it will produce. At the stage boundary, a fixed kernel reads off the actions and keeps, deletes, or splits each element accordingly. All actions evolve jointly under the same flow, so the dependence between simultaneous branching decisions is carried by the continuous dynamics and is learned by the flow map itself, while the boundary kernel introduces no learned discrete randomness. The dimension changes only at finitely many prescribed times, allowing the use of standard flow map objectives between them.

The construction composes with both self-distillation (Boffi et al., 2025) and stochastic flow maps (Holderrieth et al., 2026; McCallum et al., 2026) with sampling cost equal to the number of stages times the few steps taken within each. We characterise the behaviour of these different choices on a synthetic variable length task (sinusoid) as well as on QM9 molecule generation. We also evaluate the deterministic within-stage construction on conditional antibody sequence generation, where region lengths can not be pre-sampled and must instead be learned jointly with the state.

2 Background and Related Work

Flow matching and diffusion models (mostly subsumed under the Generator Matching framework of Holderrieth et al. (2025) and further extended in Billera et al. (2026a)) learn to generate novel samples from complicated probability distributions by simulating the a learned stochastic process X_t which satisfies $X_0 \sim p_{\text{simple}}$ and $X_1 \sim p_{\text{complicated}}$ in many steps. This *generator* is learnt by training a neural network to match the *conditional* generator $\mathcal{L}_t^Z$ of a process X_t^Z satisfying $\int p_1(x|z)p(dz) = p_{\text{data}}(x)$, available in closed form. The training procedure marginalises over Z and gives rise to a *marginal* generator. As an example, in flow matching, Z is a tuple $(X_0, X_1) \sim \pi(dx_0, dx_1)$ where π is an arbitrary coupling of $p_{\text{simple}}, p_{\text{data}}$, the conditional generator is a drift that interpolates from x_0 to x_1 and the neural network learns a marginal vector field that transports p_{simple} to p_{data}. Flow matching and diffusion models have shown great strength in image generation as well as in scientific domains, and have recently been extended to allow variable-length generation, but suffer from high computational cost stemming from the need to numerically integrate the dynamics by taking many small steps, motivating the development of so-called few step models.

2.1 Variable length models

The flow matching and diffusion framework has been extended to allow for on-the-fly variable length generation, where length does not need to be specified a priori, opening up the use of these models for tasks where the length distribution of elements is unknown, for example in protein binder design or language modelling. Examples include Campbell et al. (2023), Havasi et al. (2025), Ifriqi et al. (2026), Franke et al. (2026), Cheng et al. (2026), Tang & Chatterjee (2026) and Billera et al. (2026b).

2.1.1 Branching Flows

Billera et al. (2026b) introduce *Branching Flows*, a generative modelling framework which extends the Generator Matching framework of Holderrieth et al. (2025) to variable-dimension state spaces. Variable length dynamics are achieved by constructing conditional processes which couple elements of X_0 with elements of data samples X_1 via a forest of binary trees, with elements splitting and dying at stochastic times during the conditional process. These tree-dynamics are marginalised out by the model, leaving a process which, at inference time, evolves samples from p_0 to novel data samples of potentially different length than the X_0 sample.

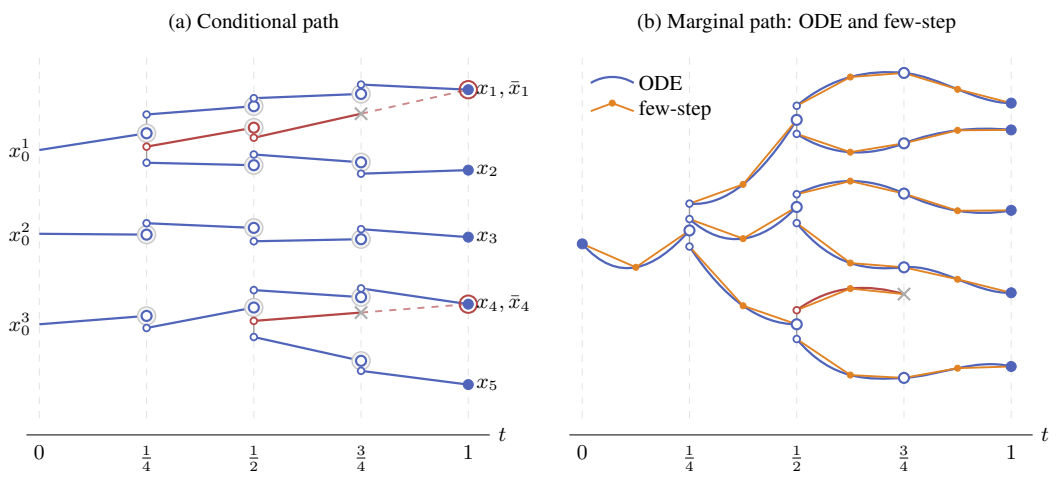


Figure 1: **Illustration of the conditional and marginal branching processes. (a)** A conditional training path. Each segment is a straight line, evolving the state x_t towards the lineage's next endpoint: either its terminal target or its next split point. While x_t evolves over each interval, an 'action' variable a_t controlling the number of descendants also evolves continuously. At every boundary, controlled by the value of a_t, each lineage either persists, is deleted, or is copied into two or more descendants. After the boundary, each surviving lineage again evolves toward its next target. Red lineages denote deletion-padded copies; the dashed red continuations indicate their corresponding latent augmented terminal targets. **(b)** A marginal path under the instantaneous ODE model (blue curves) follows the expectation of the generator of the conditional paths for the joint state $y_t = x_t, a_t$, and a few-step model (orange) learns to take finite steps (shown here: 2 per interval) along these paths. As with the conditional paths, during sampling, the value of a_t determines the number of descendants at each interval boundary, and surviving lineages are perturbed by noise (matching the conditional construction).

2.2 Few Step Models

Since the introduction of Consistency Models (Song et al., 2023; Song & Dhariwal, 2023) there has been significant work on sampling from generative models in few steps. Especially in the direction of flow maps (Frans et al., 2025; Geng et al., 2025; Boffi et al., 2025; Geng et al., 2026) which facilitate efficient few-step distillation. Later, Roos et al. (2026) and Potaptchik et al. (2026b) extend this framework to the categorical case by considering flow maps on the simplex. Lee et al. (2026) also introduces a discrete flow map via a simplex construction, but with a process that leaves the simplex at intermediate times, similar to Dieleman et al. (2022). We readily adopt this method to model discrete data. The aforementioned methods are deterministic, but stochastic few step models have been explored in for example McCallum et al. (2026), Pan et al. (2026) which both learn a path-wise consistent strong mapping as well as in Holderrieth et al. (2026) (refer to appendix B.3 for more details) and Potaptchik et al. (2026a) which learn via distillation of an ODE a function $F(x_t, \eta)$ satisfying $F(x_t, \cdot)_{\#}\mathcal{N}(0, 1) = p_{1|t}(\cdot|x_t)$, that is it learns to push forward Gaussian noise to the posterior distribution over future terminal samples given the current generative state x_t. Finally, Tang & Chatterjee (2026) introduce a few-step model over variable lengths in both the discrete and Euclidean settings, but not in the combined setting. For further discussion on this contribution, see subsection 2.2.1.

2.2.1 Expanding Flow Maps

In Tang & Chatterjee (2026) two variable length generative methods are introduced: i) EFlow, which, similar to Flowception (Ifriqi et al., 2026) achieves variable length dynamics by learning an insertion rate of new tokens. Inserted tokens are then de-noised on their own time schedules, to account for tokens inserted at later times having higher noise levels and ii) Expanding Flow Maps, which attempts to achieve variable length dynamics in few steps. This is done by utilising the deterministic few step framework of Boffi et al. (2025) to evolve already inserted coordinates in few steps, together

with an additional head which learns the expected remaining insertions in a gap over an interval $(s, t]$, which is then used to parametrise a binomial distribution which is used to sample the insertion count for that gap, followed by left-to-right truncation. We believe this sampling procedure prevents the method from sampling arbitrary length distributions. For a more expansive discussion, refer to appendix A.

3 METHOD

Many-step Branching Flows specifies instantaneous rates for branching events, in which an element is duplicated in place, and deletion events, in which an element is removed. Events can therefore happen at any time, but occur simultaneously with probability 0. As with discrete flow matching, sampling simulation uses small steps. Over larger time steps several events may occur, and naively sampling them independently from their individual rates ignores the dependence between them. The resulting sample can therefore exhibit a length that is inconsistent with the correct marginal path, see A for a concrete example.

The deterministic flow maps that exist in the literature learn the solution operator of an ODE in fixed dimensional Euclidean space, which is why flow maps over discrete data operate on continuous relaxations of the tokens (see for example Lee et al. (2026), Roos et al. (2026), Potaptchik et al. (2026b), Tang & Chatterjee (2026)). Length is inherently discrete, and it is unclear how to distil a length-varying process into a flow map. The only attempt so far Tang & Chatterjee (2026) appears to suffer from the theoretical issues mentioned earlier (we refer once again to appendix A).

In order to avoid these believed to be inherent theoretical difficulties with sampling lengths we define a generative process in which time is divided into fixed stages $0 = \tau_0 < \tau_1 < \cdots < \tau_J = 1$. Within a stage j the process evolves along a standard, fixed-dimensional velocity field, and we write Φ_j for the within-stage ODE transport map. At each stage boundary, a boundary kernel R_j defines a transition from action-states to the next stage, thereby possibly changing the dimensionality of the process. The terminal generated distribution is then the composition

$$p_1 = p_0 \, \Phi_0 R_0 \, \Phi_1 R_1 \cdots \Phi_{J-1} R_{J-1}, \tag{1}$$

which may then be decoded into p_{data} deterministically. See appendix C for additional information and proofs. The boundary event may depend on the current state, but not on the training target or other conditioning information used to construct training paths. Boundary events give rise to a lot of flexibility when constructing transport paths. Nothing here requires the boundary events to preserve the structure of the state space, so in particular they need not preserve length. In Branching Flow Maps, we keep the within-stage dynamics continuous and of fixed length, in order to allow existing few-step methods to learn Φ_j. In order to allow actions to depend on the state (as they must) we augment each element with a continuous *action* coordinate. The value of this coordinate at the end of a stage determines whether the element persists, is deleted, or splits into several descendants (Section 3.1). The actions of all elements evolve jointly within a stage, so the flow map learns the dependence between simultaneous events rather than ignoring it.

3.1 STATE, ACTIONS, AND BOUNDARY EVENTS

We consider a data target distribution supported on $\mathcal{M} = \bigsqcup_{k=1}^{L} S^k$, where each element takes values in a base space S that may be Euclidean, discrete, or a product of the two. The process must be continuous within a stage, so we represent discrete coordinates as one-hot vectors and, following Lee et al. (2026) and Dieleman et al. (2022), allow them to leave the simplex at intermediate times. For example, an atom in $\mathbb{R}^3 \times \{\mathrm{H}, \mathrm{C}, \mathrm{N}, \mathrm{O}, \mathrm{F}\}$ is represented during generation by a point in $\mathbb{R}^3 \times \mathbb{R}^5$. We write x for this continuous representation of an element.

To make branching a function of the state, each element also carries an *action* a. The action is a continuous encoding of the element's child count c at the next boundary, through a fixed embedding $c \mapsto e(c)$. A count of $c = 0$ deletes the element, $c = 1$ keeps it, and $c \geq 2$ splits it. The boundary event R_j at τ_{j+1} acts after stage j. In the conditional path it holds that $a = e(c)$ at the boundary, however For approximate outputs and inactive population constraint we decode each element's count as the nearest feasible encoding, $c = \arg\min_{c'} \|a - e(c')\|$. When the resulting total population violates the allowed bounds, the counts are decoded jointly subject to those bounds. It then replaces

the element in place by c copies of its state, perturbs the state of every surviving element with independent Gaussian noise, and gives each surviving element a fresh action $\eta \sim \mathcal{N}(0, I)$. R_j reads only the current state, as Section 3 requires. The additional noise is necessary to ensure the generative model has sufficient degrees of freedom. At the final boundary only keep/delete actions are allowed, and no noise is added.

3.2 Conditional paths

Conditional paths are indexed by a latent z, which consists of a data sample x_1, a sample x_0 from the simple distribution and a forest containing tree information. The forest connects the initial elements to the elements of x_1 across the stage boundaries, and it may include additional lineages that are deleted before $t = 1$. The children of each element form a contiguous block, so that sequence order is preserved. Any distribution over such forests satisfying some admissibility conditions (Appendix B.4) yields a valid construction. Appendix B.4 describes the ones we use here.

For every element alive in stage j, the forest fixes two targets: a state target b, which is the mean over its assigned augmented leaves, including copies marked for later deletion, and the element's child count c at the next boundary. Within the stage, the element's state moves in a straight line toward b, at the speed that would reach it at $t = 1$. Meanwhile, its action moves from its fresh noise η to $e(c)$ by the end of the stage:

$$X_t = X_{\tau_j} + \frac{t - \tau_j}{1 - \tau_j}\big(b - X_{\tau_j}\big), \qquad A_t = \frac{(\tau_{j+1} - t)\,\eta + (t - \tau_j)\,e(c)}{\tau_{j+1} - \tau_j}. \tag{2}$$

At τ_{j+1} the action equals $e(c)$, so R_j produces exactly the event that the forest prescribes. The children then start the next stage from their noisy copies of the parent's state. Every conditional boundary is therefore a draw from the same state-dependent kernel R_j (Lemma C.1), which is the requirement of Section 3. Because b averages over everything an element will become, the model's endpoint prediction for an element before a split is an unordered 'bag' of its future descendants. This bag resolves into an ordered sequence once the split occurs (Figure 5B).

3.3 Learning the stages

Instantaneous model. For each channel $v \in \{x, a\}$ we write y_v for the current value and β_v for the target, with $\beta_x = b$ and $\beta_a = e(c)$. The corresponding deadlines are $d_x = 1$ and $d_a = \tau_{j+1}$. The marginal within-stage velocity is then

$$v_v(t, y) = \frac{\mathbb{E}[\beta_v \mid Y_t = y] - y_v}{d_v - t}.$$

We learn an endpoint model $m_\theta(t, y)$ by minimising

$$\mathcal{L}(\theta) = \mathbb{E}_{t,\, z,\, Y_t} \sum_v w_v(t)\, D_v\big(\beta_v,\ m_\theta^v(t, Y_t)\big). \tag{3}$$

Each D_v is a Bregman divergence, up to a target-only term, so its minimiser is the conditional mean. We use squared error for continuous coordinates, cross-entropy for relaxed labels, and a divergence matched to the action encoding (with Rao–Blackwellized action targets for the local-tree construction; Appendix B.4). The weights $w_v(t)$ are time-dependent loss weights Billera et al. (2025).

Flow map. Within a stage, the learned process is an ODE of fixed length, so we learn its flow map in the same endpoint parametrisation. For $\tau_j \le s \le t \le \tau_{j+1}$with $s < \tau_{j+1}$,

$$\Phi_\theta^v(s, t, y) = y_v + \frac{t - s}{d_v - s}\big(P_\theta^v(s, t, y) - y_v\big), \qquad P_\theta(s, s, y) = m_\theta(s, y). \tag{4}$$

Here $P_\theta(s, t, y)$ is an effective endpoint for the jump from s to t. Composing two such maps through an intermediate time $u \in (s, t)$ gives a map of the same form, with effective endpoint

$$Q^v = \gamma_v\, P_\theta^v(s, u, y) + (1 - \gamma_v)\, P_\theta^v\big(u, t, \Phi_\theta(s, u, y)\big), \qquad \gamma_v = \frac{(d_v - t)(u - s)}{(d_v - u)(t - s)}. \tag{5}$$

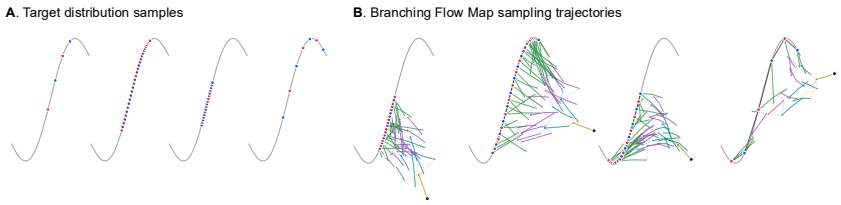


Figure 2: **A**. Training samples from the synthetic "alternating sinusoid" target distribution. **B**. Branching Flow Map (4 stages, 8 NFE total) sampling trajectories, starting from one point (black circle).

Following the self-distillation approach of Boffi et al. (2025), we train $P_\theta(s, t, y)$ toward $\mathrm{stopgrad}(Q)$ using the same divergences D_v. Since Q is a convex combination of endpoints, categorical targets remain in the simplex and cross-entropy remains valid. We freeze the trained instantaneous model and train an additional finite-map refinement branch. Appendix B describes this construction and the stochastic adaptations of Holderrieth et al. (2026) and McCallum et al. (2026).

Sampling. We start from N_0 elements with states $x_0 \sim p_0$ and actions $a_0 \sim \mathcal{N}(0, I)$. Within each stage we take n flow-map steps, and at the end of the stage we apply R_j. At $t = 1$, labels are decoded by argmax. With J stages, sampling costs Jn network evaluations.

4 Empirical Results

We evaluate joint state and length generation on the alternating-sinusoid task, QM9 molecules, and conditional antibody sequences. Unless otherwise specified, BFM denotes the four-stage model with deterministic within-stage dynamics. Extended results and experimental details are given in Appendices D–G.

4.1 A synthetic multimodal variable length task

To play the role of the now-classic 'spiral' and 'checkerboard' 2D synthetic datasets used for fixed-length flow matching and diffusion models, we introduce a synthetic variable length multimodal dataset: the 'alternating sinusoid'. A sample is a sequence of values that all lie along a sinusoid curve, evenly spaced in their x coordinate, with binary states alternating one after the other, and an a priori specified length distribution with two peaks. This requires a generator to i) handle strongly-coupled discrete and continuous states and ii) learn to match sequence length and state distribution jointly.

Few-step maps recover coupled length and state structure. BFM learns to generate sequences that jointly follow the curve, alternate their labels, and reproduce the bimodal length distribution. At eight evaluations, almost all samples have the correct alternating pattern, whereas coarse integration of the instantaneous model frequently loses this structure. Further evaluations mainly refine the geometry. The trajectories in Figure 2 show how shared predictions separate into ordered points as the population grows. Extended comparisons confirm that the improvement over coarse instantaneous sampling is shared by several BFM variants (Figure 6; Table 2).

4.2 QM9 comparisons

Distillation preserves molecular quality with fewer evaluations. Learning finite maps substantially improves QM9 generation at small sampling budgets (Figure 4). With eight evaluations, the standard BFM produces predominantly connected-valid molecules and closely approximates the reference length and chemical-property distributions. Direct integration of its instantaneous model at the same budget produces smaller molecules and poorer distributional agreement. At 16–32 evaluations, the finite model approaches its many-step instantaneous reference, while the many-step

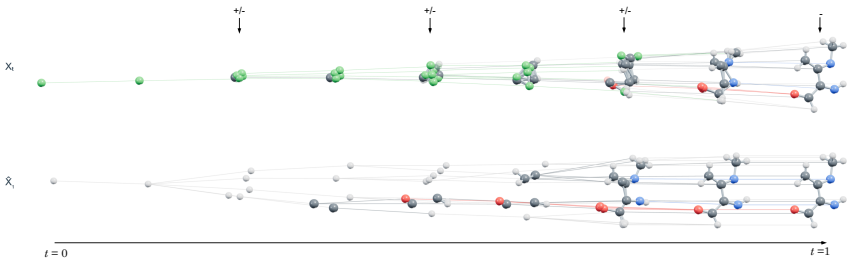


Figure 3: **Molecule (QM9) few-step sampling trajectory**. The sampling path for a Branching Flow Map QM9 sampler is shown, using 4 stages, with 2 NFEs per stage. Starting from a single atom at $t = 0$, the splits and deletions occur at stage boundaries (black down-arrows) except the final one, where only keep/delete actions occur. Top shows the X_t current state, and bottom the predicted endpoint $\hat{X}_1$.

Branching Flows SDE retains closer joint descriptor agreement. Figure 3 illustrates how a single initial element develops into a molecule through successive state refinement, splitting, and deletion.

Multiple stages constrain few-step performance. The single-stage controls perform substantially better at the smallest budgets (Figure 8; Table 5). Both the fixed-length oracle models, which receive a sampled length before generation, and the deletion-only BFM, which learns which initial atoms to retain, begin with all candidate atoms present. At four evaluations they produce substantially more connected-valid molecules than the four-stage BFM, with closer chemical-property agreement. Reducing BFM from four stages to two also improves low-budget validity, and gives a clear improvement on the sinusoid task. These results suggest that repeatedly resolving new populations consumes part of a limited evaluation budget, leaving fewer steps to refine their states. The difference between deletion-only and four-stage BFM narrows considerably at 16–32 evaluations.

Few-step performance is robust to alternative configurations Diamond Maps and strong stochastic flow-map adaptations also improve with finite-map training and additional evaluations (Figure 7). The deterministic BFM gives a favorable overall balance of validity and distributional agreement in these comparisons, while the Diamond Maps variant attains similar joint descriptor agreement at higher budgets. Lagrangian training objective Boffi et al. (2025) produces results close to the semigroup-trained model. The staged construction therefore accommodates several ways of learning the continuous dynamics between branching events. For supervising actions that control splits and deletions, replacing one-hot actions by a relaxed scalar count gives similar performance on both tasks, as does the non-Rao-Blackwellized model trained on sampled offspring counts (Tables 2–4). In the latter comparison, the tree construction also changes from the forward-local law to a backward construction. Comparable results across these settings suggest that analytic action averaging and the choice of action representation offer implementation flexibility while preserving the main few-step benefit.

4.3 Reward-guided molecule generation

Inference-time search explores length as well as molecular state. Flow Maps offer dramatic benefits for reward-tilted sampling, since rapid few-step rollouts to terminal samples allow rewards to be evaluated much more rapidly than diffusion or flow matching methods. To investigate whether reward tilted sampling would also work in the variable length context with a property that correlates with length, we use the same QM9-trained BFM, with Feynman–Kac steering and beam search Singhal et al. (2025); Oshima et al. (2025) shifting the generated atom-span (the largest distance between any two atoms) distribution toward either smaller or larger values (Figure 9). By perturbing the state and action noise at stage boundaries, their proposals can change atom count as well as geometry and atom identity, giving the search several ways to realize the requested molecular extent. The examples illustrate different structures reached in each direction. Few-step generation supplies the candidate continuations used to select intermediate states, connecting the variable-length sampler to inference-time search (Appendix G).

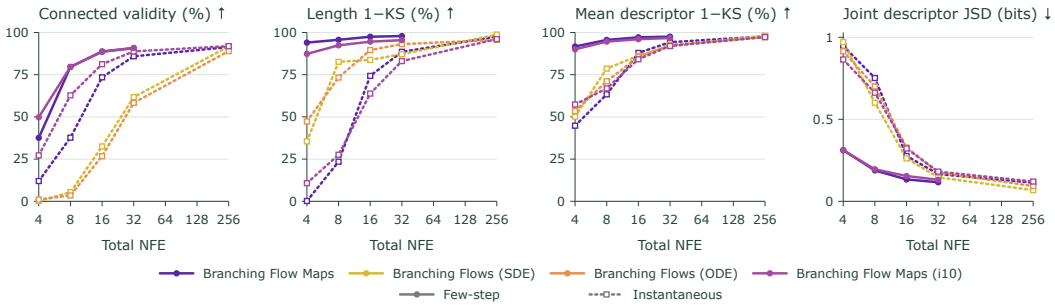


Figure 4: **Few step variable length molecule sampling.** Metrics for small molecule (QM9) sampling. At 4–32 NFE, the plotted finite maps give closer distributional agreement than their instantaneous counterparts evaluated at much larger computational budgets. Extended comparisons are in Appendix D and Tables 3–4.



Figure 5: **Few-step sampling paths from an antibody model**. Complementarity determining regions (yellow boxes) are fixed, and the rest of the sequence is variable. **A.** shows the model's endpoint ($t = 1$) prediction of the amino acid state at each of the 4 intervals, and then the final sample (5th row). **B.** shows an example where, prior to a split, the model is representing an unordered 'bag-of-tokens', which then resolves into an ordered sequence at the next interval, after splitting. Branching Flow Maps thus naturally produces a coarse-to-fine token representation, similar to that which is enforced by the process in e.g. Zhou et al. (2025).

4.4 Antibody sequence model

Antibodies are proteins, formed by a sequence of amino acids that arises from a complex biological process of recombination and mutation, providing an idealized language modeling substrate with biological applications.

Few-step sampling retains arbitrary-region conditioning. The antibody BFM can jointly generate the identities and unknown lengths of multiple editable regions while preserving any supplied sequence context. Alongside conventional CDR completion, this supports framework generation around fixed CDRs and completion of disjoint spans that cross predefined region boundaries (Figure 5). This carries the arbitrary-span conditioning capability of the many-step Branching Flows antibody model into the few-step setting (Billera et al., 2026b). We compare against the conditional Branching Flows and Edit Flows models on the same completion tasks.

Fewer evaluations translate into faster conditional generation. On arbitrary-span completion, eight to sixteen evaluations recover much of the sequence-distribution agreement of the many-step models (Table 7). Lengths are already well matched at four evaluations, while additional steps improve the diversity, novelty, and nativeness distributions. At sixteen evaluations, BFM has broadly similar distributional scores to the 128-evaluation Branching Flows model. This makes repeated conditional sampling practical while retaining the flexibility to infer each missing span's length. IgCraft, which uses a padding-based approach to length variation and can thus only sample specific predetermined antibody regions, provides a further variable-length region-inpainting comparison (Greenig et al., 2025) (Table 1). Restricting conditional sampling to these regions, on most metrics Branching Flow Maps matches or exceeds IgCraft's far more expensive sampler, with longer generation times under its 32-particle sampling procedure.

Table 1: IMGT-region infilling across inference budgets. Darker shading indicates the best result within each NFE budget and lighter shading the second best. IgCraft baselines are reported separately. The five quality columns report $1 - \mathrm{KS}$ agreement with the natural reference distributions; s/seq denotes seconds per sequence.

NFE	Model	Length ↑	Diversity ↑	Novelty ↑	CDR3 length ↑	AbNatiV ↑	s/seq ↓
4	Branching Flow Maps	**.932**	.652	**.663**	**.960**	**.560**	.0132
	BFM Instant	.235	**.672**	.233	.557	.218	.0091
	Branching Flows	.728	.338	.311	.944	.221	.0027
	Edit Flows	.094	.342	.250	.511	.191	**.0016**
8	Branching Flow Maps	**.970**	**.810**	**.848**	.982	**.820**	.0235
	BFM Instant	.325	.606	.552	.608	.412	.0164
	Branching Flows	.846	.511	.477	**.988**	.353	.0055
	Edit Flows	.350	.448	.454	.765	.283	**.0036**
16	Branching Flow Maps	**.971**	**.843**	**.887**	.980	**.868**	.0439
	BFM Instant	.705	.795	.833	.790	.742	.0319
	Branching Flows	.950	.664	.662	**.982**	.522	.0112
	Edit Flows	.654	.629	.665	.876	.441	**.0076**
128	BFM Instant	.966	.865	.899	.970	**.895**	.2370
	Branching Flows	**.982**	**.888**	**.914**	**.973**	.886	.0897
	Edit Flows	.869	.820	.860	.943	.757	**.0637**
4x32	IgCraft ($T = 1$)	.516	.340	.305	.618	.290	.5443
8x32	IgCraft ($T = 1$)	.870	.905	.922	.868	.866	1.0838
16x32	IgCraft ($T = 1$)	.742	.811	.846	.732	.925	2.1694
20x32	IgCraft ($T = .1$, default)	.631	.741	.783	.613	.853	2.7115

5 Discussion

Joint actions connect fixed-dimensional maps to variable-length generation. Branching Flow Maps learn coordinated structural changes by evolving continuous action coordinates jointly with the state and executing their decisions at stage boundaries. This separates the challenge of learning finite-time continuous dynamics from the bookkeeping of splitting and deletion. The experiments show that the construction can preserve coupled state and length structure across continuous, categorical, and mixed domains, including conditional generation of several antibody regions at once. The similar performance of several action and training variants suggests that this interface is useful across more than one particular flow-map implementation.

Deletion-only sampling trades progressive growth for initial capacity. The 'stages' introduced come at a cost: the finite step flow maps cannot sample across a stage boundary. As a result, fixed-length models and deletion-only Branching Flow Map models (which have only a single stage) can achieve excellent performance at even lower inference budgets. However, QM9 has a small, bounded size range, so an overcomplete initial population can cover the possible outputs at modest cost. Conditional antibody generation poses a different allocation problem. With fixed residues separating several editable gaps, an order-preserving deletion-only process can only shorten each gap's initial population. Extra positions in one gap cannot supply a shortage in another without moving across the fixed context. Supporting arbitrary missing regions would therefore require generous padding in every gap. Branching instead lets each gap acquire new positions as its sequence context is resolved, with deletion available to remove excess positions. This flexibility is particularly useful when the missing regions vary substantially in length, but comes at a slightly higher computational cost.

Conditional sampling creates opportunities for design-time search. While a 'toy' demonstration, the atom-span experiment illustrates how a few-step variable-length model can supply proposals for reward-guided search. For antibodies, arbitrary-region conditioning similarly allows candidate sequences to be explored around fixed functional motifs while their intervening lengths remain variable. The current sequence evaluations characterize distribution recovery and sampling speed. Extending these models to application-specific rewards, followed by experimental assessment of folding, binding, and developability, is a natural direction for testing their value in molecular design.

AI USE STATEMENT

Generative AI tools were used, via coding harnesses, for implementation and experimentation, as well as mathematical investigations. Generative AI tools also assisted with manuscript drafting, editing, and checking. The authors are responsible for the final text and scientific claims.

ETHICS STATEMENT

The reported experiments are computational studies of molecular generation and antibody sequence completion using existing datasets and introduce no ethical issues at the scale at which they have been tested.

REPRODUCIBILITY STATEMENT

Appendix B describes the conditional constructions and within-stage models, and Appendix C gives the stage-composition argument. Appendices E–G describe the datasets, architectures, training objectives, sampling settings, evaluation metrics, and reward-guided search procedures. Extended figures and numerical comparisons are provided in Appendix D.

REFERENCES

Lukas Billera, Hedwig Nora Nordlinder, and Ben Murrell. Time dependent loss reweighting for flow matching and diffusion models is theoretically justified, 2025. URL https://arxiv.org/abs/2511.16599.

Lukas Billera, Hedwig Nora Nordlinder, and Ben Murrell. Latent process generator matching, 2026a. URL https://arxiv.org/abs/2605.20547.

Lukas Billera, Hedwig Nora Nordlinder, Jack Collier Ryder, Anton Oresten, Aron Stålmarck, Theodor Mosetti Björk, and Ben Murrell. Branching flows: Discrete, continuous, and manifold flow matching with splits and deletions, 2026b. URL https://arxiv.org/abs/2511.09465.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation, 2025. URL https://arxiv.org/abs/2505.18825.

Andrew Campbell, William Harvey, Christian Weilbach, Valentin De Bortoli, Tom Rainforth, and Arnaud Doucet. Trans-dimensional generative modeling via jump diffusion models, 2023. URL https://arxiv.org/abs/2305.16261.

Chaoran Cheng, Zhanghan Ni, Yanru Qu, Yuxin Chen, Ruihan Guo, Jiajun Fan, and Ge Liu. Variable-length generative protein design via generalized poisson flow, 2026. URL https://arxiv.org/abs/2607.09039.

Kieran Didi, Zuobai Zhang, Guoqing Zhou, Danny Reidenbach, Zhonglin Cao, Sooyoung Cha, Tomas Geffner, Christian Dallago, Jian Tang, Michael M. Bronstein, Martin Steinegger, Emine Kucukbenli, Arash Vahdat, and Karsten Kreis. Scaling atomistic protein binder design with generative pretraining and test-time compute, 2026. URL https://arxiv.org/abs/2603.27950.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data, 2022. URL https://arxiv.org/abs/2211.15089.

Malte Franke, Stefan P. Schmid, Zarko Ivkovic, Kjell Jorner, and Andreas Krause. Generative molecular morphing for flexible-size design via unbalanced optimal transport, 2026. URL https://arxiv.org/abs/2606.07239.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=OlzB6LnXcS.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=uWj4s7rMnR.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J. Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models, 2026. URL https://arxiv.org/abs/2512.02012.

Matthew Greenig, Haowen Zhao, Vladimir Radenkovic, Aubin Ramon, and Pietro Sormanni. Igcraft: A versatile sequence generation framework for antibody discovery and engineering, 2025. URL https://arxiv.org/abs/2503.19821.

Marton Havasi, Brian Karrer, Itai Gat, and Ricky T. Q. Chen. Edit flows: Flow matching with edit operations, 2025. URL https://arxiv.org/abs/2506.09018.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020. URL https://arxiv.org/abs/2006.11239.

Peter Holderrieth, Marton Havasi, Jason Yim, Neta Shaul, Itai Gat, Tommi Jaakkola, Brian Karrer, Ricky T. Q. Chen, and Yaron Lipman. Generator matching: Generative modeling with arbitrary markov processes, 2025. URL https://arxiv.org/abs/2410.20587.

Peter Holderrieth, Douglas Chen, Luca Eyring, Ishin Shah, Giri Anantharaman, Yutong He, Zeynep Akata, Tommi Jaakkola, Nicholas Matthew Boffi, and Max Simchowitz. Diamond maps: Efficient reward alignment via stochastic flow maps, 2026. URL https://arxiv.org/abs/2602.05993.

Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3D. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8867–8887. PMLR, 2022. URL https://proceedings.mlr.press/v162/hoogeboom22a.html.

Tariq Berrada Ifriqi, John Nguyen, Karteek Alahari, Jakob Verbeek, and Ricky T. Q. Chen. Flowception: Temporally expansive flow matching for video generation, 2026. URL https://arxiv.org/abs/2512.11438.

Chaitanya K. Joshi, Xiang Fu, Yi-Lun Liao, Vahe Gharakhanyan, Benjamin Kurt Miller, Anuroop Sriram, and Zachary W. Ulissi. All-atom diffusion transformers: Unified generative modelling of molecules and materials, 2025. URL https://arxiv.org/abs/2503.03965.

Aleksandr Kovaltsuk, Jinwoo Leem, Sebastian Kelm, James Snowden, Charlotte M. Deane, and Konrad Krawczyk. Observed antibody space: A resource for data mining next-generation sequencing of antibody repertoires, 2018. URL https://doi.org/10.4049/jimmunol.1800708.

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M. Boffi, and Jinwoo Kim. Flow map language models: One-step language modeling via continuous denoising, 2026. URL https://arxiv.org/abs/2602.16813.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

Sam McCallum, Zander W. Blasingame, Timothy Herschell, Niklas Rindtorff, Alexander Tong, and James Foster. Strong stochastic flow maps, 2026. URL https://arxiv.org/abs/2606.01086.

Yuta Oshima, Masahiro Suzuki, Yutaka Matsuo, and Hiroki Furuta. Inference-time text-to-video alignment with diffusion latent beam search, 2025. URL https://arxiv.org/abs/2501.19252.

Zhengkai Pan, Peter Potaptchik, Wenxi Yao, Michael S. Albergo, and Jakiw Pidstrigach. Itô maps for any-step sdes, 2026. URL https://arxiv.org/abs/2606.11156.

Peter Potaptchik, Adhi Saravanan, Abbas Mammadov, Alvaro Prat, Michael S. Albergo, and Yee Whye Teh. Meta flow maps enable scalable reward alignment, 2026a. URL https://arxiv.org/abs/2601.14430.

Peter Potaptchik, Jason Yim, Adhi Saravanan, Peter Holderrieth, Eric Vanden-Eijnden, and Michael S. Albergo. Discrete flow maps, 2026b. URL https://arxiv.org/abs/2604.09784.

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1:140022, 2014. doi: 10.1038/sdata.2014.22. URL https://doi.org/10.1038/sdata.2014.22.

Daan Roos, Oscar Davis, Floor Eijkelboom, Michael Bronstein, Max Welling, İsmail İlkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. Categorical flow maps, 2026. URL https://arxiv.org/abs/2602.12233.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models, 2025. URL https://arxiv.org/abs/2501.06848.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models, 2023. URL https://arxiv.org/abs/2310.14189.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models, 2023. URL https://arxiv.org/abs/2303.01469.

Sophia Tang and Pranam Chatterjee. Expanding flow maps, 2026. URL https://arxiv.org/abs/2607.21585.

Oliver M. Turnbull, Dino Oglic, Rebecca Croasdale-Wood, and Charlotte M. Deane. p-iggen: A paired antibody generative language model, 2024. URL https://doi.org/10.1093/bioinformatics/btae659.

Cai Zhou, Chenyu Wang, Dinghuai Zhang, Shangyuan Tong, Yifei Wang, Stephen Bates, and Tommi Jaakkola. Next semantic scale prediction via hierarchical diffusion language models, 2025. URL https://arxiv.org/abs/2510.08632.

A Expanding Flow Maps

Expanding Flow Maps describes a method for few step generative modelling on variable length state spaces. To this date, code has not been released, and while we tried to the best of our ability to reconstruct the method we believe the current version does not contain sufficient information to do so. To the best of our understanding there also appears to be some issues relating to conflating conditional and marginal generative paths which we believe lead to the method, as described, not sampling from the data distribution at $t = 1$. We elaborate on these believed issues in this appendix.

A.1 Binomial insertions and truncation

The procedure by which Expanding Flow Maps samples insertions when mapping from s to t is to, for each gap i, predict the expected amount of insertions in that gap $I_{s,t}(x_s)$ and then sample the number of insertions from $\ell_i \sim \text{Bin}\left(L - n_s, \frac{I_{s,t}(x_s)[i]}{L-n_s}\right)$ (see their equation (28)), followed by left to right truncation to prevent the total sampled length from exceeding the maximum. This insertion procedure can not represent arbitrary terminal length distributions. To realise this, consider

as an example the task of learning to sample from a data distribution over sequences which are AB with probability $\frac{1}{2}$ and A with probability $\frac{1}{2}$. Suppose we wish to sample in two steps from this distribution with a perfectly trained neural network, using the insertion schedule $\alpha_t = \min(2t, 1)$ that resolves all insertions before time $t = \frac{1}{2}$. Consider now the first step, from 0 to $\frac{1}{2}$. An optimally trained network will predict $I_{0,1/2}(\emptyset) = \frac{3}{2}$. Following the sampling procedure then yields $\ell_0 \sim \mathrm{Bin}(L - n_s, \frac{\frac{3}{2}}{L-n_s}) = \mathrm{Bin}(2, \frac{3}{4})$ which yields $\mathbb{P}[\ell = 1] = 0.375, \mathbb{P}[\ell = 0] = 0.0625, \mathbb{P}[\ell = 2] = 0.5625$ while the behaviour required for distribution matching is $\mathbb{P}[\ell = 1] = \mathbb{P}[\ell = 2] = \frac{1}{2}$ so the sampling procedure of Expanding Flow Maps can not recover the correct terminal distribution even with a perfectly trained neural network. It is unclear to us if the maximum length is supposed to be a maximum tensor dimension that is never hit in practice, or the actual maximum length of the data distribution. However, truncation at this maximum length is not the sole issue, as is shown.

These inconsistencies, as well as other issues, made it difficult to faithfully reproduce the likely intended method, and we therefore caution about our empirical comparisons. However, we believe the binomial insertion and truncation issue to be a fundamental limitation of Expanding Flow Maps.

A.2 Implementation deviations

We note that our implementation deviates in substantive ways from the paper description. These deviations cover both necessary domain-specific adjustments as well as choices made when there was ambiguity in the paper. For example, to the best of our understanding, from the preprint we currently do not think that EFM allows variable length generation over continuous space, since equation 9 prescribes $p_1 \in \mathcal{P}(\mathbb{R}^{d(1)})$, however the abstract claims "Across both continuous and discrete modalities, we establish EFlows and EFMs as a principled framework for settings in which output size is itself a learned, controllable degree of freedom." (Tang & Chatterjee, 2026). We chose to follow the abstract, and note that we made other such specific choices when the paper was ambiguous or when there were apperent typos.

B Implementation Details

Here we recall the work of Dieleman et al. (2022) and Lee et al. (2026) to motivate our simplex parametrisation of discrete categories.

B.1 Simplex parametrisation

In order to model discrete data via flow matching / diffusion we consider a conditional process which at $t = 1$ is a simplex vertex (that is, a one-hot vector) but at intermediate times $0 \leq t < 1$ may drift off the simplex into the ambient space $\mathbb{R}^d$. As noted in Dieleman et al. (2022) a property of the conditional score function for this process allows the cross-entropy loss to be used despite the off-simplex nature of the intermediate states. This is due to the fact that the expectation $\mathbb{E}[X_1|X_t = x_t]$ is a continuous convex combination of the vertices of the simplex, hence $\mathbb{E}[X_1|X_t = x_t] \in \Delta^{d-1}$ so the cross-entropy loss may be used. We may then freely adapt the few step modelling frameworks of Holderrieth et al. (2026) and McCallum et al. (2026), which work for ODEs and SDEs in the continuous state case. We also introduce a new method specific to Few Step Branching Flows.

B.2 Deterministic within-stage maps

The standard model uses deterministic within-stage ODE transport and Gaussian randomization at internal boundaries. We first train an instantaneous endpoint predictor using standard flow matching, freeze it, and train an additional finite-map refinement branch using teacher rollouts and two-step composition targets. The sinusoid and QM9 specifications are given in Sections E.4–E.5; the antibody refinement branch, training objectives, and polishing procedure are described in Sections F.3–F.6.

B.3 Stochastic within-stage variants

The Diamond Maps adaptation retains the inherited state as conditioning information and draws a fresh Gaussian working state for each stage, then transports that noise toward a Gaussian conditional boundary target. The strong stochastic flow-map adaptation instead supplies path-coupled Brownian features to maps of the coordinate and relaxed-label SDEs; actions retain their ODE interpolation. These are adaptations of the respective frameworks (Holderrieth et al., 2026; McCallum et al., 2026). Their conditional processes, noise coupling, and training settings are given in Section E.7.

B.4 Conditional trees, anchors, and action targets

Sampled trees and action targets (non-Rao–Blackwellized). A conditional tree describes how an initial element becomes a group of target elements. Its root corresponds to an element of x_0, its leaves correspond to target elements, and each level records the population at a stage boundary. With several initial elements, we use a forest with one tree per initial element. To include deletions, we first augment the target by inserting marked copies of target elements. These copies have ordinary states and participate in the tree construction, but their lineages are removed before the output is returned. Each original target element appears exactly once among the unmarked leaves.

A simple construction is to work backwards from the augmented leaves: one can, for example, randomly coalesce adjacent groups at successive stage boundaries until the prescribed number of roots is reached. Each merge becomes a split when the tree is read forwards. The groups must respect the maximum offspring count and the available number of stages; retaining a group gives a unary continuation. We reserve the final boundary for keep/delete actions, so all descendant groups must have become singletons by the start of the final stage. Adjacency preserves sequence order. The tree and anchor strategies described in Branching Flows (Billera et al., 2026b) can also be adapted to this stage grid. The initial population law does not depend on the rest of the tree construction, instead augmentation and tree sampling are conditioned on constituting a valid coupling of the source and deletion-augmented target.

Once the forest and deletion events have been sampled, the action target is simply the child count c that occurred at the next boundary: $c = 0$ for deletion, $c = 1$ for continuation, and $c \geq 2$ for a split. For one-hot action encoding, the target is $e(c)$ and the action loss is the usual hard-label cross-entropy. For example, an element representing five terminal leaves that splits into groups of sizes two and three has action target $e(2)$ at that boundary. The remaining splits are supervised in subsequent stages. The same sampled-count supervision can also be used with the forward tree construction below.

Anchors: the state represented by a group. Let B_v be the set of augmented leaf indices assigned to a node v, and let $\widetilde{x}_1^{(\ell)}$ be the continuous representation of augmented leaf ℓ, with categorical components represented by one-hot vectors. We choose the node's state target, or *anchor*, to be

$$b_v = \frac{1}{|B_v|} \sum_{\ell \in B_v} \widetilde{x}_1^{(\ell)}.$$

Thus, a coordinate anchor is the mean position of the assigned leaves, while a categorical anchor gives their category proportions. For example, a group with labels (A, B, B) has categorical anchor $(e_A + 2e_B)/3$. Marked copies contribute to this average in the same way as unmarked leaves. The anchor summarizes everything currently assigned to the element; a split gives each child its own smaller group and corresponding anchor.

During stage j, the physical state moves towards b_v with deadline $t = 1$, while the action moves towards $e(c)$ with deadline τ_{j+1}, as in Section 3.2. Consequently, an internal anchor specifies the direction of travel during the stage. At an internal boundary, every child inherits the parent's attained state and receives independent boundary perturbations, including unary continuations. The children then move towards their respective anchors. By the final stage every remaining group is a singleton, and each unmarked singleton has its original target element as its anchor. The terminal boundary removes any remaining marked singletons and adds no noise.

Local trees: choosing each split from its current group. The forward-local construction used in our implementation starts by partitioning the ordered augmented leaves among the initial roots. It

then builds the tree forwards, choosing each node's children from its current descendant group. For a group containing $w > 1$ augmented leaves, the unconditioned child count at boundary τ_{j+1} is

$$c \sim 1 + \mathrm{Binomial}(w-1, p_j), \qquad p_j = \frac{G(\tau_{j+1}) - G(\tau_j)}{1 - G(\tau_j)}.$$

Here G is a growth clock that reaches one at the last splitting boundary. A group of w leaves needs $w-1$ additional elements to become fully resolved, and p_j controls how much of that growth occurs at the next boundary. The clock and numerical settings are given in Sections E.3 and F.2.

We condition this count distribution on the offspring bound and the ability of every child to finish splitting in the remaining stages. Specifically, if M is the maximum offspring count and C_{j+1} is the maximum number of leaves that one child can resolve from the next stage onwards, the feasible counts satisfy

$$\left\lceil \frac{w}{C_{j+1}} \right\rceil \le c \le \min(M, w).$$

After sampling c, we choose uniformly among the ordered partitions of the parent group into c nonempty contiguous child groups, each of size at most C_{j+1}. The root partition is sampled uniformly under the analogous constraints. Near the last splitting boundary these constraints force the groups to resolve into singletons. Each node's anchor is computed directly from its full assigned group, independently of how that group will subsequently be partitioned.

An unmarked singleton continues with $c = 1$. A marked leaf becomes eligible for deletion once it first appears as a singleton. We sample its death time uniformly over its remaining lifetime and round upwards to a stage boundary. Equivalently, a marked singleton still alive in stage j is deleted at the next boundary with probability $(\tau_{j+1} - \tau_j)/(1 - \tau_j)$, which is $1/(J-j)$ for equal stages. Until it becomes a singleton, a marked leaf participates in its group's growth and anchor just like any other leaf. These growth and deletion rules define a local count prior $\pi_j(c \mid w, \delta)$, where δ indicates that the current element is a marked singleton.

Rao–Blackwellized action targets. The local construction lets us average over the next action analytically. We still sample an integer c to construct the conditional path. At local stage time $u = (t - \tau_j)/(\tau_{j+1} - \tau_j)$, its observed action state is

$$A_u = (1-u)\eta + ue(c), \qquad \eta \sim \mathcal{N}(0, I).$$

Given the group's mass w, singleton mark δ, and observed action state, Bayes' rule gives, for $u < 1$,

$$q_j(c \mid A_u, w, \delta) \propto \pi_j(c \mid w, \delta) \exp\left[-\frac{\|A_u - ue(c)\|^2}{2(1-u)^2} \right].$$

We normalize over the feasible counts and replace the sampled one-hot target by $\sum_c q_j(c \mid A_u, w, \delta) e(c)$. For one-hot encodings, this is simply the probability vector q_j, giving the soft-target cross-entropy $-\sum_c q_j(c \mid A_u, w, \delta) \log m^a_{\theta,c}(t, Y_t)$. At $u = 0$ the target equals the local prior; at $u = 1$ the observed action is a feasible vertex and the target is that vertex. The masses and marks are used only to construct training targets.

The forward construction and descendant-average anchors are what make this local averaging exact. Given the current group's mass and singleton mark, the next count is drawn from the specified local law independently of the earlier history. The current physical state depends on that history and the group's anchor, while the next split affects the current observation through A_u. Thus, writing Y_t for the full observed state,

$$\mathbb{E}[e(c) \mid Y_t, w, \delta] = \mathbb{E}[e(c) \mid A_u, w, \delta] = \sum_c q_j(c \mid A_u, w, \delta) e(c).$$

The soft-target loss is therefore the conditional expectation of the sampled-action loss. It preserves the expected action gradient while averaging out uncertainty in the sampled count. Intuitively, early in a stage several counts may be consistent with the noisy action state. Averaging gives each plausible count its posterior weight, smoothing the training signal across random realizations of the local tree. As the action approaches a one-hot vertex, the posterior concentrates on the sampled count. This averaging is used for instantaneous action supervision; the finite-map phase uses the teacher and composition targets described in Section 3.3.

C Proofs and Derivations

C.1 Conditional paths and regression targets

Let $\delta_j := \tau_{j+1} - \tau_j$ denote the stage width, let $s = \frac{t-\tau_j}{\delta_j}$ denote the within-stage local time and let $\kappa_j = \frac{1-\tau_j}{\delta_j}$. We write $Y_{j,s} = (X_{j,s}, A_{j,s})$ for the state. For elements alive in stage j we define the anchors β_j^x for the state and the regression target β_j^a for the continuous relaxation of the boundary actions. Our conditional paths become then within j

$$X_{j,s} = X_{j,0} + \frac{s}{\kappa_j}(\beta_j^x - X_{j,0})$$
$$A_{j,s} = (1-s)A_{j,0} + s\beta_j^a.$$

That is, $A_{j,s}$ follows a linear path towards the actual target action at the end of the stage, while $X_{j,s}$ follows a path that would end at the weighted descendant mean of its nodes had it continued all the way to global time $t = 1$, following the same construction as Billera et al. (2026b). This yields the velocity fields

$$\partial_s X_{j,s} = \frac{\beta_j^x - X_{j,s}}{\kappa_j - s}$$
$$\partial_s A_{j,s} = \frac{\beta_j^a - A_{j,s}}{1-s}.$$

Define the expected endpoint conditioned on the current state to be $m_j^x(s,y) := \mathbb{E}[\beta_j^x | Y_{j,s} = y]$ and $m_j^a(s,y) = \mathbb{E}[\beta_j^a | Y_{j,s} = y]$, which are learned at optimum by standard flow matching techniques, which may then be distilled into few-step models.

Indeed, $(\beta_j^x - X_{j,s})/(\kappa_j - s) = (\beta_j^x - X_{j,0})/\kappa_j$, so this expression equals the derivative of the linear conditional path.

C.2 Kernel composition

As previously stated we consider a data distribution supported on the variable length space $\mathcal{M} := \bigsqcup_{k=0}^{L} S^k$ where L is some potentially infinite max length and S is a Euclidean space, a finite categorical space or a product of these. In order to sample from this distribution and enable on the fly variable length generation we learn a process on the augmented space $\mathcal{M}' := \bigsqcup_{k=0}^{L_{\text{aug}}} (S' \times \mathcal{A})^k$ which goes through stages $j = 0, \cdots, J-1$ between which it may change dimension. Here, S' represents the state space corresponding to a continuous relaxation of discrete states to a Euclidean representation, where discrete elements at terminal time are represented by simplex vertices (that is, one-hot vectors) but at intermediate times are allowed to drift into ambient Euclidean space (which is the setting considered by Dieleman et al. (2022) and Lee et al. (2026)) and $\mathcal{A}$ represents a continuous relaxation of the discrete *actions* that can be taken. L_{aug} is the maximum possible length when the space is augmented by to-be-deleted elements. We define a conditional stochastic process $Y_{j,s}$ on this state space in the following way: Let $Z = (X_1, X_0, \mathcal{T}, \mathcal{D}, \mathfrak{A}, \mathcal{C})$ be a random variable containing the terminal data sample X_1, the random source sample X_0, the sampled forest and descendant assignments $\mathcal{T}$, the deletion marks and augmented to-be-deleted states $\mathcal{D}$, the internal anchors $\mathfrak{A}$ and the boundary actions $\mathcal{C} = \{\mathcal{C}_{i,j}\}$. Let $\zeta_j = (\epsilon_j, \eta_j)$ contain the boundary perturbation noise and fresh action noise introduced at stage j, and write $\zeta_{0:j} = (\zeta_0, \ldots, \zeta_j)$. We denote their conditional versions given $Z = z$ by ζ_j^z. At the endpoint of stage j, the action coordinate of each element is the chosen encoding $e(c)$ of the action taken by that element. Thus the prescribed boundary actions, and hence the dimensions of the required noise, are determined by $Y_{j,1}$ independently of Z. Let $\rho_{j+1}(y, d\zeta)$ denote the prescribed fresh-noise law, with dimensions determined by the actions decoded from y. We sample this noise so that

$$\text{Law}(\zeta_{j+1} \mid Z, \zeta_{0:j}) = \rho_{j+1}(Y_{j,1}, \cdot).$$

Let $F_j(y, \zeta) \in \mathcal{M}'$ be the boundary map which executes the actions decoded from y and applies the prescribed perturbation and action initialisation using ζ. In particular, F_j depends only on y and ζ, not Z. At the final boundary this map only keeps or deletes elements, and does not re-noise. Define

$$Y_{j+1,0} = F_j(Y_{j,1}, \zeta_{j+1}), \qquad R_j(y, B) := \int \mathbf{1}_B(F_j(y,\zeta))\, \rho_{j+1}(y, d\zeta).$$

Since $Y_{j,1}$ is determined by $(Z, \zeta_{0:j})$, we obtain

$$\mathbb{P}[Y_{j+1,0} \in B \mid Z, \zeta_{0:j}] = R_j(Y_{j,1}, B).$$

Conditioning this identity on $Y_{j,1}$ establishes that $R_j(y, \cdot)$ is a version of $\mathrm{Law}(Y_{j+1,0} \mid Y_{j,1} = y)$. We let

$$p_{j,s}(B) := \int \int \mathbb{P}[Y_{j,s} \in B | \zeta^z_{0:j} = \bar{\zeta}_{0:j}, Z = z] p_{\zeta_{0:j}}(d\bar{\zeta}_{0:j} | Z = z) p_Z(dz)$$

denote the marginal paths of this process. Let $j = 0, \ldots, J-1$ index the flow stages, and let $Y_{J,0}$ denote the state after the final boundary operation. The complete reference construction satisfies $\pi(Y_{J,0}) = X_1$ almost surely, where $X_1 \sim p_{\text{data}}$ and $\pi : \mathcal{M}' \to \mathcal{M}$ is the deterministic decoding map. What we aim to show is therefore that one may learn a transport which for each stage j matches the marginal distribution $p_{j,s}$ of the process. Before proceeding, we introduce the notation $\mu_j := p_{j,0}$, $j = 0, \ldots, J$, where $p_{J,0} := \mathrm{Law}(Y_{J,0})$. For $j < J$, this is the post-action, pre-flow marginal law; μ_J is the terminal post-boundary law. Now, define for stage j the marginal vector field $v_j(s, y) := \mathbb{E}[\partial_s Y_{j,s} | Y_{j,s} = y]$ for $s \in [0, 1)$ and assume that for each fixed-dimensional component of the state space and for every $t < 1$ that this vector field generates a sufficiently regular ODE flow with solution map $\phi_j : [0, 1) \times [0, 1) \times \mathcal{M}' \to \mathcal{M}'$. We further assume that the limit $\lim_{t \nearrow 1} \phi_j(0, t, y)$ exists for μ_j-a.e y. From this we may define a trivial ODE kernel up to μ_j-a.e equality as $\Phi_j(y, B) := \mathbf{1}_B(\phi_j(0, 1, y))$. For discussions on regularity and ODE solution maps in the generative modelling setting, we refer to Holderrieth et al. (2025) and Billera et al. (2025). We are now almost ready to state and prove our main result, but the following lemmas will be useful:

Lemma C.1 (Boundary compatibility)**.** *Under the prescribed boundary update and fresh-noise law,*

$$p_{j,1} R_j = \mu_{j+1}, \qquad j = 0, \ldots, J-1.$$

Proof. Let $\mathcal{H}_j := \sigma(Z, \zeta_{0:j})$. Since $Y_{j,1}$ is $\mathcal{H}_j$-measurable, the prescribed conditional noise law gives, for every measurable $B \subseteq \mathcal{M}'$,

$$\begin{aligned} \mathbb{P}(Y_{j+1,0} \in B \mid \mathcal{H}_j) &= \int \mathbf{1}_B(F_j(Y_{j,1}, \zeta))\, \rho_{j+1}(Y_{j,1}, d\zeta) \\ &= R_j(Y_{j,1}, B). \end{aligned}$$

Taking expectations yields

$$\mu_{j+1}(B) = \mathbb{E}[R_j(Y_{j,1}, B)] = \int R_j(y, B)\, p_{j,1}(dy) = (p_{j,1} R_j)(B).$$

□

Lemma C.2. *(Single step kernel composition) It holds for all j that $\mu_j \Phi_j = p_{j,1}$*

Proof. This follows from standard results in the field under appropriate regularity conditions, as we at each stage behave like a standard flow matching / few step model, we refer in particular to assumption 5 in appendix A.2 of Holderrieth et al. (2025) as well as Billera et al. (2025). □

We are now ready to prove our main result

Theorem C.3. *(Kernel composition samples from the data distribution) Let notation be as established, and denote by p_{simple} the source distribution assumed to be equal to μ_0. It holds that*

$$p_{simple} \Phi_0 R_0 \Phi_1 R_1 \cdots \Phi_{J-1} R_{J-1} = \mu_J$$

and the decoded terminal law satisfies $\pi_{\#} \mu_J = p_{data}$.

Proof. We proceed by induction. Define $q_0 := p_{\text{simple}}$ and $q_{j+1} := q_j \Phi_j R_j$ for $j = 0, \ldots, J-1$. Suppose it holds for some j that $q_j = \mu_j$. Then by applying lemmas C.2 and C.1 it holds that

$$q_{j+1} = q_j \Phi_j R_j = \mu_j \Phi_j R_j = p_{j,1} R_j = \mu_{j+1}$$

which establishes the induction step. The base case is given by the fact that $\mu_0 = q_0 = p_{\text{simple}}$, and so induction yields that $q_J = \mu_J$. Finally, the terminal identity for the reference construction gives

$$\pi_{\#} q_J = \pi_{\#} \mu_J = \mathrm{Law}(\pi(Y_{J,0})) = \mathrm{Law}(X_1) = p_{\text{data}}.$$

□

D ADDITIONAL RESULTS

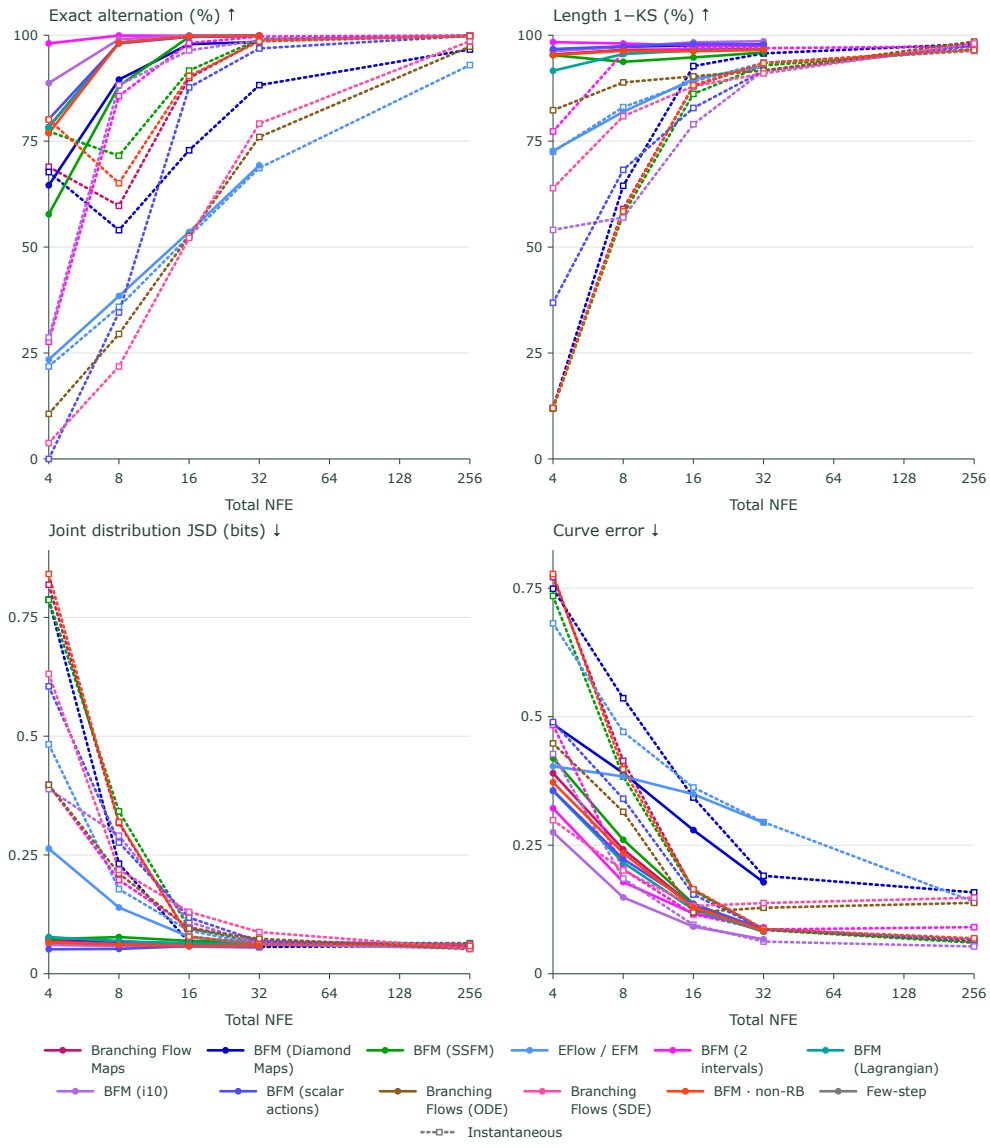


Figure 6: **Ablations, variations, and comparisons: Performance on variable-length 'alternating sinusoid'**. See table 2 and E.7 for details

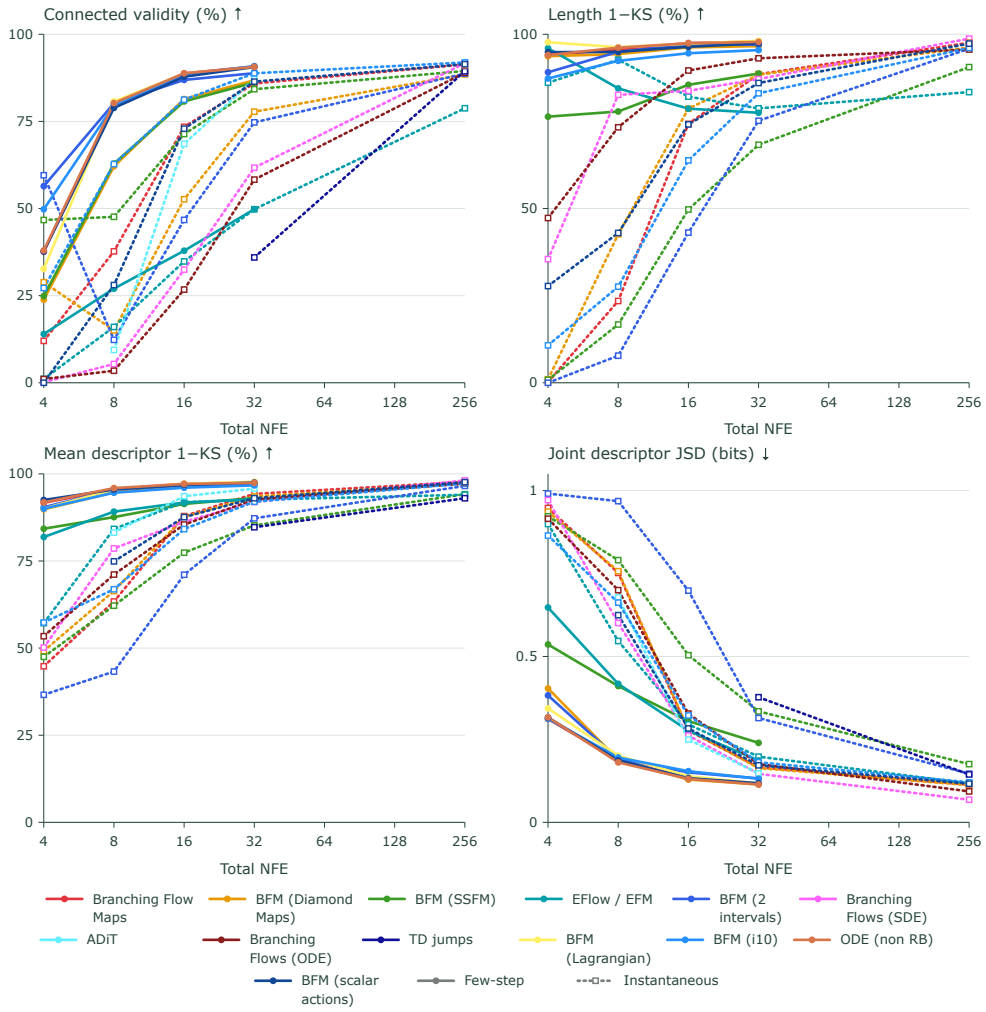


Figure 7: **Ablations, variations, and comparisons: Performance on QM9 sampling**. See 3 amd 4 and E.6 for more details.

Table 2: Comparison of selected Sinusoid methods across inference budgets. Darker shading indicates the best result within each NFE budget and lighter shading the second-best distinct value; equal values share the same rank. Rankings use unrounded values and follow the arrows. Columns without a preferred direction are not ranked. Dashes indicate missing values or observations excluded from that metric comparison. All rows use EMA weights. Each row is one saved evaluation; evaluations are not averaged.

NFE	Model	Exact alternation (%) ↑	Length 1-KS (%) ↑	Joint distribution JSD (bits) ↓	Curve error ↓
4	Branching Flow Maps (few-step)	78.360	95.440	.071	.390
	Branching Flow Maps (instant)	68.940	11.940	.819	.772
	BFM (Diamond Maps) (few-step)	64.580	96.400	.076	.485
	BFM (Diamond Maps) (instant)	67.700	12.000	.787	.749
	BFM (SSFM) (few-step)	57.740	95.280	.073	.419
	BFM (SSFM) (instant)	77.280	11.920	.787	.735
	EFlow / EFM (few-step)	23.440	72.700	.263	.403
	EFlow / EFM (instant)	21.820	72.500	.483	.681
	BFM (2 intervals) (few-step)	**98.060**	**98.360**	.066	.322
	BFM (2 intervals) (instant)	27.660	77.280	.397	.484
	BFM (Lagrangian) (few-step)	78.320	91.600	.077	.356
	BFM (i10) (few-step)	88.680	96.200	.061	**.275**
	BFM (i10) (instant)	28.660	54.060	.389	.427
	BFM (scalar actions) (few-step)	80.180	96.740	**.051**	.357
	BFM (scalar actions) (instant)	.000	36.860	.605	.489
	Branching Flows (ODE) (instant)	10.640	82.280	.398	.448
	Branching Flows (SDE) (instant)	3.760	63.900	.631	.299
	BFM · non-RB (few-step)	76.840	95.320	.066	.372
	BFM · non-RB (instant)	80.100	11.980	.841	.777
8	Branching Flow Maps (few-step)	98.200	96.560	.066	.242
	Branching Flow Maps (instant)	59.740	58.960	.317	.414
	BFM (Diamond Maps) (few-step)	89.520	97.300	.068	.390
	BFM (Diamond Maps) (instant)	54.000	64.460	.232	.536
	BFM (SSFM) (few-step)	88.040	93.720	.077	.261
	BFM (SSFM) (instant)	71.560	57.640	.342	.383
	EFlow / EFM (few-step)	38.440	81.920	.139	.384
	EFlow / EFM (instant)	35.900	83.020	.178	.470
	BFM (2 intervals) (few-step)	**99.920**	**98.040**	.062	.178
	BFM (2 intervals) (instant)	85.640	95.820	.197	.203
	BFM (Lagrangian) (few-step)	98.240	95.560	.068	.216
	BFM (i10) (few-step)	99.040	97.320	.058	**.149**
	BFM (i10) (instant)	88.220	56.980	.291	.185
	BFM (scalar actions) (few-step)	98.060	97.360	**.052**	.223
	BFM (scalar actions) (instant)	34.580	68.220	.276	.340
	Branching Flows (ODE) (instant)	29.500	88.860	.211	.315
	Branching Flows (SDE) (instant)	21.820	80.880	.219	.201
	BFM · non-RB (few-step)	98.220	96.320	.061	.235
	BFM · non-RB (instant)	65.080	58.440	.319	.398
16	Branching Flow Maps (few-step)	99.780	96.600	.064	.131
	Branching Flow Maps (instant)	89.920	87.960	.078	.163
	BFM (Diamond Maps) (few-step)	97.900	97.560	.061	.279
	BFM (Diamond Maps) (instant)	72.840	92.720	.066	.343
	BFM (SSFM) (few-step)	99.600	94.760	.069	.136
	BFM (SSFM) (instant)	91.640	86.160	.096	.162
	EFlow / EFM (few-step)	53.560	89.620	.078	.349
	EFlow / EFM (instant)	52.560	89.060	.091	.362
	BFM (2 intervals) (few-step)	99.860	97.580	.062	.117
	BFM (2 intervals) (instant)	98.140	97.080	.098	.116
	BFM (Lagrangian) (few-step)	99.860	96.380	.064	.125
	BFM (i10) (few-step)	**99.920**	**98.320**	**.058**	**.092**
	BFM (i10) (instant)	96.420	78.980	.109	.095
	BFM (scalar actions) (few-step)	99.600	97.960	.058	.136
	BFM (scalar actions) (instant)	87.680	82.820	.118	.154
	Branching Flows (ODE) (instant)	52.640	90.300	.095	.120
	Branching Flows (SDE) (instant)	52.160	87.960	.130	.132
	BFM · non-RB (few-step)	99.760	96.220	.058	.129
	BFM · non-RB (instant)	90.380	88.180	.078	.164

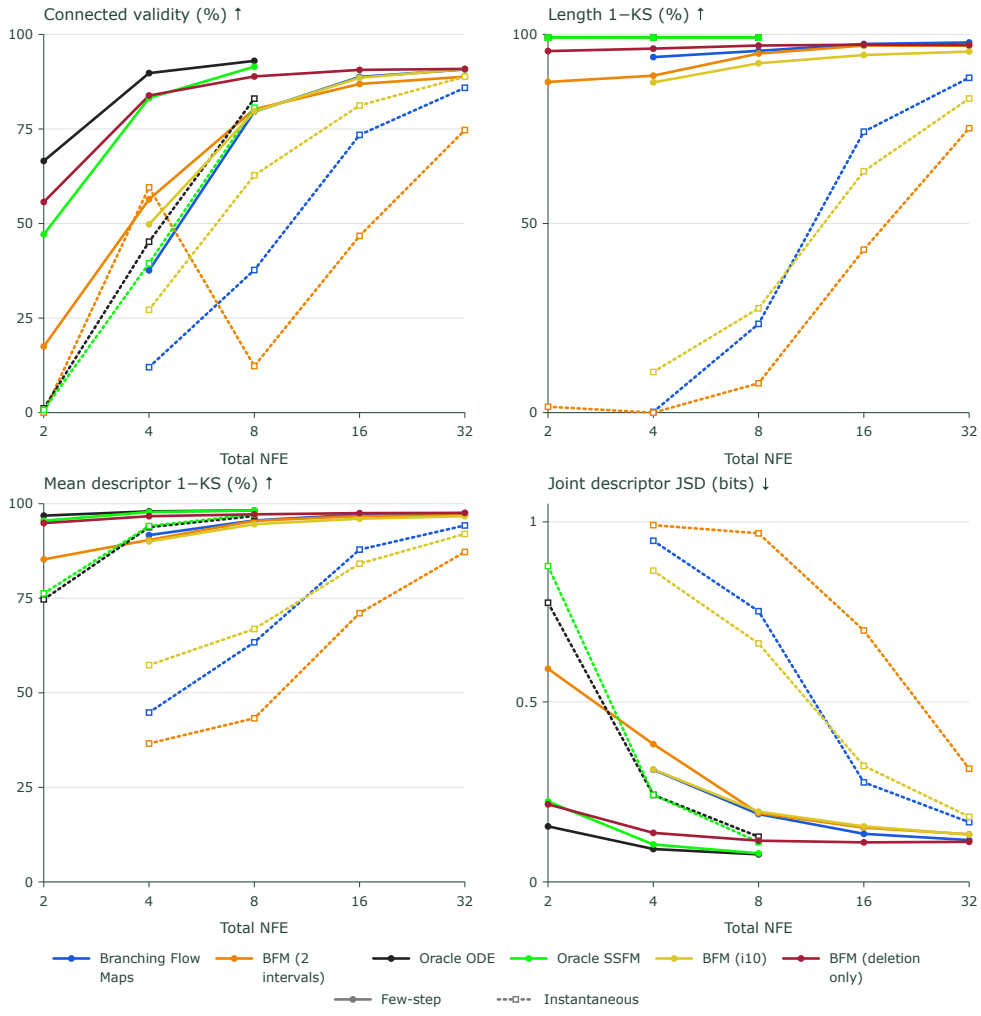


Figure 8: **Stage count and population construction affect few-step performance.** QM9 comparisons between four-stage and two-stage BFM, single-stage oracle-length maps, and single-stage deletion-only BFM. The oracle models receive the molecular length distribution in advance; deletion-only BFM learns the output length by removing elements from an overcomplete initial population.

Table 3: Comparison of selected QM9 methods across inference budgets. Darker shading indicates the best result within each NFE budget and lighter shading the second-best distinct value; equal values share the same rank. Rankings use unrounded values and follow the arrows. Columns without a preferred direction are not ranked. Dashes indicate missing values or observations excluded from that metric comparison. Each row is one saved evaluation; evaluations are not averaged.

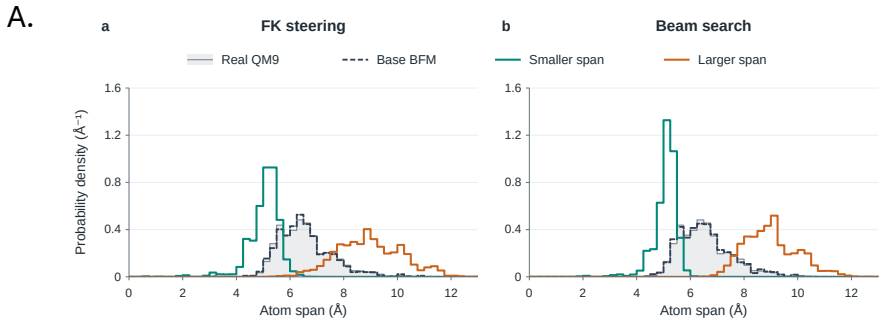
NFE	Model	Connected validity (%) ↑	Length 1-KS (%) ↑	Mean descriptor 1-KS (%) ↑	Joint descriptor JSD (bits) ↓
4	Branching Flow Maps (few-step)	37.610	94.010	91.696	**.312**
	Branching Flow Maps (instant)	12.020	.190	44.789	.948
	BFM (Diamond Maps) (few-step)	23.840	93.710	89.822	.404
	BFM (Diamond Maps) (instant)	28.850	1.000	49.173	.936
	BFM (SSFM) (few-step)	24.860	76.360	84.259	.536
	BFM (SSFM) (instant)	46.710	.840	47.496	.923
	EFlow / EFM (few-step)	13.980	96.000	81.878	.648
	EFlow / EFM (instant)	1.120	86.110	57.159	.899
	BFM (2 intervals) (few-step)	56.410	89.100	90.415	.383
	BFM (2 intervals) (instant)	**59.550**	0	36.602	.991
	Branching Flows (SDE) (instant)	.150	35.480	50.156	.971
	Branching Flows (ODE) (instant)	1.070	47.280	53.421	.915
	BFM (Lagrangian) (few-step)	32.670	**97.720**	90.046	.344
	BFM (i10) (few-step)	49.790	87.320	90.018	.312
	BFM (i10) (instant)	27.200	10.750	57.333	.864
	BFM (scalar actions) (few-step)	37.660	94.860	**92.498**	.316
	ODE (non RB) (few-step)	37.980	94.020	91.824	.317
	BFM (scalar actions) (instant)	.000	27.750	–	–
8	Branching Flow Maps (few-step)	79.580	95.660	95.579	.188
	Branching Flow Maps (instant)	37.710	23.470	63.365	.752
	BFM (Diamond Maps) (few-step)	62.120	94.310	94.688	.189
	BFM (Diamond Maps) (instant)	15.040	42.610	66.389	.757
	BFM (SSFM) (few-step)	62.890	77.840	87.588	.411
	BFM (SSFM) (instant)	47.620	16.720	62.178	.790
	EFlow / EFM (few-step)	27.020	84.470	89.158	.417
	EFlow / EFM (instant)	16.030	92.700	84.200	.547
	BFM (2 intervals) (few-step)	80.220	94.950	95.477	.191
	BFM (2 intervals) (instant)	12.320	7.770	43.273	.968
	Branching Flows (SDE) (instant)	5.390	82.620	78.599	.601
	ADiT (instant)	9.410	–	83.153	.678
	Branching Flows (ODE) (instant)	3.450	73.310	71.101	.700
	BFM (Lagrangian) (few-step)	**80.720**	**96.280**	95.465	.201
	BFM (i10) (few-step)	79.630	92.380	94.583	.195
	BFM (i10) (instant)	62.750	27.580	66.880	.662
	BFM (scalar actions) (few-step)	78.890	94.960	95.498	.187
	ODE (non RB) (few-step)	80.260	96.160	**95.915**	**.181**
	BFM (scalar actions) (instant)	28.040	42.980	74.890	.625
16	Branching Flow Maps (few-step)	88.770	**97.480**	97.144	.134
	Branching Flow Maps (instant)	73.440	74.290	87.872	.277
	BFM (Diamond Maps) (few-step)	81.410	96.240	96.793	**.129**
	BFM (Diamond Maps) (instant)	52.660	78.640	87.613	.279
	BFM (SSFM) (few-step)	80.730	85.500	91.369	.305
	BFM (SSFM) (instant)	71.380	49.720	77.396	.504
	EFlow / EFM (few-step)	37.890	78.660	91.774	.276
	EFlow / EFM (instant)	34.830	82.100	91.869	.295
	BFM (2 intervals) (few-step)	86.910	97.060	96.737	.151
	BFM (2 intervals) (instant)	46.730	43.090	71.055	.698
	Branching Flows (SDE) (instant)	32.470	83.770	86.251	.262
	ADiT (instant)	68.550	–	93.590	.250
	Branching Flows (ODE) (instant)	26.740	89.600	85.456	.328
	BFM (Lagrangian) (few-step)	88.690	97.300	**97.147**	.139
	BFM (i10) (few-step)	88.530	94.560	96.030	.154
	BFM (i10) (instant)	81.230	63.820	84.145	.322
	BFM (scalar actions) (few-step)	87.820	96.400	96.847	.133
	ODE (non RB) (few-step)	**88.800**	97.410	97.098	.131
	BFM (scalar actions) (instant)	72.910	74.140	87.586	.283

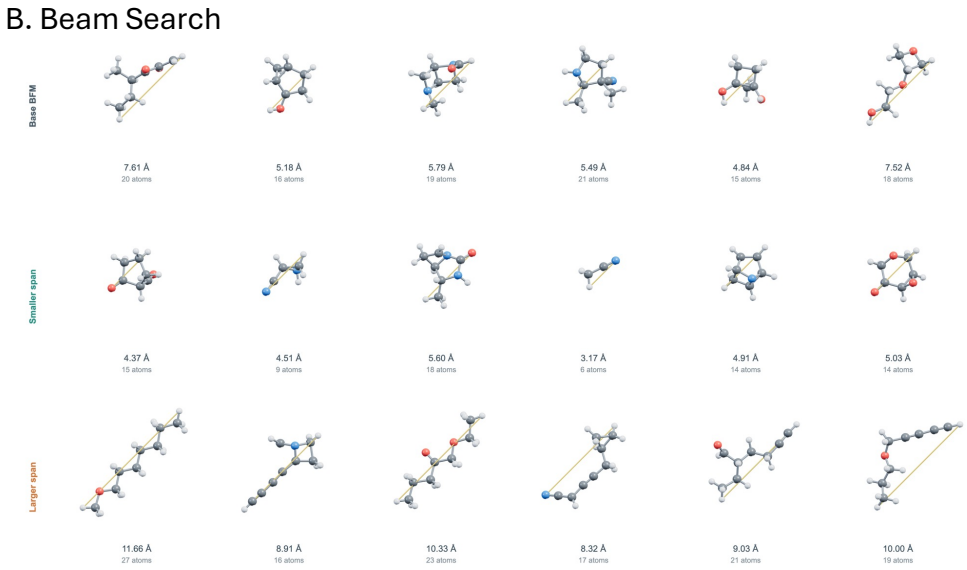
Table 4: Comparison of selected QM9 methods across inference budgets, continued. Darker shading indicates the best result within each NFE budget and lighter shading the second-best distinct value; equal values share the same rank. Rankings use unrounded values and follow the arrows. Columns without a preferred direction are not ranked. Dashes indicate missing values or observations excluded from that metric comparison. Each row is one saved evaluation; evaluations are not averaged.

NFE	Model	Connected validity (%) ↑	Length 1-KS (%) ↑	Mean descriptor 1-KS (%) ↑	Joint descriptor JSD (bits) ↓
32	Branching Flow Maps (few-step)	90.740	97.880	97.522	.116
	Branching Flow Maps (instant)	85.890	88.540	94.256	.166
	BFM (Diamond Maps) (few-step)	86.800	96.600	97.162	.114
	BFM (Diamond Maps) (instant)	77.780	88.400	93.556	.164
	BFM (SSFM) (few-step)	86.160	88.730	93.055	.240
	BFM (SSFM) (instant)	84.240	68.290	85.210	.335
	EFlow / EFM (few-step)	49.800	77.480	92.705	.187
	EFlow / EFM (instant)	49.790	78.740	92.572	.198
	BFM (2 intervals) (few-step)	88.830	97.040	96.944	.132
	BFM (2 intervals) (instant)	74.720	75.190	87.241	.314
	Branching Flows (SDE) (instant)	61.710	87.140	91.919	.146
	ADiT (instant)	87.140	–	95.757	.148
	Branching Flows (ODE) (instant)	58.300	93.110	92.603	.176
	TD jumps (instant)	35.970	–	84.688	.377
	BFM (Lagrangian) (few-step)	90.530	**98.110**	**97.708**	.117
	BFM (i10) (few-step)	**90.850**	95.460	96.695	.131
	BFM (i10) (instant)	88.810	83.070	92.012	.181
	BFM (scalar actions) (few-step)	90.600	97.400	97.506	.118
	ODE (non RB) (few-step)	90.640	97.810	97.428	**.114**
	BFM (scalar actions) (instant)	86.410	86.020	93.010	.172
256	Branching Flow Maps (instant)	91.350	97.560	97.694	.113
	BFM (Diamond Maps) (instant)	88.490	96.190	97.150	.112
	BFM (SSFM) (instant)	89.460	90.580	94.155	.176
	EFlow / EFM (instant)	78.770	83.410	93.996	.117
	BFM (2 intervals) (instant)	88.710	95.870	96.551	.147
	Branching Flows (SDE) (instant)	91.700	**98.760**	**98.099**	**.068**
	Branching Flows (ODE) (instant)	88.980	95.620	97.247	.093
	TD jumps (instant)	89.390	–	92.999	.144
	BFM (i10) (instant)	**91.920**	96.080	97.265	.121
	ODE (non RB) (instant)	91.090	97.620	97.576	.114
	BFM (scalar actions) (instant)	91.420	97.320	97.618	.117

Table 5: Comparison of BFM against oracle length and deletion only flow map models across inference budgets. Darker shading indicates the best result within each NFE budget and lighter shading the second-best distinct value; equal values share the same rank. Rankings use unrounded values and follow the arrows. Columns without a preferred direction are not ranked. Dashes indicate missing values or observations excluded from that metric comparison. All rows use EMA weights. Each row is one saved evaluation; evaluations are not averaged.

NFE	Model	Connected validity (%) ↑	Length 1-KS (%) ↑	Mean descriptor 1-KS (%) ↑	Joint descriptor JSD (bits) ↓
2	BFM (2 intervals) (few-step)	17.510	87.440	85.286	.592
	BFM (2 intervals) (instant)	.000	1.560	–	–
	Oracle ODE (few-step)	**66.550**	**99.150**	**96.849**	**.154**
	Oracle ODE (instant)	1.130	**99.150**	74.699	.775
	Oracle SSFM (few-step)	47.140	**99.150**	95.501	.223
	Oracle SSFM (instant)	.730	**99.150**	76.283	.877
	BFM (deletion only) (few-step)	55.700	95.600	94.874	.215
4	Branching Flow Maps (few-step)	37.610	94.010	91.696	.312
	Branching Flow Maps (instant)	12.020	.190	44.789	.948
	BFM (2 intervals) (few-step)	56.410	89.100	90.415	.383
	BFM (2 intervals) (instant)	59.550	0	36.602	.991
	Oracle ODE (few-step)	**89.770**	**99.150**	**97.980**	**.091**
	Oracle ODE (instant)	45.190	**99.150**	93.795	.242
	Oracle SSFM (few-step)	83.180	**99.150**	97.796	.104
	Oracle SSFM (instant)	39.470	**99.150**	94.083	.242
	BFM (i10) (few-step)	49.790	87.320	90.018	.312
	BFM (i10) (instant)	27.200	10.750	57.333	.864
	BFM (deletion only) (few-step)	83.850	96.250	96.699	.136
8	Branching Flow Maps (few-step)	79.580	95.660	95.579	.188
	Branching Flow Maps (instant)	37.710	23.470	63.365	.752
	BFM (2 intervals) (few-step)	80.220	94.950	95.477	.191
	BFM (2 intervals) (instant)	12.320	7.770	43.273	.968
	Oracle ODE (few-step)	**93.010**	**99.150**	98.217	**.077**
	Oracle ODE (instant)	83.030	**99.150**	96.753	.125
	Oracle SSFM (few-step)	91.470	**99.150**	**98.224**	.079
	Oracle SSFM (instant)	80.620	**99.150**	97.202	.111
	BFM (i10) (few-step)	79.630	92.380	94.583	.195
	BFM (i10) (instant)	62.750	27.580	66.880	.662
	BFM (deletion only) (few-step)	88.890	97.070	97.191	.115
16	Branching Flow Maps (few-step)	88.770	**97.480**	97.144	.134
	Branching Flow Maps (instant)	73.440	74.290	87.872	.277
	BFM (2 intervals) (few-step)	86.910	97.060	96.737	.151
	BFM (2 intervals) (instant)	46.730	43.090	71.055	.698
	BFM (i10) (few-step)	88.530	94.560	96.030	.154
	BFM (i10) (instant)	81.230	63.820	84.145	.322
	BFM (deletion only) (few-step)	**90.630**	97.290	**97.513**	**.110**
32	Branching Flow Maps (few-step)	90.740	**97.880**	97.522	.116
	Branching Flow Maps (instant)	85.890	88.540	94.256	.166
	BFM (2 intervals) (few-step)	88.830	97.040	96.944	.132
	BFM (2 intervals) (instant)	74.720	75.190	87.241	.314
	BFM (i10) (few-step)	90.850	95.460	96.695	.131
	BFM (i10) (instant)	88.810	83.070	92.012	.181
	BFM (deletion only) (few-step)	**90.890**	97.190	**97.577**	**.111**





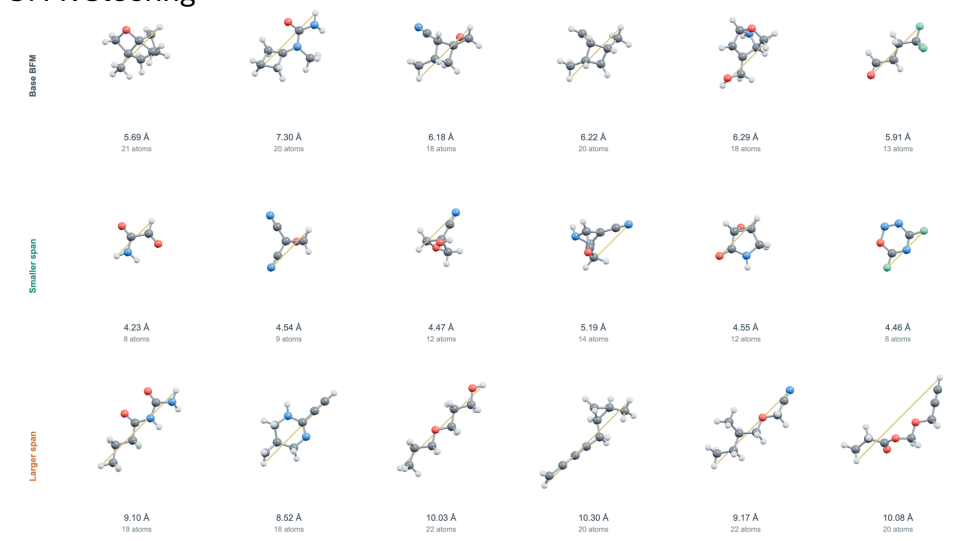


Figure 9: **Reward tilting: QM9**. **A.** shows length histograms from reward tilted sampling of QM9 molecules, either favoring short of long span (the maximum distance between two atoms) using either FK Steering or Beam Search. B. and C. show random molecules sampled from base, tilted-down, and tilted-up spans.

E MULTIMODAL EXPERIMENTAL TASKS, EVALUATION, AND MODEL CONFIGURATIONS

This section specifies the alternating-sinusoid and QM9 experiments, where a variety of model variants were compared. We first describe the tasks and evaluation criteria, then the Branching Flow Map (BFM) process, architectures, and training procedure. Finally, we describe each comparison method and its departures from this setup. We distinguish instantaneous models, which are numerically integrated, from learned few-step finite-time maps.

E.1 ALTERNATING-SINUSOID TASK

Each observation is an ordered, variable-length sequence of three-dimensional points and binary labels. To generate its length, we draw

$$N = 2 + Z, \qquad Z \sim \tfrac{1}{2}\,\mathrm{Poisson}(4) + \tfrac{1}{2}\,\mathrm{Poisson}(20). \tag{6}$$

Independently, we draw the left endpoint $L \sim \mathcal{U}[-1, 0]$, the right endpoint $R \sim \mathcal{U}[0, 1]$, and a binary phase $B \sim \mathrm{Bernoulli}(1/2)$. For $i = 0, \ldots, N-1$, the points and labels are

$$x_i = L + \frac{i}{N-1}(R - L), \qquad \mathbf{r}_i = (x_i, x_i + \sin(3x_i), 0), \qquad h_i = (i + B) \bmod 2. \tag{7}$$

Thus a successful model must reproduce a bimodal population length distribution, random spatial extent, ordered uniform spacing, and an alternating discrete pattern.

We use 100,000 training examples and independent validation and test sets of 10,000 examples each. Evaluation uses 5,000 generated sequences per method and NFE budget, compared with the same 5,000 test examples. Evaluation preserves the generated order rather than sorting points by their horizontal coordinate.

Metrics.

- **Exact alternation (%, ↑):** Percentage of generated sequences whose binary labels alternate perfectly, allowing either initial state.
- **Length $1 - \mathrm{KS}$ (%, ↑):** Agreement between generated and reference length distributions, measured by
$$S_{\mathrm{KS}} = 1 - \sup_z |\widehat{F}_{\mathrm{gen}}(z) - \widehat{F}_{\mathrm{ref}}(z)|. \tag{8}$$
We display $100 S_{\mathrm{KS}}$ for the length comparison, where $\widehat{F}$ is the empirical length CDF.
- **Joint distribution JSD (bits, ↓):** Equal-mixture Jensen–Shannon divergence between generated and reference joint distributions of sequence length and endpoints, (N, x_0, x_{N-1}), using exact integer lengths and four equal-width bins on each endpoint's target range ($[-1, 0]$ and $[0, 1]$). For empirical distributions P and Q, we use
$$\mathrm{JSD}(P, Q) = \tfrac{1}{2}\,\mathrm{KL}_2(P \| M) + \tfrac{1}{2}\,\mathrm{KL}_2(Q \| M), \qquad M = \tfrac{1}{2}(P + Q), \tag{9}$$
where KL_2 uses base-two logarithms.
- **Curve error (↓):** Vertical deviation from the target curve, measured by the residuals $y_i - [x_i + \sin(3x_i)]$.

E.2 QM9 TASK AND MOLECULAR EVALUATION

We study unconditional generation of molecular coordinates and atom types on QM9 (Ramakrishnan et al., 2014), retaining explicit hydrogen atoms and the five physical elements H, C, N, O, and F. We use the QM9 partition associated with Equivariant Diffusion (Hoogeboom et al., 2022): 100,000 training molecules, 17,748 validation molecules, and 13,083 test molecules. All models trained for our comparisons use the same training examples and the same coordinate and atom representation. The externally pretrained ADiT and jump-diffusion models are exceptions to the shared training setup, as detailed below.

Coordinates are measured in Å. Molecules are centered by their mean atomic position before augmentation. Heavy atoms retain a fixed order, with each hydrogen placed after its nearest heavy atom. During training we apply random proper rotations and independent standard-Gaussian translations. The transformer is not constrained to be rotation equivariant; augmentation supplies this inductive bias statistically. The BFM population capacity is 64, whereas the QM9 molecules themselves contain at most 29 atoms. Each evaluation contains 10,000 generated molecules. Distributional comparisons use a single fixed selection of 10,000 distinct test molecules, shared across methods and budgets.

Validity and connected validity. We use the molecular reconstruction and chemical-validity procedure of ADiT (Joshi et al., 2025), with explicit hydrogens retained. Atomic numbers and coordinates undergo OpenBabel bond perception and RDKit reconstruction and sanitization through the same conversion procedure. Let $M = 10{,}000$, let V denote the number accepted by this chemical-validity procedure, and let C denote the number that are additionally a single connected component. We report

$$\text{Validity} = \frac{V}{M}, \qquad \text{Connected validity} = \frac{C}{M}. \tag{10}$$

The same procedure accepts 9,409 of the 10,000 reference molecules as connected and valid.

Chemical descriptor distributions. We compare the ten properties used by Billera et al. (2026b): molecular weight, estimated octanol/water partition coefficient (logP), hydrogen-bond donor count, hydrogen-bond acceptor count, the five individual element counts (H, C, N, O, F), and total atom count including hydrogens. Molecular weight is the RDKit molecular-weight descriptor in daltons; logP and hydrogen-bond counts are calculated from the reconstructed chemical graph. For each descriptor we report the similarity in Eq. equation 8, together with the unweighted mean of these ten similarities.

These ten chemical-descriptor distributions are conditioned on *connected validity*. Their generated denominator is C, and their reference denominator is 9,409.

We additionally report the joint ten-dimensional descriptor JSD using Eq. equation 9. Each axis is discretized at the reference distribution's 25th, 50th, and 75th percentiles. Repeated boundaries are merged, giving at most four bins per axis; ties enter the bin to the right and the outer bins include the tails. Generated and reference joint-cell frequencies are normalized by their respective connected-valid counts.

Statistics without a validity filter. Population size and elemental composition do not require a reconstructed chemical graph. We therefore also calculate separate $1 - \mathrm{KS}$ scores for total atom count, H/C/N/O/F counts, heavy-atom count, non-QM9-label count, and the sum of the standard atomic weights of the explicitly represented atoms. Their reference distribution uses all 10,000 test molecules.

E.3 Branching Flow Map process

The standard BFM uses four equal stages with boundaries $\tau_j = j/4$. Generation begins with one element. Each live element carries Cartesian coordinates, a relaxed label vector, and an 11-dimensional relaxed action vector encoding offspring counts $c \in \{0, \ldots, 10\}$. Count zero denotes deletion and count one denotes persistence. The final boundary allows only zero or one offspring. Initial coordinates have unit Gaussian covariance and mean $(0, 0, 0)$ for QM9 or $(1, -1, 0)$ for the sinusoid (part of the task specification, to make the path visualization clearer). Initial relaxed labels and actions are independent standard Gaussians.

Conditional forests and deletion augmentation. For a target with N elements, we add $\mathrm{Poisson}(0.5N)$ 'to be deleted' element copies. Copies are placed next to sampled target elements. We construct ordered descendant groups forward through the stage boundaries. A group of w augmented leaves produces $1 + \mathrm{Binomial}(w - 1, p_j)$ children, conditioned on the offspring bound and the remaining leaf capacity, where $p_j = [G(\tau_{j+1}) - G(\tau_j)]/[1 - G(\tau_j)]$. Feasible contiguous

partitions are sampled uniformly. The growth clock has cumulative distribution

$$G(t) = \frac{6^{t/(3/4)} - 1}{6 - 1}, \qquad 0 \le t \le 3/4, \tag{11}$$

and is one thereafter. A marked lineage, once isolated as a singleton, receives a uniform death time over its remaining lifetime, rounded up to the next boundary. Its ancestral endpoints continue to average over the full augmented descendant group, including marked copies.

Appendix B.4 gives the tree construction and an intuitive explanation of the anchors and action targets.

Within-stage dynamics and boundary kernels. Coordinates and relaxed labels move toward their augmented descendant-average endpoint with global deadline one. Actions interpolate from fresh Gaussian noise to the next offspring encoding with deadline τ_{j+1}. For a channel v with deadline d_v and conditional endpoint β_v, this gives

$$Y_t^v = Y_{\tau_j}^v + \frac{t - \tau_j}{d_v - \tau_j}(\beta_v - Y_{\tau_j}^v), \qquad d_v = \begin{cases} 1, & v \in \{\text{coordinates}, \text{labels}\}, \\ \tau_{j+1}, & v = \text{actions}. \end{cases} \tag{12}$$

At an internal boundary, decoded counts determine which elements are deleted, retained, or copied. Every surviving child, including a unary continuation, receives independent Gaussian perturbations with standard deviation 0.2 in both coordinates and relaxed labels. Actions are independently reset to standard Gaussian noise. The terminal boundary adds no noise. Counts at stage boundaries and final discrete state labels are decoded by elementwise argmax.

Rao–Blackwellized action supervision. The instantaneous coordinate endpoint is trained by squared error and the physical-label endpoint by soft-target cross-entropy. For actions, we integrate out the local discrete event under the forest's conditional arity distribution. If u is local stage time, $a_u = (1-u)\eta + ue(c)$, and $\pi(c \mid \zeta)$ is the feasible local prior given training-only descendant information ζ, then

$$q(c \mid a_u, \zeta) \;\propto\; \pi(c \mid \zeta) \exp\left[-\frac{\|a_u - ue(c)\|^2}{2(1-u)^2} \right]. \tag{13}$$

Cross-entropy to this posterior replaces supervision by a single sampled offspring one-hot.

E.4 Branching Flow Map architectures

Both sinusoid and QM9 models use a packed, bidirectional transformer with the same architectural pattern at different scales. Current coordinates are embedded using two fixed Gaussian random-Fourier feature banks, at frequency scales 1 and 0.1. Relaxed labels and actions have linear embeddings. Fourier embeddings of global time, stage-local time, and a learned stage embedding, drive the conditioning in adaptive layer normalization. Each block contains multi-head attention with rotary positional embeddings, RMS-normalized queries and keys, gated attention outputs, and a SwiGLU feed-forward network of width $4d$. Attention scores are smoothly capped at magnitude 50.

Attention also receives 128 pair features: 64 from the current coordinate state and 64 from the current coordinate endpoint estimate. Each geometry contributes multiscale directional features and radial features, including 46 learned radial basis functions encoding pairwise distance. The final half of the instantaneous model blocks progressively refine the coordinate endpoint. RMS-normalized output features produce coordinate corrections and final label and action logits.

Few-step refinement. The finite-time model augments the frozen instantaneous model with a trainable refinement branch initialized by copying its endpoint-refinement blocks. Learned linear projections route features from every frozen instantaneous model depth to every refinement block. The branch additionally receives the current state and Fourier features of $(s, t, t-s)$. Its coordinate heads refine the endpoint after each block, while its label and action heads correct the final logits. Corrections are initialized to zero, so the initial finite model agrees with the instantaneous model.

Quantity	QM9	Sinusoid
Transformer width d	384	192
Instantaneous model transformer blocks	12	4
Attention heads	12	8
Dimension per attention head	64	32
Instantaneous model endpoint-refinement blocks	6	2
Additional finite-map blocks	6	2
Lateral projection width	1,024	256
Physical label classes	5	2
Instantaneous / finite-map batch size	128 / 64	64 / 64
Instantaneous / finite-map updates	800,000 / 300,000	500,000 / 400,000
Instantaneous / finite-map terminal warmdown	100,000 / 50,000	100,000 / 50,000
EMA decay	0.9999	0.999

Table 6: Branching Flow Map QM9 and sinusoid architecture and training settings. The finite-map blocks are additional to a permanently frozen instantaneous model. EMA denotes exponential moving average. Training budgets exclude pilot updates used for loss calibration.

E.5 Branching Flow Map training and sampling

Instantaneous training. We first train the instantaneous endpoint model from scratch, using the budgets in Table 6. Training examples and stage indices are sampled uniformly. Conditional states are generated from the augmented forests. Local time is zero with probability $1/8$ and is otherwise uniform. Coordinate loss is the mean squared error per coordinate. Label and action losses are cross-entropies to their conditional endpoint targets. Each component is averaged over the live elements within each example and then over examples.

Loss calibration. We balance the three components separately for each training phase. A disposable 1,000-update pilot is followed by gradient measurements on 512 training batches: four stages, eight time bins, and sixteen repeats per cell. The bins describe local time for the instantaneous model and local duration for the finite model. For each component and cell, its fixed weight is the inverse median gradient norm on the shared trainable transformer blocks. The pilot model and optimizer state are then discarded, and training restarts from the phase's original initialization with this fixed loss weighting.

Semigroup training of finite maps. For each example, we choose local map endpoints using a mixture of 10% full-stage jumps, 15% partial jumps starting at the stage boundary, and 75% ordered interior pairs. Nonzero spans have minimum local duration 0.001. The finite map for each channel has the endpoint form

$$\Phi^v_{s,t}(y) = y^v + \frac{t-s}{d_v - s}\big(P^v_\theta(s,t,y) - y^v\big). \tag{14}$$

Here s, t, d_v are expressed in a common time coordinate; in global time the deadlines are those specified in Section E.3.

The first 1,000 production updates use endpoint targets obtained by integrating the frozen instantaneous model through 16 substeps over the sampled span. Subsequently, one quarter of updates retain these teacher targets, while three quarters use two half-span maps from the finite model's EMA, following the self-distillation principle of Boffi et al. (2025). If $u = (s+t)/2$, the latter target is

$$Q^v = \gamma_v P^v_{\bar\theta}(s,u,y) + (1-\gamma_v) P^v_{\bar\theta}\big(u,t,\Phi^{\bar\theta}_{s,u}(y)\big), \tag{15}$$

$$\gamma_v = \frac{(d_v - t)(u-s)}{(d_v - u)(t-s)}. \tag{16}$$

Targets and their intermediate states are detached from differentiation. These convex endpoint combinations preserve valid soft categorical targets. The student uses coordinate MSE and label/action cross-entropy, with the same per-example reduction and a newly measured finite-phase calibration.

Optimization. Matrix-valued parameters use Muon with momentum 0.95 and Nesterov acceleration; input/output exceptions, normalization parameters, biases, and other nonmatrix parameters

use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$. AdamW's learning rate is 0.015 times the Muon base rate. Both use weight decay 0.01, and no gradient clipping is applied. The base learning rate rises from 10^{-4} to 10^{-3} over approximately 170 updates, increasing by a factor 1.15 every ten updates, and then decays exponentially. For the sinusoid instantaneous model and the finite-map phases, the decay factor is 0.99995 per ten updates, with a 10^{-4} floor before warmdown. For the QM9 instantaneous model, the corresponding decay is stretched from 400,000 to 700,000 updates. Each phase finishes with linear warmdown to zero over the number of updates in Table 6; these updates are included in the stated budgets. Training uses BF16 mixed precision with FP32 parameter and optimizer state. EMA models are used for evaluation, with the task-specific decay in the table. The frozen instantaneous model inside the finite model is not updated.

Sampling budgets. We use equally spaced evaluations inside each fixed stage, applying the population kernel only at stage boundaries. With four stages, one, two, four, or eight map calls per stage give 4, 8, 16, or 32 total neural-function evaluations (NFE). The many-step instantaneous evaluation uses 256 total NFE.

E.6 QM9 COMPARISON METHODS

Branching Flow Maps. This is the four-stage, one-initial-element QM9 model specified above: constant 0.2 coordinate and relaxed-label noise at internal boundaries, deletion augmentation coefficient 0.5, an 800,000-update instantaneous model, and a 300,000-update semigroup-trained finite model with its instantaneous model core frozen. Instantaneous and few-step curves use the respective final EMA predictions.

EFlow / EFM. We adapt Expanding Generative Flows and Expanding Flow Maps (Tang & Chatterjee, 2026) to the joint coordinate/label task. The process begins empty and learns insertions into ordered gaps, rather than splits and deletions of existing elements. Conditional birth times have cumulative distribution $\alpha(t) = \sqrt{t}$; an element born at a follows its own local clock $(t-a)/(1-a)$. Coordinate and relaxed-label transport starts from Gaussian noise. There are no BFM action states, ghost lineages, or fixed event boundaries. Instantaneous model training uses coordinate-velocity regression, label cross-entropy, and a hazard-weighted Poisson gap-count loss. EFlow sampling uses Poisson insertions; EFM uses per-gap binomial insertion counts with a finite population budget.

The backbone scale, geometric attention features, optimizer, data, batches, EMA, and instantaneous/finite training budgets match the BFM QM9 settings, with the necessary gap and birth-time interfaces replacing BFM-specific inputs and heads. EFM freezes its own raw instantaneous model and uses 75% instantaneous-diagonal distillation and 25% off-diagonal expansion/composition, with detached current-student rather than EMA targets. Calibration follows these native strata. EFlow is evaluated with explicit Euler transport. Both EFlow and EFM require two predictions per numerical step, so 2/4/8/16 steps correspond to 4/8/16/32 NFE. This is our multimodal adaptation of the expanding-flow construction.

ADiT public reproduction, oracle length. ADiT is a latent diffusion transformer whose autoencoder decodes latent atom representations into coordinates and elements (Joshi et al., 2025). We evaluate the publicly released joint QM9/materials reproduction without retraining, rather than the model underlying the original paper's reported results. Unlike BFM, it is given a population size in advance. We draw this size independently from the empirical length distribution of our 100,000 QM9 training molecules, not from test-molecule lengths. Its length score therefore measures a supplied prior, not learned population dynamics. Sampling retains classifier-free guidance of scale four, self-conditioning, and FP32 inference. We use 4/8/16/32 denoising steps; each step evaluates both conditioned and unconditioned predictions, corresponding to 8/16/32/64 per-example denoiser evaluations, plus one autoencoder decode. Physical elements outside the QM9 vocabulary are retained in evaluation rather than filtered.

Branching Flows (SDE). This baseline uses the continuous-time binary branching process with deletions of Billera et al. (2026b). Coordinates follow its Ornstein–Uhlenbeck bridge, while labels follow its discrete noisy-interpolation process with mask-valued internal anchors. Splits and deletions can occur between any two solver times; there is no fixed stage grid and no continuous BFM offspring vector. Separate heads predict the endpoint state, splitting behavior, and deletion behavior.

We use one initial element, Poisson deletion augmentation with coefficient 0.5, and split times distributed as $\mathrm{Beta}(1, 1.5)$ conditioned to lie below 0.95. The independent uniform deletion clock is retained. Training times remain uniform over the trajectory, and the original analytic time-dependent loss factors are preserved.

The QM9 architecture scale, data, augmentation, batch size, optimizer, and 800,000-update budget match the BFM instantaneous model setup, with process-specific heads. This model has a two-phase training history: its first 434,000 updates used FP32, padded sequences, atom-weighted losses, and the original fixed relative loss coefficients. It then continued with BF16, sequence packing, per-molecule loss averaging, and four globally calibrated component weights. Calibration used 512 batches at the transition without further pilot updates; optimizer and EMA state and the learning-rate clock were retained. Its final EMA is evaluated at 4/8/16/32/256/1000 NFE.

Branching Flows (ODE). This is the corresponding hybrid Branching Flows baseline in which only the coordinate process is replaced by straight-line endpoint transport. The discrete label process, mask initialization and internal labels, label inheritance, and population-event mechanism are unchanged. Each daughter at an actual split receives independent coordinate noise of standard deviation $\sigma(t) = 0.20 - 0.18t$; nonsplitting elements receive no such perturbation, and there is no Gaussian label noise. The split cutoff remains 0.95. In particular, "ODE" refers to the coordinate transport, not deterministic label evolution or the absence of random population events. This model used the same two-phase training procedure as the SDE baseline, but transitioned after 421,000 updates, continuing to 800,000 in total. The architecture and evaluation budgets are the same as for the SDE baseline.

Transdimensional Jump Diffusion. We evaluate the released unconditional QM9 model of Campbell et al. (2023), without retraining. It combines diffusion of molecular state with learned dimension-changing jumps, starting from one atom and supporting up to 29 atoms. Unlike ADiT, it learns population growth rather than receiving an oracle length. We use uniform grids with 32 or 256 steps, no corrector steps, and the model's near-atom insertion and noise-free final-step options. Each step uses one prediction, so these budgets are 32 and 256 NFE. The very low uniform-step budgets are not included because the diffusion update becomes undefined at those step sizes; the population update also permits at most one birth per step. The architecture and training procedure are those of the pretrained model.

BFM (Lagrangian). This experiment retains the four-stage QM9 BFM process and architecture but trains a new finite map for 300,000 updates using a destination-time Lagrangian condition (Boffi et al., 2025), rather than semigroup targets. Its frozen trunk is the final 800,000-update instantaneous *EMA*. No trained finite map is used as initialization. For Eq. equation 14, the condition $\partial_t \Phi_{s,t} = v(t, \Phi_{s,t})$ can be written channelwise as

$$P_\theta^v + \kappa_v \partial_t P_\theta^v = m_{\mathrm{src}}^v(t, \Phi_{s,t}), \qquad \kappa_v = \frac{(d_v - t)(t - s)}{d_v - s}. \tag{17}$$

Training uses a destination-time Jacobian–vector product of the live map and one local instantaneous model evaluation at the mapped state, with detached targets and semigradients. For categorical channels, a positive, normalized adaptive cross-entropy target is chosen to give the signed Lagrangian logit residual; signed probabilities are not clipped into a target distribution. Exact stage-end training draws are moved to 99.9% of the proposed span to avoid the action deadline singularity; inference still reaches the exact boundary. There are no teacher rollouts, teacher warmup, semigroup compositions, or refiner-EMA training targets. The finite EMA serves evaluation only. Optimizer, batch, full budget, and terminal warmdown match the QM9 BFM, with objective-specific calibration. This comparison changes both the training objective and the frozen instantaneous model from raw to EMA parameters, together with the endpoint treatment described above.

BFM (i10). The initial population is $5 + \mathrm{Poisson}(5)$, with mean ten, rather than one. Roots are sampled independently of the target length; feasible deletion augmentation accommodates excess roots. The architecture and 800,000/300,000-update budgets otherwise follow the QM9 BFM scale. However, this comparison also uses deletion augmentation coefficient 0.2 and internal coordinate/label noise scales $(0.25, 0.15, 0.05)$, rather than coefficient 0.5 and constant noise 0.2.

Oracle-length ODE and SSFM. These controls use one stage at a fixed population. At generation time, a length is sampled from the QM9 test split's empirical length distribution; no held-out coordinates or atom identities are supplied. The ODE control uses straight-line conditional paths, and the SSFM control retains the corresponding Brownian-driven state dynamics. Both omit branching, deletion, and action channels, and use the QM9 architecture scale and 800,000/300,000 instantaneous/finite update budgets.

Deletion-only BFM. This model uses a single stage and an overcomplete initial population sampled uniformly from 34 to 39 elements. Coordinates and atom identities evolve jointly with keep/delete actions $c \in \{0, 1\}$. The terminal action removes excess elements, so the model learns its output length without intermediate splitting or boundary perturbations. All candidate atoms are present throughout the continuous stage.

The QM9 Diamond Maps, SSFM, scalar-action, and non-RB variants use the corresponding process and objective changes described in Appendix E.7, at the QM9 architecture scale and training budgets. The two-stage QM9 model follows the same construction with maximum offspring count 64, matching its population capacity.

E.7 Sinusoid comparison methods

Branching Flow Maps. This is identical to the one-root, four-stage sinusoid BFM model described above, with constant 0.2 coordinate/label noise, a deletion proportion of 0.5, a 500,000-update instantaneous model, and a 400,000-update semigroup-trained few-step model. Both use the same sinusoid architecture, X_0 distribution, and evaluations from Sections E.1–E.5.

BFM (Diamond Maps). This is adapted from Diamond Maps (Holderrieth et al., 2026) but modified for our multi-stage approach. The ordered forest, the actions, the deletion proportion of 0.5, and the instantaneous/refiner scales are unchanged. We modify the coordinate and relaxed-label process such that at each stage we retain the inherited state C as conditioning information but draw a new Gaussian working state. If the global stage is $[a, b]$ and A is the augmented descendant anchor, its conditional boundary target is

$$r = (1-b)/(1-a), \qquad \alpha = r^2, \qquad \beta = b - a\alpha, \tag{18}$$

$$Z = \alpha C + \beta A + \sqrt{q}\,\xi, \qquad q = (1-b)^2(1-r^2), \qquad \xi \sim \mathcal{N}(0, I). \tag{19}$$

Within-stage, the flow interpolates from fresh noise to Z. A linear encoder provides the context C to the instantaneous model. Unlike the ODE version, no additional noise is added at the boundary. Coordinate MSE, label/action cross-entropy, the teacher/EMA-composition mixture, and the 500,000/400,000 training budgets are retained.

BFM (SSFM). This variant adapts strong stochastic flow maps (McCallum et al., 2026) to the coordinate and relaxed-label channels. Within a stage, with anchor A, these channels follow

$$dY_t = \frac{A - Y_t}{1-t}\,dt + (1-t)\,dW_t, \tag{20}$$

with independent Brownian drivers. Actions retain the BFM random-initialized ODE and Rao–Blackwellized cross-entropy supervision; they have no independent diffusion term. Attained coordinate and label states are copied at a split, without boundary jitter, after which children have independent future drivers. The finite model receives three polynomial Brownian-integral features per state component. Teacher and composed maps share the same underlying path, so the objective compares path-coupled predictions rather than independently resampled noise. The remaining architecture and 500,000/400,000 training budgets are retained. This SSFM model uses deletion augmentation coefficient 0.2, compared with 0.5 for the standard BFM.

EFlow / EFM. The insertion-only construction, birth-local clocks, native objectives, and two-predictions-per-step accounting are the same as for the QM9 EFlow/EFM comparison. Here the physical vocabulary is binary, the coordinate prior mean is $(1, -1, 0)$, and no rigid augmentation is used. The instantaneous model has width 192 and four blocks, and the finite branch has two additional blocks and lateral width 256. Instantaneous model and finite budgets are 500,000 and 400,000 updates, batch size is 64, and EMA decay is 0.999. It begins empty, has capacity 128, and does not use the BFM ghost or fixed-boundary noise mechanisms.

BFM (2 stages). We replace four stages by $[0, 1/2]$ and $[1/2, 1]$. Splitting is allowed only at the internal boundary, which uses the same independent 0.2 coordinate and label noise. The final boundary permits only keep/delete. To allow one root to reach the full population capacity in a single split, the maximum offspring count is increased from 10 to 128, expanding the categorical action head accordingly. The architecture scale, ghost coefficient, training budgets, and optimizer otherwise remain the sinusoid BFM settings. One and two evaluations per stage now give two and four total NFE. Additional evaluated budgets include 8, 16, and 32 total NFE; the instantaneous reference uses 256.

BFM (Lagrangian). This is the sinusoid counterpart of Eq. equation 17: a fresh finite refiner initialized from the final 500,000-update instantaneous EMA, trained for the full 400,000-update budget including 50,000 warmdown updates. It uses the same near-endpoint treatment and pure local Lagrangian semigradient objective as the QM9 experiment, with no teacher integration, semigroup target, or refiner-EMA target. The sinusoid architecture, data, noise/ghost settings, batch size, optimization schedule, and evaluation EMA decay are unchanged. As for QM9, the frozen EMA instantaneous model also differs from the raw instantaneous model of the semigroup-trained BFM.

BFM (i10). This variant uses $5 + \mathrm{Poisson}(5)$ initial elements, mean ten, with the sinusoid instantaneous/refiner architecture and 500,000/400,000-update budgets. As in the QM9 i10 comparison, it also uses ghost coefficient 0.2 and tapered internal noise $(0.25, 0.15, 0.05)$ in both coordinate and label channels. Consequently, initial population, deletion augmentation, and noise schedule all differ from the one-root BFM.

BFM (scalar actions). The offspring action is represented by one scalar in count units rather than an 11-dimensional one-hot relaxation. Its conditional path is $a_u = (1-u)\eta + uc$, $\eta \sim \mathcal{N}(0,1)$, and its endpoint predicts the conditional mean count μ. We use the Poisson mean loss $\mu - \bar{c}\log\mu$, where $\bar{c}$ is the scalar Rao–Blackwellized count target or a finite-map teacher/composition target. This loss estimates the required mean; it does not assert a Poisson offspring distribution and does not introduce a Poisson sampling step. The scalar posterior uses the Gaussian likelihood $\exp[-(a_u - uc)^2/(2(1-u)^2)]$ with the same feasible arity prior. The head is bounded to the stage's count range. The input count is normalized by ten and encoded with two scales of random Fourier features; the finite branch uses the same type of encoding. At a boundary, nearest-integer decoding replaces categorical argmax, subject to the same nonempty and capacity constraints. Scalar convex endpoint composition replaces categorical composition in the action channel. Coordinates, labels, forest law, noise, and the remaining architecture and training procedure are unchanged. This comparison trains both the 500,000-update instantaneous model and the 400,000-update finite model from their respective fresh initializations.

Branching Flows (ODE). This uses the continuously timed Branching Flows population process described for QM9, with straight-line coordinate transport and daughter-only split noise $0.20 - 0.18t$. Binary labels retain the original stochastic discrete process and mask state, not a relaxed-label ODE. Splits use the same beta law truncated at 0.95, deletion augmentation coefficient is 0.5, and the independent uniform deletion clock and original analytic time-dependent loss factors are retained. This sinusoid model is trained *from scratch* with BF16, sequence packing, and per-sequence loss averaging from the first production update. Four global component weights are calibrated from 512 batches after a discarded 1,000-update pilot. It uses the sinusoid backbone scale, batch size 64, EMA decay 0.999, and 500,000 instantaneous model updates including 100,000 warmdown updates. There is no additional learned finite-map stage. Sampling is evaluated at 4/8/16/32/256/1000 NFE.

Branching Flows (SDE). This is the corresponding original Ornstein–Uhlenbeck coordinate bridge and discrete-label process of Billera et al. (2026b). It has no added ODE daughter-noise rule. It shares the continuously timed forest, split cutoff, deletion augmentation, uniform training-time sampling, and preserved analytic loss factors with the sinusoid ODE baseline. The same corrected training setup is used from scratch for 500,000 updates: sinusoid-scale backbone, BF16, packing, per-sequence reduction, fresh global calibration, and the same optimizer, schedule, batch size, EMA, and evaluation budgets. Both sinusoid Branching Flows baselines therefore use the same training policy throughout, whereas the QM9 baselines change policy partway through training.

BFM (non-RB). This variant removes local posterior averaging from the instantaneous action target and supervises the sampled offspring one-hot directly with cross-entropy. It also replaces the forward local-binomial forest construction by a backward population-level construction: feasible parent counts are sampled from the split-clock law, and ordered children are grouped by uniform bounded compositions. The coordinate/label anchors still average the full augmented descendant group. The action-state interpolation, prediction head, boundary decoding, ghost coefficient 0.5, constant noise 0.2, architecture, and 500,000/400,000 training budgets are unchanged. Its finite stage still uses soft teacher and EMA semigroup targets from its own frozen raw instantaneous model. This ablation jointly changes the tree law and the action-target estimator.

F ANTIBODY SEQUENCE MODEL

Table 7: Arbitrary-span antibody infilling across inference budgets. Darker shading indicates the best result within each NFE budget and lighter shading the second best. The five quality columns report $1-\text{KS}$ agreement with the natural reference distributions; s/seq denotes seconds per sequence.

NFE	Model	Length ↑	Diversity ↑	Novelty ↑	CDR3 length ↑	AbNatiV ↑	s/seq ↓
4	Branching Flow Maps	**.966**	**.771**	**.799**	**.987**	**.715**	.0149
	BFM Instant	.290	.606	.429	.703	.311	.0111
	Branching Flows	.796	.530	.498	.967	.359	.0029
	Edit Flows	.272	.510	.460	.775	.293	**.0021**
8	Branching Flow Maps	**.984**	**.890**	**.918**	**.992**	**.900**	.0261
	BFM Instant	.559	.749 • Raw output mode on: transcript text is shown for clean terminal selection.	.757	.767	.586	.0190
	Branching Flows	.897	.653	.646	.979	.490	.0059
	Edit Flows	.572	.628	.655	.887	.420	**.0043**
16	Branching Flow Maps	**.988**	**.909**	**.936**	**.992**	**.925**	.0483
	BFM Instant	.832	.875	.915	.872	.868	.0356
	Branching Flows	.962	.753	.769	.988	.663	.0119
	Edit Flows	.793	.740	.783	.951	.599	**.0088**
128	BFM Instant	.982	.920	**.946**	.990	**.936**	.2612
	Branching Flows	**.995**	**.926**	.940	**.995**	.924	.0942
	Edit Flows	.929	.875	.909	.977	.846	**.0720**

We use a conditional Branching Flow Map (BFM) to generate variable-length human heavy-chain variable-domain sequences. Fixed residues provide sequence context, while the model jointly generates the identities and lengths of the editable regions. Training comprises instantaneous-model pretraining and human-specific fine-tuning, followed by finite-map training and a final polishing phase.

F.1 DATA AND CONDITIONING

The human training set contains 15 million sequences over the 20 canonical amino acids.

These sequences were the same filtered subset as used in the Branching Flows paper Billera et al. (2026b). The antibodies were 15 million randomly sampled sequences from the data used by Turnbull et al. (2024), a filtered subset of the Observed Antibody Space database Kovaltsuk et al. (2018). We took an instantaneous model first pretrained on an equal mixture of this human data collection and a separate collection of 15 million camelid VHH sequences, sampling uniformly within each collection. Human-specific fine-tuning, finite-map training, and polishing then sample exclusively from the human collection.

For each training example, we mark the entire sequence editable with probability 0.1. Otherwise, following Branching Flows Billera et al. (2026b), we form an editable mask from the union of randomly placed spans. The number of proposed spans is $1 + \text{Poisson}(6U)$, with $U \sim \mathcal{U}[0, 1]$. Each span has a uniformly sampled starting residue, a uniformly sampled direction, and an extension of $\text{Poisson}(30V)$ residues, where $V \sim \mathcal{U}[0, 1]$ is drawn independently for each span. Both endpoints are included, spans are clipped to the sequence boundaries, and overlaps are merged. The model receives the editable mask and the identities of the fixed residues. Fixed residues retain their identities and relative order throughout generation; their positions shift as the intervening editable regions grow or shrink.

F.2 BRANCHING PROCESS

We use four equal stages with boundaries $\tau_j = j/4$, $j = 0, \ldots, 4$. Each element carries a relaxed amino-acid state $x \in \mathbb{R}^{21}$ and an action state $a \in \mathbb{R}^{11}$. The amino-acid state contains the 20 canon-

ical residue channels and an auxiliary mask channel; endpoint probabilities are normalized over the canonical residues. Actions encode offspring counts $c \in \{0, \dots, 10\}$ as one-hot vectors $e(c)$: zero deletes an element, one retains it, and larger counts create consecutive copies. The final boundary allows counts zero and one. The initial population in each editable region is an independent draw from $5 + \mathrm{Poisson}(5)$, including the single region used for unconditional generation. Initial editable amino-acid and action states are independent standard Gaussians.

Conditional forests. For an editable target span containing N residues and assigned R initial elements, we augment the target with marked duplicate residues. Their count D has a $\mathrm{Poisson}(0.5N)$ law conditioned on $N + D \geq R$ and on the joint population capacity across all spans. Each duplicate is placed immediately before or after a uniformly sampled residue in its span. The augmented sequence is partitioned into ordered descendant groups for the initial elements, and each group is recursively partitioned into contiguous child groups. We sample feasible ordered partitions uniformly. Local offspring counts follow the bounded binomial growth construction with clock

$$G(t) = \frac{6^{t/(3/4)} - 1}{5}, \qquad 0 \leq t \leq 3/4, \tag{21}$$

and $G(t) = 1$ thereafter. At an internal boundary ($j = 0, 1, 2$), a group of $w > 1$ augmented descendants has $1 + \mathrm{Binomial}(w - 1, p_j)$ children, with $p_j = [G(\tau_{j+1}) - G(\tau_j)]/[1 - G(\tau_j)]$, conditioned on the offspring bound and the remaining capacity to reach its descendants. Unmarked singletons persist, while marked singletons are deleted at the next boundary with probability $1/(4 - j)$ in stage j.

Within-stage transport and boundaries. The amino-acid endpoint b is the mean one-hot state of an element's augmented terminal descendants. For an editable element at $t \in [\tau_j, \tau_{j+1}]$, the conditional paths are

$$\begin{aligned} x_t &= x_{\tau_j} + \frac{t - \tau_j}{1 - \tau_j}(b - x_{\tau_j}), \\ a_t &= (1 - u)\eta + ue(c), \qquad u = \frac{t - \tau_j}{\tau_{j+1} - \tau_j}, \qquad \eta \sim \mathcal{N}(0, I). \end{aligned} \tag{22}$$

At an internal boundary, every surviving editable element receives independent Gaussian amino-acid noise, including elements with one child. The standard deviations at the three internal boundaries are 0.5, 0.275, and 0.05. Each editable survivor receives fresh standard-Gaussian action noise for the next stage. Fixed residues remain at their one-hot amino-acid states with count one. The final boundary performs the terminal keep/delete operation.

F.3 Architecture

Instantaneous model. Following the architecture in Branching Flows Billera et al. (2026b) but scaling its depth, the instantaneous endpoint predictor is a packed, bidirectional transformer with 16 blocks, width 768, and 12 attention heads of dimension 64. Its input is the sum of linear projections of the current amino-acid and action states, an embedding of the editable/fixed mask, and an amino-acid embedding at fixed positions. Gaussian random Fourier features of global time, projected to width 768, are combined with a learned stage embedding to condition the transformer blocks.

Each block has an adaptively layer-normalized attention sublayer and an adaptively layer-normalized SwiGLU feed-forward sublayer, each with a residual connection. The adaptive shifts and scales are initialized to zero. Attention uses rotary positional embeddings and RMS-normalized queries and keys. The feed-forward hidden width is $4 \times 768 = 3{,}072$. Sequences are packed together with attention restricted to elements of the same sequence. A learned RMS normalization after the final residual addition precedes separate linear amino-acid and action readouts. Softmax normalization uses the 20 canonical amino-acid classes and the stage's feasible action classes.

Finite-map refinement. The finite model retains the final human instantaneous model's exponential moving average (EMA) as a frozen trunk and adds four trainable transformer blocks. These blocks are initialized from instantaneous blocks 13–16 and start from the features after block 12. Learned lateral connections also provide every refiner block with features from all 16 trunk depths.

Each depth is projected to 1,024 channels, combined using learned depth- and channel-specific mixing weights for each refiner block, and projected back to width 768.

The refinement branch additionally receives a linear projection of the current 32-dimensional joint state and random Fourier features of the normalized within-stage map times $(\bar{s}, \bar{t}, \bar{t} - \bar{s})$. A final RMS normalization precedes residual amino-acid and action readouts. The added state, time, lateral-output, and readout projections are initialized to zero. Residual logit corrections are multiplied by $\bar{t} - \bar{s}$, so the finite predictor agrees with the instantaneous predictor on the diagonal throughout training. The human instantaneous EMA remains the frozen trunk and source teacher during both finite-map training and polishing.

F.4 Training objectives

Instantaneous supervision. We sample the stage uniformly and draw normalized local time from an atom of mass $1/8$ at zero and an otherwise uniform distribution on $[0, 0.9999)$. Amino-acid predictions are trained by cross-entropy to the descendant-average endpoint b. For actions, we use Rao–Blackwellized supervision under the local forest law. Writing $\pi(c \mid \zeta)$ for the feasible count prior given the training-only descendant information ζ, the target is

$$q(c \mid a_u, \zeta) \;\propto\; \pi(c \mid \zeta) \exp\left[-\frac{\|a_u - ue(c)\|^2}{2(1-u)^2} \right]. \tag{23}$$

Action predictions are trained by cross-entropy to this posterior. Each component is averaged over editable elements within an example, then over examples.

Loss scaling and logit regularization. The two cross-entropies have fixed weights stratified by stage and by eight bins of local time for the instantaneous model or map duration for the finite model. The weights are inverse median gradient norms on the shared trainable transformer blocks, with a common rescaling that sets the mean amino-acid weight to one. Instantaneous pretraining reuses previously calibrated weights. Finite-map training uses a disposable 1,000-update pilot followed by 512 calibration batches: four stages, eight bins, and sixteen repeats per cell. Production training restarts from the pre-pilot initialization with these fixed weights. Each readout also receives a z-loss $10^{-4}(\log \sum_{k \in \mathcal{C}} \exp \ell_k)^2$, where ℓ denotes its logits and $\mathcal{C}$ its legal classes. This term is averaged over editable elements and examples and added separately from the calibrated cross-entropies.

Finite-map supervision. For global times $\tau_j \le s < t \le \tau_{j+1}$, the two channels $v \in \{x, a\}$ use the endpoint parameterization

$$\Phi^v_\theta(s,t,y) = y^v + \frac{t-s}{d_v - s}\big(P^v_\theta(s,t,y) - y^v\big), \qquad d_x = 1, \quad d_a = \tau_{j+1}. \tag{24}$$

The first 1,000 production updates use effective endpoints obtained by integrating the frozen instantaneous EMA through 16 equal substeps over the sampled span. Subsequently, every fourth update retains this teacher target and the remaining updates use two half-span maps of the finite model's own EMA. With $m = (s+t)/2$, the composition target is

$$\begin{aligned} Q^v &= \gamma_v P^v_{\bar{\theta}}(s,m,y) + (1-\gamma_v) P^v_{\bar{\theta}}\big(m,t,\Phi_{\bar{\theta}}(s,m,y)\big), \\ \gamma_v &= \frac{(d_v - t)(m-s)}{(d_v - m)(t-s)}. \end{aligned} \tag{25}$$

Targets and intermediate teacher states are detached from differentiation. Both channels use soft-target cross-entropy to these effective endpoints.

Each example independently selects a full-stage span with probability 0.10, a partial span starting at the stage boundary with probability 0.15, or an interior span with probability 0.75. In normalized local coordinates, the interior law is

$$\bar{s} = 0.999U, \qquad \bar{t} = \bar{s} + (1-\bar{s})(0.001 + 0.999V), \qquad U, V \overset{\text{iid}}{\sim} \mathcal{U}[0,1]. \tag{26}$$

Boundary-start spans set $\bar{s} = 0$ and use the same duration law; full-stage spans use $(\bar{s}, \bar{t}) = (0, 1)$. All elements of an example share its map times.

Phase	Training data	Updates
Instantaneous pretraining	Equal human/VHH mixture	240,000
Human-specific adaptation	Human	200,000 + 20,000 warmdown
Shared finite-map training	Human	225,000
Standard final continuation	Human	25,000 warmdown
Polishing continuation	Human	30,000 + 20,000 warmdown

Table 8: Training history of the human antibody model. The standard and polished finite models share their first 225,000 updates, then follow separate continuations, ending at 250,000 and 275,000 updates respectively. The finite model uses the frozen EMA of the 460,000-update human instantaneous model.

F.5 Optimization and training history

Training uses a global batch of 64 sequences over two GPUs, BF16 mixed precision, and FP32 parameters, optimizer states, continuous process states, and loss reductions. Eligible matrix-valued parameters use Muon with momentum 0.95, Nesterov acceleration, and five Newton–Schulz iterations. Amino-acid and action encoders and readouts, embeddings, biases, normalization parameters, and other nonmatrix parameters use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$. AdamW's learning rate is 0.015 times the Muon base rate, and both optimizers use weight decay 0.01. EMA decay is 0.9999. During finite-map training, optimization and EMA updates apply to the trainable refinement branch.

The instantaneous model is pretrained for 240,000 updates on the mixed human/VHH data. Its learning rate rises from 10^{-4} to 10^{-3} over 2,000 updates, then follows an exponential schedule set to reach 2×10^{-4} at update 720,000. Human-specific adaptation begins from the 240,000-update state and runs for 200,000 updates, followed by a 20,000-update linear warmdown to zero. The resulting instantaneous model has 460,000 updates in its cumulative history.

Finite-map training starts a fresh update clock. Its learning rate rises from 10^{-4} to 10^{-3} over 2,000 updates and decays exponentially to 2×10^{-4} at update 225,000. The standard finite model then completes a 25,000-update linear warmdown, ending at update 250,000. Polishing branches from the retained state at update 225,000. The two final continuations are summarized in Table 8.

F.6 Polishing and sampling

Polishing. We continue the 225,000-update finite model for 50,000 updates, retaining its trainable parameters, optimizer state, EMA, and calibrated cross-entropy weights. The training-time distribution places equal probability on full-stage spans, equal half-stage spans, and interior spans from Eq. equation 26. A half-stage span is chosen uniformly from $[0, 1/2]$ and $[1/2, 1]$ in local time.

Teacher-target and EMA-composition updates alternate, giving each half of the polishing updates. For a teacher span of local duration $\Delta = \bar{t} - \bar{s}$, we use

$$n_{\text{teacher}} = \max\{2, \lceil 32\Delta \rceil\} \tag{27}$$

equal instantaneous substeps. Full-stage spans therefore use 32 teacher evaluations and half-stage spans use 16. Composition updates retain the two-half-map EMA target in Eq. equation 25. Both update types draw from the same full/half/interior time distribution.

F.7 Experimental settings, search budget and evaluation

To evaluate the model, we consider a conditional completion task for antibody sequences, where the model fills in missing segments whose lengths are unknown. We consider two variants: (1) uniformly sampling a number of regions between two and six, then uniformly selecting that many of the seven IMGT regions to remove; and (2) randomly removing spans using the conditional masking procedure described in Section F.1. For evaluation, we exclude unconditional examples and redraw masks that remove the entire sequence. All methods receive the same retained sequence context and masks, without the lengths of the removed segments.

Models and inference budgets. For both tasks, we compare BFM and BFM Instant with the conditional Branching Flows Billera et al. (2026b) and Edit Flows Havasi et al. (2025) models described

in the Branching Flows paper. Here, BFM denotes the polished finite-map model, and BFM Instant denotes the final human-adapted instantaneous model. These four models use a hidden dimension of 768, but differ in depth: Branching Flows and Edit Flows use 8 transformer layers, BFM Instant uses 16, and BFM augments the 16-layer backbone with 4 refinement layers. Branching Flows and Edit Flows each contain approximately 95 million parameters, whereas BFM contains 254 million and BFM Instant contains 190 million. We evaluate BFM at 4, 8 and 16 function evaluations, and BFM Instant, Branching Flows and Edit Flows at 4, 8, 16 and 128 function evaluations.

For IMGT-region infilling, we additionally evaluate the VH-only IgCraft model Greenig et al. (2025), using variable-length sampling and 32 particles. We use 4, 8 and 16 steps at temperature 1.0, and additionally include its default step count and temperature of 20 and 0.1. Each IgCraft network call processes 32 particles per input, so these sampling budgets are not directly compute-matched to those of the other models. Each configuration produces one final sequence per input. Generated regions are combined with the original unmasked regions to preserve the supplied context.

Evaluation metrics. We evaluate on 10,000 human VH sequences held out from model training for BFM, Branching Flows and Edit Flows. Overlap with IgCraft's training data is unknown. For IMGT-region infilling, 9,983 sequences pass the numbering and region-capacity checks. We compare generated and natural distributions of sequence length, diversity, novelty, CDR3 length and AbNatiV VH scores. Diversity is the minimum cosine distance between a sequence's amino-acid 3-mer count vector and those of other sequences in the same set. Novelty is the minimum Levenshtein distance to sequences in the opposite reference fold from the input sequence; it therefore measures distance to held-out reference sequences rather than to the training set. CDR3 lengths are obtained using IMGT numbering. For each metric, we report $1 - \mathrm{KS}_D$, where KS_D is the two-sample Kolmogorov–Smirnov statistic. CDR3 and AbNatiV distributions are computed over successfully numbered or scored sequences, respectively.

Timing experiments. Generation times are measured on an NVIDIA GB10 over 100 inputs, taking the median of three warmed repeats. Timings include initialization, sampling and decoding, but exclude model loading, compilation and evaluation. These are batched throughput measurements, with batch sizes selected separately for each model and task. BFM, BFM Instant, Branching Flows and Edit Flows use BF16 network inference, whereas IgCraft retains FP32. All methods use compiled networks.

Results: IMGT-region infilling. Table 1 compares the methods across inference budgets. BFM consistently outperforms Edit Flows across all five distribution metrics at matched evaluation budgets. It also substantially improves on Branching Flows at low budgets, although Branching Flows achieves slightly better CDR3-length matching at 8 and 16 NFE. At matched NFE, however, BFM is consistently slower than both Edit Flows and Branching Flows.

For Branching Flows, the comparison is mixed. At similar wall-clock budgets, BFM at 4 NFE and Branching Flows at 16 NFE take approximately 13 ms and 11 ms per sequence, respectively. BFM better matches AbNatiV scores and achieves nearly identical novelty matching, whereas Branching Flows better matches sequence length, diversity and CDR3 length. At higher budgets, BFM at 16 NFE achieves broadly comparable distribution matching to Branching Flows at 128 NFE while generating approximately $2\times$ faster.

For Edit Flows, the advantage is clearer: BFM at 16 NFE outperforms Edit Flows at 128 NFE across all five metrics, while requiring approximately 44 ms rather than 64 ms per sequence.

IgCraft provides a strong baseline, but at substantially higher measured runtime. Its 8-step configuration achieves better diversity and novelty matching than BFM at 16 evaluations, with nearly identical AbNatiV matching, whereas BFM better matches sequence and CDR3 lengths. This IgCraft configuration takes approximately 1.08 s per sequence, making BFM approximately $25\times$ faster. At 16 steps, IgCraft achieves the strongest AbNatiV matching, but performs worse than its 8-step configuration on the other four metrics and takes approximately 2.17 s per sequence, around $49\times$ the time required by BFM at 16 evaluations. Thus, additional IgCraft steps do not uniformly improve distribution matching. The IgCraft configuration using the default step count and temperature is slower still and is outperformed by BFM at 16 evaluations across all five metrics.

Results: Arbitrary-span infilling. Table 7 evaluates completion of spans that need not follow IMGT boundaries. IgCraft is excluded from this comparison because its released implementation

supports variable-length infilling of predefined IMGT regions, but does not directly support variable-length infilling of arbitrary spans.

However, comparing the existing models, BFM consistently outperforms both Branching Flows and Edit Flows across all five metrics at matched budgets of 4, 8 and 16 evaluations. At 16 evaluations, BFM additionally exceeds Edit Flows at 128 evaluations on every metric, while requiring approximately 48 ms rather than 72 ms per sequence.

The comparison with Branching Flows is more nuanced at larger budgets. BFM at 16 evaluations achieves distribution matching close to Branching Flows at 128 evaluations, while taking approximately 48 ms rather than 94 ms per sequence. Branching Flows nevertheless achieves slightly better length, diversity, novelty and CDR3-length matching at this higher budget, while AbNatiV matching is nearly identical. At similar wall-clock budgets, BFM at 4 NFE also outperforms Branching Flows at 16 NFE on four out of five metrics, taking approximately 15 ms and 12 ms per sequence, respectively. BFM Instant at 128 evaluations also provides strong results, including the highest novelty and AbNatiV matching among the evaluated configurations, but takes approximately 261 ms per sequence. Although BFM is slower per function evaluation than Branching Flows and Edit Flows, its stronger performance at low evaluation budgets can therefore translate into faster generation at comparable distribution matching.

G QM9 REWARD-GUIDED GENERATION WITH FEYNMAN–KAC STEERING AND BEAM SEARCH

We adapt the rollout-based Feynman–Kac (FK) steering and beam-search procedures of Proteína-Complexa (Didi et al., 2026) to Branching Flow Maps (BFMs). Both methods allocate additional inference-time computation to promising partial trajectories, without updating the generative model or differentiating through the reward. Our extension allows a search particle to change its population through the BFM splitting and deletion process. Consequently, the search can jointly explore molecular geometry, atom identities, and atom count, rather than conditioning on a prescribed molecular length.

G.1 PROGRESSIVE ATOM-SPAN REWARD

For a molecule $M = \{(\mathbf{x}_i, h_i)\}_{i=1}^{n}$, we define its atom span by

$$d(M) = \max_{1 \le i,j \le n} \|\mathbf{x}_i - \mathbf{x}_j\|_2. \tag{28}$$

Distances are measured in Å, and all atoms, including explicit hydrogens, contribute. We conduct separate searches for smaller and larger spans, using

$$r_\delta(M) = \delta\big[d(M) - m\big], \qquad \delta \in \{-1, +1\}, \tag{29}$$

where $\delta = -1$ favors smaller spans, $\delta = +1$ favors larger spans, and $m = 6.4111$ Å is the training-set median span. This centering constant does not affect rankings or normalized exponential weights. The reward is strictly progressive: it has neither a target span nor a plateau or clipping threshold.

Let $v(M)$ indicate that the molecule is both chemically valid and connected. We assess validity using the molecular reconstruction and sanitization procedure of ADiT (Joshi et al., 2025), retaining explicit hydrogens, and additionally require the entire reconstructed molecule to contain exactly one connected component. The selection potential is

$$G_{\delta,\beta}(M) = \begin{cases} \exp\{\beta r_\delta(M)\}, & v(M) = 1, \\ 0, & v(M) = 0, \end{cases} \qquad \beta > 0. \tag{30}$$

Here β has units of inverse Å. Among eligible candidates, an improvement of Δ Å in the desired direction multiplies the potential by $\exp(\beta\Delta)$. Invalid or disconnected candidates receive no positive selection weight, irrespective of their span.

G.2 VARIABLE-POPULATION PROPOSALS AND TERMINAL LOOKAHEAD

We use the trained four-interval QM9 BFM with fixed parameters throughout search. A particle (note: not an atom) contains the current coordinates, relaxed atom labels, and decoded offspring

actions. Each search maintains W particles and expands each into B candidate continuations at every interval. Note: These search branches should not be confused with from the element-level splits and deletions performed by the model.

Write $S_{k-1}^{(i)}$ for retained particle i before interval k. A candidate is obtained by applying the usual BFM boundary transition and finite-time maps,

$$\widetilde{S}_k^{(i,b)} = \Phi_k\big(S_{k-1}^{(i)}; \boldsymbol{\epsilon}_k^{(i,b)}\big), \qquad i = 1, \dots, W, \quad b = 1, \dots, B. \tag{31}$$

At the first interval, the candidates instead start from the model's Gaussian one-root prior. Proposals vary the initial noise and the Gaussian noise already present at interval boundaries: coordinate perturbations, relaxed-label perturbations, and initialization of the next action state. The within-interval maps are deterministic given these draws.

Action-state noise changes the subsequent splitting and deletion decisions. Thus candidates descended from the same search particle can develop different atom counts in later intervals. Length is explored through the model's natural sampling behavior. Because the reward depends on absolute span, its optimization may change both atom count and conformation.

To control the correlation between sibling proposals while retaining the Gaussian marginal law, we draw

$$\boldsymbol{\epsilon}_k^{(i,b)} = \rho\, \mathbf{z}_k^{(i)} + \sqrt{1-\rho^2}\, \boldsymbol{\eta}_k^{(i,b)}, \qquad \mathbf{z}_k^{(i)}, \boldsymbol{\eta}_k^{(i,b)} \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, I). \tag{32}$$

This construction is applied separately to each Gaussian noise component. The anchor $\mathbf{z}_k^{(i)}$ is shared by siblings and drawn afresh for each parent and interval, independently of the retained state. Before selection, each proposal has unit noise covariance, while two siblings have cross-covariance $\rho^2 I$. Thus $\rho = 0$ gives independent proposals and larger ρ reduces their dispersion around a shared random anchor.

Partial states are scored by completing them to molecules. For each candidate we draw one unsteered continuation through the remaining intervals,

$$\widehat{M}_k^{(i,b)} = \mathcal{R}_{k\to K}\big(\widetilde{S}_k^{(i,b)}; \boldsymbol{\xi}_k^{(i,b)}\big), \qquad K = 4, \tag{33}$$

using fresh independent boundary noise. We evaluate $r_\delta(\widehat{M}_k^{(i,b)})$ and $v(\widehat{M}_k^{(i,b)})$, then select among the associated intermediate states $\widetilde{S}_k^{(i,b)}$. The lookahead is a single stochastic estimate of a candidate's promise, not an exact conditional expectation. Its future is not committed: a retained state is expanded again at the next interval. At the final interval, the scored molecules themselves are selected. Only these final outputs enter the sampled distribution.

G.3 Soft resampling and hard selection

Feynman–Kac steering. Index the WB candidates within a search by a. If at least one completed lookahead is connected and valid, normalize its potential to obtain

$$p_k(a) = \frac{G_{\delta,\beta}(\widehat{M}_k^{(a)})}{\sum_{c=1}^{WB} G_{\delta,\beta}(\widehat{M}_k^{(c)})}. \tag{34}$$

We draw W indices independently with replacement from this categorical distribution and retain the corresponding intermediate states. Higher rewards increase the probability of propagation, but do not deterministically exclude lower-reward eligible candidates. A state can be selected more than once; its copies receive new proposal noise at the next expansion. If all candidates are ineligible, we propagate W uniformly sampled candidates so that subsequent intervals can recover.

Beam search. The proposal and lookahead procedures are identical, but selection is deterministic. Assign each candidate the score

$$q_k(a) = \begin{cases} r_\delta(\widehat{M}_k^{(a)}), & v(\widehat{M}_k^{(a)}) = 1, \\ -\infty, & v(\widehat{M}_k^{(a)}) = 0, \end{cases} \tag{35}$$

and retain the W highest-scoring candidate indices without replacement. Ties are resolved by candidate order. If fewer than W candidates are eligible, the remaining positions are filled by ineligible candidates in that order.

Both methods repeatedly favor partial trajectories whose sampled completions have high reward. We use FK steering in this rollout-based selection sense: the procedure is not an exact sampler from a density proportional to $p_{\mathrm{BFM}}(M)G_{\delta,\beta}(M)$. In particular, intermediate lookahead potentials are reapplied rather than corrected by telescoping importance-weight ratios. Beam search likewise produces a search-biased distribution rather than samples from a specified normalized target law.

G.4 Experimental settings, search budget, and evaluation

We use $W = B = 4$ and select particles after each of the four intervals. Each interval takes four model evaluations, giving 16 neural function evaluations (NFE) along an individual unsteered trajectory. The model uses one initial element, capacity 64, and coordinate and relaxed-label noise of standard deviation 0.2 at each internal boundary; action noise has unit standard deviation.

The reported distributions use the fixed settings in Table 9, chosen to obtain substantial span shifts while preserving validity and diversity. Beam search uses reward rankings directly and therefore has no inverse temperature parameter.

Table 9: Settings used for the reported atom-span steering distributions. The noise parameter ρ is defined in Eq. equation 32.

Method	Span direction	β (Å^{-1})	ρ
FK steering	Smaller	4	0
FK steering	Larger	2	0
Beam search	Smaller	—	0.35
Beam search	Larger	—	0

Lookahead computation must be included in the search budget. At the four selection rounds, each of the 16 candidates requires respectively 16, 12, 8, and 4 model evaluations to advance and complete its lookahead. Hence one search costs

$$C_{\mathrm{search}} = WB\,(16 + 12 + 8 + 4) = 640 \tag{36}$$

model evaluations counted per candidate, and produces four retained molecules. The amortized sampling cost is therefore 160 NFE and 16 molecular reward evaluations per output, including lookahead.

Each setting produces 4,096 molecules from 1,024 independent searches, with four correlated outputs per search. We compare these with 1,024 unsteered molecules per method and 10,000 held-out QM9 structures. We report span distributions both over all outputs and conditional on connected validity, using shared 0.25 Å histogram bins normalized separately for each population. Uncertainty in mean span is estimated by resampling complete search groups.