SCAN-PARALLEL VARIATIONAL INFERENCE FOR EVENT-DRIVEN LATENT JUMP MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continuous-time latent state-space models are widely used for irregularly observed time series, typically representing uncertainty through a random initial state followed by smooth ordinary differential equation (ODE) dynamics, or Brownian perturbations accumulated by a stochastic differential equation (SDE). While these constructions are natural for smooth or diffusive systems, they do not directly encode a finite set of unobserved shocks whose timing, magnitude, and effect on memory are themselves uncertain. We instead introduce latent jump models (LJMs): the latent state follows closed-form deterministic dynamics between Poisson-driven event times and undergoes a learned instantaneous jump at each event, with state-dependent intensity and mark distributions. This defines a time-homogeneous Markov process whose jumps are intrinsic state transitions rather than numerical integration steps. For training, we derive a tractable event-augmented path-space evidence lower bound (ELBO) and introduce *Scan-Parallel Variational Inference (SPVI)*: an observation-conditioned variational posterior samples the complete latent event configuration before the state recurrence is evaluated using triangular affine scans. This enables fast and scalable training for LJMs in *logarithmic temporal depth*. Across synthetic stochastic systems and real irregular time series, LJMs remain competitive with continuous-time latent baselines in forecast accuracy, often with fewer latent updates than numerical solver evaluations, while training one to two orders of magnitude faster than the sequential baselines.

1 INTRODUCTION

Continuous-time latent state-space models provide a natural language for irregularly observed time series: a latent state evolves between observation times, and an emission model relates that state to the measurements. Latent ODE models are especially appropriate when the dynamics are smooth and nearly deterministic conditional on the initial state (Chen et al., 2018; Rubanova et al., 2019). Latent SDE models add Brownian motion, allowing uncertainty to accumulate continuously and producing rough but continuous latent paths (Li et al., 2020). These choices give expressive models of smooth and diffusive evolution but many systems instead evolve through relatively regular motion interrupted by a finite number of abrupt, unobserved changes: interventions, failures, spikes, flares, price shocks, or transitions between operating conditions. In the latter regime, it may be scientifically and computationally preferable to model their times and marks as latent random variables rather than forcing them to appear as a steep continuous vector field or as the accumulation of many Gaussian perturbations.

We take this event-driven perspective and develop *latent jump models* (LJMs), representing stochasticity through a sequence of latent events governed by a marked point process with Poisson noise. A continuous-time latent state evolves through closed-form decay between random event times and undergoes a learned marked-jump at each event. The observations remain (irregular) time-series measurements; however, the events themselves are latent. Therefore, events act as stochastic drivers of the latent state, whose intensity determines when the state is likely to jump and a random mark determines its character. Both the prior event intensity and mark distribution depend on the current state, so the consequences of one event may change the distribution of the later ones.

This produces a latent piecewise-deterministic Markov process (PDMP) (Davis, 1984), admitting a Chapman–Kolmogorov semigroup. Its trajectories can be simulated exactly, without numerical

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.

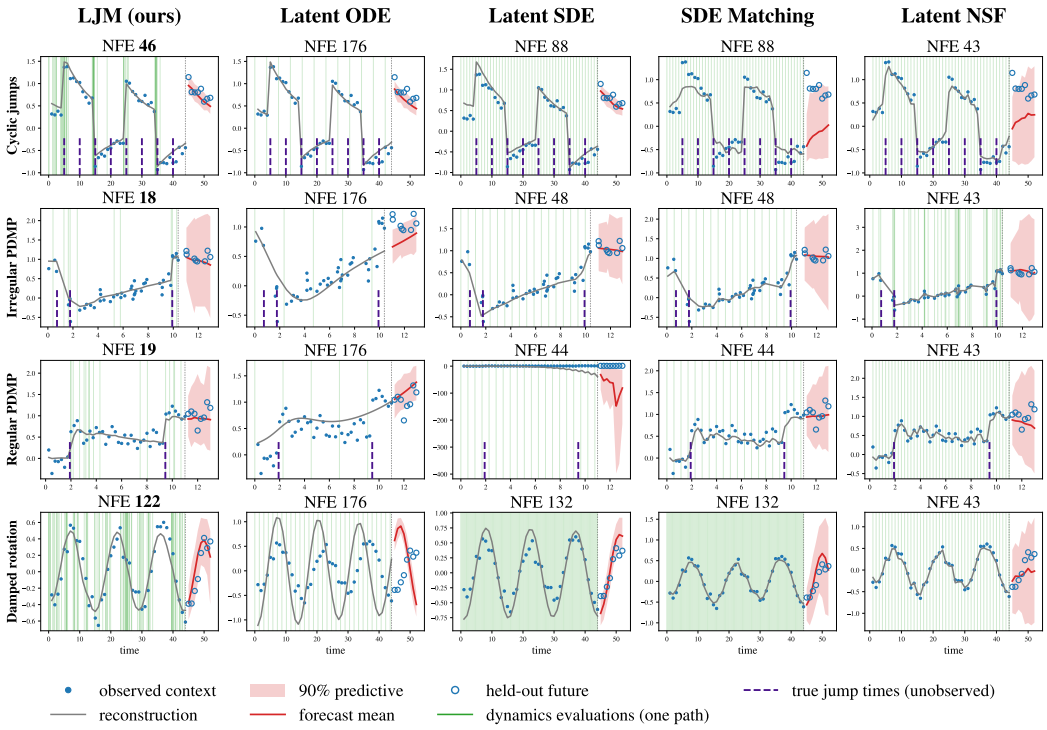

Figure 1: Comparison across four event-driven and smooth systems. All models use identical history-encoder and emission architectures. Solver step sizes are coarsened for baselines at evaluation to approach LJM event budget; NFE labels show the number of dynamics evaluations. LJM shows better reconstruction and forecast quality at fewer NFE. Latent SDE diverges on the regular PDMP, while Latent ODE and Neural Stochastic Flow (solver-free) exhibit errors in several examples.

ODE or SDE solvers, by combining the closed-form flow with jumps induced by the latent event process. This differs from neural point process models in which events themselves are typically the observed data. Therefore, learned events in LJMs may differ from the (typically unidentifiable) events of the data-generating process, however still provide adaptive stochastic updates to the latent representation which are learned rather than enforced by numerical solvers; see Figure 1. Moreover, as we show in Section 4.3, replacing diffusion with jumps does not even require giving up variational learning on the path space. By constructing the variational posterior as a *controlled marked Poisson process*, we derive a path-space evidence lower bound (ELBO) using a change-of-measure formula for marked-point processes (Brémaud, 1981). The posterior shares its deterministic dynamics with prior process, but acts as a control policy on the latent state, adjusting the initial state law, intensity at which events occur and shaping the distribution of event marks in response to observed data.

However, a tractable event likelihood is not enough. Event-driven simulation alone does not remove sequential state unrolling. The situation mirrors Latent ODEs and SDEs, whose training integrates the posterior path with a numerical solver: every step reads the state produced by the previous one, so the depth of a forward–backward pass grows with the number of solver steps, and stiff or long windows are paid for in sequential time (Chen et al., 2018; Li et al., 2020). A latent jump model inherits the same structure through its events. With a random number L of events in a window, a general neural jump map $F_\theta(z, u)$ has to be applied L times in order, because the state entering event n is the state produced by event $n-1$. Sampling every event time and mark in advance does not remove this: the recurrence is in the state, not in the events; this serial depth makes training slow. Recent work removes simulation from continuous-time training in different ways including matching posterior marginals via SDE Matching (Bartosh et al., 2025) and by learning stochastic transition laws for solver-free sampling via Neural Stochastic Flows (Kiyohara et al., 2025). We take a different route: we keep an explicit finite event configuration and a standard path-space objective, and change the transition so that the recurrence itself can be evaluated in parallel.

Hence, we introduce *Scan-Parallel Variational Inference* (SPVI) that enables fast and scalable path-space training of event-driven state space models. Our central construction parallelizes this chain so that its depth scales *logarithmically* in the number of events L without approximating the dynamics. Two design decisions make it possible. First, the variational event process is exogenous: its initial state law, intensities and mark laws are functions of the observations through an encoder network, not of the sampled latent state. Therefore the complete event configuration, i.e., event times and marks, can be drawn before any state is computed. Second, the jump map is constructed to be block-triangular and affine within a block. The state is split into J ordered blocks. Write $z_n^{(j)}$ for block j just after event n and $\tilde{z}_n^{(j)}$ for the same block just before it, i.e., the value after event $n-1$ carried forward by the closed-form decay. Event n then updates block j as

$$z_n^{(j)} = R^{(j)} \odot \tilde{z}_n^{(j)} + B^{(j)},$$

where the retention gates $R^{(j)} \in (0,1)$ and the shifts $B^{(j)}$ are produced by a neural network from three inputs: mark of event n, lower blocks $\tilde{z}_n^{(<j)}$ just before, and lower blocks $z_n^{(<j)}$ just after the event n. The network never reads block j's own value $\tilde{z}_n^{(j)}$, and it reads nothing from any later event.

Together, these decisions make every step of block j an element-wise affine map whose coefficients are known before the state is. The marks are known because events are sampled first and the lower blocks because they are computed first. We emphasize that nonlinearity is relocated rather than removed: the coefficient networks are arbitrary functions of the marks and of the lower blocks, so the transition is nonlinear across blocks. However, the closure property of affine maps enables a parallel prefix scan Kogge & Stone (1973); Ladner & Fischer (1980) which yields the state of block j after every event in $O(\log(1+L))$ dependent stages. A recurrence of length L therefore becomes J parallel scans yielding $O(J\log(1+L))$ depth. SPVI therefore reproduces the chronological trajectory exactly while changing the order of computation from a length-L event recurrence to J *parallel scans, each with logarithmic temporal depth*, enabling fast and scalable training for long time series.

We evaluate our method on six synthetic continuous-time systems spanning smooth rotations, cyclic jumps and piecewise-deterministic processes, and on two real benchmarks, REFIT Murray et al. (2017) motion capture and USHCN Menne et al. (2010) climate datasets. On event-driven systems, LJM matches the best forecast error with a fraction of the baselines' dynamics evaluations while remaining competitive even on smooth rotations. On both real datasets, LJM remains competitive with baselines in terms of forecasting error, at one to two orders of magnitude less training time.

Summary of contributions. We introduce a finite-activity latent jump model with state-dependent event laws, derive its event-augmented variational objective, and develop a triangular transition architecture that enables scan-parallel posterior computation. Experiments characterize the predictive role of latent events and demonstrate shorter training times than the evaluated baselines, with matched-model benchmarks isolating the benefit of scan-based execution.

2 Related work

Continuous-time latent models. Neural ODE (Chen et al., 2018) learns vector fields, and Latent ODE (Rubanova et al., 2019) extends them to probabilistic modeling of irregular observations. Latent SDE (Li et al., 2020) introduces continuously accumulating process noise and support variational learning through changes of drift. NCDSSM (Ansari et al., 2023) combines neural observation models with continuous-discrete filtering and smoothing; continuous recurrent units use linear latent SDEs with closed-form recursive computations (Schirmer et al., 2022).

Simulation-free and solver-free alternatives. Recent methods reduce numerical integration by changing the representation of continuous-time dynamics or their variational posterior. Neural Flows (Biloš et al., 2021) directly parameterize invertible ODE solution maps, with latent trajectories determined by their initial state; the autonomous semigroup identity is an optional regularization target rather than an architectural constraint. SDE Matching (Bartosh et al., 2025) instead retains an SDE prior and a path-space ELBO, parameterizes the posterior through one-time marginals, and constructs a compatible drift so that the objective can be estimated from states sampled at random times without integrating posterior trajectories. Amortized reparametrization uses a related marginal-based construction for latent SDE inference (Course & Nair, 2023). However, Smith et al. (2026) show that,

for diffusions with a common diffusion coefficient, one-time marginals do not uniquely determine the path law: compatible drifts can differ by a field v_t satisfying $\nabla \cdot (q_t v_t) = 0$. Selecting a particular compatible drift can therefore restrict posterior temporal dependence; Helmholtz-SDE (Smith et al., 2026) addresses this restriction by optimizing over marginal-preserving drift corrections. Neural Stochastic Flows (Kiyohara et al., 2025) take a different approach, learning transition kernels for arbitrary time gaps with conditional normalizing flows and regularizing one-step and composed kernels toward Chapman–Kolmogorov consistency. This transition-kernel construction is distinct from the one-time-marginal parameterizations analyzed in Helmholtz-SDE.

SPVI instead specifies a joint variational law over the initial state and *complete latent event configuration*. The shared flow and jump map induce the state path, and the ELBO compares the augmented event laws. Temporal dependence thus follows from the event construction, while associative scans replace chronological posterior state unrolling. Its structural restriction is different: the variational event law depends on observations but not on the sampled latent state.

Jump processes and parallel state computation. PDMPs provide the classical framework for deterministic motion interrupted by random jumps (Davis, 1984). Recurrent switching linear dynamical systems make regime transitions depend on the continuous state (Linderman et al., 2017); LJM instead applies marked transformations to the state while retaining the same between-event flow. Variational inference for Markov jump processes has been developed through mean-field (Opper & Sanguinetti, 2007) and moment-based approximations (Wildner & Koeppl, 2019). LJM uses a continuous state and an observation-conditioned marked-event posterior. Neural Jump SDEs learn hybrid dynamics for observed marked events (Jia & Benson, 2019), whereas Neural Jump ODEs (Herrera et al., 2021) and GRU-ODE-Bayes (De Brouwer et al., 2019) update their predictive states when observations arrive. TriTPP (Shchur et al., 2020) enables parallel sampling and density evaluation of event times; its variational application to latent Markov jump processes combines inferred jump times with exact conditional inference over finite states. NeuralMJP (Seifner & Sanchez, 2023) learns observation-conditioned transition rates for a discrete-state posterior and evolves its probabilities through a master equation.

LJM instead places a variational distribution over unobserved event times and marks. Its per-block recurrence is an event-indexed analogue of diagonal recurrences in S4D (Gu et al., 2022) and LRU (Orvieto et al., 2023), with event-dependent coefficients resembling Mamba's input-dependent transitions (Gu & Dao, 2024). Associative formulations also support Bayesian smoothers (Särkkä & García-Fernández, 2021) and state-space layers (Smith et al., 2023). S2P2 (Chang et al., 2025) models observed marked events using stacked state-space layers with additive event impulses. LJM, however, applies scans to a sampled latent event stream and trains through an event-augmented ELBO. Its triangular dependence resembles affine coupling in RealNVPs (Dinh et al., 2017), but upper-block coefficients depend nonlinearly on realized lower-block states, requiring successive scans rather than one globally linear recurrence.

3 Latent Jump Models

In this section, we formally describe our generative model. Let the observations be $Y = \{(t_k, y_k)\}_{k=1}^{K}$, with $0 = t_0 < t_1 < \cdots < t_K = T$. We introduce a continuous-time Markov latent state $Z_t \in \mathbb{R}^d$ to model these observations. LJMs therefore have three ingredients: a deterministic flow, a random event-jump mechanism, and an observation distribution.

Starting from $Z_0 \sim p_0 = \mathcal{N}(0, I)$, the state decays coordinatewise between events:

$$\dot{Z}_t = -\gamma \odot Z_t, \qquad \Phi_\delta(z) = e^{-\gamma\delta} \odot z, \qquad \gamma \in (0, \infty)^d. \tag{1}$$

Thus any event-free interval can be traversed in one closed-form operation. At time t, the pre-event state $z = Z_{t^-}$ determines the event intensity $\lambda_\theta(z)$ and the distribution of the mark $U \in \mathbb{R}^m$:

$$0 < \lambda_{\min} \le \lambda_\theta(z) \le \bar{\lambda}_\theta < \infty, \qquad p_\theta(u \mid z) = \mathcal{N}(u; \mu_\theta(z), \operatorname{diag} v_\theta(z)). \tag{2}$$

The bounded rate gives finitely many events on every finite horizon. The state dependence makes the process responsive to its own history: an event can move it into a region with a different future rate or mark law. At each event time τ_n, we sample $U_n \sim p_\theta(\cdot \mid Z_{\tau_n^-})$ and together call (τ_n, U_n) an *event*.

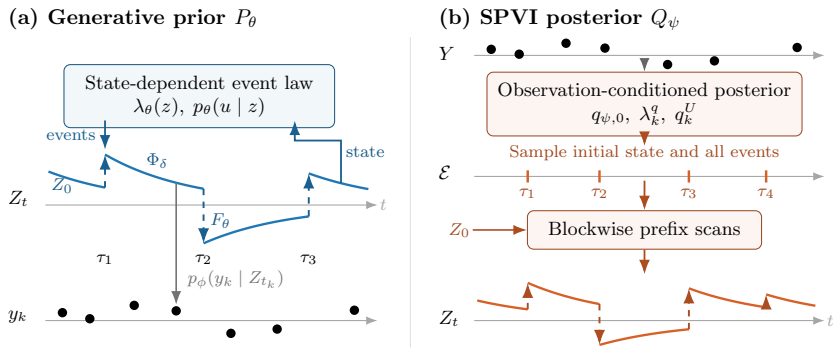

Figure 2: (a) **Generative prior**: the current latent state determines the intensity and mark distribution of subsequent events, requiring chronological simulation. (b) **SPVI posterior**: observations determine the initial-state distribution and event laws independently of the sampled latent state. The complete event configuration can therefore be sampled before constructing the trajectory through successive blockwise scans. Both processes share the closed-form flow Φ_δ and jump map F_θ.

Each event updates state $Z_{\tau_n} = F_\theta(Z_{\tau_n^-}, U_n)$ with the jump map F_θ, where the pre-jump state $Z_{\tau_n^-}$ is obtained via the closed form deterministic flow $Z_{\tau_n^-} = \Phi_{\tau_n - \tau_{n-1}}(Z_{\tau_{n-1}})$.

Although the generative model is valid for arbitrary jump maps, the state computation remains sequential when no structure is imposed on the network. In this work, we replace this sequential computation by parallel "depth" computation via our stacked block-triangular structure of the jump network F_θ which only reads the lower blocks of the sampled trajectory. This makes fast and parallel variational inference possible for LJMs with logarithmic temporal depth. We will discuss this in detail in Section 4.

Finally, the observation or emission model is $y_k \mid Z_{t_k} \sim p_\phi(y_k \mid Z_{t_k})$, evaluated on the observed coordinates. Our experiments assume a parametric form for the observation model, often a Gaussian with its mean parametrized by a neural network. Therefore, a predictive sample contains randomness from the inferred initial state, latent events, and observation model.

With measurable finite-valued updates and the bounded rate in Equation (2), these ingredients define a non-explosive, time-homogeneous Markov process. Appendix A gives the construction of the generator and transition-semigroup argument.

4 Scan-Parallel Variational Inference

In this section, we introduce *Scan-Parallel Variational Inference* (SPVI), a framework that enables fast and scalable path-space training of event-driven state space models. Learning requires a distribution over the unobserved initial state and events. SPVI defines the variational posterior family as a *controlled marked Poisson process* and proceeds in three steps: sample an observation-conditioned event configuration, construct its latent trajectory by scans, and compare that trajectory's event law with the generative prior via a tractable event-augmented path-space ELBO. We describe each of these ingredients one-by-one below.

4.1 Variational family and event configuration generation

In this section, we describe the variational family which makes the observation-conditioned posterior event sampling possible, enabling parallel state computation. It has three components: initial state head, event intensity head, and event mark head. All three heads are conditioned on a common recognition encoder, that uses causal attention over the observation sequence to produce observations embeddings $h_{1:k} \in \mathbb{R}^{K \times d_h}$ in one single forward pass. Event parameters on $I_k = (t_{k-1}, t_k]$ are then computed from h_k, which includes the right-endpoint observation y_k, while the initial-state posterior uses the final context embedding h_K. This defines an observation-conditioned variational posterior over the context trajectory, rather than a strictly online filtering procedure. At evaluation,

the recognition network receives only the context observations; forecasting beyond the context uses the learned generative prior.

For interval I_k of length Δ_k, recognition heads specify a rate $\lambda^q_{\psi,k}$ and Gaussian mark law $q^U_{\psi,k} = \mathcal{N}(\mu^q_{\psi,k}, \operatorname{diag} v^q_{\psi,k})$. This yields $q_{\psi,0}(Z_0 \mid Y)$ and the intervalwise event law

$$N_k \sim \text{Poisson}(\lambda^q_k \Delta_k), \qquad \tau_{k,n} \mid N_k \sim \text{Uniform}(I_k), \qquad U_{k,n} \sim q^U_k. \tag{3}$$

Conditional on Y, the rate and mark parameters do not depend on the sampled latent state. All event times and marks can therefore be generated before state propagation. Denote their sorted collection by $\mathcal{E} = \{(\tau_n, U_n)\}_{n=1}^{N_T}$ and the law of $(Z_0, \mathcal{E})$ by Q_ψ. This event driver therefore supplies parallelism without making the entire transition state independent. The shared jump map still reads the lower blocks of the sampled trajectory. We exploit that distinction next.

4.2 Turn chronological updates into blockwise scans

A fully general F_θ creates a sequential neural recurrence. We now introduce the stacked block-triangular structure of the jump network that retains nonlinear transitions while giving each block a simpler dependence on its own history. Split the state into J ordered blocks, $z = (z^{(1)}, \dots, z^{(J)})$ and let $z^{(<j)} = (z^{(1)}, \dots, z^{(j-1)})$. For the nth event occurring at time τ_n with mark $U_{\tau_n} = u_n$, let the pre-jump state be $Z_{\tau_n^-} = \tilde{z}_n$ and the post-jump state be $Z_{\tau_n} = z_n = F_\theta(\tilde{z}_n, u_n)$. First, by the coordinate-wise decay dynamics between events, we have

$$\tilde{z}^{(j)}_n = \Phi_{\delta_n}(z^{(j)}_{n-1}) = e^{-\gamma^{(j)}\delta_n} \odot z^{(j)}_{n-1}, \quad \text{where } \delta_n = \tau_n - \tau_{n-1}.$$

At jump time, we update blocks from bottom to top:

$$(R^{(j)}_n, B^{(j)}_n) = F^{(j)}_\theta(u_n, \tilde{z}^{(<j)}_n, z^{(<j)}_n),$$
$$z^{(j)} = R^{(j)}_n \odot \tilde{z}^{(j)} + B^{(j)}_n, \qquad R^{(j)}_n \in (\rho_{\min}, 1)^{d_j}.$$

Here $\tilde{z}^{(<j)}_n$ and $z^{(<j)}_n$ are the lower blocks immediately before and after the event (τ_n, u_n). The retention $R^{(j)}$ determines how much memory survives: a retention near one preserves the previous state, while a small retention approximates a reset where the shift $B^{(j)}$ supplies new content.

We emphasize that each $F^{(j)}_\theta$ can be any neural network, and the coefficients R, B depend on lower blocks and the mark arbitrarily, but not on block j itself. Therefore, the upper blocks respond to the realized lower-block trajectory giving a map that is nonlinear and expressive across blocks but affine in each block's own recurrent state. This structure enable parallel state computation while preserving expressivity.

One can further combine the decay over $\delta_n = \tau_n - \tau_{n-1}$ with the event update and obtain the full recurrence as

$$z^{(j)}_n = A^{(j)}_n \odot z^{(j)}_{n-1} + B^{(j)}_n, \qquad A^{(j)}_n = R^{(j)}_n \odot e^{-\gamma^{(j)}\delta_n}. \tag{4}$$

For the first block, every coefficient depends only on its mark and time gap. For the next block, every coefficient becomes available once the first block has been evaluated at all event times. The same argument applies successively to the remaining blocks. To evaluate one block, associate (a, b) with the affine map $x \mapsto a \odot x + b$. Composing two such maps gives

$$(a_2, b_2) \star (a_1, b_1) = (a_2 \odot a_1,\ a_2 \odot b_1 + b_2). \tag{5}$$

Composition is associative, so a prefix scan evaluates every state of the first block. Once that trajectory is available, the second block's coefficients can also be computed in one batch. Repeating this procedure evaluates all J blocks.

Proposition 1. *Conditional on the initial state and event stream, J associative scans under Equation (5) reproduce the chronological flow–jump trajectory. For fixed network architectures and block sizes, the scan stage has $O(JL)$ work and $O(J \log(1 + L))$ temporal depth.*

The proof uses affine composition within blocks and induction across blocks (Appendix A.2). The construction uses only decays across nonnegative time gaps. Observation and quadrature states are then obtained from the preceding stored event state using Equation (1). Scans significantly accelerate posterior state construction for identical models and inputs. We verify this empirically too: at batch-size 16 and the training padding lengths, matched-model scans accelerate posterior forward–backward computation by $125\times$ on cyclic jumps (Appendix F).

4.3 A PATH-SPACE ELBO VIA CHANGE OF MEASURE

The prior and posterior use the same flow and state transformation. Their difference lies in the initial distribution and in when and how events occur. Let P_θ denote the prior law of $(Z_0, \mathcal{E})$. Under the domination and integrability conditions in Appendix A.3, the event-augmented ELBO is

$$\mathcal{L}(\theta, \phi, \psi) = \mathbb{E}_{Q_\psi}\left[\sum_{k=1}^{K} \log p_\phi(y_k \mid Z_{t_k})\right] - \mathrm{KL}(Q_\psi \| P_\theta) \;\le\; \log p_{\theta,\phi}(Y). \tag{6}$$

The marked-point-process likelihood ratio gives a local expression for the divergence:

$$\mathrm{KL}(Q_\psi \| P_\theta) = \mathrm{KL}(q_{\psi,0} \| p_0) + \mathbb{E}_{Q_\psi} \int_0^T \kappa(t, Z_{t^-})\, dt, \tag{7}$$

$$\kappa(t, z) = \lambda^q(t) \log \frac{\lambda^q(t)}{\lambda_\theta(z)} - \lambda^q(t) + \lambda_\theta(z) + \lambda^q(t)\, \mathrm{KL}\big(q_t^U \| p_\theta(\cdot \mid z)\big)\,. \tag{8}$$

The first three terms compare event rates; the final term compares marks, weighted by their rate of occurrence. Both Gaussian KL terms are analytic. We estimate the remaining time integral using Monte Carlo random probes of the scanned trajectory. Thus the generative networks are evaluated at posterior states; full derivation is given in Theorem 1.

4.4 TRAINING WITH HARD EVENTS AND FORECASTING WITH THE PRIOR

To train posterior rates, we construct a homogeneous proposal process whose rate is determined by the maximum posterior intensity over that sequence's observation intervals. The envelope is recomputed from the current posterior rates at each training step. A candidate in interval k is accepted with probability $\lambda_k^q / \bar{\lambda}_q$, where $\bar{\lambda}_q = \mathrm{sg}(\max_k \lambda_k^q) + \rho$, and ρ is a small slack.

$$g_i^{\mathrm{h}} = \mathbf{1}\{\mathrm{logit}(p_k) + \zeta_i > 0\}, \quad g_i^{\mathrm{s}} = \mathrm{sigmoid}\left(\frac{\mathrm{logit}(p_k) + \zeta_i}{\tau_{\mathrm{ST}}}\right), \quad g_i = g_i^{\mathrm{h}} + g_i^{\mathrm{s}} - \mathrm{sg}(g_i^{\mathrm{s}}), \tag{9}$$

where ζ_i is standard logistic noise. In the forward computation, an accepted proposal applies the original jump and a rejected proposal applies only the flow, using effective coefficients $1 + g_i(R_i - 1)$ and $g_i B_i$. Initial states and marks use Gaussian reparameterization. Algorithm 1 in the appendix gives the implemented training procedure.

The experiments optimize a KL-weighted, event-regularized objective,

$$\mathcal{J} = \mathbb{E}_{Q_\psi} \sum_k \log p_\phi(y_k \mid Z_{t_k}) - w_{\mathrm{KL}}\, \mathrm{KL}(Q_\psi \| P_\theta) - \beta_{\mathrm{sparse}} \sum_k \lambda_k^q \Delta_k. \tag{10}$$

The third term controls the activity of the learned latent representation. We evaluate the resulting tradeoff between predictive quality and the number of latent jump updates, without assuming a correspondence between learned events and physical events. $\mathcal{J}$ remains a lower bound on the ELBO in our setting since we use $w_{\mathrm{KL}} \ge 1$ and $\beta_{\mathrm{sparse}} \ge 0$. Training scans all candidates; posterior reconstruction at evaluation instead samples the accepted event process directly through Equation (3).

Forecasting begins from a posterior sample of the last context state Z_T. Beyond T, the model uses the state-dependent prior. We generate proposals at rate $\bar{\lambda}_\theta$, propagate by the closed-form flow, and accept each proposal with probability $\lambda_\theta(Z_{t^-})/\bar{\lambda}_\theta$. Accepted marks are drawn from $p_\theta(\cdot \mid Z_{t^-})$. Repeating this procedure and sampling the emission model produces predictive paths, from which we calculate means and quantile intervals. Appendix B gives the reconstruction and forecasting algorithms.

5 EXPERIMENTS

In this section, we show that LJM achieves better or competitive predictive quality on four synthetic systems and two real datasets while taking fewer dynamics updates than the numerical solver baselines. Latent ODE, Latent SDE, and NCDSSM require sequential solvers during training and inference both; SDE Matching matches marginals during training however inference requires sequential solvers; Neural Stochastic Flow (NSF) is the solver-free baseline, but requires one update per observation interval. Appendix D gives definitions of the evaluated metrics and detailed training protocols.

Table 1: Synthetic forecasting benchmarks. The reported numbers are mean and range over three independent runs of the whole study. NFE is dynamics evaluations per context trajectory; in the columns marked $\times$, a value $K\times$ means LJM is K times *cheaper* than the baseline.

	LJM (ours)			Best baseline at a similar budget				
System	CRPS ↓	RMSE ↓	NFE ↓	Method	CRPS ↓	RMSE ↓	NFE ×	Train time ×
Cyclic jumps	**0.0679** (0.0032)	**0.1261** (0.0036)	89	Latent ODE	0.0729 (0.0010)	0.1276 (0.0015)	2.0×	7×
Irregular PDMP	**0.2244** (0.0012)	**0.5210** (0.0017)	19	Latent SDE	0.2410 (0.0036)	0.5467 (0.0118)	2.6×	10×
Regular PDMP	**0.2509** (0.0009)	0.5995 (0.0022)	19	SDE Matching	0.2567 (0.0009)	**0.5945** (0.0041)	2.3×	1×
Damped rotation	0.1012 (0.0056)	0.1693 (0.0225)	205	NCDSSM-NL	**0.0743** (0.0017)	**0.1274** (0.0027)	1.7×	86×

5.1 Synthetic Systems: smooth rotations, scheduled jumps and PDMPs

The synthetic suite (see Figure 1) spans damped rotations, cyclic schecduled jumps, and irregularly and regularly observed PDMPs. Each evaluation uses 1,000 trajectories, 44 context observations, eight forecast observations, and 128 predictive paths. Table 1 reports LJM at the lowest-CRPS event penalty β_{sparse} (selected on an independent validation set: 0.3 for cyclic jumps, 1 for both PDMPs, 0.1 for damped rotation) against the strongest baseline at the closest budget it can reach. Budget match is approximate because solvers cannot take fewer than one step per observation interval but the baseline's budget is always chosen to be more than LJM's.

Across the synthetic benchmarks, LJM performs the best accross all event driven benchmarks while remaining competitive on rotation with $86\times$ faster training and $1.7\times$ faster inference. Future-event suppression at inference leaves the sparsest cyclic fits nearly unchanged but degrades the PDMP and rotation fits (Appendix E).

5.2 REFIT electrical load measurement data

We evaluate long-context forecasting with $512 - 2,048$ observations for whole-house electrical demand for the 18 households of the REFIT Electrical Load Measurements release (Murray et al., 2017). The raw distribution of per-appliance readings is aggregated into one-minute arithmetic means over $[t - 60\,\text{s}, t)$, and timestamped at the closing edge. The model receives only past aggregate power and elapsed timestamps. Power is scaled per household as $x = (P + 1\,\text{W})/s_h$ with s_h the 0.99 quantile of $P + 1\,\text{W}$ over that household's unique training minutes. Given a context of 512, 1024 or 2048 minutes the task is to forecast power at the every minute for the next 30 minutes. We focus on scalable training baselines (SDE Matching and Latent NSF) for REFIT since sequential solver-based methods are computationally prohibitive to train at long contexts under our experimental budget.

As shown in Table 2, LJM performs best on all three metrics at every context length showing that the advantage is not an artefact of one scoring rule: it holds for a per-minute proper score, for a plain squared error, and for the aggregated energy over the horizon, which is the quantity a demand-response application actually consumes. Against the stronger baseline, SDE Matching, it reduces normalised CRPS by 11–12% and horizon-total energy CRPS by 12%; against the Neural Stochastic Flow the margins are 13–17% and 14–20%. The model remains well-calibrated with sharp prediction intervals for all context lengths. The reason is that aggregate household load is dominated by appliances switching on and off, which is a piecewise-continuous signal with abrupt transitions. Therefore, a model whose latent path can only diffuse must absorb those transitions into its diffusion term, widening its predictive distribution everywhere to cover jumps that occur in a few places. .

Table 7 further varies the sparsity penalty and shows that LJM performs better than both baselines at much fewer dynamics evaluations. Its coverage stays between 0.909 and 0.929 throughout and is closest to nominal at $\beta_{\text{sparse}} = 2$, where the interval width is also narrowest at 735 W, so the sparser models are not achieving their coverage by widening.

Table 2: REFIT aggregate load forecasting using 256 predictive trajectories per test window. Context is the number of observed points. CRPS$/s_h$ normalises by the household scale. Energy CRPS scores total consumption over the 30-minute forecast horizon.

Context	Method	CRPS$/s_h$ ↓	RMSE (W)↓	E-CRPS (kWh)↓	90% cov.	Width (W)
512	LJM (ours)	**0.0495**$_{\pm 0.0004}$	**551.7**$_{\pm 2.9}$	**0.0630**$_{\pm 0.0005}$	0.925$_{\pm 0.005}$	858$_{\pm 33}$
	SDE Matching	0.0544$_{\pm 0.0011}$	566.9$_{\pm 1.2}$	0.0705$_{\pm 0.0007}$	0.876$_{\pm 0.017}$	804$_{\pm 120}$
	Latent NSF†	0.0588$_{\pm 0.0018}$	601.5$_{\pm 21.3}$	0.0779$_{\pm 0.0031}$	0.877$_{\pm 0.012}$	918$_{\pm 322}$
1024	LJM (ours)	**0.0492**$_{\pm 0.0004}$	**551.4**$_{\pm 0.2}$	**0.0628**$_{\pm 0.0005}$	0.926$_{\pm 0.008}$	926$_{\pm 40}$
	SDE Matching	0.0568$_{\pm 0.0006}$	573.7$_{\pm 2.0}$	0.0730$_{\pm 0.0010}$	0.883$_{\pm 0.003}$	937$_{\pm 161}$
	Latent NSF†	0.0599$_{\pm 0.0015}$	589.3$_{\pm 9.2}$	0.0794$_{\pm 0.0030}$	0.872$_{\pm 0.013}$	740$_{\pm 135}$
2048	LJM (ours)	**0.0495**$_{\pm 0.0005}$	**551.5**$_{\pm 1.3}$	**0.0632**$_{\pm 0.0008}$	0.925$_{\pm 0.011}$	892$_{\pm 29}$
	SDE Matching	0.0555$_{\pm 0.0008}$	567.0$_{\pm 1.5}$	0.0709$_{\pm 0.0006}$	0.918$_{\pm 0.008}$	1014$_{\pm 97}$
	Latent NSF†	0.0590$_{\pm 0.0014}$	577.0$_{\pm 2.6}$	0.0765$_{\pm 0.0021}$	0.888$_{\pm 0.015}$	882$_{\pm 100}$

Table 3: USHCN test forecasts: The numbers show mean ± sd over five station folds with 128 predictive paths.

Model	Forecast MSE ↓	CRPS ↓	90% coverage	NFE ↓	Train time/fold ↓
LJM (ours)	<u>0.360</u>$_{\pm 0.069}$	**0.258**$_{\pm 0.024}$	0.837$_{\pm 0.030}$	461	**7.5 min**
NCDSSM-NL	**0.352**$_{\pm 0.055}$	0.305$_{\pm 0.013}$	0.968$_{\pm 0.006}$	6740	3.6 h
Latent SDE	0.414$_{\pm 0.078}$	0.306$_{\pm 0.040}$	0.592$_{\pm 0.127}$	1685	21 h
Latent ODE	0.451$_{\pm 0.107}$	0.371$_{\pm 0.042}$	0.284$_{\pm 0.033}$	6740	11.5 h
SDE Matching	0.433$_{\pm 0.106}$	<u>0.259</u>$_{\pm 0.030}$	0.958$_{\pm 0.018}$	1685	33 min
Latent NSF	0.829$_{\pm 0.162}$	0.477$_{\pm 0.058}$	0.605$_{\pm 0.049}$	**422**	<u>19 min</u>

5.3 USHCN CLIMATE DATA

We evaluate five station folds, each with 110–112 test stations. Context extends through $t = 150$, and the next three observations are predicted; only recorded coordinates are scored. LJM and all baselines use 450 epochs except NCDSSM-NL that uses 200 optimizer steps at batch size 100.

LJM's mean MSE is 0.360, compared with 0.352 for NCDSSM-NL, with training times of 7.5 minutes and 3.6 hours per fold: a roughly 29× difference; see Table 3 and Appendix D.1 for more details.

6 LIMITATIONS AND CONCLUSION

Limitations. Although latent events provide a probabilistic representation of the observed dynamics, but their times and marks need not be uniquely determined by sparse, noisy measurements. Several smaller jumps may explain the observations as well as a few larger jumps, while a flexible decoder introduces additional ambiguity in the latent representation. The expected-count penalty favors parsimonious event configurations but does not establish their identifiability. Our experiments evaluate predictive performance and event activity, rather than recovery of ground-truth events. Second, the variational posterior samples all events in parallel before state computation, hence posterior event times cannot react to past sampled states. Lastly, state-dependent forecasting remains chronological in our framework. Extending our framework for parallelized forecasting remains an open direction for future work.

Conclusion. We introduced latent jump models, a finite-activity latent PDMP with state-dependent event laws, learned memory retention and exact between-event propagation, together with an event-augmented path-space ELBO derived by a change of measure. SPVI replaces the length-L chain of event updates with J successive scans of $O(J \log(1 + L))$ depth while keeping nonlinear interactions across blocks. On six synthetic systems and two real datasets, LJMs matched the forecast error of the strongest evaluated baselines on event-driven tasks with fewer latent jump updates than the baselines requiring numerical solvers, and trained one to two orders of magnitude faster than the sequential ones.

AI USE STATEMENT

In this work, we used generative AI tools for implementation and editing the manuscript. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Abdul Fatir Ansari, Alvin Heng, Andre Lim, and Harold Soh. Neural continuous-discrete state space models for irregularly-sampled time series. In *Proceedings of the International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 926–951, 2023. URL https://proceedings.mlr.press/v202/ansari23a.html.

Grigory Bartosh, Dmitry Vetrov, and Christian A. Naesseth. SDE matching: Scalable and simulation-free training of latent stochastic differential equations. In *Proceedings of the International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 3054–3070, 2025. URL https://proceedings.mlr.press/v267/bartosh25a.html.

Marin Biloš, Johanna Sommer, Syama Sundar Rangapuram, Tim Januschowski, and Stephan Günnemann. Neural flows: Efficient alternative to neural ODEs. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://papers.nips.cc/paper/2021/hash/b21f9f98829dea9a48fd8aaddc1f159d-Abstract.html.

Pierre Brémaud. *Point Processes and Queues: Martingale Dynamics*. Springer, 1981.

Yuxin Chang, Alex James Boyd, Cao Xiao, Taha Kass-Hout, Parminder Bhatia, Padhraic Smyth, and Andrew Warrington. Deep continuous-time state-space models for marked event sequences. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5431.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Kevin Course and Prasanth B. Nair. Amortized reparametrization: Efficient and scalable variational inference for latent SDEs. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2312.10550.

M. H. A. Davis. Piecewise-deterministic Markov processes: A general class of non-diffusion stochastic models. *Journal of the Royal Statistical Society: Series B*, 46(3):353–376, 1984. doi: 10.1111/j.2517-6161.1984.tb01308.x.

Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1905.12374.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1605.08803.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2024. URL https://arxiv.org/abs/2312.00752.

Albert Gu, Ankit Gupta, Karan Goel, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2206.11893.

Calypso Herrera, Florian Krach, and Josef Teichmann. Neural jump ordinary differential equations: Consistent continuous-time prediction and filtering. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2006.04727.

Junteng Jia and Austin R. Benson. Neural jump stochastic differential equations. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1905.10403.

Naoki Kiyohara, Edward Johns, and Yingzhen Li. Neural stochastic flows: Solver-free modelling and inference for SDE solutions. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0571. URL https://arxiv.org/abs/2510.25769.

Peter M. Kogge and Harold S. Stone. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Transactions on Computers*, C-22(8):786–793, 1973. doi: 10.1109/TC.1973.5009159.

Richard E. Ladner and Michael J. Fischer. Parallel prefix computation. *Journal of the ACM*, 27(4): 831–838, 1980. doi: 10.1145/322217.322232.

Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David K. Duvenaud. Scalable gradients and variational inference for stochastic differential equations. In *Proceedings of the Symposium on Advances in Approximate Bayesian Inference*, volume 118 of *Proceedings of Machine Learning Research*, pp. 1–28, 2020. URL https://proceedings.mlr.press/v118/li20a.html.

Scott Linderman, Matthew Johnson, Andrew Miller, Ryan Adams, David Blei, and Liam Paninski. Bayesian learning and inference in recurrent switching linear dynamical systems. In *Proceedings of AISTATS*, volume 54, pp. 914–922, 2017. URL https://proceedings.mlr.press/v54/linderman17a.html.

Matthew J. Menne, Claude N. Williams, Jr., and Russell S. Vose. Long-term daily and monthly climate records from stations across the contiguous united states. U.S. Historical Climatology Network, Carbon Dioxide Information Analysis Center, 2010. URL http://cdiac.ornl.gov/epubs/ndp/ushcn/access.html.

David Murray, Lina Stankovic, and Vladimir Stankovic. An electrical load measurements dataset of united kingdom households from a two-year longitudinal study. *Scientific data*, 4(1):160122, 2017.

Manfred Opper and Guido Sanguinetti. Variational inference for markov jump processes. In *Advances in Neural Information Processing Systems*, volume 20, 2007. URL https://papers.nips.cc/paper/3296-variational-inference-for-markov-jump-processes.

Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *Proceedings of ICML*, volume 202, pp. 26670–26698, 2023. URL https://proceedings.mlr.press/v202/orvieto23a.html.

Yulia Rubanova, Ricky T. Q. Chen, and David K. Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://papers.nips.cc/paper/2019/hash/42a6845a557bef704ad8ac9cb4461d43-Abstract.html.

Simo Särkkä and Ángel F. García-Fernández. Temporal parallelization of Bayesian smoothers. *IEEE Transactions on Automatic Control*, 66(1):299–306, 2021. doi: 10.1109/TAC.2020.2976316. URL https://arxiv.org/abs/1905.13002.

Mona Schirmer, Mazin Eltayeb, Stefan Lessmann, and Maja Rudolph. Modeling irregular time series with continuous recurrent units. In *Proceedings of the International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 19388–19405, 2022. URL https://proceedings.mlr.press/v162/schirmer22a.html.

Patrick Seifner and Ramses J Sanchez. Neural Markov jump processes. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Oleksandr Shchur, Nicholas Gao, Marin Biloš, and Stephan Günnemann. Fast and flexible temporal point processes with triangular maps. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Henry D. Smith, Brian L. Trippe, and Scott W. Linderman. Closing the approximation gap in simulation-free latent SDEs. *arXiv preprint arXiv:2606.16138*, 2026. URL https://arxiv.org/abs/2606.16138.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2208.04933.

Christian Wildner and Heinz Koeppl. Moment-based variational inference for markov jump processes. In *Proceedings of ICML*, volume 97, pp. 6766–6775, 2019. URL https://proceedings.mlr.press/v97/wildner19a.html.

A MATHEMATICAL DETAILS

A.1 FINITE ACTIVITY, EXACT EVENT SIMULATION, AND THE MARKOV PROPERTY

The generative state has no explicit dependence on absolute time. Its flow is $\Phi_\delta(z) = e^{-\gamma\delta} \odot z$, its event intensity is $\lambda_\theta(z)$, and its mark kernel is $p_\theta(du \mid z)$. We make the following assumptions explicit.

Assumption 1 (Generative regularity)**.** *The initial law is a probability measure on $\mathbb{R}^d$; the rate is Borel measurable with $0 < \lambda_{\min} \le \lambda_\theta(z) \le \bar{\lambda}_\theta < \infty$; the mark distribution is a Borel probability kernel; and the jump map is measurable and finite-valued at every finite state and mark.*

Proposition 2 (Finite-activity generative process)**.** *Under Assumption 1, the flow–jump construction defines a non-explosive time-homogeneous Markov process. For a horizon T, its accepted event count satisfies*

$$N_T \preceq_{\mathrm{st}} \mathrm{Poisson}(\bar{\lambda}_\theta T). \tag{11}$$

Its transition operators $P_t f(z) = \mathbb{E}[f(Z_t) \mid Z_0 = z]$ satisfy $P_{s+t} = P_s P_t$.

Proof. Generate a homogeneous Poisson proposal process of rate $\bar{\lambda}_\theta$. At each proposal time, propagate the state with Φ, accept with probability $\lambda_\theta(z)/\bar{\lambda}_\theta$, and, upon acceptance, draw a mark from $p_\theta(\cdot \mid z)$ and apply F_θ. The proposal count is finite almost surely on a finite horizon, and every update produces a finite state. Accepted events are a subset of proposals, giving the count bound. The accepted conditional intensity and mark law equal those specified by the model. Future evolution depends only on the current state and fresh proposal, acceptance, and mark randomness. Since none of these transition rules depends on absolute time, the process is time homogeneous and Markov. Applying conditional expectation at time s gives $P_{s+t} f = P_s(P_t f)$. □

For smooth test functions with locally integrable jump terms, the compensated jump formula gives the extended generator

$$\mathcal{A}_\theta f(z) = \langle -\gamma \odot z, \nabla f(z) \rangle + \lambda_\theta(z) \int [f(F_\theta(z,u)) - f(z)]\, p_\theta(du \mid z). \tag{12}$$

The first term describes event-free motion; the second describes random state transformations. Event-free intervals use the exact flow, and thinning samples the specified event law without a time grid. This is exact simulation of the parameterized process, with ordinary floating-point arithmetic used to evaluate its maps.

A.2 WHY THE TRIANGULAR RECURRENCE ADMITS A SCAN

For three affine maps, both parenthesizations of Equation (5) give

$$(a_3 \odot a_2 \odot a_1,\ a_3 \odot a_2 \odot b_1 + a_3 \odot b_2 + b_3)\,. \tag{13}$$

Thus $\star$ is associative and has identity $(\mathbf{1}, \mathbf{0})$. If (α_i, β_i) is the prefix composition through event i, then $z_i = \alpha_i \odot z_0 + \beta_i$. An implementation that assumes zero initial state can instead replace b_1 with $b_1 + A_1 \odot z_0$.

Proof of Proposition 1. In block 1, the marks, gates, and inter-event gaps determine every coefficient before the recurrence is evaluated. Prefix composition therefore gives exactly the chronological block-1 states. Suppose blocks $1, \dots, j-1$ have been recovered. Their pre-event states are obtained by applying the next decay to the preceding post-event states; their post-event states are already available. Together with the marks, these are precisely the inputs to $F_\theta^{(j)}$. All block-j coefficients can therefore be evaluated in a batch. A second affine prefix calculation gives exactly its chronological states. Induction proves the result for all blocks. Each work-efficient scan has linear work and logarithmic span; there are J sequential block scans. □

This argument also covers rejected candidates: when $g_i = 0$, the event map is the identity and the coefficient contains only the decay. Identity padding uses $(A_i, b_i) = (1, 0)$ and does not alter any prefix. A trajectory with no events is simply $Z_t = \Phi_t(Z_0)$.

To answer a query at time t, find $i = \max\{n : r_n \le t\}$, with $r_0 = 0$ and stored Z_0, and return $\Phi_{t-r_i}(Z_{r_i})$. The implementation uses post-event states for observation queries and pre-event states for KL probes on exact ties. Under continuous event times, a tie with a fixed observation time occurs with probability zero.

A.3 The event-augmented likelihood ratio and ELBO

Write $\lambda^q(t) = \lambda^q_k$ and $q^U_t = q^U_k$ on I_k. Conditional on the observations, the variational event process is a marked inhomogeneous Poisson process. Its initial state is sampled independently of its event configuration under the specified variational family.

Assumption 2 (Domination and finite entropy)**.** *On $[0, T]$, $q_{\psi,0} \ll p_0$; the prior and posterior event processes are non-explosive with positive finite rates; and $q^U_t \ll p_\theta(\cdot \mid Z_{t^-})$ for almost every time and Q_ψ-almost every reachable state. The expected absolute initial, event-rate, and mark log-ratios, both compensator integrals, and the absolute observation log likelihood are finite.*

The positive Gaussian densities and bounded rates used in the implementation provide the domination conditions. The integrability requirement concerns their values along the sampled state process.

Theorem 1 (Event-augmented change of measure)**.** *Under Assumptions 1 and 2, $Q_\psi \ll P_\theta$ on the space of $(Z_0, \mathcal{E})$, and*

$$\log \frac{dQ_\psi}{dP_\theta} = \log \frac{q_{\psi,0}(Z_0 \mid Y)}{p_0(Z_0)} + \sum_{\tau_n \le T} \left[\log \frac{\lambda^q(\tau_n)}{\lambda_\theta(Z_{\tau_n^-})} + \log \frac{q^U_{\tau_n}(U_n)}{p_\theta(U_n \mid Z_{\tau_n^-})} \right] - \int_0^T [\lambda^q(t) - \lambda_\theta(Z_{t^-})]\, dt. \tag{14}$$

Taking expectations yields Equations (7)–(8) and the ELBO in Equation (6).

Proof. Fix the event count and use ordered event times and marks as coordinates. The generative density is the initial density times

$$\exp\left\{ - \int_0^T \lambda_\theta(Z_{t^-}) dt \right\} \prod_{n=1}^{N_T} \lambda_\theta(Z_{\tau_n^-}) p_\theta(U_n \mid Z_{\tau_n^-}). \tag{15}$$

The posterior density has the same form with $q_{\psi,0}$, λ^q, and q^U_t. The shared deterministic maps recover the state from these coordinates. Domination allows division of the two densities; taking logarithms gives Equation (14). Under Q_ψ, the compensation identity gives

$$\mathbb{E}_{Q_\psi} \sum_{\tau_n \le T} a(\tau_n, Z_{\tau_n^-}, U_n) = \mathbb{E}_{Q_\psi} \int_0^T \lambda^q(t) \int a(t, Z_{t^-}, u) q^U_t(du) dt. \tag{16}$$

Applying it to the event log-ratios yields the rate and mark terms in Equation (8). Finally, applying Jensen's inequality to the event-augmented marginal likelihood gives Equation (6). □

The comparison is performed before mapping events into state paths, so no jump-map Jacobian enters the objective. If several event configurations produce the same state path, the augmented KL can exceed the KL between the induced state-path laws; the augmented objective is still a valid lower bound.

For diagonal Gaussians, the mark KL at a state z is

$$\mathrm{KL}(q^U_k \| p_\theta(\cdot \mid z)) = \frac{1}{2} \sum_{a=1}^{m} \left[\log \frac{v_{\theta,a}(z)}{v^q_{k,a}} + \frac{v^q_{k,a} + (\mu^q_{k,a} - \mu_{\theta,a}(z))^2}{v_{\theta,a}(z)} - 1 \right]. \tag{17}$$

For M quadrature probes per interval, draw independent $V_{k,s} \sim \mathrm{Uniform}(0, 1)$ and use

$$\xi_{k,s} = t_{k-1} + \frac{\Delta_k}{M}(s - 1 + V_{k,s}), \qquad \widehat{K}_{\mathrm{dyn}} = \sum_{k=1}^{K} \frac{\Delta_k}{M} \sum_{s=1}^{M} \kappa(\xi_{k,s}, Z_{\xi^-_{k,s}}). \tag{18}$$

Conditional on the sampled path, this stratified estimate is unbiased for the time integral. Increasing M changes the objective estimator, not the sampled path.

B ALGORITHMS AND IMPLEMENTATION

The algorithms below use F_θ for the entire post-event state transformation. An equivalent residual implementation writes $G_\theta(z,u) = F_\theta(z,u) - z$ and updates $z \leftarrow z + G_\theta(z,u)$. The pseudocode describes one trajectory; the implementation vectorizes over trajectories and predictive samples.

B.1 TRAINING

Training uses a posterior proposal envelope, hard-forward events, and a straight-through derivative. For a proposal with acceptance probability p, the forward value of Equation (9) is Bernoulli(p). The derivative with respect to its logit is

$$\frac{\partial g_i}{\partial \operatorname{logit}(p_i)} = \frac{g_i^{\mathrm{s}}(1-g_i^{\mathrm{s}})}{\tau_{\mathrm{ST}}}. \tag{19}$$

This is the surrogate used for event occurrence. Explicit posterior-rate terms in the KL and sparsity penalty are differentiated directly. Reparameterized initial states and Gaussian marks carry ordinary derivatives. Proposed candidates and their times are treated as constant in the backward pass.

Algorithm 1 SPVI training step with hard-forward thinning

Require: observations Y on $[0,T]$, quadrature count M, small slack $\rho > 0$
Encode Y and compute $q_{\psi,0}$ and $(\lambda_k^q, \mu_k^q, v_k^q)_{k=1}^K$
Set the sequence-specific proposal rate $\bar{\lambda}_q \leftarrow \mathrm{sg}(\max_{1\le k\le K} \lambda_k^q) + \rho$
Sample $Z_0 \sim q_{\psi,0}$ by Gaussian reparameterization
Draw $L \sim \mathrm{Poisson}(\bar{\lambda}_q T)$
Draw $r_{1:L} \overset{\mathrm{iid}}{\sim} \mathrm{Uniform}(0,T)$ and sort in increasing order
for all candidates i in parallel **do**
    Find its interval k and set $p_i = \lambda_k^q / \bar{\lambda}_q$
    Draw a mark $U_i = \mu_k^q + \sqrt{v_k^q} \odot \varepsilon_i$, $\varepsilon_i \sim \mathcal{N}(0, I)$
    Draw logistic noise and construct g_i using Equation (9)
end for
Compute the candidate states with BLOCKSCAN(Z_0, r, U, g)
Query states at observation times and stratified quadrature times
Compute the observation log likelihood and the initial-state Gaussian KL
Estimate the dynamic KL with Equation (18)
Form a sample of $\mathcal{J}$ in Equation (10)
Backpropagate its negative, clip the gradient, and update with AdamW

Algorithm 2 Layerwise state construction shared by training and reconstruction

function BLOCKSCAN$(Z_0, r_{1:L}, U_{1:L}, g_{1:L})$
    if $L = 0$ **then**
        return the initial state and the event-free flow
    end if
    Set $r_0 = 0$, $H_0 = Z_0$, and compute $D_i = e^{-\gamma(r_i - r_{i-1})}$
    for $j = 1, \dots, J$ **do**
        Read known lower pre-event states $D_i^{(<j)} \odot H_{i-1}^{(<j)}$
        Read known lower post-event states $H_i^{(<j)}$
        Compute every $(R_i^{(j)}, B_i^{(j)})$ with $F_\theta^{(j)}$ in parallel
        $A_i^{(j)} \leftarrow [\mathbf{1} + g_i(R_i^{(j)} - \mathbf{1})] \odot D_i^{(j)}$, $b_i^{(j)} \leftarrow g_i B_i^{(j)}$
        Set $(A_i^{(j)}, b_i^{(j)}) = (1, 0)$ on padding positions
        Compute every affine prefix under $\star$ and apply it to $Z_0^{(j)}$
    end for
    return all post-event states, with Z_0 prepended
end function

B.2 Posterior reconstruction and generative forecasting

At evaluation, the recognition network is used only on the context. Each predictive path samples its own initial state and context event configuration. The implementation repeats the context encoding across the packed predictive samples. Equation (3) directly generates posterior events without an envelope; all of these events are accepted. Future events use the generative rate and mark law.

Algorithm 3 Context reconstruction followed by state-dependent prior forecasting

Require: context Y through T; future query times $t^f_{1:F}$; prior cap $\bar{\lambda}_\theta$

Encode the context and sample $Z_0 \sim q_{\psi,0}$
for all context intervals k in parallel **do**
  Sample $N_k \sim \mathrm{Poisson}(\lambda^q_k \Delta_k)$, uniform event times, and Gaussian marks
end for
Sort and pack the events; apply BLOCKSCAN$(Z_0, r, U, \mathbf{1})$
Query Z_T and initialize $z \leftarrow Z_T$, $s \leftarrow 0$, $H \leftarrow t^f_F - T$; store $(0, Z_T)$
Draw $M \sim \mathrm{Poisson}(\bar{\lambda}_\theta H)$ and sort $c_{1:M} \overset{\text{iid}}{\sim} \mathrm{Uniform}(0, H)$
for $i = 1, \ldots, M$ in chronological order **do**
  $z \leftarrow \Phi_{c_i - s}(z)$; $s \leftarrow c_i$
  Draw $A_i \sim \mathrm{Bernoulli}(\lambda_\theta(z)/\bar{\lambda}_\theta)$
  if $A_i = 1$ **then**
    Draw $U_i \sim p_\theta(\cdot \mid z)$ and set $z \leftarrow F_\theta(z, U_i)$
  end if
  Store the post-candidate state and time
end for
Query future states from the preceding stored states using the closed-form flow
Decode means and draw observation samples; repeat to form a predictive ensemble

B.3 Recorded synthetic configuration

Table 4 summarizes the synthetic implementation settings. The recognition network is a causal pre-normalized transformer with zero dropout. Posterior event rates use a bounded log-rate link,

$$\lambda^q_k = \exp(a + (b-a)\,\mathrm{sigmoid}(\ell_\psi(h_k)))\,, \qquad a = \log 10^{-3}, \quad b = \log 10. \tag{20}$$

Posterior mark and initial-state variances use an exponentiated log-variance clipped to $[-7, 3]$. The initial-state head reads the final context embedding. Retention gates lie between 10^{-2} and one. The reported total trainable parameter counts are 1,152,436 for LJM, 854,195 for Latent ODE, 1,105,811 for Latent SDE, and 288,659 for the synthetic NCDSSM model.

The training likelihood is a per-trajectory sum, averaged over the batch. There is one Monte Carlo trajectory per training example. The learning-rate implementation combines linear warmup from 0.2 of the peak rate with cosine decay. Candidate streams are padded to a power of two for the scan implementation.

C Synthetic data-generating processes

All six synthetic datasets share a two-dimensional latent state $z \in \mathbb{R}^2$ and a two-dimensional observation $y \in \mathbb{R}^2$. Each trajectory is observed at 52 times, of which the first 44 form the context window and the last 8 the forecast horizon. The observation model is common to all six: writing $z(t_k)$ for the latent state at the k-th observation time,

$$y_k \;=\; z(t_k) \;+\; b \;+\; \sigma\,\varepsilon_k, \qquad \varepsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0, I_2), \tag{21}$$

with an identity emission map, an additive offset $b \in \mathbb{R}^2$ that is zero for every dataset except IRREGULAR PDMP, and isotropic Gaussian noise of scale σ. The models are trained with their decoder scale pinned to this σ, so (21) is known exactly at training time. Every per-trajectory quantity below is drawn independently for each trajectory from a single seeded stream.

Table 4: Synthetic implementation settings. The KL weight takes values in the listed set; the event-regularization weight is swept separately.

Setting	Value
Latent dimension / ordered block sizes	8 / $(4, 4)$
Mark dimension / observation dimension	2 / 2
Training batch size / observations per window	512 / 52
Context / forecast observations	44 / 8
Predictive samples / evaluation batch size	128 / 16
Posterior rate range	$(10^{-3}, 10)$
Prior rate cap	10 / 15
Straight-through temperature	1
Quadrature probes per interval	4
KL weight	$\{2, 5\}$
Posterior sparsity weights	$\{0, 0.03, 0.1, 0.3, 1\}$
Prior-rate penalty	0
AdamW momentum coefficients	$(0.9, 0.99)$
Weight decay / gradient-norm cap	10^{-5} / 1
Whole-window candidate allocation guard	4096

Two of the datasets contain no jumps and serve as negative controls; two contain jumps at deterministic, periodic times; and two are genuine piecewise-deterministic Markov processes with stochastic event times.

C.1 No-jump controls

Both controls evolve by a linear map applied once per unit time step $\delta = 1$, on the regular grid $t_k = k\delta$, $k = 1, \dots, 52$, and both start from the same ring distribution

$$z_0 \;=\; \rho\,(\cos\varphi,\ \sin\varphi)^\top, \qquad \rho \sim \mathcal{U}(0.5,\ 2.0), \qquad \varphi \sim \mathcal{U}(0,\ 2\pi). \tag{22}$$

Write $R(\vartheta) = \left(\begin{smallmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{smallmatrix}\right)$ for the planar rotation.

Deterministic rotation. A noiseless rigid rotation: the latent path is fully determined by (θ, z_0) and all randomness in y comes from (21).

$$z_k \;=\; R(\theta)\, z_{k-1}, \qquad \theta \sim \mathcal{U}(0.25,\ 0.55), \qquad \sigma = 0.10, \quad b = 0. \tag{23}$$

Damped rotation. A contracting rotation *with process noise*, so unlike the deterministic control its latent path is itself stochastic:

$$z_k \;=\; e^{\alpha\delta}\, R(\omega\delta)\, z_{k-1} \;+\; \eta_k, \qquad \eta_k \overset{\text{iid}}{\sim} \mathcal{N}\big(0,\ \varsigma^2 I_2\big), \tag{24}$$

with $\alpha \sim \mathcal{U}(-0.004, -0.002)$, $\omega \sim \mathcal{U}(0.4,\ 0.8)$, process-noise scale $\varsigma = 0.02$, and $\sigma = 0.10$, $b = 0$.

C.2 Deterministic periodic jumps: the two cyclic datasets

Both cyclic variants use the same class and differ only in whether the decay rates are resampled per trajectory. Between observation times the latent decays coordinatewise, and every $m = 5$ steps a mark is added, cycling through four directions with period four. On the grid $t_k = k\delta$, $\delta = 1$, initialised from the ring law (22), the recursion is

$$z_k \;=\; F\, z_{k-1} \;+\; \mathbf{1}[k \bmod m = 0]\; u_{\,n(k) \bmod 4}, \qquad F = \operatorname{diag}\big(e^{a_1\delta},\ e^{a_2\delta}\big), \tag{25}$$

where $n(k) = \lfloor k/m \rfloor$ counts the events so far, the decay acts *before* the jump, and the four marks are

$$u_1 = \begin{pmatrix} c_1 \\ 0 \end{pmatrix}, \quad u_2 = \begin{pmatrix} 0 \\ c_2 \end{pmatrix}, \quad u_3 = \begin{pmatrix} -c_1 \\ 0 \end{pmatrix}, \quad u_0 = \begin{pmatrix} 0 \\ -c_2 \end{pmatrix}, \tag{26}$$

with amplitudes $c_1 \sim \mathcal{U}(0.9,\ 1.5)$ and $c_2 \sim \mathcal{U}(0.6,\ 1.0)$ drawn once per trajectory. With 52 steps and $m = 5$ this gives exactly 10 events per trajectory, at $k \in \{5, 10, \dots, 50\}$, at times known in

advance. Two consequences are worth stating. The amplitudes c_1, c_2 are drawn once per trajectory rather than once per event, so conditional on a trajectory the whole mark sequence is deterministic and the jump channel carries no aleatoric noise. And because events fall exactly on observation times with the decay applied first, no jump is ever hidden between two observations and the recorded latent z_k is the *post*-jump state; there is accordingly no event-localisation difficulty in these two datasets.

Cyclic jumps. The decay rates are drawn per trajectory, $a_1 \sim \mathcal{U}(-0.12, -0.08)$ and $a_2 \sim \mathcal{U}(-0.18, -0.12)$, so the flow is heterogeneous across the dataset. Here $\sigma = 0.10$, $b = 0$.

Cyclic jumps, fixed decay. Identical in every other respect, but the rates are held at $(a_1, a_2) = (-0.10, -0.15)$ for every trajectory. The two variants therefore form a paired control isolating sequence-level decay heterogeneity: the random stream is arranged so that all other sampled quantities are unchanged.

C.3 Stochastic events: the two PDMP datasets

Both PDMP datasets are marked Poisson jump processes with a nonlinear, state-dependent jump map. The event *rate* is homogeneous: events arrive as a Poisson process of constant intensity $\lambda = 0.3$, independent of the state. Between events the latent flows by the exact solution of $\dot{z} = a \odot z$ with $a = (-0.10, -0.15)^\top$,

$$z(t) \;=\; e^{a\,(t-s)} \odot z(s) \qquad \text{for } s \le t \text{ with no event in } (s, t], \tag{27}$$

starting from $z(0) = 0$. At an event time τ a mark is drawn and the state is displaced by a fixed two-layer network,

$$U_\tau \sim \mathcal{N}(0, I_2), \qquad z(\tau) \;=\; z(\tau^-) \;+\; s_J\, W_1 \tanh\big(W_2\, z(\tau^-) + W_3\, U_\tau + \beta\big), \tag{28}$$

with jump scale $s_J = 1$ and weights $W_1 \in \mathbb{R}^{2\times 8}$, $W_2, W_3 \in \mathbb{R}^{8\times 2}$, $\beta \in \mathbb{R}^8$, each entry drawn once as $\mathcal{N}(0, 0.5^2)$ under a fixed seed and then held constant across all trajectories and across both PDMP datasets. The jump is thus state-dependent through $W_2 z(\tau^-)$ while the intensity is not.

Events are simulated interval by interval: on $(t_{k-1}, t_k]$ of length Δ_k the count is $N_k \sim \mathrm{Poisson}(\lambda \Delta_k)$ and the event times are the order statistics of N_k uniform draws on that interval. Both datasets use horizon $H = 13$ with 52 observations and $\sigma = 0.15$.

Because the flow is pure decay from $z(0) = 0$, a trajectory remains identically zero until its first event: with $\lambda H = 3.9$ expected events the latent is exactly $(0, 0)$ for the first twelve to thirteen observations on average, and a fraction $e^{-\lambda H} \approx 2\%$ of trajectories contain no event at all and therefore consist of b plus observation noise throughout. The offset b is added to the observations only; the stored ground-truth latent is not offset, so any latent-space metric is compared against a target displaced from the data by b.

Regular PDMP. A deterministic, equally spaced grid $t_k = kH/52$, so $\Delta_k \equiv 0.25$, with no observation offset, $b = (0, 0)^\top$.

Irregular PDMP. The grid is itself random and differs for every trajectory: 52 points are drawn uniformly on $(0, H]$ and sorted,

$$t_1 < \cdots < t_{52}, \qquad \{t_k\} = \mathrm{sort}\,\big(H \cdot \mathcal{U}(0,1)^{52}\big), \tag{29}$$

resampled if not strictly increasing. The inter-observation gaps are therefore exponential-like rather than constant, which is what makes this the irregular grid. It additionally carries an observation offset $b = (1, 1)^\top$, and it is generated from a different seed stream, so the two PDMP datasets are *not* paired trajectory by trajectory. A comparison between them therefore confounds the irregular grid with a constant shift and with independent sampling noise, and is not a clean grid-only ablation.

Table 5: The six synthetic datasets. All have latent and observation dimension 2 and 52 observations per trajectory, of which 44 are context

Dataset	Event times	Jump / innovation	σ	Grid	Per-traj. randomness
Deterministic rotation	none	none	0.10	regular	θ, z_0
Damped rotation	none	Gaussian, ς=0.02	0.10	regular	α, ω, z_0
Cyclic jumps	every 5 steps	4-cycle, fixed dirs.	0.10	regular	a_1, a_2, c_1, c_2, z_0
Cyclic jumps, fixed decay	every 5 steps	4-cycle, fixed dirs.	0.10	regular	c_1, c_2, z_0
Regular PDMP	Poisson, λ=0.3	$W_1 \tanh(\cdot)$, Eq. (28)	0.15	regular	events, marks
Irregular PDMP	Poisson, λ=0.3	$W_1 \tanh(\cdot)$, Eq. (28)	0.15	*random*	events, marks, grid

D Evaluation definitions and protocols

Let $\mathcal{I}$ contain the held-out scalar coordinates that are actually observed. For predictive latent paths $Z^{(1:S)}$, define the point prediction

$$\widehat{y}_{t,c} = \frac{1}{S}\sum_{s=1}^{S} \mu_\phi(Z_t^{(s)})_c. \tag{30}$$

MSE is $|\mathcal{I}|^{-1}\sum_{(t,c)\in\mathcal{I}}(y_{t,c}-\widehat{y}_{t,c})^2$, and RMSE is its square root. For observation samples $\widetilde{y}^{(s)}$, the interval endpoints are the empirical 0.05 and 0.95 quantiles at each coordinate. Coverage and mean width are

$$\widehat{C}_{.90} = \frac{1}{|\mathcal{I}|}\sum_{(t,c)\in\mathcal{I}} \mathbf{1}\{q_{.05,t,c} \le y_{t,c} \le q_{.95,t,c}\}, \qquad \widehat{W}_{.90} = \frac{1}{|\mathcal{I}|}\sum_{(t,c)\in\mathcal{I}} (q_{.95,t,c} - q_{.05,t,c}). \tag{31}$$

Joint and marginal predictive density. For a single trajectory with D_f observed forecast scalars, the joint-path estimate is

$$\widehat{\mathrm{LPD}}_{\mathrm{joint}} = \frac{1}{D_f}\left[\mathrm{logsumexp}_{s=1}^{S}\left\{\sum_{(t,c)\in\mathcal{I}} \log p_\phi(y_{t,c} \mid Z_t^{(s)})\right\} - \log S\right]. \tag{32}$$

The marginal estimate instead averages separate scalar mixture log densities,

$$\widehat{\mathrm{LPD}}_{\mathrm{marginal}} = \frac{1}{D_f}\sum_{(t,c)\in\mathcal{I}}\left[\mathrm{logsumexp}_{s=1}^{S} \log p_\phi(y_{t,c} \mid Z_t^{(s)}) - \log S\right]. \tag{33}$$

Synthetic sequences have the same forecast dimensions, so the reported per-scalar joint score can be averaged across trajectories. These estimates assess different features of a distribution: the joint version preserves a shared latent path across future times, whereas the marginal version evaluates each scalar separately. Both approximate mixture densities with a finite sample ensemble.

For observation samples $\widetilde{y}^{(1:S)}$, the continuous ranked probability score at one scalar coordinate is estimated from the sample predictive distribution by the fair (unbiased) estimator

$$\widehat{\mathrm{CRPS}}(\widetilde{y}^{(1:S)}, y) = \frac{1}{S}\sum_{s=1}^{S}\left|\widetilde{y}^{(s)} - y\right| - \frac{1}{2S(S-1)}\sum_{s=1}^{S}\sum_{s'=1}^{S}\left|\widetilde{y}^{(s)} - \widetilde{y}^{(s')}\right|, \tag{34}$$

which is computed in $O(S\log S)$ from the order statistics $\widetilde{y}^{(1)} \le \cdots \le \widetilde{y}^{(S)}$ as $\sum_s (2s - S - 1)\,\widetilde{y}^{(s)}/\big(S(S-1)\big)$ for the second term. The reported CRPS is the mean over the held-out coordinates,

$$\mathrm{CRPS} = \frac{1}{|\mathcal{I}|}\sum_{(t,c)\in\mathcal{I}} \widehat{\mathrm{CRPS}}\big(\widetilde{y}_{t,c}^{(1:S)},\, y_{t,c}\big). \tag{35}$$

This is a marginal score: it evaluates each time and coordinate separately and is blind to dependence across the horizon, so a model that gets every marginal right while sampling incoherent paths is not

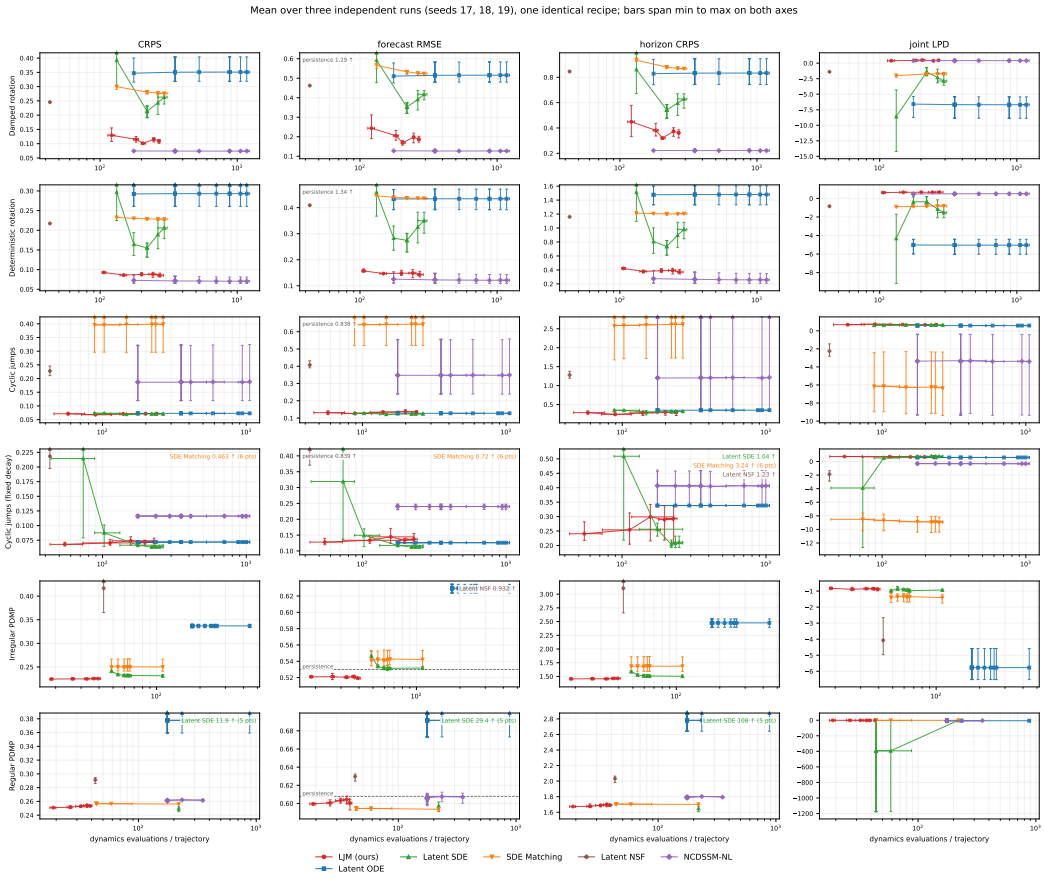

Figure 3: Forecast quality against latent update cost on the six synthetic systems. Rows are systems, columns are CRPS, forecast RMSE, horizon CRPS, joint LPD of the central 90% predictive interval, and joint log predictive density per scalar. The horizontal axis is the number of dynamics-network evaluations per trajectory over the context window, on a logarithmic scale. LJM is shown at each of $\beta_{\text{sparse}} \in \{0, 0.03, 0.1, 0.3, 1\}$; points of one family are joined in order of cost.

penalised. Horizon CRPS scores the aggregate instead. Summing over the observed times of each held-out window before scoring,

$$Y_c = \sum_{t:(t,c)\in\mathcal{I}} y_{t,c}, \qquad \widetilde{Y}_c^{(s)} = \sum_{t:(t,c)\in\mathcal{I}} \widetilde{y}_{t,c}^{(s)}, \qquad \mathrm{CRPS}_\Sigma = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \widehat{\mathrm{CRPS}}\big(\widetilde{Y}_c^{(1:S)}, Y_c\big), \tag{36}$$

where $\mathcal{C}$ indexes the window–coordinate pairs with at least one observation. Because the sum is taken within a path before the score is applied, CRPS_Σ depends on the joint law of the predicted trajectory and not only on its marginals. On REFIT the observable is power, so this quantity is reported in energy units as $E = \sum_t P_t/(60 \times 1000)$ kWh and named energy CRPS.

D.1 Real-data architectures and training protocols

On REFIT every family is built around the same components, so the comparison varies the latent dynamics and holds everything else fixed. All three share one history encoder and one decoder, and their totals land within 1.5% of each other at roughly 1.9M parameters — what differs is only how each represents the latent path between observations. LJM spends its budget on the jump map, the prior mark and intensity networks and the causal recognition heads; SDE Matching on a marginal posterior, a prior drift and a diffusion network; the Neural Stochastic Flow on a learned transition law, a GRU posterior and a bridge. Aggregate household power is positive and heavy-tailed, so every family emits through a log-normal decoder on the household-scaled series $x = (P + 1\,\mathrm{W})/s_h$, with

s_h the 0.99 quantile over that household's training minutes. LJM uses a latent dimension of 32 in four blocks of eight with eight-dimensional Gaussian marks. The USHCN models are smaller, as that dataset is: LJM there has $17{,}596$ parameters at latent dimension 8 in two blocks of four with two-dimensional marks.

Training is matched within each dataset and differs between them only because the datasets do. On REFIT every family takes $30{,}000$ AdamW steps at batch 32 with weight decay 10^{-5}, gradient clipping at 1.0 and a warmup-then-cosine schedule; the peak learning rate is 6×10^{-4} for LJM and 2×10^{-4} for both baselines, and the KL weight is 1.0 for LJM and SDE Matching and 0.1 for the flow, warmed in over the first $2{,}000$ steps. The flow additionally trains on random contiguous sub-sequences of length 50. On USHCN every family runs 450 epochs at batch 32 with the same optimiser, weight decay and clipping, a peak learning rate of 4×10^{-4} decayed to 2.5×10^{-5}, a warmup of eight epochs and a KL weight of 5.0; the decoder mean is learned with the observation scale held fixed.

E How sparsity changes the role of future events

E.1 Task structure and intervention

The cyclic-jump generator uses a fixed event schedule shared across trajectories: a jump occurs every $m = 5$ grid steps and the mark sequence repeats $(+c_1, +c_2, -c_1, -c_2)$. Trajectory-level randomness in the base task comes from the two decay rates, magnitudes c_1, c_2, initial state, and observation noise. The fixed-decay variant retains the scheduled event times and repeating mark-sign sequence. In contrast, event timing is random in the PDMP tasks. The two rotation tasks evaluate smooth oscillatory trajectories.

For each fitted synthetic configuration, we compare the learned prior rollout (Full) with a rollout whose prior intensity is held at its positive floor (Floor). Posterior context inference and the fitted model parameters remain unchanged. The five tested values are $\beta_{\text{sparse}} \in \{0, 0.03, 0.1, 0.3, 1\}$. Each condition uses 1,000 validation trajectories and 128 predictive paths. Floor is a near-zero-event intervention, not a zero-rate process; over the cyclic eight-unit forecast window, its expected count is 0.008 events per path. Table 6 reports the complete sweep.

E.2 Three observed regimes

Scheduled cyclic jumps. With the learned event mechanism, forecast RMSE varies little across the penalty sweep: 0.121–0.127 for cyclic jumps and 0.116–0.130 with fixed decay. Dependence on future events nevertheless changes substantially. Without the event-count penalty, the two fits generate 50.6 and 52.0 future events per path, and suppression raises RMSE to 1.312 and 1.499. At the intermediate penalties, suppressing events remains damaging. Only the largest tested penalty produces forecasts that are nearly unchanged by suppression: RMSE is $0.121/0.120$ for Full/Floor on cyclic jumps and $0.116/0.116$ with fixed decay, at approximately 0.4 future events per path under Full.

The path diagnostics agree with this distinction. For the sparse cyclic fit, the frequency of a largest one-step output increment above 1.0 is 0.84 under Full and 0.88 under Floor, compared with 0.88 in the reference trajectories. For the fit with no event-count penalty, the frequency drops from 0.79 to 0.37 under suppression, and per-step coverage reaches zero from the reported step 6 onward. These results are consistent with the sparse fit representing the predictable cycle through the inferred context state, closed-form flow, and decoder, whereas the other fitted configurations rely on future events. The intervention tests that reliance without retraining.

Randomly timed PDMPs. Increasing the penalty from zero to one reduces the mean future count from 8.7 to 4.2 on the irregular PDMP and from 9.1 to 4.3 on the regular PDMP. Full-rollout RMSE remains similar: 0.519 versus 0.520 and 0.605 versus 0.605, respectively. Events continue to matter throughout the sweep. Their suppression reduces coverage by approximately 13–15 percentage points at every tested penalty, while RMSE increases to 0.572–0.586 on irregular PDMP and 0.661–0.668 on regular PDMP.

Table 6: Future-event suppression across all six synthetic systems and five event-count penalties. Full uses the learned prior intensity; Floor holds it at its positive minimum during forecasting. Context inference and model parameters are unchanged. Each condition uses 1,000 validation trajectories and 128 predictive paths. The last column is the mean accepted future event count under Full. Coverage is rounded to two decimal places.

	RMSE		90% coverage		Future events/path
β_{sparse}	Full	Floor	Full	Floor	Full
Cyclic jumps					
0	0.124	1.312	0.88	0.37	50.6
0.03	0.126	1.225	0.88	0.38	35.2
0.1	0.126	0.808	0.88	0.39	22.7
0.3	0.127	0.473	0.88	0.56	14.7
1	0.121	0.120	0.86	0.86	0.4
Cyclic jumps, fixed decay					
0	0.121	1.499	0.88	0.34	52.0
0.03	0.123	1.328	0.88	0.37	34.3
0.1	0.125	1.176	0.89	0.37	21.4
0.3	0.130	0.576	0.89	0.31	15.2
1	0.116	0.116	0.87	0.87	0.4
Damped rotation					
0	0.175	0.910	0.96	0.38	47.3
0.03	0.213	0.841	0.96	0.41	40.8
0.1	0.314	0.879	0.93	0.39	36.4
0.3	0.350	0.880	0.94	0.37	27.5
1	0.274	0.778	0.95	0.47	17.3
Deterministic rotation					
0	0.147	0.712	0.97	0.46	44.3
0.03	0.147	0.779	0.97	0.44	43.6
0.1	0.190	0.822	0.96	0.40	35.4
0.3	0.163	0.784	0.97	0.50	26.1
1	0.184	0.613	0.97	0.54	15.4
Irregular PDMP					
0	0.519	0.579	0.85	0.70	8.7
0.03	0.521	0.581	0.85	0.70	8.1
0.1	0.518	0.573	0.85	0.71	7.4
0.3	0.535	0.586	0.84	0.70	5.5
1	0.520	0.572	0.86	0.72	4.2
Regular PDMP					
0	0.605	0.663	0.83	0.69	9.1
0.03	0.607	0.661	0.83	0.69	8.5
0.1	0.605	0.668	0.83	0.69	7.3
0.3	0.599	0.661	0.84	0.70	6.0
1	0.605	0.664	0.85	0.70	4.3

Table 7: Event-count penalty sweep for the latent jump model on REFIT, 18 households, context 512, test split.

β_{sparse}	$N_{\text{ev}} \downarrow$	CRPS/$s_h \downarrow$	RMSE (W)↓	Cov.→0.90	Width (W)	Energy CRPS↓
0	205.4	**0.0491**	550.3	0.923	830	**0.0627**
0.01	200.6	0.0496	**549.8**	0.914	858	0.0633
0.03	176.8	0.0508	563.8	0.929	864	0.0664
0.1	185.0	0.0507	559.8	0.928	876	0.0649
0.3	147.5	0.0517	563.5	0.919	855	0.0668
1	98.4	0.0516	565.4	0.926	920	0.0666
2	64.2	0.0526	568.6	**0.909**	**735**	0.0690
4	46.6	0.0540	573.9	0.922	920	0.0722

The distributional diagnostics show why point error alone understates this dependence. Across the reported PDMP diagnostics, the frequency of a largest one-step output increment exceeding 1.0 is approximately 0.23–0.26 in the reference trajectories, 0.12–0.18 under Full, and at most 0.02 under Floor. In the irregular-PDMP fit at $\beta_{\text{sparse}} = 0$, suppression reduces mean interval width from 1.139 to 0.573 and joint-path LPD from -0.90 to -4.70. Interval width grows with horizon from approximately 0.76 to 1.44 under Full, but changes from 0.60 to 0.55 under Floor. Future events supply much of the forecast tail and its horizon-dependent spread. The reported regular-PDMP tail frequency increases from 0.128 to 0.176 from the zero to the largest penalty, so fewer events need not imply a weaker tail. Even the full models still underproduce the largest changes relative to the reference trajectories.

Smooth rotations. Future-event suppression is damaging at every tested penalty, and the penalty also affects the full-rollout accuracy. On damped rotation, RMSE increases from 0.175 at 47.3 future events to 0.274 at 17.3 events for the largest penalty, reaching 0.350 at $\beta_{\text{sparse}} = 0.3$. The deterministic-rotation increase is smaller, from 0.147 to 0.184 between the endpoint penalties. Under suppression, their RMSEs are 0.778 and 0.613, respectively, at the largest penalty. In these fitted models, events therefore support the learned oscillatory forecasts even though the target trajectories are smooth. This contrasts with the sparse cyclic fits, which remain accurate with nearly all future events suppressed.

E.3 Additional REFIT results

We vary the sparsity penalty of LJM on context length 512 and show the results in Table 7.

Interpretation. Table 7 varies the event penalty at context 512, and the comparison that matters is against Table 2. Raising β_{sparse} from 0 to 4 cuts the mean accepted event count from 205.4 to 46.6, a factor of 4.4, and costs 10% in normalised CRPS ($0.0491 \rightarrow 0.0540$). That degradation is small relative to the gap to the baselines. At $\beta_{\text{sparse}} = 2$ the model uses 64.2 events and scores 0.0526, still ahead of SDE Matching's 0.0544 ± 0.0011 by more than its seed spread and well ahead of the Neural Stochastic Flow's 0.0588 ± 0.0018; only at $\beta_{\text{sparse}} = 4$ does it fall level with SDE Matching, and it beats the flow at every penalty in the sweep. The event budget is therefore not what buys the margin: three quarters of the events can be removed before the advantage over the stronger baseline disappears, and the advantage over the weaker one survives the whole sweep.

Coverage stays between 0.909 and 0.929 throughout and is closest to nominal at $\beta_{\text{sparse}} = 2$, where the interval width is also narrowest at 735 W, so the sparser models are not achieving their coverage by widening.

F Scan versus chronological state construction

F.1 Benchmark setup

We evaluate the same fitted models with identical inputs using two execution strategies: the chronological flow–gated-jump loop from the test suite and the vendored pure-PyTorch associative scan

Table 8: Matched-model posterior state construction on one NVIDIA A40 (float32; median times). Seq. is the chronological flow–gated-jump loop; Scan is the pure-PyTorch scan used in training, without a fused kernel. B is batch size and L is the processed proposal length. † marks training padding lengths. Speed-up refers to the forward–backward pass and are computed from unrounded timing measurements; timing entries are rounded. Peak-memory units are shown in each entry.

			Forward (ms)		Forward + backward (ms)			Peak memory	
Model	B	L	Seq.	Scan	Seq.	Scan	Speed-up	Seq.	Scan
Cyclic	16	$1024^{\dagger}$	985	5.0	3,025	24	$125\times$	155 MB	174 MB
Cyclic	128	1024	1,019	10.9	3,053	32	$96\times$	1.20 GB	1.30 GB
Cyclic	128	4096	4,120	41.8	12,506	93	$134\times$	4.81 GB	5.20 GB

used during training. The scan implementation uses no fused kernel. Timings are medians measured in float32 on one NVIDIA A40. We vary the batch size B and processed proposal length L; the latter includes padding at the training lengths and is distinct from the accepted event count. Table 8 combines forward-only time, forward–backward time, and peak memory for all eight settings. This experiment measures posterior state construction and its backward pass, separately from the encoder, full variational objective, and optimizer.

F.2 Runtime scaling

The larger-batch cyclic results expose the scan's increasing arithmetic work: at $B = 128$, increasing L from 1024 to 4096 increases forward–backward time from 32 to 93 ms, compared with $3{,}053$ to $12{,}506$ ms for the chronological loop. The approximately flat timings in smaller cases describe the measured hardware regime; the state construction still has $O(JL)$ work and $O(J \log(1 + L))$ temporal depth for fixed block sizes and coefficient-network architectures.