GALERKINFLOW: TIME-SERIES GENERATION WITH SPLINE COEFFICIENT FLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many time-series generative models produce every time step of a window directly, so the number of values they model grows with the window length, even for smooth signals that far fewer numbers could describe. Functional generative models are formulated on function spaces, but at sampling time they still update the function value at every grid point in every step, so their cost grows with the grid resolution. We introduce *GalerkinFlow*, which instead runs every sampling step on a fixed set of spline coefficients and maps them to the grid only once, through a linear decoder. Ridge fitting projects each window onto a cubic B-spline basis, and a Transformer learns the joint $K \times D$ coefficient distribution with standard flow matching. Because the decoder is linear and preserves distances in a suitable metric, the Wasserstein error of the generated trajectories splits exactly into a floor set by the basis and a term that training can reduce; a further bound accounts for fitting coefficients from finite, possibly incomplete, observations. GalerkinFlow achieves the lowest Context-FID on all six regular benchmarks, outperforms Spectral Mean Flow on irregular-to-irregular generation at every tested nonzero drop rate, and trains 2.5–4.6$\times$ faster than Spectral Mean Flow on the regular benchmarks.

1 INTRODUCTION

Time-series generation aims to learn the distribution of observed sequences and synthesize new samples that preserve their temporal dependencies and relationships across variables. Existing approaches include generative adversarial networks, variational autoencoders, and diffusion models. For instance, TimeGAN combines adversarial training with supervision on temporal dynamics in a learned embedding space (Yoon et al., 2019), while TimeVAE learns a latent representation from which new sequences are decoded (Desai et al., 2021). Diffusion-based methods generate sequences through iterative denoising; Diffusion-TS incorporates a decomposition of temporal structure into this process (Yuan & Qiao, 2024). However, models that generate every time step directly must model more values as the window grows, even when the signal is smooth enough to be described by far fewer numbers. Functional models take a different view and treat each trajectory as a single function to be generated (Kerrigan et al., 2024), but at sampling time they still update the value at every grid point in every step, so each step becomes more expensive as the grid gets finer.

Functional diffusion models extend diffusion to Hilbert spaces using Gaussian measures (Kerrigan et al., 2023) or Gaussian-process noise (Lim et al., 2025), and Functional Flow Matching learns a velocity field that moves a Gaussian reference distribution toward the distribution of observed functions (Kerrigan et al., 2024). These methods rest on a principled function-space theory, but they are implemented with neural operators that act on the function values at every grid point. Many time series, however, are smooth enough to be described accurately by a fixed set of basis functions. This suggests a simpler route: learn the distribution of the basis coefficients, and let a fixed decoder produce the values on the grid.

We propose *GalerkinFlow*, which follows this route. It fits a cubic B-spline curve to each input window, so that each window is summarized by a $K \times D$ matrix of spline coefficients, and trains a Transformer with standard flow matching to generate new coefficient matrices. A fixed linear decoder then turns generated coefficients into values on the sampling grid. The size of the generated object is thus set by the number of basis functions K, not by the window length N; on long win-

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.

dows, where $K \ll N$, the network handles far fewer tokens than a model of the raw values. In exchange, sample quality also depends on how well splines represent the data and how accurately the coefficients can be estimated from the observed samples.

Our analysis makes this trade-off precise. It splits the error of the generated distribution into three parts: the error of representing signals with splines, the error of estimating coefficients from finite samples, and the error of learning the coefficient distribution (Theorem 4.1 and Proposition 4.2). The first two parts are fixed before training begins; only the third depends on training.

Our main contributions are as follows:

- **Spline-Coefficient Generation.** We propose GalerkinFlow, which moves distribution learning from sampled values to cubic B-spline coefficients. A Transformer learns their joint distribution through standard flow matching, and a fixed linear decoder maps generated coefficients to the sampling grid. On long-horizon benchmarks where $K \ll N$, the network processes far fewer tokens than a model of the sampled values, which lowers the cost of every velocity evaluation.
- **Theoretical Analysis.** Because the decoder is linear, we show that the Wasserstein error of the generated trajectories splits exactly into a spline approximation error and a coefficient-learning error, and we bound the extra error of estimating coefficients from finite samples. This separates the roles of basis choice, coefficient estimation, and distribution learning.
- **Empirical Evaluation and Controlled Analysis.** GalerkinFlow achieves state-of-the-art Context-FID on all six regular benchmarks against both non-functional and functional baselines, with competitive results on the other metrics and on long-horizon and masked time series. Controlled studies examine how the basis, fitting accuracy, training, and sampling each affect generation quality.

2 Related Work

Time-Series Generation. Time-series generation has been approached with adversarial, variational, discrete-latent, and diffusion-based models (Yoon et al., 2019; Desai et al., 2021; Lee et al., 2023; Yuan & Qiao, 2024). These methods either model the sampled values directly, so their size grows with the window length N, or learn a latent representation whose approximation error is not quantified. Spectral Mean Flow (Kim et al., 2025) is the closest prior work: it embeds the distribution of whole sequences in a Hilbert space through an operator view of hidden Markov models and generates with a flow-matching form of an MMD gradient flow. GalerkinFlow is simpler. It represents each window with a fixed cubic B-spline basis that is local and smooth, learns the coefficient distribution with standard flow matching, and admits an exact error decomposition. On the six regular benchmarks, it also trains 2.5–4.6$\times$ faster than Spectral Mean Flow (Table 10).

Functional Generation. Functional diffusion models define generative processes on function spaces (Kerrigan et al., 2023; Lim et al., 2025), while Functional Flow Matching transports a Gaussian reference measure to the data distribution (Kerrigan et al., 2024). These methods evaluate neural operators on discretized grids, so their per-step cost typically grows with the number of time points. Functional Mean Flow targets one-step generation in Hilbert space (Li et al., 2026); Trajectory Flow Matching trains neural SDEs for irregular trajectories (Zhang et al., 2024); Generator Matching handles jump processes through Markov generators (Holderrieth et al., 2025). GalerkinFlow instead fixes a spline space in advance, so its network always processes K coefficient tokens whatever the grid resolution, and Theorem 4.1 gives its functional error exactly.

Galerkin Methods in Deep Learning. The Deep Galerkin Method uses neural networks as PDE trial functions (Sirignano & Spiliopoulos, 2018), hp-VPINNs use Petrov–Galerkin formulations (Kharazmi et al., 2021), the Galerkin Transformer applies the same view to operator learning (Cao, 2021), and Neural Galerkin schemes use variational projection for evolution equations (Bruna et al., 2024). GalerkinFlow brings this principle to generation: it projects each window onto a fixed cubic B-spline space, which is a Galerkin projection with identical trial and test spaces, and learns the distribution of the resulting coefficients.

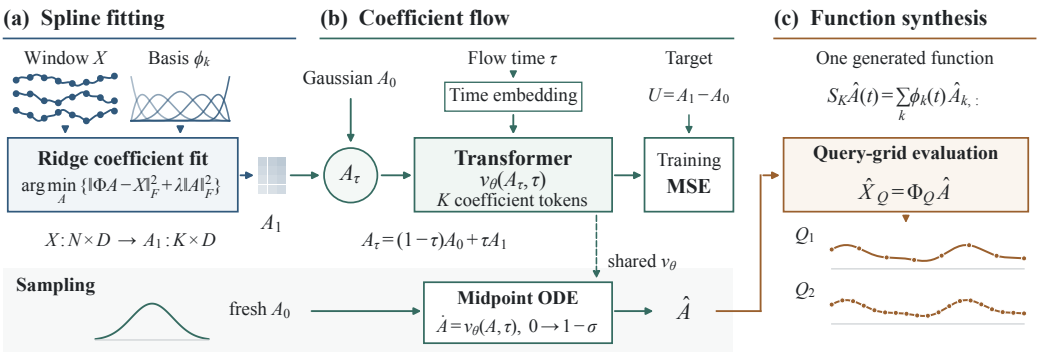


Figure 1: GalerkinFlow overview. (a) Ridge fitting maps each observed sequence to coefficients in a shared cubic B-spline basis. (b) Flow matching learns the coefficient distribution by predicting the velocity along straight paths from Gaussian noise to data; sampling follows the learned velocity from fresh noise. (c) The generated coefficients are decoded on the prescribed sampling grid.

3 Flow Matching in Spline Coefficient Space

Figure 1 summarizes the pipeline: discrete observations are mapped to spline coefficients, flow matching learns the joint distribution of these coefficients, and generated coefficients are decoded back to the prescribed sampling grid.

3.1 Representation and Sampled Fitting

We consider multivariate time series observed as discrete samples $x_n \in \mathbb{R}^D$ at times $t_n \in [0,1]$, $n = 1, \dots, N$. We do not assume that the data arise from an inherently continuous process. Instead, GalerkinFlow fits each sequence with K cubic B-spline basis functions, yielding a coefficient matrix $A \in \mathbb{R}^{K\times D}$ on which flow matching is performed. The fitted spline lies in a finite-dimensional subspace $V_K \subset \mathcal{H} = L^2([0,1];\mathbb{R}^D)$, where we analyze approximation and transport errors. As a result, the object being generated always has size $K \times D$, whatever the number of observation times N.

To instantiate this representation, we use $K \geq 4$ cubic B-splines with uniform knot spacing $h = 1/(K-3)$ and the open, clamped knot vector

$$(0,0,0,0,h,2h,\dots,(K-4)h,1,1,1,1), \tag{1}$$

which has no interior knots when $K = 4$. The resulting basis functions $\{\phi_k\}_{k=1}^K$ span V_K componentwise. They are nonnegative, sum to one at every t, are C^2 across interior knots, and each is supported on at most four knot intervals (de Boor, 1978).

The k-th row of A specifies the contribution of ϕ_k to each of the D channels, so the spline reconstruction at time t is

$$S_K A(t) = \sum_{k=1}^{K} \phi_k(t) A_{k,:}. \tag{2}$$

To evaluate it at the observation times, we define the basis matrix $\Phi \in \mathbb{R}^{N\times K}$ by $\Phi_{nk} = \phi_k(t_n)$. The reconstructed sequence is then $\widehat{X} = \Phi A$, with rows $\widehat{X}_{n,:} = S_K A(t_n)$. Thus, A serves as the generative representation of the sequence, and Φ maps it back to the prescribed sampling grid.

With the representation fixed, the remaining question is how to estimate the coefficients from discrete, possibly incomplete, observations. The ideal L^2 projector $P_K : \mathcal{H} \to V_K$ is defined through continuous inner products, but the data provide only finitely many samples. We therefore estimate the coefficients by ridge regression. For a complete sequence $X \in \mathbb{R}^{N\times D}$,

$$\widehat{C}_{K,\lambda}(X) = \arg\min_A \|\Phi A - X\|_F^2 + \lambda\|A\|_F^2 = (\Phi^\top\Phi + \lambda I)^{-1}\Phi^\top X. \tag{3}$$

In practice, we solve the corresponding linear system rather than forming the inverse. The regularization $\lambda > 0$ guarantees a unique solution even when the samples do not determine all coefficients, as on our regular benchmarks, where $K = 32 > N = 24$.

3.2 Coefficient flow and numerical sampling

Once each sequence is represented by its spline coefficients, we model their joint distribution directly in coefficient space with conditional flow matching. Because the synthesis map S_K is linear, errors in coefficient space translate directly into errors between functions, a connection we make precise in Section 4. For each sequence, let $A_1 = \widehat{C}_{K,\lambda}(X)$ denote its fitted coefficients. We draw a prior sample $A_0 \in \mathbb{R}^{K\times D}$ with i.i.d. entries $(A_0)_{kd} \sim \mathcal{N}(0, \alpha^2)$, independent of A_1, and a flow time $\tau \sim \mathcal{U}[0, 1]$. The linear interpolation path and its conditional velocity are

$$A_\tau = (1-\tau)A_0 + \tau A_1, \qquad U = A_1 - A_0. \tag{4}$$

We train the velocity field v_θ with the conditional flow-matching objective (Lipman et al., 2023; Liu et al., 2023)

$$\mathcal{L}_F(\theta) = \mathbb{E}\|v_\theta(A_\tau, \tau) - U\|_F^2. \tag{5}$$

The endpoints A_0 and A_1 are paired independently, without an optimal-transport coupling. Since S_K is linear,

$$S_K A_\tau = (1-\tau)S_K A_0 + \tau S_K A_1, \tag{6}$$

so linear interpolation between coefficients is exactly linear interpolation between the corresponding spline trajectories.

We parameterize v_θ with a Transformer that treats each row of A as one coefficient token. A linear input map and a learned basis-index embedding give $H_k^{(0)} = A_{k,:}W_{\text{in}} + e_k$, where $W_{\text{in}} \in \mathbb{R}^{D\times d}$. Sinusoidal features of τ pass through a two-layer MLP and condition each Transformer block through adaptive layer normalization (Peebles & Xie, 2023). Self-attention couples all K tokens, and a final linear map returns the $K \times D$ velocity.

At sampling time, we draw fresh Gaussian coefficients and integrate $\dot{A}(\tau) = v_\theta(A(\tau), \tau)$ up to $T_\sigma = 1 - \sigma$, with $\sigma = 0.01$. For S midpoint steps, let $\Delta\tau = T_\sigma/S$ and $\tau_s = s\Delta\tau$. Each update is

$$A^{s+1} = A^s + \Delta\tau\, v_\theta\left(A^s + \frac{\Delta\tau}{2} v_\theta(A^s, \tau_s), \tau_s + \frac{\Delta\tau}{2}\right). \tag{7}$$

The final coefficients A^S are decoded on the prescribed grid as $\Phi A^S \in \mathbb{R}^{N\times D}$. Proposition A.1 bounds the resulting error in terms of velocity learning, endpoint stopping, and numerical integration.

3.3 Masked observations

The same representation extends to incomplete sequences. Let $M \in \{0, 1\}^{N\times D}$ mark the observed entries. We linearly interpolate each channel onto the common grid and fit $A_1 = \widehat{C}_{K,\lambda}(\text{Fill}(X, M))$ with Eq. (3), so every window keeps a coefficient representation of the same size regardless of its missingness pattern. For $\|M\|_1 > 0$, we supervise the decoded velocity only at observed entries:

$$\mathcal{L}_{\text{mask}}(\theta) = \mathbb{E}\frac{\|M \odot \Phi[v_\theta(A_\tau, \tau) - (A_1 - A_0)]\|_F^2}{\|M\|_1}. \tag{8}$$

The loss averages over the available measurements: interpolated values shape the coefficient path but are never penalized directly. The shared representation and this observation-aware loss let the model train on windows with different missingness patterns while still generating complete sequences on the prescribed grid.

4 Representation, fitting, and transport error

GalerkinFlow learns a distribution over spline coefficients and decodes each sample as a function in V_K. A first question is how close such samples can get to the data, even if the coefficient distribution

were learned perfectly. We answer it exactly, and then add the errors from fitting coefficients to discrete observations and from learning and sampling their distribution.

We measure trajectory errors in $\mathcal{H} = L^2([0,1];\mathbb{R}^D)$ and write P_K for the orthogonal projector onto V_K. To compare them with errors in the coefficients, we use the norm induced by the spline basis:

$$G_{jk} = \int_0^1 \phi_j(t)\phi_k(t)\,dt, \qquad \|a\|_G^2 = \operatorname{tr}(a^\top G a). \tag{9}$$

Linear independence makes G positive definite. Moreover, $\|S_K a\|_{\mathcal{H}} = \|a\|_G$, so this norm measures a coefficient error by the size of the function it produces. We write $W_{2,G}$ for the corresponding Wasserstein distance.

With this metric, the representation and coefficient-law errors decouple.

Theorem 4.1 (Representation–Transport Decomposition)**.** *Let $F \sim \nu_X$ be a random signal with $\mathbb{E}\|F\|_{\mathcal{H}}^2 < \infty$, let $\nu_K = (S_K^{-1}P_K)_{\#}\nu_X$, and let $\widehat{\nu}_K$ be any coefficient law with a finite second moment. Define $b_K^2 = \mathbb{E}\|F - P_K F\|_{\mathcal{H}}^2$. Then*

$$W_{2,\mathcal{H}}^2((S_K)_{\#}\widehat{\nu}_K, \nu_X) = b_K^2 + W_{2,G}^2(\widehat{\nu}_K, \nu_K). \tag{10}$$

In particular, synthesis preserves Wasserstein distance when coefficients are measured in the Gram metric:

$$W_{2,\mathcal{H}}((S_K)_{\#}\mu, (S_K)_{\#}\nu) = W_{2,G}(\mu,\nu). \tag{11}$$

In words, the first term is the error caused by the spline space itself, which no amount of training can remove, and the second term is the only part that learning the coefficient distribution can reduce. Even a perfect coefficient model stays at distance b_K from the data, and no distribution on V_K can do better; $(P_K)_{\#}\nu_X$ is the unique best approximation on V_K, so ν_K is the right target in coefficient space.

The theorem needs only a finite second moment. How fast b_K shrinks as K grows depends on the smoothness of F. For the C^2 cubic-spline space with uniform knot spacing $h = 1/(K-3)$,

$$b_K \le (h/\pi)^s \big(\mathbb{E}\|F^{(s)}\|_{\mathcal{H}}^2\big)^{1/2}, \quad s \in \{1,2,3,4\}, \tag{12}$$

when $F \in H^s$ almost surely and the derivative moment is finite (Sande et al., 2019). For smooth signals ($s = 4$), halving the knot spacing reduces this bound by a factor of 16. GalerkinFlow does not require such smoothness; it only affects how large K must be.

Estimating Coefficients from Observations. The ideal coefficients use the continuous projection $P_K F$, but the encoder only sees finitely many samples: it fills missing values by interpolation and fits coefficients by ridge regression. Let A^{fit} be its output, assume $\mathbb{E}\|A^{\mathrm{fit}}\|_G^2 < \infty$, and let η_K be the distribution of A^{fit}. We measure the fitting error by comparing the fitted spline with the ideal projection of the same signal:

$$e_{\mathrm{fit}}^2 = \mathbb{E}\|S_K A^{\mathrm{fit}} - P_K F\|_{\mathcal{H}}^2. \tag{13}$$

Proposition 4.2 (Error of Sampled Encoding)**.** *Under the assumptions above, the reconstruction error satisfies*

$$e_{\mathrm{recon}}^2 := \mathbb{E}\|F - S_K A^{\mathrm{fit}}\|_{\mathcal{H}}^2 = b_K^2 + e_{\mathrm{fit}}^2. \tag{14}$$

For any output coefficient law $\widehat{\nu}_K$ with a finite second moment,

$$W_{2,\mathcal{H}}((S_K)_{\#}\widehat{\nu}_K, \nu_X) \le \Big[b_K^2 + \big(e_{\mathrm{fit}} + W_{2,G}(\widehat{\nu}_K, \eta_K)\big)^2\Big]^{1/2}. \tag{15}$$

Remarks. The reconstruction error in Eq. (14) is measured over the whole time interval, not only at the sampled times. When K exceeds the number of observations, a spline can pass through every sample yet stray from the true signal between them, so a small error at the samples does not mean an accurate reconstruction. Appendix B illustrates each error term numerically.

Learning and Sampling the Coefficient Law. The remaining term, $W_{2,G}(\widehat{\nu}_K, \eta_K)$, is the error of learning and sampling the coefficient distribution. Because S_K is linear, the straight training paths between coefficients are also straight paths between trajectories (Eq. (6)), and the best possible velocity field is the conditional mean $v^*(a,\tau) = \mathbb{E}[A_1 - A_0 \mid A_\tau = a]$. We train with the Euclidean coefficient loss, which also controls the error in function space because $\|u\|_G^2 \le \lambda_{\max}(G)\|u\|_F^2$. Under the assumptions of Proposition A.1, this error is bounded by three quantities: how well the network fits the velocity, where the sampler stops, and the error of the ODE solver.

5 EXPERIMENTS

We evaluate unconditional generation on regular, long-horizon, and missing-data benchmarks. Appendices D–E contain the ablation tables.

5.1 EVALUATION PROTOCOL

The regular benchmarks are Sines, Stocks, ETTh, MuJoCo, Energy, and fMRI, all with windows of length $N = 24$. FRED-MD ($N = 728$) and NN5 Daily ($N = 791$) test long windows; models are trained and evaluated at these lengths. For missing data, we remove 30%, 50%, or 70% of the observations on Stocks and evaluate irregular-to-regular (i2r) and irregular-to-irregular (i2i) generation. In both settings, the missing entries are removed from a common grid; neither setting tests generation at a new resolution.

We report Context-FID, correlation, discriminative, and predictive scores for regular data. Lower values are preferred for all four metrics. Specifically, Context-FID compares the empirical means and covariances of real and generated sequence embeddings:

$$\|\mu_r - \mu_g\|_2^2 + \operatorname{tr}\left[\Sigma_r + \Sigma_g - 2(\Sigma_r^{1/2}\Sigma_g\Sigma_r^{1/2})^{1/2}\right]. \tag{16}$$

The embeddings come from a 320-dimensional TS2Vec encoder (Yue et al., 2022) trained only on real data. For GalerkinFlow, the $\pm$ values of Context-FID are 95% confidence half-widths over five independently trained encoders, so they reflect evaluator variability rather than variation across trained generators; other $\pm$ values follow the conventions of the baselines (Appendix C). The correlation score compares the correlation matrices between channels of real and generated data, the discriminative score measures how far a classifier separating real from generated windows is from chance, and the predictive score is the error of a forecaster trained on generated data and tested on real data. The long-horizon metrics follow Kim et al. (2025); there, a higher classification score is better.

The default representation and prior hyperparameters are $K = 32$, $\alpha = 1$, and $\lambda = 10^{-6}$. On regular benchmarks $K = 32$ exceeds $N = 24$, so there is no token reduction on those datasets; the compression benefit applies to the long-horizon setting where $K = 32 \ll N = 728/791$. Table 1 uses 1,000 function evaluations (NFE), i.e., 500 midpoint steps; several ablations in Appendices D–E use smaller sampling budgets (40 or 50 NFE), which is why Context-FID values for the same dataset can differ across tables. Appendix C records all network, training, and sampling details.

5.2 REGULAR GENERATION

GalerkinFlow has the lowest Context-FID on all six regular benchmarks (Table 1), with especially large margins on ETTh (0.014 vs 0.058 for Spectral Mean Flow) and Energy (0.029 vs 0.051). It also has the lowest discriminative mean on four datasets and the lowest correlation mean on Stocks (tied), ETTh, and Energy; in total, it is lowest or tied for lowest in 17 of the 24 dataset–metric pairs. Against the functional baselines FFM-OT, TFM-U-SDE, and GM-Jump (Table 1), GalerkinFlow again has the lowest Context-FID on all six datasets and is lowest or tied in 20 of 24 pairs; FFM-OT is better on Sines correlation and discrimination and on Energy discrimination.

5.3 LONG-HORIZON AND IRREGULAR GENERATION

We next evaluate generation on much longer windows: 728 time steps for FRED-MD and 791 for NN5 Daily (Table 2). GalerkinFlow has the best displayed mean on all three metrics on both datasets. On FRED-MD, it lowers marginal error from 0.019 to 0.017 and predictive error from 0.030 to 0.026 relative to SMF, while raising the classification score from 0.992 to 1.027.

A similar pattern holds on NN5 Daily: GalerkinFlow improves marginal (0.007 vs 0.008), classification (0.911 vs 0.776), and predictive scores (0.418 vs 0.466), and outperforms ImagenTime on every metric. These results show that spline-coefficient flow matching scales effectively to sequences of several hundred time steps.

For irregular-to-irregular (i2i) generation, GalerkinFlow has lower mean discriminative scores than Spectral Mean Flow at every nonzero drop rate (Table 3). In irregular-to-regular (i2r) generation, it

Table 1: Regular generation with $N = 24$; lower values are better. Entries report means and uncertainty. The last three rows of each block are the functional baselines FFM-OT, GM-Jump, and TFM-U-SDE. Bold marks the lowest mean among the rows above them.

Method	Sines	Stocks	ETTh	MuJoCo	Energy	fMRI
Context-FID ↓						
GalerkinFlow	**0.003±.001**	**0.005±.000**	**0.014±.001**	**0.008±.000**	**0.029±.001**	**0.108±.004**
Spectral Mean Flow	0.004±.001	0.008±.003	0.058±.007	0.018±.002	0.051±.009	0.116±.004
Diffusion-TS	0.013±.001	0.169±.021	0.126±.007	0.015±.001	0.113±.011	0.118±.007
DiffTime	0.006±.001	0.236±.074	0.299±.044	0.188±.028	0.279±.045	0.340±.015
DiffWave	0.014±.002	0.232±.032	0.873±.061	0.393±.041	1.031±.131	0.244±.018
TimeGAN	0.101±.014	0.103±.013	0.300±.013	0.563±.052	0.767±.103	1.292±.218
TimeVAE	0.307±.060	0.215±.035	0.805±.186	0.251±.015	1.631±.142	14.449±.969
COT-GAN	1.337±.068	0.408±.086	0.980±.071	1.094±.079	1.039±.028	7.813±.550
FFM-OT	0.015±.083	0.029±.065	0.038±.002	0.018±.077	0.035±.069	0.177±.028
GM-Jump	0.012±.050	0.057±.035	0.286±.034	0.112±.055	0.570±.098	0.311±.025
TFM-U-SDE	4.158±.036	0.595±.064	4.154±.062	3.568±.037	1.383±.015	10.369±.062
Correlation ↓						
GalerkinFlow	0.019±.003	**0.006±.003**	**0.027±.007**	0.178±.014	**0.671±.091**	0.763±.024
Spectral Mean Flow	0.027±.012	0.010±.007	0.040±.015	**0.173±.016**	0.732±.107	**0.737±.021**
Diffusion-TS	**0.016±.005**	0.010±.009	0.049±.013	0.188±.035	0.788±.075	1.252±.070
DiffTime	0.017±.004	**0.006±.002**	0.067±.005	0.218±.031	1.158±.095	1.501±.048
DiffWave	0.022±.005	0.030±.020	0.175±.006	0.579±.018	5.001±.154	3.927±.049
TimeGAN	0.045±.010	0.063±.005	0.210±.006	0.886±.039	4.010±.104	23.502±.039
TimeVAE	0.131±.010	0.095±.008	0.111±.020	0.388±.041	1.688±.226	17.296±.526
COT-GAN	0.049±.010	0.087±.004	0.249±.009	1.042±.007	3.164±.061	26.824±.449
FFM-OT	0.016±.089	0.009±.054	0.069±.018	0.243±.055	0.748±.083	0.949±.010
GM-Jump	0.025±.034	0.019±.001	0.080±.090	0.227±.076	1.407±.027	3.386±.025
TFM-U-SDE	0.525±.066	0.123±.026	0.338±.087	1.038±.052	1.851±.093	3.288±.089
Discriminative ↓						
GalerkinFlow	0.007±.004	**0.012±.009**	**0.015±.006**	0.006±.006	**0.091±.061**	**0.091±.035**
Spectral Mean Flow	**0.006±.006**	0.022±.013	0.027±.010	**0.005±.004**	0.161±.021	0.136±.207
Diffusion-TS	0.030±.006	0.085±.026	0.075±.007	0.012±.006	0.154±.012	0.158±.020
DiffTime	0.013±.006	0.097±.016	0.100±.007	0.154±.045	0.445±.004	0.245±.051
DiffWave	0.017±.008	0.232±.061	0.190±.008	0.203±.096	0.493±.004	0.402±.029
TimeGAN	0.011±.008	0.102±.021	0.114±.055	0.238±.068	0.236±.012	0.484±.042
TimeVAE	0.041±.044	0.145±.120	0.209±.058	0.230±.102	0.499±.000	0.476±.044
COT-GAN	0.254±.137	0.230±.016	0.325±.099	0.426±.022	0.498±.002	0.492±.018
RNN-AR	0.495±.001	0.226±.035	—	—	0.483±.004	—
FFM-OT	0.006±.019	0.026±.037	0.022±.033	0.015±.046	0.046±.043	0.133±.036
GM-Jump	0.012±.073	0.118±.060	0.099±.076	0.095±.097	0.499±.032	0.448±.043
TFM-U-SDE	0.298±.020	0.228±.096	0.401±.037	0.453±.005	0.499±.074	0.401±.022
Predictive ↓						
GalerkinFlow	**0.093±.000**	**0.037±.000**	**0.121±.007**	0.008±.001	0.253±.000	**0.100±.000**
Spectral Mean Flow	**0.093±.000**	**0.037±.000**	0.123±.005	0.008±.001	**0.251±.000**	**0.100±.000**
Diffusion-TS	0.095±.000	**0.037±.000**	**0.121±.002**	**0.007±.001**	**0.251±.000**	**0.100±.000**
DiffTime	**0.093±.000**	0.038±.001	**0.121±.004**	0.010±.001	0.252±.000	**0.100±.000**
DiffWave	**0.093±.000**	0.047±.000	0.130±.001	0.013±.000	**0.251±.000**	0.101±.000
TimeGAN	**0.093±.019**	0.038±.001	0.124±.001	0.025±.003	0.273±.004	0.126±.002
TimeVAE	**0.093±.000**	0.039±.001	0.126±.004	0.012±.002	0.292±.000	0.113±.003
COT-GAN	0.100±.000	0.047±.001	0.129±.000	0.068±.009	0.259±.000	0.185±.003
RNN-AR	0.150±.022	0.038±.001	—	—	0.315±.005	—
FFM-OT	**0.093±.000**	**0.037±.000**	**0.121±.000**	0.011±.000	**0.251±.000**	**0.100±.000**
GM-Jump	0.093±.000	0.038±.000	0.123±.000	0.009±.000	0.251±.000	0.101±.000
TFM-U-SDE	0.096±.000	0.037±.000	0.137±.000	0.031±.000	0.290±.000	0.102±.000

has a lower reported mean at 30% missingness, matches Spectral Mean Flow at 50%, and remains close at 70% (0.016 versus 0.015). In a control experiment, removing the interpolation step raises the discriminative score from 0.021 to 0.178 (Table 15). GalerkinFlow thus learns well from incomplete

Table 2: Long-horizon generation on FRED-MD ($N = 728$) and NN5 Daily ($N = 791$). Arrows indicate the preferred direction.

	FRED-MD			NN5 Daily		
Method	Marginal ↓	Class. ↑	Pred. ↓	Marginal ↓	Class. ↑	Pred. ↓
GalerkinFlow	**0.017**	**1.027±.112**	**0.026±.006**	**0.007**	**0.911±.104**	**0.418±.037**
Spectral Mean Flow	0.019	0.992±.106	0.030±.006	0.008	0.776±.068	0.466±.045
ImagenTime	0.022	0.766±.259	0.034±.020	0.009	0.560±.174	0.584±.188

Table 3: Stocks generation with missing observations: discriminative score (↓). Bold marks the lowest displayed mean, including ties. Appendix D reports the masked-loss controls.

Method	0% drop	30% drop	50% drop	70% drop
Irregular to regular				
GalerkinFlow	**0.006±.002**	**0.019±.016**	**0.019±.014**	0.016±.012
Spectral Mean Flow	0.009±.008	0.020±.011	**0.019±.008**	**0.015±.007**
Koopman VAE	0.021±.022	0.109±.051	0.067±.038	0.049±.052
GT-GAN	0.077±.031	0.251±.097	0.265±.073	0.230±.053
TimeGAN	0.102±.021	0.411±.040	0.477±.021	0.485±.022
RCGAN	0.196±.027	0.436±.064	0.478±.049	0.381±.086
C-RNN-GAN	0.399±.028	0.500±.000	0.500±.000	0.500±.000
RNN-AR	0.226±.035	0.305±.002	0.308±.010	0.317±.019
Irregular to irregular				
GalerkinFlow	**0.006±.002**	**0.010±.004**	**0.026±.013**	**0.033±.047**
Spectral Mean Flow	0.009±.008	0.049±.017	0.044±.017	0.138±.137
Koopman VAE	0.021±.022	0.227±.096	0.211±.078	0.187±.075

sequences: interpolation gives every window a coefficient representation of the same size, and the masked loss trains only on the entries that were actually observed.

Table 4: Stocks generation across bases with matched model capacity (517,318 parameters). Lower values are better.

Basis	FID	Corr.	Disc.	Pred.
Cubic B-spline	**0.009**	**0.003**	0.012	0.0368
Fourier	0.033	0.008	0.017	0.0369
Legendre	0.032	0.006	**0.008**	0.0369
Haar	0.033	0.008	0.014	0.0369

5.4 Basis and optimization controls

We first replace splines with Fourier, Legendre, and Haar bases at matched capacity and coefficient scale (Table 4). Splines give the lowest Context-FID (0.009, about 73% lower than Legendre at 0.032) and halve the correlation error (0.003 vs 0.006), while Legendre attains the lowest discriminative score. Since splines and Legendre have nearly identical reconstruction RMSEs (0.019 vs 0.020), the gap reflects how each basis shapes the coefficient distribution rather than fit quality. Fourier and Legendre functions are global, so an error in one generated coefficient spreads across the whole window; Haar functions are local but discontinuous, so small errors appear as jumps. Cubic B-splines are the only basis here that is both local and smooth: each coefficient affects at most four knot intervals, giving every token a localized temporal meaning, while C^2 regularity avoids artificial discontinuities.

To separate the representation from the architecture, we further compare against a raw grid baseline that feeds the $N = 24$ sampled values directly to the same Transformer (Table 5). Both models share the token count, the 518,182 parameters, and the training recipe, and since the ridge fit is nearly exact at this basis size, they differ only in the coordinates in which each window is expressed.

The spline model lowers Context-FID by roughly 30% (0.005 vs 0.007), halves the correlation error (0.006 vs 0.012), and reduces the discriminative score by one third (0.012 vs 0.018). One likely reason is the prior: independent Gaussian coefficients decode to smooth random trajectories, whereas independent Gaussian values on the raw grid are white noise, so by Eq. (6) the spline flow starts from and moves through trajectories that already resemble the data. Both comparisons thus indicate that the gains stem from a deliberate representational choice rather than model capacity: a local, smooth, fixed basis yields a coefficient space that is easier to model than raw values.

Table 5: Stocks: B-spline Galerkin vs. raw-grid identity at matched state dimension and parameter count (518,182). Lower is better.

Representation	Parameters	FID	Corr.	Disc.	Pred.
B-spline Galerkin	518,182	**0.005**	**0.006**	**0.012**	0.0368
Raw-grid identity	518,182	0.007	0.012	0.018	0.0368

Training longer also matters: continuing the $K = 48$ run from 600 to 1,200 epochs lowers Context-FID by 46% ($0.0074 \to 0.004$; Table 18), so a sweep over K at a fixed number of epochs mixes the effect of the basis size with that of incomplete training. Among ODE solvers with the same budget of 40 function evaluations, the midpoint method scores best on all four metrics (Context-FID 0.003; Table 13). Its quality barely changes between 20 and 500 evaluations while sampling time grows $23\times$ (Table 14), so a moderate budget suffices. Each of these ablations holds part of the error budget of Section 4 fixed: longer training can only reduce the transport term, and the solver affects only the integration error. Appendix B reads all ablations in this way and illustrates each error term numerically.

5.5 Computational cost

Generating one window with N time points and D channels, using E velocity evaluations of a Transformer with L blocks, hidden width d, and feedforward width d_{ff} on K tokens, costs

$$\mathcal{O}\big(E\left[L(K^2 d + K d^2 + K d d_{\text{ff}}) + KDd\right] + NKD\big)\,. \tag{17}$$

Every sampling step depends only on K; the window length N enters once, in the final decoding. In practice, GalerkinFlow has the shortest training and sampling times on all six regular benchmarks (Tables 10 and 11). It trains in 5–13 minutes and generates 1,000 windows in 10–57 seconds, compared with 15–60 minutes and 29–109 seconds for Spectral Mean Flow. On Stocks, increasing the sampling budget from 50 to 500 function evaluations raises runtime from 26.6 to 237.3 seconds without improving Context-FID (Table 14).

6 Conclusion

GalerkinFlow represents each time-series window with coefficients of a fixed cubic B-spline basis. We fit these coefficients from the available observations, learn their joint distribution with standard flow matching, and decode generated coefficients with a fixed linear map. This gives regular and masked observations a common representation and makes the basis size an explicit modeling choice. Our analysis splits the generation error into what the spline space allows, what is lost when fitting coefficients to samples, and what remains from learning their distribution. GalerkinFlow achieves the lowest Context-FID on all six regular benchmarks, and in irregular-to-irregular generation its mean discriminative score is lower than that of Spectral Mean Flow at every tested nonzero drop rate. Ablation studies on Stocks favor cubic splines for Context-FID and correlation error, while showing that training duration and solver choice also contribute. Several limitations remain: uniform knot spacing handles sharp transients less well than adaptive knots, on short windows with $K > N$ some coefficient directions are not determined by the observed samples, and the theoretical analysis does not cover the masked objective. Measuring functional generation error directly rather than through grid-based proxies is a natural direction for future work.

AI USE STATEMENT

Generative AI tools assisted with manuscript writing and editing.

REFERENCES

Joan Bruna, Benjamin Peherstorfer, and Eric Vanden-Eijnden. Neural galerkin schemes with active learning for high-dimensional evolution equations. *Journal of Computational Physics*, 496: 112588, 2024. doi: 10.1016/j.jcp.2023.112588. URL https://doi.org/10.1016/j.jcp.2023.112588.

Shuhao Cao. Choose a transformer: Fourier or galerkin. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/d0921d442ee91b896ad95059d13df618-Abstract.html.

Andrea Coletta, Sriram Gopalakrishnan, Daniel Borrajo, and Svitlana Vyetrenko. On the constrained time-series generation problem. In *Advances in Neural Information Processing Systems*, volume 36, pp. 61048–61059, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/bfb6a69c0d9e2bc596e1cd31f16fcdde-Abstract-Conference.html.

Carl de Boor. *A Practical Guide to Splines*, volume 27 of *Applied Mathematical Sciences*. Springer-Verlag, 1978. doi: 10.1007/978-1-4612-6333-3.

Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. TimeVAE: A variational auto-encoder for multivariate time series generation. *arXiv preprint arXiv:2111.08095*, 2021. URL https://arxiv.org/abs/2111.08095.

Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional GANs. *arXiv preprint arXiv:1706.02633*, 2017. URL https://arxiv.org/abs/1706.02633.

Rakshitha W. Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *NeurIPS Datasets and Benchmarks*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/eddea82ad2755b24c4e168c5fc2ebd40-Paper-round2.pdf.

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. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/819aaee144cb40e887a4aa9e781b1547-Abstract-Conference.html.

Paul Jeha, Michael Bohlke-Schneider, Pedro Mercado, Shubham Kapoor, Rajbir Singh Nirwan, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. PSA-GAN: Progressive self attention GANs for synthetic time series. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Ix_mh42xq5w.

Jinsung Jeon, Jeonghak Kim, Haryong Song, Seunghyeon Cho, and Noseong Park. GT-GAN: General purpose time series synthesis with generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 35, pp. 36999–37010, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/f03ce573aa8bce26f77b76f1cb9ee979-Abstract-Conference.html.

Gavin Kerrigan, Justin Ley, and Padhraic Smyth. Diffusion generative models in infinite dimensions. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 9538–9563, 2023. URL https://proceedings.mlr.press/v206/kerrigan23a.html.

Gavin Kerrigan, Giosue Migliorini, and Padhraic Smyth. Functional flow matching. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 3934–3942, 2024. URL https://proceedings.mlr.press/v238/kerrigan24a.html.

Ehsan Kharazmi, Zhongqiang Zhang, and George Em Karniadakis. hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, 2021. doi: 10.1016/j.cma.2020.113547. URL https://doi.org/10.1016/j.cma.2020.113547.

Jinwoo Kim, Max Beier, Petar Bevanda, Nayun Kim, and Seunghoon Hong. Sequence modeling with spectral mean flows. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://papers.neurips.cc/paper_files/paper/2025/hash/c03b51b0d53f9d664572ddd37a876610-Abstract-Conference.html.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=a-xFK8Ymz5J.

Daesoo Lee, Sara Malacarne, and Erlend Aune. Vector quantized time series generation with a bidirectional prior model. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 7665–7693. PMLR, 2023. URL https://proceedings.mlr.press/v206/lee23d.html.

Zhiqi Li, Yuchen Sun, Greg Turk, and Bo Zhu. Functional mean flow in Hilbert space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1928–1938, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Li_Functional_Mean_Flow_in_Hilbert_Space_CVPR_2026_paper.html.

Jae Hyun Lim, Nikola B. Kovachki, Ricardo Baptista, Christopher Beckham, Kamyar Azizzadenesheli, Jean Kossaifi, Vikram Voleti, Jiaming Song, Karsten Kreis, Jan Kautz, Christopher Pal, Arash Vahdat, and Anima Anandkumar. Score-based diffusion models in function space. *Journal of Machine Learning Research*, 26(158):1–62, 2025. URL https://www.jmlr.org/papers/v26/23-1472.html.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

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

Olof Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016. URL https://arxiv.org/abs/1611.09904.

Ilan Naiman, Nimrod Berman, Itai Pemper, Idan Arbiv, Gal Fadlon, and Omri Azencot. Utilizing image transforms and diffusion models for generative modeling of short and long time series. In *Advances in Neural Information Processing Systems*, volume 37, pp. 121699–121730, 2024a. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/dc6748383752138af7f00b3185a0a404-Abstract-Conference.html.

Ilan Naiman, N. Benjamin Erichson, Pu Ren, Michael W. Mahoney, and Omri Azencot. Generative modeling of regular and irregular time series data via Koopman VAEs. In *International Conference on Learning Representations*, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/dbeb7e621d4a554069a6a775da0f7273-Abstract-Conference.html.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Peebles_Scalable_Diffusion_Models_with_Transformers_ICCV_2023_paper.html.

Espen Sande, Carla Manni, and Hendrik Speleers. Sharp error estimates for spline approximation: explicit constants, n-widths, and eigenfunction convergence. *Mathematical Models and Methods in Applied Sciences*, 29(6):1175–1205, 2019. doi: 10.1142/S0218202519500192. URL https://arxiv.org/abs/1810.13418.

Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018. doi: 10.1016/j.jcp.2018.08.029. URL https://doi.org/10.1016/j.jcp.2018.08.029.

Tianlin Xu, Li Kevin Wenliang, Michael Munn, and Beatrice Acciaio. COT-GAN: Generating sequential data via causal optimal transport. In *Advances in Neural Information Processing Systems*, volume 33, pp. 8798–8809, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/641d77dd5271fca28764612a028d9c8e-Abstract.html.

Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/c9efe5f26cd17ba6216bbe2a7d26d490-Abstract.html.

Xinyu Yuan and Yan Qiao. Diffusion-TS: Interpretable diffusion for general time series generation. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4h1apFjO99.

Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. TS2Vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8980–8987, 2022. doi: 10.1609/aaai.v36i8.20881. URL https://ojs.aaai.org/index.php/AAAI/article/view/20881.

Xi Zhang, Yuan Pu, Yuki Kawamura, Andrew Loza, Yoshua Bengio, Dennis L. Shung, and Alexander Tong. Trajectory flow matching with applications to clinical time series modelling. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/c1e67cde895c3c91edb43569ad0df260-Abstract-Conference.html.

A MATHEMATICAL ANALYSIS AND PROOFS

A.1 NOTATION

In this section, we introduce the notation used throughout the paper. We use t for physical time and τ for flow time. Capital letters denote random signals or coefficient matrices, and lowercase a denotes a generic coefficient matrix. The synthesis map S_K of Eq. (2) is used throughout. For a measurable map T, the pushforward $T_{\#}\nu$ is the law of $T(X)$ when $X \sim \nu$.

Symbol	Meaning
N, K, D	Observation count, spline-basis size, and channel count
$\mathcal{H}, V_K$	Ambient L^2 space and its finite spline subspace
ν_X	Target law of the random signal F
P_K, S_K	Ideal orthogonal projector and coefficient-to-function synthesis
Φ, G	Sampled design matrix and continuous spline Gram matrix
$\widehat{C}_{K,\lambda}$	Sampled ridge encoder; distinct from $S_K^{-1}P_K$
ν_K	Ideal coefficient law $(S_K^{-1}P_K)_{\#}\nu_X$
$\eta_K, \eta_K^{\mathrm{emp}}$	Population and empirical laws of fitted coefficients
$\widehat{\nu}_K$	Output coefficient law of the numerical generator
A_0, A_1, A_τ	Gaussian prior, fitted training target, and interpolated state
b_K, e_{fit}	Ideal representation RMS and sampled-fit discrepancy RMS
$\mathcal{R}_G$	Population excess velocity-regression risk in Gram geometry
$\sigma, \delta_{\mathrm{num}}$	Endpoint stopping parameter and numerical-flow error

A.2 SPLINE CONVENTIONS AND SYNTHESIS

Take $K \geq 4$, set $J = K - 3$, and use the open knot vector

$$(0, 0, 0, 0, 1/J, 2/J, \ldots, (J-1)/J, 1, 1, 1, 1).$$

The normalized degree-three B-splines span the full C^2 piecewise-cubic space on these J intervals. In the Cox–de Boor recursion, terms with zero denominators are set to zero, and endpoint values are defined by continuity. Implementations that use an extended uniform knot vector instead must define the active basis functions and the endpoint values accordingly.

Expanding the integral, for any a we obtain

$$\|S_K a\|_{\mathcal{H}}^2 = \sum_{d=1}^{D} a_{:,d}^\top G a_{:,d} = \|a\|_G^2.$$

Hence $\|S_K\|_{F\to\mathcal{H}} = \sqrt{\lambda_{\max}(G)}$. Nonnegativity and partition of unity of the normalized B-splines give

$$\sum_k G_{jk} = \int_0^1 \phi_j(t)\, dt = \frac{u_{j+3} - u_{j-1}}{4} \leq h,$$

where $u_0, \ldots, u_{K+3}$ are zero-indexed knots and $j = 1, \ldots, K$ indexes the basis functions. The row-sum bound gives $\lambda_{\max}(G) \leq h$, which is at most $2/K$ for $K \geq 6$. This argument also covers the open-knot boundary splines. Hence $\|a\|_G \leq \sqrt{2/K}\, \|a\|_F$: a Euclidean coefficient error shrinks by a factor of order $K^{-1/2}$ when measured in function space. Rescaling the basis would change this factor and rescale the coefficient distribution inversely.

A.3 PROOF OF THEOREM 4.1

We first prove the isometry property. The synthesis map is a continuous linear bijection from coefficient space onto V_K, with a continuous inverse. Pushing a coefficient coupling through $S_K \times S_K$ preserves its cost when coefficients are measured in the Gram metric. Conversely, every coupling of the synthesized laws is supported on $V_K \times V_K$ and pulls back through the inverse map with the same cost. Taking infima in both directions gives (11).

For $g \in V_K$ and $f \in \mathcal{H}$, orthogonality gives

$$\|g - f\|_{\mathcal{H}}^2 = \|g - P_K f\|_{\mathcal{H}}^2 + \|f - P_K f\|_{\mathcal{H}}^2.$$

Every coupling of $(S_K)_{\#}\widehat{\nu}_K$ and ν_X therefore has cost at least $W_{2,\mathcal{H}}^2((S_K)_{\#}\widehat{\nu}_K, (P_K)_{\#}\nu_X) + b_K^2$. To prove the reverse inequality, take an optimal coupling of the two laws on V_K and disintegrate ν_X conditional on $P_K F$. The conditional distribution exists because $\mathcal{H}$ is a separable Hilbert space. Given the projected component of the coupling, draw F from this conditional law. The resulting coupling has marginals $(S_K)_{\#}\widehat{\nu}_K$ and ν_X, and its cost equals the lower bound above. The isometry then gives (10).

A.4 Proof of Proposition 4.2

For each signal and its fitted coefficients, $F - P_K F$ is orthogonal to $P_K F - S_K A^{\mathrm{fit}}$. Expanding the squared norm and taking expectation gives

$$\mathbb{E}\|F - S_K A^{\mathrm{fit}}\|_{\mathcal{H}}^2 = b_K^2 + e_{\mathrm{fit}}^2.$$

The joint law of $S_K A^{\mathrm{fit}}$ and $P_K F$ is a coupling with squared cost e_{fit}^2. By the isometry, $W_{2,G}(\eta_K, \nu_K) \le e_{\mathrm{fit}}$. Substituting the triangle inequality into (10) proves (15).

A.5 Classical spline approximation rate

Apply Theorem 1 of Sande et al. (2019) with polynomial degree $p = 3$ to each channel, then square, sum, and take expectation to obtain (12). The finite sth-derivative moment is separate from, and does not replace, the finite second-moment assumption on F.

A.6 Error of the ridge fit

Suppose the observation model supplies $Y \in \mathbb{R}^{N \times D}$, a design matrix Φ, and weights $W = \mathrm{diag}(w_1, \dots, w_N)$ with $w_n \ge 0$. For point observations, assume a specified pointwise representative of F, since point evaluation is undefined on an arbitrary L^2 equivalence class. Let

$$c = S_K^{-1} P_K F, \qquad M_\Phi = \Phi^\top W \Phi, \qquad A^{\mathrm{fit}} = (M_\Phi + \lambda I)^{-1} \Phi^\top W Y, \quad \lambda > 0.$$

Set $e = Y - \Phi c$. Subtracting the normal equations gives the exact identity

$$A^{\mathrm{fit}} - c = (M_\Phi + \lambda I)^{-1} \big(\Phi^\top W e - \lambda c\big). \tag{18}$$

It follows that

$$\|A^{\mathrm{fit}} - c\|_G \le \frac{\sqrt{\lambda_{\max}(G)}}{\lambda_{\min}(M_\Phi) + \lambda} \left(\|\Phi^\top W e\|_F + \lambda \|c\|_F\right).$$

The residual e may include unresolved signal variation, observation noise, and interpolation error. Equation (18) separates their contribution from ridge bias without assuming either is small. The design rank is at most $\min(N, K)$ and may be lower on poorly placed grids. When $N < K$, observations do not identify all coefficients; reducing λ cannot recover directions in the null space of Φ.

A.7 A conditional flow-matching bound

The following result assumes a fixed positive-definite quadratic training metric and is conditional on the realized training sample. Let $A_1 \sim \eta_K^{\mathrm{emp}}$, the empirical law of fitted training coefficients, and independently draw $\mathrm{vec}(A_0) \sim \mathcal{N}(0, \alpha^2 I_{KD})$. Put $U = A_1 - A_0$, $A_\tau = (1-\tau)A_0 + \tau A_1$, $p_\tau = \mathcal{L}(A_\tau)$, and $v^*(a, \tau) = \mathbb{E}[U \mid A_\tau = a]$, understood as an almost-everywhere version of the conditional mean. For $0 < \sigma < 1$, abbreviate the sampler endpoint $T_\sigma = 1 - \sigma$ by T and define

$$\mathcal{R}_G = \int_0^T \mathbb{E}\|v_\theta(A_\tau, \tau) - v^*(A_\tau, \tau)\|_G^2 \, d\tau.$$

Different endpoint pairs can pass through the same state with different velocities. A state-dependent predictor estimates their conditional mean, leaving conditional variance in the regression loss. Orthogonality of conditional expectation gives

$$\int_0^T \mathbb{E}\|v_\theta(A_\tau, \tau) - U\|_G^2 \, d\tau = \mathcal{R}_G + \int_0^T \mathbb{E}\|U - v^*(A_\tau, \tau)\|_G^2 \, d\tau. \tag{19}$$

The variance term need not vanish. For the analogous Euclidean excess risk $\mathcal{R}_F$, we have $\mathcal{R}_G \leq \lambda_{\max}(G)\mathcal{R}_F$. This comparison does not require changing the implemented objective.

Proposition A.1 (From excess risk to sampled output)**.** *Assume v^* generates a nonexplosive flow with marginals p_τ on $[0,T]$. Assume v_θ generates a nonexplosive flow and is globally Lipschitz in its state under $\|\cdot\|_G$, with an integrable bound $L(\tau) \geq 0$. All quantities below are assumed finite. Define*

$$C_T = \left(\int_0^T \exp\left(2 \int_t^T L(s)\, ds \right) dt \right)^{1/2}.$$

Let $\widehat{\nu}_K$ be the numerical sampler's output law at T, and let δ_{num} bound its $W_{2,G}$ distance from the exactly integrated learned-flow law at T. Then

$$W_{2,G}(\widehat{\nu}_K, \eta_K^{\mathrm{emp}}) \leq \delta_{\mathrm{num}} + C_T \sqrt{\mathcal{R}_G} + \sigma \big(\mathbb{E} \|A_0 - A_1\|_G^2 \big)^{1/2}. \tag{20}$$

Proof. Couple a learned trajectory B_τ and a trajectory Z_τ of v^* using the same initial sample. Almost everywhere, their difference satisfies

$$\frac{d}{d\tau} \|B_\tau - Z_\tau\|_G \leq L(\tau) \|B_\tau - Z_\tau\|_G + \|v_\theta(Z_\tau, \tau) - v^*(Z_\tau, \tau)\|_G.$$

Apply Gronwall's inequality, the L^2 triangle inequality, and Cauchy–Schwarz in time to obtain

$$\big(\mathbb{E} \|B_T - Z_T\|_G^2 \big)^{1/2} \leq C_T \sqrt{\mathcal{R}_G}.$$

Because $Z_T \sim p_T$, this bounds $W_{2,G}(\mathcal{L}(B_T), p_T)$. Separately, the coupling (A_T, A_1) gives $W_{2,G}(p_T, \eta_K^{\mathrm{emp}}) \leq \sigma (\mathbb{E}\|A_0 - A_1\|_G^2)^{1/2}$. The triangle inequality with the numerical error gives the claim. The endpoint coupling concerns laws; an ODE trajectory need not follow any particular independently sampled endpoint pair. □

For the independent centered prior, $\mathbb{E}\|A_0 - A_1\|_G^2 = \alpha^2 D \operatorname{tr}(G) + \mathbb{E}\|A_1\|_G^2$. Choosing $\sigma = 0.01$ alone therefore does not guarantee a negligible endpoint error. Likewise, the midpoint solver has second-order convergence only under the required regularity and stability conditions. The bound keeps δ_{num} explicit.

Define the empirical-to-population coefficient-law gap by $e_{\mathrm{stat}} = W_{2,G}(\eta_K^{\mathrm{emp}}, \eta_K)$. Combining the preceding bounds gives

$$\begin{aligned} W_{2,\mathcal{H}}((S_K)_{\#}\widehat{\nu}_K, \nu_X) &\leq \left[b_K^2 + (e_{\mathrm{fit}} + e_{\mathrm{stat}} + e_{\mathrm{flow}})^2 \right]^{1/2}, \\ e_{\mathrm{flow}} &= \delta_{\mathrm{num}} + C_T \sqrt{\mathcal{R}_G} + \sigma \big(\mathbb{E}\|A_0 - A_1\|_G^2 \big)^{1/2}. \end{aligned} \tag{21}$$

This bound is conditional on the training sample and the stated assumptions; it is not an experimentally measured certificate. In particular, training loss alone neither estimates e_{stat} nor verifies the Lipschitz assumption.

A.8 Scope of the masked objective

Masking makes the quadratic regression metric sample-dependent. Flatten coefficient matrices consistently, and let Q_m denote the positive-semidefinite matrix of the mask-weighted quadratic form. For channel d, the diagonal block is $\Phi^\top \operatorname{diag}(m_{:,d})\Phi / \|m\|_1$, with nonempty masks assumed. For a fixed state and time, the population minimizer obeys

$$\mathbb{E}[Q_m \mid A_\tau = a] v = \mathbb{E}[Q_m U \mid A_\tau = a].$$

The ordinary marginal velocity is a minimizer only if $\mathbb{E}[Q_m(U - v^*(a,\tau)) \mid A_\tau = a] = 0$ (and uniqueness requires the conditional mean matrix to be positive definite). Random masking alone does not imply these conditions: both the fitted target and interpolated state depend on the mask. Proposition A.1 therefore does not establish consistency of the masked objective. The synthesis and representation identities still apply to its output law, and the irregular-data experiments assess this variant empirically.

Unobserved coefficient directions. Let Π_0 be the Euclidean orthogonal projector onto $\ker \Phi$, applied channelwise. For the specified fixed-grid ridge encoder, projecting the normal equations gives $\lambda \Pi_0 A_1 = 0$, so $\Pi_0 A_1 = 0$ whenever $\lambda > 0$, including for filled inputs. Along the linear path, $\Pi_0 U = -\Pi_0 A_\tau/(1-\tau)$. Taking conditional expectation yields $\Pi_0 v^*(a,\tau) = -\Pi_0 a/(1-\tau)$, p_τ-almost everywhere for $\tau < 1$. Under the exact-flow assumptions of Proposition A.1, the ideal ODE trajectory satisfies $\Pi_0 A(\tau) = (1-\tau)\Pi_0 A(0)$, leaving residual $\sigma \Pi_0 A(0)$ at T_σ. This contraction describes the unrestricted coefficient-regression minimizer. Both masked and unmasked sampled-space losses are invariant to velocity perturbations annihilated by Φ and therefore provide no direct supervision for it.

B THE ERROR DECOMPOSITION IN PRACTICE

Theorem 4.1 and Proposition 4.2 split the functional generation error of GalerkinFlow into three parts: the projection error b_K, which is fixed by the spline space; the fitting error e_{fit}, which is fixed by the encoder and the observation pattern; and the coefficient transport error $W_{2,G}$, which training and sampling can reduce. This appendix makes the decomposition concrete. We first verify the identities numerically, then visualize how the first two terms depend on the basis size, the observation density, and missing data, and finally read the ablations of Section 5.4 through the resulting error budget.

B.1 NUMERICAL VERIFICATION OF THE DECOMPOSITION

To check Eq. (10) independently of any spline code, we construct finite laws in $\mathbb{R}^9$ whose subspace V is spanned by a deliberately nonorthogonal 9×3 synthesis matrix, so that the Gram metric differs from the Euclidean one. The target law has six equally weighted atoms, and the generated coefficient law is a permuted and perturbed copy of their projections. Since both laws are uniform over six atoms, the Wasserstein distances are computed exactly by enumerating all $6! = 720$ assignments. Across three random seeds, the squared Wasserstein distance in the ambient space equals $b^2 + W_{2,G}^2$ up to an absolute error of at most 1.8×10^{-15}, and the optimal assignment in the ambient space coincides with the optimal assignment in Gram-metric coefficient space in every case, as the isometry in Eq. (11) predicts.

B.2 REPRESENTATION AND FITTING ERRORS

Setup. We draw random signals $F(t) = \sqrt{1/3} \sum_{q \in \{1,3,6\}} \left(z_{q,s} \sin 2\pi q t + z_{q,c} \cos 2\pi q t \right)$ with i.i.d. standard Gaussian weights z, so that $\mathbb{E}\|F\|_{\mathcal{H}}^2 = 1$ and all errors below are relative RMS errors. We use the same clamped cubic B-spline basis and ridge encoder as in the main experiments ($\lambda = 10^{-6}$), with $K \in \{8, 12, 16, 24, 32, 40, 48, 64, 96\}$, under three observation conditions: 24 complete uniform samples, which matches the regular benchmarks; 96 complete uniform samples; and 12 of the 24 samples retained, with the rest filled by linear interpolation exactly as in the masked-data encoder. Because F is Gaussian and every map involved is linear, the expectations in Eqs. (12)–(14) are computed exactly from response matrices by Gauss–Legendre quadrature with 12 nodes per knot interval. No network is trained and no Monte Carlo sampling is involved, and doubling the quadrature order changes the reported errors by less than 3×10^{-15}. In all 27 configurations, identity $e_{\text{recon}}^2 = b_K^2 + e_{\text{fit}}^2$ of Eq. (14) holds to within 2.5×10^{-15}. Figure 2 and Table 6 report the results.

Three regimes of basis size. Figure 2(a) shows the trade-off predicted by Proposition 4.2. When K is small, the projection term dominates. When K matches the number of observations, the projection and fitting terms balance, and the total error reaches its minimum of 0.024 at $K = N = 24$. Once K exceeds N, the projection error keeps shrinking, reaching 1.5×10^{-5} at $K = 96$, but fitting error grows from 0.10 at $K = 32$ to 0.62 at $K = 96$. The following paragraphs explain each regime.

Why the projection error falls so quickly. The decay of b_K follows the approximation theory behind Eq. (12). For this family, $\mathbb{E}\|F^{(4)}\|_{\mathcal{H}}^2 = \frac{1}{3} \sum_{q \in \{1,3,6\}} (2\pi q)^8$ in closed form, so the fourth order bound can be evaluated exactly. For every $K \geq 16$, the measured b_K lies below the bound, for example 0.014 against 0.062 at $K = 24$ and 3.4×10^{-4} against 2.9×10^{-3} at $K = 48$. The

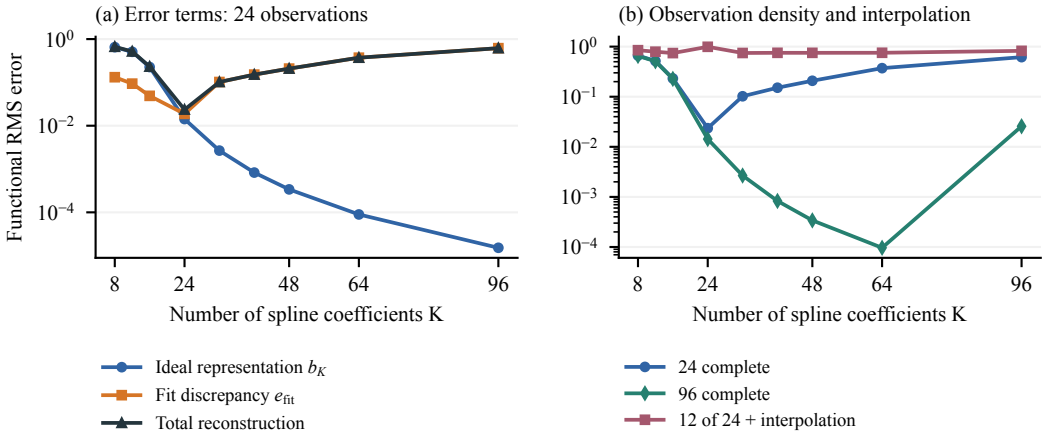


Figure 2: Functional RMS errors of the spline encoder on a synthetic family of smooth signals. (a) With 24 observations, the projection error b_K decreases monotonically with K, while the fitting error e_{fit} grows once K exceeds the number of observations, so the total reconstruction error is smallest at $K = 24$. (b) Total reconstruction error under three observation conditions: 96 observations move the optimum to larger K, whereas interpolating 12 of 24 observations leaves a large fitting error at every basis size.

Table 6: Selected values from Figure 2. The first four columns use 24 complete observations; "grid" is the RMS residual at the 24 observed times, while the remaining errors are measured over the whole interval $[0, 1]$.

	24 complete observations				96 complete	12 of 24 + interp.
K	b_K	e_{fit}	e_{recon}	grid	e_{recon}	e_{recon}
8	0.651	0.132	0.664	0.655	0.652	0.849
16	0.226	0.049	0.231	0.218	0.227	0.743
24	0.014	0.019	**0.024**	1.2×10^{-5}	0.014	0.993
32	0.003	0.102	0.102	1.7×10^{-6}	0.003	0.747
48	3.4×10^{-4}	0.208	0.208	1.9×10^{-6}	3.4×10^{-4}	0.752
64	8.9×10^{-5}	0.372	0.372	2.0×10^{-6}	$\mathbf{9.6 \times 10^{-5}}$	0.754
96	1.5×10^{-5}	0.617	0.617	2.0×10^{-6}	0.026	0.824

local slope of $\log b_K$ against $\log h$ decreases from 5.8 between $K = 16$ and $K = 24$ to 4.2 between $K = 64$ and $K = 96$, approaching the asymptotic fourth order rate. The onset of this decay is governed by resolution. The fastest component, $q = 6$, completes six periods on $[0, 1]$: $K = 16$ provides only about two knot intervals per period and leaves $b_K = 0.23$, whereas $K = 24$ provides 3.5 intervals per period and already reduces b_K to 0.014. For $K \leq 12$, fewer than 1.5 intervals fall in each period, the fastest component is essentially unresolved, and the bound becomes loose.

Where the fitting error comes from when $K < N$. Below $K = N$, the ridge fit differs from the ideal projection only because it uses a discrete inner product. Eq. (18) makes this explicit: the coefficient error is driven by $\Phi^\top e$, the correlation between the basis and the unresolved residual $F - P_K F$ sampled on the grid. In continuous L^2, this residual is orthogonal to every basis function; on a finite grid, part of it aliases onto the basis. At $K = 16$, the sampled residual has nearly the same RMS with 24 and with 96 observations (0.24 and 0.23), yet the fitting error falls from 0.049 to 0.021, because the discrete inner product approaches the continuous one as the grid becomes denser. This is why e_{fit} stays below b_K whenever $K < N$.

Why the fitting error grows when $K > N$. Once K exceeds N, the fitting error has a single, exactly identifiable source. The design matrix has rank 24, its null space has dimension $K - 24$, and the ridge encoder sets every null space component of the coefficients to zero (Appendix A.8).

Splitting the ideal coefficients $S_K^{-1} P_K F$ into the part that the samples observe and the part that they do not, we find that the unobserved part accounts for the entire fitting error at $K = 32$, 48, and 96. The discarded component is a spline that vanishes at every observation time but not in between, which is precisely the failure described after Proposition 4.2: at $K = 32$, the RMS error is 1.7×10^{-6} at the 24 sample times but 0.13 at the midpoints between them, and the midpoint error grows to 0.27 at $K = 48$ and 0.92 at $K = 96$. Regularization plays no role in this growth. Replacing $\lambda = 10^{-6}$ by $\lambda = 10^{-10}$, or by the minimum norm solution with $\lambda = 0$, leaves the reconstruction error at $K = 32$ unchanged to six digits (0.102106), because no choice of λ can recover directions that the samples never observe.

Weakly observed directions and observation density. Figure 2(b) shows that denser observations shift the optimum: with 96 samples, the fitting error stays negligible well beyond $K = 24$, and the total error follows b_K down to 9.6×10^{-5} at $K = 64$. The rise at $K = N = 96$ has the same cause as the growth above. The square 96×96 design is nearly singular, with its two smallest singular values equal to 1.5×10^{-7} against 1.0 for the largest. The corresponding coefficient directions are almost invisible to the samples: their decoded splines stay below 6×10^{-8} at every observation time but reach 0.45 between them. Since $\lambda = 10^{-6}$ greatly exceeds the squared singular values, the ridge suppresses these directions and incurs a fitting error of 0.026. The exact interpolant with $\lambda = 0$ lowers the error to 2.2×10^{-5} on exact data, but it would amplify any observation noise along these directions by a factor of about $1/\sigma_{\min} \approx 7 \times 10^6$. The square design with $K = N = 24$ is far better conditioned, with smallest singular value 0.016 and condition number 66, so there the ridge is harmless (0.02355 against 0.02360 at $\lambda = 0$). In every case, the fitting error is controlled by the coefficient directions that the observations determine weakly or not at all. This suggests a simple criterion that depends only on the grid and the basis and can be checked before training: choose K so that Φ has full column rank with its smallest singular value well above $\sqrt{\lambda}$.

Why interpolation dominates under missing data. With 12 of 24 samples retained, the linearly filled grid values already differ from the true signal by an RMS of 0.73 before any spline is fitted. This error concentrates in the fast components: it is 0.05 for $q = 1$, 0.33 for $q = 3$, and 0.64 for $q = 6$. The largest gap between retained samples spans four grid steps, slightly more than one full period of the $q = 6$ component, so the information needed to reconstruct that component is absent from the input. The encoder faithfully fits this corrupted input, and the total error stays between 0.74 and 0.99 for every basis size. Since no choice of basis can restore missing information, the remedy must lie in the training objective. This is the role of the masked loss in Eq. (8), which supervises the decoded velocity only at observed entries, so that interpolated values shape the coefficient path without serving as regression targets.

Implications for GalerkinFlow. Three practical conclusions follow. First, for smooth signals the projection floor is reached quickly: a few knot intervals per period of the fastest relevant component already bring b_K to the percent level, at the fourth order rate of Eq. (12). Second, the fitting term is governed by how well the samples determine the coefficients, so basis size should be chosen relative to the observation grid rather than in isolation. Third, the long-horizon setting is the most favorable regime for the encoder, since $K = 32$ lies far below $N = 728$ and $N = 791$. For signals of comparable smoothness, the fitting error at $K = 32$ is already 1.9×10^{-4} with 96 observations, so on long windows the budget is governed by the projection term, which the choice of K controls, and the transport term, which training reduces. On the regular benchmarks with $K = 32 > N = 24$, the between-sample deviation does not affect grid-based metrics such as Context-FID, which evaluate generated windows only at the sampling times; this is consistent with $K = 24$ and $K = 32$ performing equally well in the Stocks sweep (Table 9). It matters, however, whenever the generated trajectories are used as continuous functions, in which case $K \leq N$ is preferable.

Relation to the learned models. The Stocks basis-size sweep follows the same qualitative pattern. Context-FID is 0.012 at $K = 16$, 0.006 at $K = 24$ and $K = 32$, and rises to 0.019 and 0.030 at $K = 40$ and $K = 48$ (Table 9): too few coefficients leave a projection floor, while too many exceed what 24 samples can identify. The second sweep in Table 18 is flatter at large K, and continued training at $K = 48$ roughly halves its Context-FID, so part of the large-K degradation reflects optimization. The synthetic diagnostic isolates the representation and fitting terms, which no amount of training can change.

B.3 Reading the ablations through the error budget

Combining Proposition 4.2 with Proposition A.1 gives the budget in Eq. (21): the projection term b_K, the fitting term e_{fit}, the statistical term e_{stat}, and the flow term e_{flow}, which in turn collects velocity-learning, endpoint, and integration errors. Each ablation in Section 5.4 holds a known subset of these terms fixed, which indicates where its effect must arise.

- **Continued training** (Table 18) keeps the basis, encoder, and training data unchanged, so b_K, e_{fit}, and e_{stat} are fixed and only the velocity-learning part of e_{flow} can change. Context-FID falls from 0.0074 to 0.0040, and the coefficient sliced-W_2 distance falls from 0.0226 to 0.0165, consistent with a smaller transport error.
- **Solver and sampling budget** (Tables 13 and 14) change only the integration error δ_{num}. At 40 function evaluations, Euler yields a Context-FID of 0.0185 against 0.0031 for midpoint RK2, whereas midpoint RK2 stays between 0.0032 and 0.0033 from 20 to 500 evaluations. Beyond a moderate budget, numerical integration is therefore not the limiting term.
- **Ridge regularization** (Tables 16 and 17) changes the fitting term only slightly for $\lambda \in [10^{-8}, 10^{-4}]$, and Context-FID correspondingly stays at about 0.0032.
- **Basis family** (Table 4) compares bases with nearly equal reconstruction errors (0.0193 for splines and 0.0196 for Legendre), so the representation and fitting terms are comparable. The large Context-FID gap (0.009 against 0.032) must therefore arise in the transport term: with the same network and training budget, the spline coefficient law is easier to learn.
- **Raw grid versus spline coordinates** (Table 5) compares two models that receive essentially the same information, so again the difference lies in how easily the flow transports the prior to the data in each coordinate system.

These correspondences are qualitative. The budget bounds the functional Wasserstein distance, whereas the benchmark metrics are computed on the sampling grid; the mapping from each ablation to a term of Eq. (21) follows from which components the ablation holds fixed, not from a direct measurement of every term.

C Experimental specification

C.1 Model and training settings

Table 7: GalerkinFlow experimental settings used across all reported results.

Setting	Value
Regular windows; default basis	$N = 24$; $K = 32$
Small velocity network	4 blocks; width 96; 4 attention heads
Long-horizon network	8 blocks; width 384; 8 attention heads
Time features (small model)	96 values: sine and cosine at 48 frequencies
Optimizer	AdamW; learning rate 5×10^{-4}
Training	600 epochs; batch size 64
EMA decay	0.99; 0.98 in irregular experiments
Prior and ridge	$\alpha = 1$; ridge $\lambda = 10^{-6}$
Endpoint	$1 - \sigma$, with $\sigma = 0.01$
Main sampling	500 midpoint steps (1,000 NFE)
Budget sweep	20, 50, 100, 200, 500 NFE; 50 is the default within this sweep
Equal-NFE solver comparison	40 NFE: 40 Euler, 20 Heun/midpoint, or 10 RK4 steps
Compute	One NVIDIA A100 GPU
TS2Vec evaluator	320-dimensional full-series embedding; batch size 8; learning rate 10^{-3}

C.2 Training and inference

For masked inputs, we fill each channel separately: we interpolate linearly between observed values and repeat the nearest observed value beyond the first and last observations. A channel with one

observation is filled with that constant; a channel with no observations is filled with zeros in the preprocessed coordinates.

For a fixed grid, we construct Φ and factorize $\Phi^\top\Phi + \lambda I$. Masked windows are interpolated onto the common grid before solving for their coefficients. For each minibatch, we draw independent prior tensors A_0 of shape $B \times K \times D$ and flow times τ, form $A_\tau = (1-\tau)A_0 + \tau A_1$ and $U = A_1 - A_0$, predict $v_\theta(A_\tau, \tau)$, and update the parameters using Eq. (5) or, for masked training, Eq. (8). The encoder and basis remain fixed throughout training.

For generation, we draw fresh prior coefficients, apply Eq. (7) for the chosen number of steps, and decode ΦA^S on the prescribed grid. All reported evaluations use the same sampling resolution as training.

C.3 Metrics and uncertainty

Context-FID was introduced for time-series generation by Jeha et al. (2022). Our evaluator uses TS2Vec (Yue et al., 2022) with equal-sized real and generated sets, truncating the generated set when necessary. We fit five encoders afresh on real data; the interval half-width is $t_{0.975,4}s/\sqrt{5}$. An uncertainty shown as $\pm.000$ is below 0.0005 after rounding, not exactly zero.

The discriminative score is the absolute difference between a real-versus-generated classifier's accuracy and 0.5, rather than its raw error (Yoon et al., 2019). The predictive score is the error on real test data of a forecaster trained on generated data. The long-horizon classification score is not a probability, and some values exceed one under its scoring convention.

C.4 Baselines and data sources

The regular and masked benchmarks include TimeGAN (Yoon et al., 2019), COT-GAN (Xu et al., 2020), TimeVAE (Desai et al., 2021), RCGAN (Esteban et al., 2017), C-RNN-GAN (Mogren, 2016), GT-GAN (Jeon et al., 2022), Koopman VAE (Naiman et al., 2024b), DiffWave (Kong et al., 2021), DiffTime (Coletta et al., 2023), and Diffusion-TS (Yuan & Qiao, 2024). Long-horizon comparisons use FRED-MD and NN5 Daily from the Monash archive (Godahewa et al., 2021) and compare against ImagenTime (Naiman et al., 2024a). Spectral Mean Flow (Kim et al., 2025) is evaluated in all main settings, and RNN-AR denotes an autoregressive recurrent baseline.

The functional and continuous-time baselines are FFM-OT, TFM-U-SDE, and GM-Jump (Kerrigan et al., 2024; Zhang et al., 2024; Holderrieth et al., 2025), which we run with our shared preprocessing and evaluation protocol.

D Additional benchmark details and ablations

Tables 1, 2, and 3 give the regular, long-horizon, and irregular benchmark comparisons. Here we report the masked-loss and basis-size ablations, followed by training and sampling times.

Table 8: Stocks masked-loss ablation: discriminative score (↓). The full and no-mask variants use the settings of Table 3. Spectral Mean Flow is included for comparison.

Method	0% drop	30% drop	50% drop	70% drop
Irregular to regular				
GalerkinFlow	0.006±.002	0.019±.016	0.019±.014	0.016±.012
GalerkinFlow, no mask	0.006±.002	0.035±.012	0.045±.013	0.042±.016
Spectral Mean Flow	0.009±.008	0.020±.011	0.019±.008	0.015±.007
Irregular to irregular				
GalerkinFlow	0.006±.002	0.010±.004	0.026±.013	0.033±.047
GalerkinFlow, no mask	0.006±.002	0.018±.008	0.019±.004	0.035±.037
Spectral Mean Flow	0.009±.008	0.049±.017	0.044±.017	0.138±.137

Table 9: Stocks basis-size sweep (↓). These runs are separate from the second sweep and the continued-training run in Table 18.

K	Context-FID ↓	Correlational ↓	Discriminative ↓	Predictive ↓
16	0.012 ± 0.003	0.011 ± 0.006	0.010 ± 0.006	0.037 ± 0.000
24	0.006 ± 0.002	0.006 ± 0.007	0.018 ± 0.009	0.037 ± 0.000
32	0.006 ± 0.002	0.008 ± 0.006	0.014 ± 0.014	0.037 ± 0.000
40	0.019 ± 0.003	0.016 ± 0.007	0.016 ± 0.014	0.037 ± 0.000
48	0.030 ± 0.009	0.013 ± 0.009	0.016 ± 0.011	0.037 ± 0.000

Table 10: Regular-benchmark training time, in minutes.

Method	Sines	Stocks	ETTh	MuJoCo	Energy	fMRI
GalerkinFlow	5min	6min	8min	13min	11min	8min
Spectral Mean Flow	18min	15min	27min	60min	44min	35min
Diffusion-TS	16min	15min	30min	25min	60min	48min

Table 11: Regular-benchmark sampling time, in seconds per 1,000 generated samples.

Method	Sines	Stocks	ETTh	MuJoCo	Energy	fMRI
GalerkinFlow	19sec	11sec	10sec	29sec	57sec	29sec
Spectral Mean Flow	30sec	30sec	29sec	109sec	95sec	109sec
Diffusion-TS	66sec	66sec	67sec	133sec	202sec	262sec

E Additional Experimental Results

E.1 Coefficient paths and numerical integration

The path comparison uses $K = 24$ and $\sigma = 0.01$ and replaces our straight interpolation path (Eq. (4)) with a variance-preserving path with trigonometric coefficients (VP trigonometric) or a variance-exploding path with an exponential noise schedule (VE exponential). The solver comparison uses a single checkpoint and matches the number of function evaluations (NFE); equal NFE does not imply equal wall-clock time.

Table 12: Stocks: comparison of coefficient interpolation paths (↓).

Path	FID	Corr.	Disc.	Pred.
VP trigonometric	0.0139	0.006	0.014	0.037
VE exponential	0.0218	0.007	0.012	0.037
Ours	0.006	0.006	0.018	0.037

Table 13: Stocks: solvers compared at 40 function evaluations (↓).

Solver	FID	Corr.	Disc.	Pred.
Euler	0.018505	0.009879	0.020191	0.036737
Heun RK2	0.004066	0.005980	0.022237	0.036729
Midpoint RK2	0.003131	0.005353	0.012960	0.036717
RK4	0.003440	0.005674	0.013370	0.036724

Table 14: Stocks: effect of the midpoint sampling budget. Time is the wall-clock sampling time in seconds; lower scores are better.

NFE	Time (s)	FID	Corr.	Disc.	Pred.
20	10.403	0.003156	0.005745	0.015825	0.036707
50 (default)	26.585	0.003192	0.005307	0.010368	0.036720
100	48.227	0.003296	0.005257	0.012142	0.036723
200	96.149	0.003327	0.005205	0.010641	0.036726
500	237.291	0.003337	0.005197	0.013370	0.036726

E.2 Interpolation, prior scale, and ridge regularization

The interpolation and loss controls use $K = 32$ in the irregular-to-regular setting. Comparing the full model with a variant trained without the masked loss isolates the contribution of the masked objective. The ridge-fit diagnostics use the same Stocks grid as the main experiments; reconstruction RMSE measures residual on the observed grid, not over the full continuous interval.

Table 15: Stocks: interpolation and loss controls (↓).

Configuration	Discriminative score
Full model	0.0208 ± 0.0208
Without interpolation	0.1778 ± 0.0068
Without masked loss	0.0271 ± 0.0341

Table 16: Stocks: sensitivity to prior scale and ridge regularization (↓).

Setting	FID	Corr.	Disc.	Pred.
Prior std. 0.5	0.003263	0.004643	0.008186	0.036743
Prior std. 1.0; ridge 10^{-6}	0.003192	0.005307	0.010368	0.036720
Prior std. 2.0	0.004570	0.004402	0.012415	0.036707
Ridge 10^{-8}	0.003236	0.005334	0.010232	0.036719
Ridge 10^{-4}	0.003209	0.005608	0.011323	0.036722

Table 17: Stocks: ridge-fit diagnostics.

Ridge λ	Target coeff. RMS	Reconstruction RMSE
10^{-8}	0.634886	4.78×10^{-8}
10^{-6}	0.634885	8.45×10^{-7}
10^{-4}	0.634802	8.42×10^{-5}

E.3 Basis dimension and continued optimization

The coefficient sliced-distance proxy is defined as

$$\widehat{\mathrm{SW}}_2 = \left[\frac{1}{128} \sum_{m=1}^{128} W_2^2\big((\theta_m^\top)_{\#}\widehat{\eta}, (\theta_m^\top)_{\#}\widehat{\nu}\big) \right]^{1/2},$$

using 128 fixed random unit directions in vectorized coefficient space.

Table 18: Stocks: basis dimension and continued training. Dashes indicate quantities not computed for this run.

K	Epochs	Parameters	FID	Corr.	Disc.	Pred.	Coeff. $\widehat{\mathrm{SW}}_2$
16	600	517,414	0.013011	0.011033	0.017599	0.036781	0.026373
24	600	518,182	0.007008	0.003784	0.013506	0.036793	0.078485
32	600	518,950	0.003192	0.005307	0.010368	0.036720	—
40	600	519,718	0.003645	0.006064	0.012142	0.036714	0.023987
48	600	520,486	0.007400	0.008844	0.009004	0.036716	0.022606
48	1200	520,486	0.003982	—	—	—	0.016521