QFLOW: DETERMINISTIC QUANTILE TRANSPORT FOR PROBABILISTIC TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent diffusion forecasters have shown that long-horizon prediction can be reduced to a linear-time recursion over physical time, with noise and scale schedules estimated directly from data. However, these deterministic forecasts do not by themselves provide a principled way to turn the state evolution into a predictive distribution, while resorting to Monte Carlo sampling would reintroduce much of the cost that efficient diffusion forecasting seeks to avoid. We present QFLOW, which extends such forecasters with deterministic quantile transport for probabilistic time series forecasting. We use the Monge–Ampère equation to relate probability density to the local stretching and compression of a vector quantile map. A diagonal approximation of this relation then yields a one-step recursion that transports an entire quantile curve under the same probability-flow field as the underlying state update. At inference, QFLOW advances a dense transport grid, centers the resulting curve on a separately estimated conditional mean, and computes probabilistic forecast metrics directly from its quantiles, without sample ensembles. Across nine long-horizon forecasting datasets, QFLOW ranks among the top three recent diffusion forecasters in 33 of 36 dataset–metric comparisons. It achieves this performance using only a temporal linear layer for the conditional-mean head and either a temporal linear layer or a single cross-attention block for the denoiser.

1 INTRODUCTION

Probabilistic forecasts support decisions that depend not only on the expected future but also on the range of plausible outcomes. In energy, epidemiology, traffic, and finance, calibrated uncertainty can be as important as point accuracy (Gneiting & Katzfuss, 2014). Diffusion models are attractive for this setting because they are generative by construction, but their forecasting procedures often inherit the cost profile of image diffusion: repeated denoising, repeated sampling, and evaluation through ensembles (Rasul et al., 2021; Tashiro et al., 2021; Li et al., 2024; Ye et al., 2025).

In a standard diffusion forecaster, clean future windows are corrupted at several noise levels during training, and a denoiser learns to recover the clean signal from a noisy window. A noise schedule sets the amount of corruption and a scale schedule tracks changes in signal magnitude. At inference, conventional methods reverse this corruption repeatedly and draw many forecast samples to represent uncertainty. If S denoising iterations are used for each of H future steps, the cost scales as $\mathcal{O}(SH)$.

Recent work changes this procedure by identifying diffusion time with the physical time of the series and estimating the noise and scale schedules from the observed window (Solano-Carrillo et al., 2026). One backward-denoising Euler step then advances one physical time step. Repeating this update H times produces a deterministic probability-flow trajectory with $\mathcal{O}(H)$ sampling complexity, without a separate S-step diffusion chain at each time point. That trajectory alone does not explain how the full predictive distribution should change or where its conditional mean lies.

QFLOW supplies this missing distributional update. In QFLOW, the transport map moves each predictive quantile from the current physical time step to the next. Starting from the vector-quantile view of optimal transport (Carlier et al., 2016), we use the Monge–Ampère equation to relate the predictive probability density to local stretching of the quantile map. A diagonal approximation of the quantile map's Jacobian turns this multivariate relation into a coordinate-wise update that transports the centered quantile shape through the probability-flow dynamics. A separate lightweight head

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.

estimates each horizon step's conditional mean given the observed context window, after which a translation combines the learned location and transported shape while keeping the quantiles in order.

The empirical question is whether this quantile update remains competitive with modern diffusion forecasters when both of its learned components are intentionally small: one temporal linear layer for the conditional-mean head, where recent methods place a Transformer forecaster (Li et al., 2024; Ye et al., 2025), and one temporal linear layer or cross-attention block for the denoiser. This separates the effect of the transport from gains due to network capacity, and keeps the resource footprint small for online forecasting. Across nine standard datasets, QFLOW's quantile forecasts remain competitive with recent diffusion forecasters, and non-diffusion baselines trace most of the remaining error to the transported shape.

Our contributions are:

- We derive a deterministic quantile update from the vector-quantile Monge–Ampère relation by a diagonal approximation. It carries an entire predictive curve from one physical time step to the next under the same probability-flow dynamics that advance the forecast itself.
- We propose a location–shape construction that splits the predictive distribution into a learned conditional-mean location and a transported centered shape, ordered by construction and computed in one deterministic pass per step rather than from a sampled ensemble.
- We prove an upper bound on the Wasserstein distance between the transported curve and the true conditional distribution, separating the error of the transport approximation from that of the learned denoiser, and introduce a training regularizer that controls the coordinate mixing the diagonal approximation neglects.

2 Related Work

Diffusion forecasting. Existing methods differ in how they incorporate temporal structure and observed history. TimeGrad (Rasul et al., 2021) conditions diffusion on a recurrent representation of the past to model the joint distribution across variables at each forecast step. CSDI (Tashiro et al., 2021) learns conditional scores for probabilistic imputation, training on masked observations with attention across time and variables. Forecasting follows by treating future values as missing entries. TimeDiff (Shen & Kwok, 2023) denoises future windows non-autoregressively, conditioning on autoregressive predictions and, during training, a mixture of encoded history and future targets.

Other methods modify the denoiser or diffusion dynamics. Diffusion-TS (Yuan & Qiao, 2024) combines trend–seasonality decomposition in a Transformer with sample reconstruction and a Fourier loss, and conditional sampling extends this general generator to forecasting and imputation. TMDM (Li et al., 2024) uses Transformer predictions of the conditional mean to modulate both forward and reverse diffusion. NsDiff (Ye et al., 2025) estimates a conditional mean and variance to adapt the diffusion endpoint distribution, and adjusts its noise schedule to represent changing uncertainty. QFLOW combines a small location head with direct quantile transport through a physical-time probability flow.

Physical-time probability flow. Diffusion models can be sampled deterministically with the probability-flow ordinary differential equation (ODE) (Song et al., 2021; Karras et al., 2022). Identifying its diffusion time with physical time allows one Euler step to advance a forecast, using scale and noise schedules estimated from the observed series (Solano-Carrillo et al., 2026). We use this formulation to derive a distributional update: an ordered quantile curve propagated alongside the state, with a modular location estimate and an analysis of the resulting marginal error.

Quantiles and optimal transport. Quantile regression (Koenker & Bassett Jr, 1978; Wen et al., 2017) fits specified probability levels through the pinball loss. Spline quantile models (Gasthaus et al., 2019) instead parameterize an entire monotone curve. Like spline models, QFLOW represents a curve that can be queried at different levels without retraining. Its distinguishing feature is how that curve evolves: the update follows a denoiser-driven probability flow. To derive it, we use vector quantiles, which connect multivariate distributions to monotone maps through optimal transport (Carlier et al., 2016). A diagonal approximation of this map's Jacobian then makes the update tractable, yielding scalar marginal curves while omitting their cross-feature dependence.

3 METHOD

3.1 FORECASTING THROUGH PROBABILITY FLOW

Setting and notation. The observed context $\mathcal{C} \in \mathbb{R}^{T\times d}$ contains T past time steps of d variables. We predict a marginal distribution for each variable at each of the next H steps, whose observations form $\mathbf{X}_{1:H} \in \mathbb{R}^{H\times d}$. The forecast index $t = 0, \dots, H$ starts at the context boundary. A random scalar state is denoted by X_t, a deterministic trajectory by x_t, and a transported quantile at probability level $u \in (0, 1)$ by $q_t(u)$. Bold symbols denote vector or matrix quantities.

The denoiser processes several time positions together. Its rolling state $\mathbf{x}^{[t]}$ starts from the context and advances with the recursively predicted values. Each position and variable carries a state value and a quantile curve; the scalar equations below describe one such entry, although the denoiser can use the whole window. We reserve $\mathbf{w}$ for a clean window drawn from the training data, rather than the context of a particular forecast. Appendix B.1 gives the full window indices.

Schedules and density. We first describe the state evolution, whose joint density given the context is $p_t(\mathbf{x} \mid \mathcal{C})$. A positive *scale schedule* s_t controls multiplicative changes in the signal, while a *noise covariance schedule* $\mathbf{\Sigma}_t$ describes accumulated noise in the scaled state $\mathbf{X}_t/s_t$. The underlying linear-drift diffusion has transition $\mathbf{X}_t \mid \mathbf{X}_0 = \mathbf{x}_0 \sim \mathcal{N}(s_t\mathbf{x}_0, s_t^2\mathbf{\Sigma}_t)$ (Solano-Carrillo et al., 2026). QFLOW approximates this covariance by its diagonal and writes Σ_t for the variance of the entry under consideration. In forecasting, local level and variance estimates supply a scale and noise variance for each rolling-window entry. Recomputing them as the window advances adapts the dynamics (Appendices A.1 and B.1).

The deterministic probability-flow ODE $\dot{\mathbf{x}}_t = \mathbf{v}_t(\mathbf{x}_t)$, with velocity field $\mathbf{v}_t$, realizes the same densities: evolving $\mathbf{x}_{t_{\text{start}}} \sim p_{t_{\text{start}}}(\cdot \mid \mathcal{C})$ to t_{end} yields $\mathbf{x}_{t_{\text{end}}} \sim p_{t_{\text{end}}}(\cdot \mid \mathcal{C})$ (Song et al., 2021; Karras et al., 2022). We call a solution path a *characteristic*, and the forecast state x_t follows one characteristic. A single path need not track the distribution's mean, so QFLOW centers its recursive curve on the characteristic and then refines the output with a separate location estimate.

To evaluate this evolution, we express its velocity through a denoiser $D_\theta(x/s_t, \Sigma_t)$, which estimates the initial signal from the scaled noisy state. In scalar formulas, D_θ denotes the output entry for one coordinate, with its other window inputs and conditioning arguments suppressed. The increments $\Delta \log s_t$ and $\Delta \log \Sigma_t$ are log-ratios between adjacent physical positions in the current window, as defined in Appendix B.1. Following the velocity $\mathbf{v}_t$ in its denoising direction, the forecast advances by the Euler update (Solano-Carrillo et al., 2026)

$$x_{t+1} = x_t - (\Delta \log s_t)x_t + \tfrac{1}{2}s_t(\Delta \log \Sigma_t)\left[D_\theta(x_t/s_t, \Sigma_t) - x_t/s_t\right]. \tag{1}$$

One iteration advances the forecast by one physical step.

3.2 TRANSPORTING THE QUANTILE SHAPE

The state update supplies the motion of one characteristic, x_t. To describe uncertainty, we also track how a whole distribution stretches and compresses. Flattening the rolling state gives n coordinates: one for each variable at each window position. A vector quantile map $\mathbf{q}_t(\mathbf{u})$ carries a uniform reference on $(0, 1)^n$ to p_t. Its Jacobian $\mathbf{A}_t = \nabla_{\mathbf{u}}\mathbf{q}_t$ records how changing each reference coordinate moves each state coordinate. The Monge–Ampère change-of-variables identity relates its local volume expansion to density:

$$p_t(\mathbf{q}_t(\mathbf{u}) \mid \mathcal{C}) \det \mathbf{A}_t = 1.$$

In one dimension, with $q_t' = \partial_u q_t$, the identity reads $p_t(q_t(u) \mid \mathcal{C})q_t'(u) = 1$. For example, the interval from the 50th to the 51st percentile always contains 1% of the mass. If its endpoints move farther apart, that same mass occupies a wider interval, so its average density falls. Locally, the quantiles at levels u and $u + du$ lie $q_t'(u)\, du$ apart, so q_t' is their spacing per unit probability, and the mass between them is $p_t(q_t(u) \mid \mathcal{C})q_t'(u)\, du = du$.

To obtain an update for this spacing, we move the map with the probability-flow velocity: $\partial_t\mathbf{q}_t = \mathbf{v}_t(\mathbf{q}_t)$. Differentiation gives $\dot{\mathbf{A}}_t = [\nabla_{\mathbf{x}}\mathbf{v}_t(\mathbf{x})]_{\mathbf{x}=\mathbf{q}_t}\, \mathbf{A}_t$. QFLOW retains only the diagonal entries of $\mathbf{A}_t$: each scalar curve describes its own coordinate's spread, without tracking how a reference coordinate displaces other coordinates. The determinant becomes a product of scalar spacings. In

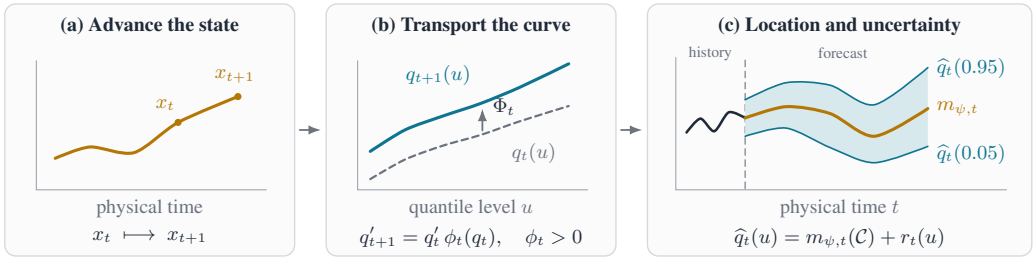

Figure 1: **Schematic of the QFLOW forecast.** (a) The characteristic advances one step. (b) The map Φ_t transports an ordered quantile curve. (c) Its centered shape r_t combines with the learned location $m_{\psi,t}(\mathcal{C})$ at each future step. The gold line shows the location forecast; the shaded band spans $\widehat{q}_t(0.05)$ to $\widehat{q}_t(0.95)$. Band width and asymmetry come from the transported shape.

scalar form, x is the state coordinate and x_t a characteristic's position. Differentiating $v_t(x)$ first, then evaluating at $x = q_t(u)$, gives

$$\partial_t q'_t(u) = [\partial_x v_t(x)]_{x=q_t(u)}\, q'_t(u), \qquad \partial_t \log q'_t(u) = [\partial_x v_t(x)]_{x=q_t(u)}\,.$$

Thus a velocity that increases with position pulls neighboring quantiles apart. This reduction is exact for separable dynamics with a diagonal initial map. With a coupled denoiser, it omits the off-diagonal products in $\dot{\mathbf{A}}_t$. The network still uses other variables and time positions, but uncertainty propagation retains only its own-coordinate derivative (Appendix A.3).

Expressing $\partial_x v_t$ through the denoiser and stepping in the same direction as Eq. (1) gives a first-order update of the log-spacing $\log q'_t$:

$$\begin{aligned} q'_{t+1}(u) &= q'_t(u)\,\phi_t(q_t(u)), \\ \phi_t(x) &= \exp\!\big[-\Delta \log s_t - \tfrac{1}{2}\Delta \log \Sigma_t + \tfrac{1}{2} s_t(\Delta \log \Sigma_t)\,\partial_x D_\theta(x/s_t, \Sigma_t)\big]\,. \end{aligned} \tag{2}$$

Here ∂_x differentiates the denoiser with respect to the physical input, including the factor $1/s_t$. Values $\phi_t > 1$ expand local spacing, while $0 < \phi_t < 1$ contract it. Exponentiation ensures positivity and hence preserves order under the exact integral map.

We obtain the curve by integrating the positive coefficient from any fixed reference point x_{ref}:

$$\Phi_t(x) = \int_{x_{\mathrm{ref}}}^{x} \phi_t(w)\,dw, \qquad q_{t+1}(u) = x_{t+1} + \Phi_t(q_t(u)) - \int_0^1 \Phi_t(q_t(v))\,dv. \tag{3}$$

Differentiating $q_{t+1}(u)$ in u gives Eq. (2) by the chain rule, and subtracting the mapped curve's mean centers it on x_{t+1}. The reference point cancels between the two integrals. We initialize $q_0(u) = m_0 + \sigma_0 \Phi_{\mathcal{N}}^{-1}(u)$ using the local context mean m_0, standard deviation σ_0, and standard normal quantile function $\Phi_{\mathcal{N}}^{-1}$. Although this initial curve is Gaussian, it can become asymmetric and change its tails as ϕ_t varies along it.

Location refinement. Equation (3) already defines a forecast curve centered on the characteristic x_t. To refine its location, we fit a temporal linear head $m_{\psi,1:H}(\mathcal{C})$ to future observations by minimizing the mean squared error (MSE). It applies the same T-to-H weight matrix and bias to each variable's history, without mixing variables. The main benchmark uses the learned-location forecast

$$\widehat{q}_t(u) = m_{\psi,t}(\mathcal{C}) + r_t(u), \qquad r_t(u) = q_t(u) - \int_0^1 q_t(v)\,dv. \tag{4}$$

This translation refines the output while the recursive state follows Eq. (1). Moving all levels equally preserves their order and interval widths, leaving the uncertainty shape determined by transport. For example, $[\widehat{q}_t(0.05), \widehat{q}_t(0.95)]$ gives a central 90% interval; Figure 1 shows these intervals across time. Using x_t as the location gives the characteristic-based forecast.

3.3 TRAINING AND INFERENCE

Training fits the denoiser while keeping its input derivatives stable. For each vectorized clean window $\mathbf{w}$, we estimate scales $\mathbf{s}$ and diagonal noise covariance $\mathbf{\Sigma}$, then add Gaussian noise directly: $\widetilde{\mathbf{w}} = \mathbf{w} + \mathbf{\Sigma}^{1/2}\boldsymbol{\varepsilon}$, with $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We minimize the expectation over windows, noise, and probes using minibatch averages of the sampled loss

$$\mathcal{L}_{\text{train}}(\theta; \mathbf{w}, \widetilde{\mathbf{w}}, \mathbf{s}, \mathbf{\Sigma}, \boldsymbol{\xi}) = \left\| \boldsymbol{\lambda}_{\mathbf{\Sigma}}^{1/2} \odot \left(D_\theta(\widetilde{\mathbf{w}}, \mathbf{\Sigma}) - \mathbf{w} \right) \right\|_2^2 + \lambda_J \left\| \mathbf{J}_\theta(\mathbf{x})^\top \boldsymbol{\xi} \right\|_2^2, \tag{5}$$

where $\mathbf{x} = \mathbf{s} \odot \widetilde{\mathbf{w}}$, $\mathbf{J}_\theta = \partial_{\mathbf{x}} D_\theta(\mathbf{x}/\mathbf{s}, \mathbf{\Sigma})$, and $\boldsymbol{\xi}$ is an independent probe with zero mean and covariance $\mathbf{I}$. Products $\odot$ and ratios such as $\mathbf{x}/\mathbf{s}$ are coordinate-wise. The reconstruction weights $\boldsymbol{\lambda}_{\mathbf{\Sigma}}$ follow the denoiser's preconditioning (Karras et al., 2022; Solano-Carrillo et al., 2026). The second term estimates $\|\mathbf{J}_\theta\|_F^2$, discouraging both large derivatives and coordinate mixing.

At inference, the required derivative for coordinate i is $(\mathbf{J}_\theta)_{ii}$. We estimate it with a random probe $\boldsymbol{\xi}$ as $[\boldsymbol{\xi} \odot \mathbf{J}_\theta^\top \boldsymbol{\xi}]_i$, evaluated at each quadrature node of Φ_t, and use this estimate in Eq. (2). This requires a vector–Jacobian product rather than the full matrix. Gaussian training probes and Rademacher inference probes (independent entries ± 1) both satisfy $\mathbb{E}[\boldsymbol{\xi}\boldsymbol{\xi}^\top] = \mathbf{I}$, so both estimates are unbiased. Rademacher probes have lower variance for the diagonal of $\mathbf{J}_\theta$ (Appendix A.3). Inference is deterministic for a fixed probe sequence.

Numerical realization. Numerical quadrature approximates an integral by a weighted sum of function values at chosen nodes. For the signal integral $\Phi_t(x)$ in Eq. (3), N_{GL} Gauss–Legendre nodes between x_{ref} and x give an accurate approximation with few denoiser evaluations. The centering integral instead runs over probability levels, where the curve is already stored at K uniformly spaced values $u_k \in [0.01, 0.99]$. Normalized trapezoidal weights ν_k, with $\sum_k \nu_k = 1$, reuse these values: $\sum_k \nu_k \Phi_t(q_t(u_k))$ approximates the mean in Eq. (3), and $\sum_k \nu_k q_t(u_k)$ the mean in Eq. (4). These averages omit the extreme tails (Appendix B.2). The learned networks do not depend on K, so the number and positions of output quantiles can change without retraining.

Derivative and quadrature errors can accumulate over the horizon, so we rescale the transported spread toward the rolling-window variance, with strength selected on validation data (Eq. (22)). Clipping limits extreme extrapolation, and sorting repairs numerical crossings. We center the final shape *after* these corrections, since clipping can shift the curve's mean. Algorithms 1 and 2 collect the procedure, and Appendix B.2 specifies the settings.

Algorithm 1 QFLOW training

Require: weights $\boldsymbol{\lambda}_{\mathbf{\Sigma}}, \lambda_J$; QICE cap τ_Q
for each minibatch of clean windows $\mathbf{w}$ **do**
  estimate $(\mathbf{s}, \mathbf{\Sigma})$ from $\mathbf{w}$
  $\boldsymbol{\varepsilon}, \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ independently
  $\widetilde{\mathbf{w}} \leftarrow \mathbf{w} + \mathbf{\Sigma}^{1/2}\boldsymbol{\varepsilon}$
  evaluate $\mathcal{L}_{\text{train}}(\theta; \mathbf{w}, \widetilde{\mathbf{w}}, \mathbf{s}, \mathbf{\Sigma}, \boldsymbol{\xi})$ by (5)
  $\theta \leftarrow \theta - \eta_\theta \nabla_\theta \mathcal{L}_{\text{train}}$
end for
select denoiser θ with lowest validation CRPS subject to QICE $\leq \tau_Q$; if none qualifies, select the denoiser with lowest QICE
for each minibatch of pairs $(\mathcal{C}, \mathbf{X}_{1:H})$ **do**
  $\psi \leftarrow \psi - \eta_\psi \nabla_\psi \|m_\psi(\mathcal{C}) - \mathbf{X}_{1:H}\|_2^2$
end for
select mean head ψ with lowest validation MSE

Algorithm 2 QFLOW inference

Require: context $\mathcal{C}$; horizon H; levels u_k, weights ν_k
initialize $\mathbf{x}^{[0]}, m_0, \sigma_0$ from $\mathcal{C}$
$q_0(u_k) \leftarrow m_0 + \sigma_0 \Phi_{\mathcal{N}}^{-1}(u_k)$
compute $m_{\psi,1:H}(\mathcal{C})$
for $t = 0, \ldots, H-1$ **do**
  estimate $(\mathbf{s}, \mathbf{\Sigma})$ from $\mathbf{x}^{[t]}$
  compute x_{t+1} by (1) for each window entry
  sample Rademacher $\boldsymbol{\xi}$ at each quadrature node
  use $[\boldsymbol{\xi} \odot \mathbf{J}_\theta^\top \boldsymbol{\xi}]_i$ for $\partial_x D_\theta$ in (2)
  integrate ϕ_t to obtain Φ_t by (3)
  $q_{t+1}(u_k) \leftarrow x_{t+1} + \Phi_t(q_t(u_k)) - \sum_j \nu_j \Phi_t(q_t(u_j))$
  apply variance rescaling and safeguards
  $r_{t+1}(u_k) \leftarrow q_{t+1}(u_k) - \sum_j \nu_j q_{t+1}(u_j)$
  emit $\widehat{q}_{t+1}(u_k) \leftarrow m_{\psi,t+1}(\mathcal{C}) + r_{t+1}(u_k)$
  advance $\mathbf{x}^{[t]}$ and its quantile curves
end for

Evaluation and cost. We evaluate forecasts with two standard scores. The continuous ranked probability score (CRPS) integrates pinball loss over the forecast curve. It penalizes misplaced or unnecessarily diffuse probability mass relative to the observed outcome. Quantile interval coverage error

(QICE) compares observed decile-bin frequencies with their nominal probabilities (Appendix B.3). In terms of cost, inference uses two denoiser calls per step: one for the state and one batch of KN_{GL} integration nodes, plus a vector–Jacobian product per probe. Batching retains $(1+KN_{\mathrm{GL}})H$ window evaluations and their memory cost; computation is linear in H for fixed grids and network size.

3.4 Bound on the distance to the true distribution

Preserving order makes a curve a valid quantile function; forecast accuracy asks how close that distribution is to the observations. For one variable, the true conditional cumulative distribution function (CDF) is $F_t^\star(x) = \Pr(X_t \le x \mid \mathcal{C})$. Its quantile $q_t^\star(u)$ is the smallest threshold containing probability at least u. Similarly, F_t and $\widehat{F}_t$ denote the distributions represented by the recursive curve q_t and final forecast $\widehat{q}_t$.

The one-dimensional Wasserstein distance compares distributions by matching equal probability ranks:

$$W_1(F, G) = \int_0^1 |q_F(u) - q_G(u)|\, du.$$

This is the minimum average distance that probability mass must move to turn one distribution into the other, measured in the forecast variable's units. It captures errors across the quantile curve, including its location and spread. Writing $m_t^\star = \int_0^1 q_t^\star(u)\, du$ and $r_t^\star = q_t^\star - m_t^\star$, Eq. (4) gives

$$W_1(\widehat{F}_t, F_t^\star) \le |m_{\psi,t}(\mathcal{C}) - m_t^\star| + \int_0^1 |r_t(u) - r_t^\star(u)|\, du. \tag{6}$$

The terms measure location and shape error. We separate shape error using F_t^{M}, the marginal CDF of the learned joint density $p_t(\mathbf{x} \mid \mathcal{C})$. QFLOW approximates this marginal, while the learned denoiser and empirical schedules determine its agreement with the data.

Proposition 3.1 (Location, transport, and model error)**.** *For a fixed context, suppose the distributions have finite first moments. If the transport error satisfies $W_1(F_H, F_H^{\mathrm{M}}) \le E_{\mathrm{tr}}$ and the model error satisfies $W_1(F_H^{\mathrm{M}}, F_H^\star) \le E_{\mathrm{model}}$, then*

$$W_1(\widehat{F}_H, F_H^\star) \le |m_{\psi,H} - m_H^\star| + 2E_{\mathrm{tr}} + 2E_{\mathrm{model}}. \tag{7}$$

This follows from Eq. (6) by inserting the learned marginal between the two shapes. Centering can at most double a Wasserstein distance, which accounts for the factors of two. Theorem A.1 in Appendix A.4 bounds E_{tr} and E_{model} through their initial errors and accumulated differences in quantile velocity, allowing for numerical error. A velocity difference means that corresponding quantiles move at different speeds; even from the same starting curve, their separation can grow over the horizon. In a coupled flow, the correct marginal velocity averages over the other coordinates, whereas QFLOW uses the scalar approximation derived above.

The distinction guides the experiments: Section 4.1 tests forecast accuracy against held-out observations; Section 4.2 compares with samples from the same learned flow to assess the scalar transport; and Section 4.3 separates location and shape contributions. A small transport error alone does not imply accuracy against the data, which also depends on E_{model}.

4 Experiments

Protocol. We use nine datasets under the NsDiff protocol (Ye et al., 2025). Each forecast uses a context of $T = 168$ time steps and predicts the next $H = 192$ steps, except on the short ILI series, where both lengths are 36. Hyperparameters are selected on validation data using seed 1. The denoiser is then trained for ten epochs with each of seeds 1, 2, and 3, keeping the weights from the epoch that Algorithm 1 selects. The mean head is fitted separately and held fixed across these runs, so seed variation reflects changes in the transported shape. Appendix B.4 reports the data splits, settings, and numbers of evaluated windows.

Comparisons. Tables 1 and 2 compare QFLOW with six diffusion forecasters using the published benchmark scores of Ye et al. (2025) under the same evaluation protocol. CRPS and QICE assess distributions; MSE and MAE assess the mean forecast, with lower values better throughout.

Table 1: Probabilistic benchmark under the NsDiff protocol. Baselines are published values from Ye et al. (2025); QFLOW entries average three denoiser seeds with one fixed learned mean head. QICE is in percentage points. Standard deviations across seeds are in Table 6. Legend: rank 1, rank 2, and rank 3 within each dataset/metric pair.

Models	Datasets	ETTh1	ETTh2	ETTm1	ETTm2	Electricity	Exchange	ILI	Solar	Traffic
TimeGrad (2021)	CRPS	0.606	1.212	0.647	0.775	0.397	0.826	1.140	**0.293**	0.407
	QICE	6.731	9.488	6.693	6.962	7.118	9.464	6.519	7.378	4.581
CSDI (2021)	CRPS	0.492	0.647	0.524	0.817	0.577	0.855	1.244	0.432	1.418
	QICE	3.107	5.331	2.828	8.106	7.506	7.864	7.693	9.957	13.613
TimeDiff (2023)	CRPS	0.465	0.471	0.464	0.316	0.750	0.433	1.153	0.700	0.771
	QICE	14.931	14.813	14.795	13.385	15.466	14.556	14.942	14.914	15.439
Diffusion-TS (2024)	CRPS	0.603	1.168	0.574	1.035	0.633	1.251	1.612	0.470	0.668
	QICE	6.423	9.577	5.605	9.959	8.205	10.411	10.090	6.627	5.958
TMDM (2024)	CRPS	0.452	0.383	0.375	0.289	0.461	0.336	0.967	0.350	0.557
	QICE	2.821	4.471	2.567	2.610	10.562	6.393	6.217	9.342	10.676
NsDiff (2025)	CRPS	0.392	0.358	0.346	**0.256**	0.290	0.324	**0.806**	0.300	0.378
	QICE	**1.470**	**2.074**	**2.041**	**2.030**	6.685	5.930	**5.598**	6.820	**3.601**
QFLOW (ours)	CRPS	**0.325**	**0.341**	**0.282**	0.268	**0.208**	**0.234**	1.116	0.304	**0.284**
	QICE	2.112	4.139	3.662	5.331	**2.107**	**3.169**	7.193	**6.456**	4.895

Table 2: Point-forecast benchmark under the NsDiff protocol. Baseline values are from Ye et al. (2025), in the MAE/MSE order given by their Table 6's caption. QFLOW uses the learned location in Eq. (4), and Table 5 (Appendix C.2) compares it with the characteristic location. Colors denote ranks as in Table 1, and lower is better.

Models	Datasets	ETTh1	ETTh2	ETTm1	ETTm2	Electricity	Exchange	ILI	Solar	Traffic
TimeGrad (2021)	MSE	1.062	3.462	1.218	1.690	0.505	1.567	4.197	0.475	0.983
	MAE	0.813	1.496	0.831	0.967	0.504	1.058	1.414	0.446	0.535
CSDI (2021)	MSE	0.949	1.226	1.002	1.723	1.007	1.701	4.515	0.763	1.731
	MAE	0.708	0.900	0.752	1.069	0.822	1.081	1.481	0.675	0.925
TimeDiff (2023)	MSE	0.517	**0.456**	0.537	0.268	0.879	0.402	3.958	0.821	1.350
	MAE	0.479	0.485	0.477	**0.333**	0.764	0.446	1.169	0.713	0.784
Diffusion-TS (2024)	MSE	1.089	3.273	1.030	2.372	1.072	3.628	6.053	0.749	1.473
	MAE	0.774	1.411	0.744	1.232	0.856	1.564	1.788	0.740	0.815
TMDM (2024)	MSE	0.696	0.512	0.494	0.315	0.257	0.334	3.636	0.250	0.679
	MAE	0.607	0.490	0.455	0.395	0.359	0.430	1.175	0.316	0.425
NsDiff (2025)	MSE	0.594	0.514	0.488	0.281	0.209	0.300	**2.846**	**0.242**	0.637
	MAE	0.523	0.490	0.455	0.352	0.306	0.412	**0.985**	**0.307**	0.373
QFLOW (ours)	MSE	**0.405**	0.457	**0.290**	**0.266**	**0.174**	**0.160**	3.513	0.268	**0.481**
	MAE	**0.403**	**0.443**	**0.349**	0.335	**0.271**	**0.299**	1.361	0.347	**0.329**

4.1 Forecast accuracy with lightweight networks

QFLOW has the lowest CRPS on six of nine datasets among the compared diffusion methods, and ranks in the top three on all nine. It also achieves the lowest QICE on Electricity, Exchange, and Solar, with top-three QICE on seven datasets. Across both tables, it ranks in the top three in $33/36$

comparisons and first in 21. The remaining QICE ranks are fourth, on ETTm1 and ILI; Appendix C.4 examines interval coverage.

QFLOW is already competitive with the characteristic as its location: against the same diffusion baselines, this variant attains top-three CRPS on seven datasets and top-three MSE on five (Table 5). Tables 1 and 2 report the learned-location refinement, which improves average CRPS and MSE on all nine datasets while retaining the transported shape. Appendix C.3 reports seed variation and non-diffusion baselines.

4.2 Fidelity to the learned flow

To assess the scalar approximation, we compare its quantiles with empirical marginal quantiles obtained by propagating whole windows through the same frozen denoiser. For each context, we construct $N = 2{,}048$ initial rolling states from QFLOW's initial Gaussian curve q_0. For coordinate i, which indexes a time position and variable, we use

$$x_{0,i}^{(\ell)} = q_{0,i}(u_{\pi_i(\ell)}) = m_{0,i} + \sigma_{0,i}\Phi_{\mathcal{N}}^{-1}(u_{\pi_i(\ell)}), \qquad u_\ell = 0.01 + 0.98\,\frac{\ell - \frac{1}{2}}{N}.$$

Each coordinate covers the same evenly spaced probability levels. An independent random permutation π_i assigns these levels to the states. Without it, every coordinate in a state would have the same percentile rank, imposing perfect rank dependence at initialization. The states share the observed conditioning context and use the same local moments as QFLOW.

Each initial state follows the full denoiser update without further noise injection. At every step, the sampled ensemble and QFLOW use schedules estimated from the same characteristic window, together with the same centering, variance rescaling, and clipping rules. Empirical quantiles across the propagated states then provide a reference for each forecast variable and lead.

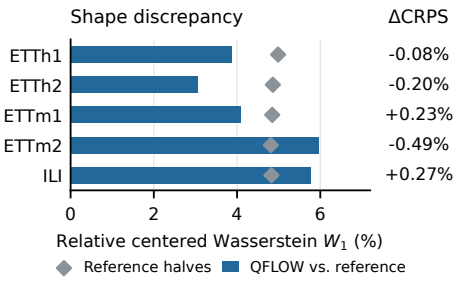

Figure 2: **Transport versus sampled flow.** Bars: centered W_1 between QFLOW and the reference, relative to the reference's spread (lower is better). Diamonds: the same measure between two halves of the reference, a finite-sample floor. ΔCRPS: QFLOW minus reference, relative to QFLOW. Averages over three seeds.

Figure 2 compares the two representations on the five datasets with the fewest test windows (Appendix C.1). Their centered shapes differ by 3.1–6.0% of the reference's spread, while their CRPS scores differ by less than 0.5%. This agreement supports the scalar approximation within the evaluated numerical procedure.

The diamonds show how much quantile estimates vary within the finite reference ensemble, as the normalized centered Wasserstein distance W_1 between the marginals of two groups of 1,024 propagated states. A bar near its diamond means that transport differs from the reference about as much as the reference's halves differ from each other. Because the halves share their initial levels and ensemble corrections, the diamonds give a descriptive scale rather than a confidence bound (Appendix C.1).

4.3 Separating the location and shape contributions

Equation (4) combines a location with a centered shape. We assess these components by changing one at a time: first the location with the transported shape fixed, then the quantile offsets with the fitted mean head fixed.

To construct the residual baseline, we collect errors $X_{t,i} - m_{\psi,t,i}(\mathcal{C})$ over validation windows, separately for each lead t and variable i. Their empirical quantile function $r_{t,i}^{\mathrm{res}}(u)$ gives the forecast

$$\widehat{q}_{t,i}^{\,\mathrm{res}}(u) = m_{\psi,t,i}(\mathcal{C}) + r_{t,i}^{\mathrm{res}}(u).$$

These offsets stay fixed across test contexts, whereas QFLOW's $r_{t,i}$ adapts through its schedules and denoiser derivatives. Residual offsets can retain mean bias, so the comparison fixes the point predictor, without requiring identical quantile means (Appendix C.3).

Learned location. We compare $x_t + r_t(u)$ with $m_{\psi,t}(\mathcal{C}) + r_t(u)$, using exactly the same r_t. Figure 3 shows lower average CRPS on every dataset after the location refinement, with the largest gain on Solar.

Context-dependent shape. The figure's numeric column compares QFLOW with the residual forecast $\widehat{q}_t^{\,\mathrm{res}}(u)$ defined above, using the same fitted mean head. This tests whether adapting uncertainty to the current context improves on reusing past forecast errors. Transport lowers CRPS on ETTh1, Electricity, and Exchange, is close on ETTh2, and trails the residual baseline on the remaining datasets.

Together with Section 4.2, these results distinguish fidelity to the learned flow from calibration against observations. A narrow curve may reproduce that flow accurately yet contain fewer observations than its nominal coverage requires. Appendix C.2 examines variation across forecast windows, and Appendix C.4 reports interval coverage and width.

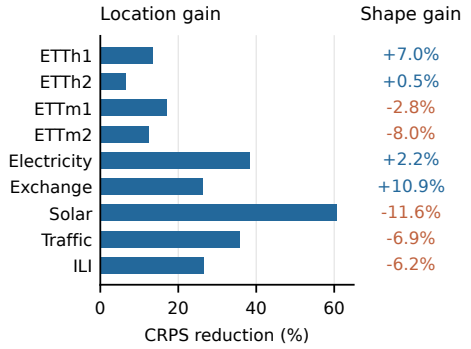

Figure 3: **Location and shape contributions.** Relative CRPS reductions from the learned location (bars, same transported shape) and from the transported shape (numbers, same mean head, versus fixed validation residuals). Positive values favor QFLOW. Reductions use three-seed mean CRPS (Table 5 and Appendix C.3).

Direct fitting and changing the quantile levels. Direct quantile regression provides another way to adapt uncertainty to the context. We fit a temporal linear head by pinball loss at the 33 evaluated levels, directly optimizing the loss that CRPS averages, with 33 times the mean head's parameters. It outperforms every compared diffusion forecaster on eight datasets in CRPS and all nine in QICE; NsDiff retains the best ILI CRPS (Appendix C.3).

QFLOW's denoiser instead defines transport independently of probability levels, whose number and positions can change without retraining. For example, levels $(0.1, 0.9)$ yield a central 80% interval, while $(0.05, 0.95)$ yield a 90% interval. A quantile-regression head must interpolate additional levels or be refitted. We test this dependence by fitting nine rather than 33 levels. Both quantile-regression heads interpolate 0.05 and 0.95, but the coarser fit incurs higher tail losses. With its existing weights, QFLOW has lower upper-tail pinball loss than both heads on ETTh2 and Exchange (Figure 5). These tail advantages do not hold on every dataset.

Modularity and numerical limits. In a single-dataset comparison on ETTm2, replacing the mean head with a Transformer leaves transport unchanged but does not improve accuracy (Appendix C.2). Numerical corrections also affect uncertainty with the networks fixed: variance rescaling improves CRPS and QICE in a separate ETTh1 configuration, while removing clipping on ILI improves coverage without restoring its nominal level (Appendix C.4). Thus correcting accumulated spread errors can help, but restricting tail values can impair calibration even when the recursion remains numerically stable.

5 Conclusion

QFLOW turns physical-time probability flow into an ordered quantile forecast by transporting log-spacing and refining location with a small conditional-mean head. It produces marginal uncertainty without forecast ensembles and remains competitive with recent diffusion baselines while using only a linear layer or a single attention block. Comparisons with sampled flow support the scalar approximation. Component comparisons distinguish the benefits of location refinement from those of adapting uncertainty.

Limitations. The scalar approximation does not recover joint dependence, and finite grids, empirical schedules, and clipping can impair tails. Fixed validation residuals outperform transport on five of nine datasets, and direct quantile regression is more accurate at the 33 levels it was trained on. Changing quantile levels without retraining is useful, but accurate uncertainty across contexts and tail levels remains the main direction for improvement.

Ethics Statement

Better calibrated long-horizon forecasts can improve planning in energy, traffic, and public-health settings, but miscalibrated uncertainty can create overconfident decisions in precisely those high-stakes domains. Calibration diagnostics, transparent reporting of failure modes, and domain-specific human oversight are therefore necessary when deploying such systems.

Reproducibility Statement

All datasets are public benchmarks. Section 3.3 specifies training and inference, and Appendix A supplies the derivations and approximation assumptions. Appendix B describes preprocessing, schedule estimation, numerical corrections, evaluation metrics, and dataset-specific settings. Appendix C reports the supporting experiments, including their sample counts and variability across seeds. The code for the method and for every experiment will be released publicly with the camera-ready version.

AI Use Statement

In this work, we used generative AI tools to provide critical ingredients for proving mathematical claims and to assist in writing proofs, to propose and refine parts of the research methodology, and to implement methods. We have not used generative AI tools to generate synthetic data sets, and the other tasks with required disclosure are not applicable to this work. Additionally, we used generative AI tools to draft parts of this paper, to edit it for readability, to identify relevant literature, and to create software code and figures. We have reviewed all AI-assisted work. The authors derived the core of the method, the probability flow and scalar transport of Appendices A.1 and A.2, and wrote its base implementation, including scale and noise schedule estimators, without AI assistance. AI tools proved the results of Appendices A.3 and A.4. AI tools also proposed computing the Jacobian penalty (Eq. (5)) efficiently with random probes, proposed the variance correction (Eq. (22)), and improved the schedule estimators, in particular with the innovation blend (Appendix B.1). The authors designed the control experiments of Sections 4.2 and 4.3, and AI tools wrote the code that runs them and produces all tables and figures. The authors checked these proofs, components, and code before using them in this work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Guillaume Carlier, Victor Chernozhukov, and Alfred Galichon. Vector quantile regression: An optimal transport approach. *The Annals of Statistics*, 44(3):1165 – 1192, 2016. doi: 10.1214/15-AOS1401. URL https://doi.org/10.1214/15-AOS1401.

Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. Probabilistic forecasting with spline quantile function rnns. In Kamalika Chaudhuri and Masashi Sugiyama (eds.), *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1901–1910. PMLR, 16–18 Apr 2019.

Tilmann Gneiting and Matthias Katzfuss. Probabilistic forecasting. *Annual Review of Statistics and Its Application*, 1(1):125–151, 2014.

Tilmann Gneiting and Roopesh Ranjan. Comparing density forecasts using threshold- and quantile-weighted scoring rules. *Journal of Business & Economic Statistics*, 29(3):411–422, 2011. doi: 10.1198/jbes.2010.08110.

Xizewen Han, Huangjie Zheng, and Mingyuan Zhou. Card: Classification and regression diffusion models. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Proc. NeurIPS*, 2022.

Roger Koenker and Gilbert Bassett Jr. Regression quantiles. *Econometrica*, 46(1):33–50, 1978. doi: 10.2307/1913643.

Yuxin Li, Wenchao Chen, Xinyue Hu, Bo Chen, Baolin Sun, and Mingyuan Zhou. Transformer-modulated diffusion models for probabilistic multivariate time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

James E. Matheson and Robert L. Winkler. Scoring rules for continuous probability distributions. *Management Science*, 22(10):1087–1096, 1976.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8857–8868. PMLR, 18–24 Jul 2021.

Lifeng Shen and James Kwok. Non-autoregressive conditional diffusion models for time series prediction. In *International Conference on Machine Learning*, pp. 31016–31029. PMLR, 2023.

Edgardo Solano-Carrillo, Sreerag Vadakkemeppully Naveenachandran, and Julia Niebling. Tedm: Time series forecasting with elucidated diffusion models. In *International Conference on Learning Representations*, 2026.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems*, 2021.

Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017. doi: 10.48550/arXiv.1711.11053.

Weiwei Ye, Zhuopeng Xu, and Ning Gui. Non-stationary diffusion for probabilistic time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

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

Michaël Zamo and Philippe Naveau. Estimation of the continuous ranked probability score with limited information and applications to ensemble weather forecasts. *Mathematical Geosciences*, 50:209–234, 2018. doi: 10.1007/s11004-017-9709-7.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pp. 11106–11115. AAAI Press, 2021.

The appendices follow the order of the paper. Appendix A derives the method of Section 3 and its error bound, Appendix B specifies how QFLOW is implemented and evaluated, and Appendix C reports the experiments that support Section 4.

A Derivations and Guarantees

This appendix follows the method in order. Appendix A.1 expresses the probability flow of Section 3.1 through the denoiser, Appendix A.2 derives the scalar transport of Section 3.2, Appendix A.3 bounds the dependence it discards and gives the Jacobian probe identities of Section 3.3, and Appendix A.4 proves the error bound of Section 3.4. Throughout, the observed context $\mathcal{C}$, the last T time steps of all d variables, is fixed. We retain it in density notation and omit it from other arguments when unambiguous. The random vector $\mathbf{X}_t$ is the forecast state at step t, with one coordinate for each variable at each updated rolling-window position. A characteristic $\mathbf{x}_t$ is one deterministic path of this state, $\mathbf{q}_t$ is a vector transport map, and q_t is one scalar transported quantile curve. A star denotes an ideal denoiser or a data-generating target; the superscript M denotes the learned joint model. The same physical index t is used at discrete steps and for their continuous interpolation.

A.1 Probability flow and the denoiser

Like score-based diffusion models (Song et al., 2021; Karras et al., 2022) and Solano-Carrillo et al. (2026), we start from a linear-drift stochastic differential equation (SDE) with state-independent diffusion, which defines the scale and noise schedules s_t and $\mathbf{\Sigma}_t$:

$$d\mathbf{X}_t = f_t \mathbf{X}_t \, dt + \mathbf{G}_t \, d\boldsymbol{\omega}_t, \qquad s_t = \exp\!\left(\int_0^t f_v \, dv \right), \quad \mathbf{\Sigma}_t = \int_0^t s_v^{-2} \mathbf{G}_v \mathbf{G}_v^\top \, dv.$$

Here f_t is a scalar drift coefficient, $\mathbf{G}_t$ is a state-independent diffusion matrix, and $\boldsymbol{\omega}_t$ is a Wiener process. Solving this linear SDE gives the Gaussian transition kernel

$$\mathbf{X}_t \mid \mathbf{X}_0 = \mathbf{x}_0 \sim \mathcal{N}(s_t \mathbf{x}_0, s_t^2 \mathbf{\Sigma}_t), \qquad \dot{\mathbf{\Sigma}}_t = s_t^{-2} \mathbf{G}_t \mathbf{G}_t^\top.$$

The transition covariance $s_t^2 \mathbf{\Sigma}_t$ measures uncertainty conditional on the initial state. The marginal density $p_t(\mathbf{x} \mid \mathcal{C}) = \int \mathcal{N}(\mathbf{x}; s_t \mathbf{x}_0, s_t^2 \mathbf{\Sigma}_t) p_0(\mathbf{x}_0 \mid \mathcal{C}) \, d\mathbf{x}_0$ also averages over the initial distribution.

The Fokker–Planck equation describes how this density evolves. Following Song et al. (2021), we use $\mathbf{G}_t \mathbf{G}_t^\top = s_t^2 \dot{\mathbf{\Sigma}}_t$ to write it as a continuity equation driven by a deterministic velocity:

$$\partial_t p_t = -\nabla \cdot (p_t \mathbf{v}_t), \qquad \mathbf{v}_t(\mathbf{x}) = f_t \mathbf{x} - \tfrac{1}{2} s_t^2 \dot{\mathbf{\Sigma}}_t \nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid \mathcal{C}). \tag{8}$$

State independence of $\mathbf{G}_t$ makes this identity exact. Under the usual existence and uniqueness conditions, the probability-flow ordinary differential equation (ODE) $\dot{\mathbf{x}}_t = \mathbf{v}_t(\mathbf{x}_t)$ preserves the SDE's time-marginal densities (Song et al., 2021).

To express that velocity through denoising, we define $D^\star(\mathbf{x}/s_t, \mathbf{\Sigma}_t) = \mathbb{E}[\mathbf{X}_0 \mid \mathbf{X}_t = \mathbf{x}, \mathcal{C}]$. For positive definite $\mathbf{\Sigma}_t$, differentiating the Gaussian kernel inside the marginal integral gives

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid \mathcal{C}) = s_t^{-1} \mathbf{\Sigma}_t^{-1} \left[D^\star(\mathbf{x}/s_t, \mathbf{\Sigma}_t) - \mathbf{x}/s_t \right]. \tag{9}$$

Substituting this score identity into Eq. (8) yields

$$\mathbf{v}_t(\mathbf{x}) = f_t \mathbf{x} - \tfrac{1}{2} s_t \dot{\mathbf{\Sigma}}_t \mathbf{\Sigma}_t^{-1} \left[D^\star(\mathbf{x}/s_t, \mathbf{\Sigma}_t) - \mathbf{x}/s_t \right]. \tag{10}$$

For a general covariance, $\dot{\mathbf{\Sigma}}_t \mathbf{\Sigma}_t^{-1}$ must be kept as this matrix product; it does not in general equal $\partial_t \log \mathbf{\Sigma}_t$. Under a diagonal schedule the product is element-wise, and one scalar coordinate has velocity

$$v_t(x) = \left(\partial_t \log s_t + \tfrac{1}{2} \partial_t \log \Sigma_t \right) x - \tfrac{1}{2} s_t (\partial_t \log \Sigma_t) D^\star(x/s_t, \Sigma_t).$$

The velocity above is oriented in the SDE's noise-adding direction. Forecasting uses the opposite, denoising direction: after replacing $D^\star$ by D_θ, a first-order step subtracts this velocity and gives Eq. (1). The index t still increases by one per physical forecast step; the sign reversal introduces no additional time variable or inner sampling loop. The empirical schedules and learned denoiser approximate the local dynamics used for forecasting.

A characteristic and the conditional mean of the state generally evolve differently: a characteristic obeys $\dot{x}_t = v_t(x_t)$, while the mean obeys $\partial_t \mathbb{E}[X_t \mid \mathcal{C}] = \mathbb{E}[v_t(X_t) \mid \mathcal{C}]$. If v_t is affine in x, then $\mathbb{E}[v_t(X_t) \mid \mathcal{C}] = v_t(\mathbb{E}[X_t \mid \mathcal{C}])$, so a characteristic started at the mean stays at the mean. A nonlinear velocity, such as one computed from a trained denoiser, breaks this identity in general. Likewise, the forward-SDE identity $\mathbb{E}[X_t \mid \mathcal{C}] = s_t \mathbb{E}[X_0 \mid \mathcal{C}]$ concerns an ensemble expectation, not an arbitrary characteristic.

A.2 From vector transport to scalar log-spacing

We now use this velocity to evolve an entire transport map. The state has n coordinates, one for each variable at each updated rolling-window position; thus n is d times the number of those positions. We consider a map $\mathbf{q}_t$ that pushes the uniform law on $(0,1)^n$ to p_t, with Jacobian $\mathbf{A}_t = \nabla_{\mathbf{u}} \mathbf{q}_t$ of positive determinant. The initial map may be a vector quantile in the optimal-transport sense (Carlier et al., 2016). Moving it with the flow, $\partial_t \mathbf{q}_t = \mathbf{v}_t(\mathbf{q}_t)$, keeps it a map that pushes the uniform law to p_t, although for a general coupled velocity it may not remain the optimal-transport map at later times. The change-of-variables identity remains valid:

$$p_t(\mathbf{q}_t(\mathbf{u}) \mid \mathcal{C}) \det \mathbf{A}_t = 1.$$

Taking its logarithmic derivative along the flow and substituting the continuity equation gives

$$\partial_t \log \det \mathbf{A}_t = -\left[\partial_t \log p_t + \nabla \log p_t \cdot \partial_t \mathbf{q}_t\right]_{\mathbf{x}=\mathbf{q}_t} = \nabla \cdot \mathbf{v}_t(\mathbf{q}_t). \tag{11}$$

For separable dynamics and a diagonal map, the determinant factors into scalar derivatives. One coordinate therefore satisfies

$$\partial_t \log q_t'(u) = \left[\partial_x v_t(x)\right]_{x=q_t(u)} = \partial_t \log s_t + \tfrac{1}{2}\partial_t \log \Sigma_t - \tfrac{1}{2} s_t (\partial_t \log \Sigma_t) \left[\partial_x D_\theta(x/s_t, \Sigma_t)\right]_{x=q_t(u)}. \tag{12}$$

As in the state update, the forecasting step uses the negative of this noise-adding velocity derivative. Integrating it over one schedule interval with increments $\Delta \log s_t, \Delta \log \Sigma_t$ gives the first-order log-spacing update in Eq. (2). Exponentiating makes the coefficient positive for every finite derivative estimate. Integrating in u gives $q_{t+1} = C_t + \Phi_t \circ q_t$; choosing the integration constant C_t so that $\int q_{t+1} = x_{t+1}$ gives Eq. (3).

A.3 Discarded dependence and Jacobian probes

QFLOW's diagonal approximation makes two simplifications: it replaces the noise covariance $\mathbf{\Sigma}_t$ by its diagonal, and it keeps only the diagonal of the transport Jacobian $\mathbf{A}_t$, which replaces the coupled transport by scalar curves. These are approximation choices, not properties guaranteed by training. To see the latter error explicitly, let $\mathbf{V}_t = [\nabla_{\mathbf{x}} \mathbf{v}_t(\mathbf{x})]_{\mathbf{x}=\mathbf{q}_t}$. Differentiating $\partial_t \mathbf{q}_t = \mathbf{v}_t(\mathbf{q}_t)$ in $\mathbf{u}$ gives $\dot{\mathbf{A}}_t = \mathbf{V}_t \mathbf{A}_t$, hence, for each coordinate i with $(\mathbf{A}_t)_{ii} > 0$,

$$\partial_t \log(\mathbf{A}_t)_{ii} = (\mathbf{V}_t)_{ii} + R_{t,i}, \qquad R_{t,i} = \sum_{j \neq i} (\mathbf{V}_t)_{ij} \frac{(\mathbf{A}_t)_{ji}}{(\mathbf{A}_t)_{ii}}. \tag{13}$$

For diagonal schedules and $\mathbf{J}_\theta = \partial_{\mathbf{x}} D_\theta(\mathbf{x}/s_t, \mathbf{\Sigma}_t)$,

$$|R_{t,i}| \leq \tfrac{1}{2} s_t |\partial_t \log \Sigma_{t,i}| \|(\mathbf{J}_\theta)_{i,-i}\|_2 \frac{\|(\mathbf{A}_t)_{-i,i}\|_2}{(\mathbf{A}_t)_{ii}}. \tag{14}$$

Thus the omitted term depends on both denoiser coupling and the current transport map. A small denoiser-Jacobian ratio alone is insufficient to certify small error.

The same diagonal replacement also changes the determinant that relates the transport map to its density. Here diag extracts a matrix's diagonal as a vector, while Diag places a vector on the diagonal of an otherwise zero matrix. Thus $\mathbf{D}_A = \mathrm{Diag}(\mathrm{diag}\, \mathbf{A}_t)$ keeps $\mathbf{A}_t$'s diagonal entries and sets every off-diagonal entry to zero. With $\mathbf{B}_A = \mathbf{D}_A^{-1/2}(\mathbf{A}_t - \mathbf{D}_A)\mathbf{D}_A^{-1/2}$, if $\mathbf{D}_A > 0$ and $\rho = \|\mathbf{B}_A\|_2 < 1$, the matrix-log series and $\mathrm{tr}\, \mathbf{B}_A = 0$ imply

$$\left| \log \det \mathbf{A}_t - \sum_i \log(\mathbf{A}_t)_{ii} \right| \leq \frac{\|\mathbf{B}_A\|_F^2}{2(1-\rho)}. \tag{15}$$

Indeed, the series begins at order two and $|\operatorname{tr}(\mathbf{B}_A^k)| \le \|\mathbf{B}_A\|_F^2 \rho^{k-2}$ for $k \ge 2$. To account for the discarded covariance terms, we define $\mathbf{M} = \dot{\mathbf{\Sigma}}\mathbf{\Sigma}^{-1}$ and $\mathbf{M}^{\mathrm{d}} = \dot{\mathbf{\Sigma}}^{\mathrm{d}}(\mathbf{\Sigma}^{\mathrm{d}})^{-1}$, where $\mathbf{\Sigma}^{\mathrm{d}}$ is the diagonal part. At fixed s and $\mathbf{x}$, Eq. (10) yields

$$\|\mathbf{v}^{\mathbf{\Sigma}} - \mathbf{v}^{\mathrm{d}}\|_2 \le \tfrac{1}{2} s \left(\|\mathbf{M} - \mathbf{M}^{\mathrm{d}}\|_2 \|D^{\mathbf{\Sigma}} - \mathbf{x}/s\|_2 + \|\mathbf{M}^{\mathrm{d}}\|_2 \|D^{\mathbf{\Sigma}} - D^{\mathrm{d}}\|_2 \right). \tag{16}$$

The two bounds identify the quantities that must be small for the diagonal approximation to be accurate. The training penalty addresses denoiser derivatives, which also determine the variability of the numerical probe estimates below.

Probe identities. The training penalty in Eq. (5) and the estimate of $\operatorname{diag} \mathbf{J}_\theta$ used at inference both avoid forming the Jacobian: each requires only a probe with zero mean and covariance $\mathbf{I}$. Gaussian and Rademacher probes both satisfy these conditions:

$$\mathbb{E}[\boldsymbol{\xi} \odot \mathbf{J}_\theta^\top \boldsymbol{\xi}] = \operatorname{diag} \mathbf{J}_\theta, \qquad \mathbb{E}\|\mathbf{J}_\theta^\top \boldsymbol{\xi}\|_2^2 = \operatorname{tr}(\mathbf{J}_\theta \mathbf{J}_\theta^\top) = \|\mathbf{J}_\theta\|_F^2. \tag{17}$$

For coordinate i, a Rademacher probe gives diagonal-estimator variance $\sum_{j \ne i} (\mathbf{J}_\theta)_{ji}^2$; a Gaussian probe adds $2(\mathbf{J}_\theta)_{ii}^2$. The difference follows from $\xi_i^2 = 1$ for Rademacher entries. We therefore use Rademacher probes for diagonal estimation, while Gaussian probes provide an unbiased training penalty with the same expected Frobenius norm. Different probe families are a variance-reduction choice, not a requirement of the transport derivation. Averaging P independent probes divides the diagonal-estimator variance by P. The training penalty therefore controls the total Jacobian energy, including the terms that create probe variance. It is not a selective off-diagonal penalty, nor does it force coordinate independence. Moreover, unbiasedness of the derivative estimate does not imply unbiasedness after exponentiation in ϕ_t.

A.4 Marginal error relative to the data distribution

The remainder $R_{t,i}$ in Eq. (13) and the bounds in Eqs. (15) and (16) describe what scalar transport discards at a single time. To connect these local errors to forecast accuracy, we compare QFLOW with the learned flow's marginal distribution and then with the data distribution. All velocities below use the forecasting direction of Eq. (1), with the denoising sign already incorporated. For a fixed coordinate i, let $p_{t,i}(x \mid \mathcal{C})$ be the ith marginal density of the learned joint flow $p_t(\mathbf{x} \mid \mathcal{C})$, and let $F_{t,i}^{\mathrm{M}}$ and $q_{t,i}^{\mathrm{M}}$ be its CDF and quantile. Assume this density is positive. Its marginal velocity is

$$\bar{v}_{t,i}(x) = \mathbb{E}[v_{t,i}(\mathbf{X}_t) \mid X_{t,i} = x, \mathcal{C}]. \tag{18}$$

Integrating the joint continuity equation over the other coordinates, with vanishing boundary flux, gives $\partial_t F_{t,i}^{\mathrm{M}}(x) = -p_{t,i}(x \mid \mathcal{C}) \bar{v}_{t,i}(x)$. Differentiating $F_{t,i}^{\mathrm{M}}(q_{t,i}^{\mathrm{M}}(u)) = u$ consequently gives

$$\partial_t q_{t,i}^{\mathrm{M}}(u) = \bar{v}_{t,i}(q_{t,i}^{\mathrm{M}}(u)). \tag{19}$$

This averaging is essential. The ith coordinate of a joint state moves with $v_{t,i}(\mathbf{x})$, which depends on the other coordinates, so a state whose ith coordinate sits at the marginal quantile $q_{t,i}^{\mathrm{M}}(u)$ generally moves at a different speed than that quantile.

We suppress i below for readability. Let q_t interpolate QFLOW's recursive curve before the final learned translation, and suppose

$$\partial_t q_t(u) = \widetilde{v}_t(q_t(u)) + e_t(u).$$

Here $\widetilde{v}_t$ is the scalar approximation and e_t accounts for integration and other numerical corrections. Fixing a probe realization makes these objects deterministic. Their accumulated discrepancies are

$$\delta_t = \int_0^1 |\widetilde{v}_t(q_t^{\mathrm{M}}(u)) - \bar{v}_t(q_t^{\mathrm{M}}(u))| \, du, \qquad \eta = \int_0^H \int_0^1 |e_t(u)| \, du \, dt.$$

As in Section 3.4, F_t and $\widehat{F}_t$ are the CDFs represented by q_t and $\widehat{q}_t$, while $F_t^\star$ is the conditional data CDF. For its quantile $q_t^\star$ with velocity $v_t^\star$, we similarly define $\gamma_t = \int_0^1 |\bar{v}_t(q_t^\star(u)) - v_t^\star(q_t^\star(u))| \, du$. We measure the distance between two distributions F and G with quantile functions q_F and q_G by the one-dimensional Wasserstein distance $W_1(F, G) = \int_0^1 |q_F(u) - q_G(u)| \, du$.

Theorem A.1 (Location, transport, and model error). *Suppose the quantile curves q_t, q_t^{M}, and $q_t^\star$ are absolutely continuous in t for every u, their distributions have finite first moments, the velocities $\widetilde{v}_t$ and $\bar{v}_t$ are uniformly L- and $\bar{L}$-Lipschitz, and the discrepancies δ_t, e_t, and γ_t are integrable. Define*

$$E_{\mathrm{tr}} = e^{LH}\left[W_1(F_0, F_0^{\mathrm{M}}) + \int_0^H \delta_t\, dt + \eta\right], \tag{20}$$

$$E_{\mathrm{model}} = e^{\bar{L}H}\left[W_1(F_0^{\mathrm{M}}, F_0^\star) + \int_0^H \gamma_t\, dt\right]. \tag{21}$$

Then $W_1(F_H, F_H^{\mathrm{M}}) \le E_{\mathrm{tr}}$, $W_1(F_H^{\mathrm{M}}, F_H^\star) \le E_{\mathrm{model}}$, and the final learned-location forecast satisfies Eq. (7).

Proof. The one-dimensional quantile identity gives $W_1(F_t, F_t^{\mathrm{M}}) = \int |q_t - q_t^{\mathrm{M}}|$. Differentiating its integrand almost everywhere, inserting Eq. (19), and using Lipschitz continuity gives

$$\partial_t |q_t - q_t^{\mathrm{M}}| \le L|q_t - q_t^{\mathrm{M}}| + |\widetilde{v}_t(q_t^{\mathrm{M}}) - \bar{v}_t(q_t^{\mathrm{M}})| + |e_t|.$$

Integrating in u and applying Grönwall's inequality proves the first bound. Repeating the argument between q_t^{M} and $q_t^\star$ proves the second. Finally, subtracting each distribution's own mean increases W_1 by at most a factor of two, since the difference of means is itself bounded by W_1. Applying this fact and the triangle inequality to the centered-shape term in Eq. (6) gives the final claim. □

For fixed H and bounded $L, \bar{L}$, vanishing initial, velocity, integration, and location errors imply $W_1(\widehat{F}_H, F_H^\star) \to 0$. This is a conditional consistency statement, not a convergence guarantee for the training algorithm. In particular, clipping and moment rescaling contribute to the approximation error whenever they change the reference dynamics. The exponential factors are worst-case amplification bounds.

Finite quantile grids. Restricting the integrals over u in the proof to $[\epsilon, 1-\epsilon]$ gives a bound of the same form for the truncated quantile integral $\int_\epsilon^{1-\epsilon} |q_H(u) - q_H^{\mathrm{M}}(u)|\, du$. For full W_1, one must also control the omitted tails, for example through uniform integrability of the approximating quantiles and convergence of numerical integration as $K, N_{\mathrm{GL}} \to \infty$ and $\epsilon \to 0$. A fixed 33-level evaluation grid does not establish that limit. The sampled-flow experiment measures a finite-grid endpoint shape discrepancy; it does not directly estimate the integrated velocity discrepancy $\int \delta_t\, dt$.

B Implementation and Evaluation

This appendix gives the details needed to implement and evaluate QFLOW. Appendix B.1 relates the scalar notation of Section 3 to the rolling window and defines the empirical scale and noise schedules. Appendix B.2 describes the networks, quadrature, and numerical corrections, Appendix B.3 the forecast metrics, and Appendix B.4 the data and dataset-specific settings.

B.1 Rolling-window notation and empirical schedules

The scalar notation in the main text suppresses the entries of the denoiser's rolling window. At forecast iteration t, we write that window as $\mathbf{x}^{[t]}$, with within-window position j and feature i. Schedule estimation returns arrays $s_{j,i}^{[t]}$ and $\Sigma_{j,i}^{[t]}$. For each updated entry, the main-text quantities mean

$$s_t = s_{j,i}^{[t]}, \quad \Sigma_t = \Sigma_{j,i}^{[t]}, \quad \Delta \log s_t = \log \frac{s_{j,i}^{[t]}}{s_{j-1,i}^{[t]}}, \quad \Delta \log \Sigma_t = \log \frac{\Sigma_{j,i}^{[t]}}{\Sigma_{j-1,i}^{[t]}}.$$

The window holds one more position than the entries it updates. Its first position is not updated and serves as the predecessor of the first updated entry, so every updated entry j has its predecessor $j-1$ in the same window, including on the first forecast step. The ratios therefore compare adjacent physical positions in the current window, not the schedule at one position as re-estimated at successive steps. The state and its quantile curves then advance together into the next window, whose last position holds the newest forecast step. Thus t counts forecast steps, j only indexes positions within the window at one step, and there is no separate diffusion-time loop. When the Jacobian is flattened, its coordinates run over both features and window positions.

Scale. For one variable, let c_j be the observed or recursively predicted value at position j of the current rolling window, in raw coordinates, that is, before standardization. The scale and noise estimates below both use these values. Under the linear-drift SDE, $\mathbb{E}[X_t] = s_t \mathbb{E}[X_0]$, so the scale is the ratio of the current mean level to the level at the start of the window. We track the level with a causal exponential moving average, $m_j^{\mathrm{loc}} = \beta m_{j-1}^{\mathrm{loc}} + (1-\beta)c_j$ with $m_0^{\mathrm{loc}} = c_0$ and β between 0.99 and 0.998 depending on the dataset, and set

$$\widehat{s}_j = \mathrm{clip}\left(\frac{|m_j^{\mathrm{loc}}| + \epsilon_s}{|m_0^{\mathrm{loc}}| + \epsilon_s},\ s_{\min}, s_{\max} \right),$$

where the small constant ϵ_s avoids division by zero and the bounds $s_{\min}, s_{\max}$ are set per dataset. We estimate the level in raw coordinates because standardization changes multiplicative mean relations: for $X' = aX + b$ and $\mathbb{E}[X_t] = s_t \mathbb{E}[X_0]$, one has $\mathbb{E}[X'_t] = s_t \mathbb{E}[X'_0] + (1 - s_t)b$. Mean ratios therefore do not survive standardization. The training-set scaler maps standardized values back to raw values for this estimate, so no test-target information enters.

Noise. The schedule needs the noise variance $\widehat{\Sigma}_j$ of the scaled state at each window position j, and Table 4 lists which of three estimators each dataset uses. The first two follow the SDE's transition variance $s_t^2 \mathbf{\Sigma}_t$ (Appendix A.1): they estimate a variance $\widehat{V}_j$ of the observed values and set $\widehat{\Sigma}_j = \widehat{V}_j / \widehat{s}_j^2$. The stable *cumulative* estimate $\widehat{V}_j$ is the sample variance of $c_0, \ldots, c_j$. The more responsive *sliding* estimate is the variance over the w positions centered at j, with fewer at the window's edges, smoothed by a causal exponential moving average with factor 0.99. Depending on the dataset, w is 24, 48, or 96. Both estimates are empirical proxies. They can differ from the SDE's transition variance, and unlike $\mathbf{\Sigma}_t$ they can decrease along the window.

The *innovation blend* instead estimates how fast the noise variance grows. Dividing the SDE's solution by s_t removes the multiplicative drift, $\mathbf{X}_t / s_t = \mathbf{X}_0 + \int_0^t s_v^{-1} \mathbf{G}_v \, d\boldsymbol{\omega}_v$, so its increment from $t-1$ to t has mean zero and covariance $\mathbf{\Sigma}_t - \mathbf{\Sigma}_{t-1}$. The squared increments $(\Delta z_j)^2$ of the de-scaled series, with $\Delta z_j = c_j / \widehat{s}_j - c_{j-1} / \widehat{s}_{j-1}$, therefore estimate the growth of Σ at each position, and their running sum estimates Σ itself. Single squared increments are noisy, so the blend averages them in two ways: r_j^{g} over all positions up to j, and r_j^{l} over the w positions centered at j, smoothed as above. With $d_j = r_j^{\mathrm{l}} - r_j^{\mathrm{g}}$ and the estimated squared standard errors $e_j^{\mathrm{g}}, e_j^{\mathrm{l}}$ of the two averages (variance divided by the number of averaged terms), the blend is

$$\omega_j = \frac{d_j^2}{d_j^2 + \lambda(e_j^{\mathrm{g}} + e_j^{\mathrm{l}}) + \epsilon_s}, \qquad \widehat{r}_j = (1 - \omega_j) r_j^{\mathrm{g}} + \omega_j r_j^{\mathrm{l}}, \qquad \widehat{\Sigma}_j = \sum_{k \le j} \widehat{r}_k.$$

The weight ω_j approaches one, favoring the local average, when the two averages differ by more than their standard errors suggest. A larger λ (0.25, 1, or 4, depending on the dataset) makes the blend less responsive. Because the two averages overlap, the sum $e_j^{\mathrm{g}} + e_j^{\mathrm{l}}$ is only a proxy for the variance of their difference. All variances are divided by the training-set scaler's variance to match the denoiser's standardized units, and positive floors, scale caps, and variance caps stabilize ratios and logarithms. Each schedule is recomputed from the current rolling state, which contains observed context and preceding predictions only.

B.2 Denoiser, grids, and numerical corrections

Networks and training. The mean head applies one T-to-H linear map separately to each variable. Its weight matrix and bias are shared across all d variables, so it has $H(T+1)$ parameters. The denoiser is either one temporal linear layer shared across variables or one cross-attention block over time. Its preconditioning, as in Karras et al. (2022) and Solano-Carrillo et al. (2026), scales the network's input and output and adds a skip connection, with factors that depend on the scale and noise level. For variance Σ, these factors and the loss weight λ_Σ of Eq. (5) are

$$c_{\mathrm{in}} = \frac{|s|}{\sqrt{\sigma_{\mathrm{data}}^2 + \Sigma}}, \quad c_{\mathrm{skip}} = \frac{\sigma_{\mathrm{data}}^2}{\sigma_{\mathrm{data}}^2 + \Sigma}, \quad c_{\mathrm{out}} = \frac{\sqrt{\Sigma}\,\sigma_{\mathrm{data}}}{\sqrt{\sigma_{\mathrm{data}}^2 + \Sigma}}, \quad \lambda_\Sigma = \frac{\Sigma + \sigma_{\mathrm{data}}^2}{\Sigma \sigma_{\mathrm{data}}^2},$$

with $D_\theta(x/s, \Sigma) = c_{\mathrm{skip}}(x/s) + c_{\mathrm{out}} f_\theta(c_{\mathrm{in}}(x/s), \mathbf{x}^{\mathrm{cond}})$ and $\sigma_{\mathrm{data}} = 1$, where f_θ is the trainable network. As in Eq. (2), x is the physical state value and x/s is the scaled denoiser input. Neither

network receives the noise level directly. It enters only through these factors. The attention block takes its queries from a conditioning window $\mathbf{x}^{\mathrm{cond}}$ of T observed or predicted values that ends $k+1$ positions before the newest state entry, with $k = 4$ on ETTh2, ETTm2, and Electricity, $k = 0$ on Traffic, and $k = 2$ on the other datasets. The linear layer does not use $\mathbf{x}^{\mathrm{cond}}$. Training adds noise at each clean window's own schedule variance rather than at a sampled noise level (Section 3.3), and the mean head is fitted separately on context–future pairs.

Quadrature and centering. The transport grid, the probability levels u_k at which each quantile curve is stored, excludes the endpoints 0 and 1, where the Gaussian initial curve q_0 diverges. For K uniform levels $u_k \in [\epsilon_u, 1-\epsilon_u]$, $\epsilon_u = 0.01$, we use $x_{\mathrm{ref}} = q_t(u_1)$. With Gauss–Legendre nodes ζ_g and weights w_g on $[-1, 1]$,

$$\Phi_t(q_t(u_k)) \approx \frac{q_t(u_k) - x_{\mathrm{ref}}}{2} \sum_{g=1}^{N_{\mathrm{GL}}} w_g \, \phi_t\left(x_{\mathrm{ref}} + \frac{q_t(u_k) - x_{\mathrm{ref}}}{2}(1+\zeta_g) \right).$$

The denoiser evaluates all KN_{GL} inputs in one batched call. The curve's mean is approximated by $\sum_k \nu_k q_t(u_k)$, with normalized trapezoidal weights ν_k that sum to one. This averages over the truncated interval $[\epsilon_u, 1-\epsilon_u]$ and omits the tails. The transport grid uses 65 levels on ETTh1 and ETTm1 and 33 on the other datasets. For evaluation, the curve is linearly interpolated onto a separate metric grid: CRPS uses 33 levels on $[0.01, 0.99]$, and QICE uses the nine deciles $0.1, \ldots, 0.9$. All reported configurations use $N_{\mathrm{GL}} = 8$ and one inference probe.

Variance correction. The transport determines spread through denoiser derivatives. Errors in those derivatives and their numerical integration can therefore make the curve progressively too narrow or too wide. The variance estimate of the noise schedule (Appendix B.1) provides a separate reference for the local spread. We use it to rescale the curve's width. The rescaling multiplies every centered quantile by the same factor, so it keeps the curve's center, order, and shape up to scale.

After transport, the curve is centered on x_{t+1} (Eq. (3)) and has weighted variance $\widetilde{V}_{t+1} = \sum_k \nu_k (q_{t+1}(u_k) - x_{t+1})^2$. The reference is the noise schedule's variance of the state at the entry's window position j, $V^{\mathrm{sch}}_{t+1} = \widehat{s}_j^2 \widehat{\Sigma}_j$. Multiplying every centered quantile by a factor a multiplies the curve's variance by a^2, so $a = (V^{\mathrm{sch}}_{t+1}/\widetilde{V}_{t+1})^{1/2}$ makes the two variances equal. We blend this factor with $a = 1$, which leaves the curve unchanged, using a weight α, and floor the curve's variance at a small ϵ_v:

$$q_{t+1}(u) \leftarrow x_{t+1} + \left[(1-\alpha) + \alpha \sqrt{\frac{V^{\mathrm{sch}}_{t+1}}{\max(\widetilde{V}_{t+1}, \epsilon_v)}} \right] \big(q_{t+1}(u) - x_{t+1} \big), \qquad 0 \le \alpha \le 1. \tag{22}$$

Thus $\alpha = 0$ retains raw transport, while $\alpha = 1$ matches V^{sch}_{t+1} whenever the floor is inactive. Intermediate values would temper the correction when the reference itself is uncertain, but validation selected $\alpha = 0$ or 1 on every dataset (Table 4). The reference is computed from observed and recursively predicted values, so matching it does not guarantee calibrated forecasts. Appendix C.4 evaluates the effect on forecast scores.

Safeguards and output. Each step applies these safeguards in order. Derivative estimates enter an exponential in Eq. (2), which can overflow, so they are clipped to $[-5, 5]$ to limit extreme expansion or compression. Non-finite values of the mapped curve are replaced by x_{t+1}. After centering and variance correction, quantile values are clipped to $[-c, c]$ in standardized units to limit extrapolation, with a bound c between 3.5 and 6 that depends on the dataset (Table 4), then sorted to repair numerical crossings. The state x_{t+1} is clipped to the range of the current window's values, and the quantiles are shifted by the same amount. Finally, Eq. (4) subtracts the curve's weighted mean and adds $m_{\psi,t}$. These steps keep the quantiles finite and ordered, but they do not make the quadrature exact.

Cost accounting. A forecast makes $2H$ sequential denoiser calls, two per step: H calls for the state and H for the grid, each carrying KN_{GL} states. With one probe, H reverse-mode vector–Jacobian products differentiate the grid calls. For $K = 33, N_{\mathrm{GL}} = 8$ the forward total is $265H$ window evaluations, or $521H$ for $K = 65$. These counts exclude network-dependent backward costs and the once-per-forecast mean-head evaluation. Batched storage scales with KN_{GL}; memory and

wall-clock time depend on the network and hardware. For sampling-based diffusion forecasters, the corresponding counts depend on whether they denoise each physical step or a whole future window, so counts of reverse steps alone do not give a general runtime ratio.

B.3 Forecast metrics

For a scalar predictive CDF F with finite first moment, $q(u) = \inf\{x : F(x) \geq u\}$ is its quantile and y is one realized outcome. The pinball loss of a forecast error $e = y - q(u)$, $\rho_u(e) = (u - \mathbb{I}\{e < 0\})e$, weights underprediction and overprediction according to the requested level u. The standard CRPS identities are (Matheson & Winkler, 1976; Gneiting & Ranjan, 2011; Zamo & Naveau, 2018)

$$\mathrm{CRPS}(F, y) = \int_{\mathbb{R}} (F(x) - \mathbb{I}\{x \geq y\})^2 \, dx = 2 \int_0^1 \rho_u(y - q(u)) \, du. \tag{23}$$

We approximate the latter integral by the trapezoidal rule on the 33-level metric grid of Appendix B.2. The grid covers $u \in [0.01, 0.99]$, and we do not rescale the result to account for the omitted tails. For QICE, the predicted deciles define ten intervals with outer endpoints $-\infty, +\infty$. If f_b is the pooled fraction of observations in bin b, then the reported percentage-point score is (Han et al., 2022)

$$\mathrm{QICE} = \frac{100}{10} \sum_{b=1}^{10} |f_b - 0.1|. \tag{24}$$

Bins use $(\mathrm{lower}, \mathrm{upper}]$ to assign every outcome once. Scores pool the evaluated windows, leads, and features. Mean squared error (MSE) and mean absolute error (MAE) use the selected point forecast: $m_{\psi,t}$ in the main benchmark and x_t in the characteristic ablation. Tail diagnostics use pinball losses at $0.05, 0.95$, empirical coverage of $[\widehat{q}_t(0.05), \widehat{q}_t(0.95)]$, and its mean width. Coverage must be considered with width and tail loss: a wide interval can attain high coverage without an accurate predictive distribution.

B.4 Data and configuration details

We use Informer's fixed chronological split indices for ETT (Zhou et al., 2021) and non-overlapping evaluation windows, following the NsDiff benchmark (Ye et al., 2025).[1] The remaining datasets use chronological 70%/10%/20% splits and overlapping windows. Normalization is fitted on training data only. Table 3 lists each series' length and its number of evaluated test windows. The ETT test sets contain only 8 or 32 windows, which limits per-window comparisons.

Table 3: Dataset properties and denoiser training settings (Section 3.3; architectures in Appendix B.2). Context and horizon lengths are 168 and 192, except 36 and 36 for ILI. Steps is the full series length, N_{test} counts test windows per seed, and B is the minibatch size. Learning rate (LR) and weight decay (WD) are shown in units of 10^{-3} and 10^{-4}, respectively.

Dataset	d	Steps	N_{test}	Denoiser	B	LR	WD
ETTh1	7	17,420	8	Attention	128	1	1
ETTh2	7	17,420	8	Attention	128	0.8	3
ETTm1	7	69,680	32	Linear	128	1.2	3
ETTm2	7	69,680	32	Attention	128	0.8	3
Electricity	321	26,304	4,897	Linear	4	0.8	1
Exchange	8	7,588	1,156	Linear	128	1	0
ILI	7	966	120	Linear	32	1	0
Solar	137	52,560	10,150	Attention	4	1.2	0
Traffic	862	17,544	3,149	Linear	4	1.2	3

Denoiser training uses Adam for ten epochs, with two minibatches accumulated per update for attention models and one for linear models. Adam's decay factors are $(0.9, 0.96)$, except approximately $(0.9462, 0.9265)$ for ETTm1 and $(0.8633, 0.9477)$ for Exchange. The learning rate is constant at the

[1] Each window holds $T + k + 1$ observed values before the forecast horizon, with the conditioning offset k of Appendix B.2. In our adaptation, each ETT validation and test split starts $T + H$ steps before its boundary, so the first forecast of a split has $H - k - 1$ targets before the boundary.

value listed in Table 3. An exponential moving average of the model weights is updated every ten optimizer steps with decay 0.995 (0.9995 for ETTm1). After each epoch, the averaged weights are scored on validation data, with the characteristic as the location. Algorithm 1 then selects the epoch with a QICE cap of $\tau_Q = 5$ percentage points. Table 4 collects the transport settings.

The mean head uses Adam with learning rate 10^{-3} for ten epochs and is selected by validation MSE. It is reused when varying the denoiser seed. The architecture comparison in Appendix C.2 additionally varies the mean-head training seed.

Table 4: Transport settings. Noise estimators are defined in Appendix B.1; λ_J weights Eq. (5), and α blends the variance correction in Eq. (22). The grid size K and clipping range $[-c, c]$ follow Appendix B.2. All use $N_{\mathrm{GL}} = 8$ quadrature nodes and one inference probe.

Dataset	Noise estimator	K	λ_J	α	c
ETTh1	cumulative	65	10^{-5}	1	3.5
ETTh2	innovation blend	33	10^{-5}	1	4.5
ETTm1	innovation blend	65	10^{-4}	1	6
ETTm2	innovation blend	33	10^{-5}	0	4.5
Electricity	innovation blend	33	0	1	4.5
Exchange	innovation blend	33	10^{-5}	1	3.5
ILI	sliding	33	0	0	3.5
Solar	innovation blend	33	10^{-4}	1	3.5
Traffic	innovation blend	33	10^{-5}	1	4.5

C Evidence for Transport, Location, and Uncertainty

The supporting experiments follow the three questions raised in the main text: whether scalar transport approximates the learned flow (Appendix C.1), how location affects forecasting (Appendix C.2), and what the resulting quantile curves offer beyond fixed-grid prediction (Appendix C.3). Appendix C.4 then examines interval coverage and numerical limits. In result tables, bold marks the lowest mean within a dataset and metric.

C.1 Sampling the same learned flow

The reference construction in Section 4.2 is evaluated on every test window of ETTh1, ETTh2, ETTm1, ETTm2, and ILI, with three frozen denoiser seeds. Each of the $2{,}048$ states contains all updated time positions and variables, together with the observed first window position and the conditioning window, which all states share. The full denoiser propagates these states jointly across coordinates. At every step, its empirical marginals receive the same corrections as QFLOW (Appendix B.2): centering on the characteristic, variance rescaling, clipping, and the window-range correction of the state. Both methods retain diagonal schedules.

For each forecast window, the centered W_1 between the curves of QFLOW and the reference, $\int_0^1 |r_t(u) - r_t^{\mathrm{ref}}(u)|\, du$, is averaged over variables and leads and divided by the reference's spread $\int_0^1 |r_t^{\mathrm{ref}}(u)|\, du$, averaged over the same entries. Reported percentages average these per-window ratios, then the seeds. The diamonds apply the same calculation to the two groups of $1{,}024$ states, with the second group providing the denominator. Because the groups share their initial levels and ensemble corrections, this half-to-half difference is a descriptive scale rather than a confidence bound.

Using sixteen Jacobian probes per derivative estimate instead of one changes the relative shape discrepancy by less than 0.1 percentage points, for example from 4.08% to 4.03% on ETTm1. This indicates low sensitivity to the probe count for these trained denoisers.

Because this comparison looks at the forecast distributions of the learned flow rather than at velocities, it does not measure the integrated velocity discrepancy $\int_0^H \delta_t\, dt$ of Theorem A.1 or the error relative to the data distribution. It also leaves the diagonal schedule unchanged, so it does not test the effect of full covariance modeling. Electricity, Exchange, Solar, and Traffic were left out of this comparison because of its computational cost.

C.2 Location accuracy and variation across windows

Table 5 isolates the learned location by holding the transported shape fixed. The characteristic already gives competitive forecasts (Section 4.1), but replacing it with $m_{\psi,t}(\mathcal{C})$ lowers average CRPS and MSE on all nine datasets. An average can hide windows that get worse, so Figure 4 compares the two locations window by window.

Table 5: Location ablation with a fixed transported shape: the characteristic x_t from Eq. (1) versus the learned mean $m_{\psi,t}$ in Eq. (4). Entries are three-seed means; bold denotes the lowest mean within each dataset and metric.

	CRPS		MSE	
Dataset	Characteristic	Learned	Characteristic	Learned
ETTh1	0.376	**0.325**	0.549	**0.405**
ETTh2	0.365	**0.341**	0.492	**0.457**
ETTm1	0.340	**0.282**	0.471	**0.290**
ETTm2	0.306	**0.268**	0.368	**0.266**
Electricity	0.337	**0.208**	0.466	**0.174**
Exchange	0.318	**0.234**	0.300	**0.160**
ILI	1.522	**1.116**	6.967	**3.513**
Solar	0.773	**0.304**	1.652	**0.268**
Traffic	0.443	**0.284**	1.155	**0.481**

For each test window and denoiser seed, the paired ratio divides the score with the learned location by the score with the characteristic. Gains are consistent on Electricity, Solar, and Traffic, while the broader ranges on the ETT datasets reveal windows that become worse. In particular, the median ETTh2 ratio is near one despite an improvement in average loss. The figure pools three denoiser seeds with one fixed mean head, so its spread describes variation across windows and transported shapes, not independent replications of all learned components.

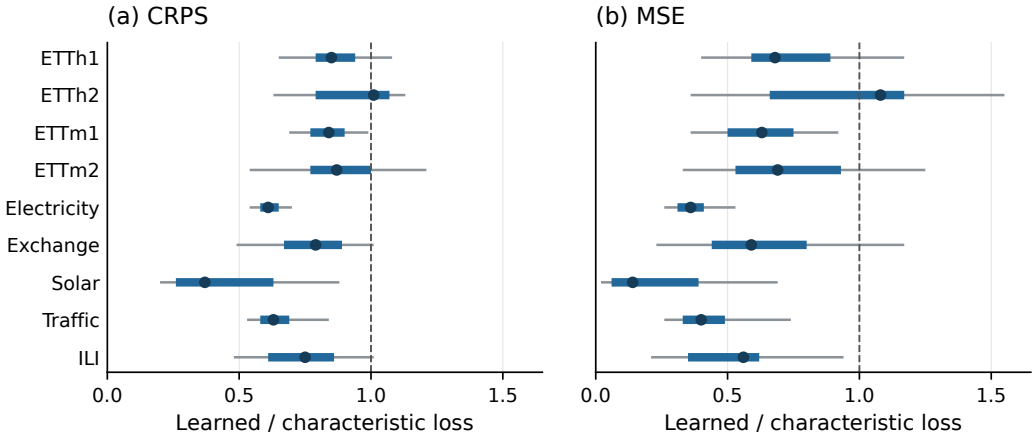

Figure 4: **Location gains across test windows.** Paired learned/characteristic score ratios across test windows and three denoiser seeds. Dots show medians, thick segments the 25th–75th percentiles, and thin segments the 5th–95th percentiles. Ratios below one favor the learned location. These are distributions of paired outcomes, not confidence intervals. Table 3 gives the test-window counts.

Changing the location architecture. Replacing the linear mean head with an NsDiff-style Transformer on ETTm2 leaves the denoiser and transport fixed. We use ETTm2 because, apart from the short ILI series, it has the lowest interval coverage in Figure 6. Across three independently trained head seeds, CRPS is 0.2738 ± 0.0103 for the linear head and 0.2917 ± 0.0027 for the Transformer. The former uses $32{,}448$ parameters versus 7.41M and takes 0.0117 versus 0.549 ms per window on the same NVIDIA V100 GPU. So the transport works unchanged with a different location architecture, although the larger head is less accurate here. Timings measure the location branch only.

C.3 FITTED-GRID ACCURACY AND PREDICTIONS AT NEW QUANTILE LEVELS

Baselines and fitting objectives. The fixed-residual baseline adds the quantiles of validation residuals to a linear mean head (Section 4.3). In Figure 3, it uses QFLOW's fitted mean head, and in Table 6, each of three seeds fits its own head. The offsets do not depend on the context, and their mean can differ from zero. The linear quantile-regression head has the mean head's architecture, one temporal map shared across variables, but 33 outputs per lead, one for each level. It fits these outputs by pinball loss and sorts them at evaluation. Both baselines are fitted without test targets.

Table 6 reports their aggregate performance. Linear quantile regression outperforms every diffusion forecaster in Table 1 on eight datasets in CRPS and all nine in QICE. NsDiff keeps the best ILI CRPS. Fixed residuals also outperform transport on five datasets. These comparisons show where transported uncertainty needs improvement, but objective and capacity both favor quantile regression. It is fitted by the pinball loss at the 33 evaluated levels, the loss that CRPS averages, while QFLOW's networks are fitted by denoising and by mean squared error. A K-level linear head also has $KH(T+1)$ parameters, about 1.07M for 33 levels with $T = 168$ and $H = 192$. QFLOW's mean head and denoiser together have 32.7k parameters with the attention denoiser on ETTh1, ETTh2, and ETTm2, 108k with the attention denoiser on Solar, and 60.8k with the linear denoiser on ETTm1, Electricity, Exchange, and Traffic, so the quantile-regression head is 10 to 33 times larger. On ILI, where $T = H = 36$, the counts are 2.7k and 44k.

Table 6: Marginal quantile baselines defined in Appendix C.3, with metrics from Appendix B.3: mean ± sample standard deviation over three seeds. QFLOW varies the denoiser seed with one fixed mean head. Bold denotes the lowest mean within each dataset and metric; deviations describe seed variation, not significance. QICE is in percentage points.

	CRPS ↓			QICE ↓	
Dataset	QFLOW	Linear QR	Fixed residual	QFLOW	Linear QR
ETTh1	0.3246 ± 0.0004	**0.2938** ± 0.0008	0.3524 ± 0.0034	2.112 ± 0.040	**1.189** ± 0.023
ETTh2	0.3409 ± 0.0044	**0.2998** ± 0.0019	0.3416 ± 0.0015	4.139 ± 0.187	**1.739** ± 0.090
ETTm1	0.2816 ± 0.0005	**0.2466** ± 0.0016	0.2757 ± 0.0013	3.662 ± 0.047	**1.493** ± 0.044
ETTm2	0.2681 ± 0.0006	**0.2328** ± 0.0023	0.2502 ± 0.0020	5.331 ± 0.054	**1.366** ± 0.110
Electricity	0.2078 ± 0.0000	**0.1881** ± 0.0008	0.2121 ± 0.0012	2.107 ± 0.000	**0.382** ± 0.027
Exchange	0.2342 ± 0.0018	**0.2314** ± 0.0086	0.2595 ± 0.0039	3.169 ± 0.718	**2.087** ± 0.178
ILI	1.1157 ± 0.0002	**0.9169** ± 0.0011	1.0447 ± 0.0060	7.193 ± 0.001	**5.526** ± 0.088
Solar	0.3037 ± 0.0001	**0.1871** ± 0.0007	0.2807 ± 0.0044	6.456 ± 0.017	**5.560** ± 0.136
Traffic	0.2844 ± 0.0001	**0.2145** ± 0.0004	0.2651 ± 0.0014	4.895 ± 0.004	**0.901** ± 0.035

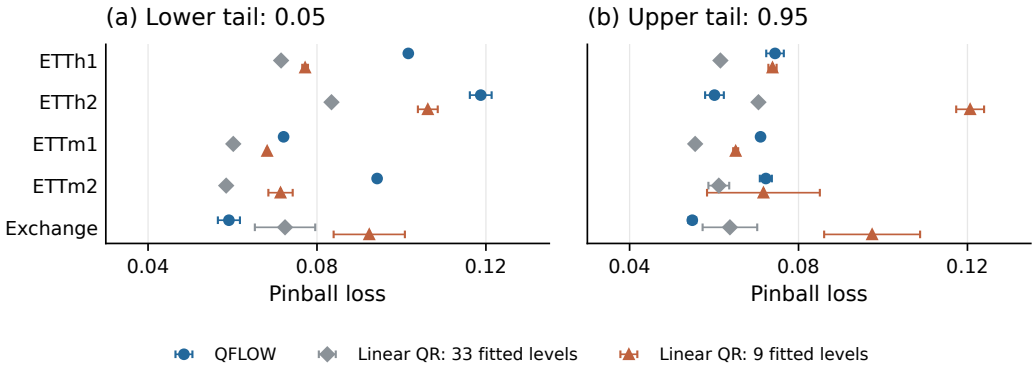

Figure 5: **Accuracy at additional tail levels.** Pinball loss at 0.05 and 0.95 for QFLOW and linear quantile-regression heads fitted on 9 or 33 levels (lower is better). Both quantile-regression heads interpolate these tails. QFLOW uses its existing denoiser and location weights. Points and error bars show three-seed means and sample standard deviations.

Predicting levels outside a fitted grid. A denoising objective does not assign model outputs to fixed probability levels. Once QFLOW is trained, we can choose the number and positions of

transport levels, or interpolate its computed curve onto a requested output grid, without changing the learned weights. Increasing the transport resolution costs additional numerical work but requires no refitting.

A quantile-regression head, in contrast, must interpolate new levels from its fitted outputs unless it is refitted. We assess this behavior by training linear quantile regression on nine equally spaced levels in $[0.01, 0.99]$, then evaluating on the standard 33-level grid. The median is fitted in both designs, while 0.05 and 0.95 are obtained through interpolation. On the five datasets of Figure 5, reducing the fitted grid increases lower-tail loss by 8–28% and upper-tail loss by 17–71%, compared with only 1–2% at the median.

Figure 5 compares these tail predictions with QFLOW using its existing weights. QFLOW has lower upper-tail loss on ETTh2 and Exchange than either quantile-regression head, and lower lower-tail loss on Exchange. The quantile-regression heads remain stronger on several other tails. So a quantile-regression head's tail accuracy depends on its fitted grid, while QFLOW's advantage at these new levels holds on some datasets only. The nine-level refit also reduces the quantile-regression head's parameter count, so the comparison does not isolate interpolation from capacity.

C.4 Coverage and numerical limits

To examine uncertainty directly, we form the central 90% prediction interval $[\widehat{q}_t(0.05), \widehat{q}_t(0.95)]$, leaving a nominal 5% probability in each tail. *Coverage* is the fraction of held-out observations inside these intervals: a calibrated 90% interval should contain about nine out of ten outcomes. Lower coverage means more outcomes fall outside than predicted. *Width* is the endpoint difference $\widehat{q}_t(0.95) - \widehat{q}_t(0.05)$, averaged over forecasts. Narrower intervals give more precise predictions when coverage is retained; widening them can improve coverage at the expense of precision.

Figure 6 therefore reports both quantities. QFLOW's coverage ranges from 79.7% to 89.3% on seven datasets but falls to 61.6% on ETTm2 and 50.8% on ILI, whose intervals are far too narrow. Exchange shows the useful case: its interval is narrower than linear quantile regression's while its coverage stays close to nominal. Together with the tail losses in Figure 5, these measurements show when a narrower interval is informative and when it misses outcomes.

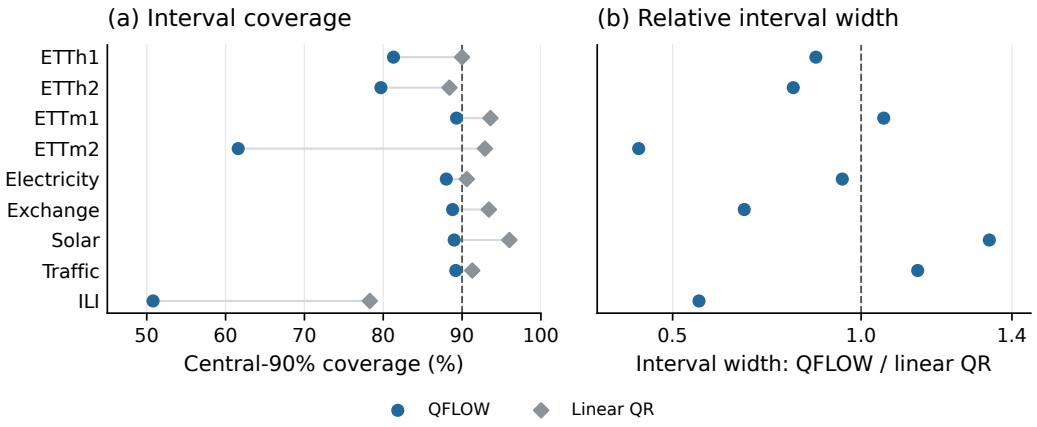

Figure 6: **Coverage and interval width.** Left: empirical central-90% coverage, with the nominal value marked. Right: QFLOW interval width divided by linear quantile regression's, with equal width marked. Values below one indicate narrower intervals. All values are three-seed means.

Variance correction and clipping. Variance rescaling (Eq. (22)) adjusts width, and clipping (Appendix B.2) limits tails. On ETTh1 with the innovation-blend noise estimator, a configuration separate from the benchmark one, full rescaling ($\alpha = 1$ instead of 0) lowers CRPS from 0.364 to 0.338 and QICE from 3.545 to 1.773. On ILI, clipping affects 0.004% of validation quantiles but 14.7% of test quantiles. Removing it raises test coverage from 50.8% to 67.4% and lowers CRPS from 1.116 to 1.076, explaining part of the undercoverage. Clipping can also move the mean, so Eq. (4) centers the final curve after all corrections.