

PROBABILISTIC FORECASTING VIA AUTOREGRESSIVE FLOW MATCHING

Anonymous authors

Paper under double-blind review

ABSTRACT

In this work, we introduce autoregressive flow matching (AFM) for probabilistic forecasting of multivariate timeseries data. Given historical measurements and optional future covariates, we formulate forecasting as sampling from a learned conditional distribution over future trajectories. Specifically, we decompose the joint distribution of future observations into a sequence of conditional densities, each modeled via a shared flow that transforms a simple base distribution into the next observation distribution, conditioned on observed covariates. To achieve this, we leverage the flow matching framework, enabling scalable and simulation-free learning of these transformations. By combining this factorization with the flow matching objective, AFM retains the benefits of classical autoregressive models—including strong extrapolation performance, compact model size, and well-calibrated uncertainty estimates—while also capturing complex multi-modal conditional distributions, as seen in modern transport-based generative models. We demonstrate the effectiveness of AFM on multiple stochastic dynamical systems and real-world forecasting tasks.

1 INTRODUCTION

A core problem in modern machine learning is probabilistic timeseries forecasting, where the aim is to extrapolate how system dynamics evolve into the future given observational data. This problem is central to a wide range of scientific, industrial and societal disciplines (Lim & Zohren, 2021; Dama & Sinoquet, 2021; Ye et al., 2024).

An emerging trend is to leverage deep generative models to tackle this problem (Karl et al., 2016; Rasul et al., 2020; Desai et al., 2021). In this setting, forecasting is framed as sampling from a future probability density conditioned on the past. Most notably, diffusion models and score-based generative models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020) have recently pushed state-of-the-art performance in multiple forecasting benchmarks (Rasul et al., 2021; Tashiro et al., 2021; Kollovich et al., 2023; Meijer & Chen, 2024). Despite their impressive performance, diffusion models typically come with high computational costs during training and inference.

Flow matching (FM) (Liu et al., 2022; Albergo et al., 2023; Lipman et al., 2022) is an emerging paradigm for generative modeling that generalizes and subsumes diffusion models while offering more flexible design choices. Unlike diffusion models, which rely on iterative stochastic denoising steps over a discretized trajectory, FM learns deterministic probability paths that transforms arbitrary base distributions into the target distribution directly through continuous normalizing flows. By optimally designing these probability paths, FM circumvents the need for handcrafted noise schedules and lengthy ancestral sampling chains inherent to diffusion, enabling more efficient training and sampling.

Recently, FM has been applied in the context of timeseries modeling and probabilistic forecasting (Tamir et al., 2024; Hu et al., 2024; Kollovich et al., 2024), showing strong empirical results and improved computational efficiency compared to diffusion models. However, current approaches rely on directly learning the conditional distribution of an entire fixed-horizon future window conditioned on a fixed-horizon context window. While this leads to fast training and sampling, it also results in models that extrapolate poorly beyond the training distribution, and miscalibrated uncertainty estimates. Additionally, it introduces a more complex optimization problem, where models attempt to learn intricate time-dependent probability paths along the flow dimension. The latter issue

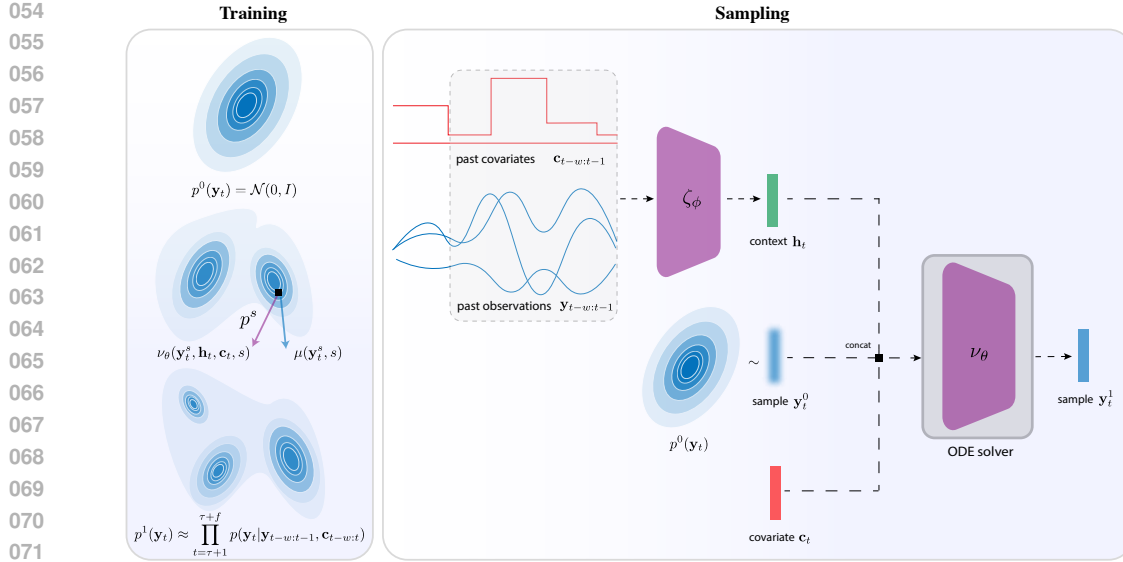


Figure 1: An overview of the training and sampling process of our approach. During training, a probability path $(p^s)_{0 \leq s \leq 1}$ is constructed between a base distribution p^0 and the target distribution p^1 . This probability path is generated by a velocity field μ . Training is done by regressing μ via a neural network ν_θ which takes in a sample from the probability path y_t^s at flow step s , a context vector h_t encoding past observations and covariates, and the current covariate c_t . After training, sampling from the target distribution is achieved via first sampling the base distribution p^0 , and integrating the trained velocity field via a ODE solver until $s = 1$.

is highlighted in (Kollovieh et al., 2024), where the authors propose to use informed priors based on Gaussian processes to simplify and speed up optimization. However, this restricts their approach to univariate forecasting problems.

In this work, we propose autoregressive flow matching (AFM) as a simple alternative that utilizes FM to learn probabilistic forecasting models that scale to high-dimensional multivariate datasets without relying on informed priors. Unlike existing FM-based methods that model the entire future window simultaneously, AFM decomposes the forecasting problem into a sequence of conditional distributions, modeling the distribution of each future time step conditioned on past observations and covariates. This is achieved using a shared flow that transforms a simple base distribution into the next observation’s distribution. This autoregressive structure enables AFM to naturally handle variable forecasting horizons, extrapolate beyond the training distribution, and provide well-calibrated uncertainty estimates, while simplifying the optimization problem.

We demonstrate that AFM achieves superior or competitive performance on both classical dynamical systems and real-world forecasting tasks, offering a practical and effective solution to probabilistic timeseries forecasting.

2 BACKGROUND: FLOW MATCHING

Flow matching (Lipman et al., 2022) is a paradigm for generative modeling that enables training of continuous normalizing flows (CNFs) (Chen et al., 2018; Grathwohl et al., 2018) at an unprecedented scale. CNFs model data transformations as solutions to ordinary differential equations (ODEs), providing invertible mappings with tractable likelihoods.

Similar to CNFs, FM aims to learn a time-dependent diffeomorphic map defined on the data space Ω , called a flow $\psi: [0, 1] \times \Omega \rightarrow \Omega$. This flow transforms a sample $\mathcal{X}^0 \sim p$ from a source distribution p into a target sample $\mathcal{X}^1 := \psi(\mathcal{X}^0, 1)$ such that $\mathcal{X}^1 \sim q$ for some target distribution q . The flow is constructed by solving the following initial value problem:

$$d\psi(\mathcal{X}, s) = \mu(\psi(\mathcal{X}, s), s)ds, \quad \psi(\mathcal{X}, 0) = \mathcal{X}, \quad s \in [0, 1] \quad (1)$$

where $\mu: [0, 1] \times \Omega \rightarrow \Omega$ is a vector field defining the velocity of the flow and generating a probability path $(p^s)_{0 \leq s \leq 1}$ where each p^s is a distribution over Ω with $p^0 = p$. The objective is thus to learn a valid vector field μ such that $p^1 = q$.

Unlike CNFs where the vector field is learned via likelihood maximization, requiring solving and differentiating through Eq. (1) during training, FM learns a vector field ν_θ with parameters θ by regressing over vector fields of fixed conditional probability paths. This enables defining a tractable objective function called conditional flow matching (CFM) defined as:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{s \sim \mathcal{U}[0,1], \mathcal{X}^1 \sim q(\mathcal{X}^1), \mathcal{X} \sim p^s(\mathcal{X} | \mathcal{X}^1)} \left\| \nu_\theta(\mathcal{X}, s) - \mu(\mathcal{X}, s | \mathcal{X}^1) \right\|^2. \quad (2)$$

Minimizing this objective allows sampling from q by solving Eq. (1) with the trained ν_θ . By avoiding gradient backpropagation through the ODE solver, FM enables stable and scalable training for high-dimensional generative tasks, such as image and video modeling (Esser et al., 2024; Polyak et al., 2024). Various probability path designs and conditioning strategies have been proposed, unifying multiple transport-based generative models under this framework (Liu et al., 2022; Albergo et al., 2023; Tong et al., 2023).

3 AUTOREGRESSIVE FLOW MATCHING

Problem Setting Let $\mathbf{y}_\tau \in \mathbb{R}^n$ denote an n -dimensional observation at time τ . Given access to its l past measurements including the current observation, denoted as $\mathbf{Y}_l = \{\mathbf{y}_{\tau-l}, \dots, \mathbf{y}_\tau\}$, our objective is to provide probabilistic forecasts of the next f future values denoted as $\mathbf{Y}_f = \{\mathbf{y}_{\tau+1}, \dots, \mathbf{y}_{\tau+f}\}$, potentially conditioned on observed covariates $\mathbf{C} = \{\mathbf{c}_{\tau-l}, \dots, \mathbf{c}_{\tau+f}\}$ where $\mathbf{c}_\tau \in \mathbb{R}^c$. Formally, our goal is to learn to sample from the conditional distribution $p(\mathbf{Y}_f | \mathbf{Y}_l, \mathbf{C})$ using a dataset $\mathcal{D} = \{(\mathbf{Y}_f^i, \mathbf{Y}_l^i, \mathbf{C}^i)\}_{i=1}^m$ with m the number of instances.

We propose to factorize the conditional distribution autoregressively across future time-steps under a Markov assumption of order w such that:

$$p(\mathbf{Y}_f | \mathbf{Y}_l, \mathbf{C}) = \prod_{t=\tau+1}^{\tau+f} p(\mathbf{y}_t | \mathbf{y}_{t-w:t-1}, \mathbf{c}_{t-w:t}) \quad (3)$$

where $w \leq l$ is the history size. Our objective thus becomes to learn the conditional distribution in Eq. (3). Instead of maximum likelihood estimation, common for autoregressive models, we employ the flow matching framework to learn this conditional distribution.

This autoregressive factorization provides several advantages. First, it reduces the learning problem to modeling simpler conditional distributions $p(\mathbf{y}_t | \mathbf{y}_{t-w:t-1}, \mathbf{c}_{t-w:t})$ rather than a complex high-dimensional joint distribution. Second, each time step’s conditional distribution can be trained within a teacher forcing framework allowing parallel training. Third, the Markov assumption with window size w offers a trade-off between model capacity and computational complexity. Finally, this formulation enables interpretable uncertainty quantification by explicitly modeling how prediction uncertainty propagates through time via the chain of conditional distributions.

3.1 CONSTRUCTING A TARGET PROBABILITY PATH

For all future time steps $t \in [\tau + 1, \tau + f]$, we define a target probability path $(p^s(\mathbf{y}_t))_{0 \leq s \leq 1}$, with marginals $p^0(\mathbf{y}_t) = \mathcal{N}(0, I)$ and $p^1(\mathbf{y}_t) \approx \prod_{t=\tau+1}^{\tau+f} p(\mathbf{y}_t | \mathbf{y}_{t-w:t-1}, \mathbf{c}_{t-w:t}) \approx p_{\mathcal{D}}$ where $p_{\mathcal{D}}$ is the empirical distribution. This probability path is generated by a flow field $\psi: [0, 1] \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ along s defined via the ordinary differential equation (ODE):

$$d\psi(\mathbf{y}_t, s) = \mu(\psi(\mathbf{y}_t, s), s) ds, \quad \psi(\mathbf{y}_t, 0) = \mathbf{y}_t \quad s \in [0, 1] \quad (4)$$

where $\mu: [0, 1] \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a vector field constructing the flow chosen such that the resulting flow satisfy the continuity equation.

To derive a tractable objective, we can model the marginal probability path as a mixture of conditional probability paths:

$$p^s(\mathbf{y}_t) = \int p^s(\mathbf{y}_t | \mathbf{z}) p(\mathbf{z}) d\mathbf{z} \quad (5)$$

where $\mathbf{z} \sim \pi_{0,1}(\mathbf{z})$ is a conditioning random variable sampled from an arbitrary data coupling $\pi_{0,1}$. Theorem 2 in (Lipman et al., 2022) shows that the conditional velocity fields $\mu(\mathbf{y}_t, s | \mathbf{z})$ associated with the conditional probability paths $p^s(\mathbf{y}_t | \mathbf{z})$ are equivalent to their respective marginal velocity field $\mu(\mathbf{y}_t, s)$.

Following (Tong et al., 2023), we choose $\mathbf{z} = \{\mathbf{y}_t^0, \mathbf{y}_t^1\}$ with $p(\mathbf{z} = (\mathbf{y}_t^0, \mathbf{y}_t^1)) = p^0 \times p^1$, where \mathbf{y}_t^0 and \mathbf{y}_t^1 are samples from p^0 and p^1 , respectively. We define the conditional probability path as:

$$p^s(\mathbf{y}_t | \mathbf{z}) = \mathcal{N}((1-s)\mathbf{y}_t^0 + s\mathbf{y}_t^1, \sigma^2 I) \quad (6)$$

with marginals satisfying $p^0(\mathbf{y}_t | \mathbf{z}) = p^0(\mathbf{y}_t)$ and $p^1(\mathbf{y}_t | \mathbf{z}) = p^1(\mathbf{y}_t)$ for $\sigma^2 \rightarrow 0$. The respective conditional velocity field is then simply $\mu(\mathbf{y}_t, s | \mathbf{z}) = \mathbf{y}_t^1 - \mathbf{y}_t^0$. This choice offers straight probability paths from the source to the target distributions in Euclidean space and closed-form solutions, enabling efficient training and sampling. Learning this velocity field allows us to sample from our target distribution $p(\mathbf{y}_t | \mathbf{y}_{t-w:t-1}, \mathbf{c}_{t-w:t})$ by solving the ODE in Eq. (4) until $s = 1$.

3.2 TRAINING

Similar to the standard FM setting, we learn this velocity field by regressing it against a neural network ν_θ with parameters θ using the conditional flow matching objective in Eq. 2. To efficiently achieve this in our setting, we define a context vector $\mathbf{h}_t \in \mathbb{R}^h$ that encodes the context window at time t defined as:

$$\mathbf{h}_t = \zeta_\phi(\mathbf{y}_{t-w:t-1}, \mathbf{c}_{t-w:t-1}) \quad (7)$$

where $\zeta: \mathbb{R}^{w \times n} \times \mathbb{R}^{w \times c} \rightarrow \mathbb{R}^h$ is a neural network with parameters ϕ . We can then define our learning objective as:

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{\mathbf{z} \sim \pi_{0,1}, s \sim \mathcal{U}(0,1), \mathbf{y}_t^s \sim p^s(\mathbf{y}_t | \mathbf{z})} \|\mu(\mathbf{y}_t^s, s | \mathbf{z}) - \nu_\theta(\mathbf{y}_t^s, \mathbf{h}_t, \mathbf{c}_t, s)\|^2 \quad (8)$$

which we train by randomly sampling an observation at time point t , its associated context window, and covariates from \mathcal{D} and jointly optimize the parameters ϕ and θ by minimizing $\mathcal{L}(\theta, \phi)$ via stochastic gradient descent. Note that this is akin to teacher forcing (Williams & Zipser, 1989), where ground-truth past observations are fed into the model during training (rather than its own past predictions), allowing parallel training across all time-steps and avoiding error accumulation. Algorithm 1 summarizes the training procedure for autoregressive flow matching.

Algorithm 1 Training: Autoregressive Flow Matching.

Require: Dataset $\mathcal{D} = \{(\mathbf{Y}_f^i, \mathbf{Y}_l^i, \mathbf{C}^i)\}_{i=1}^m$, window size w , networks ν_θ, ζ_ϕ

- 1: **while** not converged **do**
 - 2: Sample $\mathbf{y}_t^0 \sim \mathcal{N}(0, I)$ and $(\mathbf{y}_t^1, \mathbf{y}_{t-w:t-1}, \mathbf{c}_{t-w:t}) \sim \mathcal{D}$
 - 3: Set $\mathbf{h}_t = \zeta_\phi(\mathbf{y}_{t-w:t-1}, \mathbf{c}_{t-w:t-1})$
 - 4: Sample $s \sim \mathcal{U}(0, 1)$ and $\mathbf{y}_t^s \sim \mathcal{N}((1-s)\mathbf{y}_t^0 + s\mathbf{y}_t^1, \sigma^2 I)$
 - 5: Compute target $\mu(\mathbf{y}_t^s, s | \mathbf{z}) = \mathbf{y}_t^1 - \mathbf{y}_t^0$
 - 6: Compute loss $\mathcal{L}_t = \|\mu(\mathbf{y}_t^s, s | \mathbf{z}) - \nu_\theta(\mathbf{y}_t^s, \mathbf{h}_t, \mathbf{c}_t, s)\|^2$
 - 7: Update θ, ϕ
 - 8: **end while**
 - 9: **return** ν_θ, ζ_ϕ
-

3.3 INFERENCE

After training, we wish to provide probabilistic predictions for new observations \mathbf{Y}_l and corresponding covariates \mathbf{C} for f future timestamps into the future. For this we follow the sampling procedure in Algorithm 2, where predictions are made autoregressively using a rolling-window approach. This procedure can be repeated many times to obtain empirical quantiles of the uncertainty of our predictions.

4 EXPERIMENTS

In this section, we present our empirical results and compare AFM against various baselines using real world datasets and data generated via simulating various dynamical systems.

Algorithm 2 Sampling: Autoregressive Flow Matching.

Require: Past observations \mathbf{Y}_l , covariates \mathbf{C} , forecast horizon f , window size w , networks ν_θ, ζ_ϕ

- 1: Initialize $\mathbf{Y}_f \leftarrow \mathbf{Y}_l$
- 2: **for** $t = \tau + 1$ **to** $\tau + f$ **do**
- 3: Set context $\mathbf{h}_t = \zeta_\phi(\mathbf{y}_{t-w:t-1}, \mathbf{c}_{t-w:t-1})$ using the last w entries in \mathbf{Y}_f
- 4: Sample $\mathbf{y}_t^0 \sim \mathcal{N}(0, I)$
- 5: Solve $\frac{d\psi(\mathbf{y}_t, s)}{ds} = \nu_\theta(\psi(\mathbf{y}_t, s), \mathbf{h}_t, \mathbf{c}_t, s)$ with $\psi(\mathbf{y}_t, 0) = \mathbf{y}_t^0$ and $s \in [0, 1]$
- 6: Append $\psi(\mathbf{y}_t, 1)$ to \mathbf{Y}_f
- 7: **end for**
- 8: **return** \mathbf{Y}_f

4.1 DYNAMICAL SYSTEMS

Data We evaluated AFM on the task of forecasting five different stochastic dynamical systems: stochastic versions of the Lorenz system (Lorenz, 1963), FitzHugh–Nagumo model (FitzHugh, 1961), Lotka–Volterra system Lotka (1925), Brusselator (Lefever & Prigogine, 1968), and Van der Pol oscillator (van der Pol, 1926). For each system, we generated trajectories by numerically solving the corresponding stochastic differential equations, using randomly sampled initial conditions and Brownian motion realizations. The details of the dynamical systems and the data generation process are provided in Appendix A.1.

Implementation details For these experiments we set ζ_ϕ as a two-layer bi-directional LSTM with 64 hidden units, and set ν_θ as a multi-layer perceptron with three hidden layers of size 64. We use Fourier positional embeddings to encode the flow step s into a 16-dimensional vector using a fixed set of frequencies and set the context length w to be equal to the prediction length. We trained our models via stochastic gradient descent using an Adam optimizer with a learning rate of 0.003 and a batch size of 128. Training until convergence took ~ 7 minutes on a single A100-SXM4-40GB NVIDIA GPU.

Baselines To evaluate the effectiveness of the autoregressive factorization in AFM, we compared our approach against a non-autoregressive flow-matching baseline, which directly models the joint distribution of future observations conditioned on past observations and covariates, that is, $p^1 = p(\mathbf{Y}_f | \mathbf{Y}_l, \mathbf{C})$. This approach serves as a direct ablation, isolating the effect of autoregressive factorization introduced in AFM. Note that this non-autoregressive formulation closely resembles the method proposed in (Kollovich et al., 2024). Further details on this baseline are provided in Appendix B.

Evaluation To assess the model’s ability to extrapolate beyond its training distribution, we partitioned each trajectory into three segments: (1) an observation window, which serves as the historical input to the model; (2) a prediction window, which is available only during training to act as the target; and (3) an extrapolation window, which remains entirely unseen during training. We used the continuous rank probability score (CRPS) (Winkler et al., 1996) for evaluating the performance of the model in uncertainty quantification and the normalized root mean square error (NRMSE) (Hindman & Koehler, 2006) to assess the accuracy of the predictions. See Appendix C for more details about the employed evaluation metrics.

Results Our results, summarized in Table 1, demonstrates that the autoregressive factorization employed in AFM significantly improves forecasting performance across multiple stochastic dynamical systems. For the prediction task, the AR approach outperforms the FM baseline in terms of NRMSE across all five systems, with particularly dramatic improvements observed in the Lorenz (0.017 vs 0.242) and Brusselator (0.031 vs 0.658) systems, representing reductions in error of 93% and 95%, respectively. The AR factorization also yields superior CRPS in four of the five systems, with only the Van der Pol oscillator showing better uncertainty quantification with the FM approach. More importantly, these improvements extend to the extrapolation regime, where autoregressive factorization maintains its advantage in NRMSE across all systems, demonstrating its superior ability to generalize beyond the training distribution. This is particularly evident in the Brusselator system, where AR factorization reduces the extrapolation NRMSE by 98% (0.023 vs 1.046).

Table 1: Performance comparison of autoregressive (AFM) vs non-autoregressive (FM) factorization in our method for five different stochastic dynamical systems. Performance is compared in terms of CRPS and NRMSE for both prediction and extrapolation regimens on the test set. Reported results are the mean and standard deviation across five different random seeds.

System	Factorization	Prediction		Extrapolation	
		CRPS ↓	NRMSE ↓	CRPS ↓	NRMSE ↓
Lorenz	FM	0.147±0.012	0.242±0.018	0.120±0.009	0.301±0.024
	AFM	0.090±0.008	0.017±0.003	0.137±0.011	0.048±0.006
FitzHugh-Nagumo	FM	0.165±0.014	0.131±0.011	0.214±0.016	0.306±0.022
	AFM	0.144±0.013	0.090±0.007	0.146±0.012	0.278±0.019
Lotka-Volterra	FM	0.130±0.011	0.266±0.019	0.136±0.012	0.502±0.033
	AFM	0.126±0.010	0.189±0.015	0.123±0.009	0.339±0.026
Brusselator	FM	0.217±0.016	0.658±0.042	0.238±0.019	1.046±0.057
	AFM	0.125±0.011	0.031±0.004	0.109±0.008	0.023±0.003
Van der Pol	FM	0.143±0.012	0.161±0.014	0.211±0.018	0.510±0.031
	AFM	0.225±0.017	0.051±0.005	0.282±0.021	0.101±0.009

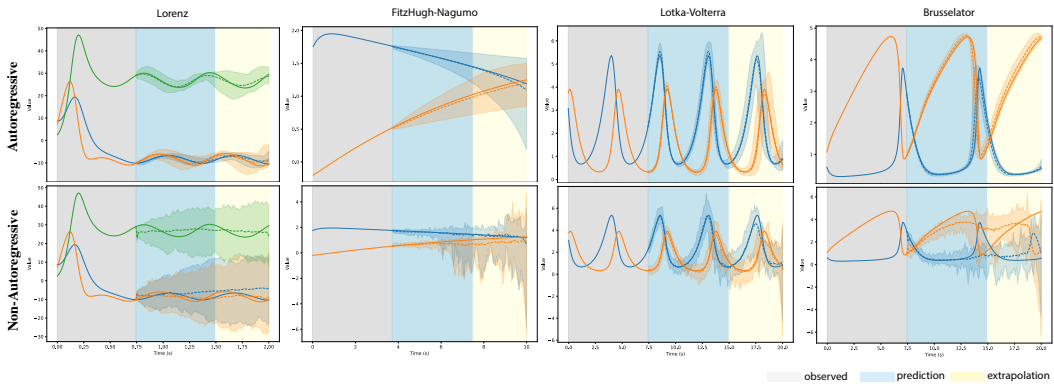


Figure 2: Samples from the forecasting results for autoregressive flow matching vs standard flow matching on four different dynamical systems. The solid lines indicate ground truth, while the dashed lines indicated the mean prediction along with the 95% confidence interval. The results are visualized during both prediction and extrapolation regimens.

Figure 2 illustrates the forecasting results for AFM compared to FM by generating future trajectories for four dynamical systems conditioned on observed data. AFM is shown to have much lower variance compared to the FM baseline, even allowing for accurate extrapolation of the dynamics beyond the prediction window it was trained on. These results confirm that modeling the temporal dependencies through autoregressive factorization enables AFM to better capture the underlying dynamics of complex stochastic systems, leading to more accurate predictions and improved uncertainty quantification, especially in challenging extrapolation scenarios.

4.2 REAL-WORLD DATASETS

Data We evaluated AFM on the tasks of forecasting multiple univariate and multivariate timeseries with different frequencies (hourly and daily) from GluonTS (Alexandrov et al., 2020). Specifically, we used the following datasets: Electricity (Lai et al., 2018), Solar (Lai et al., 2018), Exchange (Dua et al., 2017), Traffic (Dua et al., 2017), and Wikipedia (Gasthaus et al., 2019). We provide further details in Appendix A.2.

Implementation details We set ζ_ϕ as a three-layer bi-directional LSTM with 64 hidden units. We defined ν_θ via a residual neural network with five residual blocks implemented via 1-D convolutional layers with gated activation functions (Van Den Oord et al., 2016; Kong et al., 2020). The architecture is similar to the network used in (Rasul et al., 2021) but with fewer residual blocks. We used Fourier positional embeddings to encode the flow step s into a 32-dimensional vector using a fixed set of frequencies. We used the same embedding technique to encode time-dependent covariates c . We trained our models via stochastic gradient descent using an Adam optimizer with a learning rate of 0.001 and a batch size of 128. Depending on the dataset, training until convergence took from 40 – 100 minutes on a single A100-SXM4-40GB NVIDIA GPU.

Baselines We compared AFM against multiple established baselines. These include traditional statistical methods such as Seasonal Naive (SN), AutoARIMA, and AutoETS (Hyndman et al., 2008), as well as deep learning methods such as DeepAR (Salinas et al., 2020), WaveNet (Van Den Oord et al., 2016). We further include diffusion baselines such as TSDiff (Kollovich et al., 2023), SSSD (Alcaraz & Strodthoff, 2022), and TimeGrad (Rasul et al., 2021) and a flow matching baseline TSFlow (Kollovich et al., 2024).

Table 2: Test set CRPS comparison (lower is better) of different models on forecasting of five real world data sets. Mean and standard error metrics for AFM obtained by re-training and evaluating five times. Results for the baselines are from (Kollovich et al., 2024) and (Rasul et al., 2021). Best scores in **bold**, second best underlined.

Method	Electricity	Exchange	Solar	Traffic	Wikipedia
SN	0.060±0.000	0.013±0.000	0.512±0.000	0.221±0.000	0.423±0.000
ARIMA	0.344±0.000	0.013±0.000	0.558±0.003	0.486±0.000	0.654±0.000
ETS	0.056±0.000	0.008±0.000	0.550±0.000	0.492±0.000	0.651±0.000
DeepAR	0.051±0.000	0.013±0.004	0.429±0.015	0.103±0.000	0.215±0.003
WaveNet	0.058±0.008	0.012±0.001	0.360±0.000	0.099±0.000	0.207 ±0.000
CSDI	0.051±0.000	0.013±0.000	0.360±0.000	0.152±0.000	0.318±0.012
SSSD	0.048±0.000	<u>0.010</u> ±0.000	0.354±0.024	0.107±0.002	<u>0.209</u> ±0.000
TimeGrad	0.048±0.001	0.013±0.002	0.347±0.024	0.109±0.002	0.311±0.002
TSFlow	<u>0.045</u> ±0.001	0.009 ±0.001	0.343±0.002	0.083 ±0.000	0.227±0.000
AFM	0.042 ±0.001	0.009 ±0.001	0.284 ±0.002	<u>0.089</u> ±0.000	0.243±0.002

Table 3: Model efficiency comparison showing number of parameters and inference time per forecast on the Electricity and Solar datasets. AFM achieves the best accuracy with the smallest model size, while being 4× slower than the non-autoregressive TSFlow but significantly faster than the autoregressive diffusion baseline TimeGrad.

Method	Parameters	Inference Time	Electricity	Solar
TSFlow	4.1M	45ms	0.045±0.001	0.343±0.002
TimeGrad	2.8M	850ms	0.048±0.001	0.347±0.024
AFM	0.85M	180ms	0.042 ±0.001	0.284 ±0.002

Results

Results Table 2 presents the comparison of our proposed AFM model against various baselines on five real-world timeseries datasets. The results demonstrate that AFM achieves state-of-the-art performance on three of the five datasets: Electricity, Exchange, and Solar. On the Electricity dataset, AFM outperforms all baselines with a CRPS of 0.042, representing a 6.7% improvement over the second-best model, TSFlow (0.045), and a 12.5% improvement over the autoregressive diffusion baseline TimeGrad (0.048). For the Exchange dataset, AFM matches the performance of TSFlow with a CRPS of 0.009, significantly outperforming traditional statistical methods and other deep learning approaches. Most notably, on the Solar dataset, AFM achieves a substantial improvement with a CRPS of 0.284, representing a 17.2% reduction in error compared to TSFlow (0.343) and an 18.2% improvement over TimeGrad (0.347). While AFM ranks second on the Traffic dataset with a CRPS of 0.089, slightly behind TSFlow (0.083), it still outperforms all other traditional and deep

learning baselines including TimeGrad (0.109). On the Wikipedia dataset, WaveNet (0.207) and SSSD (0.209) achieve the best performance, with TSFlow (0.227) outperforming AFM (0.243).

Table 3 presents the computational efficiency analysis. AFM achieves the best predictive performance with the smallest model size (0.85M parameters compared to 4.1M for TSFlow and 2.8M for TimeGrad). In terms of inference time, AFM requires 180ms per forecast, which is 4× slower than the non-autoregressive TSFlow (45ms) but significantly faster than the autoregressive diffusion baseline TimeGrad (850ms). This demonstrates that AFM’s autoregressive flow matching framework with straight probability paths provides computational advantages over diffusion-based autoregressive approaches while maintaining compact model architectures. These results demonstrate that the autoregressive flow-matching approach employed in AFM consistently delivers superior or competitive performance across diverse real-world timeseries forecasting tasks, with favorable trade-offs between model size, inference speed, and predictive accuracy.

5 DISCUSSION

In this work, we introduced AFM, a probabilistic forecasting model that combines autoregressive modeling with flow matching to generate accurate and well-calibrated predictions for multivariate timeseries data. By factorizing the forecasting problem into a sequence of conditional distributions, AFM effectively balances the sequential dependencies in temporal data with the expressiveness of flow-based generative modeling. Our empirical results across simulated dynamical systems and real-world datasets demonstrate that AFM achieves competitive performance with state-of-the-art baselines in challenging forecasting benchmarks. The autoregressive decomposition leads to notable improvements in uncertainty calibration and distribution coverage compared to standard flow-matching approaches that model entire future windows simultaneously. Furthermore, AFM exhibits robust generalization to out-of-distribution scenarios, particularly in extrapolation tasks where it achieves 93–98% error reduction compared to non-autoregressive baselines (e.g., Brusselator: NRMSE 0.023 vs 1.046), a critical capability for real-world deployment where test conditions often diverge from training data.

AFM has limitations that warrant discussion. First, the sequential nature of autoregressive sampling introduces computational overhead during inference compared to fully parallelizable non-autoregressive models. Our experiments show that AFM is 4× slower than non-autoregressive flow matching approaches (180ms vs 45ms per forecast on real-world datasets), which may limit its applicability in real-time forecasting systems requiring latency below 50ms. However, AFM remains significantly faster than autoregressive diffusion baselines like TimeGrad (850ms) while achieving better predictive performance with smaller model sizes (0.85M vs 2.8M–4.1M parameters). Non-autoregressive methods are preferable when: (1) real-time inference is critical with strict latency requirements (<50ms), (2) forecast horizons are very short (1–5 steps) where the benefits of autoregressive modeling are minimal, and (3) training data fully covers the target prediction horizon and extrapolation beyond the training distribution is not required. In contrast, AFM excels in applications where accuracy and well-calibrated uncertainty estimates are prioritized over inference speed, such as climate modeling, demand planning, and scientific forecasting tasks that involve extrapolation. Additionally, our current implementation applies flow matching directly in the data space, which may be suboptimal for high-dimensional partially observed data. Future work could explore latent-space formulations, extend to irregularly sampled data, investigate adaptive window sizes that balance the trade-off between model capacity and computational efficiency, and explore broader applications in scientific modeling and decision-making under uncertainty. Overall, AFM represents a step forward in probabilistic timeseries forecasting, demonstrating the potential of flow-based generative models for improving both accuracy and uncertainty quantification while maintaining computational efficiency relative to diffusion-based approaches.

REFERENCES

- Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.
- Juan Miguel Lopez Alcaraz and Nils Strodthoff. Diffusion-based time series imputation and forecasting with structured state space models. *arXiv preprint arXiv:2208.09399*, 2022.

- 432 Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan
433 Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Rangapuram, David Salinas, Jasper
434 Schulz, et al. GluonTS: Probabilistic and neural time series modeling in python. *Journal of*
435 *Machine Learning Research*, 21(116):1–6, 2020.
- 436
437 Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary
438 differential equations. *Advances in Neural Information Processing Systems*, 31, 2018.
- 439 Fatoumata Dama and Christine Sinoquet. Time series analysis and modeling to forecast: a survey.
440 In *ArXiv:2104.00164*, 2021.
- 441
442 Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. TimeVAE: A variational auto-
443 encoder for multivariate time series generation. *arXiv preprint arXiv:2111.08095*, 2021.
- 444
445 Dheeru Dua, Casey Graff, et al. UCI machine learning repository. URL <http://archive.ics.uci.edu/ml/>,
446 7(1):62, 2017.
- 447 Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam
448 Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for
449 high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*,
450 2024.
- 451
452 Richard FitzHugh. Impulses and physiological states in theoretical models of nerve membrane.
453 *Biophysical Journal*, 1(6):445–466, 1961. doi: 10.1016/S0006-3495(61)86902-6.
- 454
455 Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas,
456 Valentin Flunkert, and Tim Januschowski. Probabilistic forecasting with spline quantile function
457 RNNs. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1901–
1910. PMLR, 2019.
- 458
459 Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFFORD:
460 Free-Form continuous dynamics for scalable reversible generative models. In *International Con-*
461 *ference on Learning Representations*, 2018.
- 462
463 Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in*
464 *Neural Information Processing Systems*, 33:6840–6851, 2020.
- 465
466 Yang Hu, Xiao Wang, Lirong Wu, Huatian Zhang, Stan Z. Li, Sheng Wang, and Tianlong Chen.
467 Fm-ts: Flow matching for time series generation. In *ArXiv:2411.07506v1 [cs.LG]*, 2024. URL
<http://arxiv.org/abs/2411.07506>.
- 468
469 Rob Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. *Forecasting with Exponential*
470 *Smoothing: The State Space Approach*. Springer Science & Business Media, 2008.
- 471
472 Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International*
Journal of Forecasting, 22(4):679–688, 2006.
- 473
474 Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick Van der Smagt. Deep varia-
475 tional Bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint*
476 *arXiv:1605.06432*, 2016.
- 477
478 Marcel Kollovich, Abdul Fatir Ansari, Michael Bohlke-Schneider, Jasper Zschiegner, Hao Wang,
479 and Yuyang Bernie Wang. Predict, refine, synthesize: Self-guiding diffusion models for proba-
480 bilistic time series forecasting. *Advances in Neural Information Processing Systems*, 36:28341–
28364, 2023.
- 481
482 Marcel Kollovich, Marten Lienen, David Lüdke, Leo Schwinn, and Stephan Günemann. Flow
483 matching with gaussian process priors for probabilistic time series forecasting. In *The Thirteenth*
International Conference on Learning Representations, 2024.
- 484
485 Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A versatile
diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

- 486 Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term
487 temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference*
488 *on Research & Development in Information Retrieval*, pp. 95–104, 2018.
- 489 R. Lefever and I. Prigogine. Symmetry breaking instabilities in dissipative systems. II. *The Journal*
490 *of Chemical Physics*, 48(4):1695–1700, 1968. doi: 10.1063/1.1668896.
- 492 Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: A survey. *Philosophical*
493 *Transactions of the Royal Society A*, 379(2194):20200209, 2021.
- 494 Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow match-
495 ing for generative modeling. In *The Eleventh International Conference on Learning Representa-*
496 *tions*, 2022.
- 498 Xingchao Liu, Chengyue Gong, et al. Flow straight and fast: Learning to generate and transfer data
499 with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2022.
- 501 Edward Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2), 1963.
- 502 Alfred J. Lotka. *Elements of Physical Biology*. Williams & Wilkins, 1925.
- 503 Caspar Meijer and Lydia Y Chen. The rise of diffusion models in time-series forecasting. *arXiv*
504 *preprint arXiv:2401.03006*, 2024.
- 506 Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv
507 Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. MovieGen: A cast of media founda-
508 tion models. *arXiv e-prints*, pp. arXiv–2410, 2024.
- 509 Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. Mul-
510 tivariate probabilistic time series forecasting via conditioned normalizing flows. *arXiv preprint*
511 *arXiv:2002.06103*, 2020.
- 513 Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising dif-
514 fusion models for multivariate probabilistic time series forecasting. In *International Conference*
515 *on Machine Learning*, pp. 8857–8868. PMLR, 2021.
- 516 David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus.
517 High-dimensional multivariate forecasting with low-rank gaussian copula processes. *Advances*
518 *in Neural Information Processing Systems*, 32, 2019.
- 520 David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic
521 forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):
522 1181–1191, 2020.
- 523 Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised
524 learning using nonequilibrium thermodynamics. In *International Conference on Machine Learn-*
525 *ing*, pp. 2256–2265. PMLR, 2015.
- 527 Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben
528 Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint*
529 *arXiv:2011.13456*, 2020.
- 530 Ella Tamir, Najwa Laabid, Markus Heinonen, Vikas Garg, and Arno Solin. Conditional flow match-
531 ing for time series modelling. In *ICML Workshop: Structured Probabilistic Inference and Gener-*
532 *ative Modeling*, 2024.
- 534 Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: Conditional score-based
535 diffusion models for probabilistic time series imputation. *Advances in Neural Information Pro-*
536 *cessing Systems*, 34:24804–24816, 2021.
- 537 Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-
538 Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models
539 with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

540 Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves,
541 Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, et al. WaveNet: A generative model for
542 raw audio. *arXiv preprint arXiv:1609.03499*, 12, 2016.

543
544 B. van der Pol. On 'relaxation-oscillations'. *The London, Edinburgh, and Dublin Philosophical
545 Magazine and Journal of Science*, 2(11):978–992, 1926. doi: 10.1080/14786442608564127.

546 Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent
547 neural networks. *Neural Computation*, 1(2):270–280, 1989.

548
549 Robert L Winkler, Javier Munoz, José L Cervera, José M Bernardo, Gail Blattenberger, Joseph B
550 Kadane, Dennis V Lindley, Allan H Murphy, Robert M Oliver, and David Ríos-Insua. Scoring
551 rules and the evaluation of probabilities. *Test*, 5:1–60, 1996.

552 Jiexia Ye, Weiqi Zhang, Ke Yi, Yongzi Yu, Ziyue Li, Jia Li, and Fugee Tsung. A survey of time
553 series foundation models: Generalizing time series representation with large language model. In
554 *ArXiv:2104.00164*, 2024.

555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593

A DATASETS

A.1 DYNAMICAL SYSTEMS

We generated training and testing data for five classical stochastic dynamical systems, that is, Lorenz, FitzHugh-Nagumo, Lotka-Volterra, Brusselator and Van der Pol, by numerically integrating their corresponding stochastic differential equations (SDEs). Each system has the general form

$$d\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), t; \Theta) dt + \sigma d\mathbf{W}(t)$$

where f is the deterministic drift, Θ are the system parameters, σ is a constant diffusion term, and $\mathbf{W}(t)$ is a standard Brownian motion. We sampled 2400 random initial conditions from a uniform distribution over the specified range (see Table 4) for each system and generated a trajectory for each initial condition by solving the SDE using the Euler-Heun method with fixed step size. We sampled 200 equally spaced time points for each trajectory. Out of the 200 time points in each trajectory, the first 75 were used as observed data, the next 75 for prediction, and the final 50 for extrapolation. Out of the 2400 trajectories generated, we used 2000 for training and 400 for testing.

Table 4: Summary of stochastic dynamical systems. $\mathbf{f}(\mathbf{x})$ denotes the drift function, Θ the system parameters, σ the constant diffusion term, and $p(\mathbf{x}(0))$ the initial condition distribution. Each system was integrated over the given interval $[t_0, t_1]$ with 200 steps using the Euler-Heun method.

System	$f(\mathbf{x})$	Θ	Σ	$p(\mathbf{x}(0))$	$[t_0, t_1]$
Lorenz	$f_1 = \sigma(x_2 - x_1)$	$\sigma = 10$	(1.5, 1.5, 1.5)	$\mathcal{U}([0, 10])$	[0, 2]
	$f_2 = x_1(\rho - x_3) - x_2$	$\rho = 28$			
	$f_3 = x_1 x_2 - \beta x_3$	$\beta = \frac{8}{3}$			
FitzHugh-Nagumo	$f_1 = x_1 - \frac{x_1^3}{3} - x_2 + I$ $f_2 = \frac{x_1 + a - b x_2}{\tau}$	$a = 0.7$	(1.5, 1.5)	$\mathcal{U}([-2, 2])$	[0, 10]
		$b = 0.8$			
		$\tau = 12.5$			
		$I = 0.5$			
Lotka-Volterra	$f_1 = \alpha x_1 - \beta x_1 x_2$ $f_2 = -\delta x_2 + \gamma x_1 x_2$	$\alpha = 1.3$	(1.5, 1.5)	$\mathcal{U}([0, 5])$	[0, 20]
		$\beta = 0.9$			
		$\gamma = 0.8$			
		$\delta = 1.8$			
Brusselator	$f_1 = A + x_1^2 x_2 - (B + 1) x_1$ $f_2 = B x_1 - x_1^2 x_2$	$A = 1.0$	(1.5, 1.5)	$\mathcal{U}([0, 2])$	[0, 20]
		$B = 3.0$			
Van der Pol	$f_1 = x_2$ $f_2 = \mu(1 - x_1^2) x_2 - x_1$	$\mu = 0.1$	(1.5, 1.5)	$\mathcal{U}([-2, 2])$	[0, 20]

A.2 REAL-WORLD DATASETS

We used five different datasets from GluonTS (Alexandrov et al., 2020) to evaluate our model performance. Specifically, we used the versions preprocessed as in (Salinas et al., 2019). A summary of their properties is listed in Table 5.

Table 5: Properties of the datasets used in the experiments: dimension n , domain Ω , frequency, total training timesteps, and prediction length.

Dataset	n	Ω	freq.	timesteps	prediction length
Electricity	370	\mathbb{R}^+	hourly	5833	24
Exchange	8	\mathbb{R}^+	daily	6071	30
Solar	137	\mathbb{R}^+	hourly	7009	24
Traffic	963	(0, 1)	hourly	4001	24
Wikipedia	2000	\mathbb{N}	daily	792	30

B NON-AUTOREGRESSIVE FLOW MATCHING BASELINE

Problem Setting We consider the same forecasting setup as in Section 3 where given past observations \mathbf{Y}_l and covariates \mathbf{C} , our goal remains to model the conditional distribution $p(\mathbf{Y}_f | \mathbf{Y}_l, \mathbf{C})$. However, unlike the autoregressive factorization, we directly model the joint distribution of the entire future trajectory $\mathbf{Y}_f \in \mathbb{R}^{f \times n}$ without temporal factorization:

$$p(\mathbf{Y}_f | \mathbf{Y}_l, \mathbf{C}) = p(\mathbf{y}_{\tau+1}, \dots, \mathbf{y}_{\tau+f} | \mathbf{Y}_l, \mathbf{C}). \quad (9)$$

This formulation preserves temporal correlations across all future time steps but requires learning a high-dimensional distribution.

Training We construct a probability path $(p^s(\mathbf{Y}_f))^{0 \leq s \leq 1}$ that transports samples from a prior distribution $p^0(\mathbf{Y}_f) = \mathcal{N}(0, \Sigma)$ to the target distribution $p^1(\mathbf{Y}_f) \approx p(\mathbf{Y}_f | \mathbf{Y}_l, \mathbf{C})$, where Σ is a block-diagonal covariance matrix. The prior can be interpreted as independent Brownian motion processes per dimension, where Brownian motion is defined as a stochastic process $W: [\tau + 1, \tau + f] \rightarrow \mathbb{R}^n$.

We define the conditional probability path using a linear interpolation bridge with Brownian noise:

$$p^s(\mathbf{Y}_f | \mathbf{z}) = \mathcal{N}((1-s)\mathbf{Y}_f^0 + s\mathbf{Y}_f^1, \sigma^2 s(1-s)\Sigma) \quad (10)$$

where $\mathbf{z} = (\mathbf{Y}_f^0, \mathbf{Y}_f^1)$ with $\mathbf{Y}_f^0 \sim p^0$ and $\mathbf{Y}_f^1 \sim p^1$. The corresponding conditional velocity field becomes:

$$\mu(\mathbf{Y}_f, s | \mathbf{z}) = \mathbf{Y}_f^1 - \mathbf{Y}_f^0 + \frac{\sigma^2(1-2s)}{2}\Sigma^{-1}(\mathbf{Y}_f - ((1-s)\mathbf{Y}_f^0 + s\mathbf{Y}_f^1)) \quad (11)$$

When $\sigma^2 \rightarrow 0$, this reduces to the straight path velocity $\mu(\mathbf{Y}_f, s | \mathbf{z}) = \mathbf{Y}_f^1 - \mathbf{Y}_f^0$. We learn a neural velocity field ν_θ that operates on the entire future trajectory. The context encoding remains similar to Eq. (7):

$$\mathbf{h} = \zeta_\phi(\mathbf{Y}_l, \mathbf{C}_{\tau-l:\tau+f}) \quad (12)$$

where ζ_ϕ now processes both past and future covariates. The training objective becomes:

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{\mathbf{z} \sim \pi_{0,1}, s \sim \mathcal{U}(0,1), \mathbf{Y}_f^s \sim p^s(\mathbf{Y}_f | \mathbf{z})} \left\| \mu(\mathbf{Y}_f^s, s | \mathbf{z}) - \nu_\theta(\mathbf{Y}_f^s, \mathbf{h}, \mathbf{C}_{\tau+1:\tau+f}, s) \right\|^2 \quad (13)$$

where ν_θ is implemented as a sequential neural network (e.g., Transformer or RNN) that processes the entire trajectory. Sampling requires solving the trajectory-level ODE:

$$\frac{d\psi(\mathbf{Y}_f, s)}{ds} = \nu_\theta(\psi(\mathbf{Y}_f, s), \mathbf{h}, \mathbf{C}_{\tau+1:\tau+f}, s), \quad \psi(\mathbf{Y}_f, 0) \sim p^0 \quad (14)$$

using numerical solvers. This produces joint samples from $p(\mathbf{Y}_f | \mathbf{Y}_l, \mathbf{C})$ without autoregressive decomposition.

Implementation Details We used this baseline to conduct experiments on the task of forecasting multiple dynamical systems. For this setup we set ν_θ as a 4-layer bi-directional LSTM with 128 hidden units. We kept the same training parameters as in Section 4.1. Training until convergence took ~ 2 minutes on a single A100-SXM4-40GB NVIDIA GPU.

Key Differences from Autoregressive Variant The non-autoregressive approach differs fundamentally from its autoregressive counterpart introduced in AFM. First, while the autoregressive method factorizes the joint distribution via a Markovian decomposition across time steps, the non-autoregressive baseline directly models the full future trajectory \mathbf{Y}_f as a single high-dimensional random variable. This preserves cross-temporal dependencies at the expense of learning a more complex distribution over $\mathbb{R}^{n \times f}$. Second, computational characteristics diverge significantly: the non-autoregressive version processes entire trajectories through sequential neural networks for the velocity fields enabling parallel generation of all future timepoints at the cost of higher memory requirements. Finally, their uncertainty propagation mechanisms contrast sharply: the autoregressive approach explicitly models how prediction errors accumulate through the chain of conditional distributions, while the non-autoregressive method captures joint uncertainty over all timesteps through trajectory-level sampling but lacks explicit mechanisms to model error accumulation dynamics. These differences create complementary trade-offs – the autoregressive variant offers interpretable uncertainty quantification and efficient window-based computation but assumes limited Markovian dependencies, while the non-autoregressive baseline preserves full temporal correlations at higher computational cost with less explicit uncertainty dynamics.

Algorithm 3 Non-Autoregressive Flow Matching: Training

Require: Dataset \mathcal{D} , networks ν_θ, ζ_ϕ

- 1: **while** not converged **do**
- 2: Sample $\mathbf{Y}_f^0 \sim \mathcal{N}(0, \Sigma)$ and $(\mathbf{Y}_l^1, \mathbf{Y}_l, \mathbf{C}) \sim \mathcal{D}$
- 3: Set $\mathbf{h} = \zeta_\phi(\mathbf{Y}_l, \mathbf{C}_{\tau-l:\tau+f})$
- 4: Sample $s \sim \mathcal{U}(0, 1)$ and $\mathbf{Y}_f^s \sim \mathcal{N}((1-s)\mathbf{Y}_f^0 + s\mathbf{Y}_f^1, \sigma^2 s(1-s)\Sigma)$
- 5: Compute target $\mu(\mathbf{Y}_f^s, s \mid \mathbf{z})$
- 6: Update θ, ϕ to minimize $\|\mu(\cdot) - \nu_\theta(\mathbf{Y}_f^s, \mathbf{h}, \mathbf{C}_{\tau+1:\tau+f}, s)\|_{\Sigma^{-1}}^2$
- 7: **end while**

Algorithm 4 Non-Autoregressive Flow Matching: Sampling

Require: \mathbf{Y}_l, \mathbf{C} , networks ν_θ, ζ_ϕ

- 1: Encode context $\mathbf{h} = \zeta_\phi(\mathbf{Y}_l, \mathbf{C})$
- 2: Sample initial trajectory $\mathbf{Y}_f^0 \sim \mathcal{N}(0, \Sigma)$
- 3: Solve $\frac{d\psi(\mathbf{Y}_f, s)}{ds} = \nu_\theta(\psi(\mathbf{Y}_f, s), \mathbf{h}, \mathbf{C}_{\tau+1:\tau+f}, s)$
- 4: Return $\psi(\mathbf{Y}_f, 1)$

C EVALUATION METRICS

In this section, we describe the metrics used to evaluate the quality of our forecasts. Given an n -dimensional time series $\mathbf{Y}_f = \{\mathbf{y}_{\tau+1}, \dots, \mathbf{y}_{\tau+f}\}$ and its predictive distribution or samples from our probabilistic model, we employ two metrics, as described in the following.

C.1 CONTINUOUS RANKED PROBABILITY SCORE

The continuous ranked probability score (CRPS) is a proper scoring rule commonly used to assess the quality of probabilistic forecasts. For a univariate random variable X with cumulative distribution function (CDF) F_X and an observed value x , the CRPS is defined as:

$$\text{CRPS}(F_X, x) = \int_{-\infty}^{+\infty} (F_X(y) - \mathbf{1}\{y \geq x\})^2 dy, \quad (15)$$

where $\mathbf{1}\{\cdot\}$ is the indicator function. Intuitively, the CRPS evaluates how well the entire predictive distribution aligns with the observation x . Lower CRPS values indicate better-calibrated and more accurate predictive distributions.

In the context of time series forecasting, we can compute the CRPS for each time step in the forecasting horizon and average the results:

$$\text{CRPS}_{\text{avg}} = \frac{1}{f} \sum_{k=1}^f \text{CRPS}(F_{X_{\tau+k}}, y_{\tau+k}), \quad (16)$$

where $F_{X_{\tau+k}}$ is the predicted CDF (or an empirical CDF from samples) at forecast horizon k , and $y_{\tau+k}$ is the corresponding ground-truth observation.

C.2 NORMALIZED ROOT MEAN SQUARE ERROR

The root mean square error (RMSE) is a standard metric for evaluating the accuracy of point forecasts. For a set of predictions $\{\hat{y}_{\tau+1}, \dots, \hat{y}_{\tau+f}\}$ and corresponding ground-truth values $\{y_{\tau+1}, \dots, y_{\tau+f}\}$, the RMSE is:

$$\text{RMSE} = \sqrt{\frac{1}{f} \sum_{k=1}^f (\hat{y}_{\tau+k} - y_{\tau+k})^2}. \quad (17)$$

To make the RMSE scale-invariant and facilitate comparison across different datasets or time series with different magnitudes, we use the normalized RMSE (NRMSE). One common normalization is

756 by the standard deviation of the ground-truth series, σ_y :

$$757 \quad \text{NRMSE} = \frac{\text{RMSE}}{\sigma_y}, \quad (18)$$

760 where σ_y is the sample standard deviation of the observations. Alternatively, one could normalize by
761 the range of the data ($\max(y) - \min(y)$) depending on the application. A lower NRMSE indicates
762 more accurate predictions relative to the variability of the time series.
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809