Stable Multi-Step Volatility Forecasting via Universal Differential Equations

Anonymous authors
Paper under double-blind review

Abstract

Neural differential equations such as Neural ODEs, Neural CDEs, and Universal Differential Equations (UDEs) model temporal evolution as a continuous-time flow rather than a fixed-step recurrence. Even for regularly sampled data, this formulation differs fundamentally from discrete-time architectures: it learns smooth vector fields governing instantaneous rates of change, reducing discretization bias and improving long-horizon stability. We present a systematic study of **Universal Differential Equations** for financial volatility forecasting, a domain characterized by regime shifts, heavy tails, and jump discontinuities. UDEs extend Neural ODEs by embedding mechanistic structure within learned dynamics, using neural networks to parameterize coefficients in partially known differential equations instead of learning the system purely from data. Our UDE variants incorporate volatility's empirical regularities while retaining neural flexibility for regime adaptation. Across market regimes, they outperform both continuous-time baselines and discrete-time models, achieving higher accuracy and greater long-horizon stability while remaining interpretable. These results suggest that UDEs grounded in mechanistic structure and neural flexibility offer a principled route to stable, interpretable multi-step forecasting in nonstationary domains.

1 Introduction

Time series forecasting is fundamentally challenging due to complex temporal dependencies, non-stationarity, and abrupt regime shifts that make accurate prediction difficult (24; 19). These challenges are particularly acute in domains where the data-generating process itself is unstable and where small modeling errors can have cascading consequences.

We focus on financial volatility as our evaluation domain: Volatility, the magnitude of price fluctuations, serves as a demanding testbed for forecasting methods for three reasons. First, it is practically important: volatility quantifies market risk and is fundamental to portfolio construction, option pricing, and systemic risk management (2; 12). Second, high-quality open data is widely available (e.g., via *yfinance*), enabling reproducible research without the computational burden typically associated with domains like climate modeling. Third, and most critically, volatility exhibits extreme versions of these challenges, as detailed below.

Even one-step-ahead volatility forecasting is notoriously difficult: Unlike asset prices, where trends can persist for extended periods, volatility is inherently noisy with rapid, unpredictable fluctuations. This difficulty stems from well-documented structural properties known as "stylized facts" in the empirical finance literature (7; 18). Volatility exhibits *clustering*, where high-volatility periods persist before returning to calm (11); *jump discontinuities*, where sudden, large price movements trigger instantaneous volatility spikes that cannot be predicted from continuous dynamics alone (7); and *heavy-tailed distributions*, where extreme events occur far more frequently than normal distributions predict (6). These properties mean that a model trained during calm markets can fail catastrophically when the market regime shifts, a form of structural instability that makes even single-step prediction challenging.

Multi-step forecasting amplifies these challenges dramatically: When predicting multiple time steps into the future, errors at each step compound, and the distribution shift accumulates over the forecast horizon.

Subtle model mis-specifications that are nearly invisible in one-step-ahead prediction, such as small biases, incorrect noise assumptions, or slightly mis-calibrated parameters, become larger failures at longer horizons. The model must not only predict the next value accurately but also maintain stability and calibration as it recursively feeds its own predictions forward.

For volatility, multi-step forecasting is particularly challenging: Volatility clustering means that one error during a high-volatility regime can destroy the entire forecast trajectory. If the model underestimates volatility at a step t, it will likely continue underestimating at subsequent steps $t+1, t+2, \ldots$ as the clustering persists. Unlike smoother time series, where multi-step errors degrade gradually, volatility forecasts can collapse abruptly at longer horizons.

Current approaches fall short in different ways. Recent foundation models with transformer backbones, such as TimeGPT, Chronos, TimesFM, achieve impressive zero-shot generalization but require massive computational resources and operate entirely as black boxes. Standard deep learning approaches (LSTMs, GRUs) offer flexibility but lack structural constraints, making them prone to instability. Neural ODEs (4) impose continuous-time dynamics for better inductive bias, but because the dynamics function $f_\theta(x, t)$ is a general-purpose neural network, they still learn purely from data without domain-specific constraints, allowing them to overfit to noise and spurious patterns rather than capturing the underlying generative process. Traditional econometric models, such as GARCH, FI-GARCH, encode domain structure but are too rigid to adapt to regime shifts or model misspecification. This challenge motivates hybrid approaches that combine neural flexibility with mechanistic stability when partial domain knowledge exists.

This observation points to a fundamental challenge in time series modeling. Fully data-driven approaches, LSTMs, GRUs, and Transformers learn discrete-time state transitions as black-box functions, requiring massive datasets to discover stable patterns and often failing at long horizons where they extrapolate beyond training distributions (24; 19). Neural ODEs improve upon this by imposing continuous-time dynamics, which provides better inductive bias for temporal modeling, but because the dynamics function $f_\theta(x, t)$ is a general-purpose neural network, they still learn dynamics purely from data without incorporating known physical or economic principles (4). At the other extreme, mechanistic approaches to stochastic differential equations (SDEs) like the Heston model (1) or GARCH-family models (2) encode domain knowledge for interpretable and stable predictions but are inflexible: they cannot adapt to regime shifts or learn dynamics beyond their fixed functional forms. For volatility, GARCH models assume specific parametric structures that may misspecify the true dynamics, while SDEs require manual calibration and struggle with sudden market dislocations (1; 12).

This challenge motivates hybrid approaches that combine neural flexibility with mechanistic stability when partial domain knowledge exists. Such methods can leverage known dynamics as structural inductive biases while allowing neural components to learn residual effects, regime-dependent parameters, or model misspecification that rigid parametric forms cannot capture.

ODE vs UDE. An Ordinary Differential Equation (ODE) specifies a fully known righthand side, $\dot{x} = f(x, t; \vartheta)$; a Universal Differential Equation (UDE) augments a mechanistic ODE with a learned component, $\dot{x} = f_{\text{known}}(x, t; \vartheta) + g_{\text{NN}}(x, t; \phi)$, or equivalently learns state dependent coefficients while retaining the governing form. In this paper, we adopt the modulation view: small MLPs modulate mean reversion coefficients under positivity/bound constraints to preserve dissipativity and interpretability, in contrast to a vanilla neural ODE that learns an unconstrained vector field.

Universal Differential Equations (UDEs) for volatility forecasting. We demonstrate that UDEs (21), previously applied to scientific discovery and PDE solving, provide an effective framework for multi-step volatility forecasting. Our approach uses neural networks to modulate mechanistic coefficients (mean-reversion rates, long-term levels) within a mean-reverting structure rather than learning dynamics from scratch. This preserves the stabilizing inductive bias of the mechanistic form while allowing adaptation to regime-dependent dynamics. The result is a model that maintains interpretability, each learned coefficient corresponds to a meaningful market quantity, while achieving strong multi-step stability that purely learned approaches cannot match.

Essentially, our framework leverages volatility's stylized facts as domain knowledge while allowing neural components to learn continuous, state-dependent parameter variations. This blend of structure and flexibility proves critical for multi-step stability: the mechanistic structure constrains the functional form of dynamics always to remain mean-reverting by construction, while neural modulation enables smooth adaptation to evolving market conditions within that stable template. While frequently observed, such empirical regularities can fail under structural change; our design aims to improve stability without assuming inviolable physical laws.

Model hierarchy and terminology. Our experimental framework spans a spectrum from pure data-driven to pure mechanistic models. At one extreme, **Vanilla Neural ODE** learns dynamics entirely from data with no structural priors. At the other, **ODE-MR** is a pure mean-reverting process with fixed parameters. Between these extremes, our UDE variants progressively incorporate stylized facts: **UDE-MR** adds neural modulation of mean-reversion coefficients; **UDE-MRVC** further adds volatility clustering; and **UDE-MRVCJ** additionally incorporates jump dynamics. All UDE variants enforce soft constraints, such as nonnegative mean-reversion rates, bounded long-run levels, structured clustering terms to ensure neural components respect rather than overturn the mechanistic structure.

Our contributions are:

1. **Rigorous evaluation of UDEs for multi-step time series forecasting:** We provide the first systematic assessment of UDEs on financial volatility forecasting using rolling walk-forward validation with strictly non-overlapping windows across diverse market regimes (2008, 2020, 2022) and asset classes (equity, emerging markets, crypto ,etc.).

2. **Parameter efficiency through structural inductive biases:** UDEs achieve competitive accuracy with significantly fewer parameters than foundation models, demonstrating that embedding domain structure can replace massive scale for forecasting tasks with partial mechanistic knowledge.

3. **Interpretable regime adaptation:** Unlike black-box foundation models, the hybrid architecture maintains interpretability through its mechanistic structure. Learned parameters correspond to economically meaningful quantities (mean-reversion rates, volatility clustering strength, jump intensities) that can be tracked across time, enabling post-hoc analysis of how market dynamics evolve across regimes (Figure **??**).

4. **Long-horizon stability:** UDEs remain relatively stable across longer horizons, unlike purely neural and foundation models, which degrade at a higher rate.

2 Related Work

Multi-step Time Series Forecasting: Multi-step forecasting presents unique challenges due to error accumulation and distribution shift, with approaches broadly categorized into direct, recursive, and multi-output strategies (23). Deep learning architectures have significantly advanced the field, including LSTNet (17) which combined CNNs and RNNs for temporal patterns, and transformer-based methods like Informer (26) and Autoformer (24) that capture long-term dependencies through specialized attention mechanisms. Recent state-of-the-art foundation models like TimesFM (9) leverage large-scale pretraining across diverse temporal domains to enable robust, zero-shot multi-step forecasting.

Volatility Forecasting: Financial volatility forecasting has evolved from parametric econometric models to flexible machine learning approaches. The GARCH family (2; 13) and stochastic volatility models (14) have dominated academic finance, providing interpretable but potentially misspecified parametric forms. Recent machine learning approaches include neural networks for volatility prediction (25), deep learning models capturing complex patterns (3), and hybrid methods that combine GARCH with neural networks as feature extractors (5).

Universal Differential Equations: The Universal Differential Equations paradigm represents an advancement in scientific machine learning, structurally integrating mechanistic models with data-driven components. Introduced by Rackauckas et al. (21), UDEs parameterize unknown dynamical terms using neural

networks, enabling simultaneous leverage of domain knowledge and data-driven flexibility. This extends several key lineages: Neural ODEs (4), Physics-Informed Neural Networks (22), and symbolic regression for interpretable dynamics (8). While previous UDE applications have focused on physical and biological systems, our work explores their value for financial forecasting, modeling volatility dynamics as partially-specified differential equations with learned market microstructure effects.

3 Method

We formulate financial volatility forecasting as a multi-step prediction problem. Let $\{x\}_{t=0}^{T}$ denote the sequence of log-volatilities, where $x_t = \ln(\sigma_t)$ and σ_t represents the latent daily volatility. The objective is to predict future log-volatilities $\hat{x}_{t+h}$ at fixed horizon h, leveraging past observed values within a look-back window of length K.

Log-volatility is the standard target for volatility modeling. It acts as a variance-stabilizing transformation: the original volatility process σ_t exhibits heteroskedasticity, while $x_t = \ln(\sigma_t)$ converts multiplicative dynamics into additive noise with more stable variance, simplifying modeling assumptions across methods.

Volatility is estimated using the Garman-Klass (GK) range-based estimator on daily prices. For each day t, the GK realized variance is

$$\sigma^2_{\mathrm{GK},t} = \tfrac{1}{2}\big(\ln \tfrac{H_t}{L_t}\big)^2 - \big(2\ln 2 - 1\big)\big(\ln \tfrac{C_t}{O_t}\big)^2 \tag{1}$$

where O_t, H_t, L_t, C_t are the open, high, low, and close. We take $\sigma_t = \sqrt{\sigma^2_{\mathrm{GK},t}}$ and use $x_t = \ln(\sigma_t)$ for modeling. The GK estimator is approximately 7.4 times (10) more efficient than the close-to-close squared returns volatility estimator for daily data while remaining robust to drift.

3.1 Pure Physics Model: ODE with Mean Reversion

Considering the log-volatility $x_t \in \mathbb{R}$, the pure physics model posits the continuous-time dynamics governed by the Ordinary Differential Equation:

$$d(x)/d(t) = \kappa(\theta - x) \tag{2}$$

with parameters $\kappa > 0$ representing the mean reversion speed and $\theta \in \mathbb{R}$ the long-term equilibrium. This linear vector field is globally Lipschitz continuous with Lipschitz constant κ, which by the Picard–Lindelöf theorem guarantees existence and uniqueness of solutions for all initial conditions. The closed-form solution

$$x(t) = \theta + (x_0 - \theta)e^{-\kappa t} \tag{3}$$

demonstrates exponential convergence to θ, confirming global asymptotic stability. The Jacobian matrix of the system linearised about the equilibrium point $x^* = \theta$ is $J = -\kappa < 0$, which ensures perturbations decay exponentially, embodying the mean reversion property fundamental to modeling volatility dynamics.

3.2 Pure Neural ODE Model: Data-Driven Dynamics

In contrast to the purely parametric physics model, the pure neural ODE represents the log-volatility evolution via a learnable continuous vector field parameterized by a neural network $NN_\phi : \mathbb{R} \to \mathbb{R}$,

$$d(x)/d(t) = NN_\phi(x), \tag{4}$$

where ϕ denotes network parameters. Leveraging the Universal Approximation Theorem, this neural network, with sufficient depth and non-linear smooth activations such as GELU, can approximate any continuous vector field on a compact subset of $\mathbb{R}$ arbitrarily well. The local Lipschitz continuity of NN_ϕ ensures, via classical ODE theory, local existence and uniqueness of solutions. However, neural vector fields impose no inherent structural guarantees like mean reversion or boundedness, potentially leading to unstable trajectories or divergence, which may affect forecasting performance and interpretability.

3.3 Universal Differential Equation Model: Structural Coupling via Neural Modulation

The universal differential equation framework allows us to integrate prior domain knowledge by coupling neural modulation with mean reversion, one of the strongest studied effects in volatility:

$$\frac{dx}{dt} = \kappa_{\text{eff}}(x)\left(\theta_{\text{eff}}(x) - x\right) \tag{5}$$

Rather than learning the dynamics function entirely from data (as in Vanilla Neural ODE) or fixing parameters rigidly (as in pure ODE), we use small neural networks to modulate the mechanistic coefficients in a state-dependent manner. Specifically, the mean-reversion rate and long-term level are defined as:

$$\kappa_{\text{eff}}(x) = \underbrace{\text{softplus}(\kappa) + \varepsilon}_{\kappa_+} \cdot \exp\big(s\,\tanh(NN_1(x))\big), \quad \theta_{\text{eff}}(x) = \theta + b\,\tanh(NN_2(x)) \tag{6}$$

where NN_1 and NN_2 are small MLPs (e.g., 3 hidden layers with 64 units each), $s > 0$ and $b > 0$ control the modulation strength, and $\varepsilon > 0$ is a small constant. This multiplicative structure ensures $\kappa_{\text{eff}}(x) > 0$ (preserving dissipativity) while keeping $\theta_{\text{eff}}(x)$ bounded, thereby maintaining mean-reverting stability. Critically, the neural component modulates coefficients within the mechanistic structure rather than adding a separate unconstrained drift term, which would allow the network to learn dynamics that overturn the mean-reverting property.

Incremental model composition. We first enforce physics-parameter constraints ($\kappa_+ > 0$ via softplus and bounded θ_{eff} shifts), which forces a mean-reverting representation and prevents runaway dynamics. Building on this base, we progressively incorporate additional stylized facts through compositional stacking: volatility clustering (via lagged squared returns) and jump processes (via learned jump rate and magnitude). This progressive design enables systematic ablation of each component's contribution to forecasting performance. Table 1 presents the model variants evaluated in our experiments.

Table 1: UDE model variants with progressive incorporation of stylized facts. Ablation studies (Section 4.3) show monotonic improvements from compositional stacking.

Variant	**Dynamics Equation**
UDE-Mean Reversion	$\frac{dx}{dt} = \kappa_{\text{eff}}(x)\left(\theta_{\text{eff}}(x) - x\right)$
UDE-Volatility Clustering	$\frac{dx}{dt} = \kappa_{\text{eff}}(x)\left(\theta_{\text{eff}}(x) - x\right) + \gamma_+\, r_{t-1}^2$
	where $\gamma_+ = \text{softplus}(\gamma) + \varepsilon$
UDE-Jumps	$\frac{dx}{dt} = \kappa_{\text{eff}}(x)\left(\theta_{\text{eff}}(x) - x\right) + \lambda(x)\,J(x)$
	where $\lambda(x) = \text{softplus}(NN_\lambda(x)) + \varepsilon,\ J(x) = \tanh(NN_J(x))$
UDE-MRVCJ	$\frac{dx}{dt} = \kappa_{\text{eff}}(x)\left(\theta_{\text{eff}}(x) - x\right) + \gamma_+\, r_{t-1}^2 + \lambda(x)\,J(x)$

Integrator choice: Because the volatility data are uniformly sampled at daily frequency, we employ a fixed-step fourth-order Runge–Kutta (RK4) integrator with step size $h = 1$ day for all models. This choice avoids unnecessary adaptive stepping overhead while maintaining high-order accuracy and deterministic reproducibility across thousands of rolling windows. Although adaptive solvers are beneficial for irregularly sampled data, our setting benefits more from the continuous-time formulation's structural properties: learning a smooth vector field that governs instantaneous rates of change rather than discrete one-step transitions. This continuous representation reduces discretization bias and promotes trajectory-level stability during multi-step rollouts, even under regular sampling.

4 Experiments

We evaluate our UDE-based volatility forecasting framework using rigorous rolling walk-forward validation across diverse financial assets. Our experimental design tests model robustness across varying market dynamics, volatility regimes, and stress periods spanning nearly two decades.

4.1 Datasets and Evaluation Protocol

Asset Classes: We evaluate on 10 financial indices spanning major global markets and asset classes: developed and emerging market equities, government bonds, small/large cap indices, and cryptocurrency. This diverse collection exhibits strong temporal dependencies, conditional heteroskedasticity, and stress periods, including the 2008 financial crisis, 2020 COVID crash, and 2022 market turbulence. Daily prices are obtained from Yahoo Finance, and volatility is estimated using the Garman–Klass range-based estimator. We model log-volatility $x_t = \ln(\sigma_t)$ as described in Section 3. Complete details and tickers are provided in Appendix A.3.

Rolling walk-forward validation: We use strictly non-overlapping rolling windows to ensure realistic out-of-sample evaluation. Each window consists of sequential train, validation, and test splits with no data leakage. Windows are rolled forward through the entire sample period (Aug 2005 - Aug 2025), naturally traversing multiple market regimes and crisis periods. This protocol prevents look-ahead bias and provides conservative performance estimates. Full details on window sizes, stride length, and split configuration are provided in Appendix A.4.

Baselines: We compare against statistical models (Random Walk, EWMA, GARCH-family), deep learning approaches (LSTM, GRU), foundation models (TimesFM zero-shot and few-shot), pure mechanistic baselines (ODE-MR), and pure neural ODE baselines (Vanilla Neural ODE). This comprehensive set isolates the contribution of hybrid structural constraints.

4.2 Results

We present the performance evaluation of our UDE-based architecture across multiple forecasting horizons, comparing against traditional econometric models, deep learning baselines, and foundation models. Table 2 shows aggregated results across horizons 1, 5, 10, and 20 steps ahead.

Table 2: Normalized RMSE (mean ± 95% CI). Normalized by standard deviation of observed volatility. Scale-independent metric, lower is better.

Model	**Type**	**NRMSE** $h = 1$	**NRMSE** $h = 5$	**NRMSE** $h = 10$	**NRMSE** $h = 20$
Random Walk	Baseline	0.770 ± 0.063	0.841 ± 0.080	0.882 ± 0.084	0.911 ± 0.087
EWMA	Statistical	1.381 ± 0.220	1.442 ± 0.223	1.498 ± 0.232	1.602 ± 0.245
GARCH(1,1)	Econometric	1.384 ± 0.222	1.494 ± 0.234	1.606 ± 0.247	1.852 ± 0.277
FIGARCH(1,1)	Econometric	1.430 ± 0.235	1.548 ± 0.249	1.677 ± 0.267	1.985 ± 0.314
LSTM	Sequence	0.811 ± 0.052	0.825 ± 0.048	0.855 ± 0.049	0.965 ± 0.047
GRU	Sequence	0.803 ± 0.048	0.817 ± 0.046	0.818 ± 0.048	0.906 ± 0.046
TimesFM	Transformer	0.704 ± 0.055	0.782 ± 0.051	0.794 ± 0.053	0.819 ± 0.050
TimesFM few-shot	Transformer	0.701 ± 0.054	0.780 ± 0.050	0.791 ± 0.052	0.814 ± 0.049
Vanilla Neural ODE	Neural ODE	0.735 ± 0.056	0.812 ± 0.073	0.949 ± 0.228	0.998 ± 0.090
ODE-MR	Physics	0.758 ± 0.080	0.985 ± 0.070	0.997 ± 0.069	1.020 ± 0.085
UDE-MR	UDE	0.662 ± 0.056	0.740 ± 0.050	0.753 ± 0.050	0.780 ± 0.050
UDE-MRVC	UDE	0.660 ± 0.056	0.738 ± 0.051	0.749 ± 0.052	0.774 ± 0.052
UDE-MRVCJ	UDE	**0.658** ± 0.052	**0.735** ± 0.048	**0.743** ± 0.049	**0.759** ± 0.048

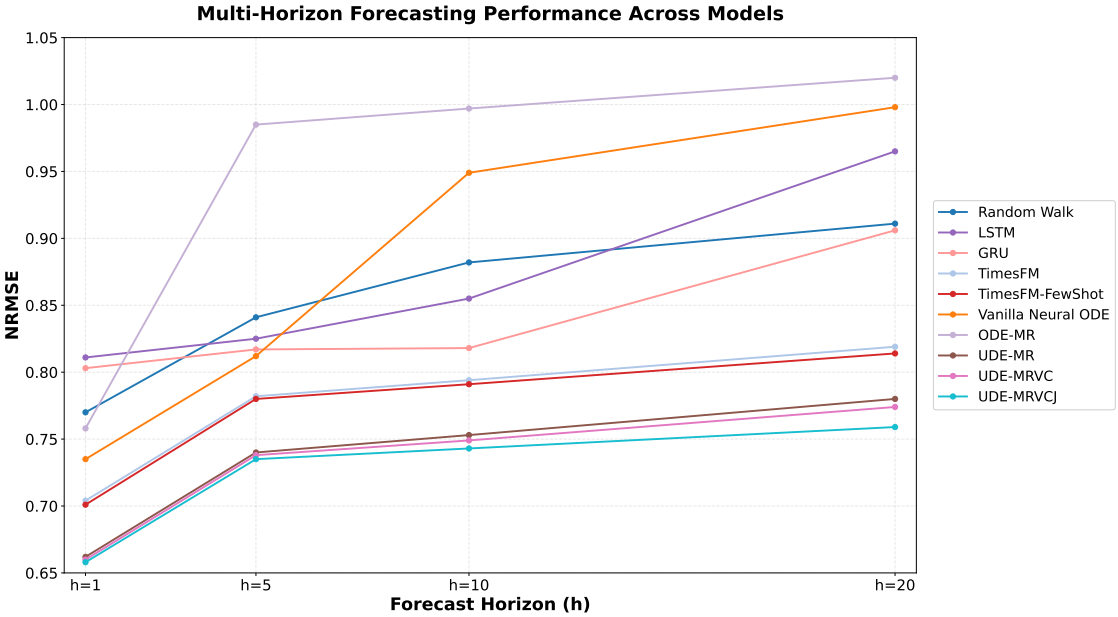


Figure 1: Multi-horizon forecasting accuracy (NRMSE; lower is better) at horizons $h \in \{1, 5, 10, 20\}$ for baseline (Random Walk), sequence models (LSTM, GRU), transformer models (TimesFM, TimesFM-FewShot), and Neural-ODE/UDE variants. The UDE-MRVCJ model (incorporating mean reversion, volatility clustering, and jumps) achieves the best performance across all horizons. Curves show mean results over rolling evaluation windows; lower lines indicate more accurate forecasts.

Table 3: Diebold-Mariano significance across assets (combined p-values, Fisher). Lower p indicates model 2 significantly outperforms model 1.

Pair	$h = 1$	$h = 5$	$h = 10$	$h = 20$
Vanilla Neural ODE → UDE-MR	3.1e-4	1.8e-5	2.2e-6	8.5e-7
UDE-MR → UDE-MRVC	0.0170	0.0857	0.0709	0.0386
UDE-MRVC → UDE-MRVCJ	0.0412	0.0013	0.0034	0.0189

We evaluate significance using the Diebold–Mariano (DM) test across rolling windows for each asset. Because the selected assets span distinct classes (equity indices, emerging-market indices, and crypto) with low cross-correlations in realized volatility, we treat test outcomes as approximately independent. Combined p-values are therefore computed using Fisher's method.

4.2.1 Performance Analysis

UDEs combine accuracy and stability. UDE variants slightly outperform the foundation model (TimesFM: 0.70; UDE-MRVCJ: 0.66) at short horizons. Across longer horizons, UDEs remain stable (0.66 → 0.76 from $h = 1$ to $h = 20$), whereas purely neural models degrade (Vanilla Neural ODE: 0.74 → 1.00) and rigid mechanistic models stay suboptimal (ODE-MR: 0.76 → 1.02).

Compositional stacking yields monotonic improvements. Table 3 shows statistically significant improvements at each composition step. Adding mean-reversion modulation to Vanilla Neural ODE provides the largest gain (DM p $<$ 3.1e-4 at $h = 1$, p $<$ 8.5e-7 at $h = 20$). Volatility clustering (UDE-MRVC) and jumps (UDE-MRVCJ) provide incremental but consistent improvements, particularly at longer horizons where structural constraints prevent catastrophic degradation.

4.2.2 Computational Efficiency

Table 4: The efficiency table highlight that UDE variants are *orders of magnitude* smaller and have a smaller footprint than large neural and foundation baselines while preserving long-horizon stability

Model	Params	Memory (MB)	Train (s)	Infer (s)	Total (s)	Epoch (s)
Random Walk	0	0.00	0.00	0.00	0.00	0.000
EWMA	1	0.05	0.00	0.00	0.00	0.000
GARCH(1,1)	4	0.10	0.36	0.00	0.36	0.000
FIGARCH(1,1)	5	0.10	1.11	0.00	1.11	0.000
LSTM	50.5K	0.19	0.49	0.19	0.68	0.010
GRU	37.9K	0.14	0.46	0.18	0.64	0.009
ODE-MR	2	0.00	0.50	0.02	0.52	0.005
Vanilla Neural ODE	4.4K	0.02	1.02	0.05	1.07	0.010
UDE-MR	4.4K	0.02	1.32	0.09	1.42	0.013
UDE-MRVC	4.4K	0.02	1.46	0.10	1.56	0.014
UDE-MRVCJ	4.4K	0.03	1.54	0.11	1.65	0.016
TimesFM	200.0M	800.00	N/A	25.52	25.52	0.000
TimesFM-FewShot	200.0M	800.00	N/A	25.25	25.25	0.000

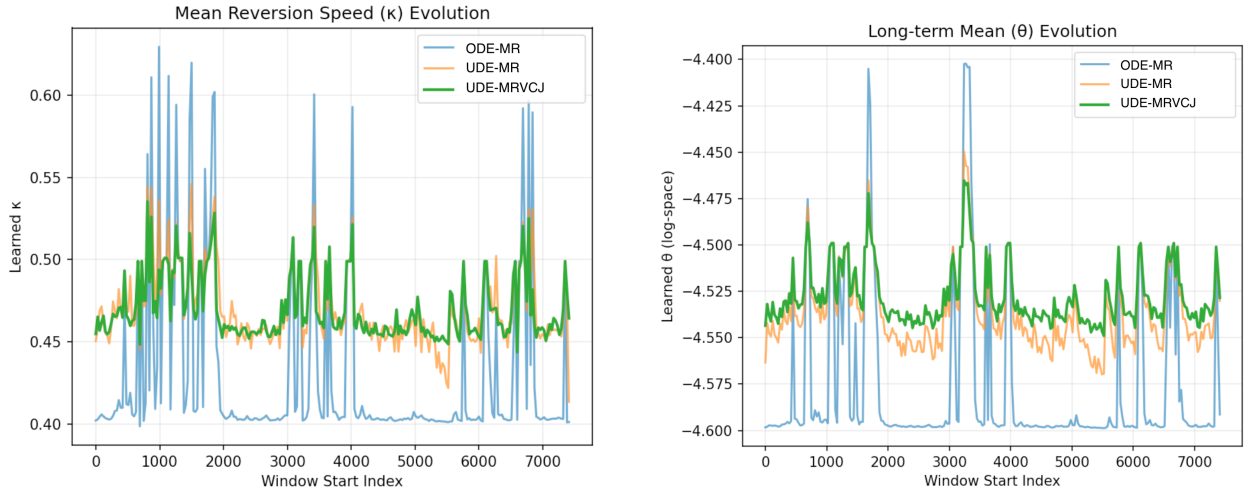
4.3 Ablation Study

Table 5: Ablation study of model components for multi-step volatility forecasting. R^2 measures explained variance; sMAPE measures scale-independent forecast accuracy (lower is better). Progressive addition of structural components improves performance, though high volatility noise explains the low R^2 at $h = 20$.

Model	**Structure**	$h = 1$		$h = 20$	
		R^2	**sMAPE**	**R^2**	**sMAPE**
ODE-MR	Pure Mechanistic	0.421	7.91	-0.198	11.58
Vanilla Neural ODE	Pure Neural	0.456	7.50	-0.136	10.46
UDE-MR	+ Mean Reversion	0.518	7.17	0.150	9.37
UDE-MRVC	+ Vol. Clustering	0.522	7.14	0.151	9.36
UDE-MRVCJ	+ Jumps	0.535	7.03	0.160	9.11

4.4 Interpretability via Parameter Evolution (S&P 500)

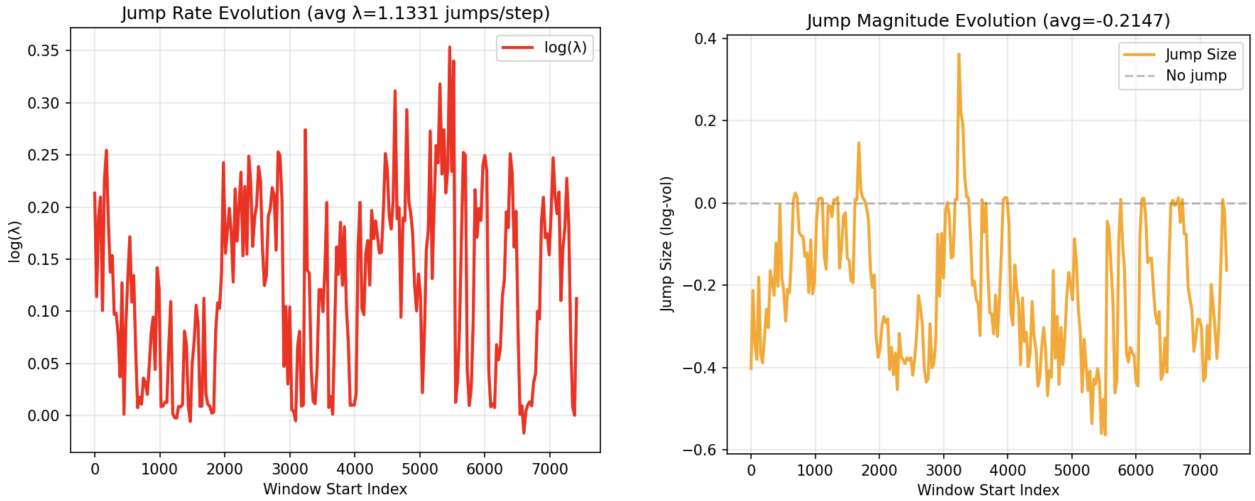
We track how the learned parameters of our UDE–MRVCJ evolve across rolling windows for the S&P 500. This reveals where the model adapts and where it remains invariant. First, the mean–reversion speed $\kappa(x)$ and the long–run level $\theta(x)$ co-move with regime shifts: κ decreases during prolonged drawdowns (slower pullback toward equilibrium) and θ rises when background volatility is persistently elevated. Second, the jump mechanism is event–sensitive: the estimated jump *rate* $\lambda(x)$ exhibits pronounced spikes around market stress (e.g., macro announcements, crisis windows), while the jump *magnitude* concentrates within those episodes, indicating a parsimonious "rare event" channel rather than persistent overfitting. Third, the volatility–clustering strength (γ) varies smoothly and remains bounded, capturing short–horizon persistence without destabilizing the core dynamics. Finally, a stability summary and correlation heatmap show that core parameters (κ, θ) remain stable across windows and are only weakly confounded with jump/clustering terms, supporting identifiability and interpretability. Together, these diagnostics let us explain forecast changes (e.g., "volatility increased because mean reversion slowed and jump intensity spiked") rather than treat predictions as opaque.



(a) Mean–reversion speed $\kappa(x)$ across windows (S&P 500).

(b) Long–run level $\theta(x)$ across windows (S&P 500).

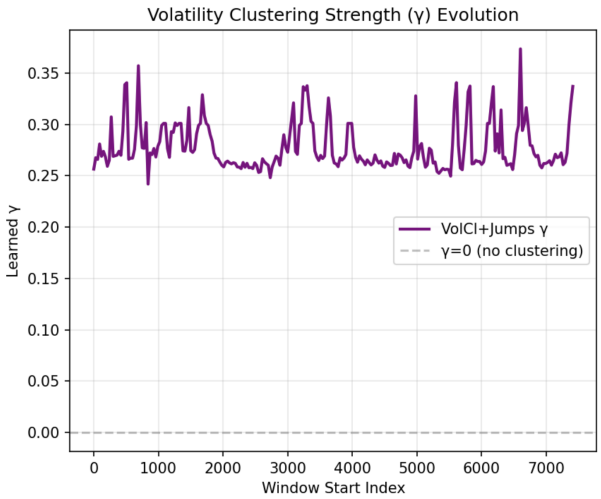
Figure 2: Core mean–reverting components adapt with regime changes while remaining well–behaved.



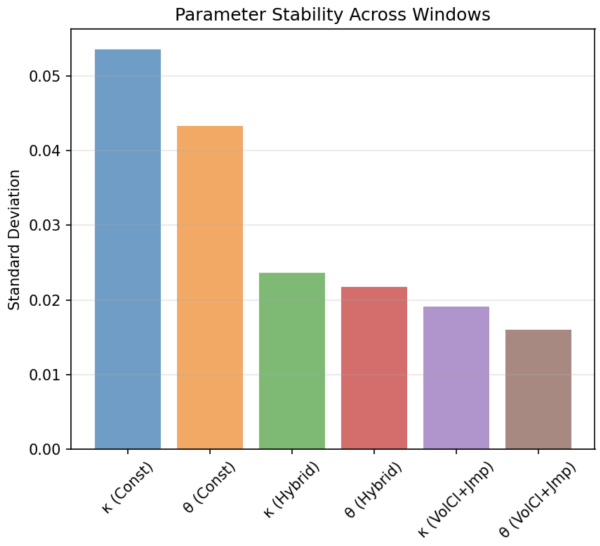
(a) Jump rate $\lambda(x)$: spikes at stress periods (S&P 500).

(b) Jump magnitude: concentrated during tail episodes (S&P 500).

Figure 3: Event–sensitivity: jumps activate during rare episodes rather than persistently.

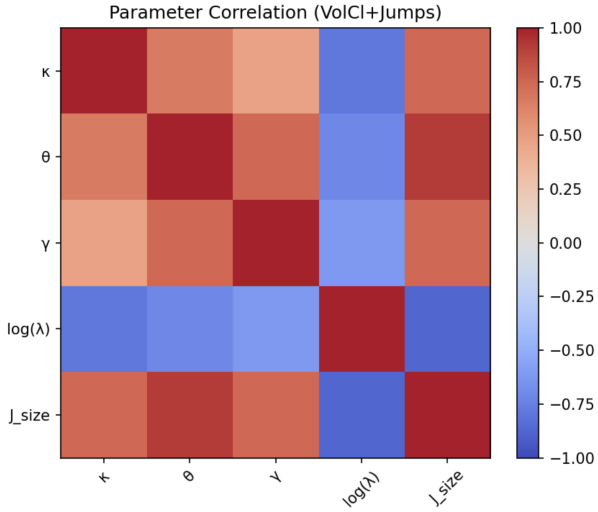


(a) Clustering strength γ: smooth, bounded variation (S&P 500).

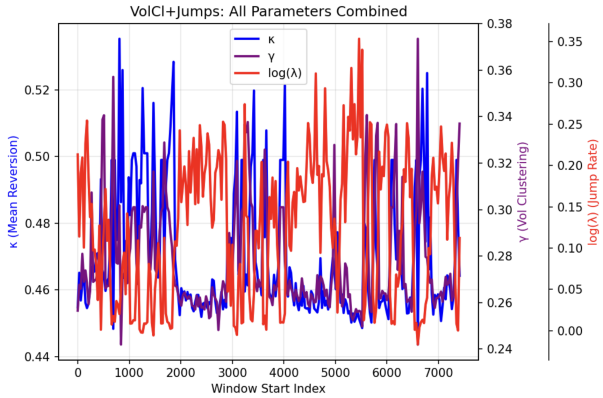


(b) Stability summary: core parameters vary modestly (S&P 500).

Figure 4: Short–horizon persistence is captured by a bounded clustering channel; core structure remains stable.



(a) Parameter correlation: weak confounding across channels (S&P 500).



(b) Composite view: clustering + jumps alignment with stress (S&P 500).

Figure 5: Identifiability: mean–reversion parameters are largely orthogonal to jump/clustering terms.

5 Limitations and Future Work

Point forecasts vs probabilistic predictions. Our framework produces deterministic point forecasts rather than distributional predictions with uncertainty quantification. While sufficient for demonstrating Multi-step stability, risk management applications require value-at-risk estimation or full predictive distributions. Neural Stochastic Differential Equations (Neural SDEs) offer a natural extension by adding learned diffusion terms:

$$dx/dt = \kappa_{\text{eff}}(x)[\theta_{\text{eff}}(x) - x] + \gamma_{\text{eff}}(x)dW_t \tag{7}$$

where $\gamma_{\text{eff}}(x)$ models state-dependent volatility and W_t represents Brownian motion. However, Neural SDEs remain notoriously difficult to train: adjoint methods require solving backward through stochastic processes with numerical instability (16). In contrast, the diffusion coefficient design requires careful constraints to prevent explosive trajectories (20). Recent advances in stable Neural SDE classes and signature-based training (15) suggest promising directions, but adapting these to compositional UDE frameworks remains open work.

Limited exogenous features: We use only lagged squared returns as exogenous input. Incorporating macroeconomic indicators, market microstructure features, or cross-asset correlations could improve regime detection and forecasting accuracy, though this risks overfitting and reduces generality.

Domain generalization: While financial volatility provides an excellent testbed due to documented stylized facts and accessible data, demonstrating the framework's effectiveness across diverse domains would strengthen claims of general methodological contribution. Each domain would require careful physics term design and validation against domain-specific baselines.

Despite these limitations, our core results remain robust: UDE variants with inductive biases achieve stable multi-step forecasting with dramatic parameter efficiency and interpretable dynamics. Extensions to probabilistic modeling and broader domain applications represent natural progressions of the framework.

6 Conclusion

We demonstrated that Universal Differential Equations (UDEs) provide an effective framework for multi-step time series forecasting, achieving superior accuracy and long-horizon stability across diverse financial regimes, including crises and structural breaks. Our compositional UDE variants consistently outperform foundation, econometric, and deep learning baselines while using orders of magnitude fewer parameters.

UDEs and neural differential equations more broadly are particularly well suited to forecasting because they model temporal evolution as a **continuous flow** rather than a fixed-step recurrence. Even with regularly sampled data, this formulation enforces trajectory-level smoothness and reduces discretization bias, the main driver of error accumulation in recurrent architectures. By learning instantaneous rates of change instead of discrete transitions, UDEs maintain stable dynamics over extended horizons.

More fundamentally, UDEs offer a principled bridge between mechanistic and data-driven modeling. By embedding domain structure ,mean reversion, volatility clustering, and jump effects , within a differential equation while allowing neural modulation of key coefficients, they preserve **stabilizing inductive biases** that confine the learned dynamics to economically plausible regimes. Our compositional framework showed that each successive incorporation of domain-informed structure yields **monotonic, statistically significant gains**, demonstrating that carefully chosen inductive biases can substitute for brute-force scale.

This combination of continuous-time modeling and structured knowledge integration explains why compact UDE models can rival or surpass massive foundation models. Structural bias, when aligned with the data-generating process, not only enhances efficiency and interpretability but also delivers robust multi-step forecasts that remain coherent across regimes ,an essential property for modeling inherently unstable systems like financial volatility.

References

[1] Leif Andersen. Efficient simulation of the Heston stochastic volatility model. *Risk Management*, 2007.

[2] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986.

[3] Federico Bucci. Deep learning volatility: A finance application. *arXiv preprint arXiv:2006.10463*, 2020.

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

[5] Ting Chen, Zhi Chen, and Bin Han. Hybrid models of garch and neural networks in financial volatility forecasting. *Expert Systems with Applications*, 176:114908, 2021.

[6] Fabienne Comte and Eric Renault. Long memory in continuous-time stochastic volatility. *Mathematical Finance*, 1998.

[7] Rama Cont. Empirical properties of asset returns: Stylized facts and statistical issues. *Quantitative Finance*, 1(2):223–236, 2001.

[8] Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models with differentiable physics. *Nature Communications*, 11(1):1–9, 2020.

[9] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *International Conference on Machine Learning (ICML)*, 2024. arXiv preprint arXiv:2310.10688.

[10] Mark B. Garman and Michael J. Klass. On the estimation of security price volatilities from historical data. *Journal of Business*, 53(1):67–78, 1980.

[11] Jim Gatheral et al. Volatility is rough. *Quantitative Finance*, 2018.

[12] Lawrence R. Glosten, Ravi Jagannathan, and David E. Runkle. On the relation between the expected value and the volatility of the nominal excess return on stocks. *Journal of Finance*, 48(5):1779–1801, 1993.

[13] Peter R. Hansen and Asger Lunde. A forecast comparison of volatility models: Does anything beat a garch(1,1)? *Journal of Applied Econometrics*, 20(7):873–889, 2005.

[14] Andrew C. Harvey, Esther Ruiz, and Neil Shephard. Multivariate stochastic variance models. *The Review of Economic Studies*, 61(2):247–264, 1994.

[15] Zacharia Issa, Blanka Horvath, Maud Lemercier, and Cristopher Salvi. Non-adversarial training of neural sdes with signature kernel scores, 2023.

[16] Patrick Kidger, James Foster, Xuechen Li, and Terry J. Lyons. Neural SDEs as infinite-dimensional GANs. In *International Conference on Machine Learning (ICML)*, pages 5453–5463, 2021.

[17] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104, 2018.

[18] Philippe Masset. Volatility stylized facts. *SSRN Electronic Journal*, 2011.

[19] Yuqi Nie, Yulin He, Qingsong Wen, Liang Sun, Rong Jin, and Ziqing Ma. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

[20] YongKyung Oh, Dong-Young Lim, and Sungil Kim. Stable neural stochastic differential equations in analyzing irregular time series data, 2025.

[21] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Ramadhan. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

[22] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[23] Souhaib Ben Taieb and Rob J. Hyndman. Recursive and direct multi-step forecasting: The best of both worlds. In *Proc. International Conference on Neural Information Processing*, pages 30–40, 2012.

[24] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems*, volume 34, pages 22419–22430, 2021.

[25] Dacheng Xiu and Siem Jan Koopman. The stochastic volatility model: A review. *Journal of Econometrics*, 187(2):433–440, 2015.

[26] 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)*, volume 35, pages 11106–11115, 2021.

A Appendix

A.1 Lipschitz Continuity for UDE-MRVCJ

We consider the one-dimensional ODE on log-volatility

$$\frac{dx}{dt} \;=\; f(x; r_{t-1}^2) \;=\; \underbrace{\kappa_{\text{eff}}(x)\,\big(\theta_{\text{eff}}(x) - x\big)}_{\text{mean reversion}} \;+\; \underbrace{\gamma_+\, r_{t-1}^2}_{\text{vol. clustering}} \;+\; \underbrace{\lambda(x)\,J(x)}_{\text{expected jumps}}\;,$$

where

$$\kappa_{\text{eff}}(x) = \kappa_0\,\exp\big(s\,\tanh(k(x))\big), \qquad \theta_{\text{eff}}(x) = \theta + b\,\tanh(\vartheta(x)),$$
$$\lambda(x) = \text{softplus}(\ell(x)) + \varepsilon, \qquad J(x) = \tanh\big(j(x)\big).$$

Here $\kappa_0 = \text{softplus}(\kappa) + \varepsilon > 0$ is constant, $s > 0$ is the modulation scale, $b > 0$ is the θ-shift bound , and $\varepsilon > 0$ is a fixed numerical slack. The maps $k, \vartheta, \ell, j : \mathbb{R} \to \mathbb{R}$ are the scalar outputs of two MLPs with GELU activations (one for $\{k, \vartheta\}$ and one for $\{\ell, j\}$). The clustering input r_{t-1}^2 is exogenous and treated as constant in x at each step. We assume the MLPs are globally Lipschitz with constants L_{base} (for k, ϑ) and L_{jump} (for ℓ, j). For GELU networks this holds with

$$L_{\text{MLP}} \;\le\; \Big(\prod_i \|W_i\|_2\Big)\,(1.129)^{\#\text{activations}}.$$

We write $K = [-M, M]$ for a compact domain of interest and use $\|\cdot\|$ for the Euclidean norm on $\mathbb{R}$.

Auxiliary bounds: (i) Since $|\tanh'| \le 1$ and $(\text{softplus})' = \sigma \in (0, 1)$,

$$|(\tanh \circ g)'| \le |g'|, \qquad |(\text{softplus} \circ g)'| \le |g'|, \qquad \forall g : \mathbb{R} \to \mathbb{R}.$$

(ii) For $x \in K$, define

$$\kappa_{\max} := \kappa_0\, e^s, \qquad L_\kappa := \kappa_0\, e^s\, s\, L_{\text{base}}, \qquad L_\theta := b\, L_{\text{base}}.$$

Then $|\kappa_{\text{eff}}(x)| \le \kappa_{\max}$ and $|\kappa'_{\text{eff}}(x)| \le L_\kappa$; also $|\theta'_{\text{eff}}(x)| \le L_\theta$ and $|\theta_{\text{eff}}(x)| \le |\theta| + b$. (iii) For the jump factors, $|J(x)| \le 1$ and

$$|\lambda'(x)| \le L_{\text{jump}}, \qquad |J'(x)| \le L_{\text{jump}}.$$

Let

$$\Lambda(M) := \sup_{x \in K} \lambda(x) \;\le\; \log 2 + |\ell(0)| + L_{\text{jump}}\, M + \varepsilon$$

(using $\text{softplus}(t) \le \log(1 + e^{|t|}) \le \log 2 + |t|$ and $|\ell(x)| \le |\ell(0)| + L_{\text{jump}}|x|$).

Theorem A.1 (Local Lipschitz on compact sets)**.** *On any compact $K = [-M, M]$, the vector field $f(\cdot; r_{t-1}^2)$ is Lipschitz continuous in x with constant*

$$L_f(M) \;\le\; \underbrace{L_\kappa\,\big(M + |\theta| + b\big) \;+\; \kappa_{\max}\,(L_\theta + 1)}_{\textit{mean reversion term}} \;+\; \underbrace{L_{\text{jump}}\,(1 + \Lambda(M))}_{\textit{expected jump term}}.$$

The exogenous clustering term $\gamma_+ r_{t-1}^2$ does not affect the Lipschitz constant in x.

Proof. Write $f(x) = g(x)h(x) + c + q(x)$ where $g = \kappa_{\text{eff}}$, $h = \theta_{\text{eff}} - x$, $c = \gamma_+ r_{t-1}^2$ (constant in x), and $q = \lambda\, J$. For any $x_1, x_2 \in K$,

$$|g_1 h_1 - g_2 h_2| \;\le\; |g_1 - g_2|\,|h_1| \;+\; |g_2|\,|h_1 - h_2|.$$

Using $|h(x)| \le |\theta_{\text{eff}}(x)| + |x| \le |\theta| + b + M$ and $|h_1 - h_2| \le (L_\theta + 1)|x_1 - x_2|$, plus $|g_1 - g_2| \le L_\kappa |x_1 - x_2|$ and $|g_2| \le \kappa_{\max}$, we obtain the first bracket. For $q = \lambda J$,

$$|q_1 - q_2| \;\le\; |\lambda_1 - \lambda_2|\,|J_1| \;+\; |\lambda_2|\,|J_1 - J_2| \;\le\; \big(L_{\text{jump}} \cdot 1 + \Lambda(M)\, L_{\text{jump}}\big)\,|x_1 - x_2|.$$

Summing the bounds yields the stated $L_f(M)$. □

Theorem A.2 (Linear growth and global existence)**.** *There exist constants $C_0(M), C_1 \geq 0$ such that*

$$|f(x; r_{t-1}^2)| \;\leq\; C_0(M) + C_1\,|x|, \qquad \forall x \in \mathbb{R}.$$

One admissible choice is

$$C_1 \;=\; \kappa_{\max} + L_{\text{jump}}, \qquad C_0(M) \;=\; \kappa_{\max}(|\theta| + b) + |\gamma_+|\,|r_{t-1}^2| + \log 2 + |\ell(0)| + \varepsilon,$$

using $\lambda(x) \leq \log 2 + |\ell(0)| + L_{\text{jump}}|x| + \varepsilon$. Consequently, by Picard–Lindelöf and Grönwall, the IVP admits a unique solution globally in time.

A.2 Convergence of RK4 for Our Constrained Neural Mean–Reversion with Clustering and Jumps

We study our model

$$\dot{x}(t) \;=\; f\big(x(t), u(t)\big) \;=\; \underbrace{\kappa_{\text{eff}}(x)\,\big(\theta_{\text{eff}}(x) - x\big)}_{\text{mean reversion}} \;+\; \underbrace{\gamma_+\,u(t)}_{\text{vol. clustering}} \;+\; \underbrace{\lambda(x)\,J(x)}_{\text{expected jumps}},$$

with

$$\kappa_{\text{eff}}(x) = \kappa_0\,\exp\big(s\,\tanh(k(x))\big), \qquad \theta_{\text{eff}}(x) = \theta + b\,\tanh(\vartheta(x)),$$
$$\lambda(x) = \text{softplus}\big(\ell(x)\big) + \varepsilon, \qquad J(x) = \tanh\big(j(x)\big),$$

where k, ϑ, ℓ, j are scalar outputs of MLPs with exact GELU activations, and $u(t) = r_{t-1}^2$ is a bounded exogenous regressor. In our solver, we hold $u(t)$ piecewise–constant on each RK4 step (matching our experimental procedure).

Let $[0, T]$ be fixed and $K \subset \mathbb{R}$ be a compact set that contains the true trajectory and all RK4 stage values (for h small enough).

Hypotheses we verify. - (H1) Uniform Lipschitz in x on K: by our Lipschitz result in Appendix A.1, there exists $L \geq 0$ such that $|f(x_1, u) - f(x_2, u)| \leq L\,|x_1 - x_2|$ for all $x_1, x_2 \in K$ and admissible u. - (H2) Smoothness in x on K: $\exp, \tanh, \text{softplus}$ and exact GELU are C^∞; compositions with affine maps preserve C^∞, so $f(\cdot, u) \in C^\infty(K)$ uniformly in u. Hence $\partial_x^m f(\cdot, u)$ are bounded on K for $m = 0, \ldots, 4$. - (H3) Stage confinement: since f is bounded on K, there exists $h_0 > 0$ such that for $0 < h \leq h_0$ all RK4 stages remain in K.

We apply classical RK4 with step $h > 0$, freezing $u(t) \equiv u_n$ on $[t_n, t_{n+1}]$, $t_n = nh$:

$$k_1 = f(x_n, u_n), \quad k_2 = f\big(x_n + \tfrac{h}{2}k_1, u_n\big), \quad k_3 = f\big(x_n + \tfrac{h}{2}k_2, u_n\big),$$
$$k_4 = f\big(x_n + hk_3, u_n\big), \qquad x_{n+1} = x_n + \tfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4).$$

Lemma A.3 (Local truncation error (order 5))**.** *There exists $C_{\text{loc}} = C_{\text{loc}}(K, f)$ such that, for all $x \in K$ and admissible (frozen) u,*

$$\big\|\varphi_h(x, u) - \Phi_h(x, u)\big\| \;\leq\; C_{\text{loc}}\,h^5,$$

where φ_h is the exact time-h flow of $x' = f(x, u)$ and Φ_h is one RK4 step.

Proof. By (H2), $f(\cdot, u) \in C^4(K)$ uniformly in u. The Taylor–Butcher expansion and RK4 order conditions give a uniform $\mathcal{O}(h^5)$ remainder. (H3) guarantees stage evaluations stay in K where the bounds apply. □

Lemma A.4 (One–step Lipschitz stability)**.** *There exist $c_1 = c_1(K, f)$ and $h_1 > 0$ such that, for $0 < h \leq h_1$ and all $x, y \in K$ and admissible u,*

$$\big\|\Phi_h(x, u) - \Phi_h(y, u)\big\| \;\leq\; (1 + c_1 h)\,\|x - y\|.$$

Proof. Write $\Phi_h(x, u) = x + h\,\Psi_h(x, u)$ with stage–averaged slope Ψ_h. By (H1)–(H2), $\Psi_h(\cdot, u)$ is Lipschitz in x with constant $L + \mathcal{O}(h)$ uniformly on K. Hence $\|\Phi_h(x, u) - \Phi_h(y, u)\| \leq (1 + h(L + C'))\|x - y\|$ for some C', proving the claim. □

Theorem A.5 (Global $\mathcal{O}(h^4)$ error for our model)**.** *Under* (H1)–(H3) *there exists* $C = C(K, f)$*, independent of* h *and* n*, such that*

$$\max_{0 \le n \le N} \left\| x(t_n) - x_n \right\| \;\le\; C\, e^{LT}\, h^4, \qquad N = \lfloor T/h \rfloor.$$

Proof. On $[t_n, t_{n+1}]$ with frozen u_n, the exact solution satisfies $x(t_{n+1}) = \varphi_h(x(t_n), u_n)$, so

$$e_{n+1} := x(t_{n+1}) - x_{n+1} = \big[\Phi_h(x(t_n), u_n) - \Phi_h(x_n, u_n)\big] + \tau_{n+1}, \qquad \|\tau_{n+1}\| \le C_{\mathrm{loc}} h^5.$$

By Lemma A.4, $\|e_{n+1}\| \le (1 + c_1 h)\|e_n\| + C_{\mathrm{loc}} h^5$. Discrete Grönwall yields $\|e_n\| \le C e^{c_1 t_n} h^4 \le C e^{LT} h^4$, since $c_1 \ge L$ and $e_0 = 0$. □

Why the hypotheses hold for our architecture. - Mean reversion term: κ_{eff} and θ_{eff} are C^∞ compositions of $\exp, \tanh$ and affine maps; on K they and their first four derivatives are bounded. Our Lipschitz constant L on K was established in Appendix A.1. - Clustering term: $\gamma_+ u(t)$ is constant in x and bounded in t, so it does not affect Lipschitz in x nor the order of the method. - Jump term: $\lambda = \mathrm{softplus} \circ \ell$ and $J = \tanh \circ j$ are C^∞; their products and derivatives are bounded on K. - Stage confinement: since f is bounded on K, taking $h \le h_0(K, f)$ ensures stage values remain in K.

A.3 Datasets

We evaluate on the following 10 indices spanning major global markets and asset classes:

- **Developed Market Equities:** S&P 500 (`^GSPC`), NASDAQ (`^IXIC`), Dow Jones (`^DJI`)
- **Emerging Market Equities:** NIFTY50 (`^NSEI`)
- **Small/Large Cap:** Russell 1000 (`^RUI`), Russell 2000 (`^RUT`)
- **Government Bonds:** iShares Core U.S. Aggregate Bond ETF (`AGG`)
- **Cryptocurrency:** Bitcoin (`BTC-USD`), Ethereum (`ETH-USD`)
- **International:** FTSE 100 (`^FTSE`)

Daily OHLC prices are obtained via `yfinance`. Realized volatility is estimated using the Garman–Klass estimator (Eq. 1), and we model log-volatility $x_t = \ln(\sigma_t)$. Evaluation period: August 2005 to August 2025.

A.4 Evaluation Protocol

Rolling window configuration. We use the following strictly temporal configuration:

- **Train:** 63 trading days (≈3 months)
- **Validation:** 20 days
- **Test:** 30 days
- **Stride:** 30 days between consecutive window start dates

We selected the training window length via a small hyperparameter sweep over candidate lookback periods 7, 21, 63, 126, 252 trading days and evaluated normalized MSE on held-out rolling windows across all assets. The 63-day window produced the lowest aggregate NRMSE across assets. Intuitively, longer windows (126, 252) often span multiple volatility regimes, causing learned modulators to average over heterogeneous dynamics and thereby reduce predictive utility; visually, parameter trajectories estimated from 126/252-day fits show piecewise regime structure that a single parameter set cannot capture reliably. A short window (7, 21) lacks sufficient information to estimate even the compact neural modulators stably. Thus, the 63-day choice represents an empirical tradeoff between enough data to fit modulators and narrowness of the regime assumed.

Walk-forward guarantee of no data leakage. The walk-forward protocol ensures strictly out-of-sample testing by moving forward in time only. For each window, we:

1. Train a model on the training segment
2. Validate on the validation segment
3. Make predictions on the test segment
4. Roll forward by stride days and repeat

Critically, **we never revisit past test periods.** Once predictions are made for a test segment, that window is complete. Even though future training windows may include data that was previously used as test data, we never re-predict those periods with the benefit of future information the predictions have already been made and recorded using only past data. This is the defining property of walk-forward validation: we only predict forward in time, never backward.

For a train size of 63 days, validation size of 20 days, test size of 30 days, and stride of 30 days:

- Window 1: Train on days 1-63, Val on days 64-83, Test on days 84-113
- Window 2: Train on days 31-93, Val on days 94-113, Test on days 114-143
- Window 3: Train on days 61-123, Val on days 124-143, Test on days 144-173

By Window 3, days 84-113 (Window 1's test set) now appear in the training data. However, predictions for days 84-113 were already made in Window 1 using only data up to day 83. We do not re-predict them. This ensures that every prediction uses only information available at the time the prediction would have been made in real-world deployment.

Forecast horizons and autoregressive rollout. We evaluate at $h \in \{1, 5, 10, 20\}$ days ahead. For $h = 20$, each 30-day test window yields 10 independent predictions (days 1-20, 2-21, ..., 10-29 of the test period).

All predictions are made using **strictly autoregressive rollout with no future data leakage**. When predicting at time t for horizon h, the model uses only information available up to time t. Critically:

- For multi-step predictions ($h > 1$), the model recursively feeds its own predictions forward
- No ground-truth values from $t + 1$ to $t + h - 1$ are provided during the rollout
- Exogenous variables (e.g., squared returns r_{t-1}^2) use only strictly causal values: when predicting at time t, we use r_{t-1}^2 (available), never r_t^2 or future returns
- At prediction time t for target $t + h$, the model has no access to any data from time $t + 1$ onward

This ensures realistic evaluation: predictions are made exactly as they would be in real-world deployment, where future information is unavailable.

Example of autoregressive rollout. To predict log-volatility at $t + 20$ starting from time t:

1. Input: observed log-volatilities up to t, exogenous features up to $t - 1$ (e.g., r_{t-1}^2)
2. Predict $\hat{x}_{t+1}$ using the ODE integrated from t to $t + 1$
3. Predict $\hat{x}_{t+2}$ using $\hat{x}_{t+1}$ as input (not ground truth x_{t+1})
4. Continue rolling forward: $\hat{x}_{t+3}$ from $\hat{x}_{t+2}$, ..., $\hat{x}_{t+20}$ from $\hat{x}_{t+19}$

5. Exogenous features at each step use only past information (e.g., at step $t+k$, use predicted $\hat{r}^2_{t+k-1}$ derived from $\hat{x}_{t+k-1}$)

This protocol tests the model's ability to maintain stability under compounding error, the key challenge in multi-step forecasting.

A.5 Training Configuration

Unless stated otherwise, we use:

- **Seeds:** 42, 43, 44 (results aggregated across all 3 seeds and 10 assets)
- **Horizon in loss:** $n_{\text{steps}} = 20$
- **Batch size:** 50
- **Networks:** Small MLP modulators with GELU activations
  - Hidden sizes tested: [64,64], [64,64,64], [64,64,64,64], [16,16],[32,32]
  - Selected: [64,64] (no further improvement with deeper architectures)
- **Optimizer:** Adam with learning rate 1×10^{-3}
- **Epochs:** 50 (with early stopping on validation loss)
- **ODE Integrator:** RK4 with fixed step size
- **Constraints:** Softplus for mean-reversion rate ($\kappa_+ > 0$), bounded shifts for long-run level (θ_{eff})

A.6 Compute Environment

Experiments were run on Linux/macOS with Python 3.10–3.11. GPU was optional; CPU results were consistent modulo runtime.

Core libraries:

- PyTorch
- Neuromancer for ODE integration, systems, and losses
- NumPy $\geq$ 1.26, SciPy $\geq$ 1.11, Pandas $\geq$ 2.0, Matplotlib $\geq$ 3.8

A.7 Evaluation Metrics

NRMSE. Normalized Root Mean Squared Error, normalized by the standard deviation of observed volatility (scale-independent metric, lower is better).

Statistical significance. We report Diebold–Mariano tests (Fisher combined across assets) to compare model pairs using two-sided squared-error differentials. No multiple-comparison correction is applied. Combined p-values are reported in Table **??**.

A.8 Econometric Baselines

GARCH-family models (GARCH, EGARCH, FIGARCH) were estimated using the `arch` Python library. Models were refit in each rolling window via maximum likelihood. Convergence was verified; the best-performing configuration after convergence is reported.