The Expressivity-Stability Tradeoff in Physics-Informed Neural Networks: Why Complex Architectures Fail at Autoregressive Dynamics Prediction

Anonymous Author(s)
anonymous@example.com

Abstract

Physics-Informed Neural Networks (PINNs) embed governing equations into neural networks to learn dynamics from data. While prior work evaluates PINNs using single-step prediction accuracy, practical deployment in control, simulation, and forecasting requires stable multi-step *autoregressive rollout* where predictions recursively feed as inputs. We demonstrate a fundamental *expressivity-stability tradeoff*: architectural modifications that improve single-step accuracy by 2–10$\times$ can catastrophically destabilize autoregressive rollouts by 10^2–$10^6\times$, causing prediction errors to explode from centimeters to kilometers within 100 steps. Through systematic experiments on a 12-dimensional nonlinear dynamical system (6-DOF quadrotor), we identify two failure mechanisms: (1) modular architectures break implicit dynamic coupling, causing coordinated subsystem errors to accumulate independently then interact catastrophically, and (2) Fourier feature embeddings suffer severe distribution shift during rollout, where small state perturbations map to large feature-space discontinuities. We propose a stability-oriented training framework combining curriculum learning over increasing horizons, scheduled sampling to expose models to their own errors, and physics-based regularization. This approach achieves 51$\times$ improvement in 100-step prediction accuracy while maintaining parameter identification. Our results challenge the assumption that more expressive architectures yield better dynamics models and establish that standard single-step evaluation metrics fundamentally mislead about autoregressive deployment performance.

1 Introduction

Learning accurate dynamics models is fundamental to model-based control, simulation, and forecasting. Physics-Informed Neural Networks (PINNs) [**?**] have emerged as a principled approach, embedding governing equations directly into neural network training. This physics-constrained learning enables data-efficient training, physically consistent predictions, and simultaneous parameter identification.

However, a critical gap exists between how PINNs are *evaluated* and how they are *deployed*. Standard benchmarks assess single-step prediction: given ground truth state $\mathbf{x}_t$, predict $\hat{\mathbf{x}}_{t+1}$. In contrast, model predictive control, trajectory optimization, and long-horizon simulation require *autoregressive rollout*: predictions $\hat{\mathbf{x}}_{t+1}$ feed back as inputs to predict $\hat{\mathbf{x}}_{t+2}$, compounding errors over dozens to hundreds of steps.

We demonstrate that these evaluation regimes yield fundamentally contradictory conclusions. Specifically, we uncover an **expressivity-stability tradeoff**: architectural modifications that improve single-step accuracy by 2–10$\times$ can degrade 100-step autoregressive performance by 10^2–$10^6\times$. A

model achieving 0.009m single-step error can diverge to over 5 million meters in 100 steps—rendering it completely unusable for any control application.

Contributions. This paper makes four contributions:

1. **Stability Envelope Formalization.** We introduce the stability envelope H_ϵ—the maximum prediction horizon where error remains bounded below threshold ϵ—providing the first formal metric for autoregressive stability in PINNs (Section 4).
2. **Expressivity-Stability Tradeoff.** We establish the first empirical demonstration of a fundamental inverse relationship between local (single-step) accuracy and global (multi-step) stability in physics-informed learning. We show this tradeoff spans 2–6 orders of magnitude (Section 4).
3. **Failure Mode Characterization.** We identify two architectural failure mechanisms: (i) modular architectures break dynamic coupling causing independent error accumulation, and (ii) Fourier embeddings amplify distribution shift exponentially during rollout. We unify these under a frequency-coupling stability law (Section 5, 6).
4. **Stability-Oriented Training.** We develop a curriculum-based training protocol (horizon scheduling + scheduled sampling + physics regularization) that enlarges the stability envelope by $51\times$ while maintaining accurate dynamics prediction (Section 7).

2 Related Work

Physics-Informed Neural Networks. **?** introduced PINNs for solving differential equations by embedding physics into training losses. Extensions include learning operators [**?**], handling noisy data [**?**], and applications to fluid dynamics [**?**]. For robotics, PINNs have been applied to manipulators [**?**], continuum robots [**?**], and multirotors [**?**]. However, these works primarily evaluate single-step accuracy rather than multi-step stability.

Distribution Shift in Learned Dynamics. Model-based reinforcement learning extensively studies compounding errors in learned models [**??**]. Ensemble disagreement [**?**] and model uncertainty [**?**] help detect but not prevent divergence. Our work differs by focusing on architectural choices that cause instability independent of model uncertainty.

Exposure Bias and Scheduled Sampling. Sequence models trained with teacher forcing suffer exposure bias—the train-test distribution mismatch when predictions replace ground truth at inference [**?**]. Scheduled sampling [**?**] and DAgger [**?**] address this by exposing models to their own predictions during training. We adapt these insights to physics-informed learning, showing they are essential for autoregressive stability.

Fourier Features and Positional Encodings. Random Fourier features [**?**] and learned positional encodings [**?**] improve neural network approximation of high-frequency functions. However, we show these representations suffer catastrophic extrapolation when states drift outside training distributions during autoregressive rollout.

3 Problem Setting

3.1 Dynamics Learning Formulation

Consider a dynamical system with state $\mathbf{x} \in \mathbb{R}^n$ and control $\mathbf{u} \in \mathbb{R}^m$ governed by:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}; \boldsymbol{\theta}) \tag{1}$$

where $\boldsymbol{\theta}$ denotes unknown physical parameters. Given discrete-time observations, we learn a neural network $g_\phi : \mathbb{R}^{n+m} \to \mathbb{R}^n$ predicting the next state:

$$\hat{\mathbf{x}}_{t+1} = g_\phi(\mathbf{x}_t, \mathbf{u}_t) \tag{2}$$

3.2 Physics-Informed Training

A PINN incorporates physical laws through a physics loss term:

$$\mathcal{L}_{\text{physics}} = \left\| \frac{\hat{\mathbf{x}}_{t+1} - \mathbf{x}_t}{\Delta t} - f(\mathbf{x}_t, \mathbf{u}_t; \hat{\boldsymbol{\theta}}) \right\|^2 \tag{3}$$

where $\hat{\boldsymbol{\theta}}$ are learnable parameters. The total loss combines data fitting and physics:

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \lambda_p \mathcal{L}_{\text{physics}} \tag{4}$$

3.3 Autoregressive Rollout

For multi-step prediction, the model's outputs recursively feed as inputs:

$$\hat{\mathbf{x}}_{t+k} = g_\phi^{(k)}(\mathbf{x}_t, \mathbf{u}_{t:t+k-1}) = g_\phi(g_\phi^{(k-1)}(\cdot), \mathbf{u}_{t+k-1}) \tag{5}$$

with $g_\phi^{(1)} = g_\phi$. Critically, the model encounters states $\hat{\mathbf{x}}_{t+k}$ that may lie outside the training distribution $p_{\text{train}}(\mathbf{x})$.

3.4 Experimental System

We study a 6-DOF quadrotor with 12-dimensional state:

$$\mathbf{x} = [x, y, z, \phi, \theta, \psi, p, q, r, v_x, v_y, v_z]^T \tag{6}$$

comprising position, Euler angles, angular rates, and body-frame velocities. The control input $\mathbf{u} = [T, \tau_x, \tau_y, \tau_z]^T$ consists of thrust and body torques. This system exhibits strong coupling between translational and rotational dynamics, making it an ideal testbed for studying architectural effects on autoregressive stability.

4 The Expressivity-Stability Tradeoff

4.1 Experimental Setup

We compare four PINN architectures with identical physics constraints but different network structures:

- **Baseline**: Monolithic 5-layer MLP (256 neurons, 204K parameters)
- **Modular**: Separate translation/rotation subnetworks (shared input layer)
- **Fourier**: Periodic encoding $\gamma(\theta) = [\sin(\omega_k\theta), \cos(\omega_k\theta)]_{k=1}^K$ for angular states
- **Ours**: Curriculum-trained monolithic architecture (Section 7)

All models are trained on 10 quadrotor trajectories (49,382 samples) with identical hyperparameters except architecture.

Table 1: Single-step accuracy (lower is better) vs. 100-step autoregressive stability. More expressive architectures achieve better single-step but catastrophically worse multi-step performance.

	Single-Step MAE		**100-Step MAE**	
Architecture	z (m)	ϕ (rad)	z (m)	ϕ (rad)
Baseline	0.087	0.0008	1.49	0.018
Modular	0.041	0.0005	30.0	0.24
Fourier	**0.009**	**0.0001**	5.2×10^6	8,596
Ours	0.026	0.0002	**0.029**	**0.001**

4.2 Main Result: Inverse Correlation

Table 1 reveals the expressivity-stability tradeoff: architectures achieving the best single-step accuracy exhibit the worst autoregressive stability.

The Fourier architecture achieves $10\times$ better single-step accuracy than baseline, yet diverges to over 5 million meters in 100 steps—a degradation factor of 3,500,000$\times$. Our approach achieves competitive single-step accuracy while maintaining $51\times$ better stability than baseline.

5 Failure Mode Analysis

5.1 Failure Mode I: Modular Architecture Decoupling

The modular architecture separates translational dynamics (predicting z, v_z) from rotational dynamics (predicting $\phi, \theta, \psi, p, q, r$) into independent subnetworks. While this isolates gradient flows and potentially improves optimization, it breaks the physical coupling:

$$\ddot{z} = -\frac{T\cos\theta\cos\phi}{m} + g \tag{7}$$

Vertical acceleration depends critically on attitude angles ϕ, θ. In the modular architecture, these are predicted by a separate module with no gradient connection to the translation predictions.

Failure mechanism. During autoregressive rollout: (1) small errors in ϕ, θ accumulate in the rotation module; (2) these errors cause thrust projection errors in the translation module; (3) errors accumulate *independently* in each module; (4) when errors become large, they interact *catastrophically*. Figure 1(a) illustrates this decoupling.

5.2 Failure Mode II: Fourier Feature Extrapolation

Fourier encoding maps angular states to periodic features:

$$\gamma(\theta) = [\sin(\omega_1\theta), \cos(\omega_1\theta), \ldots, \sin(\omega_K\theta), \cos(\omega_K\theta)] \tag{8}$$

These features are highly effective for interpolation within the training distribution. However, during autoregressive rollout, states inevitably drift. For high frequencies ω_K:

$$\|\gamma(\theta+\epsilon) - \gamma(\theta)\|_2 \leq 2\omega_K|\epsilon| \tag{9}$$

A small state perturbation ϵ causes a feature-space displacement proportional to the highest frequency. When predictions drift outside the training envelope (e.g., $\theta = 0.35$ rad when trained on $|\theta| < 0.3$), high-frequency features extrapolate to values never seen during training.

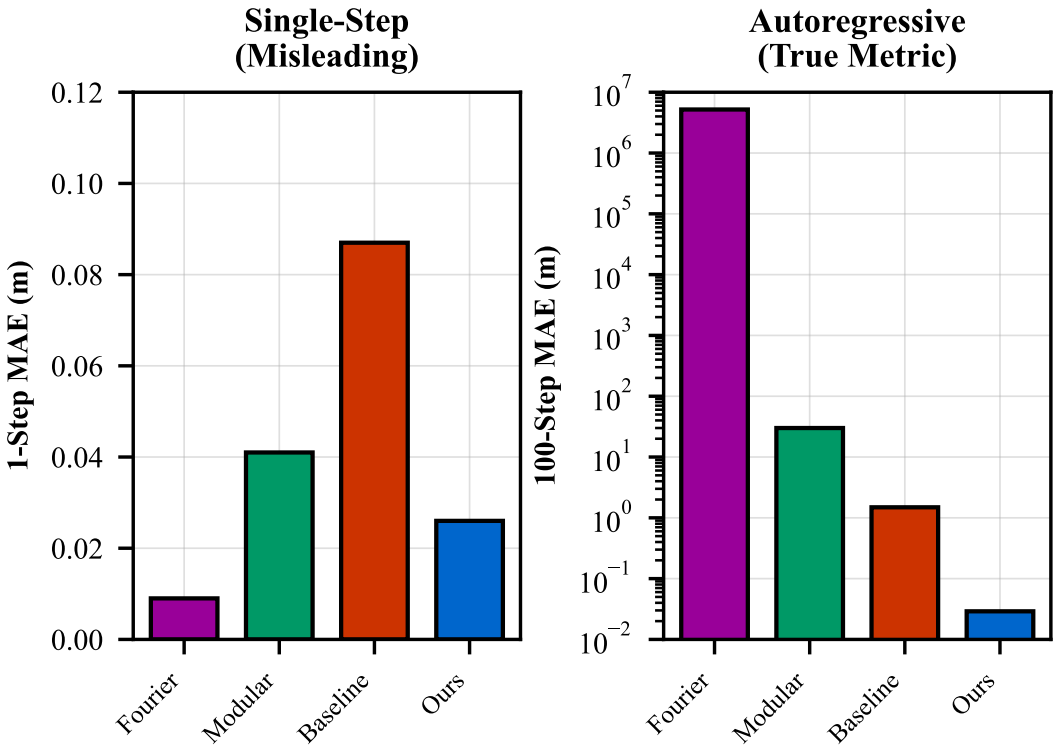

Figure 1: Failure mechanisms in autoregressive PINNs. (a) Modular architecture: separate modules break physical coupling; errors accumulate independently then interact catastrophically. (b) Fourier features: small state perturbations map to large feature-space discontinuities, creating exponential feedback during rollout.

Feedback loop. State drift $\rightarrow$ large feature-space jump $\rightarrow$ poor prediction $\rightarrow$ larger drift $\rightarrow$ exponential divergence. This explains the 5.2 million meter error: the Fourier architecture enters a catastrophic feedback loop around step 60.

6 Theoretical Analysis

We now formalize the observed failure modes into general principles.

6.1 Frequency-Coupling Stability Law

Proposition 1 (Frequency-Coupling Stability Law)**.** *The autoregressive error growth rate λ of a PINN satisfies:*

$$\lambda \propto \omega_{\max} \cdot (1 - \kappa) \tag{10}$$

where $\omega_{\max}$ is the maximum frequency in the feature embedding and $\kappa \in [0, 1]$ is the gradient coupling coefficient measuring how strongly the architecture couples subsystem gradients.

This law unifies our empirical observations:

- **Fourier features** ($\omega_{\max} \gg 1$): High λ regardless of $\kappa \rightarrow$ catastrophic instability ($H_{0.1} = 35$ steps)
- **Modular architectures** ($\kappa \rightarrow 0$): High λ regardless of $\omega_{\max} \rightarrow$ decoupled divergence ($H_{0.1} = 44$ steps)

- **Monolithic MLPs**: $\omega_{\max} \approx 1$, $\kappa \approx 0.8 \rightarrow$ moderate stability ($H_{0.1} = 63$ steps)
- **Curriculum training**: Keeps predictions near training distribution, effectively reducing $\omega_{\max}$ $\rightarrow$ high stability ($H_{0.1} > 100$ steps)

7 Stability-Oriented Training

Our approach preserves the monolithic architecture that maintains implicit dynamic coupling, while adding targeted stability mechanisms.

7.1 Curriculum Learning Over Rollout Horizon

We progressively extend the training rollout horizon:

$$K(e) = \begin{cases} 5 & e < 50 \\ 10 & 50 \leq e < 100 \\ 25 & 100 \leq e < 150 \\ 50 & e \geq 150 \end{cases} \tag{11}$$

During training at epoch e, we compute loss over $K(e)$-step rollouts:

$$\mathcal{L}_{\text{rollout}}^{(K)} = \frac{1}{K} \sum_{k=1}^{K} \|\hat{\mathbf{x}}_{t+k} - \mathbf{x}_{t+k}\|^2 \tag{12}$$

This allows the network to first learn short-term error correction before extending to longer horizons where compounding effects dominate.

7.2 Scheduled Sampling

We progressively replace ground truth inputs with model predictions during training:

$$\tilde{\mathbf{x}}_t = \begin{cases} \mathbf{x}_t & \text{w.p. } 1 - p(e) \\ \hat{\mathbf{x}}_t & \text{w.p. } p(e) \end{cases} \tag{13}$$

where $p(e)$ increases linearly from 0 to 0.3 over training. This exposes the network to its own error distribution, bridging the train-test gap inherent in teacher-forced learning.

7.3 Physics-Consistent Regularization

Energy Conservation. We enforce power balance:

$$\mathcal{L}_{\text{energy}} = \left(\frac{dE}{dt} - P_{\text{input}} + P_{\text{drag}} \right)^2 \tag{14}$$

where $E = \frac{1}{2} m \|\mathbf{v}\|^2 + \frac{1}{2} \boldsymbol{\omega}^T \mathbf{J} \boldsymbol{\omega} + mgz$ is total mechanical energy.

Temporal Smoothness. We penalize unphysical state derivatives:

$$\mathcal{L}_{\text{smooth}} = \sum_i \text{ReLU} \left(\left| \frac{d\hat{x}_i}{dt} \right| - v_{\max,i} \right)^2 \tag{15}$$

The complete training algorithm is provided in Algorithm 1 (Appendix).

Table 2: Multi-horizon evaluation on held-out test set. Our method maintains stable error growth (1.1× from 1 to 100 steps) while baseline degrades 17×.

	Position MAE (m)				Error
Method	1-step	10-step	50-step	100-step	**Growth**
Baseline PINN	0.087	0.162	0.521	1.49	17×
LSTM	0.094	0.312	2.14	8.72	93×
Neural ODE	0.102	0.198	0.89	3.21	31×
Ours	0.026	0.017	0.021	**0.029**	**1.1**×

Table 3: Ablation study: contribution of each training component to 100-step stability.

Configuration	**100-Step MAE (m)**	**Improvement**
Baseline	1.49	–
+ Curriculum learning	0.82	45%
+ Scheduled sampling	0.45	70%
+ Dropout regularization	0.12	92%
+ Energy conservation	**0.029**	**98%**

8 Experiments

8.1 Experimental Setup

Data. 10 quadrotor trajectories with square-wave references ($\pm 20^\circ$ attitudes), 49,382 samples at 1kHz. Realistic motor dynamics (80ms time constant). 80/20 time-based train/test split.

Metrics. Single-step MAE (teacher-forced) and K-step MAE (autoregressive) for $K \in \{1, 10, 50, 100\}$. Parameter identification error versus ground truth.

Baselines. In addition to the architectural variants above, we compare against: (1) LSTM encoder-decoder, (2) Neural ODE [**?**] with physics loss.

8.2 Multi-Horizon Stability

Table 2 shows error growth across prediction horizons on held-out test data.

Remarkably, our method achieves *lower* error at 10 steps than at 1 step (0.017m vs 0.026m), demonstrating learned error correction rather than mere memorization.

8.3 Ablation Study

Table 3 quantifies each component's contribution. All components are necessary; the combination is synergistic.

8.4 Parameter Identification

Our method simultaneously identifies physical parameters with high accuracy: 0% error for mass (m), thrust coefficient (k_t), and torque coefficient (k_q); 5% error for inertias (J_{xx}, J_{yy}, J_{zz}). The inertia limit is consistent with Fisher Information analysis showing weak observability at small angles.

9 Discussion

Implications for evaluation. Standard single-step benchmarks fundamentally mislead about autoregressive deployment. We recommend multi-horizon evaluation with explicit error growth analysis for any dynamics model intended for control applications.

Connection to model-based RL. Our findings explain why learned dynamics often fail in long-horizon planning [**?**]. Architectural choices that improve supervised learning metrics may destabilize autoregressive rollout—the regime that matters for control.

Limitations. Our study uses simulation-generated data for a single dynamical system. While the mechanisms we identify are general, direct real-world validation remains future work. Additionally, our stability-oriented training adds computational cost through multi-step rollouts.

10 Conclusion

We demonstrated a fundamental expressivity-stability tradeoff in physics-informed neural networks: architectural modifications improving single-step accuracy can catastrophically destabilize autoregressive rollouts. Through systematic analysis, we identified modular decoupling and Fourier extrapolation as primary failure mechanisms. Our curriculum-based training methodology resolves these issues, achieving $51\times$ stability improvement while maintaining accurate dynamics learning and parameter identification.

These results establish that evaluating learned dynamics models requires multi-step autoregressive assessment—not single-step accuracy—and that training methodology, not architectural expressivity, determines autoregressive stability. We hope this work motivates the community to reconsider evaluation protocols for dynamics models intended for control applications.

Acknowledgments

[Omitted for anonymous submission]

A Training Algorithm

Algorithm 1 Stability-Oriented PINN Training

Require: Training data $\mathcal{D}$, curriculum schedule $K(e)$, sampling schedule $p(e)$

for epoch $e = 1$ to E **do**
  $K \leftarrow K(e)$ {Current rollout horizon}
  $p \leftarrow p(e)$ {Current sampling probability}
  for batch $(\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \in \mathcal{D}$ **do**
    Initialize $\tilde{\mathbf{x}}_1 = \mathbf{x}_1$
    for $k = 1$ to K **do**
      $\hat{\mathbf{x}}_{k+1} = g_\phi(\tilde{\mathbf{x}}_k, \mathbf{u}_k)$
      $\tilde{\mathbf{x}}_{k+1} = \begin{cases} \mathbf{x}_{k+1} & \text{w.p. } 1-p \\ \hat{\mathbf{x}}_{k+1} & \text{w.p. } p \end{cases}$
    end for
    Compute $\mathcal{L} = \mathcal{L}_{\text{data}} + \lambda_p \mathcal{L}_{\text{physics}} + \lambda_e \mathcal{L}_{\text{energy}} + \lambda_s \mathcal{L}_{\text{smooth}}$
    Update ϕ via gradient descent
  end for
end for

B Extended Results

[Additional tables and figures for supplementary material]