BEYOND ONE-STEP ACCURACY: TRAJECTORY, INVARIANT, AND DISSIPATIVE FIDELITY IN MODELLING OF PDE DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Popular approaches for modelling PDE dynamics in a data-driven way include direct one-step predictors like neural operators. Recently, continuous-time generator-based methods like Trajectory Flow models have also shown strong results in this setting. These approaches typically have good one-step accuracy. However, long-horizon trajectory fidelity under autoregressive composition and physical consistency are also important metrics for physical systems. Low one-step prediction error alone does not determine long-horizon trajectory fidelity or physical consistency. We analyse why continuous-time generators achieve stronger long-horizon fidelity than the direct one-step predictors. We also introduce GENERIC structure into the Trajectory Flow model to better preserve known physical invariants while retaining a continuous-time generator formulation. Our model reduces invariant violations by orders of magnitude. This substantial gain in invariant preservation comes at the cost of some trajectory accuracy, although the structured model retains strong long-horizon fidelity. Moreover, invariant preservation alone does not ensure quantitatively correct dissipation. The structured model can evolve dissipative functionals in the correct direction while misestimating their rate. We therefore extend GENERIC Flow with explicit functional-change supervision, which partially recovers dissipative-rate fidelity.

1 INTRODUCTION

Data-driven models of surrogate PDE solvers have several notions of correctness. A model may predict the next state accurately given the true initial state yet accumulate large errors when deployed autoregressively (McCabe et al., 2023; Lippe et al., 2023; Koehler et al., 2024). Another model which is capable of reproducing accurate long-horizon trajectories may violate quantities that are supposed to be preserved. Conversely, a model which may be physically consistent with respect to conservation laws may not produce the most accurate long-horizon trajectories. Thus improvements in *one-step accuracy*, *trajectory fidelity*, and *physical fidelity* need not induce the same ordering of dynamical models.

The important characteristic of the model also depends on the application. In short-horizon forecasting or control, reaching the correct state at the required time can be more important than small violations of conserved quantities. However, in long-running physical simulations, conservation of invariant quantities may be more important as systematic drifts will result in physically inconsistent trajectories. For dissipative systems, there is a further distinction as we need to maintain both the correct direction and rate of dissipation (Grmela & Öttinger, 1997; Yu et al., 2021; Zhang et al., 2022). We study these properties across two different families of data-driven surrogate models. A direct one-step predictor modelling PDE dynamics parameterises the transition as $u_{k+1} = F_\theta(u_k, t_k)$, whereas a continuous-time generator learns a vector field, $\dot{u} = f_\theta(u, t)$, and obtains the transition through numerical integration (Chen et al., 2018; Hou et al., 2026). Both are composed autoregressively, but they impose different structure on the deployed transition. We observe that direct endpoint fitting permits many one-step maps to match the same observations, whereas a generator-induced transition and its local sensitivity arising from the same continuous vector field impose a stricter structure.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We study the aforementioned characteristics across five PDE families, including a two-dimensional Burgers setting. Our experiments show that the continuous-time generator, trained using Trajectory Flow (Zhang et al., 2024; Hou et al., 2026), consistently achieves stronger long-horizon trajectory fidelity than the direct one-step predictors (CNN and FNO (Li et al., 2021; Kovachki et al., 2023)) when used autoregressively. We further strengthened the training methodology of the direct one-step predictors with multistep rollout-aware training and explicit input Jacobian regularisation. Though this shows some improvement, it does not consistently yield performance as strong as the Trajectory Flow models. To investigate further, we also took direct measurements of the two key factors that influence error propagation during repeated model application: the local transition defect and the model's input sensitivity. The generator-induced Trajectory Flow models exhibit substantially lower local error and weaker upper-tail input sensitivity. However, achieving high trajectory fidelity through autoregressive updates does not imply that the model will also preserve the physical fidelity. If physical laws are not explicitly enforced, small systematic prediction errors can accumulate as drift in invariant quantities (McGreivy & Hakim, 2023).

We therefore introduce a GENERIC-inspired (Grmela & Öttinger, 1997) Trajectory Flow model that can embed prescribed conservative and dissipative structure while retaining the advantages of a continuous-time generator model. The structured GENERIC Flow suppresses the invariant-violating updates of an unrestricted Trajectory Flow model by orders of magnitude even though the unrestricted model still achieves a lower trajectory error. These results show that a tradeoff may exist between invariant preservation and accurate state-space representation. Invariant preservation benefits from a particularly rigid structural form, but makes optimisation for long-horizon state-space accuracy difficult.

We also note that physical fidelity encompasses both invariant preservation and quantitatively correct dissipation. We observe that the structural constraints can restrict evolution to admissible directions and enforce the sign of dissipation, but they do not uniquely determine its magnitude. Unrestricted Trajectory Flow models often track dissipative functionals more accurately than their GENERIC-inspired counterparts, even when both models evolve them in the correct direction. We therefore extend the GENERIC Flow with supervision of both local and finite-step functional change. This recovers part of the lost dissipative-rate fidelity while retaining strong invariant preservation.

Our contributions are:

1. We study how directly predicting the next state versus learning a continuous-time generator affects long-horizon autoregressive behaviour across five PDE families and both one- and two-dimensional settings. We additionally compare against direct predictors strengthened with multistep rollout-aware training and input-Jacobian regularisation.
2. We connect long-horizon trajectory behaviour to local transition defect and deployed-step sensitivity, and show empirically that generator-induced models exhibit favorable behavior on both. We further show that improved trajectory fidelity does not necessarily imply physical fidelity.
3. We introduce a GENERIC-inspired structured continuous-time generator that embeds prescribed conservative and dissipative structure into Trajectory Flow. The resulting GENERIC Flow substantially improves invariant preservation, while explicit functional-change supervision partially restores dissipative-rate fidelity.

2 Related Work

When used as autoregressive PDE surrogates, neural operators can learn finite-step maps of the form $u_{t+1} = F_\theta(u_t)$ (Lu et al., 2021; Li et al., 2021; Kovachki et al., 2023). Repeated composition of these maps can accumulate prediction errors and eventually destabilise long-horizon rollouts (McCabe et al., 2023). There are various approaches in the literature to mitigate this problem. Multistep rollout-aware training has been proposed by Brandstetter et al. (2022); Koehler et al. (2024), while Finlay et al. (2020); Nie & Suzuki (2026) used input Jacobian regularisation to control amplification of input perturbations. Some of the works modify the structure of the neural operators themselves. McCabe et al. (2023) study stability-oriented neural operator designs, DRIFT-Net (Li & Salim, 2026) combines spectral and spatial branches, and SGNO (Li et al., 2026) uses a gain-controlled spectral generator to improve autoregressive stability. Our study is complementary. Rather than in-

troducing another stabilised architecture, we investigate how directly learning a one-step transition versus learning a continuous-time generator affects long-horizon behaviour under matched autoregressive deployment.

Generative approaches, including diffusion-based PDE models (Huang et al. (2024); Lu & Xu (2024)), learn distributions over solution trajectories, often mapping initial conditions or system inputs directly to full trajectories. Neural ODEs parameterise a vector field $\dot{u} = f_\theta(u, t)$ and obtain finite-time evolution through numerical integration (Chen et al. (2018)). While expressive, training Neural ODEs can be computationally expensive because optimisation typically requires repeated numerical integration and differentiation through the solver (Baker et al. (2022); Wang et al. (2026); Kidger et al. (2020); Finlay et al. (2020)). Trajectory Flow model instead learns the generator directly from observed transitions (Zhang et al. (2024)), avoiding solver-in-the-loop training while retaining a continuous-time generator at deployment. Recent trajectory-flow-matching methods further develop this direction (Hou et al. (2026); Zhang et al. (2026)). We study this generator-based representation under long autoregressive deployment, comparing it against strengthened one-step baselines with matched spatial backbones and measuring both local transition defect and deployed-step sensitivity.

PINNs and physics-informed neural operators incorporate the governing PDE through equation residuals or related physics losses (Raissi et al., 2019; Li et al., 2024). In contrast, we assume observed trajectories and selected physical functionals, without requiring access to the full governing equation during learning. A complementary line of work embeds geometric or thermodynamic structure directly into learned dynamics. Hamiltonian and symplectic models incorporate conservative structure (Greydanus et al., 2019; Zhong et al., 2020), while OnsagerNet (Yu et al., 2021) learns thermodynamically structured dynamics using a generalised Onsager principle, including reduced-order models. GFINNs (Zhang et al., 2022) enforce GENERIC degeneracy conditions through neural network construction. MCFM (Baheri & Lindemann, 2025) combines metriplectic structure with conditional flow matching with a Strang splitting integrator. We propose to incorporate GENERIC-inspired structure into Trajectory Flow and evaluate its effects on trajectory and physical fidelity across multiple PDE families and a much larger dataset, including a two-dimensional setting. We use this framework to investigate how hard structural constraints affect the balance between trajectory accuracy, invariant preservation, and dissipative-rate fidelity.

3 BACKGROUND

In our setting, we consider a data-driven dynamical system with the actual governing equations unknown. We observe trajectories at discrete times t_k represented as $u_k = u(t_k) \in \mathbb{R}^{N_x}$. The unknown dynamics induce a finite-time flow map $u_{k+1} = F^*(u_k, t_k) = u_k + \int_{t_k}^{t_k+\Delta t} f^\star(u(t), t)\, dt$ where $f^\star$ denotes the true continuous-time vector field and $F^*(u_k, t_k)$ denotes the corresponding finite-step transition. The main goal is to learn a model that takes (u_k, t_k) as input and learns the transition to u_{k+1}. We deploy this model recursively over horizons substantially longer than those used during training and study three increasingly structured parameterisations: direct one-step maps (FNO and CNN), unrestricted continuous-time generators (Trajectory Flow models), and structure-preserving continuous-time generators (GENERIC Flow).

3.1 DIRECT ONE-STEP PREDICTORS

A direct one-step predictor parameterises the finite transition itself as $\hat{u}_{k+1} = F_\theta(u_k, t_k)$ where F_θ is a neural network that takes (u_k, t_k) as input and directly outputs the predicted $\hat{u}_{k+1}$. The base model is trained using one-step supervision, $\mathcal{L}_{1\text{step}} = \mathbb{E}_k[\|F_\theta(u_k, t_k) - u_{k+1}\|_2^2]$. At inference time, the learned map is recursively composed: $\hat{u}_{k+m} = F_\theta^{(m)}(u_k)$. To check whether failures under repeated composition can be addressed purely through stronger training objectives, we additionally consider multistep rollout-aware and Jacobian-regularised variants of the one-step supervision loss. We use the M-step objective $\mathcal{L}_{\text{MS}} = \mathbb{E}[\sum_{m=1}^{M} \|\hat{u}_{k+m} - u_{k+m}\|_2^2]$ (Brandstetter et al., 2022). Jacobian regularisation penalises local sensitivity through $\mathcal{L}_{\text{J}} = \mathcal{L}_{1\text{step}} + \lambda_J \|\nabla_u F_\theta(u_k, t_k)\|_F^2$. (Finlay et al., 2020) These variants modify the training objective while retaining the same finite-step representation.

3.2 Continuous-Time Generator Models

A continuous-time generator model parameterises a local vector field, $du/dt = f_\theta(u,t)$ (Chen et al., 2018). The corresponding transition is obtained by numerical integration of the learned vector field as $\hat{u}_{k+1} = F_\theta(u_k, t_k) = u_k + \int_{t_k}^{t_k+\Delta t} f_\theta(u(t),t)\, dt$. We use Trajectory Flow matching (Hou et al., 2026) as our continuous-time generator formulation and emphasize that its training differs from that of Neural ODEs (see 2). For the Trajectory Flow models, we construct the supervised target for the local velocity field from consecutive observations. For a transition (u_k, u_{k+1}), define $v_k = (u_{k+1} - u_k)/\Delta t$. Following trajectory-flow training, we sample $\tau \sim \mathcal{U}(0,1)$ and construct $u_\tau = (1-\tau)u_k + \tau u_{k+1}$. The generator is trained by $\mathcal{L}_{\text{Flow}} = \mathbb{E}_{k,\tau}\left[\left\| f_\theta\left(u_\tau, t_k + \tau\Delta t\right) - v_k\right\|_2^2\right]$. We use the same CNN and FNO backbone families for both direct and continuous parameterisations. Solver and architecture details are given in Appendix A.

4 Temporal Representation and Physical Structure

4.1 Finite-Step Composition versus Generator-Induced Evolution

Both direct one-step predictors and Trajectory Flow models are deployed autoregressively, so their long-horizon behaviour is governed by repeated composition of a learned finite-step transition. Let's consider a learned transition F_θ under an autoregressive setting, with

$$u_{k+1} = F^*(u_k, t_k), \qquad \hat{u}_{k+1} = F_\theta(\hat{u}_k, t_k), \qquad e_{k+1} = \hat{u}_{k+1} - u_{k+1}. \tag{1}$$

Defining the local defect $\delta_k = F_\theta(u_k,t_k) - F^*(u_k,t_k)$ and the deployed-step Jacobian $J_k = D_u F_\theta(u_k,t_k)$, a first-order expansion gives (Stuart, 1994)

$$e_{k+1} = J_k e_k + \delta_k + \mathcal{O}(\|e_k\|^2). \tag{2}$$

Thus, long-horizon accuracy depends jointly on the local error injected at each step and on how the learned transition propagates existing errors.

The distinction between direct one-step prediction and Trajectory Flow model lies in how F_θ is parameterised. A one-step predictor is trained on $(u_{k+1}, F_\theta(u_k,t_k))$ directly without taking into consideration the underlying vector field f_θ. Training constrains the values of this map at observed transitions, but does not otherwise determine how the map varies around those states. Trajectory Flow model instead learns a local generator: $\dot{u} = f_\theta(u,t); F_\theta(u,t) = u + \int_t^{t+\Delta t} f_\theta(u(s),s)\, ds$, so the deployed finite-step transition is induced by numerical integration. For small Δt,

$$F_\theta(u,t) = u + \Delta t\, f_\theta(u,t) + \mathcal{O}(\Delta t^2), \qquad D_u F_\theta(u,t) = I + \Delta t\, D_u f_\theta(u,t) + \mathcal{O}(\Delta t^2). \tag{3}$$

The corresponding true evolution satisfies the same expansion as in equation 3, with true vector field $f^\star$ instead of a learned f_θ. Hence, approximation of the generator and its local variation induces approximation of both the finite-step transition and its sensitivity. In contrast, direct endpoint fitting by direct one-step predictors leaves these properties less tightly coupled. We therefore evaluate the two quantities appearing in equation 2 directly in Table 1. One-step MSE measures the local transition defect, while $\hat{\sigma}_{\max}$ estimates the largest singular value of the state-input Jacobian of one deployed step. For Trajectory Flow model, differentiation passes through the complete numerical integrator, so the measured quantity is $D_u \Phi^\theta_{\Delta t}$.

Trajectory Flow model has substantially smaller local defect for both backbones and a tighter finite-step amplification profile. For CNN, both methods remain close to neutral amplification, with Flow showing a smaller upper tail. The difference is stronger for FNO: $\hat{\sigma}_{\max} > 1$ at 94.0% of sampled states for direct one-step predictors versus 22.9% for Trajectory Flow model, while the P95 estimate decreases from 1.39 to 1.06.

The results are consistent with the representation-level argument above. Direct endpoint fitting leaves more freedom in the learned finite-step map, whereas a continuous-time generator restricts transitions to those induced by integrating a vector field. Empirically, this is accompanied by both lower local defect and weaker upper-tail amplification, matching the observed long-horizon rollout advantage.

Table 1: **Local defect and deployed-step amplification.** $\hat{\sigma}_{\max}$ denotes a finite power-iteration estimate of the largest singular value of the state-input Jacobian of one deployed step. Flow uses midpoint integration with eight substeps ($\Delta t = 0.001$, 16 neural-function evaluations). The estimates use matched test states across methods. The Jacobian statistics pool 336 matched test states across 21 equation settings, while one-step MSE is aggregated with equal weight across PDE families.

Backbone	Model	One-Step MSE ↓	Median $\hat{\sigma}_{\max}$	P95 $\hat{\sigma}_{\max}$	Frac. $\hat{\sigma}_{\max} > 1$
CNN	Direct one-step	8.154×10^{-6}	0.9978	1.0121	22.3%
CNN	Trajectory Flow model	8.049×10^{-10}	0.9970	1.0041	14.0%
FNO	Direct one-step	2.297×10^{-5}	1.0571	1.3930	94.0%
FNO	Trajectory Flow model	1.154×10^{-8}	0.9993	1.0567	22.9%

4.2 Physical Admissibility of Learned Dynamics

Trajectory Flow models learn a continuous vector field from observed transitions. The linear training path, $u_\tau = (1-\tau)u_k + \tau u_{k+1}$, is economical in state space, but this construction imposes no physical constraints on the learned vector field. An unrestricted generator may therefore fit trajectories accurately while still violating invariants that the reference dynamics preserve. Hard structure instead restricts the model to a physically admissible hypothesis class.

A plausible geometric interpretation of why invariant preservation and trajectory accuracy can separate is provided in Appendix F.

Figure 1 illustrates this separation directly. Although the reference trajectories preserve the corresponding invariants to numerical precision, the unrestricted Trajectory Flow model produces a substantial fraction of invariant-violating rollout updates, ranging from roughly 25% on Advection and Burgers to nearly all updates on KdV. Injecting structural constraints into the Trajectory Flow model suppresses these violations to numerical precision by restricting the learned vector field to the admissible class.

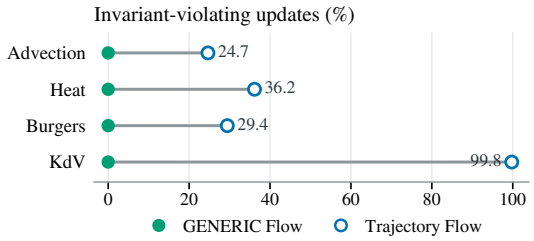


Figure 1: **Hard structure suppresses invariant violations.** Fraction of rollout updates exceeding the conservation deadband. Reference trajectories and GENERIC Flow remain near numerical precision, while unrestricted Trajectory Flow model exhibits substantial violations.

5 GENERIC Flow: Structured Continuous Evolution

A Trajectory Flow model provides a continuous-time generator, but accurate state evolution alone does not guarantee that the learned dynamics respect known physical structure. We therefore restrict the generator using a GENERIC-inspired decomposition. GENERIC (Grmela & Öttinger, 1997) provides a thermodynamic decomposition of the vector field into reversible and irreversible components. The reversible component is generated through an energy-like functional $E(u)$, while the irreversible component is driven by an entropy-like functional $S(u)$. The associated operators are constrained so that the reversible dynamics do not change S, while the irreversible dynamics do not change E. For notational simplicity, we write $u = u(t)$ when the time dependence is clear from context. Let $E(u)$ denote a declared invariant and $S(u)$ a dissipative or entropy-like functional when available. The PDE-specific E and S are defined in Appendix D. The structured core is

$$f_{\text{hard}}(u) = L_\theta(u)\nabla E(u) + M_\theta(u)\nabla S(u), \tag{4}$$

$$L_\theta^\top = -L_\theta, \qquad M_\theta \succeq 0, \qquad L_\theta \nabla S(u) = 0, \qquad M_\theta \nabla E(u) = 0. \tag{5}$$

Here, $L_\theta(u), M_\theta(u) \in \mathbb{R}^{N_x \times N_x}$ are learned state-dependent operators. L_θ represents the reversible (conservative) component of the dynamics and is constrained to be skew-symmetric, while M_θ represents the irreversible (dissipative) component and is constrained to be positive semidefinite. They

additionally satisfy the GENERIC degeneracy conditions. These conditions give

$$\frac{dE}{dt} = 0, \qquad \frac{dS}{dt} = \nabla S(u)^\top M_\theta \nabla S(u) \geq 0 \tag{6}$$

for the hard GENERIC core (Grmela & Öttinger, 1997; Zhang et al., 2022).

We enforce these algebraic relations by construction. The network predicts low-rank factors U, V, B, which are projected into the required state-dependent null spaces, $\widetilde{U} = P_{\perp \nabla S} U, \quad \widetilde{V} = P_{\perp \nabla S} V, \quad \widetilde{B} = P_{\perp \nabla E} B$, where $P_{\perp q} = I - qq^\top/(q^\top q + \varepsilon)$. The operators are then constructed as the two expressions (Horn & Johnson, 2012) $L_\theta = \frac{\widetilde{U}\widetilde{V}^\top - \widetilde{V}\widetilde{U}^\top}{\sqrt{r}}, M_\theta = \frac{\widetilde{B}\widetilde{B}^\top}{\sqrt{r}}$, so that they are skew-symmetric and positive semidefinite by construction.

Flexibility Components. The GENERIC decomposition provides useful physical constraints, but these constraints can often be too restrictive to completely specify the dynamics. Restricting the entire learned vector field to equation 4 can be unnecessarily rigid. The learned generator must represent all observations using only the directions available through the declared GENERIC operators. We therefore retain the hard structured core while adding two controlled sources of flexibility: $f_\theta = f_{\text{hard}} + f_{\text{tangent}} + f_{\text{slack}}$. The tangent branch learns additional motion that is not represented by the hard core but is still constrained to lie in the tangent space of the declared invariants. The slack branch provides a small compatibility residual for dynamics that cannot be represented by the structured components and is strongly regularised during training.

Training the structured generator. For each observed transition (u_k, u_{k+1}), we treat one stored transition as a unit interval in model time and define the observed secant velocity $v_k^{\text{src}} = u_{k+1} - u_k$. We supervise both the local generator and the finite transition induced by numerical integration:

$$\mathcal{L}_{\text{velocity}} = \frac{\text{MSE}(f_\theta(u_k, t_k), v_k^{\text{src}})}{s_v^2}, \qquad \mathcal{L}_{\text{endpoint}} = \frac{\text{MSE}(F_\theta(u_k, t_k), u_{k+1})}{s_v^2}, \tag{7}$$

where s_v^2 is the training-set scale of the observed velocity increments and F_θ denotes the finite transition obtained by integrating the complete structured generator. The complete pointwise objective is $\mathcal{L}_{\text{point}} = \mathcal{L}_{\text{velocity}} + \alpha_{\text{str}}\mathcal{L}_{\text{structure}} + \alpha_{\text{slack}}\mathcal{L}_{\text{slack}} + \alpha_{\text{end}}\mathcal{L}_{\text{endpoint}}$. $\mathcal{L}_{\text{structure}}$ softly penalises violations of the GENERIC degeneracy relations while $\mathcal{L}_{\text{slack}}$ regularises reliance on the least-constrained compatibility branch. Thus, $\mathcal{L}_{\text{velocity}}$ constrains the local dynamics, whereas $\mathcal{L}_{\text{endpoint}}$ ensures that the numerically integrated finite-step evolution remains consistent with the observed transition. The extra endpoint loss term, in addition to the velocity loss term, is important for preserving trajectory accuracy since optimisation under structural constraints is difficult, as shown in Appendix H.2.

Rollout-aware training. Pointwise supervision primarily exposes the generator to reference states, whereas autoregressive deployment repeatedly evaluates it on states produced by the model itself. We therefore additionally train through the numerical solver. Starting from a reference state u_0, we recursively apply the deployed transition to obtain $\hat{u}_h = F_\theta^{(h)}(u_0); h = 1, \ldots, H$. The rollout objective is $\mathcal{L}_{\text{roll}} = \mathcal{L}_{\text{state}} + \beta_{\text{grad}}\mathcal{L}_{\text{gradient}} + \beta_{\text{func}}\mathcal{L}_{\text{functional}} + \beta_{\text{can}}\mathcal{L}_{\text{canonical}} + \beta_{\text{JVP}}\mathcal{L}_{\text{JVP}}$.

$$\mathcal{L}_{\text{state}} = \frac{\text{mean}_{h,x}(\hat{u}_h - u_h)^2}{\max\{\text{mean}_{h,x}\, u_h^2, \epsilon\}}, \qquad \mathcal{L}_{\text{gradient}} = \frac{\text{mean}_{h,x}(D_x\hat{u}_h - D_x u_h)^2}{\max\{\text{mean}_{h,x}(D_x u_h)^2, \epsilon\}}. \tag{8}$$

For a declared physical functional, we additionally compare its change from the initial rollout state, using a normalised squared error. $\mathcal{L}_{\text{functional}}$ aggregates these errors over the prescribed E, S, and auxiliary invariants, while $\mathcal{L}_{\text{canonical}}$ applies the same principle to an additional equation-specific functional. $\mathcal{L}_{\text{JVP}}$ discourages excessive local amplification. *These auxiliary objectives provide stabilising supervision during long horizon rollout-aware optimisation.* We follow a progressive rollout curriculum, with pointwise and rollout objectives optimised separately rather than jointly on the same minibatch.

To confirm that the rollout-aware training is not the main source of invariant preservation, we perform two controlled ablations. First, an unrestricted Trajectory Flow model trained with the same progressive rollout-aware curriculum as GENERIC Flow continues to exhibit frequent invariant-violating updates, showing that deployment-aware training alone does not explain the improvement (Appendix G). Secondly, removing only the state-dependent null-space projections from the structured model, while retaining the skew-symmetric/positive-semidefinite constructions and the remaining training pipeline, causes invariant drift to increase by roughly five orders of magnitude on Burgers and KdV (Appendix H.1).

Table 2: Each entry reports nRMSE @100 / t_{acc}@30%, with nRMSE in 10^{-2} units (lower / higher is better). Multistep denotes rollout-aware training, while Jacobian denotes input-Jacobian regularisation; Trajectory Flow model is the continuous-time generator reference.

Backbone	Training strategy	Advection	Heat	Burgers	Allen-Cahn	KdV	2D Burgers
CNN	One-step	14.75/362.5	7.399/321.5	35.59/278.8	38.74/149.8	27.51/128.3	43.20/45.8
	Multistep	1.686/1592.7	2.056/1355.3	1.300/1734.5	2.018/1357.0	3.930/433.3	19.07/116.0
	One-step + Jacobian	37.97/259.2	7.500/289.3	10.80/229.3	25.33/172.5	21.97/180.3	64.60/35.3
	Multistep + Jacobian	2.293/1359.0	2.136/1716.5	1.957/1397.3	1.869/1588.0	3.292/1405.3	17.68/178.8
	Trajectory Flow	**0.07639/2000**	**0.05072/2000**	**0.04692/2000**	**0.05913/2000**	**0.2455/1442.0**	**4.949/399.5**
FNO	One-step	2.999×10^{7}/21.3	198.2/16.5	25.85/72.3	39.86/65.0	158.7/49.7	52.66/39.2
	Multistep	88.52/6.3	67.63/13.8	67.76/15.8	73.76/9.3	73.92/15.3	69.02/5.8
	One-step + Jacobian	43.23/38.8	41.87/39.5	27.01/66.8	31.86/55.5	4.201×10^{21}/23.3	42.38/40.0
	Multistep + Jacobian	93.83/5.3	68.93/12.5	73.17/13.5	76.45/8.8	78.21/13.3	71.59/5.5
	Trajectory Flow	**0.5211/1920.5**	**0.3522/2000**	**0.7203/2000**	**0.3253/2000**	**1.707/758.7**	**6.649/400.3**

Rate Supervised Extension. GENERIC constrains admissible evolution but does not determine the magnitude of functional change. We therefore augment the pointwise objective with two terms comparing the observed increment $\Delta S_k = S(u_{k+1}) - S(u_k)$ against (i) the local generator estimate $\nabla S(u_k)^\top f_\theta(u_k, t_k)$ and (ii) the functional change produced by the integrated step. Both use normalized Huber losses: $\mathcal{L}^{\mathrm{ext}}_{\mathrm{point}} = \mathcal{L}_{\mathrm{point}} + \beta_{\mathrm{rate}}\mathcal{L}_{\mathrm{rate}} + \beta_{\mathrm{step}}\mathcal{L}_{\mathrm{step}}$. For systems with a mass invariant, the slack component is additionally projected onto its mass-preserving tangent space.

Exact details on definitions, parameterisations, loss functions and the architecture are provided in Appendix C.

6 Experiments

6.1 Experimental Setup and Evaluation

We evaluate on five one-dimensional PDE families, Advection, Heat, Burgers, Allen-Cahn, and KdV, using autoregressive rollouts of up to $T = 2000$ steps. We additionally evaluate scalar conservative Burgers in two spatial dimensions on a 64×64 grid over $T = 499$ forecast steps. For 1D PDEs, we use 40 trajectories for training, 10 for validation, and 20 for testing. For 2D PDEs, we use 16,4 and 8 trajectories for training, validation and testing, respectively. We report cumulative nRMSE@H and the accuracy horizon $t_{\mathrm{acc}}(\tau) = \min\{t : e_{\mathrm{state}}(t) > \tau\}$ for trajectory fidelity. Physical fidelity is evaluated separately through maximum invariant self-drift and, for evolving quantities, functional trajectory and increment errors (Takamoto et al., 2022; McGreivy & Hakim, 2023; Hansen et al., 2023). Exact definitions, normalisation, equation-specific functionals, and aggregation rules are given in Appendices E and D. Architecture, training, data-generation, and split details are given in Appendices A.1 and A.2.

6.2 Continuous-Time Generators Improve Long-Horizon Trajectory Fidelity

First we show empirically that the stability and accuracy of the long-horizon trajectories under recursive composition are affected by how the state transition is parameterised. We compare direct one-step predictors with the Trajectory Flow model, which learns a continuous-time vector field (Table 2 and Figure 2). Trajectory flow is backbone-agnostic. So, to ensure a fair comparison, we evaluate the results separately. We further investigate whether the long-horizon trajectory failures of the direct one-step predictors can be mitigated via stronger training objectives like multistep rollout training or input-Jacobian penalisation, as discussed in Section 4.1. Table 2 reports cumulative nRMSE@100 and t_{acc}@30%. Stronger training substantially improves some direct predictors, particularly with the CNN backbone, but the gains are not uniform across PDEs to consistently match Trajectory Flow. Thus, rollout-aware and sensitivity-regularised training can mitigate failures of direct finite-step prediction, but does not consistently recover the long-horizon behaviour obtained from the continuous-time generator-based representation.

6.3 Relation Between Trajectory Fidelity and Physical Fidelity

The previous sections establish the Trajectory Flow model as the strongest and most consistent model for long-horizon state prediction. We now try to evaluate whether accurate state evolution

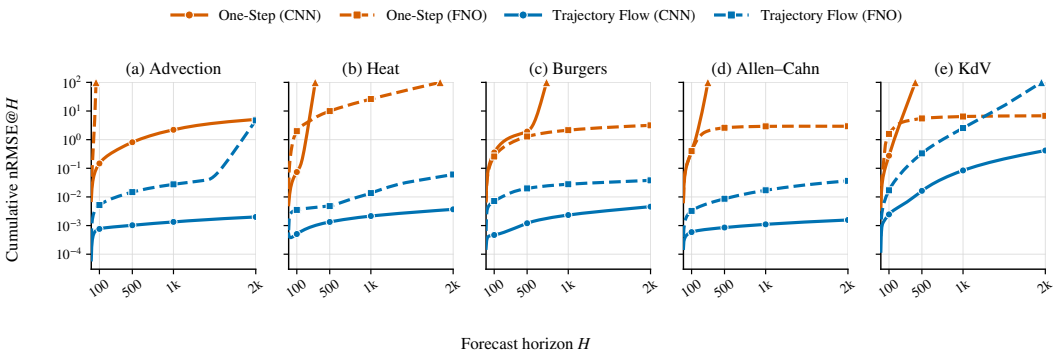


Figure 2: Empirical long-horizon rollout comparison between direct one-step prediction and Trajectory Flow model parameterizations with CNN and FNO backbones. Each panel reports the equal-equation mean of cumulative $\mathrm{nRMSE}@H$ for one PDE; curves use every forecast horizon and markers identify $H \in \{100, 500, 1000, 2000\}$. Curves are truncated at $\mathrm{nRMSE} = 10^2$ for readability, with upward triangles marking the first exceedance.

implies physically faithful evolution as discussed in 4. We compare the unrestricted *Trajectory Flow model (CNN)* against *GENERIC Flow* and the strongest one-step predictor, *Multistep + Jacobian (CNN)*, which combines multistep training and input-Jacobian regularisation. Table 3 compares state accuracy with the two complementary notions of physical fidelity. For invariants, we report the maximum self-drift $D_Q^{\max}$. For non-invariant quantities, we report the mean target-relative functional error followed by the increment nRMSE.

Table 3: Trajectory and physical fidelity of Multistep + Jacobian (CNN), Trajectory Flow model (CNN), and GENERIC Flow. nRMSE is reported in units of 10^{-3}.

PDE	Model	nRMSE@100 ↓ ($\times 10^{-3}$)	t_{acc}@10% ↑	Maximum invariant drift ↓	Functional error ↓ (trajectory / increment)
Advection	Multistep + Jacobian	22.93	375.7	M : 3.953	$2.695 \times 10^{-1}/1.084$
	Trajectory Flow model (CNN)	$\mathbf{7.785 \times 10^{-1}}$	**2000**	M : 4.949×10^{-2}	$\mathbf{1.574 \times 10^{-3}}/\mathbf{6.793 \times 10^{-3}}$
	GENERIC Flow	2.374	1959.8	$\mathbf{M : 2.946 \times 10^{-6}}$	$1.786 \times 10^{-2}/5.958 \times 10^{-2}$
Heat	Multistep + Jacobian	21.36	518.0	M : 3.572	$1.498 \times 10^{-1}/1.932 \times 10^{-1}$
	Trajectory Flow (CNN)	$\mathbf{5.804 \times 10^{-1}}$	**2000**	M : 6.266×10^{-2}	$\mathbf{3.060 \times 10^{-3}}/\mathbf{2.985 \times 10^{-3}}$
	GENERIC Flow	7.062	1802.1	$\mathbf{M : 3.067 \times 10^{-6}}$	$5.915 \times 10^{-2}/4.585 \times 10^{-2}$
Burgers	Multistep + Jacobian	19.57	469.8	M : 4.387	$1.869 \times 10^{-1}/2.368 \times 10^{-1}$
	Trajectory Flow model (CNN)	$\mathbf{4.580 \times 10^{-1}}$	**2000**	M : 5.298×10^{-2}	$\mathbf{9.977 \times 10^{-4}}/\mathbf{4.741 \times 10^{-3}}$
	GENERIC Flow	1.055	1805.8	$\mathbf{M : 2.403 \times 10^{-6}}$	$1.123 \times 10^{-2}/1.848 \times 10^{-2}$
Allen-Cahn	Multistep + Jacobian	18.69	467.5	—	—
	Trajectory Flow (CNN)	$\mathbf{6.261 \times 10^{-1}}$	**2000**	—	$\mathbf{9.304 \times 10^{-4}}/\mathbf{5.630 \times 10^{-3}}$
	GENERIC Flow	4.808	1964.8	—	$6.170 \times 10^{-3}/4.166 \times 10^{-2}$
2D Burgers	Multistep + Jacobian	171.1	26.6	M : 7.600	$3.813 \times 10^{-1}/7.361 \times 10^{-1}$
	Trajectory Flow (CNN)	**10.40**	**425.3**	M : 3.199	$3.483 \times 10^{-1}/5.086 \times 10^{-1}$
	GENERIC Flow	56.77	354.1	$\mathbf{M : 2.814 \times 10^{-7}}$	$\mathbf{6.429 \times 10^{-2}}/\mathbf{1.010 \times 10^{-1}}$

PDE	Model	nRMSE@100 ↓ ($\times 10^{-3}$)	t_{acc}@10% ↑	Maximum invariant drift ↓ (M, Q_2, H_β)
KdV	Multistep + Jacobian	32.92	596.0	$(13.870,\ 8.495 \times 10^{-1},\ 25.23)$
	Trajectory Flow (CNN)	**2.460**	**930.6**	$(61.360,\ 5.995,\ 2.229 \times 10^{4})$
	GENERIC Flow	4.561	747.4	$(\mathbf{5.680 \times 10^{-5}},\ \mathbf{3.065 \times 10^{-6}},\ \mathbf{1.426 \times 10^{-2}})$

Accurate trajectories do not guarantee invariant conservation. At $H = 100$, the Trajectory Flow model achieves the lowest state-space error among the three models on all five one-dimensional PDE families and on 2D Burgers, but this does not translate into stronger invariant preservation. On one-dimensional Advection, Heat, and Burgers, GENERIC Flow reduces maximum mass drift of the Trajectory Flow models from approximately 10^{-2} to 10^{-6}. On 2D Burgers, it reduces the drift from 3.199 to 2.814×10^{-7}. The distinction is particularly pronounced for KdV. Despite having somewhat higher state error, GENERIC Flow preserves mass, Q_2, and the Hamiltonian orders of magnitude more accurately than the Trajectory Flow model.

Invariant conservation does not guarantee correct dissipation. The opposite ordering appears for dissipation. Across the one-dimensional Advection, Heat, Burgers, and Allen-Cahn systems, the unrestricted Trajectory Flow model more accurately tracks both the functional trajectory and its step-to-step increments, whereas the GENERIC Flow is better on 2D Burgers. For 2D Burgers, we exclude the $\nu = 0.01$ setting from all three tabulated aggregates and report it separately. At this setting, the $H = 100$ nRMSE of Trajectory Flow model and GENERIC Flow is comparable (0.1669 versus 0.1690), but the unrestricted model develops a late instability over the full rollout: Its functional trajectory / increment errors rise to $1.426 \times 10^3/7.746 \times 10^4$, compared with $0.1570/0.3140$ for GENERIC Flow, while maximum mass drift is 29.58 versus 4.519×10^{-7}. This is a concrete example in which the imposed structure prevents a catastrophic late instability despite similar short-horizon state accuracy. The structured model greatly improves conservation. However, preserving invariants and enforcing the correct direction of dissipation do not determine the magnitude of that dissipation. Matched coefficient-level comparisons are provided in Appendix J.

6.4 GENERIC WITH DISSIPATIVE RATE SUPERVISION

GENERIC Flow substantially improves invariant preservation, but its dissipative functionals are generally less accurate than those of unrestricted Trajectory Flow model. We therefore evaluate the extension from Section 5, which adds local and finite-step functional supervision. Apart from a couple of cases, rate supervision mostly improves the dissipation quality over GENERIC Flow. The extension improves functional trajectory error on Advection, Heat, and Burgers, and functional-increment error on Advection, Heat, and Allen-Cahn as seen in Figure 3. Overall, explicit rate and finite-step supervision recovers part of the dissipative fidelity lost under hard structural constraints while retaining strong invariant preservation, but it does not fully close the gap to unrestricted Trajectory Flow model.

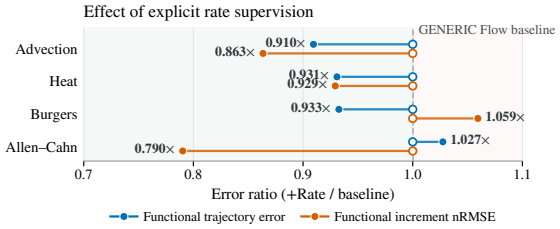


Figure 3: **Effect of GENERIC Flow + Rate.** Functional trajectory and increment-error ratios relative to GENERIC Flow; values below one indicate improvement.

7 CONCLUSION AND LIMITATIONS

We studied three distinct notions of fidelity in learned physical systems under autoregressive deployment: *local prediction accuracy, long-horizon trajectory fidelity, and physical fidelity*. We investigate how temporal representation affects long-horizon accuracy, finding that *continuous-time generators achieve stronger autoregressive fidelity than directly learned transitions, with lower local transition defect and weaker deployed-step sensitivity*. However, continuous-time generators alone do not guarantee physical fidelity and can violate conserved quantities. We address this by embedding *GENERIC-inspired structure* into the learned generator, *greatly improving invariant preservation* while retaining strong trajectory accuracy. Yet *structural admissibility alone does not determine the quantitative rate of dissipative evolution*, even when the direction of change is correct. Explicit supervision of dissipation rate partially recovers this information, indicating that it is not supplied by structure alone. Finally, the trajectory-accuracy gap between GENERIC Flow and unrestricted Trajectory Flow model narrows as temporal observations become sparser, as shown in Appendix I. Taken together, these results show that *long-horizon trajectory fidelity, invariant preservation, and quantitative dissipative fidelity are distinct properties of learned PDE dynamics and should not be treated as interchangeable proxies for dynamical correctness*.

Our study has two main limitations. Most experiments use one-dimensional PDEs, with 2D Burgers providing an initial check beyond this setting, so broader multidimensional validation remains an important direction. In addition, the structured models use selected physically meaningful functionals to define their constraints. While this requires substantially less information than the full governing PDE, extending the framework to settings where such quantities are only partially known or must themselves be learned would broaden its applicability.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools for the following tasks: **1)** Sentence- and short-paragraph-level editing of the manuscript, primarily to improve clarity and compress the writing to satisfy page limits (GPT 5.6 Sol). **2)** Cleaning and organising a large and heterogeneous code repository for release (Codex 5.6 Sol Ultra). **3)** Assisting with hyperparameter tuning across different models and training configurations, and with drafting the reproducibility sections of the appendix from the existing codebase (Codex 5.6 Sol Ultra). **4)** Reviewing and verifying author-written code, particularly where the repository contained multiple model classes, PDE systems, training procedures, and evaluation pipelines (Codex 5.6 Sol Ultra).

The research questions, mathematical formulations, implementation, analysis of experimental results, and final scientific conclusions were developed by the authors.

All AI-assisted edits, code changes, experimental choices, and documentation were manually reviewed and verified by the authors, who take responsibility for the final manuscript, code, and reported results.

ETHICS STATEMENT

(This section is **recommended** and does not count toward the page limit.)

There is no violation of ethics according to the authors in this work.

REPRODUCIBILITY STATEMENT

(This section is **recommended** and does not count toward the page limit.)

Section 5 describes the proposed GENERIC architecture in the main text. Additional reproducibility details for GENERIC Flow, including the loss functions and rollout curriculum, are provided in Appendix C. Baseline model and training details are provided in Appendix A.1, while the dataset-generation procedure is described in Appendix A.2. Mathematical definitions of the evaluation metrics are provided in Appendix E, and the physical functionals for each PDE are defined in Appendix D. The code is also provided as part of the supplementary material.

REFERENCES

Ali Baheri and Lars Lindemann. Metriplectic conditional flow matching for dissipative dynamics, 2025. URL https://arxiv.org/abs/2509.19526.

Justin Baker, Hedi Xia, Yiwei Wang, Elena Cherkaev, Akil Narayan, Long Chen, Jack Xin, Andrea L. Bertozzi, Stanley J. Osher, and Bao Wang. Proximal implicit ode solvers for accelerating learning neural odes, 2022. URL https://arxiv.org/abs/2204.08621.

Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vSix3HPYKSU.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam M Oberman. How to train your neural ode: the world of jacobian and kinetic regularization. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Sam Greydanus, Misko Dzamba, and Jason Yosinski. *Hamiltonian neural networks*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Miroslav Grmela and Hans Christian Öttinger. Dynamics and thermodynamics of complex fluids. i. development of a general formalism. *Phys. Rev. E*, 56:6620–6632, Dec 1997. doi: 10.1103/PhysRevE.56.6620. URL https://link.aps.org/doi/10.1103/PhysRevE.56.6620.

Derek Hansen, Danielle C. Maddix, Shima Alizadeh, Gaurav Gupta, and Michael W. Mahoney. Learning physical models that can respect conservation laws. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2 edition, 2012.

Xianglong Hou, Xinquan Huang, and Paris Perdikaris. CFO: Learning continuous-time PDE dynamics via flow-matched neural operators. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=IQhaeSzyup.

Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. DiffusionPDE: Generative PDE-solving under partial observation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=z0I2SbjN0R.

Patrick Kidger, James Morrill, Jamshid Sourati, and Terry Lyons. Learning differential equations that are easy to solve. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2599–2610. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/2e255d2d6bf9bb33030246d31f1a79ca-Abstract.html.

Felix Koehler, Simon Niedermayr, rüdiger westermann, and Nils Thuerey. APEBench: A benchmark for autoregressive neural emulators of PDEs. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=iWc0qE116u.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: learning maps between function spaces with applications to pdes. *J. Mach. Learn. Res.*, 24(1), January 2023. ISSN 1532-4435.

Jiayi Li and Flora D. Salim. DRIFT-net: A spectral-coupled neural operator for PDEs learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=qTvmTCDRyD.

Jiayi Li, Penghao Jiang, Hira Saleem, Zhaonan Wang, Piotr Koniusz, and Flora D. Salim. Sgno: Spectral generator neural operators for stable long horizon pde rollouts, 2026. URL https://arxiv.org/abs/2602.18801.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM / IMS J. Data Sci.*, 1(3), May 2024. doi: 10.1145/3648506. URL https://doi.org/10.1145/3648506.

Phillip Lippe, Bastiaan S. Veeling, Paris Perdikaris, Richard E Turner, and Johannes Brandstetter. Modeling accurate long rollouts with temporal neural PDE solvers. In *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023. URL https://openreview.net/forum?id=EGTY6V76b3.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021. doi: 10.1038/s42256-021-00302-5.

Yulong Lu and Wuzhe Xu. Generative downscaling of pde solvers with physics-guided diffusion models. *J. Sci. Comput.*, 101(3), November 2024. ISSN 0885-7474. doi: 10.1007/s10915-024-02709-9. URL https://doi.org/10.1007/s10915-024-02709-9.

Michael McCabe, Peter Harrington, Shashank Subramanian, and Jed Brown. Towards stability of autoregressive neural operators. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=RFfUUtKYOG.

Nick McGreivy and Ammar Hakim. Invariant preservation in machine learned PDE solvers via error correction. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023. URL https://openreview.net/forum?id=RLfs4A0ZJg-.

Fengxiang Nie and Yasuhiro Suzuki. Jaws: Enhancing long-term rollout of neural pde solvers via spatially-adaptive jacobian regularization, 2026. URL https://arxiv.org/abs/2603.05538.

M. Raissi, P. Perdikaris, and G.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. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Andrew M. Stuart. Numerical analysis of dynamical systems. *Acta Numerica*, 3:467–572, 1994. doi: 10.1017/S0962492900002488.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: an extensive benchmark for scientific machine learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Hongyu Wang, Jingquan Wang, and Dan Negrut. Fnode: Flow-matching for data-driven simulation of constrained multibody systems. *Computer Methods in Applied Mechanics and Engineering*, 455:118912, 2026. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2026.118912. URL https://www.sciencedirect.com/science/article/pii/S0045782526001854.

Haijun Yu, Xinyuan Tian, Weinan E, and Qianxiao Li. Onsagernet: Learning stable and interpretable dynamics using a generalized onsager principle. *Physical Review Fluids*, 6(11), November 2021. ISSN 2469-990X. doi: 10.1103/physrevfluids.6.114402. URL http://dx.doi.org/10.1103/PhysRevFluids.6.114402.

Jiaquan Zhang, Caiyan Qin, Haoyu Bian, Libin Cai, Yi Lu, Chaoning Zhang, Wei Dong, Yuanfang Guo, Yang Yang, and Heng Tao Shen. Autoregression-free neural operators for time-dependent pdes, 2026. URL https://arxiv.org/abs/2605.25413.

Xi Zhang, Yuan Pu, Yuki Kawamura, Andrew Loza, Yoshua Bengio, Dennis Shung, and Alexander Tong. Trajectory flow matching with applications to clinical time series modelling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=fNakQltI1N.

Zhen Zhang, Yeonjong Shin, and George Em Karniadakis. Gfinns: Generic formalism informed neural networks for deterministic and stochastic dynamical systems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 380(2229), 2022. ISSN 1471-2962. doi: 10.1098/rsta.2021.0207. URL http://dx.doi.org/10.1098/rsta.2021.0207.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ode-net: Learning hamiltonian dynamics with control. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxmb1rKDS.

A Reproducibility Details

A.1 Models

Each equation-coefficient pair is assigned a separate model. The state is standardized by a scalar mean and standard deviation computed from the fitting trajectories, and predictions are transformed back to physical units before evaluation. Time-conditioned models concatenate the state with an eight-dimensional sinusoidal embedding of the normalized source-snapshot index k/T; the largest embedding period is 100. Here, $T = 2000$ for MTS-1D and $T = 499$ for 2D Burgers. The selected spatial architectures are summarized in Table 4.

Table 4: Architectures used for the reported models.

Backbone	Configuration
CNN	Six periodic convolution blocks with kernel size 5, pointwise skips, GELU activations, and projection width 128. The widths are 128 in 1D and 64 in 2D.
FNO	Four Fourier blocks, 32 retained modes, width 48, projection width 128, pointwise local branches, GELU after the first three blocks, and a linear final Fourier block. Complex spectral-weight components are initialized with standard deviation 0.02. The 2D FNO instead retains 12 modes per axis with width 32 and projection width 64.
GENERIC Flow	The same six-block periodic CNN trunk, with rank-64 conservative factors U, V, rank-64 dissipative factor B, and separate scalar tangent and compatibility-slack output fields in 1D. The 2D model uses the 2D CNN trunk, a rank-64 PSD factor, and a tangent field, with hard mass projection and tangent orthogonalization against the squared-L_2 gradient.

The reported configurations were hyperparameter tuned on held-out development trajectories. CNN tuning screened periodic versus zero padding, learning rates $\{3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}\}$, training objectives, and random seeds. FNO tuning first crossed learning rates $3 \times 10^{-4}, \{10^{-4}, 10^{-3}\}$ with complex-Xavier, medium and small spectral initialization, then compared $(\text{modes}, \text{width}, \text{blocks}, \text{projection width})$ equal to $(32, 32, 4, 64)$, $(32, 48, 4, 128)$, $(64, 34, 4, 128)$, and $(64, 48, 4, 128)$. Flow-CNN tuning additionally screened interpolation-time sampling, input noise, padding, learning-rate schedules, and short solver-in-the-loop objectives. Direct-model selection used the mean closed-rollout $\log(1 + \text{relative-}\ell_2)$ AUC on 128-256-step windows starting at indices 0, 500, 1000, and 1500; promoted Flow settings were then checked on 2,000-step held-out rollouts. These searches selected the periodic CNN and the width-48 FNO in Table 4. The selected peak learning rates are listed in Table 5. The 2D study uses the selected 1D Burgers learning rates with the 2D capacities in Table 4 and selects checkpoints by 128-step closed validation-rollout AUC.

Table 5: Selected peak learning rates by PDE family.

Model	Advection	Allen-Cahn	Burgers	Heat	KdV	2D Burgers
Direct one-step CNN	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Trajectory Flow model CNN	3×10^{-4}	3×10^{-4}	3×10^{-4}	10^{-4}	10^{-5}	3×10^{-4}
Direct one-step FNO	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}
Trajectory Flow model FNO	10^{-4}	10^{-4}	10^{-3}	10^{-3}	3×10^{-4}	10^{-3}
GENERIC Flow	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-5}	10^{-5}	3×10^{-4}

Direct one-step predictors are trained for 20 epochs with batch size 512, Adam, and normalized one-step MSE. Trajectory Flow models are trained for 50 epochs, with 400 batches of size 512 per epoch, Adam, global-gradient clipping at 1, and a 5% linear warmup followed by cosine decay to 10^{-5}. Their target is the normalized secant velocity along uniformly sampled linear interpolants. At deployment, the direct one-step predictor uses one network evaluation and Trajectory Flow model uses eight explicit-midpoint substeps, or 16 neural-function evaluations, per stored transition. The multistep ablation of the direct predictor uses a 10-step rollout objective, 30 epochs, batch size 128, the same 5% warmup/cosine schedule, and the same architecture and per-PDE learning rate. Its Jacobian variant adds a random-direction finite-difference penalty with weight 10^{-4} and perturbation 10^{-3}; the one-step Jacobian ablation uses the same penalty settings in the 20-epoch one-step recipe.

For 2D Burgers, the corresponding batch sizes are 8 for the direct one-step predictor and Trajectory Flow model, 4 for the multistep objective, 8 for GENERIC pointwise updates, and 2 for GENERIC rollout updates.

The GENERIC Flow recipe is trained at model step $h = 1$ for 50 epochs, with 400 pointwise updates of batch size 512 and 20 rollout updates of batch size 16 per epoch. The rollout curriculum is $1 \to 4 \to 16 \to 64$. Adam uses global-gradient clipping at 1, a 5% warmup, and cosine decay to one tenth of the peak rate. The reported pointwise objective assigns coefficients $1, 1, 0, 10$, and 1 to the velocity, structure, tangent, slack, and endpoint terms, respectively. The rollout-state, spatial-gradient, functional-increment, canonical-increment, and JVP-hinge weights are $1, 0.1, 0.1, 0.1$, and 0.1, with an unpenalized JVP-gain threshold of 1. GENERIC Flow uses one explicit-midpoint substep, or two neural-function evaluations, per stored transition. The reported base GENERIC Flow results cover all 21 benchmark equation-coefficient pairs at model seeds 0, 1, and 2. The two-seed rate-supervised extension uses $\beta_{\mathrm{rate}} = \beta_{\mathrm{step}} = 0.1$ and projects the slack field onto the mass-preserving tangent space for Advection, Heat, and Burgers.

A.1.1 Two-dimensional Burgers model protocol

For each viscosity, we train eleven separate models: direct one-step and multistep predictors, with and without Jacobian regularization, using both CNN and FNO backbones; CNN and FNO Trajectory Flow model; and CNN GENERIC Flow. All models use the scalar mean and standard deviation computed over the 16 training trajectories. The experiment seeds are 42,44,45 with fixed method-specific key offsets; each Jacobian-regularized direct model shares its initialization and minibatch stream with the corresponding non-Jacobian model.

Direct one-step predictors use 20 epochs, 400 updates per epoch, and batch size 8. Multistep models use 30 epochs, 400 updates per epoch, batch size 4, and a 10-step closed-rollout loss. The Jacobian variants add a random-direction finite-difference penalty on the normalized deployed map with weight 10^{-4} and perturbation 10^{-3}. Trajectory Flow model uses 50 epochs, 400 updates of batch size 8 per epoch, and normalized secant-velocity targets at uniformly sampled points along each observed transition. Peak learning rates are 3×10^{-4} for CNN models and 10^{-3} for FNO models, as listed in Table 5. Multistep and generator models use a 5% linear warmup, cosine decay to 10^{-5}, and global-gradient clipping at one.

The 2D GENERIC model uses 50 epochs with 400 pointwise updates of batch size 8 and 20 rollout updates of batch size 2 per epoch. Its rollout curriculum is $1 \to 4 \to 16 \to 64$. Pointwise training combines secant-velocity and midpoint-endpoint losses with a kinetic-energy increment loss of weight 0.1. Rollout training combines normalized state error, spatial-gradient error of weight 0.1, and kinetic-energy increment error of weight 0.1. The rank-64 PSD branch and tangent branch are projected to zero spatial mean, while the tangent branch is additionally orthogonalized against the squared-L_2 gradient.

Checkpoints are scored at epoch 1, every five epochs, and the final epoch. The selection score is the mean $\log(1 + \text{relative-}\ell_2)$ AUC over four deterministic 128-step closed rollouts from the validation split, and the lowest-scoring checkpoint is used for test evaluation. Direct models require one network evaluation per stored transition. Trajectory Flow model uses eight explicit-midpoint substeps, or 16 network evaluations, while GENERIC Flow uses one explicit-midpoint substep, or two network evaluations. Final evaluation rolls each selected checkpoint from all eight test initial states for 499 transitions.

A.1.2 Rationale for method-specific training budgets

We intentionally do not match nominal epoch counts across methods because an epoch is not a common unit of optimization work for these objectives. In our implementations, an epoch for a direct one-step predictor is a pass over observed transition pairs, a Trajectory Flow model epoch contains 400 sampled interpolation updates, and a GENERIC Flow epoch additionally contains 20 rollout updates through a curriculum reaching horizon 64, together with its structured and JVP terms. Equalizing epoch counts would therefore equalize neither optimizer updates nor learned-function evaluations, FLOPs, or wall-clock time.

Instead, we treat the finite training horizon as a method-specific optimization choice informed by the staged pilot and held-out-development campaigns described above. Model configurations are selected using closed-rollout validation error, and the 2D study selects the best checkpoint within the declared horizon by the same validation principle. The resulting comparisons are therefore intended to assess the predictive and physical performance of independently tuned methods, not training-compute efficiency. We report each training recipe explicitly and evaluate deployment efficiency separately using the NFE- and wall-clock-matched controls in Appendix B.

A.2 Data

The one-dimensional experiments use the MTS-1D benchmark generated on the periodic domain $[0, 2\pi)$ with 256 uniformly spaced points. Each coefficient has a float32 development array of shape $(50, 2001, 256)$ and a disjoint test array of shape $(20, 2001, 256)$. The nominal stored interval is $\Delta t = 10^{-3}$, giving 2,000 forecast transitions per trajectory. The 21 equation-coefficient pairs and reference discretizations are listed in Table 6. The simulated equations are

$$\begin{aligned} \text{Advection:} \quad & u_t = -\beta u_x, \\ \text{Heat:} \quad & u_t = \kappa u_{xx}, \\ \text{Burgers:} \quad & u_t = -\partial_x(u^2/2) + (\nu/\pi) u_{xx}, \\ \text{Allen-Cahn:} \quad & u_t = \epsilon^2 u_{xx} + u - u^3, \\ \text{KdV:} \quad & u_t = -3\partial_x(u^2) - \beta u_{xxx}, \\ \text{2D Burgers:} \quad & u_t = -\partial_x(u^2/2) - \partial_y(u^2/2) + \nu \Delta u. \end{aligned} \tag{9}$$

Initial conditions have the common form

$$u_0(x) = a \sum_{j \in \mathcal{K}} A_j \sin(jx + \phi_j) + b \exp\left[-\frac{(x-c)^2}{2w^2} \right], \tag{10}$$

where $A_j \sim \mathcal{N}(0,1)$ and $\phi_j, c \sim \mathcal{U}(0, 2\pi)$. Advection, Heat, and Burgers use $(\mathcal{K}, a, b, w) = (\{1, \ldots, 6\}, 0.3, 0.5, 0.2)$; Allen-Cahn uses $(\{1, \ldots, 6\}, 0.5, 0.5, 0.3)$ and subtracts the spatial mean; KdV uses $(\{1, \ldots, 4\}, 0.25, 0.1, 0.3)$. JAX generation keys 0 and 1 produce the development and test initial conditions, respectively, and indexed initial conditions are shared across coefficient values within a PDE family. KdV uses a CFL-limited internal step selected by its advection and dispersion bounds; the effective KdV cadence is not stored in the NPZ metadata, so KdV horizons are reported in stored-step units.

A.2.1 Two-dimensional Burgers data generation

The scalar 2D benchmark solves

$$u_t + \partial_x(u^2/2) + \partial_y(u^2/2) = \nu \Delta u, \qquad \nu \in \{0.01, 0.03, 0.1, 0.3\}, \tag{11}$$

on the periodic domain $[0, 2\pi)^2$. Each state is stored as one channel on the uniform 64×64 grid. Spatial derivatives use a Fourier pseudospectral discretization with the two-thirds dealiasing mask applied to the state and nonlinear flux. Classical fourth-order Runge-Kutta advances each stored interval $\Delta t = 0.01$ using two internal steps of size 0.005. After every internal step, the spatial mean is reset to its initial value, removing floating-point drift in the analytically constant zero Fourier mode.

Each initial field is the sum of the seven modes $(1,0), (0,1), (1,1), (1,2), (2,1), (2,3), (3,2)$ with independent standard-normal amplitudes and phases uniform on $[0, 2\pi)$. A mean-centered periodic Gaussian bump of width 0.45 is added; its center is uniform on the domain and its amplitude is uniform on $[-0.35, 0.35]$. The resulting zero-mean component is rescaled to spatial standard deviation 0.5, after which a prescribed mean sampled uniformly from $[-0.35, 0.35]$ is added.

For every viscosity, split seeds 0, 1, and 2 generate 16 training, four validation, and eight test trajectories, respectively. Indexed initial conditions are shared across viscosities within each split. Every trajectory contains 500 float32 snapshots, producing arrays of shapes $(16, 500, 64, 64, 1)$, $(4, 500, 64, 64, 1)$, and $(8, 500, 64, 64, 1)$. Each archive stores the equation identifier, viscosity, grid

size, domain length, snapshot interval, number of internal substeps, and zero-mode-correction flag. The test split is used only after checkpoint selection on the validation trajectories.

Hyperparameter selection uses trajectory-level validation splits from the development arrays. The GENERIC Flow campaign fixes 40 fitting trajectories and 10 validation trajectories with validation indices $\{2, 7, 13, 21, 26, 28, 31, 34, 35, 41\}$; normalization and all train-derived functional scales are computed from the 40 fitting trajectories. Final evaluation starts from each of the 20 test initial conditions and rolls each model autoregressively for 2,000 stored steps. For 2D Burgers, selection uses the four validation trajectories and final evaluation rolls the eight test trajectories autoregressively for 499 stored steps.

Table 6: Benchmark coefficients and reference trajectory generators.

PDE	Coefficients	Spatial and temporal discretization
Advection	$\beta \in \{-2, -1, -0.5, 0.5, 1, 2\}$	Sign-aware first-order upwind difference; explicit Euler
Allen-Cahn	$\epsilon \in \{0.01, 0.03, 0.1, 0.3\}$	Centered periodic Laplacian; CFL-limited explicit Euler
Burgers	$\nu \in \{0.01, 0.03, 0.1, 0.3\}$	Upwind numerical flux and centered diffusion; CFL-limited explicit Euler
2D Burgers	$\nu \in \{0.01, 0.03, 0.1, 0.3\}$	Two-thirds-dealiased pseudospectral derivatives; RK4 with two internal substeps and zero-mode correction
Heat	$\kappa \in \{0.02, 0.05, 0.1, 0.2\}$	Centered periodic Laplacian; explicit Euler
KdV	$\beta \in \{0.01, 0.03, 0.05\}$	Two-thirds-dealiased pseudospectral derivatives; CFL-limited RK4

B Compute-Matched Deployment Controls

The main experiments deploy Trajectory Flow model using explicit midpoint integration with eight substeps per stored transition. We therefore perform two controls to separate the effect of the continuous-time generator representation from additional inference compute. First, we compare the direct one-step predictor and Trajectory Flow model at exactly one neural-function evaluation (NFE) per stored step. Second, we measure steady-state wall-clock time under a controlled GPU setup.

B.1 NFE-Matched Solver Transfer

We reuse the frozen time-conditioned checkpoints for the direct one-step predictors and Trajectory Flow models and change only the deployment rule. No solver-specific retraining is performed. The compared deployments are

Direct one-step	:	direct finite-step map, 1 NFE/step,
Trajectory Flow model-Euler	:	1 Euler step, 1 NFE/step,
Trajectory Flow model-midpoint	:	1 midpoint step, 2 NFE/step,
Trajectory Flow model-RK4	:	1 RK4 step, 4 NFE/step.

The native main-paper Flow deployment uses midpoint integration with eight substeps, corresponding to 16 NFE per stored transition; it is not part of the matched-budget comparison.

Table 7 gives the primary one-NFE control. Each cell reports cumulative nRMSE@100 followed by the 30% accuracy horizon t_{acc}. At the same one-NFE budget, Euler-deployed Trajectory Flow model outperforms the direct one-step predictor on both metrics for all five PDE families and both backbones.

Across the ten backbone-PDE comparisons, the reduction in nRMSE@100 ranges from $24.2\times$ to $5.46 \times 10^7\times$. Thus the long-horizon advantage of Trajectory Flow model is not explained solely by midpoint using an additional vector-field evaluation.

Higher-order integration provides a secondary refinement rather than the primary representation-level effect. Table 8 reports the equal-equation macro across all 21 equation settings. Euler already preserves the large advantage over direct one-step prediction, while midpoint and RK4 further improve some harder late-horizon cases. In particular, the effect is most visible for KdV and for the late FNO Advection tail.

Table 7: **One-NFE comparison.** Each entry reports nRMSE@100 / $t_{\mathrm{acc}}@30\%$, with nRMSE in 10^{-2} units. Trajectory Flow model uses a single Euler evaluation per stored step, exactly matching the one neural-function evaluation of the direct one-step predictor.

Backbone	Method	Advection	Heat	Burgers	Allen-Cahn	KdV
CNN	Direct one-step	14.749 / 362.5	7.399 / 321.5	35.586 / 278.8	38.744 / 149.8	27.506 / 128.3
CNN	Flow-Euler	0.1270 / 2000	0.05698 / 2000	0.06105 / 2000	0.07161 / 2000	1.136 / 293.0
FNO	Direct one-step	2.999×10^7 / 21.3	198.25 / 16.5	25.852 / 72.2	39.862 / 65.0	158.65 / 49.7
FNO	Flow-Euler	0.5494 / 1871.2	0.4138 / 2000	0.7362 / 2000	0.3278 / 2000	1.708 / 721.0

Table 8: **Solver-budget sensitivity.** Equal-equation macro over all 21 equation settings. The native main-paper eight-substep midpoint deployment is not included.

Backbone	Deployment	NFE	nRMSE@100 ↓	$t_{\mathrm{acc}}@30\%$ ↑
CNN	Direct one-step	1	0.23711	264.8
CNN	Flow-Euler	1	0.002347	1756.1
CNN	Flow-midpoint	2	0.0008781	1897.4
CNN	Flow-RK4	4	0.0008676	1921.5
FNO	Direct one-step	1	8.569×10^4	42.5
FNO	Flow-Euler	1	0.006824	1780.5
FNO	Flow-midpoint	2	0.006586	1798.6
FNO	Flow-RK4	4	0.006590	1798.0

Table 9: **Controlled steady-state wall-clock.** Timing covers only the compiled model rollout after warm-up. Relative cost is normalized to the direct one-step CNN.

Method	Deployment	NFE/step	Median (s)	Relative cost
Direct one-step CNN	direct	1	1.16042	1.000×
Trajectory Flow model CNN	Euler	1	1.16021	1.000×
Trajectory Flow model CNN	midpoint	2	2.32862	2.007×
GENERIC Flow	midpoint	2	3.11291	2.683×

These results support a narrow conclusion: higher-order integration can improve deployment quality, particularly in difficult late-horizon regimes, but it is not responsible for the primary gap between direct one-step prediction and Flow. The continuous-time generator advantage remains when both methods are restricted to one neural-function evaluation per stored step.

B.2 Controlled Wall-Clock Measurement

NFE controls the number of neural-network evaluations but does not by itself determine latency, since different architectures may have different per-evaluation costs. We therefore additionally measure steady-state rollout time in a controlled setting.

All methods are evaluated sequentially on a single NVIDIA A40 using the CNN backbone on Burgers with $\nu = 0.1$, float32 precision, 20 trajectories, 2,000 forecast steps, and 256 spatial points. Model loading, dataset loading, JIT compilation, host transfer, and result serialization are excluded. Two warm-up runs precede seven synchronized timed repetitions; we report the median and interquartile range.

The direct one-step CNN and one-NFE Flow-Euler are indistinguishable at the displayed precision in this controlled case. Midpoint approximately doubles the ordinary Flow runtime, consistent with doubling the number of neural-function evaluations. GENERIC Flow is more expensive than ordinary Flow at the same declared two NFE per step because each evaluation includes additional structured branches and projections. The wall-clock experiment therefore supports only a controlled implementation claim, rather than a hardware-independent efficiency ranking. In this setting, matching the direct one-step predictor and Trajectory Flow model at one NFE also approximately matches their steady-state runtime, while the large trajectory-fidelity advantage of Flow remains.

NOTATION

Table 10 summarizes the notation used most frequently in the main text and appendices.

Table 10: **Frequently used notation.**

Symbol	Meaning
States and temporal evolution	
$u(t)$, u_k	Reference state, with $u_k = u(t_k)$. In Appendix C, u_k denotes the normalized model state unless stated otherwise.
$\hat{u}_k$	Model-predicted state during autoregressive deployment.
N_x, N_t	Number of spatial degrees of freedom and number of stored states in a trajectory, respectively.
t_k, Δt	Physical observation time and spacing between consecutive stored observations.
$f^\star$, f_θ	True and learned continuous-time vector fields, respectively.
$F^\star$, F_θ	True and learned finite-step transitions. For a direct predictor, F_θ is parameterized directly; for a continuous generator it is induced by numerical integration of f_θ.
Φ_θ	Numerical flow map obtained by integrating the learned generator; $\Phi_\theta^{(h)}$ denotes repeated application over h rollout steps.
v_k, v_k^{src}	Secant/velocity targets. Trajectory Flow uses $v_k = (u_{k+1} - u_k)/\Delta t$. In the normalized 1D GENERIC training, $v_k^{\mathrm{src}} = u_{k+1} - u_k$ because one stored transition is treated as one unit of model time.
τ, u_τ	Within-transition interpolation coordinate and interpolated state, $u_\tau = (1-\tau)u_k + \tau u_{k+1}$.
s_k	Normalized source clock, $s_k = k/(N_t - 1)$, used by the structured time-conditioned generator.
e_k, δ_k, J_k	Autoregressive state error, local transition defect, and deployed-step state Jacobian, respectively.
$\mathcal{D}(u) = \mu + \sigma u$	Map from the normalized model state back to physical units; μ and σ are the training-data mean and standard deviation.
GENERIC structure	
$E(u)$, $S(u)$	Declared energy-like invariant and dissipative/entropy-like functional used by the structured generator.
$L_\theta(u)$, $M_\theta(u)$	Learned reversible and irreversible operators. L_θ is skew-symmetric and M_θ is positive semidefinite.
$P_q^\perp$	Projection onto the subspace orthogonal to q, used to impose the GENERIC degeneracy conditions.
P_{adm}	Projector onto the tangent space shared by the declared auxiliary invariants.
f_{hard}, f_{tangent}, f_{slack}	Hard GENERIC core, invariant-tangent flexibility branch, and regularized compatibility residual; their sum defines the structured generator.
Q_j, Q_{can}	Auxiliary declared invariants and the separate PDE-specific canonical functional used by the rollout objective.
Physical functionals and evaluation	
$M(u)$, $Q_2(u)$	Discrete mass and squared-ℓ_2 functionals.
$F_\epsilon(u)$, $H_\beta(u)$	Allen-Cahn free energy and KdV Hamiltonian, respectively.
$G_+(u)$, $K(u)$	Periodic discrete gradient energy used as Q_{can} for the 1D Advection/Heat/Burgers systems, and kinetic energy used by the 2D Burgers objective.
nRMSE@H	Cumulative normalized state-space error over forecast horizons up to H.
t_{acc}	First forecast step at which the trajectory-relative state error exceeds the prescribed accuracy threshold.
$D_Q(t)$, $D_Q^{\max}$	Invariant self-drift at time t and its maximum over rollout time and trajectories.
$e_Q(t)$	Target-relative error between the predicted and reference trajectories of a physical functional Q.
$\mathrm{nRMSE}_{\Delta Q}$	Normalized error in the step-to-step increments of a non-invariant physical functional.

C GENERIC FLOW: ARCHITECTURE AND TRAINING SPECIFICATION

This section provides the implementation-level specification of GENERIC Flow and the rate-supervised extension used in the experiments.

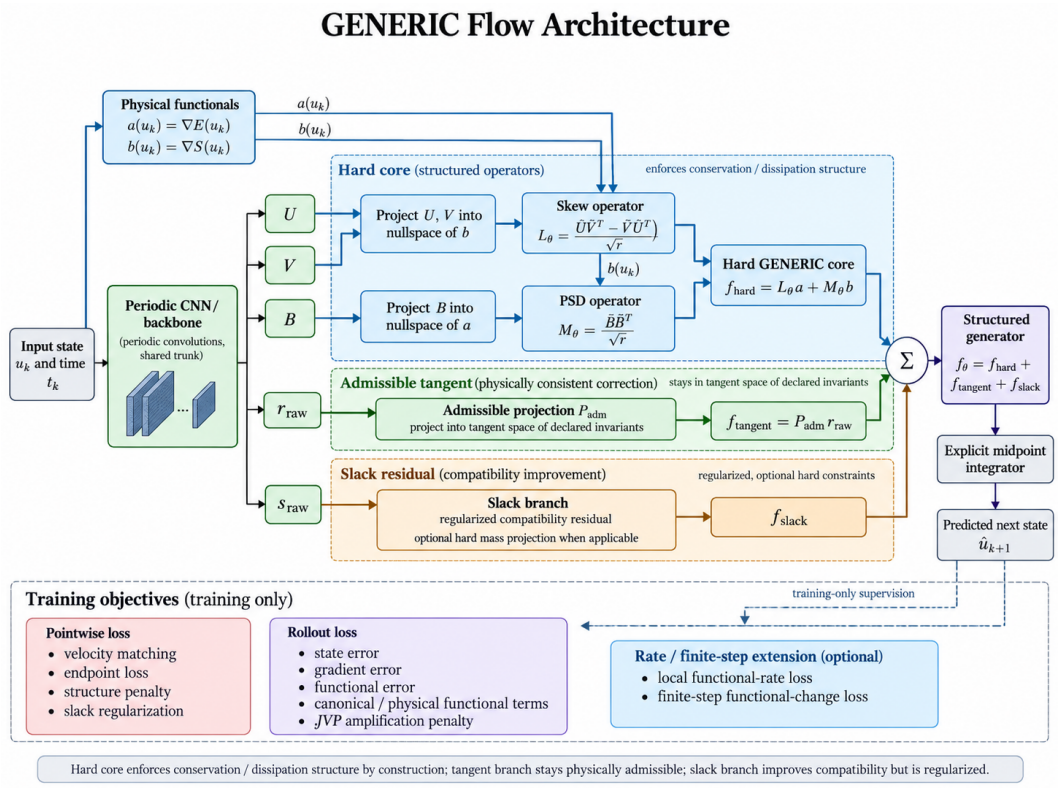


Figure 4: GENERIC Flow architecture. A shared backbone produces the factors defining the hard structured core together with tangent and slack branches. The hard core enforces conservative and dissipative constraints by construction, the tangent branch adds admissible flexibility, and the slack branch provides a regularized compatibility residual. Their sum defines the structured generator, which is numerically integrated to obtain the next state. Training uses pointwise and rollout losses, with optional rate and finite-step functional supervision.

C.1 Structured Generator

Let

$$a(u) = \nabla E(u), \qquad b(u) = \nabla S(u).$$

The hard GENERIC component is

$$f_{\text{hard}}(u) = L_\theta(u)a(u) + M_\theta(u)b(u),$$

with

$$L_\theta^\top = -L_\theta, \qquad M_\theta \succeq 0, \qquad L_\theta b = 0, \qquad M_\theta a = 0.$$

The network predicts low-rank factors U, V, B. For

$$P_{\perp q} = I - \frac{qq^\top}{q^\top q + \varepsilon},$$

we define

$$\widetilde{U} = P_{\perp b}U, \qquad \widetilde{V} = P_{\perp b}V, \qquad \widetilde{B} = P_{\perp a}B,$$

and construct

$$L_\theta = \frac{\widetilde{U}\widetilde{V}^\top - \widetilde{V}\widetilde{U}^\top}{\sqrt{r}}, \qquad M_\theta = \frac{\widetilde{B}\widetilde{B}^\top}{\sqrt{r}}.$$

The complete learned vector field is

$$f_\theta = f_{\text{hard}} + f_{\text{tangent}} + f_{\text{slack}}.$$

For invariant gradients collected in

$$A(u) = [\nabla Q_1(u), \dots, \nabla Q_m(u)],$$

the tangent projector is

$$P_{\mathrm{adm}} = I - A(A^\top A + \varepsilon I)^{-1} A^\top,$$

with

$$f_{\mathrm{tangent}} = P_{\mathrm{adm}} r_\theta(u, t).$$

The slack branch is an additional learned residual that is explicitly regularized during training.

C.2 Pointwise and Rollout Training

Unless stated otherwise, u_k denotes the normalized model state and $\mathcal{D}(u) = \mu + \sigma u$ maps it back to physical units. For an observed transition (u_k, u_{k+1}), the source-state velocity target is $v_k^{\mathrm{src}} = u_{k+1} - u_k$, since one stored transition is one unit of model time. Let $\mathrm{mse}(x, y) = \mathrm{mean}[(x - y)^2]$ and define the train-set velocity scale

$$s_v^2 = \max\left\{\mathrm{mean}_{\mathrm{train}}\left[(u_{k+1} - u_k)^2\right], 10^{-12}\right\}.$$

For a trajectory with N_t stored states, the source clock $s_k = k/(N_t - 1)$ is held fixed inside a stored transition, while τ denotes the within-transition clock. The endpoint term uses the deployed explicit-midpoint map Φ_θ:

$$k_1 = f_\theta(u_k; s_k, 0), \qquad u_{k+\frac{1}{2}} = u_k + \frac{1}{2} k_1, \tag{12}$$

$$k_2 = f_\theta(u_{k+\frac{1}{2}}; s_k, \tfrac{1}{2}), \qquad \hat{u}_{k+1} = \Phi_\theta(u_k; s_k) = u_k + k_2. \tag{13}$$

The two data-fitting terms are therefore

$$\mathcal{L}_{\mathrm{velocity}} = \frac{\mathrm{mse}\big(f_\theta(u_k; s_k, 0), v_k^{\mathrm{src}}\big)}{s_v^2}, \qquad \mathcal{L}_{\mathrm{endpoint}} = \frac{\mathrm{mse}(\hat{u}_{k+1}, u_{k+1})}{s_v^2}. \tag{14}$$

For the structural term, write $f_\theta = f_1 + f_2$ for the complete reversible and irreversible branch outputs, including the applicable tangent and slack components. With $[x]_+ = \max(x, 0)$, define

$$\varepsilon_\ell = 10^{-12}, \qquad \chi_{\varepsilon_\ell}(q) = \mathbf{1}\left\{\|q\|_2^2 > \varepsilon_\ell\right\}, \qquad \mathcal{R}_{\varepsilon_\ell}(q, g) = \mathrm{mean}\left[\chi_{\varepsilon_\ell}(q) \frac{(q^\top g)^2}{\max\{\|q\|_2^2, \varepsilon_\ell\}}\right].$$

At $\widetilde{u}_k = \mathcal{D}(u_k)$, let $a = \nabla E(\widetilde{u}_k)$, $b = \nabla S(\widetilde{u}_k)$, and $g_j = \nabla Q_j(\widetilde{u}_k)$. Set $\mathcal{A}_{\mathrm{aux}} = 0$ when $m = 0$, and otherwise set $\mathcal{A}_{\mathrm{aux}} = m^{-1} \sum_{j=1}^m \mathcal{R}_{\varepsilon_\ell}(g_j, f_1 + f_2)$. The implemented structural loss is

$$\mathcal{L}_{\mathrm{structure}} = \frac{1}{s_v^2}\left(\mathcal{R}_{\varepsilon_\ell}(b, f_1) + \mathcal{R}_{\varepsilon_\ell}(a, f_2) + \mathcal{R}_{\varepsilon_\ell}(a, f_1) + \mathrm{mean}\big[\chi_{\varepsilon_\ell}(b)[-b^\top f_2]_+^2\big] + \mathcal{A}_{\mathrm{aux}}\right). \tag{15}$$

The calibrated compatibility allowance ρ_0 gives

$$\rho_k = \frac{\|f_{\mathrm{slack}}(u_k; s_k)\|_2}{\max\{\|f_\theta(u_k; s_k)\|_2, \varepsilon_\ell\}}, \qquad \mathcal{L}_{\mathrm{slack}} = \mathrm{mean}_k[\rho_k - \rho_0]_+^2. \tag{16}$$

Thus, for the reported GENERIC Flow models,

$$\mathcal{L}_{\mathrm{point}} = \mathcal{L}_{\mathrm{velocity}} + \mathcal{L}_{\mathrm{structure}} + 10\mathcal{L}_{\mathrm{slack}} + \mathcal{L}_{\mathrm{endpoint}}.$$

For rollout training, let $\hat{u}_{b,h} = \Phi_\theta^{(h)}(u_{b,0})$ and let $u_{b,h}$ be its reference counterpart. With the periodic forward difference $(D_x u)_i = (u_{i+1} - u_i)/\Delta x$, the state-space terms are

$$\mathcal{L}_{\mathrm{state}} = \frac{\mathrm{mean}_{b,h,x}(\hat{u}_{b,h} - u_{b,h})^2}{\max\{\mathrm{mean}_{b,h,x}\, u_{b,h}^2, 10^{-6}\}}, \tag{17}$$

$$\mathcal{L}_{\mathrm{gradient}} = \frac{\mathrm{mean}_{b,h,x}(D_x \hat{u}_{b,h} - D_x u_{b,h})^2}{\max\{\mathrm{mean}_{b,h,x}(D_x u_{b,h})^2, 10^{-6}\}}. \tag{18}$$

The functional losses are evaluated in physical units. For any declared functional Q, set

$$\Delta\widehat{Q}_{b,h} = Q(\mathcal{D}(\hat{u}_{b,h})) - Q(\mathcal{D}(u_{b,0})), \tag{19}$$

$$\Delta Q_{b,h} = Q(\mathcal{D}(u_{b,h})) - Q(\mathcal{D}(u_{b,0})), \tag{20}$$

$$d_Q = \mathrm{mean}_{b,h}(\Delta Q_{b,h})^2 + 10^{-6}\,\mathrm{mean}_b\, Q(\mathcal{D}(u_{b,0}))^2 + 10^{-12}, \tag{21}$$

$$\ell_Q = \frac{\mathrm{mean}_{b,h}(\Delta\widehat{Q}_{b,h} - \Delta Q_{b,h})^2}{d_Q}. \tag{22}$$

For m auxiliary invariants, the implemented aggregation is

$$\mathcal{L}_{\text{functional}} = \frac{\mathbf{1}_E\ell_E + \mathbf{1}_S\ell_S + \sum_{j=1}^{m}\ell_{Q_j}}{2+m},$$

where an inactive E or S slot contributes zero. For the separate canonical evaluation functional Q_{can}, define the same anchored increments and

$$d_{\text{can}} = \mathrm{mean}_{b,h}(\Delta Q^{\text{can}}_{b,h})^2 + 10^{-6}\max\{\mathrm{mean}_b\, Q_{\text{can}}(\mathcal{D}(u_{b,0}))^2, 10^{-8}\} + 10^{-8}, \tag{23}$$

$$\mathcal{L}_{\text{canonical}} = \frac{\mathrm{mean}_{b,h}(\Delta\widehat{Q}^{\text{can}}_{b,h} - \Delta Q^{\text{can}}_{b,h})^2}{d_{\text{can}}}, \tag{24}$$

with $\mathcal{L}_{\text{canonical}} = 0$ when no canonical functional is declared.

Finally, for a Gaussian probe $\xi \sim \mathcal{N}(0, I)$, the deployed-map gain and its one-sided hinge are

$$g_\xi(u,s) = \frac{\|D_u\Phi_\theta(u;s)\xi\|_2}{\max\{\|\xi\|_2, \varepsilon_\ell\}}, \tag{25}$$

$$\mathcal{L}_{\text{JVP}} = \mathbb{E}_{u,s,\xi}[g_\xi(u,s) - 1]^2_+. \tag{26}$$

This is the Jacobian of one deployed midpoint step, rather than the Jacobian of the full H-step rollout. The reported rollout loss is

$$\mathcal{L}_{\text{roll}} = \mathcal{L}_{\text{state}} + 0.1\mathcal{L}_{\text{gradient}} + 0.1\mathcal{L}_{\text{functional}} + 0.1\mathcal{L}_{\text{canonical}} + 0.1\mathcal{L}_{\text{JVP}}.$$

Training uses $H = 1 \to 4 \to 16 \to 64$ in four approximately equal epoch blocks. Pointwise and rollout minibatches are optimized in separate updates: each epoch contains 400 pointwise updates and 20 rollout updates.

Two-dimensional Burgers specialization. The 2D experiment uses physical states and a chord-sampled velocity loss. Let $u_\tau = (1-\tau)u_k + \tau u_{k+1}$, $v_k = (u_{k+1} - u_k)/\Delta t$, and $\tau \sim \mathcal{U}(0,1)$. Here $s_k = k/(N_t - 1)$, $\Delta s = 1/(N_t - 1)$, and the midpoint map advances the physical state by Δt while advancing its normalized clock by Δs. If c_f is the mean per-channel train-set state standard deviation, σ_u is the per-channel train-set state standard deviation, and

$$K(u) = \frac{1}{2}\,\mathrm{mean}_{x,y}\|u(x,y)\|_2^2, \qquad c_K = \mathrm{RMS}_{\text{train}}[K(u_{k+1}) - K(u_k)] + 10^{-8},$$

then the 2D pointwise loss is

$$\begin{aligned}\mathcal{L}^{\text{2D}}_{\text{point}} = {} & \mathrm{mean}\left[\left(\frac{f_\theta(u_\tau, s_k + \tau\Delta s) - v_k}{c_f}\right)^2\right] + \mathrm{mean}\left[\left(\frac{\Phi_\theta(u_k) - u_{k+1}}{\sigma_u}\right)^2\right] \\ & + 0.1\,\mathrm{mean}\left[\left(\frac{K(\Phi_\theta(u_k)) - K(u_{k+1})}{c_K}\right)^2\right].\end{aligned} \tag{27}$$

For $e_h = (\hat{u}_h - u_h)/\sigma_u$ and periodic one-grid differences δ_x, δ_y, its rollout loss is

$$\mathcal{L}^{\text{2D}}_{\text{roll}} = \mathrm{mean}_h\left[\mathrm{mean}(e_h^2) + 0.1\,\mathrm{mean}\big((\delta_x e_h)^2 + (\delta_y e_h)^2\big) + 0.1\left(\frac{K(\hat{u}_h) - K(u_h)}{c_K}\right)^2\right].$$

Writing the last terms as kinetic-energy differences is equivalent to the implemented increment form because the predicted and target increments use the same source state. The 2D objective has no JVP term.

C.3 RATE-SUPERVISED EXTENSION

For dissipative systems, define the observed physical functional increment

$$\Delta S_k^{\mathrm{data}} = S(\mathcal{D}(u_{k+1})) - S(\mathcal{D}(u_k)).$$

The local generator estimate is

$$\widehat{\Delta S}_k^{\mathrm{rate}} = \nabla S(\mathcal{D}(u_k))^\top \big[\sigma h f_\theta(u_k; s_k, 0)\big], \qquad h = 1,$$

while the realized finite-step change is

$$\widehat{\Delta S}_k^{\mathrm{step}} = S(\mathcal{D}(\Phi_\theta(u_k; s_k))) - S(\mathcal{D}(u_k)).$$

The train-only normalizer is

$$c_S = \max\left\{ \sqrt{\mathrm{mean}_{\mathrm{train}}[(\Delta S_k^{\mathrm{data}})^2]}, 10^{-6}\sqrt{\mathrm{mean}_{\mathrm{train}}[S(\mathcal{D}(u))^2]}, 10^{-12} \right\},$$

where the second mean pools source and endpoint functional values. Define the unit-threshold Huber function

$$\ell_{\mathrm{H}}(r) = \begin{cases} \frac{1}{2}r^2, & |r| \le 1, \\ |r| - \frac{1}{2}, & |r| > 1. \end{cases}$$

The two losses are

$$\mathcal{L}_{\mathrm{rate}} = \mathrm{mean}_k\, \ell_{\mathrm{H}}\left(\frac{\widehat{\Delta S}_k^{\mathrm{rate}} - \Delta S_k^{\mathrm{data}}}{c_S} \right), \tag{28}$$

$$\mathcal{L}_{\mathrm{step}} = \mathrm{mean}_k\, \ell_{\mathrm{H}}\left(\frac{\widehat{\Delta S}_k^{\mathrm{step}} - \Delta S_k^{\mathrm{data}}}{c_S} \right). \tag{29}$$

The extended pointwise objective is

$$\mathcal{L}_{\mathrm{point}}^{\mathrm{ext}} = \mathcal{L}_{\mathrm{point}} + 0.1\mathcal{L}_{\mathrm{rate}} + 0.1\mathcal{L}_{\mathrm{step}},$$

while the rollout objective is unchanged.

For systems with a declared mass invariant, let $g_M = \nabla M$ and $P_M = I - g_M g_M^\top / \|g_M\|_2^2$. The slack branch is additionally projected onto the mass-preserving tangent space:

$$f_{\mathrm{slack}} \leftarrow P_M f_{\mathrm{slack}}.$$

All loss weights and equation-specific choices of E, S, and auxiliary invariants are reported with the experimental configuration.

D PHYSICAL FUNCTIONALS AND EQUATION-SPECIFIC CONTRACTS

We distinguish quantities that are declared invariant from quantities whose evolution is treated as dissipative or reference-dissipative. The equation-specific structural and evaluation contracts are summarized in Table 11.

We use the discrete mass and squared-ℓ_2 functionals

$$M(u) = \Delta x \sum_i u_i, \qquad Q_2(u) = \Delta x \sum_i u_i^2. \tag{30}$$

For 2D Burgers, the corresponding definitions replace $\Delta x \sum_i$ by $\Delta x \Delta y \sum_{i,j}$; the constant spatial factor cancels in the reported relative errors.

The canonical evaluation functional used in the 1D rollout loss is the periodic discrete gradient energy

$$G_+(u) = \frac{\Delta x}{2} \sum_i (D_+ u_i)^2, \tag{31}$$

Table 11: **Equation-specific structural and evaluation functionals.** For Advection, Q_2 is treated as a *reference-dissipative* diagnostic: it measures agreement with the numerical reference trajectory and is not interpreted as thermodynamic dissipation of the continuum equation. The Q_{can} column gives the PDE-specific functional used by $\mathcal{L}_{\mathrm{canonical}}$, or its counterpart in the separately defined 2D objective.

PDE	Structured E	Structured S	Evaluated invariants	Evolving functional	Q_{can}
Advection	$M(u)$	$-\frac{1}{2}Q_2(u)$	$M(u)$	$Q_2(u)$	$G_+(u)$
Heat	$M(u)$	$-\frac{1}{2}Q_2(u)$	$M(u)$	$Q_2(u)$	$G_+(u)$
Burgers	$M(u)$	$-\frac{1}{2}Q_2(u)$	$M(u)$	$Q_2(u)$	$G_+(u)$
2D Burgers	$M(u)$	$-\frac{1}{2}Q_2(u)$	$M(u)$	$Q_2(u)$	$K(u)$
Allen-Cahn	—	$-F_\epsilon(u)$	—	$F_\epsilon(u)$	$F_\epsilon(u)$
KdV	$H_\beta(u)$	—	$M(u)$, $Q_2(u)$, $H_\beta(u)$	—	—

for Advection, Heat, and Burgers. Allen-Cahn instead uses its free energy, and KdV has no canonical evaluation functional. The separate 2D Burgers objective uses the kinetic energy $K(u)$ defined above.

For Allen-Cahn, the discrete free energy is

$$F_\epsilon(u) = \Delta x \sum_i \left[\frac{\epsilon^2}{2}(D_+ u_i)^2 + \frac{(u_i^2 - 1)^2}{4} \right], \tag{32}$$

where D_+ denotes the discrete forward spatial derivative.

For KdV, the discrete Hamiltonian is

$$H_\beta(u) = \Delta x \sum_i \left[\frac{\beta}{2}(D_{\mathrm{spec}} u_i)^2 - u_i^3 \right], \tag{33}$$

where D_{spec} denotes the spectral spatial derivative.

E Evaluation Metric Definitions

Let $\hat{u}_{b,t}$ and $u_{b,t}$ denote the predicted and reference states for trajectory b at forecast step t.

E.1 Trajectory Fidelity

Following normalised rollout-error metrics commonly used in PDE surrogate evaluation (Takamoto et al., 2022; Koehler et al., 2024), cumulative normalized root-mean-square error over the first H forecast steps:

$$\mathrm{nRMSE}\,@H = \sqrt{\mathrm{mean}_{b,\, t \le H} \left[\frac{\|\hat{u}_{b,t} - u_{b,t}\|_2^2}{\max\left(\|u_{b,t}\|_2^2, \epsilon_{\mathrm{state}}\right)} \right]}. \tag{34}$$

Here $\epsilon_{\mathrm{state}}$ is a scale-aware numerical floor. Thus, nRMSE@H is cumulative over all forecasts up to H, rather than the pointwise error at step H.

The accuracy horizon is

$$t_{\mathrm{acc}}(\tau) = \min\left\{ t : e_{\mathrm{state}}(t) > \tau \right\}, \tag{35}$$

where $e_{\mathrm{state}}(t)$ is the trajectory-mean relative spatial ℓ_2 error. If the threshold is never crossed, we set $t_{\mathrm{acc}} = T$, with $T = 2000$ for MTS-1D and $T = 499$ for 2D Burgers.

For augmentation experiments, we additionally report t_{nan}, the first forecast containing either a non-finite prediction or a violation of the prescribed amplitude bound. Consequently, a rollout may remain numerically finite long after it has ceased to be accurate.

E.2 Invariant Preservation

For a declared invariant Q, we measure self-drift relative to its value at the initial rollout state:

$$D_Q(t) = \frac{|Q(\hat{u}_t) - Q(u_0)|}{|Q(u_0)| + \epsilon_{Q,0}}. \tag{36}$$

The reported conservation statistic is the maximum over trajectories and rollout time,

$$D_Q^{\max} = \max_{b,t} D_{Q,b}(t). \tag{37}$$

This is a self-conservation metric: it measures preservation of the initial invariant value rather than agreement with the reference functional trajectory.

E.3 Functional Trajectory Fidelity

For a physical functional Q, agreement with the reference trajectory is measured using

$$e_Q(t) = \frac{|Q(\hat{u}_t) - Q(u_t)|}{\max\left(\mathrm{scale}_Q(u_t), \epsilon_Q\right)}. \tag{38}$$

For positive quantities such as Q_2 and the Allen-Cahn free energy, scale_Q reduces to the usual target magnitude. For signed quantities such as mass and the KdV Hamiltonian, we instead use a positive physical scale to avoid artificial inflation near cancellation. Unless stated otherwise, we report the mean of equation 38 over trajectories and rollout time.

E.4 Functional-Increment Fidelity

For non-invariant quantities, define

$$\Delta Q_{b,t} = Q(u_{b,t+1}) - Q(u_{b,t}), \qquad \Delta \hat{Q}_{b,t} = Q(\hat{u}_{b,t+1}) - Q(\hat{u}_{b,t}). \tag{39}$$

We measure agreement in the rate of functional evolution using

$$\mathrm{nRMSE}_{\Delta Q} = \frac{\sqrt{\mathrm{mean}_{b,t}\left(\Delta \hat{Q}_{b,t} - \Delta Q_{b,t}\right)^2}}{\sqrt{\mathrm{mean}_{b,t}\left(\Delta Q_{b,t}\right)^2}}. \tag{40}$$

The relative value is reported only when the RMS target increment exceeds a scale-aware numerical floor. This metric is used only for quantities that evolve along the reference trajectory, not for declared invariants.

For Heat and both 1D and 2D Burgers, Q_2 is treated as a dissipative quantity, while Allen-Cahn is evaluated using its free energy. For Advection, Q_2 is reported as a *reference-dissipative* quantity: the metric measures whether the learned dynamics reproduce dissipation present in the numerical reference data, without asserting dissipation of the continuum equation. KdV is treated as conservative, so no functional-increment metric is used in the main comparison.

F Geometric interpretation of physical admissibility and trajectory accuracy

For a conserved scalar functional $Q(u)$, physically admissible instantaneous dynamics satisfy $\nabla Q(u)^\top f(u) = 0$. Thus, the vector field at u must lie in the tangent space $T_u\mathcal{M}_Q = \ker(\nabla Q(u)^\top)$ of the corresponding invariant level set. This constraint does not uniquely determine the correct dynamics, since many different vector fields may lie in the same tangent space. For nonlinear invariants, there is an additional finite-step effect. Even if two observations satisfy $Q(u_{k+1}) = Q(u_k)$, their secant $v_{\mathrm{ref}} = (u_{k+1} - u_k)/\Delta t$ need not be exactly tangent to the invariant manifold at u_k. A Taylor expansion gives

$$\nabla Q(u_k)^\top (u_{k+1} - u_k) = -\frac{1}{2}(u_{k+1} - u_k)^\top \nabla^2 Q(u_k)(u_{k+1} - u_k) + \mathcal{O}\left(\|u_{k+1} - u_k\|^3\right). \tag{41}$$

Table 12: **Mass-residual audit for unrestricted Trajectory Flow.** For the 1D systems, discrete mass is $M(u) = \Delta x \sum_i u^i$. We define the reference and learned instantaneous mass derivatives as $\dot{M}_{\mathrm{ref}} = \Delta x \sum_i v_{\mathrm{ref},i}$ and $\dot{M}_\theta = \Delta x \sum_i f_{\theta,i}$. The first two columns show that the reference interpolation velocities are nearly mass-preserving, whereas the learned unrestricted vector field contains a larger mass-changing component even on reference interpolation states. For each rollout, the learned instantaneous mass derivative is $\dot{M}_\theta(t) = \Delta x \sum_i f_{\theta,i}(\hat{u}(t), t)$, and the corresponding accumulated mass change is $\Delta M_{\mathrm{res}}(T) = \int_0^T \Delta x \sum_i f_{\theta,i}(\hat{u}(t), t)\, dt$. We compare this with the mass drift measured directly from the predicted trajectory, $\Delta M_{\mathrm{obs}}(T) = M(\hat{u}(T)) - M(\hat{u}(0))$. The integral is evaluated using the same explicit-midpoint solver as the rollout. Correlation and zero-intercept slope near one between ΔM_{res} and ΔM_{obs} indicate that accumulation of the learned mass-changing component accounts for the observed rollout drift.

PDE	$\mathbb{E}\lvert\dot{M}_{\mathrm{ref}}\rvert$	$\mathbb{E}\lvert\dot{M}_\theta\rvert$	Ratio	Correlation	Slope
Advection	3.694×10^{-6}	1.467×10^{-3}	397	1.000	1.000
Heat	2.444×10^{-6}	2.421×10^{-3}	990	1.000	1.00002
Burgers	2.839×10^{-6}	1.632×10^{-3}	575	1.000	0.99994
KdV	2.849×10^{-6}	3.539×10^{-2}	12400	1.000	1.000

Therefore, on a curved invariant manifold, fitting a finite observed transition and enforcing exact local tangency are related but not identical objectives.

This curvature effect does not arise for a linear invariant such as mass. Here,u^i denotes the i-th spatial component of the observation u, whereas subscripts such as those in u_k and u_{k+1} index observation time. In our 1D setting, $M(u) = \Delta x \sum_i u^i$, so if two reference states have the same mass then their secant also satisfies $\Delta x \sum_i v_{\mathrm{ref},i} = 0$. However, an unrestricted learned generator only approximates this target. If $f_\theta(u) = v_{\mathrm{ref}}(u) + \varepsilon_\theta(u)$, then the learned instantaneous mass derivative is $\dot{M}_\theta = \Delta x \sum_i f_{\theta,i} = \Delta x \sum_i \varepsilon_{\theta,i}$, which need not vanish exactly. A small coherent error while predicting the state can be almost negligible relative to $\|u\|$. However, since the mass functional sums up same-sign contributions over the spatial domain, even a small error can remain visible. If such an error is systematic, it can accumulate over repeated integration steps and produce measurable mass drift. Table 12 directly audits this mechanism. The ground-truth interpolation velocities are mass-preserving up to the numerical precision of the stored data, whereas the learned velocity from the unrestricted Trajectory Flow model shows a substantially larger mass-changing component.

For a simple intuition, consider a toy prediction of the form $\hat{u} = u + c\mathbf{1}$, where the same small offset c is introduced at every spatial location. The relative state-space error contributed by this offset is $\|\hat{u} - u\|_2 / \|u\|_2 = |c|/\mathrm{RMS}(u)$, and can therefore remain small when $|c|$ is small relative to the typical magnitude of the state. In contrast, its mass error is $M(\hat{u}) - M(u) = N_x \Delta x\, c$, since the contributions from all spatial locations add with the same sign. Thus, a small coherent component of the state error can be weak in the overall trajectory metric while remaining directly visible as mass drift. The quantity $\Delta x \sum_i f_{\theta,i}$ reported in Table 12 measures the instantaneous analogue of precisely this mass-changing component in the learned vector field.

The structured model addresses the complementary problem of enforcing admissibility. For the 1D experiments, the discretized state has $N_x = 256$ degrees of freedom, so a single linear invariant such as mass still leaves a 255-dimensional mass-preserving subspace. Conservation alone therefore does not identify the correct dynamics. GENERIC Flow imposes a substantially more restricted hypothesis class through its reversible-dissipative decomposition, degeneracy conditions, and projected low-rank factors. The model must therefore identify the particular data-consistent vector field while remaining inside this structured admissible class. This additional restriction can make trajectory fitting more difficult than for an unrestricted generator, motivating the use of both local velocity supervision and finite-step endpoint supervision.

Together, these observations provide a geometric interpretation of the empirical separation between trajectory and physical fidelity. Physical admissibility restricts the directions in which the learned system may evolve but does not uniquely determine the correct trajectory. For nonlinear invariants, finite-step observations can additionally differ from exact local tangency; for linear invariants

Table 13: **Invariant-violating rollout updates for unrestricted and structured Flow controls.** Values report the percentage of deployed transitions exceeding the $\tau = 10^{-6}$ conservation deadband. Neither rollout-aware training nor additional soft physics losses reproduce the preservation obtained from hard structure.

PDE	Invariant	Normal Flow	Rollout-Aware Flow	Rollout-Aware + Physics Losses	GENERIC Flow
Advection	Mass	24.658%	80.822%	77.158%	0.000%
Heat	Mass	36.165%	45.852%	45.859%	0.000%
Burgers	Mass	29.440%	53.958%	59.454%	0.000%
KdV	Mass	99.751%	79.581%	89.848%	0.000%
KdV	Q_2	93.115%	75.000%	86.006%	0.000%
KdV	Hamiltonian	99.743%	95.740%	97.283%	0.127%

such as mass, approximate regression can introduce small systematic mass-changing components that accumulate over long rollouts. Consequently, improved invariant preservation need not imply lower state-space error, while an accurate unrestricted trajectory model can still exhibit measurable invariant drift.

G Rollout-Aware Flow Control for Invariant Preservation

GENERIC Flow is trained through the numerical solver, analogously to multistep rollout-aware training, whereas standard Trajectory Flow does not undergo rollout-aware training. One may therefore question whether the invariant preservation shown by GENERIC Flow is due to its rollout-aware training. We train a stronger variant of Trajectory Flow with rollout-aware training to test whether it can explain the strong invariance preservation capabilities of the GENERIC Flow. We compare the standard Trajectory Flow with CNN backbone, the corresponding rollout-aware model, the same rollout-aware model augmented with soft functional and canonical losses, and GENERIC Flow on their common equation support. The rollout-aware Trajectory Flow is trained for 50 epochs (10,000 updates) using a rollout-horizon curriculum $1 \to 4 \to 16 \to 64$, with checkpoint selection based on a fixed 64-step validation rollout. The physics-aware control retains this objective and adds $0.1\mathcal{L}_{\text{functional}} + 0.1\mathcal{L}_{\text{canonical}}$. We use a deadband of $\tau = 10^{-6}$ for the headline comparison.

The common support contains six Advection, four Heat, four Burgers, and three KdV equations; Allen-Cahn is excluded because it has no declared invariant. Both rollout-aware Flow controls use one deterministic run per equation. Results are then averaged equally across equations within each PDE.

Table 13 shows that rollout-aware training does not prevent conservation violations. Adding soft physics losses slightly reduces the Advection violation rate, leaves Heat effectively unchanged, and worsens Burgers and all three KdV invariants relative to rollout-aware training alone. Across the six rows, $45.859\% - 97.283\%$ of the physics-aware control's updates violate the deadband, compared with $0\% - 0.127\%$ for GENERIC Flow.

These controls show that neither solver-in-the-loop training nor direct supervision of physical functionals is sufficient to explain the conservation behavior of GENERIC Flow. Together with the evidence in Appendix H, this indicates that enforcing the hard geometry of GENERIC is the most important factor in preserving the invariants.

H Targeted Structural Ablation of GENERIC Flow

The main experiments show that GENERIC Flow substantially improves invariant preservation relative to unrestricted Trajectory Flow model. We perform a targeted ablation to test whether this improvement is specifically attributable to the state-dependent null-space projections used inside the structured core.

Table 14: **Effect of relaxing the factor null-space projections.** State accuracy is reported using cumulative nRMSE@100 and $t_{\mathrm{acc}}@30\%$. Physical fidelity is measured by maximum invariant self-drift. Lower is better for nRMSE and drift; higher is better for t_{acc}.

PDE	Method	nRMSE@100 ↓	t_{acc} ↑	Mass drift ↓	Q_2 drift ↓	Hamiltonian drift ↓
Burgers	GENERIC Flow	0.001101	2000.0	2.356×10^{-6}	–	–
Burgers	Relaxed projections	0.001154	2000.0	0.532835	–	–
KdV	GENERIC Flow	0.004443	1262.3	6.641×10^{-5}	3.329×10^{-6}	0.015276
KdV	Relaxed projections	0.003081	1641.7	32.5619	0.829253	0.009747

H.1 Relaxing the Factor Null-Space Projections

In the GENERIC architecture, the structured core constructs reversible and dissipative actions from learned low-rank factors. In the full model, these factors are projected into state-dependent null spaces before forming the skew-symmetric and positive-semidefinite operators. The ablation removes only these factor projections while retaining the remainder of the architecture and training pipeline.

Concretely, both models use the same neural operator and parameter count, the same skew construction

$$L = \frac{UV^\top - VU^\top}{\sqrt{r}},$$

and the same positive-semidefinite construction

$$M = \frac{BB^\top}{\sqrt{r}}.$$

The full GENERIC Flow first projects the learned factors into the required state-dependent null spaces, whereas the relaxed variant feeds the learned factors directly into the same skew and PSD constructions.

The intervention therefore preserves the intrinsic identities associated with skew symmetry and positive-semidefiniteness, but removes the exact cross-degeneracy and auxiliary-invariant orthogonality imposed by the factor projections. The projected tangent branch, calibrated slack branch, soft structural losses, optimizer, data, seeds, model selection procedure, and explicit-midpoint deployment are otherwise matched.

Because both variants retain the calibrated slack component and are deployed with an explicit numerical integrator, this experiment should not be interpreted as comparing an exactly invariant numerical scheme with an unstructured model. Rather, it isolates the contribution of the exact null-space geometry inside the structured core.

We evaluate Burgers at four viscosity settings and KdV at three dispersion settings using two matched seeds per configuration. The resulting 14 paired coefficient-seed units use the same training and test protocol as the main benchmark (Table 14).

Relaxing the factor projections causes a selective collapse of the invariants directly protected by the removed geometry. Burgers mass drift increases by 2.26×10^5, while KdV mass and squared-L_2 drift increase by 4.90×10^5 and 2.49×10^5, respectively. The direction of this degradation is consistent across all matched runs: 8/8 Burgers coefficient-seed pairs for mass and 6/6 KdV pairs for both mass and squared-L_2.

Importantly, this loss of conservation is not explained by a general deterioration in state prediction. For Burgers, nRMSE@100 changes by only 4.9%, with both variants remaining below the 30% error threshold for the full 2,000-step rollout. For KdV, relaxing the projections instead *improves* nRMSE@100 by 30.6% and increases the mean accuracy horizon by 30.1%, even as the mass and squared-L_2 invariants deteriorate by approximately five orders of magnitude.

This provides a direct example of the distinction between trajectory fidelity and physical fidelity. Removing hard geometric restrictions enlarges the set of admissible learned dynamics and can improve state-space fitting, while simultaneously permitting directions that violate the protected invariants.

Table 15: **Target-functional errors under the structural ablation.** The relaxed model loses the quantities directly protected by the removed null-space geometry, while the retained Hamiltonian structure does not degrade.

PDE	Functional	GENERIC Flow	Relaxed projections
Burgers	Mass	8.594×10^{-8}	0.006017
Burgers	Squared L_2	0.012096	0.011027
KdV	Mass	8.180×10^{-7}	0.067213
KdV	Squared L_2	9.402×10^{-7}	0.072605
KdV	Hamiltonian	1.156×10^{-4}	7.783×10^{-5}

Table 16: Endpoint-loss ablation on Burgers and KdV. Results are averaged over matched seeds 0 and 1 and all four Burgers / three KdV coefficient settings. Lower nRMSE and higher t_{acc} are better.

PDE	Variant	nRMSE@100 ($\times 10^{-3}$) $\downarrow$	t_{acc}@10% $\uparrow$
Burgers	With endpoint loss	1.1008	1825.8
	Without endpoint loss	2.0593	1699.4
KdV	With endpoint loss	4.4432	750.2
	Without endpoint loss	5.2415	481.3

The KdV Hamiltonian acts as a useful negative control. Unlike mass and squared-L_2, Hamiltonian drift decreases by 36.2% in the relaxed model. This is consistent with the intervention: the skew-symmetric reversible core is retained, and therefore the intrinsic Hamiltonian tangency mechanism is not removed together with the auxiliary null-space projections. The ablation consequently produces a selective rather than universal degradation of physical functionals.

To reduce sensitivity to normalization by near-zero signed invariants, we also compare the predicted and reference functional trajectories directly. The same selective failure appears under this alternative metric (Table 15).

The target-functional metric reproduces the same pattern. Burgers mass and KdV mass and squared-L_2 errors increase by roughly four to five orders of magnitude after removing the factor projections, whereas KdV Hamiltonian error slightly improves. This agreement across invariant self-drift and target-functional error supports the narrow conclusion that the null-space projections are the mechanism responsible for the strong preservation of the quantities they directly enforce.

H.2 Endpoint-loss ablation.

GENERIC Flow has an explicit endpoint loss in addition to the local velocity loss term. The main reason is that optimisation with structural constraints might restrict the space of admissible vector fields, making it difficult to fit the trajectory accurately. Our ablation verifies this, as trajectory accuracy is reduced without the endpoint-fitting term (Table 16). For KdV, errors in the higher-order invariants Q_2 and H also increase, indicating that endpoint supervision improves some aspects of quantitative physical fidelity (Table 17).

I Effect of Temporal Observation Sparsity

In the main paper, the experiments use a temporal spacing of $\Delta t = 10^{-3}$. We wanted to see whether the trajectory accuracy gap between Trajectory Flow and GENERIC Flow narrows as Δt increases and the data becomes sparser. The major objective is to verify whether an imposed physical structure provides a better inductive bias for modelling PDE solutions.

Starting from the same reference trajectories, we retain every 1st, 5th, or 10th observation, corresponding to stored intervals $\Delta t \in \{10^{-3}, 5 \times 10^{-3}, 10^{-2}\}$. To make the comparison independent of the number of stored transitions, all models are evaluated through the same physical time $t = 0.5$.

Table 17: Physical-fidelity metrics for the endpoint-loss ablation. Lower values are better.

PDE	Variant	Max mass drift	Max Q_2 drift	Max H drift	Functional error
Burgers	With endpoint loss	2.3563×10^{-6}	–	–	0.01210/0.02038
	Without endpoint loss	2.6100×10^{-6}	–	–	0.01808/0.03075
KdV	With endpoint loss	6.6409×10^{-5}	3.3289×10^{-6}	0.01528	–
	Without endpoint loss	6.4801×10^{-5}	2.6839×10^{-5}	0.09438	–

For each PDE, we report the signed trajectory-accuracy gap

$$g(\Delta t) = \mathrm{nRMSE}_{\mathrm{GENERIC}}(\Delta t) - \mathrm{nRMSE}_{\mathrm{Flow}}(\Delta t), \tag{42}$$

where $g > 0$ indicates that the unrestricted Trajectory Flow model is more accurate.

Our findings show that the trajectory gap narrows for four of the five PDEs under the sparse setting (Figure 5). From the native $1\times$ spacing to the $10\times$ spacing, the gap closes by 26% for Advection, 42% for Burgers, 82% for Allen-Cahn, and 83% for KdV. Heat is the only exception where the trajectory gap increases.

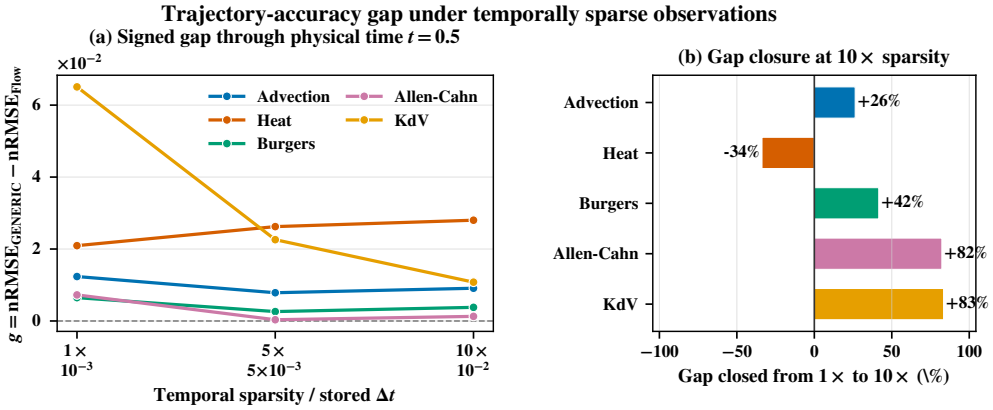


Figure 5: **Trajectory-accuracy gap under temporally sparse observations.** Left: signed nRMSE difference between GENERIC Flow and unrestricted Trajectory Flow model, evaluated through the common physical horizon $t = 0.5$. Positive values indicate lower trajectory error for Trajectory Flow model. Right: relative reduction in this gap from $\Delta t = 10^{-3}$ to 10^{-2}; positive values indicate that the GENERIC-Flow accuracy gap closes as temporal observations become sparser.

Hence, we can claim that the dominant trend is that the trajectory accuracy gap between Trajectory Flow and GENERIC Flow reduces in sparse settings. This behavior is consistent with our original hypothesis that hard structure can act as a favorable inductive bias for sparse data. When the data is dense, the unrestricted Trajectory Flow model can fit the data well using local information while retaining greater freedom in the vector field. As the interval between observations increases, the local information required to fit the data becomes weaker. This experiment shows that the accuracy-conservation tradeoff observed at the native spacing can become substantially less pronounced.

J Per-coefficient fidelity tradeoffs

In our experiments, we have trained separate models for different PDE parameters, as shown in Appendix A.2. The numbers reported in Table 3 are aggregated for each PDE. We additionally show that the tradeoff between physical fidelity and trajectory fidelity holds for each equation-coefficient pair as well. This is to rule out the possibility that the observed phenomenon is an artefact of aggregation over PDE coefficients. Figure 7 compares trajectory error with invariant drift for each PDE at matched coefficients. Figure 6 does the same for dissipative-rate error and invariant drift.

From the figures, we can see that our findings hold for each individual coefficient of the studied PDEs. Together, these matched-coefficient comparisons reinforce the main conclusion that trajectory fidelity, invariant preservation, and dissipative-rate fidelity represent distinct objectives rather than different measurements of a single notion of model quality.

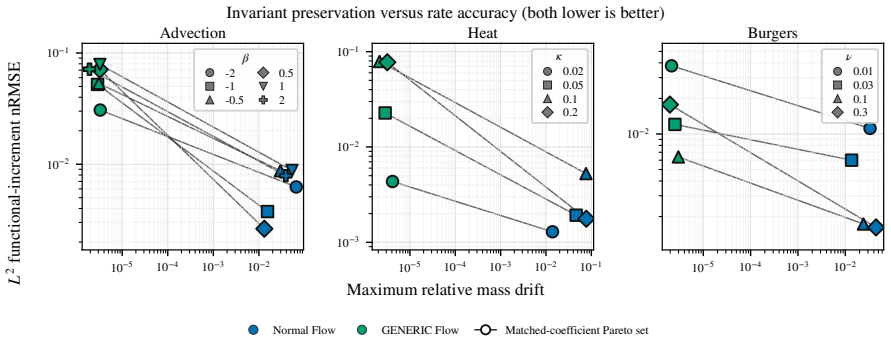


Figure 6: **Matched-coefficient comparison of invariant preservation and dissipative-rate fidelity.** Each pair connects unrestricted Trajectory Flow model and GENERIC Flow trained and evaluated at the same PDE coefficient. Both axes are lower-is-better. GENERIC Flow consistently improves mass conservation, while unrestricted Trajectory Flow model generally attains lower functional-increment error.

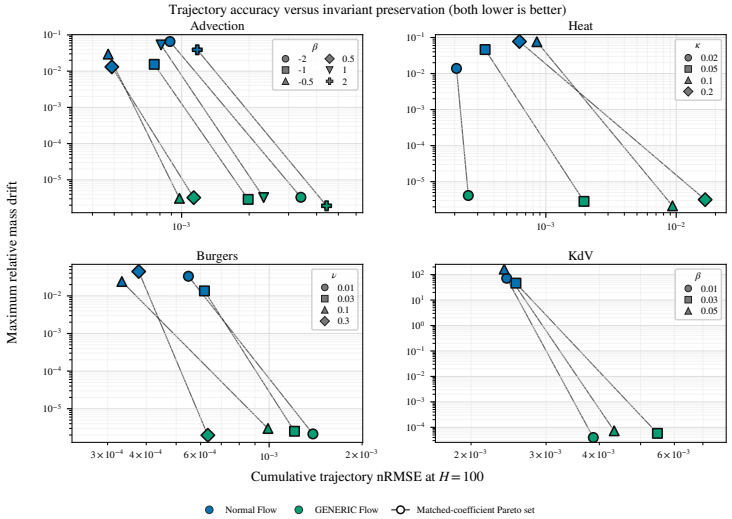


Figure 7: **Matched-coefficient comparison of trajectory fidelity and invariant preservation.** Each pair connects the unrestricted Trajectory Flow model and GENERIC Flow at the same PDE coefficient. Both axes are lower-is-better. The conservation advantage of GENERIC Flow persists across coefficients, while trajectory accuracy need not improve under the stronger structural constraint.