Physics-Informed AI for One-Dimensional Turbulence (ODT) Simulations

What should a machine-learned surrogate of a stochastic simulation learn — a realization, or the statistics?

Babak Goshayeshi, Ph.D. | code & data: `github.com/bgoshayeshi/AI-ODT` | SC26 — The International Conference for High Performance Computing, Networking, Storage, and Analysis

code + data

Motivation

One-Dimensional Turbulence (ODT) models a turbulent flow on a single line: molecular diffusion evolves the velocity profiles continuously, and stochastic **eddy events** — triplet maps applied at random times, places, and sizes — stand in for turbulent advection.

ODT is cheap enough to run by the hundreds, making it an ideal sandbox for a central question of surrogate modeling: **should a neural surrogate of a stochastic simulation learn a realization, or the statistics?** Three scenarios of increasing ambition answer it.

The ODT model

The temporal planar jet solves u(y,t), v(y,t) on a line across the jet. Between eddy events, fields evolve by **viscous diffusion only**:

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial y^2}, \quad \frac{\partial v}{\partial t} = \nu \frac{\partial^2 v}{\partial y^2}, \quad \nu = \frac{\mu_0}{\rho_0}, \quad p_0 = \rho_0 R_s T_0$$

with a smooth top-hat initial jet and ambient Dirichlet boundaries:

$$u(y,0) = U_a + \frac{U_j - U_a}{2}\left[\tanh\left(\frac{y + D/2}{\delta}\right) - \tanh\left(\frac{y - D/2}{\delta}\right)\right], \quad v(y,0) = 0$$

An eddy event (y_0, ℓ) applies the measure-preserving **triplet map** M(y) plus a kernel K(y) = y – M(y) that redistributes kinetic energy, conserving momentum and energy:

$$u^+(y) = u^-(M(y)) + c_u K(y), \qquad v^+(y) = v^-(M(y)) + c_v K(y)$$

Candidate eddies are accepted by rejection sampling from the rate

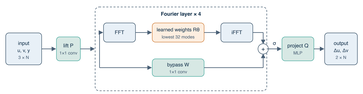
$$\lambda(y_0, \ell; t) = \frac{C\nu}{\ell^4}\sqrt{\left(\frac{q\kappa\,\ell}{\nu}\right)^2 - Z}$$

Explicit-Euler diffusion on a 1001-point uniform grid; large-eddy suppression filters unphysical events.

Fourier Neural Operator

An FNO learns maps between **functions**: each layer FFTs the whole field, multiplies the lowest k modes by learned complex weights (a global convolution), inverse-FFTs, and adds a pointwise bypass:

$$z_{l+1} = \sigma\left(\mathcal{F}^{-1}[R_\theta \cdot \mathcal{F}(z_l)|_{\le k}] + W z_l\right)$$

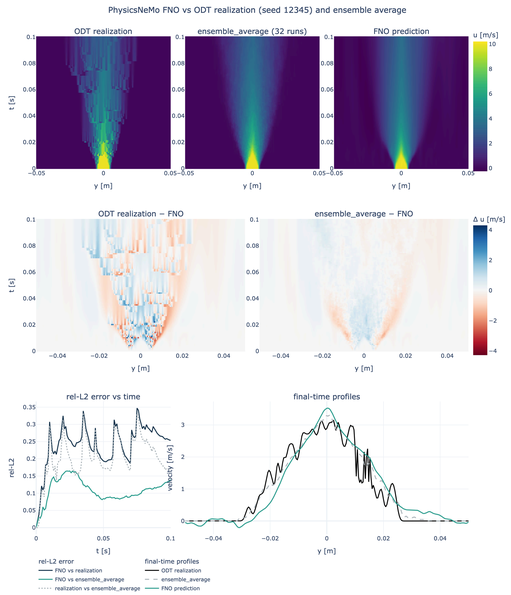


ODT fields are smooth apart from eddy wrinkles, so their energy sits in the low modes — a 1D FNO (64 latent channels, 32 modes, 4 layers, **1,071,778 parameters**, NVIDIA PhysicsNeMo) advances the full 1001-point state in one shot. The same architecture is used in all three scenarios; inputs (u,v,y), output (Δu, Δv).

SCENARIO 1 FNO on stochastic realizations

32 ODT runs (random seeds) → one-step pairs (u,v) → increment → FNO → autoregressive rollout from t = 0 only

The eddy sequence is unpredictable from the flow state, so the MSE-optimal deterministic prediction is the **conditional mean** of the next increment: the network learns the expected (ensemble-mean) dynamics **even though it never sees an averaged field**. Against the held-out realization the rollout carries the irreducible eddy noise; against the ensemble average the error roughly halves and the speckled eddy signature vanishes.



Streamwise velocity u(y,t). Top: held-out ODT realization (seed 12345), 32-run ensemble average, FNO rollout. Middle: difference maps — speckles vs the realization, nearly clean vs the mean. Bottom: rel-L2 error in time and final-time profiles.

rollout rel-L2(u)	t = 25%	t = 50%	t = final
vs single realization	0.254	0.194	0.253
vs ensemble average	**0.165**	**0.084**	**0.140**

23.5s training (20 epochs, Apple-silicon GPU)

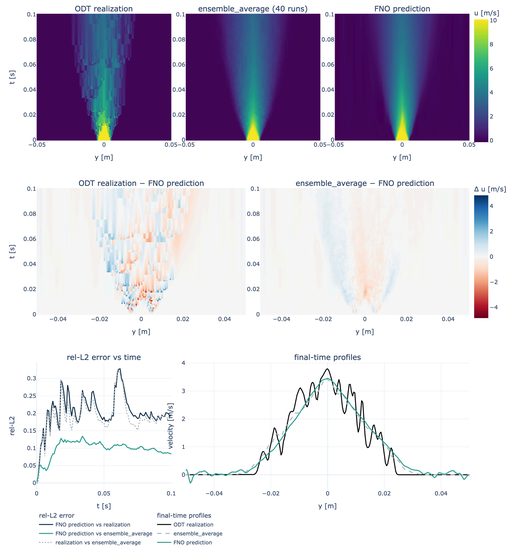
0.44s 100-step rollout (CPU)

268× vs the 32-run ensemble it stands in for

SCENARIO 2 FNO on ensemble averages

9 (U_j, D) combos × 40 seeds = 360 runs → 9 ensemble-mean trajectories → FNO (8 train combos) → rollout at any (U_j, D)

Scenario 1 showed the FNO converges to the mean anyway — so train on **ensemble averages directly** and make the problem harder: a 3×3 grid of jet velocities (U_j = 6, 10, 14 m/s) and widths (D = 6, 10, 14 mm). The combo **U_j = 10 m/s, D = 10 mm is held out of training entirely** — rollouts there test whether the operator **interpolates the mean dynamics across jet parameters** rather than memorizing trajectories.



Held-out combo (U_j = 10 m/s, D = 10 mm): the FNO prediction matches the 40-run ensemble average it never saw during training; error vs the mean stays near 0.1 throughout the rollout.

rollout rel-L2(u) vs ens. avg.	t = 25%	t = 50%	t = final
held-out combo (10 m/s, 10 mm)	**0.117**	**0.102**	**0.083**
mean over 8 training combos	0.111	0.106	0.139

11.3s training (40 epochs, Apple-silicon GPU)

0.42s 100-step rollout (CPU)

356× vs the 40-run ensemble it stands in for

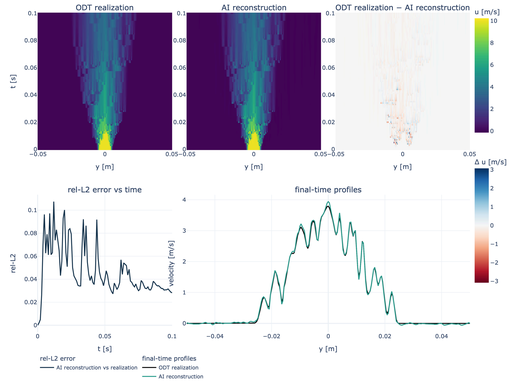
SCENARIO 3 Physics-informed advance + explicit eddies

state u, v at t → physics-informed FNO continuous (diffusion) advance → replay recorded eddies (odt_code triplet map) ↻ ×100 steps

Deterministic operators can only track the mean — the eddies are irreducible noise. So **split the problem the way the ODT solver does**: the operator advances only the continuous diffusion dynamics (trained on 8,080 states, exact-solver targets, plus a PDE-residual loss at the predicted midpoint), and the realization's **recorded eddy sequence** (227 events: time, position, size) is applied explicitly, with kernel coefficients recomputed from the AI-predicted state.

$$\mathcal{L} = \|\Delta\hat{u} - \Delta u\|^2 + 0.1\,\|\partial_t u - \nu\,\partial_y^2 u\|^2_{\text{midpoint}}$$

Because the eddy sequence matches, the hybrid is compared **directly against the individual stochastic realization** — which no purely deterministic surrogate could do. The combo U_j = 10 m/s, D = 10 mm is again fully held out.



The hybrid reconstruction follows the individual realization, speckles and all: the difference map is far cleaner than in scenarios 1–2, and the final-time profiles overlay.

reconstruction rel-L2(u) vs realization	t = 25%	t = 50%	t = final
held-out combo (10 m/s, 10 mm)	**0.071**	**0.037**	**0.028**
mean over 8 training combos	0.053	0.046	0.041

Physics-informed loss weight 0.1; trained 20 epochs in 64.4 s. Hybrid rollout 0.439 s, of which the 227 triplet-map applications cost 0.006 s. Eddies inside a snapshot interval are applied after the AI advance — an O(Δt) operator-splitting approximation.

Computational cost

Wall-clock on one machine: ODT solver and AI rollouts on CPU, training on the Apple-silicon GPU. The speedup depends on **what a rollout replaces**: scenarios 1–2 stand in for a whole ensemble (~300×), scenario 3 generates a single realization (~9×).

	S1	S2	S3
one ODT realization [s]	3.70	3.74	3.77
ODT runs in dataset	40	360	90*
full ODT dataset [s]	148	1346	339*
FNO training [s]	23.5	11.3	64.4
AI rollout [s]	0.443	0.421	0.439†
speedup vs one realization	8×	9×	9×
speedup vs its ensemble	**268×**	**356×**	—

*reused from the scenario 2 dataset application †including triplet-map

In every scenario, **the training data — not the training itself — is the dominant cost.**

Takeaways

1. **Trained on noise, learns the mean.** A deterministic FNO trained on individual stochastic realizations converges to the ensemble-mean dynamics — the eddy noise is irreducible for any deterministic map.
2. **Statistics are learnable and interpolable.** Trained on ensemble averages, the operator predicts the mean dynamics at a fully held-out jet parameter combo (rel-L2 ≲ 0.12).
3. **Split physics from stochasticity.** A physics-informed operator for the continuous dynamics plus explicitly replayed eddy events reconstructs an *individual realization* (rel-L2 ≈ 0.03) — beyond any purely deterministic surrogate.
4. **Sub-second surrogates.** One rollout replaces a whole ensemble (up to 356×); generating training data dominates the cost, not training.

Outlook

- **Batched surrogate ensembles:** thousands of sub-second rollouts per GPU for parameter-space sweeps and UQ.
- **Learned eddy sampling:** replace the replayed eddy sequence with a learned stochastic event model to make the scenario 3 hybrid fully generative.
- **Richer physics:** scalar transport and reacting ODT jets, where ODT is used as a subgrid closure for LES.

Reproducibility

All code, data-generation scripts, trained models, and this analysis: **github.com/bgoshayeshi/AI-ODT** (ODT solver in `odt_code/`, scenarios in `Scenarios/`). FNO built on NVIDIA PhysicsNeMo; figures regenerable end-to-end from the repository.