
GEOMETRY-AWARE CONTINUOUS FLOW OPERATOR FOR 2D SHALLOW WATER EQUATIONS

A PREPRINT

Ronit Raj Baishya

Indian Institute of Technology Guwahati
Guwahati - 781039, India
ronitb@vt.edu

Dr. Natesan Srinivasan

Department of Mathematics
Indian Institute of Technology Guwahati
Guwahati - 781039, India
natesan@iitg.ac.in

October 18, 2026

ABSTRACT

The github repository can be found in the following link: <https://github.com/ronitbaishya1/Pi-CFO> Continuous Flow Operators (CFO) learn a continuous-time vector field from discrete physical trajectories, rather than a fixed temporal transition as neural operators do. For two dimensional shallow-water flow over variable bathymetry, a surrogate has to respect conservation laws, terrain-dependent forcing and equilibrium states, and it has to stay stable over long rollouts.

We build a geometry-aware, well-balanced, physics-informed CFO for the 2D shallow-water equations. The conservative state is $\mathbf{q} = [h, hu, hv]$, and the bed elevation $b(x, y)$ is kept as a separate conditioning field. Beyond the CFO flow-matching loss, we added a shallow-water residual, counterfactual bed-response loss, a lake-at-rest well-balanced loss and a rollout loss. Bathymetry enters through the geometric features $[b, b_x, b_y]$, which a learned geometry encoder processes before they are coupled to a U-FNO backbone. The reason for introduction of rollout loss was due to the accumulation of small local errors pile up during free rollout even though the recursive prediction in the one-step vector-field predictions can look accurate. To counter this, we fine-tune the final 64×64 model with a short differentiable horizon intervals RK4 rollout loss with four substeps. This helps in dropping the final Geometry-U-FNO model's ID relative L_2 error, mean Gaussian terrain-effect error, the mean OOD terrain-effect error and final ID mass drift at 64×64 .

We then evaluate the same 64×64 checkpoint zero-shot on a 128×128 grid. Applying the local U-Net branch directly at the finer resolution caused strong, resolution-dependent oscillations. We fixed this with a canonical U-Net evaluation. The 128×128 latent field is area-averaged down to the 64×64 training grid, the trained U-Net runs there, and its response is reconstructed at 128×128 by cubic interpolation. The Fourier and geometry branches stay at 128×128 , and no neural training is done at that resolution. Which helped improve the errors in both Gaussian case and the unseen cases. Together, these results show stable zero-shot resolution transfer that keeps the continuous-time rollout formulation intact.

Keywords Continuous Flow Operator · Physics-Informed Machine Learning · Neural Operators · Shallow-Water Equations · Variable Bathymetry · Geometry-Aware Learning · Rollout Stability · Zero-Shot Resolution Transfer

1 Introduction

Data-driven surrogates are increasingly used to speed up predictions for nonlinear systems governed by partial differential equations (PDEs). High-fidelity numerical solvers are accurate, but running them repeatedly gets expensive in optimization, inverse problems, uncertainty quantification, parameter studies, and real-time prediction[1–3].

Many neural operators learn a discrete temporal mapping[1, 4],

$$\mathbf{q}(t_i) \mapsto \mathbf{q}(t_{i+1}).$$

The Continuous Flow Operator (CFO) instead learns a continuous-time vector field[5],

$$\mathcal{N}_\theta(t, \mathbf{q}) \approx \frac{d\mathbf{q}}{dt},$$

which an ordinary differential equation solver can then integrate. For physical systems this is a natural fit, since the learned dynamics are not tied to one fixed time step.

Shallow-water flow over variable terrain asks for more than temporal continuity. The operator has to respond correctly to changes in bathymetry, preserve equilibrium states, and stay stable when its own predictions are fed back in as input[6, 7]. Spatial resolution transfer adds another difficulty[8]. Fourier layers work across grid sizes without modification, but the local convolutional U-Net branch of U-FNO has a receptive field whose physical size depends on the grid. A 3×3 kernel at 128×128 covers about half the physical width it covers at 64×64 [4]. So a U-FNO trained at 64×64 and evaluated directly at 128×128 can pick up new errors even when the Fourier branch transfers well.

We address these issues step by step. We first extend CFO to the full conservative shallow-water state $\mathbf{q} = [h, hu, hv]$ and add the variable-bottom shallow-water residual. A counterfactual bed-response loss then isolates the effect of changing bathymetry while holding the state and time fixed, and a lake-at-rest constraint encourages well-balanced behavior. Finally, we represent bathymetry with the geometric features $[b, b_x, b_y]$ and pass them through a learned geometry encoder before they enter a U-FNO backbone.

Two further changes matter in the final study. Short-horizon differentiable RK4 fine-tuning reduces recursive rollout error at the native 64×64 resolution. For zero-shot $64 \times 64 \rightarrow 128 \times 128$ transfer, a canonical U-Net evaluation keeps the local U-Net branch at its training resolution while the Fourier and geometry branches run directly on the finer grid.

Our contributions are:

1. a continuous-time CFO formulation for the full conservative two-dimensional shallow-water state $[h, hu, hv]$;
2. the variable-bottom shallow-water residual, a counterfactual bed-response loss, and a lake-at-rest well-balanced loss;
3. explicit geometry conditioning through $[b, b_x, b_y]$ and a learned geometry encoder;
4. a Geometry-U-FNO backbone that combines global Fourier interactions with local U-Net refinement;
5. rollout-stability fine-tuning on short differentiable RK4 trajectories, in place of one-step vector-field matching alone;
6. evaluation on translated Gaussian hills and on structurally unseen terrain families;
7. a canonical U-Net strategy for zero-shot $64 \times 64 \rightarrow 128 \times 128$ transfer, with no 128×128 neural-network training.

2 Model Problem and Dataset

2.1 Two-Dimensional Shallow-Water Equations with Bathymetry

The conservative shallow-water state is

$$\mathbf{q}(x, y, t) = \begin{bmatrix} h \\ hu \\ hv \end{bmatrix},$$

where h is the water depth and u, v are the depth-averaged horizontal velocities.

The bed elevation is $b = b(x, y)$, and the free-surface elevation is

$$\eta = h + b.$$

Mass conservation reads

$$h_t + (hu)_x + (hv)_y = 0.$$

The x -momentum equation is

$$(hu)_t + \frac{\partial}{\partial x} \left(\frac{(hu)^2}{h} + \frac{1}{2}gh^2 \right) + \frac{\partial}{\partial y} \left(\frac{(hu)(hv)}{h} \right) = -gh b_x,$$

and the y -momentum equation is

$$(hv)_t + \frac{\partial}{\partial x} \left(\frac{(hu)(hv)}{h} \right) + \frac{\partial}{\partial y} \left(\frac{(hv)^2}{h} + \frac{1}{2}gh^2 \right) = -gh b_y.$$

Together they take the compact form

$$\boxed{\mathbf{q}_t + \partial_x \mathbf{F}_x(\mathbf{q}) + \partial_y \mathbf{F}_y(\mathbf{q}) = \mathbf{S}_b(\mathbf{q}, b),}$$

with fluxes

$$\mathbf{F}_x = \begin{bmatrix} hu \\ \frac{(hu)^2}{h} + \frac{1}{2}gh^2 \\ \frac{(hu)(hv)}{h} \end{bmatrix},$$

$$\mathbf{F}_y = \begin{bmatrix} hv \\ \frac{(hu)(hv)}{h} \\ \frac{(hv)^2}{h} + \frac{1}{2}gh^2 \end{bmatrix},$$

and bathymetry source term

$$\mathbf{S}_b = \begin{bmatrix} 0 \\ -ghb_x \\ -ghb_y \end{bmatrix}.$$

In the non-dimensional experiments, $g = 1$.

2.2 Variable-Bathymetry Dataset

The computational domain is $(x, y) \in [-2.5, 2.5]^2$. The final model is trained at 64×64 , where $\Delta x = \Delta y = 5/64 = 0.078125$. The cross-resolution study uses an independent PyClaw reference at 128×128 , where $\Delta x = \Delta y = 5/128 = 0.0390625$.

Each trajectory has 51 snapshots over $t \in [0, 1]$.

The initial free surface is

$$\eta(x, y, 0) = \begin{cases} 2, & r \leq R, \\ 1, & r > R, \end{cases}$$

with $r = \sqrt{x^2 + y^2}$. The initial water depth follows from the bed:

$$h(x, y, 0) = \eta(x, y, 0) - b(x, y),$$

and the fluid starts at rest:

$$hu(x, y, 0) = hv(x, y, 0) = 0.$$

We generate the reference trajectories with PyClaw, using a bathymetry-aware shallow-water solver. The in-distribution data use smooth Gaussian hills with varying geometric parameters. All final experiments share the same train/validation/test split.

2.3 Counterfactual and Unseen Terrain Tests

Two further evaluation sets check whether the operator has learned a meaningful response to terrain.

The first is a set of paired Gaussian simulations. Each physical experiment is run once over a flat bed and once over a Gaussian hill placed to the left of, at the center of, or to the right of the domain. The isolated terrain contribution is

$$\Delta \mathbf{q}_b(t) = \mathbf{q}_{\text{terrain}}(t) - \mathbf{q}_{\text{flat}}(t). \quad (2.1)$$

The second set uses terrain families that differ structurally from the single circular Gaussian:

1. two separated hills;
2. a narrow, tall Gaussian hill;
3. an elongated, axis-aligned ridge;
4. a rotated, anisotropic ridge;
5. a three-component multi-hill terrain.

These geometries are used only for testing; the final 64×64 checkpoint is not retrained on them.

3 Geometry-Aware Well-Balanced Physics-Informed CFO

3.1 Continuous-Time Vector Field

CFO learns a continuous-time vector field instead of a fixed time-step map [5],

$$\boxed{\mathcal{N}_\theta : (t, \mathbf{q}, b) \mapsto \frac{d\mathbf{q}}{dt}.}$$

The learned state evolution satisfies

$$\frac{d\mathbf{q}}{dt} = \mathcal{N}_\theta(t, \mathbf{q}, b).$$

Once the vector field is learned, we generate trajectories with a fourth-order Runge–Kutta (RK4) integrator.

3.2 Continuous Temporal Spline and CFO Loss

For an interval $[t_s, t_e]$, define[5]

$$\tau = \frac{t - t_s}{t_e - t_s}.$$

The quintic temporal spline is

$$\mathbf{s}(\tau) = \mathbf{a}_0 + \mathbf{a}_1\tau + \mathbf{a}_2\tau^2 + \mathbf{a}_3\tau^3 + \mathbf{a}_4\tau^4 + \mathbf{a}_5\tau^5,$$

and its time derivative is available analytically:

$$\dot{\mathbf{s}} = \frac{1}{\Delta t} (\mathbf{a}_1 + 2\mathbf{a}_2\tau + 3\mathbf{a}_3\tau^2 + 4\mathbf{a}_4\tau^3 + 5\mathbf{a}_5\tau^4).$$

The stochastic CFO path is

$$\boxed{\mathbf{I}(t) = \mathbf{s}(t) + \gamma\tau^3(1 - \tau)^3\boldsymbol{\epsilon},}$$

with

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

The CFO loss is

$$\mathcal{L}_{\text{CFO}} = \mathbb{E} \left[\left\| \mathcal{N}_\theta(t, \mathbf{I}(t), b) - \dot{\mathbf{I}}(t) \right\|_2^2 \right].$$

3.3 Variable-Bottom Physics Loss

The physics branch is evaluated on the clean spline state [1, 6, 7],

$$\mathbf{q} = \mathbf{s}(t),$$

not on the noise-perturbed CFO state. The network prediction is

$$\mathbf{q}_t^{NN} = \mathcal{N}_\theta(t, \mathbf{s}(t), b),$$

and the physical residual is

$$\mathcal{R}_{\text{SWE}} = \mathbf{q}_t^{NN} + \partial_x \mathbf{F}_x + \partial_y \mathbf{F}_y - \mathbf{S}_b.$$

The PDE loss is

$$\mathcal{L}_{\text{PDE}} = \text{mean}(\mathcal{R}_{\text{SWE}}^2).$$

Spatial derivatives are finite differences using the spacing of the current grid. For an $N \times N$ grid, the geometry and PDE derivatives use $\Delta x = \Delta y = 5/N$, rather than a spacing hard-coded for one resolution.

3.4 Counterfactual Bed-Response Loss

The PDE residual constrains the full shallow-water dynamics, but it does not by itself test whether the model reacts correctly when only the terrain changes.

For a fixed state $\mathbf{q}$ and time t , define

$$\Delta_b \mathcal{N}_\theta = \mathcal{N}_\theta(t, \mathbf{q}, b) - \mathcal{N}_\theta(t, \mathbf{q}, 0).$$

The direct bathymetry contribution to the momentum equations is

$$\Delta_b \begin{bmatrix} (hu)_t \\ (hv)_t \end{bmatrix} = \begin{bmatrix} -ghb_x \\ -ghb_y \end{bmatrix},$$

which gives the bed-response loss

$$\mathcal{L}_{\text{bed}} = \text{mean} \left[\left\| [\Delta_b \mathcal{N}_\theta]_{hu, hv} - \begin{bmatrix} -ghb_x \\ -ghb_y \end{bmatrix} \right\|_2^2 \right].$$

Put simply, the loss queries the model twice, once with the real bed and once with a flat bed. Because the state and time are identical, the difference between the two vector fields should be the direct physical response to terrain.

3.5 Lake-at-Rest Well-Balanced Constraint

The lake-at-rest state is an important equilibrium. The fluid is motionless,

$$hu = hv = 0,$$

and the free surface is constant,

$$h + b = \eta_0.$$

The equilibrium state used in training is

$$\mathbf{q}_{eq} = [\eta_0 - b, 0, 0],$$

with $\eta_0 = 1.5$. A steady vector field should satisfy

$$\mathcal{N}_\theta(t, \mathbf{q}_{eq}, b) = \mathbf{0},$$

so the well-balanced loss is

$$\mathcal{L}_{WB} = \text{mean} \left[\|\mathcal{N}_\theta(t, \mathbf{q}_{eq}, b)\|_2^2 \right].$$

3.6 Explicit Geometry Encoding

If only the scalar bathymetry $b(x, y)$ is supplied, the backbone has to infer local slope and terrain orientation on its own. The final model instead builds

$$\mathbf{g} = [b, b_x, b_y]$$

and passes it through a learned convolutional geometry encoder,

$$\mathbf{z}_g = \mathcal{E}_\phi(b, b_x, b_y).$$

Bathymetry is not appended to the evolving physical state. The dynamic variables remain

$$\mathbf{q} = [h, hu, hv],$$

and b, b_x, b_y enter separately as fixed geometric conditioning.

3.7 Geometry-U-FNO Backbone

The encoded geometry is fused with the lifted state before the operator blocks,

$$[b, b_x, b_y] \xrightarrow{\mathcal{E}_\phi} \mathbf{z}_g,$$

and the full map is

$$(\mathbf{q}, t, \mathbf{z}_g) \xrightarrow{\text{Geometry-U-FNO}} \mathbf{q}_t.$$

Each U-FNO block has three components[4],

$$\mathbf{z}^{\ell+1} = \mathcal{F}^\ell(\mathbf{z}^\ell) + \mathcal{W}^\ell(\mathbf{z}^\ell) + \mathcal{U}^\ell(\mathbf{z}^\ell),$$

where $\mathcal{F}^\ell$ is the Fourier branch, $\mathcal{W}^\ell$ the pointwise branch, and $\mathcal{U}^\ell$ the local U-Net branch.

The final architecture uses 12 Fourier modes per spatial direction, latent width 64, four operator blocks, two U-Net local blocks, geometry width 16, and geometry depth 2 [4].

3.8 Base Physics-Informed Objective

The base geometry-aware model is trained with [1, 9]

$$\mathcal{L}_{\text{base}} = \mathcal{L}_{\text{CFO}} + \lambda_{\text{PDE}} \mathcal{L}_{\text{PDE}} + \lambda_{\text{bed}} \mathcal{L}_{\text{bed}} + \lambda_{\text{WB}} \mathcal{L}_{\text{WB}}.$$

The retained weights are

$$\lambda_{\text{PDE}} = 0.01, \quad \lambda_{\text{bed}} = 0.70, \quad \lambda_{\text{WB}} = 0.10.$$

3.9 Rollout Error Accumulation and Its Correction

In the 64×64 study, one-step vector-field predictions were much better than long free rollouts. The model could represent the spatial field well enough; the problem was that small prediction errors were fed back into the model and grew with time, especially in the momentum channels [10].

Spectral diagnostics showed the same thing. High-frequency error stayed small for local one-step predictions but grew during recursive rollout, and phase and coherence degraded over time. Refining only the numerical RK4 resolution did not help, because the error came from the learned vector field itself.

We therefore fine-tune the final model with a short differentiable free rollout. The first state comes directly from the reference trajectory,

$$\hat{\mathbf{q}}_0 = \mathbf{q}_i,$$

and the network is then integrated with RK4,

$$\hat{\mathbf{q}}_{k+1} = \text{RK4}(\hat{\mathbf{q}}_k, \mathcal{N}_\theta, b).$$

The predicted states are compared with the next reference snapshots. After the first state, each step starts from the model’s own previous prediction, not from exact data, so the loss exposes the model to the same recursive error mechanism it meets at inference time.

For a rollout horizon K , the loss is

$$\mathcal{L}_{\text{roll}} = \frac{1}{K} \sum_{k=1}^K \sum_{c \in \{h, hu, hv\}} w_c \text{mean} \left[\left(\frac{\hat{q}_{k,c} - q_{i+k,c}}{s_c} \right)^2 \right],$$

where s_c is a channel scale. The retained channel weights are

$$w_h = 0.25, \quad w_{hu} = w_{hv} = 0.375,$$

which put more emphasis on the two momentum components.

The final fine-tuning configuration is $K = 2$, $\lambda_{\text{roll}} = 0.10$, $N_{\text{RK4}} = 4$.

We also tried an optional high-frequency error penalty but did not use it in the final checkpoint,

$$\lambda_{\text{HF}} = 0.$$

The final objective is

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{base}} + 0.10 \mathcal{L}_{\text{roll}}.$$

3.10 Why Naive $64 \rightarrow 128$ U-FNO Transfer Was Unstable

Once the 64×64 model was stable, we evaluated the same checkpoint directly on 128×128 . The Fourier branch handles a different grid naturally because it acts on Fourier modes. The local U-Net branch does not, because its convolution kernels are defined in grid cells [4, 8].

A 3×3 kernel trained at 64×64 covers a particular physical neighborhood. Applied unchanged at 128×128 , it covers about half that physical width. The weights are the same, but the physical scale the local branch sees is not. In practice this produced strong local oscillations and poor momentum profiles during zero-shot rollout.

3.11 Canonical U-Net with Cubic Reconstruction

To preserve the physical scale the U-Net learned, we evaluate only the local U-Net branch at its original training resolution.

Let z_{128} be the latent field on the finer grid. Area averaging reduces it to 64×64 ,

$$z_{64} = \mathcal{D}_{\text{area}}(z_{128}),$$

replacing each 2×2 group of cells by its average. The original trained U-Net is applied there,

$$\mathbf{u}_{64} = \mathcal{U}_{64}(z_{64}),$$

and its response is reconstructed on the finer grid by cubic interpolation,

$$\mathbf{u}_{128} = \mathcal{R}_{\text{cubic}}(\mathbf{u}_{64}).$$

The full block becomes

$$\mathbf{z}_{128}^{\ell+1} = \mathcal{F}_{128}^{\ell}(\mathbf{z}_{128}^{\ell}) + \mathcal{W}_{128}^{\ell}(\mathbf{z}_{128}^{\ell}) + \mathcal{R}_{\text{cubic}}[\mathcal{U}_{64}^{\ell}(\mathcal{D}_{\text{area}}(\mathbf{z}_{128}^{\ell}))].$$

Only the U-Net branch drops back to 64×64 . The Fourier branch and geometry encoder stay on the 128×128 grid. No trainable parameters are added, and the same 64×64 checkpoint is used.

This operation runs every time the neural vector field is evaluated, so it is active inside every RK4 stage of the 128×128 zero-shot rollout. It is not a post-processing filter applied to the prediction.

3.12 Zero-Shot Stabilization Choices

We tested several simpler alternatives before fixing the final cross-resolution configuration.

Area averaging followed by linear interpolation already removed much of the instability, which confirms that the main problem was the local U-Net scale mismatch. Switching from linear to cubic reconstruction reduced the state and momentum errors further.

Blending in a small contribution from the native 128×128 U-Net branch brought back part of the resolution mismatch. Canonicalizing the geometry encoder gave no meaningful extra benefit, and we dropped a source-resolution low-pass filter. Raising the evaluation integrator from four to eight RK4 steps per interval changed almost nothing. The final zero-shot configuration is

Fourier branch: 128×128 ,
 geometry encoder: 128×128 ,
 U-Net branch: $128 \xrightarrow{\text{area}} 64 \xrightarrow{\mathcal{U}} 64 \xrightarrow{\text{cubic}} 128$,
 native U-Net blend: 0,
 spectral low-pass filter: off,
 RK4 steps per interval: 4.

3.13 Model Development Sequence

The development sequence is

CFO $\rightarrow$ Bathymetry-CFO
 $\rightarrow$ WB-Bed-PI-CFO
 $\rightarrow$ Geometry-U-FNO
 $\rightarrow$ Rollout-Stable Geometry-U-FNO
 $\rightarrow$ Canonical Zero-Shot Geometry-U-FNO.

We also tried alternative backbones and tangent/Jacobian diagnostics during model selection[11]. They helped us settle on Geometry-U-FNO, but the final model has no tangent regularization.

3.14 Computational Framework

Figure 1 summarizes the final native-resolution training and rollout-stability framework.

Geometry-aware continuous-flow operator: training and zero-shot inference

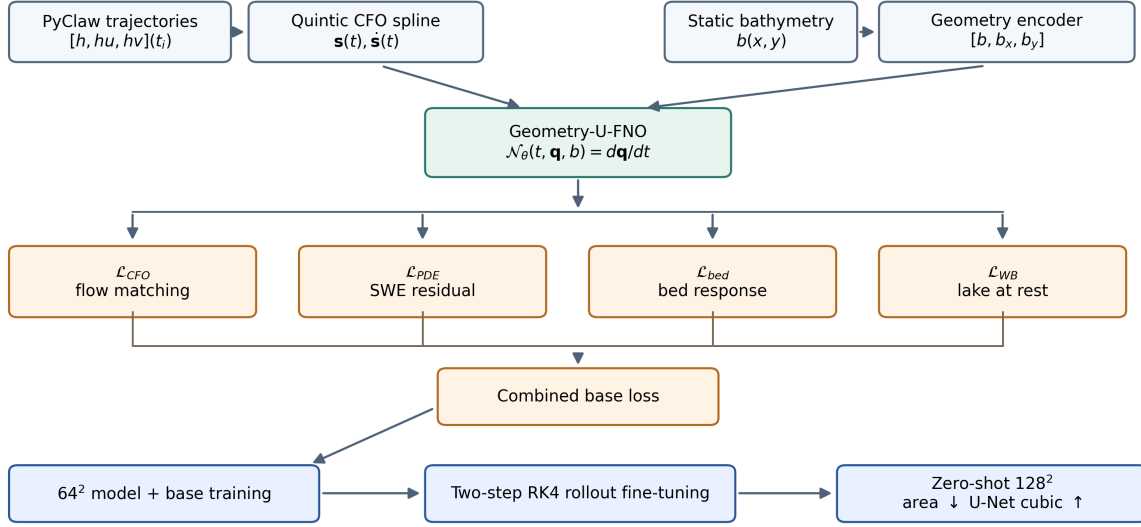


Figure 1: Simplified geometry-aware PI-CFO training framework. PyClaw trajectories provide the continuous-time training states, while bathymetry is processed through a geometry encoder. The shared Geometry-U-FNO is trained using CFO, physics-informed, bed-response, well-balanced, and short-horizon RK4 rollout losses.

3.15 Training and Inference Algorithm

Algorithm 1 summarizes the complete training, rollout-stabilization, and cross-resolution inference procedure. The main idea is to keep the shallow-water state and bathymetry separate. The state $\mathbf{q} = [h, hu, hv]$ evolves in time, while the bed is supplied to the model through the geometry features $[b, b_x, b_y]$.

Algorithm 1: Geometry-aware rollout-stable PI-CFO training and inference

Require: PyClaw trajectories $\{\mathbf{q}(t_i)\}_{i=0}^{N_t}$, bathymetry $b(x, y)$, training resolution 64×64

Require: $\lambda_{\text{PDE}} = 0.01$, $\lambda_{\text{bed}} = 0.70$, $\lambda_{\text{WB}} = 0.10$, $\lambda_{\text{roll}} = 0.10$

Ensure: Continuous-time geometry-aware operator $\mathcal{N}_\theta(t, \mathbf{q}, b)$

1: Compute bathymetry gradients

$$b_x = \frac{\partial b}{\partial x}, \quad b_y = \frac{\partial b}{\partial y}$$

2: Construct geometry input

$$\mathbf{g} = [b, b_x, b_y]$$

3: Encode geometry

$$\mathbf{z}_g = \mathcal{E}_\phi(\mathbf{g})$$

4: Fit continuous temporal splines $s(t)$ through each PyClaw trajectory

5: Construct the CFO conditional state

$$\mathbf{I}(t) = s(t) + \gamma\tau^3(1 - \tau)^3\epsilon$$

Stage 1: Geometry-aware physics-informed training

6: **for** epoch = 1, ..., 1000 **do**

7: Predict the continuous-time vector field

$$\hat{\mathbf{q}}_t = \mathcal{N}_\theta(t, \mathbf{I}(t), \mathbf{z}_g)$$

8: Compute CFO flow-matching loss

$$\mathcal{L}_{\text{CFO}} = \left\| \hat{\mathbf{q}}_t - \mathbf{I}(t) \right\|_2^2$$

9: Compute variable-bottom SWE residual loss $\mathcal{L}_{\text{PDE}}$

- 10: Compute counterfactual bed-response loss $\mathcal{L}_{\text{bed}}$
- 11: Compute lake-at-rest well-balanced loss $\mathcal{L}_{\text{WB}}$
- 12: Form the base objective

$$\mathcal{L}_{\text{base}} = \mathcal{L}_{\text{CFO}} + 0.01\mathcal{L}_{\text{PDE}} + 0.70\mathcal{L}_{\text{bed}} + 0.10\mathcal{L}_{\text{WB}}$$

- 13: Update network parameters
- 14: **end for**
- Stage 2: Rollout-stability fine-tuning**
- 15: **for** fine-tuning epoch = 1, ..., 200 **do**
- 16: Select an exact PyClaw state $\mathbf{q}_0$
- 17: Starting from $\mathbf{q}_0$, perform a two-step differentiable rollout using RK4
- 18: Use four RK4 substeps within each rollout interval
- 19: Compare predicted states $\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2$ with PyClaw targets $\mathbf{q}_1, \mathbf{q}_2$
- 20: Compute

$$\mathcal{L}_{\text{roll}} = \frac{1}{2} \sum_{k=1}^2 \frac{\|\hat{\mathbf{q}}_k - \mathbf{q}_k\|_2^2}{\|\mathbf{q}_k\|_2^2 + \epsilon}$$

- 21: Optimize

$$\mathcal{L}_{\text{FT}} = \mathcal{L}_{\text{base}} + 0.10\mathcal{L}_{\text{roll}}$$

- 22: **end for**
- Stage 3: Native 64×64 inference**
- 23: Integrate

$$\frac{d\mathbf{q}}{dt} = \mathcal{N}_{\theta}(t, \mathbf{q}, b)$$

using RK4 with four substeps per output interval

Stage 4: Zero-shot $64 \rightarrow 128$ inference

- 24: Start from the 128×128 state and bathymetry
- 25: Area-downsample the fine-grid fields to the canonical 64×64 model grid
- 26: Evaluate the trained Geometry-U-FNO on the canonical grid
- 27: Cubically reconstruct the predicted vector field from 64×64 to 128×128
- 28: Apply the reconstructed vector field inside RK4 to advance the 128×128 solution
- 29: Repeat without retraining the neural operator

4 Numerical Experiments

4.1 Final Configuration

The final configuration we decided on is shown in Table 1

4.2 Evaluation Metrics

The global relative state error is

$$E_q = \frac{\|\hat{\mathbf{q}} - \mathbf{q}\|_2}{\|\mathbf{q}\|_2}.$$

We also report E_h , E_{hu} , and E_{hv} separately. The free-surface error is computed on $\eta = h + b$. Total water mass is approximated by

$$M(t) = \sum_{i,j} h_{ij}(t) \Delta x \Delta y,$$

and the relative mass drift is

$$E_M(t) = \frac{|M(t) - M(0)|}{|M(0)|}.$$

For paired terrain experiments, the isolated terrain effect is given by Eq. (2.1). A direct-condition diagnostic also fixes the state and time and changes only the bathymetry, from $b = 0$ to the target bed. It tests whether the network vector field assigns the correct momentum response to bathymetric forcing.

In the zero-shot study, the 128×128 PyClaw trajectory is the reference. We interpolate the 64×64 PyClaw trajectory to the finer grid only when we need a direct numerical resolution-difference metric. It is not a learned-model prediction. It indicates how much error the change in numerical resolution produces by itself.

Table 1: Final native-resolution and zero-shot Geometry-U-FNO configuration.

Parameter	Value
Dynamic state	$[h, hu, hv]$
Geometry features	$[b, b_x, b_y]$
Training resolution	64×64
Zero-shot evaluation resolution	128×128
Snapshots	51
Backbone	Geometry-U-FNO
Fourier modes	12×12
Latent width	64
Operator blocks	4
U-FNO local blocks	2
Geometry hidden width	16
Geometry encoder depth	2
CFO noise γ	10^{-5}
λ_{PDE}	0.01
λ_{bed}	0.70
λ_{WB}	0.10
λ_{roll}	0.10
Rollout horizon	$K = 2$
Rollout channel weights	$(0.25, 0.375, 0.375)$
Rollout RK4 substeps	4
Final evaluation RK4 steps	4
Canonical U-Net grid	64×64
Zero-shot downsampling	Area averaging
Zero-shot reconstruction	Cubic interpolation
Native 128^2 U-Net blend	0

Table 2: Final 64×64 comparison. We keep only the metrics most directly tied to the main physical and generalization claims.

Model	ID Rel. L_2	ID mass drift	Gaussian Rel. L_2	Gaussian effect	OOD Rel. L_2	OOD effect
PyClaw	0	5.29×10^{-10}	0	0	0	0
Bathy-CFO	0.036128	0.005443	0.032233	0.628432	0.036759	0.875895
Final Geometry-U-FNO	0.015235	0.001076	0.015670	0.238144	0.016684	0.318635

5 Results and Discussion

5.1 Final 64×64 Bathymetry-Conditioned Performance

The final native-resolution comparison is

$$\text{PyClaw} \rightarrow \text{Bathy-CFO} \rightarrow \text{Rollout-Stable Geometry-U-FNO}.$$

PyClaw is the numerical reference. Bathy-CFO receives bathymetry as a condition but lacks the final geometry-aware physics and rollout-stability modifications.

Relative to Bathy-CFO, the final model cuts the ID relative error by about 57.8% and the ID mass drift by about 80.2%. The Gaussian mean state error falls by about 51.4% and the isolated Gaussian terrain-effect error by about 62.1%. On the unseen terrain families, the mean state error falls by about 54.6% and the mean terrain-effect error by about 63.6%.

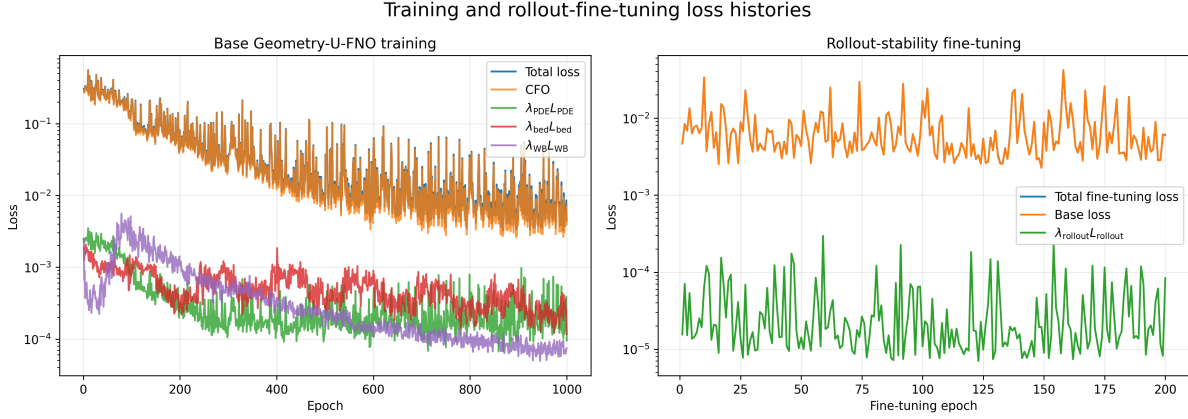


Figure 2: Training and rollout-stability fine-tuning histories for the final Geometry-U-FNO. The left panel shows the base geometry-aware physics-informed training over 1000 epochs, including the CFO, PDE, bed-response, and well-balanced contributions. The right panel shows the subsequent short-horizon rollout fine-tuning. The rollout term stays small relative to the base objective but directly penalizes recursive prediction error.

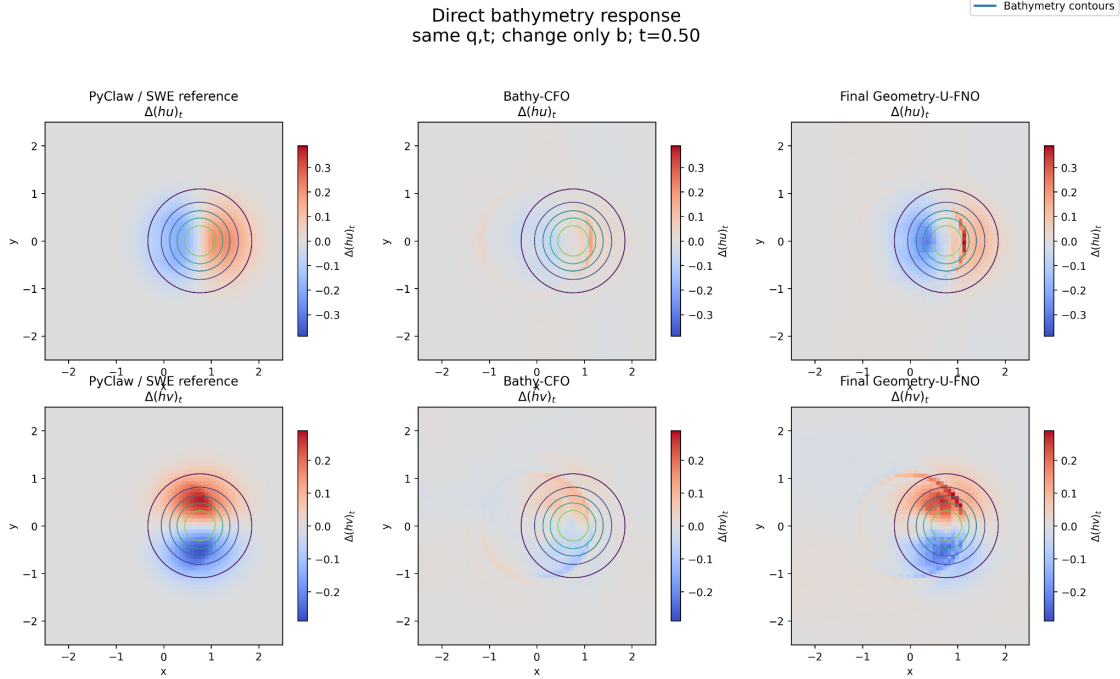


Figure 3: Direct bathymetry-response comparison at $t = 0.50$. The state $\mathbf{q}$ and time are held fixed while only the bathymetry condition changes. PyClaw/SWE gives the physical reference, followed by Bathy-CFO and the final rollout-stable Geometry-U-FNO. Contours show the Gaussian bed.

5.2 Training and Rollout-Stability Fine-Tuning

Figure 2 shows the base Geometry-U-FNO loss decreasing steadily over the full training window. The dominant CFO term falls together with the total loss, while the weighted PDE, bed-response, and well-balanced terms stay several orders of magnitude smaller and keep decreasing. The rollout stage is not a second full training phase. It exposes the already trained vector field to its own recursively predicted states. The fine-tuning histories are noisier as a result, but neither the base objective nor the weighted rollout term grows unstably. This fits the intended role of the rollout term as a targeted stability correction, not a replacement for the original CFO and physics-informed objectives.

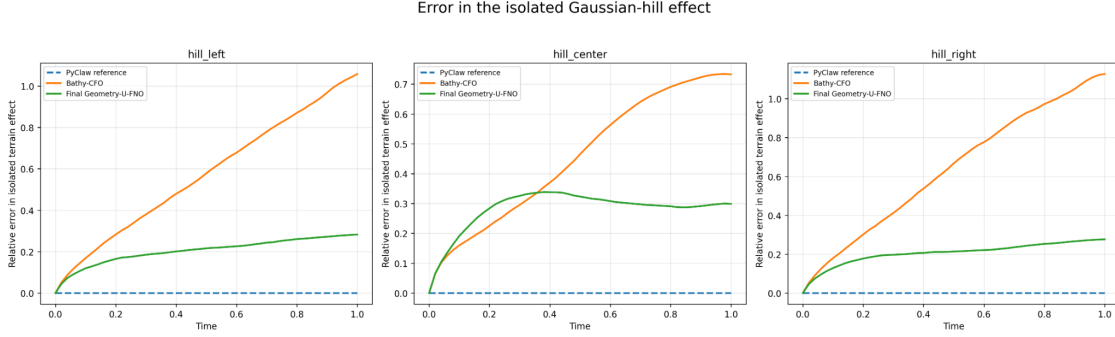


Figure 4: Relative error in the isolated terrain effect for left, center, and right Gaussian hills. The PyClaw line is the zero-error reference. The final Geometry-U-FNO stays well below Bathy-CFO over the long rollout, especially for the translated left and right hills.

5.3 Response at 64×64 for different Geometries

The direct-condition error drops from 0.781381 for Bathy-CFO to 0.359050 for the final model. The cosine similarity with the physical response rises from 0.782397 to 0.933545, and the response gain moves from 0.310485 to 0.912971. The final model therefore produces a stronger terrain response, and that response points much closer to the direction of the shallow-water source term. For isolated gaussian-hill, the two learned models separate clearly for the translated left and righthills. For the centered hill, the final model reaches a moderate error early in the rollout and then stays comparatively bounded, while Bathy-CFO keeps growing. This fits the purpose of the short-horizon rollout fine-tuning: the model learns to match a local vector field and also to stay useful once its own predictions are fed back to it. For terrain-induced velocity, as the hill moves across the domain, the final Geometry-U-FNO keeps the position and orientation of the terrain-induced response better. Bathy-CFO tends to under-represent or misplace the local terrain effect, even when its full-state rollout looks plausible. The final model keeps the dominant terrain location and orientation for multiple hills and anisotropic ridges. The narrow-tall case stays hard, since it has a shorter spatial scale and larger local bed gradients than the smooth training family. Even so, the overall OOD terrain-effect metric improves substantially over Bathy-CFO.

5.4 Error Distributions over Unseen Bathymetry and Unseen Terrain Families

Figure 7 shows OOD generalization more fully than a single averaged metric. For water depth, the Geometry-U-FNO error distribution sits at clearly smaller values and is more concentrated than the Bathy-CFO distribution. Both momentum components show the same trend. Bathy-CFO develops long high-error tails in hu and hv , while the final Geometry-U-FNO keeps most samples in a much lower error range. The improvement therefore does not come from a few favorable cases. It appears across a large fraction of the unseen-terrain rollout states. Figure 8 splits the OOD results by terrain family. The final Geometry-U-FNO lowers the mean error in all three state variables for every tested family, so the gain is not tied to one type of bed. Both methods have their largest errors on the narrow-tall terrain, especially in the momentum components. This is physically plausible: the narrow feature has a shorter length scale and stronger local bathymetry gradients than the smooth Gaussian training family. The elongated and rotated ridges are hard because of their anisotropy, and the two-hill and multi-hill cases test several interacting terrain features. Across all of these, the final model keeps a clear error reduction over Bathy-CFO. The final model follows the PyClaw free-surface structure more closely than Bathy-CFO and is less irregular in the central wave region. The momentum fields separate the models more than the depth field does. The final model removes much of the spurious local variation in Bathy-CFO and keeps the dominant positive and negative momentum structures.

5.5 Zero-Shot $64 \times 64 \rightarrow 128 \times 128$ Resolution Transfer

The cross-resolution experiment uses the same learned 64×64 checkpoints. We do no 128×128 neural-network training or fine-tuning[8].

The final comparison is

$$\text{PyClaw}_{64} \text{ vs } \text{Bathy-CFO}_{64 \rightarrow 128} \text{ vs } \text{Canonical Geometry-U-FNO}_{64 \rightarrow 128} \text{ vs } \text{PyClaw}_{128}.$$

We show the 64×64 PyClaw trajectory to illustrate the resolution change. When we report errors against PyClaw-128, the PyClaw-64 state is interpolated to the finer grid.

For the centered Gaussian hill, canonical Geometry-U-FNO cuts the overall zero-shot error by about 19.6% relative to Bathy-CFO and the terrain-effect error by about 35.7%. For the unseen two-hill case the reductions are about 20.2% and 62.6%. The PyClaw-64 interpolation rows sit below the learned models. That is expected, since they compare

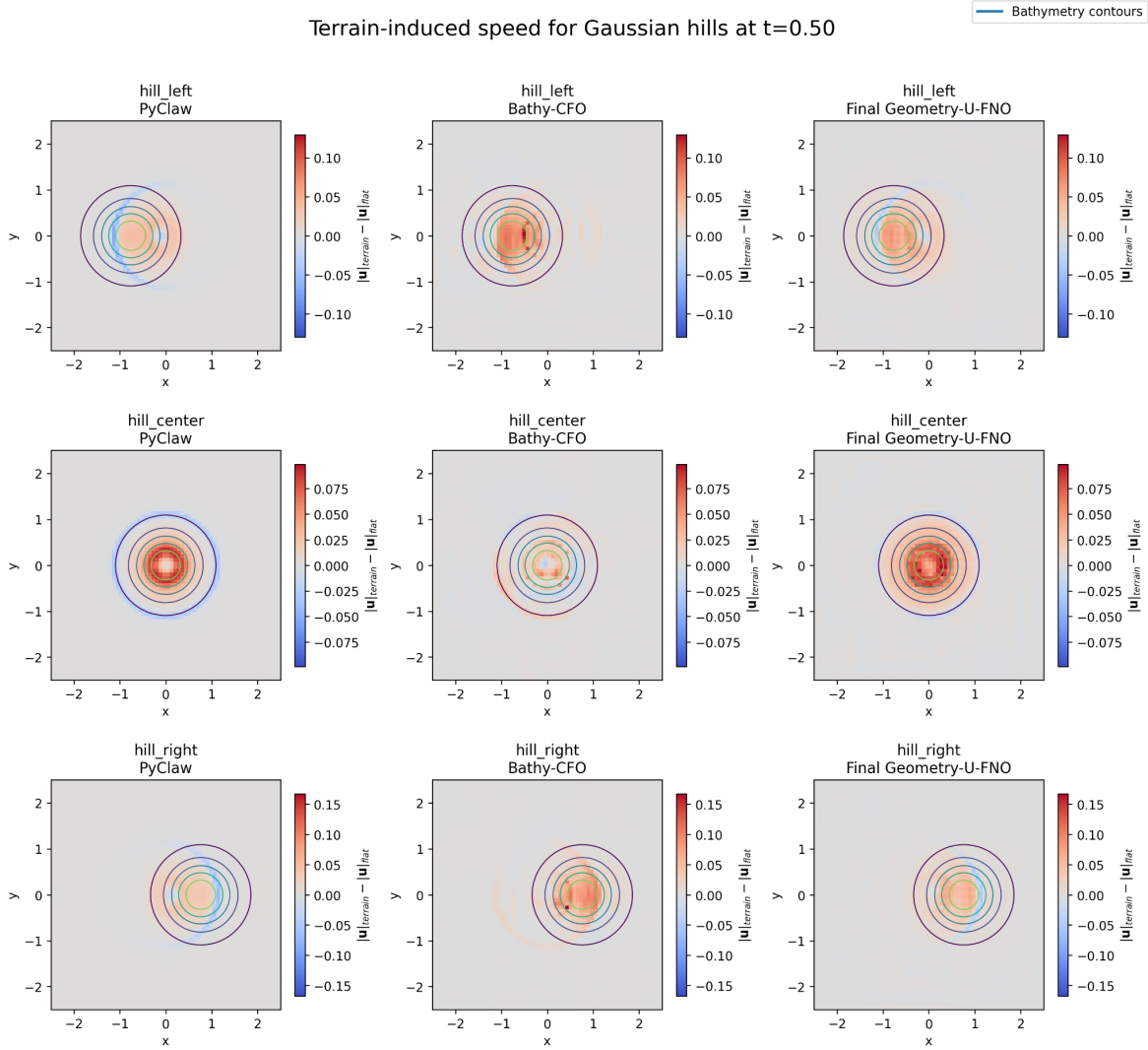


Figure 5: Terrain-induced speed response for the three Gaussian counterfactual cases at $t = 0.50$. Each row is a hill location, and the columns compare PyClaw, Bathy-CFO, and the final Geometry-U-FNO. The flat-bed response is subtracted so the displayed field shows only the part of the dynamics that terrain induces.

Table 3: Zero-shot $64 \rightarrow 128$ metrics for the centered Gaussian hill and the unseen two-hill case. PyClaw-128 is the reference. The PyClaw-64 interpolation row shows the numerical resolution-difference level and is not a learned model.

Case	Method	Rel. L_2	E_h	E_{hu}	E_{hv}	Terrain-effect error
Gaussian center	PyClaw64 interpolated	0.024145	0.015908	0.145711	0.147421	0.199125
	Bathy-CFO $64 \rightarrow 128$	0.035048	0.026177	0.176855	0.200134	0.563551
	Canonical Geo-U-FNO $64 \rightarrow 128$	0.028191	0.020713	0.158737	0.150654	0.362361
	PyClaw128 reference	0	0	0	0	0
Two hills	PyClaw64 interpolated	0.024368	0.015992	0.147592	0.146692	0.205860
	Bathy-CFO $64 \rightarrow 128$	0.035937	0.025923	0.193600	0.205471	0.817599
	Canonical Geo-U-FNO $64 \rightarrow 128$	0.028692	0.020764	0.164343	0.153369	0.305634
	PyClaw128 reference	0	0	0	0	0

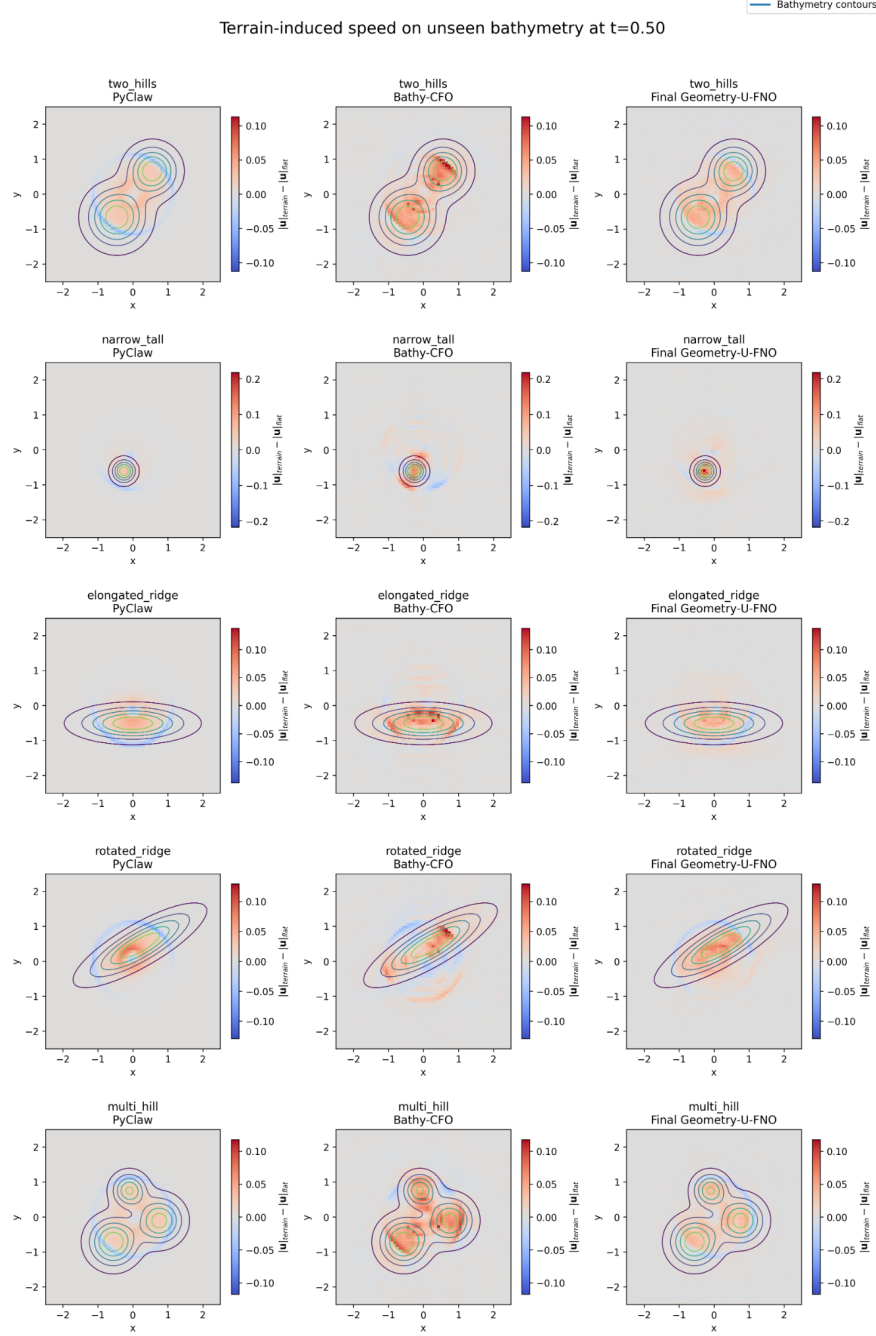


Figure 6: Terrain-induced speed on unseen bathymetry families at $t = 0.50$. So, if we generalize for unseen bathymetry, the rows show two hills, a narrow-tall hill, an elongated ridge, a rotated ridge, and a multi-hill terrain. PyClaw, Bathy-CFO, and the final Geometry-U-FNO are compared.

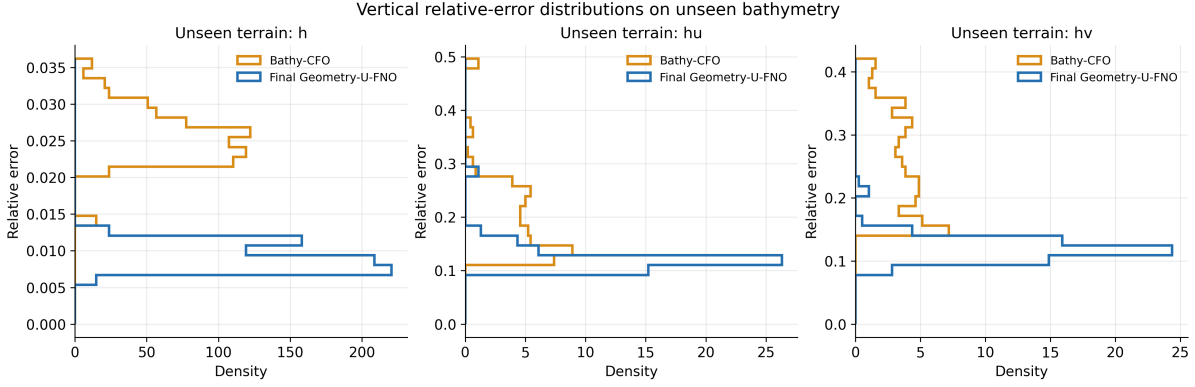


Figure 7: Distribution of relative approximation errors over the unseen bathymetry cases for h , hu , and hv . Bathy-CFO is compared with the final rollout-stable Geometry-U-FNO over the complete OOD rollouts. A leftward shift means lower approximation error.

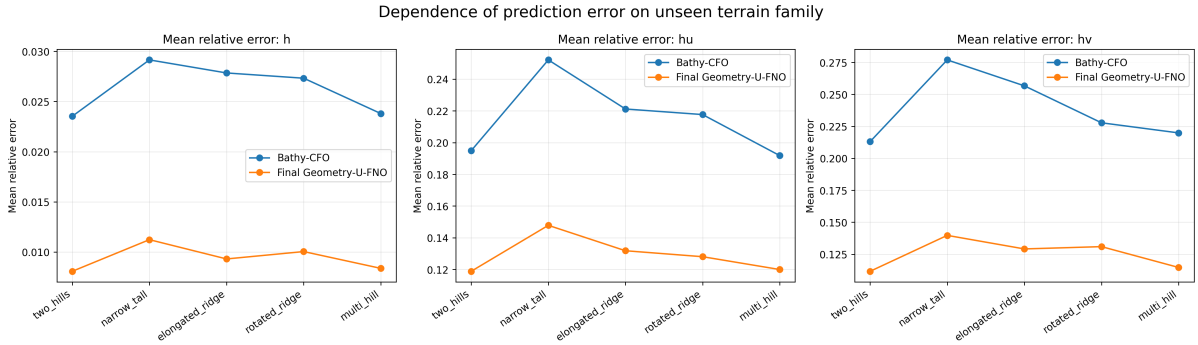


Figure 8: Mean relative error in h , hu , and hv for the five unseen bathymetry families: two hills, narrow-tall, elongated ridge, rotated ridge, and multi-hill. The final Geometry-U-FNO stays below Bathy-CFO for every state component and every terrain family.

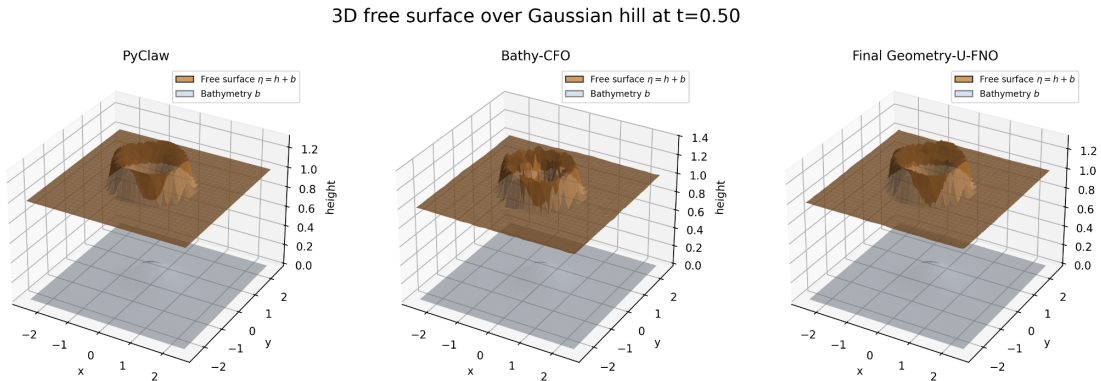


Figure 9: Three-dimensional free surface $\eta = h + b$ over the centered Gaussian bathymetry at $t = 0.50$. PyClaw, Bathy-CFO, and the final Geometry-U-FNO are shown for the same physical configuration.

3D h , hu , hv fields through Gaussian hill
 $t=0.50$, y -section=-0.039, x -section=-0.039

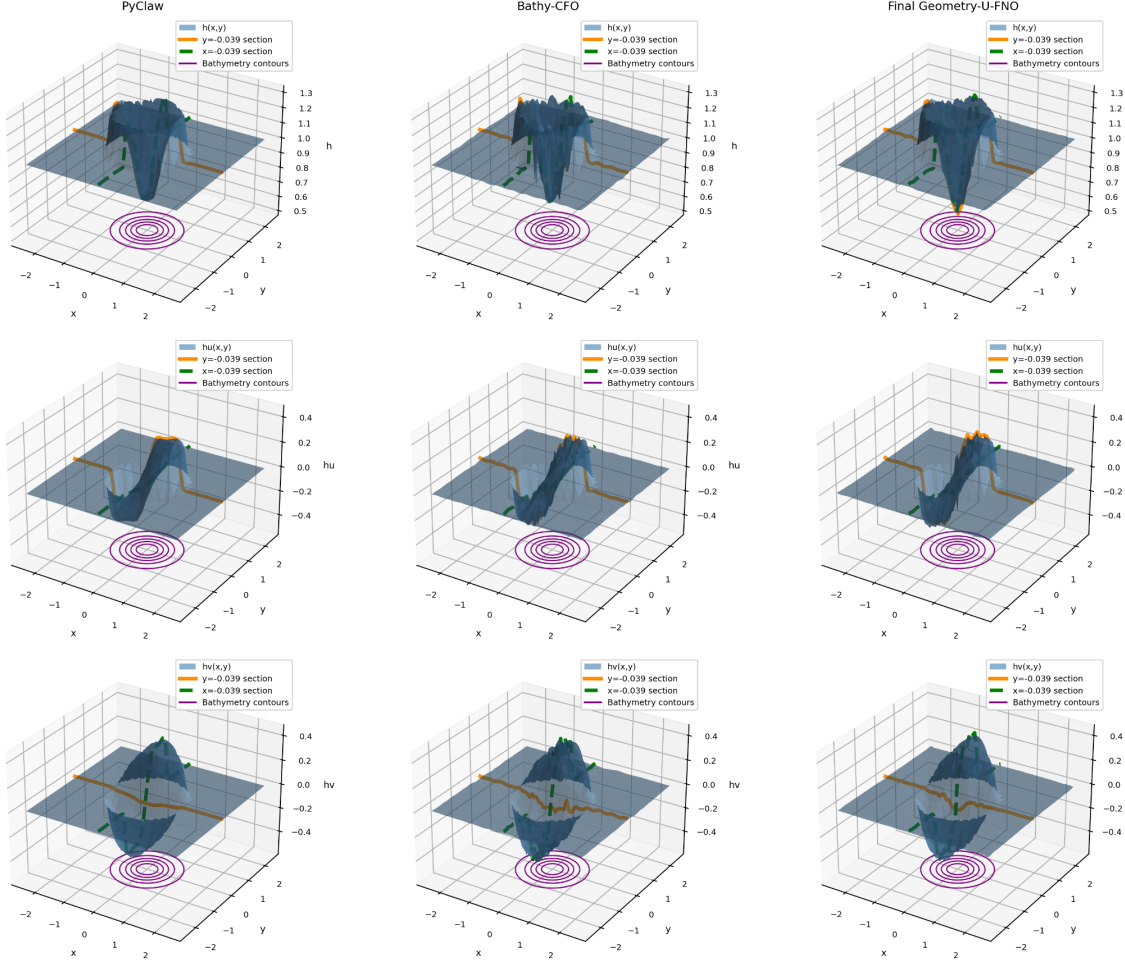


Figure 10: Three-dimensional h , hu , and hv fields through the centered Gaussian hill at $t = 0.50$. Columns compare PyClaw, Bathy-CFO, and the final Geometry-U-FNO. Highlighted paths mark central cross-sections, and projected contours mark the Gaussian bed.

two numerical solutions of the same equations rather than a surrogate against the fine-grid reference. The gap is small for the overall relative error but larger for the terrain-effect error. The canonical model still lands much closer to this numerical resolution-difference level than direct zero-shot Bathy-CFO does. For zero-shot direct bathymetry response, the canonical model reproduces the sign and main spatial location of the fine-grid PyClaw momentum response more clearly than Bathy-CFO. Some local amplitude mismatch remains, but the response is much closer to the expected terrain-conditioned structure. For zero-shot isolated Gaussian-hill effect, the canonical model sharply reduces the large error growth of zero-shot Bathy-CFO. The centered case stays harder than the translated hills, but the error remains much lower over the later part of the rollout. The canonical result is visibly closer to the fine-grid PyClaw response than Bathy-CFO, particularly in the ring-like wave structure and in the reduction of spurious interior oscillations. For zero-shot generalization to unseen bathymetry, on the two-hill and multi-hill cases, the canonical model keeps multiple separated terrain responses on the finer grid. The narrow-tall and rotated-ridge cases stay difficult because of their stronger gradients and directional structure, but the canonical model generally reduces the broad oscillatory artifacts seen in Bathy-CFO. For the bathymetry resolution used in the zero-shot study, this comparison matters for reading the zero-shot test. The physical terrain is unchanged and only the discretization is refined. The canonical U-Net modification aims to preserve the physical scale the local convolutional branch learned, not to alter the terrain.



Figure 11: Zero-shot isolated terrain-effect error for left, center, and right Gaussian hills. PyClaw-128 is the zero-error reference. The PyClaw-64 curve is obtained after interpolation to the 128×128 grid.

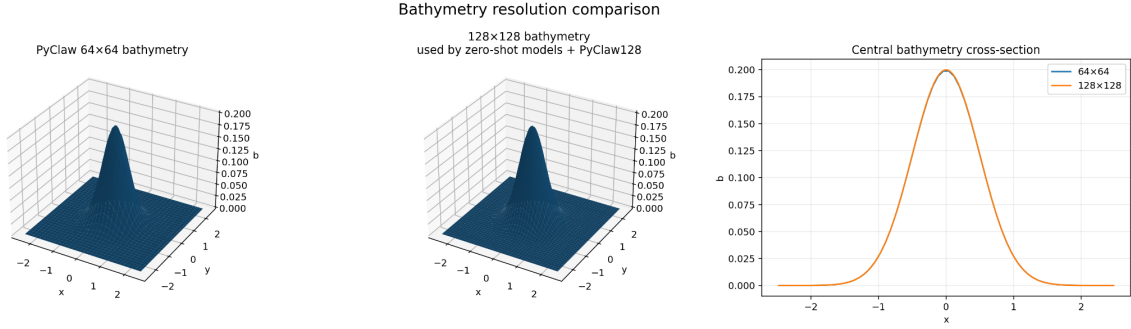


Figure 12: Comparison of the 64×64 and 128×128 Gaussian bathymetries. The central cross-section shows that both grids represent the same physical bed profile with different spatial discretizations.

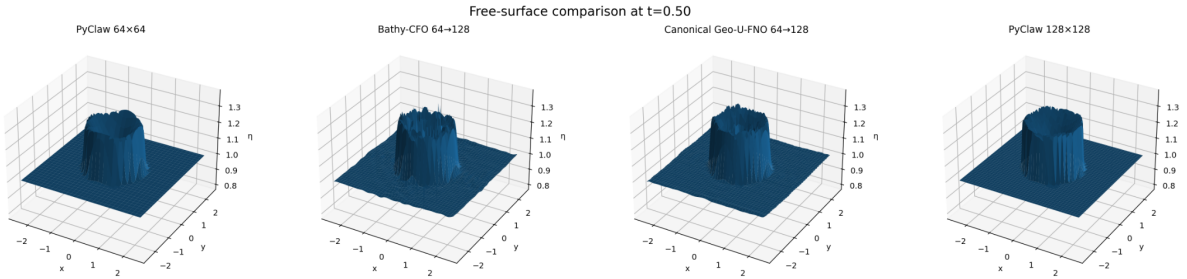


Figure 13: Three-dimensional free-surface comparison at $t = 0.50$ for PyClaw 64×64 , Bathy-CFO $64 \rightarrow 128$, canonical Geometry-U-FNO $64 \rightarrow 128$, and PyClaw 128×128 .

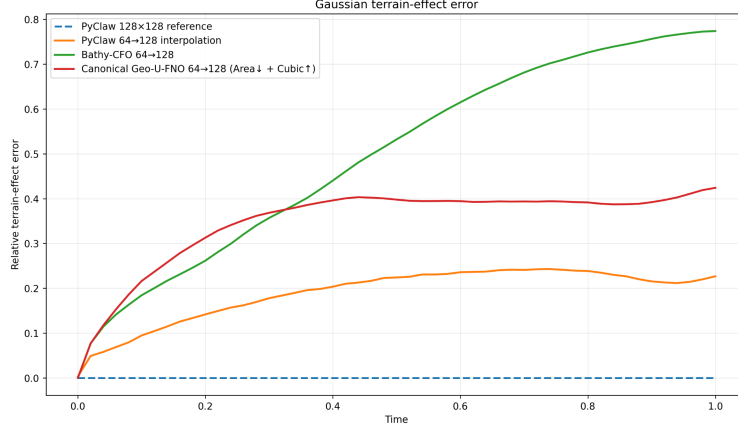


Figure 14: Relative error in the centered-Gaussian terrain effect over the full zero-shot rollout. PyClaw-128 is the reference, PyClaw-64 interpolation shows the numerical resolution difference, and the two learned models are evaluated directly at 128×128 .

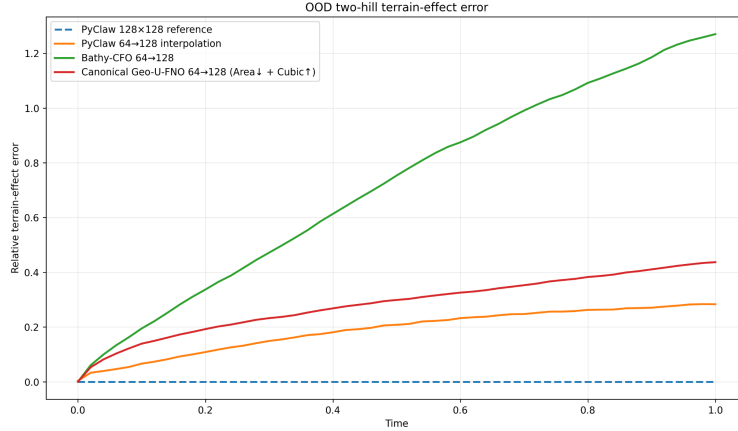


Figure 15: Relative terrain-effect error for the unseen two-hill configuration during zero-shot $64 \rightarrow 128$ rollout. PyClaw-128 is the reference, and PyClaw-64 interpolation gives the numerical resolution-difference level.

For zero-shot three-dimensional free surface, the canonical model suppresses much of the irregular fine-grid structure that the direct zero-shot baseline produces, and it keeps the main circular wave pattern of the PyClaw reference. For zero-shot Gaussian terrain-effect error over time, Bathy-CFO error grows throughout the rollout and reaches about 0.77 by the final time. The canonical Geometry-U-FNO rises early in the rollout and then stays near 0.4 for much of the trajectory, so the canonical local branch limits the accumulation of cross-resolution error. The gap is largest for the unseen two-hill geometry. Bathy-CFO error grows above one near the end of the rollout, while the canonical Geometry-U-FNO stays below about 0.45. This supports using the canonical U-Net as a cross-resolution stabilization mechanism, not just a visual smoothing step. The canonical model keeps the large-scale circular wave structure and reduces the nonphysical local oscillations in the Bathy-CFO prediction. The remaining mismatch sits mainly near steep wave fronts and in the interior terrain-response region.

5.6 Discussion

The experiments show two separate sources of instability that need different fixes.

The first appears at the native training resolution. One-step predictions can be accurate while a free rollout steadily worsens, because every predicted state becomes the input to the next prediction. More RK4 steps cannot fix a biased or unstable learned vector field. The short-horizon rollout loss targets the problem directly by training on the recursive prediction mechanism.

The second appears only when the 64×64 U-FNO moves to 128×128 . The Fourier branch can use the finer grid, but the local U-Net kernel then covers a different physical neighborhood[4]. The canonical strategy separates the two roles:

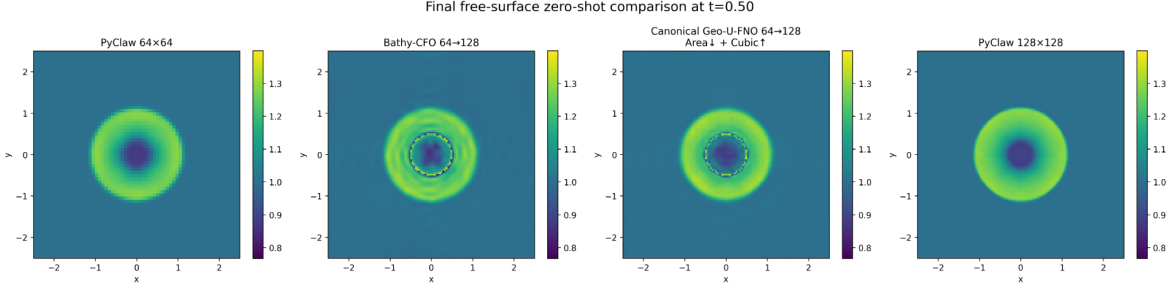


Figure 16: Two-dimensional free-surface comparison at $t = 0.50$. The panels show PyClaw 64×64 , Bathy-CFO $64 \rightarrow 128$, canonical Geometry-U-FNO $64 \rightarrow 128$ with area downsampling and cubic reconstruction, and PyClaw 128×128 .

global Fourier processing and geometry encoding stay on the fine grid, and the local U-Net runs at the spatial scale it was trained on.

The canonical method is not an output interpolation scheme. The sequence

$$128 \rightarrow 64 \rightarrow \mathcal{U}_{64} \rightarrow 128$$

runs inside the vector field at every RK4 evaluation, so the improved local response affects the whole temporal trajectory. The cubic reconstruction results also explain why we dropped the native 128×128 U-Net blend. Even a small direct fine-grid contribution partly brings back the scale mismatch that caused the original zero-shot oscillations. Likewise, going from four to eight RK4 steps did not materially improve the cross-resolution metrics, which points to spatial operator transfer, not ODE integration error, as the main remaining source of discrepancy.

The final model is not perfectly resolution invariant. The PyClaw-64 interpolation error stays below the learned zero-shot error, and the gap is wider for the terrain effect than for the full state. A fairer description is that Geometry-U-FNO generalizes stably and usefully across resolutions once its resolution-sensitive local branch runs on the canonical training grid.

6 Conclusion and Future Work

We extended the Continuous Flow Operator into a geometry-aware, well-balanced, rollout-stable, physics-informed model for two-dimensional shallow-water dynamics over variable bathymetry.

The evolving physical state remains $[h, hu, hv]$, and terrain enters separately through $[b, b_x, b_y]$. The final Geometry-U-FNO combines the original CFO flow-matching objective with the variable-bottom shallow-water residual, a counterfactual bed-response loss, and a lake-at-rest well-balanced constraint.

Recursive rollout error was a major practical difficulty. Local one-step predictions were much better than long free trajectories, which means small model errors accumulated as predictions were fed back into the network. We therefore fine-tune the final 64×64 checkpoint with a short differentiable RK4 rollout objective, using $\lambda_{\text{roll}} = 0.10$, horizon $K = 2$, and four RK4 substeps.

At 64×64 , the final model reduces the ID relative error from 0.036128 for Bathy-CFO to 0.015235. The mean Gaussian terrain-effect error falls from 0.628432 to 0.238144, and the mean OOD terrain-effect error from 0.875895 to 0.318635. The final ID mass drift falls from 5.443×10^{-3} to 1.076×10^{-3} .

Zero-shot spatial refinement brought a second difficulty. Applying the 64×64 local U-Net branch directly on a 128×128 grid changes its physical receptive field and produces large local oscillations. The canonical strategy area-averages the latent field to 64×64 , evaluates the same trained U-Net at its original scale, and reconstructs its response at 128×128 by cubic interpolation. The Fourier and geometry branches stay on the fine grid, and no 128×128 training is introduced.

For the centered Gaussian case, the canonical model reaches a zero-shot relative L_2 error of 0.028191, versus 0.035048 for Bathy-CFO. For the unseen two-hill case, the errors are 0.028692 and 0.035937. The terrain-effect error also falls from 0.563551 to 0.362361 for the Gaussian case and from 0.817599 to 0.305634 for the two-hill case.

Stable transfer of a neural operator across resolutions appears to require care with the spectral component and also with local branches whose physical receptive field changes with the grid.

Future work will treat time-dependent geometry, including sediment transport and SWE–Exner systems,

$$b = b(x, y, t),$$

where the bed becomes part of the coupled dynamics. Other extensions include wetting-and-drying interfaces, moving shorelines, and dynamically varying terrain. A longer-term goal is extension to higher-dimensional Navier–Stokes systems with more complex geometry-dependent interactions[12].

Acknowledgment

The authors gratefully acknowledge Dr. Natesan Srinivasan of the Department of Mathematics, Indian Institute of Technology Guwahati, for his valuable guidance, insightful discussions, and constructive feedback throughout this research. His expertise and encouragement have contributed to the development and refinement of the mathematical and computational aspects of this work.

Data availability

The datasets used in this study include data obtained from the original Continuous Flow Operator (CFO) project and additional datasets generated through numerical simulations using computational codes developed for this research. The newly generated datasets are reproducible using the corresponding simulation procedures and configurations described in this work. **The GitHub link is:** <https://github.com/ronitbaishya1/Pi-CFO>

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the research, methodology, results, or conclusions presented in this paper.

AI Use Disclosure

During the preparation of this manuscript, the authors used generative AI tools, including ChatGPT (OpenAI) and Claude (Anthropic), to assist with code development and debugging, clarification of technical concepts, and English grammar and language refinement. All AI-assisted content was reviewed and revised by the authors, who retain full responsibility for the scientific accuracy, originality, methodology, results, and final content of this work.

References

- [1] 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 Journal of Data Science*, 1(3):1–27, 2024. doi:10.1145/3648506.
- [2] Hanru Bai, Yuncheng Zhou, and Difan Zou. Physics-informed neural pde solvers via spatio-temporal meanflow, 2026.
- [3] Abhishek Mishra, Cosmin Anitescu, Pattabhi Ramaiah Budarapu, Sundararajan Natarajan, Pandu Ranga Vundavilli, and Timon Rabczuk. An artificial neural network based deep collocation method for the solution of transient linear and nonlinear partial differential equations. *Frontiers of Structural and Civil Engineering*, 18(8):1296–1310, 2024. doi:10.1007/s11709-024-1011-4.
- [4] Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M. Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022. doi:10.1016/j.advwatres.2022.104180.
- [5] Xianglong Hou, Xinquan Huang, and Paris Perdikaris. Cfo: Learning continuous-time pde dynamics via flow-matched neural operators. In *International Conference on Learning Representations*, 2026.
- [6] Robert Keppler, Rasmus Fensholt, and Julian Koch. Learning 2d shallow water equations with physics-informed neural operator networks. *Water Resources Research*, 62(8):e2025WR042156, 2026. doi:10.1029/2025WR042156.
- [7] Xingkai Li, Yuanjun Jiang, Yu Zhuang, Xiaobo Hu, and Binbin Zhao. A physics-informed multi-scale fourier neural operator framework for snow avalanche dynamics simulation. *Journal of Hydrology*, 675:135573, 2026. doi:10.1016/j.jhydrol.2026.135573.
- [8] Abdolmehdi Behrooz, Kathryn Lawson, and Chaopeng Shen. Physically anchored multi-resolution neural operator framework for flood inundation prediction. *EGU sphere*, 2026. doi:10.5194/egusphere-2026-1982. Preprint.
- [9] Giacomo Baldan, Qiang Liu, Alberto Guardone, and Nils Thuerey. Physics vs distributions: Pareto optimal flow matching with physics constraints. In *International Conference on Learning Representations*, 2026.
- [10] Juncai He, Xinliang Liu, and Jinchao Xu. Self-composing neural operators for high-frequency and multiscale pde surrogates. *Journal of Computational Physics*, 565:115189, 2026. doi:10.1016/j.jcp.2026.115189.
- [11] Zhen Zhang and George Em Karniadakis. Well-posedness of neural turbulence closures and tangent dissipation, 2026.

- [12] Viktor Grimm, Alexander Heinlein, and Axel Klawonn. Learning the solution operator of two-dimensional incompressible navier–stokes equations using physics-aware convolutional neural networks. *Journal of Computational Physics*, 535:114027, 2025. doi:10.1016/j.jcp.2025.114027.
- [13] Difeng Cai and Paulina Sepúlveda. Data-driven self-supervised learning for the discovery of solution singularity for partial differential equations. *Journal of Computational Physics*, 554:114751, 2026. doi:10.1016/j.jcp.2026.114751.
- [14] N. Benjamin Erichson, Lionel Mathelin, Zhewei Yao, Steven L. Brunton, Michael W. Mahoney, and J. Nathan Kutz. Shallow neural networks for fluid flow reconstruction with limited sensors. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2238):20200097, 2020. doi:10.1098/rspa.2020.0097.