Supplemental Materials

A Nomenclature

Table 2 summarizes the main symbols and notation used in this work.

$C(A, B)$	Space of continuous functions from a space A to a space B.
L^2	Hilbert space of square integrable functions.
$\mathcal{X}$	Domain for input functions, subset of $\mathbb{R}^{d_x}$.
$\mathcal{Y}$	Domain for output functions, subset of $\mathbb{R}^{d_y}$.
x	Input function arguments.
y	Output function arguments (queries).
u	Input function in $C(\mathcal{X}, \mathbb{R}^{d_u})$.
s	Output function in $C(\mathcal{Y}, \mathbb{R}^{d_s})$.
n	Latent dimension for solution manifold
$\mathcal{F}, \mathcal{G}$	Operator mapping input functions u to output functions s.

Table 2: (Nomenclature) A summary of the main symbols and notation used in this work.

B Experimental Details

B.1 Data-set generation

For all experiments, we use N_{train} number of function pairs for training and N_{test} for testing. m and P number of points where the input and output functions are evaluated, respectively. See Table 4 for the values of these parameters for the different examples along with batch sizes and total training iterations. We train and test with the same data-set on each example for both NOMAD and DeepONet.

We build collections of measurements for each of the N input/output function pairs, (u^i, s^i) as follows. The input function is measured at m locations $x_1^i, \dots, x_m^i$ to give the point-wise evaluations, $\{u^i(x_1^i), \dots, u^i(x_m^i)\}$. The output function is evaluated at P locations $y_1^i, \dots, y_P^i$, with these locations potentially varying over the data-set, to give the point-wise evaluations $\{s^i(y_1^i), \dots, s^i(y_P^i)\}$. Each data pair used in training is then given as $(\{u^i(x_j^i)\}_{j=1}^m, \{s^i(y_\ell^i)\}_{\ell=1}^P)$.

B.2 Antiderivative

We approximate the antiderivative operator

$$\mathcal{G} : u \mapsto s(x) := \int_0^x u(y)\, dy,$$

acting on a set of input functions

$$\mathcal{U} := \left\{ u(x) = 2\pi t \cos(2\pi t x) \;\middle|\; 0 \le t_0 \le t \le T \right\}.$$

The set of output functions is given by $\mathcal{G}(\mathcal{U}) = \{\sin(2\pi t x) \mid 0 < t_0 < t < T\}$. We consider $x \in \mathcal{X} = [0, 1]$ and the initial condition $s(0) = 0$. For a given forcing term u the solution operator returns the antiderivative $s(x)$. Our goal is to learn the solution operator $\mathcal{G} : C(\mathcal{X}, \mathbb{R}) \to C(\mathcal{Y}, \mathbb{R})$. In this case $d_x = d_y = d_s = d_u = 1$.

To construct the data-sets we sample input functions $u(x)$ by sampling $t \sim \mathcal{U}(0, 10)$ and evaluate these functions on $m = 500$ equispaced sensor locations. We measure the corresponding output functions on $P = 500$ equispaced locations. We construct $N_{train} = 1,000$ input/output function pairs for training and $N_{test} = 1,000$ pairs for testing the model.

B.3 Advection Equation

For demonstrating the benefits of our method, we choose a linear transport equation benchmark, similar to [14],

$$\frac{\partial}{\partial t}s(x,t) + c\frac{\partial}{\partial x}s(x,t) = 0, \tag{20}$$

with initial condition

$$s_0(x) = s(x,0) = \frac{1}{\sqrt{0.0002\pi}} \exp\Big(-\frac{(x-\mu)^2}{0.0002}\Big), \tag{21}$$

where μ is sampled from a uniform distribution $\mu \sim \mathcal{U}(0.05, 1)$. Here we have $x \in \mathcal{X} := [0,2]$, and $y = (x,t) \in \mathcal{Y} := [0,2] \times [0,1]$. Our goal is to learn the solution operator $\mathcal{G} : \mathcal{C}(\mathcal{X}, \mathbb{R}) \to \mathcal{C}(\mathcal{Y}, \mathbb{R})$. The advection equation admits an analytic solution

$$s(x,t) = s_0(x - ct, t), \tag{22}$$

where the initial condition is propagated through the domain with speed c, as shown in Figure 4a.

We construct training and testing data-sets by sampling $N_{train} = 1,000$ and $N_{test} = 1,000$ initial conditions and evaluate the analytic solution on $N_t = 100$ temporal and $N_x = 256$ spatial locations. We use a high spatio-temporal resolution for training the model to avoid missing the narrow travelling peak in the pointwise measurements.

B.4 Shallow Water Equations

The shallow water equations are a hyperbolic system of equations that describe the flow below a pressure surface, given as

$$\begin{aligned}
&\frac{\partial \rho}{\partial t} + \frac{\partial(\rho v_1)}{\partial x_1} + \frac{\partial(\rho v_2)}{\partial x_2} = 0,\\
&\frac{\partial(\rho v_1)}{\partial t} + \frac{\partial}{\partial x_1}(\rho v_1^2 + \frac{1}{2}g\rho^2) + \frac{\partial(\rho v_1 v_2)}{\partial x_2} = 0, \qquad t \in (0,1], x \in (0,1)^2\\
&\frac{\partial(\rho v_2)}{\partial t} + \frac{\partial(\rho v_1 v_2)}{\partial x_1} + \frac{\partial}{\partial x_2}(\rho v_2^2 + \frac{1}{2}g\rho^2) = 0,
\end{aligned} \tag{23}$$

where ρ is the total fluid column height, v_1 the velocity in the x_1-direction, v_2 the velocity in the x_2-direction, and g the acceleration due to gravity.

We consider impenetrable reflective boundaries

$$v_1 \cdot n_{x_1} + v_2 \cdot n_{x_2} = 0,$$

where $\hat{n} = n_{x_1}\hat{i} + n_{x_2}\hat{j}$ is the unit outward normal of the boundary.

Initial conditions are generated from a droplet of random width falling from a random height to a random spatial location and zero initial velocities

$$\begin{aligned}
&\rho = 1 + h \exp\big(-((x_1 - \xi)^2 + (x_2 - \zeta)^2)/w\big)\\
&v_1 = v_2 = 0,
\end{aligned}$$

where h corresponds to the altitude that the droplet falls from, w the width of the droplet, and ξ and ζ the coordinates that the droplet falls in time $t = 0s$. Instead of choosing the solution for v_1, v_2 at time $t_0 = 0s$ as the input function, we use the solution at $dt = 0.002s$ so the input velocities are not always zero. The components of the input functions are then

$$\begin{aligned}
&\rho = 1 + h \exp\big(-((x_1 - \xi)^2 + (x_2 - \zeta)^2)/w\big)\,,\\
&v_1 = v_1(dt, y_1, y_2),\\
&v_2 = v_2(dt, y_1, y_2).
\end{aligned}$$

We set the random variables h, w, ξ, and ζ to be distributed according to the uniform distributions

$$\begin{aligned}
h &= \mathcal{U}(1.5, 2.5),\\
w &= \mathcal{U}(0.002, 0.008),\\
\xi &= \mathcal{U}(0.4, 0.6),\\
\zeta &= \mathcal{U}(0.4, 0.6).
\end{aligned}$$

In this example, $x \in \mathcal{X} := (0,1)^2$ and $y = (x,t) \in (0,1)^2 \times (0,1]$. For a given set of input functions, the solution operator $\mathcal{G}$ of [23] maps the fluid column height and velocity fields at time dt to the fluid column height and velocity fields at later times. Therefore, our goal is to learn a solution operator $\mathcal{G} : \mathcal{C}(\mathcal{X}, \mathbb{R}^3) \to \mathcal{C}(\mathcal{Y}, \mathbb{R}^3)$.

We create a training and a testing data-set by sampling $N_{train} = 1,000$ and $N_{test} = 1,000$ input/output function samples by sampling initial conditions on a 32×32 grid, solving the equation using a Lax-Friedrichs scheme [34] and considering five snapshots $t = [0.11, 0.16, 0.21, 0.26, 0.31]s$. We randomly choose $P = 128$ measurements from the available spatio-temporal data of the output functions per data pair for training.

C Comparison Metrics

Throughout this work, we employ the relative $\mathcal{L}_2$ error as a metric to assess the test accuracy of each model, namely

$$\text{Test error metric} = \frac{||s^i(y) - \hat{s}^i(y)||_2^2}{||s^i(y)||_2^2},$$

where $\hat{s}(y)$ the model predicted solution, $s(y)$ the ground truth solution and i the realization index. The relative $\mathcal{L}_2$ error is computed across all examples in the testing data-set, and different statistics of this error vector are calculated: the mean and standard deviation. For the Shallow Water Equations where we train on a lower resolution of the output domain, we compute the testing error using a full resolution grid.

D Architecture Choices and Hyper-parameter Settings

We first present details on the architecture and hyperparameters used to train NOMAD and the DeepONet for the antiderivative and parametric advection experiments.

In the DeepONet, the approximation map $\mathcal{A} : \mathbb{R}^m \to \mathbb{R}^n$ is known as the branch network b, and the neural network whose outputs are the basis $\{\tau_1, \dots, \tau_n\}$ is known as the trunk network, τ. We use an MLP for both b and τ and present the architecture details in Table 3. The DeepONet used for the antiderivative and parametric advection experiments is the plain DeepONet version originally put forth in [30], without considering the improvements in [32]. The reason for choosing the simplest architecture possible is because we are interest in examining solely the effect of the decoder without any additional moving parts. The NOMAD architecture for these two experiments is the same as the DeepONet. The activation function for all MLPs is the GeLU activation [19].

Table 3: Architecture choices for different examples.

Example	b depth	b width	τ depth	τ width
Antiderivative	5	100	5	100
Parametric Advection	5	100	5	100

Next, we present all architecture choices and training details used in the free-surface wave experiment for the NOMAD, DeepONet, FNO, and LOCA methods.

For both NOMAD and DeepONet, we set the batch size of input and output pairs equal to 100. We consider an initial learning rate of $l_r = 0.001$, and an exponential decay with decay-rate of 0.99 every 100 training iterations unless otherwise stated.

For the results presented in Table 1, we consider 5 hidden layers and 100 neurons for the branch and the trunk for NOMAD. For the LOCA architecture, we consider one hidden layer deep MLPs with 1024 hidden neurons, GeLU [19] activation functions, number of harmonic features $H = 2$, batch size 100, dimensionality of the encoder $l = 100$. We train for 80000 iteration with learning rate $l_r = 0.001$ and an exponential decay with decay-rate of 0.95 every 100 training iterations. We consider $J = 1$ log-2 scatteting scales, $L = 3$ angles for the wavelet transform and $m_o = 2$ maximum order of scattering coefficients to compute and a Monte Carlo integration. We consider the same values for m and P as in Table 4. For the FNO architecture, we consider 8 modes and a width of

Table 4: Training details for the experiments in this work. We present the number of training and testing data pairs N_{train} and N_{test}, respectively, the number of sensor locations where the input functions are evaluated m, the number of query points where the output functions are evaluated P, the batch size, and total training iterations.

Example	N_{train}	N_{test}	m	P	Batch #	Train iterations
Antiderivative	1000	1000	500	500	100	20000
Parametric Advection	1000	1000	256	25600	100	20000
Free Surface Waves	1000	1000	1024	128	100	100000

25 for the FNO layers, as well as 4 FNO layers. We train for 400 epochs using a batch size that is equal to 100, a learning rate $l_r = 0.001$, which we then reduce by 0.5 every 100 epochs and a weight decay of 0.0001. We consider the same m, P values as in Table 4. For the DeepONet architecture, we consider the branch b and the trank τ as MLPs with 11 hidden layers each consisting of 100 neurons and train for 80000 iterations. Moreover we include the harmonic feature expansion for the DeepONet inputs proposed in [31].