nature machine intelligence

Supplementary information

https://doi.org/10.1038/s42256-024-00844-4

Laplace neural operator for solving differential equations

In the format provided by the authors and unedited

Supplementary Information

S1 Derivation to illustrate why LNO can learn two parts of the transient response

In this section, we provide a derivation to demonstrate LNO's capability to learn both components of the transient response and the process of incorporating initial conditions into the method. FNO as well as LNO are based on the solution of partial differential equations represented by the integral of Green's function. To illustrate, we employ an example of an undamped single-degree-freedom system. The equation of motion for an ordinary differential equation system is expressed as:

$$\ddot{x}(t) + \bar{\omega}^2 x(t) = f(t), \tag{1}$$

$$\text{subject to:} \quad x(t=0) = x_0 \text{and} \;\; \dot{x}(t=0) = v_0, \tag{2}$$

where $x(t)$ is a signal in the time domain, $\bar{\omega}$ is the natural frequency, and $f(t)$ is the source term. Performing the Laplace transform yields:

$$(s^2 + \bar{\omega}^2)\tilde{x}(s) = \tilde{f}(s) + sx_0 + v_0, \tag{3}$$

where $s = \sigma + i\omega$ is a complex frequency domain parameter and σ and ω are real numbers. Then,

$$\tilde{x}(s) = \tilde{h}(s)\tilde{f}(s) + \tilde{h}(s)[sx_0 + v_0], \tag{4}$$

where the transfer function is

$$\tilde{h}(s) = \frac{1}{s^2 + \bar{\omega}^2} = \frac{\beta}{s - i\bar{\omega}} + \frac{\beta^*}{s + i\bar{\omega}}\,, \tag{5}$$

in which β and $\beta^* = \pm i/(2\bar{\omega})$ are system residues and $\pm i\bar{\omega}$ are system poles. Let $s = i\omega$, then Eq. 5 can be re-written as

$$H(\omega) = \frac{1}{\bar{\omega}^2 - \omega^2}, \tag{6}$$

where $H(\omega)$ is the complex frequency response function (FRF). Eq. 4 can be categorized into two parts [1]:

$$\tilde{x}(s) = \tilde{x}_1(s) + \tilde{x}_2(s), \tag{7}$$

where

$$\tilde{x}_1(s) = \tilde{h}(s)\tilde{f}(s), \tag{8}$$

and

$$\tilde{x}_2(s) = \tilde{h}(s)[sx_0 + v_0]. \tag{9}$$

For an arbitrary external force, $f(t)$, it can be expressed as a complex Fourier series such that

$$f(t) = \sum_{m=-\infty}^{\infty} C_m e^{i\Omega_m t} \tag{10}$$

where C_m is the m^{th} complex Fourier coefficient, and $\Omega_m = m\Omega_1$, Ω_1 is the fundamental frequency. Performing Laplace transform of Eq. 10, results in

$$\tilde{f}(s) = \sum_{m=-\infty}^{\infty} \frac{C_m}{s - i\Omega_m}. \tag{11}$$

Substituting s by $i\omega$ in Eq. 11 leads to the corresponding frequency representation:

$$F(\omega) = \sum_{m=-\infty}^{\infty} \frac{C_m}{i\omega - i\Omega_m} \tag{12}$$

Substituting Eqs. 5 and 11 into Eq. 8 yields

$$\tilde{x}_1(s) = \left(\frac{\beta}{s - i\bar{\omega}} + \frac{\beta^*}{s + i\bar{\omega}}\right) \sum_{m=-\infty}^{\infty} \frac{C_m}{s - i\Omega_m}. \tag{13}$$

Since the common denominator in Eq. 13 is the product of $(s - i\bar{\omega})$, $(s + i\bar{\omega})$ and $(s - i\Omega_m)$, Eq. 13 can be re-written in a partial fraction form as

$$\tilde{x}_1(s) = \left(\frac{V}{s - i\bar{\omega}} + \frac{V^*}{s + i\bar{\omega}}\right) + \sum_{m=-\infty}^{\infty} \frac{U_m}{s - i\Omega_m}, \tag{14}$$

where V and V^* are response residues corresponding to the excitation poles; and U_m are response residues corresponding to the system poles. From Eq. 14, V can be obtained by [2]

$$V = \lim_{s \to i\bar{\omega}} (s - i\bar{\omega})\tilde{x}_1(s). \tag{15}$$

Substituting Eq. 13 into Eq. 15, we get

$$V = \beta\, \tilde{f}(i\bar{\omega}), \tag{16}$$

where

$$\tilde{f}(i\bar{\omega}) = \sum_{m=-\infty}^{\infty} \frac{C_m}{i\bar{\omega} - i\Omega_m}. \tag{17}$$

Similarly, U_m can be obtained by

$$U_m = \lim_{s \to i\Omega_m} (s - i\Omega_m)\tilde{x}(s), \tag{18}$$

and then

$$U_m = C_m\, \tilde{h}(i\Omega_m), \tag{19}$$

or

$$U_m = C_m\, H(\Omega_m). \tag{20}$$

Once V and U_m are computed, by taking the inverse Laplace transform of Eq. 14, we get

$$x_1(t) = [V \exp(i\bar{\omega}t) + V^* \exp i\bar{\omega}^* t)] + \sum_{m=-\infty}^{\infty} U_m \exp(i\Omega_m t), \tag{21}$$

where $x_1(t) = \mathscr{L}^{-1}\{\tilde{x}_1(s)\}$ is the response under the zero initial conditions. For calculating $\tilde{x}_2(s)$, one can write Eq. 9 into a partial fraction form

$$\tilde{x}_2(s) = \frac{A}{s - i\bar{\omega}} + \frac{A^*}{s + i\bar{\omega}}, \tag{22}$$

where $A = (i\bar{\omega}x_0 + v_0)/2i\bar{\omega}$. By performing the inverse Laplace transform, we get

$$x_2(t) = A \exp(i\bar{\omega}t) + A^* \exp(-i\bar{\omega}t). \tag{23}$$

where $x_2(t) = \mathscr{L}^{-1}\{\tilde{x}_2(s)\}$ is the transient response caused by the non-zero initial conditions. Finally, the total solution is the sum of Eqs. 21 and 23, and it reads as:

$$\begin{aligned} x(t) = &[A\exp(i\bar{\omega}t) + A^*\exp(-i\bar{\omega}t)] + \\ &[V\exp(i\bar{\omega}t) + V^*\exp(i\bar{\omega}^*t)] + \sum_{m=-\infty}^{\infty} U_m \exp(i\Omega_m t). \end{aligned} \tag{24}$$

In Eq. 24, the term, $A\exp(i\bar{\omega}t) + A^*\exp(-i\bar{\omega}t)$ denotes the transient response caused by non-zero initial conditions; while the term, $V\exp(i\bar{\omega}t) + V^*\exp(i\bar{\omega}^*t)$ is the transient response under zero initial conditions. These two transient terms are related to the system poles. The last term in Eq. 24, $\sum_{m=-\infty}^{\infty} U_m \exp(i\Omega_m t)$ is the steady-state response operated in the frequency domain, which can be obtained by performing the inverse Fourier transform of Eq. 20. In summary, FNO employs a linear weight to capture the described transient responses, relying on regression for data fitting. While adept at interpolation, FNO struggles with extrapolation issues, especially when datasets for training and testing involve significantly different source terms, $f(t)$. In contrast, the accuracy of LNO is significantly higher as it independently learns the three components, proving particularly effective in extrapolation scenarios. Furthermore, one can add one parallel part in the Laplace layer to compute the response caused by the non-zero initial conditions based on the above derivation. More details about the derivation and relative knowledge can be read in [3–5].

Next, we summarize the primary differences between FNO and LNO:

1. FNO chooses $K_{\boldsymbol{\phi}}$ to be a neural network parameterized by $\boldsymbol{\phi} = (K_{\boldsymbol{\phi}}(i\omega_1), \cdots, K_{\boldsymbol{\phi}}(i\omega_L))$ in the frequency domain. However, LNO chooses $K_{\boldsymbol{\phi}}$ to be a neural network parameterized by $\boldsymbol{\phi} = (\mu_1, \cdots, \mu_N, \beta_1, \cdots, \beta_N)$ in the Laplace domain, where μ_n and β_n are the trainable system poles and residues, respectively.
2. LNO learns the steady-state response, transient response under zero initial conditions, and transient responses caused by non-zero initial conditions, respectively, which guarantees the true system is learned so that LNO performs better for extrapolation problems. FNO uses a linear weight to capture two types of transient responses, which does not totally learn the system but uses the regression method to fit the data. Thus, FNO is very good at interpolation problems.

When implementing LNO, the process involves three key steps:

1. The input $\mathbf{f}(t)$ undergoes an initial transformation through a linear operation, resulting in a higher-dimensional representation $\mathbf{v}(t)$.
2. $\mathbf{v}(t)$ is then subjected to Laplace layers, involving the following substeps:
   a) Employing a signal decomposition method, such as FFT or the Prony-SS method, to express $v(t)$ in a pole-residue form, yielding $i\omega_\ell$ and α_ℓ.

b) Defining training parameters μ_n and β_n, which are learned from the data.
c) Calculating output residues γ_n and λ_ℓ based on Eq. (27) and Eq. (29).

3. In the final substep, Eq. (31) or Eq. (32) is constructed using μ_n, $i\omega_\ell$, γ_n, and λ_ℓ to obtain the desired output $u(t)$ or $u(x,t)$.

s2 Justification for selecting these benchmarks

In this section, we systematically elucidate the rationale behind the selection of each example:

1. The Duffing and pendulum systems are examined under two conditions: first without a damping term and the second with slight damping, causing transient responses to manifest more gradually. These cases serve as illustrations to showcase LNO's ability to learn transient responses.
2. The exploration of the Lorenz system with a parameter setting of $\rho = 10$ reveals two equilibrium states within its solutions. This deliberate selection affords a valuable opportunity to showcase LNO's efficacy in learning solutions characterized by multiple equilibria.
3. Both the Euler-Bernoulli beam and the diffusion equation exemplify linear operators, strategically chosen to accentuate how the pole-residue formulation accurately represents the responses of linear systems. This choice aims to illustrate the precision and fidelity of LNO in capturing the dynamics of linear operators.
4. The investigation into shallow-water equations serves a comparative purpose, specifically to assess the accuracy of Latent-LNO (L-LNO) against DeepONet and latent-DeepONet (L-DeepONet), where the latent representation is acquired through training an autoencoder. This particular example has been chosen to demonstrate and evaluate the efficacy of LNO in solving high-dimensional problems, offering insights into its capabilities beyond lower-dimensional scenarios.

s3 Data generation

This section introduces the details of the data generation for all ODE and PDE examples.

Duffing oscillator To generate N_{train} samples to train LNO, we consider a sinusoidal function, $f_{train}(t) = A\sin(5t)$, where the amplitude $A \in [0.05, 10]$ with $\delta A = 0.05$. Each sample is discretized into 2048 temporal points and the time interval is $\Delta t = 0.01$ seconds. The response is calculated by the solver **ode45** on MATLAB. For validating and testing the neural operators, we generated datasets considering $f_{test}(t) = Ae^{-0.05t}\sin(5t)$, where the amplitude $A \in [0.14, 9.09]$.

Driven gravity pendulum For the driven gravity pendulum model, we consider the same forcing functions for training, f_{train}, and testing, f_{test} as

used in the last example. Thus, $N_{train} = 200$, $N_{vali} = 50$, and $N_{test} = 130$. The corresponding responses, $x(t)$, are also computed by **ode45**.

Forced Lorenz system The number of samples for training, testing, and validation is kept the same as in the previous two examples.

Euler-Bernoulli beam To generate N_{train} samples to train LNO, we consider a function, $f_{train}(x,t) = Ae^{-0.05x}(1-10^2)\sin(10t)$, where the amplitude $A \in [0.05, 10]$ with an interval $\delta A = 0.05$, therefore $N_{train} = 200$. Each sample is discretized on 50×50 tempo-spatial grid such that the time interval, $\Delta t = 0.02$ seconds, and the spatial interval, $\Delta x = 0.03$ meters. For validating and testing the neural operator, a function, $f_{test}(x,t) = Ae^{-x}(1-10^2)\sin(10t)$, is considered, where the amplitude $A \in [1.24, 10.19]$ and $N_{vali} = 50$ and $N_{test} = 130$. The analytical particular solution of Eq.10 for f_{train} is $y_{train}(x,t) = Ae^{-0.05x}\sin(10t)$, and for f_{test} is $y_{test}(x,t) = Ae^{-x}\sin(10t)$.

Diffusion equation For learning the operator of the system in Eq.11, we consider a function, $f_{train}(x,t) = Ae^{-0.05t}(1-\pi^2)\sin(\pi x)$, where the amplitude $A \in [0.05, 10]$ with an interval $\delta A = 0.05$, therefore $N_{train} = 200$. Each sample is discretized on 50×50 tempo-spatial grid such that the time interval is $\Delta t = 0.01$ seconds and the spatial interval, $\Delta x = 0.08$ meters. For validating and testing the neural operators, we generate a dataset considering the function, $f_{test}(x,t) = Ae^{-t}(1-\pi^2)\sin(\pi x)$, where the amplitude $A \in [1.24, 10.19]$ and $N_{vali} = 50$ and $N_{test} = 130$. The analytical particular solution of Eq.11 for the source terms, f_{train} and f_{test} are $y_{train}(x,t) = Ae^{-0.05t}\sin(\pi x)$ and $y_{test}(x,t) = Ae^{-t}\sin(\pi x)$, respectively.

Reaction-diffusion system To generate N_{train} samples for training LNO, we consider the source term $f_{train}(x,t) = Ae^{-0.05t}(1-\pi^2)\sin(\pi x) + A^2e^{-0.1t}\sin(\pi x)^2$, where $A \in [0.05, 10]$, within the interval, we consider $\delta A = 0.05$, resulting in $N_{train} = 200$ samples. Each sample is discretized on a 50×50 tempo-spatial grid with a time interval of $\Delta t = 0.2$ seconds and a spatial interval of $\Delta x = 0.08$. For validation and testing the neural operators, we have employed the source term $f_{test}(x,t) = Ae^{-t}(1-\pi^2)\sin(\pi x) + A^2e^{-2t}\sin(\pi x)^2$, where $A \in [0.14, 9.09]$ with $N_{vali} = 50$ validation samples and $N_{test} = 130$ testing samples. The analytical particular solution for f_{train} source term in Eq.12 is $y_{train}(x,t) = Ae^{-0.05t}\sin(\pi x)$, and for f_{test} source term, the solution is $y_{test}(x,t) = Ae^{-t}\sin(\pi x)$.

Furthermore, we investigate the capability of LNO to map the input with low resolution to the output with high resolution. The source term $f(t)$ is sampled from a Gaussian random field according to the distribution: $N(0, K(t_1,t_2))$, where $K(t_1,t_2) = \exp[-\frac{|t_1-t_2|^2}{2(0.5)^2}]$. We use 500 samples for training, and 50, and 130 samples for validation and testing. The corresponding response $y(x,t)$ to each input sample is calculated by the PDE solver—**pdepe** on MATLAB. When solving the PDE, the time and spatial intervals are

$\Delta t \times \Delta x$ is 0.05×0.05, respectively. Thus, both the input and output have a size of 21×21. For showing the ability of interpolation, the input of size 21×21 is down-sampled to size 11×11. We aim to map the input $f(t)$ of size 11×11 to the output $y(x, t)$ of size 21×21.

We also test the ability of LNO to predict the output at locations that do not have inputs; that is, we choose the input and output at different locations, which are summarized in Supplementary Section 1. The relative $\mathcal{L}_2$ error ($7.04 \times 10^{-2} \pm 1.89 \times 10^{-4}$) is similar to the situation that maps the low-resolution input to high-resolution output ($7.43 \times 10^{-2} \pm 3.16 \times 10^{-4}$). The last two rows of Fig. S3 display error plots of two typical test samples. Another example—Burger's equation, which is also used to show the above two abilities of LNO—is demonstrated in Supplementary Section 1.

Brusselator reaction-diffusion system For training and testing the operators, slightly different functions $f_{train}(t) = A_{train} e^{-0.01t} \sin(t)$ and $f_{test}(t) = A_{test} e^{-0.05t} \sin(t)$ are considered. The amplitude A belongs to $[0.01, 10]$ with an interval $\delta A = 0.01$, in which random 800 values are used as A_{train}, and 200 values are used as A_{test}. The response $u(\mathbf{x}, t)$ is calculated by the PDE solver—**py-pde** on Python [6]. When solving the PDE, the time interval Δt is 0.02 seconds, and the spatial interval $\Delta x \times \Delta y$ is $1/28 \times 1/28$ meters. When training and testing the operators, the snapshots with the time interval $\Delta t = 0.5$ seconds for each sample are chosen.

Shallow-water equations When generating the training and testing data, the following values are chosen: $\Xi = 7.292 \times 10^{-5}$ s^{-1}, $g = 9.80616$ ms^{-1}, $\nu = 1.0 \times 10^5$ m^{2}s^{-1}, $u_{\max} = 80$ ms^{-1}, $\phi_0 = \pi/7$, $\phi_1 = \pi/2 - \phi_0$, thus the jet's mid-point is applied at $\phi = \pi/4$. Substituting these values into Eq.16, the initial velocity u is obtained. The height field is then numerically calculated by integrating the balance equation based on Gaussian quadrature

$$gh(\phi) = gh_0 - \int^{\phi} \alpha u(\phi') \left[f + \frac{\tan(\phi')}{\alpha} u(\phi') \right] d\phi', \tag{25}$$

in which $\alpha = 6.37122 \times 10^6$ m is the Earth's radius and $h_0 = 10$ km is the mean layer depth around the sphere. For generating different input data, we add a small unbalanced perturbation to the height field h to induce the development of barotropic instability. The localized Gaussian perturbation is represented as:

$$h'(\lambda, \phi, t = 0) = \hat{h} \cos(\phi) \exp[-(\lambda/\alpha)^2] \exp[-(\phi_2 - \phi)/\beta]^2, \tag{26}$$

where $-\pi < \lambda < \pi$, $\phi_2 = \pi/4$ and $\hat{h} = 120$ m define the location of the perturbation. The values $\alpha \sim U[0.\bar{1}, 0.5]$, $\beta \sim U[0.0\bar{3}, 0.2]$, which define the shape of the perturbation, are considered random variables. We generate Datasets using the *Dedalus Project*, which can be found at https://github.com/DedalusProject/dedalus. The simulation domain is a spherical domain $\Omega = [-\pi, \pi] \times [-\pi, \pi]$, which is discretized with $m_s \times m_s = 256 \times 256$ mesh

points in the directions of longitude and latitude, respectively. The time duration is $t = [0, 360h]$ with the time interval $\delta t = 1.\bar{6} \cdot 10^{-1}h$, so that $m_t = 72$ time steps are considered. To simply present the results, the time range is mapped to the dimensionless range $t = [0, 1]$. In total, $N_{\text{train}} = 230, N_{\text{test}} = 70$ are generated for training and testing, respectively.

S4 Additional results

Fig. S1 presents the error plots of two representative test samples drawn from three ODE experiments. Fig. S2 presents the error plots of two representative test samples drawn from three PDE experiments. Fig. S3 shows pointwise error plots of responses for two scenarios drawn from reaction-diffusion experiments. Scenario 1 represents the mapping from low-resolution input to high-resolution output; Scenario 2 represents the mapping whose input and output are at different locations. Fig. S4 displays error plots of two typical test samples drawn from the Brusselator reaction-diffusion experiment.

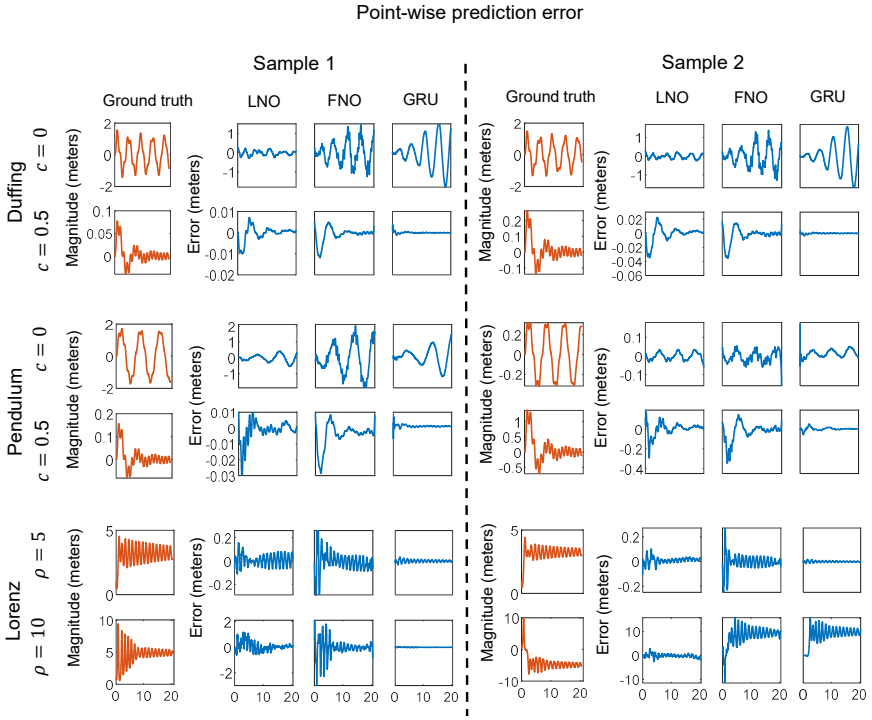

Fig. S1 Pointwise error plots of responses for two representative test samples drawn from three ODE experiments. The ground truth is plotted by red curves and the pointwise error for LNO, FNO, and GRU are presented by blue curves.

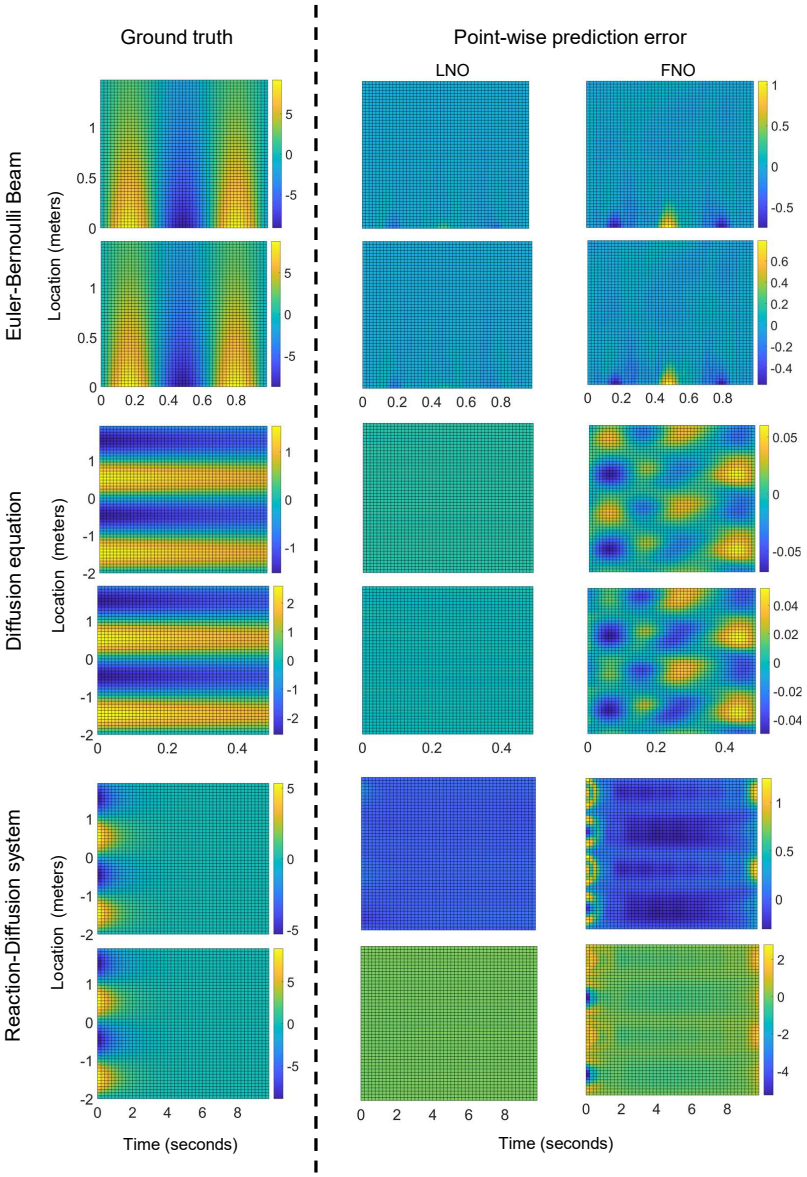

Fig. S2 Pointwise error plots of responses for two representative test samples drawn from three PDE experiments. The ground truth is plotted in the left column and the point-wise errors for LNO and FNO are presented in the right section.

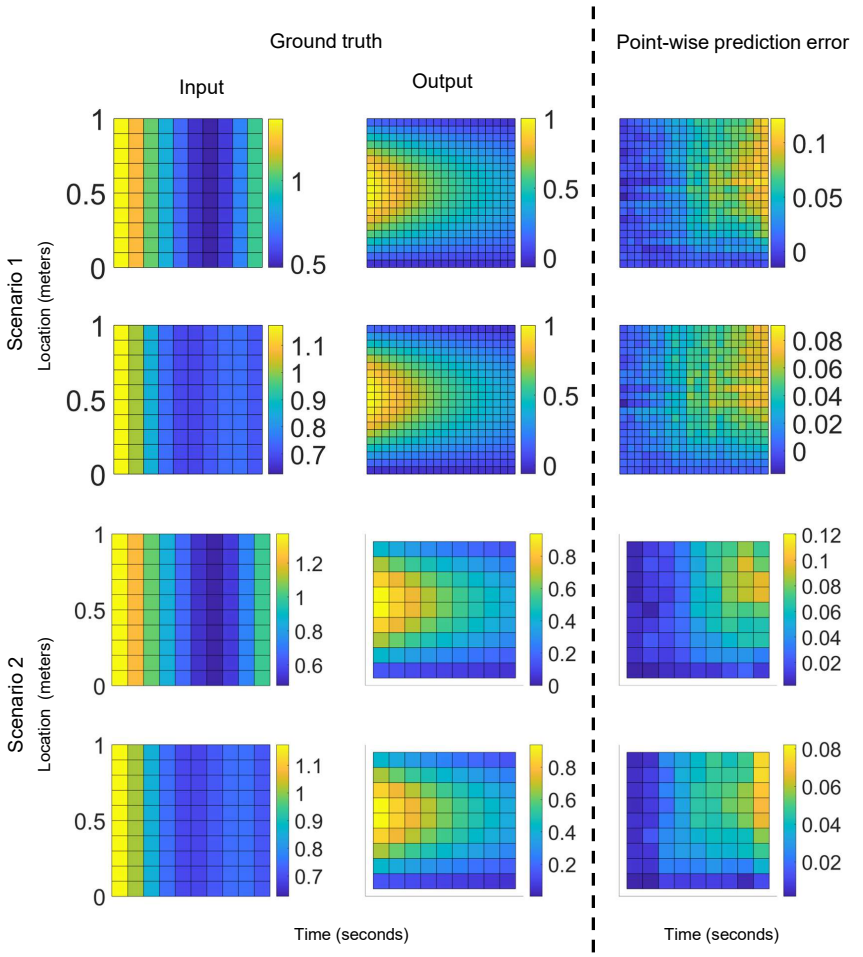

Fig. S3 Pointwise error plots of responses for two scenarios (each scenario includes two representative test samples) drawn from reaction-diffusion experiments. Scenario 1 represents the mapping from low-resolution input to high-resolution output; Scenario 2 represents the mapping whose input and output are at different locations.

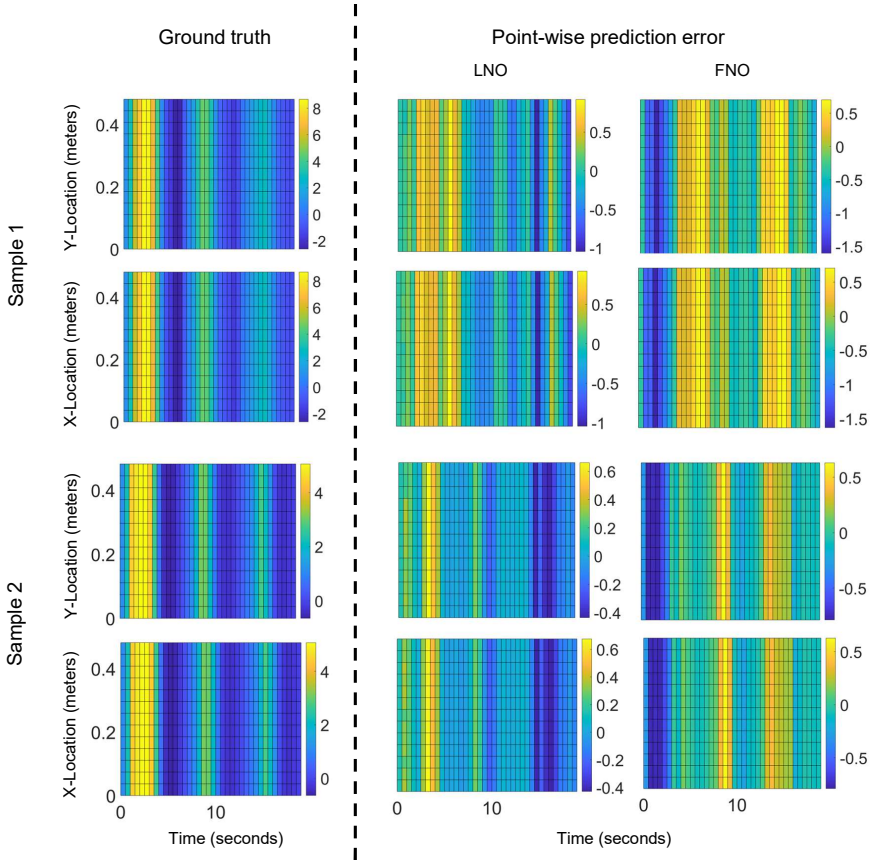

Fig. S4 Pointwise error plots of responses for two representative test samples drawn from the Brusselator reaction-diffusion experiment. The ground truth is plotted in the left column and the point-wise errors for LNO and FNO are presented in the right section.

S5 Comparing neural operators

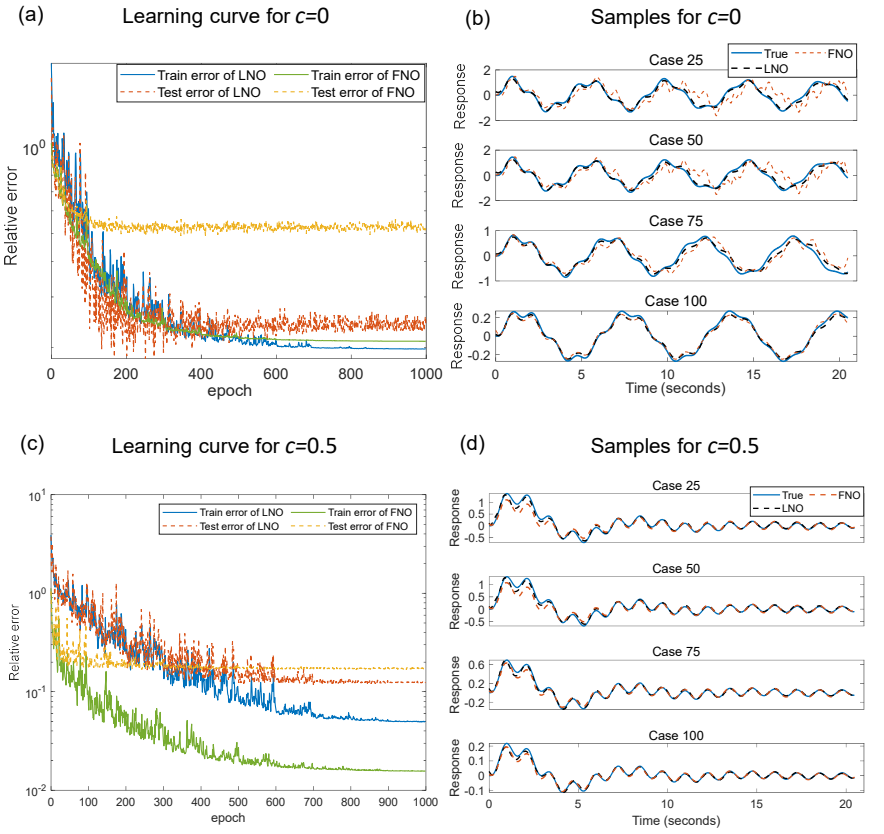

Fig. S5 Duffing oscillator: Comparison of training and testing losses and responses obtained using LNO and FNO: (a) learning curve of the system without damping, (b) representative response obtained from the system without damping, for test cases, (c) learning curve of the system with damping $c = 0.5$, (d) representative response obtained from the system with damping $c = 0.5$, for test cases.

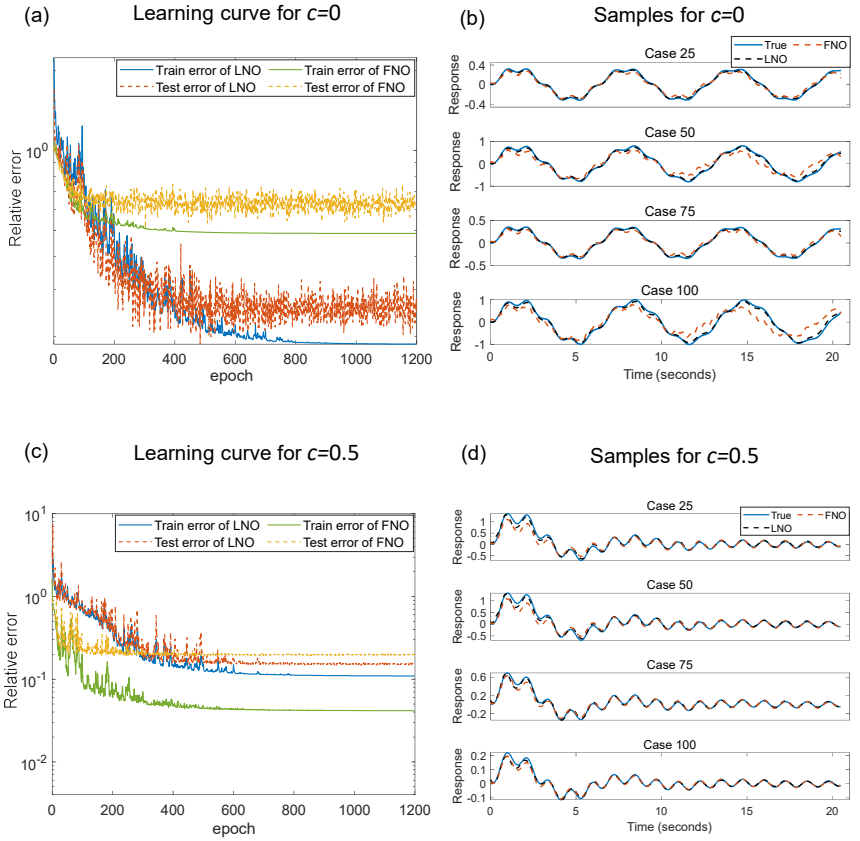

Fig. S6 Driven gravity pendulum: Comparison of training and testing losses and responses obtained using LNO and FNO: (a) learning curve of the system without damping, (b) representative response obtained from the system without damping, for test cases, (c) learning curve of the system with damping $c = 0.5$, (d) representative response obtained from the system with damping $c = 0.5$, for test cases.

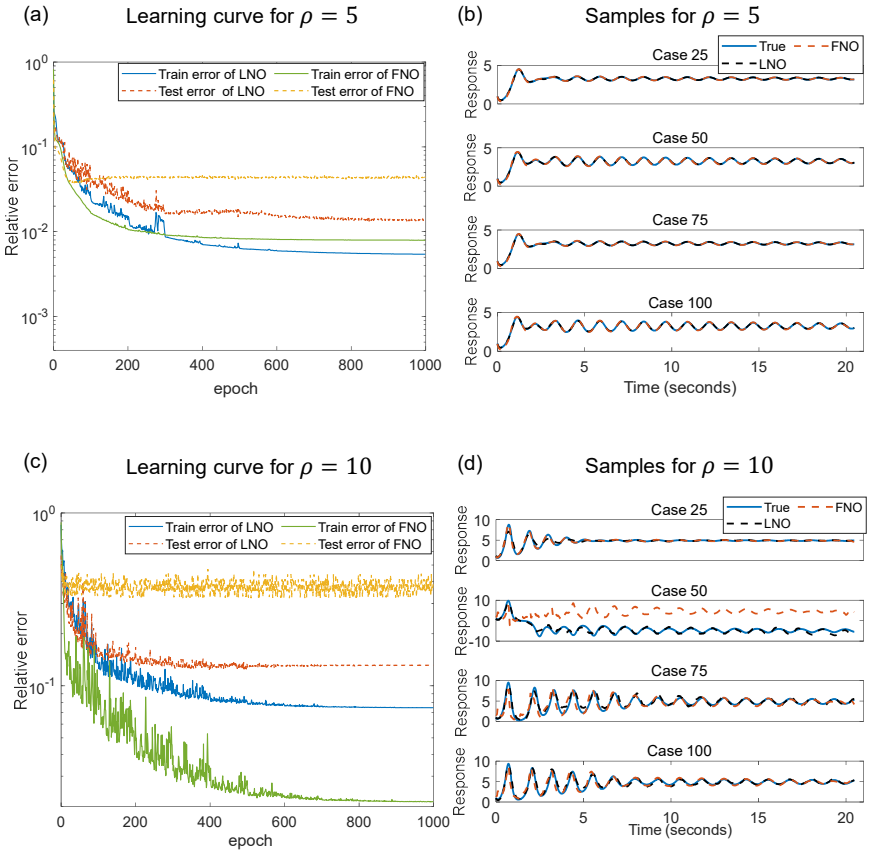

Fig. S7 Lorenz system: comparison of learning rates and responses computed by LNO and FNO: (a) learning curve of the system with $\rho = 5$, (b) response samples of the system with $\rho = 5$, (c) learning curve of the system with $\rho = 10$, (b) response samples of the system with damping $\rho = 10$

S6 Burger's equation

Burger's equation is a nonlinear PDE that occurs in various areas, such as fluid mechanics, nonlinear acoustics, gas dynamics, and traffic flow. For a given field $u(x,t)$ and kinematic viscosity coefficient ν, it takes the form:

$$\begin{aligned} &\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, x \in (0,1), t \in (0,1] \\ &u(x,0) = u_0(x) \end{aligned} \tag{27}$$

with periodic boundary conditions. The initial conditions are modeled as a Gaussian random process such that $u_0 \sim GP(0, C)$, where $C = \sigma^2(-\Delta + \tau^2 I)^{-\gamma}$. In this example, the following values: $\nu = 1/1000$, $\gamma = 2.5$, $\tau = 7$ and $\sigma = 7$ are considered. We aim to learn the operator mapping the initial condition to the solution; that is, $\mathcal{G}_{\boldsymbol{\theta}} : u_0 \to u(x, t)$. The following two scenarios are studied:

- Scenario 1: Mapping from input with low resolution to output with high resolution.
- Scenario 2: Outputs required at positions not aligned with the input sensors.

We train the LNO with a dataset of 800 different u_0, validate and test the trained LNO with a dataset of 100 and 100 random u_0, respectively. The corresponding output $u(x, t)$ is calculated by the open-source package—**chebfun**. For scenario 1, the set up of the input and the output are summarized in Table S1. FNO with the inverse discrete Fourier transform (IDFT) is essentially the latter part of Eq. 32 in the manuscript, which also can learn the mapping from low resolution to high resolution. Motivated by FNO, we introduce an advanced version of LNO. In this advanced version, we retain μ_n and β_n as training parameters for the transient term, while the steady-state term follows the approach of FNO, treating $\boldsymbol{\phi} = (K_{\boldsymbol{\phi}}(i\omega_1), \cdots, K_{\boldsymbol{\phi}}(i\omega_L))$ as the training parameters. This approach decouples the transient and steady-state terms, enhancing the flexibility of the LNO method for operator learning. For fair comparison across the different neural operator architectures (FNO with IDFT, LNO, and advanced LNO), we choose the same hyper-parameters (see Table S5) and report the relative $\mathcal{L}_2$ errors in Table S2. Both advanced LNO and FNO with IDFT achieve similar accuracy, outperforming vanilla LNO. Advanced LNO overcomes the limitations of vanilla LNO in scenarios involving initial conditions and interpolation problems, as discussed in the conclusion. However, for extrapolation problems, vanilla LNO performs better, avoiding overfitting. This will be taken up for study in our future work. Fig. S8 illustrates two representative test samples for the three architectures. The scenario 2 chooses the input and output at different locations, which are summarized in Table S3. The relative $\mathcal{L}_2$ errors for this situation is $9.26 \times 10^{-2} \pm 1.57 \times 10^{-3}$. Fig. S9 displays error plots of two typical test samples.

Here we show some techniques about how to handle different resolutions.

- **Low to High Resolution Transition**: In cases where input data is of low resolution and output data requires a higher resolution, we address this by zero-padding at points where input data is absent. This ensures uniform grids for both input and output data. Subsequently, the original LNO can predict high-resolution output values effectively.
- **Outputs required at positions not aligned with the input sensors**: Alternatively, when faced with scenarios where the outputs need to be computed at locations without input sensors at those specific locations, we resort to the second method. In this method, we augment the original architecture with a standalone Laplace layer (without the weight matrix) after the

lifting layer, $\mathcal{P}$ (as shown in Fig. 1). Following the newly discussed Laplace layer, we continue with the original framework of the Laplace layer and weight matrix. This newly added Laplace layer allows us to handle a smaller number of points in the output.

Table S1 The resolution of the input and outputs for different situations in Burger's equation and diffusion-reaction system.

Application	Type	Resolution of input	Resolution of output
Diffusion-reaction	Same resolution	$\Delta t = 0.05,\ N_t = 21$	$\Delta x = 0.05,\ N_x = 21$ $\Delta t = 0.05,\ N_x = 21$
	Low to high resolution	$\Delta t = 0.1,\ N_t = 11$	$\Delta x = 0.05,\ N_x = 21$ $\Delta t = 0.05,\ N_x = 21$
Burger	Same resolution	$\Delta x = \frac{1}{63},\ N_x = 64$	$\Delta x = \frac{1}{63},\ N_x = 64$ $\Delta t = \frac{1}{49},\ N_x = 50$
	Low to high resolution	$\Delta x = \frac{2}{63},\ N_x = 32$	$\Delta x = \frac{1}{63},\ N_x = 64$ $\Delta t = \frac{1}{49},\ N_x = 50$

Table S2 Relative $\mathcal{L}_2$ error for FNO with IDFT, LNO, and advanced LNO for scenario 1 which considers the Burgers equation and learns the mapping from a low-resolution input to a high-resolution output.

Method	LNO	Advanced LNO	FNO with IDFT
Error	9.15×10^{-2} $\pm 6.99 \times 10^{-4}$	5.31×10^{-2} $\pm 1.79 \times 10^{-4}$	5.32×10^{-2} $\pm 3.05 \times 10^{-4}$

Table S3 The locations of input and output for scenario 2 in Burger's equation and diffusion-reaction system.

Application	Input location	Output location and time
Diffusion-reaction	$x = [0, 0.1, 0.2 \cdots, 0.9, 1]$	$x = [0.05, 0.15, 0.25, \cdots, 0.85, 0.95]$ $t = [0.05, 0.15, 0.25, \cdots, 0.85, 0.95]$
Burger	$x = [0, \frac{2}{63}, \frac{4}{63}, \cdots, \frac{60}{63}, \frac{62}{63}]$	$x = [\frac{1}{63}, \frac{3}{63}, \cdots, \frac{61}{63}, 1]$ $t = [\frac{1}{49}, \frac{3}{49}, \cdots, \frac{47}{49}, 1]$

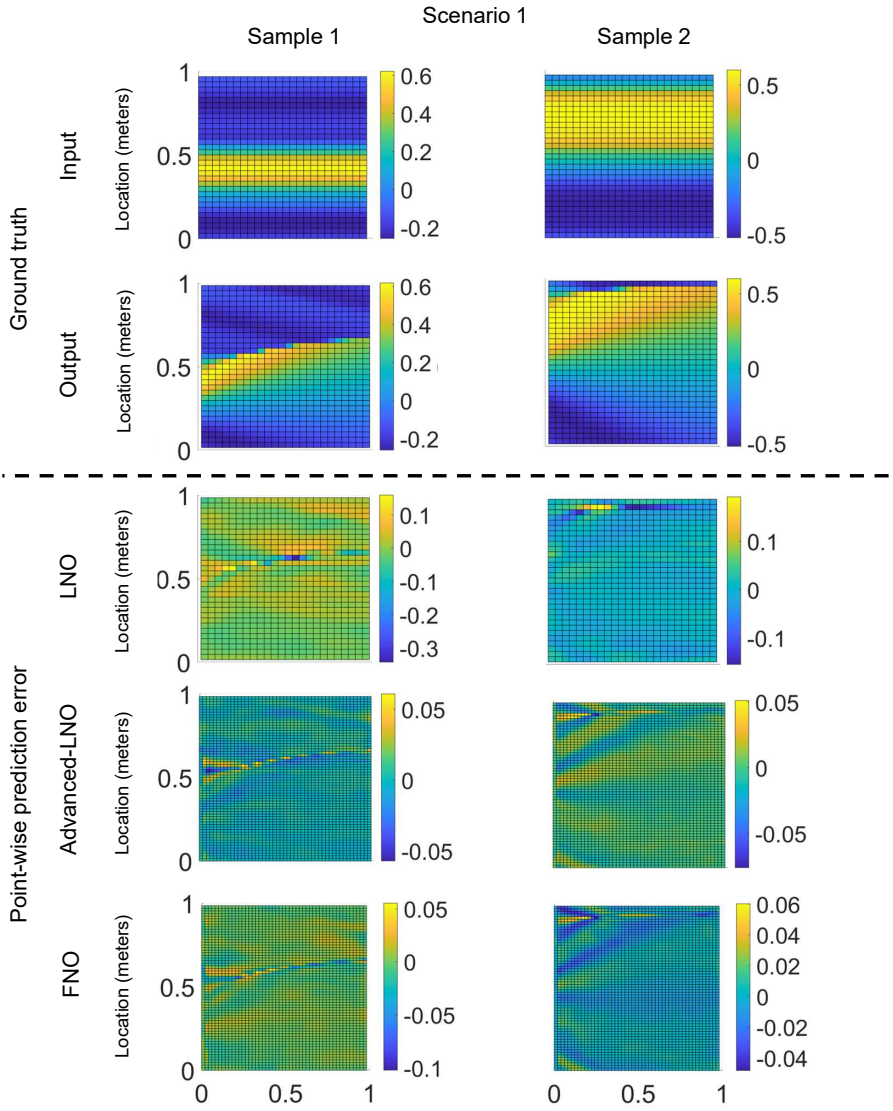

Fig. S8 Pointwise error plots of responses for two representative test samples for Scenario 1 in Burger's equation experiments. Scenario 1 represents the mapping from low-resolution input to high-resolution output.

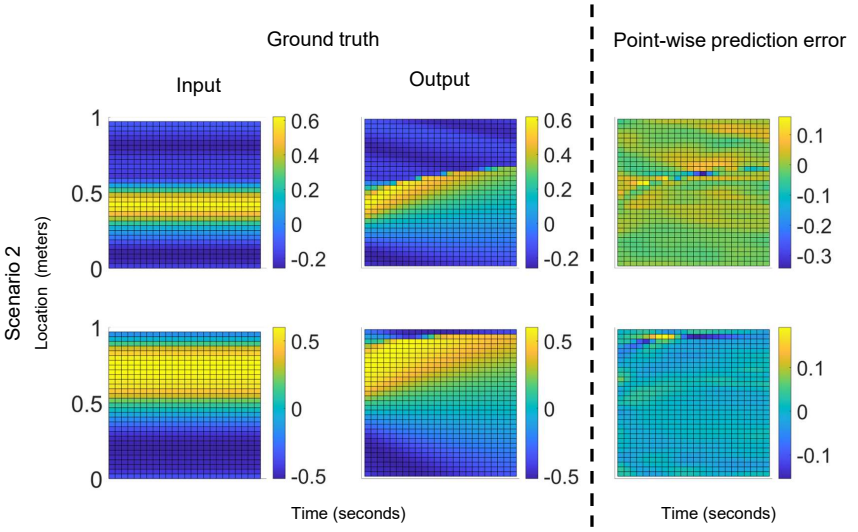

Fig. S9 Pointwise error plots of responses for two representative test samples for Scenario 2 in Burger's equation experiments. Scenario 2 represents the mapping whose input and output are at different locations.

S7 Network architecture details

Table S4 Hyperparameters used in CNO and U-Net for training an operator to approximate the responses for the beam, diffusion equation, and reaction-diffusion system. In this table, *lr* is the learning rate, *bs* is the batch size, and *act* is the activation function.

Network	Layer	Res blocks	Kernel size	Channels	lr	bs	act	Epochs
CNO	4	4	5	64	0.002	50	Leaky ReLU	1000
U-Net	10	/	/	64	0.002	50	ReLU	1000

Table S5 Hyperparameters used in LNO for training an operator to approximate the response where the input and output have different discretizations. In this table, *lr* is the learning rate, *bs* is the batch size, and *act* is the activation function.

Application		Layer	Width	Modes	lr	bs	act	Epochs
Reaction-diffusion system	Scenario 1	1	4	$8, 8$	0.001	20	sin	1000
	Scenario 2	1	4	$8, 8$	0.001	20	sin	1000
Burger's equation	Scenario 1	4	16	$4, 4$	0.01	10	sin	1000
	Scenario 2	4	16	$4, 4$	0.01	10	sin	1000

Table S6 Hyperparameters of GRU with zero initial conditions.

Application		# hidden	Width	# dense	lr	bs	Iterations
Duffing oscillator	$c = 0$	1	10	1	0.001	128	20000
	$c = 0.5$	1	10	1	0.001	128	30000
Driven pendulum	$c = 0$	1	10	1	0.001	128	20000
	$c = 0.5$	1	10	1	0.001	128	30000
Lorenz system	$\rho = 5$	1	10	1	0.001	128	30000
	$\rho = 10$	1	20	1	0.001	128	30000

Table S7 Hyperparameters used in LNO and FNO for training an operator to approximate the response of shallow-water equation

Method	Layer	Width	Modes 1/2/3	lr	bs	act	Epochs
LNO	4	64	4/2/2	0.005	8	ReLU	1200
FNO	4	64	37/5/5	0.005	8	ReLU	1200

Table S8 Hyperparameters used in LNO and FNO for training an operator to approximate the response.

Application			Layer	Width	Modes	lr	bs	act	Epochs
Duffing oscillator	$c = 0$	LNO	1	4	16	0.002	20	sin	1000
		FNO	1	32	16	0.002	20	sin	1000
	$c = 0.5$	LNO	1	4	16	0.002	20	sin	1000
		FNO	1	32	16	0.002	20	sin	1000
Driven pendulum	$c = 0$	LNO	1	4	20	0.005	40	sin	1200
		FNO	1	16	16	0.002	40	sin	1200
	$c = 0.5$	LNO	1	4	8	0.002	40	sin	1200
		FNO	1	16	16	0.002	40	sin	1200
Lorenz system	$\rho = 5$	LNO	1	4	16	0.005	20	tanh	1000
		FNO	1	32	32	0.002	20	tanh	1000
	$\rho = 10$	LNO	1	4	84	0.002	10	tanh	1000
		FNO	4	32	1025	0.002	20	tanh	1000
Euler-Bernoulli beam		LNO	1	16	$4, 4$	0.002	50	sin	1000
		FNO	1	16	$4, 4$	0.002	50	sin	1000
Diffusion equation		LNO	1	16	$4, 4$	0.002	50	sin	1000
		FNO	1	16	$4, 4$	0.002	50	sin	1000
Reaction-diffusion system		LNO	1	48	$4, 4$	0.002	50	sin	1000
		FNO	4	32	$40, 11$	0.002	50	sin	1000
Brusselator reaction -diffusion system		LNO	1	8	$4, 4, 4$	0.005	50	ReLU	300
		FNO	4	8	$11, 8, 8$	0.005	50	ReLU	300
Shallow-water equations		Latent -LNO	4	64	$4, 2, 2$	0.01	8	ReLU	1000

s8 Error metrics

The error metrics used to measure the accuracy of the models can be expressed as Eq. 28:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \quad \text{and} \quad \text{Relative} \quad \mathcal{L}_2 = \frac{|| \, y_i - \hat{y_i} \, ||_2}{|| \, y_i \, ||_2} \quad \text{for} \quad i = (1, n), \tag{28}$$

where n is the number of samples, y_i is the true value of the i^{th} sample, and $\hat{y_i}$ is the predicted value of the i^{th} sample.

References

[1] Liu, F., Tian, Z., Wang, B., Liu, P., Cheng, X., Li, Z.: A new residue-based dynamic analysis method for offshore structures with non-zero initial conditions. Ocean Engineering **162**, 138–149 (2018)

[2] Kreyszig, E., Stroud, K., Stephenson, G.: Advanced engineering mathematics. Integration **9**(4) (2008)

[3] Liu, F., Chen, J., Qin, H.: Frequency response estimation of floating structures by representation of retardation functions with complex exponentials. Marine Structures **54**, 144–166 (2017)

[4] Hu, S.-L.J., Gao, B.: Computing transient response of dynamic systems in the frequency domain. Journal of Engineering Mechanics **144**(2), 04017167 (2018)

[5] Cao, Q., Gao, B., Li, H., Hu, S.-L.J.: A new system identification method operated in the pole domain. In: Experimental Vibration Analysis for Civil Structures: Testing, Sensing, Monitoring, and Control 7, pp. 631–643 (2018). Springer

[6] Zwicker, D.: py-pde: A python package for solving partial differential equations. Journal of Open Source Software **5**(48), 2158 (2020). https://doi.org/10.21105/joss.02158