nature machine intelligence

Article

https://doi.org/10.1038/s42256-026-01265-1

Enabling local neural operators to perform equation-free system-level analysis

Received: 17 September 2025

Accepted: 3 June 2026

Published online: 15 July 2026

Check for updates

Gianluca Fabiani[1,2], Hannes Vandecasteele[2,3], Somdatta Goswami[4], Constantinos Siettos[5] ✉ & Ioannis G. Kevrekidis[2,3,6] ✉

Neural operators (NOs) offer a powerful computational framework to learn complex spatiotemporal dynamics as mappings between infinite-dimensional function spaces. Still, NOs have primarily been used as surrogates for brute-force temporal simulations and predictions. Their potential for systematic, system-level numerical analysis, such as fixed-point, stability and bifurcation analysis—crucial for predicting irreversible transitions in real-world phenomena—remains largely unexplored. Here, motivated by the equation-free approach, we develop a framework that integrates (local) NOs with iterative numerical analysis methods in the Krylov subspace. This approach expands their potential beyond simulation to efficiently analyse large-scale dynamical systems and tackle fundamental challenges in computer-assisted modelling and numerical analysis of complex systems. Furthermore, we demonstrate the utility of learning local in space–time NOs, which, combined with multiscale equation-free schemes—such as projective integration, Gap-Tooth and Patch Dynamics—accelerate system-level computations, improve the conditioning of Krylov solvers and reduce memory requirements, enabling efficient multiscale analysis of complex spatiotemporal dynamics. We illustrate our framework via three nonlinear partial differential equations (PDE) benchmarks: the one-dimensional Allen–Cahn equation, which undergoes multiple concatenated pitchfork bifurcations, the Liouville–Bratu–Gelfand PDE, which features a saddle-node tipping point, and the FitzHugh–Nagumo model, consisting of two coupled PDEs that exhibit both Hopf and saddle-node bifurcations.

Computer-assisted modelling and numerical analysis of complex dynamical systems is fundamental to understanding, predicting and analysing the behaviour of important real-world problems across science and engineering. Based on such models and their numerical/system-level analysis—which extends beyond 'brute-force' temporal simulations—we can enable physics-based decision-making, management and control of the dynamics[1–4]. Key objectives include computing long-term steady states/limit cycles and constructing bifurcation diagrams to detect and understand the mechanisms that trigger qualitative dynamic transitions, including tipping points[5–8]. Such system-level analysis is crucial for addressing real-world challenges, such as pandemic outbreaks, climate catastrophic shifts and economic crises[9–13].

The equation-free (EF) approach[2,3,14,15] was introduced in the early 2000s to address these challenges. EF provides efficient tools to bridge scales and perform direct system-level analysis of microscopic/high-fidelity models of complex systems. The main idea is to

[1]Hopkins Extreme Materials Institute, Johns Hopkins University, Baltimore, MD, USA. [2]Department of Chemical and Biomolecular Engineering, Johns Hopkins University, Baltimore, MD, USA. [3]Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD, USA. [4]Department of Civil and Systems Engineering, Johns Hopkins University, Baltimore, MD, USA. [5]Dipartimento di Matematica e Applicazioni 'Renato Caccioppoli', Università degli Studi di Napoli Federico II, Naples, Italy. [6]School of Medicine's Department of Urology, Johns Hopkins University, Baltimore, MD, USA. ✉e-mail: constantinos.siettos@unina.it; yannisk@jhu.edu

https://doi.org/10.1038/s42256-026-01265-1

'wrap around' the microscopic simulator, matrix-free methods in the Krylov subspace[16,17] and the numerical bifurcation toolkit[18,19] for the systematic analysis and control of the emergent dynamics. Although the EF approach circumvents the need to build surrogate macroscopic models, it still relies on the availability of high-fidelity microscopic simulators, which might be unavailable for many real-world systems, and data may be limited to experimental measurements.

On the other hand, data-driven modelling has its own inherent challenges: sparsity of available data, the 'curse of dimensionality', nonlinearities and stochasticity that escalate the complexity of learning robust representations. Modern scientific machine-learning (ML) methods, such as deep neural networks (DNNs) are also inherently ill-suited for capturing the infinite-dimensional nature of models such as partial differential equations (PDEs).

Neural operators (NOs) have been introduced to address the latter problem by learning mappings between function spaces. Various NO architectures have been proposed over the past few years, including deep operator networks (DeepONets)[20–22], Fourier NOs (FNOs)[23], graph NOs (GNOs)[24,25] and, more recently, random projection-based operator networks (RandONets)[26]. However, although NOs offer powerful approximation capabilities[20,23,26,27], their practical deployment for high-precision numerical analysis tasks can be challenging[28–30].

To address these challenges, we integrate NOs with the EF framework, enabling system-level analysis based on data-driven approximations of PDE solution operators. This expands the role of NOs beyond simple simulations to construct bifurcation diagrams and detect tipping points in high-dimensional systems. The framework is, in principle, architecture agnostic; its practical success depends on the accuracy and generalization of the underlying local NO. Our key idea is to train local NOs and then use them as modular building blocks of computational workflows inspired by classical numerical analysis schemes. This modularity echoes existing work on patch dynamics[31–33], as well as recent developments on learning effective dynamics from microscopic models[34,35]. Our notion of locality is distinct from recent work on NOs with localized kernels[36,37], where 'local' refers to kernel support, not spatiotemporal patches. It is also related but conceptually distinct from 'local' ML/NO approaches in domain-decomposition frameworks[38–41], which mainly aim to improve learning accuracy by partitioning the domain into overlapping subdomains or subdomains with common boundaries. Here, we use 'local' referring to both short in-space and short in-time solution operators, possibly separated by inactive gaps (as in gap-tooth and patch dynamics schemes[42]). Coupled with EF analysis, these local NOs enables scalable, modular workflows for large-scale PDEs, reducing computational cost and memory while allowing seamless integration with Krylov subspace methods.

We envision that combining these ideas with the design of local NOs can enhance training efficiency and substantially broaden their range of applicability to large multiscale complex systems even when only sparse data in time and space are available. This integration yields clear advantages in two distinct scenarios: (1) when an expensive microscopic simulator or high-fidelity PDE solver already exists, a NO trained on a limited number of its trajectories amortizes the computational cost of repeatedly invoking the original simulator, and (2) more fundamentally, when no simulator is available at any scale and only sparse observational data exist, the trained local NO provides a viable time-stepper surrogate. In this case, the integration of EF toolkit becomes essential, enabling computations that neither component could achieve independently.

EF computations with NOs

Performing accurate numerical bifurcation analysis with NOs, including the reliable detection of critical points, is a particularly challenging task. A common approach to constructing bifurcation diagrams involves long temporal simulations of NOs. As noted in refs. 43,44, it is often more efficient to learn the system's short-time evolution and then apply it iteratively. However, such autoregressive strategies can accumulate errors over time, particularly in the presence of high-frequency components, which NOs may not capture accurately, as highlighted in the HINTS framework[45].

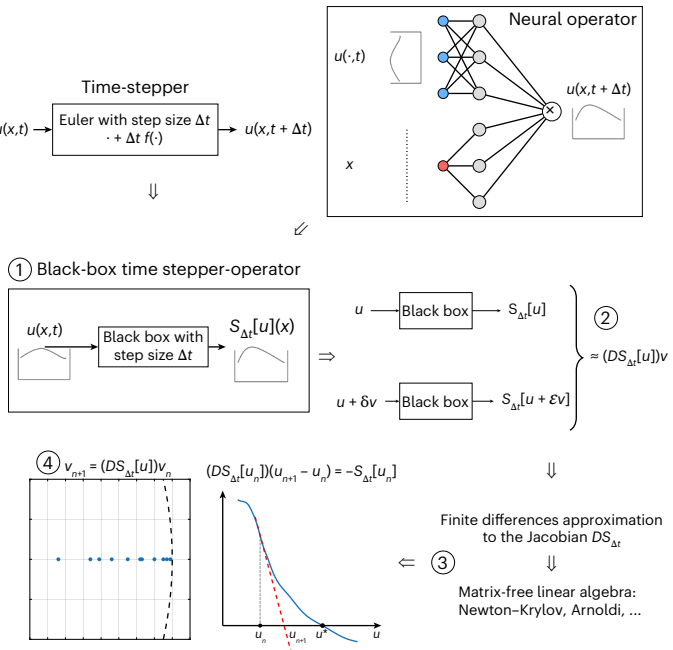


Fig. 1 | Schematic representation of EF-NO framework. (1) An NO maps the current solution profile $u(x,t)$ to the next time step $u(x,t+\Delta t) = \mathcal{S}_{\Delta t}[u](x)$, thus defining a black-box neural time-stepper; (2) numerical approximation of the action of the Jacobian via directional derivatives; (3) matrix-free Krylov subspace methods used to avoid explicit Jacobian formation; (4) Jacobian-free Newton–Krylov (respectively Arnoldi) computations for steady-state (respectively eigenvalues) computations.

To mitigate this issue, an effective approach is to use a fixed-point iteration method, such as Newton–Raphson, to directly find the steady-state solutions of the time-stepper, bypassing the need for long-time simulations. A schematic of this EF-NO workflow is shown in Fig. 1. A related idea for the construction of neural surrogate models by using root-finding algorithms appears in implicit ML methods, such as deep equilibrium models[46]. This fixed-point approach enables convergence to unstable steady states that are unreachable via simple temporal simulations. Moreover, as noted in ref. 47, the discrete-time operator has favourable spectral properties, yielding faster convergence in iterative Krylov solvers, compared with its continuous-time counterpart.

When dealing with large-scale systems, forming the Jacobian matrix explicitly—whether from a discretized PDE or via automatic differentiation (AD) of a NO—is computationally prohibitive due to its size and density. We therefore resort to numerical analysis, matrix-free iterative methods in the Krylov subspace, which explicitly avoid computing the Jacobian matrix and its inverse. In this context, here, we use the generalized minimal residual (GMRES) method. Once a steady state is located, its stability can be assessed via for example, the Arnoldi's algorithm[17,48]. Then, a full bifurcation diagrams as a function of the parameter(s) can then be computed using pseudo arc-length continuation[19].

Although our primary focus is on performing accurate numerical bifurcation and stability analysis, multiscale computations can enhance the applicability of NOs for accelerating computations and learning more efficiently from sparse and limited data. Two notable techniques in this context are (telescopic) projective integration (PI) and the 'gap-tooth' scheme, initially developed in the multiscale EF framework[42,49,50].

https://doi.org/10.1038/s42256-026-01265-1

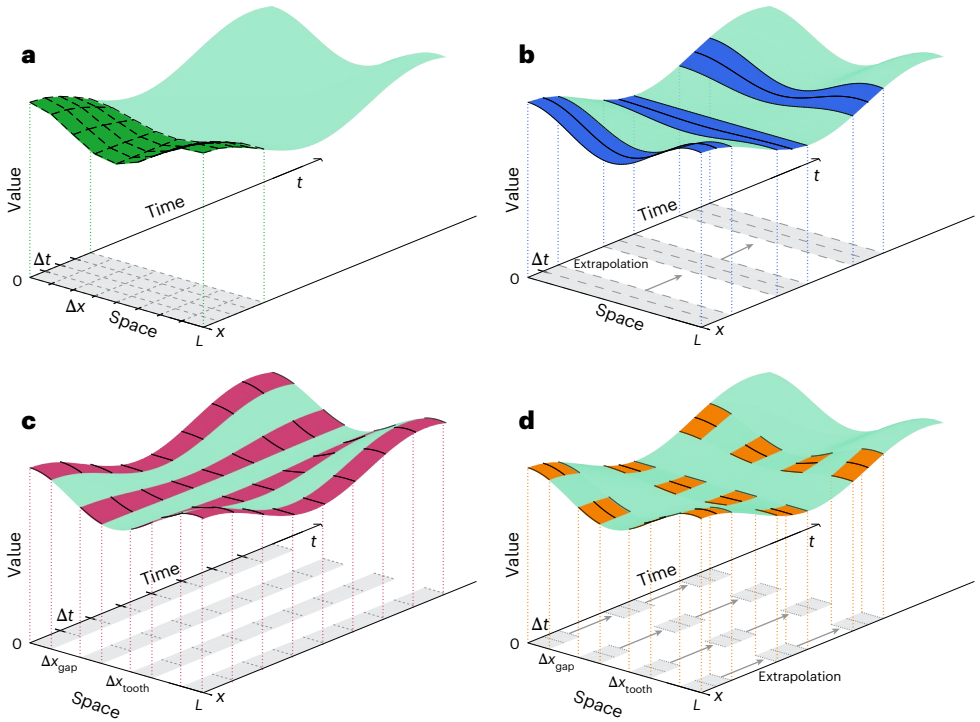


Fig. 2 | Equation-free multiscale neural operator methods. The standard iterative application of the NO time-stepper (**a**), projective integration (PI) applied to the NO time-stepper (**b**), the gap-tooth scheme coupled with the NO (**c**) and the full patch-dynamics scheme using the NO (**d**). The grey-shaded regions indicate where the NO time-stepper is active, whereas the white regions denote inactive domains.

Both exploit the smoothness of PDE solutions in time (PI) and in space (gap-tooth). The combination of PI and gap-tooth leads to the so-called patch dynamics[31], for which we have also proposed patch NOs (patch NOs). Figure 2 illustrates several multiscale extensions compatible with the EF-NO framework, all of which can improve computational efficiency and fixed-point iterations in problems with a scale separation or sparse data. These schemes represent possible alternatives that exploit the continuity and locality principles. We refer to Methods for more details.

Results

We present numerical results for three benchmark problems focused on identifying the solution operator of parameter-dependent PDEs: (1) the Allen–Cahn PDE, (2) a parabolic extension of the Liouville–Bratu–Gelfand (LBG) PDE; (3) a Lattice–Boltzmann implementation of the FitzHugh–Nagumo (FHN) PDE. In particular, we assessed the performance of our local EF-NOs to converge on steady states (stable or unstable), approximate the correct dominant eigenspectrum for stability and reconstruct the corresponding bifurcation diagrams. It is important to note that both the steady states and the associated spectra are derived from the NO surrogate. Consequently, their fidelity is determined by the accuracy with which the NO approximates not only the solution operator but also its Fréchet derivative.

Furthermore, for the LBG, we also explored the use of the 'gap-tooth' NOs: local in space solution operators aiming for greater data efficiency and for increased adaptability to changing domains and boundary conditions. For our computations, we used RandONets as our primary surrogate architecture. For the Allen–Cahn PDE, we also explored a standard DeepONet for comparison (Extended Data Fig. 2 and Supplementary Section B.3).

Case study 1, Allen–Cahn PDE

Our first illustrative case is the one-dimensional Allen–Cahn PDE[51], a model widely used to describe phase transitions and interface dynamics. The Allen–Cahn equation is given by

$$\frac{\partial \phi}{\partial t}(x,t) = \varepsilon \Delta \phi(x,t) - \frac{1}{\varepsilon} W'(\phi(x,t)), \quad x \in \Omega,\ t > 0, \tag{1}$$

with Neumann boundary conditions: $\phi_x(-1) = \phi_x(1) = 0$, $\Omega = [-1, 1]$ and $0 < \varepsilon \ll 1$. The function $W(\phi) = \frac{1}{4}(\phi^2 - 1)^2$ is a double-well potential forcing ϕ to assume values close to ±1.

The bifurcation diagram of the one-dimensional Allen–Cahn PDE has been extensively analysed in ref. 51. In particular, the system exhibits a sequence of pitchfork bifurcations originating from the trivial unstable steady state ($\phi(x) \equiv 0$), while the two trivial stable steady states ($\phi(x) \equiv \pm 1$) persist as stable solutions. Specifically,

- for each integer $n \geq 0$, the parameter values $\varepsilon_n^{(1)} = \frac{1}{\pi/2 + n\pi}$ are bifurcation points. Local solutions $(\phi_n(x,s), \varepsilon_n(s))$ around these points are well approximated by

$$\begin{aligned} &\varepsilon_n(s) = \varepsilon_n^{(1)} + s, \qquad |s| \leq 1, \\ &\phi_n(x,s) = s \sin\left(\tfrac{\pi}{2} + n\pi x\right) + O(s^2); \end{aligned} \tag{2}$$

- for each integer $n \geq 1$, the parameter values $\varepsilon_n^{(2)} = \frac{1}{n\pi}$ are bifurcation points. Local solutions $(\phi_n(x,s), \varepsilon_n(s))$ around these points are well approximated by

$$\begin{aligned} &\varepsilon_n(s) = \varepsilon_n^{(2)} + s, \qquad |s| \leq 1 \\ &\phi_n(x,s) = s \cos(n\pi x) + O(s^2); \end{aligned} \tag{3}$$

where $O(s^2)$ is the order of the truncation error. In the next we will refer to sine-type and cosine-type branches.

See Supplementary Section B.1 for more details on data generation. To assess trained NO accuracy, we compared the predictions of the trained RandONet (Supplementary Section B.3, DeepONet) models against solutions of the true equations obtained

https://doi.org/10.1038/s42256-026-01265-1

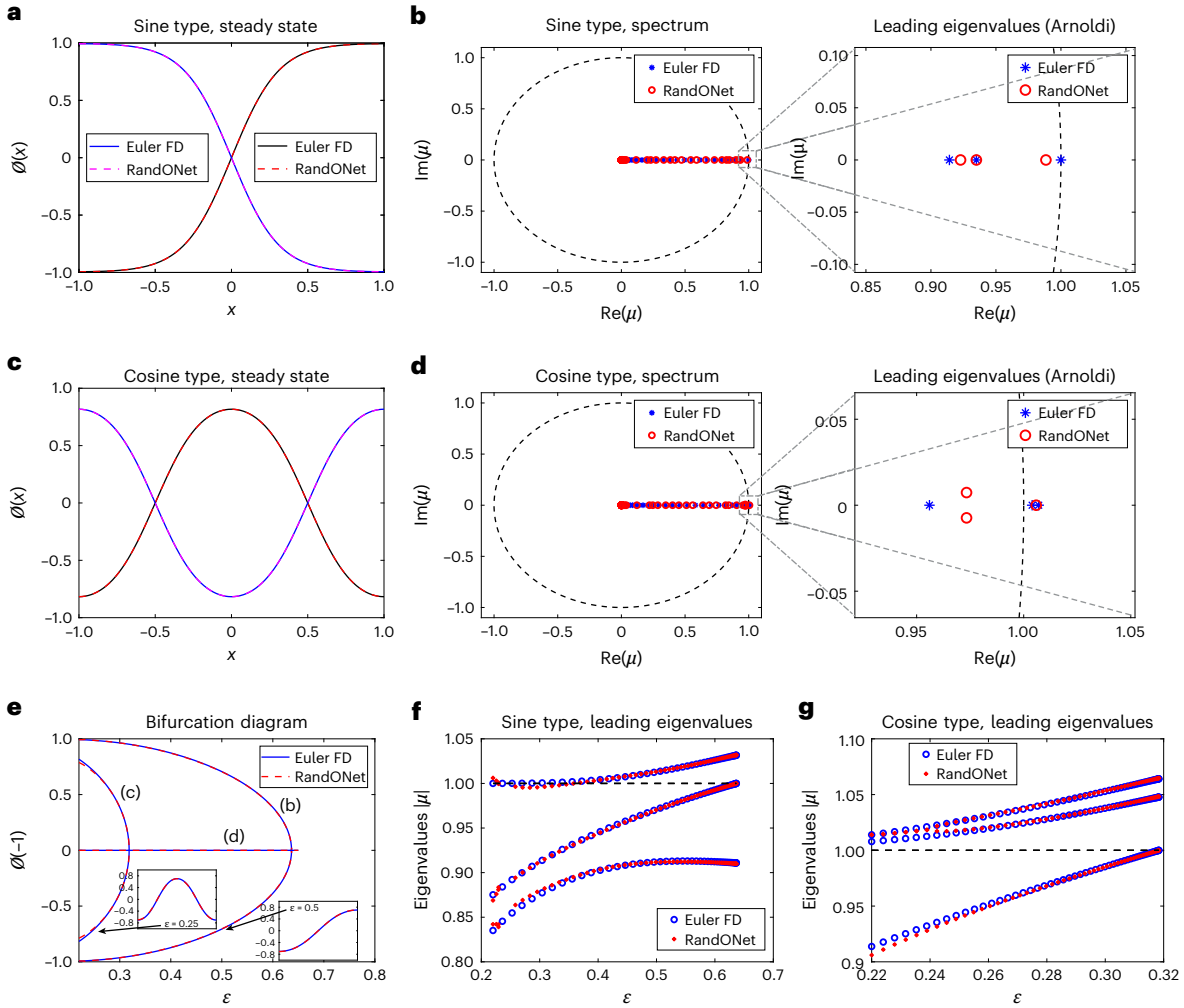


Fig. 3 | RandONet-based time-stepper for the Allen–Cahn PDE. a,**c**, Steady state computed with Newton-GMRES (and symmetric counterpart). **b**,**d**, Full eigenvalue spectrum of the full Jacobian matrix. Re and Im denote the real and imaginary parts, respectively; the dashed circle denotes the unit circle in the complex plane; the zoomed-in insets shows the three leading eigenvalues computed with Arnoldi algorithm. **a**,**b**, Sine-type solutions. **c**,**d**, Cosine-type solutions. **e**, Bifurcation diagram obtained via the homotopy-based RandONet trained with data in $\varepsilon \in (0.22, 0.7)$. **f**,**g**, Tracking absolute values of first three leading eigenvalues using the Arnoldi method. **f**, Sine-type solution (external branch). **g**, Cosine-type solution (internal branch). For **a**–**d**, results were obtained via vanilla RandONet trained using subsampled trajectories at a fixed value of parameter ($\varepsilon = 0.22$). All results are compared with a reference FD discretization of the actual PDE.

with a simple numerical scheme—finite differences (FDs) in space and forward Euler in time.

Bifurcation and stability analysis using NO-based time-steppers. We evaluated the performance of local in time RandONet (as well as DeepONet in Supplementary Section B.3) time-stepper, trained on the multiparameter dataset, with $\varepsilon \in [0.22, 0.7]$, thus enabling the construction of the full bifurcation diagram.

For the RandONet, we used random Fourier features in the branch and sigmoidal activation functions in the trunk. When learning the dynamics in the full parameter range $\varepsilon \in (0.22, 0.7)$, we used a special homotopy-based RandONet with 1,500 neurons in the branch and 300 in the trunk. This constructs homotopy-based embeddings across the parameter space using a convex combination of multiple random bases, as detailed in Methods. This homotopy-based RandONet obtained an mean square error of 10^{-9}, a median L^2 error of 10^{-5} and a maximum absolute error of 10^{-3} for the test set. Notably, for the training of the RandONet, only 5.22 s were necessary, using a single NVIDIA GeForce RTX 2060 GPU and with a `MATLAB` implementation. Further details on the training process and convergence analysis are provided in Extended Data Fig. 1a–f and Supplementary Section B.2.

Then, after the training of the NOs, we attempted to reconstruct the bifurcation diagram, including the first two of the many pitchfork bifurcations given by equations (2) and (3), using Newton-GMRES and pseudo-arclength continuation. The NOs time-steppers are compared against a reference solver based on FD discretization in space and a forward Euler method in time (Euler-FD). To respect numerical stability, the Euler-FD solver uses a fixed time step of $dt = 0.00005$. By contrast, the NOs uses a 200× larger time step of $dt = 0.01$.

The two right-most pitchfork bifurcations computed for both time-steppers are shown in Fig. 3e. As can be seen, there is excellent visual agreement between the bifurcation diagrams of the actual PDE and the one constructed via the RandONet. We also computed the absolute values of the three leading eigenvalues with respect to the bifurcation parameter. The sine-type solutions are shown in Fig. 3f and the cosine-type in Fig. 3g. These results demonstrate that RandONets also accurately approximate the actual key eigenvalues, that is, those governing the dominant dynamics and bifurcation behaviour.

To further assess the performance of the RandONet time-steppers, we first report the results obtained when RandONets were trained at a fixed value of the parameter (for our illustrations for $\varepsilon = 0.22$). Further details on the single-parameter training process are provided in

https://doi.org/10.1038/s42256-026-01265-1

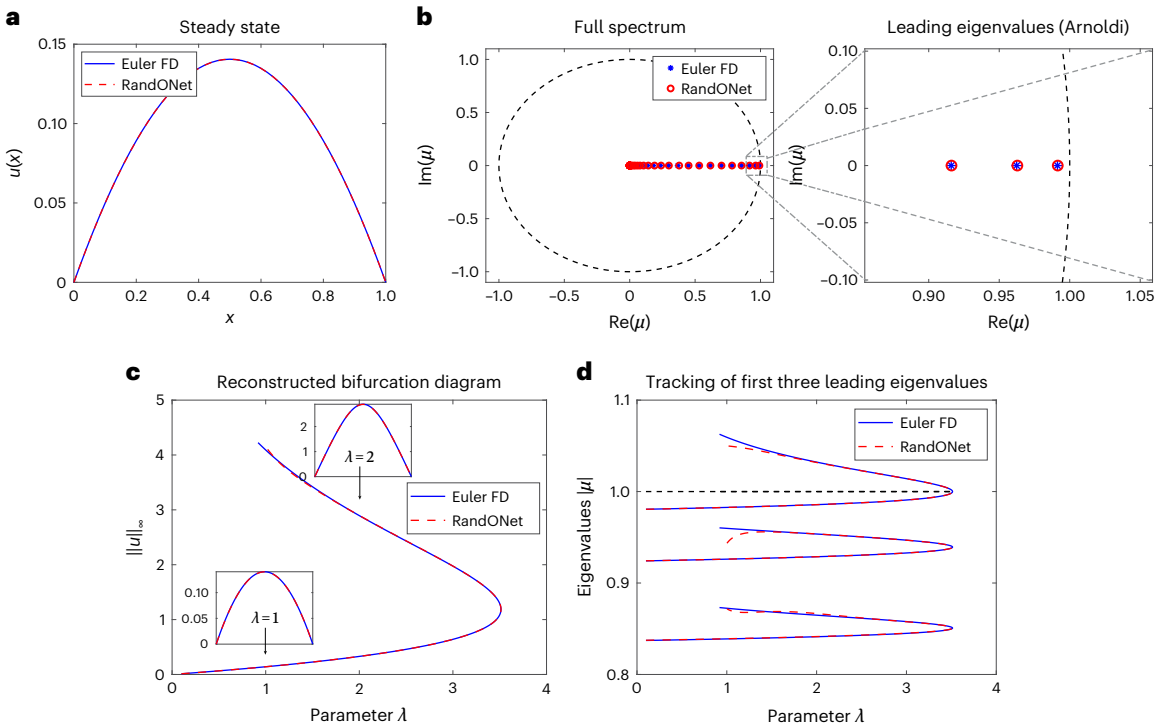


Fig. 4 | RandONet for the parabolic LBG PDE. a, Steady-state solutions obtained with Newton-GMRES. **b**, Full spectrum of the steady-state Jacobian of the actual PDE time-stepper. Re and Im denote the real and imaginary parts, respectively; the dashed circle denotes the unit circle in the complex plane; in the zoomed-in panel, the three leading eigenvalues as computed using the Arnoldi method. **c**, Reconstructed bifurcation diagram with the RandONets. In the two insets, we show two representative solutions associated with $\lambda = 1$ (stable branch) and $\lambda = 2$ (unstable branch). **d**, Tracking the absolute value of first three leading eigenvalues. For **a** and **b**, results were obtained via vanilla RandONet trained using trajectories at a fixed value of the bifurcation parameter ($\lambda = 1$). For **c** and **d**, results were obtianed via the homotopy-based RandONet trained using the full dataset of the dynamics in $\lambda \in [0, 3.8]$. All results are compared with a reference FD discretization of the actual PDE.

Extended Data Fig. 1a,d and Supplementary Section B.2. Figure 3a,c displays the steady-state solutions of the respective methods obtained through Newton-GMRES iterations. In Fig. 3b,d, we present the full eigenvalue spectrum of the steady-state Jacobian of the actual PDE time-stepper. Finally, the zoomed-in insets of Fig. 3b,d shows the three leading eigenvalues calculated by the Arnoldi's algorithm. All these panels highlight a close agreement between the RandONet and the reference solver.

Case study 2, the parabolic LBG PDE

Our second illustrative example is the parabolic LBG PDE[52], that arises in modelling several physical and chemical systems, given by

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \lambda \exp(u), \qquad x \in \Omega = [0,1], \tag{4}$$

with homogeneous Dirichlet boundary conditions $u(0, t) = u(1, t) = 0$. The one-dimensional steady state problem admits an analytical solution[53], reading

$$u(x) = 2\ln\frac{\cosh\theta}{\cosh\theta(1-2x)}, \text{ with } \cosh\theta = \frac{4\theta}{\sqrt{2\lambda}}. \tag{5}$$

It can be shown that when $0 < \lambda < \lambda_c$, the problem admits two solution branches that meet in the saddle-node point $\lambda_c \approx 3.513830719$, while beyond λ_c, no solutions exist[52]. See Supplementary Section C.1 for more details on data generation.

To preserve numerical stability, the Euler-FD solver uses a fixed time step of $dt = 0.0001$. By contrast, RandONets have been trained on larger time step of $dt = 0.001$.

Bifurcation and stability analysis. We first evaluated the accuracy of RandONets in approximating the time evolution, steady states and spectral properties of the LBG PDE at a fixed value of the bifurcation parameter (here, at $\lambda = 1$). For the fixed parameter case, we used a RandONet with 200 neurons in the branch and 150 neurons in the trunk. Full details of the training setup are provided in Supplementary Section C.2 and Extended Data Fig. 3a–d. We compared the steady-state solutions obtained using Newton-GMRES on both the RandONet time-stepper and the reference FD-based time-stepper. These results are shown in Fig. 4a. Figure 4b shows the full eigenvalue spectrum obtained with the RandONet and the reference time-stepper at steady states, showing a nearly-exact match up to an accuracy of 10^{-4}. These eigenvalues of the full Jacobian matrix were computed using a direct differentiation approach. Furthermore, as shown, the three leading eigenvalues computed using the Arnoldi method also exhibit an accuracy of 10^{-4}.

Finally, we trained a homotopy-based RandONet to learn the dynamics for $\lambda \in [0, 3.8]$. For that purpose, we used 1,500 neurons in the branch and 150 neurons in the trunk. Further details are provided in Extended Data Fig. 3c,d and Supplementary Section C.2. Figure 4c shows the bifurcation diagram obtained wrapping around the RandONet, Newton-GMRES with pseudo-arclength continuation. The saddle-node bifurcation was identified at $\lambda_c = 3.51381$, a value that corresponds to the theoretical one with an error of the order 10^{-5}. Figure 4d shows the approximation of the absolute value of the three leading eigenvalues as a function of λ as obtained by RandONets by applying the Arnoldi method.

PI for accelerating time simulations. Here, for problems with time-scale separation, we illustrate how we can further accelerate time simulations by applying PI using the RandONet time-stepper

https://doi.org/10.1038/s42256-026-01265-1

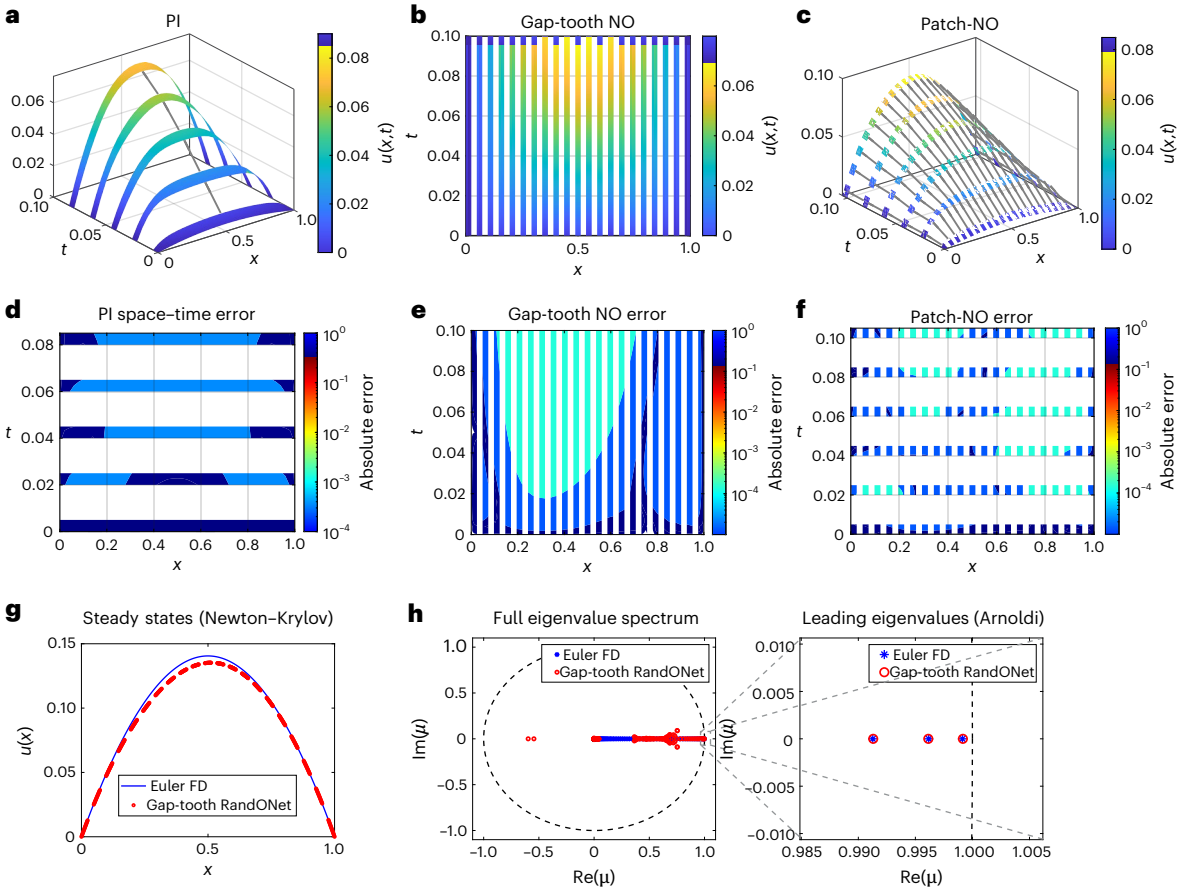


Fig. 5 | PI, gap-tooth-NO and patch-NO results of the RandONet time-stepper for the LBG PDE with $\lambda = 1$. **a**, Projective integration (PI) solution trajectory. **b**, The Gap-Tooth RandONet time-stepper. **c**, Patch RandONet time-stepper (combination of gap-tooth NO and PI). **d**, Corresponding PI space–time error compared with the reference solution. **e**, Corresponding Gap-Tooth spatiotemporal error with respect to the reference solution. **f**, Corresspponding Patch-NO spatiotemporal error with respect to the reference solution. **g**, Steady states computed with Jacobian-free Newton–Krylov GMRES. **h**, Full eigenvalue spectrum of the RandONet and reference Jacobians in their respective steady states. Re and Im denote the real and imaginary parts, respectively; the dashed circle denotes the unit circle in the complex plane; in the zoomed-in panel, the three leading eigenvalues computed with the Arnoldi method.

with a small time step ($dt = 0.001$). PI alternates between short relaxation steps and large projective steps to advance efficiently along the slow time-evolution manifold. Specifically, we performed five short relaxation steps with $\Delta t = 0.001$, followed by a large projective step of size $\Delta T = 0.015$. Figure 5a shows the evolution of the PI solution $u(x,t)$, and Fig. 5d displays the related space–time error. This absolute error remains confined within 10^{-3} (corresponding to a relative spatiotemporal L^2 error of 3.8%). Extended Data Fig. 5a–c track the evolution of the L^∞-error in the solution and its first (representing total variation) and second (representing total curvature) spatial derivatives over time, illustrating the relaxation process.

The 'gap-tooth' NO. Here, we demonstrate how a NO can be used to learn and exploit a local in space solution operator within the 'gap-tooth' framework. In particular, we discretized spatial domain of the LBG PDE into $N_{\text{teeth}} = 21$ teeth and $N_{\text{gaps}} = 20$ gaps of equal width, so that the input functions are defined 'inside the tooth', evolving over smaller spatial domains. Thus, they require smoother, less variegated families of initial conditions. This local-in-space setup avoids full-domain coverage (here, coupled local NOs only evolve over half of the domain, and the other half—the inactive gaps—are recovered by interpolation). For training and short-time predictions, we used sampled data with a local time step of $\Delta t = 0.0001$. For approximating the gap-tooth solution operator, we used a RandONet with 300 neurons in the branch and 100 in the trunk. We obtain results in a mean squared error of 1.37×10^{-11}, a median L2 error of 5.13×10^{-6} and a maximum absolute error of 9.18×10^{-5}. Notably, the training of the single-parameter 'gap-tooth' RandONet took 0.15 s (Extended Data Fig. 4 for training convergence). We provide further details on the data generation and training process in Supplementary Sections C.3 and C.4. We applied the 'gap-tooth' RandONet iteratively through forward passes to generate long-time predictions, with coupling between teeth governed by the subsequent evolution of their neighbours. Figure 5 compares the time evolution between the 'gap-tooth' RandONet time-stepper and the reference solver. Figure 5b shows the predicted evolution. As shown in Fig. 5e, this evolution matches well with the reference trajectories, with an absolute error remaining confined at an 4×10^{-3} level, and corresponding relative spatiotemporal L^2 error of 3.7%. Figure 5g compares both steady-state solutions, showing that the 'gap-tooth' RandONet reproduces the steady-state behaviour satisfactorily, albeit with a slight visible deviation that remains within acceptable bounds. The relative L^2 error between the two steady-states is 3.8%. Spectral analysis of the 'gap-tooth' scheme shown in Fig. 5h reveals that the three leading eigenvalues match very closely with the full reference ones.

The 'patch-NO dynamics'. Finally, we further extend the Gap-Tooth NO approach by incorporating PI, resulting in a patch-dynamics NO. This combines local-in-space and local-in-time evolution, enhancing computational efficiency. Specifically, we performed fifty short relaxation steps with $\Delta t = 0.0001$, followed by a large projective step

https://doi.org/10.1038/s42256-026-01265-1

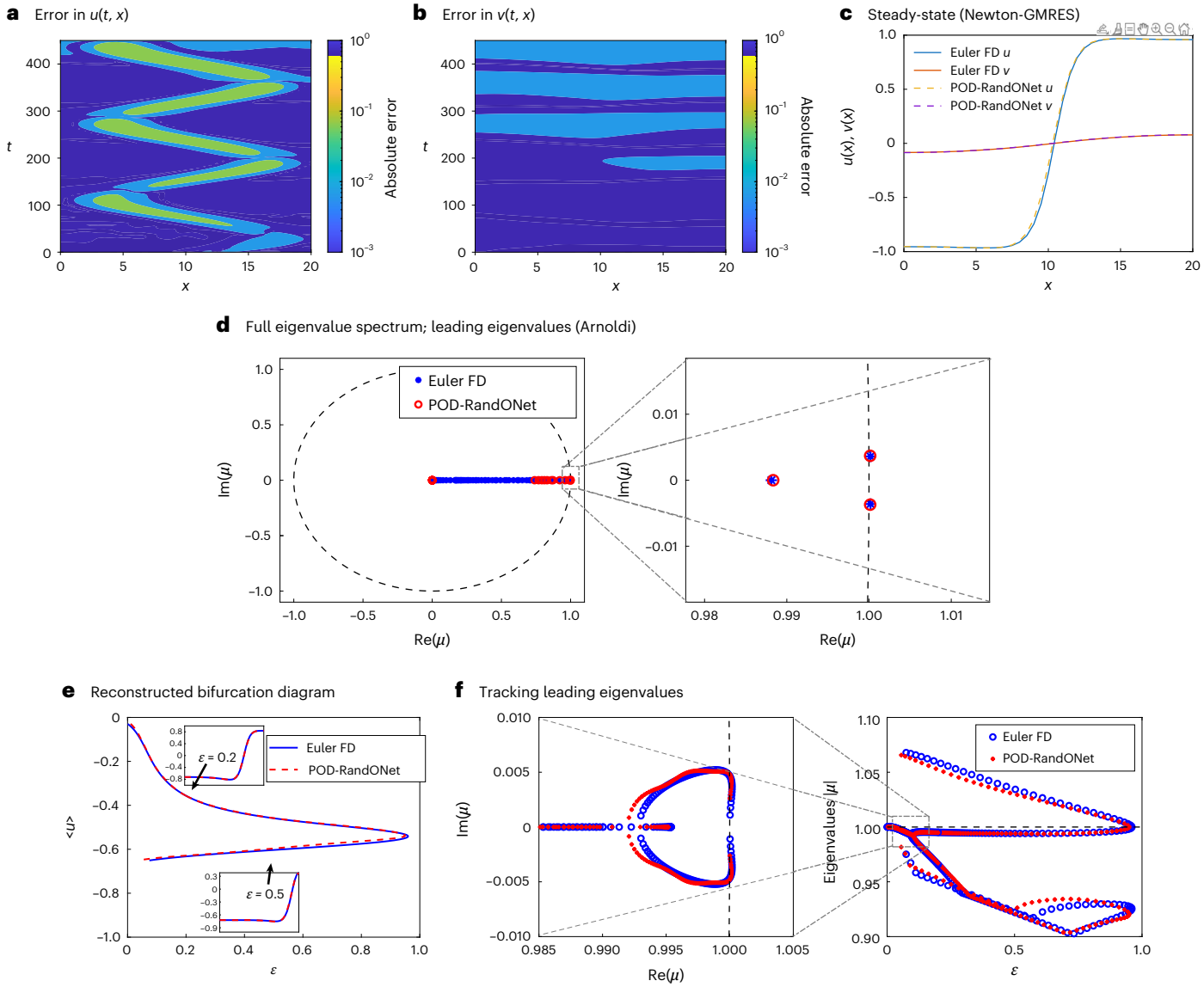


Fig. 6 | POD-RandONet time-stepper for the FHN PDEs, trained for $\varepsilon \in [0.005, 0.995]$. a,**b**, Space–time error of the RandONet versus the ground-truth integration of the actual FHN PDEs, indicatively for $\varepsilon = 0.008$. Errors are reported for the activator variable u in **a** and the inhibitor variable v in **b**. **c**–**f**, Comparison of the RandONet-based versus an FD-based time-stepper of the actual FHN PDEs: steady-states with Newton–Krylov GMRES (**c**); full eigenvalue spectrum via the full Jacobian (inset: the three leading eigenvalues obtained with the Arnoldi method) (**d**); reconstructed bifurcation diagrams (**e**); tracking the absolute value of the two leading eigenvalues via the Arnoldi method (**f**). Re and Im denote the real and imaginary parts, respectively; the dashed circle denotes the unit circle in the complex plane; in the inset, zoomed-in view of their real and imaginary parts near the Hopf bifurcation.

of size $\Delta T = 0.015$. Figure 5c,f show the Patch-NO time-stepper and the corresponding error. This absolute error remains confined within 3×10^{-3}, (corresponding to a relative spatiotemporal L^2 error of 2.5%).

Case study 3, the FHN equations

The FHN equations were first introduced in ref. 54 to describe the propagation of an action potential as a travelling wave. Here, we considered the following system of PDEs, describing the evolution of an activator u and an inhibitor v

$$\begin{aligned}\frac{\partial u(x,t)}{\partial t} &= D^u \frac{\partial^2 u(x,t)}{\partial x^2} + u(x,t) - u(x,t)^3 - v(x,t),\\ \frac{\partial v(x,t)}{\partial t} &= D^v \frac{\partial^2 v(x,t)}{\partial x^2} + \varepsilon(u(x,t) - \alpha_1 v(x,t) - \alpha_0),\end{aligned} \tag{6}$$

where, $\Omega = [0, L]$ with $L = 20$; D^u and D^v, α_0, α_1 are the respective diffusion and reaction parameters, and ε is our bifurcation parameter. This problem is naturally endowed with homogeneous Neumann boundary conditions for both variables. The bifurcation diagram is known to have a turning/tipping point and a supercritical Hopf bifurcation[5,7,55].

To generate the training data for RandONet, we used the $D1Q3$-lattice Boltzmann method[56,57], which simulate the spatiotemporal dynamics through mesoscopic particles (Supplementary Section D.1). In particular, a local time step of $\Delta t = 0.01$ was used during the simulation, and data snapshots were collected every $\Delta t = 0.1$ for training and short-time predictions of the NO.

Bifurcation and stability analysis. We assessed the performance of the RandONet time-steppers to perform bifurcation and stability analysis. To learn the full dynamics in $\varepsilon \in [0.005, 0.995]$, we trained a homotopy-based RandONet enhanced with POD modes. These modes are computed from all training data and retain 99.9% of the total eigenvalue score, eventually reducing the dimensionality of the branch input from 82 to 12. The RandONet consisted of 1,000 neurons in the

branch and 250 in the trunk. Full details on the training setup and convergence analysis are provided in Supplementary Section D.2 and Extended Data Fig. 6.

As in the other problems, we first evaluated the accuracy of RandONets at a fixed value of the bifurcation parameter (here at $\varepsilon = 0.008$). We compared the learned POD-RandONet time-stepper against the reference FD time-stepper of the actual PDE. In Fig. 6a,b, we show the space–time pointwise errors of the POD-RandONet, for a single representative trajectory. Figure 6c depicts the steady-state solutions obtained by the Jacobian-free Newton–Krylov, on POD-RandONet and the reference FD time-stepper. In Fig. 6d, we present the full eigenvalue spectrum of the POD-RandONet and compare it with the Euler-FD time-stepper. By construction, the POD-RandONets only retain 12 active eigenvalues and eigenvectors, with the remaining modes being mapped to zero due to the inherent dimensionality reduction in POD modes. Although this limits the ability to precisely reproduce the full spectrum, the leading modes still provide a coherent and accurate view of the dynamical behaviour. Finally, the the zoomed-in insets of Fig. 6d shows the three leading eigenvalues calculated with the Arnoldi method. The eigenvalues computed with the RandONet were accurate within an error of the order 10^{-3}.

Finally, the bifurcation diagrams are displayed in Fig. 6e. Figure 6f depicts the absolute values of the first three leading eigenvalues (two real and one pair of complex conjugated eigenvalues) as computed by the RandONets upon steady states using the Arnoldi method. A zoomed-in view of Fig. 6f depicts the real and imaginary parts of the leading eigenvalues for ε values close to the Hopf bifurcation.

Discussion

In this work, we have demonstrated the application of local NOs—local in time, over short-time windows and possibly even local in space, over small spatial patches—to perform EF system-level analysis, with a particular focus on long-time steady-state, stability and bifurcation analysis. Specifically, our work bridges a gap left by each standalone methodology: NOs provide fast surrogates but they are limited in providing analytical system-level insight, whereas the EF offers a powerful numerical analysis toolkit for complex systems but requires an available high-fidelity simulator. Their integration enables large-scale stability and bifurcation analysis directly from data, even in the absence of an explicit macroscopic model or microscopic simulator. Although the present work demonstrates the framework on noise-free simulation data, its architecture-agnostic nature allows integration with NOs known for noise robustness (for example, DeepONets over FNOs[21]). Moreover, through the exploration of multiscale techniques such as PI, 'gap-tooth' and patch-NO schemes, we have shown how ML can be integrated with traditional numerical analysis methods to offer data-efficient, scalable solutions for spatially distributed systems.

In this work, as presented in the Supplementary Section B.3, we performed some illustrative comparison between 'vanilla' DeepONets and their similar but shallow and randomized counterparts, RandONets. But a detailed comparison with alternative NO architectures for learning PDE solution operators (including higher-dimensional PDEs) remains an important direction for future research. An important empirical observation from this work is that RandONets accurately reproduce the Jacobian spectrum of the local time-stepper. We believe this is because RandONets reduce training to a convex, regularized linear least-squares problem for the output weights. Those regularizers act as spectral filters: they damp or remove components associated with small singular values, yielding a regularized solution that that is intrinsically 'smoothed' (spectral-filtered under, for example, the Singular Value Decomposition basis). We hypothesize that this spectral damping, by suppressing spurious high-frequency components, may improve the numerical reliability of the directional derivatives used in Krylov subspace routines. Neverthless, we note that while RandONets excelled for learning local operators in our framework, deep architectures such as DeepONets and FNOs remain powerful and may be better suited for learning highly non-local or long-time solution operators, where intrinsic low-dimensionality breaks down and their greater expressivity is required.

To mitigate spectral/Jacobian issues in deep-learning surrogates, potential improvements include Sobolev training and derivative-informed operator learning[58–60] and dissipation enforcement to steer predictions towards attractors[43]. Another promising direction is the development of 'numerical analysis-informed' training, where spectral or stability objectives are integrated directly into the loss function. This would steer NOs not only towards data fidelity but also towards the correct dynamical behaviour. More broadly, our key message is that every variant of NO—including DeepONet and FNO—can benefit from the combination of local training and classical numerical analysis workflows, regardless of the specific implementation or example. This is demonstrated, for instance, by the successful steady-state computations performed with the local DeepONet in Supplementary Section B.3.

It has already been argued (in ref. 44) that iterating short time NOs (trained in their case in a self-supervised manner as physics-informed NO) exhibits advantages over long-time training. In our case, we argue that the use of such local in time (and possibly in space) NOs, trained either on data or as physics-informed NO, opens up exciting algorithmic possibilities. We anticipate that an important benefit—in the case of data-driven NOs—is the potential to 'get away with' much smaller and parsimonious training datasets. Indeed, we expect that it is sufficient to train 'patch' NOs with relatively low-dimensional families of initial and boundary conditions (constant, linear or possibly quadratic)—much less variegated than large space initial conditions and long-time boundary conditions (and, for that matter, different domain shapes). We also expect, in the spirit of ref. 44, that self-supervised PI-NOs will also be easier to train if they are local-in-space × time.

In current (and future) work, we extend our implementation to include adaptive time stepping and to combine PI with the 'gap-tooth' scheme to obtain solutions accelerated by doubly local (in space and in time) NOs. It is important to note that, with local NOs playing the role of discretization points in traditional numerical schemes—with the 'right' flux communication between them, as prescribed by numerical analysis—our approach can also be 'geometry agnostic'. Indeed, to the extent that different macroscale domains can be usefully modelled through point, or finite volume, discretizations, local patch NOs hold the promise of liberating a class of NO computations from needing to be trained on data from specific geometric domains. Another promising direction lies in integrating the proposed framework with data-driven discovery of coarse observables. In the original EF vision, manifold learning techniques identify low-dimensional macroscopic variables on-the-fly from high-dimensional simulation data[61]. Here, our POD enhancement for the FHN system represents a first step. Extending this to nonlinear manifold learning would enable the EF-NO framework to work with emergent macroscopic descriptions—from purely microscopic settings.

Methods

This section outlines the methodological framework used in this study. We begin by formulating the problem setting and defining the operator-learning objective. We then describe the NO architectures used in this work, with emphasis on RandONets and homotopy-based RandONets. We further present the matrix-free numerical techniques used for system-level analysis, including Krylov subspace methods for fixed-point and stability computations. Finally, we summarize the multiscale computational strategies integrated within our framework.

Learning parametric operators with NOs

We address the challenging problem of learning nonlinear parametric operators $\mathcal{F}_\lambda : \mathcal{U} \times \mathbb{R}^p \to \mathcal{V}$ between Banach spaces of functions $\mathcal{U}, \mathcal{V}$ using data-driven NOs. Here, λ is a finite-dimensional parameter of the

underlying operator, for instance specifying a parametric family of differential operators. Specifically, we focus on the so-called 'time-stepper'/solution operator. The learned time-stepper provides a global solver-free surrogate of a spatially distributed system (for example, a PDE model), mapping input functions (for example, initial conditions and parameters) to the system's state at a future time. Then the learned NO time-stepper can be used iteratively for long-time computations (see, for example, refs. 43,44). Notice that, in the standard multiscale EF framework, this time-stepper is computed on demand, every time its application is required and then discarded. In our EF Local NO framework, the same trained neural time-stepper is used with different initial/boundary condition inputs as necessary.

For simplicity, let us assume that $\mathcal{U}$ and $\mathcal{V}$ are subsets of $C^1(\mathrm{X})$ and $C^1(\mathrm{Y})$, respectively. Elements $u \in \mathcal{U}$ are functions $u : \mathrm{X} \subseteq \mathbb{R}^d \to \mathbb{R}$. The parametric operator $\mathcal{F}_\lambda : \mathcal{U} \to \mathcal{V}$ maps such a function to a new function $v \in \mathcal{V}$, defined for a location $\mathbf{y} \in \mathrm{Y} \subseteq \mathbb{R}^d$ by $v(\mathbf{y}) = \mathcal{F}_\lambda[u](\mathbf{y})$.

Given an initial state $u_0(\mathbf{x})$ at time $t = 0$, the solution operator, $\mathcal{S}_{\Delta t}$, outputs the state profile $u_{\Delta t}(\cdot) : \Omega \to \mathbb{R}$ after a certain time horizon Δt

$$u_{\Delta t}(\mathbf{x}) = \mathcal{S}_{\Delta t}[u_0;\lambda](\mathbf{x}), \qquad \mathcal{S}_{\Delta t} : \mathcal{U} \times \Lambda \to \mathcal{U}, \tag{7}$$

where Λ represents the parameter space.

We emphasize that a central feature of our approach is its local-in-time formulation. Specifically, the system's short-time evolution, $\mathcal{S}_{\Delta t}$, over time steps Δt, can be learned and subsequently applied autoregressively via

$$u(\mathbf{x}, T) = \mathcal{S}_T[u,\lambda] = \underbrace{\mathcal{S}_{\Delta t} \circ \mathcal{S}_{\Delta t} \circ \cdots \circ \mathcal{S}_{\Delta t}}_{\frac{T}{\Delta t}\ \text{times}}[u,\lambda], \tag{8}$$

rather than training a solver directly for long-time integration. Although the long-time strategy is computationally efficient, it can suffer from cumulative errors over multiple time steps, particularly in the presence of high-frequency components, which NOs may not capture accurately[43–45].

NO and methodologies

Preliminaries. Here, we will briefly outline the general framework introduced in Chen and Chen work[27], which forms the basis for both DeepONets[20] and RandONets[26]. Consider the problem of approximating an operator $\mathcal{F}$ acting on functions $u \in \mathcal{U}$, sampled at m fixed locations x_j in the domain $\mathcal{X}_1$. The input vector $\mathbf{U} = (u(\mathbf{x}_1), \ldots, u(\mathbf{x}_m)) \in \mathbb{R}^{m\times 1}$ is processed by a branch network, a single (or multi) hidden layer FNN with M neurons, using an activation function φ^{br} and internal weights $\alpha_{ij}^{\mathrm{br}} \in \mathbb{R}$ and biases $\beta_i^{\mathrm{br}} \in \mathbb{R}$, where the superscript "br" denotes the branch network. Whereas the trunk network, again a single (or multi) hidden layer FNN with N neurons, with activation function φ^{tr}, weights $\boldsymbol{\alpha}_k^{\mathrm{tr}} \in \mathbb{R}^d$ and biases $\beta_k^{\mathrm{tr}} \in \mathbb{R}$, where the superscript "tr" denotes the trunk network, processes the new location $\boldsymbol{y} \in K_2 \subset \mathbb{R}^{1\times d}$ to evaluate $\mathcal{F}[u]$. The hidden values of the branch and trunk networks are represented by $\mathbf{B} = (B_1, \ldots, B_M) \in \mathbb{R}^{M\times 1}$ and $\mathbf{T} = (T_1, \ldots, T_N) \in \mathbb{R}^{1\times N}$, respectively,

$$B_i(\mathbf{U}) = \varphi^{\mathrm{br}}\left(\sum_{j=1}^{m} \alpha_{ij}^{\mathrm{br}} u(\mathbf{x}_j) + \beta_i^{\mathrm{br}}\right), \qquad i = 1, \ldots, M, \tag{9}$$

$$T_k(\mathbf{y}) = \varphi_{tr}\left(\mathbf{y} \cdot \boldsymbol{\alpha}_k^{\mathrm{tr}} + \beta_k^{\mathrm{tr}}\right), \qquad k = 1, \ldots, N. \tag{10}$$

The network output is then the weighted inner product of the branch and trunk outputs

$$\mathcal{F}[u](\mathbf{y}) \approx \sum_{k=1}^{N}\sum_{i=1}^{M} w_{ki} B_i(\mathbf{U}) T_k(\mathbf{y}) = \mathbf{T}W\mathbf{B} = \langle \mathbf{T}, \mathbf{B}\rangle_W, \tag{11}$$

where $W \in \mathbb{R}^{N\times M}$ contains the weights w_{ki}.

Although Chen and Chen's original result uses shallow networks, DeepONet extends this by using deep networks or other types of NNs. Training such large networks is computationally intensive and requires parallel computing and GPUs. To circumvent this, we used RandONets, that we introduced recently in ref. 62 and for the completeness of the presentation we briefly describe below.

Difficulties in training NOs. Although NOs learn mappings for infinite-dimensional PDEs, they still face the challenges related to the 'curse of dimensionality'. In fact, NOs built on DNNs typically suffer from the 'curse of dimensionality', as it has been shown their optimization is NP-hard[28], and their overparameterization makes training extremely computationally expensive[29,30,62]. Hence, despite their theoretical promise[27], DNNs and NOs based on them often provide only partially satisfactory numerical approximation accuracy, which in practice hinder their efficiency for high-precision numerical analysis tasks. Recently, to tackle some of the computational challenges mentioned above, we introduced RandONet[26], a randomized NO architecture. RandONets share the same fundamental structure as DeepONets—rooted in the Chen and Chen framework[27]—but replace trainable hidden layers in DNNs with just one hidden layer (as in ref. 27), using fixed randomized embeddings, drawing on random projections, leveraging the Johnson–Lindenstrauss lemma[63] and the universal approximation power of random features[64–66]. This design transforms the original non-convex, gradient-based (and computationally expensive) training used in DNNs into the solution of a linear system for the output weights. Although the resulting system may be high-dimensional and ill-conditioned, it can be solved effectively using numerical analysis methods such as the Moore–Penrose pseudoinverse, Tikhonov regularization[67], QR decomposition or complete orthogonal decomposition[68].

RandONets. In this section, we present the architecture of RandONets, which has been recently introduced in ref. 26. Also, in ref. 26, we have proven that RandONets are universal approximators of nonlinear operators. As proposed in the original Chen and Chen Theorem[27], RandONets use two single hidden layer FNNs with appropriate random bases as activation functions.

Specifically, we propose using (parsimoniously chosen) randomized hyperbolic tangent bases to efficiently embed the space of spatial locations (y)[69]. Thus, $\boldsymbol{\alpha}_k^{\mathrm{tr}} \in \mathbb{R}^d$, β_k^{tr}, $k = 1, \ldots, N$ are independent and identically distributed (i.i.d.) randomly sampled from a continuous probability distribution function, as explained in refs. 26,62,69,70. For the branch network, here we have implemented the following nonlinear random Fourier feature network embeddings[65]

$$\begin{aligned}\mathbf{B} &= \varphi_M^{\mathrm{br}}(\mathbf{U};\boldsymbol{\alpha}^{\mathrm{br}},\boldsymbol{\beta}^{\mathrm{br}}) \\ &= \sqrt{\tfrac{2}{M}}\left[\cos\left(\boldsymbol{\alpha}_1^{\mathrm{br}}\cdot\mathbf{U} + \beta_1^{\mathrm{br}}\right), \ldots, \cos\left(\boldsymbol{\alpha}_M^{\mathrm{br}}\cdot\mathbf{U} + \beta_M^{\mathrm{br}}\right)\right],\end{aligned} \tag{12}$$

where we have two vectors of random variables $\boldsymbol{\alpha}^{\mathrm{br}}$ and $\boldsymbol{\beta}^{\mathrm{br}}$, from which we sample M realizations. The weights $\boldsymbol{\alpha}_i^{\mathrm{br}}$ are i.i.d. sampled from a standard Gaussian distribution, and the biases $\boldsymbol{\beta}_i^{\mathrm{br}}$ are uniformly distributed in $\mathcal{U}[0, 2\pi]$. This explicit random lifting has a low distortion for a Gaussian shift-invariant kernel distance. RandONets training reduces to the solution of a linear least-squares problem in the unknowns W, that is, the external weights of the weighted inner product as in equation (11).

In what follows, we will discuss the particular treatment of the parameter-dependent embeddings we adopted. Specifically, we construct a homotopy across different values of the parameter. For a comprehensive description of the RandONet architecture and implementation, please refer to the original RandONets study[26] and the additional implementation notes in Supplementary Section A, where we also describe the POD-enhanced RandONets used for treating and preconditioning the training of multifield NOs.

Handling parameter-dependent operators via homotopy-based embedding. In operator learning, handling parameter-dependent operators is particularly challenging. A common strategy for parameter-dependent operators is to append the parameter directly to the branch input or to fuse it with the function representation via simple concatenation[20]. However, this approach is known to suffer from representation imbalance when high-dimensional functional inputs are combined with low-dimensional parameters, often leading to weak or ineffective conditioning[22,71]. Related issues have been reported in both NO learning[21], multi-input NO[72] and conditional neural networks[73], where naive fusion mechanisms fail to capture strong or nonlinear parameter effects, particularly near bifurcations or critical transitions. These observations motivate structured parameter embeddings and modulation-based strategies, which our homotopy-based embedding naturally implements.

To mitigate these challenges, we introduce a parameter-dependent embedding strategy based on a continuous homotopy of branch hidden features $\mathbf{B}(\lambda)$, where $\lambda \in \mathbb{R}$ is the parameter of interest. For simplicity, assume that the parameter is rescaled to $\tilde{\lambda} \in [0,1]$. Our proposed homotopy-based embedding $\mathbf{B}_{\tilde{\lambda}}$ is constructed as a convex combination of two independent branch networks $\mathbf{B}_0$ and $\mathbf{B}_1$ (or, in the case of RandONets, two randomized matrices R_0 and R_1) corresponding to the extreme parameter values $\tilde{\lambda} = 0$ and $\tilde{\lambda} = 1$

$$\mathbf{B}_{\tilde{\lambda}}(\mathbf{U},\lambda) = (1-\tilde{\lambda})\mathbf{B}_0(\mathbf{U},\lambda) + \tilde{\lambda}\mathbf{B}_1(\mathbf{U},\lambda). \tag{13}$$

While a rigorous theoretical analysis of this embedding is beyond the scope of this work, we observed empirically that it improves parameter sensitivity and helps capture bifurcations in our test cases. In particular, for the Allen–Cahn example in Supplementary Section B.2, we compared the vanilla RandONet with its homotopy-based counterpart. The results demonstrate that the vanilla architecture exhibits approximately two orders of magnitude lower accuracy across all evaluated metrics, highlighting the efficacy of the homotopy embedding in improving parameter-sensitive operator learning. Future work could investigate the theoretical properties of such embeddings and compare them to other parameter-aware architectures.

EF multiscale methods

Krylov-GMRES. In this section, we briefly describe GMRES, a Krylov subspace method for the solution of the linear system in equation (16), and provide a thorough justification for its use. We note that in classical PDE solvers, the Jacobian matrix arises from the discretization of differential operators using basis functions with local support (for example, finite element shape functions and FD stencils). This locality naturally induces strong sparsity structure (for example, banded or block-sparse), enabling the effective use of sparse direct or iterative solvers. By contrast, NOs (such as DeepONets or FNOs) typically use basis functions with global support. The network architecture creates dependencies where each output degree of freedom couples with all (or many) input degrees of freedom. As a result, this Jacobian is effectively dense, high-dimensional and expensive to form and manipulate explicitly. Therefore, even though AD or FD can be used to construct the full Jacobian, its explicit evaluation and storage is generally computationally intractable for performing system-level tasks, such as stability and bifurcation analysis, making Krylov subspace methods essential for efficient and scalable computations.

Let us consider here a time-stepper $\mathcal{S}_T$, as introduced in equation (8). A steady state u^* of $\mathcal{S}_T$ is a fixed point satisfying

$$u^* = \mathcal{S}_T[u^*,\lambda]. \tag{14}$$

Equivalently the steady-states are precisely the zeros of the operator $\psi(u;\lambda)$, defined by

$$\psi(u;\lambda) = u - \mathcal{S}_T[u,\lambda]. \tag{15}$$

The zeros of ψ for a fixed parameter value λ can then be computed, for example, using Newton's method. Starting from a current estimate $u^{(k)}$, the next iterate is calculated by solving the linear system

$$\nabla\psi(u^{(k)};\lambda)\delta^{(k)} = -\psi(u^{(k)};\lambda), \qquad \delta^{(k)} = u^{(k+1)} - u^{(k)}, \tag{16}$$

where $\nabla\psi$ is the Jacobian of ψ with respect to u.

Considering then the GMRES approach, the linear system in equation (16) consist in minimizing

$$\delta^{(k)} = \arg\min_{\delta^{(k)}\in\mathcal{K}_m} \| \psi(u^{(k)};\lambda) + \nabla\psi(u^{(k)};\lambda)\delta^{(k)}\|_2, \tag{17}$$

where the Krylov subspace $\mathcal{K}_m$ is defined as

$$\mathcal{K}_m = \text{span}\left\{r_0, \nabla\psi(u^{(k)};\lambda)r_0, \dots, \nabla\psi(u^{(k)};\lambda)^{m-1}r_0\right\}. \tag{18}$$

$\mathbf{r}_0$ can be a random vector or

$$\mathbf{r}_0 = -\psi(u^{(k)};\lambda) - \nabla\psi(u^{(k)};\lambda)\delta_0, \tag{19}$$

where δ_0 is the initial guess for GMRES. After m iterations, GMRES returns $\delta^{(k)} = \delta_m$. Note that the Krylov-GMRES method avoids the explicit construction and storage of the full Jacobian matrix by estimating the desired directional derivatives (the action of the Jacobian on a desired direction v, $\nabla\psi(u^{(k)};\lambda)v$), using directional FDs (dir-FD):

$$\nabla\psi(u^{(k)};\lambda)v \approx \frac{\psi(u^{(k)}+\varepsilon v;\lambda) - \psi(u^{(k)};\lambda)}{\varepsilon}, \tag{20}$$

where ε is a small number, typically on the order of the square root of the machine precision or the inherent noise level in the data.

Here, we used a dir-FD approach to numerically estimate the required matrix-free Jacobian action. Directional AD (dir-AD) could also be used. Comparisons between AD and FD in related ML contexts have shown comparable accuracy with modest computational advantages for FD in some settings[74–76]. However, the relative performance remains problem and implementation dependent. The choice between them involves trade-offs: dir-AD provides exact directional derivatives at machine precision but for composite architectures such as gap-tooth and patch dynamics, it would require tracking computational graphs across each local surrogate, introducing implementation complexity. Conversely, dir-FD treats the coupled system as a single black-box evaluation, preserving modularity, but requires careful selection of the perturbation step to balance truncation and round-off errors. A systematic comparison of accuracy, computational cost and implementation effort between the two approaches for NO surrogates is an interesting direction for future work. Furthermore, for composite architectures using local-in-space NOs—such as the gap-tooth and patch-dynamics frameworks introduced in this work—computing the action of the global Jacobian via dir-AD would require tracking computational graphs across each local surrogate and their coupling. This introduces substantial overhead and implementation complexity, whereas dir-FD treat the coupled system as a single black-box evaluation, preserving both efficiency and modularity.

Arnoldi algorithm. In this section, we briefly outline the Arnoldi algorithm used to estimate the leading eigenvalue spectrum of the Jacobian around a steady state of the learned NO. The method is implemented in a matrix-free manner, requiring only the action of the Jacobian on vectors. Around the computed steady state u^*, the dominant eigenvalues and eigenvectors are approximated via the implicitly restarted Arnoldi method, implemented in MATLAB's `eigs` routine (based on the ARPACK library). The algorithm constructs an orthonormal basis

Q_m of the Krylov subspace $\mathcal{K}_m$. The Jacobian $J = \nabla \mathcal{S}_{\Delta t}|_{u=u^*}$ is never assembled explicitly; its action on a vector v is computed using dir-FD as in equation (20)

$$(\nabla \mathcal{S}_T[u^*, \lambda])v = \frac{\mathcal{S}_T[u^* + \varepsilon v, \lambda] - \mathcal{S}_T[u^*, \lambda]}{\varepsilon}. \tag{21}$$

The projection $H_m = Q_m^\top J Q_m$ yields an upper Hessenberg matrix whose eigenvalues (Ritz values) approximate the critical eigenvalues of the Jacobian, thus providing stability information[17]. The associated Ritz vectors $Q_m v$ span a low-dimensional subspace approximating the dominant eigenspace of the Jacobian, enabling local linearization and stability analysis.

Finally, we note that NOs are only approximate surrogates of the underlying PDE solution operator and therefore inherit approximation errors, spectral bias and possible high-frequency artifacts. In particular, DNNs often exhibit spectral bias, a tendency to preferentially represent low-frequency components, which can distort the representation of fine-scale features. When computing directional derivatives or Jacobian-vector products, these errors can accumulate across layers due to repeated application of the chain rule, especially in regimes with steep gradients or near critical points, potentially producing inaccurate high-frequency components in the derivative. Therefore, in the dir-FD approximation, a suitably chosen perturbation step may suppress spurious high-frequency components arising from approximation errors or spectral bias in the NO. For its choise in the context of Krylov subspace methods various practical choices are possible depending on the scaling and numerical characteristics of the problem (see, for example, the discusion in ref. 77).

Comparison of FD and NO-based spectra. As noted in equation (8), any iterative application of a short time-stepper, $\mathcal{S}_{\Delta t}$, can effectively define a long-time-stepper, $\mathcal{S}_T$, which can be equivalently incorporated within the matrix-free solver described above. In our FD computations, the FD time-stepper—typically smaller than the time step used to train the NO—was applied repeatedly to match the effective length of the NO's internal time step. This ensures a consistent comparison, because differences in internal time steps can produce variations in the eigenvalue spectra of discrete-time systems.

For the NO, the time-stepper was defined using a single forward pass. However, as demonstrated in ref. 47, improving the convergence of Krylov methods may require a slightly larger time step. This prevents clustering of eigenvalues near 1, which would correspond to an ill-conditioned, nearly singular operator. A more relaxed spectrum generally improves numerical stability and solver performance. These considerations become particularly relevant when discussing PI and patch dynamics, which enable further acceleration of internal time-stepper computations and reduce memory requirements—the former being crucial for stiff systems and the latter for high-dimensional, large-scale problems.

Furthermore, we remark that using a high-accuracy Chebyshev (spectral) solver[78] to generate training data and then testing the trained NO against an FD discretization of the actual PDE allows us to emulate realistic data-scarce and low-resolution settings. The spectral method provides dense, high-fidelity reference solutions, which we deliberately subsample to construct sparse-in-space and sparse-in-time datasets that mimic the limited information typically available in FD simulations or sensor-based measurements. Testing on FD solutions then directly assesses the robustness and transferability of the learned model to coarse, practical solvers, enabling a principled evaluation of performance under realistic data limitations.

Pseudo-arclength continuation. In this section, we describe the pseudo-arclength continuation algorithm for computing the bifurcation diagram of an operator as a function of the parameter λ. We focus on continuation past saddle-node bifurcations without going into details of bifurcation point detection, types of bifurcation points and branch switching. We refer to a number of published works[18,19,70] for more details on these concepts.

Consider a parameter-dependent function

$$\psi(u;\lambda), \qquad \psi : \mathbb{R}^{n+1} \to \mathbb{R}^n \tag{22}$$

coming from the NO in equation (15), where $u \in \mathbb{R}^n$ is the n-dimensional state variable vector, and $\lambda \in \mathbb{R}$ is a scalar parameter. The goal is to construct a branch Γ

$$\Gamma = \{(u;\lambda) \in \mathbb{R}^{n+1}, \text{ such that } \psi(u;\lambda) = 0\}, \tag{23}$$

corresponding to the steady states of the system in equation (22) for various values of the parameter λ. The main concept underlying pseudo-arclength continuation[19] is to follow the tangent vector τ of equation (23) to construct a curve containing all steady-state solutions. This curve is parametrized using the (pseudo-)arclength s.

A typical implementation involves a predictor–corrector process. We start from a known point $(u(s), \lambda(s)) \in \Gamma$ and its tangent vector $\tau(s)$ to the curve, calculated as the unit vector in the null space of the extended matrix

$$\left[\nabla \psi(u(s);\lambda(s)); \frac{\partial \psi}{\partial \lambda}(u(s);\lambda(s))\right]. \tag{24}$$

We then compute a new point $(u(s + \Delta s); \lambda(s + \Delta s))$ in two steps: (1) predictor compute an initial guess along the tangent $\tau(s)$, and (2) corrector iteratively refines the guess through Jacobian-free Newton–Krylov to a new point on the curve Γ in equation (23). Note that this new point is not always a distance Δs from the starting point due to the nonlinear nature of the Newton method; this is why it is called pseudo-arclength continuation.

This method fails at bifurcation points, that is, where the extended Jacobian in equation (24) becomes singular. However, it is possible to trace solution branches past saddle-node bifurcations because the extended Jacobian does not become singular at such points.

From local NOs to global computations with gap-tooth NOs and PI. In this section, we present extensions of PI and the gap-tooth scheme for local NOs. Both PI and gap-tooth schemes offer promising ways to enhance the efficiency of NO-based solvers, particularly for multiscale systems where fine-scale resolution is computationally expensive. We remark that both PI and gap-tooth schemes rely on smoothness and locality of the underlying dynamics. PI is effective for systems with time-scale separation, such as stiff ODEs or parabolic PDEs, where fast modes (for example, diffusion) decay rapidly onto a slow manifold. After a brief relaxation phase, the slow time-derivative can be extrapolated accurately, enabling large projective steps. Gap-tooth schemes exploit spatial smoothness and local coupling, as in diffusion-dominated or weakly hyperbolic PDEs, where the solution varies slowly between subdomains. Both methods assume continuity and smoothness in time and/or space and become less effective for problems with sharp gradients, strong nonlocality, or lack of scale separation.

Projective integration. PI is useful when the learned NO, $\mathcal{S}_{\Delta t}$, solves over short time intervals, yet acceleration of long-time predictions are desired. Examples include systems that exhibit long-term periodicity. Multiscale PI consists of alternating between short bursts of fine-scale evolution with extrapolation over a longer time interval. Any errors induced by extrapolation are subsequently damped by the fine-scale evolution. This approach is appropriate for problems characterized by multiple (fast-slow) time scales; the trained NO operates over fast times, and the extrapolation allows evolution over slow times.

Given the current state u_n, at the nth PI step, we first apply the time-stepper (the short time NO) for k small steps

$$u_{n,j} = \mathcal{S}_{\Delta t}[u_{n,j-1}, \lambda], \qquad j = 1, \ldots, k, \tag{25}$$

with $u_{n,0} = u_n$. With these states, one can approximate the slow time-evolution derivative using for example, FDs, and then apply a forward Euler scheme (with estimated, slow time derivative), to perform a long-time step projection using extrapolation, for example,

$$u_{n+1} = u_{n,k} + \Delta\tau \frac{u_{n,k} - u_{n,k-1}}{\Delta t}, \tag{26}$$

where $\Delta\tau \gg \Delta t$ is a larger time step. After this extrapolation, short fine-scale steps are resumed to relax back onto the slow manifold. Telescopic PI[50] uses short-time time-steppers to systematically 'jump' over multiple time scales (multiple eigenvalue gaps in the system linearization).

The effectiveness of PI relies on the alternation between relaxation and extrapolation. After each projective leap, a brief sequence of fine-scale steps returns the system to the slow manifold, preventing error accumulation that would cause pure extrapolation to diverge. Moreover, as demonstrated in ref. 47, the discrete-time operator's spectrum 'compactify' as the effective time step increases. Thus, Jacobian eigenvalues cluster away from 1, improving conditioning and accelerating convergence of Krylov methods such as Arnoldi and GMRES. Hence, when used to construct a computationally accelerated longer time-stepper, PI can substantially enhance the convergence of these iterative solvers. This spectral benefit may enable numerical analysis tasks that would otherwise struggle (require more iterations) due to unfavourable spectral properties of short-time operators.

The gap-tooth NO scheme. The 'gap-tooth' scheme is based on the locality and smoothness of the PDE solution in space. This implies that it is theoretically sufficient to learn the (short time) solution operator in local regions in space ('teeth'), with possibly large gaps in between, rather than the full domain. Within each 'tooth', a local solution operator $\mathcal{S}^{\text{local}}_{\Delta t}$ is learned

$$u_{t+\Delta t} = \mathcal{S}^{\text{local}}_{\Delta t}[u_t, \lambda]. \tag{27}$$

Importantly, the boundary conditions for these local-in-space operators are typically prescribed boundary spatial derivatives (corresponding to physical fluxes). Once the 'gap-tooth' NO is trained for various boundary conditions, we can now combine them to evolve a large spatial domain over short times. After each step, the solutions are interpolated to update the boundary conditions for the next iteration, using standard numerical schemes. Repeating this process yields a large space–time evolution constructed from short-space, short-time NOs. In particular, we partition the domain Ω into N small intervals (teeth) centred at $\{x_i\}_{i=1}^N$, each of width Δx, defined as

$$T_i = \left[x_i - \frac{\Delta x}{2}, x_i + \frac{\Delta x}{2}\right], \tag{28}$$

which are separated by $N-1$ inactive intervals (gaps) of size typically $\Delta\xi \geq \Delta x$. At each time step $t = 0, \Delta t, 2\Delta t, \ldots, T$, we interpolate the boundary values, $u\left(x_i - \frac{\Delta x}{2}, s\right)$ and $u\left(x_i + \frac{\Delta x}{2}, s\right)$, for each tooth T_i, using computed solutions in neighbouring teeth T_{i-1} and T_{i+1}, for $s \in [t, t + \Delta t]$, estimating the values of the right and left flux (boundary) F_i^{right} and F_i^{left}. Then we learn (and compute) the local-tooth (T_i) solution operator $\mathcal{S}^{T_i}_{\Delta t}[u_s, F_i^{\text{right}}, F_i^{\text{left}}]$ within each tooth T_i over $s \in [t, t + \Delta t]$ using these interpolated boundary conditions ($F_i^{\text{right}}, F_i^{\text{left}}$), as additional branch inputs for the gap-tooth NOs. When there is no explicit spatial dependence, the learned local NOs are identical (same weights and biases) for each tooth, that is, $\mathcal{S}^{T_i}_{\Delta t} \equiv \mathcal{S}^{T_j}_{\Delta t}, \forall\, i, j = 1, \ldots, N$. This enables a 'patch' learning process that reduces computational complexity.

The size of the patches is an important hyper-parameter that has been extensively studied within the framework of the EF approach[31]. When performing time integration with patch dynamics, given the spatial resolution defined on the available data, one must choose an inner box size h, a buffer size H and a time step δt. These parameters must be selected carefully to ensure accuracy. The inner box width, that is the size of the patch, should be large enough to capture all small-scale effects, yet small enough to save computational time and, for our purposes, to facilitate efficient training of a local NO.

Another important aspect of patch construction is the coupling mechanism. In traditional gap-tooth schemes, coupling is enforced via numerical (global) interpolation: values and derivatives of the global approximant are evaluated at each patch boundary to impose artificial, yet consistent, boundary conditions, effectively allowing each local patch to evolve 'as if' it was an autonomous PDE (over a short time step before the boundaries are updated). This differs from domain-decomposition approaches where coupling is imposed as continuity constraint or due to averaging, and so, importantly, inactive or gap regions are not allowed. Similarly to the traditional schemes, in our local gap-tooth NO framework, each patch NO evolves 'as if' it was an autonomous solution operator. The only requirement is that each patch receives coarse coupling information from neighbouring or nearby local NOs. Importantly, this coupling can be learned directly from data, allowing each patch NO to infer the interaction with its neighbours without requiring explicit analytical boundary conditions or interpolation schemes.

Patch NO dynamics towards scalable and data efficient NO approaches. The two methods, PI and gap-tooth NOs, can be combined—we can have local-in-space and local-in-time NOs; these trained NOs will be used over only part of the spatiotemporal domain (taking advantage of the solution smoothness in space–time). This is the so-called 'patch dynamics' EF scheme[31], which now becomes a 'patch NO' scheme for large space–time, even geometry agnostic domains.

The combination of local NOs with gap-tooth and PI schemes enables scalable and data-efficient system-level computations. By restricting each operator to short-time evolution over a local patch, the learning task becomes substantially easier: the effective input dimension of each patch-NO is appreciably smaller than that of a global NO operator, and variations within the patch are also smaller, reducing high-frequency content and smoothing the function to be learned. This modular decomposition further allows the reuse of many local trajectories from the same dataset, effectively increasing the amount of training data available per local operator. Even in the presence of dense spatiotemporal data, patch-based methods accelerate time integration and fixed-point computations, while inactive regions (gaps) reduce memory usage and computational cost. PI further improves efficiency by enabling frequent larger timesteps in outer solvers, which relaxes the spectrum of the associated time-stepper Jacobian and enhances Krylov convergence[47]. Therefore, the utility of the gap-tooth NOs and patch-NOs dynamics is multifold: it improves computational efficiency, enables scalable system-level analysis and becomes important under data sparsity.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

All datasets generated and analysed during this study are publicly available via GitHub at https://github.com/Centrum-IntelliPhysics/local-neural-operator-time-stepper-instead-of-just-time-stepper. The repository includes scripts to generate the following datasets: (1) the one-dimensional Allen–Cahn equation for $\varepsilon \in [0.22, 0.7]$; (2) the

full-domain and (3) gap-tooth version of the parabolic LBG PDE for $\lambda \in [0, 3.8]$; and (4) the mesoscopic lattice Boltzmann simulations of the FHN PDEs for $\varepsilon \in [0.005, 0.995]$.

Code availability

The code used to produce the findings of this study is publicly available via GitHub at https://github.com/Centrum-IntelliPhysics/local-neural-operator-time-stepper-instead-of-just-time-stepper and archived via Zenodo at https://doi.org/10.5281/zenodo.20328560 (ref. 79).

References

1. Fabiani, G. et al. Task-oriented machine learning assisted surrogates for tipping points of agent-based models. *Nat. Commun.* **15**, 4117 (2024).
2. Kevrekidis, I. G. et al. Equation-free, coarse-grained multiscale computation: enabling microscopic simulators to perform system-level analysis. *Commun. Math. Sci.* **1**, 715–762 (2003).
3. Kevrekidis, I. G., Gear, C. W. & Hummer, G. Equation-free: the computer-aided analysis of complex multiscale systems. *AIChE J.* **50**, 1346–1355 (2004).
4. Karniadakis, G. E. et al. Physics-informed machine learning. *Nat. Rev. Phys.* **3**, 422–440 (2021).
5. Theodoropoulos, C., Qian, Y.-H. & Kevrekidis, I. G. 'coarse' stability and bifurcation analysis using time-steppers: a reaction-diffusion example. *Proc. Natl Acad. Sci.* **97**, 9840–9843 (2000).
6. Siettos, C., Gear, C. W. & Kevrekidis, I. G. An equation-free approach to agent-based computation: bifurcation analysis and control of stationary states. *Europhys. Lett.* **99**, 48007 (2012).
7. Galaris, E., Fabiani, G., Gallos, I., Kevrekidis, I. & Siettos, C. Numerical bifurcation analysis of PDEs from Lattice Boltzmann model simulations: a parsimonious machine learning approach. *J. Sci. Comput.* **92**, 34 (2022).
8. Evangelou, N., Giovanis, D. G., Kevrekidis, G. A., Pavliotis, G. A. & Kevrekidis, I. G. Machine learning for the identification of phase transitions in interacting agent-based systems: a desai-zwanzig example. *Phys. Rev. E* **110**, 014121 (2024).
9. Scheffer, M., Carpenter, S., Foley, J. A., Folke, C. & Walker, B. Catastrophic shifts in ecosystems. *Nature* **413**, 591–596 (2001).
10. Hirota, M., Holmgren, M., Van Nes, E. H. & Scheffer, M. Global resilience of tropical forest and savanna to critical transitions. *Science* **334**, 232–235 (2011).
11. Dai, L., Vorselen, D., Korolev, K. S. & Gore, J. Generic indicators for loss of resilience before a tipping point leading to population collapse. *Science* **336**, 1175–1177 (2012).
12. Russo, L., Russo, P., Vakalis, D. & Siettos, C. Detecting weak points of wildland fire spread: a cellular automata model risk assessment simulation approach. *Chem. Eng. Trans.* **36**, 253–258 (2014).
13. Gnanadesikan, A. et al. Tipping points in overturning circulation mediated by ocean mixing and the configuration and magnitude of the hydrological cycle: a simple model. *J. Phys. Oceanogr.* **54**, 1389–1409 (2024).
14. Rico-Martınez, R., Gear, C. W. & Kevrekidis, I. G. Coarse projective kmc integration: forward/reverse initial and boundary value problems. *J. Comput. Phys.* **196**, 474–489 (2004).
15. Erban, R. et al. Variable-free exploration of stochastic models: a gene regulatory network example. *J. Chem. Phys.* **126**, 04B618 (2007).
16. Kelley, C. T. *Iterative Methods For Linear and Nonlinear Equations* (SIAM, 1995).
17. Saad, Y. *Numerical Methods for Large Eigenvalue Problems: Revised Edition* (SIAM, 2011).
18. Doedel, E. & Tuckerman, L. S. *Numerical Methods for Bifurcation Problems and Large-scale Dynamical Systems* Vol. 119 (Springer, 2012).
19. Allgower, E. L. & Georg, K. *Numerical Continuation Methods: An Introduction* Vol. 13 (Springer, 2012).
20. Lu, L., Jin, P., Pang, G., Zhang, Z. & Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* **3**, 218–229 (2021).
21. Lu, L. et al. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Comput. Methods Appl. Mech. Eng.* **393**, 114778 (2022).
22. Goswami, S., Bora, A., Yu, Y. & Karniadakis, G. E. in *Machine Learning in Modeling and Simulation: Methods and Applications. Computational Methods in Engineering & the Sciences* (eds Rabczuk, T. & Bathe, K. J.) 219–254 (Springer, 2023).
23. Li, Z. et al. Fourier neural operator for parametric partial differential equations. In *Proc. International Conference on Learning Representations (ICLR)* (Curran Associates, 2021).
24. Li, Z. et al. Multipole graph neural operator for parametric partial differential equations. *Adv. Neural Inf. Process. Syst.* **33**, 6755–6766 (2020).
25. Kovachki, N. B. et al. Neural operator: learning maps between function spaces with applications to PDEs. *J. Mach. Learn. Res.* **24**, 4061–4157 (2023).
26. Fabiani, G., Kevrekidis, I. G., Siettos, C. & Yannacopoulos, A. N. Randonets: shallow networks with random projections for learning linear and nonlinear operators. *J. Comput. Phys.* **520**, 113433 (2025).
27. Chen, T. & Chen, H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Trans. Neural Netw.* **6**, 911–917 (1995).
28. Froese, V. & Hertrich, C. Training neural networks is np-hard in fixed dimension. *Adv. Neural Inf. Process. Syst.* **36**, 44039–44049 (2023).
29. Almira, J., Lopez-de Teruel, P., Romero-Lopez, D. & Voigtlaender, F. Negative results for approximation using single layer and multilayer feedforward neural networks. *J. Math. Anal. Appl.* **494**, 124584 (2021).
30. Karumuri, S., Graham-Brady, L. & Goswami, S. Efficient training of deep neural operator networks via randomized sampling. *World Sci. Annu. Rev. Artif. Intell.* **3**, 2540001 (2025).
31. Samaey, G., Kevrekidis, I. G. & Roose, D. Patch dynamics with buffers for homogenization problems. *J. Comput. Phys.* **213**, 264–287 (2006).
32. Runborg, O., Theodoropoulos, C. & Kevrekidis, I. G. Effective bifurcation analysis: a time-stepper-based approach. *Nonlinearity* **15**, 491 (2002).
33. Xiu, D. & Kevrekidis, I. G. Equation-free, multiscale computation for unsteady random diffusion. *Multiscale Model. Simul.* **4**, 915–935 (2005).
34. Arbabi, H., Bunder, J. E., Samaey, G., Roberts, A. J. & Kevrekidis, I. G. Linking machine learning with multiscale numerics: data-driven discovery of homogenized equations. *J. Miner. Metals Mater. Soc.* **72**, 4444–4457 (2020).
35. Chatterjee, T., Chakraborty, S., Goswami, S., Adhikari, S. & Friswell, M. I. Robust topological designs for extreme metamaterial micro-structures. *Sci. Rep.* **11**, 15221 (2021).
36. Liu-Schiaffini, M. et al. Neural operators with localized integral and differential kernels. In *Proc. 41st International Conference on Machine Learning* Vol. 235 (eds Salakhutdinov, R. et al.) 32576–32594 (PMLR, 2024).
37. Li, H., Ye, X., Jiang, P., Qin, G. & Wang, T. Local neural operator for solving transient partial differential equations on varied domains. *Comput. Methods Appl. Mech. Eng.* **427**, 117062 (2024).

38. Jagtap, A. D. & Karniadakis, G. E. Extended physics-informed neural networks (xpinns): a generalized space–time domain decomposition based deep learning framework for nonlinear partial differential equations. *Commun. Comput. Phys.* **28**, 2002–2041 (2020).
39. Wang, W., Hakimzadeh, M., Ruan, H. & Goswami, S. Accelerating multiscale modeling with hybrid solvers: coupling fem and neural operators with domain decomposition. *Comput. Methods Appl. Mech. Eng.* **446**, 118319 (2025).
40. Ouyang, W. et al. NOEM: efficient and scalable finite element method enabled by reusable neural operators. *Nat. Comput. Sci.* **6**, 417–429 (2026).
41. Jiao, A. et al. One-shot learning for solution operators of partial differential equations. *Nat. Commun.* **16**, 8386 (2025).
42. Gear, C. W., Li, J. & Kevrekidis, I. G. The gap-tooth method in particle simulations. *Phys. Lett. A* **316**, 190–195 (2003).
43. Li, Z. et al. Learning chaotic dynamics in dissipative systems. *Adv. Neural Inf. Process. Syst.* **35**, 16768–16781 (2022).
44. Wang, S. & Perdikaris, P. Long-time integration of parametric evolution equations with physics-informed deeponets. *J. Comput. Phys.* **475**, 111855 (2023).
45. Zhang, E. et al. Blending neural operators and relaxation methods in pde numerical solvers. *Nat. Mach. Intell.* **6**, 1303–1313 (2024).
46. Bai, S., Kolter, J. Z. & Koltun, V. Deep equilibrium models. *Adv. Neural Inf. Process. Syst.* **32**, 690–701 (2019).
47. Kelley, C. T., Kevrekidis, I. & Qiao, L. Newton–Krylov solvers for time-steppers. Preprint at https://arxiv.org/abs/math/0404374 (2004).
48. Arnoldi, W. E. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Q. Appl. Math.* **9**, 17–29 (1951).
49. Gear, C. W. & Kevrekidis, I. G. Projective methods for stiff differential equations: problems with gaps in their eigenvalue spectrum. *SIAM J. Sci. Comput.* **24**, 1091–1106 (2003).
50. Gear, C. & Kevrekidis, I. G. Telescopic projective methods for parabolic differential equations. *J. Comput. Phys.* **187**, 95–109 (2003).
51. Zhao, X. E., Chen, L.-Q., Hao, W. & Zhao, Y. Bifurcation analysis reveals solution structures of phase field models. *Commun. Appl. Math. Comput.* **6**, 64–89 (2024).
52. Boyd, J. P. An analytical and numerical study of the two-dimensional Bratu equation. *J. Sci. Comput.* **1**, 183–206 (1986).
53. Mohsen, A. A simple solution of the Bratu problem. *Comput. Math. Appl.* **67**, 26–33 (2014).
54. FitzHugh, R. Impulses and physiological states in theoretical models of nerve membrane. *Biophy. J.* **1**, 445–466 (1961).
55. Lee, S., Kooshkbaghi, M., Spiliotis, K., Siettos, C. I. & Kevrekidis, I. G. Coarse-scale pdes from fine-scale observations via machine learning. *Chaos* **30**, 013141 (2020).
56. Bhatnagar, P. L., Gross, E. P. & Krook, M. A model for collision processes in gases. I. small amplitude processes in charged and neutral one-component systems. *Phys. Rev.* **94**, 511–525 (1954).
57. Qian, Y. & Orszag, S. Scalings in diffusion-driven reaction $a + b \rightarrow c$: numerical simulations by lattice bgk models. *J. Stat. Phys.* **81**, 237–253 (1995).
58. Yu, J., Lu, L., Meng, X. & Karniadakis, G. E. Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. *Comput. Methods Appl. Mech. Eng.* **393**, 114823 (2022).
59. O'Leary-Roseberry, T., Chen, P., Villa, U. & Ghattas, O. Derivative-informed neural operator: an efficient framework for high-dimensional parametric derivative learning. *J. Comput. Phys.* **496**, 112555 (2024).
60. Qiu, Y., Bridges, N. & Chen, P. Derivative-enhanced deep operator network. *Adv. Neural Inf. Process. Syst.* **37**, 20945–20981 (2024).
61. Patsatzis, D. G., Russo, L., Kevrekidis, I. G. & Siettos, C. Data-driven control of agent-based models: an equation/variable-free machine learning approach. *J. Comput. Phys.* **478**, 111953 (2023).
62. Fabiani, G. Random projection neural networks of best approximation: convergence theory and practical applications. *SIAM J. Math. Data Sci.* **7**, 385–409 (2025).
63. Johnson, W. B. Extensions of Lipschitz mappings into a hilbert space. *Contemp. Math.* **26**, 189–206 (1984).
64. Igelnik, B. & Pao, Y.-H. Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Trans. Neural Netw.* **6**, 1320–1329 (1995).
65. Rahimi, A. & Recht, B. Random features for large-scale kernel machines. *Adv. Neural Inf. Process. Syst.* **20**, 1177–1184 (2007).
66. Gorban, A. N., Tyukin, I. Y., Prokhorov, D. V. & Sofeikov, K. I. Approximation with random bases: pro et contra. *Inf. Sci.* **364**, 129–145 (2016).
67. Golub, G. H., Hansen, P. C. & O'Leary, D. P. Tikhonov regularization and total least squares. *SIAM J. Matrix Anal. Appl.* **21**, 185–194 (1999).
68. Hough, P. D. & Vavasis, S. A. Complete orthogonal decomposition for weighted least squares. *SIAM J. Matrix Anal. Appl.* **18**, 369–392 (1997).
69. Fabiani, G., Galaris, E., Russo, L. & Siettos, C. Parsimonious physics-informed random projection neural networks for initial value problems of odes and index-1 daes. *Chaos* **33**, 043128 (2023).
70. Fabiani, G., Calabrò, F., Russo, L. & Siettos, C. Numerical solution and bifurcation analysis of nonlinear partial differential equations with extreme learning machines. *J. Sci. Comput.* **89**, 44 (2021).
71. Goswami, S. et al. Neural operator learning of heterogeneous mechanobiological insults contributing to aortic aneurysms. *J. R. Soc. Interface* **19**, 20220410 (2022).
72. Jin, P., Meng, S. & Lu, L. Mionet: learning multiple-input operators via tensor product. *SIAM J. Sci. Comput.* **44**, A3490–A3514 (2022).
73. Perez, E., Strub, F., De Vries, H., Dumoulin, V. & Courville, A. Film: visual reasoning with a general conditioning layer. In *Proc. AAAI Conference on Artificial Intelligence* Vol. 32, 3942–3951 (AAAI, 2018).
74. Patsatzis, D., Fabiani, G., Russo, L. & Siettos, C. Slow invariant manifolds of singularly perturbed systems via physics-informed machine learning. *SIAM J. Sci. Comput.* **46**, C297–C322 (2024).
75. Ma, Y., Dixit, V., Innes, M. J., Guo, X. & Rackauckas, C. A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions. In *2021 IEEE High Performance Extreme Computing Conference* 1–9 (IEEE, 2021).
76. Margossian, C. C. A review of automatic differentiation and its efficient implementation. *WIREs Data Min. Knowl. Discov.* **9**, e1305 (2019).
77. Knoll, D. A. & Keyes, D. E. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *J. Comput. Phys.* **193**, 357–397 (2004).
78. Platte, R. B. & Trefethen, L. N. Chebfun: a new kind of numerical computing. In *Progress in Industrial Mathematics at ECMI 2008. Mathematics in Industry* Vol. 15 (eds Fitt, A. et al.) 69–87 (Springer, 2010).
79. Fabiani, G., Vandecasteele, H., Goswami, S., Siettos, C. & Kevrekidis, I. G. LEFNO v1.0: a local equation-free neural operator framework. *Zenodo* https://doi.org/10.5281/zenodo.20328560 (2026).

Acknowledgements

We acknowledge computing support provided by the Advanced Research Computing at Hopkins (ARCH) core facility at Johns Hopkins

University and the Rockfish cluster. ARCH core facility (rockfish.jhu.edu) is supported by the National Science Foundation (NSF) grant no. OAC1920103.

Author contributions

G.F.: conceptualization, methodology, software, validation, formal analysis, investigation, writing—original draft, writing—review and editing and visualization. H.V.: methodology, software, validation, formal analysis, investigation, writing—original draft, writing—review and editing and visualization. S.G.: methodology, software, validation, formal analysis, writing—original draft, writing—review and editing and supervision. C.S.: conceptualization, methodology, validation, formal analysis, writing—review and editing, visualization and supervision. I.G.K.: conceptualization, methodology, validation, formal analysis, resources, writing—review and editing, visualization and supervision.

Funding

G.F., H.V., S.G. and I.G.K. acknowledge the Department of Energy (DOE) support under grant no. DE-SC0024162. I.G.K. also acknowledges partial support by the National Science Foundation under grant nos. CPS2223987 and FDT2436738. C.S. acknowledges partial support by the PNRR MUR projects PE0000013-Future Artificial Intelligence Research-FAIR and CN0000013 CN HPC—National Centre for HPC, Big Data and Quantum Computing, Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

Competing interests

The authors declare no competing interests.

Additional information

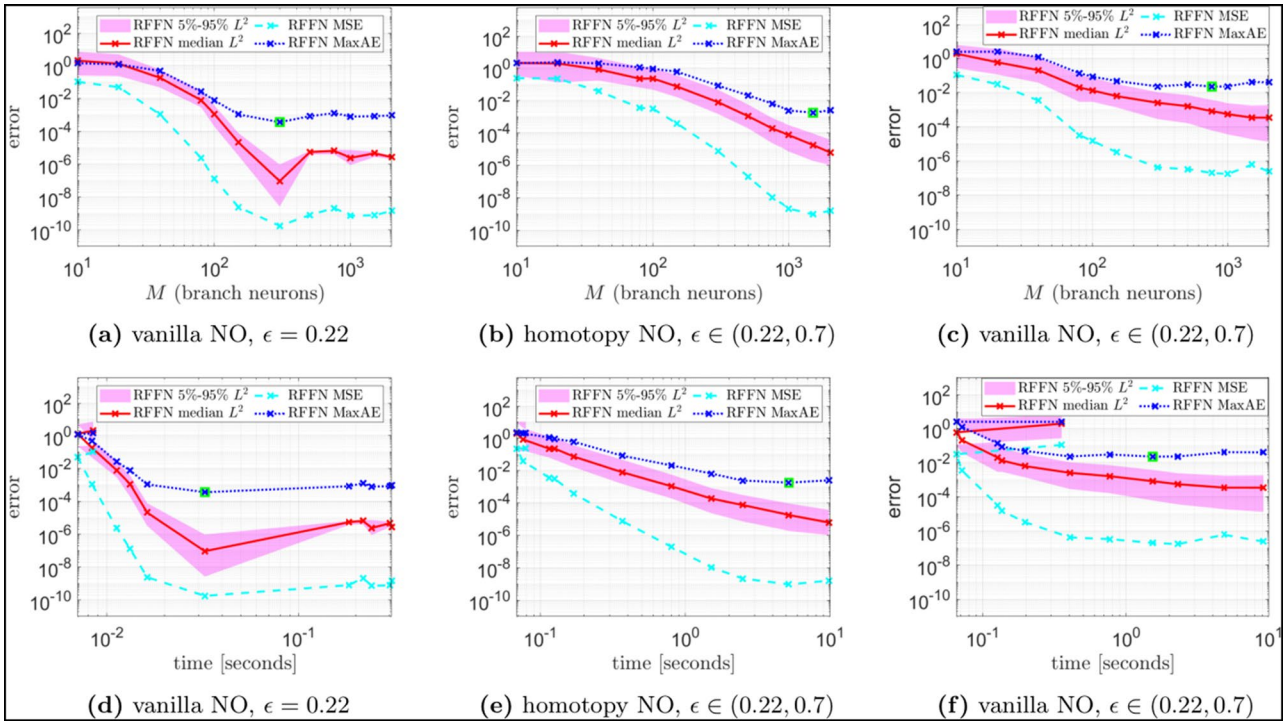
Extended data is available for this paper at https://doi.org/10.1038/s42256-026-01265-1.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42256-026-01265-1.

Correspondence and requests for materials should be addressed to Constantinos Siettos or Ioannis G. Kevrekidis.

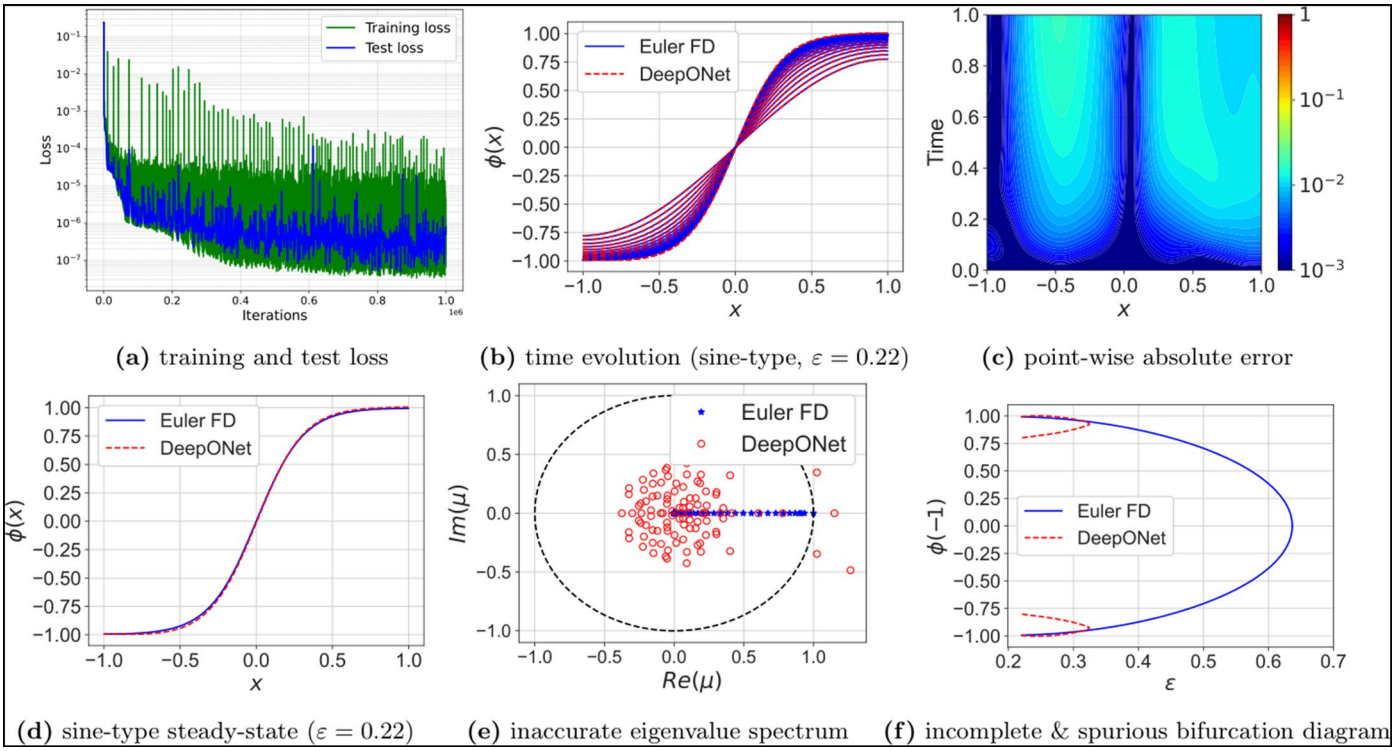
Peer review information *Nature Machine Intelligence* thanks Michael R. A. Abdelmalik, Lianghao Cao, Shaowu Pan and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Peer reviewer reports are available.

https://doi.org/10.1038/s42256-026-01265-1



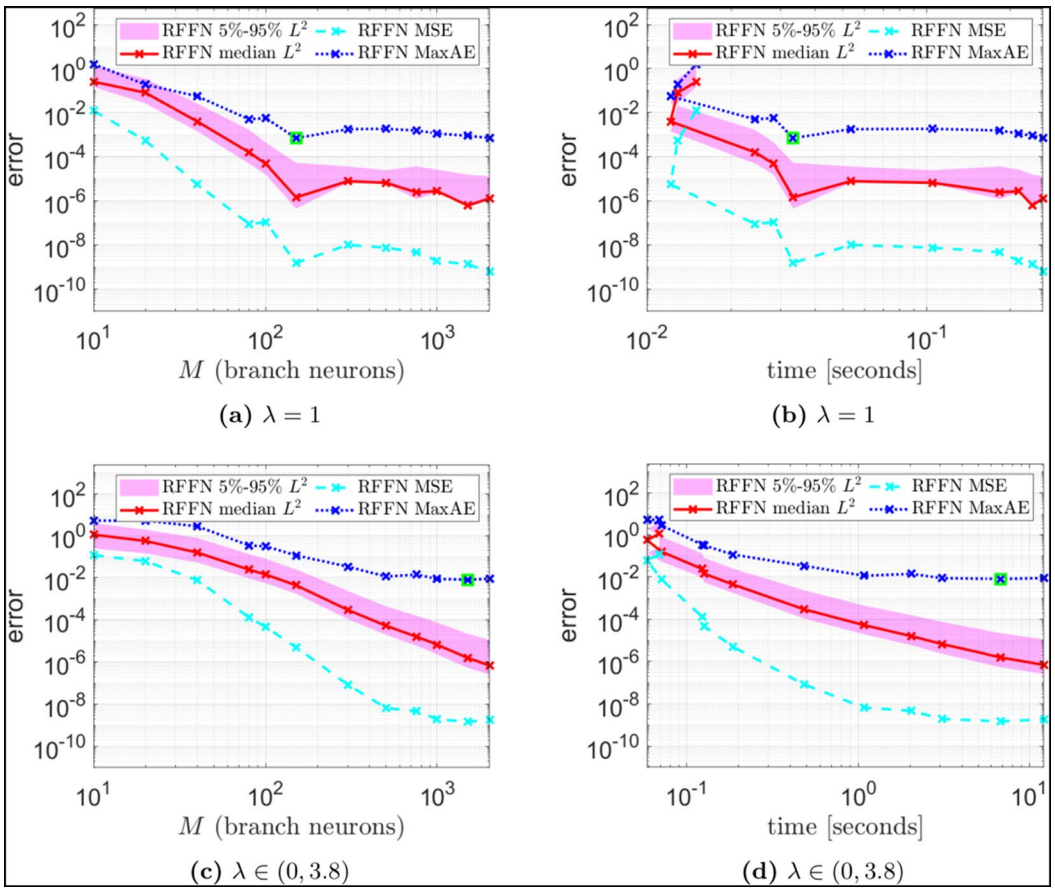
Extended Data Fig. 1 | Training convergence of the RandONets for the Allen-Cahn PDE in Eq. (1) in the main text. (**a**),(**d**) Single value $\varepsilon = 0.22$ with vanilla RandONets. (**b**),(**e**) Multi paramater $\varepsilon \in (0.22, 0.7)$ with homotopy-based RandONets. (**c**),(**f**) Multi paramater $\varepsilon \in (0.22, 0.7)$ with vanilla RandONets. Convergence of the RandONets w.r.t (**a**)–(**c**) the number of neurons in the branch and (**d**)–(**f**) the computational time. We report the Maximum absolute error (MaxAE), the Mean Square Error (MSE) and the median of the L^2 error and the range between 5% and 95% percentiles.

https://doi.org/10.1038/s42256-026-01265-1



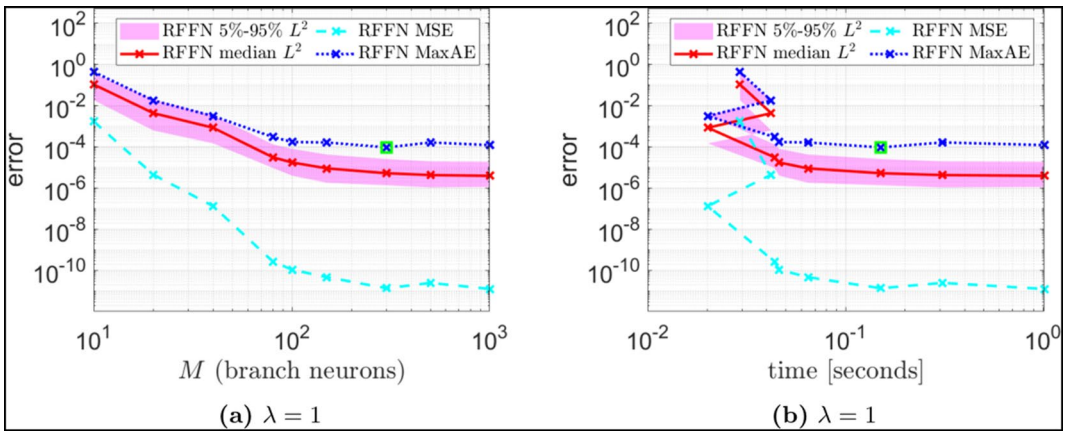
Extended Data Fig. 2 | DeepONet-based time-stepper for the Allen-Cahn PDE in Eq. (1) in the main text, trained with data with multiple parameters in the range $\varepsilon \in [0.22, 0.7]$. The results are compared with the approximated solutions, obtained using Finite-Difference (FD) discretization in space and forward Euler in time of the *known* PDE. Results are compared with a reference FD discretization of the *actual* PDE. (**a**) training and test loss during optimization process. (**b**) Time evolution for a sine-type initial condition for $\varepsilon = 0.22$; (**c**) corresponding point-wise absolute error; (**d**) sine-type steady-state solution for $\varepsilon = 0.22$; (**e**) inaccurate eigenvalue spectrum of Jacobian computed for the steady-state; (**f**) arc-length continuation of DeepONet steady-states give rise to an incomplete and spurious bifurcation diagram.

https://doi.org/10.1038/s42256-026-01265-1



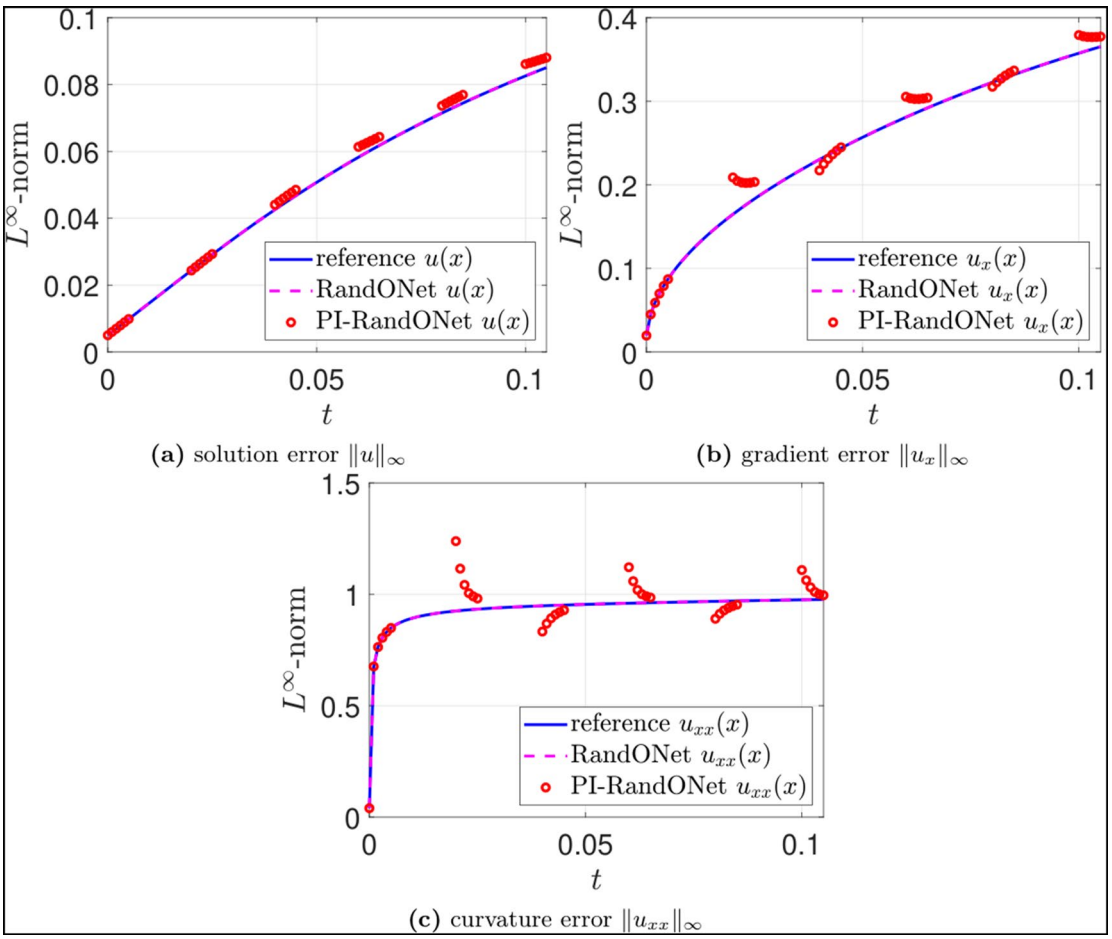
Extended Data Fig. 3 | Training convergence results of the RandONets for the Liouville-Bratu-Gelfand (LBG) PDE. (**a**),(**b**) RandONet for Single value $\lambda = 1$. (**c**),(**d**) homotopy-based RandONets for the parametric LBG PDE. Convergence of the RandONets w.r.t ((**a**),(**c**)) the number of neurons in the branch and ((**b**),(**d**)) the computational time. We report the Maximum absolute error (MaxAE), the Mean Square Error (MSE) and the median of the L^2 error and the range between 5% and 95% percentiles.

https://doi.org/10.1038/s42256-026-01265-1



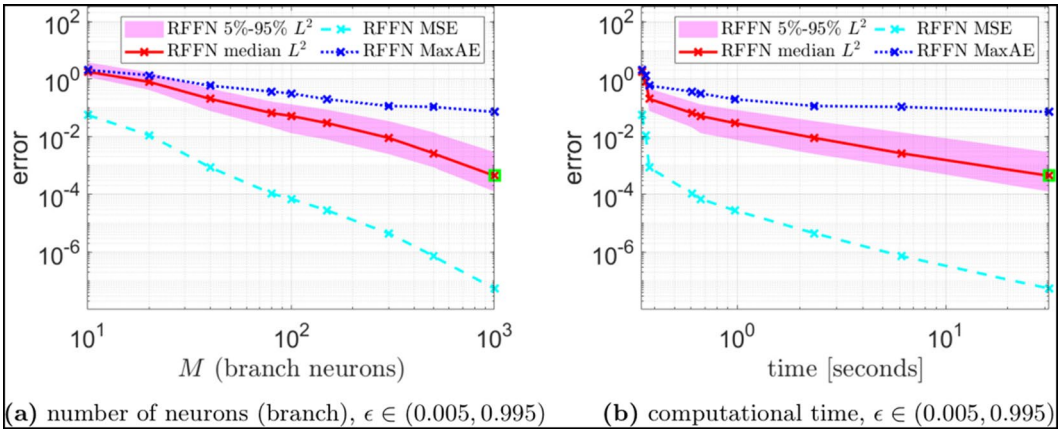
Extended Data Fig. 4 | Training convergence results of the RandONets for the Gap-Tooth Liouville-Bratu-Gelfand PDE. (**a**)-(**b**) RandONets for Single value $\lambda = 1$. Convergence of the RandONets w.r.t (**a**) the number of neurons in the branch and (**b**) the computational time. We report the Maximum absolute error (MaxAE), the Mean Square Error (MSE) and the median of the L^2 error and the range between 5% and 95% percentiles.

https://doi.org/10.1038/s42256-026-01265-1



Extended Data Fig. 5 | Error evolution during Projective Integration for the Liouville-Bratu-Gelfand equation. (**a**) L^∞-error of the solution, (**b**) error in the first spatial derivative (total variation), and (**c**) error in the second derivative (total curvature). The rapid decay in (**c**) confirms fast relaxation of high-frequency modes onto the slow manifold.

https://doi.org/10.1038/s42256-026-01265-1



Extended Data Fig. 6 | Training convergence of the RandONets for the FHN PDE, simulated with Lattice Boltzmann method. (**a**),(**b**) Multi paramater $\varepsilon \in (0.005, 0.995)$ with homotopy-based RandONets. Convergence of the RandONets w.r.t (**a**) the number of neurons in the branch and (**b**) the computational time. We report the Maximum absolute error (MaxAE), the Mean Square Error (MSE) and the median of the L^2 error and the range between 5% and 95% percentiles.

nature portfolio

Corresponding author(s): Ioannis G. Kevrekidis, Constantinos Siettos

Last updated by author(s): May 21, 2026

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed
☐	☒ The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☐	☒ A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒ The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐ A description of all covariates tested
☒	☐ A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☒	☐ A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☒	☐ For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐ For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐ For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐ Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	in the GitHub repository https://github.com/Centrum-IntelliPhysics/local-neural-operator-time-stepper-instead-of-just-time-stepper
Data analysis	in the GitHub repository https://github.com/Centrum-IntelliPhysics/local-neural-operator-time-stepper-instead-of-just-time-stepper

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our policy

Data used in this work are publicly available in the GitHub repository https://github.com/Centrum-IntelliPhysics/local-neural-operator-time-stepper-instead-of-just-time-stepper

Research involving human participants, their data, or biological material

Policy information about studies with human participants or human data. See also policy information about sex, gender (identity/presentation), and sexual orientation and race, ethnicity and racism.

Reporting on sex and gender	n/a
Reporting on race, ethnicity, or other socially relevant groupings	n/a
Population characteristics	n/a
Recruitment	n/a
Ethics oversight	n/a

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	The data were numerical simulations of PDEs
Data exclusions	n/a
Replication	n/a
Randomization	n/a
Blinding	n/a

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☒	☐ Animals and other organisms
☒	☐ Clinical data
☒	☐ Dual use research of concern
☒	☐ Plants

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging

Plants

Seed stocks	n/a
Novel plant genotypes	n/a
Authentication	n/a