Journal of Applied Mathematics and Computing (2024) 70:4395–4421
https://doi.org/10.1007/s12190-024-02131-2

ORIGINAL RESEARCH

Learning unbounded-domain spatiotemporal differential equations using adaptive spectral methods

Mingtao Xia[1] · Xiangting Li[2] · Qijing Shen[3] · Tom Chou[2,4]

Received: 20 April 2024 / Revised: 20 April 2024 / Accepted: 8 May 2024 /
Published online: 3 June 2024

Abstract
Rapidly developing machine learning methods have stimulated research interest in computationally reconstructing differential equations (DEs) from observational data, providing insight into the underlying mechanistic models. In this paper, we propose a new neural-ODE-based method that *spectrally expands* the spatial dependence of solutions to learn the spatiotemporal DEs they obey. Our spectral spatiotemporal DE learning method has the advantage of not explicitly relying on spatial discretization (e.g., meshes or grids), thus allowing reconstruction of DEs that may be defined on *unbounded* spatial domains and that may contain long-ranged, nonlocal spatial interactions. By combining spectral methods with the neural ODE framework, our proposed spectral DE method addresses the inverse-type problem of reconstructing spatiotemporal equations in *unbounded domains*. Even for bounded domain problems, our spectral approach is as accurate as some of the latest machine learning approaches for learning or numerically solving partial differential equations (PDEs). By developing a spectral framework for reconstructing both PDEs and partial integro-differential equations

✉ Tom Chou
tomchou@ucla.edu

Mingtao Xia
xiamingtao@nyu.edu

Xiangting Li
xiangting.li@ucla.edu

Qijing Shen
qijing.shen@ndm.ox.ac.uk

[1] Courant Institute of Mathematical Sciences, New York University, 251 Mercer St., New York, NY 10012, USA

[2] Department of Computational Medicine, UCLA, 621 Charles E. Young Dr. S., Los Angeles, CA 90095-1766, USA

[3] Nuffield Department of Medicine, University of Oxford, John Radcliffe Hospital, Oxford OX3 9DU, UK

[4] Department of Mathematics, UCLA, 520 Portola Plaza, Los Angeles, CA 90095-1555, USA

(PIDEs), we extend dynamical reconstruction approaches to a wider range of problems, including those in unbounded domains.

Keywords Neural ODE · Spectral method · Differential equations · Spatiotemporal inverse problem · Unbounded domains

Mathematics Subject Classification 35P05 · 62M45 · 62F30 · 62G99 · 34A55

1 Introduction

There has been much recent interest in developing machine-learning-based methods for learning the underlying physics-based equations of motion from data. In this paper, we are interested in learning the general dynamics $F[u; x, t]$ of spatiotemporal partial differential equations (PDEs) or spatiotemporal partial integro-differential equations (PIDEs) such as

$$\partial_t u = F[u; x, t], \quad x \in \Omega,\ t \in [0, T]. \tag{1}$$

Here, Ω is the spatial domain of interest and $F[u; x, t]$ represents a general spatiotemporal operator acting on the function $u(x, t)$, including linear combinations of all differential operators acting on u, such as u_x, u_{xx}, u_{xxx}, ..., and spatial convolutional operators.

Although machine learning approaches have been proposed for many types of inverse problems that reconstruct partial differential equations (PDEs) from data [1, 2], most of them make prior assumptions about the specific form of the PDE and use spatial discretization i.e., grids or meshes, of a bounded spatial variable x to approximate the solutions of the PDE. There are three main types of machine-learning-based methods for learning PDEs: (i) methods that use neural networks to reconstruct the RHS of Eq. (1), $F[u; x, t]$, by assuming that it can be well approximated by a (non)linear combination of a class of differential operators, (ii) methods that try to find an explicit mathematical expression for $F[u; x, t]$ by imposing specific forms on F, and (iii) methods that circumvent learning $F[u; x, t]$ by reconstructing a map from the initial condition to the solution at a later time. Long et al. [3] and Churchill et al. [4] used convolutional layers to construct the spatial derivatives of u, then applied a neural network [3, 4] or a symbolic network [5] to approximate $F[u; x, t]$ by $F(x, u_x, u_{xx}, \ldots)$. Sparse identification of nonlinear dynamics (SINDy) [6] and its variants [7, 8] have been developed to learn the dynamics of PDEs by using a sparse regression method to infer coefficients $\mathbf{a}$ in $\partial_t u(x, t) = \mathbf{a} \cdot (1, u, u^2, u_x, u_{xx}, \ldots)$, where $\mathbf{a}$ is the to-be-learned row vector of coefficients associated with each term in the PDE. These methods imposed an additive form for $F[u; x, t]$. Additionally, Fourier neural operator (FNO) [9] and other approaches [10] which learn the mapping between the function space of the initial condition $u_0(\cdot, 0)$ and the function space of the solution $u(\cdot, t)$ within a time range $t \in [t_1, t_2]$ have also been recently developed [10, 11].

In summary, previous methods either assume $F[u; x, t]$ can be approximated by some (non)linear combinations of differential operators, impose a specific form of

$F[u; x, t]$, or circumvent learning $F[u; x, t]$ by reconstructing a map from the initial condition to the solution at a later time. To our knowledge, there has been no method that can extract the dynamics $F[u; x, t]$ from data without making prior assumptions on its form. Moreover, since most prevailing numerical methods for time-dependent DEs rely on spatial discretization via meshes or grids, they cannot be directly applied to problems defined on an unbounded domain [12]. Nonetheless, many physical systems involve the evolution of quantities that experience long-ranged spatial interactions, requiring the solution of spatiotemporal integro-differential equations defined on unbounded domains; i.e., the dynamics F on the RHS of Eq. (1) might contain a spatial convolutional operator. Examples of such spatiotemporal integro-differential equations involve the fractional Laplacian equations for anomalous diffusion [13] and the Keller–Segel equation [14] for describing the swarming behavior. Reconstructing $F[u; x, t]$ on the RHS in Eq. (1) given some observations of the physical quantity $u(x, t)$ can help uncover the physical laws that govern their time evolution.

One major difficulty that prevents direct application of previous methods to unbounded domain problems is that one needs to truncate the unbounded domain and define appropriate boundary conditions [15, 16] on the new artificial boundaries. Although generalizations of the FNO method that include basis functions other than the Fourier series [17] can potentially be applied to unbounded domains, they do not reconstruct the dynamics $F[u; x, t]$. Moreover, they treat x and t in the same way using nonadaptive basis functions which can be inefficient for addressing unbounded-domain spatiotemporal problems where basis functions often need to be dynamically adjusted over time. Recently, an adaptive spectral PINN (s-PINN) method was proposed to solve specified unbounded-domain PDEs [18]. The method expresses the underlying unknown function in terms of adaptive spectral expansions in space and time-dependent coefficients of the basis functions, does not rely on explicit spatial discretization, and can be applied to unbounded domains. However, like many other approaches, the s-PINN approach assumes that the PDE takes the specific form $u_t = F(u, u_x, u_{xx}, ...) + f(x, t)$ in which $F(u, u_x, u_{xx}, ...)$ is known and only the u-independent source term $f(x, t)$ is unknown and to be learned. Therefore, the s-PINN method is limited to parameter inference and source reconstruction.

In this paper, we propose a spectral-based DE learning method that extracts the unknown dynamics in the spatiotemporal DE Eq. (1) by using a parameterized neural network to express $F[u; x, t] \approx F[u; x, t, \Theta]$. Specifically, our spectral-based DE learning approach aims to reconstruct both spatiotemporal PDEs and spatiotemporal PIDEs where the spatial variable x is defined on an unbounded domain. Moreover, our approach does not require prior assumptions on the form of $F[u; x, t]$. Throughout this paper, the term "spatiotemporal DE" will refer to both PDEs and PIDEs in the form of Eq. (1). The formal solution u is then represented by a spectral expansion in space,

$$u(x, t) \approx u_N(x, t) = \sum_{i=0}^{N} c_i(t)\phi_i(x), \tag{2}$$

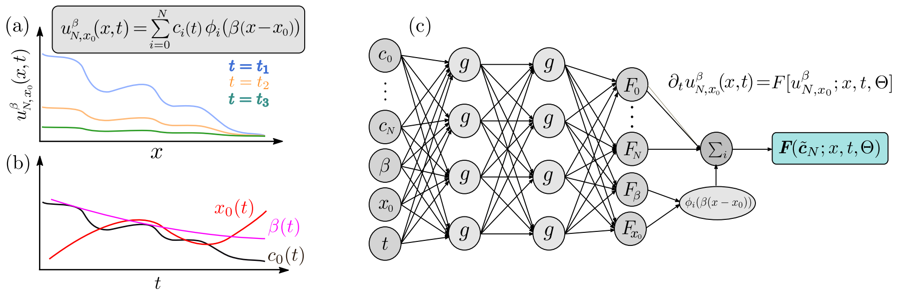
(a)
$u^{\beta}_{N,x_0}(x,t)=\sum_{i=0}^{N} c_i(t)\,\phi_i(\beta(x-x_0))$
$u^{\beta}_{N,x_0}(x,t)$
$t=t_1$
$t=t_2$
$t=t_3$
x
(b)
$x_0(t)$
$\beta(t)$
$c_0(t)$
t
(c)
c_0
c_N
β
x_0
t
g
F_0
F_N
F_β
F_{x_0}
$\partial_t u^{\beta}_{N,x_0}(x,t)=F[u^{\beta}_{N,x_0};x,t,\Theta]$
Σ_i
$\phi_i(\beta(x-x_0))$
$\boldsymbol{F}(\tilde{\boldsymbol{c}}_N;x,t,\Theta)$

Fig. 1 **a** A 1D example of the spectral expansion in an unbounded domain with scaling factor β and displacement x_0 (Eq. (7)). **b** The evolution of the coefficient $c_0(t)$ and the two tuning parameters $\beta(t)$, $x_0(t)$. **c** A schematic of how to reconstruct Eq. (1) satisfied by the spectral expansion approximation u^{β}_{N,x_0}. The time t, expansion coefficients c_i, and tuning variables $\beta(t)$, and x_0 are inputs of the neural network, which then outputs $\boldsymbol{F}(\tilde{\boldsymbol{c}}_N; t, \Theta) = (F_0, ..., F_N, F_\beta, F_{x_0})$. The basis functions $\phi_i\big(\beta(t)(x - x_0(t))\big)$ are shaped by the time-dependent scaling factor $\beta(t)$ and shift parameter $x_0(t)$ which are determined by $\frac{\mathrm{d}\beta}{\mathrm{d}t} \approx F_\beta$ and $\frac{\mathrm{d}x_0}{\mathrm{d}t} \approx F_{x_0}$, respectively

where $\{\phi_i\}_{i=0}^{N}$ is a set of appropriate basis functions that can be defined on bounded or unbounded domains and $\{c_i\}_{i=0}^{N}$ are the associated coefficients. We assume that the spectral expansion coefficients $c_i(t_j), i = 0, ..., N$ in Eq. (2) are given as inputs at various time points $\{t_j\} = t_0, ..., t_M$.

By using the spectral expansion in Eq. (2) to approximate u, we do not need an explicit spatial discretization like spatial grids or meshes, and the spatial variable x can be defined in either bounded or unbounded spatial domains. The best choice of basis functions will depend on the spatial domain. In bounded domains, any set of basis functions in the Jacobi polynomial family, including Chebyshev and Legendre polynomials, provides similar performance and convergence rates; for semibounded domains $\mathbb{R}^+$, generalized Laguerre functions are often used; for unbounded domains $\mathbb{R}$, generalized Hermite functions are used if the solution is exponentially decaying at infinity, while mapped Jacobi functions are used if the solution is algebraically decaying [19, 20].

Additionally, after using the spectral expansion Eq. (2), a numerical scheme for Eq. (1), regardless of whether it is a PDE or a PIDE involving convolution terms in the spatial variable x, can be expressed as ordinary differential equations (ODEs) in the expansion coefficients $\boldsymbol{c}_N(t) := (c_0(t), \ldots, c_N(t))$

$$\frac{\mathrm{d}\boldsymbol{c}_N(t)}{\mathrm{d}t} = \boldsymbol{F}(\boldsymbol{c}_N; t). \tag{3}$$

The spatiotemporal DE learning method proposed here differs substantially from the s-PINN framework because it does not make any assumptions on the form of the spatiotemporal DE in Eq. (1) other than that the RHS F does not contain time-derivatives or time-integrals of $u(x, t)$. Instead, the spectral neural DE method models F directly by a neural network and employs the neural ODE method [21] to fit the trajectories of the ground truth spectral expansion coefficients $\boldsymbol{c}_N(t)$. (see Fig. 1c). The method inputs both the solution $u(x, t)$ (in terms of a spectral expansion) and t into the neural network and applies a neural ODE model [21]. Thus, general DEs such as Eq. (1) can be learned with little knowledge of the RHS. To summarize, the proposed method presented in this paper has the advantage that it

(i) does not require assumptions on the explicit form of F other than it should not contain any time-derivatives or time-integrals of u. Both spatiotemporal PDEs and PIDEs can be learned in a unified way.
(ii) directly learns the dynamics of a spatiotemporal DE (RHS of Eq. (1)) by using a parameterized neural network that can time-extrapolate the solutions, and
(iii) does not rely on explicit spatial discretization and can thus learn *unbounded*-domain DEs. By further using *adaptive* spectral techniques, our neural DE learning method also learns the dynamics of the shaping parameters that adjust the basis functions. Additionally, our neural DE learning method can also take advantage of sparse spectral methods [22] for effectively reconstructing multidimensional spatiotemporal DEs using a reduced number of inputs.

In the next section, we formulate our spatiotemporal DE learning method. In Sect. 3, we use our spatiotemporal DE method to learn the underlying dynamics of DEs. Although our main focus is to address learning unbounded-domain spatiotemporal DEs, we perform benchmarking comparisons on bounded-domain problems that are solved using other recently developed machine-learning based PDE learning methods that apply only in bounded domains. Concluding remarks are given in Sect. 4. Additional numerical experiments and results are given in the Appendix.

2 Spectral spatiotemporal DE learning method

We now formalize our spectral spatiotemporal DE learning method for spatiotemporal DEs of the general structure of Eq. (1), assuming $F[u; x, t]$ does not include time-differentiation or time-integration of $u(x, t)$. However, unlike in [10], the "dynamics" $F[u; x, t]$ on the RHS of Eq. (1) can take any other form including differentiation in space, spatial convolution, and nonlinear terms. Below is a table of the notation used throughout the development of our method and the test examples in the rest of the paper.

First, consider a bounded spatial domain Ω. Suppose we have observational data $u_m(x, t)$, $m = 1, ..., M$ for all x at given time points t_j, $j = 1, ..., T$ associated with different initial conditions $u_m(x, t_0)$, $m = 1, ..., M$. Furthermore, we assume that $u_m(x, t)$ all obey the same underlying, well-posed spatiotemporal DE Eq. (1). Upon choosing proper orthogonal basis functions $\{\phi_i(x)\}_{i=0}^N$, we can approximate $u(x, t)$ by the spectral expansion in Eq. (2) and obtain the spectral expansion coefficients $\boldsymbol{c}_N(t) := (c_0(t), ..., c_N(t))$ as Eq. (3). We aim to reconstruct the dynamics $F(\boldsymbol{c}_N; t)$ in Eq. (3) by using a neural network

$$\boldsymbol{F}(\boldsymbol{c}_N; t) \approx \boldsymbol{F}(\boldsymbol{c}_N; t, \Theta), \tag{4}$$

Table 1 Overview of variables and parameters

Symbol	Definition
N	Spectral expansion order
T	Number of sampled time points
M	Number of different initial conditions/solutions
Θ	Neural network hyperparameters
ϕ_i	ith order basis function in a spatial spectral expansion
$\beta(t)$	Spatial variable scaling factor: $\phi_i(\beta(x-x_0))$
$x_0(t)$	Spatial variable translation factor: $\phi_i(\beta(x-x_0))$
$u_{N,x_0}^{\beta}(x,t)$	Spectral expansion approximation of order N, scaling factor β, translation x_0: $u_{N,x_0}^{\beta}=\sum_{i=0}^{N}c_i(t)\phi_i(\beta(t)(x-x_0(t)))$
$\hat{\mathcal{H}}_i$	Generalized Hermite function of order i

Definitions of the main variables and parameters used in this paper. Besides the expansion order N, the adaptive spectral method for unbounded domain problems employs two additional parameters (per spatial dimension) $\beta(t)$ and $x_0(t)$ that are dynamically adjusted to optimize the spectral approximation

where Θ is the set of parameters in the neural network. We can then construct the RHS of Eq. (1) using

$$F[u;x,t,\Theta]\approx\sum_{i=0}^{N}F_i(\boldsymbol{c}_N;t,\Theta)\phi_i(x) \tag{5}$$

where F_i is the i^{th} component of the vector $\boldsymbol{F}(\boldsymbol{c}_N;t,\Theta)$. We shall use the neural ODE to learn the dynamics $\boldsymbol{F}(\boldsymbol{c}_N;t,\Theta)$ by minimizing the mean loss function $L(u_N(x,t;\Theta),u(x,t))$ between the numerical solution $u_N(x,t;\Theta)$ and the observations $u(x,t)$. When data are provided at discrete time points t_j, we need to minimize

$$\sum_{m=1}^{M}\sum_{j=1}^{T}L\big(u_{N,m}(x,t_j;\Theta),u_m(x,t_j)\big), \tag{6}$$

with respect to Θ. Here, $u_m(x,t_j)$ is the solution at t_j of the m^{th} trajectory in the dataset and $u_{N,m}(x,t_j;\Theta)$ denotes the spectral expansion solution reconstructed from the coefficients $\boldsymbol{c}_{N,s}$ obtained by the neural ODE of the m^{th} solution at t_j.

To solve unbounded domain DEs (in any dimension $\Omega\subseteq\mathbb{R}^D$), two additional sets of parameters are needed to scale and translate the spatial argument $\boldsymbol{x}$, a scaling factor $\boldsymbol{\beta}:=(\beta^1,\ldots,\beta^D)\in\mathbb{R}^D$, and a shift factor $\boldsymbol{x}_0:=(x_0^1,\ldots,x_0^D)\in\mathbb{R}^D$. These factors need to be dynamically adjusted to obtain accurate spectral approximations of the original function [12, 23, 24]. When generalizing the spectral approximation $u_{N,x_0}^{\beta}(x,t)$ in Table 1 to higher spatial dimensions, we can write

$$u(\boldsymbol{x}, t) \approx u^{\boldsymbol{\beta}}_{N,\boldsymbol{x}_0}(\boldsymbol{x}, t) = \sum_{i=0}^{N} c_i(t)\phi_i\big(\boldsymbol{\beta} * (\boldsymbol{x} - \boldsymbol{x}_0)\big), \tag{7}$$

where here, $\boldsymbol{\beta} * (\boldsymbol{x} - \boldsymbol{x}_0) := (\beta^1(x - x_0^1), ..., \beta^D(x - x_0^D))$ is the Hadamard product and $\phi_i(\cdot)$ are D-dimensional basis functions.

Given observed $u(\boldsymbol{x}, t)$, the ground truth coefficients $c_i(t)$ as well as the spectral adjustment parameters $\boldsymbol{\beta}(t)$ and $\boldsymbol{x}_0(t)$ at discrete time points can be obtained by minimizing the *frequency indicator* (introduced in [12])

$$\mathcal{F}(u; \boldsymbol{\beta}, \boldsymbol{x}_0) = \sqrt{\frac{\sum_{i=N-[\frac{N}{3}]+1}^{N} c_i^2}{\sum_{i=0}^{N} c_i^2}} \tag{8}$$

that measures the error of the numerical representation of the solution u [25]. $\mathcal{F}(u; \boldsymbol{\beta}, \boldsymbol{x}_0)$ depends on $\boldsymbol{\beta}$, $\boldsymbol{x}_0$, and the expansion order N through the arguments of the basis functions and thus implicitly through their expansion coefficients c_i. Thus, minimizing $\mathcal{F}(u; \boldsymbol{\beta}, \boldsymbol{x}_0)$ will also minimize the approximation error $\|u - \sum_{i=0}^{N} c_i\phi_i(\boldsymbol{\beta}(t) * (\boldsymbol{x} - \boldsymbol{x}_0(t)))\|_2^2$. Numerically evaluating $c_i(t_j)$ usually requires setting up appropriate collocation points determined by the basis functions and adaptive parameters $\boldsymbol{\beta}$ and $\boldsymbol{x}_0$. In such unbounded domain problems, the ground truth coefficients and adaptive parameters $\tilde{\boldsymbol{c}}_N := \big(c_0(t), ..., c_N(t), \boldsymbol{\beta}(t), \boldsymbol{x}_0(t)\big)$ at times t_j are given as inputs to the neural network.

In addition to $\boldsymbol{c}_N(t)$, evolution of the adaptive parameters $\boldsymbol{\beta}(t)$, $\boldsymbol{x}_0(t)$ over time can also be learned by the neural ODE. More specifically,

$$\frac{\mathrm{d}\tilde{\boldsymbol{c}}_N}{\mathrm{d}t} = \boldsymbol{F}(\tilde{\boldsymbol{c}}_N; t) \tag{9}$$

for the ODEs satisfied by $\tilde{\boldsymbol{c}}_N := \big(\boldsymbol{c}_N(t), \boldsymbol{\beta}(t), \boldsymbol{x}_0(t)\big)$. The underlying dynamics $\boldsymbol{F}(\tilde{\boldsymbol{c}}_N; t)$ is approximated as

$$\boldsymbol{F}(\tilde{\boldsymbol{c}}_N; t) \approx \boldsymbol{F}(\tilde{\boldsymbol{c}}_N; t, \Theta) \tag{10}$$

by minimizing with respect to Θ a loss function that also penalizes the error in $\boldsymbol{\beta}$ and $\boldsymbol{x}_0$

$$\sum_{m=1}^{M}\sum_{j=1}^{T}\Big[L\big(u_m(\boldsymbol{x}, t_j), u^{\boldsymbol{\beta}_m}_{N,\boldsymbol{x}_{0,m},m}(\boldsymbol{x}, t_j; \Theta)\big) + \lambda\,\big\|\boldsymbol{\beta}_m(t_j) - \boldsymbol{\beta}_m(t_j; \Theta)\big\|_2^2 + \lambda\,\big\|\boldsymbol{x}_{0,m}(t_j) - \boldsymbol{x}_{0,m}(t_j; \Theta)\big\|_2^2\Big]. \tag{11}$$

Similarly, the DE satisfied by $u^{\boldsymbol{\beta}}_{N,\boldsymbol{x}_0}(\boldsymbol{x}, t)$ is

$$\partial_t u^{\boldsymbol{\beta}}_{N,\boldsymbol{x}_0}(\boldsymbol{x}, t) = F[u^{\boldsymbol{\beta}}_{N,\boldsymbol{x}_0}; \boldsymbol{x}, t, \Theta], \tag{12}$$

where

$$
\begin{aligned}
F[u^{\boldsymbol{\beta}}_{N,\boldsymbol{x}_0}; \boldsymbol{x}, t, \Theta] = & \sum_{i=0}^{N} F_i(\tilde{\boldsymbol{c}}_N; t, \Theta)\phi_i\big(\boldsymbol{\beta}(t) * (\boldsymbol{x} - \boldsymbol{x}_0(t))\big) \\
& + \sum_{i=0}^{N} c_i(t) \sum_{j=1}^{D} \Big(\partial_t \Big(\beta^j(t)\big(x^j - x_0^j(t)\big)\Big)\, \partial_{x^j}\phi_i\big(\boldsymbol{\beta}(t) * (\boldsymbol{x} - \boldsymbol{x}_0(t))\big)\Big)
\end{aligned}
\tag{13}
$$

and F_i is the i^{th} component of $\boldsymbol{F}(\tilde{\boldsymbol{c}}_N; t, \Theta)$. Here, $\boldsymbol{\beta}_m(t_j)$ and $\boldsymbol{x}_{0,m}(t_j)$ are the scaling factor and the displacement of the m^{th} sample at time t_j, respectively, and λ is the penalty due to squared mismatches in the scaling and shift parameters β and x_0. In this way, the dynamics of the variables $\boldsymbol{x}_0, \boldsymbol{\beta}$ are also learned by the neural ODE so they do not need to be manually adjusted as they were in [12, 18, 24, 25].

If the space Ω is high-dimensional, sufficiently smooth and well-behaved solutions can be approximated by restricting the basis functions $\{\phi^{\beta}_{i,x_0}\}$ to those in the hyperbolic cross space. If this projection is performed optimally, such sparse spectral methods with spectral expansions defined in the hyperbolic cross space can reduce the effective dimensionality of the problem by leaving out redundant basis functions [22, 26] without significant loss of accuracy. We will show that our method can also easily incorporate the hyperbolic cross spaces to enhance training efficiency in modestly higher-dimensional problems.

3 Numerical experiments

In this work, we set $L(\cdot, \cdot)$ to be the relative squared L^2 error

$$
L(u(x, t_i), u_N(x, t_i; \Theta)) := \frac{\big\|u(x, t_i) - u_N(x, t_i; \Theta)\big\|_2^2}{\|u\|_2^2} \tag{14}
$$

in the loss function Eq. (6) used for training. We carry out numerical experiments to test our spectral spatiotemporal DE reconstruction method by learning the underlying DE given data in both bounded and unbounded domains. In this section, we use the `odeint_adjoint` function along with the `dopri5` numerical integration method developed in the `torchdiffeq` package [21] to numerically integrate Eqs. (3) and (9). Stochastic gradient descent (SGD) and the Adam optimizer are used separately to optimize parameters of the neural network. Computations for all numerical experiments were implemented on a 4-core Intel® i7-8550U CPU, 1.80 GHz laptop using Python 3.8.10, Torch 1.12.1, and Torchdiffeq 0.2.3.

Since algorithms already exist for learning bounded-domain PDEs, we first examine a bounded-domain problem in order to benchmark our spatiotemporal DE method against two other recent representative methods, a convolutional neural PDE learner [10] and a Fourier neural operator PDE learning method [11].

Example 1 For our first example, we consider learning a bounded-domain Burgers' equation that describes the behavior of viscous fluid flow [27]. This example illustrates the performance of our spatiotemporal DE method in learning bounded-domain PDEs and benchmarks our approach against some recently developed methods.

$$\begin{aligned} \partial_t u + \tfrac{1}{2}\partial_x(u^2) = \tfrac{1}{10}\partial_{xx}u, \quad x \in (-1, 1),\ t \geq 0, \\ u(-1, t) = u(1, t),\ \ \partial_x u(-1, t) = \partial_x u(1, t),\ \ u(x, 0) = -\tfrac{1}{5}\frac{\psi_x(x, 0)}{\psi(x, 0)}, \end{aligned} \tag{15}$$

where

$$\psi(x, t) \equiv 5 + \left(\tfrac{2+\xi_1}{2}\right)e^{-\pi^2 t/10}\sin\pi x + \tfrac{\xi_2}{2}e^{-2\pi^2 t/5}\cos 2\pi x. \tag{16}$$

This model admits the analytic solution expressible as $u(x, t) = -\frac{\psi_x(x,t)}{5\psi(x,t)}$. We then sample two independent random variables from $\xi_1, \xi_2 \sim \mathcal{U}(0, 1)$ to generate a class of solutions to Eq. (16) $\{u\}_{\xi_1,\xi_2}$ for both training and testing. To approximate F in Eq. (4), we use a neural network that has one intermediate layer with 300 neurons and the ELU activation function. The basis functions in Eq. (2) are taken to be Chebyshev polynomials. For training, we use 200 solutions (each corresponding to an independently randomly sampled pair (ξ_1, ξ_2) of Eq. (15)) and record the expansion coefficients $\{c_i\}_{i=0}^{9}$ at different times $t_j = j\Delta t$, $\Delta t = \frac{1}{4}$, $j = 0, \ldots, 4$. The test set consists of 100 more solutions, also evaluated at times $t_j = j\Delta t$, $\Delta t = \frac{1}{4}$, $j = 0, \ldots, 4$.

In this bounded-domain problem, we can compare our results (the generated solutions $u(x, t)$) with those generated from the Fourier neural operator (FNO) and the convolutional neural PDE learner methods. In the FNO method, four intermediate Fourier convolution layers with 128 neurons in each layer were used to input the initial condition $u(i\Delta x, 0)$. Then, the FNO method outputs the function values $u(i\Delta x, t = j\Delta t)$ (with $\Delta x = \frac{1}{128}$, $\Delta t = \frac{1}{4}$) for $j > 0$ [11].

When implementing the convolutional neural PDE solver [10], we input $u(i\Delta x, (j-1)\Delta t)$ and $u(i\Delta x, j\Delta t)$ (with $\Delta x = \frac{1}{100}$, $\Delta t = \frac{1}{250}$) [10] into seven convolutional layers with 40 neurons in each layer which outputs $u(i\Delta x, (j+1)\Delta t)$ as the numerical solution at the next time step. Small Δx and Δt are used in the convolutional neural PDE solver method because this method depends on both spatial and temporal discretization, requiring fine discretization meshes in both dimensions. For all three methods, we used the Adam method to perform gradient descent with a learning rate $\eta = 0.001$ to run 10000 epochs, which was sufficient for the errors in all three methods to converge. We list in Table 2 the mean relative L^2 error

$$\frac{1}{MT}\sum_{m=1}^{M}\sum_{j=1}^{T}\frac{\left\|u_{N,m}(x, t_j; \Theta) - u_m(x, t_j)\right\|_2}{\|u_m(x, t_j)\|_2}. \tag{17}$$

For the FNO and spectral PDE learning methods, we aim to minimize the relative squared L^2 loss (Eq. (14)), while for the convolution neural PDE solver method, we must minimize the MSE loss since only partial and local spatial information on the solution is inputted during each training epoch so we cannot properly define a relative squared loss as the relative squared loss Eq. (14) needs global spatial information

Table 2 The convolutional PDE solver, the Fourier neural operator method, and our proposed spatiotemporal DE learner are used to learn or solve the dynamics of Burgers' equation Eq. (15) in a *bounded* domain

Method	Training error	Testing error
	Mean relative L^2	Mean relative L^2
Convolutional	1.68e−02 ± 7.86e−03	2.82e−02 ± 6.62e−03
Fourier	7.43e−03 ± 1.76e−03	8.61e−03 ± 2.86e−03
Spectral	9.82e−03 ± 4.95e−03	9.99e−03 ± 4.97e−03

The FNO method gives the best performance and the convolutional neural PDE solver performs the worst. Our proposed spatiotemporal DE learner achieves performance comparable to the FNO method

to calculate $\|u\|_2$. As shown in Table 2, the relative L^2 error of the FNO method is smaller than the MSEs of the other two methods on the training set while the convolutional neural PDE solver method performs the worst. Nonetheless, our proposed neural spectral DE learning approach performs comparably to the FNO method, giving comparable mean relative L^2 errors for learning the dynamics associated with the bounded-domain Burgers' equation, but can also generate new solutions given different initial conditions.

Additionally, the run times (using a 4-core i7-8550U laptop) in this example were $\sim$ 2 hours for the convolutional PDE solver method, $\sim$ 6 hours for the FNO method, and $\sim$ 5 hours for our proposed spatiotemporal DE learning approach. Overall, even in bounded domains, our proposed neural DE learning approach compares well with the recently developed convolutional neural PDE solver and FNO methods, providing comparable errors and efficiency in generating solutions to Eq. (15) given different initial conditions.

The Fourier neural operator method works well for solving Burgers' equation in Example 1, and there could be other even more efficient methods for reconstructing bounded domain spatiotemporal DEs. However, reconstructing unbounded domain spatiotemporal DEs is substantially different from reconstructing bounded domain counterparts. First, discretizing space cannot be directly applied to unbounded domains; second, if we truncate an unbounded domain into a bounded domain, appropriate artificial boundary conditions need to be imposed [15]. Constructing such boundary conditions is usually complex; improper boundary conditions can lead to large errors. A simple example of when the FNO will fail when we truncate the unbounded domain into a bounded domain is provided in Appendix A.

Since our spectral method uses basis functions, it obviates the need for explicit spatial discretization and can be used to reconstruct unbounded-domain DEs. Dynamics in unbounded domains are intrinsically different from their bounded-domain counterparts because functions can display diffusive and convective behavior leading to, e.g., time-dependent growth at large x. This growth poses intrinsic numerical challenges when using prevailing finite element/finite difference methods that truncate the domain.

Although it is difficult for most existing methods to learn the dynamics in unbounded spatial domains, our spectral approach can reconstruct unbounded-domain DEs by simultaneously learning the expansion coefficients and the evolution of the basis functions. To illustrate this, we next consider a one-dimensional unbounded domain inverse problem.

Example 2 Here, we examine a parabolic PDE in an unbounded domain and with initial conditions that depend on parameters ξ_1, ξ_2, ξ_3. The PDE and its initial condition are given by

$$\partial_t u = -\partial_x u + \tfrac{1}{4}\partial_{xx} u, \quad u(x,0) = \frac{\xi_1}{\sqrt{1+\xi_2}} \exp\left(-\frac{(x-\xi_3)^2}{1+\xi_2}\right). \tag{18}$$

This example illustrates the application of our method to learning the dynamics of a parabolic PDE given different ground truth solutions in an unbounded domain corresponding to different initial conditions. The solution of the PDE, within the domain $x \in \mathbb{R}$ and time interval $t \in [0, 1]$, is expressed as

$$u(x,t;\xi_1,\xi_2,\xi_3) = \frac{\xi_1}{\sqrt{t+1+\xi_2}} \exp\left(-\frac{(x-t-\xi_3)^2}{t+1+\xi_2}\right). \tag{19}$$

Since this problem is defined on an unbounded domain, neither the FNO nor the convolutional neural PDE methods can be used as they rely explicitly on spatial meshes or grids and apply only on bounded domains. However, given observational data $u(\cdot, t)$ for different t, we can calculate the spectral expansion of u via the generalized Hermite functions [20]

$$u(x,t) \approx u_{N,x_0}^{\beta} = \sum_{i=0}^{N} c_i(t)\hat{\mathcal{H}}_i\big(\beta(t)(x - x_0(t))\big) \tag{20}$$

and then use the spatiotemporal DE learning approach to reconstruct the dynamics F in Eq. (1) satisfied by u. Recall that the scaling factor $\beta(t)$ and the displacement of the basis functions $x_0(t)$ are also to be learned. To penalize misalignment of the spectral expansion coefficients and the scaling and displacement factors β and x_0, we use the loss function Eq. (11). Note that taking the derivative of the first term in Eq. (11) would involve evaluating the derivative of Eq. (14) which would require evaluation of integrals such as $\int \frac{u_{N,x_0}^{\beta}(x,t_j;\Theta) - u(x,t_j)}{\|u\|_2^2} \partial_x u_{N,x_0}^{\beta}(x,t_j;\Theta)\partial_\Theta\big[\beta(t_j;\Theta)(x - x_0(t_j;\Theta))\big]\mathrm{d}x$. Expressing $\partial_x u_{N,x_0}^{\beta}(x,t_j;\Theta)$ in terms of the basis functions $\hat{\mathcal{H}}_i\big(\beta(t)(x - x_0(t))\big)$ would involve a dense matrix–vector multiplication of the coefficients of the expansion $\partial_x u_{N,x_0}^{\beta}(x,t_j;\Theta)\partial_\Theta\big[\beta(t_j;\Theta)(x - x_0(t_j;\Theta))\big]$, which might be computationally expensive during backward propagation in the stochastic gradient descent (SGD) procedure.

Alternatively, let the neural network parameter after the $(j-1)^{\text{th}}$ training epoch be Θ_{j-1}. During the calculation of the gradient of the loss function Eq. (11) w.r.t. Θ at the j^{th} epoch, we define $\tilde{\beta}(t_j) := \beta(t_j;\Theta_{j-1}), \tilde{x}_0(t_j) := x_0(t_j;\Theta_{j-1})$ to be constants independent of Θ and then modify Eq. (11) to

$$\sum_{m=1}^{M}\sum_{j=1}^{T}\Bigg[\frac{\big\|u_{N,\tilde{x}_{0,m}(t_j)}^{\tilde{\beta}_m(t_j)}(x,t_j;\Theta)-u_m(x,t_j)\big\|_2^2}{\|u_m(x,t_j)\|_2^2} \\ +\lambda\big(\beta_m(t_j;\Theta)-\beta_m(t_j)\big)^2+\lambda\big(x_{0,m}(t_j;\Theta)-x_{0,m}(t_j)\big)^2\Bigg], \tag{21}$$

so that backpropagation within each epoch will *not* involve calculating gradients of $\tilde{\beta}_m(t_j), \tilde{x}_{0,m}(t_j)$ in the first term of Eq. (21). This simplified calculation reduces the computational cost of the training process but can provide gradients close to the true gradients when the reconstructed $\tilde{\beta}(t_j), \tilde{x}_0(t_j)$ are close to the ground truth values $\beta_m(t), x_{m,0}(t)$. For example, when $\beta_m(t;\Theta_{j-1})=\beta_m(t), x_{m,0}(t;\Theta_{j-1})=x_{m,0}(t)$, i.e., the reconstructed $\beta_m(t;\Theta_{j-1}), x_{m,0}(t;\Theta_{j-1})$ agree exactly with the ground truth, Eq. (11) and Eq. (21) will both become

$$\sum_{m=1}^{M}\sum_{j=1}^{T}\frac{\big\|u_{N,x_{0,m}(t_j)}^{\beta_m(t_j)}(x,t_j;\Theta)-u_m(x,t_j)\big\|_2^2}{\big\|u_m(x,t_j)\big\|_2^2}=\sum_{m=1}^{M}\sum_{j=1}^{T}\frac{\sum_{i=0}^{N}\big(c_{m,i}(t_j;\Theta)-c_i(t_j)\big)^2}{\sum_{i=0}^{N}c_{m,i}(t_j)^2}. \tag{22}$$

No derivative of β, x_0 w.r.t. Θ will be used and only the gradient of F in Eq. (4) w.r.t. Θ appears. In this case, the simplified gradient exactly reflects the true gradient. Therefore, we can fit the coefficients $c_i(t)$ and $\beta(t), x_0(t)$ separately, and then use the simplified loss gradient to update the neural network parameters.

We use 100 solutions for training and another 50 solutions for testing with $N=9, \Delta t=0.1, T=9, \lambda=0.1$. Each solution is generated from Eq. (19) with independently sampled parameters ξ_1, ξ_2, ξ_3. A neural network with two hidden layers, 200 neurons in each layer, and the ELU activation function is used for training. Both training and testing data are taken from Eq. (19) with sampled parameters $\xi_1\sim\mathcal{N}(3,\frac{1}{4}),\ \xi_2\sim\mathcal{U}(0,\frac{1}{2}),\ \xi_3\sim\mathcal{N}(0,\frac{1}{2})$.

Setting $\lambda=0.1$, we first compare the two different loss functions Eqs. (11) and (21). After running 10 independent training processes using SGD, each containing 2000 epochs and using a learning rate $\eta=0.0002$, the average relative L^2 error when using the loss function Eq. (11) are larger than the average relative L^2 errors when using the loss function Eq. (21). This difference arises in both the training and testing sets as shown in Fig. 2a.

In Fig. 2b, we plot the average learned F (RHS in Eq. (1)) for a randomly selected sample at $t=0$ in the testing set. The dynamics learned by using Eq. (21) is a little more accurate than that learned by using Eq. (11). Also, using the loss function Eq. (21) required only $\sim$ 1 hour of computational time compared to 5 days when learning with Eq. (11) (on the 4-core i7-8550U laptop). Therefore, for efficiency and accuracy, we adopt the revised loss function Eq. (21) and separately fit the dynamics of the adaptive spectral parameters (β, x_0) and the dynamics of the spectral coefficients c_i.

We also explore how network architecture and regularization affect the reconstructed dynamics. The results are shown in Appendix B, from which we observe that a wider and shallower neural network with 2 or 4 intermediate layers and 200 neurons in each layer yields the smallest errors on both the training and testing sets,

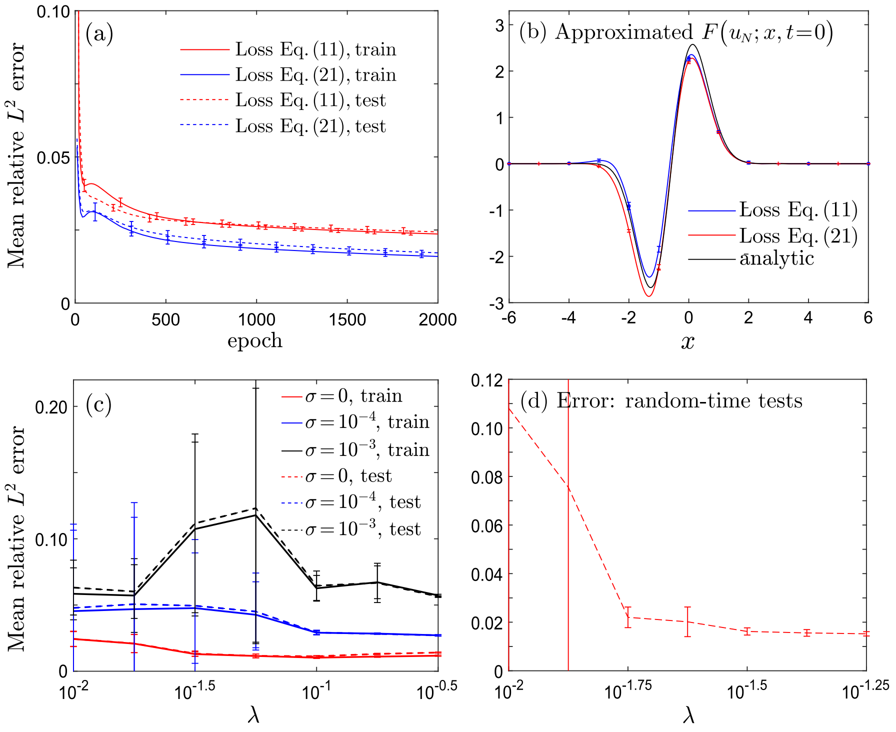

Fig. 2 **a** Errors using the different loss functions Eqs. (11) and (21). **b** Average dynamics found from using Eqs. (11) and (21). **c** Errors with λ and σ. **d** Errors on the testing set with random times $t_i \sim \mathcal{U}(0, 1.5)$

and short run times. We also apply a ResNet [28] as well as the dropout technique [29, 30] to regularize the neural network structure. Dropout regularization does not reduce either the training error or the testing error probably because even with a feedforward neural network, the errors from our spatiotemporal DE learner on the training set are close to those on the testing set and there is no overfitting issue. On the other hand, applying the ResNet technique leads to about a 20% decrease in errors. Results from using ResNets and dropout are shown in Appendix B.

Next, we investigate how noise in the observed data and changes in the adaptive parameter penalty coefficient λ in Eq. (21) impact the results. Noise is incorporated into simulated observational data as

$$u_\xi(x, t) = u(x, t)\big[1 + \xi(x, t)\big], \tag{23}$$

where $u(x, t)$ is the solution to the parabolic equation Eq. (18) given by Eq. (19) and $\xi(x, t) \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian-distributed noise that is both spatially and temporally uncorrelated (i.e., $\langle \xi(x, t)\xi(y, s)\rangle = \sigma^2\delta_{x,y}\delta_{s,t}$). The noise term is assumed to be independent for different samples. We use a neural network with 2 hidden layers, 200 neurons in each layer, to implement 10 independent training processes using SGD and a learning rate $\eta = 0.0002$, each containing 5000 epochs. Results are shown in

Fig. 2c and further tabulated in Appendix C. For $\sigma = 0$, choosing an intermediate $\lambda \in (10^{-1.5}, 10^{-1}]$ leads the smallest errors and an optimal balance between learning the coefficients c_i and learning the dynamics of β, x_0. When σ is increased to nonzero values ($\sim 10^{-4} - 10^{-3}$), a larger $\lambda \sim 10^{-0.75} - 10^{-0.5}$ is needed to keep errors small (see Fig. 2c and Appendix C). If the noise is further increased to, say, $\sigma = 10^{-2}$ (not shown in Fig. 2), an even larger $\lambda \sim 10^{-0.5}$ is needed for training to converge. This behavior arises because the independent noise $\xi(x,t) \sim \mathcal{N}(0, \sigma^2)$ contributes more to high-frequency components in the spectral expansion. In order for training to converge, fitting the shape of the basis functions by learning β, x_0 is more important than fitting noisy high-frequency components via learning c_i. A larger λ puts more weight on learning the dynamics of β, x_0 and basis function shapes.

We also investigate how intrinsic noise in the parameters ξ_1, ξ_2, ξ_3 affects the solution (Eq. (19)) and the accuracy of the learned DE. As shown in D, we find that if the intrinsic noise in ξ_1, ξ_2, ξ_3 is increased, the training errors of the learned DE models also increase. However, compared to models trained on data with lower ξ_1, ξ_2, ξ_3, training using noisier data leads to lower errors when testing data are also noisy. Additionally, we explore how the number of solutions in the training set impacts how the learned DE model makes new predictions given the initial conditions of solutions in the testing set. The results are also listed in Appendix D and show that larger numbers of training samples (solutions associated with different (ξ_1, ξ_2) in Eq. (19)) lead to smaller relative L^2 errors of the predicted solutions in both the training and testing sets.

Finally, we test whether the parameterized F (Eq. (10)) learned from the training set can extrapolate well beyond the training set sampling interval $t \in [0, 0.9]$. To do this, we generate another 50 trajectories and sample each of then at random times $t_i \sim \mathcal{U}(0, 1.5), i = 1, ..., 9$. We then use models trained with $\sigma = 0$ and different λ to test. As shown in Fig. 2d, our spatiotemporal DE learner can accurately extrapolate the solution to times beyond the training set sampling time intervals. We also observe that a stronger penalty on β and x_0 ($\lambda = 10^{-0.5}$) leads to better extrapolation results.

In the last example, we carry out a numerical experiment on learning the evolution of a Gaussian wave packet (which may depend on nonlocal interactions) across a two-dimensional unbounded domain $(x, k) \in \mathbb{R}^2$. We use this case to explore improving training efficiency by using a hyperbolic cross space to reduce the number of coefficients in multidimensional problems.

Example 3 We solve a 2D unbounded-domain problem of fitting a Gaussian wave packet's evolution

$$f(x, k, t; \xi_1, \xi_2) = 2e^{-\frac{(x-\xi_1)^2}{2a^2}} e^{2bt(x-\xi_1)(k-\xi_2)} e^{-2a^2(1+b^2t^2)(k-\xi_2)^2}, \tag{24}$$

where ξ_1 is the center of the wave packet and a is the minimum positional spread. If $\xi_2 = 0$, the Gaussian wave packets defined in Eq. (24) solves the stationary zero-potential Wigner equation, an equation often used in quantum mechanics to describe the evolution of the Wigner quasi-distribution function [31, 32]. We set $a = 1$ and $b = \frac{1}{2}$ in Eq. (24) and independently sample $\xi_1, \xi_2 \sim \mathcal{U}(-\frac{1}{2}, \frac{1}{2})$ to generate data. Thus, the DE satisfied by the highly nonlinear Eq. (24) is unknown and potentially involves nonlocal convolution terms. In fact, there could be infinitely many DEs, including complicated nonlocal DEs, that can describe the dynamics of Eq. (24). An example of such a nonlocal DE is

$$
\begin{aligned}
&\partial_t f + 2a^2 b A[f; x, k, t]\, \partial_x f(x, k, t) = 0,\\
&A[f; x, k, t] = \frac{bt(x - B[f; x, k, t])}{2a^2(1 + b^2t^2)} + \sqrt{\frac{\log D[f; k, t] - C(t) - \log(f/2)}{a^2(1 + b^2t^2)}},\\
&B[f; x, k, t] = x - \sqrt{2a^2(1 + b^2t^2)}\sqrt{2C(t) - 2\log D[f; x, k, t] + \log(f/2)},\\
&C(t) = \frac{1}{2}\log\Big[\frac{\pi}{a^2(1 + b^2t^2)}\Big],\\
&D[f; x, k, t] = \int f(x, y, t) e^{-2a^2(1+b^2t^2)(y-k)^2} \mathrm{d}y.
\end{aligned}
\tag{25}
$$

We wish to learn the underlying dynamics using a parameterized F in Eq. (10). Since the Gaussian wave packet Eq. (24) is defined in the unbounded domain $\mathbb{R}^2$, learning its evolution requires information over the entire domain. Thus, methods that depend on discretization of space are not applicable.

Our numerical experiment uses Eq. (24) as both training and testing data. We take $\Delta t = 0.1, t_j = j\Delta t$, $j = 0, ..., 10$ and generate 100 solutions for training. For testing, we generate another 50 solutions, each with starting time $t_0 = 0$ but t_j taken from $\mathcal{U}(0, 1)$, $j = 1, \ldots, 10$. The parameters ξ_1, ξ_2 in the solutions Eq. (24) are independently sampled for both the training set and the testing set. For this example, training with ResNet results in diverging gradients, whereas the use of a feedforward neural network yields convergent results. So we use a feedforward neural network with two hidden layers and 200 neurons in each hidden layer and the ELU activation function. We train across 1000 epochs using SGD with momentum (SGD M), a learning rate $\eta = 0.001$, momentum $= 0.9$, and weight decay $= 0.005$. We use a spectral expansion in the form of a two-dimensional tensorial product of Hermite basis functions $\hat{\mathcal{H}}_i\hat{\mathcal{H}}_\ell$

$$
f_N(x, k, t_j; \xi_1, \xi_2) = \sum_{i=0}^{14}\sum_{\ell=0}^{14} c_{i,\ell}(t_j)\hat{\mathcal{H}}_i(\beta^1(x - x_0^1))\hat{\mathcal{H}}_\ell(\beta^2(k - k_0^2)) \tag{26}
$$

to approximate Eq. (24). We record the coefficients $c_{i,\ell}$ as well as the scaling factors and displacements $\beta^1, \beta^2, x_0^1, k_0^2$ at different t_j as the training data.

Because $(x, k) \in \mathbb{R}^2$ are defined in a 2-dimensional space, instead of a tensor product, we can use a hyperbolic cross space for the spectral expansion to effectively reduce the total number of basis functions while preserving accuracy [22]. Similar to the use of sparse grids in the finite element method [33, 34], choosing basis functions in the space

$$
\begin{aligned}
V_{N,\gamma}^{\boldsymbol{\beta},\boldsymbol{x}_0} :=&\text{span}\Big\{\hat{\mathcal{H}}_{n_1}(\beta^1(x - x_0^1))\hat{\mathcal{H}}_{n_2}(\beta^2(k - k_0^2)) : |\boldsymbol{n}|_{\text{mix}}\|\boldsymbol{n}\|_\infty^{-\gamma} \le N^{1-\gamma}\Big\},\\
\boldsymbol{n} :=&(n_1, n_2),\ \ |\boldsymbol{n}|_{\text{mix}} := \max\{n_1, 1\}\max\{n_2, 1\}
\end{aligned}
\tag{27}
$$

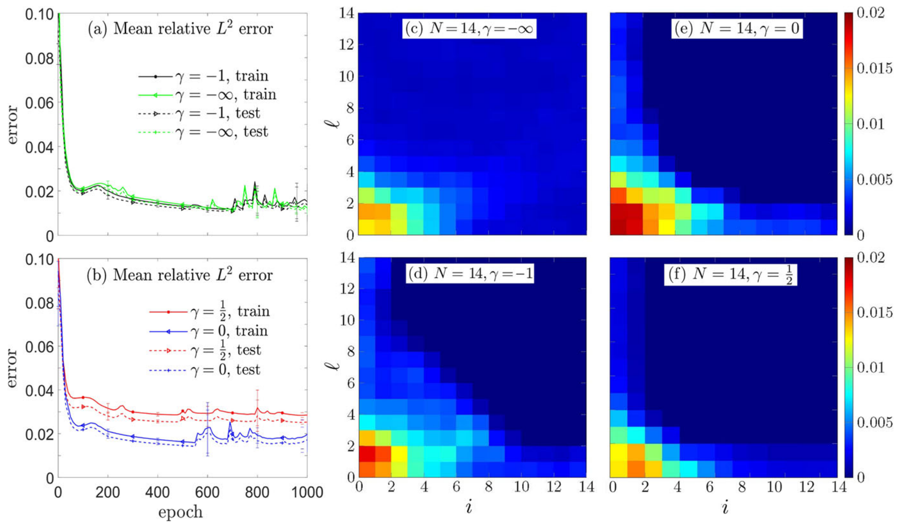

Fig. 3 **a**, **b** Mean relative L^2 errors for $N = 14$ and $\gamma = -\infty, -1, 0, 1/2$. **c–f** Saliency maps showing the mean absolute values of the partial derivative of the loss function w.r.t. to $\{c_{i,\ell}(0)\}$ for $\gamma = -\infty, -1, 0, 1/2$

can reduce the effective dimensionality of the problem. We explored different hyperbolic spaces $V_{N,\gamma}^{\boldsymbol{\beta},\boldsymbol{x}_0}$ with different N and γ. We use the loss function Eq. (21) with $\lambda = \frac{1}{50}$ for training. The results are listed in Appendix E. To show how the loss function Eq. (21) depends on the coefficients $c_{i,\ell}$ in Eq. (26), we plot saliency maps [35] for the quantity $\frac{1}{10}\sum_{j=1}^{10}\left|\frac{\partial \text{Loss}_j}{\partial c_{i,\ell}(0)}\right|$[1], the absolute value of the partial derivative of the loss function Eq. (21) w.r.t. $c_{i,\ell}$ averaged over 10 training processes.

As shown in Fig. 3a, b, using $\gamma = -1, 0$ leads to similar errors as the full tensor product $\gamma = -\infty$, but could greatly reduce the number of coefficients and improve training efficiency. Taking a too large $\gamma = 1/2$ leads to larger errors because useful coefficients are left out. From Fig. 3c–f, there is a resolution-invariance for the dependence of the loss function on the coefficients $c_{i,\ell}$ though using different hyperbolic spaces with different γ. We find that an intermediate $\gamma \in (-\infty, 1)$ (e.g., $\gamma = -1, 0$) can be used to maintain accuracy and reduce the number of inputs/outputs when reconstructing the dynamics of Eq. (24). Overall, the "curse of dimensionality" can be mitigated by adopting a hyperbolic space for the spectral representation.

Finally, in Appendix F, we consider source reconstruction in a heat equation. Our proposed spatiotemporal DE learning method achieves an average relative error $L^2 \approx 0.1$ in the reconstructed source term. On the other hand, if all terms on the RHS of Eq. (1) except an unknown source (which does not depend on the solution) are known, the recently developed s-PINN method [18] achieves a higher accuracy. However, if in addition to the source term, additional terms on on the RHS of Eq. (1) are unknown,

[1] We take derivatives w.r.t. only the coefficients $\{c_{i,\ell}(0)\}$ of the predicted $u_{N,\boldsymbol{x}_0(0),m}^{\boldsymbol{\beta}_m(0)}(x, 0; \Theta)$ in Eq. (21) and not w.r.t. the expansion coefficients of the observational data $u(x, 0)$.

s-PINNs cannot be used but our proposed spatiotemporal DE learning method remains applicable.

4 Conclusions

In this paper, we propose a spatiotemporal DE learning method that is quite suitable for learning spatiotemporal DEs from spectral expansion data of the underlying solution. Its main advantage is its applicability to learning both spatiotemporal PDEs and integro-differential equations in *unbounded domains*, while matching the performance of the most recent high-accuracy PDE learning methods applicable to only bounded domains. Moreover, our proposed method has the potential to deal with higher-dimensional problems if a proper hyperbolic cross space can be justified to effectively reduce the dimensionality.

In future investigations, we plan to apply our spatiotemporal DE learning method to many other inverse-type problems in physics with other appropriate basis functions in unbounded domains, such as the mapped Jacobi functions that characterize algebraic decay at infinity [36–38], the radial basis functions [39–41], or the Laguerre functions on the semi-unbounded half line $\mathbb{R}^+$ [42–44]. A potentially interesting application is to learn the evolution of probability densities associated with anomalous diffusion [45] in an unbounded domain, which is often described by fractional derivatives or convolutional terms in the corresponding $F[u; x, t]$ term. Finally, higher dimensional problems remain challenging since the number of inputs (expansion coefficients) grows exponentially with spatial dimension and the computational cost in may not be sufficiently mitigated by the optimal hyperbolic cross space indices N, γ (see Eq. (27)). Two possible ways to address this issue are promising. First, prior knowledge on the observed data can be used to reduce the dimension of the unknown dynamics to be learned, e.g., if we can determine an optimal hyperbolic cross space for the spectral expansion from data, we can effectively reduce the number of basis functions needed. Second, deep neural networks [46, 47], which can effectively handle a large number of inputs, could be adopted when the number of spectral expansion coefficients becomes large. Exploring these directions can further extend the applicability of our proposed spatiotemporal DE learning method to higher-dimensional problems.

A Using Fourier neural operator to solve unbounded domain DEs

We shall show through a simple example that it is usually difficult to apply bounded domain DE solve methods to reconstruct unbounded domain DEs even if we truncate the unbounded domain into a bounded domain, because appropriate boundary conditions must be provided. Here, we show how the FNO method fails to generalize well on the testing set when solutions to a PDE with a wrong boundary condition are inputted as training data. We wish to use the Fourier neural operator method to solve the unbounded domain DE

$$u_t = \tfrac{1}{4}u_{xx}, \quad x \in \mathbb{R}, \quad t \in [0, 1]. \tag{28}$$

Table 3 Training and testing errors when using the FNO method and truncating the unbounded domain. However, an incorrect boundary condition for the training set is imposed

	Training error		Testing error	
	MSE	Mean relative L^2	MSE	Mean relative L^2
Fourier	3.78e−5 ± 1.78e−5	6.09e−3 ± 1.47e−3	8.95e−4 ± 6.9e−5	3.07e−2 ± 1.2e−3

The error is significantly larger on the testing set than that on the training set because a different DE is reconstructed from the training data

If one imposes the initial condition $u(x, 0) = 10\xi e^{-100x^2}$, then

$$u(x,t) = \frac{\xi}{\sqrt{0.01+t}} \exp\left(-\frac{x^2}{0.01+t}\right) \tag{29}$$

is the analytic solution to Eq. (28). For this problem, we will assume $\xi \sim \mathcal{U}(1, \frac{3}{2})$.

Since the Fourier neural operator (FNO) method relies on spatial discretization and grids, and cannot be directly applied to unbounded domain problems, we truncate the unbounded domain. Suppose one is interested in the solution's behavior for $x \in [-1, 1]$. One approach is to truncate the unbounded domain $x \in \mathbb{R}$ to $[-1, 1]$ and use the FNO method to reconstruct the solution $u(x, t)$, $x \in [-1, 1], t \in [0, 1]$ given $u(x, 0)$. However, we show how improper boundary conditions of the truncated domain can leads to large errors.

For example, we assume the training set satisfies the boundary condition $u(x = \pm 1, t) = 0$, which is not the correct boundary condition since it is inconsistent with the ground truth solution. Therefore, we would **not** be solving the model in Eq. (28). We generate the testing dataset using the correct initial condition $u(x, 0) = 10\xi \exp(-100x^2)$, without boundary conditions. The results are given in Table 3.

From Table 3, the testing error is significantly larger than the training error because a different DE (not Eq. (28)) is constructed from the training data, which is not the heat equation we expect. Therefore, even if methods such FNO are efficient in bounded domain DE reconstruction problems, directly using them to reconstruct unbounded domain problems is not feasible if we cannot construct appropriate boundary conditions.

B Dependence on neural network architecture

The neural network structure of the parameterized $F[u; x, t, \Theta]$ may impact learned dynamics. To investigate how the neural network structure influences results, we use neural networks with various configurations to learn the dynamics of Eq. (19) in Example 2 in the noise-free limit. We set the learning rate $\eta = 0.0002$ and apply networks with 2,3,5,8 intermediate layers, and 50, 80, 120, 200 neurons in each layer.

From Tables 4 and 5, we see that a shallower and wider neural network yields the smallest error. Runtimes increase with the number of layers and the number of neurons in each layer; however, when the number of layers is small, the increase in runtime

Table 4 The relative L^2 errors on the training set and testing set (in parentheses) for Example 2 when there is no noise in both the training data and the testing data ($\sigma = 0$). $\lambda = 0.1$ in Eq. (21), and the training rate is set to be $\eta = 0.0005$ for 5000 training epochs using SGD

Neurons	Layers			
	2	3	5	8
50	0.0166 (0.0179)	0.0175 (0.0196)	0.0219 (0.0251)	0.0249 (0.0272)
80	0.0143 (0.0155)	0.0164 (0.0181)	0.0186 (0.0208)	0.0238 (0.0266)
120	0.0130 (0.0141)	0.0148 (0.0162)	0.0192 (0.0218)	0.0232 (0.0265)
200	0.0098 (0.0108)	0.0126 (0.0137)	0.0176 (0.0196)	0.0228 (0.0263)

Table 5 The training time (in seconds) for Example 2 when there is no noise in both the training data and the testing data ($\sigma = 0$)

Neurons	Layers			
	2	3	5	8
50	5306 ± 216	5447 ± 885	5780 ± 1027	6110 ± 1233
80	5204 ± 468	5415 ± 311	6291 ± 923	5717 ± 530
120	5860 ± 491	6286 ± 444	6114 ± 872	7098 ± 1141
200	5438 ± 522	6282 ± 672	6640 ± 741	9282 ± 217

$\lambda = 0.1$ in Eq. (21), and the training rate is set to be $\eta = 0.0005$ for 5000 training epochs using SGD. Training was performed on a laptop with a 4-core Intel® i7-8550U CPU @ 1.80 GHz using Python 3.8.10, Torch 1.12.1, and Torchdiffeq 0.2.3

with the number of neurons in each layer is not significant. Thus, for the best accuracy and computational efficiency, we recommend a neural network with 2 hidden layers and 200 neurons in each layer.

Regularization of the neural network can also affect the spectral neural PDE learner's ability to learn the dynamics or to reduce overfitting on the training set. We set $\lambda = 0.1$ in the loss function Eq. (21) and the training rate $\eta = 0.0002$ and train over 5000 epochs using SGD. We applied the ResNet and dropout techniques with a neural network containing 2 intermediate layers, each with 200 neurons. For the ResNet technique, we add the output of the first hidden layer to the output of the second hidden layer as the new output of the second hidden layer. For the dropout technique, each neuron in the second hidden layer is set to 0 with a probability $P_{\rm d}$. The results are presented in Table 6 which shows the relative L^2 errors on the training set and testing set for Example 2 when there is no noise in both the training and testing data ($\sigma = 0$). We apply regularization to the neural network, testing both the ResNet and the dropout techniques with different dropout probabilities $P_{\rm d}$. The errors are averaged over 10 independent training processes. Applying the ResNet technique leads to approximately a 20% decrease in the errors, whereas applying the dropout technique does not reduce the training error nor the testing error.

Table 6 Mean relative L^2 errors and their standard deviations for the training and testing (in parentheses) sets for a noiseless ($\sigma = 0$) Example 2

Method	None	ResNet
	0.0101 ± 0.0006 (0.0112 ± 0.0005)	0.0083 ± 0.0005 (0.0099 ± 0.0004)
P_d	Method	
	Dropout	Dropout & ResNet
0.1	0.0146 ± 0.0006 (0.0153 ± 0.0007)	0.0125 ± 0.0004 (0.0139 ± 0.0004)
0.5	0.0314 ± 0.0021 (0.0327 ± 0.0023)	0.0275 ± 0.0018 (0.0286 ± 0.0021)

The ResNet and dropout techniques are applied to regularization the neural network used to parameterize $F[u; x, t] \approx F[u; x, t, \Theta]$. For dropout, we experimented with different probabilities P_d of dropping out the links between neurons. The errors are averaged over 10 independent training processes

C How data noise and penalty parameter λ affect learning

We now investigate how different strengths σ and penalties λ affect the learning, including the dynamics of β and x_0. For each strength of noise $\sigma = 0, 0.0001, 0.001$, 100 trajectories are generated for training and another 50 are generated for testing according to Eq. (23). The penalty parameter tested are $\lambda = 10^{-2}, 10^{-1.5}, 10^{-1}, 10^{-0.5}$. The mean relative errors on the training set and testing set over 10 independent training processes are shown in Table 7 below and are plotted in Fig. 2c.

D How parameter noise and number of training solutions affect reconstruction

Here, we take different distributions of the three parameters ξ_1, ξ_2, ξ_3 in Eq. (19). We shall use $\xi_1 \sim \mathcal{N}(3, \frac{\sigma_\mathrm{p}^2}{4})$, $\xi_2 \sim \frac{1}{4}\mathcal{U}(-\frac{\sigma_\mathrm{p}}{4}, \frac{\sigma_\mathrm{p}}{4})$, $\xi_3 \sim \mathcal{N}(0, \frac{\sigma_\mathrm{p}^2}{2})$ and vary $\sigma_\mathrm{p} = 0, \frac{1}{4}, \frac{1}{2}, 1$. For different σ_p, we train 10 independent models and the results are given in Table 8.

The training and testing error is the same for models with $\sigma_\mathrm{p} = 0$ in Table 8 because if there is no uncertainty in the initial condition, all trajectories are the same. Though giving larger training errors, models trained on training sets with larger variances in the parameters of the initial condition could generalize better on testing sets where the variances in the parameters of the initial condition is larger.

Next, we change the number of solutions in the training set for training the spatiotemporal DE model and test the performance of the learned spatiotemporal DE models with different training solutions on the testing set. We take the first 12, 25, 50, and 100 solutions of the 100-solution training set generated by Eq. (19) with independently sampled $\xi_1 \sim \mathcal{N}(3, \frac{1}{4})$, $\xi_2 \sim \mathcal{U}(0, \frac{1}{2})$, $\xi_3 \sim \mathcal{N}(0, \frac{1}{2})$. Solutions in the testing set (a total of 50 solutions) are also generated from Eq. (19) with independently sampled $\xi_1 \sim \mathcal{N}(3, \frac{1}{4})$, $\xi_2 \sim \mathcal{U}(0, \frac{1}{2})$, $\xi_3 \sim \mathcal{N}(0, \frac{1}{2})$. The time points for both the training set and the testing set are taken to be $t_j = 0.1j, j = 0, ..., 9$.

Table 7 Mean relative L^2 errors defined in Eq. (17) and standard deviations, averaged over 10 independent processes, of the training and testing sets (in parentheses) with different λ and σ when learning the dynamics of the noisy data Eq. (23)

λ	σ		
	0	10^{-4}	10^{-3}
10^{-2}	0.0243 ± 0.0055	0.0454 ± 0.0693	0.0585 ± 0.0196
	(0.0244 ± 0.0058)	(0.0478 ± 0.0632)	(0.0632 ± 0.0207)
$10^{-1.75}$	0.0209 ± 0.0068	0.0469 ± 0.0694	0.0572 ± 0.0278
	(0.0210 ± 0.0069)	(0.0506 ± 0.0767)	(0.0602 ± 0.0203)
$10^{-1.5}$	0.0129 ± 0.0017	0.0477 ± 0.0417	0.1074 ± 0.0656
	(0.0133 ± 0.0020)	(0.0495 ± 0.0499)	(0.1117 ± 0.0675)
$10^{-1.25}$	0.0115 ± 0.0014	0.0426 ± 0.0249	0.1178 ± 0.0097
	(0.0116 ± 0.0014)	(0.0451 ± 0.0291)	(0.1231 ± 0.1023)
10^{-1}	0.0103 ± 0.0007	0.0291 ± 0.0137	0.0626 ± 0.0097
	(0.013 ± 0.0006)	(0.0292 ± 0.0181)	(0.0646 ± 0.0111)
$10^{-0.75}$	0.0111 ± 0.0006	0.0284 ± 0.0005	0.0672 ± 0.0124
	(0.0131 ± 0.0005)	(0.0283 ± 0.0004)	(0.0667 ± 0.0148)
$10^{-0.5}$	0.0117 ± 0.0005	0.0272 ± 0.0004	0.0573 ± 0.0009
	(0.0141 ± 0.0004)	(0.0271 ± 0.0006)	(0.0563 ± 0.0006)

An increase in σ typically results in higher errors. For small σ, selecting an intermediate λ around $10^{-1.5}$ balances learning of the adaptive spectral parameters (scaling factors β and displacements x_0) with that of coefficients $c_i(t)$ and leads to a minimal relative L^2 error. For large σ, choosing a larger $\lambda = 10^{-0.5}$ to better fit the dynamics of β and x_0 leads to smaller errors

Table 8 Training and testing errors of trained models on different testing sets with different variances in the initial condition parameters, averaged over ten trained models

Training σ_p	0	1/4	1/2	1
Training error	4.43e−3 ± 9.9e−4	1.25e−2 ± 5e−4	1.64e−2 ± 8e−3	1.62e−2 ± 1.1e−3
Training σ_p	testing σ_p			
	0	1/4	1/2	1
0	4.43e−3 ± 9.9e−4	2.81e−2 ± 3.2e−3	3.88e−2 ± 3.3e−3	5.64e−2 ± 3.0e−3
$\frac{1}{4}$	1.11e−2 ± 7e−4	1.28e−2 ± 6e−4	2.33e−2 ± 1.4e−3	4.55e−2 ± 2.4e−3
$\frac{1}{2}$	1.24e−2 ± 1.0e−3	1.07e−2 ± 6e−4	1.65e−2 ± 9e−4	2.95e−2 ± 1.5e−3
1	1.24e−2 ± 1.1e−3	9.22e−3 ± 9e−4	1.03e−2 ± 8e−4	1.68e−2 ± 1.2e−3

A neural network with two hidden layers, 200 neurons in each layer, and the ELU activation function is used for training. The results shown in Table 9 are averaged over 10 independent training process using SGD, each containing 2000 epochs and using a learning rate $\eta = 0.0002$. Equation (21) is minimized for training and the penalty parameter $\lambda = 0.1$.

Table 9 Comparison of the learned DE models with different numbers of training samples for Example 2. When the number of solutions in the training set increases, the relative L^2 errors of the predicted solutions generated by the learned DE model are smaller on both the training set and the testing set

Training samples	Training error Mean relative L^2	Testing error Mean relative L^2
12	$0.142 \pm 6.240\text{e}{-04}$	$0.0239 \pm 1.469\text{e}{-03}$
25	$0.0140 \pm 5.234\text{e}{-04}$	$0.0167 \pm 8.549\text{e}{-04}$
50	$0.0106 \pm 6.202\text{e}{-04}$	$0.0122 \pm 5.613\text{e}{-04}$
100	$0.0081 \pm 4.160\text{e}{-04}$	$0.0097 \pm 4.371\text{e}{-04}$

Table 10 The L^2 errors (Eq. (17)) and the total number of basis functions used to learn the evolution of the Gaussian wave packet (Eq. (24)) are listed for different N and γ

γ	N		
	5	9	14
$-\infty$	**36**, 0.0180 (0.0164)	**100**, 0.0204, (0.0194)	**225**, 0.0110, (0.0105)
-1	**23**, 0.0342, (0.0302)	**51**, 0.0185, (0.0166)	**92**, 0.0102, (0.0093)
0	**21**, 0.0414, (0.0363)	**42**, 0.0230, (0.0210)	**70**, 0.0309, (0.0279)
$\frac{1}{2}$	**20**, 0.0459, (0.0409)	**37**, 0.0413, (0.0365)	**55**, 0.0336, (0.0295)

The bold number in each cell represents the number of basis functions used, which is also sensitive to the hyperbolicity γ. The numbers in parentheses are the errors on the testing set

E Varying hyperbolic cross-space parameters N and γ

If the spectral expansion order N is sufficiently large, using a hyperbolic cross space Eq. (27) can effectively reduce the required number of basis functions while maintaining accuracy. In our experiments, we set $N = 5, 9, 14$ and $\gamma = -\infty$ (full tensor product), -1, 0, $\frac{1}{2}$. We train the network for 1000 epochs using SGDM with a learning rate $\eta = 0.001$, momentum $= 0.9$, and weight decay $= 0.005$. The penalty coefficient in Eq. (21) is $\lambda = 0.02$.

From Table 10, we see that a hyperbolic space with $N = 14, \gamma = -1$ leads to minimal errors on the testing set. Furthermore, the number of basis functions for the hyperbolic space with $N = 15, \gamma = -1$ is smaller than the full tensor product space for $N = 9, 14$ when $\gamma = -\infty$, so the hyperbolic space with $N = 14, \gamma = -1$ could be close to the most appropriate choice. We shall also use the saliency map to investigate the role of different frequencies and plot $|\frac{\partial \text{Loss}}{\partial c_{i,\ell}(0)}|$ for different N and γ in Fig. 4, where the loss function is Eq. (21). Even for different choices of N, γ, the changes in frequencies on the lower-left part of the saliency map (corresponding to a moderate γ and a large N) have the largest impact on the loss. This “resolution-invariance” justifies our choices that the proper hyperbolic space should have a larger N but a moderate $\gamma > \infty$ so that the total number of inputs or outputs are reduced to boost efficiency in higher-dimensional problems while accuracy is maintained.

The errors in Table 10 can be larger on the training set than on the testing set, especially at larger training set errors (e.g., for $N = 5$). This arises because the largest

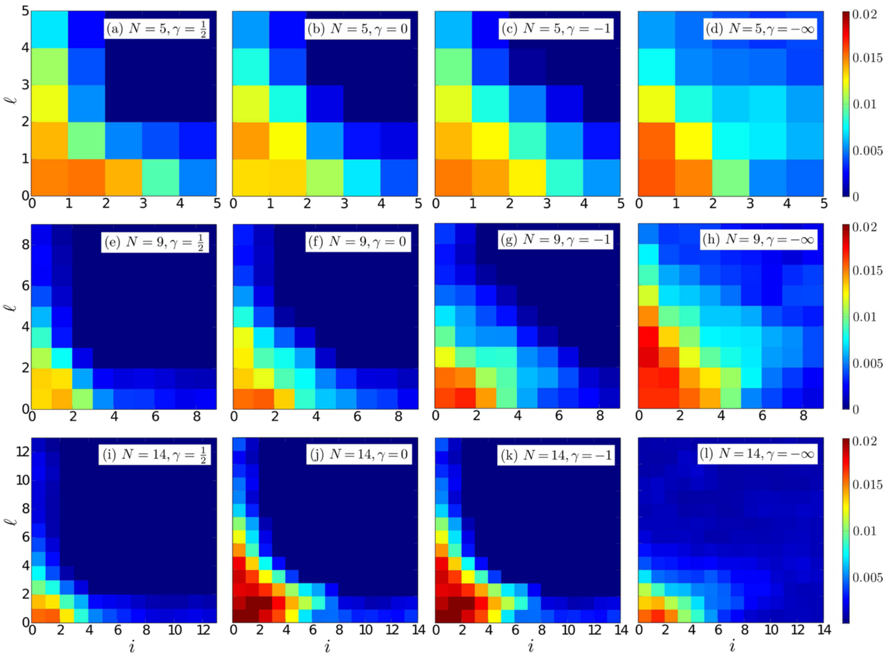

Fig. 4 Saliency maps of the absolute value of derivatives of the relative L^2 loss w.r.t. $\{c_{i,\ell}\}$. Here, $N = 5, 9, 14$ and $\gamma = -\infty, -1, 0, \frac{1}{2}$ in Eq. (27). The loss function is always most sensitive to frequencies $c_{i,\ell}$ on the lower left of the saliency maps. Such a "resolution-invariance" indicates that having a larger N but a moderate $\gamma > -\infty$ to include the frequencies in the lower-left part of this saliency map leads to a balance between efficiency and accuracy

sampling time among the training samples is $t = 1$ while it is less than 1 for testing samples. If the trained dynamics $F(\tilde{U}; t, \Theta)$ does not approximate the true dynamics $F(\tilde{U}; t)$ in Eq. (10) well, the error of the training samples with time $t = 1$ will be larger than that of testing samples due to error accumulation.

F Comparison with the s-PINN method

To make a comparison with the s-PINN method proposed in [18], we consider the following inverse-type problem of reconstructing the unknown potential $f(x,t)$ in

$$u_t = u_{xx} + f(x,t), \tag{30}$$

by approximating $f(x,t) \approx \hat{f} := F[u; x, t, \Theta] - u_{xx}$. The function u is taken to be

$$u(x,t) = \frac{\xi}{\sqrt{t+1}} \exp\left(-\frac{x^2}{4(t+1)}\right) + \frac{\sin(x)}{\sqrt{t+1}} \exp\left(-\frac{x^2}{4(t+1)}\right) \tag{31}$$

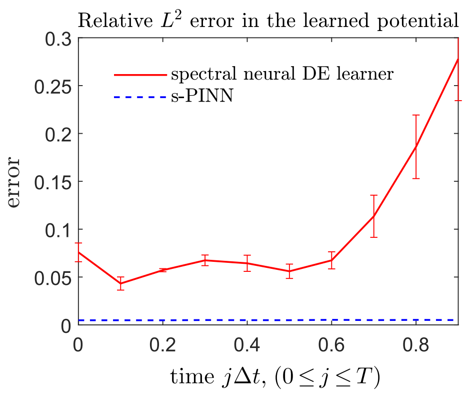

Fig. 5 Comparison of the relative L^2 error in the potential in Eq. (30) learned from our proposed spatiotemporal DE learner and from the s-PINN method. Our spectral neural DE learner achieved an average relative L^2 error of 0.1, while the s-PINN method, designed to input the exact form of the RHS of Eq. (30) with only one unknown potential, achieved better accuracy with an average relative L^2 error of about 0.01

where $\xi \sim \mathcal{U}(\frac{1}{2}, 1)$ is i.i.d. sampled for different trajectories. Therefore, the true potential in Eq. (30) is

$$f(x,t) = \Big[(t+1)\sin(x) + x\cos(x)\Big](t+1)^{-3/2}\exp\left(-\frac{x^2}{4(t+1)}\right), \tag{32}$$

which is independent of $u(x,t)$. We generate 100 solutions $u_m(x,t_i)$, $m = 1, ..., 100$ as the training set to learn the unknown potential with $t_i = i\Delta t$, $\Delta t = 0.1$, $i = 0, ..., 10$. In the s-PINN method, since only t is inputted, only one reconstructed $\hat{f}$ (which is the same for all trajectories) is outputted in the form of a spectral expansion. However, in our spatiotemporal DE learning method, $f(x,t) \approx \hat{f} = F[u; x, t, \Theta] - u_{xx}$ will be different for different inputted u giving rise to a changing error along the time horizon. The mean and variance of the relative L^2 error $\frac{\|\hat{f}-f\|_2}{\|f\|_2}$ is plotted in Fig. 5.

When all but the potential on the RHS of Eq. (1) is known, s-PINN is preferable because more information is inputted as part of the loss function in [18]. Nevertheless, our spatiotemporal DE learner can still achieve a relative L^2 error ~ 0.1, indicating that without any prior information it can still reconstruct the unknown source term with acceptable accuracy. However, the accuracy of our spatiotemporal DE learner for reconstructing the potential f deteriorates as the time horizon $t = j\Delta t$ increases. Since errors accumulate, minimizing Eq. (21) requires more accurate reconstruction of the dynamics (RHS of Eq. (1)) at earlier times than at later times.

Funding This work was supported in part by the Army Research Office under Grant W911NF-18-1-0345.

Data availability No data were used in this manuscript. All reported results in the manuscript were generated numerically from the methods presented. Codes that were used to generate numerical results can be made available on request.

Declaration

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships to disclose.

References

1. Bar, L., Sochen, N.: Unsupervised deep learning algorithm for PDE-based forward and inverse problems. arXiv preprint arXiv:1904.05417 (2019)
2. Stephany, R., Earls, C.: PDE-LEARN: using deep learning to discover partial differential equations from noisy, limited data. Neural Netw. 106242 (2024)
3. Long, Z., Lu, Y., Ma, X., Dong, B.: PDE-net: learning PDEs from data. In: International Conference on Machine Learning, pp. 3208–3216 (2018). PMLR
4. Churchill, V., Chen, Y., Xu, Z., Xiu, D.: Dnn modeling of partial differential equations with incomplete data. J. Comput. Phys. **493**, 112502 (2023)
5. Long, Z., Lu, Y., Dong, B.: PDE-net 2.0: learning PDEs from data with a numeric-symbolic hybrid deep network. J. Comput. Phys. **399**, 108925 (2019)
6. Brunton, S.L., Proctor, J.L., Kutz, J.N.: Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Natl. Acad. Sci. **113**(15), 3932–3937 (2016)
7. Rudy, S.H., Brunton, S.L., Proctor, J.L., Kutz, J.N.: Data-driven discovery of partial differential equations. Sci. Adv. **3**(4), 1602614 (2017)
8. Xu, H., Chang, H., Zhang, D.: DL-PDE: deep-learning based data-driven discovery of partial differential equations from discrete and noisy data. Commun. Comput. Phys. **29**(3), 698–728 (2021)
9. Anandkumar, A., Azizzadenesheli, K., Bhattacharya, K., Kovachki, N., Li, Z., Liu, B., Stuart, A.: Neural operator: graph kernel network for partial differential equations. In: ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations (2020)
10. Brandstetter, J., Worrall, D.E., Welling, M.: Message passing neural PDE solvers. In: International Conference on Learning Representations (2021)
11. Li, Z., Kovachki, N.B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier neural operator for parametric partial differential equations. In: International Conference on Learning Representations (2020)
12. Xia, M., Shao, S., Chou, T.: Efficient scaling and moving techniques for spectral methods in unbounded domains. SIAM J. Sci. Comput. **43**(5), 3244–3268 (2021)
13. De Pablo, A., Quirós, F., Rodrı́guez, A., Vázquez, J.L.: A general fractional porous medium equation. Commun. Pure Appl. Math. 65(9), 1242–1284 (2012)
14. Grindrod, P.M.: Patterns and Waves: The Theory and Applications of Reaction-Diffusion Equations. Oxford (1991)
15. Antoine, X., Arnold, A., Besse, C., Ehrhardt, M., Schädle, A.: A review of transparent and artificial boundary conditions techniques for linear and nonlinear Schrödinger equations. Commun. Comput. Phys. **4**(4), 729–796 (2008)
16. Zhang, W., Yang, J., Zhang, J., Du, Q.: Artificial boundary conditions for nonlocal heat equations on unbounded domain. Commun. Comput. Phys. **21**(1), 16–39 (2017)
17. Fanaskov, V., Oseledets, I.: Spectral neural operators. arXiv preprint arXiv:2205.10573 (2022)
18. Xia, M., Böttcher, L., Chou, T.: Spectrally adapted physics-informed neural networks for solving unbounded domain problems. Mach. Learn. Sci. Technol. **4**(2), 025024 (2023)

19. Burns, K.J., Vasil, G.M., Oishi, J.S., Lecoanet, D., Brown, B.P.: Dedalus: a flexible framework for numerical simulations with spectral methods. Phys. Rev. Res. **2**(2), 023068 (2020)
20. Shen, J., Tang, T., Wang, L.-L.: Spectral Methods: Algorithms, Analysis and Applications. Springer, New York (2011)
21. Chen, R.T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp. 6572–6583 (2018)
22. Shen, J., Wang, L.-L.: Sparse spectral approximations of high-dimensional problems based on hyperbolic cross. SIAM J. Numer. Anal. **48**(3), 1087–1109 (2010)
23. Tang, T.: The Hermite spectral method for Gaussian-type functions. SIAM J. Sci. Comput. **14**(3), 594–606 (1993)
24. Xia, M., Shao, S., Chou, T.: A frequency-dependent p-adaptive technique for spectral methods. J. Comput. Phys. **446**, 110627 (2021)
25. Chou, T., Shao, S., Xia, M.: Adaptive Hermite spectral methods in unbounded domains. Appl. Numer. Math. **183**, 201–220 (2023)
26. Shen, J., Yu, H.: Efficient spectral sparse grid methods and applications to high-dimensional elliptic problems. SIAM J. Sci. Comput. **32**(6), 3228–3250 (2010)
27. Bateman, H.: Some recent researches on the motion of fluids. Mon. Weather Rev. **43**(4), 163–170 (1915)
28. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
29. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012)
30. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
31. Chen, Z., Xiong, Y., Shao, S.: Numerical methods for the Wigner equation with unbounded potential. J. Sci. Comput. **79**(1), 345–368 (2019)
32. Shao, S., Lu, T., Cai, W.: Adaptive conservative cell average spectral element methods for transient Wigner equation in quantum transport. Commun. Comput. Phys. **9**(3), 711–739 (2011)
33. Bungartz, H.-J., Griebel, M.: Sparse grids. Acta Numer. **13**, 147–269 (2004)
34. Zenger, C., Hackbusch, W.: Sparse grids. In: Proceedings of the Research Workshop of the Israel Science Foundation on Multiscale Phenomenon, Modelling and Computation, p. 86 (1991)
35. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: Proceedings of the International Conference on Learning Representation (ICLR), pp. 1–8 (2014)
36. Wang, L.-L., Shen, J.: Error analysis for mapped jacobi spectral methods. J. Sci. Comput. **24**, 183–218 (2005)
37. Shen, J., Wang, L.-L.: Some recent advances on spectral methods for unbounded domains. Commun. Comput. Phys. **5**(2–4), 195–241 (2009)
38. Zhao, T., Zhao, Z., Li, C., Li, D.: Spectral approximation of ψ-fractional differential equation based on mapped Jacobi functions. arXiv preprint arXiv:2312.16426 (2023)
39. Wang, Z., Chen, M., Chen, J.: Solving multiscale elliptic problems by sparse radial basis function neural networks. J. Comput. Phys. **492**, 112452 (2023)
40. Buhmann, M.D.: Radial basis functions. Acta Numer. **9**, 1–38 (2000)
41. Chen, C.-S., Noorizadegan, A., Young, D.L., Chen, C.S.: On the selection of a better radial basis function and its shape parameter in interpolation problems. Appl. Math. Comput. **442**, 127713 (2023)
42. Clement, P.R.: Laguerre functions in signal analysis and parameter identification. J. Franklin Inst. **313**(2), 85–95 (1982)
43. Vismara, F., Benacchio, T., Bonaventura, L.: A seamless, extended DG approach for advection-diffusion problems on unbounded domains. J. Sci. Comput. **90**, 1–27 (2022)
44. Xiong, Y., Guo, X.: A short-memory operator splitting scheme for constant-Q viscoelastic wave equation. J. Comput. Phys. **449**, 110796 (2022)
45. Jin, B., Rundell, W.: A tutorial on inverse problems for anomalous diffusion processes. Inverse Prob. **31**(3), 035003 (2015)
46. Lu, L., Meng, X., Mao, Z., Karniadakis, G.E.: Deepxde: a deep learning library for solving differential equations. SIAM Rev. **63**(1), 208–228 (2021)

47. Chen, Z., Churchill, V., Wu, K., Xiu, D.: Deep neural network modeling of unknown partial differential equations in nodal space. J. Comput. Phys. **449**, 110782 (2022)